Arduino RFID Driver

Generated by Doxygen 1.8.11

ii CONTENTS

Contents

1	Hier	rarchical Index		2
	1.1	Class Hierarchy		2
2	Clas	ss Index		3
	2.1	Class List		3
3	File	Index		5
	3.1	File List		5
4	Clas	ss Documentation		5
	4.1	RadioFrequencyIdentification	MFRC522::BIT_FRAMINGbits Union Reference	5
		4.1.1 Detailed Description		5
		4.1.2 Member Data Docum	nentation	6
	4.2	RadioFrequencyIdentification	MFRC522::COLLbits Union Reference	6
		4.2.1 Detailed Description		6
		4.2.2 Member Data Docum	nentation	7
	4.3	RadioFrequencyIdentification	MFRC522::COM_I_ENbits Union Reference	7
		4.3.1 Detailed Description		8
		4.3.2 Member Data Docum	nentation	8
	4.4	RadioFrequencyIdentification	MFRC522::COM_IRQbits Union Reference	9
		4.4.1 Detailed Description		9
		4.4.2 Member Data Docum	nentation	9
	4.5	RadioFrequencyIdentification	nMFRC522::COMMANDbits Union Reference	10
		4.5.1 Detailed Description		10
		4.5.2 Member Data Docum	nentation	10
	4.6	RadioFrequencyIdentification	MFRC522::CONTROLbits Union Reference	11
		4.6.1 Detailed Description		11
		4.6.2 Member Data Docum	nentation	11
	4.7	RadioFrequencyIdentification	nMFRC522::CW_GS_Pbits Union Reference	12
		4.7.1 Detailed Description		12

	4.7.2	Member Data Documentation	12
4.8	RadioF	requencyIdentificationMFRC522::DEMODbits Union Reference	13
	4.8.1	Detailed Description	13
	4.8.2	Member Data Documentation	13
4.9	RadioF	requencyIdentificationMFRC522::DIV_I_ENbits Union Reference	14
	4.9.1	Detailed Description	14
	4.9.2	Member Data Documentation	14
4.10	RadioF	requencyIdentificationMFRC522::DIV_IRQbits Union Reference	15
	4.10.1	Detailed Description	15
	4.10.2	Member Data Documentation	15
4.11	RadioF	requencyIdentificationMFRC522::ERRORbits Union Reference	16
	4.11.1	Detailed Description	16
	4.11.2	Member Data Documentation	17
4.12	RadioF	requencyIdentificationMFRC522::FIFO_LEVELbits Union Reference	17
	4.12.1	Detailed Description	18
	4.12.2	Member Data Documentation	18
4.13	RadioF	requencyIdentificationMFRC522::GS_Nbits Union Reference	18
	4.13.1	Detailed Description	19
	4.13.2	Member Data Documentation	19
4.14	RadioF	requencyIdentificationMFRC522::MF_RXbits Union Reference	19
	4.14.1	Detailed Description	19
	4.14.2	Member Data Documentation	20
4.15	RadioF	requencyIdentificationMFRC522::MF_TXbits Union Reference	20
	4.15.1	Detailed Description	20
	4.15.2	Member Data Documentation	20
4.16	RadioF	requencyIdentificationMFRC522::MOD_GS_Pbits Union Reference	21
	4.16.1	Detailed Description	21
	4.16.2	Member Data Documentation	21
4.17	RadioF	requencyIdentificationMFRC522::MODEbits Union Reference	22
	4.17.1	Detailed Description	22

iv CONTENTS

	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
File I	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

5

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 RegisterBasedDevice	24
RadioFrequencyIdentificationMFRC522	24
RadioFrequencyIdentificationMFRC522::RF_CFGbits	34
RadioFrequencyIdentificationMFRC522::RX_MODEbits	35
RadioFrequencyIdentificationMFRC522::RX_SELbits	36
RadioFrequencyIdentificationMFRC522::RX_THRESHOLDbits	37
RadioFrequencyIdentificationMFRC522::SERIAL_SPEEDbits	38

2 Class Index

	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

2

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

3 File Index 5

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	
<pre>#include <radiofrequencyidentificationmfrc522.h></radiofrequencyidentificationmfrc522.h></pre>	
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 | 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 | ENbits::HI ALERT | 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.

 $4.3.2.9 \quad unsigned \ char \ Radio Frequency I dentification MFRC 522:: COM_I_EN bits:: TX_I_EN$

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 RadioFrequencyldentificationMFRC522::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.

 $4.8.2.3 \quad unsigned \ char \ Radio Frequency Identification MFRC522:: DEMOD bits:: FIX_IQ$

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
      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.

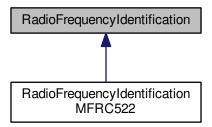
The documentation for this union was generated from the following file:

· RadioFrequencyIdentificationMFRC522.h

4.18 RadioFrequencyIdentification Class Reference

#include <RadioFrequencyIdentification.h>

Inheritance diagram for RadioFrequencyIdentification:



4.18.1 Detailed Description

Arduino - Radio Frequency Identification.

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 10 of file RadioFrequencyIdentification.h.

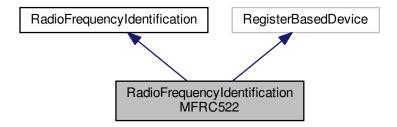
The documentation for this class was generated from the following file:

· RadioFrequencyIdentification.h

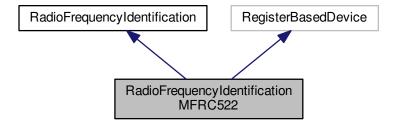
4.19 RadioFrequencyIdentificationMFRC522 Class Reference

#include <RadioFrequencyIdentificationMFRC522.h>

Inheritance diagram for RadioFrequencyIdentificationMFRC522:



Collaboration diagram for RadioFrequencyIdentificationMFRC522:



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

enum Register {

• union WATER LEVELbits

Public Types

```
COMMAND = 0x01, COM | EN = 0x02, DIV | 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 RadioFrequencyldentificationMFRC522::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 RadioFrequencyldentificationMFRC522::communicate (Command command, unsigned char * output, 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 ComlrqReg 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 ComlrqReg 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 RadioFrequencyldentificationMFRC522::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 RadioFrequencyldentificationMFRC522::writeRegisterBlock (unsigned char *reg*, 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 RadioFrequencyldentificationMFRC522::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.

 $4.21.2.3 \quad unsigned \ char \ Radio Frequency Identification MFRC 522:: RX_MODE bits:: RX_CRC_EN$

Definition at line 846 of file RadioFrequencyIdentificationMFRC522.h.

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 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 RadioFrequencyldentificationMFRC522::RX_THRESHOLDbits::MIN_LEVEL

Definition at line 1002 of file RadioFrequencyldentificationMFRC522:.h.

4.23.2.5 unsigned char RadioFrequencyldentificationMFRC522::RX_THRESHOLDbits::value

 $Definition\ at\ line\ 1004\ of\ file\ Radio Frequency Identification MFRC522.h.$

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 RadioFrequencyldentificationMFRC522::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 RadioFrequencyldentificationMFRC522::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

 $Definition\ at\ line\ 1253\ of\ file\ Radio Frequency Identification MFRC 522.h.$

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 { ... }
```

 $4.33.2.2 \quad unsigned \ char \ Radio Frequency Identification MFRC 522:: VERSION bits:: CHIPTYPE$

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>

5 File Documentation 49

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 RadioFrequencyldentificationMFRC522::WATER_LEVELbits::WATER_LEVEL

Definition at line 646 of file RadioFrequencyIdentificationMFRC522.h.

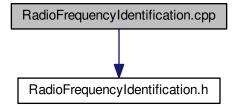
The documentation for this union was generated from the following file:

· RadioFrequencyIdentificationMFRC522.h

5 File Documentation

5.1 RadioFrequencyIdentification.cpp File Reference

```
#include "RadioFrequencyIdentification.h"
Include dependency graph for RadioFrequencyIdentification.cpp:
```

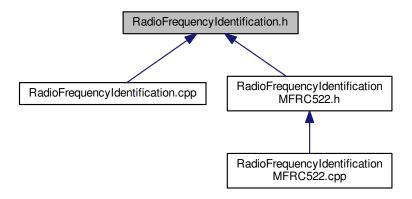


5.2 RadioFrequencyIdentification.cpp

```
00001 #include "RadioFrequencyIdentification.h"
```

5.3 RadioFrequencyldentification.h File Reference

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



Classes

· class RadioFrequencyIdentification

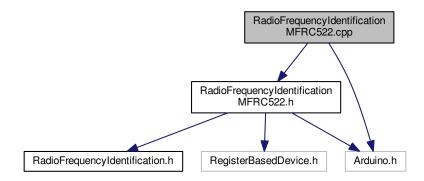
5.4 RadioFrequencyldentification.h

```
00001
00007 #ifndef __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_H_
00008 #define __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_H_ 1
00009
00010 class RadioFrequencyIdentification {
00011
00012 };
00013
00014 #endif // __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_H_
```

5.5 RadioFrequencyIdentificationMFRC522.cpp File Reference

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

Include dependency graph for RadioFrequencyIdentificationMFRC522.cpp:



5.6 RadioFrequencyIdentificationMFRC522.cpp

```
00001 #include "RadioFrequencyIdentificationMFRC522.h"
00002 #include <Arduino.h>
\tt 00004\ RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::RadioFrequencyIdentificationMFRC522::
00005
                               {\tt RegisterBasedDevice} \ \star {\tt device}, \ {\tt unsigned} \ {\tt char} \ {\tt resetPin})
00006
                               : RadioFrequencyIdentification(), device(device), resetPin(resetPin) {
00007
                      pinMode(resetPin, OUTPUT);
                      digitalWrite(resetPin, LOW);
00009 }
00010
00011 void RadioFrequencyIdentificationMFRC522::initialize() {
                    if (digitalRead(resetPin) == LOW) {
00012
00013
                             digitalWrite(resetPin, HIGH);
00014
                               delay(50);
                     } else {
00015
00016
                              softReset();
00017
00018
                      clearRegisterBits(AUTO_TEST, AUTO_TEST_ENABLE);
00019
00020 //
                           // TAuto=1; f(Timer) = 6.78MHz/TPreScaler
00021 //
                          device->writeRegister(T_MODE, 0x8d);
00022 //
00023 //
                          // TModeReg[3..0] + TPrescalerReg
                          device->writeRegister(T_PRESCALER_LOW, 0x3e);
device->writeRegister(T_RELOAD_LOW, 0x1e);
00024 //
00025 //
00026 //
                          device->writeRegister(T_RELOAD_HIGH, 0);
00027
00028 // 100% ASK
00029
                      writeRegister(TX_ASK, 0x40);
00030
                      // CRC Initial value 0x6363 ???
00031
00032
                     writeRegister (MODE, 0x3d);
00034
                      // ClearBitMask(STATUS2, 0x08);
                                                                                                          //MFCrypto10n=0
                      00035
                      // Write_AddicoreRFID(RFCfgReg, 0x7f);
00036
                                                                                                                                   //RxGain = 48dB
00037
00038
                      // Open the antenna
00039
                      setAntennaOn();
00040 }
00041
00042 void RadioFrequencyIdentificationMFRC522::softReset() {
00043
                     writeRegister(COMMAND, SOFT_RESET);
00044 }
00045
00046 void RadioFrequencyIdentificationMFRC522::setAntennaOn() {
00047
                      setRegisterBits(TX_CONTROL, TX_CONTROL_TX_RF_EN);
00048 }
00049
00050 void RadioFrequencyIdentificationMFRC522::setAntennaOff()
00051
                      clearRegisterBits(TX_CONTROL, TX_CONTROL_TX_RF_EN);
```

```
00052 }
00053
00054 int RadioFrequencyIdentificationMFRC522::readRegisterBlock
      (unsigned char reg,
00055
              unsigned char *buf, unsigned char len) {
00056
00057
          // MSB == 1 is for reading. LSB is not used in address.
00058
          return device->readRegisterBlock(((reg << 1) & 0x7e) | 0x80, buf, len);</pre>
00059 }
00060
00061 unsigned char RadioFrequencyIdentificationMFRC522::writeRegisterBlock
     (unsigned char reg,
00062
              unsigned char *buf, unsigned char len) {
00063
00064
          // MSB == 0 is for writing. LSB is not used in address.
00065
          return device->writeRegisterBlock((reg << 1) & 0x7e, buf, len);</pre>
00066 }
00067
00068 void RadioFrequencyIdentificationMFRC522::configureTimer
      (unsigned int prescaler,
00069
              unsigned int reload, bool autoStart, bool autoRestart) {
          T_MODEbits timerMode;
00070
          timerMode.value = readRegister(T_MODE);
timerMode.T_PRESCALER_HI = (prescaler >> 8) & 0x0f;
00071
00072
00073
          timerMode.T_AUTO = autoStart;
          timerMode.T_GATED = (unsigned char) 0;
00074
00075
          timerMode.T_AUTO_RESTART = autoRestart;
00076
          writeRegister(T_MODE, timerMode.value);
00077
          writeRegister(T_PRESCALER_LOW, prescaler & 0xff);
          writeRegister(T_RELOAD_HIGH, (reload >> 8) & 0xff);
00078
00079
          writeRegister (T RELOAD LOW, reload & 0xff);
00080 }
00081
00082 void RadioFrequencyIdentificationMFRC522::startTimer() {
00083
          setRegisterBits(CONTROL, CONTROL_T_START_NOW);
00084 }
00085
00086 void RadioFrequencyIdentificationMFRC522::stopTimer() {
          setRegisterBits(CONTROL, CONTROL_T_STOP_NOW);
00087
00088 }
00089
00090 void RadioFrequencyIdentificationMFRC522::enableInterrupt
      (Interrupt interrupt) {
          setRegisterBits (MFR522_INT_TO_EN_REG(interrupt),
00091
      MFR522_INT_TO_EN_MASK(interrupt));
00092 }
00093
00094 void RadioFrequencyIdentificationMFRC522::disableInterrupt
      (Interrupt interrupt) {
          clearRegisterBits (MFR522_INT_TO_EN_REG(interrupt),
00095
      MFR522_INT_TO_EN_MASK(interrupt));
00096 }
00097
00098 void RadioFrequencyIdentificationMFRC522::clearInterrupt
      (Interrupt interrupt) {
00099
00100
          // 0x7f: first bit 0 indicates that the marked bits in the register are cleared
00101
          configureRegisterBits(MFR522_INT_TO_IRQ_REG(interrupt),
00102
                   (MFR522_INT_TO_IRQ_MASK(interrupt)) | 0x80, 0x7f);
00103 }
00104
00105 void RadioFrequencyIdentificationMFRC522::flushQueue() {
          setRegisterBits(FIFO_LEVEL, FIFO_LEVEL_FLUSH_BUFFER);
00107 }
00108
00109 void RadioFrequencyIdentificationMFRC522::setWaterLevel(
     unsigned char level) {
00110
          writeRegister(WATER_LEVEL, WATER_LEVEL_WATER_LEVEL & level);
00111 }
00112
00113 int RadioFrequencyIdentificationMFRC522::generateRandomId
      (unsigned char buf[10]) {
00114
00115
          // Stop any active command.
          sendCommand(IDLE);
00116
00117
00118
          // Clear all seven interrupt request bits
00119
          clearInterrupt(COM_ALL_IRQ);
00120
00121
          // FlushBuffer = 1, FIFO initialization
00122
          flushQueue();
00123
00124
          // Send command
00125
          sendCommand(GENERATE_RANDOM_ID);
00126
          // Wait for command to complete.
00127
00128
          waitForRegisterBits(COM_IRQ, COM_IRQ_IDLE_IRQ);
```

```
00129
           // FlushBuffer = 1, FIFO initialization
00130
00131
           flushQueue();
00132
           // Transfers 25 bytes from the internal buffer to the FIFO buffer.
00133
00134
          sendCommand (MEM);
00135
00136
           \ensuremath{//} Wait for command to complete.
00137
           waitForRegisterBits(COM_IRQ, COM_IRQ_IDLE_IRQ);
00138
           sendCommand(IDLE);
           return readRegisterBlock(FIFO_DATA, buf, 10);
00139
00140 }
00141
00142 int RadioFrequencyIdentificationMFRC522::communicate(
      Command command, unsigned char *output,
00143
               unsigned char *input, unsigned char outputLen, bool checkCRC) {
00144
          int len = 0;
COM_IRQbits irq;
00145
00146
00147
00148
           // Stop any active command.
00149
           sendCommand(IDLE);
00150
00151
           \ensuremath{//} Clear all seven interrupt request bits
00152
          clearInterrupt(COM_ALL_IRQ);
00153
00154
           // FlushBuffer = 1, FIFO initialization
00155
           flushQueue();
00156
00157
           // Write sendData to the FIFO
          writeRegisterBlock(FIFO_DATA, output, outputLen);
00158
00159
00160
           // Execute the command
00161
           sendCommand(command);
00162
           if (command == TRANSCEIVE) {
00163
00164
00165
               // StartSend=1, transmission of data starts
00166
               setRegisterBits(BIT_FRAMING, BIT_FRAMING_START_SEND);
00167
00168
           // Wait for the command to complete.
00169
           // If timer was configured and T_AUTO flag is active in T_MODE register,
00170
           // timer will start automatically after all data is transmitted.
00171
00172
           // See: configureTimer method
00173
00174
               irq.value = readRegister(COM_IRQ);
00175
00176
               // Timer interrupt - nothing received
               if (irq.TIMER_IRQ) {
00177
00178
                   lastError = TIMEOUT_ERROR;
00179
                   return -1;
00180
00181
           } while (!irq.IDLE_IRQ && !irq.RX_IRQ);
00182
          // Stop now if any errors except collisions were detected.
// ErrorReg[7..0] bits are: WrErr TempErr reserved BufferOvfl CollErr CRCErr ParityErr ProtocolErr
00183
00185
           ERRORbits error;
00186
           error.value = readRegister(ERROR);
      if (error.BUFFER_OVFL || error.PARITY_ERR || error.
PROTOCOL_ERR || error.COLL_ERR) {
00187
00188
              lastError = COMMUNICATION_ERROR;
00189
               return -1;
00190
00191
00192
           // If the caller wants data back, get it from the MFRC522.
00193
          if (input != NULL) {
00194
00195
               // Number of bytes in the FIFO
00196
               int inpulLen = readRegister(FIFO_LEVEL);
00197
00198
               // Get received data from FIFO
00199
               len = readRegisterBlock(FIFO_DATA, input, inpulLen);
00200
00201
               // Perform CRC A validation if requested.
               if (len > 0 && checkCRC) {
00202
00203
00204
                   // 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];
00205
00206
                   returnedCRC <<= 8;
00207
                   returnedCRC |= input[inpulLen - 2] & 0xff;
00208
00209
                   if (expectedCRC != returnedCRC) {
00210
                        lastError = CRC_ERROR;
00211
                        return -1;
00212
                   }
00213
               }
```

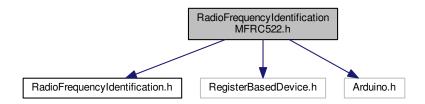
```
00214
00215
          return len;
00216 }
00217
00218 unsigned int RadioFrequencyIdentificationMFRC522::calculateCRC
      (unsigned char *buff,
00219
              unsigned char len) {
00220
00221
          unsigned int crc = 0;
00222
00223
          // Stop any active command.
          sendCommand(IDLE);
00224
00225
00226
          // Clear all seven interrupt request bits
00227
          clearInterrupt(DIV_ALL_IRQ);
00228
          // FlushBuffer = 1, FIFO initialization
00229
00230
          flushQueue();
00231
00232
          // Write sendData to the FIFO
00233
          writeRegisterBlock(FIFO_DATA, buff, len);
00234
          \ensuremath{//} Start the calculation
00235
00236
          sendCommand (CALC CRC);
00237
00238
          // Wait for the CRC calculation to complete.
00239
          waitForRegisterBits(DIV_IRQ, DIV_IRQ_CRC_IRQ);
00240
00241
          \ensuremath{//} Stop calculating CRC for new content in the FIFO.
00242
          sendCommand(IDLE);
00243
00244
          crc = readRegister(CRC_RESULT_HIDH);
00245
          crc <<= 8;
00246
          crc |= readRegister(CRC_RESULT_LOW) & 0xff;
00247
          return crc;
00248 }
00249
00250 bool RadioFrequencyIdentificationMFRC522::waitForRegisterBits
     (unsigned char reg,
00251
              unsigned char mask, unsigned long timeout) {
00252
          unsigned char v;
00253
          unsigned long start = millis();
00254
          do {
00255
              v = readRegister(reg);
00256
          } while (!(v & mask) && start + timeout > millis());
00257
          return (v & mask) > 0;
00258 }
00259
00260 bool RadioFrequencyIdentificationMFRC522::performSelfTest
     () {
00261
          unsigned char *firmwareReference;
00262
          unsigned char buffer[64] = { 0 };
00263
          writeRegister(AUTO_TEST, 0x00);
00264
          softReset();
00265
          flushQueue();
00266
          writeRegisterBlock(FIFO_DATA, buffer, 25);
00267
          sendCommand (MEM);
00268
          writeRegister(AUTO_TEST, AUTO_TEST_ENABLE);
00269
          writeRegister(FIFO_DATA, 0x00);
00270
          sendCommand(CALC_CRC);
          waitForRegisterBits(DIV_IRQ, DIV_IRQ_CRC_IRQ, 100);
00271
00272
          readRegisterBlock(FIFO_DATA, buffer, 64);
00273
          switch (getVersion()) {
00274
          case CLONE:
00275
              firmwareReference = (unsigned char *) FM17522_FIRMWARE_REFERENCE;
00276
             break;
          case V0 0:
00277
00278
              firmwareReference = (unsigned char *) MFRC522_FIRMWARE_REFERENCE_V0_0;
00279
             break:
00280
          case V1_0:
00281
             firmwareReference = (unsigned char *) MFRC522_FIRMWARE_REFERENCE_V1_0;
             break;
00282
00283
          case V2_0:
00284
             firmwareReference = (unsigned char *) MFRC522_FIRMWARE_REFERENCE_V2_0;
00285
              break;
00286
          default:
00287
             return false;
00288
          for (unsigned char i = 0; i < 64; i++) {
00289
              if (buffer[i] != pgm_read_byte(&(firmwareReference[i]))) {
00290
00291
                  return false;
00292
00293
          return true;
00294
00295 }
00296
00297 RadioFrequencyIdentificationMFRC522::Version
```

```
RadioFrequencyIdentificationMFRC522::getVersion() {
00298     return (Version) readRegister(VERSION);
00299 }
```

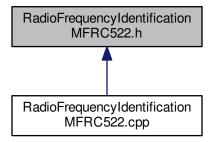
5.7 RadioFrequencyIdentificationMFRC522.h File Reference

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

Include dependency graph for RadioFrequencyIdentificationMFRC522.h:



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



Classes

- class RadioFrequencyIdentificationMFRC522
- · union RadioFrequencyIdentificationMFRC522::COMMANDbits
- union RadioFrequencyIdentificationMFRC522::COM | 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, 0x59, 0x61, 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_
80000
00009 #include <RadioFrequencyIdentification.h>
00010 #include <RegisterBasedDevice.h>
00011 #include <Arduino.h>
00012
00013 #define MFRC522 DEFAULT TIMEOUT 100
00014
                                                                                                                 (i > COM_ALL_IRQ) ? DIV_I_EN : COM_I_EN
00015 #define MFR522_INT_TO_EN_REG(i)
                                                                                                            (i > COM_ALL_IRQ) ? (i >> 8) & DIV_I_EN_INTERRUPT_EN : i &
00016 #define MFR522_INT_TO_EN_MASK(i)
                  COM_I_EN_INTERRUPT_EN
00017
                                                                                                                (i > COM_ALL_IRQ) ? DIV_IRQ : COM_IRQ
(i > COM_ALL_IRQ) ? (i >> 8) & DIV_IRQ_ALL_IRQ : i & COM_IRQ_ALL_IRQ
00018 #define MFR522_INT_TO_IRQ_REG(i)
00019 #define MFR522_INT_TO_IRQ_MASK(i)
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,\ 0xe2,\ 
                                     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,
00026
00027
00028
                                     0x4f, 0x30, 0x32, 0x8d };
00029
```

```
00030 // Version 1.0 (0x91)
00031 // NXP Semiconductors; Rev. 3.8 - 17 September 2014; 16.1.1 self-test
00032 \text{ const byte MFRC522\_FIRMWARE\_REFERENCE\_VI\_0[] } PROGMEM = \{ 0x00, 0xc6, 0x37, 0xd5, 0x32, 0xb7, 0xd5, 0x
                                                0x57, 0x5c, 0xc2, 0xd8, 0x7c, 0x4d, 0xd9, 0x70, 0xc7, 0x73, 0x10, 0xe6, 0xd2, 0xaa,
00033
00034
                                                0x5e, \ 0xa1, \ 0x3e, \ 0x5a, \ 0x14, \ 0xaf, \ 0x30, \ 0x61, \ 0xc9, \ 0x70, \ 0xdb, \ 0x2e, \ 0x64, \ 0x22, \ 0x64, \ 0x64
00035
                                                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,
00036
00037
                                                0x29, 0x79 };
00038
00039 // Version 2.0 (0x92)
00040 // NXP Semiconductors; Rev. 3.8 - 17 September 2014; 16.1.1 self-test
00041 \text{ const byte MFRC522\_FIRMWARE\_REFERENCE\_V2\_0[]} \quad \textbf{PROGMEM} = \{ \ 0 \times 00, \ 0 \times \text{eb}, \ 0 \times \text{bd}, \ 0 \times \text{bd}, \ 0 \times \text{bf}, \ 0 \times \text{bf},
                                                0x23, 0x95, 0xd0, 0xe3, 0x0d, 0x3d, 0x27, 0x89, 0x5c, 0xde, 0x9d, 0x3b, 0xa7, 0x00,
00042
                                                0x21, 0x5b, 0x89, 0x82, 0x51, 0x3a, 0xeb, 0x02, 0x0c, 0xa5, 0x00, 0x49, 0x7c, 0x84,
00043
00044
                                                0x4d, 0xb3, 0xcc, 0xd2, 0x1b, 0x81, 0x5d, 0x48, 0x76, 0xd5, 0x71, 0x61, 0x21, 0xa9,
00045
                                                0x86, 0x96, 0x83, 0x38, 0xcf, 0x9d, 0x5b, 0x6d, 0xdc, 0x15, 0xba, 0x3e, 0x7d, 0x95,
00046
                                                0x3b, 0x2f };
00047
00049 // Fudan Semiconductor FM17522 (0x88)
00050 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,
00051
                                                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,
00052
00053
00054
                                                0x2a, 0x62 };
00055
00056
00057 class RadioFrequencyIdentificationMFRC522: public
                   RadioFrequencyIdentification,
                                                public RegisterBasedDevice {
00058
00059
00060
                                  RegisterBasedDevice *device;
00061
00062
                                  unsigned char resetPin;
00063
00064
                                  unsigned char lastError;
00065
00066 public:
00067
00068
                                  enum Register {
00069
00070
                                                 // Starts and stops command execution
                                                COMMAND = 0x01,
00071
00072
00073
                                                 // Enable and disable interrupt request control bits
00074
                                                COM_I_EN = 0x02,
00075
                                                 // Enable and disable interrupt request control bits
00076
                                                DIV I EN = 0 \times 03.
00077
00078
                                                 // Interrupt request bits
00080
                                                COM_IRQ = 0x04,
00081
00082
                                                 // Interrupt request bits Table 31 on page 40
                                                DIV_IRQ = 0X05,
00083
00084
00085
                                                 // Error bits showing the error status of the last command
00086
                                                 ERROR = 0X06,
00087
00088
                                                 // Communication status bits
00089
                                                STATUS1 = 0x07.
00090
00091
                                                 // Receiver and transmitter status bits
00092
                                                 STATUS2 = 0x08,
00093
00094
                                                 // Input and output of 64 byte FIFO buffer
00095
                                                FIFO DATA = 0X09,
00096
00097
                                                 // Number of bytes stored in the FIFO buffer
00098
                                                 FIFO LEVEL = 0x0a,
00099
00100
                                                 // Level for FIFO underflow and overflow warning
00101
                                                WATER\_LEVEL = 0x0b,
00102
                                                 // Miscellaneous control registers
00103
                                                 CONTROL = 0x0c
00105
00106
                                                 // Adjustments for bit-oriented frames
00107
                                                BIT_FRAMING = 0x0d
00108
                                                 // Bit position of the first bit-collision detected on the RF
00109
                                                COLL = 0x0e,
00111
00112
                                                 // Defines general modes for transmitting and receiving
00113
                                                MODE = 0x11,
00114
00115
                                                // Defines transmission data rate and framing
```

```
00116
              TX MODE = 0x12.
00117
00118
              // Defines reception data rate and framing
00119
              RX MODE = 0x13,
00120
              // Controls the logical behavior of the antenna driver pins TX1 and TX2
00121
00122
              TX\_CONTROL = 0x14,
00123
00124
              // Controls the setting of the transmission modulation
00125
              TX\_ASK = 0x15,
00126
              // Selects the internal sources for the antenna driver
00127
00128
              TX SEL = 0x16,
00129
00130
              // Selects internal receiver settings
00131
              RX\_SEL = 0x17,
00132
00133
              // Selects thresholds for the bit decoder
              RX\_THRESHOLD = 0x18,
00134
00135
               // Defines demodulator settings
00136
00137
              DEMOD = 0x19,
00138
              // Some MIFARE communication transmit parameters
00139
00140
              MF_TX = 0x1c
00141
00142
              // Controls some MIFARE communication receive parameters
00143
              MF_RX = 0x1d,
00144
              // Selects the speed of the serial UART interface
00145
00146
              SERIAL\_SPEED = 0x1f
00147
00148
              // Shows the MSB and LSB values of the CRC calculation (HIGH)
00149
              CRC_RESULT_HIDH = 0x21,
00150
              // Shows the MSB and LSB values of the CRC calculation (LOW)
00151
              CRC RESULT LOW = 0x22,
00152
00153
00154
               // Controls the ModWidth setting
00155
              MOD_WIDTH = 0x24,
00156
00157
              // Configures the receiver gain
00158
              RFC FG = 0 \times 26.
00159
00160
              // Selects the conductance of the antenna driver pins TX1 and TX2 for modulation
00161
              GS_N = 0x27,
00162
              // The conductance of the p-driver output during periods of no modulation
00163
              CW GS P = 0x28.
00164
00165
00166
              // Defines the conductance of the p-driver output during periods of modulation
00167
              MOD\_GS\_P = 0x29,
00168
00169
              // Defines settings for the internal timer
00170
              T_MODE = 0x2a
00171
00172
              // Defines settings for the internal timer
00173
              T_PRESCALER_LOW = 0x2b,
00174
00175
              // Defines the 16-bit timer reload value (HIGH)
00176
              T RELOAD HIGH = 0x2c,
00177
00178
              // Defines the 16-bit timer reload value (LOW)
00179
              T_RELOAD_LOW = 0x2d,
00180
00181
              // Shows the 16-bit timer value (HIGH)
00182
              T COUNTER VAL HIGH = 0x2e,
00183
00184
              // Shows the 16-bit timer value (LOW)
              T_COUNTER_VAL_LOW = 0x2f,
00186
00187
              // Test signal configuration
00188
              TEST\_SEL1 = 0x31,
00189
00190
              // Test signal configuration and PRBS control
00191
              TEST\_SEL2 = 0x32,
00192
00193
              // Enables pin output driver on pins D1 to D7 \,
00194
              TEST_PIN_EN = 0x33,
00195
              // Defines the values for D1 to D7 when it is used as an I/O bus
00196
00197
              TEST_PIN_VALUE = 0x34,
00198
00199
              // Shows the status of the internal test bus
00200
              TEST_BUS = 0x35,
00201
00202
              // Controls the digital self test
```

```
00203
               AUTO\_TEST = 0x36,
00204
00205
                // Shows the software version
               VERSION = 0x37,
00206
00207
00208
                // Controls the pins AUX1 and AUX2
               ANALOG_TEST = 0x38,
00210
00211
                // Defines the test value for TestDAC1
00212
               TEST DAC1 = 0x39,
00213
00214
                // Defines the test value for TestDAC2
00215
               TEST_DAC2 = 0x3a,
00216
00217
                // Shows the value of ADC I and Q channels
00218
               TEST\_ADC = 0x3b
00219
          };
00220
          enum Command {
00222
00223
                // no action, cancels current command execution
00224
               IDLE = 0x00.
00225
                // stores 25 bytes into the internal buffer
00226
00227
               MEM = 0x01,
00228
00229
                // generates a 10-byte random ID number
00230
               GENERATE_RANDOM_ID = 0x02,
00231
               // activates the CRC calculation or performs a self test
00232
00233
               CALC CRC = 0 \times 03.
00234
00235
                // Transmit data
00236
               TRANSMIT = 0x04,
00237
               // no command change, can be used to modify the CommandReg register bits without affecting the
00238
       command, for example, the PowerDown bit NO_CMD_CHANGE = 0x07,
00239
00240
00241
                // activates the receiver circuits (receive data)
00242
               RECEIVE = 0x08,
00243
               // transmits data from FIFO buffer to antenna and automatically activates the receiver after
00244
       transmission (transmit and receive data)
00245
               TRANSCEIVE = 0x0c,
00246
00247
                // performs the MIFARE standard authentication as a reader (authentication)
00248
               MF\_AUTHENT = 0x0e,
00249
00250
                // resets the MFRC522
               SOFT_RESET = 0x0F
00252
          };
00253
00254
          enum Error {
   NO_ERROR = 0x00,
00255
00256
               GENERAL_ERROR = 0 \times 01,
00257
               TIMEOUT\_ERROR = 0x02,
00258
               COMMUNICATION_ERROR = 0 \times 03,
00259
               CRC\_ERROR = 0x04
00260
          };
00261
00262
           enum Mask {
00263
               TX\_CONTROL\_TX1\_RF\_EN = 0x01,
00264
                TX\_CONTROL\_TX2\_RF\_EN = 0x02,
00265
               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,
00266
00267
               COM_I_EN_INTERRUPT_EN = 0x7f,
00268
               COM_IRQ_TIMER_IRQ = 0x01,
00269
00270
               COM_IRQ_ERR_IRQ = 0x02,
               COM_IRQ_LO_ALERT_IRQ = 0x04,
00271
               COM_IRQ_HI_ALERT_IRQ = 0x08,
00272
               COM_IRQ_IDLE_IRQ = 0x10,
COM_IRQ_RX_IRQ = 0x20,
COM_IRQ_TX_IRQ = 0x40,
00273
00274
00275
00276
               COM_IRQ_ALL_IRQ = 0x7f,
               COM_IRQ_SET1 = 0x80,
DIV_I_EN_CRC_I_EN = 0x04,
DIV_I_EN_MFIN_ACT_I_EN = 0x10,
00277
00278
00279
               DIV_I_EN_INTERRUPT_EN = DIV_I_EN_CRC_I_EN |
00280
     DIV_I_EN_MFIN_ACT_I_EN,
DIV_IRQ_CRC_IRQ = 0x04,
00281
00282
               DIV_IRQ_MFIN_ACT_IRQ = 0x10,
00283
               DIV_IRQ_ALL_IRQ = DIV_IRQ_CRC_IRQ |
      DIV_IRO_MFIN_ACT_IRO,
DIV_IRO_SET2 = 0x80,
00284
```

```
00285
               FIFO_LEVEL_FLUSH_BUFFER = 0x80,
00286
               FIFO_LEVEL_FIFO_LEVEL = 0x7f,
00287
               WATER_LEVEL_WATER_LEVEL = 0x3f,
               BIT_FRAMING_START_SEND = 0x80,
00288
00289
               AUTO\_TEST\_ENABLE = 0x09
00290
          };
00291
00292
          enum Interrupt
00293
             : unsigned int {
00294
                  NONE_IRQ = 0x0000,
              COM\_TIMER\_IRQ = 0x0001,
00295
00296
               COM ERR IRO = 0 \times 0002.
               COM_LO_ALERT_IRQ = 0x0002,
COM_HI_ALERT_IRQ = 0x0008,
00297
00298
00299
               COM\_IDLE\_IRQ = 0x0010,
               COM_RX_IRQ = 0x0020,

COM_TX_IRQ = 0x0040,
00300
00301
               COM_ALL_IRQ = 0x007f,
DIV_CRC_IRQ = 0x0400,
00302
00303
00304
               DIV\_MFIN\_ACT\_IRQ = 0x1000,
00305
               DIV_ALL_IRQ = DIV_CRC_IRQ | DIV_MFIN_ACT_IRQ
00306
00307
          union COMMANDbits {
00312
00313
00314
              struct {
00315
00316
                    // Activates a command based on the Command value;
00317
                   // reading this register shows which command is executed
                   unsigned char COMMAND :4;
00318
00319
00320
                   // 1: Soft power-down mode entered
00321
                   // 0: when the MFRC522 is ready
00322
                   // Remark: The PowerDown bit cannot be set when the SoftReset command is activated
00323
                   unsigned char POWER_DOWN :1;
00324
                   // Analog part of the receiver is switched off unsigned char {\tt RCV\_OFF} :1;
00325
00326
00327
00328
                   // Reserved
00329
                   unsigned char :2;
00330
               }:
               unsigned char value:
00331
00332
          };
00333
00339
          union COM_I_ENbits {
00340
00341
               struct {
00342
00343
                   // Allows the timer interrupt request (TimerIRg bit) to be propagated to pin IRO
00344
                   unsigned char TIMER_I_EN :1;
00345
00346
                   // Allows the error interrupt request (ErrIRq bit) to be propagated to pin IRQ
00347
                   unsigned char ERR_I_EN :1;
00348
                   // Allows the low alert interrupt request (LoAlertIRq bit) to be propagated to pin IRQ unsigned char LO_ALERT_I_EN :1;
00349
00351
00352
                   // Allows the high alert interrupt request (HiAlertIRq bit) to be propagated to pin IRQ
00353
                   unsigned char HI_ALERT_I_EN :1;
00354
00355
                   // Allows the idle interrupt request (IdleIRq bit) to be propagated to pin IRQ
00356
                   unsigned char IDLE_I_EN :1;
00357
00358
                   // Allows the receiver interrupt request (RxIRq bit) to be propagated to pin IRQ
00359
                   unsigned char RX_I_EN :1;
00360
                   // Allows the transmitter interrupt request (TxIRq bit) to be propagated to pin IRQ
00361
00362
                   unsigned char TX_I_EN :1;
00363
00364
                   // 1: Signal on pin IRQ is inverted with respect to the StatuslReg register's IRq bit
00365
                   // 0: signal on pin IRQ is equal to the IRq bit; in combination with the DivIEnReg register's
00366
                   // IRqPushPull bit, the default value of logic 1 ensures that the output level on pin IRQ is
       3-state
00367
                   unsigned char I RO INV :1;
00368
               };
00369
              unsigned char value;
00370
00371
00377
          union DIV I ENbits (
00378
00379
               struct {
00380
00381
                   // Reserved
00382
                   unsigned char :2;
00383
                   // Allows the CRC interrupt request, indicated by the DivIrgReg register's CRCIRg bit, to be
00384
```

```
propagated to pin IRQ
00385
                  unsigned char CRC_I_EN :1;
00386
00387
                   // Reserved
00388
                  unsigned char :1;
00389
00390
                   // Allows the MFIN active interrupt request to be propagated to pin IRQ
00391
                   unsigned char MFIN_ACT_I_EN :1;
00392
                   // Reserved
00393
00394
                  unsigned char :2;
00395
00396
                   // 1: pin IRQ is a standard CMOS output pin
00397
                   // 0: pin IRQ is an open-drain output pin
00398
                   unsigned char IRQ_PUSH_PULL :1;
00399
00400
              unsigned char value:
00401
          };
00402
00408
          union COM_IRQbits {
00409
00410
              struct {
00411
                   // The timer decrements the timer value in register {\tt TCounterValReg} to zero
00412
00413
                  unsigned char TIMER_IRQ :1;
00414
00415
                   // Any error bit in the {\tt ErrorReg} register is set
00416
                  unsigned char ERR_IRQ :1;
00417
00418
                   // Status1Reg register's LoAlert bit is set in opposition to the LoAlert bit,
00419
                   // the LoAlertIRq bit stores this event and can only be reset as indicated by
00420
                   // the Set1 bit in this register
00421
                   unsigned char LO_ALERT_IRQ :1;
00422
00423
                   // Status1Reg register's HiAlert bit is set in opposition to the HiAlert bit,
                   \ensuremath{//} the HiAlertIRq bit stores this event and can only be reset as indicated by
00424
                   // the Set1 bit in this register
00425
                   unsigned char HI_ALERT_IRQ :1;
00427
00428
                   // If a command terminates, for example, when the CommandReg changes
00429
                   // 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
00430
       set
00431
                   // The microcontroller starting the Idle command does not set the IdleIRq bit
00432
                   unsigned char IDLE IRQ :1;
00433
                   // 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
00434
00435
                  // only set to logic 1 when data bytes are available in the FIFO unsigned char RX_IRQ :1;
00436
00437
00438
00439
                   \ensuremath{//} Set immediately after the last bit of the transmitted data was sent out
00440
                   unsigned char TX_IRQ :1;
00441
                   // 1: indicates that the marked bits in the ComIrqReg register are set
00442
                   // 0: indicates that the marked bits in the ComTrqReg register are cleared
00443
                   unsigned char SET1 :1;
00445
               };
00446
               unsigned char value;
00447
          };
00448
00454
          union DIV IRObits {
00455
00456
              struct {
00457
00458
                   // Reserved
00459
                  unsigned char :2;
00460
00461
                   // The CalcCRC command is active and all data is processed
                   unsigned char CRC_IRQ :1;
00462
00463
00464
                   // Reserved
00465
                  unsigned char :1;
00466
00467
                   // MFIN is active this interrupt is set when either a rising or falling signal edge is
00468
                   unsigned char MFIN_ACT_IRQ :1;
00469
                   // Reserved
00470
00471
                  unsigned char :2;
00472
00473
                   // 1: indicates that the marked bits in the DivIrqReg register are set
00474
                   // 0: indicates that the marked bits in the DivIrqReg register are cleared
00475
                   unsigned char SET2 :1;
00476
00477
              unsigned char value;
00478
          };
```

```
00479
          union ERRORbits {
00485
00486
00487
              struct {
00488
                  // Set to logic 1 if the SOF is incorrect automatically cleared during receiver start-up phase
00489
                  // bit is only valid for 106 kBd during the MFAuthent command, the ProtocolErr bit is set to
00490
00491
                  // logic 1 if the number of bytes received in one data stream is incorrect
00492
                  unsigned char PROTOCOL_ERR :1;
00493
                  // Parity check failed. Automatically cleared during receiver start-up phase
00494
                  // only valid for ISO/IEC 14443 A/MIFARE communication at 106 kBd
00495
00496
                  unsigned char PARITY_ERR :1;
00497
00498
                  // The RxModeReg register's RxCRCEn bit is set and the CRC calculation fails
00499
                  // automatically cleared to logic 0 during receiver start-up phase
                  unsigned char CRC ERR :1:
00500
00501
00502
                  // A bit-collision is detected cleared automatically at receiver start-up phase
00503
                  // only valid during the bitwise anticollision at 106 kBd always set to logic 0 during
       communication
00504
                  // protocols at 212 kBd, 424 kBd and 848 kBd \,
00505
                  unsigned char COLL_ERR :1;
00506
                  // The host or a MFRC522's internal state machine (e.g. receiver) tries to
00507
                  // write data to the FIFO buffer even though it is already full
00508
00509
                  unsigned char BUFFER_OVFL :1;
00510
00511
                  // Reserved
00512
                  unsigned char :1;
00513
00514
                  // Internal temperature sensor detects overheating, in which case the antenna drivers are
       automatically switched off
00515
                  unsigned char TEMP_ERR :1;
00516
                  // Data is written into the FIFO buffer by the host during the MFAuthent command or if data is
00517
       written
00518
                  // into the FIFO buffer by the host during the time between sending the last bit on the RF
       interface and
00519
                  // receiving the last bit on the RF interface
00520
                  unsigned char WR_ERR :1;
00521
              }:
              unsigned char value;
00523
          };
00524
00530
          union STATUS1bits {
00531
00532
              struct {
00533
                  // The number of bytes stored in the FIFO buffer corresponds to equation:
00534
00535
                  // HiAlert = FIFOLength <= WaterLevel
00536
                  // example:
00537
                  // FIFO length = 4, WaterLevel = 4 > LoAlert = 1
                  // FIFO length = 5, WaterLevel = 4 > LoAlert = 0
00538
00539
                  unsigned char LO_ALERT :1;
00540
00541
                  // The number of bytes stored in the FIFO buffer corresponds to equation:
00542
                  // HiAlert = (64 - FIFOLength) <= WaterLevel
00543
                  // example:
00544
                  // FIFO length = 60, WaterLevel = 4 > HiAlert = 1
                  // FIFO length = 59, WaterLevel = 4 > HiAlert = 0
00545
00546
                  unsigned char HI ALERT :1;
00547
00548
                  // MFRC522's timer unit is running, i.e. the timer will decrement the TCounterValReg register
       with the next timer clock
00549
                  // 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;
00550
                  // this bit is not influenced by the gated signal
00551
                  unsigned char T_RUNNING :1;
00553
                  // Indicates if any interrupt source requests attention with respect to the setting of the
       interrupt enable bits:
00554
                  // see the ComIEnReg and DivIEnReg registers
00555
                  unsigned char IRQ :1;
00556
                  // The CRC calculation has finished only valid for the CRC coprocessor calculation using the
       CalcCRC command
00558
                 unsigned char CRC_READY :1;
00559
00560
                  // The CRC result is zero
                  \ensuremath{//} for data transmission and reception, the CRCOk bit is undefined: use the
00561
                  // ErrorReg register's CRCErr bit indicates the status of the CRC coprocessor, during
00562
       calculation the value
00563
                  // changes to logic 0, when the calculation is done correctly the value changes to logic 1
00564
                  unsigned char CRC_OK :1;
00565
                  // Reserved
00566
```

```
unsigned char :1;
00568
00569
              unsigned char value;
00570
          };
00571
00577
          union STATUS2bits {
00578
00579
              struct {
00580
00581
                  // Shows the state of the transmitter and receiver state machines:
00582
                  // 000: idle
00583
                  // 001: wait for the BitFramingReg register's StartSend bit
00584
                      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
00585
       register
00586
                  // 011: transmitting
                  // 100: RxWait: wait until RF field is present if the TModeReg register's TxWaitRF bit is set
00587
       to logic 1
00588
                  // the minimum time for RxWait is defined by the RxWaitReg register
                  // 101: wait for data
// 110: receiving
00589
00590
00591
                  unsigned char MODEM_STATE :3;
00592
                  // Indicates that the MIFARE Cryptol unit is switched on and therefore all data communication
00593
       with the card is encrypted
00594
                 // can only be set to logic 1 by a successful execution of the MFAuthent command only valid in
       Read/Write mode for
00595
                  // MIFARE standard cards this bit is cleared by software
00596
                  unsigned char MF_CRYPTO1_ON :1;
00597
00598
                  // Reserved
                  unsigned char :2;
00600
00601
                  // I2C-bus input filter settings:
00602
                  // 1: the I2C-bus input filter is set to the High-speed mode independent of the I2C-bus
       protocol
00603
                  // 0: the I2C-bus input filter is set to the I2C-bus protocol used
                  unsigned char I2C_FORCE_HS :1;
00604
00605
00606
                  // Clears the temperature error if the temperature is below the alarm limit of 125C
00607
                  unsigned char TEMP_SENS_CLEAR :1;
00608
              }:
              unsigned char value:
00609
00610
          };
00611
00617
          union FIFO_LEVELbits {
00618
00619
              struct {
00620
                  // Indicates the number of bytes stored in the FIFO buffer writing to the FIFODataReg
00621
00622
                  // register increments and reading decrements the FIFOLevel value
00623
                  unsigned char FIFO_LEVEL :7;
00624
00625
                  // Immediately clears the internal FIFO buffer's read and write pointer and {\tt ErrorReg}
                  // register's BufferOvfl bit reading this bit always returns 0
00626
00627
                  unsigned char FLUSH BUFFER :1;
00628
              };
00629
              unsigned char value;
00630
00631
00637
          union WATER LEVELbits (
00638
00639
              struct {
00640
00641
                  // Defines a warning level to indicate a FIFO buffer overflow or underflow:
00642
                  // 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
00643
       WaterLevel bytes
00644
                  // Status1Reg register's LoAlert bit is set to logic 1 if equal to, or less than the WaterLevel
       bytes in the FIFO buffer
00645
                  // Remark: to calculate values for HiAlert and LoAlert see Section 9.3.1.8 on page 42.
00646
                  unsigned char WATER_LEVEL :7;
00647
                  // Reserved
00648
00649
                  unsigned char :1;
00650
              };
00651
              unsigned char value;
00652
00653
          union CONTROLbits (
00659
00660
00661
              struct {
00662
00663
                  // Indicates the number of valid bits in the last received byte if this value is 000b, the
       whole byte is valid
00664
                  unsigned char RX LAST BITS :3;
00665
```

```
// Reserved
00666
00667
                  unsigned char :2;
00668
00669
                   //\ {\tt Timer\ starts\ immediately}
                  // reading this bit always returns it to logic 0
unsigned char T_START_NOW :1;
00670
00671
00672
00673
                   // Timer stops immediately
00674
                   // reading this bit always returns it to logic0
00675
                  unsigned char T_STOP_NOW :1;
00676
              };
00677
              unsigned char value;
00678
          };
00679
00685
          union BIT_FRAMINGbits {
00686
00687
              struct {
00688
00689
                   // Used for transmission of bit oriented frames: defines the number of bits of the last byte
       that will be transmitted
00690
                  // 000b indicates that all bits of the last byte will be transmitted
00691
                  unsigned char TX_LAST_BITS :3;
00692
                  // Reserved
00693
00694
                  unsigned char :1;
00695
                  // used for reception of bit-oriented frames: defines the bit position for the first bit
00696
       received to be stored in the FIFO buffer
                  // example:
00697
                   // 0: LSB of the received bit is stored at bit position 0, the second received bit is stored at
00698
       bit position 1
00699
                   // 1: LSB of the received bit is stored at bit position 1, the second received bit is stored at
       bit position 2
00700
                  // 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 {\tt O}
                   // These bits are only to be used for bitwise anticollision at 106 kBd, for all other modes
00701
       they are set to 0
00702
                  unsigned char RX_ALIGN :3;
00703
00704
                   // Starts the transmission of data only valid in combination with the Transceive command
00705
                  unsigned char START_SEND :1;
00706
              }:
00707
              unsigned char value:
00708
          };
00709
00715
          union COLLbits {
00716
00717
              struct {
00718
00719
                  // Shows the bit position of the first detected collision in a received frame only data bits
       are interpreted
00720
00721
                   // 00h: indicates a bit-collision in the 32nd bit
00722
                   // 01h: indicates a bit-collision in the 1st bit
00723
                      08h: indicates a bit-collision in the 8th bit
00724
                   // These bits will only be interpreted if the CollPosNotValid bit is set to logic 0 \,
                  unsigned char COLL_POS :5;
00726
00727
                   // No collision detected or the position of the collision is out of the range of CollPos[4:0]
00728
                  unsigned char COLL_POS_NOT_VALID :1;
00729
                  // Reserved
00730
00731
                  unsigned char :1;
00732
00733
                  // All received bits will be cleared after a collision only used during bitwise anticollision
       at 106 kBd, otherwise it is set to logic 1
00734
                  unsigned char VALUES_AFTER_COLL :1;
00735
              };
00736
              unsigned char value:
00737
          };
00738
00744
          union MODEbits {
00745
00746
              struct {
00747
00748
                   // defines the preset value for the CRC coprocessor for the CalcCRC command
00749
                   // Remark: during any communication, the preset values are selected automatically according to
00750
                   // the definition of bits in the RxModeReg and TxModeReg registers
00751
                      00: 0000h
00752
                      01: 6363h
00753
                      10: A671h
00754
                       11: FFFFh
                  unsigned char CRC_PRESET :2;
00755
00756
00757
                   // Reserved
00758
                  unsigned char :1;
00759
```

```
00760
                  // Defines the polarity of pin MFIN
                  // Remark: the internal envelope signal is encoded active LOW, changing this bit generates a
00761
       MFinActIRq event
                  // 1: polarity of pin MFIN is active HIGH
// 0: polarity of pin MFIN is active LOW
unsigned char POL_M_FIN :1;
00762
00763
00764
00765
00766
00767
                  unsigned char :1;
00768
00769
                  // Transmitter can only be started if an RF field is generated
00770
                  unsigned char TX_WAIT_RF :1;
00771
00772
00773
                  unsigned char :1;
00774
00775
                  // CRC coprocessor calculates the CRC with MSB first in the CRCResultReg register the values
       for the
00776
                  // CRCResultMSB[7:0] bits and the CRCResultLSB[7:0] bits are bit reversed
00777
                  // Remark: during RF communication this bit is ignored
00778
                  unsigned char MSB_FIRST :1;
00779
              };
00780
              unsigned char value;
00781
          };
00782
00788
          union TX_MODEbits {
00789
00790
              struct {
00791
00792
                  // Reserved
00793
                  unsigned char :3:
00794
00795
                  // Modulation of transmitted data is inverted
00796
                  unsigned char INV_MOD :1;
00797
                  // Defines the bit rate during data transmission the MFRC522 handles transfer speeds up to 848
00798
       kBd
00799
                      000: 106 kBd
                      001: 212 kBd
00800
00801
                      010: 424 kBd
00802
                  // 011: 848 kBd
00803
                  unsigned char TX_SPEED :3;
00804
00805
                  // Enables CRC generation during data transmission
00806
                  // Remark: can only be set to logic 0 at 106 kBd
00807
                  unsigned char TX_CRC_EN :1;
00808
00809
              unsigned char value;
00810
          };
00811
00817
          union RX_MODEbits {
00818
00819
              struct {
00820
                  // Reserved
00821
00822
                  unsigned char :2;
00823
00824
                  // 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
00825
       to handle
                  // the polling command after setting this bit the Receive and Transceive commands will not
00826
       terminate automatically.
00827
                  // Multiple reception can only be deactivated by writing any command (except the Receive
       command) to the CommandReg
00828
                  // register, or by the host clearing the bit if set to logic 1, an error byte is added to the
       FIFO buffer at the
00829
                 MFRC522 version 2.0 the CRC status is
                  // reflected in the signal CRCOk, which indicates the actual status of the CRC coprocessor. For
00830
       the MFRC522 version 1.0 the CRC
00831
                  // status is reflected in the signal CRCErr.
00832
                  unsigned char RX_MULTIPLE :1;
00833
                  // An invalid received data stream (less than 4 bits received) will be ignored and the receiver
00834
       remains active
00835
                  unsigned char RX_NO_ERR :1;
00836
00837
                  // Defines the bit rate while receiving data the MFRC522 handles transfer speeds up to 848 kBd
00838
                  // 000: 106 kBd
                  // 001: 212 kBd
00839
                      010: 424 kBd
00840
                      011: 848 kBd
00841
00842
                  unsigned char RX SPEED :3;
00843
00844
                  \ensuremath{//} Enables the CRC calculation during reception
                  // Remark: can only be set to logic 0 at 106 kBd
00845
                  unsigned char RX_CRC_EN :1;
00846
```

```
00847
               };
00848
               unsigned char value;
00849
           };
00850
00856
           union TX CONTROLbits {
00857
00858
               struct {
00859
00860
                    // Output signal on pin TX1 delivers the 13.56 MHz energy carrier modulated by the transmission
       data
00861
                   unsigned char TX1 RF EN :1;
00862
00863
                   // Output signal on pin TX2 delivers the 13.56 MHz energy carrier modulated by the transmission
00864
                   unsigned char TX2_RF_EN :1;
00865
                    // Reserved
00866
00867
                   unsigned char :1;
00868
00869
                    // 1: output signal on pin TX2 continuously delivers the unmodulated 13.56 MHz energy carrier
00870
                    // 0: Tx2CW bit is enabled to modulate the 13.56 MHz energy carrier
00871
                   unsigned char TX2_CW :1;
00872
                   // Output signal on pin TX1 inverted when driver TX1 is disabled unsigned char INV_TX1_RF_OFF :1;
00873
00874
00875
00876
                    // Output signal on pin TX2 inverted when driver TX2 is disabled
00877
                    unsigned char INV_TX2_RF_OFF :1;
00878
                    // Output signal on pin TX1 inverted when driver TX1 is enabled
00879
00880
                   unsigned char INV TX1 RF ON :1:
00881
00882
                    // Output signal on pin TX2 inverted when driver TX2 is enabled
00883
                    unsigned char INV_TX2_RF_ON :1;
00884
               };
00885
00886
               struct {
00887
00888
                   // Output signal on pin TX1 delivers the 13.56 MHz energy carrier modulated by the transmission
00889
                   // Output signal on pin TX2 delivers the 13.56 MHz energy carrier modulated by the transmission
       data
00890
                   unsigned char TX RF EN :2:
00891
00892
                    // Reserved
00893
                    unsigned char :2;
00894
                    // Output signal on pin TX1 inverted when driver TX1 is disabled // Output signal on pin TX2 inverted when driver TX2 is disabled
00895
00896
00897
                   unsigned char INV_TX_RF_OFF :2;
00898
00899
                     / Output signal on pin TX1 inverted when driver TX1 is enabled
00900
                    // Output signal on pin TX2 inverted when driver TX2 is enabled
00901
                   unsigned char INV_TX_RF_ON :2;
00902
               };
00903
               unsigned char value;
00904
           };
00905
00911
          union TX_ASKbits {
00912
00913
               struct {
00914
00915
                    // Reserved
00916
                   unsigned char :6;
00917
00918
                    // Forces a 100% ASK modulation independent of the ModGsPReg register setting
00919
                   unsigned char FORCE_100_ASK :1;
00920
00921
                    // Reserved
00922
                   unsigned char :1;
00923
00924
               unsigned char value;
00925
           };
00926
00932
          union TX_SELbits {
00933
00934
               struct {
00935
00936
                    \ensuremath{//} Selects the input for pin MFOUT
00937
                        0000: 3-state
00938
                    11
                        0001: LOW
00939
                        0010: HIGH
                       0011: test bus signal as defined by the TestSellReg register's TstBusBitSel[2:0] value 0100: modulation signal (envelope) from the internal encoder, Miller pulse encoded
00940
00941
00942
                        0101: serial data stream to be transmitted, data stream before Miller encoder
00943
                        0110: reserved
00944
                        0111: serial data stream received, data stream after Manchester decoder
```

```
// 1000: to 1111 reserved
00946
                  unsigned char MF_OUT_SEL :4;
00947
00948
                  // Selects the input of drivers TX1 and TX2 \,
00949
                  // 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
00950
00951
00952
                   // 11: HIGH; the HIGH level depends on the setting of bits InvTx1RFOn/InvTx1RFOff and
       InvTx2RFOn/InvTx2RFOff
00953
                  unsigned char :2;
00954
00955
                  // Reserved
00956
                  unsigned char :2;
00957
              };
00958
              unsigned char value;
00959
          };
00960
00966
          union RX_SELbits {
00967
00968
              struct {
00969
00970
                  // After data transmission the activation of the receiver is delayed for RxWait bit-clocks,
       during this 'frame guard time
00971
                  // any signal on pin RX is ignored this parameter is ignored by the Receive command all other
       commands, such as Transceive,
00972
                  // MFAuthent use this parameter the counter starts immediately after the external RF field is
       switched on
00973
                  unsigned char RX_WAIT :6;
00974
00975
                  // Selects the input of the contactless UART
00976
                  // 00: constant LOW
00977
                  // 01: Manchester with subcarrier from pin MFIN
00978
                  // 10: modulated signal from the internal analog module, default
00979
                   // 11: NRZ coding without subcarrier from pin MFIN which is only valid for transfer speeds
       above 106 kBd
00980
                  unsigned char UART SEL :2;
00981
              };
00982
              unsigned char value;
00983
00984
00990
          union RX THRESHOLDbits {
00991
00992
              struct {
00993
00994
                  // defines the minimum signal strength at the decoder input that must be reached by the weaker
       half-bit of the
00995
                  // Manchester encoded signal to generate a bit-collision relative to the amplitude of the
       stronger half-bit
00996
                  unsigned char COLL LEVEL :3:
00997
00998
                  // Reserved
00999
                  unsigned char :1;
01000
                  // Defines the minimum signal strength at the decoder input that will be accepted if the signal
01001
       strength is below this level it is not evaluated
01002
                 unsigned char MIN_LEVEL :4;
01003
01004
              unsigned char value;
01005
          };
01006
01012
          union DEMODbits {
01013
01014
              struct {
01015
01016
                  // Changes the time-constant of the internal PLL during burst
01017
                  unsigned char TAU SYNC :2;
01018
01019
                  // Changes the time-constant of the internal PLL during data reception
                  // Remark: if set to 00b the PLL is frozen during data reception
01020
01021
                  unsigned char TAU_RCV :2;
01022
01023
                  // If set to logic 0 the following formula is used to calculate the timer frequency of the
       prescaler:
01024
                  // 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:
01026
                  // F_timer = 13.56 MHz / (2*TPreScaler+2)
01027
                  unsigned char T_PRESCAL_EVEN :1;
01028
                  // If AddIQ[1:0] are set to XOb, the reception is fixed to I channel
01029
01030
                  // If AddIQ[1:0] are set to X1b, the reception is fixed to Q channel
01031
                  unsigned char FIX IO :1;
01032
01033
                  // Defines the use of {\tt I} and {\tt Q} channel during reception
01034
                   // Remark: the FixIQ bit must be set to logic 0 to enable the following settings:
                  // 00: selects the stronger channel
01035
```

```
01036
                      01: selects the stronger channel and freezes the selected channel during communication
                  unsigned char ADD_IQ :2;
01037
01038
              };
01039
              unsigned char value;
01040
          };
01041
01047
          union MF_TXbits {
01048
01049
              struct {
01050
01051
                  // Defines the additional response time 7 bits are added to the value of the register bit by
       default
01052
                  unsigned char TX_WAIT :2;
01053
01054
                  // Reserved
01055
                  unsigned char :6;
01056
              };
01057
              unsigned char value;
01058
          };
01059
01065
          union MF_RXbits {
01066
01067
              struct {
01068
01069
                  // Reserved
01070
                  unsigned char :4;
01071
01072
                  // Generation of the parity bit for transmission and the parity check for receiving is switched
       off
01073
                  // the received parity bit is handled like a data bit
01074
                  unsigned char PARITY_DISABLE :1;
01075
01076
                   // Reserved
01077
                  unsigned char :3;
01078
01079
              unsigned char value;
01080
          };
01081
01087
          union SERIAL_SPEEDbits {
01088
01089
              struct {
01090
                  // Factor BR_T1 adjusts the transfer speed
01091
01092
                  unsigned char BR_T1 :5;
01093
01094
                  // Factor BR_TO adjusts the transfer speed
01095
                  unsigned char BR_T0 :3;
01096
              };
01097
              unsigned char value:
01098
          };
01099
01105
          union RF_CFGbits {
01106
01107
              struct {
01108
                  // Reserved
01109
01110
                  unsigned char :4;
01111
01112
                   // Defines the receiver's signal voltage gain factor:
01113
                      000: 18 dB
                      001: 23 dB
01114
01115
                  //
                      010: 18 dB
01116
                      011: 23 dB
01117
                      100: 33 dB
01118
                      101: 38 dB
01119
                      110: 43 dB
01120
                      111: 48 dB
                  unsigned char RX_GAIN :3;
01121
01122
01123
                   // Reserved
01124
                  unsigned char :1;
01125
              };
01126
01127
              unsigned char value;
          };
01128
01134
          union GS_Nbits {
01135
01136
01137
                  // Defines the conductance of the output n-driver during periods without modulation which can
01138
       be used to regulate the modulation index
01139
                  // Remark: the conductance value is binary weighted during soft Power-down mode the highest bit
       is forced to logic 1
01140
                   // value is only used if driver TX1 or TX2 is switched on
01141
                  unsigned char MOD_GS_N :4;
01142
01143
                  // defines the conductance of the output n-driver during periods without modulation which can
```

70 CONTENTS

```
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
01144
01145
       is forced to logic 1
01146
                  // value is only used if driver TX1 or TX2 is switched on
                  unsigned char CW_GS_N :4;
01147
01148
              };
01149
              unsigned char value;
01150
          };
01151
          union CW GS Pbits {
01157
01158
01159
              struct {
01160
01161
                  // defines the conductance of the p-driver output which can be used to regulate the output
       power and subsequently current consumption and operating distance
01162
                  // Remark: the conductance value is binary weighted during soft Power-down mode the highest bit
       is forced to logic 1
01163
                  unsigned char CW_GS_P :6;
01164
01165
                  // Reserved
01166
                  unsigned char :2;
01167
              }:
01168
              unsigned char value:
01169
          };
01170
01176
          union MOD_GS_Pbits {
01177
01178
              struct {
01179
                  // Defines the conductance of the p-driver output during modulation which can be used to
01180
       regulate the modulation index
01181
                  // Remark: the conductance value is binary weighted during soft Power-down mode the highest bit
       is forced to logic 1
01182
                 // if the TxASKReg register's Force100ASK bit is set to logic 1 the value of ModGsP has no
       effect
01183
                  unsigned char MOD GS P :6;
01184
01185
                  // Reserved
01186
                  unsigned char :2;
01187
              };
01188
              unsigned char value:
01189
          };
01190
01198
          union T_MODEbits {
01199
01200
              struct {
01201
01202
                  // Defines the higher 4 bits of the TPrescaler value
                  // The following formula is used to calculate the timer frequency if the DemodReg register's
01203
       TPrescalEven bit in Demot Regis set to logic 0:
01204
                  // F_timer = 13.56 MHz / (2*TPreScaler+1)
01205
                  // Where TPreScaler = [TPrescaler_Hi:TPrescaler_Lo] (TPrescaler value on 12 bits) (Default
       TPrescalEven bit is logic 0)
01206
                  ^{\prime\prime} The following formula is used to calculate the timer frequency if the DemodReg register's
       01207
01208
                  unsigned char T_PRESCALER_HI :4;
01209
                  // 1: timer automatically restarts its count-down from the 16-bit timer reload value instead of
01210
       counting down to zero
01211
                  // 0: timer decrements to 0 and the ComIrqReg register's TimerIRq bit is set to logic 1
01212
                  unsigned char T_AUTO_RESTART :1;
01213
01214
                  // Internal timer is running in gated mode
01215
                  // Remark: in gated mode, the StatuslReg register's TRunning bit is logic 1 when the timer is
       enabled by the
01216
                  // TModeReg register's TGated[1:0] bits this bit does not influence the gating signal
01217
                  // 00: non-gated mode
                  // 01: gated by pin MFIN
                  // 10: gated by pin AUX1
01219
01220
                  unsigned char T_GATED :2;
01221
                  // 1: timer starts automatically at the end of the transmission in all communication modes at
01222
       all speeds
                  // if the RxModeReg register's RxMultiple bit is not set, the timer stops immediately after
01223
       receiving the 5th bit (1 start bit, 4 data bits)
01224
                  // 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
01225
                  // TStopNow bit to logic 1
                  ^{\prime\prime} 0: indicates that the timer is not influenced by the protocol
01226
                  unsigned char T_AUTO :1;
01228
01229
              unsigned char value;
01230
          } ;
01231
          union VERSIONbits {
01237
```

```
01238
01239
             struct {
01240
                  // '1' stands for MFRC522 version 1.0 and '2' stands for MFRC522 version 2.0.
01241
01242
                  unsigned char VERSION :4;
01243
                  // '9' stands for MFRC522
01244
01245
                  unsigned char CHIPTYPE :4;
01246
01247
              unsigned char value;
01248
         };
01249
01250
         struct Uid {
01251
01252
              // Number of bytes in the UID. 4, 7 or 10.
01253
              unsigned char size;
01254
01255
              unsigned char uid[10];
01256
01257
              // The SAK (Select acknowledge) byte returned from the PICC after successful selection.
01258
              unsigned char sak;
01259
         };
01260
          enum Version {
01261
01262
             CLONE = 0x88,
01263
              V0_0 = 0x90,
01264
             V1_0 = 0x91,
01265
              V2_0 = 0x92
01266
         };
01267
01268
          RadioFrequencyIdentificationMFRC522(RegisterBasedDevice *device,
01269
                  unsigned char resetPin);
01270
01271
          void initialize();
01272
          void sendCommand(Command command) {
01273
01274
             writeRegister(COMMAND, (unsigned char) command);
01275
01276
01277
          unsigned char getLastError() {
01278
              return lastError;
01279
          }
01280
01281
          void clearLastError() {
01282
             lastError = NO_ERROR;
01283
01284
01285
          void softReset();
01286
01287
          void setAntennaOn();
01288
01289
          void setAntennaOff();
01290
01331
          void configureTimer(unsigned int prescaler, unsigned int reload, bool autoStart,
01332
                  bool autoRestart);
01333
01334
          void startTimer();
01335
01336
          void stopTimer();
01337
01341
          void enableInterrupt(Interrupt interrupt);
01342
01343
          void disableInterrupt(Interrupt interrupt);
01344
01345
          void clearInterrupt(Interrupt interrupt);
01346
01347
          void flushQueue();
01348
01349
          void setWaterLevel(unsigned char level);
01350
01351
          int generateRandomId(unsigned char buf[10]);
01352
01353
          int communicate(Command, unsigned char *output, unsigned char *input,
01354
                  unsigned char outputLen, bool checkCRC);
01355
01356
          unsigned int calculateCRC (unsigned char *buff, unsigned char len);
01357
          bool waitForRegisterBits(unsigned char reg, unsigned char mask) {
01358
01359
              return waitForRegisterBits(reg, mask,
     MFRC522 DEFAULT TIMEOUT):
01360
01361
          bool waitForRegisterBits(unsigned char reg, unsigned char mask, unsigned long
01362
      timeout);
01363
01364
          Version getVersion();
01365
```

72 CONTENTS

Index

ADD_IQ	RadioFrequencyIdentificationMFRC522, 27
RadioFrequencyIdentificationMFRC522::DEMO ←	COM_IRQ_ALL_IRQ
Dbits, 13	RadioFrequencyIdentificationMFRC522, 28
ANALOG TEST	COM_IRQ_ERR_IRQ
RadioFrequencyIdentificationMFRC522, 29	RadioFrequencyIdentificationMFRC522, 28
AUTO TEST ENABLE	COM IRQ HI ALERT IRQ
RadioFrequencyIdentificationMFRC522, 28	RadioFrequencyIdentificationMFRC522, 28
AUTO_TEST	COM_IRQ_IDLE_IRQ
RadioFrequencyIdentificationMFRC522, 29	
riadioi requericy identification wir 110322, 29	RadioFrequencyIdentificationMFRC522, 28
BIT_FRAMING_START_SEND	COM_IRQ_LO_ALERT_IRQ
RadioFrequencyIdentificationMFRC522, 28	RadioFrequencyIdentificationMFRC522, 28
	COM_IRQ_RX_IRQ
BIT_FRAMING	RadioFrequencyIdentificationMFRC522, 28
RadioFrequencyIdentificationMFRC522, 29	COM_IRQ_SET1
BR_T0	RadioFrequencyIdentificationMFRC522, 28
RadioFrequencyIdentificationMFRC522::SERIA↔	COM_IRQ_TIMER_IRQ
L_SPEEDbits, 38	RadioFrequencyIdentificationMFRC522, 28
BR_T1	COM IRQ TX IRQ
RadioFrequencyIdentificationMFRC522::SERIA ←	RadioFrequencyIdentificationMFRC522, 28
L_SPEEDbits, 38	COM IRQ
BUFFER_OVFL	RadioFrequencyIdentificationMFRC522, 28
RadioFrequencyIdentificationMFRC522::ERRO ←	
Rbits, 17	COM_LO_ALERT_IRQ
	RadioFrequencyIdentificationMFRC522, 27
CALC_CRC	COM_RX_IRQ
RadioFrequencyIdentificationMFRC522, 27	RadioFrequencyIdentificationMFRC522, 27
CHIPTYPE	COM_TIMER_IRQ
RadioFrequencyIdentificationMFRC522::VERSI←	RadioFrequencyIdentificationMFRC522, 27
ONbits, 48	COM_TX_IRQ
CLONE	RadioFrequencyIdentificationMFRC522, 27
RadioFrequencyIdentificationMFRC522, 30	COMMAND
COLL_ERR	RadioFrequencyIdentificationMFRC522, 28
RadioFrequencyIdentificationMFRC522::ERRO↔	RadioFrequencyIdentificationMFRC522::COMM ←
	ANDbits, 10
Rbits, 17	COMMUNICATION_ERROR
COLL_LEVEL	RadioFrequencyIdentificationMFRC522, 27
RadioFrequencyIdentificationMFRC522::RX_TH □ PERIOD RIVINGS	CONTROL T START NOW
RESHOLDbits, 37	RadioFrequencyIdentificationMFRC522, 28
COLL_POS_NOT_VALID	• •
RadioFrequencyIdentificationMFRC522::COLLbits,	CONTROL_T_STOP_NOW
7	RadioFrequencyIdentificationMFRC522, 28
COLL_POS	CONTROL
RadioFrequencyIdentificationMFRC522::COLLbits,	RadioFrequencyIdentificationMFRC522, 29
7	CRC_ERROR
COLL	RadioFrequencyIdentificationMFRC522, 27
RadioFrequencyIdentificationMFRC522, 29	CRC_ERR
COM_ALL_IRQ	RadioFrequencyIdentificationMFRC522::ERRO ←
RadioFrequencyIdentificationMFRC522, 27	Rbits, 17
COM_ERR_IRQ	CRC_I_EN
RadioFrequencyIdentificationMFRC522, 27	RadioFrequencyIdentificationMFRC522::DIV_I_←
COM_HI_ALERT_IRQ	ENbits, 14
RadioFrequencyIdentificationMFRC522, 27	CRC_IRQ
COM_I_EN_INTERRUPT_EN	RadioFrequencyIdentificationMFRC522::DIV_IR
RadioFrequencyIdentificationMFRC522, 28	Qbits, 15
COM_I_EN	CRC_OK
RadioFrequencyIdentificationMFRC522, 28	RadioFrequencyIdentificationMFRC522::STATU←
COM IDLE IRQ	S1bits, 39

CRC_PRESET	RadioFrequencyIdentificationMFRC522::TX_MO
RadioFrequencyIdentificationMFRC522::MOD ←	DEbits, 45
Ebits, 22	RadioFrequencyIdentificationMFRC522::TX_SE
CRC_READY	Lbits, 46
RadioFrequencyIdentificationMFRC522::STATU ←	RadioFrequencyIdentificationMFRC522::WATE ←
S1bits, 39	R_LEVELbits, 49
CRC_RESULT_HIDH	clearInterrupt RadioFrequencyIdentificationMFRC522, 30
RadioFrequencyIdentificationMFRC522, 29	clearLastError
CRC_RESULT_LOW	RadioFrequencyIdentificationMFRC522, 30
RadioFrequencyIdentificationMFRC522, 29	Command
CW_GS_N	RadioFrequencyIdentificationMFRC522, 27
RadioFrequencyIdentificationMFRC522::GS_←	communicate
Nbits, 19	RadioFrequencyIdentificationMFRC522, 30
CW_GS_P	configureTimer
RadioFrequencyIdentificationMFRC522, 29	RadioFrequencyIdentificationMFRC522, 30
RadioFrequencyIdentificationMFRC522::CW_G S. Phita 12	
S_Pbits, 12 calculateCRC	DEMOD
RadioFrequencyIdentificationMFRC522, 30	RadioFrequencyIdentificationMFRC522, 29
char	DIV_ALL_IRQ
RadioFrequencyIdentificationMFRC522::BIT_FR←	RadioFrequencyIdentificationMFRC522, 27
AMINGbits, 6	DIV_CRC_IRQ
RadioFrequencyIdentificationMFRC522::COLLbits,	RadioFrequencyIdentificationMFRC522, 27
7	DIV_I_EN_CRC_I_EN
RadioFrequencyIdentificationMFRC522::COMM←	RadioFrequencyIdentificationMFRC522, 28
ANDbits, 10	DIV_I_EN_INTERRUPT_EN
RadioFrequencyIdentificationMFRC522::CONT←	RadioFrequencyIdentificationMFRC522, 28
ROLbits, 11	DIV_I_EN_MFIN_ACT_I_EN
RadioFrequencyIdentificationMFRC522::CW_G↔	RadioFrequencyIdentificationMFRC522, 28
S_Pbits, 12	DIV_I_EN
RadioFrequencyIdentificationMFRC522::DIV_I_←	RadioFrequencyIdentificationMFRC522, 28
ENbits, 14	DIV_IRQ_ALL_IRQ
RadioFrequencyIdentificationMFRC522::DIV_IR↔	RadioFrequencyIdentificationMFRC522, 28
Qbits, 15	DIV_IRQ_CRC_IRQ RadioFrequencyIdentificationMFRC522, 28
RadioFrequencyIdentificationMFRC522::ERRO ←	DIV_IRQ_MFIN_ACT_IRQ
Rbits, 17	RadioFrequencyIdentificationMFRC522, 28
RadioFrequencyIdentificationMFRC522::MF_R ←	DIV_IRQ_SET2
Xbits, 20	RadioFrequencyIdentificationMFRC522, 28
RadioFrequencyIdentificationMFRC522::MF_T ←	DIV_IRQ
Xbits, 20	RadioFrequencyIdentificationMFRC522, 28
RadioFrequencyIdentificationMFRC522::MOD_←	DIV_MFIN_ACT_IRQ
GS_Pbits, 21	RadioFrequencyIdentificationMFRC522, 27
RadioFrequencyIdentificationMFRC522::MOD ←	device
Ebits, 22	RadioFrequencyIdentificationMFRC522, 34
RadioFrequencyIdentificationMFRC522::RF_CF←	disableInterrupt
Gbits, 34	RadioFrequencyIdentificationMFRC522, 31
RadioFrequencyIdentificationMFRC522::RX_M←	
ODEbits, 35	ERR_I_EN
RadioFrequencyIdentificationMFRC522::RX_TH ←	RadioFrequencyIdentificationMFRC522::COM_I ←
RESHOLDbits, 37	_ENbits, 8
RadioFrequencyIdentificationMFRC522::STATU←	ERR_IRQ
S1bits, 39	RadioFrequencyIdentificationMFRC522::COM_I
RadioFrequencyIdentificationMFRC522::STATU ←	RQbits, 9
S2bits, 41	ERROR
RadioFrequencyIdentificationMFRC522::TX_AS ←	RadioFrequencyIdentificationMFRC522, 28
Kbits, 43	enableInterrupt
RadioFrequencyIdentificationMFRC522::TX_CO NTROLbits, 44	RadioFrequencyIdentificationMFRC522, 31
INTROLUIS. 44	Error

RadioFrequencyIdentificationMFRC522, 27	IDLE
EIEO DATA	RadioFrequencyIdentificationMFRC522, 27
FIFO_DATA RadioFrequencyIdentificationMFRC522, 28	INV_MOD
FIFO LEVEL FIFO LEVEL	RadioFrequencyIdentificationMFRC522::TX_MO← DEbits, 45
RadioFrequencyIdentificationMFRC522, 28	INV_TX1_RF_OFF
FIFO_LEVEL_FLUSH_BUFFER	RadioFrequencyIdentificationMFRC522::TX_CO ←
RadioFrequencyIdentificationMFRC522, 28	NTROLbits, 44
FIFO LEVEL	INV_TX1_RF_ON
RadioFrequencyIdentificationMFRC522, 28	RadioFrequencyIdentificationMFRC522::TX_CO←
RadioFrequencyIdentificationMFRC522::FIFO_L←	NTROLbits, 44
EVELbits, 18	INV_TX2_RF_OFF
FIX_IQ	RadioFrequencyIdentificationMFRC522::TX_CO ←
RadioFrequencyIdentificationMFRC522::DEMO ←	NTROLbits, 44
Dbits, 13	INV_TX2_RF_ON
FLUSH_BUFFER	RadioFrequencyIdentificationMFRC522::TX_CO ←
RadioFrequencyIdentificationMFRC522::FIFO_L ←	NTROLbits, 44
EVELbits, 18	INV_TX_RF_OFF
FORCE_100_ASK	RadioFrequencyIdentificationMFRC522::TX_CO ←
RadioFrequencyIdentificationMFRC522::TX_AS ←	NTROLbits, 44
Kbits, 43	INV_TX_RF_ON
flushQueue	RadioFrequencyIdentificationMFRC522::TX_CO ←
RadioFrequencyIdentificationMFRC522, 31	NTROLbits, 44
GENERAL_ERROR	IRQ_PUSH_PULL
RadioFrequencyIdentificationMFRC522, 27	RadioFrequencyIdentificationMFRC522::DIV_I_
GENERATE RANDOM ID	ENbits, 15 IRQ
RadioFrequencyIdentificationMFRC522, 27	RadioFrequencyIdentificationMFRC522::STATU←
GS N	S1bits, 40
RadioFrequencyIdentificationMFRC522, 29	initialize
generateRandomId	RadioFrequencyIdentificationMFRC522, 31
RadioFrequencyIdentificationMFRC522, 31	Interrupt
getLastError	RadioFrequencyIdentificationMFRC522, 27
RadioFrequencyIdentificationMFRC522, 31	,
getVersion	LO_ALERT_I_EN
RadioFrequencyIdentificationMFRC522, 31	$RadioFrequencyIdentificationMFRC522::COM_I {\leftarrow}$
	_ENbits, 8
HI_ALERT_I_EN	LO_ALERT_IRQ
RadioFrequencyIdentificationMFRC522::COM_I ←	RadioFrequencyIdentificationMFRC522::COM_I ←
_ENbits, 8	RQbits, 9
HI_ALERT_IRQ	LO_ALERT
RadioFrequencyIdentificationMFRC522::COM_I ←	RadioFrequencyIdentificationMFRC522::STATU ←
RQbits, 9 HI_ALERT	S1bits, 40 lastError
RadioFrequencyIdentificationMFRC522::STATU←	RadioFrequencyIdentificationMFRC522, 34
S1bits, 39	nadioi requencyldentincationivii nobzz, 54
0.15110, 00	MEM
I2C_FORCE_HS	RadioFrequencyIdentificationMFRC522, 27
RadioFrequencyIdentificationMFRC522::STATU←	MF_AUTHENT
S2bits, 41	RadioFrequencyIdentificationMFRC522, 27
I_RQ_INV	MF_CRYPTO1_ON
$RadioFrequencyIdentificationMFRC522::COM_I {\leftarrow}$	$RadioF requency Identification MFRC522 :: STATU {\leftarrow}$
_ENbits, 8	S2bits, 41
IDLE_I_EN	MF_OUT_SEL
RadioFrequencyIdentificationMFRC522::COM_I ←	RadioFrequencyIdentificationMFRC522::TX_SE↔
_ENbits, 8	Lbits, 46
IDLE_IRQ	MF_RX
RadioFrequencyIdentificationMFRC522::COM_I ←	RadioFrequencyIdentificationMFRC522, 29
RQbits, 9	MF_TX

RadioFrequencyIdentificationMFRC522, 29	PROGMEM
MFIN_ACT_I_EN Padia Frague pay Identification MEDCE 22 up IV I	RadioFrequencyIdentificationMFRC522.h, 57
RadioFrequencyIdentificationMFRC522::DIV_I_← ENbits, 15	PROTOCOL_ERR RadioFrequencyIdentificationMFRC522::ERRO←
MFIN_ACT_IRQ	Rbits, 17
RadioFrequencyIdentificationMFRC522::DIV_IR↔	performSelfTest
Qbits, 16	RadioFrequencyIdentificationMFRC522, 31
MFR522_INT_TO_EN_MASK	riadio requesto y la critina allo rivir ri co 22, o r
RadioFrequencyIdentificationMFRC522.h, 56	RCV_OFF
MFR522 INT TO EN REG	RadioFrequencyIdentificationMFRC522::COMM ←
RadioFrequencyIdentificationMFRC522.h, 56	ANDbits, 11
MFR522_INT_TO_IRQ_MASK	RECEIVE
RadioFrequencyIdentificationMFRC522.h, 56	RadioFrequencyIdentificationMFRC522, 27
MFR522_INT_TO_IRQ_REG	RFC_FG
RadioFrequencyIdentificationMFRC522.h, 57	RadioFrequencyIdentificationMFRC522, 29
MFRC522_DEFAULT_TIMEOUT	RX_ALIGN
RadioFrequencyIdentificationMFRC522.h, 57	RadioFrequencyIdentificationMFRC522::BIT_FR←
MIN_LEVEL	AMINGbits, 6
RadioFrequencyIdentificationMFRC522::RX_TH←	RX_CRC_EN RadioFrequencyIdentificationMFRC522::RX_M ←
RESHOLDbits, 37	ODEbits, 35
MOD_GS_N	RX_GAIN
RadioFrequencyIdentificationMFRC522::GS_←	RadioFrequencyIdentificationMFRC522::RF_CF↔
Nbits, 19	Gbits, 34
MOD_GS_P	RX_I_EN
RadioFrequencyIdentificationMFRC522, 29 RadioFrequencyIdentificationMFRC522::MOD_←	RadioFrequencyIdentificationMFRC522::COM_I ←
GS_Pbits, 21	_ENbits, 8
MOD WIDTH	RX_IRQ
RadioFrequencyIdentificationMFRC522, 29	RadioFrequencyIdentificationMFRC522::COM_I ←
MODEM_STATE	RQbits, 9
RadioFrequencyIdentificationMFRC522::STATU←	RX_LAST_BITS
S2bits, 41	RadioFrequencyIdentificationMFRC522::CONT←
MODE	ROLbits, 11
RadioFrequencyIdentificationMFRC522, 29	RX_MODE
MSB_FIRST	RadioFrequencyIdentificationMFRC522, 29
RadioFrequencyIdentificationMFRC522::MOD ←	RX_MULTIPLE
Ebits, 22	RadioFrequencyIdentificationMFRC522::RX_M ORF: 12.05
Mask	ODEbits, 35
RadioFrequencyIdentificationMFRC522, 27	RX_NO_ERR
	RadioFrequencyIdentificationMFRC522::RX_M←
NO_CMD_CHANGE	ODEbits, 36
RadioFrequencyIdentificationMFRC522, 27	RX_SEL PadioFraguencyIdentificationMEPC522, 20
NO_ERROR	RadioFrequencyIdentificationMFRC522, 29 RX_SPEED
RadioFrequencyIdentificationMFRC522, 27	RadioFrequencyIdentificationMFRC522::RX_M↔
NONE_IRQ RadioFrequencyIdentificationMFRC522, 27	ODEbits, 36
nadior requency identification with nodez, 27	RX_THRESHOLD
PARITY DISABLE	RadioFrequencyIdentificationMFRC522, 29
RadioFrequencyIdentificationMFRC522::MF_R↔	RX_WAIT
Xbits, 20	
PARITY_ERR	Lbits, 36
RadioFrequencyIdentificationMFRC522::ERRO←	RadioFrequencyIdentification, 23
Rbits, 17	RadioFrequencyIdentification.cpp, 49, 50
POL_M_FIN	RadioFrequencyIdentification.h, 50
RadioFrequencyIdentificationMFRC522::MOD←	RadioFrequencyIdentificationMFRC522, 24
Ebits, 22	ANALOG_TEST, 29
POWER_DOWN	AUTO_TEST_ENABLE, 28
RadioFrequencyIdentificationMFRC522::COMM ←	AUTO_TEST, 29
ANDbits, 11	BIT_FRAMING_START_SEND, 28

BIT_FRAMING, 29 CALC_ORC, 27 CLONE, 30 CALC_ORC, 27 CLONE, 30 COLL, 29 COM_ALL_IRQ, 27 COM_ERR_IRQ, 27 COM_ERR_IRQ, 27 COM_IR_LENT_IRQ, 27 COM_IR_LEN_INTERRUPT_EN, 28 COM_IRQ_ALL_IRQ, 27 COM_IRQ_ALL_IRQ, 28 COM_IRQ_ALL_IRQ, 27 COM_TX_IRQ, 29 CRC_ERSULT_LOW_29 CRC_ERSULT_LOW_29 CRC_ERSULT_LOW_29 CRC_ERSULT_LOW_29 CRC_ERSULT_LOW_29 CRC_ERSULT_LOW_29 CRC_ERSULT_ICQ, 27 DIV_CRC_IRQ, 28 DIV_IRQ, 27 DIV_CRC_IRQ, 28 DIV_IRQ, 27 DIV_CRC_IRQ, 28 DIV_IRQ, 27 DIV_CRC_IRQ, 28 DIV_IRQ, 27 T_RELOAD_IRQ, 29 T_RELOAD_IRQ, 29 T_RELOAD_IRQ, 29 T_RELOAD_IRQ, 29 T_RELOAD_IRQ, 29 T_SET_DAC, 29 TEST_DAC, 29 TEST_DAC		
CLONE, 30 COLL, 29 COM, 29 COM, 29 COM, 27 COM, ERR, IRQ, 27 COM, ILALERT, IRQ, 27 COM, ILALERT, IRQ, 27 COM, ILALERT, IRQ, 28 COM, IEN, INTERRUPT_EN, 28 COM, IEN, INTERRUPT_EN, 28 COM, IRQ, LEN, 28 COM, IRQ, ILALERT, IRQ, 28 COM, IRQ, INC, ALERT, IRQ, 28 COM, IRQ, INC, 28 COM, IRQ, INC, INC, 28 COM, IRQ, INC, INC, 28 COM, IRQ, INC, INC, 29 COM, IRQ, IRQ, 27 COM, IRQ, IRQ, 27 COM, IRQ, 18 COM, IRQ, 27 COM, IRQ, 18 COM, IRQ, 27 COM, IRQ, 27 COM, IRQ, 27 COM, IRQ, 27 COM, INC, IRQ, 27 COM, IRQ, 29 COM, IR		FIFO_LEVEL_FIFO_LEVEL, 28
COLL, 29 COM_ALL_IRQ, 27 COM_ERR_IRQ, 27 COM_ERR_IRQ, 27 COM_HI_ALERT_IRQ, 27 COM_HI_ALERT_IRQ, 28 COM_IEN, 28 COM_IEN, 28 COM_IEN, 28 COM_IEN, 28 COM_IEN, 28 COM_IEN, 28 COM_IRQ, 27 COM_IRQ, 28 COM_IRQ, 21 COM_IRQ, 28 COM_IRQ, 19 COM_IRQ, 28 COM_IRQ, 29 COM_IRQ, 28 COM_IRQ, 27 COM_IRQ, 28 COM_IRQ, 29 CRC_ERROR, 27 CONTROL_T_STOP_NOW, 28 CRC_ERROR, 27 CRC_RESULT_LOW, 29 CRC_ERROR, 27 COM_IRQ, 29 CRC_ERROR, 27 COM_IRQ, 29 CRC_ERROR, 27 COM_IRQ, 29 CRC_ERROR, 27 COM_IRQ, 29 CRC_IRQ, 27 COM_IRQ, 28 DIV_IRQ, 29 T_RELOAD_LOW, 29 T_RELOAD_LOW, 29 T_RELOAD_LOW, 29 T_RELOAD_LOW, 29 T_REST_DAC, 29 TEST_DAC, 29	CALC_CRC, 27	FIFO_LEVEL_FLUSH_BUFFER, 28
COM ALL JRQ, 27 COM JEN JRQ, 27 COM JEN JRTERUPT_EN, 28 COM_JEN, 29 COM_JEN, 29 COM_LON, 28 COM_JEN, 27 COM_TX_JEN, 29 COM_TX_JEN, 29 COM_JEN, 27 COM_TX_JEN, 29 COM_TX_JEN, 29 COM_JEN, 2	CLONE, 30	FIFO_LEVEL, 28
COM_ERR_IRQ_27 COM_EN_INTERRUPT_EN_28 COM_IEN_28 COM_IEN_28 COM_IEN_28 COM_IEN_27 COM_IRQ_ALL_IRQ_28 COM_IRQ_ALL_IRQ_28 COM_IRQ_IDLE_IRQ_28 COM_IRQ_IDLE_IRQ_28 COM_IRQ_IDLE_IRQ_28 COM_IRQ_IDLE_IRQ_28 COM_IRQ_IDLE_IRQ_28 COM_IRQ_IDLE_IRQ_28 COM_IRQ_IDLE_IRQ_28 COM_IRQ_IDLE_IRQ_28 COM_IRQ_IDLE_IRQ_28 COM_IRQ_TIMER_IRQ_28 COM_IRQ_XIRQ_28 COM_IRQ_XIRQ_28 COM_IRQ_XIRQ_28 COM_IRQ_XIRQ_28 COM_IRQ_XIRQ_28 COM_IRQ_TIMER_IRQ_28 COM_IRQ_XIRQ_28 COM_IRQ_TIMER_IRQ_28 COM_IRQ_XIRQ_28 COM_IRQ_TIMER_IRQ_27 COM_TX_IRQ_27 COM_TX_IRQ_27 COM_TX_IRQ_27 COM_TX_IRQ_28 COM_IRQ_XIRQ_29 CRC_ERGOR_27 COM_TX_IRQ_29 CRC_ERGOR_27 COM_TX_IRQ_29 CRC_ERGOR_27 COM_TX_IRQ_29 CRC_ERGOR_27 CRC_RESULT_LOW_29 CRC_RESULT_R	COLL, 29	flushQueue, 31
COM_H_ALERT_IRQ, 27 COM_IEN_INTERRUPT_EN, 28 COM_IEN_INTERRUPT_EN, 28 COM_IEN_INTERRUPT_EN, 28 COM_IDLE_IRQ, 27 COM_IRQ_ALL_IRQ, 28 COM_IRQ_ALL_IRQ, 28 COM_IRQ_BER_IRQ, 28 COM_IRQ_IDLE_IRQ, 28 COM_IRQ_IDLE_IRQ, 28 COM_IRQ_IDLE_IRQ, 28 COM_IRQ_IDLE_IRQ, 28 COM_IRQ_IDLE_IRQ, 28 COM_IRQ_IDLE_IRQ, 28 COM_IRQ_TMER_IRQ, 27 COM_TMER_IRQ, 28 COMTROL_T_STOP_NOW, 28 COM_TRQ_L_TSTOP_NOW, 29 COM_TRQ_L_TSTOP_NOW, 28 COM_TRQ_L_TRQ_L_TSTOP_NOW, 28 COM_TRQ_L_TRQ_L_TSTOP_NOW, 28 COM_TRQ_L_TRQ_L_TSTOP_NOW, 28 COM_TRQ_L_TRQ_L_TSTOP_NOW, 28 COM_TRQ_L_TRQ_L_TSTOP_NOW, 28 COM_TRQ_L_TRQ_L_TRQ_L_TSTOP_NOW, 28 COM_TRQ_L_TRQ_L_TRQ_L_TRQ_L_TRQ	COM_ALL_IRQ, 27	GENERAL_ERROR, 27
COM_LEN_28 COM_DLE_IRQ. 27 COM_IRQ_ALL_IRQ. 28 COM_IRQ_ALL_IRQ. 28 COM_IRQ_ALL_IRQ. 28 COM_IRQ_DLE_IRQ. 28 COM_IRQ_SET1, 28 COM_IRQ_SET1, 28 COM_IRQ_SET1, 28 COM_IRQ_TIMER_IRQ. 28 COM_IRQ_TIMER_IRQ. 28 COM_IRQ_TX_IRQ. 28 COM_IRQ_TX_IRQ. 28 COM_IRQ_TX_IRQ. 27 COM_IRQ_R2 COM_IRQ_TX_IRQ. 27 COM_TX_IRQ. 28 COM_TX_IRQ. 27 COM_TX_IRQ. 29 CRC_EROR_27 CRC_RESULT_LOW_29 CRC_EROR_27 CRC_RESULT_LOW_29 CRC_EROR_27 CRC_RESULT_LOW_29 CRC_EROR_27 COM_TX_IRQ. 27 COM_TX_IRQ. 28 COM_TX_IRQ. 28 COM_TX_IRQ. 28 COM_TX_IRQ. 28 COM_TX_IRQ. 28 COM_TX_IRQ. 28 COM_TX_IRQ. 29 TX_EX_IRQ.	COM_ERR_IRQ, 27	GENERATE_RANDOM_ID, 27
COM_LEN_28 COM_DLE_IRQ. 27 COM_IRQ_ALL_IRQ. 28 COM_IRQ_ALL_IRQ. 28 COM_IRQ_ALL_IRQ. 28 COM_IRQ_DLE_IRQ. 28 COM_IRQ_SET1, 28 COM_IRQ_SET1, 28 COM_IRQ_SET1, 28 COM_IRQ_TIMER_IRQ. 28 COM_IRQ_TIMER_IRQ. 28 COM_IRQ_TX_IRQ. 28 COM_IRQ_TX_IRQ. 28 COM_IRQ_TX_IRQ. 27 COM_IRQ_R2 COM_IRQ_TX_IRQ. 27 COM_TX_IRQ. 28 COM_TX_IRQ. 27 COM_TX_IRQ. 29 CRC_EROR_27 CRC_RESULT_LOW_29 CRC_EROR_27 CRC_RESULT_LOW_29 CRC_EROR_27 CRC_RESULT_LOW_29 CRC_EROR_27 COM_TX_IRQ. 27 COM_TX_IRQ. 28 COM_TX_IRQ. 28 COM_TX_IRQ. 28 COM_TX_IRQ. 28 COM_TX_IRQ. 28 COM_TX_IRQ. 28 COM_TX_IRQ. 29 TX_EX_IRQ.	COM HI ALERT IRQ, 27	GS N, 29
COM_IEN_28 COM_IDLE_IRQ, 27 COM_IRQ_ALL_IRQ, 28 COM_IRQ_ALL_IRQ, 28 COM_IRQ_DIA_LERT_IRQ, 28 COM_IRQ_SETI, 28 COM_IRQ_SETI, 28 COM_IRQ_SETI, 28 COM_IRQ_SETI, 28 COM_IRQ_TA_IRQ, 28 COM_IRQ_TA_IRQ, 28 COM_IRQ_TA_IRQ, 28 COM_IRQ_TA_IRQ, 27 COM_TAX_IRQ, 29 COM_CR_ESULT_ILUM, 29 CRC_RESULT_ILUM, 29 CRC_RESULT_LOW, 29 CRC_RESULT_LOW, 29 CRC_RESULT_LOW, 29 CRC_RESULT_LOW, 29 CRC_RESULT_LOW, 29 COM_GS_P, 29 calculateCRC, 30 clearLastError, 30 ClearLastError, 30 Command, 27 communicate, 30 configureTimer, 30 DEMOD_2 9 DIV_ALL_IRQ, 27 DIV_CRC_IRQ, 27 DIV_IRQ_NED_CRC_LEN_28 DIV_IRQ_CRC_LEN_28 DIV_IRQ_CRC_LEN_28 DIV_IRQ_CRC_LEN_28 DIV_IRQ_CRC_LEN_28 DIV_IRQ_CRC_RESULT_IRQ, 28 DIV_IRQ_MFIN_ACT_IRQ, 28 DIV_IRQ_MFIN_ACT_IRQ, 28 DIV_IRQ_MFIN_ACT_IRQ, 28 DIV_IRQ_MFIN_ACT_IRQ, 28 DIV_IRQ_MFIN_ACT_IRQ, 28 DIV_IRQ_BET2, 28 DIV_IRQ_MFIN_ACT_IRQ, 28 DIV_IRQ_BET2, 28 DIV_IRQ_		
COM_IRQ_ALL_IRQ, 27 COM_IRQ_ERR_IRQ, 28 COM_IRQ_HI_ALERT_IRQ, 28 COM_IRQ_HI_ALERT_IRQ, 28 COM_IRQ_HI_ALERT_IRQ, 28 COM_IRQ_IDLE_IRQ, 28 COM_IRQ_IDLE_IRQ, 28 COM_IRQ_IDLE_IRQ, 28 COM_IRQ_IDLE_IRQ, 28 COM_IRQ_IN_IRQ, 28 COM_IRQ_SETI, 28 COM_IRQ_SETI, 28 COM_IRQ_TX_IRQ, 28 COM_IRQ_TX_IRQ, 28 COM_IRQ_TX_IRQ, 28 COM_IRQ_TX_IRQ, 28 COM_IRQ_TX_IRQ, 27 COM_IRQ_R2 COM_IRQ_TX_IRQ, 27 COM_IX_IRQ, 27 COM_IX_IRQ_IX_IX_IX_IX_IX_IX_IX_IX_IX_IX_IX_IX_IX_		
COM_IRQ_ALL_IRQ, 28 COM_IRQ_H_ALERT_IRQ, 28 COM_IRQ_IDLE_IRQ, 28 COM_IRQ_IDLE_IRQ, 28 COM_IRQ_IDLE_IRQ, 28 COM_IRQ_IDLE_IRQ, 28 COM_IRQ_RX_IRQ, 28 COM_IRQ_RX_IRQ, 28 COM_IRQ_RX_IRQ, 28 COM_IRQ_RX_IRQ, 28 COM_IRQ_TMER_IRQ, 28 COM_IRQ_TMER_IRQ, 28 COM_IRQ_TMER_IRQ, 28 COM_IRQ_TX_IRQ, 28 COM_IRQ_TX_IRQ, 28 COM_IRQ_TX_IRQ, 28 COM_IRQ_RX_IRQ, 27 COM_RX_IRQ, 27 COM_RX_IRQ, 27 COM_TMER_IRQ, 27 COM_TMER_IRQ, 27 COM_TMER_IRQ, 27 COM_TMER_IRQ, 27 COM_TMER_IRQ, 27 COM_TMIRC_IRQ, 27 COM_TMIRC_IRQ, 27 COM_TMIRC_IRQ, 27 COM_TMIRC_IRQ, 27 COM_TMIRC_IRQ, 27 COM_TMIRC_IRQ, 27 COM_TMIRQ, 27 COM_TMIRC_TIRQ, 28 COM_IRQ_IRQ_TRQ_TRQ_TRQ_TRQ_TRQ_TRQ_TRQ_TRQ_TRQ_T	 ·	
COM_IRQ_ERR_IRQ, 28 COM_IRQ_LD_ALERT_IRQ, 28 COM_IRQ_LD_ALERT_IRQ, 28 COM_IRQ_LD_ALERT_IRQ, 28 COM_IRQ_LD_ALERT_IRQ, 28 COM_IRQ_SET1, 28 COM_IRQ_SET1, 28 COM_IRQ_SET1, 28 COM_IRQ_SET1, 28 COM_IRQ_TMER_IRQ, 28 COM_IRQ_TMER_IRQ, 28 COM_IRQ_TMER_IRQ, 28 COM_IRQ_TMER_IRQ, 28 COM_IRQ_TMER_IRQ, 28 COM_IRQ_TMER_IRQ, 27 COM_RD_ALERT_IRQ, 27 COM_TX_IRQ, 27 COMMUNICATION_ERROR, 27 COMTROL_T_START_NOW, 28 CONTROL_T_START_NOW, 28 CONTROL_T_STOP_NOW, 28 CONTROL_T_STOP_NOW, 28 CONTROL_T_STOP_NOW, 29 CRC_ERROR, 27 CRC_RESULT_LOW, 29 CRC_RESULT_REN_RESULT_REN_RESULT		-
COM_IRQ_HI_ALERT_IRQ, 28 COM_IRQ_DLE_IRQ, 28 COM_IRQ_DLE_IRQ, 28 COM_IRQ_DALERT_IRQ, 28 COM_IRQ_RX_IRQ, 28 COM_IRQ_RX_IRQ, 28 COM_IRQ_TIMER_IRQ, 28 COM_IRQ_TIMER_IRQ, 28 COM_IRQ_TX_IRQ, 28 COM_IRQ_TX_IRQ, 28 COM_IRQ_TX_IRQ, 28 COM_IRQ_TX_IRQ, 28 COM_IRQ_TX_IRQ, 28 COM_IRQ_TX_IRQ, 27 COM_TX_IRQ, 29 RX_IRQ_IRQ_TY COM_TX_IRQ, 29 RX_IRQ_IRQ_IRQ_IRQ_IRQ_IRQ_IRQ_IRQ_IRQ_IRQ	,	•
COM_IRQ_IDLE_IRQ, 28 COM_IRQ_LO_ALERT_IRQ, 28 COM_IRQ_NX_IRQ, 28 COM_IRQ_SET1, 28 COM_IRQ_SET1, 28 COM_IRQ_SET1, 28 COM_IRQ_TMLRET_IRQ, 28 COM_IRQ_TMLRET_IRQ, 28 COM_IRQ_TX_IRQ, 28 COM_IRQ_TX_IRQ, 28 COM_IRQ_TX_IRQ, 28 COM_IRQ, 28 COM_IRQ, 28 COM_IRQ, 28 COM_IRQ, 28 COM_IRQ, 27 COM_RX_IRQ, 27 COM_RX_IRQ, 27 COM_TX_IRQ, 27 CRC_RESULT_HIDH, 29 CRC_ERROR, 27 CRC_RESULT_LOW, 29 CRC_RESULT_LOW, 29 CRC_RESULT_LOW, 29 CRC_RESULT_HOW, 29 CRC_RESULT_BROR, 27 COMMAND, 28 Command, 27 Communicate, 30 Command,		
COM_IRQ_LO_ALERT_IRQ, 28 COM_IRQ_RX_IRQ, 28 COM_IRQ_SET1, 28 COM_IRQ_SET1, 28 COM_IRQ_SET1, 28 COM_IRQ_TX_IRQ, 28 COM_IRQ_TX_IRQ, 28 COM_IRQ_TX_IRQ, 28 COM_IRQ_TX_IRQ, 27 COM_RX_IRQ, 27 COM_RX_IRQ, 27 COM_TX_IRQ, 28 COM_TX_IRQ, 29		·
COM_IRQ_SETI, 28 COM_IRQ_SETI, 28 COM_IRQ_TIMER_IRQ, 28 COM_IRQ_TX_IRQ, 28 COM_IRQ_TX_IRQ, 28 COM_IRQ_SETI, 27 COM_IRQ_SETI, 28 COM_IRQ_SETI, 29 COM		
COM_IRQ_TIMER_IRQ, 28 COM_IRQ_TX_IRQ, 28 COM_IRQ_TX_IRQ, 28 COM_IRQ_TX_IRQ, 28 COM_IRQ_TX_IRQ, 27 COM_EX_IRQ, 27 COM_EX_IRQ, 27 COM_TX_IRQ, 27 COMMAND, 28 COMMUNICATION_ERROR, 27 CONTROL_T_START_NOW, 28 CONTROL_T_STOP_NOW, 28 CONTROL_T_STOP_NOW, 28 CONTROL_T_STOP_NOW, 28 CONTROL_T_STOP_NOW, 28 CONTROL_T_STOP_NOW, 29 CRC_ERROR, 27 CRC_RESULT_HIDH, 29 CRC_RESULT_LOW, 29 CRC_RESULT_LOW, 29 CRC_RESULT_LOW, 29 CRG_RESULT_LOW, 29 CRG_RESULT_RES		•
COM_IRQ_TIMER_IRQ, 28 COM_IRQ_X2RQ, 28 COM_IRQ_X2RQ, 29 COM_LO_ALERT_IRQ, 27 COM_RX_IRQ, 27 COM_RX_IRQ, 27 COM_TIMER_IRQ, 27 COM_TIMER_IRQ, 27 COM_TX_IRQ, 27 COM_TX_IRQ, 27 COM_TX_IRQ, 27 COM_TX_IRQ, 27 COM_TX_IRQ, 27 COMMAND, 28 COMMUNICATION_ERROR, 27 CONTROL_T_START_NOW, 28 CONTROL_T_STOP_NOW, 28 CONTROL_T_STOP_NOW, 28 CONTROL_29 CRC_ERROR, 27 CRC_RESULT_HIDH, 29 CRC_RESULT_LOW, 29 CRC_RESULT_LOW, 29 CRC_RESULT_TIMER, 28 COMMUNICATION_SOME RECEIVE, 27 CRC_RESULT_RIDER, 28 DIV_IEN_IRD_RC_1EN, 28 DIV_IEN_IRTERUPT_EN, 28 DIV_IEN_MFIN_ACT_IEN, 28 DIV_IRQ_ALL_IRQ, 28 DIV_IRQ_ALL_IRQ, 28 DIV_IRQ_CRC_IRQ, 28 DIV_IRQ_CRC_IRQ, 28 DIV_IRQ_CRC_IRQ, 28 DIV_IRQ_CRC_IRQ, 28 DIV_IRQ_SET, 28 DIV_IRQ_CRC_IRQ, 28 DIV_IRQ_SET, 28 DIV_IRQ_ALL_IRQ, 28 DIV_IRQ_BET, 29 T_PRESCALER_LOW, 2		
COM_IRQ, 28 COM_IRQ, 28 COM_IRQ, 28 COM_LO_ALERT_IRQ, 27 COM_RX_IRQ, 27 COM_RX_IRQ, 27 COM_TIMER_IRQ, 27 COM_TIMER_IRQ, 27 COM_TX_IRQ, 27 COM_MODE_R29 COM_MODE_R29 COM_MODE_R29 COM_MODE_R27 COM_TX_IRQ, 27 COM_TM_IRQ, 27 COM_MODE_RCQ, 27 COM_MODE_RCQ, 27 COM_TX_IRQ, 27 COM_MODE_RCQ, 27 COM_MODE_RCQ, 27 COM_MODE_RCQ, 29 COMMUNICATION_ERROR, 27 CONTROL_T_START_NOW, 28 CONTROL_T_STOP_NOW, 28 CONTROL_2 9 RX_MODE_RCQ, 29 CRC_ERROR, 27 CRC_RESULT_HIDH, 29 CRC_RESULT_LOW, 29 CRC_RESULT_LOW, 29 CRC_RESULT_LOW, 29 CRC_RESULT_LOW, 29 CRC_RESULT_LOW, 29 CRC_RESULT_RCQ, 29 COMMAND_RCQ, 30 clearLastError, 30 Command, 27 communicate, 30 ComfigureTimer, 30 DEMOD_R29 DIV_ALL_IRQ, 27 DIV_CRC_IRQ, 27 DIV_LEN_CRC_I_EN, 28 DIV_IEN_INTERRUPT_EN, 28 DIV_IEN_INTERRUPT_EN, 28 DIV_IEN_RCRC_IEQ, 28 DIV_IRQ_ALL_IRQ, 28 DIV_IRQ_ALL_IRQ, 28 DIV_IRQ_ALL_IRQ, 28 DIV_IRQ_ACT_IRQ, 28 DIV_IRQ_CRC_IRQ, 27 T_RELOAD_HIGH, 29 T_RELOAD_HIGH, 29 T_RELOAD_LOW, 29 T_REST_DAC2, 29 ERROR_27 TEST_DAC2, 29 ERROR_27 TEST_DIN_EN, 29 TEST_DAC2, 29 TEST_DIN_EN, 29		
COM_IRQ, 28 COM_LO_ALERT_IRQ, 27 COM_LO_ALERT_IRQ, 27 COM_TIMER_IRQ, 27 COM_TIMER_IRQ, 27 COM_TIMER_IRQ, 27 COM_TIMER_IRQ, 27 COM_TX_IRQ, 27 COM_TX_IRQ, 27 COM_TX_IRQ, 27 COM_MAND, 28 COMMUNICATION_ERROR, 27 CONTROL_T_START_NOW, 28 CONTROL_T_START_NOW, 28 CONTROL_T_STOP_NOW, 28 CONTROL_T_STOP_NOW, 28 CONTROL_T_STOP_NOW, 29 CRC_ERROR, 27 CRC_RESULT_HIDH, 29 CRC_RESULT_LOW, 29 CRC_RESULT_LOW, 29 CRC_RESULT_TOW, 28 CRC_RESULT_TOW, 28 CRC_RESULT_TOW, 29 CRC_RESULT_TOW,		<u> </u>
COM_LO_ALERT_IRQ, 27 COM_RX_IRQ, 27 COM_RX_IRQ, 27 COM_TX_IRQ, 27 COM_TX_IRQ, 27 COM_TX_IRQ, 27 COM_TX_IRQ, 27 COM_MAND, 28 COMMUNICATION_ERROR, 27 COMTROL_T_START_NOW, 28 CONTROL_T_START_NOW, 28 CONTROL_T_STOP_NOW, 28 CONTROL_T_STOP_NOW, 28 CONTROL_T_STOP_NOW, 28 CONTROL_T_STOP_NOW, 28 CONTROL_T_STOP_NOW, 29 CRC_ERSULT_HIDH, 29 CRC_RESULT_LOW, 28 DIV_LEN_LOW, 28 DIV_LEN_LOW, 28 DIV_LEN_LOW, 28 DIV_LEN_LOW, 28 DIV_IRQ_ALL_IRQ, 28 DIV_IRQ_ALL_IRQ, 28 DIV_IRQ_ALL_IRQ, 28 DIV_IRQ_SET2, 29 T_RELOAD_LOW, 29 T_SET_DAC1, 29 T_SET_DAC1, 29 TEST_DAC1, 29 TEST_DAC1, 29 TEST_DAC1, 29 TEST_DAC1, 29 TEST_PIN_EN, 29		:
COM_RX_IRQ, 27 COM_TIMER_IRQ, 27 COM_TIMER_IRQ, 27 COM_TX_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 CRC_RESULT_LOW, 29 CalculateCRC, 30 clearInterrupt, 30 clearLastError, 30 Command, 27 Communicate, 30 Configure Timer, 30 DIV_ALL_IRQ, 27 DIV_EN_CRC_IEQ, 27 DIV_EN_CRC_IEN, 28 DIV_LEN_MFIN_ACT_IEN, 28 DIV_IEN_MFIN_ACT_IEN, 28 DIV_IRQ_MFIN_ACT_IRQ, 28 DIV_IRQ_SET2, 28 DIV_IRD_ACT_IRQ, 27 DIV_RESCALER_LOW, 29 T_RELOAD_LOW, 29 T_RELOAD_LOW, 29 T_RELOAD_LOW, 29 T_RELOAD_LOW, 29 T_REST_DAC1, 29 TEST_DAC2, 29 TEST_DAC2, 29 TEST_PIN_EN, 29		MOD_WIDTH, 29
COM_TIMER_IRQ, 27 COM_TX_IRQ, 27 COM_TX_IRQ, 27 COMMAND, 28 COMMUNICATION_ERROR, 27 CONTROL_T_START_NOW, 28 CONTROL_T_STOP_NOW, 28 CONTROL_T_STOP_NOW, 28 CONTROL_29 CRC_ERROR, 27 CRC_RESULT_HIDH, 29 CRC_RESULT_LOW, 29 CRC_ESULT_LOW, 29 CRC_extroup and a clear tast and a clear t		MODE, 29
COM_TX_IRQ, 27 COMMAND, 28 COMMUNICATION_ERROR, 27 COMTROL_T_START_NOW, 28 CONTROL_T_START_NOW, 28 CONTROL_T_START_NOW, 28 CONTROL_T_STOP_NOW, 28 CONTROL_STOP_NOW, 29 CRC_ERROR, 27 CRC_RESULT_HIDH, 29 CRC_RESULT_LOW, 29 CRC_RC_RESULT_LOW, 29 CRC_RC_RESULT_LOW, 29 CRC_RC_RESULT_LOW, 29 CRC_RC_RC_RC_RC_RC_RC_RC_RC_RC_RC_RC_RC_	COM_RX_IRQ, 27	Mask, 27
COMMAND, 28 COMMUNICATION_ERROR, 27 CONTROL_T_START_NOW, 28 CONTROL_T_STOP_NOW, 28 CONTROL_T_STOP_NOW, 28 CONTROL_T_STOP_NOW, 28 CONTROL_29 CRC_ERROR, 27 CRC_ERROR, 27 CRC_RESULT_HIDH, 29 CRC_RESULT_LOW, 29 CRC_RESULT_LOW, 29 CRC_RESULT_STOP_NOW, 29 CRC_RESULT_LOW, 29 CRC_ROM, 34 CRC_RESULT_LOW, 29 CRC_ROM, 34 CRC_ROM,	COM_TIMER_IRQ, 27	NO_CMD_CHANGE, 27
COMMUNICATION_ERROR, 27 CONTROL_T_START_NOW, 28 CONTROL_T_STOP_NOW, 28 CONTROL_T_STOP_NOW, 28 CONTROL_29 CRC_ERROR, 27 CRC_RESULT_HIDH, 29 CRC_RESULT_LOW, 29 CRC_RESULT_LOW, 29 CRC_RESULT_LOW, 29 CRC_SP, 29 calculateCRC, 30 clearLastError, 30 ClearLastError, 30 Command, 27 communicate, 30 configureTimer, 30 DEMOD, 29 DIV_ALL_IRQ, 27 DIV_LEN_CRC_I_EN, 28 DIV_I_EN_CRC_I_EN, 28 DIV_I_EN_MFIN_ACT_I_EN, 28 DIV_IRQ_ALL_IRQ, 28 DIV_IRQ_CRC_IRQ, 28 DIV_IRQ_CRC_IRQ, 28 DIV_IRQ_SET2, 29	COM_TX_IRQ, 27	NO_ERROR, 27
CONTROL_T_START_NOW, 28 CONTROL_T_STOP_NOW, 28 CONTROL_Z9 CRC_ERROR, 27 CRC_ERROR, 27 CRC_ERSULT_HIDH, 29 CRC_RESULT_LOW, 29 CRC_RESULT_LOW, 29 CRC_RESULT_LOW, 29 CRC_RESULT_LOW, 29 CRC_RESULT_T_LOW, 29 CRC_RESULT_LOW, 28 CRC_RESULT_RESHOLD, 28 CRC_RESULT_LOW, 28 CRC_RESULT_RESHOLD, 28 CRC_RESULT_LOW, 29 CRC_RC_RESULT_LOW, 29 CRC_RC_RESULT_LOW, 29 CRC_RC_RESULT_LOW, 29 CRC_RC_RESULT_LOW, 29 CRC_RC_RESULT_LOW, 29 CRC_RC_RC_RESULT_LOW, 29 CRC_RC_RC_ROM, 28 CRC_RC_ROM, 28 CRC_RC_ROM, 28 CRC_RC_ROM, 28 CRC_RC_ROM, 28 CRC_RC_ROM, 28 CRC_RC_ROM, 28 CRC_ROM, 28 CRC_RC_ROM, 28 CRC_RC_ROM, 28 CRC_RC_ROM, 28 CRC_ROM, 28 CRC_RC_ROM, 28 CRC_RC_ROM, 28 CRC_RC_ROM, 28 CRC_ROM, 28 CRC_ROM, 28 CRC_RC_ROM, 28 CRC_ROM, 28 CRC_ROM, 28 CRC_ROM, 28 CRC_ROM, 29 CRC_RC_ROM, 29 CRC_RC_ROM, 29 CRC_RC_ROM, 20 CRC_ROM, 20 CRC_	COMMAND, 28	NONE_IRQ, 27
CONTROL_T_STOP_NOW, 28 CONTROL, 29 CRC_ERROR, 27 CRC_ERROR, 27 CRC_RESULT_HIDH, 29 CRC_RESULT_LOW, 28 CRC_RESULT_LOW, 28 CRC_RESULT_LOW, 29 CRC_RESULT_LOW, 28 CRC_RESULT_LOW, 28 CRC_RESULT_LOW, 29 CRC_RC_RESULT_LOW, 29 CRC_RESULT_LOW, 29 CRC_RC_RC_RC_RC_RC_RC_RC_RC_RC_RC_RC_RC_	COMMUNICATION_ERROR, 27	performSelfTest, 31
CONTROL, 29 CRC_ERROR, 27 CRC_ERROR, 27 CRC_RESULT_HIDH, 29 CRC_RESULT_LOW, 20 CRC_RESULT_RESULT_LOW, 20 CRC_RESULT_RESULT_RESULT_LOW, 20 CRC_RESULT_RESUL	CONTROL_T_START_NOW, 28	RECEIVE, 27
CONTROL, 29 CRC_ERROR, 27 CRC_ERROR, 27 CRC_RESULT_HIDH, 29 CRC_RESULT_LOW, 20 CRC_RESULT_RESULT_LOW, 20 CRC_RESULT_RESULT_RESULT_LOW, 20 CRC_RESULT_RESUL	CONTROL T STOP NOW, 28	RFC FG, 29
CRC_ERROR, 27 CRC_RESULT_HIDH, 29 CRC_RESULT_LOW, 20 CRC_RESULT_LOW, 20 CRC_RESULT_LOW, 27 CRC_RESULT_LOW, 28 CRC_RESULT_LOW, 29 CRC_RESULT_LOW, 2		
CRC_RESULT_HIDH, 29 CRC_RESULT_LOW, 27 CRC_RESULT_LOW, 27 CRC_RESULT_LOW, 28 CRC_RESULT_LOW, 29 CRC_RESULT_L		_
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_IEN_CRC_I_EN, 28 DIV_IEN_MFIN_ACT_IEN, 28 DIV_IEQ_ALL_IRQ, 28 DIV_IRQ_ALL_IRQ, 28 DIV_IRQ_SET2, 28 DIV_IRQ_SET2, 28 DIV_IRQ, 28 DIV_IRQ_SET2, 28 DIV_IRQ, 28 DIV_IRQ, 28 DIV_IRQ, 28 DIV_IRQ, 28 DIV_IRQ, 28 DIV_IRQ, 28 DIV_MCT_IRQ, 27 DIV_RCQ, 28 DIV_IRQ, 28 DIV_MRIN_ACT_IRQ, 27 DIV_MEN_ACT_IRQ, 27 DIV_RCQ, 28 DIV_IRQ, 28 DIV_IRQ, 28 DIV_MFIN_ACT_IRQ, 27 device, 34 disableInterrupt, 31 ERROR, 28 enableInterrupt, 31 Error, 27 RadioFrequencyIdentificationMFRC522, 30 readRegisterBlock, 32 restPin, 34 restPin, 34		
CW_GS_P, 29 readRegisterBlock, 32 calculateCRC, 30 Register, 28 clearInterrupt, 30 resetPin, 34 clearLastError, 30 SERIAL_SPEED, 29 Command, 27 SOFT_RESET, 27 communicate, 30 STATUS1, 28 configureTimer, 30 STATUS2, 28 DEMOD, 29 sendCommand, 32 DIV_ALL_IRQ, 27 setAntennaOff, 32 DIV_CRC_IRQ, 27 setAntennaOn, 33 DIV_I_EN_CRC_I_EN, 28 setWaterLevel, 33 DIV_I_EN_INTERRUPT_EN, 28 softReset, 33 DIV_I_EN_MFIN_ACT_I_EN, 28 startTimer, 33 DIV_I_EN, 28 stopTimer, 33 DIV_I_EQ, 28 T_COUNTER_VAL_HIGH, 29 DIV_IRQ_ALL_IRQ, 28 T_COUNTER_VAL_LOW, 29 DIV_IRQ_MFIN_ACT_IRQ, 28 T_MODE, 29 DIV_IRQ_SET2, 28 T_PRESCALER_LOW, 29 DIV_IRQ, 28 T_RELOAD_HIGH, 29 DIV_MFIN_ACT_IRQ, 27 T_RELOAD_LOW, 29 device, 34 TEST_ADC, 29 disableInterrupt, 31 TEST_BUS, 29 enableInterrupt, 31 TEST_DAC1, 29 enableInterrupt, 31		-
calculateCRC, 30 Register, 28 clearInterrupt, 30 resetPin, 34 clearLastError, 30 SERIAL_SPEED, 29 Command, 27 SOFT_RESET, 27 communicate, 30 STATUS1, 28 configureTimer, 30 STATUS2, 28 DEMOD, 29 sendCommand, 32 DIV_ALL_IRQ, 27 setAntennaOff, 32 DIV_CRC_IRQ, 27 setAntennaOn, 33 DIV_IEN_CRC_IEN, 28 setWaterLevel, 33 DIV_IEN_INTERRUPT_EN, 28 softReset, 33 DIV_IEN_MFIN_ACT_IEN, 28 startTimer, 33 DIV_IEN, 28 T_COUNTER_VAL_HIGH, 29 DIV_IRQ_ALL_IRQ, 28 T_COUNTER_VAL_LOW, 29 DIV_IRQ_GRC_IRQ, 28 T_COUNTER_VAL_LOW, 29 DIV_IRQ_SET2, 28 T_MODE, 29 DIV_IRQ_SET2, 28 T_PRESCALER_LOW, 29 DIV_IRQ, 28 T_RELOAD_HIGH, 29 DIV_IRQ, 28 T_RELOAD_LOW, 29 DIV_IRQ, 28 T_RELOAD_LOW, 29 DIV_MFIN_ACT_IRQ, 27 T_RELOAD_LOW, 29 device, 34 TEST_BUS, 29 disableInterrupt, 31 TEST_DAC1, 29 enableInterrupt, 31 TEST_DAC2, 29 Error, 27 TEST_DAC2, 29 <td></td> <td></td>		
clearInterrupt, 30 resetPin, 34 clearLastError, 30 SERIAL_SPEED, 29 Command, 27 SOFT_RESET, 27 communicate, 30 STATUS1, 28 configureTimer, 30 STATUS2, 28 DEMOD, 29 sendCommand, 32 DIV_ALL_IRQ, 27 setAntennaOff, 32 DIV_CRC_IRQ, 27 setAntennaOn, 33 DIV_I_EN_CRC_I_EN, 28 setWaterLevel, 33 DIV_I_EN_MFIN_ACT_I_EN, 28 startTimer, 33 DIV_I_EN_MFIN_ACT_I_EN, 28 stopTimer, 33 DIV_IRQ_ALL_IRQ, 28 T_COUNTER_VAL_HIGH, 29 DIV_IRQ_ALL_IRQ, 28 T_COUNTER_VAL_LOW, 29 DIV_IRQ_MFIN_ACT_IRQ, 28 T_MODE, 29 DIV_IRQ_SET2, 28 T_PRESCALER_LOW, 29 DIV_IRQ_SE T_RELOAD_HIGH, 29 DIV_MFIN_ACT_IRQ, 27 T_RELOAD_LOW, 29 device, 34 TEST_ADC, 29 disableInterrupt, 31 TEST_BUS, 29 ERROR, 28 TEST_DAC1, 29 enableInterrupt, 31 TEST_DAC2, 29 Error, 27 TEST_PIN_EN, 29		
clearLastError, 30 SERIAL_SPEED, 29 Command, 27 SOFT_RESET, 27 communicate, 30 STATUS1, 28 configureTimer, 30 STATUS2, 28 DEMOD, 29 sendCommand, 32 DIV_ALL_IRQ, 27 setAntennaOff, 32 DIV_CRC_IRQ, 27 setAntennaOn, 33 DIV_IEN_CRC_I_EN, 28 setWaterLevel, 33 DIV_IEN_INTERRUPT_EN, 28 softReset, 33 DIV_IEN_MFIN_ACT_I_EN, 28 startTimer, 33 DIV_IEN_AE stopTimer, 33 DIV_IRQ_ALL_IRQ, 28 T_COUNTER_VAL_HIGH, 29 DIV_IRQ_CRC_IRQ, 28 T_COUNTER_VAL_LOW, 29 DIV_IRQ_MFIN_ACT_IRQ, 28 T_MODE, 29 DIV_IRQ_SET2, 28 T_RELOAD_HIGH, 29 DIV_IRQ, 28 T_RELOAD_LOW, 29 DIV_MFIN_ACT_IRQ, 27 T_RELOAD_LOW, 29 device, 34 TEST_BUS, 29 disableInterrupt, 31 TEST_BUS, 29 enableInterrupt, 31 TEST_DAC2, 29 Error, 27 TEST_PIN_EN, 29		_
Command, 27 SOFT_RESET, 27 communicate, 30 STATUS1, 28 configure Timer, 30 STATUS2, 28 DEMOD, 29 sendCommand, 32 DIV_ALL_IRQ, 27 setAntennaOff, 32 DIV_CRC_IRQ, 27 setAntennaOn, 33 DIV_I_EN_CRC_I_EN, 28 setWaterLevel, 33 DIV_I_EN_INTERRUPT_EN, 28 softReset, 33 DIV_I_EN_MFIN_ACT_I_EN, 28 startTimer, 33 DIV_I_EN_82 stopTimer, 33 DIV_IRQ_ALL_IRQ, 28 T_COUNTER_VAL_HIGH, 29 DIV_IRQ_CRC_IRQ, 28 T_COUNTER_VAL_LOW, 29 DIV_IRQ_MFIN_ACT_IRQ, 28 T_MODE, 29 DIV_IRQ_SET2, 28 T_PRESCALER_LOW, 29 DIV_IRQ, 28 T_RELOAD_HIGH, 29 DIV_IRQ, 28 T_RELOAD_LOW, 29 DIV_MFIN_ACT_IRQ, 27 T_RELOAD_LOW, 29 device, 34 TEST_ADC, 29 disableInterrupt, 31 TEST_BUS, 29 ERROR, 28 TEST_DAC1, 29 enableInterrupt, 31 TEST_DAC2, 29 Error, 27 TEST_PIN_EN, 29	•	•
communicate, 30 STATUS1, 28 configureTimer, 30 STATUS2, 28 DEMOD, 29 sendCommand, 32 DIV_ALL_IRQ, 27 setAntennaOff, 32 DIV_CRC_IRQ, 27 setAntennaOn, 33 DIV_I_EN_CRC_I_EN, 28 setWaterLevel, 33 DIV_I_EN_INTERRUPT_EN, 28 softReset, 33 DIV_I_EN_MFIN_ACT_I_EN, 28 startTimer, 33 DIV_I_EN, 28 T_COUNTER_VAL_HIGH, 29 DIV_IRQ_ALL_IRQ, 28 T_COUNTER_VAL_LOW, 29 DIV_IRQ_MFIN_ACT_IRQ, 28 T_MODE, 29 DIV_IRQ_SET2, 28 T_PRESCALER_LOW, 29 DIV_IRQ, 28 T_RELOAD_HIGH, 29 DIV_IRQ, 28 T_RELOAD_LOW, 29 DIV_MFIN_ACT_IRQ, 27 T_RELOAD_LOW, 29 device, 34 TEST_ADC, 29 disableInterrupt, 31 TEST_BUS, 29 ERROR, 28 TEST_DAC1, 29 enableInterrupt, 31 TEST_DAC2, 29 Error, 27 TEST_PIN_EN, 29		- :
configureTimer, 30 DEMOD, 29 SendCommand, 32 DIV_ALL_IRQ, 27 SetAntennaOff, 32 SetAntennaOn, 33 SetWaterLevel, 33 DIV_I_EN_CRC_I_EN, 28 SetWaterLevel, 33 SetWaterLevel, 33 DIV_I_EN_INTERRUPT_EN, 28 SetWaterLevel, 33 SetWaterLevel, 32 SetAttennaOff, 32 SetAntennaOff, 32 SetWaterLevel, 33 SetWaterLevel, 33 SetWaterLevel, 33 SetWaterLevel, 33 SetWaterLevel, 36 SetWaterLevel, 33 SetWaterLevel, 36 SetWaterLevel, 36 SetWaterLevel, 36 SetWaterLevel, 36 SetWaterLevel, 36 SetAntennaOff, 32 SetWaterLevel, 33 SetWaterLevel, 33 SetWaterLevel, 33 SetWaterLevel, 32 SetWaterLevel, 33 SetWaterLevel, 32 SetWaterLevel, 33 SetWaterLevel, 32 SetWaterLevel, 33 SetWaterLevel, 32 SetWaterLeve		
DEMOD, 29 DIV_ALL_IRQ, 27 DIV_CRC_IRQ, 27 DIV_CRC_IEN, 28 DIV_I_EN_CRC_I_EN, 28 DIV_I_EN_INTERRUPT_EN, 28 DIV_I_EN_MFIN_ACT_I_EN, 28 DIV_I_EN_MFIN_ACT_I_EN, 28 DIV_I_EN_QALL_IRQ, 28 DIV_IRQ_ALL_IRQ, 28 DIV_IRQ_MFIN_ACT_IRQ, 28 DIV_IRQ_MFIN_ACT_IRQ, 28 DIV_IRQ_MFIN_ACT_IRQ, 28 DIV_IRQ_BET2, 28 DIV_IRQ_SET2, 28 DIV_IRQ, 28 DIV_MRD, 29 DIV_MFIN_ACT_IRQ, 27 device, 34 disableInterrupt, 31 ERROR, 28 enableInterrupt, 31 ERROC, 29 Error, 27 TEST_DAC2, 29 Error, 27 TEST_PIN_EN, 29	•	
DIV_ALL_IRQ, 27 DIV_CRC_IRQ, 27 DIV_IEN_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_SET2, 28 DIV_IRQ, 28 DIV_MRIN_ACT_IRQ, 27 device, 34 disableInterrupt, 31 ERROR, 28 enableInterrupt, 31 Error, 27 setAntennaOff, 32 setAntennaOff, 36 setMaterLevel, 33 setWaterLevel, 33 setWaterLevel, 33 setWaterLevel, 33 setWaterLevel, 36 setWaterLevel, 33 setWaterLevel, 36 storTime, 33 stertine, 33 stertine, 33 stertine,	•	
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_IRQ, 28 DIV_IRQ, 28 DIV_IRQ, 28 DIV_IRQ, 29 DIV_IRQ, 29 DIV_IRQ, 29 DIV_IRQ, 29 DIV_IRQ, 28 DIV_MFIN_ACT_IRQ, 27 device, 34 T_EST_ADC, 29 disableInterrupt, 31 TEST_BUS, 29 ERROR, 28 enableInterrupt, 31 TEST_DAC2, 29 Error, 27 TEST_PIN_EN, 29		
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_IRQ, 28 DIV_IRQ, 28 DIV_IRQ, 28 DIV_IRQ, 29 DIV_IRQ, 29 DIV_IRQ, 29 DIV_IRQ, 29 DIV_IRQ, 29 DIV_MFIN_ACT_IRQ, 27 device, 34 disableInterrupt, 31 ERROR, 28 enableInterrupt, 31 Error, 27 SetWaterLevel, 33 softReset, 32 startTimer, 33 T_COUNTER_VAL_HIGH, 29 T_MODE, 29 T_MODE, 29 T_PRESCALER_LOW, 29 T_PRESCALER_LOW, 29 T_RELOAD_LIOW, 29 TEST_ADC, 29 TEST_BUS, 29 TEST_DAC1, 29 Error, 27 TEST_DAC2, 29 TEST_DAC2, 29 TEST_DAC2, 29 TEST_PIN_EN, 29		
DIV_I_EN_INTERRUPT_EN, 28 DIV_I_EN_MFIN_ACT_I_EN, 28 DIV_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_SET2, 28 DIV_IRQ, 28 DIV_IRQ, 28 DIV_MFIN_ACT_IRQ, 27 device, 34 disableInterrupt, 31 ERROR, 28 enableInterrupt, 31 Error, 27 SoftReset, 33 startTimer, 33 T_COUNTER_VAL_HIGH, 29 T_COUNTER_VAL_LOW, 29 T_MODE, 29 T_MODE, 29 T_PRESCALER_LOW, 29 T_RELOAD_HIGH, 29 T_RELOAD_LOW, 29 TEST_ADC, 29 TEST_ADC, 29 TEST_DAC1, 29 TEST_DAC1, 29 TEST_DAC2, 29 TEST_DAC2, 29 TEST_DAC2, 29 TEST_DAC2, 29 TEST_PIN_EN, 29		
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_SET2, 28 DIV_MRQ, 28 DIV_MFIN_ACT_IRQ, 27 device, 34 disableInterrupt, 31 ERROR, 28 enableInterrupt, 31 Error, 27 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_DAC1, 29 TEST_DAC2, 29 TEST_DAC2, 29 TEST_DAC2, 29 TEST_PIN_EN, 29		
DIV_I_EN, 28 stopTimer, 33 DIV_IRQ_ALL_IRQ, 28 T_COUNTER_VAL_HIGH, 29 DIV_IRQ_CRC_IRQ, 28 T_COUNTER_VAL_LOW, 29 DIV_IRQ_MFIN_ACT_IRQ, 28 T_MODE, 29 DIV_IRQ_SET2, 28 T_PRESCALER_LOW, 29 DIV_IRQ, 28 T_RELOAD_HIGH, 29 DIV_MFIN_ACT_IRQ, 27 T_RELOAD_LOW, 29 device, 34 TEST_ADC, 29 disableInterrupt, 31 TEST_BUS, 29 ERROR, 28 TEST_DAC1, 29 enableInterrupt, 31 TEST_DAC2, 29 Error, 27 TEST_PIN_EN, 29		•
DIV_IRQ_ALL_IRQ, 28 DIV_IRQ_CRC_IRQ, 28 DIV_IRQ_MFIN_ACT_IRQ, 28 DIV_IRQ_SET2, 28 DIV_IRQ, 28 DIV_IRQ, 28 DIV_IRQ, 28 DIV_IRQ, 28 DIV_MFIN_ACT_IRQ, 27 device, 34 disableInterrupt, 31 ERROR, 28 enableInterrupt, 31 Error, 27 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_DAC2, 29 TEST_DAC2, 29 TEST_DAC2, 29 TEST_PIN_EN, 29		
DIV_IRQ_CRC_IRQ, 28 T_COUNTER_VAL_LOW, 29 DIV_IRQ_MFIN_ACT_IRQ, 28 T_MODE, 29 DIV_IRQ_SET2, 28 T_PRESCALER_LOW, 29 DIV_IRQ, 28 T_RELOAD_HIGH, 29 DIV_MFIN_ACT_IRQ, 27 T_RELOAD_LOW, 29 device, 34 TEST_ADC, 29 disableInterrupt, 31 TEST_BUS, 29 ERROR, 28 TEST_DAC1, 29 enableInterrupt, 31 TEST_DAC2, 29 Error, 27 TEST_PIN_EN, 29		•
DIV_IRQ_MFIN_ACT_IRQ, 28 T_MODE, 29 DIV_IRQ_SET2, 28 T_PRESCALER_LOW, 29 DIV_IRQ, 28 T_RELOAD_HIGH, 29 DIV_MFIN_ACT_IRQ, 27 T_RELOAD_LOW, 29 device, 34 TEST_ADC, 29 disableInterrupt, 31 TEST_BUS, 29 ERROR, 28 TEST_DAC1, 29 enableInterrupt, 31 TEST_DAC2, 29 Error, 27 TEST_PIN_EN, 29		:
DIV_IRQ_SET2, 28 T_PRESCALER_LOW, 29 DIV_IRQ, 28 T_RELOAD_HIGH, 29 DIV_MFIN_ACT_IRQ, 27 T_RELOAD_LOW, 29 device, 34 TEST_ADC, 29 disableInterrupt, 31 TEST_BUS, 29 ERROR, 28 TEST_DAC1, 29 enableInterrupt, 31 TEST_DAC2, 29 Error, 27 TEST_PIN_EN, 29		T_COUNTER_VAL_LOW, 29
DIV_IRQ, 28 T_RELOAD_HIGH, 29 DIV_MFIN_ACT_IRQ, 27 T_RELOAD_LOW, 29 device, 34 TEST_ADC, 29 disableInterrupt, 31 TEST_BUS, 29 ERROR, 28 TEST_DAC1, 29 enableInterrupt, 31 TEST_DAC2, 29 Error, 27 TEST_PIN_EN, 29	DIV_IRQ_MFIN_ACT_IRQ, 28	T_MODE, 29
DIV_MFIN_ACT_IRQ, 27 device, 34 disableInterrupt, 31 ERROR, 28 enableInterrupt, 31 Error, 27 T_RELOAD_LOW, 29 TEST_ADC, 29 TEST_BUS, 29 TEST_DAC1, 29 TEST_DAC2, 29 TEST_DAC2, 29 TEST_PIN_EN, 29	DIV_IRQ_SET2, 28	T_PRESCALER_LOW, 29
device, 34 disableInterrupt, 31 ERROR, 28 enableInterrupt, 31 Error, 27 TEST_ADC, 29 TEST_BUS, 29 TEST_DAC1, 29 TEST_DAC2, 29 TEST_DAC2, 29 TEST_PIN_EN, 29	DIV_IRQ, 28	T_RELOAD_HIGH, 29
disableInterrupt, 31 ERROR, 28 enableInterrupt, 31 Error, 27 TEST_BUS, 29 TEST_DAC1, 29 TEST_DAC2, 29 TEST_DAC2, 29 TEST_PIN_EN, 29	DIV_MFIN_ACT_IRQ, 27	T_RELOAD_LOW, 29
disableInterrupt, 31 ERROR, 28 enableInterrupt, 31 Error, 27 TEST_BUS, 29 TEST_DAC1, 29 TEST_DAC2, 29 TEST_DAC2, 29 TEST_PIN_EN, 29	device, 34	TEST_ADC, 29
ERROR, 28 TEST_DAC1, 29 enableInterrupt, 31 TEST_DAC2, 29 Error, 27 TEST_PIN_EN, 29	disableInterrupt, 31	
enableInterrupt, 31 TEST_DAC2, 29 Error, 27 TEST_PIN_EN, 29	• •	- · · ·
Error, 27 TEST_PIN_EN, 29		_
	•	
_ , , ,		— — ·
	_ ,	, -

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_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 ← Noits, 7 ERR_I_EN, 8 H_ALERT_I_EN, 8 I_RQ_INV, 8 IDLE_I_EN, 8 RX_I_EN, 8 TIMER_I_EN, 8 TX_I_EN, 8 TX_I_EN, 8 Value, 8 CW_GS_P, 12 char, 12 value, 13 RadioFrequencyIdentificationMFRC522::DEMODbits, 13 ADD_IQ, 13 FPRESCAL_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, value, 15 RadioFrequencyIdentificationMFRC522::DIV_IRQbits, value, 16 RadioFrequencyIdentificationMFRC522::ERRORbits, 16 BUFFER_OVFL, 17 COLL_ERR, 17 CRC_ERR, 17 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 CRC_I_EN, 15 value, 15 RadioFrequencyIdentificationMFRC522::DIV_IRQbits, value, 15 RadioFrequencyIdentificationMFRC522::DIV_IRQbits, value, 16 RAD_IQ, 13 FIX_I_EN, 8 FIX_I_EN, 8 Value, 18 RadioFrequencyIdentificationMFRC522::DIV_IRQbits, value, 15 RadioFrequencyIdentificationMFRC522::DIV_IRQbits, value, 16 RadioFrequencyIdentificationMFRC522::DIV_IRQbits, value, 15 RadioFrequencyIdentificationMFR		
THEST_SEL2_29 TIMEOUT_ERROR, 27 TRANSCEIVE, 27 TRANSMIT_27 TX_ASK_29 TX_CONTROL_TX_RF_EN_28 TX_LON_TROL_TX_RF_EN_28 TX_LON_TR	TEST SEL1, 29	LO ALERT IRQ, 9
TRANSCEIVE, 27 TTAASMT, 27 TTAASMT, 27 TTAASMT, 29 TX_CONTROL_TX2_RF_EN, 28 TX_CONTROL_TX3_RF_EN, 29 TX_LEN, 8 TX_CONTROL_TX3_RF_EN, 28 TX_CONTROL_TX3_RF_EN, 29 TX_LEN, 8 TX_CONTROL_TX3_RF_EN, 29		
TRANSCEIVE, 27 TTAASMT, 27 TTAASMT, 27 TTAASMT, 29 TX_CONTROL_TX2_RF_EN, 28 TX_CONTROL_TX3_RF_EN, 29 TX_LEN, 8 TX_CONTROL_TX3_RF_EN, 28 TX_CONTROL_TX3_RF_EN, 29 TX_LEN, 8 TX_CONTROL_TX3_RF_EN, 29		SET1, 9
TRANSMT, 27 TX ASK, 29 TX CONTROL_TX1_RF_EN, 28 TX CONTROL_TX RF_EN, 28 TX COMTROL_TX RF_EN, 28 TX CONTROL_TX RF_EN, 28 TX COMTROL_TX RF_EN, 28 TX CONTROL_TX RF_EN, 28 TX COMTROL_TX RF_EN, 28 TX CONTROL_TX RF_EN, 28 TX COMTROL_TX RF_EN, 28 TX COMTROL_TX RF_EN, 28 TX COMTROL_TX RF_EN, 28 TX COMTROL_TX RF_EN, 29 TX COMTROL_TX RF_EN, 29 TX COMTROL_TX RF_EN, 29 TX COMTROL_TX RF_EN, 29 TX T COMTROL_TX RF_EN, 20 TX COMTROL_TX RF_EN, 20 TX COMTROL_TX RF_EN, 20 TX COMTROL_TX RF_EN,		
TX_CONTROL_TX1_RF_EN, 28 TX_CONTROL_TX2_RF_EN, 28 TX_CONTROL_TX2_RF_EN, 28 TX_CONTROL_TX2_RF_EN, 28 TX_CONTROL_TX3_RF_EN, 28 TX_CONTROL_TX3_RF_EN, 28 TX_CONTROL_TX3_RF_EN, 28 TX_CONTROL_TX3_RF_EN, 28 TX_CONTROL_TX3_RF_EN, 28 TX_CONTROL_TX4_RF_EN, 28 TX_CONTROL_TX3_RF_EN, 28 TX_CONTROL_TX4_RF_EN, 28 TX_CONTROL_TX4_RF_EN, 28 TX_CONTROL_TX4_RF_EN, 28 TX_CONTROL_TX4_RF_EN, 28 TX_CONTROL_TX4_RF_EN, 28 WATER_LEVEL_WATER_LEVEL, 28 WATER_LEVEL_WATER_LEVEL, 28 WATER_LEVEL_WATER_LEVEL, 28 WATER_LEVEL_WATER_LEVEL, 28 WATER_LEVEL_WATER_LEVEL, 28 WATER_LEVEL_DEN, 29 WAIFOR Register Bits, 39 walfor Feequency IdentificationMFRC S22.x0 p., 50, 51 RadioFrequency IdentificationMFRC S22.x0 p., 50, 51 RadioFrequency IdentificationMFRC S22.x0 p., 50, 51 MFRS22_INT_TO_IRQ_MASK, 56 MFRS22_INT_TO_IRQ_MAS		
TX_CONTROL_TX1_RF_EN_28 TX_CONTROL_TX2_RF_EN_28 TX_CONTROL_TX2_RF_EN_28 TX_CONTROL_TX_RF_EN_28 TX_SEL_Z9 TX_SEL_Z9 VV_0_0_30 V2_0_30 V2_0_30 V2_0_30 V2_0_30 V2_D_SO_TROL_TT CONTROL_TX_RF_EN_28 WATER_LEVEL_Z9 WAIFOREPSIEVERIS, 33 WATER_LEVEL_WATER_LEVEL_Z8 WAIFOREPSIEVERIS, 33 WATER_LEVEL_TX_TO_EN_CS_22 WAIFOREPSIEVERIS, 33 WAIFOR_TX_LEN_TY_CONTROL_TX_RF_EN_28 WARER_LEVEL_Z9 WAIFOR_TX_RF_EN_Z9 WAITER_LEVEL_Z9 WAIFOR_TX_RF_EN_Z9 WAITER_LEVEL_WATER_LEVEL_Z8 WAIFOR_TX_RF_EN_Z9 WAITER_LEVEL_Z9 WAIFOR_TX_RF_EN_Z9 WAITER_LEVEL_WATER_LEVEL_Z8 WAIFOR_TX_RF_EN_Z9 WAITER_LEVEL_WATER_LEVEL_Z8 WAITER_LEVEL_Z9	•	——————————————————————————————————————
TX_CONTROL_TX_RF_EN, 28 TX_LEN, 8 T		
TX_CONTROL_TX_RF_EN, 28 TX_CONTROL_72 PTX_MODE, 29 TX_MODE, 29 TX_MODE, 29 TX_SEL, 29 VX_O, 30 V1_0, 30 V1_0, 30 V2_0,		
TX_CONTROL_29 TX_SEL_29 TX_SEL_29 VX_0_30 VX_0_0_30 VX_		
TX_MODE, 29 TX_SEL, 29 TX_SEL, 29 V0_0, 30 V1_0, 30 V1_0, 30 V2_0, 30 V3_0, 30 V3_0		
TX_SEL, 29 V0_0, 30 V1_0, 30 V2_0, 30 V2_0, 30 V2_0, 30 V2_Sion, 29 WATER_LEVEL_VATER_LEVEL, 28 WATER_LEVEL_29 waitForRegisterBlock, 33 RadioFrequencyIdentificationMFRC522.pd, 50, 51 RadioFrequencyIdentificationMFRC522.pd, 50, 51 MFRS22_INT_TO_EN_REG, 56 MFR522_INT_TO_IRQ_MASK, 56 MFR522_INT_TO_IRQ_MEG, 57 PROGMEM, 57 RAdioFrequencyIdentificationMFRC522::BIT_FRAMINGBITS, 5 char, 6 RX_ALIGN, 6 START_SEND, 6 TX_LAST_BITS, 6 value, 6 RX_ALIGN, 6 START_SEND, 6 TX_LAST_BITS, 6 value, 6 RAMINGROUPH AND	_ '	
V0_0, 30 V1_0, 30 V1_0, 30 V2_0, 30 VERSION, 29 WATER_LEVEL_WATER_LEVEL, 28 WATER_LEVEL_S0 waitForRegisterBits, 33 writeRegisterBits, 39 pits, 11 RadioFrequencyIdentificationMFRC522::CW_GS Pbits, 12 RedioFrequencyIdentificationMFRC522::CW_GS Pbits, 12 RadioFrequencyIdentificationMFRC522::CW_GS Pobits, 12 RadioFrequencyIdentificationMFRC522::CW_GS 13 RadioFrequencyIdentificationMFRC522::DIV_IENbits, 6 RadioFrequencyIdentificationMFRC522::DIV_IENbits, 6 RadioFrequencyIdentific		
V2_0, 30 V2_0, 30 V2_0, 30 V2_0, 30 V2_0, 30 VERSION, 29 Version, 29 Warter_LEVEL_VATER_LEVEL, 28 WATER_LEVEL_VB 29 waitforRegisterBits, 33 RadioFrequencyIdentificationMFRC522.pp, 50, 51 RadioFrequencyIdentificationMFRC522.pp, 50, 51 MFR522_INT_TO_EN_MASK, 56 MFR522_INT_TO_EN_MASK, 56 MFR522_INT_TO_IRQ_MASK, 56 MFR5		_ <i>'</i>
V2_0_30 VERSION, 29 WATER_LEVEL, 29 WATER_LEVEL, 29 WAIFORGejisterBits, 33 RadioFrequencyIdentificationMFRC522.cpp, 50, 51 RadioFrequencyIdentificationMFRC522.pp, 50, 51 RadioFrequencyIdentificationMFRC522.pp, 50, 57 MFR522_INT_TO_EN_MASK, 56 MFR522_INT_TO_IRQ_MASK, 56 M		
VERSION, 29 Version, 29 Version, 29 Warter, LEVEL_WATER_LEVEL, 28 WATER_LEVEL, 29 wait-orRegisterBits, 33 writeRegisterBits, 33 writeRegisterBits, 33 writeRegisterBits, 33 writeRegisterBits, 33 RadioFrequencyIdentificationMFRC522.cpp, 50, 51 RadioFrequencyIdentificationMFRC522.pp, 50, 51 RadioFrequencyIdentificationMFRC522.pp, 50, 51 RAFIS22_INT_TO_EN_MASK, 56 MFR522_INT_TO_IRQ_MASK, 56 MFR522_INT_TO_IRQ_MSK, 56 MFR522_INT_TO_IRQ_REG, 57 MFR6522_INT_TO_IRQ_REG, 57 MFR6522_DEFAULT_TIMEOUT, 57 PROGMEM, 57 RadioFrequencyIdentificationMFRC522::BIT_FRAMI← NGbits, 5 char, 6 START_SEND, 6 TX_LAST_BITS, 6 value, 6 RALIGN, 6 START_SEND, 6 TX_LAST_BITS, 6 value, 16 RadioFrequencyIdentificationMFRC522::COLLbits, 6 COLL_POS, 7 char, 7 VALUES_AFTER_COLL, 7 value, 7 RACIDER, 8 H_LALERT_IEN, 8 H_LALERT_IEN, 8 LQ_INV, 8 IDLE_IEN, 8 RX_IEN, 8 TX_IEN, 8 TX_IEN, 8 TX_IEN, 8 RAZI_EN, 8 RAGIOFrequencyIdentificationMFRC522::COM_IRObits, 9 ERR_IRQ, 9 H_LERT_IRQ, 9 H_LERT_IRQ, 9 H_LERT_IRQ, 9		
Warter, LeVel_ Water, LeVel, 28 Warter, LeVel, 29 waitForRegisterBids, 33 Warter, etvel, 29 waitForRegisterBids, 33 Walue, 12 RadioFrequencyIdentificationMFRC522::CW_GS_← Pibits, 12 CW_GS_P, 12 char, 12 value, 13 ADD_IQ, 13 FIX_IQ, 13 FIX_IQ, 13 TAU_EV, 14 value, 14 RadioFrequencyIdentificationMFRC522::DIV_IENbits, 14 Value, 14 RadioFrequencyIdentificationMFRC522::DIV_IENbits, 14 CRC_IEN, 14 RadioFrequencyIdentificationMFRC522::DIV_IENbits, 14 RadioFrequencyIdentificationMFRC522::DIV_IENbits, 14 RadioFrequencyIdentificationMFRC522::DIV_IENbits, 14 RadioFrequencyIdentificationMFRC522::DIV_IENbits, 14 RadioFrequencyIdentificationMFRC522::DIV_IENbits, 14 RadioFrequencyIdentificationMFRC522::DIV_IENbits, 15 RadioFrequencyIdentificationMFRC522::DIV_IENbits, 15 WFIN_ACT_IEN, 17 CAL_ERR, 17 CAL_ERR, 17 CAL_ERR, 17 CAL_ERR, 17 CAL_ERR, 17 PARITY_ERR, 17 PARITY_ERR, 17 Value, 17		
WATER_LEVEL_WATER_LEVEL, 28 WATER_LEVEL, 29 WaitForRegisterBits, 33 writeRegisterBlock, 33 writeRegisterBlock, 33 writeRegisterBlock, 33 RadioFrequencyIdentificationMFRC522.cpp, 50, 51 RadioFrequencyIdentificationMFRC522.h, 55, 57 MFR522_INT_TO_EN_MASK, 56 MFR522_INT_TO_EN_MASK, 56 MFR522_INT_TO_IRQ_MASK, 56 MFR522_INT_TO_IRQ_MASK, 56 MFR622_INT_TO_IRQ_MASK, 56 MFR622_INT_TO_IRQ_MSK, 5		
WATER_LEVEL, 29 waitForRegisterBlox, 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_MASK, 56 MFR522_INT_TO_IRQ_MASK, 56 MFR522_INT_TO_IRQ_MASK, 56 MFR522_INT_TO_IRQ_MEG, 57 MFR0522_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_NOT_VALID, 7 COLL_POS_NOT_VALID, 7 VALUES_AFTER_COLL, 7 value, 7 VALUES_AFTER_COLL, 7 value, 7 VALUES_AFTER_COLL, 7 value, 7 VALUES_AFTER_COLL, 7 value, 7 FERR_I_EN, 8 H_ALERT_LEN, 8 LO_ALERT_EN, 8 I_RQ_INV, 8 IDLE_I_EN, 8 TMER_I_EN, 8 TAJ_EN, 8 Value, 8 RadioFrequencyIdentificationMFRC522::COM_IRQbits, 9 H_ALERT_IRQ, 9 H_ALERT_IRQ, 9 T_STOP_NOW, 12 value, 12 RadioFrequencyIdentificationMFRC522::CW_GS_ value, 12 RadioFrequencyIdentificationMFRC522::CW_GS_ ABdioFrequencyIdentificationMFRC522::DEMODbits, 13 RadioFrequencyIdentificationMFRC522::DEMODbits, 13 RadioFrequencyIdentificationMFRC522::DEMODbits, 14 Value, 13 RadioFrequencyIdentificationMFRC522::DEMODbits, 13 RadioFrequencyIdentificationMFRC522::DEMODbits, 14 Value, 14 RadioFrequencyIdentificationMFRC522::DIV_I_ENbits, 14 RadioFrequencyIdentificationMFRC522::DIV_I_ENbits, 15		:
waitForRegisterBits, 33 writeRegisterBitock, 33 RadioFrequencyIdentificationMFRC522.cpp, 50, 51 RadioFrequencyIdentificationMFRC522.pp, 50, 51 RadioFrequencyIdentificationMFRC522.h, 55, 57 MFR522_INT_TO_EN_MASK, 56 MFR522_INT_TO_EN_MASK, 56 MFR522_INT_TO_IRQ_MASK, 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_IE Nbits, 7 ERR_I_EN, 8 I_RQ_INV, 19 I_RQ_INV, 8 I	WATER_LEVEL_WATER_LEVEL, 28	
RadioFrequencyIdentificationMFRC522.cpp, 50, 51 RadioFrequencyIdentificationMFRC522.ph, 55, 57 MFR522_INT_TO_EN_MASK, 56 MFR522_INT_TO_IRQ_MASK, 56 MFR6522_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_IE← Nbits, 7 ERR_I_EN, 8 I_RQ_INV, 10 I_RQ_INV	WATER_LEVEL, 29	
RadioFrequencyIdentificationMFRC522.cp, 50, 51 RadioFrequencyIdentificationMFRC522.h, 55, 57 MFR522_INT_TO_EN_MASK, 56 MFR522_INT_TO_EN_REG, 56 MFR522_INT_TO_ENQ_REG, 57 MFR6522_DEFAULT_TIMEOUT, 57 PROGMEM, 57 RadioFrequencyIdentificationMFRC522::BIT_FRAMI Nobits, 5 char, 6 RX_ALIGN, 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 I_RQ_INV, 8 IDLE_I_EN, 8 I_RQ_INV, 8 IDLE_I_EN, 8 TX_I_EN, 8 RadioFrequencyIdentificationMFRC522::COM_IRObits, 9 HI_ALERT_IRQ, 9 HI_ALERT_IRQ, 9		
RadioFrequencyIdentificationMFRC522.h, 55, 57 MFRS22_INT_TO_EN_MASK, 56 MFRS22_INT_TO_IRQ_MASK, 56 MFRS22_INT_TO_IRQ_MASK, 56 MFRS22_INT_TO_IRQ_REG, 57 MFRC522_DEFAULT_TIMEOUT, 57 PROGMEM, 57 RadioFrequencyIdentificationMFRC522::BIT_FRAMI—	writeRegisterBlock, 33	RadioFrequencyIdentificationMFRC522::CW_GS_←
MFR522_INT_TO_EN_MASK, 56 MFR522_INT_TO_EN_REG, 56 MFR522_INT_TO_IRQ_MASK, 56 MFR522_INT_TO_IRQ_MSK, 56 MFRC522_IDMOIDSIS, MDI_IQ_I 3 MDI_IQ_I 3 MDD_IQ_I 3 MD_IQ_I 3 MDD_IQ_I 3 MDD_IQ_I 3 MDD_IQ_I 3 MDD_IQ_I 3 MDD_IQ_I 3 M	RadioFrequencyIdentificationMFRC522.cpp, 50, 51	Pbits, 12
MFR522_INT_TO_EN_REG, 56 MFR522_INT_TO_IRQ_RASK, 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_NOT_VALID, 7 VALUES_AFTER_COLL, 7 value, 7 RadioFrequencyIdentificationMFRC522::COM_I_E ← Nbits, 7 ERR_I_EN, 8 H_I_ALERT_I_EN, 8 I_DC_ALERT_I_EN, 8 I_DC_ALERT_I_EN, 8 RX_I_EN, 8 TIMER_I_EN, 8 TX_I_EN, 8	RadioFrequencyIdentificationMFRC522.h, 55, 57	CW_GS_P, 12
MFR522_INT_TO_IRQ_MASK, 56 MFR522_INT_TO_IRQ_REG, 57 MFRC522_DEFAULT_TIMEOUT, 57 PROGMEM, 57 RadioFrequencyIdentificationMFRC522::BIT_FRAMI→	MFR522_INT_TO_EN_MASK, 56	char, 12
MFR522_INT_TO_IRQ_REG, 57 MFRC522_DEFAULT_TIMEOUT, 57 PROGMEM, 57 RadioFrequencyIdentificationMFRC522::BIT_FRAMI.→	MFR522_INT_TO_EN_REG, 56	value, 13
MFRC522_DEFAULT_TIMEOUT, 57 PROGMEM, 57 RadioFrequencyIdentificationMFRC522::BIT_FRAMI⇔	MFR522 INT TO IRQ MASK, 56	RadioFrequencyIdentificationMFRC522::DEMODbits,
MFRC522_DEFAULT_TIMEOUT, 57 PROGMEM, 57 RadioFrequencyIdentificationMFRC522::BIT_FRAMI⇔	MFR522 INT TO IRQ REG, 57	13
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_IE↔ Nbits, 7 ERR_IEN, 8 I_RQ_INV, 8 IDLE_IEN, 8 LO_ALERT_IEN, 8 IZ_EN, 8 TX_IEN, 8 TX_IEN, 8 TX_IEN, 8 Value, 8 RadioFrequencyIdentificationMFRC522::COM_IRQbits, 9 RadioFrequencyIdentificationMFRC522::COM_IRQbits, 9 RadioFrequencyIdentificationMFRC522::COM_IRQbits, 9 RADIOR RADI		ADD IQ, 13
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::COLLbits, 6 COLL_POS_NOT_VALID, 7 Char, 7 VALUES_AFTER_COLL, 7 value, 7 RadioFrequencyIdentificationMFRC522::COM_IE↔ Nbits, 7 RadioFrequencyIdentificationMFRC522::COM_IE↔ Nbits, 7 RadioFrequencyIdentificationMFRC522::COM_IE↔ Nbits, 7 RER_I_EN, 8 HI_ALERT_I_EN, 8 I_RQ_INV, 8 IDLE_I_EN, 8 INLE_I_EN, 8 TIMER_I_EN, 8 TIMER_I_EN, 8 TIMER_I_EN, 8 TIMER_I_EN, 8 TIMER_I_EN, 8 value, 8 RadioFrequencyIdentificationMFRC522::COM_IRQbits, 9 ER_IRQ, 9 HI_ALERT_IRQ, 9 RADIOFRED T_PRESCAL_EVEN, 13 TAU_RCV, 14 TAU_SYNC, 14 value, 14 RadioFrequencyIdentificationMFRC522::DIV_IENbits, 14 CRC_I_EN, 14 char, 14 CRC_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 COLL_ERR, 17 COLL_ERR, 17 COLL_ERR, 17 CRC_I_EN, 14 char, 14 CRC_I_EN, 14 char, 14 CRC_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 COLL_ERR, 17 COLL_ERR, 17 CRC_I_EN, 17 PARITY_ERR, 17 PARITY_ERR, 17 PARITY_ERR, 17 PARITY_ERR, 17 Value, 17		- '
NGbits, 5 char, 6 RX_ALIGN, 6 RX_ALIGN, 6 RX_ALIGN, 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 I_RQ_INV, 8 IDLE_I_EN, 8 I_RQ_INV, 8 IDLE_I_EN, 8 RX_I_EN, 8 RX_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 TAU_RCV, 14 TAU_SYNC, 14 Value, 14 RadioFrequencyIdentificationMFRC522::DIV_I_ENbits, 14 CRC_I_EN, 14 CRC_I_EN, 14 CRC_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 CRC_ERR, 17 PARITY_ERR, 17 PARITY_ERR, 17 PROTOCOL_ERR, 17 TEMP_ERR, 17 Value, 17		
char, 6 RX_ALIGN, 6 RX_ALIGN, 6 START_SEND, 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 RX_I_EN, 8 RX_I_EN, 8 TIMER_I_EN, 8 TX_I_EN, 8 TX_I_EN, 8 TX_I_EN, 8 TX_I_EN, 8 TX_I_EN, 8 RADIOFREQUENCYIDENTIFICATION FRC522::COM_IRQbits, 9 ERR_IRQ, 9 HI_ALERT_IRQ, 9 TAU_SYNC, 14 value, 14 RadioFrequencyIdentificationMFRC522::DIV_I_ENbits, 14 CRC_I_EN, 14 CRC_I_EN, 14 CRC_I_EN, 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 CRC_ERR, 17 PARITY_ERR, 17 PARITY_ERR, 17 PROTOCOL_ERR, 17 TEMP_ERR, 17 value, 17	· · · ·	
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 IDLE_I_EN, 8 IDLE_I_EN, 8 RX_I_EN, 8 TIMER_I_EN, 8 TIMER_I_EN, 8 TIMER_I_EN, 8 TX_I_EN, 8 TIMER_I_EN, 8 RadioFrequencyIdentificationMFRC522::COM_IRQbits, 9 ERR_IRQ, 9 HI_ALERT_IRQ, 9 RadioFrequencyIdentificationMFRC522::COM_IRQbits, 16 RadioFrequencyIdentificationMFRC522::COM_IRQbits, 16 RadioFrequencyIdentificationMFRC522::COM_IRQbits, 16 RadioFrequencyIdentificationMFRC522::ERRORbits, 16 RADIOFREQUENCYIDENTIFICATION FRC522::ERRORbits, 16 RADIOFREQUENCYIDENTIFICATION FRC522::ERRORbits, 16 RADIOFREQUENCYIDENTIFICATION FRC522::ERRORbits, 16 RADIOFREQUENCYIDENTIFICATION FRC522::ERRORbits, 17 CRC_ERR, 17 CRC_ERR, 17 CRC_ERR, 17 CRC_ERR, 17 TEMP_ERR, 17 PARITY_ERR, 17 PROTOCOL_ERR, 17 TEMP_ERR, 17 value, 17		-
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 IDLE_I_EN, 8 IDLE_I_EN, 8 TIMER_I_EN, 8 TIMER_I_EN, 8 TIMER_I_EN, 8 TRADIOFREQUENCYIDENTIFICATION FRC522::COM_IRQbits, 16 RadioFrequencyIdentificationMFRC522::COM_IRQbits, 16 RadioFrequencyIdentificationMFRC522::COM_IRQbits, 16 RadioFrequencyIdentificationMFRC522::ERRORbits, 16 RadioFrequencyIdentificationMFRC522::ERRORbits, 16 RadioFrequencyIdentificationMFRC522::COM_IRQbits, 16 RadioFrequencyIdentificationMFRC522::ERRORbits, 16 RadioFrequencyIdentificationMFRC522::ERRORbits, 16 RadioFrequencyIdentificationMFRC522::ERRORbits, 16 RadioFrequencyIdentificationMFRC522::ERRORbits, 16 RadioFrequencyIdentificationMFRC522::ERRORbits, 17 COLL_ERR, 17 CRC_ERR, 17 CRC_ERR, 17 CRC_ERR, 17 CRC_ERR, 17 CRC_ERR, 17 TALER, 17 PARITY_ERR, 17 PARITY_ERR, 17 PROTOCOL_ERR, 17 TEMP_ERR, 17 value, 17		
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 ← Noits, 7 ERR_I_EN, 8 HI_ALERT_I_EN, 8 I_RQ_INV, 8 IDLE_I_EN, 8 TIMER_I_EN, 8 TX_I_EN, 8 TX_I_EN, 8 TX_I_EN, 8 TX_I_EN, 8 TX_IEN, 8 RadioFrequencyIdentificationMFRC522::COM_IRQbits, 9 ERR_IRQ, 9 HI_ALERT_IRQ, 9 TEMP_ERR, 17 PROTOCOL_ERR, 17 TEMP_ERR, 17 VALUES_AFTER_COLL, 7 RADIOFRATION AND TRACTICATION		
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 IDLE_I_EN, 8 IDLE_I_EN, 8 RX_I_EN, 8 TIMER_I_EN, 8 TX_I_EN, 8 TX_I_EN, 8 value, 8 RadioFrequencyIdentificationMFRC522::COM_IRQbits, 9 ERR_IRQ, 9 HI_ALERT_IRQ, 9 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 RadioFrequencyIdentificationMFRC522::ERRORbits, 16 RAGIOFREQUENCYIDENTIFICATION CRC_ERR, 17 CRC_ERR, 17 CRC_ERR, 17 PARITY_ERR, 17 PROTOCOL_ERR, 17 TEMP_ERR, 17 TEMP_ERR, 17 value, 17	——————————————————————————————————————	
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_EQ_NV, 8 RX_I_EN, 8 RX_I_EN, 8 TX_I_EN, 8 TX_I_EN, 8 value, 8 RadioFrequencyIdentificationMFRC522::COM_IRQbits, 15 CRC_IRQ, 15 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 CRC_ERR, 17 CRC_ERR, 17 PARITY_ERR, 17 PROTOCOL_ERR, 17 TEMP_ERR, 17 value, 17		
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 H_ALERT_I_EN, 8 IDLE_I_EN, 8 IDLE_I_EN, 8 RX_I_EN, 8 RX_I_EN, 8 TX_I_EN, 8 TX_I_EN, 8 TX_I_EN, 8 value, 8 RadioFrequencyIdentificationMFRC522::COM_IRQbits, PROMITY OF THE PULL, 15 MFIN_ACT_I_EN, 15 CRC_IRQ, 15 char, 15 MFIN_ACT_IRQ, 16 SET2, 16 value, 16 RadioFrequencyIdentificationMFRC522::ERRORbits, 16 BUFFER_OVFL, 17 COLL_ERR, 17 COLL_ERR, 17 CRC_ERR, 17 CRC_ERR, 17 CRC_ERR, 17 CRC_ERR, 17 PROTOCOL_ERR, 17 TEMP_ERR, 17 TEMP_ERR, 17 Value, 17		—— ·
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 IDLE_I_EN, 8 IDLE_I_EN, 8 RX_I_EN, 8 RX_I_EN, 8 RX_I_EN, 8 TIMER_I_EN, 8 TX_I_EN, 17 TX_I		
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 RX_I_EN, 8 RX_I_EN, 8 RX_I_EN, 8 TX_I_EN, 8 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 COLL_ERR, 17 CRC_ERR, 17 CRC_ERR, 17 CRC_ERR, 17 CRC_ERR, 17 PARITY_ERR, 17 PARITY_ERR, 17 PARITY_ERR, 17 PROTOCOL_ERR, 17 TEMP_ERR, 17 TEMP_ERR, 17 value, 17		
VALUES_AFTER_COLL, 7 value, 7 RadioFrequencyIdentificationMFRC522::COM_I_E ←		
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 IDLE_I_EN, 8 RX_I_EN, 8 RX_I_EN, 8 TX_I_EN, 8 TX_I_EN, 8 RadioFrequencyIdentificationMFRC522::ERRORbits, Value, 8 RadioFrequencyIdentificationMFRC522::COM_IRQbits, PARITY_ERR, 17 CRC_ERR, 17 CRC_ERR, 17 CRC_ERR, 17 CRC_ERR, 17 CRC_ERR, 17 Char, 15 CRC_IRQ, 15 char, 15 SET2, 16 Value, 16 RadioFrequencyIdentificationMFRC522::ERRORbits, 16 BUFFER_OVFL, 17 CRC_ERR, 17 CRC_ERR, 17 CRC_ERR, 17 CART, 17 PARITY_ERR, 17 PARITY_ERR, 17 PARITY_ERR, 17 PROTOCOL_ERR, 17 TEMP_ERR, 17 Value, 17		
RadioFrequencyIdentificationMFRC522::COM_I_E ← Nbits, 7 ERR_I_EN, 8 HI_ALERT_I_EN, 8 IDLE_I_EN, 8 IDLE_I_EN, 8 RX_I_EN, 8 RX_I_EN, 8 TX_I_EN, 8 TX_I_EN, 8 TX_I_EN, 8 RadioFrequencyIdentificationMFRC522::COM_IRQbits, PREMICE OF THE PROTOCOL_ERR, 17 TERR_IRQ, 9 HI_ALERT_IRQ, 9 CRC_IRQ, 15 char, 15 MFIN_ACT_IRQ, 16 SET2, 16 RadioFrequencyIdentificationMFRC522::ERRORbits, 16 BUFFER_OVFL, 17 COLL_ERR, 17 CRC_ERR, 17 Char, 17 PARITY_ERR, 17 PROTOCOL_ERR, 17 TEMP_ERR, 17 TEMP_ERR, 17 value, 17		· · ·
Nbits, 7 ERR_I_EN, 8 HI_ALERT_I_EN, 8 I_RQ_INV, 8 IDLE_I_EN, 8 RX_I_EN, 8 RX_I_EN, 8 TX_I_EN, 8 Value, 8 RadioFrequencyIdentificationMFRC522::COM_IRQbits, 9 RadioFrequencyIdentificationMFRC522::COM_IRQbits, 9 PROTOCOL_ERR, 17 CRC_ERR, 17 PROTOCOL_ERR, 17 PROTOCOL_ERR, 17 TEMP_ERR, 17 TEMP_ERR, 17 Value, 9 Value, 17		
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 RX_I_EN, 8 TIMER_I_EN, 8 Value, 16 BUFFER_OVFL, 17 COLL_ERR, 17 CCL_ERR, 17 CRC_ERR, 17 CRC_ERR, 17 Char, 17 PARITY_ERR, 17 PROTOCOL_ERR, 17 ERR_IRQ, 9 HI_ALERT_IRQ, 9 WFIN_ACT_IRQ, 16 SET2, 16 Value, 16 RadioFrequencyIdentificationMFRC522::ERRORbits, COLL_ERR, 17 CRC_ERR, 17 PARITY_ERR, 17 PROTOCOL_ERR, 17 TEMP_ERR, 17 value, 17		- :
HI_ALERT_I_EN, 8 I_RQ_INV, 8 Value, 16 RAZ_I_EN, 8 RX_I_EN, 8 RX_I_EN, 8 TX_I_EN, 8 Value, 16 BUFFER_OVFL, 17 COLL_ERR, 17 COLL_ERR, 17 CRC_ERR, 17 CRC_ERR, 17 Char, 17 PARITY_ERR, 17 PROTOCOL_ERR, 17 PROTOCOL_ERR, 17 TEMP_ERR, 17 Value, 17 TEMP_ERR, 17 Value, 17		
I_RQ_INV, 8 IDLE_I_EN, 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::ERRORbits, 16 BUFFER_OVFL, 17 COLL_ERR, 17 CRC_ERR, 17 CRC_ERR, 17 Char, 17 PARITY_ERR, 17 PARITY_ERR, 17 PROTOCOL_ERR, 17 TEMP_ERR, 17 TEMP_ERR, 17 Value, 17		
IDLE_I_EN, 8 LO_ALERT_I_EN, 8 RX_I_EN, 8 RX_I_EN, 8 TIMER_I_EN, 8 TX_I_EN, 8 Value, 8 RadioFrequencyIdentificationMFRC522::COM_IRQbits, PARITY_ERR, 17 PROTOCOL_ERR, 17 PROTOCOL_ERR, 17 TEMP_ERR, 17 TEMP_ERR, 17 Value, 17	:	
LO_ALERT_I_EN, 8 RX_I_EN, 8 RX_I_EN, 8 COLL_ERR, 17 TX_I_EN, 8 value, 8 RadioFrequencyIdentificationMFRC522::COM_IRQbits, 9 PROTOCOL_ERR, 17 PROTOCOL_ERR, 17 ERR_IRQ, 9 HI_ALERT_IRQ, 9 value, 17	:	
RX_I_EN, 8 BUFFER_OVFL, 17 TIMER_I_EN, 8 COLL_ERR, 17 TX_I_EN, 8 CRC_ERR, 17 value, 8 char, 17 RadioFrequencyIdentificationMFRC522::COM_IRQbits, 9 PARITY_ERR, 17 ERR_IRQ, 9 TEMP_ERR, 17 HI_ALERT_IRQ, 9 value, 17		
TIMER_I_EN, 8 TX_I_EN, 8 Value, 8 RadioFrequencyIdentificationMFRC522::COM_IRQbits, 9 ERR_IRQ, 9 HI_ALERT_IRQ, 9 Value, 17 CCC_ERR, 17 CRC_ERR, 17 PARITY_ERR, 17 PROTOCOL_ERR, 17 TEMP_ERR, 17 value, 17		
TX_I_EN, 8 value, 8 RadioFrequencyIdentificationMFRC522::COM_IRQbits, 9 PROTOCOL_ERR, 17 PROTOCOL_ERR, 17 TEMP_ERR, 17 TEMP_ERR, 17 value, 17	:	
value, 8 char, 17 RadioFrequencyIdentificationMFRC522::COM_IRQbits, 9 PROTOCOL_ERR, 17 ERR_IRQ, 9 TEMP_ERR, 17 HI_ALERT_IRQ, 9 value, 17		
RadioFrequencyIdentificationMFRC522::COM_IRQbits, 9 PARITY_ERR, 17 PROTOCOL_ERR, 17 ERR_IRQ, 9 TEMP_ERR, 17 HI_ALERT_IRQ, 9 value, 17		
9 PROTOCOL_ERR, 17 ERR_IRQ, 9 TEMP_ERR, 17 HI_ALERT_IRQ, 9 value, 17	value, 8	char, 17
ERR_IRQ, 9 TEMP_ERR, 17 HI_ALERT_IRQ, 9 value, 17	RadioFrequencyIdentificationMFRC522::COM_IRQbits,	
HI_ALERT_IRQ, 9 value, 17	9	PROTOCOL_ERR, 17
	ERR_IRQ, 9	TEMP_ERR, 17
IDLE_IRQ, 9 WR_ERR, 17	HI_ALERT_IRQ, 9	value, 17
	IDLE_IRQ, 9	WR_ERR, 17

FIFO_LEVEL, 18 FLUSH_BUFFER, 18 value, 18 RadioFrequencyIdentificationMFRC522::GS_Nbits, 18 CW_GS_N, 19 MOD_GS_N, 19 MOD_GS_N, 19 RadioFrequencyIdentificationMFRC522::MF_RXbits, 19 char, 20 PARITY_DISABLE, 20 value, 20 RadioFrequencyIdentificationMFRC522::MF_TXbits, 20 char, 20 pARITY_DISABLE, 20 value, 20 RadioFrequencyIdentificationMFRC522::MF_TXbits, 20 char, 20 radioFrequencyIdentificationMFRC522::MOD_GS Pbits, 21 char, 21 MOD_GS_P, 21 value, 21 RadioFrequencyIdentificationMFRC522::MODEbits, 22 CRC_PRESET, 22 char, 22 MSB_FIRST, 22 pCD_M_FIN, 22 TX_WAIT_RF, 23 value, 23 RadioFrequencyIdentificationMFRC522::RC_FGbits, 34 char, 34 RX_GAIN, 34 avalue, 35 RX_GC_EN, 35 RX_MC_RC_EN, 35 RX_MC_RC_EN, 36 RX_MC_RC_EN	RadioFrequencyIdentificationMFRC522::FIFO_LEVE ← Lbits, 17	RadioFrequencyIdentificationMFRC522::STATUS1bits, 39
value, 18	FIFO_LEVEL, 18	CRC_OK, 39
RadioFrequencyIdentificationMFRC522::MF_RXbits, 19	FLUSH_BUFFER, 18	CRC_READY, 39
CW GS N, 19 MOD_GS_N, 19 value, 19 RadioFrequencyIdentificationMFRC522::MF_RXbits, 19 robar, 20 PARITY_DISABLE, 20 value, 20 RadioFrequencyIdentificationMFRC522::MF_TXbits, 20 char, 20 robar, 20 RadioFrequencyIdentificationMFRC522::MF_TXbits, 20 char, 20 robar, 20 robar, 20 robar, 20 robar, 21 robar, 21 robar, 21 robar, 21 robar, 21 robar, 22 char, 21 robar, 22 char, 22 char, 22 char, 22 robar, 22 robar, 23 RadioFrequencyIdentificationMFRC522::MODEbits, 22 char, 22 robar, 22 robar, 23 RadioFrequencyIdentificationMFRC522::RF_CFGbits, 34 robar, 34 robar, 34 robar, 34 robar, 34 robar, 35 RadioFrequencyIdentificationMFRC522::RX_MOD+Ebits, 35 RadioFrequencyIdentificationMFRC522::RX_MOD+Ebits, 36 RX_NO_ERR, 36 RX_SPEED, 36 robar, 36 RX_MAIT, 36 UART_SEL, 36 value, 37 RadioFrequencyIdentificationMFRC522::RX_THRES+HOLDits, 37 coll_LEVEL, 37 robar, 37 MIN_LEVEL, 37 robar, 37 MIN_LEVEL, 37 robar, 38 RadioFrequencyIdentificationMFRC522::SERIAL_SP+EDbits, 38 RadioFrequencyIdentification	value, 18	char, 39
MOD_GS_N_19 value, 19 RadioFrequencyIdentificationMFRC522::MF_RXbits, 19 char, 20 value, 20 RadioFrequencyIdentificationMFRC522::MF_TXbits, 20 value, 20 TX_WAIT, 20 value, 21 RadioFrequencyIdentificationMFRC522::MOD_GS_Pbits, 21 char, 21 MOD_GS_P, 21 value, 21 RadioFrequencyIdentificationMFRC522::MOD_Ebits, 21 char, 21 MOD_GS_P, 21 value, 21 RadioFrequencyIdentificationMFRC522::MODEbits, 22 CRC_PRESET, 22 char, 22 MSB_FIRST, 23 value, 23 RadioFrequencyIdentificationMFRC522::RF_CFGbits, 34 RX_GAIN, 34 RX_GAIN, 34 RX_GAIN, 34 RX_GAIN, 34 RX_GAIN, 34 RX_GAIN, 34 RX_MIT_RF_S3 value, 35 RACIOFrequencyIdentificationMFRC522::RX_MOD→Ebits, 35 RX_NO_ERR, 36 RX_WAIT, 36 RX_WAIT, 36 RX_WAIT, 36 VART_SEL, 36 value, 37 RadioFrequencyIdentificationMFRC522::RX_THRES→HOLDits, 37 COLL_LEVEL, 37 char, 37 value, 38 RAdioFrequencyIdentificationMFRC522::RX_THRES→HOLDits, 37 COLL_LEVEL, 37 value, 38 RAdioFrequencyIdentificationMFRC522::RX_SELbits, 36 RAGIOFrequencyIdentificationMFRC522::RX_THRES→HOLDits, 37 COLL_LEVEL, 37 value, 38 RAdioFrequencyIdentificationMFRC522::RX_SELbits, 38 BR_TO, 38 BR_TO, 38 BR_TIO, 38 BR_TIO, 38 BR_TIO, 38 RAGIOFrequencyIdentificationMFRC522::SERIAL_SP→EDDbits, 38 BR_TIO, 38 BR_TIO, 38 RAGIOFrequencyIdentificationMFRC522::SERIAL_SP→EDDbits, 38 RAGIOFrequencyIdentificationMFRC522::SERIAL_SP→EDDbits, 38 RAGIOFrequencyIdentificationMFRC522::TX_SELbits, 38 RAGIOFrequencyIdentificationMFRC522::TX_SELbits, 38 RAGIOFrequencyIdentificationMFRC522::TX_SELbits, 38 RAGIOFrequencyIdentificationMFRC522::TX_SELbits, 38 RAGIOFrequencyIdentificationMFRC522::TX_SELbits, 38 RAGIOFr	RadioFrequencyIdentificationMFRC522::GS_Nbits, 18	HI_ALERT, 39
value, 19 char, 20 PARITY_DISABLE, 20 value, 20 RadioFrequencyIdentificationMFRC522::MF_TXbits, 20 char, 20 TX_WAIT, 20 value, 21 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 char, 34 Achar, 34 RX_GAIN, 34 value, 35 RAGIOFrequencyIdentificationMFRC522::RX_MODE-Ebits, 35 char, 35 RX_OC_ER, 36 RX_SPEED, 36 value, 36 RX_MAIT_SEL, 36 value, 37 COLL_LEVEL, 37 char, 37 MIN_LEVEL, 37 value, 38 RadioFrequencyIdentificationMFRC522::SERIAL_SP+ ECDbits, 38 RAdioFrequencyIdentificationMFRC522::SERIAL_SP+ RAdioFrequencyIdentificationMFRC522		IRQ, 40
value, 19 char, 20 PARITY_DISABLE, 20 value, 20 RadioFrequencyIdentificationMFRC522::MF_TXbits, 19 radioFrequencyIdentificationMFRC522::MF_TXbits, 20 char, 20 RadioFrequencyIdentificationMFRC522::MF_TXbits, 20 char, 20 RadioFrequencyIdentificationMFRC522::MF_TXbits, 20 char, 20 radioFrequencyIdentificationMFRC522::MOD_GS_ Pbits, 21 char, 21 dNOD_GS_P, 21 value, 21 RadioFrequencyIdentificationMFRC522::MODEbits, 22 cRC_PRESET, 22 char, 22 dNSB_FIRST, 22 POL_M_FIN, 22 TX_WAIT_RF, 23 value, 23 RadioFrequencyIdentificationMFRC522::RF_CFGbits, 34 char, 34 char, 34 RX_GAIN, 34 value, 35 RadioFrequencyIdentificationMFRC522::RX_MOD+Ebits, 35 char, 35 RX_OE_RS, 36 RX_SPEED, 36 value, 36 RX_MAIT, 36 UART_SEL, 36 value, 37 COL_L_EVEL, 37 char, 37 MIN_LEVEL, 37 value, 38 RadioFrequencyIdentificationMFRC522::SERIAL_SP+EEDbits, 38 RadioFrequencyIdentificationMFRC522::	MOD GS N, 19	LO ALERT, 40
RadioFrequencyIdentificationMFRC522::MF_RXbits, 19 char, 20 value, 20 rX_WAIT, 20 value, 20 rX_WAIT, 20 value, 21 RadioFrequencyIdentificationMFRC522::MOD_GS_Ditlant, 21 Achar, 21 RadioFrequencyIdentificationMFRC522::MOD_GS_Ditlant, 21 Achar, 21 Achar, 21 Achar, 21 Achar, 21 RadioFrequencyIdentificationMFRC522::MODEbits, 22 CRC_PRESET, 22 char, 22 MSB_FIRST, 22 POL_M_FIN, 22 rX_WAIT_RF, 23 value, 23 RadioFrequencyIdentificationMFRC522::RF_CFGbits, 34 char, 34 char, 34 char, 34 char, 34 char, 34 char, 35 RX_GRC_EN, 35 RX_CRC_EN, 35 RX_MULTIPLE, 35 RX_MO_ERR, 36 RX_SPEED, 36 value, 37 COLL_LEVEL, 37 char, 37 MIN_LEVEL, 37 char, 45 RadioFrequencyIdentificationMFRC522::SERIAL_SP-EEDbits, 38 BR_TI, 38 RadioFrequencyIdentificationMFRC522::SERIAL_SP-EEDbits, 38 BR_TI, 38 RadioFrequencyIdentificationMFRC522::SERIAL_SP-EEDbits, 38 BR_TI, 38 RadioFrequencyIdentificationMFRC522::SERIAL_SP-EEDbits, 38 BR_TI, 38 RadioFrequencyIdentificationMFRC522::Uid, 47		
char, 20	RadioFrequencyIdentificationMFRC522::MF_RXbits, 19	value, 40
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, 33 RadioFrequencyIdentificationMFRC522::RF_CFGbits, 34 char, 34 RX_GAIN, 34 value, 35 RadioFrequencyIdentificationMFRC522::RX_MOD_Ebits, 35 char, 35 RX_NO_ERR, 36 RX_VO_ERR, 36 RX_SPEED, 36 value, 37 COLL_LEVEL, 37 char, 37 MIN_LEVEL, 37 value, 38 RadioFrequencyIdentificationMFRC522::SERIAL_SP_EDDits, 38 BR_TI, 38 RAdioFrequencyIdentificationMFRC522::SERIAL_SP_EDDits, 38 BR_TI, 38 RAdioFrequencyIdentificationMFRC522::SERIAL_SP_EDDits, 38 BR_TI, 38 40 char, 41 INC_CPC_ENS, 41 MC_CPCPCE_HS, 41 MF_CRYPTOI_ON, 41 TEMP_SENS_CLEAR, 41 value, 41 T_AUTO_RESTART, 42 T_AUTO, 42 T_AU		RadioFrequencyIdentificationMFRC522::STATUS2bits,
value, 20 Char, 20 TX_WAIT, 20 value, 21 RadioFrequencyIdentificationMFRC522::MOD_GS_→ Pbits, 21 MOD_GS_P, 21 value, 21 RadioFrequencyIdentificationMFRC522::MOD_Ebits, 22 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 RX_CRC_EN, 35 RX_CRC_EN, 35 RX_MULTIPLE, 35 RX_NO_ERR, 36 RX_WHIT_BE, 36 RX_WAIT, 36 UART_SEL, 36 value, 37 COLL_LEVEL, 37 char, 37 MIN_LEVEL, 37 value, 38 RadioFrequencyIdentificationMFRC522::RSERIAL_SP—ECDbits, 38 BR_TO, 38 BR_TO, 38 BR_TO, 38 BR_TO, 38 BR_TO, 38 BR_TO, 38 BR_TI, 38 Char, 45 MCDEM_STATE, 41 MCDEM_STATE, 41 TEMP_SENS_CLEAR, 41 value, 41 RadioFrequencyIdentificationMFRC522::T_MODEbits, 42 T_AUTO, 42 T_	PARITY DISABLE, 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 Char, 34 RX_GAIN, 34 value, 35 RadioFrequencyIdentificationMFRC522::RS_MODE-Ebits, 35 RX_CRC_EN, 35 RX_CRC_EN, 35 RX_NO_ERR, 36 RX_SPEED, 36 value, 36 RAIOFrequencyIdentificationMFRC522::RX_SELbits, 36 RX_CRC_EN, 36 RX_WAIT_36 UART_SEL, 36 value, 37 RadioFrequencyIdentificationMFRC522::RX_THRES-HOLDbits, 37 COLL_LEVEL, 37 value, 38 RadioFrequencyIdentificationMFRC522::SERIAL_SP-EEDbits, 38 BR_TO, 38 RR_TO, 38 RRAIOFREQUENCYIdentificationMFRC522::SERIAL_SP-EEDbits, 38 BR_TO, 38 RRAIOFREQUENCYIdentificationMFRC522::Uid, 47 RadioFrequencyIdentificationMFRC522::Uid, 47 RadioFrequencyIdentificationMFRC522::Uid, 47 RAIOFREQUENCYIdentificationMFRC522::Uid, 47 RadioFrequencyIdentificationMFRC522::TX_SELbits, 46 value, 47 RadioFrequencyIdentificationMFRC522::Uid, 47 RadioFrequencyIdentificationMFRC522::Uid, 47	value, 20	char, 41
char. 20	RadioFrequencyIdentificationMFRC522::MF TXbits, 20	I2C_FORCE_HS, 41
TX_WAIT, 20 value, 21 RadioFrequencyIdentificationMFRC522::MOD_GS_→ Pbits, 21 char, 21 dNOD_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 char, 34 char, 34 char, 34 char, 34 RX_GAIN, 34 value, 35 RX_CRC_EN, 35 RX_CRC_EN, 35 RX_NO_ERR, 36 RX_SPEED, 36 value, 36 RX_WAIT, 36 UART_SEL, 36 value, 37 RadioFrequencyIdentificationMFRC522::RX_SELbits, 36 avalue, 37 RadioFrequencyIdentificationMFRC522::RX_THRES←HOLDbits, 37 char, 37 MIN_LEVEL, 37 value, 38 RadioFrequencyIdentificationMFRC522::SERIAL_SP←EEDbits, 38 BR_T1, 38 BR_T1, 38 BR_T1, 38 BR_T1, 38 MODEM_STATE, 41 TEMP_SENS_CLEAR, 41 value, 41 RadioFrequencyIdentificationMFRC522::T_MODEbits, 44 TX_AUTO, 42 T_AUTO, 42 T_ATED, 44 INV_TX_RF_OF, 44 INV_TX_RF_OF, 44 INV_TX_RF_OF, 44 INV_TX	,	MF CRYPTO1 ON, 41
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 T_X_WAIT_RF, 23 value, 23 RadioFrequencyIdentificationMFRC522::RF_CFGbits, 34 char, 34 char, 34 char, 34 char, 34 char, 35 RX_GAIN, 34 value, 35 RX_NO_ERR, 36 RX_NO_ERR, 36 RX_NO_ERR, 36 RX_NO_ERR, 36 RX_NO_ERR, 36 RX_WAIT_136 UART_SEL, 36 value, 37 RadioFrequencyIdentificationMFRC522::RX_THRES→ HOLDbits, 37 COLL_LEVEL, 37 value, 38 RadioFrequencyIdentificationMFRC522::RX_THRES→ HOLDbits, 37 COLL_LEVEL, 37 value, 38 RadioFrequencyIdentificationMFRC522::SERIAL_SP→ EEDbits, 38 BR_T0, 38 RAGIOFrequencyIdentificationMFRC522::SERIAL_SP→ EEDbits, 38 BR_T1, 38 RAGIOFrequencyIdentificationMFRC522::Uid, 47 RadioFrequencyIdentificationMFRC522::Uid, 47 RadioFrequencyIdentificationMFRC522::Uid, 47 RadioFrequencyIdentificationMFRC522::Uid, 47 RadioFrequencyIdentificationMFRC522::Uid, 47		MODEM STATE, 41
RadioFrequencyIdentificationMFRC522::MOD_GS_ Pbits, 21 MOD_GS_P, 21 value, 21 RadioFrequencyIdentificationMFRC522::MODEbits, 22 RadioFrequencyIdentificationMFRC522::MODEbits, 22 CRC_PRESET, 22 char, 22 MSB_FIRST, 22 POL_M_FIN, 22 T_X_WAIT_RF, 23 value, 23 RadioFrequencyIdentificationMFRC522::RF_CFGbits, 34 RX_GAIN, 34 value, 35 RX_GAIN, 34 value, 35 RX_MULTIPLE, 35 RX_MULTIPLE, 35 RX_NO_ERR, 36 RX_SPEED, 36 value, 36 RX_WAIT, 36 UART_SEL, 36 value, 37 RadioFrequencyIdentificationMFRC522::RX_THRES← HOLDbits, 37 COL_LEVEL, 37 value, 38 RadioFrequencyIdentificationMFRC522::SERIAL_SP← EEDbits, 38 RADIOFREQUENCYIdentificationMFRC522::TX_MODEbits, 43 Value, 41 RadioFrequencyIdentificationMFRC522::T_MODEbits, 41 T_AUTO_RESTART, 42 T_AUTO_RESTART T_AUTO_RATE T_AUTO_RATE T_AUTO_RATE T_AUTO_RATE T_AUTO_RE		TEMP SENS CLEAR, 41
Pbits, 21 char, 21 MOD_GS_P, 21 value, 21 RadioFrequencyIdentificationMFRC522::MODEbits, 22 CRC_PRESET, 22 char, 22 MSB_FIRST, 22 MSB_FIRST, 22 MSB_FIRST, 22 T_WAIT_RF, 23 value, 23 RadioFrequencyIdentificationMFRC522::RF_CFGbits, 34 char, 34 char, 34 RX_GAIN, 34 value, 35 RAIOFrequencyIdentificationMFRC522::RS_MODE Ebits, 35 char, 35 RX_CRC_EN, 35 RX_MULTIPLE, 35 RX_NDL_TIPLE, 35 RX_NDL_TIPLE, 36 value, 37 RadioFrequencyIdentificationMFRC522::RX_TRESE-HOLDbits, 37 cOLL_LEVEL, 37 char, 37 MIN_LEVEL, 37 value, 38 RAIOFrequencyIdentificationMFRC522::SERIAL_SPEEDbits, 38 BR_T0, 38 BR_T0, 38 BR_T0, 38 BR_T1, 38 RAIOFREQUENCYIdentificationMFRC522::TX_SELbits, 38 BR_T1, 38 RAIOFREQUENCYIdentificationMFRC522::SERIAL_SPEEDbits, 38 BR_T0, 38 BR_T1, 38 RAIOFREQUENCYIdentificationMFRC522::TMODEbits, 41 T_AUTO, 42 T_AUTO, 42 T_AUTO, 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_ON, 44 INV_TX1_RF_OFF, 44 INV_TX2_RF_OFF, 44 INV_TX_RF_OFF, 4		
char, 21 MOD_GS_P, 21 value, 21 RadioFrequencyIdentificationMFRC522::MODEbits, 22 CRC_PRESET, 22 char, 22 MSB_FIRST, 22 MSB_FIRST, 22 POL_M_FIN, 22 T_WAIT_RF, 23 value, 23 RadioFrequencyIdentificationMFRC522::RF_CFGbits, 34 char, 34 char, 34 RX_GAIN, 34 value, 35 RadioFrequencyIdentificationMFRC522::RX_MOD← Ebits, 35 char, 35 RX_CRC_EN, 35 RX_MULTIPLE, 35 RX_MULTIPLE, 35 RX_NO_ERR, 36 RX_WAIT_, 86 UART_SEL, 36 value, 37 RadioFrequencyIdentificationMFRC522::RX_THRES← HOLDbits, 37 COLL_LEVEL, 37 value, 38 RadioFrequencyIdentificationMFRC522::SERIAL_SP← EEDbits, 38 RAGIOFREQuencyIdentificationMFRC522::SERIAL_SP← EEDbits, 38 RAGIOFREQuencyIdentificationMFRC522::SERIAL_SP← EEDbits, 38 BR_T0, 38 BR_T0, 38 BR_T1, 38 HAITO_RESTART, 42 T_AUTO_ 42 T_PRESCALER_HI, 42 value, 42 RadioFrequencyIdentificationMFRC522::TX_ASKbits, 42 char, 43 RadioFrequencyIdentificationMFRC522::TX_CONTR↔ OLbits, 43 char, 44 INV_TX1_RF_OFF, 44 INV_TX2_RF_OFF, 44 INV_TX2_RF_OFF, 44 INV_TX_RF_OFF,		
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 char, 34 RX_GAIN, 34 value, 35 RadioFrequencyIdentificationMFRC522::RS_MOD← Ebits, 35 char, 35 RX_CRC_EN, 35 RX_CRC_EN, 35 RX_MULTIPLE, 35 RX_MULTIPLE, 35 RX_NO_ERR, 36 RX_SPEED, 36 value, 37 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 M		
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 RX_GAIN, 34 value, 35 RadioFrequencyIdentificationMFRC522::RX_MOD← Ebits, 35 char, 35 RX_CRC_EN, 35 RX_NO_ERR, 36 RX_SPEED, 36 value, 37 RadioFrequencyIdentificationMFRC522::RX_SELbits, 36 RAZ_MAIT, 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 RR_T0, 38 BR_T0, 38 BR_T1, 38 BR_T1, 38 BR_T1, 38 RadioFrequencyIdentificationMFRC522::Uid, 47 RadioFrequencyIdentificationMFRC522::Uid, 47 RadioFrequencyIdentificationMFRC522::Uid, 47 RadioFrequencyIdentificationMFRC522::Uid, 47 RadioFrequencyIdentificationMFRC522::Uid, 47 RadioFrequencyIdentificationMFRC522::Uid, 47		T AUTO RESTART, 42
RadioFrequencyIdentificationMFRC522::MODEbits, 22 CRC_PRESCALER_HI, 42 value, 42 MSB_FIRST, 22 POL_M_FIN, 22 TX_WAIT_RF, 23 value, 23 RadioFrequencyIdentificationMFRC522::RF_CFGbits, 34 RX_GAIN, 34 value, 35 RadioFrequencyIdentificationMFRC522::RX_MOD← Ebits, 35 Char, 35 RX_CRC_EN, 35 RX_NO_ERR, 36 RX_MULTIPLE, 35 RX_SPEED, 36 value, 36 RadioFrequencyIdentificationMFRC522::RX_SELbits, 36 RAZ_MAIT, 36 UART_SEL, 36 value, 37 RadioFrequencyIdentificationMFRC522::RX_THRES← HOLDbits, 37 COLL_LEVEL, 37 char, 37 value, 38 RadioFrequencyIdentificationMFRC522::SERIAL_SP← EEDbits, 38 RadioFrequencyIdentificationMFRC522::SERIAL_SP← EEDbits, 38 RATI, 38 BR_T0, 38 BR_T1, 38 BR_T1, 38 RADIOFREQUENCYIdentificationMFRC522::Uid, 47 RadioFrequencyIdentificationMFRC522::Uid, 47 RadioFrequencyIdentificationMFRC522::SERIAL_SP← EEDbits, 38 RR_T1, 38 RR_T1, 38 RR_T1, 38 RR_T1, 38 RADIOFREQUENCYIDENTIFICATIONMFRC522::Uid, 47 RadioFrequencyIdentificationMFRC522::Uid, 47 RadioFrequencyIdentificationMFRC522::Uid, 47		
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 RAWAIT, 36 UART_SEL, 36 value, 37 RadioFrequencyIdentificationMFRC522::RX_THRES← HOLDbits, 37 COLL_LEVEL, 37 char, 37 value, 38 RadioFrequencyIdentificationMFRC522::SERIAL_SP← EEDbits, 38 RAGIOFrequencyIdentificationMFRC522::SERIAL_SP← EEDbits, 38 BR_T0, 38 BR_T0, 38 BR_T0, 38 BR_T0, 38 BR_T1, 38 T_RRESCALER_HI, 42 value, 42 RadioFrequencyIdentificationMFRC522::TX_ASKbits, 42 char, 43 RadioFrequencyIdentificationMFRC522::TX_CONTR← OLbits, 43 char, 44 INV_TX1_RF_OF, 44 INV_TX2_RF_OF, 44 INV_TX2_RF_OF, 44 INV_TX_RF_OF, 44 IN		
char, 22 MSB_FIRST, 22 POL_M_FIN, 22 TX_WAIT_RF, 23 value, 23 RadioFrequencyIdentificationMFRC522::RF_CFGbits, 34 RX_GAIN, 34 value, 35 RadioFrequencyIdentificationMFRC522::RX_MOD← Ebits, 35 char, 35 RX_CRO_EN, 35 RX_MULTIPLE, 35 RX_NO_ERR, 36 RX_SPEED, 36 value, 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 RadioFrequencyIdentificationMFRC522::SERIAL_SP← EEDbits, 38 BR_T0, 38 BR_T1, 38 RadioFrequencyIdentificationMFRC522::Uid, 47 RadioFrequencyIdentificationMFRC522::Uid, 47 RadioFrequencyIdentificationMFRC522::SERIAL_SP← EEDbits, 38 BR_T1, 38 RadioFrequencyIdentificationMFRC522::Uid, 47 RadioFrequencyIdentificationMFRC522::Uid, 47 RadioFrequencyIdentificationMFRC522::Uid, 47		
MSB_FIRST, 22 POL_M_FIN, 22 TX_WAIT_RF, 23 value, 23 RadioFrequencyIdentificationMFRC522::RF_CFGbits, 34 RX_GAIN, 34 value, 35 RadioFrequencyIdentificationMFRC522::RX_MOD⊷ Lbits, 35 RX_GC_EN, 35 RX_MULTIPLE, 35 RX_MULTIPLE, 35 RX_MULTIPLE, 35 RX_NO_ERR, 36 RX_SPEED, 36 Value, 36 RX_WAIT, 36 UART_SEL, 36 Value, 37 RadioFrequencyIdentificationMFRC522::RX_THRES← HOLbits, 37 COLL_LEVEL, 37 char, 37 MIN_LEVEL, 37 value, 38 RadioFrequencyIdentificationMFRC522::SERIAL_SP← EEDbits, 38 BR_T0, 38 BR_T0, 38 BR_T1, 38 RAGIoFrequencyIdentificationMFRC522::SERIAL_SP← EEDbits, 38 BR_T1, 38 RadioFrequencyIdentificationMFRC522::Uid, 47 RadioFrequencyIdentificationMFRC522::Uid, 47 RadioFrequencyIdentificationMFRC522::Uid, 47 RadioFrequencyIdentificationMFRC522::SERIAL_SP← LEDbits, 38 BR_T1, 38 RadioFrequencyIdentificationMFRC522::Uid, 47 RadioFrequencyIdentificationMFRC522::Uid, 47 RadioFrequencyIdentificationMFRC522::Uid, 47		
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_CRC_EN, 35 RX_NO_ERR, 36 RX_NO_ERR, 36 RX_SPEED, 36 value, 37 RadioFrequencyIdentificationMFRC522::RX_SELbits, 36 RX_WAIT, 36 UART_SEL, 36 value, 37 RadioFrequencyIdentificationMFRC522::RX_THRES→ HOLDbits, 37 COLL_LEVEL, 37 value, 38 RadioFrequencyIdentificationMFRC522::SERIAL_SP← EEDbits, 38 RadioFrequencyIdentificationMFRC522::SERIAL_SP← EEDbits, 38 RadioFrequencyIdentificationMFRC522::SERIAL_SP← EEDbits, 38 RadioFrequencyIdentificationMFRC522::SERIAL_SP← EEDbits, 38 RAT, 0, 38 RAT, 38 RADIOREM		
TX_WAIT_RF, 23 value, 23 RadioFrequencyldentificationMFRC522::RF_CFGbits, 34 char, 34 RX_GAIN, 34 value, 35 RadioFrequencyldentificationMFRC522::RX_MOD← Ebits, 35 char, 35 RX_CRC_EN, 35 RX_MULTIPLE, 35 RX_NO_ERR, 36 RX_SPEED, 36 value, 36 RX_WAIT, 36 UART_SEL, 36 value, 37 RadioFrequencyldentificationMFRC522::RX_THRES← HOLDbits, 37 COLL_LEVEL, 37 value, 38 RadioFrequencyldentificationMFRC522::SERIAL_SP← EEDbits, 38 RAGIOFrequencyldentificationMFRC522::SERIAL_SP← EEDbits, 38 RAGIOFrequencyldentificationMFRC522::SERIAL_SP← EEDbits, 38 RAGIOFrequencyldentificationMFRC522::SERIAL_SP← EEDbits, 38 BR_T0, 38 BR_T1, 38 char, 43 FORCE_100_ASK, 43 value, 43 RadioFrequencyldentificationMFRC522::TX_CONTR← OLbits, 43 Char, 44 INV_TX1_RF_OFF, 44 INV_TX2_RF_OFF, 44 INV_TX2_RF_OFF, 44 INV_TX_RF_OFF, 44 INV_TX_RF_OFF, 44 INV_TX_RF_OFF, 44 INV_TX_RF_OFF, 44 INV_TX_RF_OF,		
value, 23 RadioFrequencyIdentificationMFRC522::RF_CFGbits,		
RadioFrequencyIdentificationMFRC522::RF_CFGbits, 34		
RadioFrequencyIdentificationMFRC522::TX_CONTR← char, 34 RX_GAIN, 34 value, 35 RadioFrequencyIdentificationMFRC522::RX_MOD← Ebits, 35 char, 35 RX_GRC_EN, 35 RX_MULTIPLE, 35 RX_MULTIPLE, 35 RX_NO_ERR, 36 RX_SPEED, 36 value, 36 RAZ_WAIT, 36 UART_SEL, 36 value, 37 RadioFrequencyIdentificationMFRC522::RX_THRES← HOLDbits, 37 COLL_LEVEL, 37 value, 38 RadioFrequencyIdentificationMFRC522::SERIAL_SP← EEDbits, 38 RAGIOFREQUENCYIdentificationMFRC522::SERIAL_SP← EEDbits, 38 RR_T0, 38 RR_T1, 38 RadioFrequencyIdentificationMFRC522::Uid, 47 RadioFrequencyIdentificationMFRC522::Uid, 47 RadioFrequencyIdentificationMFRC522::Uid, 47 RadioFrequencyIdentificationMFRC522::Uid, 47 RadioFrequencyIdentificationMFRC522::Uid, 47		
char, 34		
RX_GAIN, 34 value, 35 RadioFrequencyIdentificationMFRC522::RX_MOD← Ebits, 35 Char, 35 RX_CRC_EN, 35 RX_MULTIPLE, 35 RX_MODERR, 36 RX_SPEED, 36 value, 36 RX_MAIT, 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_T0, 38 BR_T1, 38 Char, 44 INV_TX1_RF_OFF, 44 INV_TX2_RF_ON, 44 INV_TX_RF_OFF, 44 INV_TX_RF_OF, 44 INV_TX_RF_OF, 44 INV_TX_RF_OF, 44 INV_T		
value, 35 RadioFrequencyIdentificationMFRC522::RX_MOD←		
RadioFrequencyIdentificationMFRC522::RX_MOD→		
Ebits, 35		
char, 35 INV_TX2_RF_ON, 44 RX_CRC_EN, 35 INV_TX_RF_OFF, 44 RX_MULTIPLE, 35 INV_TX_RF_ON, 44 RX_NO_ERR, 36 TX1_RF_EN, 44 RX_SPEED, 36 TX2_CW, 44 value, 36 TX2_RF_EN, 44 RadioFrequencyIdentificationMFRC522::RX_SELbits, 36 TX_RF_EN, 45 UART_SEL, 36 45 value, 37 RadioFrequencyIdentificationMFRC522::RX_THRES→ 6hQLbits, 37 INV_MOD, 45 TCOLL_LEVEL, 37 TX_CRC_EN, 46 TX_SPEED, 46 value, 46 RadioFrequencyIdentificationMFRC522::SERIAL_SP→ 6had RadioFrequencyIdentificationMFRC522::TX_SELbits, 46 RadioFrequencyIdentificationMFRC522::SERIAL_SP→ 6had Char, 46 EEDbits, 38 MF_OUT_SEL, 46 BR_T0, 38 Value, 47 BR_T1, 38 RadioFrequencyIdentificationMFRC522::Uid, 47		
RX_CRC_EN, 35		
RX_MULTIPLE, 35	•	
RX_NO_ERR, 36 RX_SPEED, 36 Value, 36 RX_WAIT, 36 RX_WAIT, 36 Value, 37 RadioFrequencyIdentificationMFRC522::RX_SELbits, HOLDbits, 37 COLL_LEVEL, 37 Char, 37 MIN_LEVEL, 37 Value, 38 RadioFrequencyIdentificationMFRC522::SERIAL_SP EEDbits, 38 BR_T0, 38 BR_T1, 38 TX1_RF_EN, 44 TX2_CW, 44 TX2_RF_EN, 44 TX2_RF_EN, 45 TX_RF_EN, 44 TX2_RF_EN, 44 TX2_RF_EN, 44 TX2_RF_EN, 45 TX_RF_EN, 44 TX_RF_EN, 44 TX_RE_EN, 45 TX_RF_EN, 45 TX_RE_EN, 46 TX_RE_EN, 46 TX_RET_EN, 45 TX_RET_	:	
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 TX2_CW, 44 TX2_RF_EN, 44 TX_RF_EN, 45 RadioFrequencyIdentificationMFRC522::TX_MODEbits, TX_RF_EN, 45 RadioFrequencyIdentificationMFRC522::TX_MODEbits, TX_RF_EN, 45 RadioFrequencyIdentificationMFRC522::TX_MODEbits, 45		
value, 36 RadioFrequencyIdentificationMFRC522::RX_SELbits,		
RadioFrequencyIdentificationMFRC522::RX_SELbits,		
RX_WAIT, 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 RadioFrequencyIdentificationMFRC522::Uid, 47 RadioFrequencyIdentificationMFRC522::TX_SELbits, 45 char, 45 INV_MOD, 45 TX_SPEED, 46 value, 46 RadioFrequencyIdentificationMFRC522::TX_SELbits, 46 char, 46 char, 46 value, 47 RadioFrequencyIdentificationMFRC522::Uid, 47		
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 RadioFrequencyIdentificationMFRC522::Uid, 47 RadioFrequencyIdentificationMFRC522::Uid, 47 RadioFrequencyIdentificationMFRC522::Uid, 47	· · ·	
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 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		
value, 37 RadioFrequencyIdentificationMFRC522::RX_THRES HOLDbits, 37 COLL_LEVEL, 37 Char, 45 INV_MOD, 45 TX_CRC_EN, 46 TX_SPEED, 46 value, 46 RadioFrequencyIdentificationMFRC522::TX_SELbits, value, 38 RadioFrequencyIdentificationMFRC522::SERIAL_SP← EEDbits, 38 BR_T0, 38 BR_T1, 38 Char, 45 INV_MOD, 45 TX_CRC_EN, 46 value, 46 RadioFrequencyIdentificationMFRC522::TX_SELbits, 46 char, 46 char, 46 MF_OUT_SEL, 46 value, 47 RadioFrequencyIdentificationMFRC522::Uid, 47	-	
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 INV_MOD, 45 TX_CRC_EN, 46 TX_SPEED, 46 value, 46 RadioFrequencyIdentificationMFRC522::TX_SELbits, 46 char, 46 char, 46 walue, 47 RadioFrequencyIdentificationMFRC522::Uid, 47		char. 45
HOLDbits, 37 COLL_LEVEL, 37 char, 37 MIN_LEVEL, 37 value, 38 RadioFrequencyIdentificationMFRC522::SERIAL_SP← EEDbits, 38 BR_T0, 38 BR_T1, 38 TX_CRC_EN, 46 TX_SPEED, 46 value, 46 RadioFrequencyIdentificationMFRC522::TX_SELbits, 46 char, 46 MF_OUT_SEL, 46 value, 47 RadioFrequencyIdentificationMFRC522::Uid, 47		
COLL_LEVEL, 37	· · · · · · · · · · · · · · · · · · ·	
char, 37 value, 46 MIN_LEVEL, 37 value, 38 RadioFrequencyIdentificationMFRC522::TX_SELbits, 46 RadioFrequencyIdentificationMFRC522::SERIAL_SP← EEDbits, 38 BR_T0, 38 BR_T1, 38 value, 46 RadioFrequencyIdentificationMFRC522::TX_SELbits, 46 MF_OUT_SEL, 46 value, 47 RadioFrequencyIdentificationMFRC522::Uid, 47		
MIN_LEVEL, 37 value, 38 RadioFrequencyIdentificationMFRC522::TX_SELbits, 46 Char, 46 EEDbits, 38 BR_T0, 38 BR_T1, 38 RadioFrequencyIdentificationMFRC522::SERIAL_SP← Value, 47 RadioFrequencyIdentificationMFRC522::Uid, 47		
value, 38 RadioFrequencyIdentificationMFRC522::SERIAL_SP← EEDbits, 38 BR_T0, 38 BR_T1, 38 46 char, 46 MF_OUT_SEL, 46 value, 47 RadioFrequencyIdentificationMFRC522::Uid, 47		
RadioFrequencyIdentificationMFRC522::SERIAL_SP EEDbits, 38 BR_T0, 38 BR_T1, 38 Char, 46 MF_OUT_SEL, 46 value, 47 RadioFrequencyIdentificationMFRC522::Uid, 47		
BR_T0, 38 MF_OUT_SEL, 46 value, 47 BR_T1, 38 RadioFrequencyIdentificationMFRC522::Uid, 47		
BR_T0, 38 value, 47 BR_T1, 38 RadioFrequencyIdentificationMFRC522::Uid, 47	· · · · · —	
BR_T1, 38 RadioFrequencyIdentificationMFRC522::Uid, 47		
		sak, 47

size, 47 uid, 47	RadioFrequencyIdentificationMFRC522::T_MO ← DEbits, 42
RadioFrequencyIdentificationMFRC522::VERSIONbits,	T_COUNTER_VAL_HIGH
48	RadioFrequencyIdentificationMFRC522, 29
CHIPTYPE, 48	T_COUNTER_VAL_LOW
VERSION, 48	RadioFrequencyIdentificationMFRC522, 29
value, 48	T_GATED
RadioFrequencyIdentificationMFRC522::WATER_LE ← VELbits, 48	RadioFrequencyIdentificationMFRC522::T_MO ← DEbits, 42
char, 49	T_MODE
value, 49 WATER_LEVEL, 49	RadioFrequencyIdentificationMFRC522, 29
readRegisterBlock	T_PRESCAL_EVEN
RadioFrequencyIdentificationMFRC522, 32	RadioFrequencyIdentificationMFRC522::DEMO← Dbits, 13
Register	T PRESCALER HI
RadioFrequencyIdentificationMFRC522, 28	RadioFrequencyIdentificationMFRC522::T_MO↔
resetPin	DEbits, 42
RadioFrequencyIdentificationMFRC522, 34	T_PRESCALER_LOW
	RadioFrequencyIdentificationMFRC522, 29
SERIAL_SPEED	T RELOAD HIGH
RadioFrequencyIdentificationMFRC522, 29	RadioFrequencyIdentificationMFRC522, 29
SET1	T_RELOAD_LOW
RadioFrequencyIdentificationMFRC522::COM_I ←	RadioFrequencyIdentificationMFRC522, 29
RQbits, 9	T_RUNNING
SET2	RadioFrequencyIdentificationMFRC522::STATU←
RadioFrequencyIdentificationMFRC522::DIV_IR ←	S1bits, 40
Qbits, 16 SOFT_RESET	T_START_NOW
RadioFrequencyIdentificationMFRC522, 27	$RadioF requency Identification MFRC 522 :: CONT \leftarrow$
START SEND	ROLbits, 12
RadioFrequencyIdentificationMFRC522::BIT_FR←	T_STOP_NOW
AMINGbits, 6	RadioFrequencyIdentificationMFRC522::CONT ←
STATUS1	ROLbits, 12
RadioFrequencyIdentificationMFRC522, 28	TAU_RCV
STATUS2	RadioFrequencyIdentificationMFRC522::DEMO ←
RadioFrequencyIdentificationMFRC522, 28	Dbits, 14
sak	TAU_SYNC RadioFrequencyIdentificationMFRC522::DEMO↔
RadioFrequencyIdentificationMFRC522::Uid, 47	Dbits, 14
sendCommand	TEMP ERR
RadioFrequencyIdentificationMFRC522, 32	RadioFrequencyIdentificationMFRC522::ERRO ←
setAntennaOff	Rbits, 17
RadioFrequencyIdentificationMFRC522, 32	TEMP_SENS_CLEAR
setAntennaOn	RadioFrequencyIdentificationMFRC522::STATU
RadioFrequencyIdentificationMFRC522, 33	S2bits, 41
setWaterLevel	TEST ADC
RadioFrequencyIdentificationMFRC522, 33 size	RadioFrequencyIdentificationMFRC522, 29
RadioFrequencyIdentificationMFRC522::Uid, 47	TEST_BUS
softReset	RadioFrequencyIdentificationMFRC522, 29
RadioFrequencyIdentificationMFRC522, 33	TEST_DAC1
startTimer	RadioFrequencyIdentificationMFRC522, 29
RadioFrequencyIdentificationMFRC522, 33	TEST_DAC2
stopTimer	RadioFrequencyIdentificationMFRC522, 29
RadioFrequencyIdentificationMFRC522, 33	TEST_PIN_EN
	RadioFrequencyIdentificationMFRC522, 29
T_AUTO_RESTART	TEST_PIN_VALUE
RadioFrequencyIdentificationMFRC522::T_MO ←	RadioFrequencyIdentificationMFRC522, 29
DEbits, 42	TEST_SEL1
T AUTO	RadioFrequencyIdentificationMFRC522, 29

TEST_SEL2	TX_WAIT
RadioFrequencyIdentificationMFRC522, 29	RadioFrequencyIdentificationMFRC522::MF_T ←
TIMEOUT_ERROR	Xbits, 20
RadioFrequencyIdentificationMFRC522, 27	
TIMER_I_EN	UART_SEL
RadioFrequencyIdentificationMFRC522::COM_I ←ENbits, 8	RadioFrequencyIdentificationMFRC522::RX_SE← Lbits, 36
TIMER_IRQ	uid
RadioFrequencyIdentificationMFRC522::COM_I ← RQbits, 10	RadioFrequencyIdentificationMFRC522::Uid, 47
TRANSCEIVE	V0 0
RadioFrequencyIdentificationMFRC522, 27 TRANSMIT	RadioFrequencyIdentificationMFRC522, 30 V1_0
RadioFrequencyIdentificationMFRC522, 27	RadioFrequencyIdentificationMFRC522, 30
TX1 RF EN	V2_0
RadioFrequencyIdentificationMFRC522::TX_CO←	RadioFrequencyIdentificationMFRC522, 30
NTROLbits, 44	VALUES_AFTER_COLL
TX2_CW	RadioFrequencyIdentificationMFRC522::COLLbits
RadioFrequencyIdentificationMFRC522::TX_CO←	7
NTROLbits, 44	VERSION
TX2_RF_EN	RadioFrequencyIdentificationMFRC522, 29
RadioFrequencyIdentificationMFRC522::TX_CO ←	RadioFrequencyIdentificationMFRC522::VERSI←
NTROLbits, 44	ONbits, 48
TX_ASK	value
RadioFrequencyIdentificationMFRC522, 29	RadioFrequencyIdentificationMFRC522::BIT_FR
TX_CONTROL_TX1_RF_EN	AMINGbits, 6
RadioFrequencyIdentificationMFRC522, 28	RadioFrequencyIdentificationMFRC522::COLLbits
TX_CONTROL_TX2_RF_EN	7
RadioFrequencyIdentificationMFRC522, 28	RadioFrequencyIdentificationMFRC522::COM_I ←
TX_CONTROL_TX_RF_EN	_ENbits, 8
RadioFrequencyIdentificationMFRC522, 28	RadioFrequencyIdentificationMFRC522::COM_I ←
TX_CONTROL	RQbits, 10
RadioFrequencyIdentificationMFRC522, 29	RadioFrequencyIdentificationMFRC522::COMM ←
TX_CRC_EN	ANDbits, 11
RadioFrequencyIdentificationMFRC522::TX_MO← DEbits, 46	RadioFrequencyIdentificationMFRC522::CONT ← ROLbits, 12
TX_I_EN	RadioFrequencyIdentificationMFRC522::CW_G←
RadioFrequencyIdentificationMFRC522::COM_I ←	S_Pbits, 13
_ENbits, 8	RadioFrequencyIdentificationMFRC522::DEMO ←
TX_IRQ	Dbits, 14
RadioFrequencyIdentificationMFRC522::COM_I ← RQbits, 10	RadioFrequencyIdentificationMFRC522::DIV_I_← ENbits, 15
TX_LAST_BITS	RadioFrequencyIdentificationMFRC522::DIV_IR ↔
RadioFrequencyIdentificationMFRC522::BIT_FR←	Qbits, 16
AMINGbits, 6	RadioFrequencyIdentificationMFRC522::ERRO ←
TX_MODE	Rbits, 17
RadioFrequencyIdentificationMFRC522, 29 TX_RF_EN	RadioFrequencyIdentificationMFRC522::FIFO_L EVELbits, 18
RadioFrequencyIdentificationMFRC522::TX_CO← NTROLbits, 45	RadioFrequencyIdentificationMFRC522::GS_← Nbits, 19
TX_SEL	RadioFrequencyIdentificationMFRC522::MF_R ←
RadioFrequencyIdentificationMFRC522, 29	Xbits, 20
TX_SPEED	$RadioF requency Identification MFRC 522 :: MF_T \leftarrow$
RadioFrequencyIdentificationMFRC522::TX_MO←	Xbits, 21
DEbits, 46	$RadioFrequencyIdentificationMFRC522::MOD_{\hookleftarrow}$
TX_WAIT_RF	GS_Pbits, 21
RadioFrequencyIdentificationMFRC522::MOD↔	RadioFrequencyIdentificationMFRC522::MOD ←
Ebits, 23	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
    RadioF requency Identification MFRC522::TX\_SE \leftarrow
         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
```