Arduino RFID Driver

Generated by Doxygen 1.8.11

Contents

1 Hierarchical Index

1.1 Class Hierarchy

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

2 Class Index

2.1 Class List

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

3 File Index

3.1 File List

Here is a list of all files with brief descriptions:

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::DEMODbits 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

DEMOD 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::DEMODbits::ADD_IQ

Definition at line 1023 of file ReaderMFRC522.h.

4.8.2.3 unsigned char ReaderMFRC522::DEMODbits::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.

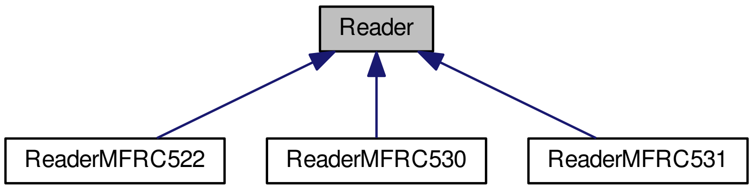
The documentation for this union was generated from the following file:

- 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.

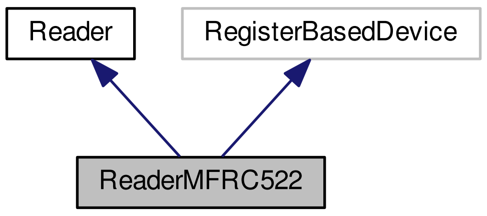
The documentation for this class was generated from the following files:

- Reader.h
- Reader.cpp

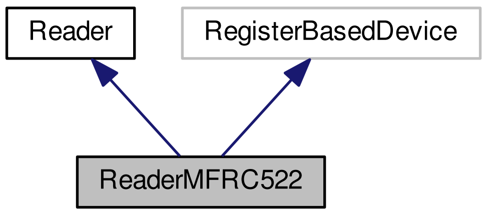
4.19 ReaderMFRC522 Class Reference

`#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 DEMODbits
- union DIV_I_ENbits
- union DIV_IRQbits
- union ERRORbits
- union FIFO_LEVELbits
- union GS_Nbits
- union MF_RXbits
- union MF_TXbits
- union MOD_GS_Pbits
- union MODEbits
- union RF_CFGbits
- union RX_MODEbits
- union RX_SELbits
- union RX_THRESHOLDbits
- union SERIAL_SPEEDbits
- union STATUS1bits
- union STATUS2bits
- union T_MODEbits
- union TX_ASKbits
- union TX_CONTROLbits
- union TX_MODEbits
- union TX_SELbits
- union VERSIONbits
- union WATER_LEVELbits

Public Types

- enum Register {
  COMMAND = 0x01, COM_I_EN = 0x02, DIV_I_EN = 0x03, COM_IRQ = 0x04,
  DIV_IRQ = 0X05, ERROR = 0X06, STATUS1 = 0x07, STATUS2 = 0x08,
  FIFO_DATA = 0X09, FIFO_LEVEL = 0x0a, WATER_LEVEL = 0x0b, CONTROL = 0x0c,
  BIT_FRAMING = 0x0d, COLL = 0x0e, MODE = 0x11, TX_MODE = 0x12,
  RX_MODE = 0x13, TX_CONTROL = 0x14, TX_ASK = 0x15, TX_SEL = 0x16,
  RX_SEL = 0x17, RX_THRESHOLD = 0x18, DEMOD = 0x19, MF_TX = 0x1c,
  MF_RX = 0x1d, SERIAL_SPEED = 0x1f, CRC_RESULT_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, TRANSCEIVE = 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, C↩
  OM_IRQ_RX_IRQ = 0x20,
  COM_IRQ_TX_IRQ = 0x40, COM_IRQ_ALL_IRQ = 0x7f, COM_IRQ_SET1 = 0x80, DIV_I_EN_CRC_I_EN = 0x04,
  DIV_I_EN_MFIN_ACT_I_EN = 0x10, DIV_I_EN_INTERRUPT_EN = DIV_I_EN_CRC_I_EN | DIV_I_EN_↩
  MFIN_ACT_I_EN, DIV_IRQ_CRC_IRQ = 0x04, DIV_IRQ_MFIN_ACT_IRQ = 0x10,
  DIV_IRQ_ALL_IRQ = DIV_IRQ_CRC_IRQ | DIV_IRQ_MFIN_ACT_IRQ, DIV_IRQ_SET2 = 0x80, FIFO_L↩
  EVEL_FLUSH_BUFFER = 0x80, FIFO_LEVEL_FIFO_LEVEL = 0x7f,
  WATER_LEVEL_WATER_LEVEL = 0x3f, BIT_FRAMING_START_SEND = 0x80, AUTO_TEST_ENABLE = 0x09, COLL_VALUES_AFTER_COLL = 0x80,
  STATUS2_MF_CRYPTO1_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 = DI↩
  V_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

TRANSCEIVE

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_CRYPTO1_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

send	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

buf	The buffer to calculate the CRC.
len	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

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

interrupt	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

command	The command to be executed.
send	Buffer to place into the FIFO before executing the command.
receive	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.
sendLen	How many bytes the send buffer has.
checkCrc	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

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

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

interrupt	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

interrupt	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

buf	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

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

Returns

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

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

command	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

rxAlign	Defines the bit position for the first bit received to be stored in the FIFO buffer.
txLastBits	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

level	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

output	Pointer to the data to transfer to the FIFO.
input	NULL or pointer to buffer if data should be read back after executing the command. (max 64 bytes).
sendLen	Size of the data to transfer to the FIFO.
checkCrc	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

reg	The register to check.
mask	The bits we want to check if are one.
timeout	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

reg	The register number.
buf	The buffer.
len	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.

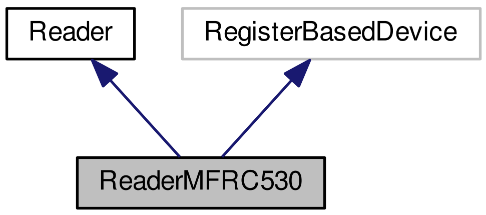
The documentation for this class was generated from the following files:

- ReaderMFRC522.h
- ReaderMFRC522.cpp

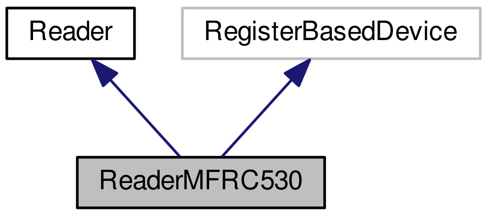
4.20 ReaderMFRC530 Class Reference

```
#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*)

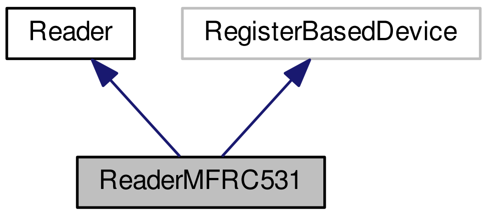
The documentation for this class was generated from the following file:

- ReaderMFRC530.h

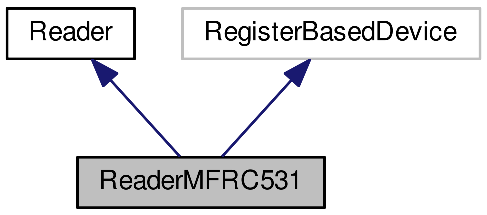
4.21 ReaderMFRC531 Class Reference

```
#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_CRYPTO1_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_CRYPTO1_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.

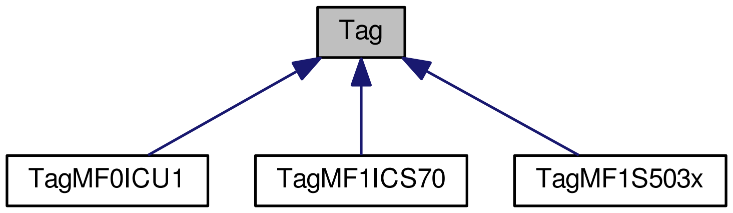
The documentation for this union was generated from the following file:

- ReaderMFRC522.h

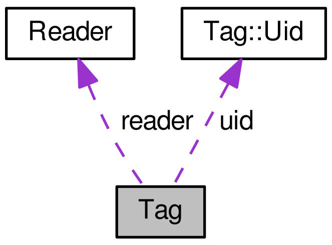
4.30 Tag Class Reference

```
#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.com`

Definition 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.

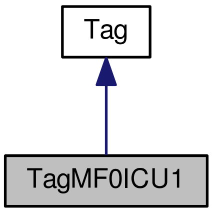
The documentation for this class was generated from the following files:

- Tag.h
- Tag.cpp

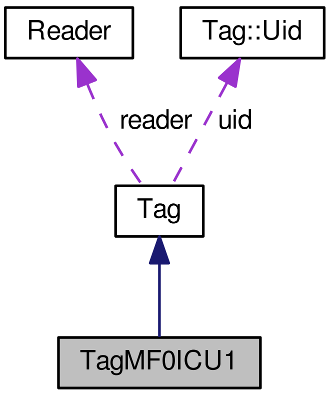
4.31 TagMF0ICU1 Class Reference

```
#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.

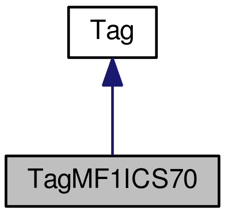
The documentation for this class was generated from the following files:

- TagMF0ICU1.h
- TagMF0ICU1.cpp

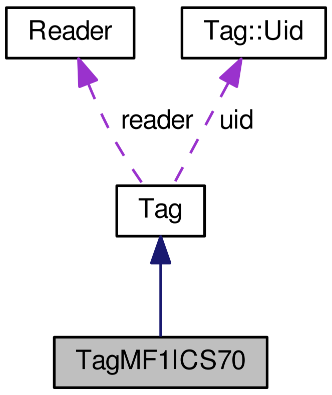
4.32 TagMF1ICS70 Class Reference

`#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.

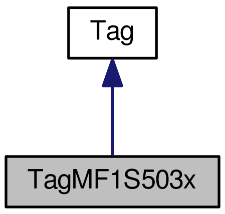
The documentation for this class was generated from the following files:

- TagMF1ICS70.h
- TagMF1ICS70.cpp

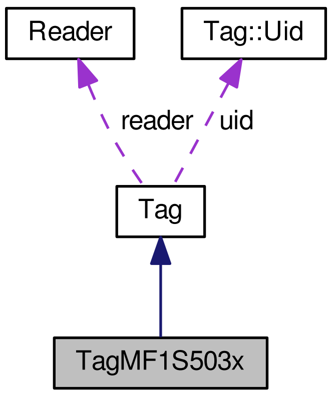
4.33 TagMF1S503x Class Reference

```
#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:

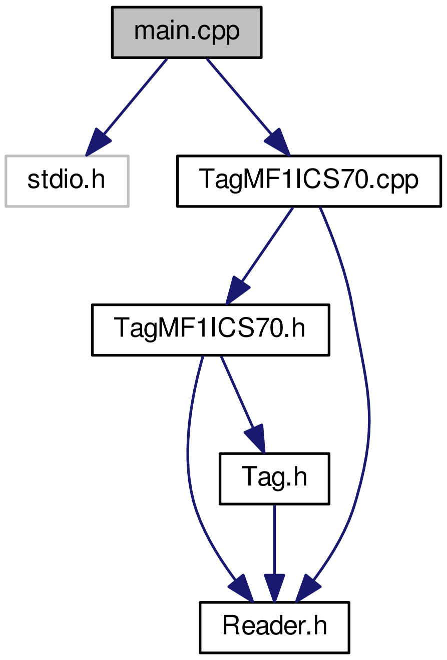
- ReaderMFRC522.h

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.

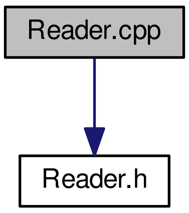
5.2 main.cpp

```cpp
#include <stdio.h>
#include "TagMF1ICS70.cpp"

int main() {

    TagMF1ICS70 tag;

    // getSectorSize

    if (tag.getSectorSize(0) != 4) {
        printf("error in getSectorSize#0\n");
    }

    if (tag.getSectorSize(31) != 4) {
        printf("error in getSectorSize#1\n");
    }

    if (tag.getSectorSize(32) != 16) {
        printf("error in getSectorSize#2\n");
    }

    if (tag.getSectorSize(40) != 16) {
        printf("error in getSectorSize#3\n");
    }

    // addressToSector

    if (tag.addressToSector(10) != 2) {
        printf("error in addressToSector#0\n");
    }

    if (tag.addressToSector(4) != 1) {
        printf("error in addressToSector#1\n");
    }

    if (tag.addressToSector(0) != 0) {
        printf("error in addressToSector#2\n");
    }

    if (tag.addressToSector(12) != 3) {
        printf("error in addressToSector#3\n");
    }

    if (tag.addressToSector(127) != 31) {
        printf("error in addressToSector#4\n");
    }

    if (tag.addressToSector(128) != 32) {
        printf("error in addressToSector#5\n");
    }

    if (tag.addressToSector(143) != 32) {
        printf("error in addressToSector#6\n");
    }

    if (tag.addressToSector(144) != 33) {
        printf("error in addressToSector#7\n");
    }

    if (tag.addressToSector(255) != 39) {
        printf("error in addressToSector#8\n");
    }

    // isAddressSectorTrailer

    if (tag.isAddressSectorTrailer(12)) {
        printf("error in isAddressSectorTrailer#0\n");
    }

    if (!tag.isAddressSectorTrailer(3)) {
        printf("error in isAddressSectorTrailer#1\n");
    }

    if (!tag.isAddressSectorTrailer(19)) {
        printf("error in isAddressSectorTrailer#2\n");
    }

    if (tag.isAddressSectorTrailer(22)) {
        printf("error in isAddressSectorTrailer#3\n");
    }

    if (!tag.isAddressSectorTrailer(127)) {
        printf("error in isAddressSectorTrailer#4\n");
    }

    if (!tag.isAddressSectorTrailer(255)) {
        printf("error in isAddressSectorTrailer#5\n");
    }

    if (tag.isAddressSectorTrailer(254-16)) {
        printf("error in isAddressSectorTrailer#6\n");
    }

    // getSectorTrailerAddress

    if (tag.getSectorTrailerAddress(31) != 127) {
        printf("error in getSectorTrailerAddress#4\n");
    }

    if (tag.getSectorTrailerAddress(32) != 143) {
        printf("error in getSectorTrailerAddress#5, got %d expect %d\n", tag.
getSectorTrailerAddress(32), 143);
    }

    if (tag.getSectorTrailerAddress(39) != 255) {
        printf("error in getSectorTrailerAddress#6, got %d expect %d\n", tag.
getSectorTrailerAddress(39), 255);
    }

    if (tag.getSectorTrailerAddress(1) != 7) {
        printf("error in getSectorTrailerAddress#1\n");
    }

    if (tag.getSectorTrailerAddress(2) != 11) {
        printf("error in getSectorTrailerAddress#2\n");
    }

    if (tag.getSectorTrailerAddress(3) != 15) {
        printf("error in getSectorTrailerAddress#3\n");
    }

    if (tag.getSectorTrailerAddress(31) != 127) {
        printf("error in getSectorTrailerAddress#5\n");
    }

    if (tag.getSectorTrailerAddress(32) != 143) {
        printf("error in getSectorTrailerAddress#6\n");
    }

    if (tag.getSectorTrailerAddress(39) != 255) {
        printf("error in getSectorTrailerAddress#7\n");
    }

    return 0;
}
```

5.3 Reader.cpp File Reference

`#include <Reader.h>`

Include dependency graph for Reader.cpp:

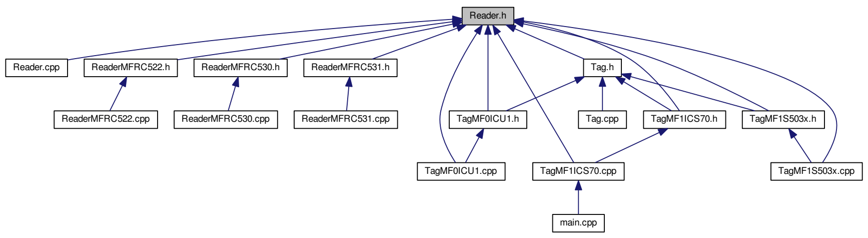


5.4 Reader.cpp

```
#include <Reader.h>

Reader::Reader() : lastError(NO_ERROR) {
}

Reader::~Reader() {
}

unsigned char Reader::getLastError() {
    return lastError;
}

void Reader::clearLastError() {
    lastError = NO_ERROR;
}
```

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.

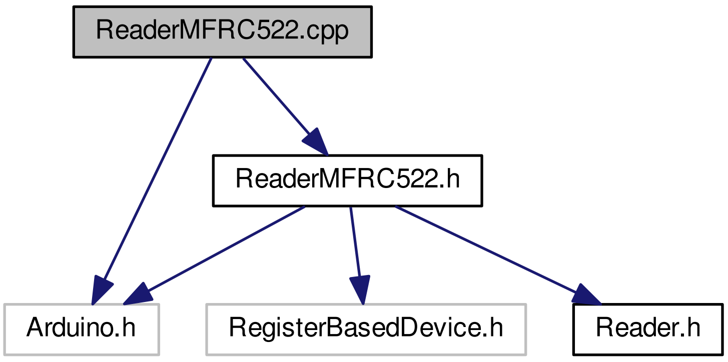
5.6 Reader.h

```

#ifndef __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_READER_H__
#define __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_READER_H__ 1

#define READER_DEFAULT_TIMEOUT          300

class Reader {

public:

    enum Error {
        NO_ERROR = 0x00,
        GENERAL_ERROR = 0x01,
        TIMEOUT_ERROR = 0x02,
        COMMUNICATION_ERROR = 0x03,
        CRC_ERROR = 0x04,
        NACK = 0x05,
        COLLISION_ERROR = 0x06
    };

    Reader();

    virtual ~Reader();

    virtual inline 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:

    Error lastError;
};

#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:



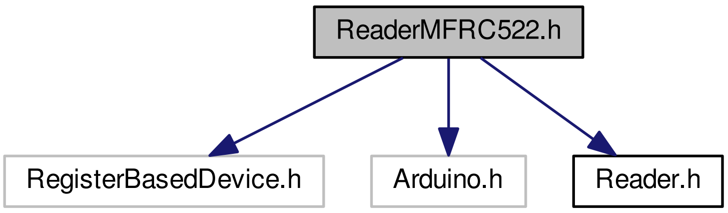
5.8 ReaderMFRC522.cpp

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

ReaderMFRC522::ReaderMFRC522(RegisterBasedDevice *device, unsigned char
resetPin)
        : Reader(), device(device), resetPin(resetPin) {
    pinMode(resetPin, OUTPUT);
    digitalWrite(resetPin, LOW);
}

ReaderMFRC522::~ReaderMFRC522() {
}

inline void ReaderMFRC522::sendCommand(unsigned char command) {
    writeRegister(COMMAND, command);
}

void ReaderMFRC522::initialize() {
    if (digitalRead(resetPin) == LOW) {
        digitalWrite(resetPin, HIGH);
        delay(50);
    } else {
        softReset();
    }
    clearRegisterBits(AUTO_TEST, AUTO_TEST_ENABLE);

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

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

    // Open the antenna
    setAntennaOn();
}

void ReaderMFRC522::softReset() {
    sendCommand(SOFT_RESET);
}

void ReaderMFRC522::setAntennaOn() {
    setRegisterBits(TX_CONTROL, TX_CONTROL_TX_RF_EN);
}

void ReaderMFRC522::setAntennaOff() {
    clearRegisterBits(TX_CONTROL, TX_CONTROL_TX_RF_EN);
}

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

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

int ReaderMFRC522::readRegisterBlock(unsigned char reg, unsigned char *buf,
 unsigned char len, unsigned char rxAlign) {
    int blockSize = readRegisterBlock(reg, buf, len);
    if (blockSize > 0 && rxAlign > 0) {
        rxAlign &= 0x07;
        unsigned char mask = 0;
        for (unsigned char i = rxAlign; i <= 7; i++) {
            mask |= (1 << i);
        }

        // Only bit positions rxAlign..7 in buf[0] are updated.
        buf[0] = (buf[0] & ~mask) | (buf[0] & mask);
    }
    return blockSize;
}

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

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

void ReaderMFRC522::configureTimer(unsigned int prescaler, unsigned int reload
, bool autoStart, bool autoRestart) {
    T_MODEbits timerMode;
    timerMode.value = readRegister(T_MODE);
    timerMode.T_PRESCALER_HI = (prescaler >> 8) & 0x0f;
    timerMode.T_AUTO = autoStart;
    timerMode.T_GATED = (unsigned char) 0;
    timerMode.T_AUTO_RESTART = autoRestart;
    writeRegister(T_MODE, timerMode.value);
    writeRegister(T_PRESCALER_LOW, prescaler & 0xff);
    writeRegister(T_RELOAD_HIGH, (reload >> 8) & 0xff);
    writeRegister(T_RELOAD_LOW, reload & 0xff);
}

void ReaderMFRC522::startTimer() {
    setRegisterBits(CONTROL, CONTROL_T_START_NOW);
}

void ReaderMFRC522::stopTimer() {
    setRegisterBits(CONTROL, CONTROL_T_STOP_NOW);
}

void ReaderMFRC522::enableInterrupt(unsigned int interrupt) {
    setRegisterBits(MFR522_INT_TO_EN_REG(interrupt),
MFR522_INT_TO_EN_MASK(interrupt));
}

void ReaderMFRC522::disableInterrupt(unsigned int interrupt) {
    clearRegisterBits(MFR522_INT_TO_EN_REG(interrupt),
MFR522_INT_TO_EN_MASK(interrupt));
}

void ReaderMFRC522::clearInterrupt(unsigned int interrupt) {

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

void ReaderMFRC522::flushQueue() {
    setRegisterBits(FIFO_LEVEL, FIFO_LEVEL_FLUSH_BUFFER);
}

void ReaderMFRC522::setWaterLevel(unsigned char level) {
    writeRegister(WATER_LEVEL, WATER_LEVEL_WATER_LEVEL & level);
}

int ReaderMFRC522::generateRandomId(unsigned char *buf) {

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

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

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

    // Send command
    sendCommand(GENERATE_RANDOM_ID);

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

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

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

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

int ReaderMFRC522::tranceive(unsigned char *send, unsigned char *receive, unsigned
char sendLen, bool checkCrc) {
    return communicate(TRANSCEIVE, send, receive, sendLen, checkCrc);
}

inline int ReaderMFRC522::tranceive(unsigned char *send, unsigned char *receive,
unsigned char sendLen) {
    return tranceive(send, receive, sendLen, false);
}

int ReaderMFRC522::communicate(unsigned char command, unsigned char *send,
unsigned char *receive, unsigned char sendLen, bool checkCrc) {

    unsigned char len = 0;
    COM_IRQbits irq;
    ERRORbits error;
    CONTROLbits control;

    lastError = NO_ERROR;

    // 25ms before timeout, auto start timer at the end of the transmission
    configureTimer(0xf9, 0x03e8, true, false);

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

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

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

    // Write sendData to the FIFO
    writeRegisterBlock(FIFO_DATA, send, sendLen);

    // Execute the command
    sendCommand(command);

    if (command == TRANSCEIVE) {

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

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

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

    // Stop now if any errors except collisions were detected.
    // ErrorReg[7..0] bits are: WrErr TempErr reserved BufferOvfl CollErr CRCErr ParityErr ProtocolErr
    error.value = readRegister(ERROR);

    if (error.COLL_ERR) {
        lastError = COLLISION_ERROR;
        return -1;
    }

    if (error.BUFFER_OVFL || error.PARITY_ERR || error.
PROTOCOL_ERR) {
        lastError = COMMUNICATION_ERROR;
        return -1;
    }

    len = readRegister(FIFO_LEVEL);
    control.value = readRegister(CONTROL);

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

    // In this case a MIFARE Classic NAK is not OK.
    if (len == 1 && control.RX_LAST_BITS == 4 && (receive[0] != SAK && receive[0] !=
ACK)) {
        lastError = NACK;
        return -1;
    }

    // We need at least the CRC_A value and all 8 bits of the last byte must be received.
    // NOTE: casting (unsigned char) len is fine here, len > 0 and is less than FIFO size: 64
    // NOTE: control.RX_LAST_BITS = 0 means 8 bits.
    if (checkCrc && (len < 2 || control.RX_LAST_BITS != 0 || !
hasValidCrc(receive, (unsigned char) len))) {
        lastError = CRC_ERROR;
        return -1;
    }

    return len;
}

inline int ReaderMFRC522::communicate(unsigned char command, unsigned char *send,
 unsigned char *receive, unsigned char sendLen) {
    return communicate(command, send, receive, sendLen, false);
}

int ReaderMFRC522::authenticate(unsigned char *send) {
    unsigned char receive;
    return communicate(MF_AUTHENT, send, &receive, 12);
}

void ReaderMFRC522::turnOffEncryption() {
    clearRegisterBits(STATUS2, STATUS2_MF_CRYPTO1_ON);
}

bool ReaderMFRC522::hasValidCrc(unsigned char *buf, unsigned char len) {
    if (len <= 2) {
        return false;
    }
    unsigned char crc[2];
    calculateCrc(buf, len - 2, crc);
    return (buf[len - 2] == crc[0]) && (buf[len - 1] == crc[1]);
}

unsigned int ReaderMFRC522::calculateCrc(unsigned char *buf, unsigned char len)
{
    unsigned int dst;
    calculateCrc(buf, len, (unsigned char *) &dst);
    return dst;
}

void ReaderMFRC522::calculateCrc(unsigned char *buf, unsigned char len, unsigned
 char *dst) {

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

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

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

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

    // Start the calculation
    sendCommand(CALC_CRC);

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

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

    if (dst != NULL) {
        dst[0] = readRegister(CRC_RESULT_LOW);
        dst[1] = readRegister(CRC_RESULT_HIGH);
    }
}

bool ReaderMFRC522::waitForRegisterBits(unsigned char reg, unsigned char mask, unsigned long timeout) {
    unsigned char v;
    unsigned long start = millis();
    do {
        v = readRegister(reg);
    } while (!(v & mask) && start + timeout > millis());
    return (v & mask) > 0;
}

inline bool ReaderMFRC522::waitForRegisterBits(unsigned char reg, unsigned char mask) {
    return waitForRegisterBits(reg, mask, READER_DEFAULT_TIMEOUT);
}

bool ReaderMFRC522::performSelfTest() {
    unsigned char *firmwareReference;
    unsigned char buffer[64] = { 0 };
    writeRegister(AUTO_TEST, 0x00);
    softReset();
    flushQueue();
    writeRegisterBlock(FIFO_DATA, buffer, 25);
    sendCommand(MEM);
    writeRegister(AUTO_TEST, AUTO_TEST_ENABLE);
    writeRegister(FIFO_DATA, 0x00);
    sendCommand(CALC_CRC);
    waitForRegisterBits(DIV_IRQ, DIV_IRQ_CRC_IRQ, 100);
    readRegisterBlock(FIFO_DATA, buffer, 64);
    switch (getVersion()) {
    case CLONE:
        firmwareReference = (unsigned char *) FM17522_FIRMWARE_REFERENCE;
        break;
    case V0_0:
        firmwareReference = (unsigned char *) MFRC522_FIRMWARE_REFERENCE_V0_0;
        break;
    case V1_0:
        firmwareReference = (unsigned char *) MFRC522_FIRMWARE_REFERENCE_V1_0;
        break;
    case V2_0:
        firmwareReference = (unsigned char *) MFRC522_FIRMWARE_REFERENCE_V2_0;
        break;
    default:
        return false;
    }
    for (unsigned char i = 0; i < 64; i++) {
        if (buffer[i] != pgm_read_byte(&(firmwareReference[i]))) {
            return false;
        }
    }
    return true;
}

void ReaderMFRC522::setBitFraming(unsigned char rxAlign, unsigned char txLastBits) {
    BIT_FRAMINGbits f;
    f.value = readRegister(BIT_FRAMING);
    f.RX_ALIGN = rxAlign;
    f.TX_LAST_BITS = txLastBits;
    writeRegister(BIT_FRAMING, f.value);
}

unsigned char ReaderMFRC522::getCollisionPosition() {
    COLLbits coll;
    coll.value = readRegister(COLL);
    return coll.COLL_POS > 0 ? coll.COLL_POS : 32;
}

void ReaderMFRC522::setuptForAnticollision() {
    clearRegisterBits(COLL, COLL_VALUES_AFTER_COLL);
}

ReaderMFRC522::Version ReaderMFRC522::getVersion() {
    return (Version) readRegister(VERSION);
}
```

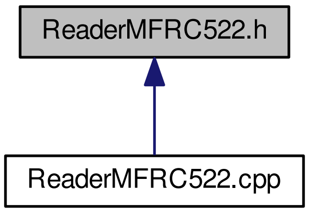
5.9 ReaderMFRC522.h File Reference

```
#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

```

#ifndef __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_READER_MFRC522_H__
#define __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_READER_MFRC522_H__ 1

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

#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

// Version 0.0 (0x90)
// Philips Semiconductors; Preliminary Specification Revision 2.0 - 01 August 2005; 16.1 self-test
const unsigned char MFRC522_FIRMWARE_REFERENCE_V0_0[] PROGMEM = { 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 };

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

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

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

class ReaderMFRC522: public Reader, public RegisterBasedDevice {

    static const unsigned char SAK = 0x08;
    static const unsigned char ACK = 0x0a;

    RegisterBasedDevice *device;

    unsigned char resetPin;

public:

    enum Register {

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

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

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

        // Interrupt request bits
        COM_IRQ = 0x04,

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

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

        // Communication status bits
        STATUS1 = 0x07,

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

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

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

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

        // Miscellaneous control registers
        CONTROL = 0x0c,

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

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

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

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

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

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

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

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

        // Selects internal receiver settings
        RX_SEL = 0x17,

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

        // Defines demodulator settings
        DEMOD = 0x19,

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

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

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

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

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

        // Controls the ModWidth setting
        MOD_WIDTH = 0x24,

        // Configures the receiver gain
        RFC_FG = 0x26,

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

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

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

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

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

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

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

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

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

        // Test signal configuration
        TEST_SEL1 = 0x31,

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

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

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

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

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

        // Shows the software version
        VERSION = 0x37,

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

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

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

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

    enum Command {

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

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

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

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

        // Transmit data
        TRANSMIT = 0x04,

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

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

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

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

        // Resets the MFRC522
        SOFT_RESET = 0x0F
    };

    enum 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_CRYPTO1_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
    };

    union COMMANDbits {

        struct {

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

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

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

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

    union COM_I_ENbits {

        struct {

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

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

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

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

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

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

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

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

    union DIV_I_ENbits {

        struct {

            // Reserved
            unsigned char :2;

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

            // Reserved
            unsigned char :1;

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

            // Reserved
            unsigned char :2;

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

    union COM_IRQbits {

        struct {

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

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

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

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

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

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

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

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

    union DIV_IRQbits {

        struct {

            // Reserved
            unsigned char :2;

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

            // Reserved
            unsigned char :1;

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

            // Reserved
            unsigned char :2;

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

    union ERRORbits {

        struct {

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

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

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

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

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

            // Reserved
            unsigned char :1;

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

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

    union STATUS1bits {

        struct {

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

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

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

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

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

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

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

    union STATUS2bits {

        struct {

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

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

            // Reserved
            unsigned char :2;

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

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

    union FIFO_LEVELbits {

        struct {

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

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

    union WATER_LEVELbits {

        struct {

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

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

    union CONTROLbits {

        struct {

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

            // Reserved
            unsigned char :2;

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

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

    union BIT_FRAMINGbits {

        struct {

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

            // Reserved
            unsigned char :1;

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

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

    union COLLbits {

        struct {

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

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

            // Reserved
            unsigned char :1;

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

    union MODEbits {

        struct {

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

            // Reserved
            unsigned char :1;

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

            // Reserved
            unsigned char :1;

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

            // Reserved
            unsigned char :1;

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

    union TX_MODEbits {

        struct {

            // Reserved
            unsigned char :3;

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

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

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

    union RX_MODEbits {

        struct {

            // Reserved
            unsigned char :2;

            // 0: receiver is deactivated after receiving a data frame
            // 1: able to receive more than one data frame only valid for data rates above 106 kBd in order to handle
            // the polling command after setting this bit the Receive and Transceive commands will not
```

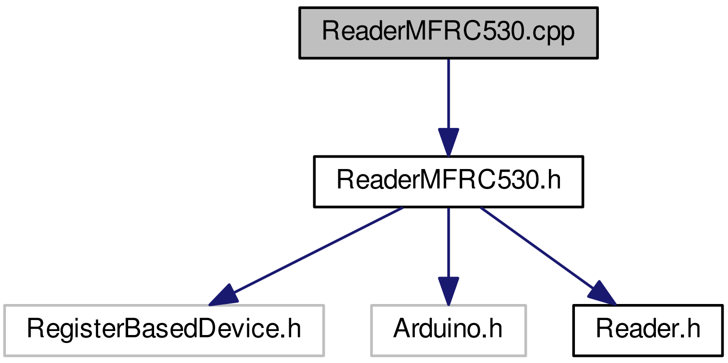
```
terminate automatically.
            // Multiple reception can only be deactivated by writing any command (except the Receive command) to the CommandReg
            // register, or by the host clearing the bit if set to logic 1, an error byte is added to the FIFO buffer at the
            // end of a received data stream which is a copy of the ErrorReg register value. For the MFRC522 version 2.0 the CRC status is
            // reflected in the signal CRCOk, which indicates the actual status of the CRC coprocessor. For the MFRC522 version 1.0 the CRC
            // status is reflected in the signal CRCErr.
            unsigned char RX_MULTIPLE :1;

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

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

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

    union TX_CONTROLbits {

        struct {

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

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

            // Reserved
            unsigned char :1;

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

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

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

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

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

        struct {

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

            // Reserved
            unsigned char :2;

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

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

    union TX_ASKbits {

        struct {

            // Reserved
            unsigned char :6;

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

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

    union TX_SELbits {

        struct {

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

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

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

    union RX_SELbits {

        struct {

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

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

    union RX_THRESHOLDbits {

        struct {

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

            // Reserved
            unsigned char :1;

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

    union DEMODbits {

        struct {

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

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

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

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

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

    union MF_TXbits {

        struct {

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

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

    union MF_RXbits {

        struct {

            // Reserved
            unsigned char :4;

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

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

    union SERIAL_SPEEDbits {

        struct {

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

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

    union RF_CFGbits {

        struct {

            // Reserved
            unsigned char :4;

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

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

    union GS_Nbits {

        struct {

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

            // defines the conductance of the output n-driver during periods without modulation which can be used to regulate the output power and
            // subsequently current consumption and operating distance
            // Remark: the conductance value is binary-weighted during soft Power-down mode the highest bit is forced to logic 1
            // value is only used if driver TX1 or TX2 is switched on
            unsigned char CW_GS_N :4;
        };
        unsigned char value;
    };

    union CW_GS_Pbits {

        struct {

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

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

    union MOD_GS_Pbits {

        struct {

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

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

    union T_MODEbits {

        struct {

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

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

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

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

    union VERSIONbits {

        struct {

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

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

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

    ReaderMFRC522(RegisterBasedDevice *device, unsigned char resetPin);

    virtual ~ReaderMFRC522();

    void initialize();

    inline 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);

    inline 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);

```

```
    inline 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);

    inline bool waitForRegisterBits(unsigned char reg, unsigned char mask);

    Version getVersion();

    bool performSelfTest();

    void setBitFraming(unsigned char rxAlign, unsigned char txLastBits);

    unsigned char getCollisionPosition();

    void setupForAnticollision();

    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);

};

#endif // __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_READER_MFRC522_H__
```

5.11 ReaderMFRC530.cpp File Reference

`#include "ReaderMFRC530.h"`

Include dependency graph for ReaderMFRC530.cpp:



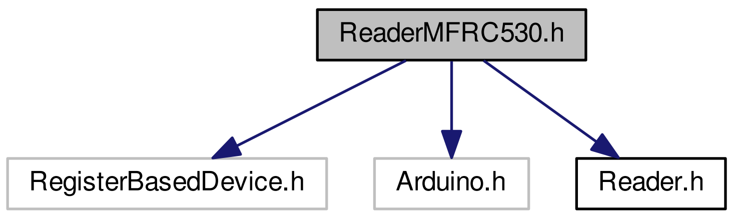
5.12 ReaderMFRC530.cpp

```
#include "ReaderMFRC530.h"

```

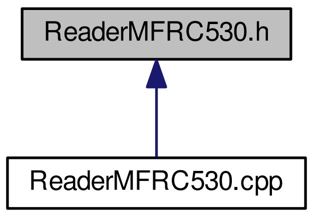
5.13 ReaderMFRC530.h File Reference

```
#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

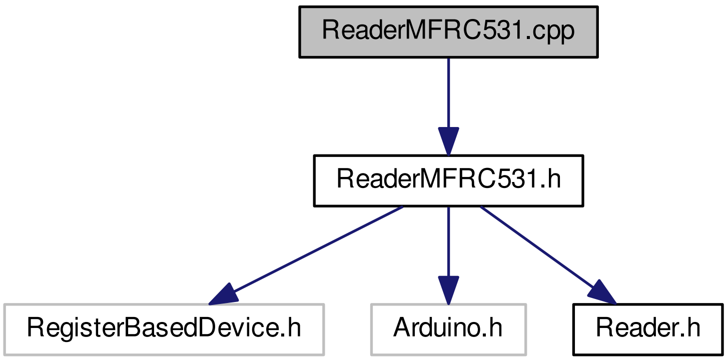
5.14 ReaderMFRC530.h

```
#ifndef __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_READER_MFRC530_H__
#define __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_READER_MFRC530_H__ 1

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

class ReaderMFRC530: public Reader, public RegisterBasedDevice {

public:

    enum Register {
    };

    enum Command {
    };

    enum Mask {
    };

    enum Interrupt
        : unsigned int {
    };

    enum Version {
    };

    ReaderMFRC530(RegisterBasedDevice *device, unsigned char resetPin);

    virtual ~ReaderMFRC530();

    void initialize();

    inline 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);

    inline 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);

    inline 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);

    inline 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);

};

#endif // __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_READER_MFRC530_H__
```

5.15 ReaderMFRC531.cpp File Reference

`#include "ReaderMFRC531.h"`

Include dependency graph for ReaderMFRC531.cpp:



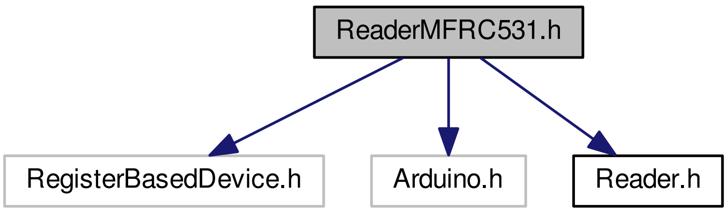
5.16 ReaderMFRC531.cpp

```
#include "ReaderMFRC531.h"

```

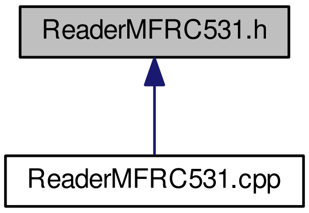
5.17 ReaderMFRC531.h File Reference

```
#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

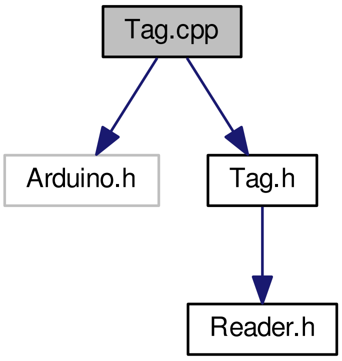
5.18 ReaderMFRC531.h

```
#ifndef __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_READER_MFRC531_H__
#define __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_READER_MFRC531_H__ 1

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

class ReaderMFRC531: public Reader, public RegisterBasedDevice {

public:

    enum Register {
    };

    enum Command {
    };

    enum Mask {
    };

    enum Interrupt
        : unsigned int {
    };

    enum Version {
    };

    ReaderMFRC531(RegisterBasedDevice *device, unsigned char resetPin);

    virtual ~ReaderMFRC531();

    void initialize();

    inline 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);

    inline 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);

    inline 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);

    inline 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);

};

#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:



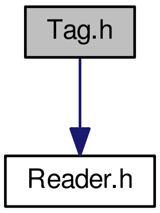
5.20 Tag.cpp

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

Tag::Tag(Reader *reader)
        : reader(reader), tagType(MIFARE_UNKNOWN), uid( { 0 }),
supportsAnticollision(false), state(POWER_OFF),
keyType(KEY_A), key(NULL), sectorTrailerProtected(
                true) {
}

Tag::~Tag() {
}

Tag::Uid Tag::getUid() {
    return uid;
}

bool Tag::hasAnticollisionSupport() {
    return supportsAnticollision;
}

Tag::TagType Tag::getTagType() {
    return tagType;
}

void Tag::setState(Tag::State state) {
    this->state = state;
}

Tag::State Tag::getState() {
    return state;
}

bool Tag::activate() {
    return request() && hasAnticollisionSupport() &&
select();
}

bool Tag::activateWakeUp() {
    return wakeUp() && hasAnticollisionSupport() &&
select();
}

bool Tag::detect(unsigned char command) {
    unsigned char buf[2] = { command, 0x00 };
    reader->turnOffEncryption();
    reader->setBitFraming(0, 0x07);
    bool ok = reader->tranceive(buf, buf, 1) >= 0;
    if (ok) {
        setState(READY);
        supportsAnticollision = buf[0] &
TAG_ATQA_ANTICOLLISION_BIT;
    }
    return ok;
}

bool Tag::request() {
    return detect(REQUEST);
}

bool Tag::wakeUp() {
    return detect(WAKE_UP);
}

bool Tag::select() {
    if (getState() != READY) {
        return false;
    }
    Command cascadeLevels[3] = { SEL_CL1, SEL_CL2, SEL_CL3 };
    unsigned char collisionPosistion = 0, knownBytes, lastBits, send[9] = { 0 }, receive[9] = { 0 }, *p = &
uid.uid[0];
    Reader::Error error;
    bool needNextCascadeLevel = true;
    uid.size = 0;
    reader->setuptForAnticollision();

    // Loop for each cascade levels.
    // Each cascade level we receive 4 bytes corresponding to the CT + ID at
    // that cascade level, followed by the BCC byte (xor of the id).
    for (unsigned char k = 0; needNextCascadeLevel && k < sizeof(cascadeLevels); k++) {

        do {
            lastBits = collisionPosistion % 8;
            knownBytes = (collisionPosistion / 8) + (lastBits ? 1 : 0);
            send[0] = cascadeLevels[k];
            send[1] = computeNvb(collisionPosistion);
            memcpy(&send[2], receive, knownBytes);
            reader->setBitFraming(lastBits, lastBits);
            reader->tranceive(send, receive, knownBytes + 2);
            collisionPosistion = 0;
            error = (Reader::Error) reader->getLastError();
            if (error != Reader::NO_ERROR && error !=
Reader::COLLISION_ERROR) {
                setState(IDLE);
                return false;
            }
            if (error == Reader::COLLISION_ERROR) {
                collisionPosistion = reader->getCollisionPosition();
            } else {

                // End of i_nth iteration.
                send[1] = 0x70;
                memcpy(&send[2], receive, 0x05);
                reader->calculateCrc(send, 7, &send[7]);
                reader->tranceive(send, receive, 0x09);
                if (reader->getLastError() != Reader::NO_ERROR) {
                    setState(IDLE);
                    return false;
                }
                uid.sak = receive[0];

                // TODO: Need more tests
                needNextCascadeLevel = (uid.sak & TAG_SAK_BIT) > 0;
                unsigned char size = 4 - needNextCascadeLevel;
                memcpy(p, &send[2 + needNextCascadeLevel], size);
                p += size;
                uid.size += size;
            }
        } while (error == Reader::COLLISION_ERROR);
    }
    computeTagType();
    setState(ACTIVE);
    return true;
}

bool Tag::halt() {
    unsigned char buf[4] = { HLT_A, 0, 0, 0 };
    reader->turnOffEncryption();
    setState(HALT);
    reader->calculateCrc(buf, 2, &buf[2]);
    reader->tranceive(buf, buf, 4);

    // If the PICC responds with any modulation during a period of 1 ms after the end of the frame
 containing the
    // HLTA command, this response shall be interpreted as ’not acknowledge’.
    return reader->getLastError() == Reader::TIMEOUT_ERROR;
}

bool Tag::authenticate(unsigned char address, KeyType type, unsigned char *
key) {
    unsigned char buf[12];
    if (getState() != ACTIVE) {
        return false;
    }
    buf[0] = (type == KEY_A) ? AUTH_KEY_A : AUTH_KEY_B;
    buf[1] = address;
    for (unsigned char i = 0; i < TAG_KEY_SIZE; i++) {
        buf[2 + i] = key[i];
    }
    for (unsigned char i = 0; i < 4; i++) {
        buf[8 + i] = uid.uid[i];
    }
    return reader->authenticate(buf) >= 0;
}

bool Tag::readBlock(unsigned char address, unsigned char *buf) {
    if (key != NULL && !authenticate(address, keyType, key)) {
        return false;
    }
    buf[0] = READ;
    buf[1] = address;
    reader->calculateCrc(buf, 2, &buf[2]);
    return reader->tranceive(buf, buf, 4, true) == 18;
}

bool Tag::writeBlock(unsigned char address, unsigned char *buf) {
    unsigned char cmd[4];
    if (isAddressSectorTrailer(address) &&
sectorTrailerProtected) {
        return false;
    }
    if (key != NULL && !authenticate(address, keyType, key)) {
        return false;
    }
    cmd[0] = WRITE;
    cmd[1] = address;
    reader->calculateCrc(cmd, 2, &cmd[2]);
    reader->tranceive(cmd, cmd, 4);
    if (reader->getLastError() == Reader::NACK) {
        return false;
    }
    reader->calculateCrc(buf, 16, &buf[16]);
    reader->tranceive(buf, buf, 18);
    return reader->getLastError() != Reader::NACK;
}

bool Tag::readBlockSlice(unsigned char address, unsigned char from, unsigned char len,
unsigned char *buf) {
    unsigned char receive[18];
    if (len == 0 || from + len > 16) {
        return false;
    }
    if (!readBlock(address, receive)) {
        return false;
    }
    memcpy(buf, &receive[from], len);
    return true;
}

bool Tag::writeBlockSlice(unsigned char address, unsigned char from, unsigned char len,
 unsigned char *buf) {
    unsigned char receive[18];
    if (len == 0 || from + len > 16) {
        return false;
    }
    if (!readBlock(address, receive)) {
        return false;
    }
    memcpy(&receive[from], buf, len);
    return writeBlock(address, receive);
}

int Tag::readByte(unsigned char address, unsigned char pos) {

    unsigned char buf[18];
    if (!readBlock(address, buf)) {
        return -1;
    }
    return buf[pos];
}

bool Tag::writeByte(unsigned char address, unsigned char pos, unsigned char value) {

    unsigned char buf[18];
    if (!readBlock(address, buf)) {
        return false;
    }
    buf[pos] = value;
    return writeBlock(address, buf);
}

bool Tag::decrement() {
    return true;
}

bool Tag::increment() {
    return true;
}

bool Tag::restore() {
    return true;
}

bool Tag::transfer() {
    return true;
}

bool Tag::setBlockType(unsigned char address, BlockType type) {

    return true;
}

bool Tag::readAccessBits(unsigned char sector, unsigned char *buf) {
    return readBlockSlice(getSectorTrailerAddress(sector), 6, 10, buf)
;
}

bool Tag::writeAccessBits(unsigned char sector, unsigned char *buf) {
    return writeBlockSlice(getSectorTrailerAddress(sector), 6, 10,
buf);
}

bool Tag::setBlockPermission(unsigned char address, unsigned char permission) {
    return true;
}

bool Tag::writeKey(unsigned char sector, KeyType type, unsigned char *
key) {
    unsigned from = TAG_KEY_TO_POS(key);
    return writeBlockSlice(getSectorTrailerAddress(sector), from,
TAG_KEY_SIZE, key);
}

bool Tag::readKey(unsigned char sector, KeyType type, unsigned char *
key) {
    unsigned from = TAG_KEY_TO_POS(key);
    return readBlockSlice(getSectorTrailerAddress(sector), from,
TAG_KEY_SIZE, key);
}

void Tag::setupAuthenticationKey(KeyType
keyType, unsigned char *key) {
    this->keyType = keyType;
    this->key = key;
}

unsigned char Tag::computeNvb(unsigned char collisionPos) {
    unsigned char bytes = collisionPos / 8;
    unsigned char bits = collisionPos % 8;
    return (((bytes << 4) & 0xf0) | (bits & 0x0f)) + 0x20;
}

void Tag::computeTagType() {
    switch (uid.sak & 0x7f) {
    case 0x04:
        tagType = MIFARE_NOT_COMPLETE;
        break;
    case 0x09:
        tagType = MIFARE_MINI;
        break;
    case 0x08:
        tagType = MIFARE_1K;
        break;
    case 0x18:
        tagType = MIFARE_4K;
        break;
    case 0x00:
        tagType = MIFARE_UL;
        break;
    case 0x10:
    case 0x11:
        tagType = MIFARE_PLUS;
        break;
    default:
        tagType = MIFARE_UNKNOWN;
    }
}

void Tag::setSectorTrailerProtected(bool protect) {
    sectorTrailerProtected = protect;
}
```

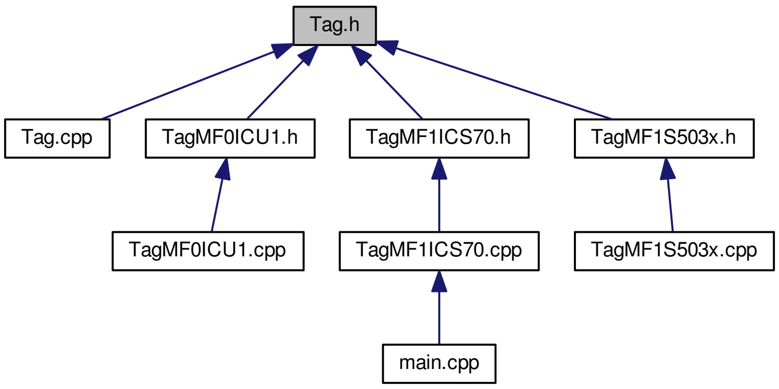
5.21 Tag.h File Reference

`#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

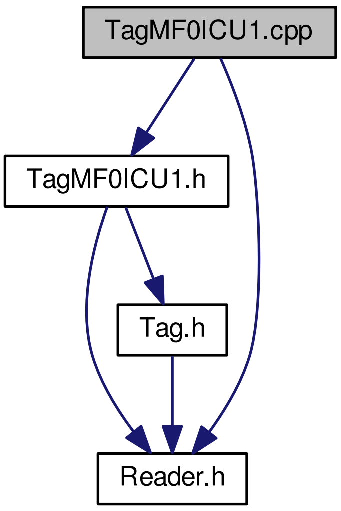
```

#ifndef __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_TAG_H__
#define __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_TAG_H__ 1

#include <Reader.h>

#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)

class Tag {

public:

    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,
    };

    // All MIFARE Classic commands use the MIFARE Crypto1 and require an authentication.
    enum Command {

        // REQuest command, Type A. Invites PICCs in state IDLE to go to READY and prepare for anticollision or selection. 7 bit frame.
        REQUEST = 0x26,

        // Wake-UP command, Type A. Invites PICCs in state IDLE and HALT to go to READY(*) and prepare for anticollision or selection. 7 bit frame.
        WAKE_UP = 0x52,

        // Anti collision/Select, Cascade Level 1
        SEL_CL1 = 0x93,

        // Anti collision/Select, Cascade Level 2
        SEL_CL2 = 0x95,

        // Anti collision/Select, Cascade Level 3
        SEL_CL3 = 0x97,

        // HaLT command, Type A. Instructs an ACTIVE PICC to go to state HALT.
        HLT_A = 0x50,

        // Perform authentication with Key A.
        AUTH_KEY_A = 0x60,

        // Perform authentication with Key B.
        AUTH_KEY_B = 0x61,

        // Reads one 16 byte block from the authenticated sector of the PICC. Also used for MIFARE Ultralight.
        READ = 0x30,

        // Writes one 16 byte block to the authenticated sector of the PICC. Called "COMPATIBILITY WRITE" for MIFARE Ultralight.
        WRITE = 0xa0,

        // Decrements the contents of a block and stores the result in the internal data register.
        DECREMENT = 0xc0,

        // Increments the contents of a block and stores the result in the internal data register.
        INCREMENT = 0xc1,

        // Reads the contents of a block into the internal data register.
        RESTORE = 0xc2,

        // Writes the contents of the internal data register to a block.
        TRANSFER = 0xb0,
    };

    struct Uid {

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

        unsigned char uid[10];

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

    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:

    Reader *reader;

    TagType tagType;

    Uid uid;

    bool supportsAnticollision;

    State state;

    KeyType keyType;

    unsigned char *key;

    bool sectorTrailerProtected;

    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();
};

#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:



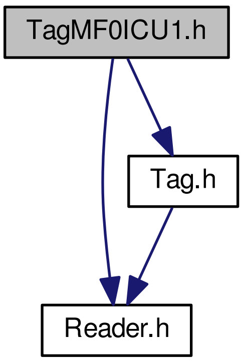
5.24 TagMF0ICU1.cpp

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

TagMF0ICU1::TagMF0ICU1(Reader *reader)
        : Tag(reader) {
}

```

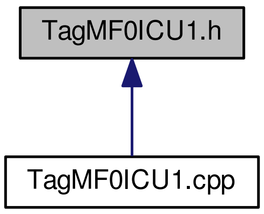
5.25 TagMF0ICU1.h File Reference

```
#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

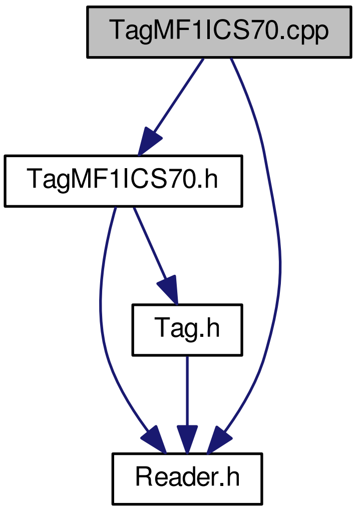
5.26 TagMF0ICU1.h

```
#ifndef __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_TAG_MF0ICU1_H__
#define __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_TAG_MF0ICU1_H__ 1

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

class TagMF0ICU1: public Tag {

public:

    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
    };

    TagMF0ICU1(Reader *reader);
};

#endif // __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_TAG_MF1S503X_H__
```

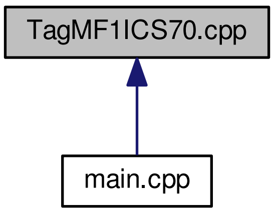
5.27 TagMF1ICS70.cpp File Reference

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

Include dependency graph for TagMF1ICS70.cpp:



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



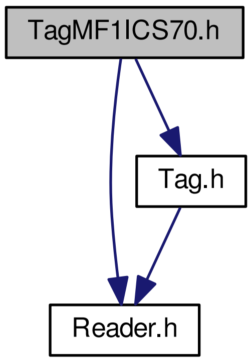
5.28 TagMF1ICS70.cpp

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

TagMF1ICS70::TagMF1ICS70(Reader *reader)
        : Tag(reader) {
}

bool TagMF1ICS70::writeBlock(unsigned char address, unsigned char *buf) {
    if (getTagType() != MIFARE_4K) {
        return false;
    }
    return Tag::writeBlock(address, buf);
}

unsigned char TagMF1ICS70::getSectorSize(unsigned char sector) {
    unsigned char size = TAG_MF1ICS70_LOW_SECTOR_SIZE;
    if (sector >= TAG_MF1ICS70_LOW_SECTOR_COUNT) {
        size = TAG_MF1ICS70_HIGH_SECTOR_SIZE;
    }
    return size;
}

unsigned char TagMF1ICS70::addressToSector(unsigned char address) {
    unsigned char sector = 0;
    if (address < TAG_MF1ICS70_LOW_MEMORY_SIZE) {
        sector = address / TAG_MF1ICS70_LOW_SECTOR_SIZE;
    } else {
        address -= TAG_MF1ICS70_LOW_MEMORY_SIZE;
        sector = TAG_MF1ICS70_LOW_SECTOR_COUNT + (address /
TAG_MF1ICS70_HIGH_SECTOR_SIZE);
    }
    return sector;
}

unsigned char TagMF1ICS70::isAddressSectorTrailer(unsigned char address)
 {
    unsigned char sectorSize = getSectorSize(addressToSector(address));
    if (address >= TAG_MF1ICS70_LOW_MEMORY_SIZE) {
        address -= TAG_MF1ICS70_LOW_MEMORY_SIZE;
    }
    return ((address % sectorSize) == (sectorSize - 1));
}

unsigned char TagMF1ICS70::getSectorTrailerAddress(unsigned char sector
) {
    unsigned char sectorSize, offset = 0;
    sectorSize = getSectorSize(sector);
    if (sector >= TAG_MF1ICS70_LOW_SECTOR_COUNT) {
        offset = TAG_MF1ICS70_LOW_MEMORY_SIZE;
        sector -= TAG_MF1ICS70_LOW_SECTOR_COUNT;
    }
    return offset + (sector * sectorSize) + (sectorSize - 1);
}
```

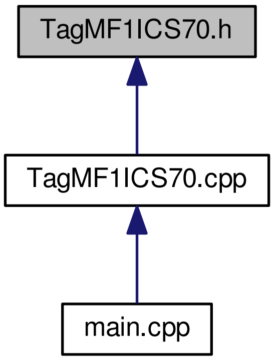
5.29 TagMF1ICS70.h File Reference

```
#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_MF1←ICS70_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.

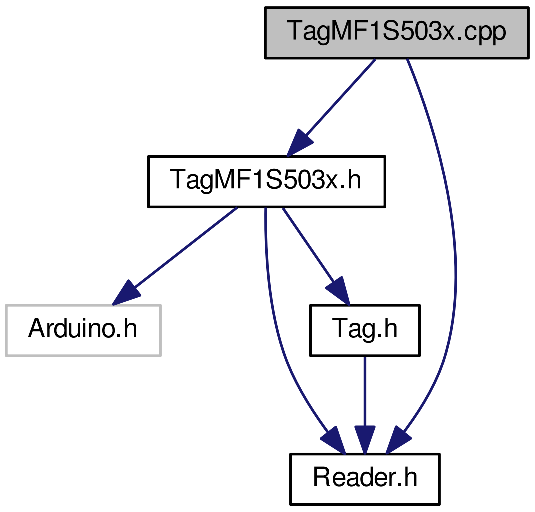
5.30 TagMF1ICS70.h

```
#ifndef __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_TAG_MF1ICS70_H__
#define __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_TAG_MF1ICS70_H__ 1

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

// The 4 kByte EEPROM memory is organized in 32 sectors with 4 blocks
// and in 8 sectors with 16 blocks. One block consists of 16 bytes.

#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

class TagMF1ICS70: public Tag {

public:

    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
    };

    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);
};

#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

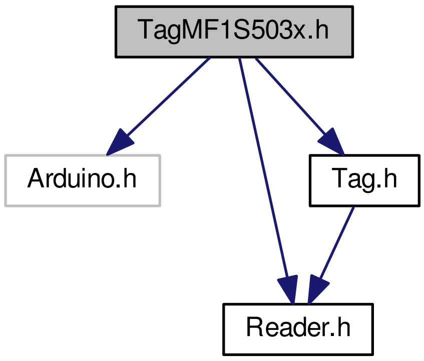
```
#include "TagMF1S503x.h"
#include <Reader.h>

TagMF1S503x::TagMF1S503x(Reader *reader)
        : Tag(reader) {
}

bool TagMF1S503x::writeBlock(unsigned char address, unsigned char *buf) {
    if (getTagType() != MIFARE_1K) {
        return false;
    }
    return Tag::writeBlock(address, buf);
}

unsigned char TagMF1S503x::getSectorSize(unsigned char sector) {
    return TAG_MF1S503X_SECTOR_SIZE;
}

unsigned char TagMF1S503x::addressToSector(unsigned char address) {
    return address / TAG_MF1S503X_SECTOR_SIZE;
}

unsigned char TagMF1S503x::isAddressSectorTrailer(unsigned char address)
 {
    return ((address % TAG_MF1S503X_SECTOR_SIZE) == (TAG_MF1S503X_SECTOR_SIZE - 1))
;
}

unsigned char TagMF1S503x::getSectorTrailerAddress(unsigned char sector
) {
    return (sector * TAG_MF1S503X_SECTOR_SIZE) + (TAG_MF1S503X_SECTOR_SIZE - 1);
}
```

5.33 TagMF1S503x.h File Reference

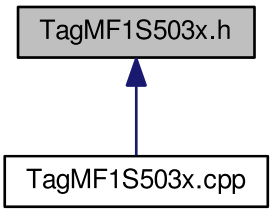
`#include <Arduino.h>`

```
#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

```

#ifndef __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_TAG_MF1S503X_H__
#define __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_TAG_MF1S503X_H__ 1

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

#define TAG_MF1S503X_SECTOR_SIZE    4

class TagMF1S503x: public Tag {

public:

    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
    };

    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);
};

#endif // __ARDUINO_RADIO_FREQUENCY_IDENTIFICATION_TAG_MF1S503X_H__
```

Index