

Arduino RFID Driver

Generated by Doxygen 1.8.11

Contents

1	Hierarchical Index	2
1.1	Class Hierarchy	2
2	Class Index	3
2.1	Class List	3
3	File Index	5
3.1	File List	5
4	Class Documentation	6
4.1	ReaderMFRC522::BIT_FRAMINGbits Union Reference	6
4.1.1	Detailed Description	6
4.1.2	Member Data Documentation	6
4.2	ReaderMFRC522::COLLbits Union Reference	7
4.2.1	Detailed Description	7
4.2.2	Member Data Documentation	7
4.3	ReaderMFRC522::COM_I_ENbits Union Reference	8
4.3.1	Detailed Description	8
4.3.2	Member Data Documentation	9
4.4	ReaderMFRC522::COM_IRQbits Union Reference	9
4.4.1	Detailed Description	10
4.4.2	Member Data Documentation	10
4.5	ReaderMFRC522::COMMANDbits Union Reference	11
4.5.1	Detailed Description	11
4.5.2	Member Data Documentation	11
4.6	ReaderMFRC522::CONTROLbits Union Reference	12
4.6.1	Detailed Description	12
4.6.2	Member Data Documentation	12
4.7	ReaderMFRC522::CW_GS_Pbits Union Reference	13
4.7.1	Detailed Description	13

4.7.2	Member Data Documentation	13
4.8	ReaderMFRC522::DEMObits Union Reference	14
4.8.1	Detailed Description	14
4.8.2	Member Data Documentation	14
4.9	ReaderMFRC522::DIV_I_ENbits Union Reference	15
4.9.1	Detailed Description	15
4.9.2	Member Data Documentation	15
4.10	ReaderMFRC522::DIV_IRQbits Union Reference	16
4.10.1	Detailed Description	16
4.10.2	Member Data Documentation	16
4.11	ReaderMFRC522::ERRORbits Union Reference	17
4.11.1	Detailed Description	17
4.11.2	Member Data Documentation	18
4.12	ReaderMFRC522::FIFO_LEVELbits Union Reference	18
4.12.1	Detailed Description	19
4.12.2	Member Data Documentation	19
4.13	ReaderMFRC522::GS_Nbits Union Reference	19
4.13.1	Detailed Description	20
4.13.2	Member Data Documentation	20
4.14	ReaderMFRC522::MF_RXbits Union Reference	20
4.14.1	Detailed Description	20
4.14.2	Member Data Documentation	21
4.15	ReaderMFRC522::MF_TXbits Union Reference	21
4.15.1	Detailed Description	21
4.15.2	Member Data Documentation	21
4.16	ReaderMFRC522::MOD_GS_Pbits Union Reference	22
4.16.1	Detailed Description	22
4.16.2	Member Data Documentation	22
4.17	ReaderMFRC522::MODEbits Union Reference	23
4.17.1	Detailed Description	23

4.17.2 Member Data Documentation	23
4.18 Reader Class Reference	24
4.18.1 Detailed Description	25
4.18.2 Member Enumeration Documentation	26
4.18.3 Constructor & Destructor Documentation	26
4.18.4 Member Function Documentation	26
4.18.5 Member Data Documentation	29
4.19 ReaderMFRC522 Class Reference	29
4.19.1 Detailed Description	32
4.19.2 Member Enumeration Documentation	32
4.19.3 Constructor & Destructor Documentation	35
4.19.4 Member Function Documentation	35
4.19.5 Member Data Documentation	44
4.20 ReaderMFRC530 Class Reference	45
4.20.1 Detailed Description	46
4.20.2 Member Enumeration Documentation	46
4.20.3 Constructor & Destructor Documentation	47
4.20.4 Member Function Documentation	47
4.21 ReaderMFRC531 Class Reference	50
4.21.1 Detailed Description	51
4.21.2 Member Enumeration Documentation	52
4.21.3 Constructor & Destructor Documentation	52
4.21.4 Member Function Documentation	52
4.22 ReaderMFRC522::RF_CFGbits Union Reference	55
4.22.1 Detailed Description	55
4.22.2 Member Data Documentation	55
4.23 ReaderMFRC522::RX_MODEbits Union Reference	56
4.23.1 Detailed Description	56
4.23.2 Member Data Documentation	56
4.24 ReaderMFRC522::RX_SELbits Union Reference	57

4.24.1 Detailed Description	57
4.24.2 Member Data Documentation	57
4.25 ReaderMFRC522::RX_THRESHOLDbits Union Reference	58
4.25.1 Detailed Description	58
4.25.2 Member Data Documentation	58
4.26 ReaderMFRC522::SERIAL_SPEEDbits Union Reference	59
4.26.1 Detailed Description	59
4.26.2 Member Data Documentation	59
4.27 ReaderMFRC522::STATUS1bits Union Reference	60
4.27.1 Detailed Description	60
4.27.2 Member Data Documentation	60
4.28 ReaderMFRC522::STATUS2bits Union Reference	61
4.28.1 Detailed Description	61
4.28.2 Member Data Documentation	62
4.29 ReaderMFRC522::T_MODEbits Union Reference	62
4.29.1 Detailed Description	63
4.29.2 Member Data Documentation	63
4.30 Tag Class Reference	64
4.30.1 Detailed Description	66
4.30.2 Member Enumeration Documentation	66
4.30.3 Constructor & Destructor Documentation	67
4.30.4 Member Function Documentation	67
4.30.5 Member Data Documentation	71
4.31 TagMF0ICU1 Class Reference	72
4.31.1 Detailed Description	73
4.31.2 Member Enumeration Documentation	73
4.31.3 Constructor & Destructor Documentation	73
4.32 TagMF1ICS70 Class Reference	74
4.32.1 Detailed Description	75
4.32.2 Member Enumeration Documentation	75

4.32.3	Constructor & Destructor Documentation	75
4.32.4	Member Function Documentation	75
4.33	TagMF1S503x Class Reference	76
4.33.1	Detailed Description	77
4.33.2	Member Enumeration Documentation	77
4.33.3	Constructor & Destructor Documentation	77
4.33.4	Member Function Documentation	78
4.34	ReaderMFRC522::TX_ASKbits Union Reference	78
4.34.1	Detailed Description	79
4.34.2	Member Data Documentation	79
4.35	ReaderMFRC522::TX_CONTROLbits Union Reference	79
4.35.1	Detailed Description	80
4.35.2	Member Data Documentation	80
4.36	ReaderMFRC522::TX_MODEbits Union Reference	81
4.36.1	Detailed Description	81
4.36.2	Member Data Documentation	81
4.37	ReaderMFRC522::TX_SELbits Union Reference	82
4.37.1	Detailed Description	82
4.37.2	Member Data Documentation	82
4.38	Tag::Uid Struct Reference	83
4.38.1	Detailed Description	83
4.38.2	Member Data Documentation	83
4.39	ReaderMFRC522::VERSIONbits Union Reference	84
4.39.1	Detailed Description	84
4.39.2	Member Data Documentation	84
4.40	ReaderMFRC522::WATER_LEVELbits Union Reference	84
4.40.1	Detailed Description	85
4.40.2	Member Data Documentation	85

5	File Documentation	86
5.1	main.cpp File Reference	86
5.1.1	Function Documentation	86
5.2	main.cpp	87
5.3	Reader.cpp File Reference	88
5.4	Reader.cpp	89
5.5	Reader.h File Reference	89
5.5.1	Macro Definition Documentation	89
5.6	Reader.h	90
5.7	ReaderMFRC522.cpp File Reference	91
5.8	ReaderMFRC522.cpp	91
5.9	ReaderMFRC522.h File Reference	96
5.9.1	Macro Definition Documentation	97
5.9.2	Variable Documentation	98
5.10	ReaderMFRC522.h	99
5.11	ReaderMFRC530.cpp File Reference	113
5.12	ReaderMFRC530.cpp	113
5.13	ReaderMFRC530.h File Reference	114
5.14	ReaderMFRC530.h	114
5.15	ReaderMFRC531.cpp File Reference	116
5.16	ReaderMFRC531.cpp	116
5.17	ReaderMFRC531.h File Reference	116
5.18	ReaderMFRC531.h	117
5.19	Tag.cpp File Reference	118
5.20	Tag.cpp	119
5.21	Tag.h File Reference	123
5.21.1	Macro Definition Documentation	124
5.22	Tag.h	125
5.23	TagMF0ICU1.cpp File Reference	127
5.24	TagMF0ICU1.cpp	128
5.25	TagMF0ICU1.h File Reference	128
5.26	TagMF0ICU1.h	129
5.27	TagMF1ICS70.cpp File Reference	129
5.28	TagMF1ICS70.cpp	130
5.29	TagMF1ICS70.h File Reference	131
5.29.1	Macro Definition Documentation	132
5.30	TagMF1ICS70.h	133
5.31	TagMF1S503x.cpp File Reference	133
5.32	TagMF1S503x.cpp	134
5.33	TagMF1S503x.h File Reference	134
5.33.1	Macro Definition Documentation	135
5.34	TagMF1S503x.h	136

[Index](#)

137

1 Hierarchical Index

1.1 Class Hierarchy

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

ReaderMFRC522::BIT_FRAMINGbits	6
ReaderMFRC522::COLLbits	7
ReaderMFRC522::COM_I_ENbits	8
ReaderMFRC522::COM_IRQbits	9
ReaderMFRC522::COMMANDbits	11
ReaderMFRC522::CONTROLbits	12
ReaderMFRC522::CW_GS_Pbits	13
ReaderMFRC522::DEMODbits	14
ReaderMFRC522::DIV_I_ENbits	15
ReaderMFRC522::DIV_IRQbits	16
ReaderMFRC522::ERRORbits	17
ReaderMFRC522::FIFO_LEVELbits	18
ReaderMFRC522::GS_Nbits	19
ReaderMFRC522::MF_RXbits	20
ReaderMFRC522::MF_TXbits	21
ReaderMFRC522::MOD_GS_Pbits	22
ReaderMFRC522::MODEbits	23
Reader	24
ReaderMFRC522	29
ReaderMFRC530	45
ReaderMFRC531	50
RegisterBasedDevice	
ReaderMFRC522	29
ReaderMFRC530	45
ReaderMFRC531	50
ReaderMFRC522::RF_CFGbits	55

ReaderMFRC522::RX_MODEbits	56
ReaderMFRC522::RX_SELbits	57
ReaderMFRC522::RX_THRESHOLDbits	58
ReaderMFRC522::SERIAL_SPEEDbits	59
ReaderMFRC522::STATUS1bits	60
ReaderMFRC522::STATUS2bits	61
ReaderMFRC522::T_MODEbits	62
Tag	64
TagMF0ICU1	72
TagMF1ICS70	74
TagMF1S503x	76
ReaderMFRC522::TX_ASKbits	78
ReaderMFRC522::TX_CONTROLbits	79
ReaderMFRC522::TX_MODEbits	81
ReaderMFRC522::TX_SELbits	82
Tag::Uid	83
ReaderMFRC522::VERSIONbits	84
ReaderMFRC522::WATER_LEVELbits	84

2 Class Index

2.1 Class List

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

ReaderMFRC522::BIT_FRAMINGbits BIT_FRAMING register	6
ReaderMFRC522::COLLbits COLL register	7
ReaderMFRC522::COM_I_ENbits COM_I_EN register	8
ReaderMFRC522::COM_IRQbits COM_IRQ register	9
ReaderMFRC522::COMMANDbits COMMAND register (address 01h) Reset value: 20h bit allocation	11
ReaderMFRC522::CONTROLbits CONTROL register	12

ReaderMFRC522::CW_GS_Pbits	
CW_GS_P register	13
ReaderMFRC522::DEMODbits	
DEMOD register	14
ReaderMFRC522::DIV_I_ENbits	
DIV_I_EN register	15
ReaderMFRC522::DIV_IRQbits	
DIV_IRQ register	16
ReaderMFRC522::ERRORbits	
ERROR register	17
ReaderMFRC522::FIFO_LEVELbits	
FIFO_LEVEL register	18
ReaderMFRC522::GS_Nbits	
GS_N register	19
ReaderMFRC522::MF_RXbits	
MF_RX register	20
ReaderMFRC522::MF_TXbits	
MF_TX register	21
ReaderMFRC522::MOD_GS_Pbits	
MOD_GS_P register	22
ReaderMFRC522::MODEbits	
MODE register	23
Reader	24
ReaderMFRC522	29
ReaderMFRC530	
Arduino - Radio Frequency Identification MFRC530	45
ReaderMFRC531	
Arduino - Radio Frequency Identification MFRC531	50
ReaderMFRC522::RF_CFGbits	
RF_CFG register	55
ReaderMFRC522::RX_MODEbits	
RX_MODE register	56
ReaderMFRC522::RX_SELbits	
RX_SEL register	57
ReaderMFRC522::RX_THRESHOLDbits	
RX_THRESHOLD register	58
ReaderMFRC522::SERIAL_SPEEDbits	
SERIAL_SPEED register	59
ReaderMFRC522::STATUS1bits	
STATUS1 register	60

ReaderMFRC522::STATUS2bits	
STATUS2 register	61
ReaderMFRC522::T_MODEbits	
T_MODE register	62
Tag	64
TagMF0ICU1	
Arduino - Radio Frequency Identification MFRC522	72
TagMF1ICS70	74
TagMF1S503x	76
ReaderMFRC522::TX_ASKbits	
TX_ASK register	78
ReaderMFRC522::TX_CONTROLbits	
TX_CONTROL register	79
ReaderMFRC522::TX_MODEbits	
TX_MODE register	81
ReaderMFRC522::TX_SELbits	
TX_SEL register	82
Tag::Uid	83
ReaderMFRC522::VERSIONbits	
VERSION register	84
ReaderMFRC522::WATER_LEVELbits	
WATER_LEVEL register	84

3 File Index

3.1 File List

Here is a list of all files with brief descriptions:

main.cpp	86
Reader.cpp	88
Reader.h	89
ReaderMFRC522.cpp	91
ReaderMFRC522.h	96
ReaderMFRC530.cpp	113
ReaderMFRC530.h	114
ReaderMFRC531.cpp	116
ReaderMFRC531.h	116

Tag.cpp	118
Tag.h	123
TagMF0ICU1.cpp	127
TagMF0ICU1.h	128
TagMF1ICS70.cpp	129
TagMF1ICS70.h	131
TagMF1S503x.cpp	133
TagMF1S503x.h	134

4 Class Documentation

4.1 ReaderMFRC522::BIT_FRAMINGbits Union Reference

```
#include <ReaderMFRC522.h>
```

Public Attributes

- struct {
 - unsigned [char TX_LAST_BITS](#):3
 - unsigned [char](#):1
 - unsigned [char RX_ALIGN](#):3
 - unsigned [char START_SEND](#):1
- unsigned [char value](#)

4.1.1 Detailed Description

BIT_FRAMING register.

Miscellaneous control bits.

Definition at line [671](#) of file [ReaderMFRC522.h](#).

4.1.2 Member Data Documentation

4.1.2.1 struct { ... }

4.1.2.2 unsigned ReaderMFRC522::BIT_FRAMINGbits::char

Definition at line [680](#) of file [ReaderMFRC522.h](#).

4.1.2.3 unsigned char ReaderMFRC522::BIT_FRAMINGbits::RX_ALIGN

Definition at line 688 of file [ReaderMFRC522.h](#).

4.1.2.4 unsigned char ReaderMFRC522::BIT_FRAMINGbits::START_SEND

Definition at line 691 of file [ReaderMFRC522.h](#).

4.1.2.5 unsigned char ReaderMFRC522::BIT_FRAMINGbits::TX_LAST_BITS

Definition at line 677 of file [ReaderMFRC522.h](#).

4.1.2.6 unsigned char ReaderMFRC522::BIT_FRAMINGbits::value

Definition at line 693 of file [ReaderMFRC522.h](#).

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

- [ReaderMFRC522.h](#)

4.2 ReaderMFRC522::COLLbits Union Reference

```
#include <ReaderMFRC522.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 701 of file [ReaderMFRC522.h](#).

4.2.2 Member Data Documentation

4.2.2.1 struct { ... }

4.2.2.2 unsigned ReaderMFRC522::COLLbits::char

Definition at line 717 of file [ReaderMFRC522.h](#).

4.2.2.3 unsigned char ReaderMFRC522::COLLbits::COLL_POS

Definition at line 711 of file [ReaderMFRC522.h](#).

4.2.2.4 unsigned char ReaderMFRC522::COLLbits::COLL_POS_NOT_VALID

Definition at line 714 of file [ReaderMFRC522.h](#).

4.2.2.5 unsigned char ReaderMFRC522::COLLbits::value

Definition at line 722 of file [ReaderMFRC522.h](#).

4.2.2.6 unsigned char ReaderMFRC522::COLLbits::VALUES_AFTER_COLL

Definition at line 720 of file [ReaderMFRC522.h](#).

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

- [ReaderMFRC522.h](#)

4.3 ReaderMFRC522::COM_I_ENbits Union Reference

```
#include <ReaderMFRC522.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 325 of file [ReaderMFRC522.h](#).

4.3.2 Member Data Documentation

4.3.2.1 struct { ... }

4.3.2.2 unsigned char ReaderMFRC522::COM_I_ENbits::ERR_I_EN

Definition at line 333 of file [ReaderMFRC522.h](#).

4.3.2.3 unsigned char ReaderMFRC522::COM_I_ENbits::HI_ALERT_I_EN

Definition at line 339 of file [ReaderMFRC522.h](#).

4.3.2.4 unsigned char ReaderMFRC522::COM_I_ENbits::I_RQ_INV

Definition at line 353 of file [ReaderMFRC522.h](#).

4.3.2.5 unsigned char ReaderMFRC522::COM_I_ENbits::IDLE_I_EN

Definition at line 342 of file [ReaderMFRC522.h](#).

4.3.2.6 unsigned char ReaderMFRC522::COM_I_ENbits::LO_ALERT_I_EN

Definition at line 336 of file [ReaderMFRC522.h](#).

4.3.2.7 unsigned char ReaderMFRC522::COM_I_ENbits::RX_I_EN

Definition at line 345 of file [ReaderMFRC522.h](#).

4.3.2.8 unsigned char ReaderMFRC522::COM_I_ENbits::TIMER_I_EN

Definition at line 330 of file [ReaderMFRC522.h](#).

4.3.2.9 unsigned char ReaderMFRC522::COM_I_ENbits::TX_I_EN

Definition at line 348 of file [ReaderMFRC522.h](#).

4.3.2.10 unsigned char ReaderMFRC522::COM_I_ENbits::value

Definition at line 355 of file [ReaderMFRC522.h](#).

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

- [ReaderMFRC522.h](#)

4.4 ReaderMFRC522::COM_IRQbits Union Reference

```
#include <ReaderMFRC522.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 [394](#) of file [ReaderMFRC522.h](#).

4.4.2 Member Data Documentation

4.4.2.1 struct { ... }

4.4.2.2 unsigned char ReaderMFRC522::COM_IRQbits::ERR_IRQ

Definition at line [402](#) of file [ReaderMFRC522.h](#).

4.4.2.3 unsigned char ReaderMFRC522::COM_IRQbits::HI_ALERT_IRQ

Definition at line [412](#) of file [ReaderMFRC522.h](#).

4.4.2.4 unsigned char ReaderMFRC522::COM_IRQbits::IDLE_IRQ

Definition at line [418](#) of file [ReaderMFRC522.h](#).

4.4.2.5 unsigned char ReaderMFRC522::COM_IRQbits::LO_ALERT_IRQ

Definition at line [407](#) of file [ReaderMFRC522.h](#).

4.4.2.6 unsigned char ReaderMFRC522::COM_IRQbits::RX_IRQ

Definition at line [423](#) of file [ReaderMFRC522.h](#).

4.4.2.7 unsigned char ReaderMFRC522::COM_IRQbits::SET1

Definition at line [430](#) of file [ReaderMFRC522.h](#).

4.4.2.8 unsigned char ReaderMFRC522::COM_IRQbits::TIMER_IRQ

Definition at line 399 of file [ReaderMFRC522.h](#).

4.4.2.9 unsigned char ReaderMFRC522::COM_IRQbits::TX_IRQ

Definition at line 426 of file [ReaderMFRC522.h](#).

4.4.2.10 unsigned char ReaderMFRC522::COM_IRQbits::value

Definition at line 432 of file [ReaderMFRC522.h](#).

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

- [ReaderMFRC522.h](#)

4.5 ReaderMFRC522::COMMANDbits Union Reference

```
#include <ReaderMFRC522.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 298 of file [ReaderMFRC522.h](#).

4.5.2 Member Data Documentation

4.5.2.1 struct { ... }

4.5.2.2 unsigned ReaderMFRC522::COMMANDbits::char

Definition at line 315 of file [ReaderMFRC522.h](#).

4.5.2.3 unsigned char ReaderMFRC522::COMMANDbits::COMMAND

Definition at line 304 of file [ReaderMFRC522.h](#).

4.5.2.4 unsigned char ReaderMFRC522::COMMANDbits::POWER_DOWN

Definition at line 309 of file [ReaderMFRC522.h](#).

4.5.2.5 unsigned char ReaderMFRC522::COMMANDbits::RCV_OFF

Definition at line 312 of file [ReaderMFRC522.h](#).

4.5.2.6 unsigned char ReaderMFRC522::COMMANDbits::value

Definition at line 317 of file [ReaderMFRC522.h](#).

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

- [ReaderMFRC522.h](#)

4.6 ReaderMFRC522::CONTROLbits Union Reference

```
#include <ReaderMFRC522.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 645 of file [ReaderMFRC522.h](#).

4.6.2 Member Data Documentation

4.6.2.1 struct { ... }

4.6.2.2 unsigned ReaderMFRC522::CONTROLbits::char

Definition at line 653 of file [ReaderMFRC522.h](#).

4.6.2.3 unsigned char ReaderMFRC522::CONTROLbits::RX_LAST_BITS

Definition at line 650 of file [ReaderMFRC522.h](#).

4.6.2.4 unsigned char ReaderMFRC522::CONTROLbits::T_START_NOW

Definition at line 657 of file [ReaderMFRC522.h](#).

4.6.2.5 unsigned char ReaderMFRC522::CONTROLbits::T_STOP_NOW

Definition at line 661 of file [ReaderMFRC522.h](#).

4.6.2.6 unsigned char ReaderMFRC522::CONTROLbits::value

Definition at line 663 of file [ReaderMFRC522.h](#).

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

- [ReaderMFRC522.h](#)

4.7 ReaderMFRC522::CW_GS_Pbits Union Reference

```
#include <ReaderMFRC522.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 1143 of file [ReaderMFRC522.h](#).

4.7.2 Member Data Documentation

4.7.2.1 struct { ... }

4.7.2.2 unsigned ReaderMFRC522::CW_GS_Pbits::char

Definition at line 1152 of file [ReaderMFRC522.h](#).

4.7.2.3 unsigned char ReaderMFRC522::CW_GS_Pbits::CW_GS_P

Definition at line 1149 of file [ReaderMFRC522.h](#).

4.7.2.4 unsigned char ReaderMFRC522::CW_GS_Pbits::value

Definition at line 1154 of file [ReaderMFRC522.h](#).

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

- [ReaderMFRC522.h](#)

4.8 ReaderMFRC522::DEMObits Union Reference

```
#include <ReaderMFRC522.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

DEMObits register.

Defines demodulator settings.

Definition at line 998 of file [ReaderMFRC522.h](#).

4.8.2 Member Data Documentation

4.8.2.1 struct { ... }

4.8.2.2 unsigned char ReaderMFRC522::DEMObits::ADD_IQ

Definition at line 1023 of file [ReaderMFRC522.h](#).

4.8.2.3 unsigned char ReaderMFRC522::DEMObits::FIX_IQ

Definition at line 1017 of file [ReaderMFRC522.h](#).

4.8.2.4 unsigned char ReaderMFRC522::DEMODbits::T_PRESCAL_EVEN

Definition at line 1013 of file [ReaderMFRC522.h](#).

4.8.2.5 unsigned char ReaderMFRC522::DEMODbits::TAU_RCV

Definition at line 1007 of file [ReaderMFRC522.h](#).

4.8.2.6 unsigned char ReaderMFRC522::DEMODbits::TAU_SYNC

Definition at line 1003 of file [ReaderMFRC522.h](#).

4.8.2.7 unsigned char ReaderMFRC522::DEMODbits::value

Definition at line 1025 of file [ReaderMFRC522.h](#).

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

- [ReaderMFRC522.h](#)

4.9 ReaderMFRC522::DIV_I_ENbits Union Reference

```
#include <ReaderMFRC522.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 363 of file [ReaderMFRC522.h](#).

4.9.2 Member Data Documentation

4.9.2.1 struct { ... }

4.9.2.2 unsigned ReaderMFRC522::DIV_I_ENbits::char

Definition at line 368 of file [ReaderMFRC522.h](#).

4.9.2.3 unsigned char ReaderMFRC522::DIV_I_ENbits::CRC_I_EN

Definition at line 371 of file [ReaderMFRC522.h](#).

4.9.2.4 unsigned char ReaderMFRC522::DIV_I_ENbits::IRQ_PUSH_PULL

Definition at line 384 of file [ReaderMFRC522.h](#).

4.9.2.5 unsigned char ReaderMFRC522::DIV_I_ENbits::MFIN_ACT_I_EN

Definition at line 377 of file [ReaderMFRC522.h](#).

4.9.2.6 unsigned char ReaderMFRC522::DIV_I_ENbits::value

Definition at line 386 of file [ReaderMFRC522.h](#).

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

- [ReaderMFRC522.h](#)

4.10 ReaderMFRC522::DIV_IRQbits Union Reference

```
#include <ReaderMFRC522.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 440 of file [ReaderMFRC522.h](#).

4.10.2 Member Data Documentation

4.10.2.1 struct { ... }

4.10.2.2 unsigned ReaderMFRC522::DIV_IRQbits::char

Definition at line 445 of file [ReaderMFRC522.h](#).

4.10.2.3 unsigned char ReaderMFRC522::DIV_IRQbits::CRC_IRQ

Definition at line 448 of file [ReaderMFRC522.h](#).

4.10.2.4 unsigned char ReaderMFRC522::DIV_IRQbits::MFIN_ACT_IRQ

Definition at line 454 of file [ReaderMFRC522.h](#).

4.10.2.5 unsigned char ReaderMFRC522::DIV_IRQbits::SET2

Definition at line 461 of file [ReaderMFRC522.h](#).

4.10.2.6 unsigned char ReaderMFRC522::DIV_IRQbits::value

Definition at line 463 of file [ReaderMFRC522.h](#).

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

- [ReaderMFRC522.h](#)

4.11 ReaderMFRC522::ERRORbits Union Reference

```
#include <ReaderMFRC522.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 471 of file [ReaderMFRC522.h](#).

4.11.2 Member Data Documentation

4.11.2.1 struct { ... }

4.11.2.2 unsigned char ReaderMFRC522::ERRORbits::BUFFER_OVFL

Definition at line 495 of file [ReaderMFRC522.h](#).

4.11.2.3 unsigned ReaderMFRC522::ERRORbits::char

Definition at line 498 of file [ReaderMFRC522.h](#).

4.11.2.4 unsigned char ReaderMFRC522::ERRORbits::COLL_ERR

Definition at line 491 of file [ReaderMFRC522.h](#).

4.11.2.5 unsigned char ReaderMFRC522::ERRORbits::CRC_ERR

Definition at line 486 of file [ReaderMFRC522.h](#).

4.11.2.6 unsigned char ReaderMFRC522::ERRORbits::PARITY_ERR

Definition at line 482 of file [ReaderMFRC522.h](#).

4.11.2.7 unsigned char ReaderMFRC522::ERRORbits::PROTOCOL_ERR

Definition at line 478 of file [ReaderMFRC522.h](#).

4.11.2.8 unsigned char ReaderMFRC522::ERRORbits::TEMP_ERR

Definition at line 501 of file [ReaderMFRC522.h](#).

4.11.2.9 unsigned char ReaderMFRC522::ERRORbits::value

Definition at line 508 of file [ReaderMFRC522.h](#).

4.11.2.10 unsigned char ReaderMFRC522::ERRORbits::WR_ERR

Definition at line 506 of file [ReaderMFRC522.h](#).

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

- [ReaderMFRC522.h](#)

4.12 ReaderMFRC522::FIFO_LEVELbits Union Reference

```
#include <ReaderMFRC522.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 603 of file [ReaderMFRC522.h](#).

4.12.2 Member Data Documentation

4.12.2.1 struct { ... }

4.12.2.2 unsigned char ReaderMFRC522::FIFO_LEVELbits::FIFO_LEVEL

Definition at line 609 of file [ReaderMFRC522.h](#).

4.12.2.3 unsigned char ReaderMFRC522::FIFO_LEVELbits::FLUSH_BUFFER

Definition at line 613 of file [ReaderMFRC522.h](#).

4.12.2.4 unsigned char ReaderMFRC522::FIFO_LEVELbits::value

Definition at line 615 of file [ReaderMFRC522.h](#).

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

- [ReaderMFRC522.h](#)

4.13 ReaderMFRC522::GS_Nbits Union Reference

```
#include <ReaderMFRC522.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 1120 of file [ReaderMFRC522.h](#).

4.13.2 Member Data Documentation

4.13.2.1 struct { ... }

4.13.2.2 unsigned char ReaderMFRC522::GS_Nbits::CW_GS_N

Definition at line 1133 of file [ReaderMFRC522.h](#).

4.13.2.3 unsigned char ReaderMFRC522::GS_Nbits::MOD_GS_N

Definition at line 1127 of file [ReaderMFRC522.h](#).

4.13.2.4 unsigned char ReaderMFRC522::GS_Nbits::value

Definition at line 1135 of file [ReaderMFRC522.h](#).

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

- [ReaderMFRC522.h](#)

4.14 ReaderMFRC522::MF_RXbits Union Reference

```
#include <ReaderMFRC522.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 1051 of file [ReaderMFRC522.h](#).

4.14.2 Member Data Documentation

4.14.2.1 struct { ... }

4.14.2.2 unsigned ReaderMFRC522::MF_RXbits::char

Definition at line 1056 of file [ReaderMFRC522.h](#).

4.14.2.3 unsigned char ReaderMFRC522::MF_RXbits::PARITY_DISABLE

Definition at line 1060 of file [ReaderMFRC522.h](#).

4.14.2.4 unsigned char ReaderMFRC522::MF_RXbits::value

Definition at line 1065 of file [ReaderMFRC522.h](#).

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

- [ReaderMFRC522.h](#)

4.15 ReaderMFRC522::MF_TXbits Union Reference

```
#include <ReaderMFRC522.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 1033 of file [ReaderMFRC522.h](#).

4.15.2 Member Data Documentation

4.15.2.1 struct { ... }

4.15.2.2 unsigned ReaderMFRC522::MF_TXbits::char

Definition at line 1041 of file [ReaderMFRC522.h](#).

4.15.2.3 unsigned char ReaderMFRC522::MF_TXbits::TX_WAIT

Definition at line 1038 of file [ReaderMFRC522.h](#).

4.15.2.4 unsigned char ReaderMFRC522::MF_TXbits::value

Definition at line 1043 of file [ReaderMFRC522.h](#).

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

- [ReaderMFRC522.h](#)

4.16 ReaderMFRC522::MOD_GS_Pbits Union Reference

```
#include <ReaderMFRC522.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 1162 of file [ReaderMFRC522.h](#).

4.16.2 Member Data Documentation

4.16.2.1 struct { ... }

4.16.2.2 unsigned ReaderMFRC522::MOD_GS_Pbits::char

Definition at line 1172 of file [ReaderMFRC522.h](#).

4.16.2.3 unsigned char ReaderMFRC522::MOD_GS_Pbits::MOD_GS_P

Definition at line 1169 of file [ReaderMFRC522.h](#).

4.16.2.4 unsigned char ReaderMFRC522::MOD_GS_Pbits::value

Definition at line 1174 of file [ReaderMFRC522.h](#).

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

- [ReaderMFRC522.h](#)

4.17 ReaderMFRC522::MODEbits Union Reference

```
#include <ReaderMFRC522.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 730 of file [ReaderMFRC522.h](#).

4.17.2 Member Data Documentation

4.17.2.1 struct { ... }

4.17.2.2 unsigned ReaderMFRC522::MODEbits::char

Definition at line 744 of file [ReaderMFRC522.h](#).

4.17.2.3 unsigned char ReaderMFRC522::MODEbits::CRC_PRESET

Definition at line 741 of file [ReaderMFRC522.h](#).

4.17.2.4 unsigned char ReaderMFRC522::MODEbits::MSB_FIRST

Definition at line 764 of file [ReaderMFRC522.h](#).

4.17.2.5 unsigned char ReaderMFRC522::MODEbits::POL_M_FIN

Definition at line 750 of file [ReaderMFRC522.h](#).

4.17.2.6 unsigned char ReaderMFRC522::MODEbits::TX_WAIT_RF

Definition at line 756 of file [ReaderMFRC522.h](#).

4.17.2.7 unsigned char ReaderMFRC522::MODEbits::value

Definition at line 766 of file [ReaderMFRC522.h](#).

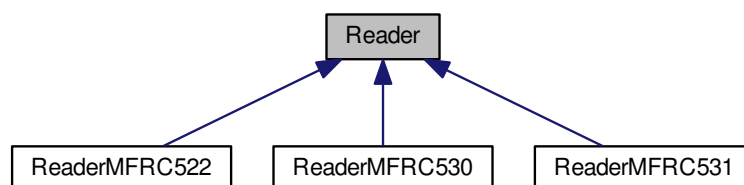
The documentation for this union was generated from the following file:

- [ReaderMFRC522.h](#)

4.18 Reader Class Reference

```
#include <Reader.h>
```

Inheritance diagram for Reader:



Public Types

- enum [Error](#) {
[NO_ERROR](#) = 0x00, [GENERAL_ERROR](#) = 0x01, [TIMEOUT_ERROR](#) = 0x02, [COMMUNICATION_ERROR](#) = 0x03,
[CRC_ERROR](#) = 0x04, [NACK](#) = 0x05, [COLLISION_ERROR](#) = 0x06 }

Public Member Functions

- [Reader](#) ()
- virtual [~Reader](#) ()
- virtual void [sendCommand](#) (unsigned char command)=0
- virtual void [softReset](#) ()=0
- virtual void [setAntennaOn](#) ()=0
- virtual void [setAntennaOff](#) ()=0
- virtual void [configureTimer](#) (unsigned int prescaler, unsigned int reload, bool autoStart, bool autoRestart)=0
- virtual void [startTimer](#) ()=0
- virtual void [stopTimer](#) ()=0
- virtual void [enableInterrupt](#) (unsigned int interrupt)=0
- virtual void [disableInterrupt](#) (unsigned int interrupt)=0
- virtual void [clearInterrupt](#) (unsigned int interrupt)=0
- virtual void [flushQueue](#) ()=0
- virtual void [setWaterLevel](#) (unsigned char level)=0
- virtual int [generateRandomId](#) (unsigned char *buf)=0
- virtual int [communicate](#) (unsigned char command, unsigned char *send, unsigned char *receive, unsigned char sendLen, bool checkCrc)=0
- virtual int [communicate](#) (unsigned char command, unsigned char *send, unsigned char *receive, unsigned char sendLen)=0
- virtual int [tranceive](#) (unsigned char *send, unsigned char *receive, unsigned char sendLen, bool checkCrc)=0
- virtual int [tranceive](#) (unsigned char *send, unsigned char *receive, unsigned char sendLen)=0
- virtual int [authenticate](#) (unsigned char *send)=0
- virtual void [turnOffEncryption](#) ()=0
- virtual unsigned int [calculateCrc](#) (unsigned char *buf, unsigned char len)=0
- virtual void [calculateCrc](#) (unsigned char *buf, unsigned char len, unsigned char *dst)=0
- virtual bool [waitForRegisterBits](#) (unsigned char reg, unsigned char mask, unsigned long timeout)=0
- virtual bool [waitForRegisterBits](#) (unsigned char reg, unsigned char mask)=0
- virtual bool [performSelfTest](#) ()=0
- virtual void [setBitFraming](#) (unsigned char rxAlign, unsigned char txLastBits)=0
- virtual unsigned char [getCollisionPosition](#) ()=0
- virtual void [setuptForAnticollision](#) ()=0
- unsigned char [getLastError](#) ()
- void [clearLastError](#) ()
- virtual bool [hasValidCrc](#) (unsigned char *buf, unsigned char len)=0

Protected Attributes

- [Error lastError](#)

4.18.1 Detailed Description

Definition at line 12 of file [Reader.h](#).

4.18.2 Member Enumeration Documentation

4.18.2.1 enum Reader::Error

Enumerator

NO_ERROR
GENERAL_ERROR
TIMEOUT_ERROR
COMMUNICATION_ERROR
CRC_ERROR
NACK
COLLISION_ERROR

Definition at line 16 of file [Reader.h](#).

4.18.3 Constructor & Destructor Documentation

4.18.3.1 Reader::Reader ()

Definition at line 3 of file [Reader.cpp](#).

4.18.3.2 Reader::~Reader () [virtual]

Definition at line 6 of file [Reader.cpp](#).

4.18.4 Member Function Documentation

4.18.4.1 virtual int Reader::authenticate (unsigned char * *send*) [pure virtual]

Implemented in [ReaderMFRC522](#), [ReaderMFRC530](#), and [ReaderMFRC531](#).

4.18.4.2 virtual unsigned int Reader::calculateCrc (unsigned char * *buf*, unsigned char *len*) [pure virtual]

Implemented in [ReaderMFRC522](#), [ReaderMFRC530](#), and [ReaderMFRC531](#).

4.18.4.3 virtual void Reader::calculateCrc (unsigned char * *buf*, unsigned char *len*, unsigned char * *dst*) [pure virtual]

Implemented in [ReaderMFRC522](#), [ReaderMFRC530](#), and [ReaderMFRC531](#).

4.18.4.4 virtual void Reader::clearInterrupt (unsigned int *interrupt*) [pure virtual]

Implemented in [ReaderMFRC522](#), [ReaderMFRC530](#), and [ReaderMFRC531](#).

4.18.4.5 void Reader::clearLastError ()

Definition at line 13 of file [Reader.cpp](#).

4.18.4.6 `virtual int Reader::communicate (unsigned char command, unsigned char * send, unsigned char * receive, unsigned char sendLen, bool checkCrc)` [pure virtual]

Implemented in [ReaderMFRC522](#), [ReaderMFRC530](#), and [ReaderMFRC531](#).

4.18.4.7 `virtual int Reader::communicate (unsigned char command, unsigned char * send, unsigned char * receive, unsigned char sendLen)` [pure virtual]

Implemented in [ReaderMFRC522](#), [ReaderMFRC530](#), and [ReaderMFRC531](#).

4.18.4.8 `virtual void Reader::configureTimer (unsigned int prescaler, unsigned int reload, bool autoStart, bool autoRestart)` [pure virtual]

Implemented in [ReaderMFRC522](#), [ReaderMFRC530](#), and [ReaderMFRC531](#).

4.18.4.9 `virtual void Reader::disableInterrupt (unsigned int interrupt)` [pure virtual]

Implemented in [ReaderMFRC522](#), [ReaderMFRC530](#), and [ReaderMFRC531](#).

4.18.4.10 `virtual void Reader::enableInterrupt (unsigned int interrupt)` [pure virtual]

Implemented in [ReaderMFRC522](#), [ReaderMFRC530](#), and [ReaderMFRC531](#).

4.18.4.11 `virtual void Reader::flushQueue ()` [pure virtual]

Implemented in [ReaderMFRC522](#), [ReaderMFRC530](#), and [ReaderMFRC531](#).

4.18.4.12 `virtual int Reader::generateRandomId (unsigned char * buf)` [pure virtual]

Implemented in [ReaderMFRC522](#), [ReaderMFRC530](#), and [ReaderMFRC531](#).

4.18.4.13 `virtual unsigned char Reader::getCollisionPosition ()` [pure virtual]

Implemented in [ReaderMFRC522](#), [ReaderMFRC530](#), and [ReaderMFRC531](#).

4.18.4.14 `unsigned char Reader::getLastError ()`

Definition at line 9 of file [Reader.cpp](#).

4.18.4.15 `virtual bool Reader::hasValidCrc (unsigned char * buf, unsigned char len)` [pure virtual]

Implemented in [ReaderMFRC522](#), [ReaderMFRC530](#), and [ReaderMFRC531](#).

4.18.4.16 `virtual bool Reader::performSelfTest ()` [pure virtual]

Implemented in [ReaderMFRC522](#), [ReaderMFRC530](#), and [ReaderMFRC531](#).

4.18.4.17 `virtual void Reader::sendCommand (unsigned char command)` [inline], [pure virtual]

Implemented in [ReaderMFRC522](#), [ReaderMFRC530](#), and [ReaderMFRC531](#).

4.18.4.18 `virtual void Reader::setAntennaOff () [pure virtual]`

Implemented in [ReaderMFRC522](#), [ReaderMFRC530](#), and [ReaderMFRC531](#).

4.18.4.19 `virtual void Reader::setAntennaOn () [pure virtual]`

Implemented in [ReaderMFRC522](#), [ReaderMFRC530](#), and [ReaderMFRC531](#).

4.18.4.20 `virtual void Reader::setBitFraming (unsigned char rxAlign, unsigned char txLastBits) [pure virtual]`

Implemented in [ReaderMFRC522](#), [ReaderMFRC530](#), and [ReaderMFRC531](#).

4.18.4.21 `virtual void Reader::setuptForAnticollision () [pure virtual]`

Implemented in [ReaderMFRC522](#), [ReaderMFRC530](#), and [ReaderMFRC531](#).

4.18.4.22 `virtual void Reader::setWaterLevel (unsigned char level) [pure virtual]`

Implemented in [ReaderMFRC522](#), [ReaderMFRC530](#), and [ReaderMFRC531](#).

4.18.4.23 `virtual void Reader::softReset () [pure virtual]`

Implemented in [ReaderMFRC522](#), [ReaderMFRC530](#), and [ReaderMFRC531](#).

4.18.4.24 `virtual void Reader::startTimer () [pure virtual]`

Implemented in [ReaderMFRC522](#), [ReaderMFRC530](#), and [ReaderMFRC531](#).

4.18.4.25 `virtual void Reader::stopTimer () [pure virtual]`

Implemented in [ReaderMFRC522](#), [ReaderMFRC530](#), and [ReaderMFRC531](#).

4.18.4.26 `virtual int Reader::tranceive (unsigned char * send, unsigned char * receive, unsigned char sendLen, bool checkCrc) [pure virtual]`

Implemented in [ReaderMFRC522](#), [ReaderMFRC530](#), and [ReaderMFRC531](#).

4.18.4.27 `virtual int Reader::tranceive (unsigned char * send, unsigned char * receive, unsigned char sendLen) [pure virtual]`

Implemented in [ReaderMFRC522](#), [ReaderMFRC530](#), and [ReaderMFRC531](#).

4.18.4.28 `virtual void Reader::turnOffEncryption () [pure virtual]`

Implemented in [ReaderMFRC522](#), [ReaderMFRC530](#), and [ReaderMFRC531](#).

4.18.4.29 `virtual bool Reader::waitForRegisterBits (unsigned char reg, unsigned char mask, unsigned long timeout) [pure virtual]`

Implemented in [ReaderMFRC522](#), [ReaderMFRC530](#), and [ReaderMFRC531](#).

4.18.4.30 `virtual bool Reader::waitForRegisterBits (unsigned char reg, unsigned char mask)` [pure virtual]

Implemented in [ReaderMFRC522](#), [ReaderMFRC530](#), and [ReaderMFRC531](#).

4.18.5 Member Data Documentation

4.18.5.1 **Error** `Reader::lastError` [protected]

Definition at line 92 of file [Reader.h](#).

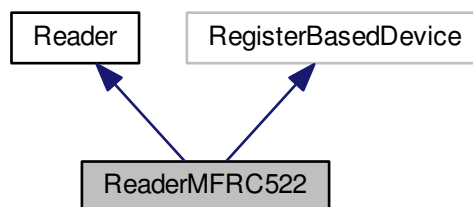
The documentation for this class was generated from the following files:

- [Reader.h](#)
- [Reader.cpp](#)

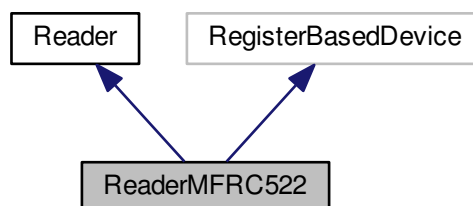
4.19 ReaderMFRC522 Class Reference

```
#include <ReaderMFRC522.h>
```

Inheritance diagram for ReaderMFRC522:



Collaboration diagram for ReaderMFRC522:



Classes

- union [BIT_FRAMINGbits](#)
- union [COLLbits](#)
- union [COM_I_ENbits](#)
- union [COM_IRQbits](#)
- union [COMMANDbits](#)
- union [CONTROLbits](#)
- union [CW_GS_Pbits](#)
- union [DEMODObits](#)
- 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](#)
- union [VERSIONbits](#)
- union [WATER_LEVELbits](#)

Public Types

- enum [Register](#) {
[COMMAND](#) = 0x01, [COM_I_EN](#) = 0x02, [DIV_I_EN](#) = 0x03, [COM_IRQ](#) = 0x04,
[DIV_IRQ](#) = 0x05, [ERROR](#) = 0x06, [STATUS1](#) = 0x07, [STATUS2](#) = 0x08,
[FIFO_DATA](#) = 0x09, [FIFO_LEVEL](#) = 0x0a, [WATER_LEVEL](#) = 0x0b, [CONTROL](#) = 0x0c,
[BIT_FRAMING](#) = 0x0d, [COLL](#) = 0x0e, [MODE](#) = 0x11, [TX_MODE](#) = 0x12,
[RX_MODE](#) = 0x13, [TX_CONTROL](#) = 0x14, [TX_ASK](#) = 0x15, [TX_SEL](#) = 0x16,
[RX_SEL](#) = 0x17, [RX_THRESHOLD](#) = 0x18, [DEMODO](#) = 0x19, [MF_TX](#) = 0x1c,
[MF_RX](#) = 0x1d, [SERIAL_SPEED](#) = 0x1f, [CRC_RESULT_HIGH](#) = 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, [TRANSCIVE](#) = 0x0c,
[MF_AUTHENT](#) = 0x0e, [SOFT_RESET](#) = 0x0f }

- enum `Mask` {
`TX_CONTROL_TX1_RF_EN` = 0x01, `TX_CONTROL_TX2_RF_EN` = 0x02, `TX_CONTROL_TX_RF_EN` =
`TX_CONTROL_TX1_RF_EN` | `TX_CONTROL_TX2_RF_EN`, `CONTROL_T_STOP_NOW` = 0x80,
`CONTROL_T_START_NOW` = 0x40, `COM_I_EN_INTERRUPT_EN` = 0x7f, `COM_IRQ_TIMER_IRQ` = 0x01,
`COM_IRQ_ERR_IRQ` = 0x02,
`COM_IRQ_LO_ALERT_IRQ` = 0x04, `COM_IRQ_HI_ALERT_IRQ` = 0x08, `COM_IRQ_IDLE_IRQ` = 0x10, `COM_IRQ_RX_IRQ` = 0x20,
`COM_IRQ_TX_IRQ` = 0x40, `COM_IRQ_ALL_IRQ` = 0x7f, `COM_IRQ_SET1` = 0x80, `DIV_I_EN_CRC_I_EN` =
0x04,
`DIV_I_EN_MFIN_ACT_I_EN` = 0x10, `DIV_I_EN_INTERRUPT_EN` = `DIV_I_EN_CRC_I_EN` | `DIV_I_EN_MFIN_ACT_I_EN`,
`DIV_IRQ_CRC_IRQ` = 0x04, `DIV_IRQ_MFIN_ACT_IRQ` = 0x10,
`DIV_IRQ_ALL_IRQ` = `DIV_IRQ_CRC_IRQ` | `DIV_IRQ_MFIN_ACT_IRQ`, `DIV_IRQ_SET2` = 0x80, `FIFO_LEVEL_FLUSH_BUFFER` = 0x80,
`FIFO_LEVEL_FIFO_LEVEL` = 0x7f,
`WATER_LEVEL_WATER_LEVEL` = 0x3f, `BIT_FRAMING_START_SEND` = 0x80, `AUTO_TEST_ENABLE` =
0x09, `COLL_VALUES_AFTER_COLL` = 0x80,
`STATUS2_MF_CRYPT01_ON` = 0x08 }
- enum `Interrupt` : unsigned int {
`NONE_IRQ` = 0x0000, `COM_TIMER_IRQ` = 0x0001, `COM_ERR_IRQ` = 0x0002, `COM_LO_ALERT_IRQ` =
0x0004,
`COM_HI_ALERT_IRQ` = 0x0008, `COM_IDLE_IRQ` = 0x0010, `COM_RX_IRQ` = 0x0020, `COM_TX_IRQ` =
0x0040,
`COM_ALL_IRQ` = 0x007f, `DIV_CRC_IRQ` = 0x0400, `DIV_MFIN_ACT_IRQ` = 0x1000, `DIV_ALL_IRQ` = `DIV_CRC_IRQ` | `DIV_MFIN_ACT_IRQ` }
- enum `Version` { `CLONE` = 0x88, `V0_0` = 0x90, `V1_0` = 0x91, `V2_0` = 0x92 }

Public Member Functions

- `ReaderMFRC522` (RegisterBasedDevice *device, unsigned char resetPin)
- virtual `~ReaderMFRC522` ()
- void `initialize` ()
- void `sendCommand` (unsigned char command)
- void `softReset` ()
- void `setAntennaOn` ()
- void `setAntennaOff` ()
- void `configureTimer` (unsigned int prescaler, unsigned int reload, bool autoStart, bool autoRestart)
- void `startTimer` ()
- void `stopTimer` ()
- void `enableInterrupt` (unsigned int interrupt)
- void `disableInterrupt` (unsigned int interrupt)
- void `clearInterrupt` (unsigned int interrupt)
- void `flushQueue` ()
- void `setWaterLevel` (unsigned char level)
- int `generateRandomId` (unsigned char *buf)
- int `communicate` (unsigned char command, unsigned char *send, unsigned char *receive, unsigned char sendLen, bool checkCrc)
- int `communicate` (unsigned char command, unsigned char *send, unsigned char *receive, unsigned char sendLen)
- int `tranceive` (unsigned char *send, unsigned char *receive, unsigned char sendLen, bool checkCrc)
- int `tranceive` (unsigned char *send, unsigned char *receive, unsigned char sendLen)
- int `authenticate` (unsigned char *send)
- unsigned int `calculateCrc` (unsigned char *buf, unsigned char len)
- void `calculateCrc` (unsigned char *buf, unsigned char len, unsigned char *dst)
- bool `waitForRegisterBits` (unsigned char reg, unsigned char mask, unsigned long timeout)
- bool `waitForRegisterBits` (unsigned char reg, unsigned char mask)
- `Version` `getVersion` ()

- bool [performSelfTest](#) ()
- void [setBitFraming](#) (unsigned char rxAlign, unsigned char txLastBits)
- unsigned char [getCollisionPosition](#) ()
- void [setuptForAnticollision](#) ()
- int [readRegisterBlock](#) (unsigned char reg, unsigned char *buf, unsigned char len)
- int [readRegisterBlock](#) (unsigned char reg, unsigned char *buf, unsigned char len, unsigned char rxAlign)
- unsigned char [writeRegisterBlock](#) (unsigned char reg, unsigned char *buf, unsigned char len)
- void [turnOffEncryption](#) ()
- bool [hasValidCrc](#) (unsigned char *buf, unsigned char len)

Private Attributes

- RegisterBasedDevice * [device](#)
- unsigned char [resetPin](#)

Static Private Attributes

- static const unsigned char [SAK](#) = 0x08
- static const unsigned char [ACK](#) = 0x0a

Additional Inherited Members

4.19.1 Detailed Description

Definition at line 49 of file [ReaderMFRC522.h](#).

4.19.2 Member Enumeration Documentation

4.19.2.1 enum ReaderMFRC522::Command

Enumerator

IDLE
MEM
GENERATE_RANDOM_ID
CALC_CRC
TRANSMIT
NO_CMD_CHANGE
RECEIVE
TRANSCIVE
MF_AUTHENT
SOFT_RESET

Definition at line 213 of file [ReaderMFRC522.h](#).

4.19.2.2 enum ReaderMFRC522::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 278 of file [ReaderMFRC522.h](#).

4.19.2.3 enum ReaderMFRC522::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  
COLL_VALUES_AFTER_COLL  
STATUS2_MF_CRYPT01_ON
```

Definition at line 246 of file [ReaderMFRC522.h](#).

4.19.2.4 enum ReaderMFRC522::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_HIGH
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 60 of file [ReaderMFRC522.h](#).

4.19.2.5 enum ReaderMFRC522::Version

Enumerator

CLONE
V0_0
V1_0
V2_0

Definition at line 1236 of file [ReaderMFRC522.h](#).

4.19.3 Constructor & Destructor Documentation

4.19.3.1 ReaderMFRC522::ReaderMFRC522 (RegisterBasedDevice * *device*, unsigned char *resetPin*)

Definition at line 4 of file [ReaderMFRC522.cpp](#).

4.19.3.2 ReaderMFRC522::~ReaderMFRC522 () [virtual]

Definition at line 10 of file [ReaderMFRC522.cpp](#).

4.19.4 Member Function Documentation

4.19.4.1 int ReaderMFRC522::authenticate (unsigned char * *send*) [virtual]

Performs the authentication by sending the MF_AUTHENT command to the device.

This command manages MIFARE authentication to enable a secure communication to any MIFARE Mini, MIFARE 1K and MIFARE 4K card. The following data is written to the FIFO buffer before the command can be activated:

- Authentication command code (60h, 61h)
- Block address
- Sector key byte 0
- Sector key byte 1
- Sector key byte 2
- Sector key byte 3
- Sector key byte 4
- Sector key byte 5
- Card serial number byte 0
- Card serial number byte 1
- Card serial number byte 2
- Card serial number byte 3

Parameters

<i>send</i>	The buffer containing the above data to be send to the module.
-------------	--

Implements [Reader](#).

Definition at line 243 of file [ReaderMFRC522.cpp](#).

4.19.4.2 `unsigned int ReaderMFRC522::calculateCrc (unsigned char * buf, unsigned char len)` [virtual]

Calculate CRC of the buffer.

Parameters

<i>buf</i>	The buffer to calculate the CRC.
<i>len</i>	The length of the buffer. It must be <= 64 bytes.

Returns

The 2 bytes wide CRC.

Implements [Reader](#).

Definition at line 261 of file [ReaderMFRC522.cpp](#).

4.19.4.3 `void ReaderMFRC522::calculateCrc (unsigned char * buf, unsigned char len, unsigned char * dst)` [virtual]

Calculate CRC of the buffer.

Parameters

<i>buf</i>	The buffer to calculate the CRC.
<i>len</i>	The length of the buffer. It must be <= 64 bytes.
<i>dst</i>	The destination where the 2 bytes wide CRC will be placed.

Implements [Reader](#).

Definition at line 267 of file [ReaderMFRC522.cpp](#).

4.19.4.4 `void ReaderMFRC522::clearInterrupt (unsigned int interrupt)` [virtual]

Clear the interrupt bit at DIV_IRQ or COM_IRQ registers.

If the interrupt param is higher than 0xff it upper byte is used for the mask and the reg is DIV_IRQ otherwise the low byte is used as mask to clear the COM_IRQ register.

Parameters

<i>interrupt</i>	The interrupt to be cleared.
------------------	------------------------------

Implements [Reader](#).

Definition at line 104 of file [ReaderMFRC522.cpp](#).

4.19.4.5 `int ReaderMFRC522::communicate (unsigned char command, unsigned char * send, unsigned char * receive, unsigned char sendLen, bool checkCrc)` [virtual]

Perform a communication with the reader.

It puts the content of the send buffer into the FIFO and execute the command. Then, it reads the content from the FIFO and place it into the receive buffer.

Parameters

<i>command</i>	The command to be executed.
<i>send</i>	Buffer to place into the FIFO before executing the command.
<i>receive</i>	Buffer to receive the FIFO data after the command is executed. NOTE: different commands receive incoming bytes with different lengths, it is your duty to provide the receive buffer big enough to hold the incoming bytes. The FIFO size (64 bytes) is the maximum value for the length of this buffer. When reading blocks of the tag sectors, this buffer needs to be 18 bytes wide, to fit 16 bytes of data plus 2 bytes of the CRC.
<i>sendLen</i>	How many bytes the send buffer has.
<i>checkCrc</i>	Whether or not it is needed to check the incoming bytes CRC.

Implements [Reader](#).

Definition at line 155 of file [ReaderMFRC522.cpp](#).

4.19.4.6 `int ReaderMFRC522::communicate (unsigned char command, unsigned char * send, unsigned char * receive, unsigned char sendLen)` [inline],[virtual]

Implements [Reader](#).

Definition at line 239 of file [ReaderMFRC522.cpp](#).

4.19.4.7 `void ReaderMFRC522::configureTimer (unsigned int prescaler, unsigned int reload, bool autoStart, bool autoRestart)` [virtual]

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

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

Parameters

<i>prescaler</i>	12 bit prescaler value.
<i>reload</i>	16 bit reload value.
<i>autoStart</i>	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
<i>autoRestart</i>	1: timer automatically restarts its count-down from the 16-bit timer reload value instead of counting down to zero 0 timer decrements to 0 and the ComIrqReg register's TimerIRq bit is set to logic 1

Implements [Reader](#).

Definition at line 75 of file [ReaderMFRC522.cpp](#).

4.19.4.8 void ReaderMFRC522::disableInterrupt (unsigned int *interrupt*) [virtual]

Disable the interrupt bit at DIV_I_EN or COM_I_EN registers.

If the interrupt param is higher than 0xff it upper byte is used for the mask and the reg is DIV_I_EN otherwise the low byte is used as mask to clear the COM_I_EN register.

Parameters

<i>interrupt</i>	The interrupt to be disables.
------------------	-------------------------------

Implements [Reader](#).

Definition at line 100 of file [ReaderMFRC522.cpp](#).

4.19.4.9 void ReaderMFRC522::enableInterrupt (unsigned int *interrupt*) [virtual]

Enable the interrupt bit at DIV_I_EN or COM_I_EN registers.

If the interrupt param is higher than 0xff it upper byte is used for the mask and the reg is DIV_I_EN otherwise the low byte is used as mask to set the COM_I_EN register.

Parameters

<i>interrupt</i>	The interrupt to be disables.
------------------	-------------------------------

Implements [Reader](#).

Definition at line 96 of file [ReaderMFRC522.cpp](#).

4.19.4.10 void ReaderMFRC522::flushQueue () [virtual]

Immediately clears the internal FIFO buffer's read and write pointer and ErrorReg register's BufferOvfl bit reading this bit always returns 0.

Implements [Reader](#).

Definition at line 110 of file [ReaderMFRC522.cpp](#).

4.19.4.11 `int ReaderMFRC522::generateRandomId (unsigned char * buf) [virtual]`

Generates a 10-byte random ID number.

Parameters

<i>buf</i>	The 10-byte wide buffer where to place the random number.
------------	---

Implements [Reader](#).

Definition at line 118 of file [ReaderMFRC522.cpp](#).

4.19.4.12 `unsigned char ReaderMFRC522::getCollisionPosition () [virtual]`

Implements [Reader](#).

Definition at line 354 of file [ReaderMFRC522.cpp](#).

4.19.4.13 `ReaderMFRC522::Version ReaderMFRC522::getVersion ()`

Return the version of the device.

Definition at line 364 of file [ReaderMFRC522.cpp](#).

4.19.4.14 `bool ReaderMFRC522::hasValidCrc (unsigned char * buf, unsigned char len) [virtual]`

Implements [Reader](#).

Definition at line 252 of file [ReaderMFRC522.cpp](#).

4.19.4.15 `void ReaderMFRC522::initialize ()`

Setup the module.

Definition at line 17 of file [ReaderMFRC522.cpp](#).

4.19.4.16 `bool ReaderMFRC522::performSelfTest () [virtual]`

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

Implements [Reader](#).

Definition at line 309 of file [ReaderMFRC522.cpp](#).

4.19.4.17 `int ReaderMFRC522::readRegisterBlock (unsigned char reg, unsigned char * buf, unsigned char len)`

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

Parameters

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

Returns

If ≥ 0 : How many bytes were read. If < 0 : when error. Check `getLastError`.

Definition at line 48 of file [ReaderMFRC522.cpp](#).

4.19.4.18 `int ReaderMFRC522::readRegisterBlock (unsigned char reg, unsigned char * buf, unsigned char len, unsigned char rxAlign)`

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

Parameters

<i>reg</i>	The register number.
<i>buf</i>	The buffer where to place read bytes. MSB become LSB inside buffer.
<i>len</i>	How many bytes to read.
<i>rxAlign</i>	Defines the bit position for the first bit received to be stored in the FIFO buffer.

Returns

If ≥ 0 : How many bytes were read. If < 0 : Error. Check `getLastError`.

Definition at line 54 of file [ReaderMFRC522.cpp](#).

4.19.4.19 `void ReaderMFRC522::sendCommand (unsigned char command) [inline], [virtual]`

Sends a command to the module.

Parameters

<i>command</i>	The command to be executed.
----------------	-----------------------------

Implements [Reader](#).

Definition at line 13 of file [ReaderMFRC522.cpp](#).

4.19.4.20 `void ReaderMFRC522::setAntennaOff () [virtual]`

Disables the antenna by clearing the `TX_RF_EN` bits of the `TX_CONTROL` register.

Implements [Reader](#).

Definition at line 44 of file [ReaderMFRC522.cpp](#).

4.19.4.21 void ReaderMFRC522::setAntennaOn () [virtual]

Enables the antenna by setting the TX_RF_EN bits of the TX_CONTROL register.

Implements [Reader](#).

Definition at line 40 of file [ReaderMFRC522.cpp](#).

4.19.4.22 void ReaderMFRC522::setBitFraming (unsigned char rxAlign, unsigned char txLastBits) [virtual]

Adjusts for bit-oriented frames.

Parameters

<i>rxAlign</i>	Defines the bit position for the first bit received to be stored in the FIFO buffer.
<i>txLastBits</i>	Defines the number of bits of the last byte that will be transmitted. 000b indicates that all bits of the last byte will be transmitted.

Implements [Reader](#).

Definition at line 346 of file [ReaderMFRC522.cpp](#).

4.19.4.23 void ReaderMFRC522::setuptForAnticollision () [virtual]

Implements [Reader](#).

Definition at line 360 of file [ReaderMFRC522.cpp](#).

4.19.4.24 void ReaderMFRC522::setWaterLevel (unsigned char level) [virtual]

Set level for FIFO underflow and overflow warning.

Parameters

<i>level</i>	The FIFO level.
--------------	-----------------

Implements [Reader](#).

Definition at line 114 of file [ReaderMFRC522.cpp](#).

4.19.4.25 void ReaderMFRC522::softReset () [virtual]

Performs a soft reset to the device by sending the SOFT_RESET command.

Implements [Reader](#).

Definition at line 36 of file [ReaderMFRC522.cpp](#).

4.19.4.26 void ReaderMFRC522::startTimer () [virtual]

Stops immediately the internal timer by writting 1 to the T_START_NOW bit of the CONTROL register.

Implements [Reader](#).

Definition at line 88 of file [ReaderMFRC522.cpp](#).

4.19.4.27 `void ReaderMFRC522::stopTimer () [virtual]`

Stops immediately the internal timer by writting 1 to the T_STOP_NOW bit of the CONTROL register.

Implements [Reader](#).

Definition at line 92 of file [ReaderMFRC522.cpp](#).

4.19.4.28 `int ReaderMFRC522::tranceive (unsigned char * send, unsigned char * receive, unsigned char sendLen, bool checkCrc) [virtual]`

Tranceive data with the card.

It sends data and wait for data from the card.

Parameters

<i>output</i>	Pointer to the data to transfer to the FIFO.
<i>input</i>	NULL or pointer to buffer if data should be read back after executing the command. (max 64 bytes).
<i>sendLen</i>	Size of the data to transfer to the FIFO.
<i>checkCrc</i>	Boolean flag indicating if it is needed to calculate the CRC of the incoming data.

Implements [Reader](#).

Definition at line 147 of file [ReaderMFRC522.cpp](#).

4.19.4.29 `int ReaderMFRC522::tranceive (unsigned char * send, unsigned char * receive, unsigned char sendLen) [inline],[virtual]`

Implements [Reader](#).

Definition at line 151 of file [ReaderMFRC522.cpp](#).

4.19.4.30 `void ReaderMFRC522::turnOffEncryption () [virtual]`

Implements [Reader](#).

Definition at line 248 of file [ReaderMFRC522.cpp](#).

4.19.4.31 `bool ReaderMFRC522::waitForRegisterBits (unsigned char reg, unsigned char mask, unsigned long timeout) [virtual]`

Busy wait until a register turns some of the mask bits on.

Parameters

<i>reg</i>	The register to check.
<i>mask</i>	The bits we want to check if are one.
<i>timeout</i>	Timeout in milliseconds.

Returns

It return true if any of the mask bits become active or false if timeout.

Implements [Reader](#).

Definition at line 296 of file [ReaderMFRC522.cpp](#).

```
4.19.4.32  bool ReaderMFRC522::waitForRegisterBits ( unsigned char reg, unsigned char mask )  [inline],
           [virtual]
```

Implements [Reader](#).

Definition at line 305 of file [ReaderMFRC522.cpp](#).

```
4.19.4.33  unsigned char ReaderMFRC522::writeRegisterBlock ( unsigned char reg, unsigned char * buf, unsigned char len )
```

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

Parameters

<i>reg</i>	The register number.
<i>buf</i>	The buffer.
<i>len</i>	Buffer length.

Returns

The result of Wire.endTransmission().

Definition at line 69 of file [ReaderMFRC522.cpp](#).

4.19.5 Member Data Documentation

```
4.19.5.1  const unsigned char ReaderMFRC522::ACK = 0x0a  [static], [private]
```

Definition at line 52 of file [ReaderMFRC522.h](#).

```
4.19.5.2  RegisterBasedDevice* ReaderMFRC522::device  [private]
```

Definition at line 54 of file [ReaderMFRC522.h](#).

```
4.19.5.3  unsigned char ReaderMFRC522::resetPin  [private]
```

Definition at line 56 of file [ReaderMFRC522.h](#).

```
4.19.5.4  const unsigned char ReaderMFRC522::SAK = 0x08  [static], [private]
```

Definition at line 51 of file [ReaderMFRC522.h](#).

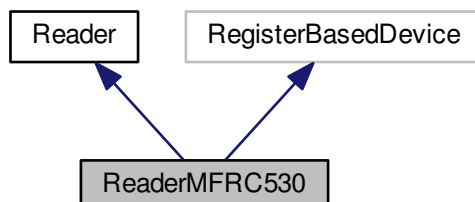
The documentation for this class was generated from the following files:

- [ReaderMFRC522.h](#)
- [ReaderMFRC522.cpp](#)

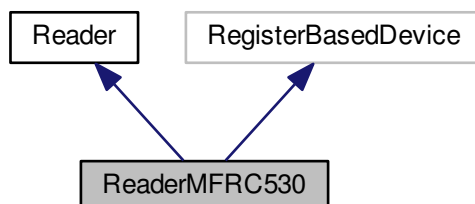
4.20 ReaderMFRC530 Class Reference

```
#include <ReaderMFRC530.h>
```

Inheritance diagram for ReaderMFRC530:



Collaboration diagram for ReaderMFRC530:



Public Types

- enum [Register](#)
- enum [Command](#)
- enum [Mask](#)
- enum [Interrupt](#) : unsigned int
- enum [Version](#)

Public Member Functions

- [ReaderMFRC530](#) (RegisterBasedDevice *device, unsigned char resetPin)
- virtual [~ReaderMFRC530](#) ()
- void [initialize](#) ()
- void [sendCommand](#) (unsigned char command)
- void [softReset](#) ()
- void [setAntennaOn](#) ()

- void [setAntennaOff](#) ()
- void [configureTimer](#) (unsigned int prescaler, unsigned int reload, bool autoStart, bool autoRestart)
- void [startTimer](#) ()
- void [stopTimer](#) ()
- void [enableInterrupt](#) (unsigned int interrupt)
- void [disableInterrupt](#) (unsigned int interrupt)
- void [clearInterrupt](#) (unsigned int interrupt)
- void [flushQueue](#) ()
- void [setWaterLevel](#) (unsigned char level)
- int [generateRandomId](#) (unsigned char *buf)
- int [communicate](#) (unsigned char command, unsigned char *send, unsigned char *receive, unsigned char sendLen, bool checkCrc)
- int [communicate](#) (unsigned char command, unsigned char *send, unsigned char *receive, unsigned char sendLen)
- int [tranceive](#) (unsigned char *send, unsigned char *receive, unsigned char sendLen, bool checkCrc)
- int [tranceive](#) (unsigned char *send, unsigned char *receive, unsigned char sendLen)
- int [authenticate](#) (unsigned char *send)
- unsigned int [calculateCrc](#) (unsigned char *buf, unsigned char len)
- void [calculateCrc](#) (unsigned char *buf, unsigned char len, unsigned char *dst)
- bool [waitForRegisterBits](#) (unsigned char reg, unsigned char mask, unsigned long timeout)
- bool [waitForRegisterBits](#) (unsigned char reg, unsigned char mask)
- [Version](#) [getVersion](#) ()
- bool [performSelfTest](#) ()
- void [setBitFraming](#) (unsigned char rxAlign, unsigned char txLastBits)
- unsigned char [getCollisionPosition](#) ()
- void [setuptForAnticollision](#) ()
- int [readRegisterBlock](#) (unsigned char reg, unsigned char *buf, unsigned char len)
- int [readRegisterBlock](#) (unsigned char reg, unsigned char *buf, unsigned char len, unsigned char rxAlign)
- unsigned char [writeRegisterBlock](#) (unsigned char reg, unsigned char *buf, unsigned char len)
- void [turnOffEncryption](#) ()
- bool [hasValidCrc](#) (unsigned char *buf, unsigned char len)

Additional Inherited Members

4.20.1 Detailed Description

Arduino - Radio Frequency Identification MFRC530.

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 13 of file [ReaderMFRC530.h](#).

4.20.2 Member Enumeration Documentation

4.20.2.1 enum [ReaderMFRC530::Command](#)

Definition at line 20 of file [ReaderMFRC530.h](#).

4.20.2.2 `enum ReaderMFRC530::Interrupt : unsigned int`

Definition at line 26 of file [ReaderMFRC530.h](#).

4.20.2.3 `enum ReaderMFRC530::Mask`

Definition at line 23 of file [ReaderMFRC530.h](#).

4.20.2.4 `enum ReaderMFRC530::Register`

Definition at line 17 of file [ReaderMFRC530.h](#).

4.20.2.5 `enum ReaderMFRC530::Version`

Definition at line 30 of file [ReaderMFRC530.h](#).

4.20.3 Constructor & Destructor Documentation

4.20.3.1 `ReaderMFRC530::ReaderMFRC530 (RegisterBasedDevice * device, unsigned char resetPin)`

4.20.3.2 `virtual ReaderMFRC530::~ReaderMFRC530 () [virtual]`

4.20.4 Member Function Documentation

4.20.4.1 `int ReaderMFRC530::authenticate (unsigned char * send) [virtual]`

Implements [Reader](#).

4.20.4.2 `unsigned int ReaderMFRC530::calculateCrc (unsigned char * buf, unsigned char len) [virtual]`

Implements [Reader](#).

4.20.4.3 `void ReaderMFRC530::calculateCrc (unsigned char * buf, unsigned char len, unsigned char * dst) [virtual]`

Implements [Reader](#).

4.20.4.4 `void ReaderMFRC530::clearInterrupt (unsigned int interrupt) [virtual]`

Implements [Reader](#).

4.20.4.5 `int ReaderMFRC530::communicate (unsigned char command, unsigned char * send, unsigned char * receive, unsigned char sendLen, bool checkCrc) [virtual]`

Implements [Reader](#).

4.20.4.6 `int ReaderMFRC530::communicate (unsigned char command, unsigned char * send, unsigned char * receive, unsigned char sendLen) [inline],[virtual]`

Implements [Reader](#).

4.20.4.7 void ReaderMFRC530::configureTimer (unsigned int *prescaler*, unsigned int *reload*, bool *autoStart*, bool *autoRestart*) [virtual]

Implements [Reader](#).

4.20.4.8 void ReaderMFRC530::disableInterrupt (unsigned int *interrupt*) [virtual]

Implements [Reader](#).

4.20.4.9 void ReaderMFRC530::enableInterrupt (unsigned int *interrupt*) [virtual]

Implements [Reader](#).

4.20.4.10 void ReaderMFRC530::flushQueue () [virtual]

Implements [Reader](#).

4.20.4.11 int ReaderMFRC530::generateRandomId (unsigned char * *buf*) [virtual]

Implements [Reader](#).

4.20.4.12 unsigned char ReaderMFRC530::getCollisionPosition () [virtual]

Implements [Reader](#).

4.20.4.13 Version ReaderMFRC530::getVersion ()

4.20.4.14 bool ReaderMFRC530::hasValidCrc (unsigned char * *buf*, unsigned char *len*) [virtual]

Implements [Reader](#).

4.20.4.15 void ReaderMFRC530::initialize ()

4.20.4.16 bool ReaderMFRC530::performSelfTest () [virtual]

Implements [Reader](#).

4.20.4.17 int ReaderMFRC530::readRegisterBlock (unsigned char *reg*, unsigned char * *buf*, unsigned char *len*)

4.20.4.18 int ReaderMFRC530::readRegisterBlock (unsigned char *reg*, unsigned char * *buf*, unsigned char *len*, unsigned char *rxAlign*)

4.20.4.19 void ReaderMFRC530::sendCommand (unsigned char *command*) [inline], [virtual]

Implements [Reader](#).

4.20.4.20 void ReaderMFRC530::setAntennaOff () [virtual]

Implements [Reader](#).

4.20.4.21 void ReaderMFRC530::setAntennaOn () [virtual]

Implements [Reader](#).

4.20.4.22 void ReaderMFRC530::setBitFraming (unsigned char *rxAlign*, unsigned char *txLastBits*) [virtual]

Implements [Reader](#).

4.20.4.23 void ReaderMFRC530::setuptForAnticollision () [virtual]

Implements [Reader](#).

4.20.4.24 void ReaderMFRC530::setWaterLevel (unsigned char *level*) [virtual]

Implements [Reader](#).

4.20.4.25 void ReaderMFRC530::softReset () [virtual]

Implements [Reader](#).

4.20.4.26 void ReaderMFRC530::startTimer () [virtual]

Implements [Reader](#).

4.20.4.27 void ReaderMFRC530::stopTimer () [virtual]

Implements [Reader](#).

4.20.4.28 int ReaderMFRC530::tranceive (unsigned char * *send*, unsigned char * *receive*, unsigned char *sendLen*, bool *checkCrc*) [virtual]

Implements [Reader](#).

4.20.4.29 int ReaderMFRC530::tranceive (unsigned char * *send*, unsigned char * *receive*, unsigned char *sendLen*) [inline], [virtual]

Implements [Reader](#).

4.20.4.30 void ReaderMFRC530::turnOffEncryption () [virtual]

Implements [Reader](#).

4.20.4.31 bool ReaderMFRC530::waitForRegisterBits (unsigned char *reg*, unsigned char *mask*, unsigned long *timeout*) [virtual]

Implements [Reader](#).

4.20.4.32 bool ReaderMFRC530::waitForRegisterBits (unsigned char *reg*, unsigned char *mask*) [inline], [virtual]

Implements [Reader](#).

4.20.4.33 unsigned char ReaderMFRC530::writeRegisterBlock (unsigned char *reg*, unsigned char * *buf*, unsigned char *len*)

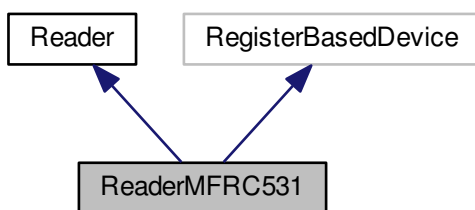
The documentation for this class was generated from the following file:

- [ReaderMFRC530.h](#)

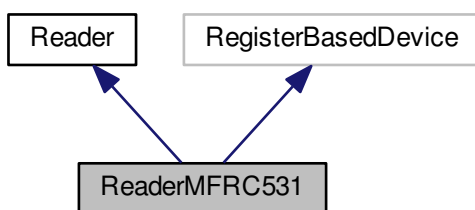
4.21 ReaderMFRC531 Class Reference

```
#include <ReaderMFRC531.h>
```

Inheritance diagram for ReaderMFRC531:



Collaboration diagram for ReaderMFRC531:



Public Types

- enum [Register](#)
- enum [Command](#)
- enum [Mask](#)
- enum [Interrupt](#) : unsigned int
- enum [Version](#)

Public Member Functions

- [ReaderMFRC531](#) (RegisterBasedDevice *device, unsigned char resetPin)
- virtual [~ReaderMFRC531](#) ()
- void [initialize](#) ()
- void [sendCommand](#) (unsigned char command)
- void [softReset](#) ()
- void [setAntennaOn](#) ()
- void [setAntennaOff](#) ()
- void [configureTimer](#) (unsigned int prescaler, unsigned int reload, bool autoStart, bool autoRestart)
- void [startTimer](#) ()
- void [stopTimer](#) ()
- void [enableInterrupt](#) (unsigned int interrupt)
- void [disableInterrupt](#) (unsigned int interrupt)
- void [clearInterrupt](#) (unsigned int interrupt)
- void [flushQueue](#) ()
- void [setWaterLevel](#) (unsigned char level)
- int [generateRandomId](#) (unsigned char *buf)
- int [communicate](#) (unsigned char command, unsigned char *send, unsigned char *receive, unsigned char sendLen, bool checkCrc)
- int [communicate](#) (unsigned char command, unsigned char *send, unsigned char *receive, unsigned char sendLen)
- int [tranceive](#) (unsigned char *send, unsigned char *receive, unsigned char sendLen, bool checkCrc)
- int [tranceive](#) (unsigned char *send, unsigned char *receive, unsigned char sendLen)
- int [authenticate](#) (unsigned char *send)
- unsigned int [calculateCrc](#) (unsigned char *buf, unsigned char len)
- void [calculateCrc](#) (unsigned char *buf, unsigned char len, unsigned char *dst)
- bool [waitForRegisterBits](#) (unsigned char reg, unsigned char mask, unsigned long timeout)
- bool [waitForRegisterBits](#) (unsigned char reg, unsigned char mask)
- [Version](#) [getVersion](#) ()
- bool [performSelfTest](#) ()
- void [setBitFraming](#) (unsigned char rxAlign, unsigned char txLastBits)
- unsigned char [getCollisionPosition](#) ()
- void [setuptForAnticollision](#) ()
- int [readRegisterBlock](#) (unsigned char reg, unsigned char *buf, unsigned char len)
- int [readRegisterBlock](#) (unsigned char reg, unsigned char *buf, unsigned char len, unsigned char rxAlign)
- unsigned char [writeRegisterBlock](#) (unsigned char reg, unsigned char *buf, unsigned char len)
- void [turnOffEncryption](#) ()
- bool [hasValidCrc](#) (unsigned char *buf, unsigned char len)

Additional Inherited Members

4.21.1 Detailed Description

Arduino - Radio Frequency Identification MFRC531.

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 13 of file [ReaderMFRC531.h](#).

4.21.2 Member Enumeration Documentation

4.21.2.1 enum ReaderMFRC531::Command

Definition at line 20 of file [ReaderMFRC531.h](#).

4.21.2.2 enum ReaderMFRC531::Interrupt : unsigned int

Definition at line 26 of file [ReaderMFRC531.h](#).

4.21.2.3 enum ReaderMFRC531::Mask

Definition at line 23 of file [ReaderMFRC531.h](#).

4.21.2.4 enum ReaderMFRC531::Register

Definition at line 17 of file [ReaderMFRC531.h](#).

4.21.2.5 enum ReaderMFRC531::Version

Definition at line 30 of file [ReaderMFRC531.h](#).

4.21.3 Constructor & Destructor Documentation

4.21.3.1 ReaderMFRC531::ReaderMFRC531 (RegisterBasedDevice * *device*, unsigned char *resetPin*)

4.21.3.2 virtual ReaderMFRC531::~~ReaderMFRC531 () [virtual]

4.21.4 Member Function Documentation

4.21.4.1 int ReaderMFRC531::authenticate (unsigned char * *send*) [virtual]

Implements [Reader](#).

4.21.4.2 unsigned int ReaderMFRC531::calculateCrc (unsigned char * *buf*, unsigned char *len*) [virtual]

Implements [Reader](#).

4.21.4.3 void ReaderMFRC531::calculateCrc (unsigned char * *buf*, unsigned char *len*, unsigned char * *dst*) [virtual]

Implements [Reader](#).

4.21.4.4 void ReaderMFRC531::clearInterrupt (unsigned int *interrupt*) [virtual]

Implements [Reader](#).

4.21.4.5 `int ReaderMFRC531::communicate (unsigned char command, unsigned char * send, unsigned char * receive, unsigned char sendLen, bool checkCrc)` [virtual]

Implements [Reader](#).

4.21.4.6 `int ReaderMFRC531::communicate (unsigned char command, unsigned char * send, unsigned char * receive, unsigned char sendLen)` [inline],[virtual]

Implements [Reader](#).

4.21.4.7 `void ReaderMFRC531::configureTimer (unsigned int prescaler, unsigned int reload, bool autoStart, bool autoRestart)` [virtual]

Implements [Reader](#).

4.21.4.8 `void ReaderMFRC531::disableInterrupt (unsigned int interrupt)` [virtual]

Implements [Reader](#).

4.21.4.9 `void ReaderMFRC531::enableInterrupt (unsigned int interrupt)` [virtual]

Implements [Reader](#).

4.21.4.10 `void ReaderMFRC531::flushQueue ()` [virtual]

Implements [Reader](#).

4.21.4.11 `int ReaderMFRC531::generateRandomId (unsigned char * buf)` [virtual]

Implements [Reader](#).

4.21.4.12 `unsigned char ReaderMFRC531::getCollisionPosition ()` [virtual]

Implements [Reader](#).

4.21.4.13 `Version ReaderMFRC531::getVersion ()`

4.21.4.14 `bool ReaderMFRC531::hasValidCrc (unsigned char * buf, unsigned char len)` [virtual]

Implements [Reader](#).

4.21.4.15 `void ReaderMFRC531::initialize ()`

4.21.4.16 `bool ReaderMFRC531::performSelfTest ()` [virtual]

Implements [Reader](#).

4.21.4.17 `int ReaderMFRC531::readRegisterBlock (unsigned char reg, unsigned char * buf, unsigned char len)`

4.21.4.18 `int ReaderMFRC531::readRegisterBlock (unsigned char reg, unsigned char * buf, unsigned char len, unsigned char rxAlign)`

4.21.4.19 `void ReaderMFRC531::sendCommand (unsigned char command)` `[inline],[virtual]`

Implements [Reader](#).

4.21.4.20 `void ReaderMFRC531::setAntennaOff ()` `[virtual]`

Implements [Reader](#).

4.21.4.21 `void ReaderMFRC531::setAntennaOn ()` `[virtual]`

Implements [Reader](#).

4.21.4.22 `void ReaderMFRC531::setBitFraming (unsigned char rxAlign, unsigned char txLastBits)` `[virtual]`

Implements [Reader](#).

4.21.4.23 `void ReaderMFRC531::setuptForAnticollision ()` `[virtual]`

Implements [Reader](#).

4.21.4.24 `void ReaderMFRC531::setWaterLevel (unsigned char level)` `[virtual]`

Implements [Reader](#).

4.21.4.25 `void ReaderMFRC531::softReset ()` `[virtual]`

Implements [Reader](#).

4.21.4.26 `void ReaderMFRC531::startTimer ()` `[virtual]`

Implements [Reader](#).

4.21.4.27 `void ReaderMFRC531::stopTimer ()` `[virtual]`

Implements [Reader](#).

4.21.4.28 `int ReaderMFRC531::tranceive (unsigned char * send, unsigned char * receive, unsigned char sendLen, bool checkCrc)` `[virtual]`

Implements [Reader](#).

4.21.4.29 `int ReaderMFRC531::tranceive (unsigned char * send, unsigned char * receive, unsigned char sendLen)` `[inline],[virtual]`

Implements [Reader](#).

4.21.4.30 void ReaderMFRC531::turnOffEncryption () [virtual]

Implements [Reader](#).

4.21.4.31 bool ReaderMFRC531::waitForRegisterBits (unsigned char *reg*, unsigned char *mask*, unsigned long *timeout*) [virtual]

Implements [Reader](#).

4.21.4.32 bool ReaderMFRC531::waitForRegisterBits (unsigned char *reg*, unsigned char *mask*) [inline], [virtual]

Implements [Reader](#).

4.21.4.33 unsigned char ReaderMFRC531::writeRegisterBlock (unsigned char *reg*, unsigned char * *buf*, unsigned char *len*)

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

- [ReaderMFRC531.h](#)

4.22 ReaderMFRC522::RF_CFGbits Union Reference

```
#include <ReaderMFRC522.h>
```

Public Attributes

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

4.22.1 Detailed Description

RF_CFG register.

Configures the receiver gain.

Definition at line 1091 of file [ReaderMFRC522.h](#).

4.22.2 Member Data Documentation

4.22.2.1 struct { ... }

4.22.2.2 unsigned ReaderMFRC522::RF_CFGbits::char

Definition at line 1096 of file [ReaderMFRC522.h](#).

4.22.2.3 unsigned char ReaderMFRC522::RF_CFGbits::RX_GAIN

Definition at line 1107 of file [ReaderMFRC522.h](#).

4.22.2.4 unsigned char ReaderMFRC522::RF_CFGbits::value

Definition at line 1112 of file [ReaderMFRC522.h](#).

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

- [ReaderMFRC522.h](#)

4.23 ReaderMFRC522::RX_MODEbits Union Reference

```
#include <ReaderMFRC522.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

4.23.1 Detailed Description

RX_MODE register.

Defines the data rate during reception.

Definition at line 803 of file [ReaderMFRC522.h](#).

4.23.2 Member Data Documentation

4.23.2.1 struct { ... }

4.23.2.2 unsigned ReaderMFRC522::RX_MODEbits::char

Definition at line 808 of file [ReaderMFRC522.h](#).

4.23.2.3 unsigned char ReaderMFRC522::RX_MODEbits::RX_CRC_EN

Definition at line 832 of file [ReaderMFRC522.h](#).

4.23.2.4 unsigned char ReaderMFRC522::RX_MODEbits::RX_MULTIPLE

Definition at line 818 of file [ReaderMFRC522.h](#).

4.23.2.5 unsigned char ReaderMFRC522::RX_MODEbits::RX_NO_ERR

Definition at line 821 of file [ReaderMFRC522.h](#).

4.23.2.6 unsigned char ReaderMFRC522::RX_MODEbits::RX_SPEED

Definition at line 828 of file [ReaderMFRC522.h](#).

4.23.2.7 unsigned char ReaderMFRC522::RX_MODEbits::value

Definition at line 834 of file [ReaderMFRC522.h](#).

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

- [ReaderMFRC522.h](#)

4.24 ReaderMFRC522::RX_SELbits Union Reference

```
#include <ReaderMFRC522.h>
```

Public Attributes

- struct {
 unsigned char [RX_WAIT](#):6
 unsigned char [UART_SEL](#):2
};
- unsigned char [value](#)

4.24.1 Detailed Description

RX_SEL register.

Selects internal receiver settings.

Definition at line 952 of file [ReaderMFRC522.h](#).

4.24.2 Member Data Documentation

4.24.2.1 struct { ... }

4.24.2.2 unsigned char ReaderMFRC522::RX_SELbits::RX_WAIT

Definition at line 959 of file [ReaderMFRC522.h](#).

4.24.2.3 unsigned char ReaderMFRC522::RX_SELbits::UART_SEL

Definition at line 966 of file [ReaderMFRC522.h](#).

4.24.2.4 unsigned char ReaderMFRC522::RX_SELbits::value

Definition at line 968 of file [ReaderMFRC522.h](#).

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

- [ReaderMFRC522.h](#)

4.25 ReaderMFRC522::RX_THRESHOLDbits Union Reference

```
#include <ReaderMFRC522.h>
```

Public Attributes

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

4.25.1 Detailed Description

RX_THRESHOLD register.

Selects thresholds for the bit decoder.

Definition at line 976 of file [ReaderMFRC522.h](#).

4.25.2 Member Data Documentation

4.25.2.1 struct { ... }

4.25.2.2 unsigned ReaderMFRC522::RX_THRESHOLDbits::char

Definition at line 985 of file [ReaderMFRC522.h](#).

4.25.2.3 unsigned char ReaderMFRC522::RX_THRESHOLDbits::COLL_LEVEL

Definition at line 982 of file [ReaderMFRC522.h](#).

4.25.2.4 unsigned char ReaderMFRC522::RX_THRESHOLDbits::MIN_LEVEL

Definition at line 988 of file [ReaderMFRC522.h](#).

4.25.2.5 unsigned char ReaderMFRC522::RX_THRESHOLDbits::value

Definition at line 990 of file [ReaderMFRC522.h](#).

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

- [ReaderMFRC522.h](#)

4.26 ReaderMFRC522::SERIAL_SPEEDbits Union Reference

```
#include <ReaderMFRC522.h>
```

Public Attributes

- struct {
 unsigned char [BR_T1](#):5
 unsigned char [BR_T0](#):3
};
- unsigned char [value](#)

4.26.1 Detailed Description

SERIAL_SPEED register.

Selects the speed of the serial UART interface.

Definition at line 1073 of file [ReaderMFRC522.h](#).

4.26.2 Member Data Documentation

4.26.2.1 struct { ... }

4.26.2.2 unsigned char ReaderMFRC522::SERIAL_SPEEDbits::BR_T0

Definition at line 1081 of file [ReaderMFRC522.h](#).

4.26.2.3 unsigned char ReaderMFRC522::SERIAL_SPEEDbits::BR_T1

Definition at line 1078 of file [ReaderMFRC522.h](#).

4.26.2.4 unsigned char ReaderMFRC522::SERIAL_SPEEDbits::value

Definition at line 1083 of file [ReaderMFRC522.h](#).

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

- [ReaderMFRC522.h](#)

4.27 ReaderMFRC522::STATUS1bits Union Reference

```
#include <ReaderMFRC522.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

4.27.1 Detailed Description

STATUS1 register.

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

Definition at line 516 of file [ReaderMFRC522.h](#).

4.27.2 Member Data Documentation

4.27.2.1 struct { ... }

4.27.2.2 unsigned ReaderMFRC522::STATUS1bits::char

Definition at line 553 of file [ReaderMFRC522.h](#).

4.27.2.3 unsigned char ReaderMFRC522::STATUS1bits::CRC_OK

Definition at line 550 of file [ReaderMFRC522.h](#).

4.27.2.4 unsigned char ReaderMFRC522::STATUS1bits::CRC_READY

Definition at line 544 of file [ReaderMFRC522.h](#).

4.27.2.5 unsigned char ReaderMFRC522::STATUS1bits::HI_ALERT

Definition at line 532 of file [ReaderMFRC522.h](#).

4.27.2.6 unsigned char ReaderMFRC522::STATUS1bits::IRQ

Definition at line 541 of file [ReaderMFRC522.h](#).

4.27.2.7 unsigned char ReaderMFRC522::STATUS1bits::LO_ALERT

Definition at line 525 of file [ReaderMFRC522.h](#).

4.27.2.8 unsigned char ReaderMFRC522::STATUS1bits::T_RUNNING

Definition at line 537 of file [ReaderMFRC522.h](#).

4.27.2.9 unsigned char ReaderMFRC522::STATUS1bits::value

Definition at line 555 of file [ReaderMFRC522.h](#).

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

- [ReaderMFRC522.h](#)

4.28 ReaderMFRC522::STATUS2bits Union Reference

```
#include <ReaderMFRC522.h>
```

Public Attributes

- struct {
 - unsigned char MODEM_STATE:3
 - unsigned char MF_CRYPT01_ON:1
 - unsigned char:2
 - unsigned char I2C_FORCE_HS:1
 - unsigned char TEMP_SENS_CLEAR:1};
- unsigned char value

4.28.1 Detailed Description

STATUS2 register.

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

Definition at line 563 of file [ReaderMFRC522.h](#).

4.28.2 Member Data Documentation

4.28.2.1 struct { ... }

4.28.2.2 unsigned ReaderMFRC522::STATUS2bits::char

Definition at line 585 of file [ReaderMFRC522.h](#).

4.28.2.3 unsigned char ReaderMFRC522::STATUS2bits::I2C_FORCE_HS

Definition at line 590 of file [ReaderMFRC522.h](#).

4.28.2.4 unsigned char ReaderMFRC522::STATUS2bits::MF_CRYPT01_ON

Definition at line 582 of file [ReaderMFRC522.h](#).

4.28.2.5 unsigned char ReaderMFRC522::STATUS2bits::MODEM_STATE

Definition at line 577 of file [ReaderMFRC522.h](#).

4.28.2.6 unsigned char ReaderMFRC522::STATUS2bits::TEMP_SENS_CLEAR

Definition at line 593 of file [ReaderMFRC522.h](#).

4.28.2.7 unsigned char ReaderMFRC522::STATUS2bits::value

Definition at line 595 of file [ReaderMFRC522.h](#).

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

- [ReaderMFRC522.h](#)

4.29 ReaderMFRC522::T_MODEbits Union Reference

```
#include <ReaderMFRC522.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](#)

4.29.1 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 1184 of file [ReaderMFRC522.h](#).

4.29.2 Member Data Documentation

4.29.2.1 struct { ... }

4.29.2.2 unsigned char ReaderMFRC522::T_MODEbits::T_AUTO

Definition at line 1213 of file [ReaderMFRC522.h](#).

4.29.2.3 unsigned char ReaderMFRC522::T_MODEbits::T_AUTO_RESTART

Definition at line 1198 of file [ReaderMFRC522.h](#).

4.29.2.4 unsigned char ReaderMFRC522::T_MODEbits::T_GATED

Definition at line 1206 of file [ReaderMFRC522.h](#).

4.29.2.5 unsigned char ReaderMFRC522::T_MODEbits::T_PRESCALER_HI

Definition at line 1194 of file [ReaderMFRC522.h](#).

4.29.2.6 unsigned char ReaderMFRC522::T_MODEbits::value

Definition at line 1215 of file [ReaderMFRC522.h](#).

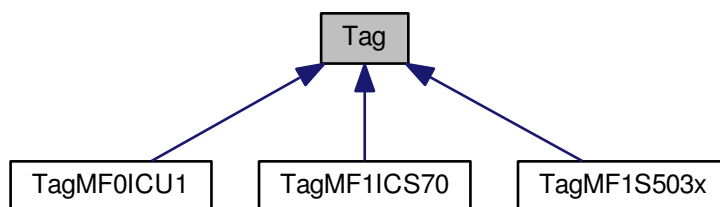
The documentation for this union was generated from the following file:

- [ReaderMFRC522.h](#)

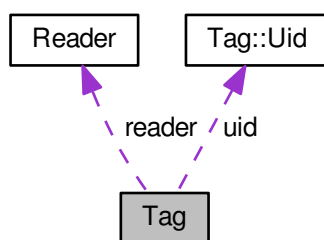
4.30 Tag Class Reference

```
#include <Tag.h>
```

Inheritance diagram for Tag:



Collaboration diagram for Tag:



Classes

- struct [Uid](#)

Public Types

- enum [State](#) {
[POWER_OFF](#) = 0x00, [IDLE](#) = 0x01, [READY](#) = 0x02, [ACTIVE](#) = 0x03,
[HALT](#) = 0x04 }
- enum [TagType](#) {
[MIFARE_UNKNOWN](#) = 0x00, [MIFARE_MINI](#) = 0x01, [MIFARE_1K](#) = 0x02, [MIFARE_4K](#) = 0x03,
[MIFARE_UL](#) = 0x04, [MIFARE_PLUS](#) = 0x06, [MIFARE_NOT_COMPLETE](#) = 0xff }
- enum [KeyType](#) { [KEY_A](#) = 0x00, [KEY_B](#) = 0x01 }
- enum [BlockType](#) { [DATA_BLOCK](#) = 0x00, [VALUE_BLOCK](#) = 0x01 }
- enum [Command](#) {
[REQUEST](#) = 0x26, [WAKE_UP](#) = 0x52, [SEL_CL1](#) = 0x93, [SEL_CL2](#) = 0x95,
[SEL_CL3](#) = 0x97, [HLT_A](#) = 0x50, [AUTH_KEY_A](#) = 0x60, [AUTH_KEY_B](#) = 0x61,
[READ](#) = 0x30, [WRITE](#) = 0xa0, [DECREMENT](#) = 0xc0, [INCREMENT](#) = 0xc1,
[RESTORE](#) = 0xc2, [TRANSFER](#) = 0xb0 }

Public Member Functions

- [Tag](#) ([Reader](#) *reader)
- virtual [~Tag](#) ()
- [Uid](#) [getUid](#) ()
- bool [hasAnticollisionSupport](#) ()
- [TagType](#) [getTagType](#) ()
- void [setState](#) ([State](#) state)
- [State](#) [getState](#) ()
- virtual bool [detect](#) (unsigned char command)
- virtual bool [activate](#) ()
- virtual bool [activateWakeUp](#) ()
- virtual bool [request](#) ()
- virtual bool [wakeUp](#) ()
- virtual bool [select](#) ()
- virtual bool [halt](#) ()
- virtual bool [authenticate](#) (unsigned char address, [KeyType](#) type, unsigned char *key)
- virtual bool [readBlock](#) (unsigned char address, unsigned char *buf)
- virtual bool [writeBlock](#) (unsigned char address, unsigned char *buf)
- virtual bool [readBlockSlice](#) (unsigned char address, unsigned char from, unsigned char len, unsigned char *buf)
- virtual bool [writeBlockSlice](#) (unsigned char address, unsigned char from, unsigned char len, unsigned char *buf)
- virtual int [readByte](#) (unsigned char address, unsigned char pos)
- virtual bool [writeByte](#) (unsigned char address, unsigned char pos, unsigned char value)
- virtual bool [decrement](#) ()
- virtual bool [increment](#) ()
- virtual bool [restore](#) ()
- virtual bool [transfer](#) ()
- virtual bool [setBlockType](#) (unsigned char address, [BlockType](#) type)
- virtual bool [readAccessBits](#) (unsigned char sector, unsigned char *buf)
- virtual bool [writeAccessBits](#) (unsigned char sector, unsigned char *buf)
- virtual bool [setBlockPermission](#) (unsigned char address, unsigned char permission)
- virtual bool [writeKey](#) (unsigned char sector, [KeyType](#) type, unsigned char *key)
- virtual bool [readKey](#) (unsigned char sector, [KeyType](#) type, unsigned char *key)
- virtual void [setupAuthenticationKey](#) ([KeyType](#) keyType, unsigned char *key)
- virtual void [setSectorTrailerProtected](#) (bool protect)

Protected Member Functions

- unsigned char [computeNvb](#) (unsigned char collisionPos)
- virtual unsigned char [getSectorSize](#) (unsigned char sector)=0
- virtual unsigned char [isAddressSectorTrailer](#) (unsigned char address)=0
- virtual unsigned char [addressToSector](#) (unsigned char address)=0
- virtual unsigned char [getSectorTrailerAddress](#) (unsigned char sector)=0
- void [computeTagType](#) ()

Protected Attributes

- [Reader](#) * reader
- [TagType](#) tagType
- [Uid](#) uid
- bool supportsAnticollision
- [State](#) state
- [KeyType](#) keyType
- unsigned char * key
- bool sectorTrailerProtected

4.30.1 Detailed Description

Definition at line 20 of file [Tag.h](#).

4.30.2 Member Enumeration Documentation

4.30.2.1 enum Tag::BlockType

Enumerator

DATA_BLOCK
VALUE_BLOCK

Definition at line 47 of file [Tag.h](#).

4.30.2.2 enum Tag::Command

Enumerator

REQUEST
WAKE_UP
SEL_CL1
SEL_CL2
SEL_CL3
HLT_A
AUTH_KEY_A
AUTH_KEY_B
READ
WRITE
DECREMENT
INCREMENT
RESTORE
TRANSFER

Definition at line 53 of file [Tag.h](#).

4.30.2.3 enum Tag::KeyType

Enumerator

KEY_A
KEY_B

Definition at line 42 of file [Tag.h](#).

4.30.2.4 enum Tag::State

Enumerator

POWER_OFF
IDLE
READY
ACTIVE
HALT

Definition at line 24 of file [Tag.h](#).

4.30.2.5 enum Tag::TagType

Enumerator

MIFARE_UNKNOWN
MIFARE_MINI
MIFARE_1K
MIFARE_4K
MIFARE_UL
MIFARE_PLUS
MIFARE_NOT_COMPLETE

Definition at line 32 of file [Tag.h](#).

4.30.3 Constructor & Destructor Documentation

4.30.3.1 Tag::Tag (Reader * reader)

Arduino - Radio Frequency Identification.

Author

Dalmir da Silva dalmirdasilva@gmail.comDefinition at line 10 of file [Tag.cpp](#).

4.30.3.2 Tag::~Tag () [virtual]

Definition at line 15 of file [Tag.cpp](#).

4.30.4 Member Function Documentation

4.30.4.1 bool Tag::activate () [virtual]

This function performs a 'Request-Idle', 'Anticollision', 'Select' sequence to activate the PICC and change its state from IDLE to ACTIVE state.

Cascaded serial numbers are handled correctly.

Definition at line 38 of file [Tag.cpp](#).

4.30.4.2 `bool Tag::activateWakeUp () [virtual]`

This function performs a 'Request-All', 'Anticollision', 'Select' sequence to activate the PICC and change its state from IDLE to ACTIVE state.

Cascaded serial numbers are handled correctly.

Definition at line 42 of file [Tag.cpp](#).

4.30.4.3 `virtual unsigned char Tag::addressToSector (unsigned char address) [protected],[pure virtual]`

Implemented in [TagMF1ICS70](#), and [TagMF1S503x](#).

4.30.4.4 `bool Tag::authenticate (unsigned char address, KeyType type, unsigned char * key) [virtual]`

This function authenticates one card's sector (according to the block address) using the specified master key A or B, addressed with `auth_mode`.

Having send the command to the card the function waits for the card's answer. This function is calling compatible with authentication functions former reader IC's. The keys are stored by the microcontroller, which should be capable for the key management.

Definition at line 137 of file [Tag.cpp](#).

4.30.4.5 `unsigned char Tag::computeNvb (unsigned char collisionPos) [protected]`

Definition at line 274 of file [Tag.cpp](#).

4.30.4.6 `void Tag::computeTagType () [protected]`

Definition at line 280 of file [Tag.cpp](#).

4.30.4.7 `bool Tag::decrement () [virtual]`

Definition at line 226 of file [Tag.cpp](#).

4.30.4.8 `bool Tag::detect (unsigned char command) [virtual]`

Definition at line 46 of file [Tag.cpp](#).

4.30.4.9 `virtual unsigned char Tag::getSectorSize (unsigned char sector) [protected],[pure virtual]`

Implemented in [TagMF1ICS70](#), and [TagMF1S503x](#).

4.30.4.10 `virtual unsigned char Tag::getSectorTrailerAddress (unsigned char sector) [protected],[pure virtual]`

Implemented in [TagMF1ICS70](#), and [TagMF1S503x](#).

4.30.4.11 `Tag::State Tag::getState ()`

Definition at line 34 of file [Tag.cpp](#).

4.30.4.12 Tag::TagType Tag::getTagType ()

Definition at line 26 of file [Tag.cpp](#).

4.30.4.13 Tag::Uid Tag::getUid ()

Definition at line 18 of file [Tag.cpp](#).

4.30.4.14 bool Tag::halt () [virtual]

Definition at line 125 of file [Tag.cpp](#).

4.30.4.15 bool Tag::hasAnticollisionSupport ()

Definition at line 22 of file [Tag.cpp](#).

4.30.4.16 bool Tag::increment () [virtual]

Definition at line 230 of file [Tag.cpp](#).

4.30.4.17 virtual unsigned char Tag::isAddressSectorTrailer (unsigned char *address*) [protected], [pure virtual]

Implemented in [TagMF1ICS70](#), and [TagMF1S503x](#).

4.30.4.18 bool Tag::readAccessBits (unsigned char *sector*, unsigned char * *buf*) [virtual]

Definition at line 247 of file [Tag.cpp](#).

4.30.4.19 bool Tag::readBlock (unsigned char *address*, unsigned char * *buf*) [virtual]

Definition at line 153 of file [Tag.cpp](#).

4.30.4.20 bool Tag::readBlockSlice (unsigned char *address*, unsigned char *from*, unsigned char *len*, unsigned char * *buf*) [virtual]

Definition at line 183 of file [Tag.cpp](#).

4.30.4.21 int Tag::readByte (unsigned char *address*, unsigned char *pos*) [virtual]

Definition at line 207 of file [Tag.cpp](#).

4.30.4.22 bool Tag::readKey (unsigned char *sector*, KeyType *type*, unsigned char * *key*) [virtual]

Definition at line 264 of file [Tag.cpp](#).

4.30.4.23 bool Tag::request () [virtual]

Definition at line 58 of file [Tag.cpp](#).

4.30.4.24 `bool Tag::restore () [virtual]`

Definition at line 234 of file [Tag.cpp](#).

4.30.4.25 `bool Tag::select () [virtual]`

Definition at line 66 of file [Tag.cpp](#).

4.30.4.26 `bool Tag::setBlockPermission (unsigned char address, unsigned char permission) [virtual]`

Definition at line 255 of file [Tag.cpp](#).

4.30.4.27 `bool Tag::setBlockType (unsigned char address, BlockType type) [virtual]`

Definition at line 242 of file [Tag.cpp](#).

4.30.4.28 `void Tag::setSectorTrailerProtected (bool protect) [virtual]`

Definition at line 306 of file [Tag.cpp](#).

4.30.4.29 `void Tag::setState (Tag::State state)`

Definition at line 30 of file [Tag.cpp](#).

4.30.4.30 `void Tag::setupAuthenticationKey (KeyType keyType, unsigned char * key) [virtual]`

Definition at line 269 of file [Tag.cpp](#).

4.30.4.31 `bool Tag::transfer () [virtual]`

Definition at line 238 of file [Tag.cpp](#).

4.30.4.32 `bool Tag::wakeUp () [virtual]`

Definition at line 62 of file [Tag.cpp](#).

4.30.4.33 `bool Tag::writeAccessBits (unsigned char sector, unsigned char * buf) [virtual]`

Definition at line 251 of file [Tag.cpp](#).

4.30.4.34 `bool Tag::writeBlock (unsigned char address, unsigned char * buf) [virtual]`

Reimplemented in [TagMF1ICS70](#), and [TagMF1S503x](#).

Definition at line 163 of file [Tag.cpp](#).

4.30.4.35 `bool Tag::writeBlockSlice (unsigned char address, unsigned char from, unsigned char len, unsigned char * buf)
[virtual]`

Definition at line 195 of file [Tag.cpp](#).

4.30.4.36 `bool Tag::writeByte (unsigned char address, unsigned char pos, unsigned char value)` [virtual]

Definition at line 216 of file [Tag.cpp](#).

4.30.4.37 `bool Tag::writeKey (unsigned char sector, KeyType type, unsigned char * key)` [virtual]

Definition at line 259 of file [Tag.cpp](#).

4.30.5 Member Data Documentation

4.30.5.1 `unsigned char* Tag::key` [protected]

Definition at line 203 of file [Tag.h](#).

4.30.5.2 `KeyType Tag::keyType` [protected]

Definition at line 201 of file [Tag.h](#).

4.30.5.3 `Reader* Tag::reader` [protected]

Definition at line 191 of file [Tag.h](#).

4.30.5.4 `bool Tag::sectorTrailerProtected` [protected]

Definition at line 205 of file [Tag.h](#).

4.30.5.5 `State Tag::state` [protected]

Definition at line 199 of file [Tag.h](#).

4.30.5.6 `bool Tag::supportsAnticollision` [protected]

Definition at line 197 of file [Tag.h](#).

4.30.5.7 `TagType Tag::tagType` [protected]

Definition at line 193 of file [Tag.h](#).

4.30.5.8 `Uid Tag::uid` [protected]

Definition at line 195 of file [Tag.h](#).

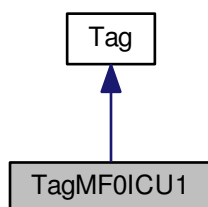
The documentation for this class was generated from the following files:

- [Tag.h](#)
- [Tag.cpp](#)

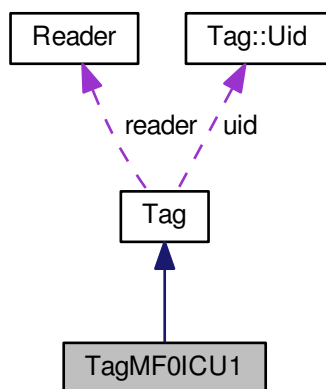
4.31 TagMF0ICU1 Class Reference

```
#include <TagMF0ICU1.h>
```

Inheritance diagram for TagMF0ICU1:



Collaboration diagram for TagMF0ICU1:



Public Types

- enum [Permission](#) {
[LEVEL_0](#) = 0x00, [LEVEL_1](#) = 0x01, [LEVEL_2](#) = 0x02, [LEVEL_3](#) = 0x03,
[LEVEL_4](#) = 0x04, [LEVEL_5](#) = 0x05, [LEVEL_6](#) = 0x06, [LEVEL_7](#) = 0x07 }

Public Member Functions

- [TagMF0ICU1](#) ([Reader](#) *[reader](#))

Additional Inherited Members

4.31.1 Detailed Description

Arduino - Radio Frequency Identification MFRC522.

Author

Dalmir da Silva dalmirdasilva@gmail.com MIFARE Classic 1K

Definition at line 16 of file [TagMF0ICU1.h](#).

4.31.2 Member Enumeration Documentation

4.31.2.1 enum TagMF0ICU1::Permission

Enumerator

LEVEL_0
LEVEL_1
LEVEL_2
LEVEL_3
LEVEL_4
LEVEL_5
LEVEL_6
LEVEL_7

Definition at line 20 of file [TagMF0ICU1.h](#).

4.31.3 Constructor & Destructor Documentation

4.31.3.1 TagMF0ICU1::TagMF0ICU1 (Reader * reader)

Definition at line 4 of file [TagMF0ICU1.cpp](#).

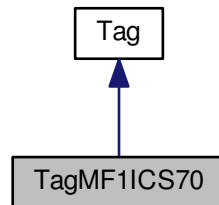
The documentation for this class was generated from the following files:

- [TagMF0ICU1.h](#)
- [TagMF0ICU1.cpp](#)

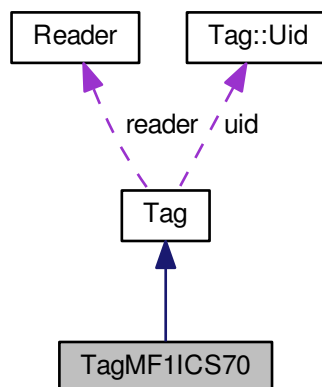
4.32 TagMF1ICS70 Class Reference

```
#include <TagMF1ICS70.h>
```

Inheritance diagram for TagMF1ICS70:



Collaboration diagram for TagMF1ICS70:



Public Types

- enum [Permission](#) {
[LEVEL_0](#) = 0x00, [LEVEL_1](#) = 0x01, [LEVEL_2](#) = 0x02, [LEVEL_3](#) = 0x03,
[LEVEL_4](#) = 0x04, [LEVEL_5](#) = 0x05, [LEVEL_6](#) = 0x06, [LEVEL_7](#) = 0x07 }

Public Member Functions

- [TagMF1ICS70](#) ([Reader](#) *[reader](#))
- bool [writeBlock](#) (unsigned char address, unsigned char *buf)
- unsigned char [getSectorSize](#) (unsigned char sector)
- unsigned char [isAddressSectorTrailer](#) (unsigned char address)
- unsigned char [addressToSector](#) (unsigned char address)
- unsigned char [getSectorTrailerAddress](#) (unsigned char sector)

Additional Inherited Members

4.32.1 Detailed Description

Definition at line 24 of file [TagMF1ICS70.h](#).

4.32.2 Member Enumeration Documentation

4.32.2.1 enum TagMF1ICS70::Permission

Enumerator

LEVEL_0
LEVEL_1
LEVEL_2
LEVEL_3
LEVEL_4
LEVEL_5
LEVEL_6
LEVEL_7

Definition at line 28 of file [TagMF1ICS70.h](#).

4.32.3 Constructor & Destructor Documentation

4.32.3.1 TagMF1ICS70::TagMF1ICS70 (Reader * reader)

Definition at line 4 of file [TagMF1ICS70.cpp](#).

4.32.4 Member Function Documentation

4.32.4.1 unsigned char TagMF1ICS70::addressToSector (unsigned char *address*) [virtual]

Implements [Tag](#).

Definition at line 23 of file [TagMF1ICS70.cpp](#).

4.32.4.2 unsigned char TagMF1ICS70::getSectorSize (unsigned char *sector*) [virtual]

Implements [Tag](#).

Definition at line 15 of file [TagMF1ICS70.cpp](#).

4.32.4.3 unsigned char TagMF1ICS70::getSectorTrailerAddress (unsigned char *sector*) [virtual]

Implements [Tag](#).

Definition at line 42 of file [TagMF1ICS70.cpp](#).

4.32.4.4 `unsigned char TagMF1ICS70::isAddressSectorTrailer (unsigned char address)` `[virtual]`

Implements [Tag](#).

Definition at line 34 of file [TagMF1ICS70.cpp](#).

4.32.4.5 `bool TagMF1ICS70::writeBlock (unsigned char address, unsigned char * buf)` `[virtual]`

Reimplemented from [Tag](#).

Definition at line 8 of file [TagMF1ICS70.cpp](#).

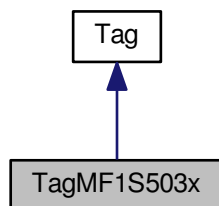
The documentation for this class was generated from the following files:

- [TagMF1ICS70.h](#)
- [TagMF1ICS70.cpp](#)

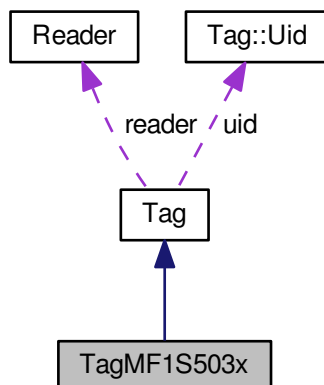
4.33 TagMF1S503x Class Reference

```
#include <TagMF1S503x.h>
```

Inheritance diagram for TagMF1S503x:



Collaboration diagram for TagMF1S503x:



Public Types

- enum [Permission](#) {
 [LEVEL_0](#) = 0x00, [LEVEL_1](#) = 0x01, [LEVEL_2](#) = 0x02, [LEVEL_3](#) = 0x03,
 [LEVEL_4](#) = 0x04, [LEVEL_5](#) = 0x05, [LEVEL_6](#) = 0x06, [LEVEL_7](#) = 0x07 }

Public Member Functions

- [TagMF1S503x](#) ([Reader](#) *reader)
- bool [writeBlock](#) (unsigned char address, unsigned char *buf)
- unsigned char [getSectorSize](#) (unsigned char sector)
- unsigned char [isAddressSectorTrailer](#) (unsigned char address)
- unsigned char [addressToSector](#) (unsigned char address)
- unsigned char [getSectorTrailerAddress](#) (unsigned char sector)

Additional Inherited Members

4.33.1 Detailed Description

Definition at line 16 of file [TagMF1S503x.h](#).

4.33.2 Member Enumeration Documentation

4.33.2.1 enum TagMF1S503x::Permission

Enumerator

LEVEL_0
LEVEL_1
LEVEL_2
LEVEL_3
LEVEL_4
LEVEL_5
LEVEL_6
LEVEL_7

Definition at line 20 of file [TagMF1S503x.h](#).

4.33.3 Constructor & Destructor Documentation

4.33.3.1 TagMF1S503x::TagMF1S503x ([Reader](#) * reader)

Definition at line 4 of file [TagMF1S503x.cpp](#).

4.33.4 Member Function Documentation

4.33.4.1 `unsigned char TagMF1S503x::addressToSector (unsigned char address)` `[virtual]`

Implements [Tag](#).

Definition at line 19 of file [TagMF1S503x.cpp](#).

4.33.4.2 `unsigned char TagMF1S503x::getSectorSize (unsigned char sector)` `[virtual]`

Implements [Tag](#).

Definition at line 15 of file [TagMF1S503x.cpp](#).

4.33.4.3 `unsigned char TagMF1S503x::getSectorTrailerAddress (unsigned char sector)` `[virtual]`

Implements [Tag](#).

Definition at line 27 of file [TagMF1S503x.cpp](#).

4.33.4.4 `unsigned char TagMF1S503x::isAddressSectorTrailer (unsigned char address)` `[virtual]`

Implements [Tag](#).

Definition at line 23 of file [TagMF1S503x.cpp](#).

4.33.4.5 `bool TagMF1S503x::writeBlock (unsigned char address, unsigned char * buf)` `[virtual]`

Reimplemented from [Tag](#).

Definition at line 8 of file [TagMF1S503x.cpp](#).

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

- [TagMF1S503x.h](#)
- [TagMF1S503x.cpp](#)

4.34 ReaderMFRC522::TX_ASKbits Union Reference

```
#include <ReaderMFRC522.h>
```

Public Attributes

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

4.34.1 Detailed Description

TX_ASK register.

Controls transmit modulation settings.

Definition at line 897 of file [ReaderMFRC522.h](#).

4.34.2 Member Data Documentation

4.34.2.1 struct { ... }

4.34.2.2 unsigned ReaderMFRC522::TX_ASKbits::char

Definition at line 902 of file [ReaderMFRC522.h](#).

4.34.2.3 unsigned char ReaderMFRC522::TX_ASKbits::FORCE_100_ASK

Definition at line 905 of file [ReaderMFRC522.h](#).

4.34.2.4 unsigned char ReaderMFRC522::TX_ASKbits::value

Definition at line 910 of file [ReaderMFRC522.h](#).

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

- [ReaderMFRC522.h](#)

4.35 ReaderMFRC522::TX_CONTROLbits Union Reference

```
#include <ReaderMFRC522.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

4.35.1 Detailed Description

TX_CONTROL register.

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

Definition at line 842 of file [ReaderMFRC522.h](#).

4.35.2 Member Data Documentation

4.35.2.1 struct { ... }

4.35.2.2 struct { ... }

4.35.2.3 unsigned ReaderMFRC522::TX_CONTROLbits::char

Definition at line 853 of file [ReaderMFRC522.h](#).

4.35.2.4 unsigned char ReaderMFRC522::TX_CONTROLbits::INV_TX1_RF_OFF

Definition at line 860 of file [ReaderMFRC522.h](#).

4.35.2.5 unsigned char ReaderMFRC522::TX_CONTROLbits::INV_TX1_RF_ON

Definition at line 866 of file [ReaderMFRC522.h](#).

4.35.2.6 unsigned char ReaderMFRC522::TX_CONTROLbits::INV_TX2_RF_OFF

Definition at line 863 of file [ReaderMFRC522.h](#).

4.35.2.7 unsigned char ReaderMFRC522::TX_CONTROLbits::INV_TX2_RF_ON

Definition at line 869 of file [ReaderMFRC522.h](#).

4.35.2.8 unsigned char ReaderMFRC522::TX_CONTROLbits::INV_TX_RF_OFF

Definition at line 883 of file [ReaderMFRC522.h](#).

4.35.2.9 unsigned char ReaderMFRC522::TX_CONTROLbits::INV_TX_RF_ON

Definition at line 887 of file [ReaderMFRC522.h](#).

4.35.2.10 unsigned char ReaderMFRC522::TX_CONTROLbits::TX1_RF_EN

Definition at line 847 of file [ReaderMFRC522.h](#).

4.35.2.11 unsigned char ReaderMFRC522::TX_CONTROLbits::TX2_CW

Definition at line 857 of file [ReaderMFRC522.h](#).

4.35.2.12 unsigned char ReaderMFRC522::TX_CONTROLbits::TX2_RF_EN

Definition at line 850 of file [ReaderMFRC522.h](#).

4.35.2.13 unsigned char ReaderMFRC522::TX_CONTROLbits::TX_RF_EN

Definition at line 876 of file [ReaderMFRC522.h](#).

4.35.2.14 unsigned char ReaderMFRC522::TX_CONTROLbits::value

Definition at line 889 of file [ReaderMFRC522.h](#).

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

- [ReaderMFRC522.h](#)

4.36 ReaderMFRC522::TX_MODEbits Union Reference

```
#include <ReaderMFRC522.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

4.36.1 Detailed Description

TX_MODE register.

Defines the data rate during transmission.

Definition at line 774 of file [ReaderMFRC522.h](#).

4.36.2 Member Data Documentation

4.36.2.1 struct { ... }

4.36.2.2 unsigned ReaderMFRC522::TX_MODEbits::char

Definition at line 779 of file [ReaderMFRC522.h](#).

4.36.2.3 unsigned char ReaderMFRC522::TX_MODEbits::INV_MOD

Definition at line 782 of file [ReaderMFRC522.h](#).

4.36.2.4 unsigned char ReaderMFRC522::TX_MODEbits::TX_CRC_EN

Definition at line 793 of file [ReaderMFRC522.h](#).

4.36.2.5 unsigned char ReaderMFRC522::TX_MODEbits::TX_SPEED

Definition at line 789 of file [ReaderMFRC522.h](#).

4.36.2.6 unsigned char ReaderMFRC522::TX_MODEbits::value

Definition at line 795 of file [ReaderMFRC522.h](#).

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

- [ReaderMFRC522.h](#)

4.37 ReaderMFRC522::TX_SELbits Union Reference

```
#include <ReaderMFRC522.h>
```

Public Attributes

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

4.37.1 Detailed Description

TX_SEL register.

Selects the internal sources for the analog module.

Definition at line 918 of file [ReaderMFRC522.h](#).

4.37.2 Member Data Documentation

4.37.2.1 struct { ... }

4.37.2.2 unsigned ReaderMFRC522::TX_SELbits::char

Definition at line 939 of file [ReaderMFRC522.h](#).

4.37.2.3 unsigned char ReaderMFRC522::TX_SELbits::MF_OUT_SEL

Definition at line 932 of file [ReaderMFRC522.h](#).

4.37.2.4 unsigned char ReaderMFRC522::TX_SELbits::value

Definition at line 944 of file [ReaderMFRC522.h](#).

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

- [ReaderMFRC522.h](#)

4.38 Tag::Uid Struct Reference

```
#include <Tag.h>
```

Public Attributes

- unsigned char [size](#)
- unsigned char [uid](#) [10]
- unsigned char [sak](#)

4.38.1 Detailed Description

Definition at line 98 of file [Tag.h](#).

4.38.2 Member Data Documentation

4.38.2.1 unsigned char Tag::Uid::sak

Definition at line 106 of file [Tag.h](#).

4.38.2.2 unsigned char Tag::Uid::size

Definition at line 101 of file [Tag.h](#).

4.38.2.3 unsigned char Tag::Uid::uid[10]

Definition at line 103 of file [Tag.h](#).

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

- [Tag.h](#)

4.39 ReaderMFRC522::VERSIONbits Union Reference

```
#include <ReaderMFRC522.h>
```

Public Attributes

- struct {
 unsigned char [VERSION](#):4
 unsigned char [CHIPTYPE](#):4
};
- unsigned char [value](#)

4.39.1 Detailed Description

VERSION register.

Shows the MFRC522 software version.

Definition at line [1223](#) of file [ReaderMFRC522.h](#).

4.39.2 Member Data Documentation

4.39.2.1 struct { ... }

4.39.2.2 unsigned char ReaderMFRC522::VERSIONbits::CHIPTYPE

Definition at line [1231](#) of file [ReaderMFRC522.h](#).

4.39.2.3 unsigned char ReaderMFRC522::VERSIONbits::value

Definition at line [1233](#) of file [ReaderMFRC522.h](#).

4.39.2.4 unsigned char ReaderMFRC522::VERSIONbits::VERSION

Definition at line [1228](#) of file [ReaderMFRC522.h](#).

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

- [ReaderMFRC522.h](#)

4.40 ReaderMFRC522::WATER_LEVELbits Union Reference

```
#include <ReaderMFRC522.h>
```

Public Attributes

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

4.40.1 Detailed Description

WATER_LEVEL register.

Defines the level for FIFO under- and overflow warning.

Definition at line 623 of file [ReaderMFRC522.h](#).

4.40.2 Member Data Documentation

4.40.2.1 struct { ... }

4.40.2.2 unsigned ReaderMFRC522::WATER_LEVELbits::char

Definition at line 635 of file [ReaderMFRC522.h](#).

4.40.2.3 unsigned char ReaderMFRC522::WATER_LEVELbits::value

Definition at line 637 of file [ReaderMFRC522.h](#).

4.40.2.4 unsigned char ReaderMFRC522::WATER_LEVELbits::WATER_LEVEL

Definition at line 632 of file [ReaderMFRC522.h](#).

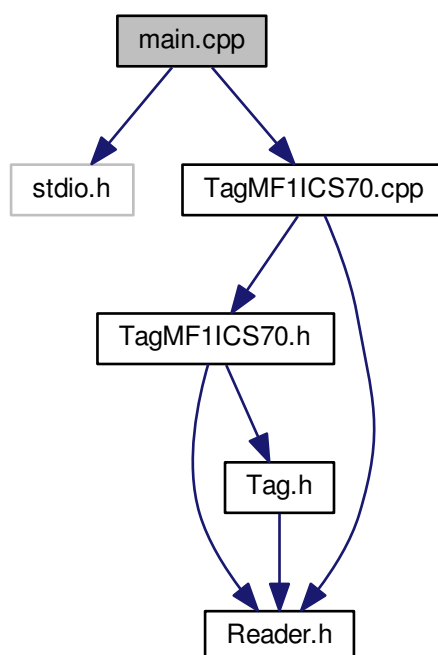
The documentation for this union was generated from the following file:

- [ReaderMFRC522.h](#)

5 File Documentation

5.1 main.cpp File Reference

```
#include <stdio.h>
#include "TagMF1ICS70.cpp"
Include dependency graph for main.cpp:
```



Functions

- `int main ()`

5.1.1 Function Documentation

5.1.1.1 `int main ()`

Definition at line 4 of file `main.cpp`.

5.2 main.cpp

```
00001 #include <stdio.h>
00002 #include "TagMF1ICS70.cpp"
00003
00004 int main() {
00005     TagMF1ICS70 tag;
00006
00007     // getSectorSize
00008
00009     if (tag.getSectorSize(0) != 4) {
00010         printf("error in getSectorSize#0\n");
00011     }
00012
00013     if (tag.getSectorSize(31) != 4) {
00014         printf("error in getSectorSize#1\n");
00015     }
00016
00017     if (tag.getSectorSize(32) != 16) {
00018         printf("error in getSectorSize#2\n");
00019     }
00020
00021     if (tag.getSectorSize(40) != 16) {
00022         printf("error in getSectorSize#3\n");
00023     }
00024
00025     // addressToSector
00026
00027     if (tag.addressToSector(10) != 2) {
00028         printf("error in addressToSector#0\n");
00029     }
00030
00031     if (tag.addressToSector(4) != 1) {
00032         printf("error in addressToSector#1\n");
00033     }
00034
00035     if (tag.addressToSector(0) != 0) {
00036         printf("error in addressToSector#2\n");
00037     }
00038
00039     if (tag.addressToSector(12) != 3) {
00040         printf("error in addressToSector#3\n");
00041     }
00042
00043     if (tag.addressToSector(127) != 31) {
00044         printf("error in addressToSector#4\n");
00045     }
00046
00047     if (tag.addressToSector(128) != 32) {
00048         printf("error in addressToSector#5\n");
00049     }
00050
00051     if (tag.addressToSector(143) != 32) {
00052         printf("error in addressToSector#6\n");
00053     }
00054
00055     if (tag.addressToSector(144) != 33) {
00056         printf("error in addressToSector#7\n");
00057     }
00058
00059     if (tag.addressToSector(255) != 39) {
00060         printf("error in addressToSector#8\n");
00061     }
00062
00063     // isAddressSectorTrailer
00064
00065     if (tag.isAddressSectorTrailer(12)) {
00066         printf("error in isAddressSectorTrailer#0\n");
00067     }
00068
00069     if (!tag.isAddressSectorTrailer(3)) {
00070         printf("error in isAddressSectorTrailer#1\n");
00071     }
00072
00073     if (!tag.isAddressSectorTrailer(19)) {
00074         printf("error in isAddressSectorTrailer#2\n");
00075     }
00076
00077     if (tag.isAddressSectorTrailer(22)) {
00078         printf("error in isAddressSectorTrailer#3\n");
00079     }
00080
00081     if (!tag.isAddressSectorTrailer(127)) {
00082         printf("error in isAddressSectorTrailer#4\n");
00083     }
00084 }
```

```

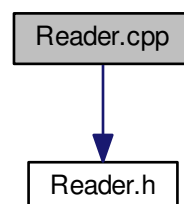
00085
00086     if (!tag.isAddressSectorTrailer(255)) {
00087         printf("error in isAddressSectorTrailer#5\n");
00088     }
00089
00090     if (tag.isAddressSectorTrailer(254-16)) {
00091         printf("error in isAddressSectorTrailer#6\n");
00092     }
00093
00094     // getSectorTrailerAddress
00095
00096     if (tag.getSectorTrailerAddress(31) != 127) {
00097         printf("error in getSectorTrailerAddress#4\n");
00098     }
00099
00100     if (tag.getSectorTrailerAddress(32) != 143) {
00101         printf("error in getSectorTrailerAddress#5, got %d expect %d\n", tag.
getSectorTrailerAddress(32), 143);
00102     }
00103
00104     if (tag.getSectorTrailerAddress(39) != 255) {
00105         printf("error in getSectorTrailerAddress#6, got %d expect %d\n", tag.
getSectorTrailerAddress(39), 255);
00106     }
00107
00108     if (tag.getSectorTrailerAddress(1) != 7) {
00109         printf("error in getSectorTrailerAddress#1\n");
00110     }
00111
00112     if (tag.getSectorTrailerAddress(2) != 11) {
00113         printf("error in getSectorTrailerAddress#2\n");
00114     }
00115
00116     if (tag.getSectorTrailerAddress(3) != 15) {
00117         printf("error in getSectorTrailerAddress#3\n");
00118     }
00119
00120     if (tag.getSectorTrailerAddress(31) != 127) {
00121         printf("error in getSectorTrailerAddress#5\n");
00122     }
00123
00124     if (tag.getSectorTrailerAddress(32) != 143) {
00125         printf("error in getSectorTrailerAddress#6\n");
00126     }
00127
00128     if (tag.getSectorTrailerAddress(39) != 255) {
00129         printf("error in getSectorTrailerAddress#7\n");
00130     }
00131
00132     return 0;
00133 }

```

5.3 Reader.cpp File Reference

#include <Reader.h>

Include dependency graph for Reader.cpp:



5.4 Reader.cpp

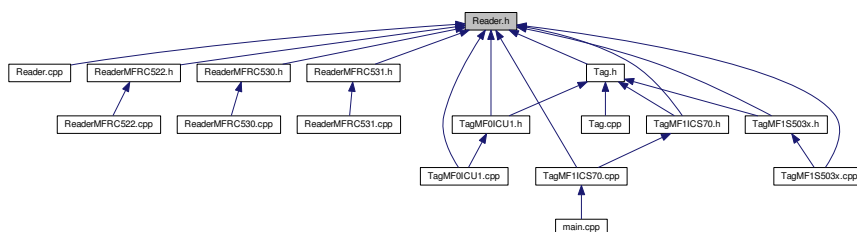
```

00001 #include <Reader.h>
00002
00003 Reader::Reader() : lastError(NO_ERROR) {
00004 }
00005
00006 Reader::~Reader() {
00007 }
00008
00009 unsigned char Reader::getLastError() {
00010     return lastError;
00011 }
00012
00013 void Reader::clearLastError() {
00014     lastError = NO_ERROR;
00015 }

```

5.5 Reader.h File Reference

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



Classes

- class [Reader](#)

Macros

- `#define` [READER_DEFAULT_TIMEOUT](#) 300

5.5.1 Macro Definition Documentation

5.5.1.1 `#define` [READER_DEFAULT_TIMEOUT](#) 300

Arduino - Radio Frequency Identification.

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 10 of file [Reader.h](#).

5.6 Reader.h

```

00001
00007 #ifndef __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_READER_H__
00008 #define __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_READER_H__ 1
00009
00010 #define READER_DEFAULT_TIMEOUT          300
00011
00012 class Reader {
00013
00014 public:
00015
00016     enum Error {
00017         NO_ERROR = 0x00,
00018         GENERAL_ERROR = 0x01,
00019         TIMEOUT_ERROR = 0x02,
00020         COMMUNICATION_ERROR = 0x03,
00021         CRC_ERROR = 0x04,
00022         NACK = 0x05,
00023         COLLISION_ERROR = 0x06
00024     };
00025
00026     Reader();
00027
00028     virtual ~Reader();
00029
00030     virtual inline void sendCommand(unsigned char command) = 0;
00031
00032     virtual void softReset() = 0;
00033
00034     virtual void setAntennaOn() = 0;
00035
00036     virtual void setAntennaOff() = 0;
00037
00038     virtual void configureTimer(unsigned int prescaler, unsigned int reload, bool autoStart,
00039 bool autoRestart) = 0;
00039
00040     virtual void startTimer() = 0;
00041
00042     virtual void stopTimer() = 0;
00043
00044     virtual void enableInterrupt(unsigned int interrupt) = 0;
00045
00046     virtual void disableInterrupt(unsigned int interrupt) = 0;
00047
00048     virtual void clearInterrupt(unsigned int interrupt) = 0;
00049
00050     virtual void flushQueue() = 0;
00051
00052     virtual void setWaterLevel(unsigned char level) = 0;
00053
00054     virtual int generateRandomId(unsigned char *buf) = 0;
00055
00056     virtual int communicate(unsigned char command, unsigned char *send, unsigned char *receive,
00057 unsigned char sendLen, bool checkCrc) = 0;
00058
00058     virtual int communicate(unsigned char command, unsigned char *send, unsigned char *receive,
00059 unsigned char sendLen) = 0;
00059
00060     virtual int tranceive(unsigned char *send, unsigned char *receive, unsigned char sendLen, bool
00061 checkCrc) = 0;
00061
00062     virtual int tranceive(unsigned char *send, unsigned char *receive, unsigned char sendLen) = 0;
00063
00064     virtual int authenticate(unsigned char *send) = 0;
00065
00066     virtual void turnOffEncryption() = 0;
00067
00068     virtual unsigned int calculateCrc(unsigned char *buf, unsigned char len) = 0;
00069
00070     virtual void calculateCrc(unsigned char *buf, unsigned char len, unsigned char *dst) = 0;
00071
00072     virtual bool waitForRegisterBits(unsigned char reg, unsigned char mask, unsigned
00073 long timeout) = 0;
00073
00074     virtual bool waitForRegisterBits(unsigned char reg, unsigned char mask) = 0;
00075
00076     virtual bool performSelfTest() = 0;
00077
00078     virtual void setBitFraming(unsigned char rxAlign, unsigned char txLastBits) = 0;
00079
00080     virtual unsigned char getCollisionPosition() = 0;
00081
00082     virtual void setupForAnticollision() = 0;
00083
00084     unsigned char getLastError();

```



```

00085
00086     void clearLastError();
00087
00088     virtual bool hasValidCrc(unsigned char *buf, unsigned char len) = 0;
00089
00090 protected:
00091
00092     Error lastError;
00093 };
00094
00095 #endif // __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_READER_H__

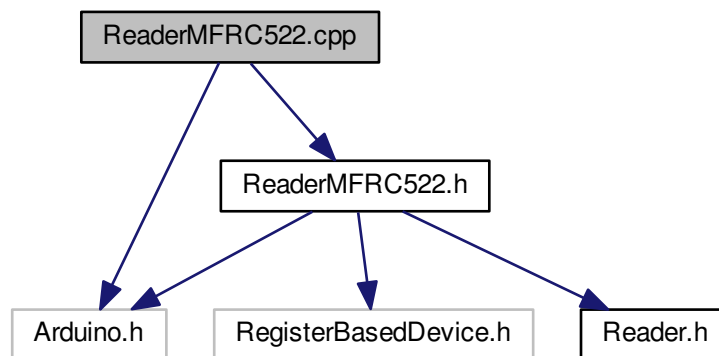
```

5.7 ReaderMFRC522.cpp File Reference

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

```
#include <ReaderMFRC522.h>
```

Include dependency graph for ReaderMFRC522.cpp:



5.8 ReaderMFRC522.cpp

```

00001 #include <Arduino.h>
00002 #include <ReaderMFRC522.h>
00003
00004 ReaderMFRC522::ReaderMFRC522(RegisterBasedDevice *device, unsigned char
    resetPin)
00005     : Reader(), device(device), resetPin(resetPin) {
00006     pinMode(resetPin, OUTPUT);
00007     digitalWrite(resetPin, LOW);
00008 }
00009
00010 ReaderMFRC522::~ReaderMFRC522() {
00011 }
00012
00013 inline void ReaderMFRC522::sendCommand(unsigned char command) {
00014     writeRegister(COMMAND, command);
00015 }
00016
00017 void ReaderMFRC522::initialize() {
00018     if (digitalRead(resetPin) == LOW) {
00019         digitalWrite(resetPin, HIGH);
00020         delay(50);
00021     } else {
00022         softReset();
00023     }
00024     clearRegisterBits(AUTO_TEST, AUTO_TEST_ENABLE);
00025
00026     // 100% ASK

```

```

00027     writeRegister(TX_ASK, 0x40);
00028
00029     // CRC Initial value 0x6363
00030     writeRegister(MODE, 0x3d);
00031
00032     // Open the antenna
00033     setAntennaOn();
00034 }
00035
00036 void ReaderMFRC522::softReset() {
00037     sendCommand(SOFT_RESET);
00038 }
00039
00040 void ReaderMFRC522::setAntennaOn() {
00041     setRegisterBits(TX_CONTROL, TX_CONTROL_TX_RF_EN);
00042 }
00043
00044 void ReaderMFRC522::setAntennaOff() {
00045     clearRegisterBits(TX_CONTROL, TX_CONTROL_TX_RF_EN);
00046 }
00047
00048 int ReaderMFRC522::readRegisterBlock(unsigned char reg, unsigned char *buf,
    unsigned char len) {
00049
00050     // MSB == 1 is for reading. LSB is not used in address.
00051     return device->readRegisterBlock((reg << 1) & 0x7e | 0x80, buf, len);
00052 }
00053
00054 int ReaderMFRC522::readRegisterBlock(unsigned char reg, unsigned char *buf,
    unsigned char len, unsigned char rxAlign) {
00055     int blockSize = readRegisterBlock(reg, buf, len);
00056     if (blockSize > 0 && rxAlign > 0) {
00057         rxAlign &= 0x07;
00058         unsigned char mask = 0;
00059         for (unsigned char i = rxAlign; i <= 7; i++) {
00060             mask |= (1 << i);
00061         }
00062
00063         // Only bit positions rxAlign..7 in buf[0] are updated.
00064         buf[0] = (buf[0] & ~mask) | (buf[0] & mask);
00065     }
00066     return blockSize;
00067 }
00068
00069 unsigned char ReaderMFRC522::writeRegisterBlock(unsigned char reg,
    unsigned char *buf, unsigned char len) {
00070
00071     // MSB == 0 is for writing. LSB is not used in address.
00072     return device->writeRegisterBlock((reg << 1) & 0x7e, buf, len);
00073 }
00074
00075 void ReaderMFRC522::configureTimer(unsigned int prescaler, unsigned int reload
    , bool autoStart, bool autoRestart) {
00076     T_MODEbits timerMode;
00077     timerMode.value = readRegister(T_MODE);
00078     timerMode.T_PRESCALER_HI = (prescaler >> 8) & 0x0f;
00079     timerMode.T_AUTO = autoStart;
00080     timerMode.T_GATED = (unsigned char) 0;
00081     timerMode.T_AUTO_RESTART = autoRestart;
00082     writeRegister(T_MODE, timerMode.value);
00083     writeRegister(T_PRESCALER_LOW, prescaler & 0xff);
00084     writeRegister(T_RELOAD_HIGH, (reload >> 8) & 0xff);
00085     writeRegister(T_RELOAD_LOW, reload & 0xff);
00086 }
00087
00088 void ReaderMFRC522::startTimer() {
00089     setRegisterBits(CONTROL, CONTROL_T_START_NOW);
00090 }
00091
00092 void ReaderMFRC522::stopTimer() {
00093     setRegisterBits(CONTROL, CONTROL_T_STOP_NOW);
00094 }
00095
00096 void ReaderMFRC522::enableInterrupt(unsigned int interrupt) {
00097     setRegisterBits(MFR522_INT_TO_EN_REG(interrupt),
    MFR522_INT_TO_EN_MASK(interrupt));
00098 }
00099
00100 void ReaderMFRC522::disableInterrupt(unsigned int interrupt) {
00101     clearRegisterBits(MFR522_INT_TO_EN_REG(interrupt),
    MFR522_INT_TO_EN_MASK(interrupt));
00102 }
00103
00104 void ReaderMFRC522::clearInterrupt(unsigned int interrupt) {
00105
00106     // 0x7f: first bit 0 indicates that the marked bits in the register are cleared
00107     configureRegisterBits(MFR522_INT_TO_IRQ_REG(interrupt), (

```

```

    MFR522_INT_TO_IRQ_MASK(interrupt)) | 0x80, 0x7f);
00108 }
00109
00110 void ReaderMFRC522::flushQueue() {
00111     setRegisterBits(FIFO_LEVEL, FIFO_LEVEL_FLUSH_BUFFER);
00112 }
00113
00114 void ReaderMFRC522::setWaterLevel(unsigned char level) {
00115     writeRegister(WATER_LEVEL, WATER_LEVEL_WATER_LEVEL & level);
00116 }
00117
00118 int ReaderMFRC522::generateRandomId(unsigned char *buf) {
00119     // Stop any active command.
00120     sendCommand(IDLE);
00121
00122     // Clear all seven interrupt request bits
00123     clearInterrupt(COM_ALL_IRQ);
00124
00125     // FlushBuffer = 1, FIFO initialization
00126     flushQueue();
00127
00128     // Send command
00129     sendCommand(GENERATE_RANDOM_ID);
00130
00131     // Wait for command to complete.
00132     waitForRegisterBits(COM_IRQ, COM_IRQ_IDLE_IRQ);
00133
00134     // FlushBuffer = 1, FIFO initialization
00135     flushQueue();
00136
00137     // Transfers 25 bytes from the internal buffer to the FIFO buffer.
00138     sendCommand(MEM);
00139
00140     // Wait for command to complete.
00141     waitForRegisterBits(COM_IRQ, COM_IRQ_IDLE_IRQ);
00142     sendCommand(IDLE);
00143     return readRegisterBlock(FIFO_DATA, buf, 10);
00144 }
00145
00146 int ReaderMFRC522::tranceive(unsigned char *send, unsigned char *receive, unsigned
char sendLen, bool checkCrc) {
00147     return communicate(TRANSCIVE, send, receive, sendLen, checkCrc);
00148 }
00149
00150 inline int ReaderMFRC522::tranceive(unsigned char *send, unsigned char *receive,
unsigned char sendLen) {
00151     return tranceive(send, receive, sendLen, false);
00152 }
00153
00154 int ReaderMFRC522::communicate(unsigned char command, unsigned char *send,
unsigned char *receive, unsigned char sendLen, bool checkCrc) {
00155     unsigned char len = 0;
00156     COM_IRQbits irq;
00157     ERRORbits error;
00158     CONTROLbits control;
00159
00160     lastError = NO_ERROR;
00161
00162     // 25ms before timeout, auto start timer at the end of the transmission
00163     configureTimer(0xf9, 0x03e8, true, false);
00164
00165     // Stop any active command.
00166     sendCommand(IDLE);
00167
00168     // Clear all seven interrupt request bits
00169     clearInterrupt(COM_ALL_IRQ);
00170
00171     // FlushBuffer = 1, FIFO initialization
00172     flushQueue();
00173
00174     // Write sendData to the FIFO
00175     writeRegisterBlock(FIFO_DATA, send, sendLen);
00176
00177     // Execute the command
00178     sendCommand(command);
00179
00180     if (command == TRANSCIVE) {
00181         // StartSend=1, transmission of data starts
00182         setRegisterBits(BIT_FRAMING, BIT_FRAMING_START_SEND);
00183     }
00184
00185     // Wait for the command to complete.
00186     // If timer was configured and T_AUTO flag is active in T_MODE register,
00187     // timer will start automatically after all data is transmitted.
00188
00189
00190

```

```

00191 // See: configureTimer method
00192 do {
00193     irq.value = readRegister(COM_IRQ);
00194
00195     // Timer interrupt - nothing received
00196     if (irq.TIMER_IRQ) {
00197         lastError = TIMEOUT_ERROR;
00198         return -1;
00199     }
00200 } while (!irq.IDLE_IRQ && !irq.RX_IRQ);
00201
00202 // Stop now if any errors except collisions were detected.
00203 // ErrorReg[7..0] bits are: WrErr TempErr reserved BufferOvfl CollErr CRCErr ParityErr ProtocolErr
00204 error.value = readRegister(ERROR);
00205
00206 if (error.COLL_ERR) {
00207     lastError = COLLISION_ERROR;
00208     return -1;
00209 }
00210
00211 if (error.BUFFER_OVFL || error.PARITY_ERR || error.
00212     PROTOCOL_ERR) {
00213     lastError = COMMUNICATION_ERROR;
00214     return -1;
00215 }
00216 len = readRegister(FIFO_LEVEL);
00217 control.value = readRegister(CONTROL);
00218
00219 // Get received data from FIFO
00220 len = readRegisterBlock(FIFO_DATA, receive, len);
00221
00222 // In this case a MIFARE Classic NAK is not OK.
00223 if (len == 1 && control.RX_LAST_BITS == 4 && (receive[0] != SAK && receive[0] !=
00224     ACK)) {
00225     lastError = NACK;
00226     return -1;
00227 }
00228 // We need at least the CRC_A value and all 8 bits of the last byte must be received.
00229 // NOTE: casting (unsigned char) len is fine here, len > 0 and is less than FIFO size: 64
00230 // NOTE: control.RX_LAST_BITS = 0 means 8 bits.
00231 if (checkCrc && (len < 2 || control.RX_LAST_BITS != 0 || !
00232     hasValidCrc(receive, (unsigned char) len))) {
00233     lastError = CRC_ERROR;
00234     return -1;
00235 }
00236 return len;
00237 }
00238
00239 inline int ReaderMFRC522::communicate(unsigned char command, unsigned char *send,
00240     unsigned char *receive, unsigned char sendLen) {
00241     return communicate(command, send, receive, sendLen, false);
00242 }
00243 int ReaderMFRC522::authenticate(unsigned char *send) {
00244     unsigned char receive;
00245     return communicate(MF_AUTHENT, send, &receive, 12);
00246 }
00247
00248 void ReaderMFRC522::turnOffEncryption() {
00249     clearRegisterBits(STATUS2, STATUS2_MF_CRYPT01_ON);
00250 }
00251
00252 bool ReaderMFRC522::hasValidCrc(unsigned char *buf, unsigned char len) {
00253     if (len <= 2) {
00254         return false;
00255     }
00256     unsigned char crc[2];
00257     calculateCrc(buf, len - 2, crc);
00258     return (buf[len - 2] == crc[0] && (buf[len - 1] == crc[1]));
00259 }
00260
00261 unsigned int ReaderMFRC522::calculateCrc(unsigned char *buf, unsigned char len)
00262 {
00263     unsigned int dst;
00264     calculateCrc(buf, len, (unsigned char *) &dst);
00265     return dst;
00266 }
00267 void ReaderMFRC522::calculateCrc(unsigned char *buf, unsigned char len, unsigned
00268     char *dst) {
00269     // Stop any active command.
00270     sendCommand(IDLE);
00271

```

```

00272 // Clear all seven interrupt request bits
00273 clearInterrupt(DIV_ALL_IRQ);
00274
00275 // FlushBuffer = 1, FIFO initialization
00276 flushQueue();
00277
00278 // Write sendData to the FIFO
00279 writeRegisterBlock(FIFO_DATA, buf, len);
00280
00281 // Start the calculation
00282 sendCommand(CALC_CRC);
00283
00284 // Wait for the CRC calculation to complete.
00285 waitForRegisterBits(DIV_IRQ, DIV_IRQ_CRC_IRQ);
00286
00287 // Stop calculating CRC for new content in the FIFO.
00288 sendCommand(IDLE);
00289
00290 if (dst != NULL) {
00291     dst[0] = readRegister(CRC_RESULT_LOW);
00292     dst[1] = readRegister(CRC_RESULT_HIGH);
00293 }
00294 }
00295
00296 bool ReaderMFRC522::waitForRegisterBits(unsigned char reg, unsigned char
mask, unsigned long timeout) {
00297     unsigned char v;
00298     unsigned long start = millis();
00299     do {
00300         v = readRegister(reg);
00301     } while (!(v & mask) && start + timeout > millis());
00302     return (v & mask) > 0;
00303 }
00304
00305 inline bool ReaderMFRC522::waitForRegisterBits(unsigned char reg,
unsigned char mask) {
00306     return waitForRegisterBits(reg, mask,
READER_DEFAULT_TIMEOUT);
00307 }
00308
00309 bool ReaderMFRC522::performSelfTest() {
00310     unsigned char *firmwareReference;
00311     unsigned char buffer[64] = { 0 };
00312     writeRegister(AUTO_TEST, 0x00);
00313     softReset();
00314     flushQueue();
00315     writeRegisterBlock(FIFO_DATA, buffer, 25);
00316     sendCommand(MEM);
00317     writeRegister(AUTO_TEST, AUTO_TEST_ENABLE);
00318     writeRegister(FIFO_DATA, 0x00);
00319     sendCommand(CALC_CRC);
00320     waitForRegisterBits(DIV_IRQ, DIV_IRQ_CRC_IRQ, 100);
00321     readRegisterBlock(FIFO_DATA, buffer, 64);
00322     switch (getVersion()) {
00323     case CLONE:
00324         firmwareReference = (unsigned char *) FM17522_FIRMWARE_REFERENCE;
00325         break;
00326     case V0_0:
00327         firmwareReference = (unsigned char *) MFRC522_FIRMWARE_REFERENCE_V0_0;
00328         break;
00329     case V1_0:
00330         firmwareReference = (unsigned char *) MFRC522_FIRMWARE_REFERENCE_V1_0;
00331         break;
00332     case V2_0:
00333         firmwareReference = (unsigned char *) MFRC522_FIRMWARE_REFERENCE_V2_0;
00334         break;
00335     default:
00336         return false;
00337     }
00338     for (unsigned char i = 0; i < 64; i++) {
00339         if (buffer[i] != pgm_read_byte(&(firmwareReference[i]))) {
00340             return false;
00341         }
00342     }
00343     return true;
00344 }
00345
00346 void ReaderMFRC522::setBitFraming(unsigned char rxAlign, unsigned char
txLastBits) {
00347     BIT_FRAMINGbits f;
00348     f.value = readRegister(BIT_FRAMING);
00349     f.RX_ALIGN = rxAlign;
00350     f.TX_LAST_BITS = txLastBits;
00351     writeRegister(BIT_FRAMING, f.value);
00352 }
00353
00354 unsigned char ReaderMFRC522::getCollisionPosition() {

```

```

00355     COLLbits coll;
00356     coll.value = readRegister(COLL);
00357     return coll.COLL_POS > 0 ? coll.COLL_POS : 32;
00358 }
00359
00360 void ReaderMFRC522::setUpForAnticollision() {
00361     clearRegisterBits(COLL, COLL_VALUES_AFTER_COLL);
00362 }
00363
00364 ReaderMFRC522::Version ReaderMFRC522::getVersion() {
00365     return (Version) readRegister(VERSION);
00366 }

```

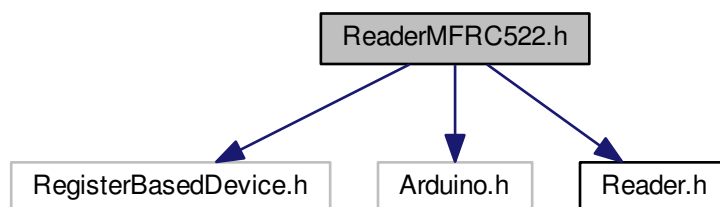
5.9 ReaderMFRC522.h File Reference

```

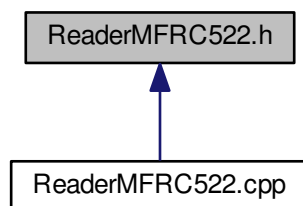
#include <RegisterBasedDevice.h>
#include <Arduino.h>
#include <Reader.h>

```

Include dependency graph for ReaderMFRC522.h:



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



Classes

- class [ReaderMFRC522](#)
- union [ReaderMFRC522::COMMANDbits](#)
- union [ReaderMFRC522::COM_I_ENbits](#)
- union [ReaderMFRC522::DIV_I_ENbits](#)

- union [ReaderMFRC522::COM_IRQbits](#)
- union [ReaderMFRC522::DIV_IRQbits](#)
- union [ReaderMFRC522::ERRORbits](#)
- union [ReaderMFRC522::STATUS1bits](#)
- union [ReaderMFRC522::STATUS2bits](#)
- union [ReaderMFRC522::FIFO_LEVELbits](#)
- union [ReaderMFRC522::WATER_LEVELbits](#)
- union [ReaderMFRC522::CONTROLbits](#)
- union [ReaderMFRC522::BIT_FRAMINGbits](#)
- union [ReaderMFRC522::COLLbits](#)
- union [ReaderMFRC522::MODEbits](#)
- union [ReaderMFRC522::TX_MODEbits](#)
- union [ReaderMFRC522::RX_MODEbits](#)
- union [ReaderMFRC522::TX_CONTROLbits](#)
- union [ReaderMFRC522::TX_ASKbits](#)
- union [ReaderMFRC522::TX_SELbits](#)
- union [ReaderMFRC522::RX_SELbits](#)
- union [ReaderMFRC522::RX_THRESHOLDbits](#)
- union [ReaderMFRC522::DEMODbits](#)
- union [ReaderMFRC522::MF_TXbits](#)
- union [ReaderMFRC522::MF_RXbits](#)
- union [ReaderMFRC522::SERIAL_SPEEDbits](#)
- union [ReaderMFRC522::RF_CFGbits](#)
- union [ReaderMFRC522::GS_Nbits](#)
- union [ReaderMFRC522::CW_GS_Pbits](#)
- union [ReaderMFRC522::MOD_GS_Pbits](#)
- union [ReaderMFRC522::T_MODEbits](#)
- union [ReaderMFRC522::VERSIONbits](#)

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](#)

5.9.1 Macro Definition Documentation

- 5.9.1.1 `#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 [ReaderMFRC522.h](#).

5.9.1.2 `#define MFR522_INT_TO_EN_REG(i)(i > COM_ALL_IRQ) ? DIV_I_EN : COM_I_EN`

Definition at line 15 of file [ReaderMFRC522.h](#).

5.9.1.3 `#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 [ReaderMFRC522.h](#).

5.9.1.4 `#define MFR522_INT_TO_IRQ_REG(i)(i > COM_ALL_IRQ) ? DIV_IRQ : COM_IRQ`

Definition at line 18 of file [ReaderMFRC522.h](#).

5.9.1.5 `#define MFRC522_DEFAULT_TIMEOUT 100`

Arduino - Radio Frequency Identification MFRC522.

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 13 of file [ReaderMFRC522.h](#).

5.9.2 Variable Documentation

5.9.2.1 `const byte FM17522_FIRMWARE_REFERENCE [] PROGMEM`

Initial value:

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

Definition at line 23 of file [ReaderMFRC522.h](#).

5.10 ReaderMFRC522.h

```

00001
00006 #ifndef __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_READER_MFRC522_H__
00007 #define __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_READER_MFRC522_H__ 1
00008
00009 #include <RegisterBasedDevice.h>
00010 #include <Arduino.h>
00011 #include <Reader.h>
00012
00013 #define MFRC522_DEFAULT_TIMEOUT      100
00014
00015 #define MFR522_INT_TO_EN_REG(i)      (i > COM_ALL_IRQ) ? DIV_I_EN : COM_I_EN
00016 #define MFR522_INT_TO_EN_MASK(i)    (i > COM_ALL_IRQ) ? (i >> 8) & DIV_I_EN_INTERRUPT_EN : i &
    COM_I_EN_INTERRUPT_EN
00017
00018 #define MFR522_INT_TO_IRQ_REG(i)     (i > COM_ALL_IRQ) ? DIV_IRQ : COM_IRQ
00019 #define MFR522_INT_TO_IRQ_MASK(i)   (i > COM_ALL_IRQ) ? (i >> 8) & DIV_IRQ_ALL_IRQ : i & COM_IRQ_ALL_IRQ
00020
00021 // Version 0.0 (0x90)
00022 // Philips Semiconductors; Preliminary Specification Revision 2.0 - 01 August 2005; 16.1 self-test
00023 const unsigned char MFRC522_FIRMWARE_REFERENCE_V0_0[] PROGMEM = { 0x00, 0x87, 0x98, 0x0f, 0x49, 0xff
    , 0x07, 0x19, 0xbf, 0x22, 0x30, 0x49, 0x59,
00024     0x63, 0xad, 0xca, 0x7f, 0xe3, 0x4e, 0x03, 0x5c, 0x4e, 0x49, 0x50, 0x47, 0x9a, 0x37, 0x61, 0xe7,
    0xe2, 0xc6, 0x2e, 0x75, 0x5a, 0xed,
00025     0x04, 0x3d, 0x02, 0x4b, 0x78, 0x32, 0xff, 0x58, 0x3b, 0x7c, 0xe9, 0x00, 0x94, 0xb4, 0x4a, 0x59,
    0x5b, 0xfd, 0xc9, 0x29, 0xdf, 0x35,
00026     0x96, 0x98, 0x9e, 0x4f, 0x30, 0x32, 0x8d };
00027
00028 // Version 1.0 (0x91)
00029 // NXP Semiconductors; Rev. 3.8 - 17 September 2014; 16.1.1 self-test
00030 const byte MFRC522_FIRMWARE_REFERENCE_V1_0[] PROGMEM = { 0x00, 0xc6, 0x37, 0xd5, 0x32, 0xb7, 0x57,
    0x5c, 0xc2, 0xd8, 0x7c, 0x4d, 0xd9, 0x70,
00031     0xc7, 0x73, 0x10, 0xe6, 0xd2, 0xaa, 0x5e, 0xa1, 0x3e, 0x5a, 0x14, 0xaf, 0x30, 0x61, 0xc9, 0x70,
    0xdb, 0x2e, 0x64, 0x22, 0x72, 0xb5,
00032     0xbd, 0x65, 0xf4, 0xec, 0x22, 0xbc, 0xd3, 0x72, 0x35, 0xcd, 0xaa, 0x41, 0x1f, 0xa7, 0xf3, 0x53,
    0x14, 0xde, 0x7e, 0x02, 0xd9, 0x0f,
00033     0xb5, 0x5e, 0x25, 0x1d, 0x29, 0x79 };
00034
00035 // Version 2.0 (0x92)
00036 // NXP Semiconductors; Rev. 3.8 - 17 September 2014; 16.1.1 self-test
00037 const byte MFRC522_FIRMWARE_REFERENCE_V2_0[] PROGMEM = { 0x00, 0xeb, 0x66, 0xba, 0x57, 0xbf, 0x23,
    0x95, 0xd0, 0xe3, 0x0d, 0x3d, 0x27, 0x89,
00038     0x5c, 0xde, 0x9d, 0x3b, 0xa7, 0x00, 0x21, 0x5b, 0x89, 0x82, 0x51, 0x3a, 0xeb, 0x02, 0x0c, 0xa5,
    0x00, 0x49, 0x7c, 0x84, 0x4d, 0xb3,
00039     0xcc, 0xd2, 0x1b, 0x81, 0x5d, 0x48, 0x76, 0xd5, 0x71, 0x61, 0x21, 0xa9, 0x86, 0x96, 0x83, 0x38,
    0xcf, 0x9d, 0x5b, 0x6d, 0xdc, 0x15,
00040     0xba, 0x3e, 0x7d, 0x95, 0x3b, 0x2f };
00041
00042 // Clone
00043 // Fudan Semiconductor FM17522 (0x88)
00044 const byte FM17522_FIRMWARE_REFERENCE[] PROGMEM = { 0x00, 0xd6, 0x78, 0x8c, 0xe2, 0xaa, 0x0c, 0x18,
    0x2a, 0xb8, 0x7a, 0x7f, 0xd3, 0x6a, 0xcf,
00045     0x0b, 0xb1, 0x37, 0x63, 0x4b, 0x69, 0xae, 0x91, 0xc7, 0xc3, 0x97, 0xae, 0x77, 0xf4, 0x37, 0xd7,
    0x9b, 0x7c, 0xf5, 0x3c, 0x11, 0x8f,
00046     0x15, 0xc3, 0xd7, 0xc1, 0x5b, 0x00, 0x2a, 0xd0, 0x75, 0xde, 0x9e, 0x51, 0x64, 0xab, 0x3e, 0xe9,
    0x15, 0xb5, 0xab, 0x56, 0x9a, 0x98,
00047     0x82, 0x26, 0xea, 0x2a, 0x62 };
00048
00049 class ReaderMFRC522: public Reader, public RegisterBasedDevice {
00050
00051     static const unsigned char SAK = 0x08;
00052     static const unsigned char ACK = 0x0a;
00053
00054     RegisterBasedDevice *device;
00055
00056     unsigned char resetPin;
00057
00058 public:
00059
00060     enum Register {
00061
00062         // Starts and stops command execution
00063         COMMAND = 0x01,
00064
00065         // Enable and disable interrupt request control bits
00066         COM_I_EN = 0x02,
00067
00068         // Enable and disable interrupt request control bits
00069         DIV_I_EN = 0x03,
00070
00071         // Interrupt request bits
00072         COM_IRQ = 0x04,
00073
00074         // Interrupt request bits Table 31 on page 40
00075         DIV_IRQ = 0x05,

```

```

00076
00077 // Error bits showing the error status of the last command
00078 ERROR = 0x06,
00079
00080 // Communication status bits
00081 STATUS1 = 0x07,
00082
00083 // Receiver and transmitter status bits
00084 STATUS2 = 0x08,
00085
00086 // Input and output of 64 byte FIFO buffer
00087 FIFO_DATA = 0x09,
00088
00089 // Number of bytes stored in the FIFO buffer
00090 FIFO_LEVEL = 0x0a,
00091
00092 // Level for FIFO underflow and overflow warning
00093 WATER_LEVEL = 0x0b,
00094
00095 // Miscellaneous control registers
00096 CONTROL = 0x0c,
00097
00098 // Adjustments for bit-oriented frames
00099 BIT_FRAMING = 0x0d,
00100
00101 // Bit position of the first bit-collision detected on the RF
00102 COLL = 0x0e,
00103
00104 // Defines general modes for transmitting and receiving
00105 MODE = 0x11,
00106
00107 // Defines transmission data rate and framing
00108 TX_MODE = 0x12,
00109
00110 // Defines reception data rate and framing
00111 RX_MODE = 0x13,
00112
00113 // Controls the logical behavior of the antenna driver pins TX1 and TX2
00114 TX_CONTROL = 0x14,
00115
00116 // Controls the setting of the transmission modulation
00117 TX_ASK = 0x15,
00118
00119 // Selects the internal sources for the antenna driver
00120 TX_SEL = 0x16,
00121
00122 // Selects internal receiver settings
00123 RX_SEL = 0x17,
00124
00125 // Selects thresholds for the bit decoder
00126 RX_THRESHOLD = 0x18,
00127
00128 // Defines demodulator settings
00129 DEMOD = 0x19,
00130
00131 // Some MIFARE communication transmit parameters
00132 MF_TX = 0x1c,
00133
00134 // Controls some MIFARE communication receive parameters
00135 MF_RX = 0x1d,
00136
00137 // Selects the speed of the serial UART interface
00138 SERIAL_SPEED = 0x1f,
00139
00140 // Shows the MSB and LSB values of the CRC calculation (HIGH)
00141 CRC_RESULT_HIGH = 0x21,
00142
00143 // Shows the MSB and LSB values of the CRC calculation (LOW)
00144 CRC_RESULT_LOW = 0x22,
00145
00146 // Controls the ModWidth setting
00147 MOD_WIDTH = 0x24,
00148
00149 // Configures the receiver gain
00150 RFC_FG = 0x26,
00151
00152 // Selects the conductance of the antenna driver pins TX1 and TX2 for modulation
00153 GS_N = 0x27,
00154
00155 // The conductance of the p-driver output during periods of no modulation
00156 CW_GS_P = 0x28,
00157
00158 // Defines the conductance of the p-driver output during periods of modulation
00159 MOD_GS_P = 0x29,
00160
00161 // Defines settings for the internal timer
00162 T_MODE = 0x2a,

```

```

00163
00164 // Defines settings for the internal timer
00165 T_PRESCALER_LOW = 0x2b,
00166
00167 // Defines the 16-bit timer reload value (HIGH)
00168 T_RELOAD_HIGH = 0x2c,
00169
00170 // Defines the 16-bit timer reload value (LOW)
00171 T_RELOAD_LOW = 0x2d,
00172
00173 // Shows the 16-bit timer value (HIGH)
00174 T_COUNTER_VAL_HIGH = 0x2e,
00175
00176 // Shows the 16-bit timer value (LOW)
00177 T_COUNTER_VAL_LOW = 0x2f,
00178
00179 // Test signal configuration
00180 TEST_SEL1 = 0x31,
00181
00182 // Test signal configuration and PRBS control
00183 TEST_SEL2 = 0x32,
00184
00185 // Enables pin output driver on pins D1 to D7
00186 TEST_PIN_EN = 0x33,
00187
00188 // Defines the values for D1 to D7 when it is used as an I/O bus
00189 TEST_PIN_VALUE = 0x34,
00190
00191 // Shows the status of the internal test bus
00192 TEST_BUS = 0x35,
00193
00194 // Controls the digital self test
00195 AUTO_TEST = 0x36,
00196
00197 // Shows the software version
00198 VERSION = 0x37,
00199
00200 // Controls the pins AUX1 and AUX2
00201 ANALOG_TEST = 0x38,
00202
00203 // Defines the test value for TestDAC1
00204 TEST_DAC1 = 0x39,
00205
00206 // Defines the test value for TestDAC2
00207 TEST_DAC2 = 0x3a,
00208
00209 // Shows the value of ADC I and Q channels
00210 TEST_ADC = 0x3b
00211 };
00212
00213 enum Command {
00214
00215 // No action, cancels current command execution
00216 IDLE = 0x00,
00217
00218 // Stores 25 bytes into the internal buffer
00219 MEM = 0x01,
00220
00221 // Generates a 10-byte random ID number
00222 GENERATE_RANDOM_ID = 0x02,
00223
00224 // Activates the CRC calculation or performs a self test
00225 CALC_CRC = 0x03,
00226
00227 // Transmit data
00228 TRANSMIT = 0x04,
00229
00230 // No command change, can be used to modify the CommandReg register bits without affecting the
command, for example, the PowerDown bit
00231 NO_CMD_CHANGE = 0x07,
00232
00233 // Activates the receiver circuits (receive data)
00234 RECEIVE = 0x08,
00235
00236 // Transmits data from FIFO buffer to antenna and automatically activates the receiver after
transmission (transmit and receive data)
00237 TRANSCEIVE = 0x0c,
00238
00239 // Performs the MIFARE standard authentication as a reader (authentication)
00240 MF_AUTHENT = 0x0e,
00241
00242 // Resets the MFRC522
00243 SOFT_RESET = 0x0f
00244 };
00245
00246 enum Mask {
00247 TX_CONTROL_TX1_RF_EN = 0x01,

```

```

00248     TX_CONTROL_TX2_RF_EN = 0x02,
00249     TX_CONTROL_TX_RF_EN = TX_CONTROL_TX1_RF_EN |
TX_CONTROL_TX2_RF_EN,
00250     CONTROL_T_STOP_NOW = 0x80,
00251     CONTROL_T_START_NOW = 0x40,
00252     COM_I_EN_INTERRUPT_EN = 0x7f,
00253     COM_IRQ_TIMER_IRQ = 0x01,
00254     COM_IRQ_ERR_IRQ = 0x02,
00255     COM_IRQ_LO_ALERT_IRQ = 0x04,
00256     COM_IRQ_HI_ALERT_IRQ = 0x08,
00257     COM_IRQ_IDLE_IRQ = 0x10,
00258     COM_IRQ_RX_IRQ = 0x20,
00259     COM_IRQ_TX_IRQ = 0x40,
00260     COM_IRQ_ALL_IRQ = 0x7f,
00261     COM_IRQ_SET1 = 0x80,
00262     DIV_I_EN_CRC_I_EN = 0x04,
00263     DIV_I_EN_MFIN_ACT_I_EN = 0x10,
00264     DIV_I_EN_INTERRUPT_EN = DIV_I_EN_CRC_I_EN |
DIV_I_EN_MFIN_ACT_I_EN,
00265     DIV_IRQ_CRC_IRQ = 0x04,
00266     DIV_IRQ_MFIN_ACT_IRQ = 0x10,
00267     DIV_IRQ_ALL_IRQ = DIV_IRQ_CRC_IRQ |
DIV_IRQ_MFIN_ACT_IRQ,
00268     DIV_IRQ_SET2 = 0x80,
00269     FIFO_LEVEL_FLUSH_BUFFER = 0x80,
00270     FIFO_LEVEL_FIFO_LEVEL = 0x7f,
00271     WATER_LEVEL_WATER_LEVEL = 0x3f,
00272     BIT_FRAMING_START_SEND = 0x80,
00273     AUTO_TEST_ENABLE = 0x09,
00274     COLL_VALUES_AFTER_COLL = 0x80,
00275     STATUS2_MF_CRYPT01_ON = 0x08
00276 };
00277
00278 enum Interrupt
00279 : unsigned int {
00280     NONE_IRQ = 0x0000,
00281     COM_TIMER_IRQ = 0x0001,
00282     COM_ERR_IRQ = 0x0002,
00283     COM_LO_ALERT_IRQ = 0x0004,
00284     COM_HI_ALERT_IRQ = 0x0008,
00285     COM_IDLE_IRQ = 0x0010,
00286     COM_RX_IRQ = 0x0020,
00287     COM_TX_IRQ = 0x0040,
00288     COM_ALL_IRQ = 0x007f,
00289     DIV_CRC_IRQ = 0x0400,
00290     DIV_MFIN_ACT_IRQ = 0x1000,
00291     DIV_ALL_IRQ = DIV_CRC_IRQ | DIV_MFIN_ACT_IRQ
00292 };
00293
00298 union COMMANDbits {
00299
00300     struct {
00301
00302         // Activates a command based on the Command value;
00303         // reading this register shows which command is executed
00304         unsigned char COMMAND :4;
00305
00306         // 1: Soft power-down mode entered
00307         // 0: when the MFRC522 is ready
00308         // Remark: The PowerDown bit cannot be set when the SoftReset command is activated
00309         unsigned char POWER_DOWN :1;
00310
00311         // Analog part of the receiver is switched off
00312         unsigned char RCV_OFF :1;
00313
00314         // Reserved
00315         unsigned char :2;
00316     };
00317     unsigned char value;
00318 };
00319
00325 union COM_I_ENbits {
00326
00327     struct {
00328
00329         // Allows the timer interrupt request (TimerIRq bit) to be propagated to pin IRQ
00330         unsigned char TIMER_I_EN :1;
00331
00332         // Allows the error interrupt request (ErrIRq bit) to be propagated to pin IRQ
00333         unsigned char ERR_I_EN :1;
00334
00335         // Allows the low alert interrupt request (LoAlertIRq bit) to be propagated to pin IRQ
00336         unsigned char LO_ALERT_I_EN :1;
00337
00338         // Allows the high alert interrupt request (HiAlertIRq bit) to be propagated to pin IRQ
00339         unsigned char HI_ALERT_I_EN :1;
00340

```

```

00341         // Allows the idle interrupt request (IdleIRq bit) to be propagated to pin IRQ
00342         unsigned char IDLE_I_EN :1;
00343
00344         // Allows the receiver interrupt request (RxIRq bit) to be propagated to pin IRQ
00345         unsigned char RX_I_EN :1;
00346
00347         // Allows the transmitter interrupt request (TxIRq bit) to be propagated to pin IRQ
00348         unsigned char TX_I_EN :1;
00349
00350         // 1: Signal on pin IRQ is inverted with respect to the Status1Reg register's IRq bit
00351         // 0: signal on pin IRQ is equal to the IRq bit; in combination with the DivIENReg register's
00352         // IRqPushPull bit, the default value of logic 1 ensures that the output level on pin IRQ is
3-state
00353         unsigned char I_IRQ_INV :1;
00354     };
00355     unsigned char value;
00356 };
00357
00363     union DIV_I_ENbits {
00364     struct {
00365         // Reserved
00366         unsigned char :2;
00367
00371         // Allows the CRC interrupt request, indicated by the DivIrqReg register's CRCIRq bit, to be
propagated to pin IRQ
        unsigned char CRC_I_EN :1;
00372
00373         // Reserved
00374         unsigned char :1;
00375
00376         // Allows the MFIN active interrupt request to be propagated to pin IRQ
00377         unsigned char MFIN_ACT_I_EN :1;
00378
00379         // Reserved
00380         unsigned char :2;
00381
00382         // 1: pin IRQ is a standard CMOS output pin
00383         // 0: pin IRQ is an open-drain output pin
00384         unsigned char IRQ_PUSH_PULL :1;
00385     };
00386     unsigned char value;
00387 };
00388
00394     union COM_IRQbits {
00395     struct {
00396
00397
00398         // The timer decrements the timer value in register TCounterValReg to zero
00399         unsigned char TIMER_IRQ :1;
00400
00401         // Any error bit in the ErrorReg register is set
00402         unsigned char ERR_IRQ :1;
00403
00404         // Status1Reg register's LoAlert bit is set in opposition to the LoAlert bit,
00405         // the LoAlertIRq bit stores this event and can only be reset as indicated by
00406         // the Set1 bit in this register
00407         unsigned char LO_ALERT_IRQ :1;
00408
00409         // Status1Reg register's HiAlert bit is set in opposition to the HiAlert bit,
00410         // the HiAlertIRq bit stores this event and can only be reset as indicated by
00411         // the Set1 bit in this register
00412         unsigned char HI_ALERT_IRQ :1;
00413
00414         // If a command terminates, for example, when the CommandReg changes
00415         // its value from any command to the Idle command if an unknown command is started,
00416         // the CommandReg register Command[3:0] value changes to the idle state and the IdleIRq bit is
set
00417         // The microcontroller starting the Idle command does not set the IdleIRq bit
00418         unsigned char IDLE_IRQ :1;
00419
00420         // Receiver has detected the end of a valid data stream
00421         // if the RxModeReg register's RxNoErr bit is set to logic 1, the RxIRq bit is
00422         // only set to logic 1 when data bytes are available in the FIFO
00423         unsigned char RX_IRQ :1;
00424
00425         // Set immediately after the last bit of the transmitted data was sent out
00426         unsigned char TX_IRQ :1;
00427
00428         // 1: indicates that the marked bits in the ComIrqReg register are set
00429         // 0: indicates that the marked bits in the ComIrqReg register are cleared
00430         unsigned char SET1 :1;
00431     };
00432     unsigned char value;
00433 };
00434

```

```

00440     union DIV_IRQbits {
00441
00442         struct {
00443
00444             // Reserved
00445             unsigned char :2;
00446
00447             // The CalcCRC command is active and all data is processed
00448             unsigned char CRC_IRQ :1;
00449
00450             // Reserved
00451             unsigned char :1;
00452
00453             // MFIN is active this interrupt is set when either a rising or falling signal edge is
00454             detected.
00455             unsigned char MFIN_ACT_IRQ :1;
00456
00457             // Reserved
00458             unsigned char :2;
00459
00460             // 1: indicates that the marked bits in the DivIrqReg register are set
00461             // 0: indicates that the marked bits in the DivIrqReg register are cleared
00462             unsigned char SET2 :1;
00463         };
00464     };
00465     unsigned char value;
00466 };
00471     union ERRORbits {
00472
00473         struct {
00474
00475             // Set to logic 1 if the SOF is incorrect automatically cleared during receiver start-up phase
00476             // bit is only valid for 106 kBd during the MFAuthent command, the ProtocolErr bit is set to
00477             // logic 1 if the number of bytes received in one data stream is incorrect
00478             unsigned char PROTOCOL_ERR :1;
00479
00480             // Parity check failed. Automatically cleared during receiver start-up phase
00481             // only valid for ISO/IEC 14443 A/MIFARE communication at 106 kBd
00482             unsigned char PARITY_ERR :1;
00483
00484             // The RxModeReg register's RxCRCEn bit is set and the CRC calculation fails
00485             // automatically cleared to logic 0 during receiver start-up phase
00486             unsigned char CRC_ERR :1;
00487
00488             // A bit-collision is detected cleared automatically at receiver start-up phase
00489             // only valid during the bitwise anticollision at 106 kBd always set to logic 0 during
00490             communication
00491             // protocols at 212 kBd, 424 kBd and 848 kBd
00492             unsigned char COLL_ERR :1;
00493
00494             // The host or a MFRC522's internal state machine (e.g. receiver) tries to
00495             // write data to the FIFO buffer even though it is already full
00496             unsigned char BUFFER_OVFL :1;
00497
00498             // Reserved
00499             unsigned char :1;
00500
00501             // Internal temperature sensor detects overheating, in which case the antenna drivers are
00502             // automatically switched off
00503             unsigned char TEMP_ERR :1;
00504
00505             // Data is written into the FIFO buffer by the host during the MFAuthent command or if data is
00506             // written
00507             // into the FIFO buffer by the host during the time between sending the last bit on the RF
00508             // interface and
00509             // receiving the last bit on the RF interface
00510             unsigned char WR_ERR :1;
00511         };
00512     };
00513     unsigned char value;
00514 };
00515     union STATUS1bits {
00516
00517         struct {
00518
00519             // The number of bytes stored in the FIFO buffer corresponds to equation:
00520             // HiAlert = FIFOLength <= WaterLevel
00521             // example:
00522             // FIFO length = 4, WaterLevel = 4 > LoAlert = 1
00523             // FIFO length = 5, WaterLevel = 4 > LoAlert = 0
00524             unsigned char LO_ALERT :1;
00525
00526
00527             // The number of bytes stored in the FIFO buffer corresponds to equation:
00528             // HiAlert = (64 - FIFOLength) <= WaterLevel
00529             // example:
00530             // FIFO length = 60, WaterLevel = 4 > HiAlert = 1
00531             // FIFO length = 59, WaterLevel = 4 > HiAlert = 0

```

```

00532         unsigned char HI_ALERT :1;
00533
00534         // MFRC522's timer unit is running, i.e. the timer will decrement the TCounterValReg register
with the next timer clock
00535         // 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;
00536         // this bit is not influenced by the gated signal
00537         unsigned char T_RUNNING :1;
00538
00539         // Indicates if any interrupt source requests attention with respect to the setting of the
interrupt enable bits:
00540         // see the ComIEEnReg and DivIEEnReg registers
00541         unsigned char IRQ :1;
00542
00543         // The CRC calculation has finished only valid for the CRC coprocessor calculation using the
CalcCRC command
00544         unsigned char CRC_READY :1;
00545
00546         // The CRC result is zero
00547         // for data transmission and reception, the CRCOK bit is undefined: use the
00548         // ErrorReg register's CRCError bit indicates the status of the CRC coprocessor, during
calculation the value
00549         // changes to logic 0, when the calculation is done correctly the value changes to logic 1
00550         unsigned char CRC_OK :1;
00551
00552         // Reserved
00553         unsigned char :1;
00554     };
00555     unsigned char value;
00556 };
00557
00563     union STATUS2bits {
00564     struct {
00565
00566         // Shows the state of the transmitter and receiver state machines:
00567         // 000: idle
00568         // 001: wait for the BitFramingReg register's StartSend bit
00569         // 010: TxWait: wait until RF field is present if the TModeReg register's
00570         // TxWaitRF bit is set to logic 1 the minimum time for TxWait is defined by the TxWaitReg
register
00571         // 011: transmitting
00572         // 100: RxWait: wait until RF field is present if the TModeReg register's TxWaitRF bit is set
to logic 1
00573         // the minimum time for RxWait is defined by the RxWaitReg register
00574         // 101: wait for data
00575         // 110: receiving
00576         unsigned char MODEM_STATE :3;
00577
00578         // Indicates that the MIFARE Cryptol unit is switched on and therefore all data communication
with the card is encrypted
00579         // can only be set to logic 1 by a successful execution of the MFAuthent command only valid in
Read/Write mode for
00580         // MIFARE standard cards this bit is cleared by software
00581         unsigned char MF_CRYPTOL_ON :1;
00582
00583         // Reserved
00584         unsigned char :2;
00585
00586         // I2C-bus input filter settings:
00587         // 1: the I2C-bus input filter is set to the High-speed mode independent of the I2C-bus
protocol
00588         // 0: the I2C-bus input filter is set to the I2C-bus protocol used
00589         unsigned char I2C_FORCE_HS :1;
00590
00591         // Clears the temperature error if the temperature is below the alarm limit of 125C
00592         unsigned char TEMP_SENS_CLEAR :1;
00593     };
00594     unsigned char value;
00595 };
00596
00597     union FIFO_LEVELbits {
00603     struct {
00604
00605         // Indicates the number of bytes stored in the FIFO buffer writing to the FIFODataReg
00606         // register increments and reading decrements the FIFOLevel value
00607         unsigned char FIFO_LEVEL :7;
00608
00609         // Immediately clears the internal FIFO buffer's read and write pointer and ErrorReg
00610         // register's BufferOvfl bit reading this bit always returns 0
00611         unsigned char FLUSH_BUFFER :1;
00612     };
00613     unsigned char value;
00614 };
00615
00616     union WATER_LEVELbits {
00623

```

```

00624
00625     struct {
00626
00627         // Defines a warning level to indicate a FIFO buffer overflow or underflow:
00628         // Status1Reg register's HiAlert bit is set to logic 1 if the remaining
00629         // number of bytes in the FIFO buffer space is equal to, or less than the defined number of
WaterLevel bytes
00630         // Status1Reg register's LoAlert bit is set to logic 1 if equal to, or less than the WaterLevel
bytes in the FIFO buffer
00631         // Remark: to calculate values for HiAlert and LoAlert see Section 9.3.1.8 on page 42.
00632         unsigned char WATER_LEVEL :7;
00633
00634         // Reserved
00635         unsigned char :1;
00636     };
00637     unsigned char value;
00638 };
00639
00640     union CONTROLbits {
00641     struct {
00642
00643         // Indicates the number of valid bits in the last received byte if this value is 000b, the
whole byte is valid
00644         unsigned char RX_LAST_BITS :3;
00645
00646         // Reserved
00647         unsigned char :2;
00648
00649         // Timer starts immediately
00650         // reading this bit always returns it to logic 0
00651         unsigned char T_START_NOW :1;
00652
00653         // Timer stops immediately
00654         // reading this bit always returns it to logic0
00655         unsigned char T_STOP_NOW :1;
00656     };
00657     unsigned char value;
00658 };
00659
00660     union BIT_FRAMINGbits {
00661     struct {
00662
00663         // Used for transmission of bit oriented frames: defines the number of bits of the last byte
that will be transmitted
00664         // 000b indicates that all bits of the last byte will be transmitted
00665         unsigned char TX_LAST_BITS :3;
00666
00667         // Reserved
00668         unsigned char :1;
00669
00670         // used for reception of bit-oriented frames: defines the bit position for the first bit
received to be stored in the FIFO buffer
00671         // example:
00672         // 0: LSB of the received bit is stored at bit position 0, the second received bit is stored at
bit position 1
00673         // 1: LSB of the received bit is stored at bit position 1, the second received bit is stored at
bit position 2
00674         // 7: LSB of the received bit is stored at bit position 7, the second received bit is stored in
the next byte that follows at bit position 0
00675         // These bits are only to be used for bitwise anticollision at 106 kBd, for all other modes
they are set to 0
00676         unsigned char RX_ALIGN :3;
00677
00678         // Starts the transmission of data only valid in combination with the Transceive command
00679         unsigned char START_SEND :1;
00680     };
00681     unsigned char value;
00682 };
00683
00684     union COLLbits {
00685     struct {
00686
00687         // Shows the bit position of the first detected collision in a received frame only data bits
are interpreted
00688         // example:
00689         // 00h: indicates a bit-collision in the 32nd bit
00690         // 01h: indicates a bit-collision in the 1st bit
00691         // 08h: indicates a bit-collision in the 8th bit
00692         // These bits will only be interpreted if the CollPosNotValid bit is set to logic 0
00693         unsigned char COLL_POS :5;
00694
00695         // No collision detected or the position of the collision is out of the range of CollPos[4:0]
00696         unsigned char COLL_POS_NOT_VALID :1;
00697     };
00698     };
00699
00700
00701
00702
00703
00704
00705
00706
00707
00708
00709
00710
00711
00712
00713
00714
00715

```



```

00716          // Reserved
00717          unsigned char :1;
00718
00719          // All received bits will be cleared after a collision only used during bitwise anticollision
at 106 kBd, otherwise it is set to logic 1
00720          unsigned char VALUES_AFTER_COLL :1;
00721      };
00722      unsigned char value;
00723  };
00724
00730      union MODEbits {
00731
00732          struct {
00733
00734              // defines the preset value for the CRC coprocessor for the CalcCRC command
00735              // Remark: during any communication, the preset values are selected automatically according to
00736              // the definition of bits in the RxModeReg and TxModeReg registers
00737              // 00: 0000h
00738              // 01: 6363h
00739              // 10: A671h
00740              // 11: FFFFh
00741              unsigned char CRC_PRESET :2;
00742
00743              // Reserved
00744              unsigned char :1;
00745
00746              // Defines the polarity of pin MFIN
00747              // Remark: the internal envelope signal is encoded active LOW, changing this bit generates a
MFinActIRq event
00748              // 1: polarity of pin MFIN is active HIGH
00749              // 0: polarity of pin MFIN is active LOW
00750              unsigned char POL_M_FIN :1;
00751
00752              // Reserved
00753              unsigned char :1;
00754
00755              // Transmitter can only be started if an RF field is generated
00756              unsigned char TX_WAIT_RF :1;
00757
00758              // Reserved
00759              unsigned char :1;
00760
00761              // CRC coprocessor calculates the CRC with MSB first in the CRCResultReg register the values
for the
00762              // CRCResultMSB[7:0] bits and the CRCResultLSB[7:0] bits are bit reversed
00763              // Remark: during RF communication this bit is ignored
00764              unsigned char MSB_FIRST :1;
00765          };
00766          unsigned char value;
00767      };
00768
00774      union TX_MODEbits {
00775
00776          struct {
00777
00778              // Reserved
00779              unsigned char :3;
00780
00781              // Modulation of transmitted data is inverted
00782              unsigned char INV_MOD :1;
00783
00784              // Defines the bit rate during data transmission the MFRC522 handles transfer speeds up to 848
kBd
00785              // 000: 106 kBd
00786              // 001: 212 kBd
00787              // 010: 424 kBd
00788              // 011: 848 kBd
00789              unsigned char TX_SPEED :3;
00790
00791              // Enables CRC generation during data transmission
00792              // Remark: can only be set to logic 0 at 106 kBd
00793              unsigned char TX_CRC_EN :1;
00794          };
00795          unsigned char value;
00796      };
00797
00803      union RX_MODEbits {
00804
00805          struct {
00806
00807              // Reserved
00808              unsigned char :2;
00809
00810              // 0: receiver is deactivated after receiving a data frame
00811              // 1: able to receive more than one data frame only valid for data rates above 106 kBd in order
to handle
00812              // the polling command after setting this bit the Receive and Transceive commands will not

```

```

        terminate automatically.
00813        // Multiple reception can only be deactivated by writing any command (except the Receive
        command) to the CommandReg
00814        // register, or by the host clearing the bit if set to logic 1, an error byte is added to the
        FIFO buffer at the
00815        // end of a received data stream which is a copy of the ErrorReg register value. For the
        MFR522 version 2.0 the CRC status is
00816        // reflected in the signal CRCOk, which indicates the actual status of the CRC coprocessor. For
        the MFR522 version 1.0 the CRC
00817        // status is reflected in the signal CRCErr.
00818        unsigned char RX_MULTIPLE :1;
00819
00820        // An invalid received data stream (less than 4 bits received) will be ignored and the receiver
        remains active
00821        unsigned char RX_NO_ERR :1;
00822
00823        // Defines the bit rate while receiving data the MFR522 handles transfer speeds up to 848 kBd
00824        // 000: 106 kBd
00825        // 001: 212 kBd
00826        // 010: 424 kBd
00827        // 011: 848 kBd
00828        unsigned char RX_SPEED :3;
00829
00830        // Enables the CRC calculation during reception
00831        // Remark: can only be set to logic 0 at 106 kBd
00832        unsigned char RX_CRC_EN :1;
00833    };
00834    unsigned char value;
00835};
00836
00842    union TX_CONTROLbits {
00843        struct {
00844
00845            // Output signal on pin TX1 delivers the 13.56 MHz energy carrier modulated by the transmission
00846        data
00847            unsigned char TX1_RF_EN :1;
00848
00849            // Output signal on pin TX2 delivers the 13.56 MHz energy carrier modulated by the transmission
00850        data
00851            unsigned char TX2_RF_EN :1;
00852
00853            // Reserved
00854            unsigned char :1;
00855
00856            // 1: output signal on pin TX2 continuously delivers the unmodulated 13.56 MHz energy carrier
00857            // 0: Tx2CW bit is enabled to modulate the 13.56 MHz energy carrier
00858            unsigned char TX2_CW :1;
00859
00860            // Output signal on pin TX1 inverted when driver TX1 is disabled
00861            unsigned char INV_TX1_RF_OFF :1;
00862
00863            // Output signal on pin TX2 inverted when driver TX2 is disabled
00864            unsigned char INV_TX2_RF_OFF :1;
00865
00866            // Output signal on pin TX1 inverted when driver TX1 is enabled
00867            unsigned char INV_TX1_RF_ON :1;
00868
00869            // Output signal on pin TX2 inverted when driver TX2 is enabled
00870            unsigned char INV_TX2_RF_ON :1;
00871        };
00872        struct {
00873
00874            // Output signal on pin TX1 delivers the 13.56 MHz energy carrier modulated by the transmission
00875        data
00876            // Output signal on pin TX2 delivers the 13.56 MHz energy carrier modulated by the transmission
00877        data
00878            unsigned char TX_RF_EN :2;
00879
00880            // Reserved
00881            unsigned char :2;
00882
00883            // Output signal on pin TX1 inverted when driver TX1 is disabled
00884            // Output signal on pin TX2 inverted when driver TX2 is disabled
00885            unsigned char INV_TX_RF_OFF :2;
00886
00887            // Output signal on pin TX1 inverted when driver TX1 is enabled
00888            // Output signal on pin TX2 inverted when driver TX2 is enabled
00889            unsigned char INV_TX_RF_ON :2;
00890        };
00891        unsigned char value;
00892    };
00893
00897    union TX_ASKbits {
00898        struct {
00899

```

```

00900
00901         // Reserved
00902         unsigned char :6;
00903
00904         // Forces a 100% ASK modulation independent of the ModGsPReg register setting
00905         unsigned char FORCE_100_ASK :1;
00906
00907         // Reserved
00908         unsigned char :1;
00909     };
00910     unsigned char value;
00911 };
00912
00913 union TX_SELbits {
00914     struct {
00915         // Selects the input for pin MFOUT
00916         // 0000: 3-state
00917         // 0001: LOW
00918         // 0010: HIGH
00919         // 0011: test bus signal as defined by the TestSelReg register's TstBusBitSel[2:0] value
00920         // 0100: modulation signal (envelope) from the internal encoder, Miller pulse encoded
00921         // 0101: serial data stream to be transmitted, data stream before Miller encoder
00922         // 0110: reserved
00923         // 0111: serial data stream received, data stream after Manchester decoder
00924         // 1000: to 1111 reserved
00925         unsigned char MF_OUT_SEL :4;
00926
00927         // Selects the input of drivers TX1 and TX2
00928         // 00: 3-state; in soft power-down the drivers are only in 3-state mode if the DriverSel[1:0]
00929         value is set to 3-state mode
00930         // 01: modulation signal (envelope) from the internal encoder, Miller pulse encoded
00931         // 10: modulation signal (envelope) from pin MFIN
00932         // 11: HIGH; the HIGH level depends on the setting of bits InvTx1RFOn/InvTx1RFOff and
00933         InvTx2RFOn/InvTx2RFOff
00934         unsigned char :2;
00935
00936         // Reserved
00937         unsigned char :2;
00938     };
00939     unsigned char value;
00940 };
00941
00942 union RX_SELbits {
00943     struct {
00944         // After data transmission the activation of the receiver is delayed for RxWait bit-clocks,
00945         during this 'frame guard time
00946         // any signal on pin RX is ignored this parameter is ignored by the Receive command all other
00947         commands, such as Transceive,
00948         // MFAuthent use this parameter the counter starts immediately after the external RF field is
00949         switched on
00950         unsigned char RX_WAIT :6;
00951
00952         // Selects the input of the contactless UART
00953         // 00: constant LOW
00954         // 01: Manchester with subcarrier from pin MFIN
00955         // 10: modulated signal from the internal analog module, default
00956         // 11: NRZ coding without subcarrier from pin MFIN which is only valid for transfer speeds
00957         above 106 kBd
00958         unsigned char UART_SEL :2;
00959     };
00960     unsigned char value;
00961 };
00962
00963 union RX_THRESHOLDbits {
00964     struct {
00965         // defines the minimum signal strength at the decoder input that must be reached by the weaker
00966         half-bit of the
00967         // Manchester encoded signal to generate a bit-collision relative to the amplitude of the
00968         stronger half-bit
00969         unsigned char COLL_LEVEL :3;
00970
00971         // Reserved
00972         unsigned char :1;
00973
00974         // Defines the minimum signal strength at the decoder input that will be accepted if the signal
00975         strength is below this level it is not evaluated
00976         unsigned char MIN_LEVEL :4;
00977     };
00978     unsigned char value;
00979 };
00980
00981 };
00982
00983 };
00984
00985 };
00986
00987 };
00988
00989 };
00990
00991 };
00992

```

```

00998     union DEMODbits {
00999
01000         struct {
01001
01002             // Changes the time-constant of the internal PLL during burst
01003             unsigned char TAU_SYNC :2;
01004
01005             // Changes the time-constant of the internal PLL during data reception
01006             // Remark: if set to 00b the PLL is frozen during data reception
01007             unsigned char TAU_RCV :2;
01008
01009             // If set to logic 0 the following formula is used to calculate the timer frequency of the
prescaler:
01010             // F_timer = 13.56 MHz / (2*TPreScaler+1).
01011             // If set to logic 1 the following formula is used to calculate the timer frequency of the
prescaler:
01012             // F_timer = 13.56 MHz / (2*TPreScaler+2)
01013             unsigned char T_PRESCAL_EVEN :1;
01014
01015             // If AddIQ[1:0] are set to X0b, the reception is fixed to I channel
01016             // If AddIQ[1:0] are set to X1b, the reception is fixed to Q channel
01017             unsigned char FIX_IQ :1;
01018
01019             // Defines the use of I and Q channel during reception
01020             // Remark: the FixIQ bit must be set to logic 0 to enable the following settings:
01021             // 00: selects the stronger channel
01022             // 01: selects the stronger channel and freezes the selected channel during communication
01023             unsigned char ADD_IQ :2;
01024         };
01025         unsigned char value;
01026     };
01027     union MF_TXbits {
01033         struct {
01034
01035             // Defines the additional response time 7 bits are added to the value of the register bit by
default
01038             unsigned char TX_WAIT :2;
01039
01040             // Reserved
01041             unsigned char :6;
01042         };
01043         unsigned char value;
01044     };
01045     union MF_RXbits {
01052         struct {
01053
01054             // Reserved
01055             unsigned char :4;
01056
01057             // Generation of the parity bit for transmission and the parity check for receiving is switched
off
01059             // the received parity bit is handled like a data bit
01060             unsigned char PARITY_DISABLE :1;
01061
01062             // Reserved
01063             unsigned char :3;
01064         };
01065         unsigned char value;
01066     };
01067     union SERIAL_SPEEDbits {
01073         struct {
01074
01075             // Factor BR_T1 adjusts the transfer speed
01078             unsigned char BR_T1 :5;
01079
01080             // Factor BR_T0 adjusts the transfer speed
01081             unsigned char BR_T0 :3;
01082         };
01083         unsigned char value;
01084     };
01085     union RF_CFGbits {
01091         struct {
01092
01093             // Reserved
01096             unsigned char :4;
01097
01098             // Defines the receiver's signal voltage gain factor:
01099             // 000: 18 dB
01100             // 001: 23 dB

```

```

01101          // 010: 18 dB
01102          // 011: 23 dB
01103          // 100: 33 dB
01104          // 101: 38 dB
01105          // 110: 43 dB
01106          // 111: 48 dB
01107          unsigned char RX_GAIN :3;
01108
01109          // Reserved
01110          unsigned char :1;
01111      };
01112      unsigned char value;
01113  };
01114
01120      union GS_Nbits {
01121          struct {
01122              // Defines the conductance of the output n-driver during periods without modulation which can
01123              // be used to regulate the modulation index
01124              // Remark: the conductance value is binary weighted during soft Power-down mode the highest bit
01125              // is forced to logic 1
01126              // value is only used if driver TX1 or TX2 is switched on
01127              unsigned char MOD_GS_N :4;
01128
01129              // defines the conductance of the output n-driver during periods without modulation which can
01130              // be used to regulate the output power and
01131              // subsequently current consumption and operating distance
01132              // Remark: the conductance value is binary-weighted during soft Power-down mode the highest bit
01133              // is forced to logic 1
01134              // value is only used if driver TX1 or TX2 is switched on
01135              unsigned char CW_GS_N :4;
01136          };
01137          unsigned char value;
01138      };
01139
01143      union CW_GS_Pbits {
01144          struct {
01145              // defines the conductance of the p-driver output which can be used to regulate the output
01146              // power and subsequently current consumption and operating distance
01147              // Remark: the conductance value is binary weighted during soft Power-down mode the highest bit
01148              // is forced to logic 1
01149              unsigned char CW_GS_P :6;
01150
01151              // Reserved
01152              unsigned char :2;
01153          };
01154          unsigned char value;
01155      };
01156
01162      union MOD_GS_Pbits {
01163          struct {
01164              // Defines the conductance of the p-driver output during modulation which can be used to
01165              // regulate the modulation index
01166              // Remark: the conductance value is binary weighted during soft Power-down mode the highest bit
01167              // is forced to logic 1
01168              // if the TxASKReg register's Force100ASK bit is set to logic 1 the value of ModGsP has no
01169              // effect
01170              unsigned char MOD_GS_P :6;
01171
01172              // Reserved
01173              unsigned char :2;
01174          };
01175          unsigned char value;
01176      };
01177
01184      union T_MODEbits {
01185          struct {
01186              // Defines the higher 4 bits of the TPrescaler value
01187              // The following formula is used to calculate the timer frequency if the DemodReg register's
01188              // TPrescalEven bit in Demot Regis set to logic 0:
01189              //  $F_{timer} = 13.56 \text{ MHz} / (2 * TPrescaler + 1)$ 
01190              // Where TPrescaler = [TPrescaler_Hi:TPrescaler_Lo] (TPrescaler value on 12 bits) (Default
01191              // TPrescalEven bit is logic 0)
01192              // The following formula is used to calculate the timer frequency if the DemodReg register's
01193              // TPrescalEven bit is set to logic 1:
01194              //  $F_{timer} = 13.56 \text{ MHz} / (2 * TPrescaler + 2)$ .
01195              unsigned char T_PRESCALER_HI :4;
01196
01197              // 1: timer automatically restarts its count-down from the 16-bit timer reload value instead of
01198              // counting down to zero

```

```

01197         // 0: timer decrements to 0 and the ComIrqReg register's TimerIRq bit is set to logic 1
01198         unsigned char T_AUTO_RESTART :1;
01199
01200         // Internal timer is running in gated mode
01201         // Remark: in gated mode, the Status1Reg register's TRunning bit is logic 1 when the timer is
enabled by the
01202         // TModeReg register's TGated[1:0] bits this bit does not influence the gating signal
01203         // 00: non-gated mode
01204         // 01: gated by pin MFIN
01205         // 10: gated by pin AUX1
01206         unsigned char T_GATED :2;
01207
01208         // 1: timer starts automatically at the end of the transmission in all communication modes at
all speeds
01209         // 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)
01210         // 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
01211         // TStopNow bit to logic 1
01212         // 0: indicates that the timer is not influenced by the protocol
01213         unsigned char T_AUTO :1;
01214     };
01215     unsigned char value;
01216 };
01217
01223     union VERSIONbits {
01224
01225         struct {
01226
01227             // '1' stands for MFRC522 version 1.0 and '2' stands for MFRC522 version 2.0.
01228             unsigned char VERSION :4;
01229
01230             // '9' stands for MFRC522
01231             unsigned char CHIPTYPE :4;
01232         };
01233         unsigned char value;
01234     };
01235
01236     enum Version {
01237         CLONE = 0x88,
01238         V0_0 = 0x90,
01239         V1_0 = 0x91,
01240         V2_0 = 0x92
01241     };
01242
01243     ReaderMFRC522(RegisterBasedDevice *device, unsigned char resetPin);
01244
01245     virtual ~ReaderMFRC522();
01246
01250     void initialize();
01251
01257     inline void sendCommand(unsigned char command);
01258
01262     void softReset();
01263
01268     void setAntennaOn();
01269
01273     void setAntennaOff();
01274
01315     void configureTimer(unsigned int prescaler, unsigned int reload, bool autoStart, bool
autoRestart);
01316
01321     void startTimer();
01322
01327     void stopTimer();
01328
01336     void enableInterrupt(unsigned int interrupt);
01337
01345     void disableInterrupt(unsigned int interrupt);
01346
01354     void clearInterrupt(unsigned int interrupt);
01355
01360     void flushQueue();
01361
01367     void setWaterLevel(unsigned char level);
01368
01374     int generateRandomId(unsigned char *buf);
01375
01392     int communicate(unsigned char command, unsigned char *send, unsigned char *receive, unsigned
char sendLen, bool checkCrc);
01393
01394     inline int communicate(unsigned char command, unsigned char *send, unsigned char *receive,
unsigned char sendLen);
01395
01405     int tranceive(unsigned char *send, unsigned char *receive, unsigned char sendLen, bool
checkCrc);
01406

```

```

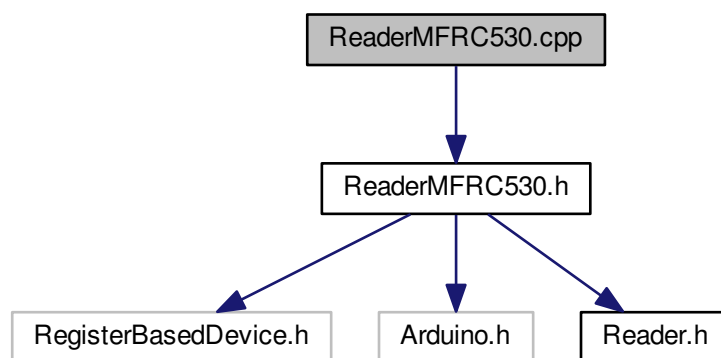
01407     inline int tranceive(unsigned char *send, unsigned char *receive, unsigned char sendLen);
01408
01429     int authenticate(unsigned char *send);
01430
01438     unsigned int calculateCrc(unsigned char *buf, unsigned char len);
01439
01447     void calculateCrc(unsigned char *buf, unsigned char len, unsigned char *dst);
01448
01457     bool waitForRegisterBits(unsigned char reg, unsigned char mask, unsigned long
timeout);
01458
01459     inline bool waitForRegisterBits(unsigned char reg, unsigned char mask);
01460
01464     Version getVersion();
01465
01495     bool performSelfTest();
01496
01504     void setBitFraming(unsigned char rxAlign, unsigned char txLastBits);
01505
01506     unsigned char getCollisionPosition();
01507
01508     void setupForAnticollision();
01509
01520     int readRegisterBlock(unsigned char reg, unsigned char *buf, unsigned char len);
01521
01533     int readRegisterBlock(unsigned char reg, unsigned char *buf, unsigned char len,
unsigned char rxAlign);
01534
01543     unsigned char writeRegisterBlock(unsigned char reg, unsigned char *buf, unsigned char
len);
01544
01545     void turnOffEncryption();
01546
01550     bool hasValidCrc(unsigned char *buf, unsigned char len);
01551
01552 };
01553
01554 #endif // __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_READER_MFRC522_H__

```

5.11 ReaderMFRC530.cpp File Reference

```
#include "ReaderMFRC530.h"
```

Include dependency graph for ReaderMFRC530.cpp:



5.12 ReaderMFRC530.cpp

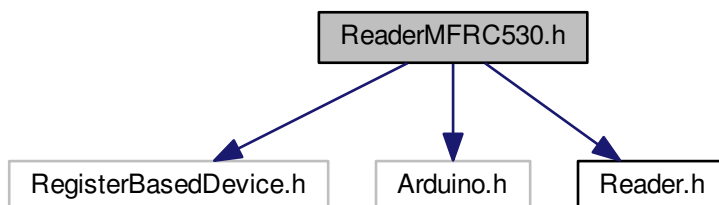
```

00001 #include "ReaderMFRC530.h"
00002

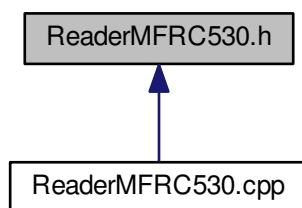
```

5.13 ReaderMFRC530.h File Reference

```
#include <RegisterBasedDevice.h>
#include <Arduino.h>
#include <Reader.h>
Include dependency graph for ReaderMFRC530.h:
```



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



Classes

- class [ReaderMFRC530](#)

5.14 ReaderMFRC530.h

```
00001
00006 #ifndef __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_READER_MFRC530_H__
00007 #define __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_READER_MFRC530_H__ 1
00008
00009 #include <RegisterBasedDevice.h>
00010 #include <Arduino.h>
00011 #include <Reader.h>
00012
00013 class ReaderMFRC530: public Reader, public RegisterBasedDevice {
00014
00015 public:
00016
00017     enum Register {
```



```
00018     };
00019
00020     enum Command {
00021     };
00022
00023     enum Mask {
00024     };
00025
00026     enum Interrupt
00027         : unsigned int {
00028     };
00029
00030     enum Version {
00031     };
00032
00033     ReaderMFRC530(RegisterBasedDevice *device, unsigned char resetPin);
00034
00035     virtual ~ReaderMFRC530();
00036
00037     void initialize();
00038
00039     inline void sendCommand(unsigned char command);
00040
00041     void softReset();
00042
00043     void setAntennaOn();
00044
00045     void setAntennaOff();
00046
00047     void configureTimer(unsigned int prescaler, unsigned int reload, bool autoStart, bool
autoRestart);
00048
00049     void startTimer();
00050
00051     void stopTimer();
00052
00053     void enableInterrupt(unsigned int interrupt);
00054
00055     void disableInterrupt(unsigned int interrupt);
00056
00057     void clearInterrupt(unsigned int interrupt);
00058
00059     void flushQueue();
00060
00061     void setWaterLevel(unsigned char level);
00062
00063     int generateRandomId(unsigned char *buf);
00064
00065     int communicate(unsigned char command, unsigned char *send, unsigned char *receive, unsigned
char sendLen, bool checkCrc);
00066
00067     inline int communicate(unsigned char command, unsigned char *send, unsigned char *receive,
unsigned char sendLen);
00068
00069     int tranceive(unsigned char *send, unsigned char *receive, unsigned char sendLen, bool
checkCrc);
00070
00071     inline int tranceive(unsigned char *send, unsigned char *receive, unsigned char sendLen);
00072
00073     int authenticate(unsigned char *send);
00074
00075     unsigned int calculateCrc(unsigned char *buf, unsigned char len);
00076
00077     void calculateCrc(unsigned char *buf, unsigned char len, unsigned char *dst);
00078
00079     bool waitForRegisterBits(unsigned char reg, unsigned char mask, unsigned long
timeout);
00080
00081     inline bool waitForRegisterBits(unsigned char reg, unsigned char mask);
00082
00083     Version getVersion();
00084
00085     bool performSelfTest();
00086
00087     void setBitFraming(unsigned char rxAlign, unsigned char txLastBits);
00088
00089     unsigned char getCollisionPosition();
00090
00091     void setupForAnticollision();
00092
00093     int readRegisterBlock(unsigned char reg, unsigned char *buf, unsigned char len);
00094
00095     int readRegisterBlock(unsigned char reg, unsigned char *buf, unsigned char len,
unsigned char rxAlign);
00096
00097     unsigned char writeRegisterBlock(unsigned char reg, unsigned char *buf, unsigned char
len);
```

```

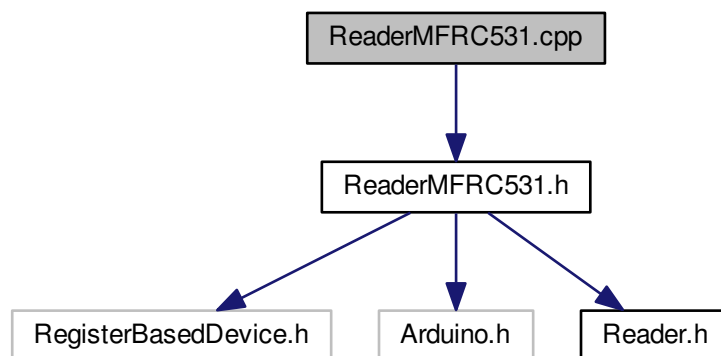
00098
00099     void turnOffEncryption();
00100
00101     bool hasValidCrc(unsigned char *buf, unsigned char len);
00102
00103 };
00104
00105 #endif // __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_READER_MFRC530_H__

```

5.15 ReaderMFRC531.cpp File Reference

```
#include "ReaderMFRC531.h"
```

Include dependency graph for ReaderMFRC531.cpp:



5.16 ReaderMFRC531.cpp

```

00001 #include "ReaderMFRC531.h"
00002

```

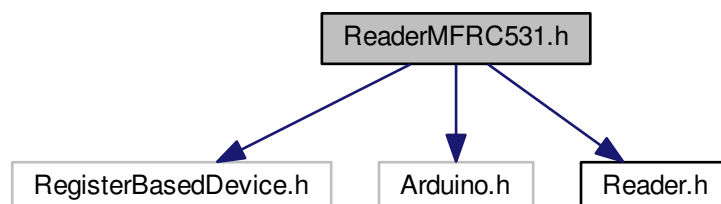
5.17 ReaderMFRC531.h File Reference

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

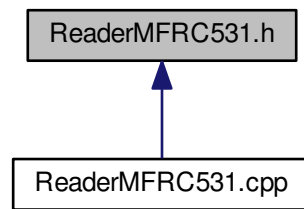
```
#include <Arduino.h>
```

```
#include <Reader.h>
```

Include dependency graph for ReaderMFRC531.h:



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



Classes

- class [ReaderMFRC531](#)

5.18 ReaderMFRC531.h

```

00001
00006 #ifndef __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_READER_MFRC531_H__
00007 #define __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_READER_MFRC531_H__ 1
00008
00009 #include <RegisterBasedDevice.h>
00010 #include <Arduino.h>
00011 #include <Reader.h>
00012
00013 class ReaderMFRC531: public Reader, public RegisterBasedDevice {
00014
00015 public:
00016
00017     enum Register {
00018     };
00019
00020     enum Command {
00021     };
00022
00023     enum Mask {
00024     };
00025
00026     enum Interrupt
00027         : unsigned int {
00028     };
00029
00030     enum Version {
00031     };
00032
00033     ReaderMFRC531(RegisterBasedDevice *device, unsigned char resetPin);
00034
00035     virtual ~ReaderMFRC531();
00036
00037     void initialize();
00038
00039     inline void sendCommand(unsigned char command);
00040
00041     void softReset();
00042
00043     void setAntennaOn();
00044
00045     void setAntennaOff();
00046
00047     void configureTimer(unsigned int prescaler, unsigned int reload, bool autoStart, bool
00048         autoRestart);
00049
00049     void startTimer();
00050
00050     void stopTimer();
00051
  
```

```

00052
00053     void enableInterrupt(unsigned int interrupt);
00054
00055     void disableInterrupt(unsigned int interrupt);
00056
00057     void clearInterrupt(unsigned int interrupt);
00058
00059     void flushQueue();
00060
00061     void setWaterLevel(unsigned char level);
00062
00063     int generateRandomId(unsigned char *buf);
00064
00065     int communicate(unsigned char command, unsigned char *send, unsigned char *receive, unsigned
char sendLen, bool checkCrc);
00066
00067     inline int communicate(unsigned char command, unsigned char *send, unsigned char *receive,
unsigned char sendLen);
00068
00069     int tranceive(unsigned char *send, unsigned char *receive, unsigned char sendLen, bool
checkCrc);
00070
00071     inline int tranceive(unsigned char *send, unsigned char *receive, unsigned char sendLen);
00072
00073     int authenticate(unsigned char *send);
00074
00075     unsigned int calculateCrc(unsigned char *buf, unsigned char len);
00076
00077     void calculateCrc(unsigned char *buf, unsigned char len, unsigned char *dst);
00078
00079     bool waitForRegisterBits(unsigned char reg, unsigned char mask, unsigned long
timeout);
00080
00081     inline bool waitForRegisterBits(unsigned char reg, unsigned char mask);
00082
00083     Version getVersion();
00084
00085     bool performSelfTest();
00086
00087     void setBitFraming(unsigned char rxAlign, unsigned char txLastBits);
00088
00089     unsigned char getCollisionPosition();
00090
00091     void setupForAnticollision();
00092
00093     int readRegisterBlock(unsigned char reg, unsigned char *buf, unsigned char len);
00094
00095     int readRegisterBlock(unsigned char reg, unsigned char *buf, unsigned char len,
unsigned char rxAlign);
00096
00097     unsigned char writeRegisterBlock(unsigned char reg, unsigned char *buf, unsigned char
len);
00098
00099     void turnOffEncryption();
00100
00101     bool hasValidCrc(unsigned char *buf, unsigned char len);
00102
00103 };
00104
00105 #endif // __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_READER_MFRC531_H__

```

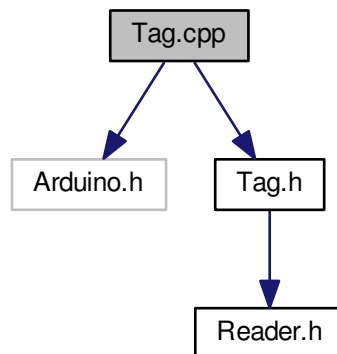
5.19 Tag.cpp File Reference

```

#include <Arduino.h>
#include "Tag.h"

```

Include dependency graph for Tag.cpp:



5.20 Tag.cpp

```

00001
00007 #include <Arduino.h>
00008 #include "Tag.h"
00009
00010 Tag::Tag(Reader *reader)
00011     : reader(reader), tagType(MIFARE_UNKNOWN), uid( { 0 } ),
00012       supportsAnticollision(false), state(POWER_OFF),
00013       keyType(KEY_A), key(NULL), sectorTrailerProtected(
00014           true) {
00015 }
00016
00017 Tag::~Tag() {
00018 }
00019
00020 Tag::Uid Tag::getUid() {
00021     return uid;
00022 }
00023
00024 bool Tag::hasAnticollisionSupport() {
00025     return supportsAnticollision;
00026 }
00027
00028 Tag::TagType Tag::getTagType() {
00029     return tagType;
00030 }
00031
00032 void Tag::setState(Tag::State state) {
00033     this->state = state;
00034 }
00035
00036 Tag::State Tag::getState() {
00037     return state;
00038 }
00039
00040 bool Tag::activate() {
00041     return request() && hasAnticollisionSupport() &&
00042         select();
00043 }
00044
00045 bool Tag::activateWakeUp() {
00046     return wakeUp() && hasAnticollisionSupport() &&
00047         select();
00048 }
00049
00050 bool Tag::detect(unsigned char command) {
00051     unsigned char buf[2] = { command, 0x00 };
00052     reader->turnOffEncryption();
00053     reader->setBitFraming(0, 0x07);
00054     bool ok = reader->tranceive(buf, buf, 1) >= 0;
00055 }
  
```

```

00051     if (ok) {
00052         setState(READY);
00053         supportsAnticollision = buf[0] &
TAG_ATQA_ANTICOLLISION_BIT;
00054     }
00055     return ok;
00056 }
00057
00058 bool Tag::request() {
00059     return detect(REQUEST);
00060 }
00061
00062 bool Tag::wakeUp() {
00063     return detect(WAKE_UP);
00064 }
00065
00066 bool Tag::select() {
00067     if (getState() != READY) {
00068         return false;
00069     }
00070     Command cascadeLevels[3] = { SEL_CL1, SEL_CL2, SEL_CL3 };
00071     unsigned char collisionPosistion = 0, knownBytes, lastBits, send[9] = { 0 }, receive[9] = { 0 }, *p = &
uid.uid[0];
00072     Reader::Error error;
00073     bool needNextCascadeLevel = true;
00074     uid.size = 0;
00075     reader->setuptForAnticollision();
00076
00077     // Loop for each cascade levels.
00078     // Each cascade level we receive 4 bytes corresponding to the CT + ID at
00079     // that cascade level, followed by the BCC byte (xor of the id).
00080     for (unsigned char k = 0; needNextCascadeLevel && k < sizeof(cascadeLevels); k++) {
00081
00082         do {
00083             lastBits = collisionPosistion % 8;
00084             knownBytes = (collisionPosistion / 8) + (lastBits ? 1 : 0);
00085             send[0] = cascadeLevels[k];
00086             send[1] = computeNvb(collisionPosistion);
00087             memcpy(&send[2], receive, knownBytes);
00088             reader->setBitFraming(lastBits, lastBits);
00089             reader->tranceive(send, receive, knownBytes + 2);
00090             collisionPosistion = 0;
00091             error = (Reader::Error) reader->getLastError();
00092             if (error != Reader::NO_ERROR && error !=
Reader::COLLISION_ERROR) {
00093                 setState(IDLE);
00094                 return false;
00095             }
00096             if (error == Reader::COLLISION_ERROR) {
00097                 collisionPosistion = reader->getCollisionPosition();
00098             } else {
00099
00100                 // End of i_nth iteration.
00101                 send[1] = 0x70;
00102                 memcpy(&send[2], receive, 0x05);
00103                 reader->calculateCrc(send, 7, &send[7]);
00104                 reader->tranceive(send, receive, 0x09);
00105                 if (reader->getLastError() != Reader::NO_ERROR) {
00106                     setState(IDLE);
00107                     return false;
00108                 }
00109                 uid.sak = receive[0];
00110
00111                 // TODO: Need more tests
00112                 needNextCascadeLevel = (uid.sak & TAG_SAK_BIT) > 0;
00113                 unsigned char size = 4 - needNextCascadeLevel;
00114                 memcpy(p, &send[2 + needNextCascadeLevel], size);
00115                 p += size;
00116                 uid.size += size;
00117             }
00118         } while (error == Reader::COLLISION_ERROR);
00119     }
00120     computeTagType();
00121     setState(ACTIVE);
00122     return true;
00123 }
00124
00125 bool Tag::halt() {
00126     unsigned char buf[4] = { HLT_A, 0, 0, 0 };
00127     reader->turnOffEncryption();
00128     setState(HALT);
00129     reader->calculateCrc(buf, 2, &buf[2]);
00130     reader->tranceive(buf, buf, 4);
00131
00132     // If the PICC responds with any modulation during a period of 1 ms after the end of the frame
00133     // containing the
00134     // HLTA command, this response shall be interpreted as 'not acknowledge'.

```

```

00134     return reader->getLastError() == Reader::TIMEOUT_ERROR;
00135 }
00136
00137 bool Tag::authenticate(unsigned char address, KeyType type, unsigned char *
key) {
00138     unsigned char buf[12];
00139     if (getState() != ACTIVE) {
00140         return false;
00141     }
00142     buf[0] = (type == KEY_A) ? AUTH_KEY_A : AUTH_KEY_B;
00143     buf[1] = address;
00144     for (unsigned char i = 0; i < TAG_KEY_SIZE; i++) {
00145         buf[2 + i] = key[i];
00146     }
00147     for (unsigned char i = 0; i < 4; i++) {
00148         buf[8 + i] = uid.uid[i];
00149     }
00150     return reader->authenticate(buf) >= 0;
00151 }
00152
00153 bool Tag::readBlock(unsigned char address, unsigned char *buf) {
00154     if (key != NULL && !authenticate(address, keyType, key)) {
00155         return false;
00156     }
00157     buf[0] = READ;
00158     buf[1] = address;
00159     reader->calculateCrc(buf, 2, &buf[2]);
00160     return reader->tranceive(buf, buf, 4, true) == 18;
00161 }
00162
00163 bool Tag::writeBlock(unsigned char address, unsigned char *buf) {
00164     unsigned char cmd[4];
00165     if (isAddressSectorTrailer(address) &&
sectorTrailerProtected) {
00166         return false;
00167     }
00168     if (key != NULL && !authenticate(address, keyType, key)) {
00169         return false;
00170     }
00171     cmd[0] = WRITE;
00172     cmd[1] = address;
00173     reader->calculateCrc(cmd, 2, &cmd[2]);
00174     reader->tranceive(cmd, cmd, 4);
00175     if (reader->getLastError() == Reader::NACK) {
00176         return false;
00177     }
00178     reader->calculateCrc(buf, 16, &buf[16]);
00179     reader->tranceive(buf, buf, 18);
00180     return reader->getLastError() != Reader::NACK;
00181 }
00182
00183 bool Tag::readBlockSlice(unsigned char address, unsigned char from, unsigned char len,
unsigned char *buf) {
00184     unsigned char receive[18];
00185     if (len == 0 || from + len > 16) {
00186         return false;
00187     }
00188     if (!readBlock(address, receive)) {
00189         return false;
00190     }
00191     memcpy(buf, &receive[from], len);
00192     return true;
00193 }
00194
00195 bool Tag::writeBlockSlice(unsigned char address, unsigned char from, unsigned char len,
unsigned char *buf) {
00196     unsigned char receive[18];
00197     if (len == 0 || from + len > 16) {
00198         return false;
00199     }
00200     if (!readBlock(address, receive)) {
00201         return false;
00202     }
00203     memcpy(&receive[from], buf, len);
00204     return writeBlock(address, receive);
00205 }
00206
00207 int Tag::readByte(unsigned char address, unsigned char pos) {
00208     unsigned char buf[18];
00209     if (!readBlock(address, buf)) {
00210         return -1;
00211     }
00212     return buf[pos];
00213 }
00214 }
00215
00216 bool Tag::writeByte(unsigned char address, unsigned char pos, unsigned char value) {

```

```

00217
00218     unsigned char buf[18];
00219     if (!readBlock(address, buf)) {
00220         return false;
00221     }
00222     buf[pos] = value;
00223     return writeBlock(address, buf);
00224 }
00225
00226 bool Tag::decrement() {
00227     return true;
00228 }
00229
00230 bool Tag::increment() {
00231     return true;
00232 }
00233
00234 bool Tag::restore() {
00235     return true;
00236 }
00237
00238 bool Tag::transfer() {
00239     return true;
00240 }
00241
00242 bool Tag::setBlockType(unsigned char address, BlockType type) {
00243
00244     return true;
00245 }
00246
00247 bool Tag::readAccessBits(unsigned char sector, unsigned char *buf) {
00248     return readBlockSlice(getSectorTrailerAddress(sector), 6, 10, buf)
;
00249 }
00250
00251 bool Tag::writeAccessBits(unsigned char sector, unsigned char *buf) {
00252     return writeBlockSlice(getSectorTrailerAddress(sector), 6, 10,
buf);
00253 }
00254
00255 bool Tag::setBlockPermission(unsigned char address, unsigned char permission) {
00256     return true;
00257 }
00258
00259 bool Tag::writeKey(unsigned char sector, KeyType type, unsigned char *
key) {
00260     unsigned from = TAG_KEY_TO_POS(key);
00261     return writeBlockSlice(getSectorTrailerAddress(sector), from,
TAG_KEY_SIZE, key);
00262 }
00263
00264 bool Tag::readKey(unsigned char sector, KeyType type, unsigned char *
key) {
00265     unsigned from = TAG_KEY_TO_POS(key);
00266     return readBlockSlice(getSectorTrailerAddress(sector), from,
TAG_KEY_SIZE, key);
00267 }
00268
00269 void Tag::setupAuthenticationKey(KeyType
keyType, unsigned char *key) {
00270     this->keyType = keyType;
00271     this->key = key;
00272 }
00273
00274 unsigned char Tag::computeNvb(unsigned char collisionPos) {
00275     unsigned char bytes = collisionPos / 8;
00276     unsigned char bits = collisionPos % 8;
00277     return ((bytes << 4) & 0xf0) | (bits & 0x0f) + 0x20;
00278 }
00279
00280 void Tag::computeTagType() {
00281     switch (uid.sak & 0x7f) {
00282     case 0x04:
00283         tagType = MIFARE_NOT_COMPLETE;
00284         break;
00285     case 0x09:
00286         tagType = MIFARE_MINI;
00287         break;
00288     case 0x08:
00289         tagType = MIFARE_1K;
00290         break;
00291     case 0x18:
00292         tagType = MIFARE_4K;
00293         break;
00294     case 0x00:
00295         tagType = MIFARE_UL;
00296         break;

```

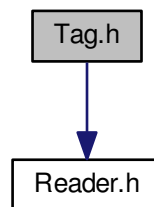


```
00297     case 0x10:
00298     case 0x11:
00299         tagType = MIFARE_PLUS;
00300         break;
00301     default:
00302         tagType = MIFARE_UNKNOWN;
00303     }
00304 }
00305
00306 void Tag::setSectorTrailerProtected(bool protect) {
00307     sectorTrailerProtected = protect;
00308 }
```

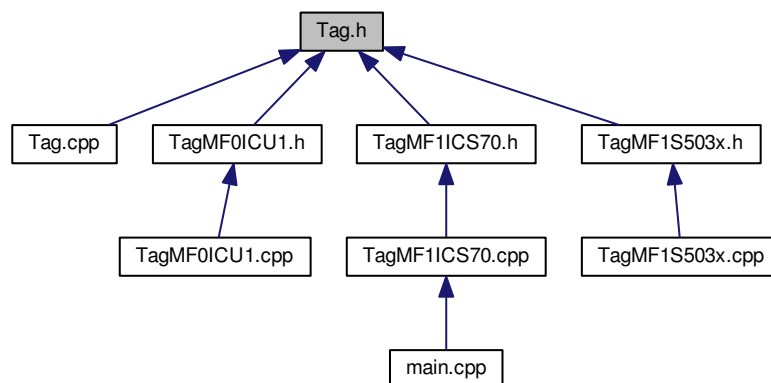
5.21 Tag.h File Reference

```
#include <Reader.h>
```

Include dependency graph for Tag.h:



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



Classes

- class [Tag](#)
- struct [Tag::Uid](#)

Macros

- `#define TAG_SAK_BIT 0x20`
- `#define TAG_ATQA_ANTICOLLISION_BIT 0x04`
- `#define TAG_KEY_SIZE 0x06`
- `#define TAG_DEFAULT_SECTOR_SIZE 0x04`
- `#define TAG_KEY_TO_POS(key) ((type == KEY_A) ? 0 : 10)`

5.21.1 Macro Definition Documentation

5.21.1.1 `#define TAG_ATQA_ANTICOLLISION_BIT 0x04`

Definition at line 13 of file [Tag.h](#).

5.21.1.2 `#define TAG_DEFAULT_SECTOR_SIZE 0x04`

Definition at line 16 of file [Tag.h](#).

5.21.1.3 `#define TAG_KEY_SIZE 0x06`

Definition at line 15 of file [Tag.h](#).

5.21.1.4 `#define TAG_KEY_TO_POS(key) ((type == KEY_A) ? 0 : 10)`

Definition at line 18 of file [Tag.h](#).

5.21.1.5 `#define TAG_SAK_BIT 0x20`

Arduino - Radio Frequency Identification.

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 12 of file [Tag.h](#).

5.22 Tag.h

```

00001
00007 #ifndef __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_TAG_H__
00008 #define __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_TAG_H__ 1
00009
00010 #include <Reader.h>
00011
00012 #define TAG_SAK_BIT 0x20
00013 #define TAG_ATQA_ANTICOLLISION_BIT 0x04
00014
00015 #define TAG_KEY_SIZE 0x06
00016 #define TAG_DEFAULT_SECTOR_SIZE 0x04
00017
00018 #define TAG_KEY_TO_POS(key) ((type == KEY_A) ? 0 : 10)
00019
00020 class Tag {
00021
00022 public:
00023
00024     enum State {
00025         POWER_OFF = 0x00,
00026         IDLE = 0x01,
00027         READY = 0x02,
00028         ACTIVE = 0x03,
00029         HALT = 0x04
00030     };
00031
00032     enum TagType {
00033         MIFARE_UNKNOWN = 0x00,
00034         MIFARE_MINI = 0x01,
00035         MIFARE_1K = 0x02,
00036         MIFARE_4K = 0x03,
00037         MIFARE_UL = 0x04,
00038         MIFARE_PLUS = 0x06,
00039         MIFARE_NOT_COMPLETE = 0xff
00040     };
00041
00042     enum KeyType {
00043         KEY_A = 0x00,
00044         KEY_B = 0x01
00045     };
00046
00047     enum BlockType {
00048         DATA_BLOCK = 0x00,
00049         VALUE_BLOCK = 0x01,
00050     };
00051
00052     // All MIFARE Classic commands use the MIFARE Crypto1 and require an authentication.
00053     enum Command {
00054
00055         // REQuest command, Type A. Invites PICCs in state IDLE to go to READY and prepare for
00056         // anticollision or selection. 7 bit frame.
00057         REQUEST = 0x26,
00058
00059         // Wake-UP command, Type A. Invites PICCs in state IDLE and HALT to go to READY(*) and prepare for
00060         // anticollision or selection. 7 bit frame.
00061         WAKE_UP = 0x52,
00062
00063         // Anti collision/Select, Cascade Level 1
00064         SEL_CL1 = 0x93,
00065
00066         // Anti collision/Select, Cascade Level 2
00067         SEL_CL2 = 0x95,
00068
00069         // Anti collision/Select, Cascade Level 3
00070         SEL_CL3 = 0x97,
00071
00072         // HaLT command, Type A. Instructs an ACTIVE PICC to go to state HALT.
00073         HLT_A = 0x50,
00074
00075         // Perform authentication with Key A.
00076         AUTH_KEY_A = 0x60,
00077
00078         // Perform authentication with Key B.
00079         AUTH_KEY_B = 0x61,
00080
00081         // Reads one 16 byte block from the authenticated sector of the PICC. Also used for MIFARE
00082         // Ultralight.
00083         READ = 0x30,
00084
00085         // Writes one 16 byte block to the authenticated sector of the PICC. Called "COMPATIBILITY WRITE"
00086         // for MIFARE Ultralight.
00087         WRITE = 0xa0,
00088
00089         // Decrements the contents of a block and stores the result in the internal data register.
00090     };

```

```

00086         DECREMENT = 0xc0,
00087
00088         // Increments the contents of a block and stores the result in the internal data register.
00089         INCREMENT = 0xc1,
00090
00091         // Reads the contents of a block into the internal data register.
00092         RESTORE = 0xc2,
00093
00094         // Writes the contents of the internal data register to a block.
00095         TRANSFER = 0xb0,
00096     };
00097
00098     struct Uid {
00099
00100         // Number of bytes in the UID. 4, 7 or 10.
00101         unsigned char size;
00102
00103         unsigned char uid[10];
00104
00105         // The SAK (Select acknowledge) byte returned from the tag after successful selection.
00106         unsigned char sak;
00107     };
00108
00109     Tag(Reader *reader);
00110
00111     virtual ~Tag();
00112
00113     Uid getUid();
00114
00115     bool hasAnticollisionSupport();
00116
00117     TagType getTagType();
00118
00119     void setState(State state);
00120
00121     State getState();
00122
00123     virtual bool detect(unsigned char command);
00124
00125     virtual bool activate();
00126
00127     virtual bool activateWakeUp();
00128
00129     virtual bool request();
00130
00131     virtual bool wakeUp();
00132
00133     virtual bool select();
00134
00135     virtual bool halt();
00136
00137     virtual bool authenticate(unsigned char address, KeyType type, unsigned char *
00138 key);
00139
00140     virtual bool readBlock(unsigned char address, unsigned char *buf);
00141
00142     virtual bool writeBlock(unsigned char address, unsigned char *buf);
00143
00144     virtual bool readBlockSlice(unsigned char address, unsigned char from, unsigned char len,
00145 unsigned char *buf);
00146
00147     virtual bool writeBlockSlice(unsigned char address, unsigned char from, unsigned char
00148 len, unsigned char *buf);
00149
00150     virtual int readByte(unsigned char address, unsigned char pos);
00151
00152     virtual bool writeByte(unsigned char address, unsigned char pos, unsigned char value);
00153
00154     virtual bool decrement();
00155
00156     virtual bool increment();
00157
00158     virtual bool restore();
00159
00160     virtual bool transfer();
00161
00162     virtual bool setBlockType(unsigned char address, BlockType type);
00163
00164     virtual bool readAccessBits(unsigned char sector, unsigned char *buf);
00165
00166     virtual bool writeAccessBits(unsigned char sector, unsigned char *buf);
00167
00168     virtual bool setBlockPermission(unsigned char address, unsigned char permission);
00169
00170     virtual bool writeKey(unsigned char sector, KeyType type, unsigned char *
00171 key);
00172
00173
00174
00175
00176
00177
00178
00179
00180
00181
00182

```

```

00183     virtual bool readKey(unsigned char sector, KeyType type, unsigned char *
00184     key);
00185     virtual void setupAuthenticationKey(KeyType
00186     keyType, unsigned char *key);
00187     virtual void setSectorTrailerProtected(bool protect);
00188     protected:
00189     Reader *reader;
00190     TagType tagType;
00191     Uid uid;
00192     bool supportsAnticollision;
00193     State state;
00194     KeyType keyType;
00195     unsigned char *key;
00196     bool sectorTrailerProtected;
00197     unsigned char computeNvb(unsigned char collisionPos);
00198     virtual unsigned char getSectorSize(unsigned char sector) = 0;
00199     virtual unsigned char isAddressSectorTrailer(unsigned char address) = 0;
00200     virtual unsigned char addressToSector(unsigned char address) = 0;
00201     virtual unsigned char getSectorTrailerAddress(unsigned char sector) = 0;
00202     void computeTagType();
00203 };
00204 #endif // __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_TAG_H__

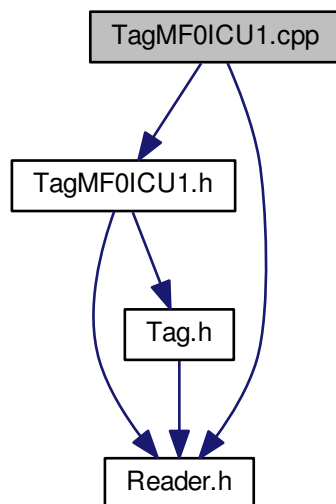
```

5.23 TagMF0ICU1.cpp File Reference

```
#include "TagMF0ICU1.h"
```

```
#include <Reader.h>
```

Include dependency graph for TagMF0ICU1.cpp:



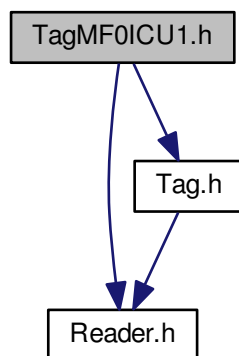
5.24 TagMF0ICU1.cpp

```
00001 #include "TagMF0ICU1.h"
00002 #include <Reader.h>
00003
00004 TagMF0ICU1::TagMF0ICU1(Reader *reader)
00005     : Tag(reader) {
00006 }
00007
```

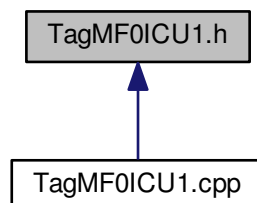
5.25 TagMF0ICU1.h File Reference

```
#include <Reader.h>
#include <Tag.h>
```

Include dependency graph for TagMF0ICU1.h:



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



Classes

- class [TagMF0ICU1](#)

5.26 TagMF0ICU1.h

```

00001
00007 #ifndef __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_TAG_MF0ICU1_H__
00008 #define __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_TAG_MF0ICU1_H__ 1
00009
00010 #include <Reader.h>
00011 #include <Tag.h>
00012
00016 class TagMF0ICU1: public Tag {
00017
00018 public:
00019
00020     enum Permission {
00021         LEVEL_0 = 0x00,
00022         LEVEL_1 = 0x01,
00023         LEVEL_2 = 0x02,
00024         LEVEL_3 = 0x03,
00025         LEVEL_4 = 0x04,
00026         LEVEL_5 = 0x05,
00027         LEVEL_6 = 0x06,
00028         LEVEL_7 = 0x07
00029     };
00030
00031     TagMF0ICU1(Reader *reader);
00032 };
00033
00034 #endif // __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_TAG_MF1S503X_H__

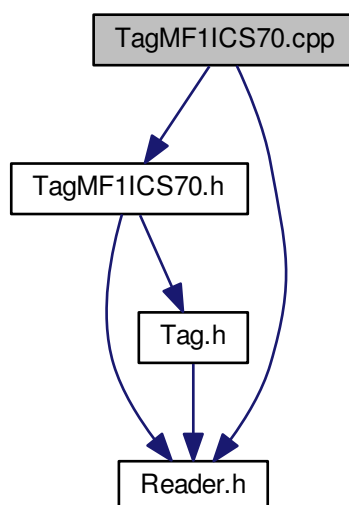
```

5.27 TagMF1ICS70.cpp File Reference

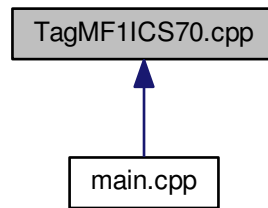
```
#include "TagMF1ICS70.h"
```

```
#include <Reader.h>
```

Include dependency graph for TagMF1ICS70.cpp:



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



5.28 TagMF1ICS70.cpp

```

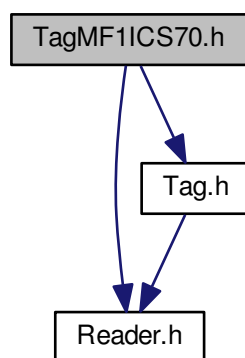
00001 #include "TagMF1ICS70.h"
00002 #include <Reader.h>
00003
00004 TagMF1ICS70::TagMF1ICS70(Reader *reader)
00005     : Tag(reader) {
00006 }
00007
00008 bool TagMF1ICS70::writeBlock(unsigned char address, unsigned char *buf) {
00009     if (getTagType() != MIFARE_4K) {
00010         return false;
00011     }
00012     return Tag::writeBlock(address, buf);
00013 }
00014
00015 unsigned char TagMF1ICS70::getSectorSize(unsigned char sector) {
00016     unsigned char size = TAG_MF1ICS70_LOW_SECTOR_SIZE;
00017     if (sector >= TAG_MF1ICS70_LOW_SECTOR_COUNT) {
00018         size = TAG_MF1ICS70_HIGH_SECTOR_SIZE;
00019     }
00020     return size;
00021 }
00022
00023 unsigned char TagMF1ICS70::addressToSector(unsigned char address) {
00024     unsigned char sector = 0;
00025     if (address < TAG_MF1ICS70_LOW_MEMORY_SIZE) {
00026         sector = address / TAG_MF1ICS70_LOW_SECTOR_SIZE;
00027     } else {
00028         address -= TAG_MF1ICS70_LOW_MEMORY_SIZE;
00029         sector = TAG_MF1ICS70_LOW_SECTOR_COUNT + (address /
TAG_MF1ICS70_HIGH_SECTOR_SIZE);
00030     }
00031     return sector;
00032 }
00033
00034 unsigned char TagMF1ICS70::isAddressSectorTrailer(unsigned char address)
{
00035     unsigned char sectorSize = getSectorSize(addressToSector(address));
00036     if (address >= TAG_MF1ICS70_LOW_MEMORY_SIZE) {
00037         address -= TAG_MF1ICS70_LOW_MEMORY_SIZE;
00038     }
00039     return ((address % sectorSize) == (sectorSize - 1));
00040 }
00041
00042 unsigned char TagMF1ICS70::getSectorTrailerAddress(unsigned char sector
) {
00043     unsigned char sectorSize, offset = 0;
00044     sectorSize = getSectorSize(sector);
00045     if (sector >= TAG_MF1ICS70_LOW_SECTOR_COUNT) {
00046         offset = TAG_MF1ICS70_LOW_MEMORY_SIZE;
00047         sector -= TAG_MF1ICS70_LOW_SECTOR_COUNT;
00048     }
00049     return offset + (sector * sectorSize) + (sectorSize - 1);
00050 }
  
```


5.29 TagMF1ICS70.h File Reference

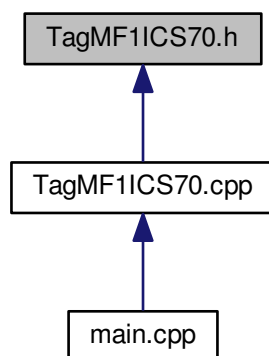
```
#include <Reader.h>
```

```
#include <Tag.h>
```

Include dependency graph for TagMF1ICS70.h:



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



Classes

- class [TagMF1ICS70](#)

Macros

- `#define TAG_MF1ICS70_LOW_SECTOR_COUNT 32`
- `#define TAG_MF1ICS70_HIGH_SECTOR_COUNT 8`
- `#define TAG_MF1ICS70_LOW_SECTOR_SIZE 4`
- `#define TAG_MF1ICS70_HIGH_SECTOR_SIZE 16`
- `#define TAG_MF1ICS70_LOW_MEMORY_SIZE TAG_MF1ICS70_LOW_SECTOR_COUNT * TAG_MF1ICS70_LOW_SECTOR_SIZE`

5.29.1 Macro Definition Documentation

5.29.1.1 `#define TAG_MF1ICS70_HIGH_SECTOR_COUNT 8`

Definition at line 17 of file [TagMF1ICS70.h](#).

5.29.1.2 `#define TAG_MF1ICS70_HIGH_SECTOR_SIZE 16`

Definition at line 20 of file [TagMF1ICS70.h](#).

5.29.1.3 `#define TAG_MF1ICS70_LOW_MEMORY_SIZE TAG_MF1ICS70_LOW_SECTOR_COUNT * TAG_MF1ICS70_LOW_SECTOR_SIZE`

Definition at line 22 of file [TagMF1ICS70.h](#).

5.29.1.4 `#define TAG_MF1ICS70_LOW_SECTOR_COUNT 32`

Arduino - Radio Frequency Identification MFRC522.

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 16 of file [TagMF1ICS70.h](#).

5.29.1.5 `#define TAG_MF1ICS70_LOW_SECTOR_SIZE 4`

Definition at line 19 of file [TagMF1ICS70.h](#).

5.30 TagMF1ICS70.h

```

00001
00007 #ifndef __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_TAG_MF1ICS70_H__
00008 #define __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_TAG_MF1ICS70_H__ 1
00009
00010 #include <Reader.h>
00011 #include <Tag.h>
00012
00013 // The 4 kByte EEPROM memory is organized in 32 sectors with 4 blocks
00014 // and in 8 sectors with 16 blocks. One block consists of 16 bytes.
00015
00016 #define TAG_MF1ICS70_LOW_SECTOR_COUNT      32
00017 #define TAG_MF1ICS70_HIGH_SECTOR_COUNT    8
00018
00019 #define TAG_MF1ICS70_LOW_SECTOR_SIZE      4
00020 #define TAG_MF1ICS70_HIGH_SECTOR_SIZE    16
00021
00022 #define TAG_MF1ICS70_LOW_MEMORY_SIZE      TAG_MF1ICS70_LOW_SECTOR_COUNT * TAG_MF1ICS70_LOW_SECTOR_SIZE
00023
00024 class TagMF1ICS70: public Tag {
00025 public:
00026
00027     enum Permission {
00028         LEVEL_0 = 0x00,
00029         LEVEL_1 = 0x01,
00030         LEVEL_2 = 0x02,
00031         LEVEL_3 = 0x03,
00032         LEVEL_4 = 0x04,
00033         LEVEL_5 = 0x05,
00034         LEVEL_6 = 0x06,
00035         LEVEL_7 = 0x07
00036     };
00037
00038     TagMF1ICS70(Reader *reader);
00039
00040     bool writeBlock(unsigned char address, unsigned char *buf);
00041
00042     unsigned char getSectorSize(unsigned char sector);
00043
00044     unsigned char isAddressSectorTrailer(unsigned char address);
00045
00046     unsigned char addressToSector(unsigned char address);
00047
00048     unsigned char getSectorTrailerAddress(unsigned char sector);
00049 };
00050
00051
00052 #endif // __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_TAG_MF1S503X_H__

```

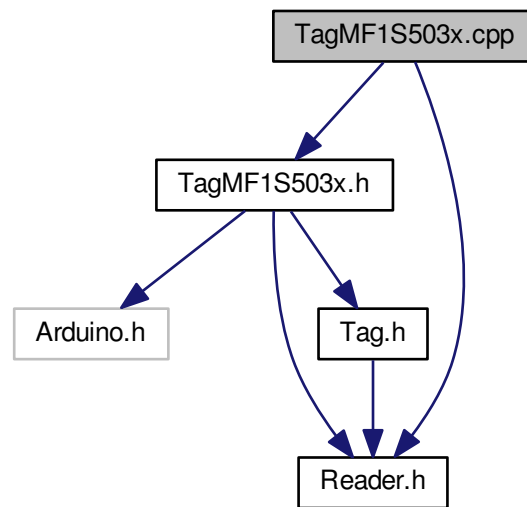
5.31 TagMF1S503x.cpp File Reference

```

#include "TagMF1S503x.h"
#include <Reader.h>

```

Include dependency graph for TagMF1S503x.cpp:



5.32 TagMF1S503x.cpp

```

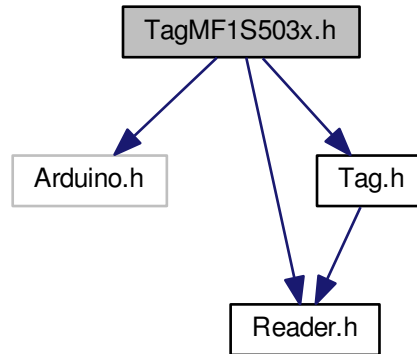
00001 #include "TagMF1S503x.h"
00002 #include <Reader.h>
00003
00004 TagMF1S503x::TagMF1S503x(Reader *reader)
00005     : Tag(reader) {
00006 }
00007
00008 bool TagMF1S503x::writeBlock(unsigned char address, unsigned char *buf) {
00009     if (getTagType() != MIFARE_1K) {
00010         return false;
00011     }
00012     return Tag::writeBlock(address, buf);
00013 }
00014
00015 unsigned char TagMF1S503x::getSectorSize(unsigned char sector) {
00016     return TAG_MF1S503X_SECTOR_SIZE;
00017 }
00018
00019 unsigned char TagMF1S503x::addressToSector(unsigned char address) {
00020     return address / TAG_MF1S503X_SECTOR_SIZE;
00021 }
00022
00023 unsigned char TagMF1S503x::isAddressSectorTrailer(unsigned char address)
00024 {
00025     return ((address % TAG_MF1S503X_SECTOR_SIZE) == (TAG_MF1S503X_SECTOR_SIZE - 1))
00026 ;
00027 }
00028 unsigned char TagMF1S503x::getSectorTrailerAddress(unsigned char sector)
00029 {
00030     return (sector * TAG_MF1S503X_SECTOR_SIZE) + (TAG_MF1S503X_SECTOR_SIZE - 1);
00031 }
  
```

5.33 TagMF1S503x.h File Reference

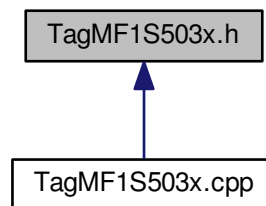
```
#include <Arduino.h>
```

```
#include <Reader.h>
#include <Tag.h>
```

Include dependency graph for TagMF1S503x.h:



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



Classes

- class [TagMF1S503x](#)

Macros

- `#define TAG_MF1S503X_SECTOR_SIZE 4`

5.33.1 Macro Definition Documentation

5.33.1.1 `#define TAG_MF1S503X_SECTOR_SIZE 4`

Arduino - Radio Frequency Identification MFRC522.

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 14 of file [TagMF1S503x.h](#).

5.34 TagMF1S503x.h

```
00001
00007 #ifndef __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_TAG_MF1S503X_H__
00008 #define __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_TAG_MF1S503X_H__ 1
00009
00010 #include <Arduino.h>
00011 #include <Reader.h>
00012 #include <Tag.h>
00013
00014 #define TAG_MF1S503X_SECTOR_SIZE    4
00015
00016 class TagMF1S503x: public Tag {
00017 public:
00018
00019     enum Permission {
00020         LEVEL_0 = 0x00,
00021         LEVEL_1 = 0x01,
00022         LEVEL_2 = 0x02,
00023         LEVEL_3 = 0x03,
00024         LEVEL_4 = 0x04,
00025         LEVEL_5 = 0x05,
00026         LEVEL_6 = 0x06,
00027         LEVEL_7 = 0x07
00028     };
00029
00030     TagMF1S503x(Reader *reader);
00031
00032     bool writeBlock(unsigned char address, unsigned char *buf);
00033
00034     unsigned char getSectorSize(unsigned char sector);
00035
00036     unsigned char isAddressSectorTrailer(unsigned char address);
00037
00038     unsigned char addressToSector(unsigned char address);
00039
00040     unsigned char getSectorTrailerAddress(unsigned char sector);
00041 };
00042
00043 #endif // __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_TAG_MF1S503X_H__
```

Index

- ~Reader
 - Reader, [26](#)
- ~ReaderMFRC522
 - ReaderMFRC522, [35](#)
- ~ReaderMFRC530
 - ReaderMFRC530, [47](#)
- ~ReaderMFRC531
 - ReaderMFRC531, [52](#)
- ~Tag
 - Tag, [67](#)
- ACTIVE
 - Tag, [67](#)
- ACK
 - ReaderMFRC522, [44](#)
- ADD_IQ
 - ReaderMFRC522::DEMODObits, [14](#)
- ANALOG_TEST
 - ReaderMFRC522, [35](#)
- AUTH_KEY_A
 - Tag, [66](#)
- AUTH_KEY_B
 - Tag, [66](#)
- AUTO_TEST_ENABLE
 - ReaderMFRC522, [33](#)
- AUTO_TEST
 - ReaderMFRC522, [35](#)
- activate
 - Tag, [67](#)
- activateWakeUp
 - Tag, [67](#)
- addressToSector
 - Tag, [68](#)
 - TagMF1ICS70, [75](#)
 - TagMF1S503x, [78](#)
- authenticate
 - Reader, [26](#)
 - ReaderMFRC522, [35](#)
 - ReaderMFRC530, [47](#)
 - ReaderMFRC531, [52](#)
 - Tag, [68](#)
- BIT_FRAMING_START_SEND
 - ReaderMFRC522, [33](#)
- BIT_FRAMING
 - ReaderMFRC522, [34](#)
- BR_T0
 - ReaderMFRC522::SERIAL_SPEEDbits, [59](#)
- BR_T1
 - ReaderMFRC522::SERIAL_SPEEDbits, [59](#)
- BUFFER_OVFL
 - ReaderMFRC522::ERRORbits, [18](#)
- BlockType
 - Tag, [66](#)
- CALC_CRC
 - ReaderMFRC522, [32](#)
- CHIPTYPE
 - ReaderMFRC522::VERSIONbits, [84](#)
- CLONE
 - ReaderMFRC522, [35](#)
- COLL_ERR
 - ReaderMFRC522::ERRORbits, [18](#)
- COLL_LEVEL
 - ReaderMFRC522::RX_THRESHOLDbits, [58](#)
- COLL_POS_NOT_VALID
 - ReaderMFRC522::COLLbits, [8](#)
- COLL_POS
 - ReaderMFRC522::COLLbits, [7](#)
- COLL_VALUES_AFTER_COLL
 - ReaderMFRC522, [33](#)
- COLLISION_ERROR
 - Reader, [26](#)
- COLL
 - ReaderMFRC522, [34](#)
- COM_ALL_IRQ
 - ReaderMFRC522, [33](#)
- COM_ERR_IRQ
 - ReaderMFRC522, [33](#)
- COM_HI_ALERT_IRQ
 - ReaderMFRC522, [33](#)
- COM_I_EN_INTERRUPT_EN
 - ReaderMFRC522, [33](#)
- COM_I_EN
 - ReaderMFRC522, [34](#)
- COM_IDLE_IRQ
 - ReaderMFRC522, [33](#)
- COM_IRQ_ALL_IRQ
 - ReaderMFRC522, [33](#)
- COM_IRQ_ERR_IRQ
 - ReaderMFRC522, [33](#)
- COM_IRQ_HI_ALERT_IRQ
 - ReaderMFRC522, [33](#)
- COM_IRQ_IDLE_IRQ
 - ReaderMFRC522, [33](#)
- COM_IRQ_LO_ALERT_IRQ
 - ReaderMFRC522, [33](#)
- COM_IRQ_RX_IRQ
 - ReaderMFRC522, [33](#)
- COM_IRQ_SET1
 - ReaderMFRC522, [33](#)
- COM_IRQ_TIMER_IRQ
 - ReaderMFRC522, [33](#)
- COM_IRQ_TX_IRQ
 - ReaderMFRC522, [33](#)
- COM_IRQ
 - ReaderMFRC522, [34](#)
- COM_LO_ALERT_IRQ
 - ReaderMFRC522, [33](#)
- COM_RX_IRQ
 - ReaderMFRC522, [33](#)

COM_TIMER_IRQ
 ReaderMFRC522, [33](#)
 COM_TX_IRQ
 ReaderMFRC522, [33](#)
 COMMAND
 ReaderMFRC522, [34](#)
 ReaderMFRC522::COMMANDbits, [11](#)
 COMMUNICATION_ERROR
 Reader, [26](#)
 CONTROL_T_START_NOW
 ReaderMFRC522, [33](#)
 CONTROL_T_STOP_NOW
 ReaderMFRC522, [33](#)
 CONTROL
 ReaderMFRC522, [34](#)
 CRC_ERROR
 Reader, [26](#)
 CRC_ERR
 ReaderMFRC522::ERRORbits, [18](#)
 CRC_I_EN
 ReaderMFRC522::DIV_I_ENbits, [15](#)
 CRC_IRQ
 ReaderMFRC522::DIV_IRQbits, [16](#)
 CRC_OK
 ReaderMFRC522::STATUS1bits, [60](#)
 CRC_PRESET
 ReaderMFRC522::MODEbits, [23](#)
 CRC_READY
 ReaderMFRC522::STATUS1bits, [60](#)
 CRC_RESULT_HIGH
 ReaderMFRC522, [34](#)
 CRC_RESULT_LOW
 ReaderMFRC522, [34](#)
 CW_GS_N
 ReaderMFRC522::GS_Nbits, [20](#)
 CW_GS_P
 ReaderMFRC522, [34](#)
 ReaderMFRC522::CW_GS_Pbits, [13](#)
 calculateCrc
 Reader, [26](#)
 ReaderMFRC522, [36](#)
 ReaderMFRC530, [47](#)
 ReaderMFRC531, [52](#)
 char
 ReaderMFRC522::BIT_FRAMINGbits, [6](#)
 ReaderMFRC522::COLLbits, [7](#)
 ReaderMFRC522::COMMANDbits, [11](#)
 ReaderMFRC522::CONTROLbits, [12](#)
 ReaderMFRC522::CW_GS_Pbits, [13](#)
 ReaderMFRC522::DIV_I_ENbits, [15](#)
 ReaderMFRC522::DIV_IRQbits, [16](#)
 ReaderMFRC522::ERRORbits, [18](#)
 ReaderMFRC522::MF_RXbits, [21](#)
 ReaderMFRC522::MF_TXbits, [21](#)
 ReaderMFRC522::MOD_GS_Pbits, [22](#)
 ReaderMFRC522::MODEbits, [23](#)
 ReaderMFRC522::RF_CFGbits, [55](#)
 ReaderMFRC522::RX_MODEbits, [56](#)
 ReaderMFRC522::RX_THRESHOLDbits, [58](#)
 ReaderMFRC522::STATUS1bits, [60](#)
 ReaderMFRC522::STATUS2bits, [62](#)
 ReaderMFRC522::TX_ASKbits, [79](#)
 ReaderMFRC522::TX_CONTROLbits, [80](#)
 ReaderMFRC522::TX_MODEbits, [81](#)
 ReaderMFRC522::TX_SELbits, [82](#)
 ReaderMFRC522::WATER_LEVELbits, [85](#)
 clearInterrupt
 Reader, [26](#)
 ReaderMFRC522, [36](#)
 ReaderMFRC530, [47](#)
 ReaderMFRC531, [52](#)
 clearLastError
 Reader, [26](#)
 Command
 ReaderMFRC522, [32](#)
 ReaderMFRC530, [46](#)
 ReaderMFRC531, [52](#)
 Tag, [66](#)
 communicate
 Reader, [26](#), [27](#)
 ReaderMFRC522, [37](#)
 ReaderMFRC530, [47](#)
 ReaderMFRC531, [52](#), [53](#)
 computeNvb
 Tag, [68](#)
 computeTagType
 Tag, [68](#)
 configureTimer
 Reader, [27](#)
 ReaderMFRC522, [37](#)
 ReaderMFRC530, [47](#)
 ReaderMFRC531, [53](#)
 DATA_BLOCK
 Tag, [66](#)
 DECREMENT
 Tag, [66](#)
 DEMOD
 ReaderMFRC522, [34](#)
 DIV_ALL_IRQ
 ReaderMFRC522, [33](#)
 DIV_CRC_IRQ
 ReaderMFRC522, [33](#)
 DIV_I_EN_CRC_I_EN
 ReaderMFRC522, [33](#)
 DIV_I_EN_INTERRUPT_EN
 ReaderMFRC522, [33](#)
 DIV_I_EN_MFIN_ACT_I_EN
 ReaderMFRC522, [33](#)
 DIV_I_EN
 ReaderMFRC522, [34](#)
 DIV_IRQ_ALL_IRQ
 ReaderMFRC522, [33](#)
 DIV_IRQ_CRC_IRQ
 ReaderMFRC522, [33](#)
 DIV_IRQ_MFIN_ACT_IRQ
 ReaderMFRC522, [33](#)

- DIV_IRQ_SET2
 - ReaderMFRC522, [33](#)
- DIV_IRQ
 - ReaderMFRC522, [34](#)
- DIV_MFIN_ACT_IRQ
 - ReaderMFRC522, [33](#)
- decrement
 - Tag, [68](#)
- detect
 - Tag, [68](#)
- device
 - ReaderMFRC522, [44](#)
- disableInterrupt
 - Reader, [27](#)
 - ReaderMFRC522, [38](#)
 - ReaderMFRC530, [48](#)
 - ReaderMFRC531, [53](#)
- ERR_I_EN
 - ReaderMFRC522::COM_I_ENbits, [9](#)
- ERR_IRQ
 - ReaderMFRC522::COM_IRQbits, [10](#)
- ERROR
 - ReaderMFRC522, [34](#)
- enableInterrupt
 - Reader, [27](#)
 - ReaderMFRC522, [38](#)
 - ReaderMFRC530, [48](#)
 - ReaderMFRC531, [53](#)
- Error
 - Reader, [26](#)
- FIFO_DATA
 - ReaderMFRC522, [34](#)
- FIFO_LEVEL_FIFO_LEVEL
 - ReaderMFRC522, [33](#)
- FIFO_LEVEL_FLUSH_BUFFER
 - ReaderMFRC522, [33](#)
- FIFO_LEVEL
 - ReaderMFRC522, [34](#)
 - ReaderMFRC522::FIFO_LEVELbits, [19](#)
- FIX_IQ
 - ReaderMFRC522::DEMODObits, [14](#)
- FLUSH_BUFFER
 - ReaderMFRC522::FIFO_LEVELbits, [19](#)
- FORCE_100_ASK
 - ReaderMFRC522::TX_ASKbits, [79](#)
- flushQueue
 - Reader, [27](#)
 - ReaderMFRC522, [38](#)
 - ReaderMFRC530, [48](#)
 - ReaderMFRC531, [53](#)
- GENERAL_ERROR
 - Reader, [26](#)
- GENERATE_RANDOM_ID
 - ReaderMFRC522, [32](#)
- GS_N
 - ReaderMFRC522, [34](#)
- generateRandomId
 - Reader, [27](#)
 - ReaderMFRC522, [38](#)
 - ReaderMFRC530, [48](#)
 - ReaderMFRC531, [53](#)
- getCollisionPosition
 - Reader, [27](#)
 - ReaderMFRC522, [39](#)
 - ReaderMFRC530, [48](#)
 - ReaderMFRC531, [53](#)
- getLastError
 - Reader, [27](#)
- getSectorSize
 - Tag, [68](#)
 - TagMF1ICS70, [75](#)
 - TagMF1S503x, [78](#)
- getSectorTrailerAddress
 - Tag, [68](#)
 - TagMF1ICS70, [75](#)
 - TagMF1S503x, [78](#)
- getState
 - Tag, [68](#)
- getTagType
 - Tag, [68](#)
- getUid
 - Tag, [69](#)
- getVersion
 - ReaderMFRC522, [39](#)
 - ReaderMFRC530, [48](#)
 - ReaderMFRC531, [53](#)
- HALT
 - Tag, [67](#)
- HI_ALERT_I_EN
 - ReaderMFRC522::COM_I_ENbits, [9](#)
- HI_ALERT_IRQ
 - ReaderMFRC522::COM_IRQbits, [10](#)
- HI_ALERT
 - ReaderMFRC522::STATUS1bits, [60](#)
- HLT_A
 - Tag, [66](#)
- halt
 - Tag, [69](#)
- hasAnticollisionSupport
 - Tag, [69](#)
- hasValidCrc
 - Reader, [27](#)
 - ReaderMFRC522, [39](#)
 - ReaderMFRC530, [48](#)
 - ReaderMFRC531, [53](#)
- I2C_FORCE_HS
 - ReaderMFRC522::STATUS2bits, [62](#)
- I_RQ_INV
 - ReaderMFRC522::COM_I_ENbits, [9](#)
- IDLE_I_EN
 - ReaderMFRC522::COM_I_ENbits, [9](#)
- IDLE_IRQ
 - ReaderMFRC522::COM_IRQbits, [10](#)

IDLE
 ReaderMFRC522, [32](#)
 Tag, [67](#)
 INCREMENT
 Tag, [66](#)
 INV_MOD
 ReaderMFRC522::TX_MODEbits, [81](#)
 INV_TX1_RF_OFF
 ReaderMFRC522::TX_CONTROLbits, [80](#)
 INV_TX1_RF_ON
 ReaderMFRC522::TX_CONTROLbits, [80](#)
 INV_TX2_RF_OFF
 ReaderMFRC522::TX_CONTROLbits, [80](#)
 INV_TX2_RF_ON
 ReaderMFRC522::TX_CONTROLbits, [80](#)
 INV_TX_RF_OFF
 ReaderMFRC522::TX_CONTROLbits, [80](#)
 INV_TX_RF_ON
 ReaderMFRC522::TX_CONTROLbits, [80](#)
 IRQ_PUSH_PULL
 ReaderMFRC522::DIV_I_ENbits, [16](#)
 IRQ
 ReaderMFRC522::STATUS1bits, [61](#)
 increment
 Tag, [69](#)
 initialize
 ReaderMFRC522, [39](#)
 ReaderMFRC530, [48](#)
 ReaderMFRC531, [53](#)
 Interrupt
 ReaderMFRC522, [32](#)
 ReaderMFRC530, [46](#)
 ReaderMFRC531, [52](#)
 isAddressSectorTrailer
 Tag, [69](#)
 TagMF1ICS70, [75](#)
 TagMF1S503x, [78](#)

 KEY_A
 Tag, [66](#)
 KEY_B
 Tag, [66](#)
 key
 Tag, [71](#)
 KeyType
 Tag, [66](#)
 keyType
 Tag, [71](#)

 LEVEL_0
 TagMF0ICU1, [73](#)
 TagMF1ICS70, [75](#)
 TagMF1S503x, [77](#)
 LEVEL_1
 TagMF0ICU1, [73](#)
 TagMF1ICS70, [75](#)
 TagMF1S503x, [77](#)
 LEVEL_2
 TagMF0ICU1, [73](#)
 TagMF1ICS70, [75](#)
 TagMF1S503x, [77](#)
 LEVEL_3
 TagMF0ICU1, [73](#)
 TagMF1ICS70, [75](#)
 TagMF1S503x, [77](#)
 LEVEL_4
 TagMF0ICU1, [73](#)
 TagMF1ICS70, [75](#)
 TagMF1S503x, [77](#)
 LEVEL_5
 TagMF0ICU1, [73](#)
 TagMF1ICS70, [75](#)
 TagMF1S503x, [77](#)
 LEVEL_6
 TagMF0ICU1, [73](#)
 TagMF1ICS70, [75](#)
 TagMF1S503x, [77](#)
 LEVEL_7
 TagMF0ICU1, [73](#)
 TagMF1ICS70, [75](#)
 TagMF1S503x, [77](#)
 LO_ALERT_I_EN
 ReaderMFRC522::COM_I_ENbits, [9](#)
 LO_ALERT_IRQ
 ReaderMFRC522::COM_IRQbits, [10](#)
 LO_ALERT
 ReaderMFRC522::STATUS1bits, [61](#)
 lastError
 Reader, [29](#)

 MEM
 ReaderMFRC522, [32](#)
 MF_AUTHENT
 ReaderMFRC522, [32](#)
 MF_CRYPT01_ON
 ReaderMFRC522::STATUS2bits, [62](#)
 MF_OUT_SEL
 ReaderMFRC522::TX_SELbits, [82](#)
 MF_RX
 ReaderMFRC522, [34](#)
 MF_TX
 ReaderMFRC522, [34](#)
 MFIN_ACT_I_EN
 ReaderMFRC522::DIV_I_ENbits, [16](#)
 MFIN_ACT_IRQ
 ReaderMFRC522::DIV_IRQbits, [17](#)
 MFR522_INT_TO_EN_MASK
 ReaderMFRC522.h, [97](#)
 MFR522_INT_TO_EN_REG
 ReaderMFRC522.h, [97](#)
 MFR522_INT_TO_IRQ_MASK
 ReaderMFRC522.h, [98](#)
 MFR522_INT_TO_IRQ_REG
 ReaderMFRC522.h, [98](#)
 MFR522_DEFAULT_TIMEOUT
 ReaderMFRC522.h, [98](#)
 MIFARE_1K
 Tag, [67](#)

- MIFARE_4K
 - Tag, [67](#)
- MIFARE_MINI
 - Tag, [67](#)
- MIFARE_NOT_COMPLETE
 - Tag, [67](#)
- MIFARE_PLUS
 - Tag, [67](#)
- MIFARE_UNKNOWN
 - Tag, [67](#)
- MIFARE_UL
 - Tag, [67](#)
- MIN_LEVEL
 - ReaderMFRC522::RX_THRESHOLDbits, [58](#)
- MOD_GS_N
 - ReaderMFRC522::GS_Nbits, [20](#)
- MOD_GS_P
 - ReaderMFRC522, [34](#)
 - ReaderMFRC522::MOD_GS_Pbits, [22](#)
- MOD_WIDTH
 - ReaderMFRC522, [34](#)
- MODEM_STATE
 - ReaderMFRC522::STATUS2bits, [62](#)
- MODE
 - ReaderMFRC522, [34](#)
- MSB_FIRST
 - ReaderMFRC522::MODEbits, [23](#)
- main
 - main.cpp, [86](#)
- main.cpp, [86](#), [87](#)
 - main, [86](#)
- Mask
 - ReaderMFRC522, [33](#)
 - ReaderMFRC530, [47](#)
 - ReaderMFRC531, [52](#)
- NACK
 - Reader, [26](#)
- NO_CMD_CHANGE
 - ReaderMFRC522, [32](#)
- NO_ERROR
 - Reader, [26](#)
- NONE_IRQ
 - ReaderMFRC522, [33](#)
- PARITY_DISABLE
 - ReaderMFRC522::MF_RXbits, [21](#)
- PARITY_ERR
 - ReaderMFRC522::ERRORbits, [18](#)
- POL_M_FIN
 - ReaderMFRC522::MODEbits, [23](#)
- POWER_DOWN
 - ReaderMFRC522::COMMANDbits, [11](#)
- POWER_OFF
 - Tag, [67](#)
- PROGMEM
 - ReaderMFRC522.h, [98](#)
- PROTOCOL_ERR
 - ReaderMFRC522::ERRORbits, [18](#)
- performSelfTest
 - Reader, [27](#)
 - ReaderMFRC522, [39](#)
 - ReaderMFRC530, [48](#)
 - ReaderMFRC531, [53](#)
- Permission
 - TagMF0ICU1, [73](#)
 - TagMF1ICS70, [75](#)
 - TagMF1S503x, [77](#)
- RCV_OFF
 - ReaderMFRC522::COMMANDbits, [12](#)
- READER_DEFAULT_TIMEOUT
 - Reader.h, [89](#)
- READY
 - Tag, [67](#)
- READ
 - Tag, [66](#)
- RECEIVE
 - ReaderMFRC522, [32](#)
- REQUEST
 - Tag, [66](#)
- RESTORE
 - Tag, [66](#)
- RFC_FG
 - ReaderMFRC522, [34](#)
- RX_ALIGN
 - ReaderMFRC522::BIT_FRAMINGbits, [6](#)
- RX_CRC_EN
 - ReaderMFRC522::RX_MODEbits, [56](#)
- RX_GAIN
 - ReaderMFRC522::RF_CFGbits, [55](#)
- RX_I_EN
 - ReaderMFRC522::COM_I_ENbits, [9](#)
- RX_IRQ
 - ReaderMFRC522::COM_IRQbits, [10](#)
- RX_LAST_BITS
 - ReaderMFRC522::CONTROLbits, [12](#)
- RX_MODE
 - ReaderMFRC522, [34](#)
- RX_MULTIPLE
 - ReaderMFRC522::RX_MODEbits, [56](#)
- RX_NO_ERR
 - ReaderMFRC522::RX_MODEbits, [57](#)
- RX_SEL
 - ReaderMFRC522, [34](#)
- RX_SPEED
 - ReaderMFRC522::RX_MODEbits, [57](#)
- RX_THRESHOLD
 - ReaderMFRC522, [34](#)
- RX_WAIT
 - ReaderMFRC522::RX_SELbits, [57](#)
- readAccessBits
 - Tag, [69](#)
- readBlock
 - Tag, [69](#)
- readBlockSlice
 - Tag, [69](#)
- readByte

- Tag, [69](#)
- readKey
 - Tag, [69](#)
- readRegisterBlock
 - ReaderMFRC522, [39](#), [41](#)
 - ReaderMFRC530, [48](#)
 - ReaderMFRC531, [53](#), [54](#)
- Reader, [24](#)
 - ~Reader, [26](#)
 - authenticate, [26](#)
 - COLLISION_ERROR, [26](#)
 - COMMUNICATION_ERROR, [26](#)
 - CRC_ERROR, [26](#)
 - calculateCrc, [26](#)
 - clearInterrupt, [26](#)
 - clearLastError, [26](#)
 - communicate, [26](#), [27](#)
 - configureTimer, [27](#)
 - disableInterrupt, [27](#)
 - enableInterrupt, [27](#)
 - Error, [26](#)
 - flushQueue, [27](#)
 - GENERAL_ERROR, [26](#)
 - generateRandomId, [27](#)
 - getCollisionPosition, [27](#)
 - getLastError, [27](#)
 - hasValidCrc, [27](#)
 - lastError, [29](#)
 - NACK, [26](#)
 - NO_ERROR, [26](#)
 - performSelfTest, [27](#)
 - Reader, [26](#)
 - sendCommand, [27](#)
 - setAntennaOff, [27](#)
 - setAntennaOn, [28](#)
 - setBitFraming, [28](#)
 - setWaterLevel, [28](#)
 - setUpForAnticollision, [28](#)
 - softReset, [28](#)
 - startTimer, [28](#)
 - stopTimer, [28](#)
 - TIMEOUT_ERROR, [26](#)
 - tranceive, [28](#)
 - turnOffEncryption, [28](#)
 - waitForRegisterBits, [28](#)
- reader
 - Tag, [71](#)
- Reader.cpp, [88](#), [89](#)
- Reader.h, [89](#), [90](#)
 - READER_DEFAULT_TIMEOUT, [89](#)
- ReaderMFRC522, [29](#)
 - ~ReaderMFRC522, [35](#)
 - ACK, [44](#)
 - ANALOG_TEST, [35](#)
 - AUTO_TEST_ENABLE, [33](#)
 - AUTO_TEST, [35](#)
 - authenticate, [35](#)
 - BIT_FRAMING_START_SEND, [33](#)
 - BIT_FRAMING, [34](#)
 - CALC_CRC, [32](#)
 - CLONE, [35](#)
 - COLL_VALUES_AFTER_COLL, [33](#)
 - COLL, [34](#)
 - COM_ALL_IRQ, [33](#)
 - COM_ERR_IRQ, [33](#)
 - COM_HI_ALERT_IRQ, [33](#)
 - COM_I_EN_INTERRUPT_EN, [33](#)
 - COM_I_EN, [34](#)
 - COM_IDLE_IRQ, [33](#)
 - COM_IRQ_ALL_IRQ, [33](#)
 - COM_IRQ_ERR_IRQ, [33](#)
 - COM_IRQ_HI_ALERT_IRQ, [33](#)
 - COM_IRQ_IDLE_IRQ, [33](#)
 - COM_IRQ_LO_ALERT_IRQ, [33](#)
 - COM_IRQ_RX_IRQ, [33](#)
 - COM_IRQ_SET1, [33](#)
 - COM_IRQ_TIMER_IRQ, [33](#)
 - COM_IRQ_TX_IRQ, [33](#)
 - COM_IRQ, [34](#)
 - COM_LO_ALERT_IRQ, [33](#)
 - COM_RX_IRQ, [33](#)
 - COM_TIMER_IRQ, [33](#)
 - COM_TX_IRQ, [33](#)
 - COMMAND, [34](#)
 - CONTROL_T_START_NOW, [33](#)
 - CONTROL_T_STOP_NOW, [33](#)
 - CONTROL, [34](#)
 - CRC_RESULT_HIGH, [34](#)
 - CRC_RESULT_LOW, [34](#)
 - CW_GS_P, [34](#)
 - calculateCrc, [36](#)
 - clearInterrupt, [36](#)
 - Command, [32](#)
 - communicate, [37](#)
 - configureTimer, [37](#)
 - DEMOD, [34](#)
 - DIV_ALL_IRQ, [33](#)
 - DIV_CRC_IRQ, [33](#)
 - DIV_I_EN_CRC_I_EN, [33](#)
 - DIV_I_EN_INTERRUPT_EN, [33](#)
 - DIV_I_EN_MFIN_ACT_I_EN, [33](#)
 - DIV_I_EN, [34](#)
 - DIV_IRQ_ALL_IRQ, [33](#)
 - DIV_IRQ_CRC_IRQ, [33](#)
 - DIV_IRQ_MFIN_ACT_IRQ, [33](#)
 - DIV_IRQ_SET2, [33](#)
 - DIV_IRQ, [34](#)
 - DIV_MFIN_ACT_IRQ, [33](#)
 - device, [44](#)
 - disableInterrupt, [38](#)
 - ERROR, [34](#)
 - enableInterrupt, [38](#)
 - FIFO_DATA, [34](#)
 - FIFO_LEVEL_FIFO_LEVEL, [33](#)
 - FIFO_LEVEL_FLUSH_BUFFER, [33](#)
 - FIFO_LEVEL, [34](#)

- flushQueue, 38
- GENERATE_RANDOM_ID, 32
- GS_N, 34
- generateRandomId, 38
- getCollisionPosition, 39
- getVersion, 39
- hasValidCrc, 39
- IDLE, 32
- initialize, 39
- Interrupt, 32
- MEM, 32
- MF_AUTHENT, 32
- MF_RX, 34
- MF_TX, 34
- MOD_GS_P, 34
- MOD_WIDTH, 34
- MODE, 34
- Mask, 33
- NO_CMD_CHANGE, 32
- NONE_IRQ, 33
- performSelfTest, 39
- RECEIVE, 32
- RFC_FG, 34
- RX_MODE, 34
- RX_SEL, 34
- RX_THRESHOLD, 34
- readRegisterBlock, 39, 41
- ReaderMFRC522, 35
- Register, 33
- resetPin, 44
- SAK, 44
- SERIAL_SPEED, 34
- SOFT_RESET, 32
- STATUS1, 34
- STATUS2, 34
- STATUS2_MF_CRYPT01_ON, 33
- sendCommand, 41
- setAntennaOff, 41
- setAntennaOn, 41
- setBitFraming, 42
- setWaterLevel, 42
- setUpForAnticollision, 42
- softReset, 42
- startTimer, 42
- stopTimer, 42
- T_COUNTER_VAL_HIGH, 34
- T_COUNTER_VAL_LOW, 34
- T_MODE, 34
- T_PRESCALER_LOW, 34
- T_RELOAD_HIGH, 34
- T_RELOAD_LOW, 34
- TEST_ADC, 35
- TEST_BUS, 35
- TEST_DAC1, 35
- TEST_DAC2, 35
- TEST_PIN_EN, 34
- TEST_PIN_VALUE, 34
- TEST_SEL1, 34
- TEST_SEL2, 34
- TRANSCIVE, 32
- TRANSMIT, 32
- TX_ASK, 34
- TX_CONTROL_TX1_RF_EN, 33
- TX_CONTROL_TX2_RF_EN, 33
- TX_CONTROL_TX_RF_EN, 33
- TX_CONTROL, 34
- TX_MODE, 34
- TX_SEL, 34
- tranceive, 43
- turnOffEncryption, 43
- V0_0, 35
- V1_0, 35
- V2_0, 35
- VERSION, 35
- Version, 35
- WATER_LEVEL_WATER_LEVEL, 33
- WATER_LEVEL, 34
- waitForRegisterBits, 43, 44
- writeRegisterBlock, 44
- ReaderMFRC522.cpp, 91
- ReaderMFRC522.h, 96, 99
 - MFR522_INT_TO_EN_MASK, 97
 - MFR522_INT_TO_EN_REG, 97
 - MFR522_INT_TO_IRQ_MASK, 98
 - MFR522_INT_TO_IRQ_REG, 98
 - MFRC522_DEFAULT_TIMEOUT, 98
 - PROGMEM, 98
- ReaderMFRC522::BIT_FRAMINGbits, 6
 - char, 6
 - RX_ALIGN, 6
 - START_SEND, 7
 - TX_LAST_BITS, 7
 - value, 7
- ReaderMFRC522::COLLbits, 7
 - COLL_POS_NOT_VALID, 8
 - COLL_POS, 7
 - char, 7
 - VALUES_AFTER_COLL, 8
 - value, 8
- ReaderMFRC522::COM_I_ENbits, 8
 - ERR_I_EN, 9
 - HI_ALERT_I_EN, 9
 - I_RQ_INV, 9
 - IDLE_I_EN, 9
 - LO_ALERT_I_EN, 9
 - RX_I_EN, 9
 - TIMER_I_EN, 9
 - TX_I_EN, 9
 - value, 9
- ReaderMFRC522::COM_IRQbits, 9
 - ERR_IRQ, 10
 - HI_ALERT_IRQ, 10
 - IDLE_IRQ, 10
 - LO_ALERT_IRQ, 10
 - RX_IRQ, 10
 - SET1, 10

- TIMER_IRQ, 10
- TX_IRQ, 11
- value, 11
- ReaderMFRC522::COMMANDbits, 11
 - COMMAND, 11
 - char, 11
 - POWER_DOWN, 11
 - RCV_OFF, 12
 - value, 12
- ReaderMFRC522::CONTROLbits, 12
 - char, 12
 - RX_LAST_BITS, 12
 - T_START_NOW, 13
 - T_STOP_NOW, 13
 - value, 13
- ReaderMFRC522::CW_GS_Pbits, 13
 - CW_GS_P, 13
 - char, 13
 - value, 14
- ReaderMFRC522::DEMObits, 14
 - ADD_IQ, 14
 - FIX_IQ, 14
 - T_PRESCAL_EVEN, 14
 - TAU_RCV, 15
 - TAU_SYNC, 15
 - value, 15
- ReaderMFRC522::DIV_I_ENbits, 15
 - CRC_I_EN, 15
 - char, 15
 - IRQ_PUSH_PULL, 16
 - MFIN_ACT_I_EN, 16
 - value, 16
- ReaderMFRC522::DIV_IRQbits, 16
 - CRC_IRQ, 16
 - char, 16
 - MFIN_ACT_IRQ, 17
 - SET2, 17
 - value, 17
- ReaderMFRC522::ERRORbits, 17
 - BUFFER_OVFL, 18
 - COLL_ERR, 18
 - CRC_ERR, 18
 - char, 18
 - PARITY_ERR, 18
 - PROTOCOL_ERR, 18
 - TEMP_ERR, 18
 - value, 18
 - WR_ERR, 18
- ReaderMFRC522::FIFO_LEVELbits, 18
 - FIFO_LEVEL, 19
 - FLUSH_BUFFER, 19
 - value, 19
- ReaderMFRC522::GS_Nbits, 19
 - CW_GS_N, 20
 - MOD_GS_N, 20
 - value, 20
- ReaderMFRC522::MF_RXbits, 20
 - char, 21
- PARITY_DISABLE, 21
 - value, 21
- ReaderMFRC522::MF_TXbits, 21
 - char, 21
 - TX_WAIT, 21
 - value, 22
- ReaderMFRC522::MOD_GS_Pbits, 22
 - char, 22
 - MOD_GS_P, 22
 - value, 22
- ReaderMFRC522::MODEbits, 23
 - CRC_PRESET, 23
 - char, 23
 - MSB_FIRST, 23
 - POL_M_FIN, 23
 - TX_WAIT_RF, 24
 - value, 24
- ReaderMFRC522::RF_CFGbits, 55
 - char, 55
 - RX_GAIN, 55
 - value, 56
- ReaderMFRC522::RX_MODEbits, 56
 - char, 56
 - RX_CRC_EN, 56
 - RX_MULTIPLE, 56
 - RX_NO_ERR, 57
 - RX_SPEED, 57
 - value, 57
- ReaderMFRC522::RX_SELbits, 57
 - RX_WAIT, 57
 - UART_SEL, 57
 - value, 58
- ReaderMFRC522::RX_THRESHOLDbits, 58
 - COLL_LEVEL, 58
 - char, 58
 - MIN_LEVEL, 58
 - value, 59
- ReaderMFRC522::SERIAL_SPEEDbits, 59
 - BR_T0, 59
 - BR_T1, 59
 - value, 59
- ReaderMFRC522::STATUS1bits, 60
 - CRC_OK, 60
 - CRC_READY, 60
 - char, 60
 - HI_ALERT, 60
 - IRQ, 61
 - LO_ALERT, 61
 - T_RUNNING, 61
 - value, 61
- ReaderMFRC522::STATUS2bits, 61
 - char, 62
 - I2C_FORCE_HS, 62
 - MF_CRYPT01_ON, 62
 - MODEM_STATE, 62
 - TEMP_SENS_CLEAR, 62
 - value, 62
- ReaderMFRC522::T_MODEbits, 62

- T_AUTO_RESTART, 63
- T_AUTO, 63
- T_GATED, 63
- T_PRESCALER_HI, 63
- value, 63
- ReaderMFRC522::TX_ASKbits, 78
 - char, 79
 - FORCE_100_ASK, 79
 - value, 79
- ReaderMFRC522::TX_CONTROLbits, 79
 - char, 80
 - INV_TX1_RF_OFF, 80
 - INV_TX1_RF_ON, 80
 - INV_TX2_RF_OFF, 80
 - INV_TX2_RF_ON, 80
 - INV_TX_RF_OFF, 80
 - INV_TX_RF_ON, 80
 - TX1_RF_EN, 80
 - TX2_CW, 80
 - TX2_RF_EN, 80
 - TX_RF_EN, 81
 - value, 81
- ReaderMFRC522::TX_MODEbits, 81
 - char, 81
 - INV_MOD, 81
 - TX_CRC_EN, 82
 - TX_SPEED, 82
 - value, 82
- ReaderMFRC522::TX_SELbits, 82
 - char, 82
 - MF_OUT_SEL, 82
 - value, 83
- ReaderMFRC522::VERSIONbits, 84
 - CHIPTYPE, 84
 - VERSION, 84
 - value, 84
- ReaderMFRC522::WATER_LEVELbits, 84
 - char, 85
 - value, 85
 - WATER_LEVEL, 85
- ReaderMFRC530, 45
 - ~ReaderMFRC530, 47
 - authenticate, 47
 - calculateCrc, 47
 - clearInterrupt, 47
 - Command, 46
 - communicate, 47
 - configureTimer, 47
 - disableInterrupt, 48
 - enableInterrupt, 48
 - flushQueue, 48
 - generateRandomId, 48
 - getCollisionPosition, 48
 - getVersion, 48
 - hasValidCrc, 48
 - initialize, 48
 - Interrupt, 46
 - Mask, 47
 - performSelfTest, 48
 - readRegisterBlock, 48
 - ReaderMFRC530, 47
 - Register, 47
 - sendCommand, 48
 - setAntennaOff, 48
 - setAntennaOn, 48
 - setBitFraming, 49
 - setWaterLevel, 49
 - setuptForAnticollision, 49
 - softReset, 49
 - startTimer, 49
 - stopTimer, 49
 - tranceive, 49
 - turnOffEncryption, 49
 - Version, 47
 - waitForRegisterBits, 49
 - writeRegisterBlock, 49
- ReaderMFRC530.cpp, 113
- ReaderMFRC530.h, 114
- ReaderMFRC531, 50
 - ~ReaderMFRC531, 52
 - authenticate, 52
 - calculateCrc, 52
 - clearInterrupt, 52
 - Command, 52
 - communicate, 52, 53
 - configureTimer, 53
 - disableInterrupt, 53
 - enableInterrupt, 53
 - flushQueue, 53
 - generateRandomId, 53
 - getCollisionPosition, 53
 - getVersion, 53
 - hasValidCrc, 53
 - initialize, 53
 - Interrupt, 52
 - Mask, 52
 - performSelfTest, 53
 - readRegisterBlock, 53, 54
 - ReaderMFRC531, 52
 - Register, 52
 - sendCommand, 54
 - setAntennaOff, 54
 - setAntennaOn, 54
 - setBitFraming, 54
 - setWaterLevel, 54
 - setuptForAnticollision, 54
 - softReset, 54
 - startTimer, 54
 - stopTimer, 54
 - tranceive, 54
 - turnOffEncryption, 54
 - Version, 52
 - waitForRegisterBits, 55
 - writeRegisterBlock, 55
- ReaderMFRC531.cpp, 116
- ReaderMFRC531.h, 116, 117

- Register
 - ReaderMFRC522, [33](#)
 - ReaderMFRC530, [47](#)
 - ReaderMFRC531, [52](#)
- request
 - Tag, [69](#)
- resetPin
 - ReaderMFRC522, [44](#)
- restore
 - Tag, [69](#)
- SAK
 - ReaderMFRC522, [44](#)
- SEL_CL1
 - Tag, [66](#)
- SEL_CL2
 - Tag, [66](#)
- SEL_CL3
 - Tag, [66](#)
- SERIAL_SPEED
 - ReaderMFRC522, [34](#)
- SET1
 - ReaderMFRC522::COM_IRQbits, [10](#)
- SET2
 - ReaderMFRC522::DIV_IRQbits, [17](#)
- SOFT_RESET
 - ReaderMFRC522, [32](#)
- START_SEND
 - ReaderMFRC522::BIT_FRAMINGbits, [7](#)
- STATUS1
 - ReaderMFRC522, [34](#)
- STATUS2
 - ReaderMFRC522, [34](#)
- STATUS2_MF_CRYPT01_ON
 - ReaderMFRC522, [33](#)
- sak
 - Tag::Uid, [83](#)
- sectorTrailerProtected
 - Tag, [71](#)
- select
 - Tag, [70](#)
- sendCommand
 - Reader, [27](#)
 - ReaderMFRC522, [41](#)
 - ReaderMFRC530, [48](#)
 - ReaderMFRC531, [54](#)
- setAntennaOff
 - Reader, [27](#)
 - ReaderMFRC522, [41](#)
 - ReaderMFRC530, [48](#)
 - ReaderMFRC531, [54](#)
- setAntennaOn
 - Reader, [28](#)
 - ReaderMFRC522, [41](#)
 - ReaderMFRC530, [48](#)
 - ReaderMFRC531, [54](#)
- setBitFraming
 - Reader, [28](#)
 - ReaderMFRC522, [42](#)
 - ReaderMFRC530, [49](#)
 - ReaderMFRC531, [54](#)
- setBlockPermission
 - Tag, [70](#)
- setBlockType
 - Tag, [70](#)
- setSectorTrailerProtected
 - Tag, [70](#)
- setState
 - Tag, [70](#)
- setWaterLevel
 - Reader, [28](#)
 - ReaderMFRC522, [42](#)
 - ReaderMFRC530, [49](#)
 - ReaderMFRC531, [54](#)
- setupAuthenticationKey
 - Tag, [70](#)
- setuptForAnticollision
 - Reader, [28](#)
 - ReaderMFRC522, [42](#)
 - ReaderMFRC530, [49](#)
 - ReaderMFRC531, [54](#)
- size
 - Tag::Uid, [83](#)
- softReset
 - Reader, [28](#)
 - ReaderMFRC522, [42](#)
 - ReaderMFRC530, [49](#)
 - ReaderMFRC531, [54](#)
- startTimer
 - Reader, [28](#)
 - ReaderMFRC522, [42](#)
 - ReaderMFRC530, [49](#)
 - ReaderMFRC531, [54](#)
- State
 - Tag, [66](#)
- state
 - Tag, [71](#)
- stopTimer
 - Reader, [28](#)
 - ReaderMFRC522, [42](#)
 - ReaderMFRC530, [49](#)
 - ReaderMFRC531, [54](#)
- supportsAnticollision
 - Tag, [71](#)
- T_AUTO_RESTART
 - ReaderMFRC522::T_MODEbits, [63](#)
- T_AUTO
 - ReaderMFRC522::T_MODEbits, [63](#)
- T_COUNTER_VAL_HIGH
 - ReaderMFRC522, [34](#)
- T_COUNTER_VAL_LOW
 - ReaderMFRC522, [34](#)
- T_GATED
 - ReaderMFRC522::T_MODEbits, [63](#)
- T_MODE
 - ReaderMFRC522, [34](#)
- T_PRESCAL_EVEN

- ReaderMFRC522::DEMObits, [14](#)
- T_PRESCALER_HI
 - ReaderMFRC522::T_MODEbits, [63](#)
- T_PRESCALER_LOW
 - ReaderMFRC522, [34](#)
- T_RELOAD_HIGH
 - ReaderMFRC522, [34](#)
- T_RELOAD_LOW
 - ReaderMFRC522, [34](#)
- T_RUNNING
 - ReaderMFRC522::STATUS1bits, [61](#)
- T_START_NOW
 - ReaderMFRC522::CONTROLbits, [13](#)
- T_STOP_NOW
 - ReaderMFRC522::CONTROLbits, [13](#)
- TAG_ATQA_ANTICOLLISION_BIT
 - Tag.h, [124](#)
- TAG_DEFAULT_SECTOR_SIZE
 - Tag.h, [124](#)
- TAG_KEY_SIZE
 - Tag.h, [124](#)
- TAG_KEY_TO_POS
 - Tag.h, [124](#)
- TAG_MF1ICS70_HIGH_SECTOR_COUNT
 - TagMF1ICS70.h, [132](#)
- TAG_MF1ICS70_HIGH_SECTOR_SIZE
 - TagMF1ICS70.h, [132](#)
- TAG_MF1ICS70_LOW_MEMORY_SIZE
 - TagMF1ICS70.h, [132](#)
- TAG_MF1ICS70_LOW_SECTOR_COUNT
 - TagMF1ICS70.h, [132](#)
- TAG_MF1ICS70_LOW_SECTOR_SIZE
 - TagMF1ICS70.h, [132](#)
- TAG_MF1S503X_SECTOR_SIZE
 - TagMF1S503x.h, [135](#)
- TAG_SAK_BIT
 - Tag.h, [124](#)
- TAU_RCV
 - ReaderMFRC522::DEMObits, [15](#)
- TAU_SYNC
 - ReaderMFRC522::DEMObits, [15](#)
- TEMP_ERR
 - ReaderMFRC522::ERRORbits, [18](#)
- TEMP_SENS_CLEAR
 - ReaderMFRC522::STATUS2bits, [62](#)
- TEST_ADC
 - ReaderMFRC522, [35](#)
- TEST_BUS
 - ReaderMFRC522, [35](#)
- TEST_DAC1
 - ReaderMFRC522, [35](#)
- TEST_DAC2
 - ReaderMFRC522, [35](#)
- TEST_PIN_EN
 - ReaderMFRC522, [34](#)
- TEST_PIN_VALUE
 - ReaderMFRC522, [34](#)
- TEST_SEL1
 - ReaderMFRC522, [34](#)
- TEST_SEL2
 - ReaderMFRC522, [34](#)
- TIMEOUT_ERROR
 - Reader, [26](#)
- TIMER_I_EN
 - ReaderMFRC522::COM_I_ENbits, [9](#)
- TIMER_IRQ
 - ReaderMFRC522::COM_IRQbits, [10](#)
- TRANSCIVE
 - ReaderMFRC522, [32](#)
- TRANSFER
 - Tag, [66](#)
- TRANSMIT
 - ReaderMFRC522, [32](#)
- TX1_RF_EN
 - ReaderMFRC522::TX_CONTROLbits, [80](#)
- TX2_CW
 - ReaderMFRC522::TX_CONTROLbits, [80](#)
- TX2_RF_EN
 - ReaderMFRC522::TX_CONTROLbits, [80](#)
- TX_ASK
 - ReaderMFRC522, [34](#)
- TX_CONTROL_TX1_RF_EN
 - ReaderMFRC522, [33](#)
- TX_CONTROL_TX2_RF_EN
 - ReaderMFRC522, [33](#)
- TX_CONTROL_TX_RF_EN
 - ReaderMFRC522, [33](#)
- TX_CONTROL
 - ReaderMFRC522, [34](#)
- TX_CRC_EN
 - ReaderMFRC522::TX_MODEbits, [82](#)
- TX_I_EN
 - ReaderMFRC522::COM_I_ENbits, [9](#)
- TX_IRQ
 - ReaderMFRC522::COM_IRQbits, [11](#)
- TX_LAST_BITS
 - ReaderMFRC522::BIT_FRAMINGbits, [7](#)
- TX_MODE
 - ReaderMFRC522, [34](#)
- TX_RF_EN
 - ReaderMFRC522::TX_CONTROLbits, [81](#)
- TX_SEL
 - ReaderMFRC522, [34](#)
- TX_SPEED
 - ReaderMFRC522::TX_MODEbits, [82](#)
- TX_WAIT_RF
 - ReaderMFRC522::MODEbits, [24](#)
- TX_WAIT
 - ReaderMFRC522::MF_TXbits, [21](#)
- Tag, [64](#)
 - ~Tag, [67](#)
 - ACTIVE, [67](#)
 - AUTH_KEY_A, [66](#)
 - AUTH_KEY_B, [66](#)
 - activate, [67](#)
 - activateWakeUp, [67](#)

[addressToSector, 68](#)
[authenticate, 68](#)
[BlockType, 66](#)
[Command, 66](#)
[computeNvb, 68](#)
[computeTagType, 68](#)
[DATA_BLOCK, 66](#)
[DECREMENT, 66](#)
[decrement, 68](#)
[detect, 68](#)
[getSectorSize, 68](#)
[getSectorTrailerAddress, 68](#)
[getState, 68](#)
[getTagType, 68](#)
[getUid, 69](#)
[HALT, 67](#)
[HLT_A, 66](#)
[halt, 69](#)
[hasAnticollisionSupport, 69](#)
[IDLE, 67](#)
[INCREMENT, 66](#)
[increment, 69](#)
[isAddressSectorTrailer, 69](#)
[KEY_A, 66](#)
[KEY_B, 66](#)
[key, 71](#)
[KeyType, 66](#)
[keyType, 71](#)
[MIFARE_1K, 67](#)
[MIFARE_4K, 67](#)
[MIFARE_MINI, 67](#)
[MIFARE_NOT_COMPLETE, 67](#)
[MIFARE_PLUS, 67](#)
[MIFARE_UNKNOWN, 67](#)
[MIFARE_UL, 67](#)
[POWER_OFF, 67](#)
[READY, 67](#)
[READ, 66](#)
[REQUEST, 66](#)
[RESTORE, 66](#)
[readAccessBits, 69](#)
[readBlock, 69](#)
[readBlockSlice, 69](#)
[readByte, 69](#)
[readKey, 69](#)
[reader, 71](#)
[request, 69](#)
[restore, 69](#)
[SEL_CL1, 66](#)
[SEL_CL2, 66](#)
[SEL_CL3, 66](#)
[sectorTrailerProtected, 71](#)
[select, 70](#)
[setBlockPermission, 70](#)
[setBlockType, 70](#)
[setSectorTrailerProtected, 70](#)
[setState, 70](#)
[setupAuthenticationKey, 70](#)
[State, 66](#)
[state, 71](#)
[supportsAnticollision, 71](#)
[TRANSFER, 66](#)
[Tag, 67](#)
[TagType, 67](#)
[tagType, 71](#)
[transfer, 70](#)
[uid, 71](#)
[VALUE_BLOCK, 66](#)
[WAKE_UP, 66](#)
[WRITE, 66](#)
[wakeUp, 70](#)
[writeAccessBits, 70](#)
[writeBlock, 70](#)
[writeBlockSlice, 70](#)
[writeByte, 70](#)
[writeKey, 71](#)
[Tag.cpp, 118, 119](#)
[Tag.h, 123, 125](#)
[TAG_ATQA_ANTICOLLISION_BIT, 124](#)
[TAG_DEFAULT_SECTOR_SIZE, 124](#)
[TAG_KEY_SIZE, 124](#)
[TAG_KEY_TO_POS, 124](#)
[TAG_SAK_BIT, 124](#)
[Tag::Uid, 83](#)
[sak, 83](#)
[size, 83](#)
[uid, 83](#)
[TagMF0ICU1, 72](#)
[LEVEL_0, 73](#)
[LEVEL_1, 73](#)
[LEVEL_2, 73](#)
[LEVEL_3, 73](#)
[LEVEL_4, 73](#)
[LEVEL_5, 73](#)
[LEVEL_6, 73](#)
[LEVEL_7, 73](#)
[Permission, 73](#)
[TagMF0ICU1, 73](#)
[TagMF0ICU1.cpp, 127, 128](#)
[TagMF0ICU1.h, 128, 129](#)
[TagMF1ICS70, 74](#)
[addressToSector, 75](#)
[getSectorSize, 75](#)
[getSectorTrailerAddress, 75](#)
[isAddressSectorTrailer, 75](#)
[LEVEL_0, 75](#)
[LEVEL_1, 75](#)
[LEVEL_2, 75](#)
[LEVEL_3, 75](#)
[LEVEL_4, 75](#)
[LEVEL_5, 75](#)
[LEVEL_6, 75](#)
[LEVEL_7, 75](#)
[Permission, 75](#)
[TagMF1ICS70, 75](#)
[writeBlock, 76](#)

- TagMF1ICS70.cpp, [129](#), [130](#)
- TagMF1ICS70.h, [131](#), [133](#)
 - TAG_MF1ICS70_HIGH_SECTOR_COUNT, [132](#)
 - TAG_MF1ICS70_HIGH_SECTOR_SIZE, [132](#)
 - TAG_MF1ICS70_LOW_MEMORY_SIZE, [132](#)
 - TAG_MF1ICS70_LOW_SECTOR_COUNT, [132](#)
 - TAG_MF1ICS70_LOW_SECTOR_SIZE, [132](#)
- TagMF1S503x, [76](#)
 - addressToSector, [78](#)
 - getSectorSize, [78](#)
 - getSectorTrailerAddress, [78](#)
 - isAddressSectorTrailer, [78](#)
 - LEVEL_0, [77](#)
 - LEVEL_1, [77](#)
 - LEVEL_2, [77](#)
 - LEVEL_3, [77](#)
 - LEVEL_4, [77](#)
 - LEVEL_5, [77](#)
 - LEVEL_6, [77](#)
 - LEVEL_7, [77](#)
 - Permission, [77](#)
 - TagMF1S503x, [77](#)
 - writeBlock, [78](#)
- TagMF1S503x.cpp, [133](#), [134](#)
- TagMF1S503x.h, [134](#), [136](#)
 - TAG_MF1S503X_SECTOR_SIZE, [135](#)
- TagType
 - Tag, [67](#)
- tagType
 - Tag, [71](#)
- tranceive
 - Reader, [28](#)
 - ReaderMFRC522, [43](#)
 - ReaderMFRC530, [49](#)
 - ReaderMFRC531, [54](#)
- transfer
 - Tag, [70](#)
- turnOffEncryption
 - Reader, [28](#)
 - ReaderMFRC522, [43](#)
 - ReaderMFRC530, [49](#)
 - ReaderMFRC531, [54](#)
- UART_SEL
 - ReaderMFRC522::RX_SELbits, [57](#)
- uid
 - Tag, [71](#)
 - Tag::Uid, [83](#)
- V0_0
 - ReaderMFRC522, [35](#)
- V1_0
 - ReaderMFRC522, [35](#)
- V2_0
 - ReaderMFRC522, [35](#)
- VALUE_BLOCK
 - Tag, [66](#)
- VALUES_AFTER_COLL
 - ReaderMFRC522::COLLbits, [8](#)
- VERSION
 - ReaderMFRC522, [35](#)
 - ReaderMFRC522::VERSIONbits, [84](#)
- value
 - ReaderMFRC522::BIT_FRAMINGbits, [7](#)
 - ReaderMFRC522::COLLbits, [8](#)
 - ReaderMFRC522::COM_I_ENbits, [9](#)
 - ReaderMFRC522::COM_IRQbits, [11](#)
 - ReaderMFRC522::COMMANDbits, [12](#)
 - ReaderMFRC522::CONTROLbits, [13](#)
 - ReaderMFRC522::CW_GS_Pbits, [14](#)
 - ReaderMFRC522::DEMODbits, [15](#)
 - ReaderMFRC522::DIV_I_ENbits, [16](#)
 - ReaderMFRC522::DIV_IRQbits, [17](#)
 - ReaderMFRC522::ERRORbits, [18](#)
 - ReaderMFRC522::FIFO_LEVELbits, [19](#)
 - ReaderMFRC522::GS_Nbits, [20](#)
 - ReaderMFRC522::MF_RXbits, [21](#)
 - ReaderMFRC522::MF_TXbits, [22](#)
 - ReaderMFRC522::MOD_GS_Pbits, [22](#)
 - ReaderMFRC522::MODEbits, [24](#)
 - ReaderMFRC522::RF_CFGbits, [56](#)
 - ReaderMFRC522::RX_MODEbits, [57](#)
 - ReaderMFRC522::RX_SELbits, [58](#)
 - ReaderMFRC522::RX_THRESHOLDbits, [59](#)
 - ReaderMFRC522::SERIAL_SPEEDbits, [59](#)
 - ReaderMFRC522::STATUS1bits, [61](#)
 - ReaderMFRC522::STATUS2bits, [62](#)
 - ReaderMFRC522::T_MODEbits, [63](#)
 - ReaderMFRC522::TX_ASKbits, [79](#)
 - ReaderMFRC522::TX_CONTROLbits, [81](#)
 - ReaderMFRC522::TX_MODEbits, [82](#)
 - ReaderMFRC522::TX_SELbits, [83](#)
 - ReaderMFRC522::VERSIONbits, [84](#)
 - ReaderMFRC522::WATER_LEVELbits, [85](#)
- Version
 - ReaderMFRC522, [35](#)
 - ReaderMFRC530, [47](#)
 - ReaderMFRC531, [52](#)
- WAKE_UP
 - Tag, [66](#)
- WATER_LEVEL_WATER_LEVEL
 - ReaderMFRC522, [33](#)
- WATER_LEVEL
 - ReaderMFRC522, [34](#)
 - ReaderMFRC522::WATER_LEVELbits, [85](#)
- WR_ERR
 - ReaderMFRC522::ERRORbits, [18](#)
- WRITE
 - Tag, [66](#)
- waitForRegisterBits
 - Reader, [28](#)
 - ReaderMFRC522, [43](#), [44](#)
 - ReaderMFRC530, [49](#)
 - ReaderMFRC531, [55](#)
- wakeUp
 - Tag, [70](#)
- writeAccessBits

- Tag, [70](#)
- writeBlock
 - Tag, [70](#)
 - TagMF1ICS70, [76](#)
 - TagMF1S503x, [78](#)
- writeBlockSlice
 - Tag, [70](#)
- writeByte
 - Tag, [70](#)
- writeKey
 - Tag, [71](#)
- writeRegisterBlock
 - ReaderMFRC522, [44](#)
 - ReaderMFRC530, [49](#)
 - ReaderMFRC531, [55](#)