RFAL

Modules

Data Structures

Files

Examples

RFAL (RF Abstraction Layer)

RFAL (RF Abstraction Layer) provides several functionalities required to perform RF/NFC communications.
The RFAL encapsulates the different RF ICs (ST25R3911, ST25R3916, ST25R200, ST25R500, ST25R95 and future ST25R devices) into a common and easy to use interface.

RFAL is MISRA C 2012 compliant (Motor Industry Software Reliability Association). Detailed MISRA Compliance report are available on documentation folder.

The technologies currently supported by RFAL are:

- NFC-A \ ISO14443A (T1T, T2T, T4TA)
- NFC-B \ ISO14443B (T4TB)
- NFC-F \ FeliCa (T3T)
- NFC-V \ ISO15693 (T5T)
- P2P \ ISO18092 (NFCIP1, Passive-Active P2P)
- ST25TB (ISO14443-2 Type B with Proprietary Protocol)
- PicoPass \ iClass
- B' \ Calypso
- CTS \ CTM

The protocols provided by RFAL are:

- ISO-DEP (ISO14443-4 Data Link Layer, T=CL)
- NFC-DEP (ISO18092 Data Exchange Protocol)

Internally the RFAL is divided into two sub layers:

- RF HL - RF Higher Layer
- RF HAL - RF Hardware Abstraction Layer
- RF AL - RF Abstraction Layer

RFAL

<table>
<tr><td colspan="2">RF HL</td><td colspan="9">RFAL NFC</td></tr>
<tr><td rowspan="2">R
F
A
L</td><td>Protocols</td><td colspan="5">ISO DEP</td><td colspan="4">NFC DEP</td></tr>
<tr><td>Technolo
gies</td><td>NF
C-A</td><td>NF
C-B</td><td>NF
C-F</td><td>NF
C-V</td><td>T1
T</td><td>T2
T</td><td>T4
T</td><td>ST25
TB</td><td>ST25xV</td></tr>
<tr><td colspan="2" rowspan="3">RF HAL</td><td colspan="9">RF</td></tr>
<tr><td colspan="9">RF Configs</td></tr>
<tr><td colspan="2">ST25R3911</td><td colspan="2">ST25R3916</td><td colspan="2">ST25R200</td><td colspan="2">ST25R500</td><td>ST25R95
*</td></tr>
</table>

* Future devices added to the ST25R family

Modules inside RF HAL are chip dependent, they implement the RF IC driver, config tables and the way to handle the HW specifically to perform the same functionalities.
The interface for the caller is a shared RF header file which provides the same interface for upper layers (for all chips).

Above, the RF AL is also composed by another two different sub layers:

- Technologies - Technology modules which implement all its specifics, framing, timings, etc.
- Protocols - Protocol implementation with the all framing, timings, error handling, etc

Then on top of these, there will be the application layer that will make use of the RFAL functionalities, for example: *NFC Forum Activities* (NFCC), *EMVco*, DISCO/NUCLEO demo, etc.
The RFAL NFC module provides an interface to perform the common activities as Poller/Listener device.

Access to the lowest functionalities of the ICs shall be granted by the RF module. Caller can make direct use of any of the RF, Technology or Protocol layer in a non hardware specific way.

Device support

A summary of supported modes per each device can be found bellow.
Further details are also available on each device's *rfal_features.h* file.

		ST25R95	ST25R200	ST25R500	ST25R3911	ST253916
RW	NFC-A	x	x	x	x	x
	NFC-B	x	x	x	x	x
	NFC-F	x	-	x	x	x
	NFC-V	x	x	x	x	x
CE	NFC-A	-	-	x	-	x
	NFC-F	-	-	x	-	x
P2P		-	-	x	x	x
AP2P	Initiator	-	-	-	x	x
	Target	-	-	-	x	x

Along this documentation the device names used also apply to the following part numbers:

- ST25R3911: ST25R3911B, ST25R3912, ST25R3913, ST25R3914, ST25R3915
- ST25R3916: ST25R3917, ST25R3918, ST25R3920, ST25R3915, ST25R3916B, ST25R3917B, ST25R3919B, ST25R3920B
- ST25R200: ST25R100

- ST25R500: ST25R300
- ST25R95: CR95HF, ST95HF

Setting up RFAL

In order to set up and configure the RFAL a few settings need to be defined by the user.
These settings must be provided on a file named ***rfal_platform.h*** which every RFAL modules rely on (this file can be found with every ST25R demo).
On **rfal_platform.h** user must provide:

- ***Pin definitions***
  - ST25R_SS_PIN / ST25R_SS_PORT

    Pin and Port definitions used for SPI Chip Select (CS) / Slave Select (SS) ***(mandatory)***
    The SPI CS must be controlled by SW, not automatically handled by the SPI peripheral.
    If HAL driver uses single definition to address GPIOs PORT and PIN only one definition must be defined, and then used on GPIO macros

  - ST25R_INT_PIN / ST25R_INT_PORT

    Pin and Port definitions used for ST25R IRQ line ***(mandatory)***
    The state of this GPIO is checked on the ISR.
    If HAL driver uses single definition to address GPIOs PORT and PIN only one definition must be defined, and then used on GPIO macros

  - ST25R_RESET_PIN / ST25R_RESET_PORT

    Pin and Port definitions used for ST25R RESET line ***(optional)***
    When available (applicable devices) and defined, the RESET line will be used during device initialization and power state management.
    If HAL driver uses single definition to address GPIOs PORT and PIN only one definition must be defined, and then used on GPIO macros

  - PLATFORM_LED_RX_PIN / PLATFORM_LED_RX_PORT

    Pin and Port definitions used for Reception LED ***(optional)***
    If defined LED will drive this GPIO high during a reception. If optional LED featre is not be used remove

these definitions

If HAL driver uses single definition to address GPIOs PORT and PIN only one definition must be defined, and then used on LED macros

- PLATFORM_LED_FIELD_PIN / PLATFORM_LED_FIELD_PORT

  Pin and Port definitions used for Field LED ***(optional)***

  If defined RFAL will drive this LED high when the field is turned On. This optional feature can be disabled by removing this definitions

  If HAL driver uses single definition to address GPIOs PORT and PIN only one definition must be defined, and then used on LED macros

- ***Interfaces to the user specific platform hardware driver (MCU HAL)***

  A few interfaces for the HW drivers must be provided:

  - ***platformProtectST25RComm()***

    Protects the unique access to ST25R communication channel - IRQ disable on single thread environment (MCU) ; Mutex lock on a multi thread environment ***(mandatory)***

  - **platformUnprotectST25RComm()**

    Unprotects the unique access to ST25R communication channel - IRQ enable on a single thread environment (MCU) ; Mutex unlock on a multi thread environment ***(mandatory)***

  - ***platformProtectST25RIrqStatus()***

    Protects the unique access to IRQ status var - IRQ disable on single thread environment (MCU) ; Mutex lock on a multi thread environment ***(optional)***

  - **platformUnprotectST25RIrqStatus()**

    Unprotects the IRQ status var - IRQ enable on a single thread environment (MCU) ; Mutex unlock on a multi thread environment ***(optional)***

  - **platformProtectWorker()**

    Protects \ locks the single execution of the RFAL worker ***(optional)***

    If not required, this macro shall be left empty.

  - **platformUnprotectWorker()**

Unprotects \ unlocks the unique execution of the RFAL worker ***(optional)***

If not required, this macro shall be left empty.

- **platformIrqST25RPinInitialize()**

  Initializes the ST25R IRQ pin as input / external interrupt ***(optional)***

  Upon initializing, RFAL will call this macro to configure the IRQ pin.

  If already configured by the user, this macro shall be left empty.

- **platformIrqST25RSetCallback(cb)**

  Sets the ST25R ISR ***(optional)***

  Upon initializing, RFAL will call this macro to set the callback (cb) of the IRQ pin.

  If the interrupt is already configured by the user and st25r391xIsr() being called upon an IRQ, this macro shall be left empty.

- **platformLedsInitialize()**

  Configures LEDs as outputs. ***(optional)***

  Upon initializing, RFAL will call this macro to initialize the required pins as outputs.

  If the LEDs not used or already configured by the user, this macro shall be left empty.

- **platformLedOff(port, pin)**

  Turns LED Off ***(optional)***

  If the LEDs are not used, this macro shall be left empty.

- **platformLedOn(port, pin)**

  Turns LED On ***(optional)***

  If the LEDs are not used, this macro shall be left empty.

- **platformGpioSet(port, pin)**

  Sets GPIO to logical high state ***(mandatory)***

- **platformGpioClear(port, pin)**

  Sets GPIO to logical low state ***(mandatory)***

- platformGpioToogle(port, pin)

  Toogles GPIO logical state ***(mandatory)***

- **platformGpioIsHigh(port, pin)**

Checks if the state of a GPIO is in logical high state ***(mandatory)***

- **platformGpioIsLow(port, pin)**

  Checks if the state of a GPIO is in logical low state ***(mandatory)***

- **platformTimerCreate(t)**

  Creates a timer that shall expire in the given time (t) expressed in milliseconds ***(mandatory)***
  This shall calculate a timer/tick/reference to a timeout that will be expired after the time defined. It will return this timer/tick/reference expressed as an unsigned 32 bit variable.

- **platformTimerIsExpired(timer)**

  Checks if the timer previously created is expired ***(mandatory)***
  This shall receive the timer/tick/reference returned by **platformTimerCreate()** and return true if expired.

- **platformTimerGetRemaining(timer)**

  Gets the remaining time until timer expiration ***(optional)***
  This shall receive the timer/tick/reference returned by **platformTimerCreate()** and return the remaining time until timer expiration. In case already expired zero shall be returned.

- **platformTimerDestroy(timer)**

  Stops and releases the given timer if previously created. ***(optional)***
  If not required, this macro shall be left empty.

- **platformDelay(t)**

  Delays/Blocks the MCU for the given time (t) in milliseconds ***(mandatory)***

- **platformGetSysTick()**

  Returns the current tick counter in millisecods ***(optional)***
  If not required, this macro shall be left empty.

- **platformAssert()**

  Asserts whether the given expression is true. Logging, error handle or trap otherwise. ***(optional)***
  If not required, this macro shall be left empty.

- **platformErrorHandle()**

Global error handler or trap ***(optional)***

If not required, this macro shall be left empty.

- ***platformSpiSelect()***

  SPI SS \ CS: Chip|Slave Select ***(mandatory)***

- **platformSpiDeselect()**

  SPI SS \ CS: Chip|Slave Deselect ***(mandatory)***

- **platformSpiTxRx(txBuf, rxBuf, len)**

  SPI Transceive ***(mandatory)***

  This shall transmit/receive from/to the given buffers for the given amount of bytes (len).

  The SPI Bus shall be clocked for the given number of bytes (len).

  If transmission buffer (txBuff) equals NULL, only reception shall be done, leading to zeros on the MOSI line, and the MISO bytes into rxBuff

  If reception buffer (rxBuff) equals NULL, only transmission shall be done, leading to transmission on the MOSI line, and data from MISO being ignored / not placed into the rxBuff

- **platformLog(...)**

  Logs the given message ***(optional)***

- ***RFAL modules configuration***

  Individual RFAL modules can be enabled or disabled according to the system specific needs.

  Details can be found on the following section.

RFAL Configuration

In order to enable or disable specific features user defined switches are available. The resources required by the RFAL vary according to the features that shall be supported.

The feature switches available are:

- ***RFAL_FEATURE_NFCA*** RFAL support for NFC-A (ISO14443A)
- ***RFAL_FEATURE_NFCB*** RFAL support for NFC-B (ISO14443B)

• ***RFAL_FEATURE_NFCF***	RFAL support for NFC-F (FeliCa)
• ***RFAL_FEATURE_NFCV***	RFAL support for NFC-V (ISO15693)
• ***RFAL_FEATURE_T1T***	RFAL support for T1T (Topaz)
• ***RFAL_FEATURE_T2T***	RFAL support for T2T (MIFARE Ultralight)
• ***RFAL_FEATURE_T4T***	RFAL support for T4T
• ***RFAL_FEATURE_ST25TB***	RFAL support for ST25TB
• ***RFAL_FEATURE_ST25xV***	RFAL support for ST25TV/ST25DV
• ***RFAL_FEATURE_ISO_DEP***	RFAL support for ISO-DEP (ISO14443-4)
• ***RFAL_FEATURE_ISO_DEP_POLL***	RFAL support for Poller mode (PCD) ISO-DEP (ISO14443-4)
• ***RFAL_FEATURE_ISO_DEP_LISTEN***	RFAL support for Listen mode (PICC) ISO-DEP (ISO14443-4)
• ***RFAL_FEATURE_NFC_DEP***	RFAL support for NFC-DEP (NFCIP1/P2P)
• ***RFAL_FEATURE_LISTEN_MODE***	RFAL support for Listen Mode
• ***RFAL_FEATURE_WAKEUP_MODE***	RFAL support for the Wake-Up mode
• ***RFAL_FEATURE_DPO***	RFAL support for Dynamic Power Output
• ***RFAL_FEATURE_DLMA***	RFAL support for Dynamic LMA

It is also possible to further tailor the memory resources required by configuring the max blocks to be supported with the following options:

• ***RFAL_FEATURE_ISO_DEP_IBLOCK_MAX_LEN***	ISO-DEP I-Block max length. Please use values as defined by rfalIsoDepFSx
• ***RFAL_FEATURE_NFC_DEP_BLOCK_MAX_LEN***	NFC-DEP Block/Payload length. Allowed values: 64, 128, 192, 254
• ***RFAL_FEATURE_NFC_RF_BUF_LEN***	RF buffer length used by RFAL NFC layer
• ***RFAL_FEATURE_ISO_DEP_APDU_MAX_LEN***	ISO-DEP APDU (Application Protocol Data Unit) max length. An APDU is composed by one or several chained ISO-DEP blocks
• ***RFAL_FEATURE_NFC_DEP_PDU_MAX_LEN***	NFC-DEP PDU (Protocol Data Unit) maximum length. A PDU is composed by one or several chained NFC-DEP blocks

Apart from the mandatory feature switches defined on ***rfal_platform.h***, there are certain switches that influence particular behaviours.
Those are:

- ***ST25R_COM_SINGLETXRX* -** Uses a single transceive operation over the serial interface (SPI/I2C). (Available for the ST25R3911, ST25R3916, ST25R200 and ST25R500)
  It requires an additional buffer to compute the full command which will then be used for a unique serial operation reducing its duration.

- ***RFAL_COLRES_AGC* -** Reduces bit collision recognition capabilities. (Available for the ST25R3911)
  Helpful with certain devices using noisy Active Load Modulation (e.g. Samsung Galaxy S8-US (SM-G950U)) which can be interpreted as two cards colliding: one weak and one strong.

- ***RFAL_CMATH* -** Enables the usage of C mathematical operations.
  Certain operations e.g. RSSI, Amplitude, Phase calculation, may require the usage of mathematical operations and floating point types.

- ***ST25R_SELFTEST* -** Enables ST25R-Host communication selftest during initialization.
  Helpful to diagnose porting issues.

- ***ST25R_SELFTEST_TIMER* -** Enable timer selftest during initialization.
  Helpful to diagnose porting issues.

An example ***rfal_platform.h*** for a STM32 project can be found here.

Analog Config

The Analog Config module allows to define a set configurations that will be applied at certain moments of the execution depending on the mode and protocol selected.
Each set of configurations is applied in a defined order of execution depending on the RFAL APIs used.
Further details can be found here.

The RFAL comes with a default Analog Config table for every supported device.
In case a customized user defined table is required, it can be easily deployed side by side to the RFAL package.
To do so the ***RFAL_ANALOG_CONFIG_CUSTOM*** preprocessor definition must be added together with the following variables:

- rfalAnalogConfigCustomSettings - the Analog Config table following the same RFAL format with the custom configurations
- rfalAnalogConfigCustomSettingsLength - the size in bytes of the rfalAnalogConfigCustomSettings table

Resources

Detailed analysis of compiled RFAL modules:

```
   text     data        bss        dec        hex    filename
    736        0       1036       1772        6ec    rfal_analogConfig.o
   1024        0        136       1160        488    rfal_cd.o
    288        0          0        288        120    rfal_cdHb.o
     48        0          0         48         30    rfal_crc.o
    532        0         36        536        238    rfal_dlma.o
    644        0         36        680        2a8    rfal_dpo.o
    970        4          8        982        3d6    rfal_iso15693_2.o
   5890        0        188       6078       17be    rfal_isoDep.o
   5336        0       2402       7738       1e3a    rfal_nfc.o
   4892        0        176       5068       13cc    rfal_nfcDep.o
   2246        0         84       2330        91a    rfal_nfca.o
   1510        0         48       1558        616    rfal_nfcb.o
   1380        0        336       1716        6b4    rfal_nfcf.o
   2300        0          0       2300        8fc    rfal_nfcv.o
   1012        0          0       1012        3f4    rfal_st25tb.o
   1828        0          0       1828        724    rfal_st25xv.o
    336        0          0        336        150    rfal_t1t.o
    320        0          0        320        140    rfal_t2t.o
    664        0          0        664        298    rfal_t4t.o
  31956        4       4486      36414       8e3e    (TOTALS)

  ---- ST25R3911B ----
   text     data        bss        dec        hex    filename
   9332        0       1072      10404       28a4    rfal_rfst25r3911.o
   1900        0          4       1904        770    st25r3911.o
   1482        0          0       1482        5ca    st25r3911_com.o
    832        0         20        852        354    st25r3911_interrupt.o
  13546        0       1096      14642       3932    (TOTALS)

  ---- ST25R3916 ----
   text     data        bss        dec        hex    filename
  11526        0       1104      12630       3156    rfal_rfst25r3916.o
   2010        0          4       2014        7de    st25r3916.o
```

```
   798        0          0        798        31e    st25r3916_aat.o
  1278        0          0       1278        4fe    st25r3916_com.o
   650        0         16        666        29a    st25r3916_irq.o
   340        0          0        340        154    st25r3916_led.o
 16602        0       1124      17726       453e    (TOTALS)

---- ST25R200 ----
  text     data        bss        dec        hex    filename
  6670        0        132       6802       1a92    rfal_rfst25r200.o
  1560        0          4       1564        61c    st25r200.o
  1402        0          0       1402        57a    st25r200_com.o
   670        0         16        686        2ae    st25r200_irq.o
 10302        0        152      10454       28d6    (TOTALS)

---- ST25R500 ----
  text     data        bss        dec        hex    filename
  9500        0        524      10024       2728    rfal_rfst25r500.o
  2304        0          4       2308        904    st25r500.o
  1318        0          0       1318        526    st25r500_com.o
  1188        0         36       1224        4c8    st25r500_dpocr.o
   670        0         16        686        2ae    st25r500_irq.o
 14980        0        580      15560       3cc8    (TOTALS)

---- ST25R95 ----
  text     data        bss        dec        hex    filename
  4302        0        472       4774       12a6    rfal_rfst25r95.o
   302        0          0        302        12e    st25r95.o
  1178      112          0       1290        50a    st25r95_com.o
  1756       16         28       1800        708    st25r95_com_spi.o
  8162      149        500       8811       226b    (TOTALS)
```

Examples

A RFAL usage example as a Poller device is provided here: **exampleRfalPoller.c**

On this example different devices are detected, activated and data is exchanged implementing a presence check

mechanism.

Once removed or upon error, the device is deactivated and the discovery loop restarts.

Generated on Fri Mar 14 2025 11:45:11 for RFAL by doxygen 1.9.5

RFAL

Modules

Here is a list of all modules:

[detail level 1234]

▼RFAL	
▼RFAL-HAL	RFAL Hardware Abstraction Layer
AnalogConfig	RFAL Analog Config Module
Chip	RFAL RF Chip Module
DLMA	RFAL Dynamic Load modulation
DPO	RFAL Dynamic Power Module
RF	RFAL RF Abstraction Layer
▼ST25R200	RFAL ST25R200 Driver
ST25R200_Driver	RFAL ST25R200 Driver
ST25R200_COM	RFAL ST25R200 Communications
ST25R200_IRQ	RFAL ST25R200 IRQ
▼ST25R3911	RFAL ST25R3911 Driver
ST25R3911_Driver	RFAL ST25R3911 Driver
ST25R3911_Com	RFAL ST25R3911 Communication
ST25R3911_Interrupt	RFAL ST25R3911 Interrupt

▼ST25R3916	RFAL ST25R3916 Driver
ST25R3916_Driver	RFAL ST25R3916 Driver
ST25R3916_COM	RFAL ST25R3916 Communications
ST25R3916_IRQ	RFAL ST25R3916 IRQ
ST25R3916_LED	RFAL ST25R3916 LED
▼ST25R500	RFAL ST25R500 Driver
ST25R500_Driver	RFAL ST25R500 Driver
ST25R500_COM	RFAL ST25R500 Communications
ST25R500_IRQ	RFAL ST25R500 IRQ
▼ST25R95	RFAL ST25R95 Driver
ST25R95_Driver	RFAL ST25R95 Driver
ST25R95_Com	RFAL ST25R95 Communication
▼RFAL-HL	RFAL Higher Layer
CD	RFAL Card Detection
NFC	RFAL NFC Device
▼RFAL-AL	RFAL Abstraction Layer
ISO-DEP	RFAL ISO-DEP Module
NFC-A	RFAL NFC-A Module
NFC-B	RFAL NFC-B Module
NFC-DEP	RFAL NFC-DEP Module
NFC-F	RFAL NFC-F Module
NFC-V	RFAL NFC-V Module
ST25TB	RFAL ST25TB Module
ST25xV	RFAL ST25xV Module
T1T	RFAL T1T Module
T2T	RFAL T2T Module
T4T	RFAL T4T Module

RFAL

Collaboration diagram for RFAL:

Modules

RFAL-HAL
RFAL Hardware Abstraction Layer.

RFAL-HL
RFAL Higher Layer.

RFAL-AL
RFAL Abstraction Layer.

Macros

#define **RFAL_FEATURE_WAKEUP_MODE** true

#define **RFAL_FEATURE_LOWPOWER_MODE** false

#define **RFAL_FEATURE_NFC_RF_BUF_LEN** 258U

#define **RFAL_FEATURE_ST25xV** false

#define **RFAL_FEATURE_DYNAMIC_ANALOG_CONFIG** false

#define **RFAL_FEATURE_DPO** false

#define **RFAL_FEATURE_DLMA** false

#define **platformProtectST25RIrqStatus**()

#define **platformUnprotectST25RIrqStatus**()

#define **platformProtectWorker**()

#define **platformUnprotectWorker**()

#define **platformIrqST25RPinInitialize**()

#define **platformIrqST25RSetCallback**(cb)

#define **platformLedsInitialize**()

#define **platformLedOff**(port, pin)

#define **platformLedOn**(port, pin)

#define **platformLedToggle**(port, pin)

#define **platformGetSysTick**()

#define **platformTimerGetRemaining**(timer) (0U)

#define **platformTimerDestroy**(timer)

#define **platformLog**(...)

#define **platformAssert**(exp)

#define **platformErrorHandle**()

#define **platformI2CTx**(txBuf, len, last, txOnly)

#define **platformI2CRx**(txBuf, len)

```
#define platformI2CStart()
#define platformI2CStop()
#define platformI2CRepeatStart()
#define platformI2CSlaveAddrWR(add)
#define platformI2CSlaveAddrRD(add)
#define RFAL_ERR_NONE ((ReturnCode)0U)
#define RFAL_ERR_NOMEM ((ReturnCode)1U)
#define RFAL_ERR_BUSY ((ReturnCode)2U)
#define RFAL_ERR_IO ((ReturnCode)3U)
#define RFAL_ERR_TIMEOUT ((ReturnCode)4U)
#define RFAL_ERR_REQUEST ((ReturnCode)5U)
#define RFAL_ERR_NOMSG ((ReturnCode)6U)
#define RFAL_ERR_PARAM ((ReturnCode)7U)
#define RFAL_ERR_SYSTEM ((ReturnCode)8U)
#define RFAL_ERR_FRAMING ((ReturnCode)9U)
#define RFAL_ERR_OVERRUN ((ReturnCode)10U)
#define RFAL_ERR_PROTO ((ReturnCode)11U)
#define RFAL_ERR_INTERNAL ((ReturnCode)12U)
#define RFAL_ERR_AGAIN ((ReturnCode)13U)
#define RFAL_ERR_MEM_CORRUPT ((ReturnCode)14U)
#define RFAL_ERR_NOT_IMPLEMENTED ((ReturnCode)15U)
```

```
#define RFAL_ERR_PC_CORRUPT ((ReturnCode)16U)
#define RFAL_ERR_SEND ((ReturnCode)17U)
#define RFAL_ERR_IGNORE ((ReturnCode)18U)
#define RFAL_ERR_SEMANTIC ((ReturnCode)19U)
#define RFAL_ERR_SYNTAX ((ReturnCode)20U)
#define RFAL_ERR_CRC ((ReturnCode)21U)
#define RFAL_ERR_NOTFOUND ((ReturnCode)22U)
#define RFAL_ERR_NOTUNIQUE ((ReturnCode)23U)
#define RFAL_ERR_NOTSUPP ((ReturnCode)24U)
#define RFAL_ERR_WRITE ((ReturnCode)25U)
#define RFAL_ERR_FIFO ((ReturnCode)26U)
#define RFAL_ERR_PAR ((ReturnCode)27U)
#define RFAL_ERR_DONE ((ReturnCode)28U)
#define RFAL_ERR_RF_COLLISION ((ReturnCode)29U)
#define RFAL_ERR_HW_OVERRUN ((ReturnCode)30U)
#define RFAL_ERR_RELEASE_REQ ((ReturnCode)31U)
#define RFAL_ERR_SLEEP_REQ ((ReturnCode)32U)
#define RFAL_ERR_WRONG_STATE ((ReturnCode)33U)
#define RFAL_ERR_MAX_RERUNS ((ReturnCode)34U)
#define RFAL_ERR_DISABLED ((ReturnCode)35U)
#define RFAL_ERR_HW_MISMATCH ((ReturnCode)36U)
```

```
#define RFAL_ERR_LINK_LOSS ((ReturnCode)37U)
#define RFAL_ERR_INVALID_HANDLE ((ReturnCode)38U)
#define RFAL_ERR_INCOMPLETE_BYTE ((ReturnCode)40U)
#define RFAL_ERR_INCOMPLETE_BYTE_01 ((ReturnCode)41U)
#define RFAL_ERR_INCOMPLETE_BYTE_02 ((ReturnCode)42U)
#define RFAL_ERR_INCOMPLETE_BYTE_03 ((ReturnCode)43U)
#define RFAL_ERR_INCOMPLETE_BYTE_04 ((ReturnCode)44U)
#define RFAL_ERR_INCOMPLETE_BYTE_05 ((ReturnCode)45U)
#define RFAL_ERR_INCOMPLETE_BYTE_06 ((ReturnCode)46U)
#define RFAL_ERR_INCOMPLETE_BYTE_07 ((ReturnCode)47U)
#define RFAL_EXIT_ON_ERR(r, f)
#define RFAL_EXIT_ON_BUSY(r, f)
#define RFAL_SIZEOF_ARRAY(a) (sizeof(a) / sizeof((a)[0]))
#define RFAL_MAX(a, b) (((a) > (b)) ? (a) : (b))
#define RFAL_MIN(a, b) (((a) < (b)) ? (a) : (b))
#define RFAL_GETU16(a) (((uint16_t)(a)[0] << 8) | (uint16_t)(a)[1])
#define RFAL_GETU32(a) (((uint32_t)(a)[0] << 24) | ((uint32_t)(a)[1] << 16) | ((uint32_t)(a)[2] << 8) | ((uint32_t)(a)[3]))
#define RFAL_MEMMOVE memmove
#define RFAL_MEMCPY memcpy
#define RFAL_MEMSET memset
#define RFAL_BYTECMP memcmp
```

#define **RFAL_NO_WARNING**(v) ((void) (v))

#define **NULL** (void*)0

Typedefs

typedef uint16_t **ReturnCode**

Detailed Description

This driver provides basic abstraction for communication with the ST25R3911. It uses the SPI driver for interfacing with the ST25R3911.

API:

- Read Register: **st25r3911ReadRegister**
- Modify Register: **st25r3911ModifyRegister**
- Write Register: **st25r3911WriteRegister**
- Write Multiple Registers: **st25r3911WriteMultipleRegisters**
- Load ST25R3911 FIFO with data: **st25r3911WriteFifo**
- Read from ST25R3911 FIFO: **st25r3911ReadFifo**
- Execute direct command: **st25r3911ExecuteCommand**

This driver provides basic abstraction for communication with the ST25R95. It uses the SPI driver for interfacing with the ST25R95.

API:

Macro Definition Documentation

◆ NULL

#define NULL (void*)0

represents a NULL pointer

Examples

exampleRfalPoller.c.

◆ platformAssert

```
#define platformAssert ( exp )
```

Asserts whether the given expression is true

◆ platformErrorHandle

```
#define platformErrorHandle ( )
```

Global error handler or trap

◆ platformGetSysTick

```
#define platformGetSysTick ( )
```

Get System Tick (1 tick = 1 ms)

◆ platformI2CRepeatStart

```
#define platformI2CRepeatStart ( )
```

I2C Repeat Start

◆ platformI2CRx

```
#define platformI2CRx ( txBuf,
                        len
                      )
```

I2C Receive

◆ platformI2CSlaveAddrRD

```
#define platformI2CSlaveAddrRD ( add )
```

I2C Slave address for Read operation

◆ platformI2CSlaveAddrWR

```
#define platformI2CSlaveAddrWR (  add )
```

I2C Slave address for Write operation

◆ platformI2CStart

```
#define platformI2CStart (  )
```

I2C Start condition

◆ platformI2CStop

```
#define platformI2CStop (  )
```

I2C Stop condition

◆ platformI2CTx

```
#define platformI2CTx (  txBuf,
                         len,
                         last,
                         txOnly
                      )
```

I2C Transmit

◆ platformIrqST25RPinInitialize

```
#define platformIrqST25RPinInitialize (  )
```

Initializes ST25R IRQ pin

◆ platformIrqST25RSetCallback

```
#define platformIrqST25RSetCallback ( cb )
```

Sets ST25R ISR callback

◆ platformLedOff

```
#define platformLedOff ( port,
                         pin
                       )
```

Turns the given LED Off

◆ platformLedOn

```
#define platformLedOn ( port,
                        pin
                      )
```

Turns the given LED On

◆ platformLedsInitialize

```
#define platformLedsInitialize ( )
```

Initializes the pins used as LEDs to outputs

◆ platformLedToggle

```
#define platformLedToggle ( port,
                            pin
                          )
```

Toggles the given LED

◆ platformLog

#define platformLog (...)

Log method

Examples

exampleRfalNfca.c, and **exampleRfalPoller.c**.

◆ platformProtectST25RIrqStatus

#define platformProtectST25RIrqStatus ( )

Protect unique access to IRQ status var - IRQ disable on single thread environment (MCU) ; Mutex lock on a multi thread environment

◆ platformProtectWorker

#define platformProtectWorker ( )

Protect RFAL Worker/Task/Process from concurrent execution on multi thread platforms

◆ platformTimerDestroy

#define platformTimerDestroy (timer)

Stops and released the given timer

◆ platformTimerGetRemaining

#define platformTimerGetRemaining (timer) (0U)

Gets the remaining time until expiration

◆ platformUnprotectST25RIrqStatus

#define platformUnprotectST25RIrqStatus ( )

Unprotect the IRQ status var - IRQ enable on a single thread environment (MCU) ; Mutex unlock on a multi thread environment

◆ platformUnprotectWorker

```
#define platformUnprotectWorker ( )
```

Unprotect RFAL Worker/Task/Process from concurrent execution on multi thread platforms

◆ RFAL_BYTECMP

```
#define RFAL_BYTECMP memcmp
```

map bytecmp to string library code

◆ RFAL_ERR_AGAIN

```
#define RFAL_ERR_AGAIN ((ReturnCode)13U)
```

Call again

◆ RFAL_ERR_BUSY

```
#define RFAL_ERR_BUSY ((ReturnCode)2U)
```

device or resource busy

Examples

exampleRfalPoller.c.

◆ RFAL_ERR_CRC

```
#define RFAL_ERR_CRC ((ReturnCode)21U)
```

crc error

◆ RFAL_ERR_DISABLED

#define RFAL_ERR_DISABLED ((**ReturnCode**)35U)

operation aborted due to disabled configuration

◆ RFAL_ERR_DONE

#define RFAL_ERR_DONE ((**ReturnCode**)28U)

transfer has already finished

◆ RFAL_ERR_FIFO

#define RFAL_ERR_FIFO ((**ReturnCode**)26U)

fifo over or underflow error

◆ RFAL_ERR_FRAMING

#define RFAL_ERR_FRAMING ((**ReturnCode**)9U)

Framing error

◆ RFAL_ERR_HW_MISMATCH

#define RFAL_ERR_HW_MISMATCH ((**ReturnCode**)36U)

expected hw do not match

◆ RFAL_ERR_HW_OVERRUN

#define RFAL_ERR_HW_OVERRUN ((**ReturnCode**)30U)

lost one or more received bytes

◆ RFAL_ERR_IGNORE

#define RFAL_ERR_IGNORE ((**ReturnCode**)18U)

indicates error detected but to be ignored

◆ RFAL_ERR_INCOMPLETE_BYTE

#define RFAL_ERR_INCOMPLETE_BYTE ((**ReturnCode**)40U)

Incomplete byte rcvd

◆ RFAL_ERR_INCOMPLETE_BYTE_01

#define RFAL_ERR_INCOMPLETE_BYTE_01 ((**ReturnCode**)41U)

Incomplete byte rcvd - 1 bit

◆ RFAL_ERR_INCOMPLETE_BYTE_02

#define RFAL_ERR_INCOMPLETE_BYTE_02 ((**ReturnCode**)42U)

Incomplete byte rcvd - 2 bit

◆ RFAL_ERR_INCOMPLETE_BYTE_03

#define RFAL_ERR_INCOMPLETE_BYTE_03 ((**ReturnCode**)43U)

Incomplete byte rcvd - 3 bit

◆ RFAL_ERR_INCOMPLETE_BYTE_04

#define RFAL_ERR_INCOMPLETE_BYTE_04 ((**ReturnCode**)44U)

Incomplete byte rcvd - 4 bit

◆ RFAL_ERR_INCOMPLETE_BYTE_05

#define RFAL_ERR_INCOMPLETE_BYTE_05 ((**ReturnCode**)45U)

Incomplete byte rcvd - 5 bit

◆ RFAL_ERR_INCOMPLETE_BYTE_06

#define RFAL_ERR_INCOMPLETE_BYTE_06 ((**ReturnCode**)46U)

Incomplete byte rcvd - 6 bit

◆ RFAL_ERR_INCOMPLETE_BYTE_07

#define RFAL_ERR_INCOMPLETE_BYTE_07 ((**ReturnCode**)47U)

Incomplete byte rcvd - 7 bit

◆ RFAL_ERR_INTERNAL

#define RFAL_ERR_INTERNAL ((**ReturnCode**)12U)

Internal Error

◆ RFAL_ERR_INVALID_HANDLE

#define RFAL_ERR_INVALID_HANDLE ((**ReturnCode**)38U)

invalid or not initialized device handle

◆ RFAL_ERR_IO

#define RFAL_ERR_IO ((**ReturnCode**)3U)

generic IO error

◆ RFAL_ERR_LINK_LOSS

#define RFAL_ERR_LINK_LOSS ((**ReturnCode**)37U)

Other device's field didn't behave as expected: turned off by Initiator in Passive mode, or AP2P did not turn on field

◆ RFAL_ERR_MAX_RERUNS

#define RFAL_ERR_MAX_RERUNS ((**ReturnCode**)34U)

blocking procedure reached maximum runs

◆ RFAL_ERR_MEM_CORRUPT

#define RFAL_ERR_MEM_CORRUPT ((**ReturnCode**)14U)

memory corruption

◆ RFAL_ERR_NOMEM

#define RFAL_ERR_NOMEM ((**ReturnCode**)1U)

not enough memory to perform the requested operation

◆ RFAL_ERR_NOMSG

#define RFAL_ERR_NOMSG ((**ReturnCode**)6U)

No message of desired type

◆ RFAL_ERR_NONE

#define RFAL_ERR_NONE ((**ReturnCode**)0U)

no error occurred

Examples

exampleRfalNfca.c, and **exampleRfalPoller.c**.

◆ RFAL_ERR_NOT_IMPLEMENTED

#define RFAL_ERR_NOT_IMPLEMENTED ((**ReturnCode**)15U)

not implemented

◆ RFAL_ERR_NOTFOUND

#define RFAL_ERR_NOTFOUND ((**ReturnCode**)22U)

transponder not found

◆ RFAL_ERR_NOTSUPP

#define RFAL_ERR_NOTSUPP ((**ReturnCode**)24U)

requested operation not supported

◆ RFAL_ERR_NOTUNIQUE

#define RFAL_ERR_NOTUNIQUE ((**ReturnCode**)23U)

transponder not unique - more than one transponder in field

◆ RFAL_ERR_OVERRUN

#define RFAL_ERR_OVERRUN ((**ReturnCode**)10U)

lost one or more received bytes

◆ RFAL_ERR_PAR

#define RFAL_ERR_PAR ((**ReturnCode**)27U)

parity error

◆ RFAL_ERR_PARAM

#define RFAL_ERR_PARAM ((**ReturnCode**)7U)

Parameter error

Examples

exampleRfalPoller.c.

◆ RFAL_ERR_PC_CORRUPT

#define RFAL_ERR_PC_CORRUPT ((**ReturnCode**)16U)

Program Counter has been manipulated or spike/noise trigger illegal operation

◆ RFAL_ERR_PROTO

#define RFAL_ERR_PROTO ((**ReturnCode**)11U)

protocol error

◆ RFAL_ERR_RELEASE_REQ

#define RFAL_ERR_RELEASE_REQ ((**ReturnCode**)31U)

device requested release

◆ RFAL_ERR_REQUEST

#define RFAL_ERR_REQUEST ((**ReturnCode**)5U)

invalid request or requested function can't be executed at the moment

Examples

exampleRfalPoller.c.

◆ RFAL_ERR_RF_COLLISION

#define RFAL_ERR_RF_COLLISION ((**ReturnCode**)29U)

collision error (Bit Collision or during RF Collision avoidance)

◆ RFAL_ERR_SEMANTIC

#define RFAL_ERR_SEMANTIC ((**ReturnCode**)19U)

indicates error in state machine (unexpected cmd)

◆ RFAL_ERR_SEND

#define RFAL_ERR_SEND ((**ReturnCode**)17U)

error sending

◆ RFAL_ERR_SLEEP_REQ

#define RFAL_ERR_SLEEP_REQ ((**ReturnCode**)32U)

device requested sleep

◆ RFAL_ERR_SYNTAX

#define RFAL_ERR_SYNTAX ((**ReturnCode**)20U)

indicates error in state machine (unknown cmd)

◆ RFAL_ERR_SYSTEM

#define RFAL_ERR_SYSTEM ((**ReturnCode**)8U)

System error

◆ RFAL_ERR_TIMEOUT

#define RFAL_ERR_TIMEOUT ((**ReturnCode**)4U)

error due to timeout

◆ RFAL_ERR_WRITE

#define RFAL_ERR_WRITE ((**ReturnCode**)25U)

write error

◆ RFAL_ERR_WRONG_STATE

#define RFAL_ERR_WRONG_STATE ((**ReturnCode**)33U)

incorrent state for requested operation

◆ RFAL_EXIT_ON_BUSY

#define RFAL_EXIT_ON_BUSY (r, f)

Value:

```
(r) = (f);                    \
if (RFAL_ERR_BUSY == (r))     \
{                             \
    return (r);               \
}
```

Common code to exit a function if process/function f has not concluded

◆ RFAL_EXIT_ON_ERR

#define RFAL_EXIT_ON_ERR (r, f)

Value:

```
(r) = (f);                    \
if (RFAL_ERR_NONE != (r))     \
{                             \
    return (r);               \
}
```

Common code to exit a function with the error if function f return error

◆ RFAL_FEATURE_DLMA

```
#define RFAL_FEATURE_DLMA   false
```

Dynamic LMA Module configuration missing. Disabled by default

◆ RFAL_FEATURE_DPO

```
#define RFAL_FEATURE_DPO   false
```

Dynamic Power Module configuration missing. Disabled by default

◆ RFAL_FEATURE_DYNAMIC_ANALOG_CONFIG

```
#define RFAL_FEATURE_DYNAMIC_ANALOG_CONFIG   false
```

Dynamic Analog Configs configuration missing. Disabled by default

◆ RFAL_FEATURE_LOWPOWER_MODE

```
#define RFAL_FEATURE_LOWPOWER_MODE   false
```

RFAL support for the Low Power mode, Disabled by default

◆ RFAL_FEATURE_NFC_RF_BUF_LEN

```
#define RFAL_FEATURE_NFC_RF_BUF_LEN   258U
```

RF buffer length used by RFAL NFC layer

◆ RFAL_FEATURE_ST25xV

```
#define RFAL_FEATURE_ST25xV   false
```

ST25xV Module configuration missing. Disabled by default

◆ RFAL_FEATURE_WAKEUP_MODE

```
#define RFAL_FEATURE_WAKEUP_MODE   true
```

Enable RFAL support for the Wake-Up mode

◆ RFAL_GETU16

```
#define RFAL_GETU16 (   a )   (((uint16_t)(a)[0] << 8) | (uint16_t)(a)[1])
```

Cast two Big Endian 8-bits byte array to 16-bits unsigned

◆ RFAL_GETU32

```
#define RFAL_GETU32 (   a )   (((uint32_t)(a)[0] << 24) | ((uint32_t)(a)[1] << 16) | ((uint32_t)(a)[2] << 8) | ((uint32_t)(a)[3]))
```

Cast four Big Endian 8-bit byte array to 32-bit unsigned

◆ RFAL_MAX

```
#define RFAL_MAX (   a,
                     b
                 )    (((a) > (b)) ? (a) : (b))
```

Return the maximum of the 2 values

◆ RFAL_MEMCPY

```
#define RFAL_MEMCPY   memcpy
```

map memcpy to string library code

Examples

exampleRfalPoller.c.

◆ RFAL_MEMMOVE

```
#define RFAL_MEMMOVE   memmove
```

map memmove to string library code

◆ RFAL_MEMSET

```
#define RFAL_MEMSET   memset
```

map memset to string library code

◆ RFAL_MIN

```
#define RFAL_MIN (   a,
                     b
                 )    (((a) < (b)) ? (a) : (b))
```

Return the minimum of the 2 values

◆ RFAL_NO_WARNING

```
#define RFAL_NO_WARNING (   v )    ((void) (v))
```

Macro to suppress compiler warning

◆ RFAL_SIZEOF_ARRAY

```
#define RFAL_SIZEOF_ARRAY (   a )    (sizeof(a) / sizeof((a)[0]))
```

Compute the size of an array

Typedef Documentation

◆ ReturnCode

typedef uint16_t **ReturnCode**

Standard Return Code type from function.

RFAL

Main Page

Data Structures

Files

Examples

Modules

RFAL-HAL

RFAL

RFAL Hardware Abstraction Layer. More...

Collaboration diagram for RFAL-HAL:

Modules

AnalogConfig
RFAL Analog Config Module.

Chip
RFAL RF Chip Module.

DLMA
RFAL Dynamic Load modulation.

DPO
RFAL Dynamic Power Module.

RF
RFAL RF Abstraction Layer.

ST25R200
RFAL ST25R200 Driver.

ST25R3911
RFAL ST25R3911 Driver.

ST25R3916
RFAL ST25R3916 Driver.

ST25R500
RFAL ST25R500 Driver.

ST25R95
RFAL ST25R95 Driver.

Detailed Description

Generated on Fri Mar 14 2025 11:45:08 for RFAL by doxygen 1.9.5

RFAL

AnalogConfig

RFAL Analog Config Module. More...

Collaboration diagram for AnalogConfig:

Data Structures

struct **rfalAnalogConfigRegAddrMaskVal**

struct **rfalAnalogConfig**

Macros

#define **RFAL_ANALOG_CONFIG_TBL_SIZE** (1024U)

#define **RFAL_ANALOG_CONFIG_LUT_SIZE** (87U)

#define **RFAL_ANALOG_CONFIG_LUT_NOT_FOUND** (0xFFU)

#define **RFAL_ANALOG_CONFIG_POLL_LISTEN_MODE_MASK** (0x8000U)

#define **RFAL_ANALOG_CONFIG_TECH_MASK** (0x7F00U)

#define **RFAL_ANALOG_CONFIG_BITRATE_MASK** (0x00F0U)

#define **RFAL_ANALOG_CONFIG_DIRECTION_MASK** (0x000FU)

#define **RFAL_ANALOG_CONFIG_CHIP_SPECIFIC_MASK** (0x00FFU)

#define **RFAL_ANALOG_CONFIG_POLL_LISTEN_MODE_SHIFT** (15U)

```
#define RFAL_ANALOG_CONFIG_TECH_SHIFT (8U)
```

```
#define RFAL_ANALOG_CONFIG_BITRATE_SHIFT (4U)
```

```
#define RFAL_ANALOG_CONFIG_DIRECTION_SHIFT (0U)
```

```
#define RFAL_ANALOG_CONFIG_POLL (0x0000U)
```

```
#define RFAL_ANALOG_CONFIG_LISTEN (0x8000U)
```

```
#define RFAL_ANALOG_CONFIG_TECH_CHIP (0x0000U)
```

```
#define RFAL_ANALOG_CONFIG_TECH_NFCA (0x0100U)
```

```
#define RFAL_ANALOG_CONFIG_TECH_NFCB (0x0200U)
```

```
#define RFAL_ANALOG_CONFIG_TECH_NFCF (0x0400U)
```

```
#define RFAL_ANALOG_CONFIG_TECH_AP2P (0x0800U)
```

```
#define RFAL_ANALOG_CONFIG_TECH_NFCV (0x1000U)
```

```
#define RFAL_ANALOG_CONFIG_TECH_RFU (0x2000U)
```

```
#define RFAL_ANALOG_CONFIG_TECH_RFU2 (0x4000U)
```

```
#define RFAL_ANALOG_CONFIG_BITRATE_COMMON (0x0000U)
```

#define
RFAL_ANALOG_CONFIG_BITRATE_106 (0x0010U)

#define
RFAL_ANALOG_CONFIG_BITRATE_212 (0x0020U)

#define
RFAL_ANALOG_CONFIG_BITRATE_424 (0x0030U)

#define
RFAL_ANALOG_CONFIG_BITRATE_848 (0x0040U)

#define
RFAL_ANALOG_CONFIG_BITRATE_1695 (0x0050U)

#define
RFAL_ANALOG_CONFIG_BITRATE_3390 (0x0060U)

#define
RFAL_ANALOG_CONFIG_BITRATE_6780 (0x0070U)

#define
RFAL_ANALOG_CONFIG_BITRATE_211p88 (0x0090U)

#define
RFAL_ANALOG_CONFIG_BITRATE_105p94 (0x00A0U)

#define
RFAL_ANALOG_CONFIG_BITRATE_53 (0x00B0U)

#define
RFAL_ANALOG_CONFIG_BITRATE_26 (0x00C0U)

#define
RFAL_ANALOG_CONFIG_BITRATE_1p6 (0x00D0U)

#define
RFAL_ANALOG_CONFIG_BITRATE_RFU (0x00E0U)

#define
RFAL_ANALOG_CONFIG_BITRATE_RFU2 (0x00F0U)

#define
RFAL_ANALOG_CONFIG_NO_DIRECTION (0x0000U)

#define
RFAL_ANALOG_CONFIG_TX (0x0001U)

#define
RFAL_ANALOG_CONFIG_RX (0x0002U)

#define
RFAL_ANALOG_CONFIG_ANTICOL (0x0003U)

#define
RFAL_ANALOG_CONFIG_DPO (0x0004U)

#define
RFAL_ANALOG_CONFIG_DLMA (0x0005U)

#define
RFAL_ANALOG_CONFIG_DIRECTION_RFU2 (0x0006U)

#define
RFAL_ANALOG_CONFIG_DIRECTION_RFU3 (0x0007U)

#define
RFAL_ANALOG_CONFIG_DIRECTION_RFU4 (0x0008U)

#define
RFAL_ANALOG_CONFIG_DIRECTION_RFU5 (0x0009U)

#define
RFAL_ANALOG_CONFIG_DIRECTION_RFU6 (0x000AU)

#define
RFAL_ANALOG_CONFIG_DIRECTION_RFU7 (0x000BU)

#define
RFAL_ANALOG_CONFIG_DIRECTION_RFU8 (0x000CU)

#define
RFAL_ANALOG_CONFIG_DIRECTION_RFU9 (0x000DU)

#define
RFAL_ANALOG_CONFIG_DIRECTION_RFU10 (0x000EU)

#define
RFAL_ANALOG_CONFIG_DIRECTION_RFU11 (0x000FU)

#define
RFAL_ANALOG_CONFIG_CHIP_INIT (0x0000U)

#define
RFAL_ANALOG_CONFIG_CHIP_DEINIT (0x0001U)

#define
RFAL_ANALOG_CONFIG_CHIP_FIELD_ON (0x0002U)

#define
RFAL_ANALOG_CONFIG_CHIP_FIELD_OFF (0x0003U)

#define
RFAL_ANALOG_CONFIG_CHIP_WAKEUP_ON (0x0004U)

#define
RFAL_ANALOG_CONFIG_CHIP_WAKEUP_OFF (0x0005U)

#define
RFAL_ANALOG_CONFIG_CHIP_LISTEN_ON (0x0006U)

#define
RFAL_ANALOG_CONFIG_CHIP_LISTEN_OFF (0x0007U)

#define
RFAL_ANALOG_CONFIG_CHIP_POLL_COMMON (0x0008U)

#define
RFAL_ANALOG_CONFIG_CHIP_LISTEN_COMMON (0x0009U)

#define
RFAL_ANALOG_CONFIG_CHIP_LOWPOWER_ON (0x000AU)

#define
RFAL_ANALOG_CONFIG_CHIP_LOWPOWER_OFF (0x000BU)

#define
RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_00 (0x0010U)

#define
RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_01 (0x0011U)

#define
RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_02 (0x0012U)

#define
RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_03 (0x0013U)

#define
RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_04 (0x0014U)

#define
RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_05 (0x0015U)

#define
RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_06 (0x0016U)

#define
RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_07 (0x0017U)

#define
RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_08 (0x0018U)

#define
RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_09 (0x0019U)

#define
RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_10 (0x001AU)

#define
RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_11 (0x001BU)

#define RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_12 (0x001CU)

#define RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_13 (0x001DU)

#define RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_14 (0x001EU)

#define RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_15 (0x001FU)

#define RFAL_ANALOG_CONFIG_UPDATE_LAST (0x00U)

#define RFAL_ANALOG_CONFIG_UPDATE_MORE (0x01U)

#define RFAL_ANALOG_CONFIG_ID_GET_POLL_LISTEN(id) (RFAL_ANALOG_CONFIG_POLL_LISTEN_MODE_MASK & (id))

#define RFAL_ANALOG_CONFIG_ID_GET_TECH(id) (RFAL_ANALOG_CONFIG_TECH_MASK & (id))

#define RFAL_ANALOG_CONFIG_ID_IS_CHIP(id) (RFAL_ANALOG_CONFIG_TECH_MASK & (id))

#define RFAL_ANALOG_CONFIG_ID_IS_NFCA(id) (RFAL_ANALOG_CONFIG_TECH_NFCA & (id))

#define RFAL_ANALOG_CONFIG_ID_IS_NFCB(id) (RFAL_ANALOG_CONFIG_TECH_NFCB & (id))

#define RFAL_ANALOG_CONFIG_ID_IS_NFCF(id) (RFAL_ANALOG_CONFIG_TECH_NFCF & (id))

#define RFAL_ANALOG_CONFIG_ID_IS_AP2P(id) (RFAL_ANALOG_CONFIG_TECH_AP2P & (id))

```
#define RFAL_ANALOG_CONFIG_ID_IS_NFCV(id)   (RFAL_ANALOG_CONFIG_TECH_NFCV & (id))
```

```
#define RFAL_ANALOG_CONFIG_ID_GET_BITRATE(id)   (RFAL_ANALOG_CONFIG_BITRATE_MASK & (id))
```

```
#define RFAL_ANALOG_CONFIG_ID_IS_COMMON(id)   (RFAL_ANALOG_CONFIG_BITRATE_MASK & (id))
```

```
#define RFAL_ANALOG_CONFIG_ID_IS_106(id)   (RFAL_ANALOG_CONFIG_BITRATE_106 & (id))
```

```
#define RFAL_ANALOG_CONFIG_ID_IS_212(id)   (RFAL_ANALOG_CONFIG_BITRATE_212 & (id))
```

```
#define RFAL_ANALOG_CONFIG_ID_IS_424(id)   (RFAL_ANALOG_CONFIG_BITRATE_424 & (id))
```

```
#define RFAL_ANALOG_CONFIG_ID_IS_848(id)   (RFAL_ANALOG_CONFIG_BITRATE_848 & (id))
```

```
#define RFAL_ANALOG_CONFIG_ID_IS_1695(id)   (RFAL_ANALOG_CONFIG_BITRATE_1695 & (id))
```

```
#define RFAL_ANALOG_CONFIG_ID_IS_3390(id)   (RFAL_ANALOG_CONFIG_BITRATE_3390 & (id))
```

```
#define RFAL_ANALOG_CONFIG_ID_IS_6780(id)   (RFAL_ANALOG_CONFIG_BITRATE_6780 & (id))
```

```
#define RFAL_ANALOG_CONFIG_ID_IS_26(id)   (RFAL_ANALOG_CONFIG_BITRATE_26 & (id))
```

```
#define RFAL_ANALOG_CONFIG_ID_IS_1p6(id)   (RFAL_ANALOG_CONFIG_BITRATE_1p6 & (id))
```

```
#define RFAL_ANALOG_CONFIG_ID_GET_DIRECTION(id)   (RFAL_ANALOG_CONFIG_DIRECTION_MASK & (id))
```

```
#define RFAL_ANALOG_CONFIG_ID_IS_TX(id)   (RFAL_ANALOG_CONFIG_TX & (id))
```

#define **RFAL_ANALOG_CONFIG_ID_IS_RX**(id) (**RFAL_ANALOG_CONFIG_RX** & (id))

#define **RFAL_ANALOG_CONFIG_CONFIG_NUM**(x) (sizeof(x)/sizeof((x)[0]))

#define **RFAL_ANALOG_CONFIG_ID_SET**(mode, tech, br, direction)

Typedefs

typedef uint8_t **rfalAnalogConfigMode**

typedef uint8_t **rfalAnalogConfigTech**

typedef uint8_t **rfalAnalogConfigBitrate**

typedef uint8_t **rfalAnalogConfigDirection**

typedef uint8_t **rfalAnalogConfigRegAddr**[2]

typedef uint8_t **rfalAnalogConfigRegMask**

typedef uint8_t **rfalAnalogConfigRegVal**

typedef uint16_t **rfalAnalogConfigId**

typedef uint16_t **rfalAnalogConfigOffset**

typedef uint8_t **rfalAnalogConfigNum**

Functions

void **rfalAnalogConfigInitialize** (void)

Initialize the Analog Configuration. More...

bool **rfalAnalogConfigIsReady** (void)

Indicate if the current Analog Configuration Table is complete and ready to be used. More...

ReturnCode	**rfalAnalogConfigListWriteRaw** (const uint8_t *configTbl, uint16_t configTblSize) Write the whole Analog Configuration table in raw format. More...
ReturnCode	**rfalAnalogConfigListWrite** (uint8_t more, const **rfalAnalogConfig** *config) Write the Analog Configuration table with new analog settings. More...
ReturnCode	**rfalAnalogConfigListReadRaw** (uint8_t *tblBuf, uint16_t tblBufLen, uint16_t *configTblSize) Read the whole Analog Configuration table in raw format. More...
ReturnCode	**rfalAnalogConfigListRead** (**rfalAnalogConfigOffset** *configOffset, uint8_t *more, **rfalAnalogConfig** *config, **rfalAnalogConfigNum** numConfig) Read the Analog Configuration table. More...
ReturnCode	**rfalSetAnalogConfig** (**rfalAnalogConfigId** configId) Set the Analog settings of indicated Configuration ID. More...
uint16_t	**rfalAnalogConfigGenModeID** (**rfalMode** md, **rfalBitRate** br, uint16_t dir) Generates Analog Config mode ID. More...

Detailed Description

Macro Definition Documentation

◆ RFAL_ANALOG_CONFIG_ANTICOL

#define RFAL_ANALOG_CONFIG_ANTICOL (0x0003U)

Anticollision setting in Analog Configuration ID

◆ RFAL_ANALOG_CONFIG_BITRATE_105p94

#define RFAL_ANALOG_CONFIG_BITRATE_105p94 (0x00A0U)

105.94kbits/s (ISO15693 x4) in Analog Configuration ID

◆ RFAL_ANALOG_CONFIG_BITRATE_106

```
#define RFAL_ANALOG_CONFIG_BITRATE_106   (0x0010U)
```

106kbits/s settings in Analog Configuration ID

◆ RFAL_ANALOG_CONFIG_BITRATE_1695

```
#define RFAL_ANALOG_CONFIG_BITRATE_1695   (0x0050U)
```

1695kbits/s settings in Analog Configuration ID

◆ RFAL_ANALOG_CONFIG_BITRATE_1p6

```
#define RFAL_ANALOG_CONFIG_BITRATE_1p6   (0x00D0U)
```

1.6kbit/s (1 out of 256) NFC-V setting Analog Config ID

◆ RFAL_ANALOG_CONFIG_BITRATE_211p88

```
#define RFAL_ANALOG_CONFIG_BITRATE_211p88   (0x0090U)
```

211.88kbits/s (ISO15693 x8) in Analog Configuration ID

◆ RFAL_ANALOG_CONFIG_BITRATE_212

```
#define RFAL_ANALOG_CONFIG_BITRATE_212   (0x0020U)
```

212kbits/s settings in Analog Configuration ID

◆ RFAL_ANALOG_CONFIG_BITRATE_26

```
#define RFAL_ANALOG_CONFIG_BITRATE_26   (0x00C0U)
```

26kbit/s (1 out of 4) NFC-V setting Analog Configuration ID

◆ RFAL_ANALOG_CONFIG_BITRATE_3390

```
#define RFAL_ANALOG_CONFIG_BITRATE_3390   (0x0060U)
```

3390kbits/s settings in Analog Configuration ID

◆ RFAL_ANALOG_CONFIG_BITRATE_424

```
#define RFAL_ANALOG_CONFIG_BITRATE_424   (0x0030U)
```

424kbits/s settings in Analog Configuration ID

◆ RFAL_ANALOG_CONFIG_BITRATE_53

```
#define RFAL_ANALOG_CONFIG_BITRATE_53   (0x00B0U)
```

53kbits/s (ISO15693 x2) setting in Analog Configuration ID

◆ RFAL_ANALOG_CONFIG_BITRATE_6780

```
#define RFAL_ANALOG_CONFIG_BITRATE_6780   (0x0070U)
```

6780kbits/s settings in Analog Configuration ID

◆ RFAL_ANALOG_CONFIG_BITRATE_848

```
#define RFAL_ANALOG_CONFIG_BITRATE_848   (0x0040U)
```

848kbits/s settings in Analog Configuration ID

◆ RFAL_ANALOG_CONFIG_BITRATE_COMMON

```
#define RFAL_ANALOG_CONFIG_BITRATE_COMMON   (0x0000U)
```

Common settings for all bit rates in Analog Configuration ID

◆ RFAL_ANALOG_CONFIG_BITRATE_MASK

```
#define RFAL_ANALOG_CONFIG_BITRATE_MASK   (0x00F0U)
```

Mask bits for Bit rate in Analog Configuration ID

◆ RFAL_ANALOG_CONFIG_BITRATE_RFU

```
#define RFAL_ANALOG_CONFIG_BITRATE_RFU   (0x00E0U)
```

RFU for Bitrate bits

◆ RFAL_ANALOG_CONFIG_BITRATE_RFU2

```
#define RFAL_ANALOG_CONFIG_BITRATE_RFU2   (0x00F0U)
```

RFU for Bitrate bits

◆ RFAL_ANALOG_CONFIG_BITRATE_SHIFT

```
#define RFAL_ANALOG_CONFIG_BITRATE_SHIFT   (4U)
```

Shift value for Technology in Analog Configuration ID

◆ RFAL_ANALOG_CONFIG_CHIP_DEINIT

```
#define RFAL_ANALOG_CONFIG_CHIP_DEINIT   (0x0001U)
```

Chip-Specific event: Deinitialize

◆ RFAL_ANALOG_CONFIG_CHIP_FIELD_OFF

```
#define RFAL_ANALOG_CONFIG_CHIP_FIELD_OFF   (0x0003U)
```

Chip-Specific event: Field Off

◆ RFAL_ANALOG_CONFIG_CHIP_FIELD_ON

```
#define RFAL_ANALOG_CONFIG_CHIP_FIELD_ON   (0x0002U)
```

Chip-Specific event: Field On

◆ RFAL_ANALOG_CONFIG_CHIP_INIT

```
#define RFAL_ANALOG_CONFIG_CHIP_INIT   (0x0000U)
```

Chip-Specific event: Startup;Reset;Initialize

◆ RFAL_ANALOG_CONFIG_CHIP_LISTEN_COMMON

```
#define RFAL_ANALOG_CONFIG_CHIP_LISTEN_COMMON   (0x0009U)
```

Chip-Specific event: Listen common

◆ RFAL_ANALOG_CONFIG_CHIP_LISTEN_OFF

```
#define RFAL_ANALOG_CONFIG_CHIP_LISTEN_OFF   (0x0007U)
```

Chip-Specific event: Listen Off

◆ RFAL_ANALOG_CONFIG_CHIP_LISTEN_ON

```
#define RFAL_ANALOG_CONFIG_CHIP_LISTEN_ON   (0x0006U)
```

Chip-Specific event: Listen On

◆ RFAL_ANALOG_CONFIG_CHIP_LOWPOWER_OFF

```
#define RFAL_ANALOG_CONFIG_CHIP_LOWPOWER_OFF   (0x000BU)
```

Chip-Specific event: Low Power Off

◆ RFAL_ANALOG_CONFIG_CHIP_LOWPOWER_ON

```
#define RFAL_ANALOG_CONFIG_CHIP_LOWPOWER_ON   (0x000AU)
```

Chip-Specific event: Low Power On

◆ RFAL_ANALOG_CONFIG_CHIP_POLL_COMMON

```
#define RFAL_ANALOG_CONFIG_CHIP_POLL_COMMON   (0x0008U)
```

Chip-Specific event: Poll common

◆ RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_00

```
#define RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_00   (0x0010U)
```

Chip-Specific event: Power Level 00 (e.g DPO, WLC)

◆ RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_01

```
#define RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_01   (0x0011U)
```

Chip-Specific event: Power Level 01 (e.g DPO, WLC)

◆ RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_02

```
#define RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_02   (0x0012U)
```

Chip-Specific event: Power Level 02 (e.g DPO, WLC)

◆ RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_03

```
#define RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_03   (0x0013U)
```

Chip-Specific event: Power Level 03 (e.g DPO, WLC)

◆ RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_04

```
#define RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_04   (0x0014U)
```

Chip-Specific event: Power Level 04 (e.g DPO, WLC)

◆ RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_05

```
#define RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_05   (0x0015U)
```

Chip-Specific event: Power Level 05 (e.g DPO, WLC)

◆ RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_06

```
#define RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_06   (0x0016U)
```

Chip-Specific event: Power Level 06 (e.g DPO, WLC)

◆ RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_07

```
#define RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_07   (0x0017U)
```

Chip-Specific event: Power Level 07 (e.g DPO, WLC)

◆ RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_08

```
#define RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_08   (0x0018U)
```

Chip-Specific event: Power Level 08 (e.g DPO, WLC)

◆ RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_09

```
#define RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_09   (0x0019U)
```

Chip-Specific event: Power Level 09 (e.g DPO, WLC)

◆ RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_10

```
#define RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_10   (0x001AU)
```

Chip-Specific event: Power Level 10 (e.g DPO, WLC)

◆ RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_11

```
#define RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_11   (0x001BU)
```

Chip-Specific event: Power Level 11 (e.g DPO, WLC)

◆ RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_12

```
#define RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_12   (0x001CU)
```

Chip-Specific event: Power Level 12 (e.g DPO, WLC)

◆ RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_13

```
#define RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_13   (0x001DU)
```

Chip-Specific event: Power Level 13 (e.g DPO, WLC)

◆ RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_14

```
#define RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_14   (0x001EU)
```

Chip-Specific event: Power Level 14 (e.g DPO, WLC)

◆ RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_15

```
#define RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_15   (0x001FU)
```

Chip-Specific event: Power Level 15 (e.g DPO, WLC)

◆ RFAL_ANALOG_CONFIG_CHIP_SPECIFIC_MASK

```
#define RFAL_ANALOG_CONFIG_CHIP_SPECIFIC_MASK   (0x00FFU)
```

Mask bits for Chip Specific Technology

◆ RFAL_ANALOG_CONFIG_CHIP_WAKEUP_OFF

```
#define RFAL_ANALOG_CONFIG_CHIP_WAKEUP_OFF   (0x0005U)
```

Chip-Specific event: Wake-up Off

◆ RFAL_ANALOG_CONFIG_CHIP_WAKEUP_ON

```
#define RFAL_ANALOG_CONFIG_CHIP_WAKEUP_ON   (0x0004U)
```

Chip-Specific event: Wake-up On

◆ RFAL_ANALOG_CONFIG_CONFIG_NUM

```
#define RFAL_ANALOG_CONFIG_CONFIG_NUM (   x )   (sizeof(x)/sizeof((x)[0]))
```

Get Analog Config number

◆ RFAL_ANALOG_CONFIG_DIRECTION_MASK

```
#define RFAL_ANALOG_CONFIG_DIRECTION_MASK   (0x000FU)
```

Mask bits for Direction in Analog Configuration ID

◆ RFAL_ANALOG_CONFIG_DIRECTION_RFU10

```
#define RFAL_ANALOG_CONFIG_DIRECTION_RFU10   (0x000EU)
```

RFU for Direction bits

◆ RFAL_ANALOG_CONFIG_DIRECTION_RFU11

```
#define RFAL_ANALOG_CONFIG_DIRECTION_RFU11   (0x000FU)
```

RFU for Direction bits

◆ RFAL_ANALOG_CONFIG_DIRECTION_RFU2

```
#define RFAL_ANALOG_CONFIG_DIRECTION_RFU2   (0x0006U)
```

RFU for Direction bits

◆ RFAL_ANALOG_CONFIG_DIRECTION_RFU3

```
#define RFAL_ANALOG_CONFIG_DIRECTION_RFU3   (0x0007U)
```

RFU for Direction bits

◆ RFAL_ANALOG_CONFIG_DIRECTION_RFU4

```
#define RFAL_ANALOG_CONFIG_DIRECTION_RFU4   (0x0008U)
```

RFU for Direction bits

◆ RFAL_ANALOG_CONFIG_DIRECTION_RFU5

```
#define RFAL_ANALOG_CONFIG_DIRECTION_RFU5   (0x0009U)
```

RFU for Direction bits

◆ RFAL_ANALOG_CONFIG_DIRECTION_RFU6

```
#define RFAL_ANALOG_CONFIG_DIRECTION_RFU6   (0x000AU)
```

RFU for Direction bits

◆ RFAL_ANALOG_CONFIG_DIRECTION_RFU7

```
#define RFAL_ANALOG_CONFIG_DIRECTION_RFU7   (0x000BU)
```

RFU for Direction bits

◆ RFAL_ANALOG_CONFIG_DIRECTION_RFU8

```
#define RFAL_ANALOG_CONFIG_DIRECTION_RFU8   (0x000CU)
```

RFU for Direction bits

◆ RFAL_ANALOG_CONFIG_DIRECTION_RFU9

```
#define RFAL_ANALOG_CONFIG_DIRECTION_RFU9   (0x000DU)
```

RFU for Direction bits

◆ RFAL_ANALOG_CONFIG_DIRECTION_SHIFT

```
#define RFAL_ANALOG_CONFIG_DIRECTION_SHIFT   (0U)
```

Shift value for Direction in Analog Configuration ID

◆ RFAL_ANALOG_CONFIG_DLMA

```
#define RFAL_ANALOG_CONFIG_DLMA   (0x0005U)
```

DLMA setting in Analog Configuration ID

◆ RFAL_ANALOG_CONFIG_DPO

```
#define RFAL_ANALOG_CONFIG_DPO   (0x0004U)
```

DPO setting in Analog Configuration ID

◆ RFAL_ANALOG_CONFIG_ID_GET_BITRATE

#define RFAL_ANALOG_CONFIG_ID_GET_BITRATE (id) (**RFAL_ANALOG_CONFIG_BITRATE_MASK** & (id))

Get Bitrate of Configuration ID

◆ RFAL_ANALOG_CONFIG_ID_GET_DIRECTION

#define RFAL_ANALOG_CONFIG_ID_GET_DIRECTION (id) (**RFAL_ANALOG_CONFIG_DIRECTION_MASK** & (id))

Get Direction of Configuration ID

◆ RFAL_ANALOG_CONFIG_ID_GET_POLL_LISTEN

#define RFAL_ANALOG_CONFIG_ID_GET_POLL_LISTEN (id) (**RFAL_ANALOG_CONFIG_POLL_LISTEN_MODE_MASK** & (id))

Check if id indicates Listen mode

◆ RFAL_ANALOG_CONFIG_ID_GET_TECH

#define RFAL_ANALOG_CONFIG_ID_GET_TECH (id) (**RFAL_ANALOG_CONFIG_TECH_MASK** & (id))

Get the technology of Configuration ID

◆ RFAL_ANALOG_CONFIG_ID_IS_106

#define RFAL_ANALOG_CONFIG_ID_IS_106 (id) (**RFAL_ANALOG_CONFIG_BITRATE_106** & (id))

Check if ID indicates 106kbits/s

◆ RFAL_ANALOG_CONFIG_ID_IS_1695

#define RFAL_ANALOG_CONFIG_ID_IS_1695 (id) (**RFAL_ANALOG_CONFIG_BITRATE_1695** & (id))

Check if ID indicates 1695kbits/s

◆ RFAL_ANALOG_CONFIG_ID_IS_1p6

#define RFAL_ANALOG_CONFIG_ID_IS_1p6 (id) (**RFAL_ANALOG_CONFIG_BITRATE_1p6** & (id))

Check if ID indicates 1 out of 256 bitrate

◆ RFAL_ANALOG_CONFIG_ID_IS_212

#define RFAL_ANALOG_CONFIG_ID_IS_212 (id) (**RFAL_ANALOG_CONFIG_BITRATE_212** & (id))

Check if ID indicates 212kbits/s

◆ RFAL_ANALOG_CONFIG_ID_IS_26

#define RFAL_ANALOG_CONFIG_ID_IS_26 (id) (**RFAL_ANALOG_CONFIG_BITRATE_26** & (id))

Check if ID indicates 1 out of 4 bitrate

◆ RFAL_ANALOG_CONFIG_ID_IS_3390

#define RFAL_ANALOG_CONFIG_ID_IS_3390 (id) (**RFAL_ANALOG_CONFIG_BITRATE_3390** & (id))

Check if ID indicates 3390kbits/s

◆ RFAL_ANALOG_CONFIG_ID_IS_424

#define RFAL_ANALOG_CONFIG_ID_IS_424 (id) (**RFAL_ANALOG_CONFIG_BITRATE_424** & (id))

Check if ID indicates 424kbits/s

◆ RFAL_ANALOG_CONFIG_ID_IS_6780

#define RFAL_ANALOG_CONFIG_ID_IS_6780 (id) (**RFAL_ANALOG_CONFIG_BITRATE_6780** & (id))

Check if ID indicates 6780kbits/s

◆ RFAL_ANALOG_CONFIG_ID_IS_848

#define RFAL_ANALOG_CONFIG_ID_IS_848 (id) (**RFAL_ANALOG_CONFIG_BITRATE_848** & (id))

Check if ID indicates 848kbits/s

◆ RFAL_ANALOG_CONFIG_ID_IS_AP2P

#define RFAL_ANALOG_CONFIG_ID_IS_AP2P (id) (**RFAL_ANALOG_CONFIG_TECH_AP2P** & (id))

Check if ID indicates AP2P

◆ RFAL_ANALOG_CONFIG_ID_IS_CHIP

#define RFAL_ANALOG_CONFIG_ID_IS_CHIP (id) (**RFAL_ANALOG_CONFIG_TECH_MASK** & (id))

Check if ID indicates Chip-specific

◆ RFAL_ANALOG_CONFIG_ID_IS_COMMON

#define RFAL_ANALOG_CONFIG_ID_IS_COMMON (id) (**RFAL_ANALOG_CONFIG_BITRATE_MASK** & (id))

Check if ID indicates common bitrate

◆ RFAL_ANALOG_CONFIG_ID_IS_NFCA

#define RFAL_ANALOG_CONFIG_ID_IS_NFCA (id) (**RFAL_ANALOG_CONFIG_TECH_NFCA** & (id))

Check if ID indicates NFC-A

◆ RFAL_ANALOG_CONFIG_ID_IS_NFCB

#define RFAL_ANALOG_CONFIG_ID_IS_NFCB (id) (**RFAL_ANALOG_CONFIG_TECH_NFCB** & (id))

Check if ID indicates NFC-B

◆ RFAL_ANALOG_CONFIG_ID_IS_NFCF

#define RFAL_ANALOG_CONFIG_ID_IS_NFCF (id) (**RFAL_ANALOG_CONFIG_TECH_NFCF** & (id))

Check if ID indicates NFC-F

◆ RFAL_ANALOG_CONFIG_ID_IS_NFCV

#define RFAL_ANALOG_CONFIG_ID_IS_NFCV (id) (**RFAL_ANALOG_CONFIG_TECH_NFCV** & (id))

Check if ID indicates NFC-V

◆ RFAL_ANALOG_CONFIG_ID_IS_RX

#define RFAL_ANALOG_CONFIG_ID_IS_RX (id) (**RFAL_ANALOG_CONFIG_RX** & (id))

Check if id indicates RX

◆ RFAL_ANALOG_CONFIG_ID_IS_TX

#define RFAL_ANALOG_CONFIG_ID_IS_TX (id) (**RFAL_ANALOG_CONFIG_TX** & (id))

Check if id indicates TX

◆ RFAL_ANALOG_CONFIG_ID_SET

#define RFAL_ANALOG_CONFIG_ID_SET (mode, tech, br, direction)

Value:

```
(  RFAL_ANALOG_CONFIG_ID_GET_POLL_LISTEN(mode) \
 | RFAL_ANALOG_CONFIG_ID_GET_TECH(tech) \
 | RFAL_ANALOG_CONFIG_ID_GET_BITRATE(br) \
 | RFAL_ANALOG_CONFIG_ID_GET_DIRECTION(direction) \
)
```

Set Analog Config ID value by: Mode, Technology, Bitrate and Direction

◆ RFAL_ANALOG_CONFIG_LISTEN

```
#define RFAL_ANALOG_CONFIG_LISTEN   (0x8000U)
```

Listen Mode bit setting in Analog Configuration ID

◆ RFAL_ANALOG_CONFIG_LUT_NOT_FOUND

```
#define RFAL_ANALOG_CONFIG_LUT_NOT_FOUND   (0xFFU)
```

Index value indicating no Configuration IDs found

◆ RFAL_ANALOG_CONFIG_LUT_SIZE

```
#define RFAL_ANALOG_CONFIG_LUT_SIZE   (87U)
```

Maximum number of Configuration IDs in the Loop Up Table

◆ RFAL_ANALOG_CONFIG_NO_DIRECTION

```
#define RFAL_ANALOG_CONFIG_NO_DIRECTION   (0x0000U)
```

No direction setting in Analog Conf ID (Chip Specific only)

◆ RFAL_ANALOG_CONFIG_POLL

```
#define RFAL_ANALOG_CONFIG_POLL   (0x0000U)
```

Poll Mode bit setting in Analog Configuration ID

◆ RFAL_ANALOG_CONFIG_POLL_LISTEN_MODE_MASK

```
#define RFAL_ANALOG_CONFIG_POLL_LISTEN_MODE_MASK   (0x8000U)
```

Mask bit of Poll Mode in Analog Configuration ID

◆ RFAL_ANALOG_CONFIG_POLL_LISTEN_MODE_SHIFT

```
#define RFAL_ANALOG_CONFIG_POLL_LISTEN_MODE_SHIFT   (15U)
```

Shift value of Poll Mode in Analog Configuration ID

◆ RFAL_ANALOG_CONFIG_RX

```
#define RFAL_ANALOG_CONFIG_RX   (0x0002U)
```

Reception bit setting in Analog Configuration ID

◆ RFAL_ANALOG_CONFIG_TBL_SIZE

```
#define RFAL_ANALOG_CONFIG_TBL_SIZE   (1024U)
```

Maximum number of Register-Mask-Value in the Setting List

◆ RFAL_ANALOG_CONFIG_TECH_AP2P

```
#define RFAL_ANALOG_CONFIG_TECH_AP2P   (0x0800U)
```

AP2P Technology bits setting in Analog Configuration ID

◆ RFAL_ANALOG_CONFIG_TECH_CHIP

```
#define RFAL_ANALOG_CONFIG_TECH_CHIP   (0x0000U)
```

Chip-Specific bit setting in Analog Configuration ID

◆ RFAL_ANALOG_CONFIG_TECH_MASK

```
#define RFAL_ANALOG_CONFIG_TECH_MASK   (0x7F00U)
```

Mask bits for Technology in Analog Configuration ID

◆ RFAL_ANALOG_CONFIG_TECH_NFCA

```
#define RFAL_ANALOG_CONFIG_TECH_NFCA   (0x0100U)
```

NFC-A Technology bits setting in Analog Configuration ID

◆ RFAL_ANALOG_CONFIG_TECH_NFCB

```
#define RFAL_ANALOG_CONFIG_TECH_NFCB   (0x0200U)
```

NFC-B Technology bits setting in Analog Configuration ID

◆ RFAL_ANALOG_CONFIG_TECH_NFCF

```
#define RFAL_ANALOG_CONFIG_TECH_NFCF   (0x0400U)
```

NFC-F Technology bits setting in Analog Configuration ID

◆ RFAL_ANALOG_CONFIG_TECH_NFCV

```
#define RFAL_ANALOG_CONFIG_TECH_NFCV   (0x1000U)
```

NFC-V Technology bits setting in Analog Configuration ID

◆ RFAL_ANALOG_CONFIG_TECH_RFU

```
#define RFAL_ANALOG_CONFIG_TECH_RFU   (0x2000U)
```

RFU for Technology bits

◆ RFAL_ANALOG_CONFIG_TECH_RFU2

```
#define RFAL_ANALOG_CONFIG_TECH_RFU2   (0x4000U)
```

RFU for Technology bits

◆ RFAL_ANALOG_CONFIG_TECH_SHIFT

```
#define RFAL_ANALOG_CONFIG_TECH_SHIFT   (8U)
```

Shift value for Technology in Analog Configuration ID

◆ RFAL_ANALOG_CONFIG_TX

```
#define RFAL_ANALOG_CONFIG_TX   (0x0001U)
```

Transmission bit setting in Analog Configuration ID

◆ RFAL_ANALOG_CONFIG_UPDATE_LAST

```
#define RFAL_ANALOG_CONFIG_UPDATE_LAST   (0x00U)
```

Value indicating Last configuration set during update

◆ RFAL_ANALOG_CONFIG_UPDATE_MORE

```
#define RFAL_ANALOG_CONFIG_UPDATE_MORE   (0x01U)
```

Value indicating More configuration set coming during update

Typedef Documentation

◆ rfalAnalogConfigBitrate

typedef uint8_t **rfalAnalogConfigBitrate**

Bitrate of Configuration

◆ rfalAnalogConfigDirection

typedef uint8_t **rfalAnalogConfigDirection**

Transmit/Receive direction of Configuration

◆ rfalAnalogConfigId

typedef uint16_t **rfalAnalogConfigId**

Analog Configuration ID

◆ rfalAnalogConfigMode

typedef uint8_t **rfalAnalogConfigMode**

Polling or Listening Mode of Configuration

◆ rfalAnalogConfigNum

typedef uint8_t **rfalAnalogConfigNum**

Number of Analog settings for the respective Configuration ID

◆ rfalAnalogConfigOffset

typedef uint16_t **rfalAnalogConfigOffset**

Analog Configuration offset address in the table

◆ rfalAnalogConfigRegAddr

typedef uint8_t rfalAnalogConfigRegAddr[2]

Register Address to ST Chip

◆ rfalAnalogConfigRegMask

typedef uint8_t **rfalAnalogConfigRegMask**

Register Mask Value

◆ rfalAnalogConfigRegVal

typedef uint8_t **rfalAnalogConfigRegVal**

Register Value

◆ rfalAnalogConfigTech

typedef uint8_t **rfalAnalogConfigTech**

Technology of Configuration

Function Documentation

◆ rfalAnalogConfigGenModeID()

```
uint16_t rfalAnalogConfigGenModeID ( rfalMode     md,
                                     rfalBitRate  br,
                                     uint16_t     dir
                                   )
```

Converts RFAL mode and bitrate into Analog Config Mode ID.

Update the chip with indicated analog settings of indicated Configuration ID.

Parameters

[in] **md** : RFAL mode format

[in] **br** : RFAL bit rate format

[in] **dir** : Analog Config communication direction

Returns

Analog Config Mode ID

◆ rfalAnalogConfigInitialize()

void rfalAnalogConfigInitialize (void)

Reset the Analog Configuration LUT pointer to reference to default settings.

Examples

exampleRfalPoller.c.

◆ rfalAnalogConfigIsReady()

```
bool rfalAnalogConfigIsReady ( void )
```

Returns

true if current Analog Configuration Table is complete and ready to be used.

false if current Analog Configuration Table is incomplete

◆ rfalAnalogConfigListRead()

```
ReturnCode rfalAnalogConfigListRead ( rfalAnalogConfigOffset * configOffset,
                                      uint8_t *                 more,
                                      rfalAnalogConfig *        config,
                                      rfalAnalogConfigNum       numConfig
                                    )
```

Read the Analog Configuration Table

Parameters

[in]	**configOffset**	: offset to the next Configuration ID in the List Table to be read.
[out]	**more**	: 0x00 indicates it is last Configuration ID settings; 0x01 indicates more Configuration ID setting(s) are coming.
[out]	**config**	: configuration id, number of configuration sets and register-mask-value sets
[in]	**numConfig**	: the remaining configuration settings space available;

Returns

RFAL_ERR_NOMEM : if number of Configuration for respective Configuration ID is greater the the remaining configuration setting space available

RFAL_ERR_NONE : if read is successful

◆ rfalAnalogConfigListReadRaw()

```
ReturnCode rfalAnalogConfigListReadRaw ( uint8_t *    tblBuf,
```

```
                                        uint16_t    tblBufLen,
                                        uint16_t *  configTblSize
                                      )
```

Reads the whole Analog Configuration Table in raw format

Parameters

- [out] **tblBuf** : location to the buffer to place the Config Table
- [in] **tblBufLen** : length of the buffer to place the Config Table
- [out] **configTblSize** : Config Table size

Returns

RFAL_ERR_PARAM : if configTbl or configTblSize is invalid

RFAL_ERR_NOMEM : if configTblSize is not enough for the whole table

RFAL_ERR_NONE : if read is successful

◆ rfalAnalogConfigListWrite()

```
ReturnCode rfalAnalogConfigListWrite ( uint8_t                    more,
                                       const rfalAnalogConfig *   config
                                     )
```

Writes the Analog Configuration and Look Up Table with the new list of register-mask-value and Configuration ID respectively.

NOTE: Function does not check for the validity of the Register Address.

Parameters

- [in] **more** : 0x00 indicates it is last Configuration ID settings; 0x01 indicates more Configuration ID setting(s) are coming.
- [in] ***config** : reference to the configuration list of current Configuraiton ID.

Returns

RFAL_ERR_PARAM : if Configuration ID or parameter is invalid

RFAL_ERR_NOMEM : if LUT is full

RFAL_ERR_REQUEST : if the update Configuration Id is disabled

RFAL_ERR_NONE : if setting is updated

◆ rfalAnalogConfigListWriteRaw()

```
ReturnCode rfalAnalogConfigListWriteRaw ( const uint8_t *  configTbl,
                                          uint16_t         configTblSize
                                        )
```

Writes the Analog Configuration and Look Up Table with the given raw table

NOTE: Function does not check the validity of the given Table contents

Parameters

[in] **configTbl** : location of config Table to be loaded

[in] **configTblSize** : size of the config Table to be loaded

Returns

RFAL_ERR_NONE : if setting is updated

RFAL_ERR_PARAM : if configTbl is invalid

RFAL_ERR_NOMEM : if the given Table is bigger exceeds the max size

RFAL_ERR_REQUEST : if the update Configuration Id is disabled

◆ rfalSetAnalogConfig()

```
ReturnCode rfalSetAnalogConfig ( rfalAnalogConfigId  configId )
```

Update the chip with indicated analog settings of indicated Configuration ID.

Parameters

[in] **configId** : configuration ID

Returns

RFAL_ERR_PARAM : if Configuration ID is invalid

RFAL_ERR_INTERNAL : if error updating setting to chip

RFAL_ERR_NONE : if new settings is applied to chip

rfalAnalogConfigRegAddrMaskVal Struct Reference


```
#include <rfal_analogConfig.h>
```

Data Fields

rfalAnalogConfigRegAddr	addr
rfalAnalogConfigRegMask	mask
rfalAnalogConfigRegVal	val

Detailed Description

Struct that contain the Register-Mask-Value set. Make sure that the whole structure size is even and unaligned!

Field Documentation

◆ addr

rfalAnalogConfigRegAddr rfalAnalogConfigRegAddrMaskVal::addr

Register Address

◆ mask

rfalAnalogConfigRegMask rfalAnalogConfigRegAddrMaskVal::mask

Register Mask Value

◆ val

rfalAnalogConfigRegVal rfalAnalogConfigRegAddrMaskVal::val

Register Value

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

- **rfal_analogConfig.h**

---

rfalAnalogConfig Struct Reference


```
#include <rfal_analogConfig.h>
```

Collaboration diagram for rfalAnalogConfig:

[legend]

Data Fields

uint8_t	**id** [sizeof(**rfalAnalogConfigId**)]
rfalAnalogConfigNum	**num**
rfalAnalogConfigRegAddrMaskVal	**regSet** []

Detailed Description

Struct that represents the Analog Configs

Field Documentation

◆ id

uint8_t rfalAnalogConfig::id[sizeof(**rfalAnalogConfigId**)]

Configuration ID

◆ num

rfalAnalogConfigNum rfalAnalogConfig::num

Number of Config Sets to follow

◆ regSet

rfalAnalogConfigRegAddrMaskVal rfalAnalogConfig::regSet[]

Register-Mask-Value sets

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

- **rfal_analogConfig.h**

Generated on Fri Mar 14 2025 11:45:10 for RFAL by doxygen 1.9.5

RFAL

Chip

RFAL RF Chip Module. More...

Collaboration diagram for Chip:

Functions

ReturnCode **rfalChipWriteReg** (uint16_t reg, const uint8_t *values, uint8_t len)
Writes a register on the RF Chip. More...

ReturnCode **rfalChipReadReg** (uint16_t reg, uint8_t *values, uint8_t len)
Reads a register on the RF Chip. More...

ReturnCode **rfalChipChangeRegBits** (uint16_t reg, uint8_t valueMask, uint8_t value)
Change a register on the RF Chip. More...

ReturnCode **rfalChipWriteTestReg** (uint16_t reg, uint8_t value)
Writes a Test register on the RF Chip. More...

ReturnCode **rfalChipReadTestReg** (uint16_t reg, uint8_t *value)
Reads a Test register on the RF Chip. More...

ReturnCode **rfalChipChangeTestRegBits** (uint16_t reg, uint8_t valueMask, uint8_t value)
Change a Test register on the RF Chip. More...

ReturnCode **rfalChipExecCmd** (uint16_t cmd)
Execute command on the RF Chip. More...

ReturnCode **rfalChipSetRFO** (uint8_t rfo)
Set RFO. More...

ReturnCode **rfalChipGetRFO** (uint8_t *result)
Get RFO. More...

ReturnCode **rfalChipGetLmFieldInd** (uint8_t *result)
Get LM Field Indicator. More...

ReturnCode **rfalChipSetLMMod** (uint8_t mod, uint8_t unmod)
Set Listen Mode Modulation. More...

ReturnCode **rfalChipGetLMMod** (uint8_t *mod, uint8_t *unmod)

Get Listen Mode Modulation. More...

ReturnCode **rfalChipMeasureAmplitude** (uint8_t *result)
Measure Amplitude. More...

ReturnCode **rfalChipMeasurePhase** (uint8_t *result)
Measure Phase. More...

ReturnCode **rfalChipMeasureCapacitance** (uint8_t *result)
Measure Capacitance. More...

ReturnCode **rfalChipMeasurePowerSupply** (uint8_t param, uint8_t *result)
Measure Power Supply. More...

ReturnCode **rfalChipMeasureIQ** (int8_t *resI, int8_t *resQ)
Measure I and Q. More...

ReturnCode **rfalChipMeasureCombinedIQ** (uint8_t *result)
Measure combined I and Q. More...

ReturnCode **rfalChipMeasureI** (uint8_t *result)
Measure I channel. More...

ReturnCode **rfalChipMeasureQ** (uint8_t *result)
Measure Q channel. More...

ReturnCode **rfalChipMeasureCurrent** (uint8_t *result)
Measure Current. More...

ReturnCode **rfalChipSetAntennaMode** (bool single, bool rfiox)
Set Antenna mode. More...

Detailed Description

Function Documentation

◆ rfalChipChangeRegBits()

```
ReturnCode rfalChipChangeRegBits ( uint16_t  reg,
                                   uint8_t   valueMask,
                                   uint8_t   value
                                 )
```

Change the value of the register bits on the RF Chip Test set in the valueMask.

Parameters

- [in] **reg** : register address to be modified
- [in] **valueMask** : mask value of the register bits to be changed
- [in] **value** : register value to be set

Returns

RFAL_ERR_PARAM : Invalid register or bad request

RFAL_ERR_NOTSUPP : Feature not supported

RFAL_ERR_OK : Change done with no error

◆ rfalChipChangeTestRegBits()

```
ReturnCode rfalChipChangeTestRegBits ( uint16_t  reg,
                                       uint8_t   valueMask,
                                       uint8_t   value
                                     )
```

Change the value of the register bits on the RF Chip Test set in the valueMask.

Parameters

- [in] **reg** : test register address to be modified
- [in] **valueMask** : mask value of the register bits to be changed
- [in] **value** : register value to be set

Returns

RFAL_ERR_PARAM : Invalid register or bad request

RFAL_ERR_NOTSUPP : Feature not supported

RFAL_ERR_OK : Change done with no error

◆ rfalChipExecCmd()

ReturnCode rfalChipExecCmd (uint16_t cmd)

Checks if the given command is valid and if so, executes it on the RF Chip

Parameters

[in] **cmd** : direct command to be executed

Returns

RFAL_ERR_PARAM : Invalid command or bad request

RFAL_ERR_NOTSUPP : Feature not supported

RFAL_ERR_NONE : Direct command executed with no error

◆ rfalChipGetLmFieldInd()

ReturnCode rfalChipGetLmFieldInd (uint8_t * result)

Gets an indicator of the signal on RFI while in Passive Listen Mode

Parameters

[out] **result** : the current RFI value

Returns

RFAL_ERR_IO : Internal error

RFAL_ERR_NOTSUPP : Feature not supported

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_NONE : No error

◆ rfalChipGetLMMod()

ReturnCode rfalChipGetLMMod (uint8_t * mod,
uint8_t * unmod
)

Gets the modulation (modulated and unmodulated state) when Passive Listen Mode is used

Parameters

[out] **mod** : modulation set in modulated state

[out] **unmod** : modulation set in unmodulated state

Returns

RFAL_ERR_IO : Internal error

RFAL_ERR_NOTSUPP : Feature not supported

RFAL_ERR_NONE : No error

◆ rfalChipGetRFO()

ReturnCode rfalChipGetRFO (uint8_t * result)

Gets the RFO driver resistance value used when the field is on (unmodulated/active)

Parameters

[out] **result** : the current RFO value

Returns

RFAL_ERR_IO : Internal error

RFAL_ERR_NOTSUPP : Feature not supported

RFAL_ERR_NONE : No error

◆ rfalChipMeasureAmplitude()

ReturnCode rfalChipMeasureAmplitude (uint8_t * result)

Measures the RF Amplitude

Parameters

[out] **result** : result of RF measurement

Returns

RFAL_ERR_IO : Internal error

RFAL_ERR_NOTSUPP : Feature not supported

RFAL_ERR_NONE : No error

◆ rfalChipMeasureCapacitance()

ReturnCode rfalChipMeasureCapacitance (uint8_t * result)

Measures the Capacitance

Parameters

[out] **result** : result of Capacitance measurement

Returns

RFAL_ERR_IO : Internal error

RFAL_ERR_NOTSUPP : Feature not supported

RFAL_ERR_NONE : No error

◆ rfalChipMeasureCombinedIQ()

ReturnCode rfalChipMeasureCombinedIQ (uint8_t * result)

Measures I and Q channels and combines them

Parameters

[out] **result** : I and Q combined

Returns

RFAL_ERR_IO : Internal error

RFAL_ERR_NOTSUPP : Feature not supported

RFAL_ERR_NONE : No error

◆ rfalChipMeasureCurrent()

ReturnCode rfalChipMeasureCurrent (uint8_t * result)

Measures the electrical current

Parameters

[out] **result** : current

Returns

RFAL_ERR_IO : Internal error

RFAL_ERR_NOTSUPP : Feature not supported

RFAL_ERR_NONE : No error

◆ rfalChipMeasureI()

ReturnCode rfalChipMeasureI (uint8_t * result)

Measures I channel

Parameters

[out] **result** : I channel

Returns

RFAL_ERR_IO : Internal error

RFAL_ERR_NOTSUPP : Feature not supported

RFAL_ERR_NONE : No error

◆ rfalChipMeasureIQ()

ReturnCode rfalChipMeasureIQ (int8_t * resI,
int8_t * resQ
)

Measures I and Q channels

Parameters

[out] **resI** : 8 bit long result of the I channel (signed)

[out] **resQ** : 8 bit long result of the Q channel (signed)

Returns

RFAL_ERR_IO : Internal error

RFAL_ERR_NOTSUPP : Feature not supported

RFAL_ERR_NONE : No error

◆ rfalChipMeasurePhase()

ReturnCode rfalChipMeasurePhase (uint8_t * result)

Measures the Phase

Parameters

[out] **result** : result of Phase measurement

Returns

RFAL_ERR_IO : Internal error

RFAL_ERR_NOTSUPP : Feature not supported

RFAL_ERR_NONE : No error

◆ rfalChipMeasurePowerSupply()

```
ReturnCode rfalChipMeasurePowerSupply ( uint8_t    param,
                                        uint8_t *  result
                                      )
```

Measures the Power Supply

Parameters

[in] **param** : measurement parameter (chip specific)

[out] **result** : result of the measurement

Returns

RFAL_ERR_IO : Internal error

RFAL_ERR_NOTSUPP : Feature not supported

RFAL_ERR_NONE : No error

◆ rfalChipMeasureQ()

```
ReturnCode rfalChipMeasureQ ( uint8_t *  result )
```

Measures Q channel

Parameters

[out] **result** : Q channel

Returns

RFAL_ERR_IO : Internal error

RFAL_ERR_NOTSUPP : Feature not supported

RFAL_ERR_NONE : No error

◆ rfalChipReadReg()

```
ReturnCode rfalChipReadReg ( uint16_t  reg,
                             uint8_t *  values,
                             uint8_t    len
                           )
```

Checks if the given register is valid and if so, reads the value(s) of the RF Chip register(s)

Parameters

- [in] **reg** : register address to be read, or the first if len > 1
- [out] **values** : pointer where the register(s) read content will be placed
- [in] **len** : number of consecutive registers to be read

Returns

RFAL_ERR_PARAM : Invalid register or bad request

RFAL_ERR_NOTSUPP : Feature not supported

RFAL_ERR_NONE : Read done with no error

◆ rfalChipReadTestReg()

```
ReturnCode rfalChipReadTestReg ( uint16_t  reg,
                                 uint8_t *  value
                               )
```

Reads the value of the RF Chip Test register

Parameters

- [in] **reg** : register address to be read
- [out] **value** : pointer where the register content will be placed

Returns

RFAL_ERR_PARAM :Invalid register or bad request

RFAL_ERR_NOTSUPP : Feature not supported

RFAL_ERR_NONE : Read done with no error

◆ rfalChipSetAntennaMode()

```
ReturnCode rfalChipSetAntennaMode ( bool  single,
```

```
                                        bool  rfiox
                                      )
```

Sets the antenna mode. Differential or single ended antenna mode (RFO1 or RFO2)

Parameters

[in] **single** : FALSE differential ; single ended mode

[in] **rfiox** : FALSE RFI1/RFO1 ; TRUE RFI2/RFO2

Returns

RFAL_ERR_IO : Internal error

RFAL_ERR_NOTSUPP : Feature not supported

RFAL_ERR_NONE : No error

◆ rfalChipSetLMMod()

```
ReturnCode rfalChipSetLMMod ( uint8_t  mod,
                              uint8_t  unmod
                            )
```

Sets the modulation (modulated and unmodulated state) when Passive Listen Mode is used

Parameters

[in] **mod** : modulation to be used in modulated state

[in] **unmod** : modulation to be used in unmodulated state

Returns

RFAL_ERR_IO : Internal error

RFAL_ERR_NOTSUPP : Feature not supported

RFAL_ERR_NONE : No error

◆ rfalChipSetRFO()

```
ReturnCode rfalChipSetRFO ( uint8_t  rfo )
```

Sets the RFO driver resistance value used when the field is on (unmodulated/active)

Parameters

[in] **rfo** : the RFO value to be used

Returns

RFAL_ERR_IO : Internal error

RFAL_ERR_NOTSUPP : Feature not supported

RFAL_ERR_NONE : No error

◆ rfalChipWriteReg()

```
ReturnCode rfalChipWriteReg ( uint16_t        reg,
                              const uint8_t * values,
                              uint8_t         len
                            )
```

Checks if the given register is valid and if so, writes the value(s) on the RF Chip register

Parameters

[in] **reg** : register address to be written, or the first if len > 1

[in] **values** : pointer with content to be written on the register(s)

[in] **len** : number of consecutive registers to be written

Returns

RFAL_ERR_PARAM : Invalid register or bad request

RFAL_ERR_NOTSUPP : Feature not supported

RFAL_ERR_NONE : Write done with no error

◆ rfalChipWriteTestReg()

```
ReturnCode rfalChipWriteTestReg ( uint16_t reg,
                                  uint8_t  value
                                )
```

Writes the value on the RF Chip Test register

Parameters

[in] **reg** : register address to be written

[in] **value** : value to be written on the register

Returns

RFAL_ERR_PARAM : Invalid register or bad request

RFAL_ERR_NOTSUPP : Feature not supported

RFAL_ERR_NONE : Write done with no error

DLMA

RFAL » RFAL-HAL

RFAL Dynamic Load modulation. More...

Collaboration diagram for DLMA:

Data Structures

struct **rfalDlmaEntry**

struct **rfalDlmaInfo**

Macros

#define **RFAL_DLMA_TABLE_MAX_ENTRIES** 4U

#define **RFAL_DLMA_TABLE_PARAM_LEN** sizeof(**rfalDlmaEntry**)

#define **RFAL_DLMA_TABLE_SIZE_MAX** (**RFAL_DLMA_TABLE_MAX_ENTRIES** * **RFAL_DLMA_TABLE_PARAM_LEN**)

Typedefs

typedef **ReturnCode**(* **rfalDlmaMeasureFunc**) (uint8_t *res)

typedef **ReturnCode**(* **rfalDlmaAdjustFunc**) (uint8_t mod, uint8_t unmod)

Functions

void **rfalDlmaInitialize** (void)
Initialize dynamic power table. More...

void **rfalDlmaSetMeasureCallback** (**rfalDlmaMeasureFunc** pFunc)
Set the measurement methode. More...

void **rfalDlmaSetAdjustCallback** (**rfalDlmaAdjustFunc** pFunc)
Set the measurement methode. More...

ReturnCode **rfalDlmaTableWrite** (const **rfalDlmaEntry** *powerTbl, uint8_t powerTblEntries)
Dynamic LMA table write. More...

ReturnCode **rfalDlmaTableRead** (**rfalDlmaEntry** *tblBuf, uint8_t tblBufEntries, uint8_t *tableEntries)
Dynamic LMA table Read. More...

ReturnCode **rfalDlmaAdjust** (void)
Dynamic LMA adjust. More...

void **rfalDlmaSetEnabled** (bool enable)
Dynamic LMA set enabled state. More...

bool **rfalDlmaIsEnabled** (void)
Get the Dynamic LMA enabled state. More...

ReturnCode **rfalDlmaGetInfo** (**rfalDlmaInfo** *info)
Get DLMA information. More...

Detailed Description

Macro Definition Documentation

◆ RFAL_DLMA_TABLE_MAX_ENTRIES

```
#define RFAL_DLMA_TABLE_MAX_ENTRIES   4U
```

Max DLMA entries

◆ RFAL_DLMA_TABLE_PARAM_LEN

```
#define RFAL_DLMA_TABLE_PARAM_LEN   sizeof(rfalDlmaEntry)
```

DLMA Parameter length

◆ RFAL_DLMA_TABLE_SIZE_MAX

```
#define RFAL_DLMA_TABLE_SIZE_MAX   (RFAL_DLMA_TABLE_MAX_ENTRIES *
RFAL_DLMA_TABLE_PARAM_LEN)
```

Max DLMA table size

Typedef Documentation

◆ rfalDlmaAdjustFunc

```
typedef ReturnCode(* rfalDlmaAdjustFunc) (uint8_t mod, uint8_t unmod)
```

Function pointer to the Adjustment method

◆ rfalDlmaMeasureFunc

typedef **ReturnCode**(* rfalDlmaMeasureFunc) (uint8_t *res)

Function pointer to methode doing the reference measurement

Function Documentation

◆ rfalDlmaAdjust()

ReturnCode rfalDlmaAdjust (void)

It measures the current output and adjusts the power accordingly to the dynamic power table

Returns

RFAL_ERR_NONE : No error

RFAL_ERR_PARAM : If configTbl is invalid or parameters are invalid

RFAL_ERR_WRONG_STATE : If the current state is valid for DLMA Adjustment

◆ rfalDlmaGetInfo()

ReturnCode rfalDlmaGetInfo (**rfalDlmaInfo** * info)

Get the DLMA information|status

Parameters

`[out]` **info** : DLMA information

Returns

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_NONE : No Error

◆ rfalDlmaInitialize()

void rfalDlmaInitialize (void)

This function sets the internal dynamic power table to the default values stored in rfal_DlmaTbl.h

◆ rfalDlmaIsEnabled()

bool rfalDlmaIsEnabled (void)

Get state of the Dynamic power adjustment

Returns

true : DLMA is enabled

false : DLMA is disabled

◆ rfalDlmaSetAdjustCallback()

void rfalDlmaSetAdjustCallback (**rfalDlmaAdjustFunc** pFunc)

This function sets the adjust method.

Parameters

[in] **pFunc** callback of adjust function

◆ rfalDlmaSetEnabled()

void rfalDlmaSetEnabled (bool enable)

Parameters

[in] **enable** : new active state

Set state to enable or disable the Dynamic LMA adjustment

◆ rfalDlmaSetMeasureCallback()

void rfalDlmaSetMeasureCallback (**rfalDlmaMeasureFunc** pFunc)

This function sets the measurement method used for reference measurement. Based on the measurement the power will then be adjusted

Parameters

[in] **pFunc** callback of measurement function

◆ rfalDlmaTableRead()

```
ReturnCode rfalDlmaTableRead ( rfalDlmaEntry *  tblBuf,
                               uint8_t          tblBufEntries,
                               uint8_t *        tableEntries
                             )
```

Read the dynamic power table

Parameters

- [out] **tblBuf** : location to the **rfalDlmaEntry**[] to place the Table
- [in] **tblBufEntries** : number of entries available in tblBuf to place the power Table
- [out] **tableEntries** : returned number of entries actually written into tblBuf

Returns

RFAL_ERR_NONE : No error

RFAL_ERR_WRONG_STATE : If configTbl is invalid

RFAL_ERR_PARAM : If parameters are invalid

◆ rfalDlmaTableWrite()

```
ReturnCode rfalDlmaTableWrite ( const rfalDlmaEntry *  powerTbl,
                                uint8_t                powerTblEntries
                              )
```

Load the dynamic power table

Parameters

- [in] **powerTbl** : location of power Table to be loaded
- [in] **powerTblEntries** : number of entries of the power Table to be loaded

Returns

RFAL_ERR_NONE : No error

RFAL_ERR_PARAM : If configTbl is invalid

RFAL_ERR_NOMEM : If the given Table is bigger exceeds the max size

RFAL

Main Page

Modules

Files

Examples

Data Structures

Data Structure Index

Data Fields

Data Fields

rfalDlmaEntry Struct Reference

RFAL » RFAL-HAL » DLMA

```
#include <rfal_dlma.h>
```

Data Fields

uint8_t	**unmodRes**
uint8_t	**modRes**
uint8_t	**inc**
uint8_t	**dec**

Detailed Description

DLMA table entry struct

Field Documentation

◆ dec

uint8_t rfalDlmaEntry::dec

Threshold for decrementing

◆ inc

uint8_t rfalDlmaEntry::inc

Threshold for incrementing

◆ modRes

uint8_t rfalDlmaEntry::modRes

RFO resistance in Passive Listen Mode: Modulated state

◆ unmodRes

uint8_t rfalDlmaEntry::unmodRes

RFO resistance in Passive Listen Mode: Unmodulated state

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

- **rfal_dlma.h**

rfalDlmaInfo Struct Reference


```
#include <rfal_dlma.h>
```

Data Fields

bool	**enabled**
uint8_t	**tableEntry**
uint8_t	**tableEntries**
uint8_t	**refMeasurement**
rfalDlmaAdjustFunc	**adjustCallback**
rfalDlmaMeasureFunc	**measureCallback**

Detailed Description

DLMA information struct

Field Documentation

◆ adjustCallback

rfalDlmaAdjustFunc rfalDlmaInfo::adjustCallback

Pointer to the adjust callback

◆ enabled

bool rfalDlmaInfo::enabled

Enabled state

◆ measureCallback

rfalDlmaMeasureFunc rfalDlmaInfo::measureCallback

Pointer to the measure callback

◆ refMeasurement

uint8_t rfalDlmaInfo::refMeasurement

Last measurement used to adjust

◆ tableEntries

uint8_t rfalDlmaInfo::tableEntries

Number of entries used

◆ tableEntry

uint8_t rfalDlmaInfo::tableEntry

Current entry used

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

- **rfal_dlma.h**

DPO

RFAL » **RFAL-HAL**

RFAL Dynamic Power Module. More...

Collaboration diagram for DPO:

Data Structures

struct	**rfalDpoEntry**
struct	**rfalDpoInfo**

Macros

#define **RFAL_DPO_TABLE_MAX_ENTRIES** 4U

#define **RFAL_DPO_TABLE_PARAM_LEN** sizeof(**rfalDpoEntry**)

#define **RFAL_DPO_TABLE_SIZE_MAX** (**RFAL_DPO_TABLE_MAX_ENTRIES** * **RFAL_DPO_TABLE_PARAM_LEN**)

Typedefs

typedef **ReturnCode**(* **rfalDpoMeasureFunc**) (uint8_t *res)

typedef **ReturnCode**(* **rfalDpoAdjustFunc**) (uint8_t res)

Functions

void **rfalDpoInitialize** (void)
Initialize dynamic power table. More...

void **rfalDpoSetMeasureCallback** (**rfalDpoMeasureFunc** pFunc)
Set the Measurement method. More...

void **rfalDpoSetAdjustCallback** (**rfalDpoAdjustFunc** pFunc)
Set the Adjust method. More...

ReturnCode **rfalDpoTableWrite** (const **rfalDpoEntry** *powerTbl, uint8_t powerTblEntries)
Write dynamic power table. More...

ReturnCode **rfalDpoTableRead** (**rfalDpoEntry** *tblBuf, uint8_t tblBufEntries, uint8_t *tableEntries)
Dynamic power table Read. More...

ReturnCode **rfalDpoAdjust** (void)
Dynamic power adjust. More...

void **rfalDpoSetEnabled** (bool enable)
Dynamic power set enabled state. More...

bool **rfalDpoIsEnabled** (void)
Get the Dynamic power enabled state. More...

void **rfalDpoReqAdj** (void)

	Request DPO adjust. More...
ReturnCode	**rfalDpoGetInfo** (**rfalDpoInfo** *info) Get DPO information. More...

Detailed Description

Macro Definition Documentation

◆ RFAL_DPO_TABLE_MAX_ENTRIES

#define RFAL_DPO_TABLE_MAX_ENTRIES 4U

Max DPO table entries

◆ RFAL_DPO_TABLE_PARAM_LEN

#define RFAL_DPO_TABLE_PARAM_LEN sizeof(**rfalDpoEntry**)

DPO Parameter length

◆ RFAL_DPO_TABLE_SIZE_MAX

#define RFAL_DPO_TABLE_SIZE_MAX (**RFAL_DPO_TABLE_MAX_ENTRIES** * **RFAL_DPO_TABLE_PARAM_LEN**)

Max DPO table size

Typedef Documentation

◆ rfalDpoAdjustFunc

typedef **ReturnCode**(* rfalDpoAdjustFunc) (uint8_t res)

Function pointer to the Adjustment method

◆ rfalDpoMeasureFunc

typedef **ReturnCode**(* rfalDpoMeasureFunc) (uint8_t *res)

Function pointer to the reference Measurement method

Function Documentation

◆ rfalDpoAdjust()

ReturnCode rfalDpoAdjust (void)

It measures the current output and adjusts the power accordingly to the dynamic power table. This method | The adjustment shall be performed when the device is already emiting RF field

Returns

RFAL_ERR_NONE : No error

RFAL_ERR_PARAM : if configTbl is invalid or parameters are invalid

RFAL_ERR_WRONG_STATE : if the current state is valid for DPO Adjustment

◆ rfalDpoGetInfo()

ReturnCode rfalDpoGetInfo (**rfalDpoInfo** * info)

Get the DPO information|status

Parameters

`[out]` **info** : pointer where DPO info is to be stored

Returns

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_NONE : No Error

◆ rfalDpoInitialize()

void rfalDpoInitialize (void)

This function sets the internal dynamic power table to the default values stored in rfal_DpoTbl.h

◆ rfalDpoIsEnabled()

bool rfalDpoIsEnabled (void)

Get state of the Dynamic power adjustment

Returns

true : DPO is enabled

false : DPO is disabled

◆ rfalDpoReqAdj()

void rfalDpoReqAdj (void)

Flags the DPO module to perform the adjustment on the next round, even if no change has been perceived from previous measurement|state

◆ rfalDpoSetAdjustCallback()

void rfalDpoSetAdjustCallback (**rfalDpoAdjustFunc** pFunc)

This function sets the Adjust method used during DPO adjust

Parameters

[in] **pFunc** callback of adjust function

◆ rfalDpoSetEnabled()

void rfalDpoSetEnabled (bool enable)

Parameters

[in] **enable** : new active state

Set state to enable or disable the Dynamic power adjustment

◆ rfalDpoSetMeasureCallback()

```
void rfalDpoSetMeasureCallback ( rfalDpoMeasureFunc  pFunc )
```

This function sets the measurement method used for reference measurement. Based on the measurement the power will then be adjusted

Parameters

[in] **pFunc** callback of measurement function

◆ rfalDpoTableRead()

```
ReturnCode rfalDpoTableRead ( rfalDpoEntry *  tblBuf,
                              uint8_t         tblBufEntries,
                              uint8_t *       tableEntries
                            )
```

Read the dynamic power table

Parameters

[out] **tblBuf** : location to the **rfalDpoEntry**[] to place the Table

[in] **tblBufEntries** : number of entries available in tblBuf to place the power Table

[out] **tableEntries** : returned number of entries actually written into tblBuf

Returns

RFAL_ERR_NONE : No error

RFAL_ERR_PARAM : if configTbl is invalid or parameters are invalid

◆ rfalDpoTableWrite()

```
ReturnCode rfalDpoTableWrite ( const rfalDpoEntry *  powerTbl,
                               uint8_t               powerTblEntries
                             )
```

Load the dynamic power table

Parameters

[in] **powerTbl** : location of power Table to be loaded

[in] **powerTblEntries** : number of entries of the power Table to be loaded

Returns

RFAL_ERR_NONE : No error

RFAL_ERR_PARAM : if configTbl is invalid

RFAL_ERR_NOMEM : if the given Table is bigger exceeds the max size

rfalDpoEntry Struct Reference


```
#include <rfal_dpo.h>
```

Data Fields

uint8_t	**rfoRes**
uint8_t	**inc**
uint8_t	**dec**

Detailed Description

DPO table entry struct

Field Documentation

◆ dec

uint8_t rfalDpoEntry::dec

Threshold for decrementing the output power

◆ inc

uint8_t rfalDpoEntry::inc

Threshold for incrementing the output power

◆ rfoRes

uint8_t rfalDpoEntry::rfoRes

Setting for the resistance level of the RFO

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

- **rfal_dpo.h**

rfalDpoInfo Struct Reference


```
#include <rfal_dpo.h>
```

Data Fields

bool	**enabled**
uint8_t	**tableEntries**
uint8_t	**tableEntry**
uint8_t	**refMeasurement**
rfalDpoAdjustFunc	**adjustCallback**
rfalDpoMeasureFunc	**measureCallback**

Detailed Description

DPO information struct

Field Documentation

◆ adjustCallback

rfalDpoAdjustFunc rfalDpoInfo::adjustCallback

Pointer to the adjust callback

◆ enabled

bool rfalDpoInfo::enabled

Enabled state

◆ measureCallback

rfalDpoMeasureFunc rfalDpoInfo::measureCallback

Pointer to the measure callback

◆ refMeasurement

uint8_t rfalDpoInfo::refMeasurement

Last measurement used to adjust

◆ tableEntries

uint8_t rfalDpoInfo::tableEntries

Number of entries used

◆ tableEntry

uint8_t rfalDpoInfo::tableEntry

Current entry used

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

- **rfal_dpo.h**

RFAL

RF

RFAL » **RFAL-HAL**

RFAL RF Abstraction Layer. More...

Collaboration diagram for RF:

Data Structures

struct	**rfalTransceiveContext**
struct	**rfalLmConfPA**
struct	**rfalLmConfPB**
struct	**rfalLmConfPF**

Macros

#define **RFAL_VERSION** 0x040000U

#define **RFAL_FWT_NONE** 0xFFFFFFFFU

#define **RFAL_GT_NONE** **RFAL_TIMING_NONE**

#define **RFAL_TIMING_NONE** 0x00U

#define **RFAL_1FC_IN_4096FC** (uint32_t)4096U

#define **RFAL_1FC_IN_2048FC** (uint32_t)2048U

#define **RFAL_1FC_IN_512FC** (uint32_t)512U

#define **RFAL_1FC_IN_64FC** (uint32_t)64U

#define **RFAL_1FC_IN_8FC** (uint32_t)8U

#define **RFAL_US_IN_MS** (uint32_t)1000U

#define **RFAL_1MS_IN_1FC** (uint32_t)13560U

#define **RFAL_BITS_IN_BYTE** (uint16_t)8U

#define **RFAL_CRC_LEN** 2U

#define **RFAL_TXRX_FLAGS_DEFAULT** ((uint32_t)**RFAL_TXRX_FLAGS_CRC_TX_AUTO** | (uint32_t)**RFAL_TXRX_FLAGS_CRC_RX_REMV** | (uint32_t)**RFAL_TXRX_FLAGS_NFCIP1_OFF** | (uint32_t)**RFAL_TXRX_FLAGS_AGC_ON** | (uint32_t)**RFAL_TXRX_FLAGS_PAR_RX_REMV** | (uint32_t)**RFAL_TXRX_FLAGS_PAR_TX_AUTO** | (uint32_t)**RFAL_TXRX_FLAGS_NFCV_FLAG_AUTO**)

#define **RFAL_LM_MASK_NFCA** ((uint32_t)1U<<(uint8_t)**RFAL_MODE_LISTEN_NFCA**)

#define **RFAL_LM_MASK_NFCB** ((uint32_t)1U<<(uint8_t)**RFAL_MODE_LISTEN_NFCB**)

#define **RFAL_LM_MASK_NFCF** ((uint32_t)1U<<(uint8_t)**RFAL_MODE_LISTEN_NFCF**)

#define **RFAL_LM_MASK_ACTIVE_P2P** ((uint32_t)1U<<(uint8_t)**RFAL_MODE_LISTEN_ACTIVE_P2P**)

```
#define RFAL_LM_SENS_RES_LEN 2U
```

```
#define RFAL_LM_SENSB_RES_LEN 13U
```

```
#define RFAL_LM_SENSF_RES_LEN 19U
```

```
#define RFAL_LM_SENSF_SC_LEN 2U
```

```
#define RFAL_NFCID3_LEN 10U
```

```
#define RFAL_NFCID2_LEN 8U
```

```
#define RFAL_NFCID1_TRIPLE_LEN 10U
```

```
#define RFAL_NFCID1_DOUBLE_LEN 7U
```

```
#define RFAL_NFCID1_SINGLE_LEN 4U
```

```
#define rfalGetMaxBrRW() ( ((RFAL_SUPPORT_BR_RW_6780) ? RFAL_BR_6780 : ((RFAL_SUPPORT_BR_RW_3390) ? RFAL_BR_3390 : ((RFAL_SUPPORT_BR_RW_1695) ? RFAL_BR_1695 : ((RFAL_SUPPORT_BR_RW_848) ? RFAL_BR_848 : ((RFAL_SUPPORT_BR_RW_424) ? RFAL_BR_424 : ((RFAL_SUPPORT_BR_RW_212) ? RFAL_BR_212 : RFAL_BR_106 ) ) ) ) ) ) ) )
```

```
#define rfalGetMaxBrAP2P() ( ((RFAL_SUPPORT_BR_AP2P_848) ? RFAL_BR_848 : ((RFAL_SUPPORT_BR_AP2P_424) ? RFAL_BR_424 : ((RFAL_SUPPORT_BR_AP2P_212) ? RFAL_BR_212 : RFAL_BR_106 ) ) ) ) )
```

```
#define rfalGetMaxBrCEA() ( ((RFAL_SUPPORT_BR_CE_A_848) ? RFAL_BR_848 : ((RFAL_SUPPORT_BR_CE_A_424) ? RFAL_BR_424 : ((RFAL_SUPPORT_BR_CE_A_212) ? RFAL_BR_212 : RFAL_BR_106 ) ) ) ) )
```

```
#define rfalGetMaxBrCEB() ( ((RFAL_SUPPORT_BR_CE_B_848) ? RFAL_BR_848 : ((RFAL_SUPPORT_BR_CE_B_424) ? RFAL_BR_424 : ((RFAL_SUPPORT_BR_CE_B_212) ? RFAL_BR_212 : RFAL_BR_106 ) ) ) ) )
```

```
#define rfalGetMaxBrCEF() ( ((RFAL_SUPPORT_BR_CE_F_424) ? RFAL_BR_424 : RFAL_BR_212 ) )
```

```
#define rfalIsModeActiveComm(md) ( ((md) == RFAL_MODE_POLL_ACTIVE_P2P) || ((md) == RFAL_MODE_LISTEN_ACTIVE_P2P) )
```

```
#define rfalIsModePassiveComm(md) ( !rfalIsModeActiveComm(md) )
```

```
#define rfalIsModePassiveListen(md)   ( ((md) == RFAL_MODE_LISTEN_NFCA) || ((md) ==
        RFAL_MODE_LISTEN_NFCB) || ((md) == RFAL_MODE_LISTEN_NFCF) )

#define rfalIsModePassivePoll(md)   ( rfalIsModePassiveComm(md) && (!rfalIsModePassiveListen(md)) )

#define rfalConv1fcTo8fc(t)   (uint32_t)( (uint32_t)(t) / RFAL_1FC_IN_8FC )

#define rfalConv8fcTo1fc(t)   (uint32_t)( (uint32_t)(t) * RFAL_1FC_IN_8FC )

#define rfalConv1fcTo64fc(t)   (uint32_t)( (uint32_t)(t) / RFAL_1FC_IN_64FC )

#define rfalConv64fcTo1fc(t)   (uint32_t)( (uint32_t)(t) * RFAL_1FC_IN_64FC )

#define rfalConv1fcTo512fc(t)   (uint32_t)( (uint32_t)(t) / RFAL_1FC_IN_512FC )

#define rfalConv512fcTo1fc(t)   (uint32_t)( (uint32_t)(t) * RFAL_1FC_IN_512FC )

#define rfalConv1fcTo2018fc(t)   (uint32_t)( (uint32_t)(t) / RFAL_1FC_IN_2048FC )

#define rfalConv2048fcTo1fc(t)   (uint32_t)( (uint32_t)(t) * RFAL_1FC_IN_2048FC )

#define rfalConv1fcTo4096fc(t)   (uint32_t)( (uint32_t)(t) / RFAL_1FC_IN_4096FC )

#define rfalConv4096fcTo1fc(t)   (uint32_t)( (uint32_t)(t) * RFAL_1FC_IN_4096FC )

#define rfalConv1fcToMs(t)   (uint32_t)( (uint32_t)(t) / RFAL_1MS_IN_1FC )

#define rfalConvMsTo1fc(t)   (uint32_t)( (uint32_t)(t) * RFAL_1MS_IN_1FC )

#define rfalConv1fcToUs(t)   (uint32_t)( ((uint32_t)(t) * RFAL_US_IN_MS) / RFAL_1MS_IN_1FC)

#define rfalConvUsTo1fc(t)   (uint32_t)( ((uint32_t)(t) * RFAL_1MS_IN_1FC) / RFAL_US_IN_MS)

#define rfalConv64fcToMs(t)   (uint32_t)( (uint32_t)(t) / (RFAL_1MS_IN_1FC / RFAL_1FC_IN_64FC) )

#define rfalConvMsTo64fc(t)   (uint32_t)( (uint32_t)(t) * (RFAL_1MS_IN_1FC / RFAL_1FC_IN_64FC) )

#define rfalConvBitsToBytes(n)   (uint16_t)( ((uint16_t)(n)+(RFAL_BITS_IN_BYTE-1U)) / (RFAL_BITS_IN_BYTE) )

#define rfalConvBytesToBits(n)   (uint32_t)( (uint32_t)(n) * (RFAL_BITS_IN_BYTE) )

#define rfalRunBlocking(e, fn)   do{ (e)=(fn); rfalWorker(); }while( (e) == RFAL_ERR_BUSY )
```

#define **rfalCreateByteTxRxContext**(ctx, tB, tBL, rB, rBL, rdL, t)

#define **rfalCreateByteFlagsTxRxContext**(ctx, tB, tBL, rB, rBL, rdL, fl, t)

#define **rfalLogE**(...) **platformLog**(__VA_ARGS__)

#define **rfalLogW**(...) **platformLog**(__VA_ARGS__)

#define **rfalLogI**(...) **platformLog**(__VA_ARGS__)

#define **rfalLogD**(...) **platformLog**(__VA_ARGS__)

#define **RFAL_GT_NFCA** **rfalConvMsTo1fc**(5U)

#define **RFAL_GT_NFCB** **rfalConvMsTo1fc**(5U)

#define **RFAL_GT_NFCF** **rfalConvMsTo1fc**(20U)

#define **RFAL_GT_NFCV** **rfalConvMsTo1fc**(5U)

#define **RFAL_GT_PICOPASS** **rfalConvMsTo1fc**(1U)

#define **RFAL_GT_AP2P** **rfalConvMsTo1fc**(5U)

#define **RFAL_GT_AP2P_ADJUSTED** **rfalConvMsTo1fc**(5U+25U)

#define **RFAL_FDT_LISTEN_NFCA_POLLER** 1172U

#define **RFAL_FDT_LISTEN_NFCB_POLLER** 1008U

#define **RFAL_FDT_LISTEN_NFCF_POLLER** 2672U

#define **RFAL_FDT_LISTEN_NFCV_POLLER** 4310U

#define **RFAL_FDT_LISTEN_PICOPASS_POLLER** 3400U

#define **RFAL_FDT_LISTEN_AP2P_POLLER** 64U

#define **RFAL_FDT_LISTEN_NFCA_LISTENER** 1172U

#define **RFAL_FDT_LISTEN_NFCB_LISTENER** 1024U

#define **RFAL_FDT_LISTEN_NFCF_LISTENER** 2688U

#define **RFAL_FDT_LISTEN_AP2P_LISTENER** 64U

#define **RFAL_FDT_POLL_NFCA_POLLER** 6780U

#define **RFAL_FDT_POLL_NFCA_T1T_POLLER** 384U

#define **RFAL_FDT_POLL_NFCB_POLLER** 6780U

#define **RFAL_FDT_POLL_NFCF_POLLER** 6800U

#define **RFAL_FDT_POLL_NFCV_POLLER** 4192U

#define **RFAL_FDT_POLL_PICOPASS_POLLER** 1790U

#define **RFAL_FDT_POLL_AP2P_POLLER** 6800U

#define **RFAL_FELICA_LEN_LEN** 1U

#define **RFAL_FELICA_POLL_REQ_LEN** (**RFAL_FELICA_LEN_LEN** + 1U + 2U + 1U + 1U)

#define **RFAL_FELICA_POLL_RES_LEN** (**RFAL_FELICA_LEN_LEN** + 1U + 8U + 8U + 2U)

#define **RFAL_FELICA_POLL_MAX_SLOTS** 16U

#define **RFAL_WUM_REFERENCE_AUTO** 0xFFU

Typedefs

typedef void(* **rfalUpperLayerCallback**) (void)

typedef void(* **rfalPreTxRxCallback**) (void)

typedef void(* **rfalPostTxRxCallback**) (void)

typedef bool(* **rfalSyncTxRxCallback**) (void)

typedef void(* **rfalLmEonCallback**) (void)

typedef uint8_t **rfalFeliCaPollRes**[**RFAL_FELICA_POLL_RES_LEN**]

Enumerations

```
enum rfalMode {
  RFAL_MODE_NONE = 0 , RFAL_MODE_POLL_NFCA = 1 , RFAL_MODE_POLL_NFCA_T1T = 2 ,
RFAL_MODE_POLL_NFCB = 3 ,
  RFAL_MODE_POLL_B_PRIME = 4 , RFAL_MODE_POLL_B_CTS = 5 , RFAL_MODE_POLL_NFCF = 6 ,
RFAL_MODE_POLL_NFCV = 7 ,
  RFAL_MODE_POLL_PICOPASS = 8 , RFAL_MODE_POLL_ACTIVE_P2P = 9 , RFAL_MODE_LISTEN_NFCA =
10 , RFAL_MODE_LISTEN_NFCB = 11 ,
  RFAL_MODE_LISTEN_NFCF = 12 , RFAL_MODE_LISTEN_ACTIVE_P2P = 13
}
```

```
enum rfalBitRate {
  RFAL_BR_106 = 0 , RFAL_BR_212 = 1 , RFAL_BR_424 = 2 , RFAL_BR_848 = 3 ,
  RFAL_BR_1695 = 4 , RFAL_BR_3390 = 5 , RFAL_BR_6780 = 6 , RFAL_BR_13560 = 7 ,
  RFAL_BR_211p88 = 0xE9 , RFAL_BR_105p94 = 0xEA , RFAL_BR_52p97 = 0xEB , RFAL_BR_26p48 = 0xEC ,
  RFAL_BR_1p66 = 0xED , RFAL_BR_KEEP = 0xFF
}
```

```
enum rfalComplianceMode { RFAL_COMPLIANCE_MODE_NFC , RFAL_COMPLIANCE_MODE_EMV ,
RFAL_COMPLIANCE_MODE_ISO }
```

```
enum rfalState {
  RFAL_STATE_IDLE = 0 , RFAL_STATE_INIT = 1 , RFAL_STATE_MODE_SET = 2 , RFAL_STATE_TXRX = 3 ,
  RFAL_STATE_LM = 4 , RFAL_STATE_WUM = 5
}
```

```
enum rfalTransceiveState {
  RFAL_TXRX_STATE_IDLE = 0 , RFAL_TXRX_STATE_INIT = 1 , RFAL_TXRX_STATE_START = 2 ,
RFAL_TXRX_STATE_TX_IDLE = 11 ,
  RFAL_TXRX_STATE_TX_WAIT_GT = 12 , RFAL_TXRX_STATE_TX_WAIT_FDT = 13 ,
RFAL_TXRX_STATE_TX_PREP_TX = 14 , RFAL_TXRX_STATE_TX_TRANSMIT = 15 ,
  RFAL_TXRX_STATE_TX_WAIT_WL = 16 , RFAL_TXRX_STATE_TX_RELOAD_FIFO = 17 ,
RFAL_TXRX_STATE_TX_WAIT_TXE = 18 , RFAL_TXRX_STATE_TX_DONE = 19 ,
  RFAL_TXRX_STATE_TX_FAIL = 20 , RFAL_TXRX_STATE_RX_IDLE = 81 ,
RFAL_TXRX_STATE_RX_WAIT_EON = 82 , RFAL_TXRX_STATE_RX_WAIT_RXS = 83 ,
  RFAL_TXRX_STATE_RX_WAIT_RXE = 84 , RFAL_TXRX_STATE_RX_READ_FIFO = 85 ,
RFAL_TXRX_STATE_RX_ERR_CHECK = 86 , RFAL_TXRX_STATE_RX_READ_DATA = 87 ,
  RFAL_TXRX_STATE_RX_WAIT_EOF = 88 , RFAL_TXRX_STATE_RX_DONE = 89 ,
```

RFAL_TXRX_STATE_RX_FAIL = 90
}

enum {
RFAL_TXRX_FLAGS_CRC_TX_AUTO = (0U<<0) , **RFAL_TXRX_FLAGS_CRC_TX_MANUAL** = (1U<<0) , **RFAL_TXRX_FLAGS_CRC_RX_KEEP** = (1U<<1) , **RFAL_TXRX_FLAGS_CRC_RX_REMV** = (0U<<1) , **RFAL_TXRX_FLAGS_NFCIP1_ON** = (1U<<2) , **RFAL_TXRX_FLAGS_NFCIP1_OFF** = (0U<<2) , **RFAL_TXRX_FLAGS_AGC_OFF** = (1U<<3) , **RFAL_TXRX_FLAGS_AGC_ON** = (0U<<3) , **RFAL_TXRX_FLAGS_PAR_RX_KEEP** = (1U<<4) , **RFAL_TXRX_FLAGS_PAR_RX_REMV** = (0U<<4) , **RFAL_TXRX_FLAGS_PAR_TX_NONE** = (1U<<5) , **RFAL_TXRX_FLAGS_PAR_TX_AUTO** = (0U<<5) , **RFAL_TXRX_FLAGS_NFCV_FLAG_MANUAL** = (1U<<6) , **RFAL_TXRX_FLAGS_NFCV_FLAG_AUTO** = (0U<<6) , **RFAL_TXRX_FLAGS_CRC_RX_MANUAL** = (1U<<7) , **RFAL_TXRX_FLAGS_CRC_RX_AUTO** = (0U<<7)
}

enum **rfalEHandling** { **RFAL_ERRORHANDLING_NONE** = 0 , **RFAL_ERRORHANDLING_EMD** = 1 }

enum **rfal14443AShortFrameCmd** { **RFAL_14443A_SHORTFRAME_CMD_WUPA** = 0x52 , **RFAL_14443A_SHORTFRAME_CMD_REQA** = 0x26 }

enum { **RFAL_FELICA_POLL_RC_NO_REQUEST** = 0x00U , **RFAL_FELICA_POLL_RC_SYSTEM_CODE** = 0x01U , **RFAL_FELICA_POLL_RC_COM_PERFORMANCE** = 0x02U }

enum **rfalFeliCaPollSlots** {
RFAL_FELICA_1_SLOT = 0 , **RFAL_FELICA_2_SLOTS** = 1 , **RFAL_FELICA_4_SLOTS** = 3 , **RFAL_FELICA_8_SLOTS** = 7 ,
RFAL_FELICA_16_SLOTS = 15
}

enum **rfalLmNfcidLen** { **RFAL_LM_NFCID_LEN_04** = RFAL_NFCID1_SINGLE_LEN , **RFAL_LM_NFCID_LEN_07** = RFAL_NFCID1_DOUBLE_LEN , **RFAL_LM_NFCID_LEN_10** = RFAL_NFCID1_TRIPLE_LEN }

enum **rfalLmState** {
RFAL_LM_STATE_NOT_INIT = 0x00 , **RFAL_LM_STATE_POWER_OFF** = 0x01 , **RFAL_LM_STATE_IDLE** = 0x02 , **RFAL_LM_STATE_READY_A** = 0x03 ,
RFAL_LM_STATE_READY_B = 0x04 , **RFAL_LM_STATE_READY_F** = 0x05 , **RFAL_LM_STATE_ACTIVE_A** = 0x06 , **RFAL_LM_STATE_CARDEMU_4A** = 0x07 ,
RFAL_LM_STATE_CARDEMU_4B = 0x08 , **RFAL_LM_STATE_CARDEMU_3** = 0x09 , **RFAL_LM_STATE_TARGET_A** = 0x0A , **RFAL_LM_STATE_TARGET_F** = 0x0B ,
RFAL_LM_STATE_SLEEP_A = 0x0C , **RFAL_LM_STATE_SLEEP_B** = 0x0D , **RFAL_LM_STATE_READY_Ax** = 0x0E , **RFAL_LM_STATE_ACTIVE_Ax** = 0x0F ,

RFAL_LM_STATE_SLEEP_AF = 0x10
}

enum **rfalLpMode** { **RFAL_LP_MODE_PD** = 0 , **RFAL_LP_MODE_HR** = 1 }

enum **rfalWumState** { **RFAL_WUM_STATE_NOT_INIT** = 0x00 , **RFAL_WUM_STATE_INITIALIZING** = 0x01 , **RFAL_WUM_STATE_ENABLED** = 0x02 , **RFAL_WUM_STATE_ENABLED_WOKE** = 0x03 }

Functions

ReturnCode **rfalInitialize** (void)
RFAL Initialize. More...

ReturnCode **rfalCalibrate** (void)
RFAL Calibrate. More...

ReturnCode **rfalAdjustRegulators** (uint16_t *result)
RFAL Adjust Regulators. More...

ReturnCode **rfalSetRegulators** (uint8_t regulation)
RFAL Set Regulators. More...

void **rfalSetUpperLayerCallback** (**rfalUpperLayerCallback** pFunc)
RFAL Set System Callback. More...

void **rfalSetPreTxRxCallback** (**rfalPreTxRxCallback** pFunc)
RFAL Set Pre Tx Callback. More...

void **rfalSetSyncTxRxCallback** (**rfalSyncTxRxCallback** pFunc)
RFAL Sync Pre Tx Callback. More...

void **rfalSetPostTxRxCallback** (**rfalPostTxRxCallback** pFunc)
RFAL Set Post Tx Callback. More...

void **rfalSetLmEonCallback** (**rfalLmEonCallback** pFunc)
RFAL Set LM EON Callback. More...

ReturnCode **rfalDeinitialize** (void)
RFAL Deinitialize. More...

ReturnCode	**rfalSetMode** (**rfalMode** mode, **rfalBitRate** txBR, **rfalBitRate** rxBR) RFAL Set Mode. More...
rfalMode	**rfalGetMode** (void) RFAL Get Mode. More...
ReturnCode	**rfalSetBitRate** (**rfalBitRate** txBR, **rfalBitRate** rxBR) RFAL Set Bit Rate. More...
ReturnCode	**rfalGetBitRate** (**rfalBitRate** *txBR, **rfalBitRate** *rxBR) RFAL Get Bit Rate. More...
void	**rfalSetErrorHandling** (**rfalEHandling** eHandling) Set Error Handling Mode. More...
rfalEHandling	**rfalGetErrorHandling** (void) Get Error Handling Mode. More...
void	**rfalSetObsvMode** (uint32_t txMode, uint32_t rxMode) Set Observation Mode. More...
void	**rfalGetObsvMode** (uint8_t *txMode, uint8_t *rxMode) Get Observation Mode. More...
void	**rfalDisableObsvMode** (void) Disable Observation Mode. More...
void	**rfalSetFDTPoll** (uint32_t FDTPoll) RFAL Set FDT Poll. More...
uint32_t	**rfalGetFDTPoll** (void) RFAL Set FDT Poll. More...
void	**rfalSetFDTListen** (uint32_t FDTListen) RFAL Set FDT Listen. More...
uint32_t	**rfalGetFDTListen** (void) RFAL Set FDT Listen. More...

uint32_t	**rfalGetGT** (void) RFAL Get GT. More...
void	**rfalSetGT** (uint32_t GT) RFAL Set GT. More...
bool	**rfalIsGTExpired** (void) RFAL Is GT expired. More...
ReturnCode	**rfalFieldOnAndStartGT** (void) RFAL Turn Field On and Start GT. More...
ReturnCode	**rfalFieldOff** (void) RFAL Turn Field Off. More...
ReturnCode	**rfalStartTransceive** (const **rfalTransceiveContext** *ctx) RFAL Set transceive context. More...
rfalTransceiveState	**rfalGetTransceiveState** (void) Get Transceive State. More...
ReturnCode	**rfalGetTransceiveStatus** (void) Get Transceive Status. More...
bool	**rfalIsTransceiveInTx** (void) Is Transceive in Tx. More...
bool	**rfalIsTransceiveInRx** (void) Is Transceive in Rx. More...
ReturnCode	**rfalGetTransceiveRSSI** (uint16_t *rssi) Get Transceive RSSI. More...
bool	**rfalIsTransceiveSubcDetected** (void) Is Transceive Subcarrier Detected. More...
void	**rfalWorker** (void) RFAL Worker. More...

ReturnCode **rfalISO14443ATransceiveShortFrame** (**rfal14443AShortFrameCmd** txCmd, uint8_t *rxBuf, uint8_t rxBufLen, uint16_t *rxRcvdLen, uint32_t fwt)

Transceives an ISO14443A ShortFrame

More...

ReturnCode **rfalISO14443ATransceiveAnticollisionFrame** (uint8_t *buf, uint8_t *bytesToSend, uint8_t *bitsToSend, uint16_t *rxLength, uint32_t fwt)

Sends an ISO14443A Anticollision Frame. More...

ReturnCode **rfalISO14443AStartTransceiveAnticollisionFrame** (uint8_t *buf, uint8_t *bytesToSend, uint8_t *bitsToSend, uint16_t *rxLength, uint32_t fwt)

Start ISO14443A Anticollision Frame transceive. More...

ReturnCode **rfalISO14443AGetTransceiveAnticollisionFrameStatus** (void)

Get ISO14443A Anticollision Frame Status. More...

ReturnCode **rfalFeliCaPoll** (**rfalFeliCaPollSlots** slots, uint16_t sysCode, uint8_t reqCode, **rfalFeliCaPollRes** *pollResList, uint8_t pollResListSize, uint8_t *devicesDetected, uint8_t *collisionsDetected)

FeliCa Poll. More...

ReturnCode **rfalStartFeliCaPoll** (**rfalFeliCaPollSlots** slots, uint16_t sysCode, uint8_t reqCode, **rfalFeliCaPollRes** *pollResList, uint8_t pollResListSize, uint8_t *devicesDetected, uint8_t *collisionsDetected)

Start FeliCa Poll. More...

ReturnCode **rfalGetFeliCaPollStatus** (void)

Get FeliCa Poll Status. More...

ReturnCode **rfalISO15693TransceiveAnticollisionFrame** (uint8_t *txBuf, uint8_t txBufLen, uint8_t *rxBuf, uint8_t rxBufLen, uint16_t *actLen)

Sends an ISO15693 Anticollision Frame. More...

ReturnCode **rfalISO15693TransceiveEOFAnticollision** (uint8_t *rxBuf, uint8_t rxBufLen, uint16_t *actLen)

Sends an ISO15693 Anticollision EOF. More...

ReturnCode **rfalISO15693TransceiveEOF** (uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *actLen)

Sends an ISO15693 EOF. More...

ReturnCode	**rfalTransceiveBlockingTx** (uint8_t *txBuf, uint16_t txBufLen, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *actLen, uint32_t flags, uint32_t fwt) Transceive Blocking Tx. More...
ReturnCode	**rfalTransceiveBlockingRx** (void) Transceive Blocking Rx. More...
ReturnCode	**rfalTransceiveBlockingTxRx** (uint8_t *txBuf, uint16_t txBufLen, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *actLen, uint32_t flags, uint32_t fwt) Transceive Blocking. More...
bool	**rfalIsExtFieldOn** (void) Is external Field On. More...
ReturnCode	**rfalListenStart** (uint32_t lmMask, const **rfalLmConfPA** *confA, const **rfalLmConfPB** *confB, const **rfalLmConfPF** *confF, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rxLen) Listen Mode start. More...
ReturnCode	**rfalListenSleepStart** (**rfalLmState** sleepSt, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rxLen) Listen Mode start Sleeping. More...
ReturnCode	**rfalListenStop** (void) Listen Mode Stop. More...
rfalLmState	**rfalListenGetState** (bool *dataFlag, **rfalBitRate** *lastBR) Listen Mode get state. More...
ReturnCode	**rfalListenSetState** (**rfalLmState** newSt) Listen Mode set state. More...
ReturnCode	**rfalWakeUpModeStart** (const **rfalWakeUpConfig** *config) Wake-Up Mode Start. More...
bool	**rfalWakeUpModeHasWoke** (void) Wake-Up has Woke. More...
bool	**rfalWakeUpModeIsEnabled** (void) Wake-Up is Enabled. More...

ReturnCode	**rfalWakeUpModeGetInfo** (bool force, **rfalWakeUpInfo** *info) Wake-Up Get Info. More...
ReturnCode	**rfalWakeUpModeStop** (void) Wake-Up Mode Stop. More...
ReturnCode	**rfalWlcPWptMonitorStart** (const **rfalWakeUpConfig** *config) WLC-P WPT Monitor Start. More...
ReturnCode	**rfalWlcPWptMonitorStop** (void) WLC-P WPT Monitor Start Stop. More...
bool	**rfalWlcPWptIsFodDetected** (void) WLC-P WPT FOD is Detected. More...
bool	**rfalWlcPWptIsStopDetected** (void) WLC-P WPT Stop is Detected. More...
ReturnCode	**rfalLowPowerModeStart** (**rfalLpMode** mode) Low Power Mode Start. More...
ReturnCode	**rfalLowPowerModeStop** (void) Low Power Mode Stop. More...

Detailed Description

Macro Definition Documentation

◆ RFAL_1FC_IN_2048FC

```
#define RFAL_1FC_IN_2048FC   (uint32_t)2048U
```

Number of 1/fc cycles in one 2048/fc

◆ RFAL_1FC_IN_4096FC

```
#define RFAL_1FC_IN_4096FC   (uint32_t)4096U
```

Number of 1/fc cycles in one 4096/fc

◆ RFAL_1FC_IN_512FC

```
#define RFAL_1FC_IN_512FC   (uint32_t)512U
```

Number of 1/fc cycles in one 512/fc

◆ RFAL_1FC_IN_64FC

```
#define RFAL_1FC_IN_64FC   (uint32_t)64U
```

Number of 1/fc cycles in one 64/fc

◆ RFAL_1FC_IN_8FC

```
#define RFAL_1FC_IN_8FC   (uint32_t)8U
```

Number of 1/fc cycles in one 8/fc

◆ RFAL_1MS_IN_1FC

```
#define RFAL_1MS_IN_1FC   (uint32_t)13560U
```

Number of 1/fc cycles in 1ms

◆ RFAL_BITS_IN_BYTE

```
#define RFAL_BITS_IN_BYTE   (uint16_t)8U
```

Number of bits in one byte

◆ RFAL_CRC_LEN

```
#define RFAL_CRC_LEN   2U
```

RF CRC LEN

◆ RFAL_FDT_LISTEN_AP2P_LISTENER

```
#define RFAL_FDT_LISTEN_AP2P_LISTENER   64U
```

FDT AP2P No actual FDTListen exists as fields switch and collision avoidance

◆ RFAL_FDT_LISTEN_AP2P_POLLER

```
#define RFAL_FDT_LISTEN_AP2P_POLLER   64U
```

FDT AP2P No actual FDTListen is required as fields switch and collision avoidance

◆ RFAL_FDT_LISTEN_NFCA_LISTENER

```
#define RFAL_FDT_LISTEN_NFCA_LISTENER   1172U
```

FDTA,LISTEN,MIN Digital 1.1 6.10

◆ RFAL_FDT_LISTEN_NFCA_POLLER

```
#define RFAL_FDT_LISTEN_NFCA_POLLER   1172U
```

FDTA,LISTEN,MIN (n=9) Last bit: Logic "1" - tnn,min/2 Digital 1.1 6.10 ; EMV CCP Spec Book D v2.01 4.8.1.3

◆ RFAL_FDT_LISTEN_NFCB_LISTENER

```
#define RFAL_FDT_LISTEN_NFCB_LISTENER   1024U
```

TR0B,MIN Digital 1.1 7.1.3 & A.3 ; EMV CCP Spec Book D v2.01 4.8.1.3 & Table A.5

◆ RFAL_FDT_LISTEN_NFCB_POLLER

```
#define RFAL_FDT_LISTEN_NFCB_POLLER   1008U
```

TR0B,MIN Digital 1.1 7.1.3 & A.3 ; EMV CCP Spec Book D v2.01 4.8.1.3 & Table A.5

◆ RFAL_FDT_LISTEN_NFCF_LISTENER

```
#define RFAL_FDT_LISTEN_NFCF_LISTENER   2688U
```

TR0F,LISTEN,MIN Digital 2.1 8.7.1.1 & B.4

◆ RFAL_FDT_LISTEN_NFCF_POLLER

```
#define RFAL_FDT_LISTEN_NFCF_POLLER   2672U
```

TR0F,LISTEN,MIN Digital 1.1 8.7.1.1 & A.4

◆ RFAL_FDT_LISTEN_NFCV_POLLER

```
#define RFAL_FDT_LISTEN_NFCV_POLLER   4310U
```

FDTV,LISTEN,MIN t1 min Digital 2.1 B.5 ; ISO15693-3 2009 9.1

◆ RFAL_FDT_LISTEN_PICOPASS_POLLER

```
#define RFAL_FDT_LISTEN_PICOPASS_POLLER   3400U
```

ISO15693 t1 min - observed adjustment

◆ RFAL_FDT_POLL_AP2P_POLLER

```
#define RFAL_FDT_POLL_AP2P_POLLER   6800U
```

AP2P inhere FDT from the Technology used (use longest: TR0F,POLL,MIN + TR1F) Digital 2.2 17.11.1

◆ RFAL_FDT_POLL_NFCA_POLLER

```
#define RFAL_FDT_POLL_NFCA_POLLER   6780U
```

FDTA,POLL,MIN Digital 1.1 6.10.3.1 & A.2

◆ RFAL_FDT_POLL_NFCA_T1T_POLLER

```
#define RFAL_FDT_POLL_NFCA_T1T_POLLER   384U
```

RRDDT1T,MIN,B1 Digital 1.1 10.7.1 & A.5

◆ RFAL_FDT_POLL_NFCB_POLLER

```
#define RFAL_FDT_POLL_NFCB_POLLER   6780U
```

FDTB,POLL,MIN = TR2B,MIN,DEFAULT Digital 1.1 7.9.3 & A.3 ; EMVCo 3.0 FDTB,PCD,MIN Table A.5

◆ RFAL_FDT_POLL_NFCF_POLLER

```
#define RFAL_FDT_POLL_NFCF_POLLER   6800U
```

FDTF,POLL,MIN Digital 2.1 8.7.3 & B.4

◆ RFAL_FDT_POLL_NFCV_POLLER

```
#define RFAL_FDT_POLL_NFCV_POLLER   4192U
```

FDTV,POLL Digital 2.1 9.7.3.1 & B.5

◆ RFAL_FDT_POLL_PICOPASS_POLLER

```
#define RFAL_FDT_POLL_PICOPASS_POLLER   1790U
```

FDT Max

◆ RFAL_FELICA_LEN_LEN

```
#define RFAL_FELICA_LEN_LEN   1U
```

FeliCa LEN byte length

◆ RFAL_FELICA_POLL_MAX_SLOTS

```
#define RFAL_FELICA_POLL_MAX_SLOTS   16U
```

Maximum number of slots (TSN) on FeliCa Poll

◆ RFAL_FELICA_POLL_REQ_LEN

```
#define RFAL_FELICA_POLL_REQ_LEN   (RFAL_FELICA_LEN_LEN + 1U + 2U + 1U + 1U)
```

FeliCa Poll Request length (LEN + CMD + SC + RC + TSN)

◆ RFAL_FELICA_POLL_RES_LEN

```
#define RFAL_FELICA_POLL_RES_LEN   (RFAL_FELICA_LEN_LEN + 1U + 8U + 8U + 2U)
```

Maximum FeliCa Poll Response length (LEN + CMD + NFCID2 + PAD + RD)

◆ RFAL_FWT_NONE

```
#define RFAL_FWT_NONE   0xFFFFFFFFU
```

Disabled FWT: Wait forever for a response

◆ RFAL_GT_AP2P

```
#define RFAL_GT_AP2P   rfalConvMsTo1fc(5U)
```

TIRFG Ecma 340 11.1.1

◆ RFAL_GT_AP2P_ADJUSTED

```
#define RFAL_GT_AP2P_ADJUSTED   rfalConvMsTo1fc(5U+25U)
```

Adjusted GT for greater interoperability (Sony XPERIA P, Nokia N9, Huawei P2)

◆ RFAL_GT_NFCA

#define RFAL_GT_NFCA **rfalConvMsTo1fc**(5U)

GTA Digital 2.0 6.10.4.1 & B.2

◆ RFAL_GT_NFCB

#define RFAL_GT_NFCB **rfalConvMsTo1fc**(5U)

GTB Digital 2.0 7.9.4.1 & B.3

◆ RFAL_GT_NFCF

#define RFAL_GT_NFCF **rfalConvMsTo1fc**(20U)

GTF Digital 2.0 8.7.4.1 & B.4

◆ RFAL_GT_NFCV

#define RFAL_GT_NFCV **rfalConvMsTo1fc**(5U)

GTV Digital 2.0 9.7.5.1 & B.5

◆ RFAL_GT_NONE

#define RFAL_GT_NONE **RFAL_TIMING_NONE**

Disabled GT: No GT will be applied after Field On

◆ RFAL_GT_PICOPASS

#define RFAL_GT_PICOPASS **rfalConvMsTo1fc**(1U)

GT Picopass

◆ RFAL_LM_MASK_ACTIVE_P2P

```
#define RFAL_LM_MASK_ACTIVE_P2P   ((uint32_t)1U<<(uint8_t)RFAL_MODE_LISTEN_ACTIVE_P2P)
```

Bitmask for Listen Mode enabling AP2P

◆ RFAL_LM_MASK_NFCA

```
#define RFAL_LM_MASK_NFCA   ((uint32_t)1U<<(uint8_t)RFAL_MODE_LISTEN_NFCA)
```

Bitmask for Listen Mode enabling NFCA

◆ RFAL_LM_MASK_NFCB

```
#define RFAL_LM_MASK_NFCB   ((uint32_t)1U<<(uint8_t)RFAL_MODE_LISTEN_NFCB)
```

Bitmask for Listen Mode enabling NFCB

◆ RFAL_LM_MASK_NFCF

```
#define RFAL_LM_MASK_NFCF   ((uint32_t)1U<<(uint8_t)RFAL_MODE_LISTEN_NFCF)
```

Bitmask for Listen Mode enabling NFCF

◆ RFAL_LM_SENS_RES_LEN

```
#define RFAL_LM_SENS_RES_LEN   2U
```

NFC-A SENS_RES (ATQA) length

◆ RFAL_LM_SENSB_RES_LEN

```
#define RFAL_LM_SENSB_RES_LEN   13U
```

NFC-B SENSB_RES (ATQB) length

◆ RFAL_LM_SENSF_RES_LEN

```
#define RFAL_LM_SENSF_RES_LEN   19U
```

NFC-F SENSF_RES length

◆ RFAL_LM_SENSF_SC_LEN

```
#define RFAL_LM_SENSF_SC_LEN   2U
```

NFC-F System Code length

◆ RFAL_NFCID1_DOUBLE_LEN

```
#define RFAL_NFCID1_DOUBLE_LEN   7U
```

NFCID1 length

◆ RFAL_NFCID1_SINGLE_LEN

```
#define RFAL_NFCID1_SINGLE_LEN   4U
```

NFCID1 length

◆ RFAL_NFCID1_TRIPLE_LEN

```
#define RFAL_NFCID1_TRIPLE_LEN   10U
```

NFCID1 length

◆ RFAL_NFCID2_LEN

```
#define RFAL_NFCID2_LEN   8U
```

NFCID2 length

◆ RFAL_NFCID3_LEN

```
#define RFAL_NFCID3_LEN   10U
```

NFCID3 length

◆ RFAL_TIMING_NONE

```
#define RFAL_TIMING_NONE   0x00U
```

Timing disabled | Don't apply

◆ RFAL_TXRX_FLAGS_DEFAULT

#define RFAL_TXRX_FLAGS_DEFAULT ((uint32_t)**RFAL_TXRX_FLAGS_CRC_TX_AUTO** | (uint32_t)**RFAL_TXRX_FLAGS_CRC_RX_REMV** | (uint32_t)**RFAL_TXRX_FLAGS_NFCIP1_OFF** | (uint32_t)**RFAL_TXRX_FLAGS_AGC_ON** | (uint32_t)**RFAL_TXRX_FLAGS_PAR_RX_REMV** | (uint32_t)**RFAL_TXRX_FLAGS_PAR_TX_AUTO** | (uint32_t)**RFAL_TXRX_FLAGS_NFCV_FLAG_AUTO**)

Default TxRx flags: Tx CRC automatic, Rx CRC removed, NFCIP1 mode off, AGC On, Tx Parity automatic, Rx Parity removed

Examples

exampleRfalPoller.c.

◆ RFAL_US_IN_MS

#define RFAL_US_IN_MS (uint32_t)1000U

Number of us in one ms

◆ RFAL_VERSION

```
#define RFAL_VERSION   0x040000U
```

RFAL Current Version: v4.0.0

◆ RFAL_WUM_REFERENCE_AUTO

```
#define RFAL_WUM_REFERENCE_AUTO   0xFFU
```

Indicates new reference is set by the driver

◆ rfalConv1fcTo2018fc

```
#define rfalConv1fcTo2018fc (   t )   (uint32_t)( (uint32_t)(t) / RFAL_1FC_IN_2048FC )
```

Converts the given t from 1/fc to 2048/fc

◆ rfalConv1fcTo4096fc

```
#define rfalConv1fcTo4096fc (   t )   (uint32_t)( (uint32_t)(t) / RFAL_1FC_IN_4096FC )
```

Converts the given t from 1/fc to 4096/fc

◆ rfalConv1fcTo512fc

```
#define rfalConv1fcTo512fc (   t )   (uint32_t)( (uint32_t)(t) / RFAL_1FC_IN_512FC )
```

Converts the given t from 1/fc to 512/fc

◆ rfalConv1fcTo64fc

```
#define rfalConv1fcTo64fc (   t )   (uint32_t)( (uint32_t)(t) / RFAL_1FC_IN_64FC )
```

Converts the given t from 1/fc to 64/fc

◆ rfalConv1fcTo8fc

```
#define rfalConv1fcTo8fc (   t )   (uint32_t)( (uint32_t)(t) / RFAL_1FC_IN_8FC )
```

Converts the given t from 1/fc to 8/fc

◆ rfalConv1fcToMs

```
#define rfalConv1fcToMs ( t ) (uint32_t)( (uint32_t)(t) / RFAL_1MS_IN_1FC )
```

Converts the given t from 1/fc to ms

◆ rfalConv1fcToUs

```
#define rfalConv1fcToUs ( t ) (uint32_t)( ((uint32_t)(t) * RFAL_US_IN_MS) / RFAL_1MS_IN_1FC)
```

Converts the given t from 1/fc to us

◆ rfalConv2048fcTo1fc

```
#define rfalConv2048fcTo1fc ( t ) (uint32_t)( (uint32_t)(t) * RFAL_1FC_IN_2048FC )
```

Converts the given t from 2048/fc to 1/fc

◆ rfalConv4096fcTo1fc

```
#define rfalConv4096fcTo1fc ( t ) (uint32_t)( (uint32_t)(t) * RFAL_1FC_IN_4096FC )
```

Converts the given t from 4096/fc to 1/fc

◆ rfalConv512fcTo1fc

```
#define rfalConv512fcTo1fc ( t ) (uint32_t)( (uint32_t)(t) * RFAL_1FC_IN_512FC )
```

Converts the given t from 512/fc to 1/fc

◆ rfalConv64fcTo1fc

```
#define rfalConv64fcTo1fc ( t ) (uint32_t)( (uint32_t)(t) * RFAL_1FC_IN_64FC )
```

Converts the given t from 64/fc to 1/fc

◆ rfalConv64fcToMs

#define rfalConv64fcToMs (t) (uint32_t)((uint32_t)(t) / (**RFAL_1MS_IN_1FC** / **RFAL_1FC_IN_64FC**))

Converts the given t from 64/fc to ms

◆ rfalConv8fcTo1fc

#define rfalConv8fcTo1fc (t) (uint32_t)((uint32_t)(t) * **RFAL_1FC_IN_8FC**)

Converts the given t from 8/fc to 1/fc

◆ rfalConvBitsToBytes

#define rfalConvBitsToBytes (n) (uint16_t)(((uint16_t)(n)+(**RFAL_BITS_IN_BYTE**-1U)) / (**RFAL_BITS_IN_BYTE**))

Converts the given n from bits to bytes

◆ rfalConvBytesToBits

#define rfalConvBytesToBits (n) (uint32_t)((uint32_t)(n) * (**RFAL_BITS_IN_BYTE**))

Converts the given n from bytes to bits

◆ rfalConvMsTo1fc

#define rfalConvMsTo1fc (t) (uint32_t)((uint32_t)(t) * **RFAL_1MS_IN_1FC**)

Converts the given t from ms to 1/fc

Examples

exampleRfalPoller.c.

◆ rfalConvMsTo64fc

#define rfalConvMsTo64fc (t) (uint32_t)((uint32_t)(t) * (**RFAL_1MS_IN_1FC** / **RFAL_1FC_IN_64FC**))

Converts the given t from ms to 64/fc

◆ rfalConvUsTo1fc

#define rfalConvUsTo1fc (t) (uint32_t)(((uint32_t)(t) * **RFAL_1MS_IN_1FC**) / **RFAL_US_IN_MS**)

Converts the given t from us to 1/fc

◆ rfalCreateByteFlagsTxRxContext

#define rfalCreateByteFlagsTxRxContext (ctx, tB, tBL, rB, rBL, rdL, fl, t)

Value:

```
(ctx).txBuf     = (uint8_t*)(tB);                                  \
(ctx).txBufLen  = (uint16_t)rfalConvBytesToBits(tBL);              \
(ctx).rxBuf     = (uint8_t*)(rB);                                  \
(ctx).rxBufLen  = (uint16_t)rfalConvBytesToBits(rBL);              \
(ctx).rxRcvdLen = (uint16_t*)(rdL);                                \
(ctx).flags     = (uint32_t)(fl);                                  \
(ctx).fwt       = (uint32_t)(t);
```

Computes a Transceive context *ctx* using lengths in bytes with the given flags and arguments *ctx* : Transceive context to be assigned

tB : txBuf the pointer to the buffer to be sent *tBL* : txBuf length in bytes *rB* : rxBuf the pointer to the buffer to place the received frame *rBL* : rxBuf length in bytes *rBL* : rxBuf length in bytes *t* : FWT to be used on this transceive in 1/fc

Examples

exampleRfalPoller.c.

◆ rfalCreateByteTxRxContext

```
#define rfalCreateByteTxRxContext (  ctx,
                                     tB,
                                     tBL,
                                     rB,
                                     rBL,
                                     rdL,
                                     t
                                  )
```

Value:

```
    (ctx).txBuf     = (uint8_t*)(tB);                                   \
    (ctx).txBufLen  = (uint16_t)rfalConvBytesToBits(tBL);               \
    (ctx).rxBuf     = (uint8_t*)(rB);                                   \
    (ctx).rxBufLen  = (uint16_t)rfalConvBytesToBits(rBL);               \
    (ctx).rxRcvdLen = (uint16_t*)(rdL);                                 \
    (ctx).flags     = (uint32_t)RFAL_TXRX_FLAGS_DEFAULT;                \
    (ctx).fwt       = (uint32_t)(t);
```

Computes a Transceive context *ctx* with default flags and the lengths in bytes with the given arguments *ctx* : Transceive context to be assigned
tB : txBuf the pointer to the buffer to be sent *tBL* : txBuf length in bytes *rB* : rxBuf the pointer to the buffer to place the received frame *rBL* : rxBuf length in bytes *rdL* : rxRcvdLen the pointer to place the rx length *t* : FWT to be used on this transceive in 1/fc

◆ rfalGetMaxBrAP2P

```
#define rfalGetMaxBrAP2P ( ((RFAL_SUPPORT_BR_AP2P_848) ? RFAL_BR_848 : ((RFAL_SUPPORT_BR_AP2P_424) ? RFAL_BR_424 : ((RFAL_SUPPORT_BR_AP2P_212) ? RFAL_BR_212 : RFAL_BR_106 ) ) ) )
```

Returns the maximum supported bit rate for AP2P mode. Caller must check if mode is supported before, as even if mode is not supported will return the min

◆ rfalGetMaxBrCEA

```
#define rfalGetMaxBrCEA ( ((RFAL_SUPPORT_BR_CE_A_848) ? RFAL_BR_848 : ((RFAL_SUPPORT_BR_CE_A_424) ? RFAL_BR_424 : ((RFAL_SUPPORT_BR_CE_A_212) ? RFAL_BR_212 : RFAL_BR_106 ) ) ) )
```

Returns the maximum supported bit rate for CE-A mode. Caller must check if mode is supported before, as even if mode is not supported will return the min

◆ rfalGetMaxBrCEB

```
#define rfalGetMaxBrCEB ( ) ( ((RFAL_SUPPORT_BR_CE_B_848) ? RFAL_BR_848 : ((RFAL_SUPPORT_BR_CE_B_424) ? RFAL_BR_424 : ((RFAL_SUPPORT_BR_CE_B_212) ? RFAL_BR_212 : RFAL_BR_106 ) ) ) ) )
```

Returns the maximum supported bit rate for CE-B mode. Caller must check if mode is supported before, as even if mode is not supported will return the min

◆ rfalGetMaxBrCEF

```
#define rfalGetMaxBrCEF ( ) ( ((RFAL_SUPPORT_BR_CE_F_424) ? RFAL_BR_424 : RFAL_BR_212 ) )
```

Returns the maximum supported bit rate for CE-F mode. Caller must check if mode is supported before, as even if mode is not supported will return the min

◆ rfalGetMaxBrRW

```
#define rfalGetMaxBrRW ( ) ( ((RFAL_SUPPORT_BR_RW_6780) ? RFAL_BR_6780 : ((RFAL_SUPPORT_BR_RW_3390) ? RFAL_BR_3390 : ((RFAL_SUPPORT_BR_RW_1695) ? RFAL_BR_1695 : ((RFAL_SUPPORT_BR_RW_848) ? RFAL_BR_848 : ((RFAL_SUPPORT_BR_RW_424) ? RFAL_BR_424 : ((RFAL_SUPPORT_BR_RW_212) ? RFAL_BR_212 : RFAL_BR_106 ) ) ) ) ) ) ) )
```

Returns the maximum supported bit rate for RW mode. Caller must check if mode is supported before, as even if mode is not supported will return the min

◆ rfalIsModeActiveComm

```
#define rfalIsModeActiveComm ( md ) ( ((md) == RFAL_MODE_POLL_ACTIVE_P2P) || ((md) == RFAL_MODE_LISTEN_ACTIVE_P2P) )
```

Checks if mode md is Active Communication

◆ rfalIsModePassiveComm

```
#define rfalIsModePassiveComm ( md ) ( !rfalIsModeActiveComm(md) )
```

Checks if mode md is Passive Communication

◆ rfalIsModePassiveListen

```
#define rfalIsModePassiveListen ( md ) ( ((md) == RFAL_MODE_LISTEN_NFCA) || ((md) == RFAL_MODE_LISTEN_NFCB) || ((md) == RFAL_MODE_LISTEN_NFCF) )
```

Checks if mode md is Passive Listen

◆ rfalIsModePassivePoll

```
#define rfalIsModePassivePoll ( md ) ( rfalIsModePassiveComm(md) && (!rfalIsModePassiveListen(md)) )
```

Checks if mode md is Passive Poll

◆ rfalLogD

```
#define rfalLogD ( ... ) platformLog(__VA_ARGS__)
```

Macro for the debug log method

◆ rfalLogE

```
#define rfalLogE ( ... ) platformLog(__VA_ARGS__)
```

Macro for the error log method

◆ rfalLogI

```
#define rfalLogI ( ... ) platformLog(__VA_ARGS__)
```

Macro for the info log method

◆ rfalLogW

```
#define rfalLogW ( ... ) platformLog(__VA_ARGS__)
```

Macro for the warning log method

◆ rfalRunBlocking

```
#define rfalRunBlocking (   e,
                            fn
                        )     do{ (e)=(fn); rfalWorker(); }while( (e) == RFAL_ERR_BUSY )
```

Macro used for the blocking methods

Typedef Documentation

◆ rfalFeliCaPollRes

```
typedef uint8_t rfalFeliCaPollRes[RFAL_FELICA_POLL_RES_LEN]
```

NFCF Poll Response NFC Forum Digital 1.1 Table 44

◆ rfalLmEonCallback

```
typedef void(* rfalLmEonCallback) (void)
```

Callback upon External Field detected while in Listen Mode

◆ rfalPostTxRxCallback

```
typedef void(* rfalPostTxRxCallback) (void)
```

Callback to be executed after a Transceive

◆ rfalPreTxRxCallback

```
typedef void(* rfalPreTxRxCallback) (void)
```

Callback to be executed before a Transceive

◆ rfalSyncTxRxCallback

typedef bool(* rfalSyncTxRxCallback) (void)

Callback to sync actual transmission start

◆ rfalUpperLayerCallback

typedef void(* rfalUpperLayerCallback) (void)

System callback to indicate an event that requires a system reRun

Enumeration Type Documentation

◆ anonymous enum

anonymous enum

RFAL transceive flags

Enumerator	
RFAL_TXRX_FLAGS_CRC_TX_AUTO	CRC will be generated automatic upon transmission
RFAL_TXRX_FLAGS_CRC_TX_MANUAL	CRC was calculated manually, included in txBuffer
RFAL_TXRX_FLAGS_CRC_RX_KEEP	Upon Reception keep the CRC in rxBuffer (reflected on rcvd length)
RFAL_TXRX_FLAGS_CRC_RX_REMV	Remove the CRC from rxBuffer
RFAL_TXRX_FLAGS_NFCIP1_ON	Enable NFCIP1 mode: Add SB(F0) and LEN bytes during Tx and skip SB(F0) byte during Rx
RFAL_TXRX_FLAGS_NFCIP1_OFF	Disable NFCIP1 mode: do not append protocol bytes while Tx nor skip while Rx
RFAL_TXRX_FLAGS_AGC_OFF	Disable Automatic Gain Control, improving multiple devices collision detection. **DEPRECATED:** flag is depreacted, usage of Anticollision APIs based on Analog Config table with RFAL_ANALOG_CONFIG_ANTICOL settings
RFAL_TXRX_FLAGS_AGC_ON	Enable Automatic Gain Control, improving single device reception **DEPRECATED:** flag is deprecated, usage of Anticollision APIs based on Analog Config table with RFAL_ANALOG_CONFIG_ANTICOL settings

RFAL_TXRX_FLAGS_PAR_RX_KEEP	Disable Parity check and keep the Parity and CRC bits in the received buffer
RFAL_TXRX_FLAGS_PAR_RX_REMV	Enable Parity check and remove the parity bits from the received buffer
RFAL_TXRX_FLAGS_PAR_TX_NONE	Disable automatic Parity generation (ISO14443A) and use the one provided in the buffer
RFAL_TXRX_FLAGS_PAR_TX_AUTO	Enable automatic Parity generation (ISO14443A)
RFAL_TXRX_FLAGS_NFCV_FLAG_MANUAL	Disable automatic adaption of flag byte (ISO15693) according to current comm params
RFAL_TXRX_FLAGS_NFCV_FLAG_AUTO	Enable automatic adaption of flag byte (ISO115693) according to current comm params
RFAL_TXRX_FLAGS_CRC_RX_MANUAL	Disable automatic CRC check
RFAL_TXRX_FLAGS_CRC_RX_AUTO	Enable automatic CRC check

◆ anonymous enum

anonymous enum

NFC-F RC (Request Code) codes NFC Forum Digital 1.1 Table 42

Enumerator	
RFAL_FELICA_POLL_RC_NO_REQUEST	RC: No System Code information requested
RFAL_FELICA_POLL_RC_SYSTEM_CODE	RC: System Code information requested
RFAL_FELICA_POLL_RC_COM_PERFORMANCE	RC: Advanced protocol features supported

◆ rfal14443AShortFrameCmd

enum **rfal14443AShortFrameCmd**

RFAL ISO 14443A Short Frame Command

Enumerator	
RFAL_14443A_SHORTFRAME_CMD_WUPA	ISO14443A WUPA / NFC-A ALL_REQ
RFAL_14443A_SHORTFRAME_CMD_REQA	ISO14443A REQA / NFC-A SENS_REQ

◆ rfalBitRate

enum **rfalBitRate**

RFAL Bit rates

Enumerator	
RFAL_BR_106	Bit Rate 106 kbit/s (fc/128)
RFAL_BR_212	Bit Rate 212 kbit/s (fc/64)
RFAL_BR_424	Bit Rate 424 kbit/s (fc/32)
RFAL_BR_848	Bit Rate 848 kbit/s (fc/16)
RFAL_BR_1695	Bit Rate 1695 kbit/s (fc/8)
RFAL_BR_3390	Bit Rate 3390 kbit/s (fc/4)
RFAL_BR_6780	Bit Rate 6780 kbit/s (fc/2)
RFAL_BR_13560	Bit Rate 13560 kbit/s (fc)
RFAL_BR_211p88	Bit Rate 211,88 kbit/s (fc/64) Fast Mode VICC->VCD
RFAL_BR_105p94	Bit Rate 105,94 kbit/s (fc/128) Fast Mode VICC->VCD
RFAL_BR_52p97	Bit Rate 52.97 kbit/s (fc/256) Fast Mode VICC->VCD
RFAL_BR_26p48	Bit Rate 26,48 kbit/s (fc/512) NFCV VICC->VCD & VCD->VICC 1of4
RFAL_BR_1p66	Bit Rate 1,66 kbit/s (fc/8192) NFCV VCD->VICC 1of256
RFAL_BR_KEEP	Value indicating to keep the same previous bit rate

◆ rfalComplianceMode

enum **rfalComplianceMode**

RFAL Compliance modes for upper modules

Enumerator	
RFAL_COMPLIANCE_MODE_NFC	Perform with NFC Forum 1.1 compliance
RFAL_COMPLIANCE_MODE_EMV	Perform with EMVCo compliance
RFAL_COMPLIANCE_MODE_ISO	Perform with ISO10373 compliance

◆ rfalEHandling

enum **rfalEHandling**

RFAL error handling

Enumerator	
RFAL_ERRORHANDLING_NONE	No special error handling will be performed
RFAL_ERRORHANDLING_EMD	EMD suppression enabled Digital 2.1 4.1.1.1 ; EMVCo 3.0 4.9.2 ; ISO 14443-3 8.3

◆ rfalFeliCaPollSlots

enum **rfalFeliCaPollSlots**

NFC-F TSN (Time Slot Number) codes NFC Forum Digital 1.1 Table 43

Enumerator	
RFAL_FELICA_1_SLOT	TSN with number of Time Slots: 1
RFAL_FELICA_2_SLOTS	TSN with number of Time Slots: 2
RFAL_FELICA_4_SLOTS	TSN with number of Time Slots: 4
RFAL_FELICA_8_SLOTS	TSN with number of Time Slots: 8
RFAL_FELICA_16_SLOTS	TSN with number of Time Slots: 16

◆ rfalLmNfcidLen

enum **rfalLmNfcidLen**

RFAL Listen Mode NFCID Length

Enumerator	
RFAL_LM_NFCID_LEN_04	Listen mode indicates 4 byte NFCID
RFAL_LM_NFCID_LEN_07	Listen mode indicates 7 byte NFCID
RFAL_LM_NFCID_LEN_10	Listen mode indicates 10 byte NFCID

◆ rfalLmState

enum **rfalLmState**

RFAL Listen Mode States

Enumerator	
RFAL_LM_STATE_NOT_INIT	Not Initialized state
RFAL_LM_STATE_POWER_OFF	Power Off state
RFAL_LM_STATE_IDLE	Idle state Activity 1.1 5.2
RFAL_LM_STATE_READY_A	Ready A state Activity 1.1 5.3 5.4 & 5.5
RFAL_LM_STATE_READY_B	Ready B state Activity 1.1 5.11 5.12
RFAL_LM_STATE_READY_F	Ready F state Activity 1.1 5.15
RFAL_LM_STATE_ACTIVE_A	Active A state Activity 1.1 5.6
RFAL_LM_STATE_CARDEMU_4A	Card Emulation 4A state Activity 1.1 5.10
RFAL_LM_STATE_CARDEMU_4B	Card Emulation 4B state Activity 1.1 5.14
RFAL_LM_STATE_CARDEMU_3	Card Emulation 3 state Activity 1.1 5.18
RFAL_LM_STATE_TARGET_A	Target A state Activity 1.1 5.9
RFAL_LM_STATE_TARGET_F	Target F state Activity 1.1 5.17
RFAL_LM_STATE_SLEEP_A	Sleep A state Activity 1.1 5.7
RFAL_LM_STATE_SLEEP_B	Sleep B state Activity 1.1 5.13
RFAL_LM_STATE_READY_Ax	Ready A* state Activity 1.1 5.3 5.4 & 5.5
RFAL_LM_STATE_ACTIVE_Ax	Active A* state Activity 1.1 5.6
RFAL_LM_STATE_SLEEP_AF	Sleep AF state Activity 1.1 5.19

◆ rfalLpMode

enum **rfalLpMode**

RFAL low power modes

Enumerator

RFAL_LP_MODE_PD	Set RF Chip in Power Down state
RFAL_LP_MODE_HR	Set RF Chip in Hold Reset state (available for specific devices)

◆ rfalMode

enum **rfalMode**

RFAL modes

Enumerator	
RFAL_MODE_NONE	No mode selected/defined
RFAL_MODE_POLL_NFCA	Mode to perform as NFCA (ISO14443A) Poller (PCD)
RFAL_MODE_POLL_NFCA_T1T	Mode to perform as NFCA T1T (Topaz) Poller (PCD)
RFAL_MODE_POLL_NFCB	Mode to perform as NFCB (ISO14443B) Poller (PCD)
RFAL_MODE_POLL_B_PRIME	Mode to perform as B' Calypso (Innovatron) (PCD)
RFAL_MODE_POLL_B_CTS	Mode to perform as CTS Poller (PCD)
RFAL_MODE_POLL_NFCF	Mode to perform as NFCF (FeliCa) Poller (PCD)
RFAL_MODE_POLL_NFCV	Mode to perform as NFCV (ISO15963) Poller (PCD)
RFAL_MODE_POLL_PICOPASS	Mode to perform as PicoPass / iClass Poller (PCD)
RFAL_MODE_POLL_ACTIVE_P2P	Mode to perform as Active P2P (ISO18092) Initiator
RFAL_MODE_LISTEN_NFCA	Mode to perform as NFCA (ISO14443A) Listener (PICC)
RFAL_MODE_LISTEN_NFCB	Mode to perform as NFCA (ISO14443B) Listener (PICC)
RFAL_MODE_LISTEN_NFCF	Mode to perform as NFCA (ISO15963) Listener (PICC)
RFAL_MODE_LISTEN_ACTIVE_P2P	Mode to perform as Active P2P (ISO18092) Target

◆ rfalState

enum **rfalState**

RFAL main states flags

◆ rfalTransceiveState

enum **rfalTransceiveState**

RFAL transceive states

◆ rfalWumState

enum **rfalWumState**

RFAL Wake-Up Mode States

Enumerator	
RFAL_WUM_STATE_NOT_INIT	Not Initialized state
RFAL_WUM_STATE_INITIALIZING	Wake-Up mode is starting
RFAL_WUM_STATE_ENABLED	Wake-Up mode is enabled
RFAL_WUM_STATE_ENABLED_WOKE	Wake-Up mode enabled and has received IRQ(s)

Function Documentation

◆ rfalAdjustRegulators()

ReturnCode rfalAdjustRegulators (uint16_t * result)

Adjusts ST25R regulators

Parameters

[out] **result** : the result of the calibrate antenna in mV NULL if result not requested

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized

RFAL_ERR_NONE : No error

◆ rfalCalibrate()

ReturnCode rfalCalibrate (void)

Performs necessary calibration of RF chip in case it is indicated by current register settings. E.g. antenna calibration and regulator calibration

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized

RFAL_ERR_NONE : No error

◆ rfalDeinitialize()

```
ReturnCode rfalDeinitialize ( void )
```

Deinitializes RFAL layer and the ST25R

Returns

RFAL_ERR_NONE : No error

◆ rfalDisableObsvMode()

```
void rfalDisableObsvMode ( void )
```

Disables the ST25R observation mode

◆ rfalFeliCaPoll()

```
ReturnCode rfalFeliCaPoll ( rfalFeliCaPollSlots  slots,
                            uint16_t             sysCode,
                            uint8_t              reqCode,
                            rfalFeliCaPollRes *  pollResList,
                            uint8_t              pollResListSize,
                            uint8_t *            devicesDetected,
                            uint8_t *            collisionsDetected
                          )
```

Sends a Poll Request and collects all Poll Responses according to the given slots

Parameters

[in] **slots** : number of slots for the Poll Request

[in]	**sysCode**	: system code (SC) for the Poll Request
[in]	**reqCode**	: request code (RC) for the Poll Request
[out]	**pollResList**	: list of all responses
[in]	**pollResListSize**	: number of responses that can be placed in pollResList
[out]	**devicesDetected**	: number of cards found
[out]	**collisionsDetected**	number of collisions detected

Returns

RFAL_ERR_NONE : If there is no error

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_WRONG_STATE : RFAL not initialized or mode not set

RFAL_ERR_TIMEOUT : If there is no response

◆ rfalFieldOff()

ReturnCode rfalFieldOff (void)

Turns the Field Off

Returns

RFAL_ERR_NONE : Field turned Off

Examples

exampleRfalNfca.c, and **exampleRfalPoller.c**.

◆ rfalFieldOnAndStartGT()

ReturnCode rfalFieldOnAndStartGT (void)

Turns the Field On, performing Initial Collision Avoidance

After Field On, if GT was set before, it starts the GT timer to be used on the following communications.

Returns

RFAL_ERR_RF_COLLISION : External field detected

RFAL_ERR_NONE : Field turned On

Examples

exampleRfalNfca.c, and **exampleRfalPoller.c**.

◆ rfalGetBitRate()

```
ReturnCode rfalGetBitRate ( rfalBitRate *  txBR,
                            rfalBitRate *  rxBR
                          )
```

Gets the Tx and Rx current bit rates

If RFAL is not initialized or mode not set the bit rates return will be invalid RFAL_BR_KEEP

Parameters

[out] **txBR** : RFAL's current Tx Bit Rate

[out] **rxBR** : RFAL's current Rx Bit Rate

See also

rfalSetBitRate

rfalBitRate

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or mode not set

RFAL_ERR_NONE : No error

◆ rfalGetErrorHandling()

```
rfalEHandling rfalGetErrorHandling ( void )
```

Gets the error handling mode currently used by the RFAL

Returns

rfalEHandling : Current error handling mode

◆ rfalGetFDTListen()

```
uint32_t rfalGetFDTListen ( void )
```

Gets the Frame Delay Time (FDT) Listen minimum

FDT Listen is the minimum time between a Poll Frame and a Listen Frame FDTx,LISTEN,MIN - Digital 1.1 6.10.1 & 7.9.1 & 8.7.1

Returns

FDT : current FDT value in 1/fc cycles

◆ rfalGetFDTPoll()

uint32_t rfalGetFDTPoll (void)

Gets the current Frame Delay Time (FDT)

FDT Poll is the minimum time following a Poll Frame during which no subsequent Poll Frame can be sent (without a response from the Listener in between) FDTx,PP,MIN - Digital 1.1 6.10.2 & 7.9.2 & 8.7.2

Returns

FDT : current FDT value in 1/fc cycles

◆ rfalGetFeliCaPollStatus()

ReturnCode rfalGetFeliCaPollStatus (void)

Gets the current state of the Felica Poll Request triggered before by **rfalStartFeliCaPoll()**

Returns

RFAL_ERR_NONE : If there is no error

RFAL_ERR_BUSY : Operation ongoing

RFAL_ERR_TIMEOUT : If there is no response

◆ rfalGetGT()

uint32_t rfalGetGT (void)

Gets the current Guard Time (GT)

GT is the minimum time when a device in Listen Mode is exposed to an unmodulated carrier

Returns

GT : Guard Time in 1/fc cycles

◆ rfalGetMode()

rfalMode rfalGetMode (void)

Gets the mode that RFAL is set to operate

See also

rfalMode

Returns

rfalMode : The current RFAL mode

◆ rfalGetObsvMode()

void rfalGetObsvMode (uint8_t * txMode,
uint8_t * rxMode
)

Gets ST25R the current configured observation modes

Parameters

[in] **txMode** : the current observation mode configured for transmission

[in] **rxMode** : the current observation mode configured for reception

◆ rfalGetTransceiveRSSI()

ReturnCode rfalGetTransceiveRSSI (uint16_t * rssi)

Gets the RSSI value of the last executed Transceive in mV

Parameters

[out] **rssi** : RSSI value

Returns

RFAL_ERR_NOTSUPP : Feature not supported

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error

◆ rfalGetTransceiveState()

rfalTransceiveState rfalGetTransceiveState (void)

Gets current Transceive internal State

Returns

rfalTransceiveState : the current Transceive internal State

◆ rfalGetTransceiveStatus()

ReturnCode rfalGetTransceiveStatus (void)

Gets current Transceive status

Returns

RFAL_ERR_NONE : Transceive done with no error

RFAL_ERR_BUSY : Transceive ongoing

RFAL_ERR_XXXX : Error occurred

RFAL_ERR_TIMEOUT : No response

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PAR : Parity error detected

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_LINK_LOSS : Link Loss - External Field is Off

RFAL_ERR_RF_COLLISION : Collision detected

RFAL_ERR_IO : Internal error

Examples

exampleRfalPoller.c.

◆ rfalInitialize()

ReturnCode rfalInitialize (void)

Initializes RFAL layer and the ST25R Ensures that ST25R is properly connected and returns error if any problem is detected

Warning

rfalAnalogConfigInitialize() should be called before so that the Analog config table has been previously initialized.

Returns

RFAL_ERR_HW_MISMATCH : Expected HW do not match or communication error

RFAL_ERR_NONE : No error

Examples

exampleRfalNfca.c, and **exampleRfalPoller.c**.

◆ rfalIsExtFieldOn()

bool rfalIsExtFieldOn (void)

Checks if external field (other peer/device) is on/detected

Returns

true External field is On

false No external field is detected

◆ rfalIsGTExpired()

bool rfalIsGTExpired (void)

Checks whether the GT timer has expired

Returns

true : GT has expired or not running

false : GT is still running

◆ rfalISO14443AGetTransceiveAnticollisionFrameStatus()

ReturnCode rfalISO14443AGetTransceiveAnticollisionFrameStatus (void)

This gets the ISO14443A anti-collision frame status.

Returns

RFAL_ERR_NONE if there is no error

◆ rfalISO14443AStartTransceiveAnticollisionFrame()

ReturnCode rfalISO14443AStartTransceiveAnticollisionFrame (uint8_t * buf,

```
                                                    uint8_t *    bytesToSend,
                                                    uint8_t *    bitsToSend,
                                                    uint16_t *   rxLength,
                                                    uint32_t     fwt
                                                  )
```

This starts the transceive of an ISO14443A anti-collision frame.

Note

Anticollision is sent without CRC

Parameters

[in,out]	**buf**	: reference to ANTICOLLISION command (with known UID if any) to be sent (also out param) reception will be place on this buf after bytesToSend buffer must be capable of holding a whole Anticollison frame (**rfalNfcaSelReq**)
[in,out]	**bytesToSend**	reference number of full bytes to be sent (including CMD byte and SEL_PAR) if a collision occurs will contain the number of clear bytes
[in,out]	**bitsToSend**	: reference to number of bits (0-7) to be sent; and received (also out param) if a collision occurs will indicate the number of clear bits (also out param)
[out]	**rxLength**	: reference to the return the received length in bits
[in]	**fwt**	: Frame Waiting Time in 1/fc

Returns

RFAL_ERR_NONE if there is no error

◆ rfalISO14443ATransceiveAnticollisionFrame()

```
ReturnCode rfalISO14443ATransceiveAnticollisionFrame ( uint8_t *    buf,
                                                       uint8_t *    bytesToSend,
                                                       uint8_t *    bitsToSend,
                                                       uint16_t *   rxLength,
                                                       uint32_t     fwt
                                                     )
```

This is used to perform ISO14443A anti-collision.

Note

Anticollision is sent without CRC

Parameters

[in,out]	**buf**	: reference to ANTICOLLISION command (with known UID if any) to be sent (also out param) reception will be place on this buf after bytesToSend buffer must be capable of holding a whole Anticollison frame (**rfalNfcaSelReq**)
[in,out]	**bytesToSend**	reference number of full bytes to be sent (including CMD byte and SEL_PAR) if a collision occurs will contain the number of clear bytes
[in,out]	**bitsToSend**	: reference to number of bits (0-7) to be sent; and received (also out param) if a collision occurs will indicate the number of clear bits (also out param)
[out]	**rxLength**	: reference to the return the received length in bits
[in]	**fwt**	: Frame Waiting Time in 1/fc

Returns

RFAL_ERR_NONE if there is no error

◆ rfalISO14443ATransceiveShortFrame()

```
ReturnCode rfalISO14443ATransceiveShortFrame ( rfal14443AShortFrameCmd  txCmd,
                                               uint8_t *                 rxBuf,
                                               uint8_t                   rxBufLen,
                                               uint16_t *                rxRcvdLen,
                                               uint32_t                  fwt
                                             )
```

Sends REQA or WUPA to detect if there is any PICC in the field

Parameters

[in]	**txCmd**	Command to be sent: 0x52 WUPA / ALL_REQ 0x26 REQA / SENS_REQ
[out]	**rxBuf**	: buffer to place the response
[in]	**rxBufLen**	: length of rxBuf in bytes
[out]	**rxRcvdLen**	received length in bits
[in]	**fwt**	: Frame Waiting Time in 1/fc

Warning

If fwt is set to RFAL_FWT_NONE it will make endlessly for a response, which on a blocking method may not be the desired usage

Returns

RFAL_ERR_NONE : If there is response

RFAL_ERR_TIMEOUT : If there is no response

RFAL_ERR_RF_COLLISION : A collision was detected

◆ rfalISO15693TransceiveAnticollisionFrame()

```
ReturnCode rfalISO15693TransceiveAnticollisionFrame ( uint8_t *   txBuf,
                                                      uint8_t     txBufLen,
                                                      uint8_t *   rxBuf,
                                                      uint8_t     rxBufLen,
                                                      uint16_t *  actLen
                                                    )
```

This send the Anticollision|Inventory frame (INVENTORY_REQ)

Warning

rxBuf must be able to contain the payload and CRC

Parameters

- [in] **txBuf** : Buffer where outgoing message is located
- [in] **txBufLen** : Length of the outgoing message in bytes
- [out] **rxBuf** : Buffer where incoming message will be placed
- [in] **rxBufLen** : Maximum length of the incoming message in bytes
- [out] **actLen** : Actual received length in bits

Returns

RFAL_ERR_NONE : Transceive done with no error

RFAL_ERR_WRONG_STATE : RFAL not initialized or mode not set

RFAL_ERR_IO : Internal error

◆ rfalISO15693TransceiveEOF()

```
ReturnCode rfalISO15693TransceiveEOF ( uint8_t *   rxBuf,
                                       uint16_t    rxBufLen,
                                       uint16_t *  actLen
                                     )
```

This is method sends an ISO15693 (EoF) used for a Write operation

Warning

rxBuf must be able to contain the payload and CRC

Parameters

[out] **rxBuf** : Buffer where incoming message will be placed

[in] **rxBufLen** : Maximum length of the incoming message in bytes

[out] **actLen** : Actual received length in bytes

Returns

RFAL_ERR_NONE : Transceive done with no error

RFAL_ERR_IO : Internal error

◆ rfalISO15693TransceiveEOFAnticollision()

```
ReturnCode rfalISO15693TransceiveEOFAnticollision ( uint8_t *   rxBuf,
                                                    uint8_t     rxBufLen,
                                                    uint16_t *  actLen
                                                  )
```

This sends the Anticollision|Inventory EOF used as a slot marker

Warning

rxBuf must be able to contain the payload and CRC

Parameters

[out] **rxBuf** : Buffer where incoming message will be placed

[in] **rxBufLen** : Maximum length of the incoming message in bytes

[out] **actLen** : Actual received length in bits

Returns

RFAL_ERR_NONE : Transceive done with no error

RFAL_ERR_WRONG_STATE : RFAL not initialized or mode not set

RFAL_ERR_IO : Internal error

◆ rfalIsTransceiveInRx()

```
bool rfalIsTransceiveInRx ( void )
```

Checks if Transceive is in Reception state

Returns

true Transmission done/reception ongoing

false Not in reception state

◆ rfalIsTransceiveInTx()

bool rfalIsTransceiveInTx (void)

Checks if Transceive is in Transmission state

Returns

true Transmission ongoing

false Not in transmission state

◆ rfalIsTransceiveSubcDetected()

bool rfalIsTransceiveSubcDetected (void)

Checks on the last executed Transceive a subcarrier was detected

Returns

true Subcarrier was detected

false No subcarrier detected | Not supported

◆ rfalListenGetState()

rfalLmState rfalListenGetState (bool * dataFlag,
rfalBitRate * lastBR
)

Sets the new state of the Listen Mode and applies the necessary changes on the RF Chip

Parameters

[out] **dataFlag** indicates that Listen Mode has rcvd data and caller must process it. The received message is located at the rxBuf passed on **rfalListenStart()**. **rfalListenSetState()** will clear this flag if NULL output parameter will no be written/returned

[out] **lastBR** bit rate detected of the last initiator request if NULL output parameter will no be written/returned

Returns

rfalLmState RFAL_LM_STATE_NOT_INIT : LM not initialized properly Any Other : LM State

◆ rfalListenSetState()

```
ReturnCode rfalListenSetState ( rfalLmState newSt )
```

Sets the new state of the Listen Mode and applies the necessary changes on the RF Chip

Parameters

[in] **newSt** : New state to go to

Returns

RFAL_ERR_WRONG_STATE : Not initialized properly

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : Done with no error

◆ rfalListenSleepStart()

```
ReturnCode rfalListenSleepStart ( rfalLmState sleepSt,
                                  uint8_t *   rxBuf,
                                  uint16_t    rxBufLen,
                                  uint16_t *  rxLen
                                )
```

Parameters

[in] **sleepSt** : sleep state to be set

[in] **rxBuf** : buffer to place incoming data

[in] **rxBufLen** : length in bits of rxBuf

[in] **rxLen** : pointer to write the data length in bits placed into rxBuf

Returns

RFAL_ERR_WRONG_STATE : Not initialized properly

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : Done with no error

◆ rfalListenStart()

```
ReturnCode rfalListenStart ( uint32_t                lmMask,
                             const rfalLmConfPA *    confA,
                             const rfalLmConfPB *    confB,
                             const rfalLmConfPF *    confF,
                             uint8_t *               rxBuf,
                             uint16_t                rxBufLen,
                             uint16_t *              rxLen
                           )
```

Configures RF Chip to go into listen mode enabling the given technologies

Parameters

- [in] **lmMask** mask with the enabled/disabled listen modes use: RFAL_LM_MASK_NFCA ; RFAL_LM_MASK_NFCB ; RFAL_LM_MASK_NFCF ; RFAL_LM_MASK_ACTIVE_P2P
- [in] **confA** pointer to Passive A configurations (NULL if disabled)
- [in] **confB** pointer to Passive B configurations (NULL if disabled)
- [in] **confF** pointer to Passive F configurations (NULL if disabled)
- [in] **rxBuf** buffer to place incoming data
- [in] **rxBufLen** length in bits of rxBuf
- [in] **rxLen** pointer to write the data length in bits placed into rxBuf

Returns

RFAL_ERR_WRONG_STATE : Not initialized properly

RFAL_ERR_PARAM : Invalid parametere mask

RFAL_ERR_NONE : Done with no error

◆ rfalListenStop()

```
ReturnCode rfalListenStop ( void )
```

Disables the listen mode on the RF Chip

Warning

the listen mode will be disabled immediately on the RFchip regardless of any ongoing operations like Transceive

Returns

RFAL_ERR_NONE : Done with no error

◆ rfalLowPowerModeStart()

ReturnCode rfalLowPowerModeStart (**rfalLpMode** mode)

Sets the RF Chip in Low Power Mode. In this mode the RF Chip is placed in Low Power Mode, similar to Wake-up mode but no operation nor period measurement is performed. Mode must be terminated by **rfalLowPowerModeStop()**

Parameters

[in] **mode** : low power mode to be set

Returns

RFAL_ERR_WRONG_STATE : Not initialized properly

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : Done with no error

◆ rfalLowPowerModeStop()

ReturnCode rfalLowPowerModeStop (void)

Stops the Low Power Mode re-enabling the device

Returns

RFAL_ERR_WRONG_STATE : Not initialized properly

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : Done with no error

◆ rfalSetBitRate()

ReturnCode rfalSetBitRate (**rfalBitRate** txBR,
rfalBitRate rxBR
)

Sets the Tx and Rx bit rates with the given values The bit rate change is applied on the RF chip remaining in the same mode previous defined with **rfalSetMode()**

If no mode is defined bit rates will not be applied and an error is returned

Parameters

[in] **txBR** : transmit bit rate

[in] **rxBR** : receive bit rate

See also

rfalSetMode

rfalMode

rfalBitRate

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NOT_IMPLEMENTED : Mode not implemented

RFAL_ERR_NONE : No error

◆ rfalSetErrorHandling()

void rfalSetErrorHandling (**rfalEHandling** eHandling)

Sets the error handling mode to be used by the RFAL

Parameters

[in] **eHandling** : the error handling mode

◆ rfalSetFDTListen()

void rfalSetFDTListen (uint32_t FDTListen)

Sets the Frame Delay Time (FDT) Listen minimum to be used on the following communications.

FDT Listen is the minimum time between a Poll Frame and a Listen Frame FDTx,LISTEN,MIN - Digital 1.1 6.10.1 & 7.9.1 & 8.7.1

Parameters

[in] **FDTListen** : Frame Delay Time in 1/fc cycles

◆ rfalSetFDTPoll()

void rfalSetFDTPoll (uint32_t FDTPoll)

Sets the Frame Delay Time (FDT) to be used on the following communications.

FDT Poll is the minimum time following a Poll Frame during which no subsequent Poll Frame can be sent (without a response from the Listener in between) FDTx,PP,MIN - Digital 1.1 6.10.2 & 7.9.2 & 8.7.2

Parameters

[in] **FDTPoll** : Frame Delay Time in 1/fc cycles

◆ rfalSetGT()

void rfalSetGT (uint32_t GT)

Sets the Guard Time (GT) to be used on the following communications.

GT is the minimum time when a device in Listen Mode is exposed to an unmodulated carrier

Parameters

[in] **GT** : Guard Time in 1/fc cycles RFAL_GT_NONE if no GT should be applied

◆ rfalSetLmEonCallback()

void rfalSetLmEonCallback (**rfalLmEonCallback** pFunc)

Sets a callback upon External Field On detected while in Passive Listen Mode

Warning

callabck available only on applicable devices, supporting Passive Listen Mode

Parameters

[in] **pFunc** : method pointer for the LM EON callback

◆ rfalSetMode()

ReturnCode rfalSetMode (**rfalMode** mode,
rfalBitRate txBR,
rfalBitRate rxBR
)

Sets the mode that RFAL will operate on the following communications. Proper initializations will be performed on the ST25R

Warning

bit rate value RFAL_BR_KEEP is not allowed, only in **rfalSetBitRate()**

the mode will be applied immediately on the RFchip regardless of any ongoing operations like Transceive, ListenMode

Parameters

[in] **mode** : mode for the RFAL/RFchip to perform

[in] **txBR** : transmit bit rate

[in] **rxBR** : receive bit rate

See also

rfalIsGTExpired

rfalMode

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error

◆ rfalSetObsvMode()

```
void rfalSetObsvMode ( uint32_t  txMode,
                       uint32_t  rxMode
                     )
```

Sets ST25R observation modes for RF debug purposes

Parameters

[in] **txMode** : the observation mode to be used during transmission

[in] **rxMode** : the observation mode to be used during reception

Warning

The Observation Mode is an advanced feature and should be set according to the documentation of the part number in use. Please refer to the corresponding Datasheet or Application Note(s)

◆ rfalSetPostTxRxCallback()

```
void rfalSetPostTxRxCallback ( rfalPostTxRxCallback  pFunc )
```

Sets a callback for the driver to call after a Transceive

Parameters

[in] **pFunc** : method pointer for the Post Tx callback

◆ rfalSetPreTxRxCallback()

void rfalSetPreTxRxCallback (**rfalPreTxRxCallback** pFunc)

Sets a callback for the driver to call before a Transceive

Parameters

[in] **pFunc** : method pointer for the Pre Tx callback

◆ rfalSetRegulators()

ReturnCode rfalSetRegulators (uint8_t regulation)

Sets the ST25R regulators to manual setting

Parameters

[in] **regulation** : Regulated voltage setting (ST25R device specific)

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized

RFAL_ERR_PARAM : Invalid regulator setting

RFAL_ERR_NONE : No error

◆ rfalSetSyncTxRxCallback()

void rfalSetSyncTxRxCallback (**rfalSyncTxRxCallback** pFunc)

Sets a callback for the driver to execute in order to Syncronize actual transmission start. If the callback is set TxRx will hold until Sync callback returns true.

Parameters

[in] **pFunc** : method pointer for the Sync Tx callback

◆ rfalSetUpperLayerCallback()

void rfalSetUpperLayerCallback (**rfalUpperLayerCallback** pFunc)

Sets a callback for the driver to call when an event has occurred that may require the system to be notified

Parameters

[in] **pFunc** : method pointer for the upper layer callback

◆ rfalStartFeliCaPoll()

```
ReturnCode rfalStartFeliCaPoll ( rfalFeliCaPollSlots  slots,
                                 uint16_t            sysCode,
                                 uint8_t             reqCode,
                                 rfalFeliCaPollRes *  pollResList,
                                 uint8_t             pollResListSize,
                                 uint8_t *           devicesDetected,
                                 uint8_t *           collisionsDetected
                               )
```

Triggers a Poll Request and all Poll Responses will be collected according to the given nuber of slots

Parameters

[in] **slots** : number of slots for the Poll Request

[in] **sysCode** : system code (SC) for the Poll Request

[in] **reqCode** : request code (RC) for the Poll Request

[out] **pollResList** : list of all responses

[in] **pollResListSize** : number of responses that can be placed in pollResList

[out] **devicesDetected** : number of cards found

[out] **collisionsDetected** number of collisions detected

Returns

RFAL_ERR_NONE : If there is no error

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_WRONG_STATE : RFAL not initialized or mode not set

◆ rfalStartTransceive()

```
ReturnCode rfalStartTransceive ( const rfalTransceiveContext *  ctx )
```

Set the context that will be used for the following Transceive Output and input buffers have to be passed and all other details prior to the Transceive itself has been started

This method only sets the context. Once set, rfalWorker has to be executed until is done

Parameters

[in] **ctx** : the context for the following Transceive

See also

rfalWorker

rfalGetTransceiveStatus

Returns

RFAL_ERR_NONE : Done with no error

RFAL_ERR_WRONG_STATE : Not initialized properly

RFAL_ERR_PARAM : Invalid parameter or configuration

Examples

exampleRfalPoller.c.

◆ rfalTransceiveBlockingRx()

ReturnCode rfalTransceiveBlockingRx (void)

This is method executes the reception of an ongoing Transceive triggered before by **rfalTransceiveBlockingTx()**

Returns

RFAL_ERR_NONE : Transceive done with no error

RFAL_ERR_BUSY : Transceive ongoing

RFAL_ERR_XXXX : Error occurred

RFAL_ERR_TIMEOUT : No response

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PAR : Parity error detected

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_LINK_LOSS : Link Loss - External Field is Off

RFAL_ERR_RF_COLLISION : Collision detected

RFAL_ERR_IO : Internal error

◆ rfalTransceiveBlockingTx()

```
ReturnCode rfalTransceiveBlockingTx ( uint8_t *   txBuf,
                                      uint16_t    txBufLen,
                                      uint8_t *   rxBuf,
                                      uint16_t    rxBufLen,
                                      uint16_t *  actLen,
                                      uint32_t    flags,
                                      uint32_t    fwt
                                    )
```

This is method triggers a Transceive and executes it blocking until the Tx has been completed

Parameters

- [in] **txBuf** : Buffer where outgoing message is located
- [in] **txBufLen** : Length of the outgoing message in bytes
- [out] **rxBuf** : Buffer where incoming message will be placed
- [in] **rxBufLen** : Maximum length of the incoming message in bytes
- [out] **actLen** : Actual received length in bits
- [in] **flags** : TransceiveFlags indication special handling
- [in] **fwt** : Frame Waiting Time in 1/fc

Returns

RFAL_ERR_NONE : Transceive done with no error

RFAL_ERR_BUSY : Transceive ongoing

RFAL_ERR_XXXX : Error occurred

RFAL_ERR_LINK_LOSS : Link Loss - External Field is Off

RFAL_ERR_RF_COLLISION : Collision detected

RFAL_ERR_IO : Internal error

◆ rfalTransceiveBlockingTxRx()

```
ReturnCode rfalTransceiveBlockingTxRx ( uint8_t *   txBuf,
                                        uint16_t    txBufLen,
                                        uint8_t *   rxBuf,
```

```
                                    uint16_t    rxBufLen,
                                    uint16_t *  actLen,
                                    uint32_t    flags,
                                    uint32_t    fwt
                                  )
```

This is method triggers a Transceive and executes it blocking until it has been completed

Parameters

[in]	**txBuf**	: Buffer where outgoing message is located
[in]	**txBufLen**	: Length of the outgoing message in bytes
[out]	**rxBuf**	: Buffer where incoming message will be placed
[in]	**rxBufLen**	: Maximum length of the incoming message in bytes
[out]	**actLen**	: Actual received length in bytes
[in]	**flags**	: TransceiveFlags indication special handling
[in]	**fwt**	: Frame Waiting Time in 1/fc

Returns

RFAL_ERR_NONE : Transceive done with no error

RFAL_ERR_BUSY : Transceive ongoing

RFAL_ERR_XXXX : Error occurred

RFAL_ERR_TIMEOUT : No response

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PAR : Parity error detected

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_LINK_LOSS : Link Loss - External Field is Off

RFAL_ERR_RF_COLLISION : Collision detected

RFAL_ERR_IO : Internal error

◆ rfalWakeUpModeGetInfo()

```
ReturnCode rfalWakeUpModeGetInfo ( bool                force,
                                   rfalWakeUpInfo *    info
                                 )
```

Returns the current information while Wake-up mode is running

Warning

The information returned will only be updated in case force is enabled, or if an event IRQ has happen. Otherwise the info will be filled with zeros.

Parameters

`[in]` **force** : Force info update info by retrieving it from device

`[out]` **info** : pointer where WU mode info is to be stored

Returns

RFAL_ERR_WRONG_STATE : Not initialized properly

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : Done with no error

◆ rfalWakeUpModeHasWoke()

bool rfalWakeUpModeHasWoke (void)

Returns true if the Wake-Up mode is enabled and it has already received the indication from the RF Chip that the surrounding environment has changed and flagged at least one wake-Up interrupt

Returns

true : Wake-Up mode enabled and has received a wake-up IRQ

false : no Wake-Up IRQ has been received

◆ rfalWakeUpModeIsEnabled()

bool rfalWakeUpModeIsEnabled (void)

Returns true if the Wake-Up mode is enabled and it has already completed its starting up sequence. When the option to obtain a reference value from WU is enabled, the startup sequence takes longer. Otherwise WU mode is running after rfalWakeUpModeStart

Returns

true : Wake-Up mode enabled

false : Wake-Up mode not enabled

◆ rfalWakeUpModeStart()

ReturnCode rfalWakeUpModeStart (const **rfalWakeUpConfig** * config)

Sets the RF Chip in Low Power Wake-Up Mode according to the given configuration.

Parameters

[in] **config** : Generic Wake-Up configuration provided by lower layers. If NULL will automatically configure the Wake-Up mode

Returns

RFAL_ERR_WRONG_STATE : Not initialized properly

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : Done with no error

◆ rfalWakeUpModeStop()

ReturnCode rfalWakeUpModeStop (void)

Stops the Wake-Up Mode

Returns

RFAL_ERR_WRONG_STATE : Not initialized properly

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : Done with no error

◆ rfalWlcPWptIsFodDetected()

bool rfalWlcPWptIsFodDetected (void)

Returns true if the WLC-P WPT is monitored and it has already received the indication from the RF Chip that FOD was detected.

Returns

true : WLC-P WPT is monitored and has identified a FOD

false : no FOD has been identified

◆ rfalWlcPWptIsStopDetected()

bool rfalWlcPWptIsStopDetected (void)

Returns true if the WLC-P WPT is monitored and it has already received the indication from the RF Chip that a WPT Stop sequence was detected.

Returns

true : WLC-P WPT is monitored and has identified a WPT Stop

false : no WPT Stop IRQ has been identified

◆ rfalWlcPWptMonitorStart()

ReturnCode rfalWlcPWptMonitorStart (const **rfalWakeUpConfig** * config)

After WLC-P reaches its WPT state it starts the monitoring for Impedance change and WPT Stop sequeence.

Parameters

[in] **config** : Generic Wake-Up configuration provided by lower layers. If NULL will automatically configure the WLC-P WPT Phase

Warning

several parameters held in config will be overwritten by the driver with the appropriate settings for WPT monitoring.

Returns

RFAL_ERR_WRONG_STATE : Not initialized properly

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : Done with no error

◆ rfalWlcPWptMonitorStop()

ReturnCode rfalWlcPWptMonitorStop (void)

Stops the monitoring of WLC-P WPT Phase

Returns

RFAL_ERR_WRONG_STATE : Not initialized properly

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : Done with no error

◆ rfalWorker()

void rfalWorker (void)

This runs RFAL layer, which drives the actual Transceive procedure It MUST be executed frequently in order to execute the RFAL internal states and perform the requested operations

Examples

exampleRfalPoller.c.

Generated on Fri Mar 14 2025 11:45:09 for RFAL by doxygen 1.9.5

RFAL

Main Page

Modules

Files

Examples

Data Structures

Data Structure Index

Data Fields

Data Fields

rfalTransceiveContext Struct Reference


```
#include <rfal_rf.h>
```

Data Fields

uint8_t * **txBuf**

uint16_t	**txBufLen**
uint8_t *	**rxBuf**
uint16_t	**rxBufLen**
uint16_t *	**rxRcvdLen**
uint32_t	**flags**
uint32_t	**fwt**

Detailed Description

Struct that holds all context to be used on a Transceive

Examples

exampleRfalPoller.c.

Field Documentation

◆ flags

uint32_t rfalTransceiveContext::flags

(In) TransceiveFlags indication special handling

◆ fwt

uint32_t rfalTransceiveContext::fwt

(In) Frame Waiting Time in 1/fc

◆ rxBuf

uint8_t* rfalTransceiveContext::rxBuf

(Out) Buffer where incoming message will be placed

◆ rxBufLen

uint16_t rfalTransceiveContext::rxBufLen

(In) Maximum length of the incoming message in bits

◆ rxRcvdLen

uint16_t* rfalTransceiveContext::rxRcvdLen

(Out) Actual received length in bits

◆ txBuf

uint8_t* rfalTransceiveContext::txBuf

(In) Buffer where outgoing message is located

◆ txBufLen

uint16_t rfalTransceiveContext::txBufLen

(In) Length of the outgoing message in bits

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

- **rfal_rf.h**

rfalLmConfPA Struct Reference


```
#include <rfal_rf.h>
```

Data Fields

rfalLmNfcidLen	**nfcidLen**
uint8_t	**nfcid** [**RFAL_NFCID1_TRIPLE_LEN**]
uint8_t	**SENS_RES** [**RFAL_LM_SENS_RES_LEN**]
uint8_t	**SEL_RES**

Detailed Description

RFAL Listen Mode Passive A configs

Field Documentation

◆ nfcid

uint8_t rfalLmConfPA::nfcid[**RFAL_NFCID1_TRIPLE_LEN**]

NFCID

◆ nfcidLen

rfalLmNfcidLen rfalLmConfPA::nfcidLen

NFCID Len (4, 7 or 10 bytes)

◆ SEL_RES

uint8_t rfalLmConfPA::SEL_RES

SEL_RES (SAK) with complete NFCID1 (UID)

◆ SENS_RES

uint8_t rfalLmConfPA::SENS_RES[**RFAL_LM_SENS_RES_LEN**]

NFC-106k; SENS_REQ Response

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

- **rfal_rf.h**

rfalLmConfPB Struct Reference


```
#include <rfal_rf.h>
```

Data Fields

uint8_t **SENSB_RES** [**RFAL_LM_SENSB_RES_LEN**]

Detailed Description

RFAL Listen Mode Passive B configs

Field Documentation

◆ SENSB_RES

uint8_t rfalLmConfPB::SENSB_RES[**RFAL_LM_SENSB_RES_LEN**]

SENSF_RES

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

- rfal_rf.h

rfalLmConfPF Struct Reference

RFAL » RFAL-HAL » RF

```
#include <rfal_rf.h>
```

Data Fields

uint8_t	SC [RFAL_LM_SENSF_SC_LEN]
uint8_t	SENSF_RES [RFAL_LM_SENSF_RES_LEN]

Detailed Description

RFAL Listen Mode Passive F configs

Field Documentation

◆ SC

uint8_t rfalLmConfPF::SC[**RFAL_LM_SENSF_SC_LEN**]

System Code to listen for

◆ SENSF_RES

uint8_t rfalLmConfPF::SENSF_RES[**RFAL_LM_SENSF_RES_LEN**]

SENSF_RES

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

- **rfal_rf.h**

ST25R200

RFAL ST25R200 Driver. More...

Collaboration diagram for ST25R200:

Modules

ST25R200_Driver
RFAL ST25R200 Driver.

ST25R200_COM
RFAL ST25R200 Communications.

ST25R200_IRQ
RFAL ST25R200 IRQ.

Detailed Description

RFAL

Main Page

Data Structures

Files

ST25R200_Driver

RFAL ST25R200 Driver. More...

Collaboration diagram for ST25R200_Driver:

Macros

#define **ST25R200_CMD_SET_DEFAULT** 0x60U

#define **ST25R200_CMD_STOP** 0x62U

#define **ST25R200_CMD_CLEAR_FIFO** 0x64U

#define **ST25R200_CMD_CLEAR_RXGAIN** 0x66U

#define **ST25R200_CMD_ADJUST_REGULATORS** 0x68U

#define **ST25R200_CMD_TRANSMIT** 0x6AU

#define **ST25R200_CMD_TRANSMIT_EOF** 0x6CU

#define **ST25R200_CMD_MASK_RECEIVE_DATA** 0x70U

#define **ST25R200_CMD_UNMASK_RECEIVE_DATA** 0x72U

#define **ST25R200_CMD_CALIBRATE_WU** 0x74U

#define **ST25R200_CMD_CLEAR_WU_CALIB** 0x76U

#define **ST25R200_CMD_MEASURE_WU** 0x78U

#define **ST25R200_CMD_MEASURE_IQ** 0x7AU

#define **ST25R200_CMD_SENSE_RF** 0x7CU

```
#define ST25R200_CMD_TRANSPARENT_MODE 0xE0U
#define ST25R200_CMD_START_GP_TIMER 0xE2U
#define ST25R200_CMD_START_WUT 0xE4U
#define ST25R200_CMD_START_MRT 0xE6U
#define ST25R200_CMD_START_NRT 0xE8U
#define ST25R200_CMD_STOP_NRT 0xEAU
#define ST25R200_CMD_TEST_ACCESS 0xFCU
#define ST25R200_BR_DO_NOT_SET 0xFFU
#define ST25R200_BR_106_26 0x00U
#define ST25R200_BR_212 0x01U
#define ST25R200_BR_424_53 0x02U
#define ST25R200_BR_848 0x03U
#define ST25R200_REG_DROP_200 0U
#define ST25R200_REG_DROP_250 1U
#define ST25R200_REG_DROP_300 2U
#define ST25R200_REG_DROP_350 3U
#define ST25R200_REG_DROP_400 4U
#define ST25R200_REG_DROP_450 5U
#define ST25R200_REG_DROP_500 6U
#define ST25R200_REG_DROP_550 7U
#define ST25R200_REG_DROP_DO_NOT_SET 0xFFU
```

#define **ST25R200_REG_LEN** 1U

#define **ST25R200_CMD_LEN** 1U

#define **ST25R200_FIFO_DEPTH** 256U

#define **ST25R200_WRITE_MODE** (0U << 7)

#define **ST25R200_READ_MODE** (1U << 7)

#define **ST25R200_CMD_MODE** **ST25R200_WRITE_MODE**

#define **ST25R200_FIFO_ACCESS** (0x5FU)

#define **st25r200TxRxOn**() **st25r200SetRegisterBits**(**ST25R200_REG_OPERATION**, (ST25R200_REG_OPERATION_rx_en | ST25R200_REG_OPERATION_tx_en))

#define **st25r200TxRxOff**() **st25r200ClrRegisterBits**(**ST25R200_REG_OPERATION**, (ST25R200_REG_OPERATION_rx_en | ST25R200_REG_OPERATION_tx_en))

#define **st25r200TxOff**() **st25r200ClrRegisterBits**(**ST25R200_REG_OPERATION**, ST25R200_REG_OPERATION_tx_en)

#define **st25r200IsGPTRunning**() **st25r200CheckReg**(**ST25R200_REG_STATUS**, ST25R200_REG_STATUS_gpt_on, ST25R200_REG_STATUS_gpt_on)

#define **st25r200IsTxEnabled**() **st25r200CheckReg**(**ST25R200_REG_OPERATION**, ST25R200_REG_OPERATION_tx_en, ST25R200_REG_OPERATION_tx_en)

#define **st25r200IsNRTinEMV**() **st25r200CheckReg**(**ST25R200_REG_NRT_GPT_CONF**, ST25R200_REG_NRT_GPT_CONF_nrt_emd, ST25R200_REG_NRT_GPT_CONF_nrt_emd_on)

#define **st25r200IsLastFIFOComplete**() **st25r200CheckReg**(**ST25R200_REG_FIFO_STATUS2**, ST25R200_REG_FIFO_STATUS2_fifo_lb_mask, 0)

#define **st25r200IsOscOn**() **st25r200CheckReg**(**ST25R200_REG_OPERATION**, ST25R200_REG_OPERATION_en, ST25R200_REG_OPERATION_en)

#define **st25r200IsTxOn**() **st25r200CheckReg**(**ST25R200_REG_OPERATION**, ST25R200_REG_OPERATION_tx_en, ST25R200_REG_OPERATION_tx_en)

Functions

ReturnCode	**st25r200Initialize** (void) Initialise ST25R200 driver. More...
void	**st25r200Deinitialize** (void) Deinitialize ST25R200 driver. More...
ReturnCode	**st25r200OscOn** (void) Turn on Oscillator and Regulator. More...
ReturnCode	**st25r200SetBitrate** (uint8_t txRate, uint8_t rxRate) Sets the bitrate. More...
ReturnCode	**st25r200AdjustRegulators** (uint8_t drop, uint16_t *result) Adjusts supply regulators according to the current supply voltage. More...
ReturnCode	**st25r200SetRegulators** (uint8_t regulation) Sets supply regulators. More...
ReturnCode	**st25r200MeasureIQ** (int8_t *resI, int8_t *resQ) Measure I/Q. More...
ReturnCode	**st25r200MeasureCombinedIQ** (uint8_t *res) Measure Combined I/Q. More...
ReturnCode	**st25r200MeasureI** (uint8_t *res) Measure I. More...
ReturnCode	**st25r200MeasureQ** (uint8_t *res) Measure Q. More...
ReturnCode	**st25r200CalibrateWU** (uint8_t *resI, uint8_t *resQ) Calibrate WU. More...
ReturnCode	**st25r200ClearCalibration** (void) Clear WU Calibration. More...

ReturnCode	**st25r200MeasureWU** (uint8_t *resI, uint8_t *resQ) Measure WU. More...
bool	**st25r200IsExtFieldOn** (void) Check External Filed. More...
uint32_t	**st25r200GetNoResponseTime** (void) Get NRT time. More...
ReturnCode	**st25r200SetNoResponseTime** (uint32_t nrt) Set NRT time. More...
ReturnCode	**st25r200SetStartNoResponseTimer** (uint32_t nrt) Set and Start NRT. More...
void	**st25r200SetGPTime** (uint16_t gpt) Set GPT time. More...
ReturnCode	**st25r200SetStartGPTimer** (uint16_t gpt, uint8_t trigger) Set and Start GPT. More...
void	**st25r200SetNumTxBits** (uint16_t nBits) Sets the number Tx Bits. More...
uint16_t	**st25r200GetNumFIFOBytes** (void) Get amount of bytes in FIFO. More...
uint8_t	**st25r200GetNumFIFOLastBits** (void) Get amount of bits of the last FIFO byte if incomplete. More...
ReturnCode	**st25r200PerformCollisionAvoidance** (void) Perform Collision Avoidance. More...
bool	**st25r200CheckChipID** (uint8_t *rev) Check Identity. More...
ReturnCode	**st25r200GetRegsDump** (uint8_t *resRegDump, uint8_t *sizeRegDump) Register Dump. More...

bool	**st25r200IsCmdValid** (uint8_t cmd) Check if command is valid. More...
ReturnCode	**st25r200ExecuteCommandAndGetResult** (uint8_t cmd, uint8_t resReg, uint8_t tOut, uint8_t *result) Executes a direct command and returns the result. More...
ReturnCode	**st25r200GetRSSI** (uint16_t *iRssi, uint16_t *qRssi) Gets the RSSI values. More...
ReturnCode	**st25r200SetAntennaMode** (bool single, bool rfiox) Set Antenna mode. More...

Detailed Description

Macro Definition Documentation

◆ ST25R200_BR_106_26

```
#define ST25R200_BR_106_26   0x00U
```

ST25R200 Bit Rate 106 kbps (fc/128) / 26 kbps(fc/512)

◆ ST25R200_BR_212

```
#define ST25R200_BR_212   0x01U
```

ST25R200 Bit Rate 212 kbps (fc/64)

◆ ST25R200_BR_424_53

```
#define ST25R200_BR_424_53   0x02U
```

ST25R200 Bit Rate 424 kbps (fc/32) / 53 kbps(fc/256)

◆ ST25R200_BR_848

```
#define ST25R200_BR_848   0x03U
```

ST25R200 Bit Rate 848 kbps (fc/16)

◆ ST25R200_BR_DO_NOT_SET

```
#define ST25R200_BR_DO_NOT_SET   0xFFU
```

Indicates not to change this Bit Rate

◆ ST25R200_CMD_ADJUST_REGULATORS

```
#define ST25R200_CMD_ADJUST_REGULATORS   0x68U
```

Adjust regulators

◆ ST25R200_CMD_CALIBRATE_WU

```
#define ST25R200_CMD_CALIBRATE_WU   0x74U
```

Calibrate Wake-up Measurement

◆ ST25R200_CMD_CLEAR_FIFO

```
#define ST25R200_CMD_CLEAR_FIFO   0x64U
```

Clears FIFO, Collision and IRQ status

◆ ST25R200_CMD_CLEAR_RXGAIN

```
#define ST25R200_CMD_CLEAR_RXGAIN   0x66U
```

Clears FIFO, Collision and IRQ status

◆ ST25R200_CMD_CLEAR_WU_CALIB

```
#define ST25R200_CMD_CLEAR_WU_CALIB   0x76U
```

Clear Wake-up Calibratation

◆ ST25R200_CMD_LEN

```
#define ST25R200_CMD_LEN   1U
```

ST25R200 CMD length

◆ ST25R200_CMD_MASK_RECEIVE_DATA

```
#define ST25R200_CMD_MASK_RECEIVE_DATA   0x70U
```

Mask receive data

◆ ST25R200_CMD_MEASURE_IQ

```
#define ST25R200_CMD_MEASURE_IQ   0x7AU
```

Measure I and Q components

◆ ST25R200_CMD_MEASURE_WU

```
#define ST25R200_CMD_MEASURE_WU   0x78U
```

Measure Wake-up I and Q components

◆ ST25R200_CMD_MODE

```
#define ST25R200_CMD_MODE   ST25R200_WRITE_MODE
```

ST25R200 Operation Mode: Direct Command

◆ ST25R200_CMD_SENSE_RF

```
#define ST25R200_CMD_SENSE_RF   0x7CU
```

Sense RF on RFI pins

◆ ST25R200_CMD_SET_DEFAULT

```
#define ST25R200_CMD_SET_DEFAULT   0x60U
```

Puts the chip in default state (same as after power-up)

◆ ST25R200_CMD_START_GP_TIMER

```
#define ST25R200_CMD_START_GP_TIMER   0xE2U
```

Start the general purpose timer

◆ ST25R200_CMD_START_MRT

```
#define ST25R200_CMD_START_MRT   0xE6U
```

Start the mask-receive timer

◆ ST25R200_CMD_START_NRT

```
#define ST25R200_CMD_START_NRT   0xE8U
```

Start the no-response timer

◆ ST25R200_CMD_START_WUT

```
#define ST25R200_CMD_START_WUT   0xE4U
```

Start the wake-up timer

◆ ST25R200_CMD_STOP

```
#define ST25R200_CMD_STOP   0x62U
```

Stops all activities and clears FIFO

◆ ST25R200_CMD_STOP_NRT

```
#define ST25R200_CMD_STOP_NRT   0xEAU
```

Stop No Response Timer

◆ ST25R200_CMD_TEST_ACCESS

```
#define ST25R200_CMD_TEST_ACCESS   0xFCU
```

Enable R/W access to the test registers

◆ ST25R200_CMD_TRANSMIT

```
#define ST25R200_CMD_TRANSMIT   0x6AU
```

Transmit

◆ ST25R200_CMD_TRANSMIT_EOF

```
#define ST25R200_CMD_TRANSMIT_EOF   0x6CU
```

Transmit ISO15693 EOF

◆ ST25R200_CMD_TRANSPARENT_MODE

```
#define ST25R200_CMD_TRANSPARENT_MODE   0xE0U
```

Transparent mode

◆ ST25R200_CMD_UNMASK_RECEIVE_DATA

```
#define ST25R200_CMD_UNMASK_RECEIVE_DATA   0x72U
```

Unmask receive data

◆ ST25R200_FIFO_ACCESS

```
#define ST25R200_FIFO_ACCESS   (0x5FU)
```

ST25R200 FIFO Access

◆ ST25R200_FIFO_DEPTH

```
#define ST25R200_FIFO_DEPTH   256U
```

Depth of FIFO

◆ ST25R200_READ_MODE

```
#define ST25R200_READ_MODE   (1U << 7)
```

ST25R200 Operation Mode: Read

◆ ST25R200_REG_DROP_200

```
#define ST25R200_REG_DROP_200   0U
```

ST25R200 target drop for regulator adjustment: 200mV

◆ ST25R200_REG_DROP_250

```
#define ST25R200_REG_DROP_250   1U
```

ST25R200 target drop for regulator adjustment: 250mV

◆ ST25R200_REG_DROP_300

```
#define ST25R200_REG_DROP_300   2U
```

ST25R200 target drop for regulator adjustment: 300mV

◆ ST25R200_REG_DROP_350

```
#define ST25R200_REG_DROP_350   3U
```

ST25R200 target drop for regulator adjustment: 350mV

◆ ST25R200_REG_DROP_400

```
#define ST25R200_REG_DROP_400   4U
```

ST25R200 target drop for regulator adjustment: 400mV

◆ ST25R200_REG_DROP_450

```
#define ST25R200_REG_DROP_450   5U
```

ST25R200 target drop for regulator adjustment: 450mV

◆ ST25R200_REG_DROP_500

```
#define ST25R200_REG_DROP_500   6U
```

ST25R200 target drop for regulator adjustment: 500mV

◆ ST25R200_REG_DROP_550

```
#define ST25R200_REG_DROP_550   7U
```

ST25R200 target drop for regulator adjustment: 550mV

◆ ST25R200_REG_DROP_DO_NOT_SET

```
#define ST25R200_REG_DROP_DO_NOT_SET   0xFFU
```

Indicates not to change this setting (regd)

◆ ST25R200_REG_LEN

```
#define ST25R200_REG_LEN   1U
```

Number of bytes in a ST25R200 register

◆ ST25R200_WRITE_MODE

```
#define ST25R200_WRITE_MODE   (0U << 7)
```

ST25R200 Operation Mode: Write

◆ st25r200IsGPTRunning

```
#define st25r200IsGPTRunning ( ) st25r200CheckReg( ST25R200_REG_STATUS, ST25R200_REG_STATUS_gpt_on, ST25R200_REG_STATUS_gpt_on )
```

Checks if General Purpose Timer is still running by reading gpt_on flag

◆ st25r200IsLastFIFOComplete

```
#define st25r200IsLastFIFOComplete ( ) st25r200CheckReg( ST25R200_REG_FIFO_STATUS2, ST25R200_REG_FIFO_STATUS2_fifo_lb_mask, 0 )
```

Checks if last FIFO byte is complete

◆ st25r200IsNRTinEMV

```
#define st25r200IsNRTinEMV ( ) st25r200CheckReg( ST25R200_REG_NRT_GPT_CONF, ST25R200_REG_NRT_GPT_CONF_nrt_emd, ST25R200_REG_NRT_GPT_CONF_nrt_emd_on )
```

Checks if NRT is in EMV mode

◆ st25r200IsOscOn

```
#define st25r200IsOscOn ( ) st25r200CheckReg( ST25R200_REG_OPERATION, ST25R200_REG_OPERATION_en, ST25R200_REG_OPERATION_en )
```

Checks if the Oscillator is enabled

◆ st25r200IsTxEnabled

```
#define st25r200IsTxEnabled ( ) st25r200CheckReg( ST25R200_REG_OPERATION, ST25R200_REG_OPERATION_tx_en, ST25R200_REG_OPERATION_tx_en )
```

Checks if Transmitter is enabled (Field On)

◆ st25r200IsTxOn

```
#define st25r200IsTxOn ( ) st25r200CheckReg( ST25R200_REG_OPERATION, ST25R200_REG_OPERATION_tx_en, ST25R200_REG_OPERATION_tx_en )
```

Checks if Transmitter (Field On) is enabled

◆ st25r200TxOff

```
#define st25r200TxOff ( ) st25r200ClrRegisterBits( ST25R200_REG_OPERATION, ST25R200_REG_OPERATION_tx_en )
```

Disables the Transmitter (Field Off)

◆ st25r200TxRxOff

```
#define st25r200TxRxOff ( ) st25r200ClrRegisterBits( ST25R200_REG_OPERATION, (ST25R200_REG_OPERATION_rx_en | ST25R200_REG_OPERATION_tx_en ) )
```

Disables the Transmitter (Field Off) and Receiver

◆ st25r200TxRxOn

```
#define st25r200TxRxOn ( ) st25r200SetRegisterBits( ST25R200_REG_OPERATION, (ST25R200_REG_OPERATION_rx_en | ST25R200_REG_OPERATION_tx_en ) )
```

Enables the Transmitter (Field On) and Receiver

Function Documentation

◆ st25r200AdjustRegulators()

```
ReturnCode st25r200AdjustRegulators ( uint8_t drop,
                                      uint16_t * result
                                    )
```

The power level is measured in maximum load conditions and the regulated voltage reference is set below this level.

The regulated voltages will be set to the result of Adjust Regulators

Parameters

- [in] **drop** : Targeted drop from supply voltage
- [out] **result** : Result of calibration in milliVolts

Returns

RFAL_ERR_IO : Error during communication with ST25R200

RFAL_ERR_NONE : No error

◆ st25r200CalibrateWU()

```
ReturnCode st25r200CalibrateWU ( uint8_t * resI,
                                 uint8_t * resQ
                               )
```

This function executes Wake-up Calibration

Parameters

- [out] **resI** I channel calibration (unsigned)

[out] **resQ** Q channel calibration (unsigned)

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error

◆ st25r200CheckChipID()

bool st25r200CheckChipID (uint8_t * rev)

Checks if the chip ID is as expected.

5 bit IC type code for ST25R200: 00101 The 3 lsb contain the IC revision code

Parameters

[out] **rev** : the IC revision code

Returns

true when IC type is as expected

false otherwise

◆ st25r200ClearCalibration()

ReturnCode st25r200ClearCalibration (void)

This function clears WU calibration

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error

◆ st25r200Deinitialize()

void st25r200Deinitialize (void)

Calling this function deinitializes the ST25R200 driver.

◆ st25r200ExecuteCommandAndGetResult()

ReturnCode st25r200ExecuteCommandAndGetResult (uint8_t cmd,

```
                                                    uint8_t    resReg,
                                                    uint8_t    tOut,
                                                    uint8_t *  result
                                                  )
```

This function executes the direct command given by *cmd* waits for *sleeptime* for I_dct and returns the result read from register *resreg*. The value of cmd is not checked.

Parameters

- [in] **cmd** : direct command to execute
- [in] **resReg** address of the register containing the result
- [in] **tOut** : time in milliseconds to wait before reading the result
- [out] **result** result

Returns

RFAL_ERR_NONE : No error

◆ st25r200GetNoResponseTime()

```
uint32_t st25r200GetNoResponseTime ( void )
```

This returns the last value set on the NRT

Warning

it does not read chip register, just the sw var that contains the last value set before

Returns

the value of the NRT in 64/fc

◆ st25r200GetNumFIFOBytes()

```
uint16_t st25r200GetNumFIFOBytes ( void )
```

Gets the number of bytes currently in the FIFO

Returns

the number of bytes currently in the FIFO

◆ st25r200GetNumFIFOLastBits()

```
uint8_t st25r200GetNumFIFOLastBits ( void )
```

Gets the number of bits of the last FIFO byte if incomplete

Returns

the number of bits of the last FIFO byte if incomplete, 0 if the last byte is complete

◆ st25r200GetRegsDump()

```
ReturnCode st25r200GetRegsDump ( uint8_t * resRegDump,
                                 uint8_t * sizeRegDump
                               )
```

Retrieves all internal registers from ST25R200

Parameters

[out] **resRegDump** : pointer to the struct/buffer where the reg dump will be written

[in,out] **sizeRegDump** : number of registers requested and the ones actually written

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error

◆ st25r200GetRSSI()

```
ReturnCode st25r200GetRSSI ( uint16_t * iRssi,
                             uint16_t * qRssi
                           )
```

This function gets the RSSI value of the previous reception taking into account the gain reductions that were used. RSSI value for both AM and PM channel can be retrieved.

Parameters

[out] **iRssi** the RSSI on the I channel expressed in mV

[out] **qRssi** the RSSI on the Q channel expressed in mV

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error

◆ st25r200Initialize()

ReturnCode st25r200Initialize (void)

This function initialises the ST25R200 driver.

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_HW_MISMATCH : Expected HW do not match or communication error

◆ st25r200IsCmdValid()

bool st25r200IsCmdValid (uint8_t cmd)

Checks if the given command is a valid ST25R200 command

Parameters

[in] **cmd** Command to check

Returns

true if is a valid command

false otherwise

◆ st25r200IsExtFieldOn()

bool st25r200IsExtFieldOn (void)

This function checks if External Field is detected by performing Sense RF procedure

Warning

ST25R200 is not equipped with an External Field Detector block. One can try to assess the presence of an external RF carrier by observing the signal on the RFIs while own field is turned off. WU calibration will be cleared.

Returns

true external signal was sensed

false unable to detect the presence of an external field

◆ st25r200MeasureCombinedIQ()

ReturnCode st25r200MeasureCombinedIQ (uint8_t * res)

This function performs an I/Q measurement and returns the vectorial magnitude

Parameters

[out] **res** 8 bit result of the I Q magnitude Max value is: 127

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error

◆ st25r200MeasureI()

ReturnCode st25r200MeasureI (uint8_t * res)

This function measures I channel

Parameters

[out] **res** I channel

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error

◆ st25r200MeasureIQ()

ReturnCode st25r200MeasureIQ (int8_t * resI,
int8_t * resQ
)

This function performs an I/Q measurement The result is stored on the *resI* and *resQ* parameters.

Parameters

[out] **resI** 8 bit long result of the I channel (signed)

[out] **resQ** 8 bit long result of the Q channel (signed)

Warning

Before executing I|Q Measurement the WU calibration shall be cleared,

See also

st25r200ClearCalibration()

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error

◆ st25r200MeasureQ()

```
ReturnCode st25r200MeasureQ ( uint8_t *  res )
```

This function measures Q channel

Parameters

[out] **res** Q channel

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error

◆ st25r200MeasureWU()

```
ReturnCode st25r200MeasureWU ( uint8_t *  resI,
                               uint8_t *  resQ
                             )
```

This function performs measuremnt as executed during WU mode

Parameters

[out] **resI** 8 bit long result of the I channel (signed)

[out] **resQ** 8 bit long result of the Q channel (signed)

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error

◆ st25r200OscOn()

```
ReturnCode st25r200OscOn ( void  )
```

This function turn on oscillator and regulator and waits for the oscillator to become stable

Returns

RFAL_ERR_SYSTEM: Unable to verify oscilator enable|stable

RFAL_ERR_NONE : No error

◆ st25r200PerformCollisionAvoidance()

ReturnCode st25r200PerformCollisionAvoidance (void)

Performs Collision Avoidance

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_RF_COLLISION : Collision detected

RFAL_ERR_NONE : No collision detected

◆ st25r200SetAntennaMode()

ReturnCode st25r200SetAntennaMode (bool single,
bool rfiox
)

Sets the antenna mode. Differential or single ended antenna mode (RFO1 or RFO2)

Parameters

[in] **single** FALSE differential ; single ended mode

[in] **rfiox** FALSE RFI1/RFO1 ; TRUE RFI2/RFO2

Returns

RFAL_ERR_IO : Internal error

RFAL_ERR_NOTSUPP : Feature not supported

RFAL_ERR_NONE : No error

◆ st25r200SetBitrate()

ReturnCode st25r200SetBitrate (uint8_t txRate,

```
                                    uint8_t  rxRate
                                )
```

This function sets the bitrates for rx and tx

Parameters

txRate : speed is 2^txrate * 106 kb/s 0xff : don't set txrate (ST25R200_BR_DO_NOT_SET)

rxRate : speed is 2^rxrate * 106 kb/s 0xff : don't set rxrate (ST25R200_BR_DO_NOT_SET)

Returns

RFAL_ERR_PARAM: At least one bit rate was invalid

RFAL_ERR_NONE : No error, both bit rates were set

◆ st25r200SetGPTime()

```
void st25r200SetGPTime ( uint16_t  gpt )
```

This function sets the General Purpose Timer time registers

Parameters

[in] **gpt** : general purpose timer timeout in steps of 8/fc (590ns)

◆ st25r200SetNoResponseTime()

```
ReturnCode st25r200SetNoResponseTime ( uint32_t  nrt )
```

This function sets the No Response Time with the given value

Parameters

[in] **nrt** : no response time in steps of 64/fc (4.72us)

Returns

RFAL_ERR_PARAM : Invalid parameter (time is too large)

RFAL_ERR_NONE : No error

◆ st25r200SetNumTxBits()

```
void st25r200SetNumTxBits ( uint16_t  nBits )
```

Sets ST25R200 internal registers with correct number of complete bytes and bits to be sent

Parameters

[in] **nBits** : number of bits to be set/transmitted

◆ st25r200SetRegulators()

ReturnCode st25r200SetRegulators (uint8_t regulation)

Manually sets the regulated voltage setting (rege). The regulated voltages will be set to the manual configuratiom.

Regulation shall be set according to the desired voltage considering the supplied voltage.

Parameters

[in] **regulation** : Regulator setting

Returns

RFAL_ERR_IO : Error during communication with ST25R200

RFAL_ERR_PARAM : Invalid regulator setting

RFAL_ERR_NONE : No error

◆ st25r200SetStartGPTimer()

ReturnCode st25r200SetStartGPTimer (uint16_t gpt,
uint8_t trigger
)

This function sets the General Purpose Timer with the given timeout and immediately starts it ONLY if the trigger source is not set to none.

Parameters

[in] **gpt** : general purpose timer timeout in steps of8/fc (590ns)

[in] **trigger** : no trigger, start of Rx, end of Rx, end of Tx in NFC mode

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error

◆ st25r200SetStartNoResponseTimer()

ReturnCode st25r200SetStartNoResponseTimer (uint32_t nrt)

This function sets the No Response Time with the given value and immediately starts it Used when needs to add more time before timeout without performing Tx

Parameters

[in] **nrt** : no response time in steps of 64/fc (4.72us)

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error

Generated on Fri Mar 14 2025 11:45:09 for RFAL by doxygen 1.9.5

RFAL

ST25R200_COM

RFAL » **RFAL-HAL** » **ST25R200**

RFAL ST25R200 Communications. More...

Collaboration diagram for ST25R200_COM:

Macros

#define **ST25R200_FIFO_STATUS_LEN** 2

```
#define ST25R200_OBS_MODE_LEN 2
#define ST25R200_REG_OPERATION 0x00U
#define ST25R200_REG_GENERAL 0x01U
#define ST25R200_REG_REGULATOR 0x02U
#define ST25R200_REG_TX_DRIVER 0x03U
#define ST25R200_REG_TX_MOD 0x04U
#define ST25R200_REG_TX_RES_MOD 0x05U
#define ST25R200_REG_RX_ANA1 0x06U
#define ST25R200_REG_RX_ANA2 0x07U
#define ST25R200_REG_RX_DIG 0x08U
#define ST25R200_REG_CORR1 0x09U
#define ST25R200_REG_CORR2 0x0AU
#define ST25R200_REG_CORR3 0x0BU
#define ST25R200_REG_CORR4 0x0CU
#define ST25R200_REG_CORR5 0x0DU
#define ST25R200_REG_CORR6 0x0EU
#define ST25R200_REG_DISPLAY1 0x0FU
#define ST25R200_REG_DISPLAY2 0x10U
#define ST25R200_REG_STATUS 0x11U
#define ST25R200_REG_PROTOCOL 0x12U
#define ST25R200_REG_PROTOCOL_TX1 0x13U
```

```
#define ST25R200_REG_PROTOCOL_TX2 0x14U
#define ST25R200_REG_PROTOCOL_TX3 0x15U
#define ST25R200_REG_PROTOCOL_RX1 0x16U
#define ST25R200_REG_PROTOCOL_RX2 0x17U
#define ST25R200_REG_PROTOCOL_RX3 0x18U
#define ST25R200_REG_EMD1 0x19U
#define ST25R200_REG_EMD2 0x1AU
#define ST25R200_REG_MRT_SQT_CONF 0x1BU
#define ST25R200_REG_MRT 0x1CU
#define ST25R200_REG_SQT 0x1DU
#define ST25R200_REG_NRT_GPT_CONF 0x1EU
#define ST25R200_REG_NRT1 0x1FU
#define ST25R200_REG_NRT2 0x20U
#define ST25R200_REG_GPT1 0x21U
#define ST25R200_REG_GPT2 0x22U
#define ST25R200_REG_DISPLAY3 0x23U
#define ST25R200_REG_DISPLAY4 0x24U
#define ST25R200_REG_OVUNSHOOT_CONF 0x25U
#define ST25R200_REG_OVERSHOOT_CONF 0x26U
#define ST25R200_REG_UNDERSHOOT_CONF 0x27U
#define ST25R200_REG_WAKEUP_CONF1 0x28U
```

```
#define ST25R200_REG_WAKEUP_CONF2 0x29U
#define ST25R200_REG_WU_I_CONF 0x2AU
#define ST25R200_REG_WU_I_DELTA 0x2BU
#define ST25R200_REG_WU_I_ADC 0x2CU
#define ST25R200_REG_WU_I_REF 0x2DU
#define ST25R200_REG_WU_I_CAL 0x2EU
#define ST25R200_REG_WU_Q_CONF 0x2FU
#define ST25R200_REG_WU_Q_DELTA 0x30U
#define ST25R200_REG_WU_Q_ADC 0x31U
#define ST25R200_REG_WU_Q_REF 0x32U
#define ST25R200_REG_WU_Q_CAL 0x33U
#define ST25R200_REG_TX_FRAME1 0x34U
#define ST25R200_REG_TX_FRAME2 0x35U
#define ST25R200_REG_FIFO_STATUS1 0x36U
#define ST25R200_REG_FIFO_STATUS2 0x37U
#define ST25R200_REG_COLLISION 0x38U
#define ST25R200_REG_IRQ_MASK1 0x39U
#define ST25R200_REG_IRQ_MASK2 0x3AU
#define ST25R200_REG_IRQ_MASK3 0x3BU
#define ST25R200_REG_IRQ1 0x3CU
#define ST25R200_REG_IRQ2 0x3DU
```

#define **ST25R200_REG_IRQ3** 0x3EU

#define **ST25R200_REG_IC_ID** 0x3FU

Functions

ReturnCode **st25r200ReadRegister** (uint8_t reg, uint8_t *val)
Returns the content of a register within the ST25R200. More...

ReturnCode **st25r200ReadMultipleRegisters** (uint8_t reg, uint8_t *values, uint16_t length)
Reads from multiple ST25R200 registers. More...

ReturnCode **st25r200WriteRegister** (uint8_t reg, uint8_t val)
Writes a given value to a register within the ST25R200. More...

ReturnCode **st25r200WriteMultipleRegisters** (uint8_t reg, const uint8_t *values, uint16_t length)
Writes multiple values to ST25R200 registers. More...

ReturnCode **st25r200WriteFifo** (const uint8_t *values, uint16_t length)
Writes values to ST25R200 FIFO. More...

ReturnCode **st25r200ReadFifo** (uint8_t *buf, uint16_t length)
Read values from ST25R200 FIFO. More...

ReturnCode **st25r200ExecuteCommand** (uint8_t cmd)
Execute a direct command. More...

ReturnCode **st25r200ExecuteCommands** (const uint8_t *cmds, uint8_t length)
Execute direct commands. More...

ReturnCode **st25r200ReadTestRegister** (uint8_t reg, uint8_t *val)
Read a test register within the ST25R200. More...

ReturnCode **st25r200WriteTestRegister** (uint8_t reg, uint8_t val)
Writes a given value to a test register within the ST25R200. More...

ReturnCode **st25r200WriteMultipleTestRegister** (uint8_t reg, const uint8_t *values, uint8_t length)
Writes a multiple values to test registers within the ST25R200. More...

ReturnCode	**st25r200ClrRegisterBits** (uint8_t reg, uint8_t clr_mask) Cleart bits on Register. More...
ReturnCode	**st25r200SetRegisterBits** (uint8_t reg, uint8_t set_mask) Set bits on Register. More...
ReturnCode	**st25r200ChangeRegisterBits** (uint8_t reg, uint8_t valueMask, uint8_t value) Changes the given bits on a ST25R200 register. More...
ReturnCode	**st25r200ModifyRegister** (uint8_t reg, uint8_t clr_mask, uint8_t set_mask) Modifies a value within a ST25R200 register. More...
ReturnCode	**st25r200ChangeTestRegisterBits** (uint8_t reg, uint8_t valueMask, uint8_t value) Changes the given bits on a ST25R200 Test register. More...
bool	**st25r200CheckReg** (uint8_t reg, uint8_t mask, uint8_t val) Checks if register contains a expected value. More...
bool	**st25r200IsRegValid** (uint8_t reg) Check if register ID is valid. More...
void	**st25r200ledInit** (void) ST25R200 LED Initialize. More...
void	**st25r200ledEvtIrq** (uint32_t irqs) ST25R200 LED Event Interrupt. More...
void	**st25r200ledEvtWrReg** (uint8_t reg, uint8_t val) ST25R200 LED Event Write Register. More...
void	**st25r200ledEvtWrMultiReg** (uint8_t reg, const uint8_t *vals, uint8_t len) ST25R200 LED Event Write Multiple Register. More...
void	**st25r200ledEvtCmd** (uint8_t cmd) ST25R200 LED Event Direct Command. More...
void	**st25r200ledEvtCmds** (const uint8_t *cmds, uint8_t len) ST25R200 LED Event Direct Commands. More...

Detailed Description

Macro Definition Documentation

◆ ST25R200_FIFO_STATUS_LEN

```
#define ST25R200_FIFO_STATUS_LEN   2
```

Number of FIFO Status Register

◆ ST25R200_OBS_MODE_LEN

```
#define ST25R200_OBS_MODE_LEN   2
```

Number of Observation Mode Config Registers

◆ ST25R200_REG_COLLISION

```
#define ST25R200_REG_COLLISION   0x38U
```

RO Collision Register 2

◆ ST25R200_REG_CORR1

```
#define ST25R200_REG_CORR1   0x09U
```

RW Correlator Register 1

◆ ST25R200_REG_CORR2

```
#define ST25R200_REG_CORR2   0x0AU
```

RW Correlator Register 2

◆ ST25R200_REG_CORR3

```
#define ST25R200_REG_CORR3   0x0BU
```

RW Correlator Register 3

◆ ST25R200_REG_CORR4

```
#define ST25R200_REG_CORR4   0x0CU
```

RW Correlator Register 4

◆ ST25R200_REG_CORR5

```
#define ST25R200_REG_CORR5   0x0DU
```

RW Correlator Register 5

◆ ST25R200_REG_CORR6

```
#define ST25R200_REG_CORR6   0x0EU
```

RW Correlator Register 6

◆ ST25R200_REG_DISPLAY1

```
#define ST25R200_REG_DISPLAY1   0x0FU
```

RW Correlator Register 7

◆ ST25R200_REG_DISPLAY2

```
#define ST25R200_REG_DISPLAY2   0x10U
```

RO Regulator and Gain Display Register

◆ ST25R200_REG_DISPLAY3

```
#define ST25R200_REG_DISPLAY3   0x23U
```

RO Display Register 2

◆ ST25R200_REG_DISPLAY4

```
#define ST25R200_REG_DISPLAY4   0x24U
```

RO Display Register 3

◆ ST25R200_REG_EMD1

```
#define ST25R200_REG_EMD1   0x19U
```

RW Protocol EMD Register 1

◆ ST25R200_REG_EMD2

```
#define ST25R200_REG_EMD2   0x1AU
```

RW Protocol EMD Register 2

◆ ST25R200_REG_FIFO_STATUS1

```
#define ST25R200_REG_FIFO_STATUS1   0x36U
```

RO FIFO Status Register 1

◆ ST25R200_REG_FIFO_STATUS2

```
#define ST25R200_REG_FIFO_STATUS2   0x37U
```

RO FIFO Status Register 2

◆ ST25R200_REG_GENERAL

```
#define ST25R200_REG_GENERAL   0x01U
```

RW General Register

◆ ST25R200_REG_GPT1

```
#define ST25R200_REG_GPT1   0x21U
```

RW General Purpose Timer (MSB) Register 1

◆ ST25R200_REG_GPT2

```
#define ST25R200_REG_GPT2   0x22U
```

RW General Purpose Timer (LSB) Register 2

◆ ST25R200_REG_IC_ID

```
#define ST25R200_REG_IC_ID   0x3FU
```

RO IC Identity Register

◆ ST25R200_REG_IRQ1

```
#define ST25R200_REG_IRQ1   0x3CU
```

RO IRQ Register 1

◆ ST25R200_REG_IRQ2

```
#define ST25R200_REG_IRQ2   0x3DU
```

RO IRQ Register 2

◆ ST25R200_REG_IRQ3

```
#define ST25R200_REG_IRQ3   0x3EU
```

RO IRQ Register 3

◆ ST25R200_REG_IRQ_MASK1

```
#define ST25R200_REG_IRQ_MASK1   0x39U
```

RO IRQ Mask Register 1

◆ ST25R200_REG_IRQ_MASK2

```
#define ST25R200_REG_IRQ_MASK2   0x3AU
```

RO IRQ Mask Register 2

◆ ST25R200_REG_IRQ_MASK3

```
#define ST25R200_REG_IRQ_MASK3   0x3BU
```

RO IRQ Mask Register 3

◆ ST25R200_REG_MRT

```
#define ST25R200_REG_MRT   0x1CU
```

RW Mask Receive Timer Register

◆ ST25R200_REG_MRT_SQT_CONF

```
#define ST25R200_REG_MRT_SQT_CONF   0x1BU
```

RW Mask Receive and Squelch Timer Configuration Register

◆ ST25R200_REG_NRT1

```
#define ST25R200_REG_NRT1   0x1FU
```

RW No Response Timer (MSB) Register 1

◆ ST25R200_REG_NRT2

```
#define ST25R200_REG_NRT2   0x20U
```

RW No Response Timer (LSB) Register 2

◆ ST25R200_REG_NRT_GPT_CONF

```
#define ST25R200_REG_NRT_GPT_CONF   0x1EU
```

RW No Response and Genereal Purpose Timer Configuration Register

◆ ST25R200_REG_OPERATION

```
#define ST25R200_REG_OPERATION   0x00U
```

RW Operation Register

◆ ST25R200_REG_OVERSHOOT_CONF

```
#define ST25R200_REG_OVERSHOOT_CONF   0x26U
```

RW Overshoot Pattern Register

◆ ST25R200_REG_OVUNSHOOT_CONF

```
#define ST25R200_REG_OVUNSHOOT_CONF   0x25U
```

RW Overshoot and Undershoot Configuration Register

◆ ST25R200_REG_PROTOCOL

```
#define ST25R200_REG_PROTOCOL   0x12U
```

RW Protocol Register

◆ ST25R200_REG_PROTOCOL_RX1

```
#define ST25R200_REG_PROTOCOL_RX1   0x16U
```

RW Protocol RX Register 1

◆ ST25R200_REG_PROTOCOL_RX2

```
#define ST25R200_REG_PROTOCOL_RX2   0x17U
```

RW Protocol RX Register 2

◆ ST25R200_REG_PROTOCOL_RX3

```
#define ST25R200_REG_PROTOCOL_RX3   0x18U
```

RW Protocol RX Register 3

◆ ST25R200_REG_PROTOCOL_TX1

```
#define ST25R200_REG_PROTOCOL_TX1   0x13U
```

RW Protocol TX Register 1

◆ ST25R200_REG_PROTOCOL_TX2

```
#define ST25R200_REG_PROTOCOL_TX2   0x14U
```

RW Protocol TX Register 2

◆ ST25R200_REG_PROTOCOL_TX3

```
#define ST25R200_REG_PROTOCOL_TX3   0x15U
```

RW Protocol TX Register 3

◆ ST25R200_REG_REGULATOR

```
#define ST25R200_REG_REGULATOR   0x02U
```

RW Regulator Register

◆ ST25R200_REG_RX_ANA1

```
#define ST25R200_REG_RX_ANA1   0x06U
```

RW RX Path Analog Register 1

◆ ST25R200_REG_RX_ANA2

```
#define ST25R200_REG_RX_ANA2   0x07U
```

RW RX Path Analog Register 2

◆ ST25R200_REG_RX_DIG

```
#define ST25R200_REG_RX_DIG   0x08U
```

RW RX Path Digital Register

◆ ST25R200_REG_SQT

```
#define ST25R200_REG_SQT   0x1DU
```

RW Squelch Timer Register

◆ ST25R200_REG_STATUS

```
#define ST25R200_REG_STATUS   0x11U
```

RO Status Register

◆ ST25R200_REG_TX_DRIVER

```
#define ST25R200_REG_TX_DRIVER   0x03U
```

RW TX Driver Register

◆ ST25R200_REG_TX_FRAME1

```
#define ST25R200_REG_TX_FRAME1   0x34U
```

RW TX Frame Register 1

◆ ST25R200_REG_TX_FRAME2

```
#define ST25R200_REG_TX_FRAME2   0x35U
```

RW TX Frame Register 2

◆ ST25R200_REG_TX_MOD

```
#define ST25R200_REG_TX_MOD   0x04U
```

RW TX Modulation Register

◆ ST25R200_REG_TX_RES_MOD

```
#define ST25R200_REG_TX_RES_MOD   0x05U
```

RW TX Resistive Modulation Register

◆ ST25R200_REG_UNDERSHOOT_CONF

```
#define ST25R200_REG_UNDERSHOOT_CONF   0x27U
```

RW Undershoot Pattern Register

◆ ST25R200_REG_WAKEUP_CONF1

```
#define ST25R200_REG_WAKEUP_CONF1   0x28U
```

RW Wake-up COnfiguration Register 1

◆ ST25R200_REG_WAKEUP_CONF2

```
#define ST25R200_REG_WAKEUP_CONF2   0x29U
```

RW Wake-up COnfiguration Register 2

◆ ST25R200_REG_WU_I_ADC

```
#define ST25R200_REG_WU_I_ADC   0x2CU
```

RO Wake-up I-Channel Calibration Display Register

◆ ST25R200_REG_WU_I_CAL

```
#define ST25R200_REG_WU_I_CAL   0x2EU
```

RO Wake-up I-Channel Calibration Display Register

◆ ST25R200_REG_WU_I_CONF

```
#define ST25R200_REG_WU_I_CONF   0x2AU
```

RW Wake-up I-Channel Configuration Register

◆ ST25R200_REG_WU_I_DELTA

```
#define ST25R200_REG_WU_I_DELTA   0x2BU
```

RW Wake-up I-Channel Delta Register

◆ ST25R200_REG_WU_I_REF

```
#define ST25R200_REG_WU_I_REF   0x2DU
```

RO Wake-up I-Channel Reference Display Register

◆ ST25R200_REG_WU_Q_ADC

```
#define ST25R200_REG_WU_Q_ADC   0x31U
```

RO Wake-up Q-Channel ADC Display Register

◆ ST25R200_REG_WU_Q_CAL

```
#define ST25R200_REG_WU_Q_CAL   0x33U
```

RO Wake-up Q-Channel Calibration Display Register

◆ ST25R200_REG_WU_Q_CONF

```
#define ST25R200_REG_WU_Q_CONF   0x2FU
```

RW Wake-up Q-Channel Configuration Register

◆ ST25R200_REG_WU_Q_DELTA

```
#define ST25R200_REG_WU_Q_DELTA   0x30U
```

RW Wake-up Q-Channel Delta Register

◆ ST25R200_REG_WU_Q_REF

```
#define ST25R200_REG_WU_Q_REF   0x32U
```

RO Wake-up Q-Channel Reference Display Register

Function Documentation

◆ st25r200ChangeRegisterBits()

```
ReturnCode st25r200ChangeRegisterBits ( uint8_t reg,
                                        uint8_t valueMask,
                                        uint8_t value
                                      )
```

This function is used if only a particular bits should be changed within an ST25R200 register.

Parameters

- [in] **reg** Address of the register to change.
- [in] **valueMask** bitmask of bits to be changed
- [in] **value** the bits to be written on the enabled valueMask bits

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r200ChangeTestRegisterBits()

```
ReturnCode st25r200ChangeTestRegisterBits ( uint8_t reg,
                                            uint8_t valueMask,
                                            uint8_t value
                                          )
```

This function is used if only a particular bits should be changed within an ST25R200 register.

Parameters

- [in] **reg** Address of the Test register to change.
- [in] **valueMask** bitmask of bits to be changed
- [in] **value** the bits to be written on the enabled valueMask bits

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r200CheckReg()

```
bool st25r200CheckReg ( uint8_t  reg,
                        uint8_t  mask,
                        uint8_t  val
                      )
```

This function checks if the given reg contains a value that once masked equals the expected value

Parameters

reg : the register to check the value

mask : the mask apply on register value

val : expected value to be compared to

Returns

true when reg contains the expected value | false otherwise

◆ st25r200ClrRegisterBits()

```
ReturnCode st25r200ClrRegisterBits ( uint8_t  reg,
                                     uint8_t  clr_mask
                                   )
```

This function clears the given bitmask on the register

Parameters

[in] **reg** Address of the register clear

[in] **clr_mask** Bitmask of bit to be cleared

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r200ExecuteCommand()

ReturnCode st25r200ExecuteCommand (uint8_t cmd)

This function is used to trigger a direct command. These commands are implemented inside the chip and each command has unique code

Parameters

[in] **cmd** : code of the direct command to be executed

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r200ExecuteCommands()

ReturnCode st25r200ExecuteCommands (const uint8_t * cmds,
uint8_t length
)

This function is used to trigger execution of direct commands. These commands are implemented inside the chip and each command has unique code (see datasheet).

Warning

Commands to be executed must support command chaining

Parameters

[in] **cmds** : code of the direct commands to be executed

[in] **length** : number of direct command to be executed

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r200IsRegValid()

bool st25r200IsRegValid (uint8_t reg)

Checks if the given register ID a valid ST25R200 register

Parameters

[in] **reg** Address of register to check

Returns

true if is a valid register ID

false otherwise

◆ st25r200ledEvtCmd()

```
void st25r200ledEvtCmd ( uint8_t  cmd )
```

This function should be called upon a ST25R200 direct command, providing the event with the command executed

Parameters

[in] **cmd** ST25R200 cmd executed

◆ st25r200ledEvtCmds()

```
void st25r200ledEvtCmds ( const uint8_t *  cmds,
                          uint8_t          len
                        )
```

This function should be called upon the execution of ST25R200 direct commands, providing the event with the commands executed

Parameters

[in] **cmds** ST25R200 cmds executed

[in] **len** : number of commands executed

◆ st25r200ledEvtIrq()

```
void st25r200ledEvtIrq ( uint32_t  irqs )
```

This function should be called upon a ST25R200 Interrupt, providing the interrupt event with ST25R200 irq flags to update LEDs

Parameters

[in] **irqs** ST25R200 irqs mask

◆ st25r200ledEvtWrMultiReg()

```
void st25r200ledEvtWrMultiReg ( uint8_t          reg,
                                const uint8_t *  vals,
                                uint8_t          len
                              )
```

This function should be called upon a ST25R200 Write Multiple Registers, providing the event with the registers and values to update LEDs

Parameters

[in] **reg** : ST25R200 first register written

[in] **vals** pointer to the values written

[in] **len** : number of registers written

◆ st25r200ledEvtWrReg()

```
void st25r200ledEvtWrReg ( uint8_t  reg,
                           uint8_t  val
                         )
```

This function should be called on a ST25R200 Write Register operation providing the event with the register and value to update LEDs

Parameters

[in] **reg** ST25R200 register to be written

[in] **val** value to be written on the register

◆ st25r200ledInit()

```
void st25r200ledInit ( void )
```

This function initializes the LEDs that represent ST25R200 activity

◆ st25r200ModifyRegister()

```
ReturnCode st25r200ModifyRegister ( uint8_t  reg,
                                    uint8_t  clr_mask,
                                    uint8_t  set_mask
                                  )
```

This function is used if only a particular bits should be changed within an ST25R200 register.

Parameters

- [in] **reg** Address of the register to write.
- [in] **clr_mask** bitmask of bits to be cleared to 0.
- [in] **set_mask** bitmask of bits to be set to 1.

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r200ReadFifo()

```
ReturnCode st25r200ReadFifo ( uint8_t *  buf,
                              uint16_t  length
                            )
```

This function needs to be called in order to read from ST25R200 FIFO.

Parameters

- [out] **buf** pointer to a buffer where the FIFO content shall be written to
- [in] **length** Number of bytes to read

Note

: This function doesn't check whether *length* is really the number of available bytes in FIFO

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r200ReadMultipleRegisters()

```
ReturnCode st25r200ReadMultipleRegisters ( uint8_t    reg,
                                           uint8_t *  values,
                                           uint16_t   length
                                         )
```

This function is used to read from multiple registers using the auto-increment feature. That is, after each read the address pointer inside the ST25R200 gets incremented automatically

Parameters

- [in] **reg** Address of the frist register to read from
- [in] **values** pointer to a buffer where the result shall be written to
- [in] **length** Number of registers to be read out

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r200ReadRegister()

```
ReturnCode st25r200ReadRegister ( uint8_t    reg,
                                  uint8_t *  val
                                )
```

Register bit definitions
DOXYGEN_SUPPRESS

This function is used to read out the content of ST25R200 registers

Parameters

- [in] **reg** Address of register to read
- [out] **val** Returned value

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r200ReadTestRegister()

```
ReturnCode st25r200ReadTestRegister ( uint8_t    reg,
                                      uint8_t *  val
                                    )
```

This function is used to read the content of test address *reg* within the ST25R200

Parameters

[in] **reg** Address of the register to read

[out] **val** Returned read value

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r200SetRegisterBits()

```
ReturnCode st25r200SetRegisterBits ( uint8_t  reg,
                                     uint8_t  set_mask
                                   )
```

This function sets the given bitmask on the register

Parameters

[in] **reg** Address of the register clear

[in] **set_mask** Bitmask of bit to be cleared

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r200WriteFifo()

```
ReturnCode st25r200WriteFifo ( const uint8_t *  values,
                               uint16_t         length
```

```
)
```

This function needs to be called in order to write to the ST25R200 FIFO.

Parameters

[in] **values** pointer to a buffer containing the values to be written to the FIFO

[in] **length** Number of values to be written

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r200WriteMultipleRegisters()

```
ReturnCode st25r200WriteMultipleRegisters ( uint8_t          reg,
                                            const uint8_t *  values,
                                            uint16_t         length
                                          )
```

This function is used to write multiple values to the ST25R200 using the auto-increment feature. That is, after each write the address pointer inside the ST25R200 gets incremented automatically

Parameters

[in] **reg** : Address of the frist register to write

[in] **values** pointer to a buffer containing the values to be written

[in] **length** Number of values to be written

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r200WriteMultipleTestRegister()

```
ReturnCode st25r200WriteMultipleTestRegister ( uint8_t          reg,
                                               const uint8_t *  values,
                                               uint8_t          length
```

```
)
```

This function is used to write *val* to test address *reg* within the ST25R200

Parameters

[in] **reg** : Address of the frist register to write

[in] **values** Pointer to a buffer containing the values to be written

[in] **length** Number of values to be written

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r200WriteRegister()

```
ReturnCode st25r200WriteRegister ( uint8_t reg,
                                   uint8_t val
                                 )
```

This function is used to write *val* to address *reg* within the ST25R200.

Parameters

[in] **reg** Address of the register to write

[in] **val** Value to be written

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r200WriteTestRegister()

```
ReturnCode st25r200WriteTestRegister ( uint8_t reg,
                                       uint8_t val
                                     )
```

This function is used to write *val* to test address *reg* within the ST25R200

Parameters

[in] **reg** Address of the register to write

[in] **val** Value to be written

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

---

ST25R200_IRQ

RFAL ST25R200 IRQ. More...

Collaboration diagram for ST25R200_IRQ:

Macros

#define **ST25R200_IRQ_MASK_ALL** (uint32_t)(0xFFFFFFFFUL)

#define **ST25R200_IRQ_MASK_NONE** (uint32_t)(0x00000000UL)

#define **ST25R200_IRQ_MASK_SUBC_START** (uint32_t)(0x00000080U)

```
#define ST25R200_IRQ_MASK_COL (uint32_t)(0x00000040U)
#define ST25R200_IRQ_MASK_WL (uint32_t)(0x00000020U)
#define ST25R200_IRQ_MASK_RX_REST (uint32_t)(0x00000010U)
#define ST25R200_IRQ_MASK_RXE (uint32_t)(0x00000008U)
#define ST25R200_IRQ_MASK_RXS (uint32_t)(0x00000004U)
#define ST25R200_IRQ_MASK_TXE (uint32_t)(0x00000002U)
#define ST25R200_IRQ_MASK_RFU1 (uint32_t)(0x00000001U)
#define ST25R200_IRQ_MASK_GPE (uint32_t)(0x00008000U)
#define ST25R200_IRQ_MASK_NRE (uint32_t)(0x00004000U)
#define ST25R200_IRQ_MASK_RFU2 (uint32_t)(0x00002000U)
#define ST25R200_IRQ_MASK_RFU3 (uint32_t)(0x00001000U)
#define ST25R200_IRQ_MASK_CRC (uint32_t)(0x00000800U)
#define ST25R200_IRQ_MASK_PAR (uint32_t)(0x00000400U)
#define ST25R200_IRQ_MASK_HFE (uint32_t)(0x00000200U)
#define ST25R200_IRQ_MASK_SFE (uint32_t)(0x00000100U)
#define ST25R200_IRQ_MASK_RFU4 (uint32_t)(0x00800000U)
#define ST25R200_IRQ_MASK_RFU5 (uint32_t)(0x00400000U)
#define ST25R200_IRQ_MASK_RFU6 (uint32_t)(0x00200000U)
#define ST25R200_IRQ_MASK_DCT (uint32_t)(0x00100000U)
#define ST25R200_IRQ_MASK_WUQ (uint32_t)(0x00080000U)
#define ST25R200_IRQ_MASK_WUI (uint32_t)(0x00040000U)
```

#define **ST25R200_IRQ_MASK_WUT** (uint32_t)(0x00020000U)

#define **ST25R200_IRQ_MASK_OSC** (uint32_t)(0x00010000U)

Functions

uint32_t **st25r200WaitForInterruptsTimed** (uint32_t mask, uint16_t tmo)
Wait until an ST25R200 interrupt occurs. More...

uint32_t **st25r200GetInterrupt** (uint32_t mask)
Get status for the given interrupt. More...

void **st25r200InitInterrupts** (void)
Init the 200 interrupt. More...

void **st25r200ModifyInterrupts** (uint32_t clr_mask, uint32_t set_mask)
Modifies the Interrupt. More...

void **st25r200CheckForReceivedInterrupts** (void)
Checks received interrupts. More...

void **st25r200Isr** (void)
ISR Service routine. More...

void **st25r200EnableInterrupts** (uint32_t mask)
Enable a given ST25R200 Interrupt source. More...

void **st25r200DisableInterrupts** (uint32_t mask)
Disable one or more a given ST25R200 Interrupt sources. More...

void **st25r200ClearInterrupts** (void)
Clear all ST25R200 irq flags. More...

void **st25r200ClearAndEnableInterrupts** (uint32_t mask)
Clears and then enables the given ST25R200 Interrupt sources. More...

void **st25r200IRQCallbackSet** (void(*cb)(void))
Sets IRQ callback for the ST25R200 interrupt. More...

void **st25r200IRQCallbackRestore** (void)

Sets IRQ callback for the ST25R200 interrupt. More...

Detailed Description

Macro Definition Documentation

◆ ST25R200_IRQ_MASK_ALL

```
#define ST25R200_IRQ_MASK_ALL   (uint32_t)(0xFFFFFFFFUL)
```

All ST25R200 interrupt sources

◆ ST25R200_IRQ_MASK_COL

```
#define ST25R200_IRQ_MASK_COL   (uint32_t)(0x00000040U)
```

ST25R200 bit collision interrupt

◆ ST25R200_IRQ_MASK_CRC

```
#define ST25R200_IRQ_MASK_CRC   (uint32_t)(0x00000800U)
```

ST25R200 CRC error interrupt

◆ ST25R200_IRQ_MASK_DCT

```
#define ST25R200_IRQ_MASK_DCT   (uint32_t)(0x00100000U)
```

ST25R200 termination of direct command interrupt

◆ ST25R200_IRQ_MASK_GPE

```
#define ST25R200_IRQ_MASK_GPE   (uint32_t)(0x00008000U)
```

ST25R200 general purpose timer expired interrupt

◆ ST25R200_IRQ_MASK_HFE

```
#define ST25R200_IRQ_MASK_HFE   (uint32_t)(0x00000200U)
```

ST25R200 hard framing error interrupt

◆ ST25R200_IRQ_MASK_NONE

```
#define ST25R200_IRQ_MASK_NONE   (uint32_t)(0x00000000UL)
```

No ST25R200 interrupt source

◆ ST25R200_IRQ_MASK_NRE

```
#define ST25R200_IRQ_MASK_NRE   (uint32_t)(0x00004000U)
```

ST25R200 no-response timer expired interrupt

◆ ST25R200_IRQ_MASK_OSC

```
#define ST25R200_IRQ_MASK_OSC   (uint32_t)(0x00010000U)
```

ST25R200 oscillator stable interrupt

◆ ST25R200_IRQ_MASK_PAR

```
#define ST25R200_IRQ_MASK_PAR   (uint32_t)(0x00000400U)
```

ST25R200 parity error interrupt

◆ ST25R200_IRQ_MASK_RFU1

```
#define ST25R200_IRQ_MASK_RFU1   (uint32_t)(0x00000001U)
```

ST25R200 RFU interrupt

◆ ST25R200_IRQ_MASK_RFU2

```
#define ST25R200_IRQ_MASK_RFU2   (uint32_t)(0x00002000U)
```

ST25R200 RFU interrupt

◆ ST25R200_IRQ_MASK_RFU3

```
#define ST25R200_IRQ_MASK_RFU3   (uint32_t)(0x00001000U)
```

ST25R200 RFU interrupt

◆ ST25R200_IRQ_MASK_RFU4

```
#define ST25R200_IRQ_MASK_RFU4   (uint32_t)(0x00800000U)
```

ST25R200 RFU interrupt

◆ ST25R200_IRQ_MASK_RFU5

```
#define ST25R200_IRQ_MASK_RFU5   (uint32_t)(0x00400000U)
```

ST25R200 RFU interrupt

◆ ST25R200_IRQ_MASK_RFU6

```
#define ST25R200_IRQ_MASK_RFU6   (uint32_t)(0x00200000U)
```

ST25R200 RFU interrupt

◆ ST25R200_IRQ_MASK_RX_REST

```
#define ST25R200_IRQ_MASK_RX_REST   (uint32_t)(0x00000010U)
```

ST25R200 automatic reception restart interrupt

◆ ST25R200_IRQ_MASK_RXE

```
#define ST25R200_IRQ_MASK_RXE   (uint32_t)(0x00000008U)
```

ST25R200 end of receive interrupt

◆ ST25R200_IRQ_MASK_RXS

```
#define ST25R200_IRQ_MASK_RXS   (uint32_t)(0x00000004U)
```

ST25R200 start of receive interrupt

◆ ST25R200_IRQ_MASK_SFE

```
#define ST25R200_IRQ_MASK_SFE   (uint32_t)(0x00000100U)
```

ST25R200 soft framing error interrupt

◆ ST25R200_IRQ_MASK_SUBC_START

```
#define ST25R200_IRQ_MASK_SUBC_START   (uint32_t)(0x00000080U)
```

ST25R200 subcarrier start interrupt

◆ ST25R200_IRQ_MASK_TXE

```
#define ST25R200_IRQ_MASK_TXE   (uint32_t)(0x00000002U)
```

ST25R200 end of transmission interrupt

◆ ST25R200_IRQ_MASK_WL

```
#define ST25R200_IRQ_MASK_WL   (uint32_t)(0x00000020U)
```

ST25R200 FIFO water level interrupt

◆ ST25R200_IRQ_MASK_WUI

#define ST25R200_IRQ_MASK_WUI (uint32_t)(0x00040000U)

ST25R200 wake-up I-Channel interrupt

◆ ST25R200_IRQ_MASK_WUQ

#define ST25R200_IRQ_MASK_WUQ (uint32_t)(0x00080000U)

ST25R200 wake-up Q-Channel interrupt

◆ ST25R200_IRQ_MASK_WUT

#define ST25R200_IRQ_MASK_WUT (uint32_t)(0x00020000U)

ST25R200 wake-up interrupt

Function Documentation

◆ st25r200CheckForReceivedInterrupts()

void st25r200CheckForReceivedInterrupts (void)

Checks received interrupts and saves the result into global params

◆ st25r200ClearAndEnableInterrupts()

void st25r200ClearAndEnableInterrupts (uint32_t mask)

Parameters

[in] **mask** mask indicating the interrupts to be cleared and enabled

◆ st25r200ClearInterrupts()

void st25r200ClearInterrupts (void)

◆ st25r200DisableInterrupts()

void st25r200DisableInterrupts (uint32_t mask)

This function disables all interrupts given by *mask*. 0xff disables all.

Parameters

[in] **mask** mask indicating the interrupts to be disabled.

◆ st25r200EnableInterrupts()

void st25r200EnableInterrupts (uint32_t mask)

This function enables all interrupts given by *mask*, ST25R200_IRQ_MASK_ALL enables all interrupts.

Parameters

[in] **mask** mask indicating the interrupts to be enabled

◆ st25r200GetInterrupt()

uint32_t st25r200GetInterrupt (uint32_t mask)

This function is used to check whether the interrupt given by *mask* has occured. If yes the interrupt gets cleared. This function returns only status bits which are inside *mask*.

Parameters

[in] **mask** : mask indicating the interrupt to check for.

Returns

the mask of the interrupts occurred

◆ st25r200InitInterrupts()

void st25r200InitInterrupts (void)

This function is used to check whether the interrupt given by *mask* has occured.

◆ st25r200IRQCallbackRestore()

```
void st25r200IRQCallbackRestore ( void )
```

◆ st25r200IRQCallbackSet()

```
void st25r200IRQCallbackSet ( void(*)(void) cb )
```

Parameters

[in] **cb** pointer to the callback method

◆ st25r200Isr()

```
void st25r200Isr ( void )
```

This function modiefies the interupt

◆ st25r200ModifyInterrupts()

```
void st25r200ModifyInterrupts ( uint32_t clr_mask,
                                uint32_t set_mask
                              )
```

This function modifies the interrupt

Parameters

[in] **clr_mask** : bit mask to be cleared on the interrupt mask

[in] **set_mask** : bit mask to be set on the interrupt mask

◆ st25r200WaitForInterruptsTimed()

```
uint32_t st25r200WaitForInterruptsTimed ( uint32_t mask,
                                          uint16_t tmo
                                        )
```

This function is used to access the ST25R200 interrupt flags. Use this to wait for max. *tmo* milliseconds for the **first** interrupt indicated with mask *mask* to occur.

Parameters

[in] **mask** : mask indicating the interrupts to wait for.

[in] **tmo** : time in milliseconds until timeout occurs. If set to 0 the functions waits forever.

Returns

: 0 if timeout occured otherwise a mask indicating the cleared interrupts.

ST25R3911

RFAL ST25R3911 Driver. More...

Collaboration diagram for ST25R3911:

Modules

ST25R3911_Driver

RFAL ST25R3911 Driver.

ST25R3911_Com

RFAL ST25R3911 Communication.

ST25R3911_Interrupt
RFAL ST25R3911 Interrupt.

Detailed Description

ST25R3911_Driver

RFAL ST25R3911 Driver. More...

Collaboration diagram for ST25R3911_Driver:

Data Structures

struct **st25r3911StreamConfig**

Macros

```
#define ST25R3911_FDT_NONE 0x00U
#define MS_TO_64FCS(A) ((A) * 212U)
#define MS_FROM_64FCS(A) ((A) / 212U)
#define ST25R3911_CMD_SET_DEFAULT 0xC1U
#define ST25R3911_CMD_CLEAR_FIFO 0xC2U
#define ST25R3911_CMD_TRANSMIT_WITH_CRC 0xC4U
#define ST25R3911_CMD_TRANSMIT_WITHOUT_CRC 0xC5U
#define ST25R3911_CMD_TRANSMIT_REQA 0xC6U
#define ST25R3911_CMD_TRANSMIT_WUPA 0xC7U
#define ST25R3911_CMD_INITIAL_RF_COLLISION 0xC8U
#define ST25R3911_CMD_RESPONSE_RF_COLLISION_N 0xC9U
#define ST25R3911_CMD_RESPONSE_RF_COLLISION_0 0xCAU
#define ST25R3911_CMD_NORMAL_NFC_MODE 0xCBU
#define ST25R3911_CMD_ANALOG_PRESET 0xCCU
#define ST25R3911_CMD_MASK_RECEIVE_DATA 0xD0U
#define ST25R3911_CMD_UNMASK_RECEIVE_DATA 0xD1U
#define ST25R3911_CMD_MEASURE_AMPLITUDE 0xD3U
#define ST25R3911_CMD_SQUELCH 0xD4U
#define ST25R3911_CMD_CLEAR_SQUELCH 0xD5U
#define ST25R3911_CMD_ADJUST_REGULATORS 0xD6U
#define ST25R3911_CMD_CALIBRATE_MODULATION 0xD7U
```

```
#define ST25R3911_CMD_CALIBRATE_ANTENNA 0xD8U
#define ST25R3911_CMD_MEASURE_PHASE 0xD9U
#define ST25R3911_CMD_CLEAR_RSSI 0xDAU
#define ST25R3911_CMD_TRANSPARENT_MODE 0xDCU
#define ST25R3911_CMD_CALIBRATE_C_SENSOR 0xDDU
#define ST25R3911_CMD_MEASURE_CAPACITANCE 0xDEU
#define ST25R3911_CMD_MEASURE_VDD 0xDFU
#define ST25R3911_CMD_START_GP_TIMER 0xE0U
#define ST25R3911_CMD_START_WUP_TIMER 0xE1U
#define ST25R3911_CMD_START_MASK_RECEIVE_TIMER 0xE2U
#define ST25R3911_CMD_START_NO_RESPONSE_TIMER 0xE3U
#define ST25R3911_CMD_TEST_CLEARA 0xFAU
#define ST25R3911_CMD_TEST_CLEARB 0xFBU
#define ST25R3911_CMD_TEST_ACCESS 0xFCU
#define ST25R3911_CMD_LOAD_PPROM 0xFDU
#define ST25R3911_CMD_FUSE_PPROM 0xFEU
#define ST25R3911_FIFO_DEPTH 96U
#define ST25R3911_THRESHOLD_DO_NOT_SET 0xFFU
#define ST25R3911_BR_DO_NOT_SET 0xFFU
#define ST25R3911_BR_106 0x00U
#define ST25R3911_BR_212 0x01U
```

#define **ST25R3911_BR_424** 0x02U

#define **ST25R3911_BR_848** 0x03U

#define **ST25R3911_BR_1695** 0x04U

#define **ST25R3911_BR_3390** 0x05U

#define **ST25R3911_BR_6780** 0x06U

#define **st25r3911IsGPTRunning**() (**st25r3911CheckReg**(**ST25R3911_REG_REGULATOR_RESULT**, ST25R3911_REG_REGULATOR_RESULT_gpt_on, ST25R3911_REG_REGULATOR_RESULT_gpt_on))

#define **st25r3911IsCRCinFIFO**() (**st25r3911CheckReg**(**ST25R3911_REG_AUX**, ST25R3911_REG_AUX_crc_2_fifo, ST25R3911_REG_AUX_crc_2_fifo))

#define **st25r3911IsExtFieldOn**() (**st25r3911CheckReg**(**ST25R3911_REG_AUX_DISPLAY**, ST25R3911_REG_AUX_DISPLAY_efd_o, ST25R3911_REG_AUX_DISPLAY_efd_o))

#define **st25r3911IsTxEnabled**() (**st25r3911CheckReg**(**ST25R3911_REG_OP_CONTROL**, ST25R3911_REG_OP_CONTROL_tx_en, ST25R3911_REG_OP_CONTROL_tx_en))

#define **st25r3911TxOff**() **st25r3911ClrRegisterBits**(**ST25R3911_REG_OP_CONTROL**, ST25R3911_REG_OP_CONTROL_tx_en);

#define **st25r3911IsLastFIFOComplete**() **st25r3911CheckReg**(**ST25R3911_REG_FIFO_RX_STATUS2**, ST25R3911_REG_FIFO_RX_STATUS2_mask_fifo_lb, 0)

#define **st25r3911IsOscOn**() **st25r3911CheckReg**(**ST25R3911_REG_OP_CONTROL**, ST25R3911_REG_OP_CONTROL_en, ST25R3911_REG_OP_CONTROL_en)

Functions

ReturnCode **st25r3911OscOn** (void)

Turn on Oscillator and Regulator. More...

void **st25r3911TxRxOn** (void)

Turn On Tx and Rx. More...

void **st25r3911TxRxOff** (void)

Turn Off Tx and Rx. More...

ReturnCode	**st25r3911Initialize** (void) Initialise ST25R3911 driver. More...
void	**st25r3911Deinitialize** (void) Deinitialize ST25R3911 driver. More...
ReturnCode	**st25r3911SetBitrate** (uint8_t txRate, uint8_t rxRate) Sets the bitrate registers. More...
ReturnCode	**st25r3911AdjustRegulators** (uint16_t *result_mV) Adjusts supply regulators according to the current supply voltage. More...
ReturnCode	**st25r3911SetRegulators** (uint8_t regulation) Sets supply regulators. More...
void	**st25r3911MeasureAmplitude** (uint8_t *result) Measure Amplitude. More...
void	**st25r3911MeasureCapacitance** (uint8_t *result) Measure Capacitance. More...
uint8_t	**st25r3911MeasurePowerSupply** (uint8_t mpsv) Measure Voltage. More...
uint16_t	**st25r3911MeasureVoltage** (uint8_t mpsv) Measure Voltage. More...
void	**st25r3911CalibrateAntenna** (uint8_t *result) Calibrate antenna. More...
void	**st25r3911MeasurePhase** (uint8_t *result) Measure Phase. More...
void	**st25r3911CalibrateModulationDepth** (uint8_t *result) Calibrate modulation depth. More...
ReturnCode	**st25r3911CalibrateCapacitiveSensor** (uint8_t *result)

Calibrate Capacitive Sensor. More...

ReturnCode	**st25r3911SetNoResponseTime_64fcs** (uint32_t nrt_64fcs) set no response time More...
ReturnCode	**st25r3911SetStartNoResponseTime_64fcs** (uint32_t nrt_64fcs) set no response time More...
ReturnCode	**st25r3911PerformCollisionAvoidance** (uint8_t FieldONCmd, uint8_t pdThreshold, uint8_t caThreshold, uint8_t nTRFW) Perform Collision Avoidance. More...
uint8_t	**st25r3911GetNumFIFOLastBits** (void) Get amount of bits of the last FIFO byte if incomplete. More...
uint32_t	**st25r3911GetNoResponseTime_64fcs** (void) Get NRT time. More...
void	**st25r3911SetGPTime_8fcs** (uint16_t gpt_8fcs) set general purpose timer timeout More...
void	**st25r3911StartGPTimer_8fcs** (uint16_t gpt_8fcs, uint8_t trigger_source) Starts GPT with given timeout. More...
bool	**st25r3911CheckReg** (uint8_t reg, uint8_t mask, uint8_t value) Checks if register contains a expected value. More...
void	**st25r3911SetNumTxBits** (uint32_t nBits) Sets the number Tx Bits. More...
bool	**st25r3911CheckChipID** (uint8_t *rev) Check Identity. More...
bool	**st25r3911IsCmdValid** (uint8_t cmd) Check if command is valid. More...
ReturnCode	**st25r3911StreamConfigure** (const struct **st25r3911StreamConfig** *config) Configure the stream mode of ST25R3911. More...

ReturnCode	**st25r3911GetRegsDump** (uint8_t *resRegDump, uint8_t *sizeRegDump) Register Dump. More...
ReturnCode	**st25r3911GetRSSI** (uint16_t *amRssi, uint16_t *pmRssi) Gets the RSSI values. More...
ReturnCode	**st25r3911SetAntennaMode** (bool single, bool rfiox) Set Antenna mode. More...

Detailed Description

Macro Definition Documentation

◆ MS_FROM_64FCS

```
#define MS_FROM_64FCS ( A ) ((A) / 212U)
```

Converts from 64/fc steps to ms

◆ MS_TO_64FCS

```
#define MS_TO_64FCS ( A ) ((A) * 212U)
```

Converts from ms to 64/fc steps

◆ ST25R3911_BR_106

```
#define ST25R3911_BR_106 0x00U
```

ST25R3911 Bit Rate 106 kbit/s (fc/128)

◆ ST25R3911_BR_1695

```
#define ST25R3911_BR_1695 0x04U
```

ST25R3911 Bit Rate 1696 kbit/s (fc/8)

◆ ST25R3911_BR_212

```
#define ST25R3911_BR_212   0x01U
```

ST25R3911 Bit Rate 212 kbit/s (fc/64)

◆ ST25R3911_BR_3390

```
#define ST25R3911_BR_3390   0x05U
```

ST25R3911 Bit Rate 3390 kbit/s (fc/4)

◆ ST25R3911_BR_424

```
#define ST25R3911_BR_424   0x02U
```

ST25R3911 Bit Rate 424 kbit/s (fc/32)

◆ ST25R3911_BR_6780

```
#define ST25R3911_BR_6780   0x06U
```

ST25R3911 Bit Rate 6780 kbit/s (fc/2)

◆ ST25R3911_BR_848

```
#define ST25R3911_BR_848   0x03U
```

ST25R3911 Bit Rate 848 kbit/s (fc/16)

◆ ST25R3911_BR_DO_NOT_SET

```
#define ST25R3911_BR_DO_NOT_SET   0xFFU
```

Indicates not to change this Bit Rate

◆ ST25R3911_CMD_ADJUST_REGULATORS

```
#define ST25R3911_CMD_ADJUST_REGULATORS   0xD6U
```

Adjust regulators

◆ ST25R3911_CMD_ANALOG_PRESET

```
#define ST25R3911_CMD_ANALOG_PRESET   0xCCU
```

Analog Preset

◆ ST25R3911_CMD_CALIBRATE_ANTENNA

```
#define ST25R3911_CMD_CALIBRATE_ANTENNA   0xD8U
```

Calibrate antenna

◆ ST25R3911_CMD_CALIBRATE_C_SENSOR

```
#define ST25R3911_CMD_CALIBRATE_C_SENSOR   0xDDU
```

Calibrate the capacitive sensor

◆ ST25R3911_CMD_CALIBRATE_MODULATION

```
#define ST25R3911_CMD_CALIBRATE_MODULATION   0xD7U
```

Calibrate modulation depth

◆ ST25R3911_CMD_CLEAR_FIFO

```
#define ST25R3911_CMD_CLEAR_FIFO   0xC2U
```

Stops all activities and clears FIFO

◆ ST25R3911_CMD_CLEAR_RSSI

```
#define ST25R3911_CMD_CLEAR_RSSI   0xDAU
```

clear RSSI bits and restart the measurement

◆ ST25R3911_CMD_CLEAR_SQUELCH

```
#define ST25R3911_CMD_CLEAR_SQUELCH   0xD5U
```

Clear Squelch

◆ ST25R3911_CMD_FUSE_PPROM

```
#define ST25R3911_CMD_FUSE_PPROM   0xFEU
```

Fuse poly fuses with data from the RAM

◆ ST25R3911_CMD_INITIAL_RF_COLLISION

```
#define ST25R3911_CMD_INITIAL_RF_COLLISION   0xC8U
```

NFC transmit with Initial RF Collision Avoidance

◆ ST25R3911_CMD_LOAD_PPROM

```
#define ST25R3911_CMD_LOAD_PPROM   0xFDU
```

Load data from the poly fuses to RAM

◆ ST25R3911_CMD_MASK_RECEIVE_DATA

```
#define ST25R3911_CMD_MASK_RECEIVE_DATA   0xD0U
```

Mask receive data

◆ ST25R3911_CMD_MEASURE_AMPLITUDE

```
#define ST25R3911_CMD_MEASURE_AMPLITUDE   0xD3U
```

Measure singal amplitude on RFI inputs

◆ ST25R3911_CMD_MEASURE_CAPACITANCE

```
#define ST25R3911_CMD_MEASURE_CAPACITANCE   0xDEU
```

Measure capacitance

◆ ST25R3911_CMD_MEASURE_PHASE

```
#define ST25R3911_CMD_MEASURE_PHASE   0xD9U
```

Measure phase between RFO and RFI signal

◆ ST25R3911_CMD_MEASURE_VDD

```
#define ST25R3911_CMD_MEASURE_VDD   0xDFU
```

Measure power supply voltage

◆ ST25R3911_CMD_NORMAL_NFC_MODE

```
#define ST25R3911_CMD_NORMAL_NFC_MODE   0xCBU
```

NFC switch to normal NFC mode

◆ ST25R3911_CMD_RESPONSE_RF_COLLISION_0

```
#define ST25R3911_CMD_RESPONSE_RF_COLLISION_0   0xCAU
```

NFC transmit with Response RF Collision Avoidance with n=0

◆ ST25R3911_CMD_RESPONSE_RF_COLLISION_N

```
#define ST25R3911_CMD_RESPONSE_RF_COLLISION_N   0xC9U
```

NFC transmit with Response RF Collision Avoidance

◆ ST25R3911_CMD_SET_DEFAULT

```
#define ST25R3911_CMD_SET_DEFAULT   0xC1U
```

Puts the chip in default state (same as after power-up)

◆ ST25R3911_CMD_SQUELCH

```
#define ST25R3911_CMD_SQUELCH   0xD4U
```

Squelch

◆ ST25R3911_CMD_START_GP_TIMER

```
#define ST25R3911_CMD_START_GP_TIMER   0xE0U
```

Start the general purpose timer

◆ ST25R3911_CMD_START_MASK_RECEIVE_TIMER

```
#define ST25R3911_CMD_START_MASK_RECEIVE_TIMER   0xE2U
```

Start the mask-receive timer

◆ ST25R3911_CMD_START_NO_RESPONSE_TIMER

```
#define ST25R3911_CMD_START_NO_RESPONSE_TIMER   0xE3U
```

Start the no-repsonse timer

◆ ST25R3911_CMD_START_WUP_TIMER

```
#define ST25R3911_CMD_START_WUP_TIMER   0xE1U
```

Start the wake-up timer

◆ ST25R3911_CMD_TEST_ACCESS

```
#define ST25R3911_CMD_TEST_ACCESS   0xFCU
```

Enable R/W access to the test registers

◆ ST25R3911_CMD_TEST_CLEARA

```
#define ST25R3911_CMD_TEST_CLEARA   0xFAU
```

Clear Test register

◆ ST25R3911_CMD_TEST_CLEARB

```
#define ST25R3911_CMD_TEST_CLEARB   0xFBU
```

Clear Test register

◆ ST25R3911_CMD_TRANSMIT_REQA

```
#define ST25R3911_CMD_TRANSMIT_REQA   0xC6U
```

Transmit REQA

◆ ST25R3911_CMD_TRANSMIT_WITH_CRC

```
#define ST25R3911_CMD_TRANSMIT_WITH_CRC   0xC4U
```

Transmit with CRC

◆ ST25R3911_CMD_TRANSMIT_WITHOUT_CRC

```
#define ST25R3911_CMD_TRANSMIT_WITHOUT_CRC   0xC5U
```

Transmit without CRC

◆ ST25R3911_CMD_TRANSMIT_WUPA

```
#define ST25R3911_CMD_TRANSMIT_WUPA   0xC7U
```

Transmit WUPA

◆ ST25R3911_CMD_TRANSPARENT_MODE

```
#define ST25R3911_CMD_TRANSPARENT_MODE   0xDCU
```

Transparent mode

◆ ST25R3911_CMD_UNMASK_RECEIVE_DATA

```
#define ST25R3911_CMD_UNMASK_RECEIVE_DATA   0xD1U
```

Unmask receive data

◆ ST25R3911_FDT_NONE

```
#define ST25R3911_FDT_NONE   0x00U
```

Value indicating not to perform FDT

◆ ST25R3911_FIFO_DEPTH

```
#define ST25R3911_FIFO_DEPTH   96U
```

Depth of FIFO

◆ ST25R3911_THRESHOLD_DO_NOT_SET

```
#define ST25R3911_THRESHOLD_DO_NOT_SET   0xFFU
```

Indicates not to change this Threshold

◆ st25r3911IsCRCinFIFO

```
#define st25r3911IsCRCinFIFO ( ) ( st25r3911CheckReg(ST25R3911_REG_AUX, ST25R3911_REG_AUX_crc_2_fifo, ST25R3911_REG_AUX_crc_2_fifo) )
```

Checks if CRC is configured to be in FIFO

◆ st25r3911IsExtFieldOn

```
#define st25r3911IsExtFieldOn ( ) ( st25r3911CheckReg(ST25R3911_REG_AUX_DISPLAY, ST25R3911_REG_AUX_DISPLAY_efd_o, ST25R3911_REG_AUX_DISPLAY_efd_o ) )
```

Checks if External Filed is detected by reading ST25R3911 External Field Detector output

◆ st25r3911IsGPTRunning

```
#define st25r3911IsGPTRunning ( ) ( st25r3911CheckReg(ST25R3911_REG_REGULATOR_RESULT, ST25R3911_REG_REGULATOR_RESULT_gpt_on, ST25R3911_REG_REGULATOR_RESULT_gpt_on) )
```

Checks if General Purpose Timer is still running by reading gpt_on flag

◆ st25r3911IsLastFIFOComplete

```
#define st25r3911IsLastFIFOComplete ( ) st25r3911CheckReg( ST25R3911_REG_FIFO_RX_STATUS2, ST25R3911_REG_FIFO_RX_STATUS2_mask_fifo_lb, 0 )
```

Checks if last FIFO byte is complete

◆ st25r3911IsOscOn

```
#define st25r3911IsOscOn ( ) st25r3911CheckReg( ST25R3911_REG_OP_CONTROL, ST25R3911_REG_OP_CONTROL_en, ST25R3911_REG_OP_CONTROL_en )
```

Checks if the Oscillator is enabled

◆ st25r3911IsTxEnabled

```
#define st25r3911IsTxEnabled ( ) ( st25r3911CheckReg(ST25R3911_REG_OP_CONTROL, ST25R3911_REG_OP_CONTROL_tx_en, ST25R3911_REG_OP_CONTROL_tx_en ) )
```

Checks if Transmitter is enabled (Field On)

◆ st25r3911TxOff

```
#define st25r3911TxOff ( ) st25r3911ClrRegisterBits(ST25R3911_REG_OP_CONTROL, ST25R3911_REG_OP_CONTROL_tx_en );
```

Turn Off Tx (Field Off)

Function Documentation

◆ st25r3911AdjustRegulators()

```
ReturnCode st25r3911AdjustRegulators ( uint16_t * result_mV )
```

On this function the power level is measured in maximum load conditions and the regulated voltage reference is set to 250mV below this level. Execution of this function lasts arround 5ms.

The regulated voltages will be set to the result of Adjust Regulators

Parameters

[out] **result_mV** : Result of calibration in milliVolts.

Returns

RFAL_ERR_IO : Error during communication with ST25R3911.

RFAL_ERR_NONE : No error.

◆ st25r3911CalibrateAntenna()

void st25r3911CalibrateAntenna (uint8_t * result)

This function is used to calibrate the antenna using a special sequence. The result is stored in the *result* parameter.

Parameters

[out] **result** 8 bit long result of antenna calibration algorithm.

◆ st25r3911CalibrateCapacitiveSensor()

ReturnCode st25r3911CalibrateCapacitiveSensor (uint8_t * result)

This function performs automatic calibration of the capacitive sensor and stores the result in parameter *result*.

Warning

To avoid interference with Xtal oscillator and reader magnetic field, it is strongly recommended to perform calibration in Power-down mode only. This method does not modify the Oscillator nor transmitter state, these should be configured before by user.

Parameters

[out] **result** 5 bit long result of the calibration. Binary weighted, step 0.1 pF, max 3.1 pF

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_IO : The calibration was not successful

RFAL_ERR_NONE : No error

◆ st25r3911CalibrateModulationDepth()

void st25r3911CalibrateModulationDepth (uint8_t * result)

This function is used to calibrate the modulation depth using a special sequence. The result is stored in the *result* parameter.

Parameters

[out] **result** 8 bit long result of antenna calibration algorithm.

◆ st25r3911CheckChipID()

```
bool st25r3911CheckChipID ( uint8_t *  rev )
```

Checks if the chip ID is as expected.

5 bit IC type code for ST25R3911: 00001 The 3 lsb contain the IC revision code

Parameters

[out] **rev** : the IC revision code

Returns

true when IC type is as expected

◆ st25r3911CheckReg()

```
bool st25r3911CheckReg ( uint8_t  reg,
                         uint8_t  mask,
                         uint8_t  value
                       )
```

This function checks if the given reg contains a value that once masked equals the expected value

Parameters

reg : the register to check the value

mask : the mask apply on register value

value : expected value to be compared to

Returns

true when reg contains the expected value | false otherwise

◆ st25r3911Deinitialize()

```
void st25r3911Deinitialize ( void )
```

Calling this function deinitializes the ST25R3911 driver.

◆ st25r3911GetNoResponseTime_64fcs()

```
uint32_t st25r3911GetNoResponseTime_64fcs ( void )
```

This returns the last value set on the NRT

Warning

it does not reads chip register, just the sw var that contains the last value set before

Returns

the value of the NRT

◆ st25r3911GetNumFIFOLastBits()

```
uint8_t st25r3911GetNumFIFOLastBits ( void )
```

Gets the number of bits of the last FIFO byte if incomplete

Returns

the number of bits of the last FIFO byte if incomplete, 0 if the last byte is complete

◆ st25r3911GetRegsDump()

```
ReturnCode st25r3911GetRegsDump ( uint8_t * resRegDump,
                                  uint8_t * sizeRegDump
                                )
```

Retrieves all internal registers from ST25R3911

Parameters

[out]	**resRegDump**	: pointer to the struct/buffer where the reg dump will be written
[in,out]	**sizeRegDump**	: number of registers requested and the ones actually written

Returns

RFAL_ERR_NONE : No error, stream mode driver initialized.

◆ st25r3911GetRSSI()

```
ReturnCode st25r3911GetRSSI ( uint16_t * amRssi,
                              uint16_t * pmRssi
                            )
```

This function gets the RSSI value of the previous reception taking into account the gain reductions that were used. RSSI value for both AM and PM channel can be retrieved.

Parameters

[out] **amRssi** the RSSI on the AM channel expressed in mV

[out] **pmRssi** the RSSI on the PM channel expressed in mV

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error

◆ st25r3911Initialize()

ReturnCode st25r3911Initialize (void)

This function initialises the ST25R3911 driver.

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_HW_MISMATCH : Expected HW do not match or communication error

RFAL_ERR_IO : Error during selftest - check communication interface

RFAL_ERR_TIMEOUT : Timeout during selftest - check IRQ handling

RFAL_ERR_SYSTEM : Failure during seltest - check oscillator or timers

◆ st25r3911IsCmdValid()

bool st25r3911IsCmdValid (uint8_t cmd)

Checks if the given command is a valid ST25R3911 command

Parameters

[in] **cmd** Command to check

Returns

true if is a valid command

false otherwise

◆ st25r3911MeasureAmplitude()

void st25r3911MeasureAmplitude (uint8_t * result)

This function measured the amplitude on the RFI inputs and stores the result in parameter *result*.

Parameters

[out] **result** 8 bit long result of RF measurement.

◆ st25r3911MeasureCapacitance()

void st25r3911MeasureCapacitance (uint8_t * result)

This function performs the capacitance measurement and stores the result in parameter *result*.

Parameters

[out] **result** 8 bit long result of RF measurement.

◆ st25r3911MeasurePhase()

void st25r3911MeasurePhase (uint8_t * result)

This function performs a Phase measurement. The result is stored in the *result* parameter.

Parameters

[out] **result** 8 bit long result of the measurement.

◆ st25r3911MeasurePowerSupply()

uint8_t st25r3911MeasurePowerSupply (uint8_t mpsv)

This function measures the voltage on one of VDD and VSP_* result in parameter *result*.

Parameters

[in] **mpsv** : one of ST25R3911_REG_REGULATOR_CONTROL_mpsv_vdd
ST25R3911_REG_REGULATOR_CONTROL_mpsv_vsp_rf
ST25R3911_REG_REGULATOR_CONTROL_mpsv_vsp_a or
ST25R3911_REG_REGULATOR_CONTROL_mpsv_vsp_d

Returns

the measured voltage raw values

◆ st25r3911MeasureVoltage()

uint16_t st25r3911MeasureVoltage (uint8_t mpsv)

This function measures the voltage on one of VDD and VSP_* result in parameter *result*.

Parameters

[in] **mpsv** : one of ST25R3911_REG_REGULATOR_CONTROL_mpsv_vdd
ST25R3911_REG_REGULATOR_CONTROL_mpsv_vsp_rf
ST25R3911_REG_REGULATOR_CONTROL_mpsv_vsp_a or
ST25R3911_REG_REGULATOR_CONTROL_mpsv_vsp_d

Returns

the measured voltage in mV

◆ st25r3911OscOn()

ReturnCode st25r3911OscOn (void)

This function turn on oscillator and regulator and wait for the oscillator to become stable.

Returns

RFAL_ERR_SYSTEM : Failure during Oscillator activation

RFAL_ERR_NONE : No error, Oscillator is active and stable, Regulator is on

◆ st25r3911PerformCollisionAvoidance()

ReturnCode st25r3911PerformCollisionAvoidance (uint8_t FieldONCmd,
uint8_t pdThreshold,
uint8_t caThreshold,
uint8_t nTRFW
)

Performs Collision Avoidance with the given threshold and with the
n number of TRFW

Parameters

[in] **FieldONCmd** : Field ON command to be executed ST25R3911_CMD_INITIAL_RF_COLLISION or
ST25R3911_CMD_RESPONSE_RF_COLLISION_0/N

[in] **pdThreshold** : Peer Detection Threshold (ST25R3911_REG_FIELD_THRESHOLD_trg_xx) 0xff : don't set Threshold (ST25R3911_THRESHOLD_DO_NOT_SET)

[in] **caThreshold** : Collision Avoidance Threshold (ST25R3911_REG_FIELD_THRESHOLD_rfe_xx) 0xff : don't set Threshold (ST25R3911_THRESHOLD_DO_NOT_SET)

[in] **nTRFW** : Number of TRFW

Returns

RFAL_ERR_NONE : no collision detected

RFAL_ERR_RF_COLLISION : collision detected

◆ st25r3911SetAntennaMode()

```
ReturnCode st25r3911SetAntennaMode ( bool  single,
                                     bool  rfiox
                                   )
```

Sets the antenna mode. Differential or single ended antenna mode (RFO1 or RFO2)

Parameters

[in] **single** FALSE differential ; single ended mode

[in] **rfiox** FALSE RFI1/RFO1 ; TRUE RFI2/RFO2

Returns

RFAL_ERR_IO : Internal error

RFAL_ERR_NOTSUPP : Feature not supported

RFAL_ERR_NONE : No error

◆ st25r3911SetBitrate()

```
ReturnCode st25r3911SetBitrate ( uint8_t  txRate,
                                 uint8_t  rxRate
                               )
```

This function sets the bitrate register for rx and tx

Parameters

txRate : speed is 2^txrate * 106 kb/s 0xff : don't set txrate

rxRate : speed is 2^rxrate * 106 kb/s 0xff : don't set rxrate

Returns

RFAL_ERR_NONE : No error, both bit rates were set

RFAL_ERR_PARAM: At least one bit rate was invalid

◆ st25r3911SetGPTime_8fcs()

void st25r3911SetGPTime_8fcs (uint16_t gpt_8fcs)

This function sets the proper registers but does not start the timer actually

Parameters

gpt_8fcs : general purpose timer timeout in 8/fc = 590ns

◆ st25r3911SetNoResponseTime_64fcs()

ReturnCode st25r3911SetNoResponseTime_64fcs (uint32_t nrt_64fcs)

This function executes sets the no response time to the defines value

Parameters

nrt_64fcs : no response time in 64/fc = 4.72us completion interrupt

Returns

RFAL_ERR_PARAM : if time is too large

◆ st25r3911SetNumTxBits()

void st25r3911SetNumTxBits (uint32_t nBits)

Sets ST25R3911 internal registers with correct number of complete bytes and bits to be sent

Parameters

nBits : the number bits to be transmitted

◆ st25r3911SetRegulators()

ReturnCode st25r3911SetRegulators (uint8_t regulation)

Manually sets the regulated voltage setting (rege). The regulated voltages will be set to the manual configuratiom.

Regulation shall be set according to the desired voltage considering the supplied voltage.

Parameters

[in] **regulation** : Regulator setting

Returns

RFAL_ERR_IO : Error during communication with ST25R3911

RFAL_ERR_PARAM : Invalid regulator setting

RFAL_ERR_NONE : No error

◆ st25r3911SetStartNoResponseTime_64fcs()

ReturnCode st25r3911SetStartNoResponseTime_64fcs (uint32_t nrt_64fcs)

This function executes sets and immediately start the no response timer to the defines value This is used when needs to add more time before timeout whitout Tx

Parameters

nrt_64fcs : no response time in 64/fc = 4.72us completion interrupt

Returns

RFAL_ERR_PARAM : if time is too large

◆ st25r3911StartGPTimer_8fcs()

void st25r3911StartGPTimer_8fcs (uint16_t gpt_8fcs,
uint8_t trigger_source
)

This function starts the general purpose timer with the given timeout

Parameters

gpt_8fcs : general purpose timer timeout in 8/fc = 590ns

trigger_source : no trigger, start of Rx, end of Rx, end of Tx in NFC mode

◆ st25r3911StreamConfigure()

ReturnCode st25r3911StreamConfigure (const struct **st25r3911StreamConfig** * config)

This function initializes the stream with the given parameters

Parameters

[in] **config** : all settings for bitrates, type, etc.

Returns

RFAL_ERR_NONE : No error, stream mode driver initialized.

◆ st25r3911TxRxOff()

void st25r3911TxRxOff (void)

This function turns Off Tx and Rx (Field Off)

◆ st25r3911TxRxOn()

void st25r3911TxRxOn (void)

This function turns On Tx and Rx (Field On)

st25r3911StreamConfig Struct Reference


```
#include <st25r3911.h>
```

Data Fields

uint8_t	**useBPSK**
uint8_t	**din**
uint8_t	**dout**
uint8_t	**report_period_length**

Detailed Description

Parameters how the stream mode should work

Field Documentation

◆ din

uint8_t st25r3911StreamConfig::din

the divider for the in subcarrier frequency: fc/2^din

◆ dout

uint8_t st25r3911StreamConfig::dout

the divider for the in subcarrier frequency fc/2^dout

◆ report_period_length

uint8_t st25r3911StreamConfig::report_period_length

the length of the reporting period 2^report_period_length

◆ useBPSK

uint8_t st25r3911StreamConfig::useBPSK

0: subcarrier, 1:BPSK

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

- **st25r3911.h**

RFAL

ST25R3911_Com

RFAL ST25R3911 Communication. More...

Collaboration diagram for ST25R3911_Com:

Macros

#define **ST25R3911_FIFO_STATUS_LEN** 2

#define **ST25R3911_REG_IO_CONF1** 0x00U

#define **ST25R3911_REG_IO_CONF2** 0x01U

#define **ST25R3911_REG_OP_CONTROL** 0x02U

#define **ST25R3911_REG_MODE** 0x03U

#define **ST25R3911_REG_BIT_RATE** 0x04U

#define **ST25R3911_REG_ISO14443A_NFC** 0x05U

#define **ST25R3911_REG_ISO14443B_1** 0x06U

#define **ST25R3911_REG_ISO14443B_2** 0x07U

#define **ST25R3911_REG_STREAM_MODE** 0x08U

#define **ST25R3911_REG_AUX** 0x09U

#define **ST25R3911_REG_RX_CONF1** 0x0AU

#define **ST25R3911_REG_RX_CONF2** 0x0BU

#define **ST25R3911_REG_RX_CONF3** 0x0CU

#define **ST25R3911_REG_RX_CONF4** 0x0DU

#define **ST25R3911_REG_MASK_RX_TIMER** 0x0EU

#define **ST25R3911_REG_NO_RESPONSE_TIMER1** 0x0FU

#define **ST25R3911_REG_NO_RESPONSE_TIMER2** 0x10U

#define **ST25R3911_REG_GPT_CONTROL** 0x11U

```
#define ST25R3911_REG_GPT1 0x12U
#define ST25R3911_REG_GPT2 0x13U
#define ST25R3911_REG_IRQ_MASK_MAIN 0x14U
#define ST25R3911_REG_IRQ_MASK_TIMER_NFC 0x15U
#define ST25R3911_REG_IRQ_MASK_ERROR_WUP 0x16U
#define ST25R3911_REG_IRQ_MAIN 0x17U
#define ST25R3911_REG_IRQ_TIMER_NFC 0x18U
#define ST25R3911_REG_IRQ_ERROR_WUP 0x19U
#define ST25R3911_REG_FIFO_RX_STATUS1 0x1AU
#define ST25R3911_REG_FIFO_RX_STATUS2 0x1BU
#define ST25R3911_REG_COLLISION_STATUS 0x1CU
#define ST25R3911_REG_NUM_TX_BYTES1 0x1DU
#define ST25R3911_REG_NUM_TX_BYTES2 0x1EU
#define ST25R3911_REG_NFCIP1_BIT_RATE 0x1FU
#define ST25R3911_REG_AD_RESULT 0x20U
#define ST25R3911_REG_ANT_CAL_CONTROL 0x21U
#define ST25R3911_REG_ANT_CAL_TARGET 0x22U
#define ST25R3911_REG_ANT_CAL_RESULT 0x23U
#define ST25R3911_REG_AM_MOD_DEPTH_CONTROL 0x24U
#define ST25R3911_REG_AM_MOD_DEPTH_RESULT 0x25U
#define ST25R3911_REG_RFO_AM_ON_LEVEL 0x26U
```

```
#define ST25R3911_REG_RFO_AM_OFF_LEVEL 0x27U
#define ST25R3911_REG_FIELD_THRESHOLD 0x29U
#define ST25R3911_REG_REGULATOR_CONTROL 0x2AU
#define ST25R3911_REG_REGULATOR_RESULT 0x2BU
#define ST25R3911_REG_RSSI_RESULT 0x2CU
#define ST25R3911_REG_GAIN_RED_STATE 0x2DU
#define ST25R3911_REG_CAP_SENSOR_CONTROL 0x2EU
#define ST25R3911_REG_CAP_SENSOR_RESULT 0x2FU
#define ST25R3911_REG_AUX_DISPLAY 0x30U
#define ST25R3911_REG_WUP_TIMER_CONTROL 0x31U
#define ST25R3911_REG_AMPLITUDE_MEASURE_CONF 0x32U
#define ST25R3911_REG_AMPLITUDE_MEASURE_REF 0x33U
#define ST25R3911_REG_AMPLITUDE_MEASURE_AA_RESULT 0x34U
#define ST25R3911_REG_AMPLITUDE_MEASURE_RESULT 0x35U
#define ST25R3911_REG_PHASE_MEASURE_CONF 0x36U
#define ST25R3911_REG_PHASE_MEASURE_REF 0x37U
#define ST25R3911_REG_PHASE_MEASURE_AA_RESULT 0x38U
#define ST25R3911_REG_PHASE_MEASURE_RESULT 0x39U
#define ST25R3911_REG_CAPACITANCE_MEASURE_CONF 0x3AU
#define ST25R3911_REG_CAPACITANCE_MEASURE_REF 0x3BU
#define ST25R3911_REG_CAPACITANCE_MEASURE_AA_RESULT 0x3CU
```

#define **ST25R3911_REG_CAPACITANCE_MEASURE_RESULT** 0x3DU

#define **ST25R3911_REG_IC_IDENTITY** 0x3FU

Functions

void **st25r3911ReadRegister** (uint8_t reg, uint8_t *value)
Returns the content of a register within the ST25R3911. More...

void **st25r3911WriteRegister** (uint8_t reg, uint8_t value)
Writes a given value to a register within the ST25R3911. More...

void **st25r3911ClrRegisterBits** (uint8_t reg, uint8_t clr_mask)
Cleart bits on Register. More...

void **st25r3911SetRegisterBits** (uint8_t reg, uint8_t set_mask)
Set bits on Register. More...

void **st25r3911ChangeRegisterBits** (uint8_t reg, uint8_t valueMask, uint8_t value)
Changes the given bits on a ST25R3911 register. More...

void **st25r3911ReadTestRegister** (uint8_t reg, uint8_t *value)
Read a test register within the ST25R3911. More...

void **st25r3911WriteTestRegister** (uint8_t reg, uint8_t value)
Writes a given value to a test register within the ST25R3911. More...

void **st25r3911ModifyRegister** (uint8_t reg, uint8_t clr_mask, uint8_t set_mask)
Modifies a value within a ST25R3911 register. More...

void **st25r3911ChangeTestRegisterBits** (uint8_t reg, uint8_t valueMask, uint8_t value)
Changes the given bits on a ST25R3911 Test register. More...

void **st25r3911WriteMultipleRegisters** (uint8_t reg, const uint8_t *values, uint8_t length)
Writes multiple values to ST25R3911 registers. More...

void **st25r3911ReadMultipleRegisters** (uint8_t reg, uint8_t *values, uint8_t length)
Reads from multiple ST25R3911 registers. More...

void	**st25r3911WriteFifo** (const uint8_t *values, uint8_t length) Writes values to ST25R3911 FIFO. More...
void	**st25r3911ReadFifo** (uint8_t *buf, uint8_t length) Read values from ST25R3911 FIFO. More...
void	**st25r3911ExecuteCommand** (uint8_t cmd) Execute a direct command. More...
void	**st25r3911ExecuteCommands** (const uint8_t *cmds, uint8_t length) Execute several direct commands. More...
bool	**st25r3911IsRegValid** (uint8_t reg) Check if register ID is valid. More...

Detailed Description

Macro Definition Documentation

◆ ST25R3911_FIFO_STATUS_LEN

```
#define ST25R3911_FIFO_STATUS_LEN   2
```

Number of FIFO Status Register

◆ ST25R3911_REG_AD_RESULT

```
#define ST25R3911_REG_AD_RESULT   0x20U
```

R A/D Converter Output Register

◆ ST25R3911_REG_AM_MOD_DEPTH_CONTROL

```
#define ST25R3911_REG_AM_MOD_DEPTH_CONTROL   0x24U
```

RW AM Modulation Depth Control Register

◆ ST25R3911_REG_AM_MOD_DEPTH_RESULT

```
#define ST25R3911_REG_AM_MOD_DEPTH_RESULT   0x25U
```

R AM Modulation Depth Display Register

◆ ST25R3911_REG_AMPLITUDE_MEASURE_AA_RESULT

```
#define ST25R3911_REG_AMPLITUDE_MEASURE_AA_RESULT   0x34U
```

R Amplitude Measurement Auto Averaging Display Register

◆ ST25R3911_REG_AMPLITUDE_MEASURE_CONF

```
#define ST25R3911_REG_AMPLITUDE_MEASURE_CONF   0x32U
```

RW Amplitude Measurement Configuration Register

◆ ST25R3911_REG_AMPLITUDE_MEASURE_REF

```
#define ST25R3911_REG_AMPLITUDE_MEASURE_REF   0x33U
```

RW Amplitude Measurement Reference Register

◆ ST25R3911_REG_AMPLITUDE_MEASURE_RESULT

```
#define ST25R3911_REG_AMPLITUDE_MEASURE_RESULT   0x35U
```

R Amplitude Measurement Display Register

◆ ST25R3911_REG_ANT_CAL_CONTROL

```
#define ST25R3911_REG_ANT_CAL_CONTROL   0x21U
```

RW Antenna Calibration Control Register

◆ ST25R3911_REG_ANT_CAL_RESULT

```
#define ST25R3911_REG_ANT_CAL_RESULT   0x23U
```

R Antenna Calibration Display Register

◆ ST25R3911_REG_ANT_CAL_TARGET

```
#define ST25R3911_REG_ANT_CAL_TARGET   0x22U
```

RW Antenna Calibration Target Register

◆ ST25R3911_REG_AUX

```
#define ST25R3911_REG_AUX   0x09U
```

RW Auxiliary Definition Register

◆ ST25R3911_REG_AUX_DISPLAY

```
#define ST25R3911_REG_AUX_DISPLAY   0x30U
```

R Auxiliary Display Register

◆ ST25R3911_REG_BIT_RATE

```
#define ST25R3911_REG_BIT_RATE   0x04U
```

RW Bit Rate Definition Register

◆ ST25R3911_REG_CAP_SENSOR_CONTROL

```
#define ST25R3911_REG_CAP_SENSOR_CONTROL   0x2EU
```

RW Capacitive Sensor Control Register

◆ ST25R3911_REG_CAP_SENSOR_RESULT

```
#define ST25R3911_REG_CAP_SENSOR_RESULT   0x2FU
```

R Capacitive Sensor Display Register

◆ ST25R3911_REG_CAPACITANCE_MEASURE_AA_RESULT

```
#define ST25R3911_REG_CAPACITANCE_MEASURE_AA_RESULT   0x3CU
```

R Capacitance Measurement Auto Averaging Display Register

◆ ST25R3911_REG_CAPACITANCE_MEASURE_CONF

```
#define ST25R3911_REG_CAPACITANCE_MEASURE_CONF   0x3AU
```

RW Capacitance Measurement Configuration Register

◆ ST25R3911_REG_CAPACITANCE_MEASURE_REF

```
#define ST25R3911_REG_CAPACITANCE_MEASURE_REF   0x3BU
```

RW Capacitance Measurement Reference Register

◆ ST25R3911_REG_CAPACITANCE_MEASURE_RESULT

```
#define ST25R3911_REG_CAPACITANCE_MEASURE_RESULT   0x3DU
```

R Capacitance Measurement Display Register

◆ ST25R3911_REG_COLLISION_STATUS

```
#define ST25R3911_REG_COLLISION_STATUS   0x1CU
```

R Collision Display Register

◆ ST25R3911_REG_FIELD_THRESHOLD

```
#define ST25R3911_REG_FIELD_THRESHOLD   0x29U
```

RW External Field Detector Threshold Register

◆ ST25R3911_REG_FIFO_RX_STATUS1

```
#define ST25R3911_REG_FIFO_RX_STATUS1   0x1AU
```

R FIFO RX Status Register 1

◆ ST25R3911_REG_FIFO_RX_STATUS2

```
#define ST25R3911_REG_FIFO_RX_STATUS2   0x1BU
```

R FIFO RX Status Register 2

◆ ST25R3911_REG_GAIN_RED_STATE

```
#define ST25R3911_REG_GAIN_RED_STATE   0x2DU
```

R Gain Reduction State Register

◆ ST25R3911_REG_GPT1

```
#define ST25R3911_REG_GPT1   0x12U
```

RW General Purpose Timer Register 1

◆ ST25R3911_REG_GPT2

```
#define ST25R3911_REG_GPT2   0x13U
```

RW General Purpose Timer Register 2

◆ ST25R3911_REG_GPT_CONTROL

```
#define ST25R3911_REG_GPT_CONTROL   0x11U
```

RW General Purpose Timer Control Register

◆ ST25R3911_REG_IC_IDENTITY

```
#define ST25R3911_REG_IC_IDENTITY   0x3FU
```

R Chip Id: 0 for old silicon, v2 silicon: 0x09

◆ ST25R3911_REG_IO_CONF1

```
#define ST25R3911_REG_IO_CONF1   0x00U
```

RW IO Configuration Register 1

◆ ST25R3911_REG_IO_CONF2

```
#define ST25R3911_REG_IO_CONF2   0x01U
```

RW IO Configuration Register 2

◆ ST25R3911_REG_IRQ_ERROR_WUP

```
#define ST25R3911_REG_IRQ_ERROR_WUP   0x19U
```

R Error and Wake-up Interrupt Register

◆ ST25R3911_REG_IRQ_MAIN

```
#define ST25R3911_REG_IRQ_MAIN   0x17U
```

R Main Interrupt Register

◆ ST25R3911_REG_IRQ_MASK_ERROR_WUP

```
#define ST25R3911_REG_IRQ_MASK_ERROR_WUP   0x16U
```

RW Mask Error and Wake-up Interrupt Register

◆ ST25R3911_REG_IRQ_MASK_MAIN

```
#define ST25R3911_REG_IRQ_MASK_MAIN   0x14U
```

RW Mask Main Interrupt Register

◆ ST25R3911_REG_IRQ_MASK_TIMER_NFC

```
#define ST25R3911_REG_IRQ_MASK_TIMER_NFC   0x15U
```

RW Mask Timer and NFC Interrupt Register

◆ ST25R3911_REG_IRQ_TIMER_NFC

```
#define ST25R3911_REG_IRQ_TIMER_NFC   0x18U
```

R Timer and NFC Interrupt Register

◆ ST25R3911_REG_ISO14443A_NFC

```
#define ST25R3911_REG_ISO14443A_NFC   0x05U
```

RW ISO14443A and NFC 106 kBit/s Settings Register

◆ ST25R3911_REG_ISO14443B_1

```
#define ST25R3911_REG_ISO14443B_1   0x06U
```

RW ISO14443B Settings Register 1

◆ ST25R3911_REG_ISO14443B_2

```
#define ST25R3911_REG_ISO14443B_2   0x07U
```

RW ISO14443B Settings Register 2

◆ ST25R3911_REG_MASK_RX_TIMER

```
#define ST25R3911_REG_MASK_RX_TIMER   0x0EU
```

RW Mask Receive Timer Register

◆ ST25R3911_REG_MODE

```
#define ST25R3911_REG_MODE   0x03U
```

RW Mode Definition Register

◆ ST25R3911_REG_NFCIP1_BIT_RATE

```
#define ST25R3911_REG_NFCIP1_BIT_RATE   0x1FU
```

R NFCIP Bit Rate Detection Display Register

◆ ST25R3911_REG_NO_RESPONSE_TIMER1

```
#define ST25R3911_REG_NO_RESPONSE_TIMER1   0x0FU
```

RW No-response Timer Register 1

◆ ST25R3911_REG_NO_RESPONSE_TIMER2

```
#define ST25R3911_REG_NO_RESPONSE_TIMER2   0x10U
```

RW No-response Timer Register 2

◆ ST25R3911_REG_NUM_TX_BYTES1

```
#define ST25R3911_REG_NUM_TX_BYTES1   0x1DU
```

RW Number of Transmitted Bytes Register 1

◆ ST25R3911_REG_NUM_TX_BYTES2

```
#define ST25R3911_REG_NUM_TX_BYTES2   0x1EU
```

RW Number of Transmitted Bytes Register 2

◆ ST25R3911_REG_OP_CONTROL

```
#define ST25R3911_REG_OP_CONTROL   0x02U
```

RW Operation Control Register

◆ ST25R3911_REG_PHASE_MEASURE_AA_RESULT

```
#define ST25R3911_REG_PHASE_MEASURE_AA_RESULT   0x38U
```

R Phase Measurement Auto Averaging Display Register

◆ ST25R3911_REG_PHASE_MEASURE_CONF

```
#define ST25R3911_REG_PHASE_MEASURE_CONF   0x36U
```

RW Phase Measurement Configuration Register

◆ ST25R3911_REG_PHASE_MEASURE_REF

```
#define ST25R3911_REG_PHASE_MEASURE_REF   0x37U
```

RW Phase Measurement Reference Register

◆ ST25R3911_REG_PHASE_MEASURE_RESULT

```
#define ST25R3911_REG_PHASE_MEASURE_RESULT   0x39U
```

R Phase Measurement Display Register

◆ ST25R3911_REG_REGULATOR_CONTROL

```
#define ST25R3911_REG_REGULATOR_CONTROL   0x2AU
```

RW Regulated Voltage Control Register

◆ ST25R3911_REG_REGULATOR_RESULT

```
#define ST25R3911_REG_REGULATOR_RESULT   0x2BU
```

R Regulator Display Register

◆ ST25R3911_REG_RFO_AM_OFF_LEVEL

```
#define ST25R3911_REG_RFO_AM_OFF_LEVEL   0x27U
```

RW RFO Normal (AM Off) Level Definition Register

◆ ST25R3911_REG_RFO_AM_ON_LEVEL

```
#define ST25R3911_REG_RFO_AM_ON_LEVEL   0x26U
```

RW RFO AM Modulation (On) Level Definition Register

◆ ST25R3911_REG_RSSI_RESULT

```
#define ST25R3911_REG_RSSI_RESULT   0x2CU
```

R RSSI Display Register

◆ ST25R3911_REG_RX_CONF1

```
#define ST25R3911_REG_RX_CONF1   0x0AU
```

RW Receiver Configuration Register 1

◆ ST25R3911_REG_RX_CONF2

```
#define ST25R3911_REG_RX_CONF2   0x0BU
```

RW Receiver Configuration Register 2

◆ ST25R3911_REG_RX_CONF3

```
#define ST25R3911_REG_RX_CONF3   0x0CU
```

RW Receiver Configuration Register 3

◆ ST25R3911_REG_RX_CONF4

```
#define ST25R3911_REG_RX_CONF4   0x0DU
```

RW Receiver Configuration Register 4

◆ ST25R3911_REG_STREAM_MODE

```
#define ST25R3911_REG_STREAM_MODE   0x08U
```

RW Stream Mode Definition Register

◆ ST25R3911_REG_WUP_TIMER_CONTROL

```
#define ST25R3911_REG_WUP_TIMER_CONTROL   0x31U
```

RW Wake-up Timer Control Register

Function Documentation

◆ st25r3911ChangeRegisterBits()

```
void st25r3911ChangeRegisterBits ( uint8_t  reg,
                                   uint8_t  valueMask,
                                   uint8_t  value
                                 )
```

This function is used if only a particular bits should be changed within an ST25R3911 register.

Warning

This method does not guarantee consistency of register content when called from multiple contexts (task, ISR, thread)

Parameters

[in]	**reg**	Address of the register to write.
[in]	**valueMask**	bitmask of bits to be changed
[in]	**value**	the bits to be written on the enabled valueMask bits

◆ st25r3911ChangeTestRegisterBits()

```
void st25r3911ChangeTestRegisterBits ( uint8_t  reg,
                                       uint8_t  valueMask,
                                       uint8_t  value
                                     )
```

This function is used if only a particular bits should be changed within an ST25R3911 register.

Parameters

[in]	**reg**	Address of the Test register to change.
[in]	**valueMask**	bitmask of bits to be changed
[in]	**value**	the bits to be written on the enabled valueMask bits

Warning

This method does not guarantee consistency of register content when called from multiple contexts (task, ISR, thread)

◆ st25r3911ClrRegisterBits()

```
void st25r3911ClrRegisterBits ( uint8_t  reg,
                                uint8_t  clr_mask
                              )
```

This function clears the given bitmask on the register

Warning

This method does not guarantee consistency of register content when called from multiple contexts (task, ISR, thread)

Parameters

[in] **reg** Address of the register clear

[in] **clr_mask** Bitmask of bit to be cleared

◆ st25r3911ExecuteCommand()

```
void st25r3911ExecuteCommand ( uint8_t  cmd )
```

This function is used to start so-called direct command. These commands are implemented inside the chip and each command has unique code (see datasheet).

Parameters

[in] **cmd** : code of the direct command to be executed.

◆ st25r3911ExecuteCommands()

```
void st25r3911ExecuteCommands ( const uint8_t *  cmds,
                                uint8_t          length
                              )
```

This function is used to start so-called direct command. These commands are implemented inside the chip and each command has unique code (see datasheet).

Warning

Commands to be executed must support command chaining

Parameters

[in] **cmds** : codes of the direct command to be executed.

[in] **length** : number of commands to be executed.

◆ st25r3911IsRegValid()

```
bool st25r3911IsRegValid ( uint8_t reg )
```

Checks if the given register ID a valid ST25R3911 register

Parameters

[in] **reg** Address of register to check

Returns

true if is a valid register ID

false otherwise

◆ st25r3911ModifyRegister()

```
void st25r3911ModifyRegister ( uint8_t reg,
                               uint8_t clr_mask,
                               uint8_t set_mask
                             )
```

This function is used if only a particular bits should be changed within an ST25R3911 register.

Warning

This method does not guarantee consistency of register content when called from multiple contexts (task, ISR, thread)

Parameters

[in] **reg** Address of the register to write.

[in] **clr_mask** bitmask of bits to be cleared to 0.

[in] **set_mask** bitmask of bits to be set to 1.

◆ st25r3911ReadFifo()

```
void st25r3911ReadFifo ( uint8_t * buf,
                         uint8_t   length
```

```
)
```

This function needs to be called in order to read from ST25R3911 FIFO.

Parameters

[out] **buf** pointer to a buffer where the FIFO content shall be written to.

[in] **length** Number of bytes to read. (= size of *buf*)

Note

: This function doesn't check whether *length* is really the number of available bytes in FIFO

◆ st25r3911ReadMultipleRegisters()

```
void st25r3911ReadMultipleRegisters ( uint8_t    reg,
                                      uint8_t *  values,
                                      uint8_t    length
                                    )
```

This function is used to read from multiple registers using the auto-increment feature. That is, after each read the address pointer inside the ST25R3911 gets incremented automatically.

Parameters

[in] **reg** Address of the frist register to read from.

[in] **values** pointer to a buffer where the result shall be written to.

[in] **length** Number of registers to be read out.

◆ st25r3911ReadRegister()

```
void st25r3911ReadRegister ( uint8_t    reg,
                             uint8_t *  value
                           )
```

Register bit definitions
DOXYGEN_SUPPRESS

This function is used to read out the content of ST25R3911 registers.

Parameters

[in] **reg** Address of register to read.

[out] **value** Returned value.

◆ st25r3911ReadTestRegister()

```
void st25r3911ReadTestRegister ( uint8_t    reg,
                                 uint8_t *  value
                               )
```

This function is used to read the content of test address *reg* within the ST25R3911.

Parameters

[in] **reg** Address of the register to read.

[out] **value** Returned read.

◆ st25r3911SetRegisterBits()

```
void st25r3911SetRegisterBits ( uint8_t  reg,
                                uint8_t  set_mask
                              )
```

This function sets the given bitmask on the register

Warning

This method does not guarantee consistency of register content when called from multiple contexts (task, ISR, thread)

Parameters

[in] **reg** Address of the register clear

[in] **set_mask** Bitmask of bit to be cleared

◆ st25r3911WriteFifo()

```
void st25r3911WriteFifo ( const uint8_t *  values,
                          uint8_t          length
                        )
```

This function needs to be called in order to write to the ST25R3911 FIFO.

Parameters

[in] **values** pointer to a buffer containing the values to be written to the FIFO.

[in] **length** Number of values to be written.

◆ st25r3911WriteMultipleRegisters()

```
void st25r3911WriteMultipleRegisters ( uint8_t        reg,
                                       const uint8_t * values,
                                       uint8_t        length
                                     )
```

This function is used to write multiple values to the ST25R3911 using the auto-increment feature. That is, after each write the address pointer inside the ST25R3911 gets incremented automatically.

Parameters

[in] **reg** Address of the frist register to write.

[in] **values** pointer to a buffer containing the values to be written.

[in] **length** Number of values to be written.

◆ st25r3911WriteRegister()

```
void st25r3911WriteRegister ( uint8_t reg,
                              uint8_t value
                            )
```

This function is used to write *value* to address *reg* within the ST25R3911.

Parameters

[in] **reg** Address of the register to write.

[in] **value** Value to be written.

◆ st25r3911WriteTestRegister()

```
void st25r3911WriteTestRegister ( uint8_t reg,
                                  uint8_t value
                                )
```

This function is used to write *value* to test address *reg* within the ST25R3911.

Parameters

[in] **reg** Address of the register to write.

[in] **value** Value to be written.

ST25R3911_Interrupt

RFAL ST25R3911 Interrupt. More...

Collaboration diagram for ST25R3911_Interrupt:

Macros

#define **ST25R3911_IRQ_MASK_ALL** (uint32_t)(0xFFFFFFU)

#define **ST25R3911_IRQ_MASK_NONE** (uint32_t)(0U)

#define **ST25R3911_IRQ_MASK_OSC** (uint32_t)(0x80U)

#define **ST25R3911_IRQ_MASK_FWL** (uint32_t)(0x40U)

#define **ST25R3911_IRQ_MASK_RXS** (uint32_t)(0x20U)

#define **ST25R3911_IRQ_MASK_RXE** (uint32_t)(0x10U)

#define **ST25R3911_IRQ_MASK_TXE** (uint32_t)(0x08U)

#define **ST25R3911_IRQ_MASK_COL** (uint32_t)(0x04U)

#define **ST25R3911_IRQ_MASK_DCT** (uint32_t)(0x8000U)

#define **ST25R3911_IRQ_MASK_NRE** (uint32_t)(0x4000U)

#define **ST25R3911_IRQ_MASK_GPE** (uint32_t)(0x2000U)

#define **ST25R3911_IRQ_MASK_EON** (uint32_t)(0x1000U)

#define **ST25R3911_IRQ_MASK_EOF** (uint32_t)(0x0800U)

#define **ST25R3911_IRQ_MASK_CAC** (uint32_t)(0x0400U)

#define **ST25R3911_IRQ_MASK_CAT** (uint32_t)(0x0200U)

#define **ST25R3911_IRQ_MASK_NFCT** (uint32_t)(0x0100U)

#define **ST25R3911_IRQ_MASK_CRC** (uint32_t)(0x800000U)

#define **ST25R3911_IRQ_MASK_PAR** (uint32_t)(0x400000U)

#define **ST25R3911_IRQ_MASK_ERR2** (uint32_t)(0x200000U)

#define **ST25R3911_IRQ_MASK_ERR1** (uint32_t)(0x100000U)

#define **ST25R3911_IRQ_MASK_WT** (uint32_t)(0x080000U)

#define **ST25R3911_IRQ_MASK_WAM** (uint32_t)(0x040000U)

#define **ST25R3911_IRQ_MASK_WPH** (uint32_t)(0x020000U)

#define **ST25R3911_IRQ_MASK_WCAP** (uint32_t)(0x010000U)

#define **ST25R3911_IRQ_MASK_TIM** (0x02U)

#define **ST25R3911_IRQ_MASK_ERR** (0x01U)

Functions

uint32_t **st25r3911WaitForInterruptsTimed** (uint32_t mask, uint16_t tmo)
Wait until an ST25R3911 interrupt occurs. More...

uint32_t **st25r3911GetInterrupt** (uint32_t mask)
Get status for the given interrupt. More...

void **st25r3911InitInterrupts** (void)
Init the 3911 interrupt. More...

void **st25r3911ModifyInterrupts** (uint32_t clr_mask, uint32_t set_mask)
Modifies the Interrupt. More...

void **st25r3911CheckForReceivedInterrupts** (void)
Checks received interrupts. More...

void **st25r3911Isr** (void)
ISR Service routine. More...

void **st25r3911EnableInterrupts** (uint32_t mask)
Enable a given ST25R3911 Interrupt source. More...

void **st25r3911DisableInterrupts** (uint32_t mask)
Disable one or more a given ST25R3911 Interrupt sources. More...

void **st25r3911ClearInterrupts** (void)
Clear all st25r3911 irq flags. More...

void **st25r3911IRQCallbackSet** (void(*cb)(void))
Sets IRQ callback for the ST25R3911 interrupt. More...

void **st25r3911IRQCallbackRestore** (void)
Sets IRQ callback for the ST25R3911 interrupt. More...

Detailed Description

Macro Definition Documentation

◆ ST25R3911_IRQ_MASK_ALL

```
#define ST25R3911_IRQ_MASK_ALL   (uint32_t)(0xFFFFFFU)
```

All ST25R3911 interrupt sources

◆ ST25R3911_IRQ_MASK_CAC

```
#define ST25R3911_IRQ_MASK_CAC   (uint32_t)(0x0400U)
```

ST25R3911 collision during RF collision avoidance interrupt

◆ ST25R3911_IRQ_MASK_CAT

```
#define ST25R3911_IRQ_MASK_CAT   (uint32_t)(0x0200U)
```

ST25R3911 minimum guard time expired interrupt

◆ ST25R3911_IRQ_MASK_COL

```
#define ST25R3911_IRQ_MASK_COL   (uint32_t)(0x04U)
```

ST25R3911 bit collision interrupt

◆ ST25R3911_IRQ_MASK_CRC

```
#define ST25R3911_IRQ_MASK_CRC   (uint32_t)(0x800000U)
```

ST25R3911 CRC error interrupt

◆ ST25R3911_IRQ_MASK_DCT

```
#define ST25R3911_IRQ_MASK_DCT   (uint32_t)(0x8000U)
```

ST25R3911 termination of direct command interrupt

◆ ST25R3911_IRQ_MASK_EOF

```
#define ST25R3911_IRQ_MASK_EOF   (uint32_t)(0x0800U)
```

ST25R3911 external field off interrupt

◆ ST25R3911_IRQ_MASK_EON

```
#define ST25R3911_IRQ_MASK_EON   (uint32_t)(0x1000U)
```

ST25R3911 external field on interrupt

◆ ST25R3911_IRQ_MASK_ERR

```
#define ST25R3911_IRQ_MASK_ERR   (0x01U)
```

additional interrupts in ST25R3911_REG_IRQ_ERROR_WUP

◆ ST25R3911_IRQ_MASK_ERR1

```
#define ST25R3911_IRQ_MASK_ERR1   (uint32_t)(0x100000U)
```

ST25R3911 hard framing error interrupt

◆ ST25R3911_IRQ_MASK_ERR2

```
#define ST25R3911_IRQ_MASK_ERR2   (uint32_t)(0x200000U)
```

ST25R3911 soft framing error interrupt

◆ ST25R3911_IRQ_MASK_FWL

```
#define ST25R3911_IRQ_MASK_FWL   (uint32_t)(0x40U)
```

ST25R3911 FIFO water level interrupt

◆ ST25R3911_IRQ_MASK_GPE

```
#define ST25R3911_IRQ_MASK_GPE   (uint32_t)(0x2000U)
```

ST25R3911 general purpose timer expired interrupt

◆ ST25R3911_IRQ_MASK_NFCT

```
#define ST25R3911_IRQ_MASK_NFCT   (uint32_t)(0x0100U)
```

ST25R3911 initiator bit rate recognized interrupt

◆ ST25R3911_IRQ_MASK_NONE

```
#define ST25R3911_IRQ_MASK_NONE   (uint32_t)(0U)
```

No ST25R3911 interrupt source

◆ ST25R3911_IRQ_MASK_NRE

```
#define ST25R3911_IRQ_MASK_NRE   (uint32_t)(0x4000U)
```

ST25R3911 no-response timer expired interrupt

◆ ST25R3911_IRQ_MASK_OSC

```
#define ST25R3911_IRQ_MASK_OSC   (uint32_t)(0x80U)
```

ST25R3911 oscillator stable interrupt

◆ ST25R3911_IRQ_MASK_PAR

```
#define ST25R3911_IRQ_MASK_PAR   (uint32_t)(0x400000U)
```

ST25R3911 parity error interrupt

◆ ST25R3911_IRQ_MASK_RXE

```
#define ST25R3911_IRQ_MASK_RXE   (uint32_t)(0x10U)
```

ST25R3911 end of receive interrupt

◆ ST25R3911_IRQ_MASK_RXS

```
#define ST25R3911_IRQ_MASK_RXS   (uint32_t)(0x20U)
```

ST25R3911 start of receive interrupt

◆ ST25R3911_IRQ_MASK_TIM

```
#define ST25R3911_IRQ_MASK_TIM   (0x02U)
```

additional interrupts in ST25R3911_REG_IRQ_TIMER_NFC

◆ ST25R3911_IRQ_MASK_TXE

```
#define ST25R3911_IRQ_MASK_TXE   (uint32_t)(0x08U)
```

ST25R3911 end of transmission interrupt

◆ ST25R3911_IRQ_MASK_WAM

```
#define ST25R3911_IRQ_MASK_WAM   (uint32_t)(0x040000U)
```

ST25R3911 wake-up due to amplitude interrupt

◆ ST25R3911_IRQ_MASK_WCAP

```
#define ST25R3911_IRQ_MASK_WCAP   (uint32_t)(0x010000U)
```

ST25R3911 wake-up due to capacitance measurement

◆ ST25R3911_IRQ_MASK_WPH

#define ST25R3911_IRQ_MASK_WPH (uint32_t)(0x020000U)

ST25R3911 wake-up due to phase interrupt

◆ ST25R3911_IRQ_MASK_WT

#define ST25R3911_IRQ_MASK_WT (uint32_t)(0x080000U)

ST25R3911 wake-up interrupt

Function Documentation

◆ st25r3911CheckForReceivedInterrupts()

void st25r3911CheckForReceivedInterrupts (void)

Checks received interrupts and saves the result into global params

◆ st25r3911ClearInterrupts()

void st25r3911ClearInterrupts (void)

◆ st25r3911DisableInterrupts()

void st25r3911DisableInterrupts (uint32_t mask)

This function disables all interrupts given by *mask*. 0xff disables all.

Parameters

[in] **mask** mask indicating the interrupts to be disabled.

◆ st25r3911EnableInterrupts()

void st25r3911EnableInterrupts (uint32_t mask)

This function enables all interrupts given by *mask*, ST25R3911_IRQ_MASK_ALL enables all interrupts.

Parameters

[in] **mask** mask indicating the interrupts to be enabled

◆ st25r3911GetInterrupt()

uint32_t st25r3911GetInterrupt (uint32_t mask)

This function is used to check whether the interrupt given by *mask* has occurred. If yes the interrupt gets cleared. This function returns only status bits which are inside *mask*.

Parameters

[in] **mask** : mask indicating the interrupt to check for.

Returns

the mask of the interrupts occurred

◆ st25r3911InitInterrupts()

void st25r3911InitInterrupts (void)

This function initiates the 3911 interrupts.

◆ st25r3911IRQCallbackRestore()

void st25r3911IRQCallbackRestore (void)

◆ st25r3911IRQCallbackSet()

void st25r3911IRQCallbackSet (void(*)(void) cb)

Parameters

[in] **cb** pointer to the callback method

◆ st25r3911Isr()

void st25r3911Isr (void)

This function modifies the interrupt

◆ st25r3911ModifyInterrupts()

```
void st25r3911ModifyInterrupts ( uint32_t  clr_mask,
                                 uint32_t  set_mask
                               )
```

This function modifies the interrupt

Parameters

[in] **clr_mask** : bit mask to be cleared on the interrupt mask

[in] **set_mask** : bit mask to be set on the interrupt mask

◆ st25r3911WaitForInterruptsTimed()

```
uint32_t st25r3911WaitForInterruptsTimed ( uint32_t  mask,
                                           uint16_t  tmo
                                         )
```

This function is used to access the ST25R3911 interrupt flags. Use this to wait for max. *tmo* milliseconds for the **first** interrupt indicated with mask *mask* to occur.

Parameters

[in] **mask** : mask indicating the interrupts to wait for.

[in] **tmo** : time in milliseconds until timeout occurs. If set to 0 the functions waits forever.

Returns

: 0 if timeout occurred otherwise a mask indicating the cleared interrupts.

ST25R3916

RFAL ST25R3916 Driver. More...

Collaboration diagram for ST25R3916:

Modules

ST25R3916_Driver
RFAL ST25R3916 Driver.

ST25R3916_COM
RFAL ST25R3916 Communications.

ST25R3916_IRQ
RFAL ST25R3916 IRQ.

ST25R3916_LED
RFAL ST25R3916 LED.

Detailed Description

ST25R3916_Driver

RFAL ST25R3916 Driver. More...

Collaboration diagram for ST25R3916_Driver:

Data Structures

struct **t_st25r3916Regs**

struct **st25r3916StreamConfig**

Macros

#define **ST25R3916_CMD_SET_DEFAULT** 0xC1U

#define **ST25R3916_CMD_STOP** 0xC2U

#define **ST25R3916_CMD_TRANSMIT_WITH_CRC** 0xC4U

```
#define ST25R3916_CMD_TRANSMIT_WITHOUT_CRC 0xC5U
#define ST25R3916_CMD_TRANSMIT_REQA 0xC6U
#define ST25R3916_CMD_TRANSMIT_WUPA 0xC7U
#define ST25R3916_CMD_INITIAL_RF_COLLISION 0xC8U
#define ST25R3916_CMD_RESPONSE_RF_COLLISION_N 0xC9U
#define ST25R3916_CMD_GOTO_SENSE 0xCDU
#define ST25R3916_CMD_GOTO_SLEEP 0xCEU
#define ST25R3916_CMD_MASK_RECEIVE_DATA 0xD0U
#define ST25R3916_CMD_UNMASK_RECEIVE_DATA 0xD1U
#define ST25R3916_CMD_AM_MOD_STATE_CHANGE 0xD2U
#define ST25R3916_CMD_MEASURE_AMPLITUDE 0xD3U
#define ST25R3916_CMD_RESET_RXGAIN 0xD5U
#define ST25R3916_CMD_ADJUST_REGULATORS 0xD6U
#define ST25R3916_CMD_CALIBRATE_DRIVER_TIMING 0xD8U
#define ST25R3916_CMD_MEASURE_PHASE 0xD9U
#define ST25R3916_CMD_CLEAR_RSSI 0xDAU
#define ST25R3916_CMD_CLEAR_FIFO 0xDBU
#define ST25R3916_CMD_TRANSPARENT_MODE 0xDCU
#define ST25R3916_CMD_MEASURE_VDD 0xDFU
#define ST25R3916_CMD_START_GP_TIMER 0xE0U
#define ST25R3916_CMD_START_WUP_TIMER 0xE1U
```

#define **ST25R3916_CMD_START_MASK_RECEIVE_TIMER** 0xE2U

#define **ST25R3916_CMD_START_NO_RESPONSE_TIMER** 0xE3U

#define **ST25R3916_CMD_START_PPON2_TIMER** 0xE4U

#define **ST25R3916_CMD_STOP_NRT** 0xE8U

#define **ST25R3916_CMD_SPACE_B_ACCESS** 0xFBU

#define **ST25R3916_CMD_TEST_ACCESS** 0xFCU

#define **ST25R3916_THRESHOLD_DO_NOT_SET** 0xFFU

#define **ST25R3916_BR_DO_NOT_SET** 0xFFU

#define **ST25R3916_BR_106** 0x00U

#define **ST25R3916_BR_212** 0x01U

#define **ST25R3916_BR_424** 0x02U

#define **ST25R3916_BR_848** 0x03U

#define **ST25R3916_BR_1695** 0x04U

#define **ST25R3916_BR_3390** 0x05U

#define **ST25R3916_BR_6780** 0x07U

#define **ST25R3916_FIFO_DEPTH** 512U

#define **ST25R3916_TOUT_OSC_STABLE** 10U

#define **st25r3916TxRxOn**() **st25r3916SetRegisterBits**(**ST25R3916_REG_OP_CONTROL**, (ST25R3916_REG_OP_CONTROL_rx_en | ST25R3916_REG_OP_CONTROL_tx_en))

#define **st25r3916TxRxOff**() **st25r3916ClrRegisterBits**(**ST25R3916_REG_OP_CONTROL**, (ST25R3916_REG_OP_CONTROL_rx_en | ST25R3916_REG_OP_CONTROL_tx_en))

#define **st25r3916TxOff**() **st25r3916ClrRegisterBits**(**ST25R3916_REG_OP_CONTROL**, ST25R3916_REG_OP_CONTROL_tx_en)

#define **st25r3916IsGPTRunning**() **st25r3916CheckReg**(**ST25R3916_REG_NFCIP1_BIT_RATE**, ST25R3916_REG_NFCIP1_BIT_RATE_gpt_on, ST25R3916_REG_NFCIP1_BIT_RATE_gpt_on)

#define **st25r3916IsExtFieldOn**() **st25r3916CheckReg**(**ST25R3916_REG_AUX_DISPLAY**, ST25R3916_REG_AUX_DISPLAY_efd_o, ST25R3916_REG_AUX_DISPLAY_efd_o)

#define **st25r3916IsTxEnabled**() **st25r3916CheckReg**(**ST25R3916_REG_OP_CONTROL**, ST25R3916_REG_OP_CONTROL_tx_en, ST25R3916_REG_OP_CONTROL_tx_en)

#define **st25r3916IsNRTinEMV**() **st25r3916CheckReg**(**ST25R3916_REG_TIMER_EMV_CONTROL**, ST25R3916_REG_TIMER_EMV_CONTROL_nrt_emv, ST25R3916_REG_TIMER_EMV_CONTROL_nrt_emv_on)

#define **st25r3916IsLastFIFOComplete**() **st25r3916CheckReg**(**ST25R3916_REG_FIFO_STATUS2**, ST25R3916_REG_FIFO_STATUS2_fifo_lb_mask, 0)

#define **st25r3916IsOscOn**() **st25r3916CheckReg**(**ST25R3916_REG_OP_CONTROL**, ST25R3916_REG_OP_CONTROL_en, ST25R3916_REG_OP_CONTROL_en)

#define **st25r3916IsAATOn**() **st25r3916CheckReg**(**ST25R3916_REG_IO_CONF2**, ST25R3916_REG_IO_CONF2_aat_en, ST25R3916_REG_IO_CONF2_aat_en)

Functions

ReturnCode **st25r3916Initialize** (void)

Initialise ST25R3916 driver. More...

void **st25r3916Deinitialize** (void)

Deinitialize ST25R3916 driver. More...

ReturnCode **st25r3916OscOn** (void)

Turn on Oscillator and Regulator. More...

ReturnCode **st25r3916SetBitrate** (uint8_t txrate, uint8_t rxrate)

Sets the bitrate. More...

ReturnCode **st25r3916AdjustRegulators** (uint16_t *result_mV)

Adjusts supply regulators according to the current supply voltage. More...

ReturnCode **st25r3916SetRegulators** (uint8_t regulation)

Sets supply regulators. More...

ReturnCode	**st25r3916MeasureAmplitude** (uint8_t *result) Measure Amplitude. More...
uint8_t	**st25r3916MeasurePowerSupply** (uint8_t mpsv) Measure Power Supply. More...
uint16_t	**st25r3916MeasureVoltage** (uint8_t mpsv) Measure Voltage. More...
ReturnCode	**st25r3916MeasurePhase** (uint8_t *result) Measure Phase. More...
ReturnCode	**st25r3916MeasureCapacitance** (uint8_t *result) Measure Capacitance. More...
ReturnCode	**st25r3916CalibrateCapacitiveSensor** (uint8_t *result) Calibrates Capacitive Sensor. More...
uint32_t	**st25r3916GetNoResponseTime** (void) Get NRT time. More...
ReturnCode	**st25r3916SetNoResponseTime** (uint32_t nrt_64fcs) Set NRT time. More...
ReturnCode	**st25r3916SetStartNoResponseTimer** (uint32_t nrt_64fcs) Set and Start NRT. More...
void	**st25r3916SetGPTime** (uint16_t gpt_8fcs) Set GPT time. More...
ReturnCode	**st25r3916SetStartGPTimer** (uint16_t gpt_8fcs, uint8_t trigger_source) Set and Start GPT. More...
void	**st25r3916SetNumTxBits** (uint16_t nBits) Sets the number Tx Bits. More...
uint16_t	**st25r3916GetNumFIFOBytes** (void)

	Get amount of bytes in FIFO. More...
uint8_t	**st25r3916GetNumFIFOLastBits** (void) Get amount of bits of the last FIFO byte if incomplete. More...
ReturnCode	**st25r3916PerformCollisionAvoidance** (uint8_t FieldONCmd, uint8_t pdThreshold, uint8_t caThreshold, uint8_t nTRFW) Perform Collision Avoidance. More...
bool	**st25r3916CheckChipID** (uint8_t *rev) Check Identity. More...
ReturnCode	**st25r3916GetRegsDump** (**t_st25r3916Regs** *regDump) Retrieves all internal registers from ST25R3916. More...
bool	**st25r3916IsCmdValid** (uint8_t cmd) Check if command is valid. More...
ReturnCode	**st25r3916StreamConfigure** (const struct **st25r3916StreamConfig** *config) Configure the stream mode of ST25R3916. More...
ReturnCode	**st25r3916ExecuteCommandAndGetResult** (uint8_t cmd, uint8_t resReg, uint8_t tOut, uint8_t *result) Executes a direct command and returns the result. More...
ReturnCode	**st25r3916GetRSSI** (uint16_t *amRssi, uint16_t *pmRssi) Gets the RSSI values. More...
ReturnCode	**st25r3916SetAntennaMode** (bool single, bool rfiox) Set Antenna mode. More...

Detailed Description

Macro Definition Documentation

◆ ST25R3916_BR_106

```
#define ST25R3916_BR_106   0x00U
```

ST25R3916 Bit Rate 106 kbit/s (fc/128)

◆ ST25R3916_BR_1695

```
#define ST25R3916_BR_1695   0x04U
```

ST25R3916 Bit Rate 1696 kbit/s (fc/8)

◆ ST25R3916_BR_212

```
#define ST25R3916_BR_212   0x01U
```

ST25R3916 Bit Rate 212 kbit/s (fc/64)

◆ ST25R3916_BR_3390

```
#define ST25R3916_BR_3390   0x05U
```

ST25R3916 Bit Rate 3390 kbit/s (fc/4)

◆ ST25R3916_BR_424

```
#define ST25R3916_BR_424   0x02U
```

ST25R3916 Bit Rate 424 kbit/s (fc/32)

◆ ST25R3916_BR_6780

```
#define ST25R3916_BR_6780   0x07U
```

ST25R3916 Bit Rate 6780 kbit/s (fc/2)

◆ ST25R3916_BR_848

```
#define ST25R3916_BR_848   0x03U
```

ST25R3916 Bit Rate 848 kbit/s (fc/16)

◆ ST25R3916_BR_DO_NOT_SET

```
#define ST25R3916_BR_DO_NOT_SET   0xFFU
```

Indicates not to change this Bit Rate

◆ ST25R3916_CMD_ADJUST_REGULATORS

```
#define ST25R3916_CMD_ADJUST_REGULATORS   0xD6U
```

Adjust regulators

◆ ST25R3916_CMD_AM_MOD_STATE_CHANGE

```
#define ST25R3916_CMD_AM_MOD_STATE_CHANGE   0xD2U
```

AM Modulation state change

◆ ST25R3916_CMD_CALIBRATE_DRIVER_TIMING

```
#define ST25R3916_CMD_CALIBRATE_DRIVER_TIMING   0xD8U
```

Starts the sequence to adjust the driver timing

◆ ST25R3916_CMD_CLEAR_FIFO

```
#define ST25R3916_CMD_CLEAR_FIFO   0xDBU
```

Clears FIFO, Collision and IRQ status

◆ ST25R3916_CMD_CLEAR_RSSI

```
#define ST25R3916_CMD_CLEAR_RSSI   0xDAU
```

Clear RSSI bits and restart the measurement

◆ ST25R3916_CMD_GOTO_SENSE

```
#define ST25R3916_CMD_GOTO_SENSE   0xCDU
```

Passive target logic to Sense/Idle state

◆ ST25R3916_CMD_GOTO_SLEEP

```
#define ST25R3916_CMD_GOTO_SLEEP   0xCEU
```

Passive target logic to Sleep/Halt state

◆ ST25R3916_CMD_INITIAL_RF_COLLISION

```
#define ST25R3916_CMD_INITIAL_RF_COLLISION   0xC8U
```

NFC transmit with Initial RF Collision Avoidance

◆ ST25R3916_CMD_MASK_RECEIVE_DATA

```
#define ST25R3916_CMD_MASK_RECEIVE_DATA   0xD0U
```

Mask receive data

◆ ST25R3916_CMD_MEASURE_AMPLITUDE

```
#define ST25R3916_CMD_MEASURE_AMPLITUDE   0xD3U
```

Measure singal amplitude on RFI inputs

◆ ST25R3916_CMD_MEASURE_PHASE

```
#define ST25R3916_CMD_MEASURE_PHASE   0xD9U
```

Measure phase between RFO and RFI signal

◆ ST25R3916_CMD_MEASURE_VDD

```
#define ST25R3916_CMD_MEASURE_VDD   0xDFU
```

Measure power supply voltage

◆ ST25R3916_CMD_RESET_RXGAIN

```
#define ST25R3916_CMD_RESET_RXGAIN   0xD5U
```

Reset RX Gain

◆ ST25R3916_CMD_RESPONSE_RF_COLLISION_N

```
#define ST25R3916_CMD_RESPONSE_RF_COLLISION_N   0xC9U
```

NFC transmit with Response RF Collision Avoidance

◆ ST25R3916_CMD_SET_DEFAULT

```
#define ST25R3916_CMD_SET_DEFAULT   0xC1U
```

Puts the chip in default state (same as after power-up)

◆ ST25R3916_CMD_SPACE_B_ACCESS

```
#define ST25R3916_CMD_SPACE_B_ACCESS   0xFBU
```

Enable R/W access to the space B registers

◆ ST25R3916_CMD_START_GP_TIMER

```
#define ST25R3916_CMD_START_GP_TIMER   0xE0U
```

Start the general purpose timer

◆ ST25R3916_CMD_START_MASK_RECEIVE_TIMER

```
#define ST25R3916_CMD_START_MASK_RECEIVE_TIMER   0xE2U
```

Start the mask-receive timer

◆ ST25R3916_CMD_START_NO_RESPONSE_TIMER

```
#define ST25R3916_CMD_START_NO_RESPONSE_TIMER   0xE3U
```

Start the no-response timer

◆ ST25R3916_CMD_START_PPON2_TIMER

```
#define ST25R3916_CMD_START_PPON2_TIMER   0xE4U
```

Start PPon2 timer

◆ ST25R3916_CMD_START_WUP_TIMER

```
#define ST25R3916_CMD_START_WUP_TIMER   0xE1U
```

Start the wake-up timer

◆ ST25R3916_CMD_STOP

```
#define ST25R3916_CMD_STOP   0xC2U
```

Stops all activities and clears FIFO

◆ ST25R3916_CMD_STOP_NRT

```
#define ST25R3916_CMD_STOP_NRT   0xE8U
```

Stop No Response Timer

◆ ST25R3916_CMD_TEST_ACCESS

```
#define ST25R3916_CMD_TEST_ACCESS   0xFCU
```

Enable R/W access to the test registers

◆ ST25R3916_CMD_TRANSMIT_REQA

```
#define ST25R3916_CMD_TRANSMIT_REQA   0xC6U
```

Transmit REQA

◆ ST25R3916_CMD_TRANSMIT_WITH_CRC

```
#define ST25R3916_CMD_TRANSMIT_WITH_CRC   0xC4U
```

Transmit with CRC

◆ ST25R3916_CMD_TRANSMIT_WITHOUT_CRC

```
#define ST25R3916_CMD_TRANSMIT_WITHOUT_CRC   0xC5U
```

Transmit without CRC

◆ ST25R3916_CMD_TRANSMIT_WUPA

```
#define ST25R3916_CMD_TRANSMIT_WUPA   0xC7U
```

Transmit WUPA

◆ ST25R3916_CMD_TRANSPARENT_MODE

```
#define ST25R3916_CMD_TRANSPARENT_MODE   0xDCU
```

Transparent mode

◆ ST25R3916_CMD_UNMASK_RECEIVE_DATA

```
#define ST25R3916_CMD_UNMASK_RECEIVE_DATA   0xD1U
```

Unmask receive data

◆ ST25R3916_FIFO_DEPTH

```
#define ST25R3916_FIFO_DEPTH   512U
```

Depth of FIFO

◆ ST25R3916_THRESHOLD_DO_NOT_SET

```
#define ST25R3916_THRESHOLD_DO_NOT_SET   0xFFU
```

Indicates not to change this Threshold

◆ ST25R3916_TOUT_OSC_STABLE

```
#define ST25R3916_TOUT_OSC_STABLE   10U
```

Max timeout for Oscillator to get stable DS: 700us

◆ st25r3916IsAATOn

```
#define st25r3916IsAATOn ( ) st25r3916CheckReg( ST25R3916_REG_IO_CONF2, ST25R3916_REG_IO_CONF2_aat_en, ST25R3916_REG_IO_CONF2_aat_en )
```

Checks if the AAT is enabled

◆ st25r3916IsExtFieldOn

```
#define st25r3916IsExtFieldOn ( ) st25r3916CheckReg( ST25R3916_REG_AUX_DISPLAY, ST25R3916_REG_AUX_DISPLAY_efd_o, ST25R3916_REG_AUX_DISPLAY_efd_o )
```

Checks if External Filed is detected by reading ST25R3916 External Field Detector output

◆ st25r3916IsGPTRunning

```
#define st25r3916IsGPTRunning ( ) st25r3916CheckReg( ST25R3916_REG_NFCIP1_BIT_RATE, ST25R3916_REG_NFCIP1_BIT_RATE_gpt_on, ST25R3916_REG_NFCIP1_BIT_RATE_gpt_on )
```

Checks if General Purpose Timer is still running by reading gpt_on flag

◆ st25r3916IsLastFIFOComplete

```
#define st25r3916IsLastFIFOComplete ( ) st25r3916CheckReg( ST25R3916_REG_FIFO_STATUS2, ST25R3916_REG_FIFO_STATUS2_fifo_lb_mask, 0 )
```

Checks if last FIFO byte is complete

◆ st25r3916IsNRTinEMV

```
#define st25r3916IsNRTinEMV ( ) st25r3916CheckReg( ST25R3916_REG_TIMER_EMV_CONTROL, ST25R3916_REG_TIMER_EMV_CONTROL_nrt_emv, ST25R3916_REG_TIMER_EMV_CONTROL_nrt_emv_on )
```

Checks if NRT is in EMV mode

◆ st25r3916IsOscOn

```
#define st25r3916IsOscOn ( ) st25r3916CheckReg( ST25R3916_REG_OP_CONTROL, ST25R3916_REG_OP_CONTROL_en, ST25R3916_REG_OP_CONTROL_en )
```

Checks if the Oscillator is enabled

◆ st25r3916IsTxEnabled

```
#define st25r3916IsTxEnabled ( ) st25r3916CheckReg( ST25R3916_REG_OP_CONTROL, ST25R3916_REG_OP_CONTROL_tx_en, ST25R3916_REG_OP_CONTROL_tx_en )
```

Checks if Transmitter is enabled (Field On)

◆ st25r3916TxOff

```
#define st25r3916TxOff ( ) st25r3916ClrRegisterBits( ST25R3916_REG_OP_CONTROL, ST25R3916_REG_OP_CONTROL_tx_en )
```

Disables the Transmitter (Field Off)

◆ st25r3916TxRxOff

```
#define st25r3916TxRxOff ( ) st25r3916ClrRegisterBits( ST25R3916_REG_OP_CONTROL, (ST25R3916_REG_OP_CONTROL_rx_en | ST25R3916_REG_OP_CONTROL_tx_en ) )
```

Disables the Transmitter (Field Off) and Receiver

◆ st25r3916TxRxOn

```
#define st25r3916TxRxOn ( ) st25r3916SetRegisterBits( ST25R3916_REG_OP_CONTROL, (ST25R3916_REG_OP_CONTROL_rx_en | ST25R3916_REG_OP_CONTROL_tx_en ) )
```

Enables the Transmitter (Field On) and Receiver

Function Documentation

◆ st25r3916AdjustRegulators()

```
ReturnCode st25r3916AdjustRegulators ( uint16_t * result_mV )
```

This function the power level is measured in maximum load conditions and the regulated voltage reference is set to 250mV below this level. Execution of this function lasts arround 5ms.

The regulated voltages will be set to the result of Adjust Regulators

Parameters

[out] **result_mV** : Result of calibration in milliVolts

Returns

RFAL_ERR_IO : Error during communication with ST25R3916

RFAL_ERR_NONE : No error

◆ st25r3916CalibrateCapacitiveSensor()

ReturnCode st25r3916CalibrateCapacitiveSensor (uint8_t * result)

This function performs automatic calibration of the capacitive sensor and stores the result in parameter *result*.

Warning

To avoid interference with Xtal oscillator and reader magnetic field, it is strongly recommended to perform calibration in Power-down mode only. This method does not modify the Oscillator nor transmitter state, these should be configured before by user.

Parameters

[out] **result** 5 bit long result of the calibration. Binary weighted, step 0.1 pF, max 3.1 pF

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_IO : The calibration was not successful

RFAL_ERR_NONE : No error

◆ st25r3916CheckChipID()

bool st25r3916CheckChipID (uint8_t * rev)

Checks if the chip ID is as expected.

5 bit IC type code for ST25R3916: 00101 The 3 lsb contain the IC revision code

Parameters

[out] **rev** : the IC revision code

Returns

true when IC type is as expected

false otherwise

◆ st25r3916Deinitialize()

void st25r3916Deinitialize (void)

Calling this function deinitializes the ST25R3916 driver.

◆ st25r3916ExecuteCommandAndGetResult()

```
ReturnCode st25r3916ExecuteCommandAndGetResult ( uint8_t    cmd,
                                                 uint8_t    resReg,
                                                 uint8_t    tOut,
                                                 uint8_t *  result
                                               )
```

This function executes the direct command given by *cmd* waits for *sleeptime* for I_dct and returns the result read from register *resreg*. The value of cmd is not checked.

Parameters

- [in] **cmd** : direct command to execute
- [in] **resReg** address of the register containing the result
- [in] **tOut** : time in milliseconds to wait before reading the result
- [out] **result** result

Returns

RFAL_ERR_NONE : No error

◆ st25r3916GetNoResponseTime()

```
uint32_t st25r3916GetNoResponseTime ( void )
```

This returns the last value set on the NRT

Warning

it does not read chip register, just the sw var that contains the last value set before

Returns

the value of the NRT in 64/fc

◆ st25r3916GetNumFIFOBytes()

```
uint16_t st25r3916GetNumFIFOBytes ( void )
```

Gets the number of bytes currently in the FIFO

Returns

the number of bytes currently in the FIFO

◆ st25r3916GetNumFIFOLastBits()

uint8_t st25r3916GetNumFIFOLastBits (void)

Gets the number of bits of the last FIFO byte if incomplete

Returns

the number of bits of the last FIFO byte if incomplete, 0 if the last byte is complete

◆ st25r3916GetRegsDump()

ReturnCode st25r3916GetRegsDump (**t_st25r3916Regs** * regDump)

Parameters

[out] **regDump** : pointer to the struct/buffer where the reg dump will be written

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error

◆ st25r3916GetRSSI()

ReturnCode st25r3916GetRSSI (uint16_t * amRssi,
uint16_t * pmRssi
)

This function gets the RSSI value of the previous reception taking into account the gain reductions that were used. RSSI value for both AM and PM channel can be retrieved.

Parameters

[out] **amRssi** the RSSI on the AM channel expressed in mV

[out] **pmRssi** the RSSI on the PM channel expressed in mV

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error

◆ st25r3916Initialize()

ReturnCode st25r3916Initialize (void)

This function initialises the ST25R3916 driver.

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_HW_MISMATCH : Expected HW do not match or communication error

RFAL_ERR_IO : Error during communication selftest. Check communication interface

RFAL_ERR_TIMEOUT : Timeout during IRQ selftest. Check IRQ handling

RFAL_ERR_SYSTEM : Failure during oscillator activation or timer error

◆ st25r3916IsCmdValid()

bool st25r3916IsCmdValid (uint8_t cmd)

Checks if the given command is a valid ST25R3916 command

Parameters

`[in]` **cmd** Command to check

Returns

true if is a valid command

false otherwise

◆ st25r3916MeasureAmplitude()

ReturnCode st25r3916MeasureAmplitude (uint8_t * result)

This function measures the amplitude on the RFI inputs and stores the result in parameter *result*.

Parameters

`[out]` **result** result of RF measurement.

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error

◆ st25r3916MeasureCapacitance()

ReturnCode st25r3916MeasureCapacitance (uint8_t * result)

This function performs the capacitance measurement and stores the result in parameter *result*.

Parameters

[out] **result** 8 bit long result of RF measurement.

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error

◆ st25r3916MeasurePhase()

ReturnCode st25r3916MeasurePhase (uint8_t * result)

This function performs a Phase measurement. The result is stored in the *result* parameter.

Parameters

[out] **result** 8 bit long result of the measurement.

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error

◆ st25r3916MeasurePowerSupply()

uint8_t st25r3916MeasurePowerSupply (uint8_t mpsv)

This function executes Measure Power Supply and returns the raw value

Parameters

[in] **mpsv** : one of ST25R3916_REG_REGULATOR_CONTROL_mpsv_vdd
ST25R3916_REG_REGULATOR_CONTROL_mpsv_vdd_rf
ST25R3916_REG_REGULATOR_CONTROL_mpsv_vdd_a
ST25R3916_REG_REGULATOR_CONTROL_mpsv_vdd_d
ST25R3916_REG_REGULATOR_CONTROL_mpsv_vdd_am

Returns

the measured voltage in raw format.

◆ st25r3916MeasureVoltage()

uint16_t st25r3916MeasureVoltage (uint8_t mpsv)

This function measures the voltage on one of VDD and VDD_* and returns the result in mV

Parameters

[in] **mpsv** : one of ST25R3916_REG_REGULATOR_CONTROL_mpsv_vdd
ST25R3916_REG_REGULATOR_CONTROL_mpsv_vdd_rf
ST25R3916_REG_REGULATOR_CONTROL_mpsv_vdd_a
ST25R3916_REG_REGULATOR_CONTROL_mpsv_vdd_d or
ST25R3916_REG_REGULATOR_CONTROL_mpsv_vdd_am

Returns

the measured voltage in mV

◆ st25r3916OscOn()

ReturnCode st25r3916OscOn (void)

This function turn on oscillator and regulator and waits for the oscillator to become stable

Returns

RFAL_ERR_SYSTEM : Failure dusring Oscillator activation

RFAL_ERR_NONE : No error, Oscillator is active and stable, Regulator is on

◆ st25r3916PerformCollisionAvoidance()

ReturnCode st25r3916PerformCollisionAvoidance (uint8_t FieldONCmd,
uint8_t pdThreshold,
uint8_t caThreshold,
uint8_t nTRFW
)

Performs Collision Avoidance with the given threshold and with the
n number of TRFW

Parameters

[in] **FieldONCmd** : Field ON command to be executed ST25R3916_CMD_INITIAL_RF_COLLISION or ST25R3916_CMD_RESPONSE_RF_COLLISION_N

[in] **pdThreshold** : Peer Detection Threshold (ST25R3916_REG_FIELD_THRESHOLD_trg_xx) 0xff : don't set Threshold (ST25R3916_THRESHOLD_DO_NOT_SET)

[in] **caThreshold** : Collision Avoidance Threshold (ST25R3916_REG_FIELD_THRESHOLD_rfe_xx) 0xff : don't set Threshold (ST25R3916_THRESHOLD_DO_NOT_SET)

[in] **nTRFW** : Number of TRFW

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_RF_COLLISION : Collision detected

RFAL_ERR_NONE : No collision detected

◆ st25r3916SetAntennaMode()

```
ReturnCode st25r3916SetAntennaMode ( bool single,
                                     bool rfiox
                                   )
```

Sets the antenna mode. Differential or single ended antenna mode (RFO1 or RFO2)

Parameters

[in] **single** FALSE differential ; single ended mode

[in] **rfiox** FALSE RFI1/RFO1 ; TRUE RFI2/RFO2

Returns

RFAL_ERR_IO : Internal error

RFAL_ERR_NOTSUPP : Feature not supported

RFAL_ERR_NONE : No error

◆ st25r3916SetBitrate()

```
ReturnCode st25r3916SetBitrate ( uint8_t txrate,
                                 uint8_t rxrate
                               )
```

This function sets the bitrates for rx and tx

Parameters

txrate : speed is 2^txrate * 106 kb/s 0xff : don't set txrate (ST25R3916_BR_DO_NOT_SET)

rxrate : speed is 2^rxrate * 106 kb/s 0xff : don't set rxrate (ST25R3916_BR_DO_NOT_SET)

Returns

RFAL_ERR_PARAM: At least one bit rate was invalid

RFAL_ERR_NONE : No error, both bit rates were set

◆ st25r3916SetGPTime()

void st25r3916SetGPTime (uint16_t gpt_8fcs)

This function sets the General Purpose Timer time registers

Parameters

[in] **gpt_8fcs** : general purpose timer timeout in steps of 8/fc (590ns)

◆ st25r3916SetNoResponseTime()

ReturnCode st25r3916SetNoResponseTime (uint32_t nrt_64fcs)

This function sets the No Response Time with the given value

Parameters

[in] **nrt_64fcs** : no response time in steps of 64/fc (4.72us)

Returns

RFAL_ERR_PARAM : Invalid parameter (time is too large)

RFAL_ERR_NONE : No error

◆ st25r3916SetNumTxBits()

void st25r3916SetNumTxBits (uint16_t nBits)

Sets ST25R3916 internal registers with correct number of complete bytes and bits to be sent

Parameters

[in] **nBits** : number of bits to be set/transmitted

◆ st25r3916SetRegulators()

ReturnCode st25r3916SetRegulators (uint8_t regulation)

Manually sets the regulated voltage setting (rege). The regulated voltages will be set to the manual configuratiom.

Regulation shall be set according to the desired voltage considering the supplied voltage.

Parameters

[in] **regulation** : Regulator setting

Returns

RFAL_ERR_IO : Error during communication with ST25R3916

RFAL_ERR_PARAM : Invalid regulator setting

RFAL_ERR_NONE : No error

◆ st25r3916SetStartGPTimer()

ReturnCode st25r3916SetStartGPTimer (uint16_t gpt_8fcs,
uint8_t trigger_source
)

This function sets the General Purpose Timer with the given timeout and immediately starts it ONLY if the trigger source is not set to none.

Parameters

[in] **gpt_8fcs** : general purpose timer timeout in steps of8/fc (590ns)

[in] **trigger_source** : no trigger, start of Rx, end of Rx, end of Tx in NFC mode

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error

◆ st25r3916SetStartNoResponseTimer()

ReturnCode st25r3916SetStartNoResponseTimer (uint32_t nrt_64fcs)

This function sets the No Response Time with the given value and immediately starts it Used when needs to add more time before timeout without performing Tx

Parameters

[in] **nrt_64fcs** : no response time in steps of 64/fc (4.72us)

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error

◆ st25r3916StreamConfigure()

ReturnCode st25r3916StreamConfigure (const struct **st25r3916StreamConfig** * config)

This function initializes the stream with the given parameters

Parameters

[in] **config** : all settings for bitrates, type, etc.

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error, stream mode driver initialized

RFAL

Main Page

Modules

Files

Examples

Data Structures

Data Structure Index

Data Fields

t_st25r3916Regs Struct Reference


```
#include <st25r3916.h>
```

Data Fields

uint8_t **RsA** [(**ST25R3916_REG_IC_IDENTITY**+1U)]

uint8_t **RsB** [**ST25R3916_SPACE_B_REG_LEN**]

Detailed Description

Struct to represent all regs on ST25R3916

Field Documentation

◆ RsA

uint8_t t_st25r3916Regs::RsA[(**ST25R3916_REG_IC_IDENTITY**+1U)]

Registers contained on ST25R3916 space A (Rs-A)

◆ RsB

uint8_t t_st25r3916Regs::RsB[**ST25R3916_SPACE_B_REG_LEN**]

Registers contained on ST25R3916 space B (Rs-B)

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

- **st25r3916.h**

RFAL

Main Page

Modules

Files

Examples

Data Structures

Data Structure Index

Data Fields

Data Fields

st25r3916StreamConfig Struct Reference

RFAL » RFAL-HAL » ST25R3916 » ST25R3916_Driver

```
#include <st25r3916.h>
```

Data Fields

uint8_t	**useBPSK**
uint8_t	**din**
uint8_t	**dout**
uint8_t	**report_period_length**

Detailed Description

Parameters how the stream mode should work

Field Documentation

◆ din

uint8_t st25r3916StreamConfig::din

Divider for the in subcarrier frequency: fc/2^din

◆ dout

uint8_t st25r3916StreamConfig::dout

Divider for the in subcarrier frequency fc/2^dout

◆ report_period_length

uint8_t st25r3916StreamConfig::report_period_length

Length of the reporting period 2^report_period_length

◆ useBPSK

uint8_t st25r3916StreamConfig::useBPSK

0: subcarrier, 1:BPSK

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

- **st25r3916.h**

ST25R3916_COM

RFAL ST25R3916 Communications. More...

Collaboration diagram for ST25R3916_COM:

Macros

#define **ST25R3916_SPACE_B** 0x40U

#define **ST25R3916_SPACE_B_REG_LEN** 16U

#define **ST25R3916_FIFO_STATUS_LEN** 2

#define **ST25R3916_PTM_A_LEN** 15U

#define **ST25R3916_PTM_B_LEN** 0U

#define **ST25R3916_PTM_F_LEN** 21U

#define **ST25R3916_PTM_TSN_LEN** 12U

#define **ST25R3916_PTM_LEN** (**ST25R3916_PTM_A_LEN** + **ST25R3916_PTM_B_LEN** + **ST25R3916_PTM_F_LEN** + **ST25R3916_PTM_TSN_LEN**)

#define **ST25R3916_REG_IO_CONF1** 0x00U

#define **ST25R3916_REG_IO_CONF2** 0x01U

```
#define ST25R3916_REG_OP_CONTROL 0x02U
#define ST25R3916_REG_MODE 0x03U
#define ST25R3916_REG_BIT_RATE 0x04U
#define ST25R3916_REG_ISO14443A_NFC 0x05U
#define ST25R3916_REG_EMD_SUP_CONF (ST25R3916_SPACE_B|0x05U)
#define ST25R3916_REG_ISO14443B_1 0x06U
#define ST25R3916_REG_SUBC_START_TIME (ST25R3916_SPACE_B|0x06U)
#define ST25R3916_REG_ISO14443B_2 0x07U
#define ST25R3916_REG_PASSIVE_TARGET 0x08U
#define ST25R3916_REG_STREAM_MODE 0x09U
#define ST25R3916_REG_AUX 0x0AU
#define ST25R3916_REG_RX_CONF1 0x0BU
#define ST25R3916_REG_RX_CONF2 0x0CU
#define ST25R3916_REG_RX_CONF3 0x0DU
#define ST25R3916_REG_RX_CONF4 0x0EU
#define ST25R3916_REG_P2P_RX_CONF (ST25R3916_SPACE_B|0x0BU)
#define ST25R3916_REG_CORR_CONF1 (ST25R3916_SPACE_B|0x0CU)
#define ST25R3916_REG_CORR_CONF2 (ST25R3916_SPACE_B|0x0DU)
#define ST25R3916_REG_MASK_RX_TIMER 0x0FU
#define ST25R3916_REG_NO_RESPONSE_TIMER1 0x10U
#define ST25R3916_REG_NO_RESPONSE_TIMER2 0x11U
```

```
#define ST25R3916_REG_TIMER_EMV_CONTROL 0x12U
#define ST25R3916_REG_GPT1 0x13U
#define ST25R3916_REG_GPT2 0x14U
#define ST25R3916_REG_PPON2 0x15U
#define ST25R3916_REG_SQUELCH_TIMER (ST25R3916_SPACE_B|0x0FU)
#define ST25R3916_REG_FIELD_ON_GT (ST25R3916_SPACE_B|0x15U)
#define ST25R3916_REG_IRQ_MASK_MAIN 0x16U
#define ST25R3916_REG_IRQ_MASK_TIMER_NFC 0x17U
#define ST25R3916_REG_IRQ_MASK_ERROR_WUP 0x18U
#define ST25R3916_REG_IRQ_MASK_TARGET 0x19U
#define ST25R3916_REG_IRQ_MAIN 0x1AU
#define ST25R3916_REG_IRQ_TIMER_NFC 0x1BU
#define ST25R3916_REG_IRQ_ERROR_WUP 0x1CU
#define ST25R3916_REG_IRQ_TARGET 0x1DU
#define ST25R3916_REG_FIFO_STATUS1 0x1EU
#define ST25R3916_REG_FIFO_STATUS2 0x1FU
#define ST25R3916_REG_COLLISION_STATUS 0x20U
#define ST25R3916_REG_PASSIVE_TARGET_STATUS 0x21U
#define ST25R3916_REG_NUM_TX_BYTES1 0x22U
#define ST25R3916_REG_NUM_TX_BYTES2 0x23U
#define ST25R3916_REG_NFCIP1_BIT_RATE 0x24U
```

```
#define ST25R3916_REG_AD_RESULT 0x25U
#define ST25R3916_REG_ANT_TUNE_A 0x26U
#define ST25R3916_REG_ANT_TUNE_B 0x27U
#define ST25R3916_REG_TX_DRIVER 0x28U
#define ST25R3916_REG_PT_MOD 0x29U
#define ST25R3916_REG_AUX_MOD (ST25R3916_SPACE_B|0x28U)
#define ST25R3916_REG_TX_DRIVER_TIMING (ST25R3916_SPACE_B|0x29U)
#define ST25R3916_REG_RES_AM_MOD (ST25R3916_SPACE_B|0x2AU)
#define ST25R3916_REG_TX_DRIVER_STATUS (ST25R3916_SPACE_B|0x2BU)
#define ST25R3916_REG_FIELD_THRESHOLD_ACTV 0x2AU
#define ST25R3916_REG_FIELD_THRESHOLD_DEACTV 0x2BU
#define ST25R3916_REG_REGULATOR_CONTROL 0x2CU
#define ST25R3916_REG_REGULATOR_RESULT (ST25R3916_SPACE_B|0x2CU)
#define ST25R3916_REG_RSSI_RESULT 0x2DU
#define ST25R3916_REG_GAIN_RED_STATE 0x2EU
#define ST25R3916_REG_CAP_SENSOR_CONTROL 0x2FU
#define ST25R3916_REG_CAP_SENSOR_RESULT 0x30U
#define ST25R3916_REG_AUX_DISPLAY 0x31U
#define ST25R3916_REG_OVERSHOOT_CONF1 (ST25R3916_SPACE_B|0x30U)
#define ST25R3916_REG_OVERSHOOT_CONF2 (ST25R3916_SPACE_B|0x31U)
#define ST25R3916_REG_UNDERSHOOT_CONF1 (ST25R3916_SPACE_B|0x32U)
```

#define **ST25R3916_REG_UNDERSHOOT_CONF2** (**ST25R3916_SPACE_B**|0x33U)

#define **ST25R3916_REG_WUP_TIMER_CONTROL** 0x32U

#define **ST25R3916_REG_AMPLITUDE_MEASURE_CONF** 0x33U

#define **ST25R3916_REG_AMPLITUDE_MEASURE_REF** 0x34U

#define **ST25R3916_REG_AMPLITUDE_MEASURE_AA_RESULT** 0x35U

#define **ST25R3916_REG_AMPLITUDE_MEASURE_RESULT** 0x36U

#define **ST25R3916_REG_PHASE_MEASURE_CONF** 0x37U

#define **ST25R3916_REG_PHASE_MEASURE_REF** 0x38U

#define **ST25R3916_REG_PHASE_MEASURE_AA_RESULT** 0x39U

#define **ST25R3916_REG_PHASE_MEASURE_RESULT** 0x3AU

#define **ST25R3916_REG_IC_IDENTITY** 0x3FU

Functions

ReturnCode **st25r3916ReadRegister** (uint8_t reg, uint8_t *val)

Returns the content of a register within the ST25R3916. More...

ReturnCode **st25r3916ReadMultipleRegisters** (uint8_t reg, uint8_t *values, uint8_t length)

Reads from multiple ST25R3916 registers. More...

ReturnCode **st25r3916WriteRegister** (uint8_t reg, uint8_t val)

Writes a given value to a register within the ST25R3916. More...

ReturnCode **st25r3916WriteMultipleRegisters** (uint8_t reg, const uint8_t *values, uint8_t length)

Writes multiple values to ST25R3916 registers. More...

ReturnCode **st25r3916WriteFifo** (const uint8_t *values, uint16_t length)

Writes values to ST25R3916 FIFO. More...

ReturnCode **st25r3916ReadFifo** (uint8_t *buf, uint16_t length)

Read values from ST25R3916 FIFO. More...

ReturnCode **st25r3916WritePTMem** (const uint8_t *values, uint16_t length)
Writes values to ST25R3916 PTM. More...

ReturnCode **st25r3916ReadPTMem** (uint8_t *values, uint16_t length)
Reads the ST25R3916 PTM. More...

ReturnCode **st25r3916WritePTMemF** (const uint8_t *values, uint16_t length)
Writes values to ST25R3916 PTM F config. More...

ReturnCode **st25r3916WritePTMemTSN** (const uint8_t *values, uint16_t length)
Writes values to ST25R3916 PTM TSN Data. More...

ReturnCode **st25r3916ExecuteCommand** (uint8_t cmd)
Execute a direct command. More...

ReturnCode **st25r3916ExecuteCommands** (const uint8_t *cmds, uint8_t length)
Execute direct commands. More...

ReturnCode **st25r3916ReadTestRegister** (uint8_t reg, uint8_t *val)
Read a test register within the ST25R3916. More...

ReturnCode **st25r3916WriteTestRegister** (uint8_t reg, uint8_t val)
Writes a given value to a test register within the ST25R3916. More...

ReturnCode **st25r3916ClrRegisterBits** (uint8_t reg, uint8_t clr_mask)
Cleart bits on Register. More...

ReturnCode **st25r3916SetRegisterBits** (uint8_t reg, uint8_t set_mask)
Set bits on Register. More...

ReturnCode **st25r3916ChangeRegisterBits** (uint8_t reg, uint8_t valueMask, uint8_t value)
Changes the given bits on a ST25R3916 register. More...

ReturnCode **st25r3916ModifyRegister** (uint8_t reg, uint8_t clr_mask, uint8_t set_mask)
Modifies a value within a ST25R3916 register. More...

ReturnCode **st25r3916ChangeTestRegisterBits** (uint8_t reg, uint8_t valueMask, uint8_t value)

Changes the given bits on a ST25R3916 Test register. More...

bool **st25r3916CheckReg** (uint8_t reg, uint8_t mask, uint8_t val)
Checks if register contains a expected value. More...

bool **st25r3916IsRegValid** (uint8_t reg)
Check if register ID is valid. More...

Detailed Description

Macro Definition Documentation

◆ ST25R3916_FIFO_STATUS_LEN

```
#define ST25R3916_FIFO_STATUS_LEN  2
```

Number of FIFO Status Register

◆ ST25R3916_PTM_A_LEN

```
#define ST25R3916_PTM_A_LEN  15U
```

Passive target memory A config length

◆ ST25R3916_PTM_B_LEN

```
#define ST25R3916_PTM_B_LEN  0U
```

Passive target memory B config length

◆ ST25R3916_PTM_F_LEN

```
#define ST25R3916_PTM_F_LEN  21U
```

Passive target memory F config length

◆ ST25R3916_PTM_LEN

#define ST25R3916_PTM_LEN (**ST25R3916_PTM_A_LEN** + **ST25R3916_PTM_B_LEN** + **ST25R3916_PTM_F_LEN** + **ST25R3916_PTM_TSN_LEN**)

Full Passive target memory length

◆ ST25R3916_PTM_TSN_LEN

#define ST25R3916_PTM_TSN_LEN 12U

Passive target memory TSN data length

◆ ST25R3916_REG_AD_RESULT

#define ST25R3916_REG_AD_RESULT 0x25U

R A/D Converter Output Register

◆ ST25R3916_REG_AMPLITUDE_MEASURE_AA_RESULT

#define ST25R3916_REG_AMPLITUDE_MEASURE_AA_RESULT 0x35U

R Amplitude Measurement Auto Averaging Display Reg

◆ ST25R3916_REG_AMPLITUDE_MEASURE_CONF

#define ST25R3916_REG_AMPLITUDE_MEASURE_CONF 0x33U

RW Amplitude Measurement Configuration Register

◆ ST25R3916_REG_AMPLITUDE_MEASURE_REF

#define ST25R3916_REG_AMPLITUDE_MEASURE_REF 0x34U

RW Amplitude Measurement Reference Register

◆ ST25R3916_REG_AMPLITUDE_MEASURE_RESULT

#define ST25R3916_REG_AMPLITUDE_MEASURE_RESULT 0x36U

R Amplitude Measurement Display Register

◆ ST25R3916_REG_ANT_TUNE_A

#define ST25R3916_REG_ANT_TUNE_A 0x26U

RW Antenna Tuning Control (AAT-A) Register 1

◆ ST25R3916_REG_ANT_TUNE_B

#define ST25R3916_REG_ANT_TUNE_B 0x27U

RW Antenna Tuning Control (AAT-B) Register 2

◆ ST25R3916_REG_AUX

#define ST25R3916_REG_AUX 0x0AU

RW Auxiliary Definition Register

◆ ST25R3916_REG_AUX_DISPLAY

#define ST25R3916_REG_AUX_DISPLAY 0x31U

R Auxiliary Display Register

◆ ST25R3916_REG_AUX_MOD

#define ST25R3916_REG_AUX_MOD (**ST25R3916_SPACE_B**|0x28U)

RW Aux Modulation setting Register

◆ ST25R3916_REG_BIT_RATE

```
#define ST25R3916_REG_BIT_RATE   0x04U
```

RW Bit Rate Definition Register

◆ ST25R3916_REG_CAP_SENSOR_CONTROL

```
#define ST25R3916_REG_CAP_SENSOR_CONTROL   0x2FU
```

RW Capacitive Sensor Control Register

◆ ST25R3916_REG_CAP_SENSOR_RESULT

```
#define ST25R3916_REG_CAP_SENSOR_RESULT   0x30U
```

R Capacitive Sensor Display Register

◆ ST25R3916_REG_COLLISION_STATUS

```
#define ST25R3916_REG_COLLISION_STATUS   0x20U
```

R Collision Display Register

◆ ST25R3916_REG_CORR_CONF1

```
#define ST25R3916_REG_CORR_CONF1   (ST25R3916_SPACE_B|0x0CU)
```

RW Correlator configuration register 1

◆ ST25R3916_REG_CORR_CONF2

```
#define ST25R3916_REG_CORR_CONF2   (ST25R3916_SPACE_B|0x0DU)
```

RW Correlator configuration register 2

◆ ST25R3916_REG_EMD_SUP_CONF

#define ST25R3916_REG_EMD_SUP_CONF (**ST25R3916_SPACE_B**|0x05U)

RW EMD Suppression Configuration Register

◆ ST25R3916_REG_FIELD_ON_GT

#define ST25R3916_REG_FIELD_ON_GT (**ST25R3916_SPACE_B**|0x15U)

RW NFC Field on guard time

◆ ST25R3916_REG_FIELD_THRESHOLD_ACTV

#define ST25R3916_REG_FIELD_THRESHOLD_ACTV 0x2AU

RW External Field Detector Activation Threshold Reg

◆ ST25R3916_REG_FIELD_THRESHOLD_DEACTV

#define ST25R3916_REG_FIELD_THRESHOLD_DEACTV 0x2BU

RW External Field Detector Deactivation Threshold Reg

◆ ST25R3916_REG_FIFO_STATUS1

#define ST25R3916_REG_FIFO_STATUS1 0x1EU

R FIFO Status Register 1

◆ ST25R3916_REG_FIFO_STATUS2

#define ST25R3916_REG_FIFO_STATUS2 0x1FU

R FIFO Status Register 2

◆ ST25R3916_REG_GAIN_RED_STATE

```
#define ST25R3916_REG_GAIN_RED_STATE   0x2EU
```

R Gain Reduction State Register

◆ ST25R3916_REG_GPT1

```
#define ST25R3916_REG_GPT1   0x13U
```

RW General Purpose Timer Register 1

◆ ST25R3916_REG_GPT2

```
#define ST25R3916_REG_GPT2   0x14U
```

RW General Purpose Timer Register 2

◆ ST25R3916_REG_IC_IDENTITY

```
#define ST25R3916_REG_IC_IDENTITY   0x3FU
```

R Chip Id: 0 for old silicon, v2 silicon: 0x09

◆ ST25R3916_REG_IO_CONF1

```
#define ST25R3916_REG_IO_CONF1   0x00U
```

RW IO Configuration Register 1

◆ ST25R3916_REG_IO_CONF2

```
#define ST25R3916_REG_IO_CONF2   0x01U
```

RW IO Configuration Register 2

◆ ST25R3916_REG_IRQ_ERROR_WUP

```
#define ST25R3916_REG_IRQ_ERROR_WUP   0x1CU
```

R Error and Wake-up Interrupt Register

◆ ST25R3916_REG_IRQ_MAIN

```
#define ST25R3916_REG_IRQ_MAIN   0x1AU
```

R Main Interrupt Register

◆ ST25R3916_REG_IRQ_MASK_ERROR_WUP

```
#define ST25R3916_REG_IRQ_MASK_ERROR_WUP   0x18U
```

RW Mask Error and Wake-up Interrupt Register

◆ ST25R3916_REG_IRQ_MASK_MAIN

```
#define ST25R3916_REG_IRQ_MASK_MAIN   0x16U
```

RW Mask Main Interrupt Register

◆ ST25R3916_REG_IRQ_MASK_TARGET

```
#define ST25R3916_REG_IRQ_MASK_TARGET   0x19U
```

RW Mask 3916 Target Interrupt Register

◆ ST25R3916_REG_IRQ_MASK_TIMER_NFC

```
#define ST25R3916_REG_IRQ_MASK_TIMER_NFC   0x17U
```

RW Mask Timer and NFC Interrupt Register

◆ ST25R3916_REG_IRQ_TARGET

```
#define ST25R3916_REG_IRQ_TARGET   0x1DU
```

R ST25R3916 Target Interrupt Register

◆ ST25R3916_REG_IRQ_TIMER_NFC

```
#define ST25R3916_REG_IRQ_TIMER_NFC   0x1BU
```

R Timer and NFC Interrupt Register

◆ ST25R3916_REG_ISO14443A_NFC

```
#define ST25R3916_REG_ISO14443A_NFC   0x05U
```

RW ISO14443A and NFC 106 kBit/s Settings Register

◆ ST25R3916_REG_ISO14443B_1

```
#define ST25R3916_REG_ISO14443B_1   0x06U
```

RW ISO14443B Settings Register 1

◆ ST25R3916_REG_ISO14443B_2

```
#define ST25R3916_REG_ISO14443B_2   0x07U
```

RW ISO14443B Settings Register 2

◆ ST25R3916_REG_MASK_RX_TIMER

```
#define ST25R3916_REG_MASK_RX_TIMER   0x0FU
```

RW Mask Receive Timer Register

◆ ST25R3916_REG_MODE

```
#define ST25R3916_REG_MODE   0x03U
```

RW Mode Definition Register

◆ ST25R3916_REG_NFCIP1_BIT_RATE

```
#define ST25R3916_REG_NFCIP1_BIT_RATE   0x24U
```

R NFCIP Bit Rate Detection Display Register

◆ ST25R3916_REG_NO_RESPONSE_TIMER1

```
#define ST25R3916_REG_NO_RESPONSE_TIMER1   0x10U
```

RW No-response Timer Register 1

◆ ST25R3916_REG_NO_RESPONSE_TIMER2

```
#define ST25R3916_REG_NO_RESPONSE_TIMER2   0x11U
```

RW No-response Timer Register 2

◆ ST25R3916_REG_NUM_TX_BYTES1

```
#define ST25R3916_REG_NUM_TX_BYTES1   0x22U
```

RW Number of Transmitted Bytes Register 1

◆ ST25R3916_REG_NUM_TX_BYTES2

```
#define ST25R3916_REG_NUM_TX_BYTES2   0x23U
```

RW Number of Transmitted Bytes Register 2

◆ ST25R3916_REG_OP_CONTROL

#define ST25R3916_REG_OP_CONTROL 0x02U

RW Operation Control Register

◆ ST25R3916_REG_OVERSHOOT_CONF1

#define ST25R3916_REG_OVERSHOOT_CONF1 (**ST25R3916_SPACE_B**|0x30U)

RW Overshoot Protection Configuration Register 1

◆ ST25R3916_REG_OVERSHOOT_CONF2

#define ST25R3916_REG_OVERSHOOT_CONF2 (**ST25R3916_SPACE_B**|0x31U)

RW Overshoot Protection Configuration Register 2

◆ ST25R3916_REG_P2P_RX_CONF

#define ST25R3916_REG_P2P_RX_CONF (**ST25R3916_SPACE_B**|0x0BU)

RW P2P Receiver Configuration Register 1

◆ ST25R3916_REG_PASSIVE_TARGET

#define ST25R3916_REG_PASSIVE_TARGET 0x08U

RW Passive Target Definition Register

◆ ST25R3916_REG_PASSIVE_TARGET_STATUS

#define ST25R3916_REG_PASSIVE_TARGET_STATUS 0x21U

R Passive target state status

◆ ST25R3916_REG_PHASE_MEASURE_AA_RESULT

```
#define ST25R3916_REG_PHASE_MEASURE_AA_RESULT  0x39U
```

R Phase Measurement Auto Averaging Display Register

◆ ST25R3916_REG_PHASE_MEASURE_CONF

```
#define ST25R3916_REG_PHASE_MEASURE_CONF  0x37U
```

RW Phase Measurement Configuration Register

◆ ST25R3916_REG_PHASE_MEASURE_REF

```
#define ST25R3916_REG_PHASE_MEASURE_REF  0x38U
```

RW Phase Measurement Reference Register

◆ ST25R3916_REG_PHASE_MEASURE_RESULT

```
#define ST25R3916_REG_PHASE_MEASURE_RESULT  0x3AU
```

R Phase Measurement Display Register

◆ ST25R3916_REG_PPON2

```
#define ST25R3916_REG_PPON2  0x15U
```

RW PPON2 Field waiting Timer Register

◆ ST25R3916_REG_PT_MOD

```
#define ST25R3916_REG_PT_MOD  0x29U
```

RW PT modulation Register

◆ ST25R3916_REG_REGULATOR_CONTROL

#define ST25R3916_REG_REGULATOR_CONTROL 0x2CU

RW Regulated Voltage Control Register

◆ ST25R3916_REG_REGULATOR_RESULT

#define ST25R3916_REG_REGULATOR_RESULT (**ST25R3916_SPACE_B**|0x2CU)

R Regulator Display Register

◆ ST25R3916_REG_RES_AM_MOD

#define ST25R3916_REG_RES_AM_MOD (**ST25R3916_SPACE_B**|0x2AU)

RW Resistive AM modulation register

◆ ST25R3916_REG_RSSI_RESULT

#define ST25R3916_REG_RSSI_RESULT 0x2DU

R RSSI Display Register

◆ ST25R3916_REG_RX_CONF1

#define ST25R3916_REG_RX_CONF1 0x0BU

RW Receiver Configuration Register 1

◆ ST25R3916_REG_RX_CONF2

#define ST25R3916_REG_RX_CONF2 0x0CU

RW Receiver Configuration Register 2

◆ ST25R3916_REG_RX_CONF3

#define ST25R3916_REG_RX_CONF3 0x0DU

RW Receiver Configuration Register 3

◆ ST25R3916_REG_RX_CONF4

#define ST25R3916_REG_RX_CONF4 0x0EU

RW Receiver Configuration Register 4

◆ ST25R3916_REG_SQUELCH_TIMER

#define ST25R3916_REG_SQUELCH_TIMER (**ST25R3916_SPACE_B**|0x0FU)

RW Squelch timeout Register

◆ ST25R3916_REG_STREAM_MODE

#define ST25R3916_REG_STREAM_MODE 0x09U

RW Stream Mode Definition Register

◆ ST25R3916_REG_SUBC_START_TIME

#define ST25R3916_REG_SUBC_START_TIME (**ST25R3916_SPACE_B**|0x06U)

RW Subcarrier Start Time Register

◆ ST25R3916_REG_TIMER_EMV_CONTROL

#define ST25R3916_REG_TIMER_EMV_CONTROL 0x12U

RW Timer and EMV Control

◆ ST25R3916_REG_TX_DRIVER

#define ST25R3916_REG_TX_DRIVER 0x28U

RW TX driver register

◆ ST25R3916_REG_TX_DRIVER_STATUS

#define ST25R3916_REG_TX_DRIVER_STATUS (**ST25R3916_SPACE_B**|0x2BU)

R TX driver timing readout Register

◆ ST25R3916_REG_TX_DRIVER_TIMING

#define ST25R3916_REG_TX_DRIVER_TIMING (**ST25R3916_SPACE_B**|0x29U)

RW TX driver timing Register

◆ ST25R3916_REG_UNDERSHOOT_CONF1

#define ST25R3916_REG_UNDERSHOOT_CONF1 (**ST25R3916_SPACE_B**|0x32U)

RW Undershoot Protection Configuration Register 1

◆ ST25R3916_REG_UNDERSHOOT_CONF2

#define ST25R3916_REG_UNDERSHOOT_CONF2 (**ST25R3916_SPACE_B**|0x33U)

RW Undershoot Protection Configuration Register 2

◆ ST25R3916_REG_WUP_TIMER_CONTROL

#define ST25R3916_REG_WUP_TIMER_CONTROL 0x32U

RW Wake-up Timer Control Register

◆ ST25R3916_SPACE_B

```
#define ST25R3916_SPACE_B   0x40U
```

ST25R3916 Space-B indicator

◆ ST25R3916_SPACE_B_REG_LEN

```
#define ST25R3916_SPACE_B_REG_LEN   16U
```

Number of register in the space B

Function Documentation

◆ st25r3916ChangeRegisterBits()

```
ReturnCode st25r3916ChangeRegisterBits ( uint8_t  reg,
                                         uint8_t  valueMask,
                                         uint8_t  value
                                       )
```

This function is used if only a particular bits should be changed within an ST25R3916 register.

Parameters

[in]	**reg**	Address of the register to change.
[in]	**valueMask**	bitmask of bits to be changed
[in]	**value**	the bits to be written on the enabled valueMask bits

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r3916ChangeTestRegisterBits()

```
ReturnCode st25r3916ChangeTestRegisterBits ( uint8_t  reg,
```

```
                                                uint8_t  valueMask,
                                                uint8_t  value
                                              )
```

This function is used if only a particular bits should be changed within an ST25R3916 register.

Parameters

- [in] **reg** Address of the Test register to change.
- [in] **valueMask** bitmask of bits to be changed
- [in] **value** the bits to be written on the enabled valueMask bits

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r3916CheckReg()

```
bool st25r3916CheckReg ( uint8_t  reg,
                         uint8_t  mask,
                         uint8_t  val
                       )
```

This function checks if the given reg contains a value that once masked equals the expected value

Parameters

- **reg** : the register to check the value
- **mask** : the mask apply on register value
- **val** : expected value to be compared to

Returns

true when reg contains the expected value | false otherwise

◆ st25r3916ClrRegisterBits()

```
ReturnCode st25r3916ClrRegisterBits ( uint8_t  reg,
                                      uint8_t  clr_mask
                                    )
```

This function clears the given bitmask on the register

Parameters

[in] **reg** Address of the register clear

[in] **clr_mask** Bitmask of bit to be cleared

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r3916ExecuteCommand()

```
ReturnCode st25r3916ExecuteCommand ( uint8_t  cmd )
```

This function is used to start so-called direct command. These commands are implemented inside the chip and each command has unique code (see datasheet).

Parameters

[in] **cmd** : code of the direct command to be executed.

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r3916ExecuteCommands()

```
ReturnCode st25r3916ExecuteCommands ( const uint8_t *  cmds,
                                      uint8_t          length
                                    )
```

This function is used to start so-called direct command. These commands are implemented inside the chip and each command has unique code (see datasheet).

Warning

Commands to be executed must support command chaining

Parameters

[in] **cmds** : code of the direct commands to be executed.

[in] **length** : number of direct command to be executed.

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r3916IsRegValid()

```
bool st25r3916IsRegValid ( uint8_t reg )
```

Checks if the given register ID a valid ST25R3916 register

Parameters

[in] **reg** Address of register to check

Returns

true if is a valid register ID

false otherwise

◆ st25r3916ModifyRegister()

```
ReturnCode st25r3916ModifyRegister ( uint8_t reg,
                                     uint8_t clr_mask,
                                     uint8_t set_mask
                                   )
```

This function is used if only a particular bits should be changed within an ST25R3916 register.

Parameters

[in] **reg** Address of the register to write.

[in] **clr_mask** bitmask of bits to be cleared to 0.

[in] **set_mask** bitmask of bits to be set to 1.

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r3916ReadFifo()

```
ReturnCode st25r3916ReadFifo ( uint8_t *  buf,
                               uint16_t   length
                             )
```

This function needs to be called in order to read from ST25R3916 FIFO.

Parameters

[out] **buf** pointer to a buffer where the FIFO content shall be written to.

[in] **length** Number of bytes to read.

Note

: This function doesn't check whether *length* is really the number of available bytes in FIFO

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r3916ReadMultipleRegisters()

```
ReturnCode st25r3916ReadMultipleRegisters ( uint8_t    reg,
                                            uint8_t *  values,
                                            uint8_t    length
                                          )
```

This function is used to read from multiple registers using the auto-increment feature. That is, after each read the address pointer inside the ST25R3916 gets incremented automatically.

Parameters

[in] **reg** Address of the frist register to read from.

[in] **values** pointer to a buffer where the result shall be written to.

[in] **length** Number of registers to be read out.

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r3916ReadPTMem()

```
ReturnCode st25r3916ReadPTMem ( uint8_t *  values,
                                uint16_t   length
                              )
```

Accesses to the begging of ST25R3916 Passive Target Memory (PTM A Config) and reads the memory for the given length

Parameters

[out] **values** pointer to a buffer where the PTM content shall be written to.

[in] **length** Number of bytes to read.

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r3916ReadRegister()

```
ReturnCode st25r3916ReadRegister ( uint8_t    reg,
                                   uint8_t *  val
                                 )
```

Register bit definitions
DOXYGEN_SUPPRESS

This function is used to read out the content of ST25R3916 registers.

Parameters

[in] **reg** Address of register to read.

[out] **val** Returned value.

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r3916ReadTestRegister()

```
ReturnCode st25r3916ReadTestRegister ( uint8_t    reg,
                                       uint8_t *  val
                                     )
```

This function is used to read the content of test address *reg* within the ST25R3916

Parameters

[in] **reg** Address of the register to read

[out] **val** Returned read value

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r3916SetRegisterBits()

```
ReturnCode st25r3916SetRegisterBits ( uint8_t  reg,
                                      uint8_t  set_mask
                                    )
```

This function sets the given bitmask on the register

Parameters

[in] **reg** Address of the register clear

[in] **set_mask** Bitmask of bit to be cleared

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r3916WriteFifo()

```
ReturnCode st25r3916WriteFifo ( const uint8_t *  values,
                                uint16_t         length
```

```
)
```

This function needs to be called in order to write to the ST25R3916 FIFO.

Parameters

[in] **values** pointer to a buffer containing the values to be written to the FIFO.

[in] **length** Number of values to be written.

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r3916WriteMultipleRegisters()

```
ReturnCode st25r3916WriteMultipleRegisters ( uint8_t        reg,
                                             const uint8_t * values,
                                             uint8_t        length
                                           )
```

This function is used to write multiple values to the ST25R3916 using the auto-increment feature. That is, after each write the address pointer inside the ST25R3916 gets incremented automatically.

Parameters

[in] **reg** Address of the frist register to write.

[in] **values** pointer to a buffer containing the values to be written.

[in] **length** Number of values to be written.

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r3916WritePTMem()

```
ReturnCode st25r3916WritePTMem ( const uint8_t * values,
                                 uint16_t       length
                               )
```

Accesses to the begging of ST25R3916 Passive Target Memory (PTM A Config) and writes the given values

Parameters

[in] **values** pointer to a buffer containing the values to be written to the Passive Target Memory.

[in] **length** Number of values to be written.

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r3916WritePTMemF()

```
ReturnCode st25r3916WritePTMemF ( const uint8_t *  values,
                                  uint16_t         length
                                )
```

Accesses ST25R3916 Passive Target Memory F config and writes the given values

Parameters

[in] **values** pointer to a buffer containing the values to be written to the Passive Target Memory

[in] **length** Number of values to be written.

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r3916WritePTMemTSN()

```
ReturnCode st25r3916WritePTMemTSN ( const uint8_t *  values,
                                    uint16_t         length
                                  )
```

Accesses ST25R3916 Passive Target Memory TSN data and writes the given values

Parameters

[in] **values** pointer to a buffer containing the values to be written to the Passive Target Memory.

[in] **length** Number of values to be written.

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r3916WriteRegister()

```
ReturnCode st25r3916WriteRegister ( uint8_t reg,
                                    uint8_t val
                                  )
```

This function is used to write *val* to address *reg* within the ST25R3916.

Parameters

[in] **reg** Address of the register to write.

[in] **val** Value to be written.

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r3916WriteTestRegister()

```
ReturnCode st25r3916WriteTestRegister ( uint8_t reg,
                                        uint8_t val
                                      )
```

This function is used to write *val* to test address *reg* within the ST25R3916

Parameters

[in] **reg** Address of the register to write

[in] **val** Value to be written

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

ST25R3916_IRQ

RFAL ST25R3916 IRQ. More...

Collaboration diagram for ST25R3916_IRQ:

Macros

#define **ST25R3916_IRQ_MASK_ALL** (uint32_t)(0xFFFFFFFFUL)

#define **ST25R3916_IRQ_MASK_NONE** (uint32_t)(0x00000000UL)

#define **ST25R3916_IRQ_MASK_OSC** (uint32_t)(0x00000080U)

#define **ST25R3916_IRQ_MASK_FWL** (uint32_t)(0x00000040U)

#define **ST25R3916_IRQ_MASK_RXS** (uint32_t)(0x00000020U)

#define **ST25R3916_IRQ_MASK_RXE** (uint32_t)(0x00000010U)

#define **ST25R3916_IRQ_MASK_TXE** (uint32_t)(0x00000008U)

#define **ST25R3916_IRQ_MASK_COL** (uint32_t)(0x00000004U)

#define **ST25R3916_IRQ_MASK_RX_REST** (uint32_t)(0x00000002U)

#define **ST25R3916_IRQ_MASK_RFU** (uint32_t)(0x00000001U)

#define **ST25R3916_IRQ_MASK_DCT** (uint32_t)(0x00008000U)

#define **ST25R3916_IRQ_MASK_NRE** (uint32_t)(0x00004000U)

#define **ST25R3916_IRQ_MASK_GPE** (uint32_t)(0x00002000U)

#define **ST25R3916_IRQ_MASK_EON** (uint32_t)(0x00001000U)

#define **ST25R3916_IRQ_MASK_EOF** (uint32_t)(0x00000800U)

#define **ST25R3916_IRQ_MASK_CAC** (uint32_t)(0x00000400U)

#define **ST25R3916_IRQ_MASK_CAT** (uint32_t)(0x00000200U)

#define **ST25R3916_IRQ_MASK_NFCT** (uint32_t)(0x00000100U)

#define **ST25R3916_IRQ_MASK_CRC** (uint32_t)(0x00800000U)

#define **ST25R3916_IRQ_MASK_PAR** (uint32_t)(0x00400000U)

#define **ST25R3916_IRQ_MASK_ERR2** (uint32_t)(0x00200000U)

#define **ST25R3916_IRQ_MASK_ERR1** (uint32_t)(0x00100000U)

#define **ST25R3916_IRQ_MASK_WT** (uint32_t)(0x00080000U)

#define **ST25R3916_IRQ_MASK_WAM** (uint32_t)(0x00040000U)

#define **ST25R3916_IRQ_MASK_WPH** (uint32_t)(0x00020000U)

#define **ST25R3916_IRQ_MASK_PPON2** (uint32_t)(0x80000000U)

#define **ST25R3916_IRQ_MASK_SL_WL** (uint32_t)(0x40000000U)

#define **ST25R3916_IRQ_MASK_APON** (uint32_t)(0x20000000U)

#define **ST25R3916_IRQ_MASK_RXE_PTA** (uint32_t)(0x10000000U)

#define **ST25R3916_IRQ_MASK_WU_F** (uint32_t)(0x08000000U)

#define **ST25R3916_IRQ_MASK_RFU2** (uint32_t)(0x04000000U)

#define **ST25R3916_IRQ_MASK_WU_A_X** (uint32_t)(0x02000000U)

#define **ST25R3916_IRQ_MASK_WU_A** (uint32_t)(0x01000000U)

Functions

uint32_t **st25r3916WaitForInterruptsTimed** (uint32_t mask, uint16_t tmo)
Wait until an ST25R3916 interrupt occurs. More...

uint32_t **st25r3916GetInterrupt** (uint32_t mask)
Get status for the given interrupt. More...

void **st25r3916InitInterrupts** (void)
Init the 3916 interrupt. More...

void **st25r3916ModifyInterrupts** (uint32_t clr_mask, uint32_t set_mask)
Modifies the Interrupt. More...

void **st25r3916CheckForReceivedInterrupts** (void)
Checks received interrupts. More...

void **st25r3916Isr** (void)
ISR Service routine. More...

void **st25r3916EnableInterrupts** (uint32_t mask)
Enable a given ST25R3916 Interrupt source. More...

void **st25r3916DisableInterrupts** (uint32_t mask)
Disable one or more a given ST25R3916 Interrupt sources. More...

void **st25r3916ClearInterrupts** (void)

Clear all ST25R3916 irq flags. More...

void **st25r3916ClearAndEnableInterrupts** (uint32_t mask)
Clears and then enables the given ST25R3916 Interrupt sources. More...

void **st25r3916IRQCallbackSet** (void(*cb)(void))
Sets IRQ callback for the ST25R3916 interrupt. More...

void **st25r3916IRQCallbackRestore** (void)
Sets IRQ callback for the ST25R3916 interrupt. More...

Detailed Description

Macro Definition Documentation

◆ ST25R3916_IRQ_MASK_ALL

```
#define ST25R3916_IRQ_MASK_ALL   (uint32_t)(0xFFFFFFFFUL)
```

All ST25R3916 interrupt sources

◆ ST25R3916_IRQ_MASK_APON

```
#define ST25R3916_IRQ_MASK_APON   (uint32_t)(0x20000000U)
```

ST25R3916 Anticollision done and Field On interrupt

◆ ST25R3916_IRQ_MASK_CAC

```
#define ST25R3916_IRQ_MASK_CAC   (uint32_t)(0x00000400U)
```

ST25R3916 collision during RF collision avoidance interrupt

◆ ST25R3916_IRQ_MASK_CAT

```
#define ST25R3916_IRQ_MASK_CAT   (uint32_t)(0x00000200U)
```

ST25R3916 minimum guard time expired interrupt

◆ ST25R3916_IRQ_MASK_COL

```
#define ST25R3916_IRQ_MASK_COL   (uint32_t)(0x00000004U)
```

ST25R3916 bit collision interrupt

◆ ST25R3916_IRQ_MASK_CRC

```
#define ST25R3916_IRQ_MASK_CRC   (uint32_t)(0x00800000U)
```

ST25R3916 CRC error interrupt

◆ ST25R3916_IRQ_MASK_DCT

```
#define ST25R3916_IRQ_MASK_DCT   (uint32_t)(0x00008000U)
```

ST25R3916 termination of direct command interrupt.

◆ ST25R3916_IRQ_MASK_EOF

```
#define ST25R3916_IRQ_MASK_EOF   (uint32_t)(0x00000800U)
```

ST25R3916 external field off interrupt

◆ ST25R3916_IRQ_MASK_EON

```
#define ST25R3916_IRQ_MASK_EON   (uint32_t)(0x00001000U)
```

ST25R3916 external field on interrupt

◆ ST25R3916_IRQ_MASK_ERR1

```
#define ST25R3916_IRQ_MASK_ERR1   (uint32_t)(0x00100000U)
```

ST25R3916 hard framing error interrupt

◆ ST25R3916_IRQ_MASK_ERR2

```
#define ST25R3916_IRQ_MASK_ERR2   (uint32_t)(0x00200000U)
```

ST25R3916 soft framing error interrupt

◆ ST25R3916_IRQ_MASK_FWL

```
#define ST25R3916_IRQ_MASK_FWL   (uint32_t)(0x00000040U)
```

ST25R3916 FIFO water level interrupt

◆ ST25R3916_IRQ_MASK_GPE

```
#define ST25R3916_IRQ_MASK_GPE   (uint32_t)(0x00002000U)
```

ST25R3916 general purpose timer expired interrupt

◆ ST25R3916_IRQ_MASK_NFCT

```
#define ST25R3916_IRQ_MASK_NFCT   (uint32_t)(0x00000100U)
```

ST25R3916 initiator bit rate recognised interrupt

◆ ST25R3916_IRQ_MASK_NONE

```
#define ST25R3916_IRQ_MASK_NONE   (uint32_t)(0x00000000UL)
```

No ST25R3916 interrupt source

◆ ST25R3916_IRQ_MASK_NRE

```
#define ST25R3916_IRQ_MASK_NRE   (uint32_t)(0x00004000U)
```

ST25R3916 no-response timer expired interrupt

◆ ST25R3916_IRQ_MASK_OSC

```
#define ST25R3916_IRQ_MASK_OSC   (uint32_t)(0x00000080U)
```

ST25R3916 oscillator stable interrupt

◆ ST25R3916_IRQ_MASK_PAR

```
#define ST25R3916_IRQ_MASK_PAR   (uint32_t)(0x00400000U)
```

ST25R3916 parity error interrupt

◆ ST25R3916_IRQ_MASK_PPON2

```
#define ST25R3916_IRQ_MASK_PPON2   (uint32_t)(0x80000000U)
```

ST25R3916 PPON2 Field on waiting Timer interrupt

◆ ST25R3916_IRQ_MASK_RFU

```
#define ST25R3916_IRQ_MASK_RFU   (uint32_t)(0x00000001U)
```

ST25R3916 RFU interrupt

◆ ST25R3916_IRQ_MASK_RFU2

```
#define ST25R3916_IRQ_MASK_RFU2   (uint32_t)(0x04000000U)
```

ST25R3916 RFU2 interrupt

◆ ST25R3916_IRQ_MASK_RX_REST

```
#define ST25R3916_IRQ_MASK_RX_REST   (uint32_t)(0x00000002U)
```

ST25R3916 automatic reception restart interrupt

◆ ST25R3916_IRQ_MASK_RXE

```
#define ST25R3916_IRQ_MASK_RXE   (uint32_t)(0x00000010U)
```

ST25R3916 end of receive interrupt

◆ ST25R3916_IRQ_MASK_RXE_PTA

```
#define ST25R3916_IRQ_MASK_RXE_PTA   (uint32_t)(0x10000000U)
```

ST25R3916 RXE with an automatic response interrupt

◆ ST25R3916_IRQ_MASK_RXS

```
#define ST25R3916_IRQ_MASK_RXS   (uint32_t)(0x00000020U)
```

ST25R3916 start of receive interrupt

◆ ST25R3916_IRQ_MASK_SL_WL

```
#define ST25R3916_IRQ_MASK_SL_WL   (uint32_t)(0x40000000U)
```

ST25R3916 Passive target slot number water level interrupt

◆ ST25R3916_IRQ_MASK_TXE

```
#define ST25R3916_IRQ_MASK_TXE   (uint32_t)(0x00000008U)
```

ST25R3916 end of transmission interrupt

◆ ST25R3916_IRQ_MASK_WAM

```
#define ST25R3916_IRQ_MASK_WAM   (uint32_t)(0x00040000U)
```

ST25R3916 wake-up due to amplitude interrupt

◆ ST25R3916_IRQ_MASK_WPH

```
#define ST25R3916_IRQ_MASK_WPH   (uint32_t)(0x00020000U)
```

ST25R3916 wake-up due to phase interrupt

◆ ST25R3916_IRQ_MASK_WT

```
#define ST25R3916_IRQ_MASK_WT   (uint32_t)(0x00080000U)
```

ST25R3916 wake-up interrupt

◆ ST25R3916_IRQ_MASK_WU_A

```
#define ST25R3916_IRQ_MASK_WU_A   (uint32_t)(0x01000000U)
```

ST25R3916 106kb/s Passive target state interrupt: Active

◆ ST25R3916_IRQ_MASK_WU_A_X

```
#define ST25R3916_IRQ_MASK_WU_A_X   (uint32_t)(0x02000000U)
```

ST25R3916 106kb/s Passive target state interrupt: Active*

◆ ST25R3916_IRQ_MASK_WU_F

```
#define ST25R3916_IRQ_MASK_WU_F   (uint32_t)(0x08000000U)
```

ST25R3916 212/424b/s Passive target interrupt: Active

Function Documentation

◆ st25r3916CheckForReceivedInterrupts()

```
void st25r3916CheckForReceivedInterrupts ( void )
```

Checks received interrupts and saves the result into global params

◆ st25r3916ClearAndEnableInterrupts()

void st25r3916ClearAndEnableInterrupts (uint32_t mask)

Parameters

[in] **mask** mask indicating the interrupts to be cleared and enabled

◆ st25r3916ClearInterrupts()

void st25r3916ClearInterrupts (void)

◆ st25r3916DisableInterrupts()

void st25r3916DisableInterrupts (uint32_t mask)

This function disables all interrupts given by *mask*. 0xff disables all.

Parameters

[in] **mask** mask indicating the interrupts to be disabled.

◆ st25r3916EnableInterrupts()

void st25r3916EnableInterrupts (uint32_t mask)

This function enables all interrupts given by *mask*, ST25R3916_IRQ_MASK_ALL enables all interrupts.

Parameters

[in] **mask** mask indicating the interrupts to be enabled

◆ st25r3916GetInterrupt()

uint32_t st25r3916GetInterrupt (uint32_t mask)

This function is used to check whether the interrupt given by *mask* has occured. If yes the interrupt gets cleared. This function returns only status bits which are inside *mask*.

Parameters

[in] **mask** : mask indicating the interrupt to check for.

Returns

the mask of the interrupts occurred

◆ st25r3916InitInterrupts()

void st25r3916InitInterrupts (void)

This function is used to check whether the interrupt given by *mask* has occured.

◆ st25r3916IRQCallbackRestore()

void st25r3916IRQCallbackRestore (void)

◆ st25r3916IRQCallbackSet()

void st25r3916IRQCallbackSet (void(*)(void) cb)

Parameters

[in] **cb** pointer to the callback method

◆ st25r3916Isr()

void st25r3916Isr (void)

This function modiefies the interupt

◆ st25r3916ModifyInterrupts()

void st25r3916ModifyInterrupts (uint32_t clr_mask,
uint32_t set_mask
)

This function modifies the interrupt

Parameters

[in] **clr_mask** : bit mask to be cleared on the interrupt mask

[in] **set_mask** : bit mask to be set on the interrupt mask

◆ st25r3916WaitForInterruptsTimed()

```
uint32_t st25r3916WaitForInterruptsTimed ( uint32_t  mask,
                                           uint16_t  tmo
                                         )
```

This function is used to access the ST25R3916 interrupt flags. Use this to wait for max. *tmo* milliseconds for the **first** interrupt indicated with mask *mask* to occur.

Parameters

[in] **mask** : mask indicating the interrupts to wait for.

[in] **tmo** : time in milliseconds until timeout occurs. If set to 0 the functions waits forever.

Returns

: 0 if timeout occured otherwise a mask indicating the cleared interrupts.

Generated on Fri Mar 14 2025 11:45:09 for RFAL by doxygen 1.9.5

RFAL

ST25R3916_LED

RFAL ST25R3916 LED. More...

Collaboration diagram for ST25R3916_LED:

Functions

void **st25r3916ledInit** (void)
ST25R3916 LED Initialize. More...

void **st25r3916ledEvtIrq** (uint32_t irqs)
ST25R3916 LED Event Interrupt. More...

void **st25r3916ledEvtWrReg** (uint8_t reg, uint8_t val)
ST25R3916 LED Event Write Register. More...

void **st25r3916ledEvtWrMultiReg** (uint8_t reg, const uint8_t *vals, uint8_t len)
ST25R3916 LED Event Write Multiple Register. More...

void **st25r3916ledEvtCmd** (uint8_t cmd)
ST25R3916 LED Event Direct Command. More...

void **st25r3916ledEvtCmds** (const uint8_t *cmds, uint8_t len)
ST25R3916 LED Event Direct Commands. More...

Detailed Description

Function Documentation

◆ st25r3916ledEvtCmd()

void st25r3916ledEvtCmd (uint8_t cmd)

This function should be called upon a ST25R3916 direct command, providing the event with the command executed

Parameters

[in] **cmd** ST25R3916 cmd executed

◆ st25r3916ledEvtCmds()

```
void st25r3916ledEvtCmds ( const uint8_t *  cmds,
                           uint8_t          len
                         )
```

This function should be called upon the execution of ST25R3916 direct commands, providing the event with the commands executed

Parameters

[in] **cmds** ST25R3916 cmds executed

[in] **len** : number of commands executed

◆ st25r3916ledEvtIrq()

```
void st25r3916ledEvtIrq ( uint32_t  irqs )
```

This function should be called upon a ST25R3916 Interrupt, providing the interrupt event with ST25R3916 irq flags to update LEDs

Parameters

[in] **irqs** ST25R3916 irqs mask

◆ st25r3916ledEvtWrMultiReg()

```
void st25r3916ledEvtWrMultiReg ( uint8_t          reg,
                                 const uint8_t *  vals,
                                 uint8_t          len
                               )
```

This function should be called upon a ST25R3916 Write Multiple Registers, providing the event with the registers and values to update LEDs

Parameters

[in] **reg** : ST25R3916 first register written

[in] **vals** pointer to the values written

[in] **len** : number of registers written

◆ st25r3916ledEvtWrReg()

```
void st25r3916ledEvtWrReg ( uint8_t  reg,
                            uint8_t  val
                          )
```

This function should be called on a ST25R3916 Write Register operation providing the event with the register and value to update LEDs

Parameters

[in] **reg** ST25R3916 register to be written

[in] **val** value to be written on the register

◆ st25r3916ledInit()

```
void st25r3916ledInit ( void  )
```

This function initializes the LEDs that represent ST25R3916 activity

ST25R500

RFAL ST25R500 Driver. More...

Collaboration diagram for ST25R500:

Modules

ST25R500_Driver
RFAL ST25R500 Driver.

ST25R500_COM
RFAL ST25R500 Communications.

ST25R500_IRQ
RFAL ST25R500 IRQ.

Detailed Description

RFAL

ST25R500_Driver

RFAL ST25R500 Driver. More...

Collaboration diagram for ST25R500_Driver:

Data Structures

struct **st25r500DpocrConfig**

struct **st25r500DpocrInfo**

Macros

#define **ST25R500_CMD_SET_DEFAULT** 0x60U

#define **ST25R500_CMD_STOP** 0x62U

#define **ST25R500_CMD_CLEAR_FIFO** 0x64U

#define **ST25R500_CMD_CLEAR_RXGAIN** 0x66U

#define **ST25R500_CMD_ADJUST_REGULATORS** 0x68U

#define **ST25R500_CMD_TRANSMIT** 0x6AU

#define **ST25R500_CMD_TRANSMIT_EOF** 0x6CU

#define **ST25R500_CMD_NFC_FIELD_ON** 0x6EU

#define **ST25R500_CMD_MASK_RECEIVE_DATA** 0x70U

#define **ST25R500_CMD_UNMASK_RECEIVE_DATA** 0x72U

#define **ST25R500_CMD_CALIBRATE_WU** 0x74U

#define **ST25R500_CMD_CLEAR_WU_CALIB** 0x76U

```
#define ST25R500_CMD_MEASURE_WU 0x78U
#define ST25R500_CMD_MEASURE_IQ 0x7AU
#define ST25R500_CMD_SENSE_RF 0x7CU
#define ST25R500_CMD_TRIGGER_WU_EV 0x7EU
#define ST25R500_CMD_START_GP_TIMER 0xE2U
#define ST25R500_CMD_START_WUT 0xE4U
#define ST25R500_CMD_START_MRT 0xE6U
#define ST25R500_CMD_START_NRT 0xE8U
#define ST25R500_CMD_STOP_NRT 0xEAU
#define ST25R500_CMD_CALIBRATE_RC 0xEEU
#define ST25R500_CMD_TRIGGER_DIAG 0xF8U
#define ST25R500_CMD_TEST_ACCESS 0xFCU
#define ST25R500_BR_DO_NOT_SET 0xFFU
#define ST25R500_BR_106_26 0x00U
#define ST25R500_BR_212_53 0x01U
#define ST25R500_BR_424 0x02U
#define ST25R500_BR_848 0x03U
#define ST25R500_REG_DROP_200 0U
#define ST25R500_REG_DROP_250 1U
#define ST25R500_REG_DROP_300 2U
#define ST25R500_REG_DROP_350 3U
```

```
#define ST25R500_REG_DROP_400 4U
#define ST25R500_REG_DROP_450 5U
#define ST25R500_REG_DROP_500 6U
#define ST25R500_REG_DROP_550 7U
#define ST25R500_REG_DROP_DO_NOT_SET 0xFFU
#define ST25R500_THRESHOLD_DO_NOT_SET 0xFFU
#define ST25R500_REG_LEN 1U
#define ST25R500_CMD_LEN 1U
#define ST25R500_FIFO_DEPTH 256U
#define ST25R500_TOUT_OSC_STABLE 5U
#define ST25R500_WRITE_MODE (0U << 7)
#define ST25R500_READ_MODE (1U << 7)
#define ST25R500_CMD_MODE ST25R500_WRITE_MODE
#define ST25R500_FIFO_ACCESS (0x5FU)
#define ST25R500_DIAG_MEAS_CMD 0x01U
#define ST25R500_DIAG_MEAS_CMD_LEN 0x02U
#define ST25R500_DIAG_MEAS_RES_LEN 0x04U
#define ST25R500_DIAG_MEAS_I_VDD_DR 0x01U
#define ST25R500_DIAG_MEAS_VDD_TX 0x02U
#define ST25R500_DIAG_MEAS_VDD_DR 0x03U
#define ST25R500_DIAG_MEAS_VDD_IO 0x04U
```

#define **ST25R500_DIAG_MEAS_VDD_D** 0x0CU

#define **ST25R500_DIAG_MEAS_VDD_A** 0x0DU

#define **ST25R500_DIAG_MEAS_VDD_VDD** 0x11U

#define **ST25R500_DIAG_MEAS_VDD_AGD** 0x12U

#define **st25r500TxRxOn**() **st25r500SetRegisterBits**(**ST25R500_REG_OPERATION**, (ST25R500_REG_OPERATION_rx_en | ST25R500_REG_OPERATION_tx_en))

#define **st25r500TxRxOff**() **st25r500ClrRegisterBits**(**ST25R500_REG_OPERATION**, (ST25R500_REG_OPERATION_rx_en | ST25R500_REG_OPERATION_tx_en))

#define **st25r500VDDDROn**() **st25r500SetRegisterBits**(**ST25R500_REG_OPERATION**, (ST25R500_REG_OPERATION_vdddr_en | ST25R500_REG_OPERATION_vdddr_en))

#define **st25r500VDDDROff**() **st25r500ClrRegisterBits**(**ST25R500_REG_OPERATION**, (ST25R500_REG_OPERATION_vdddr_en | ST25R500_REG_OPERATION_vdddr_en))

#define **st25r500TxOff**() **st25r500ClrRegisterBits**(**ST25R500_REG_OPERATION**, ST25R500_REG_OPERATION_tx_en)

#define **st25r500IsGPTRunning**() **st25r500CheckReg**(**ST25R500_REG_STATUS2**, ST25R500_REG_STATUS2_gpt_on, ST25R500_REG_STATUS2_gpt_on)

#define **st25r500IsExtFieldOn**() **st25r500CheckReg**(**ST25R500_REG_STATUS1**, ST25R500_REG_STATUS1_efd_out, ST25R500_REG_STATUS1_efd_out)

#define **st25r500IsTxEnabled**() **st25r500CheckReg**(**ST25R500_REG_OPERATION**, ST25R500_REG_OPERATION_tx_en, ST25R500_REG_OPERATION_tx_en)

#define **st25r500IsNRTinEMV**() **st25r500CheckReg**(**ST25R500_REG_NRT_GPT_CONF**, ST25R500_REG_NRT_GPT_CONF_nrt_emv, ST25R500_REG_NRT_GPT_CONF_nrt_emv_on)

#define **st25r500IsLastFIFOComplete**() **st25r500CheckReg**(**ST25R500_REG_FIFO_STATUS2**, ST25R500_REG_FIFO_STATUS2_fifo_lb_mask, 0)

#define **st25r500IsOscOn**() **st25r500CheckReg**(**ST25R500_REG_OPERATION**, ST25R500_REG_OPERATION_en, ST25R500_REG_OPERATION_en)

#define **st25r500IsTxOn**() **st25r500CheckReg**(**ST25R500_REG_OPERATION**, ST25R500_REG_OPERATION_tx_en, ST25R500_REG_OPERATION_tx_en)

#define **ST25R500_DPOCR_MAX_ENTRIES** 4U

Functions

ReturnCode **st25r500Initialize** (void)
Initialise ST25R500 driver. More...

void **st25r500Deinitialize** (void)
Deinitialize ST25R500 driver. More...

ReturnCode **st25r500OscOn** (void)
Turn on Oscillator and Regulator. More...

ReturnCode **st25r500SetBitrate** (uint8_t txRate, uint8_t rxRate)
Sets the bitrate. More...

ReturnCode **st25r500AdjustRegulators** (uint8_t drop, uint16_t *result)
Adjusts supply regulators according to the current supply voltage. More...

ReturnCode **st25r500SetRegulators** (uint8_t regulation)
Sets supply regulators. More...

ReturnCode **st25r500MeasureAmplitude** (uint8_t *result)
Measure Amplitude. More...

ReturnCode **st25r500GetCeGain** (uint8_t *result)
Get Ce gain. More...

ReturnCode **st25r500MeasureIQ** (int8_t *resI, int8_t *resQ)
Measure I/Q. More...

ReturnCode **st25r500MeasureCombinedIQ** (uint8_t *res)
Measure Combined I/Q. More...

ReturnCode **st25r500MeasureI** (uint8_t *res)

Measure I. More...

ReturnCode	**st25r500MeasureQ** (uint8_t *res) Measure Q. More...
ReturnCode	**st25r500MeasurePhase** (uint8_t *res) Measure Phase. More...
ReturnCode	**st25r500MeasureCurrent** (uint8_t *res) Measure electrical current. More...
ReturnCode	**st25r500CalibrateWU** (uint16_t *resI, uint16_t *resQ) Calibrate WU. More...
ReturnCode	**st25r500ClearCalibration** (void) Clear WU Calibration. More...
ReturnCode	**st25r500MeasureWU** (uint8_t *resI, uint8_t *resQ) Measure WU. More...
uint32_t	**st25r500GetNoResponseTime** (void) Get NRT time. More...
ReturnCode	**st25r500SetNoResponseTime** (uint32_t nrt) Set NRT time. More...
ReturnCode	**st25r500SetMRT** (uint32_t value_fc) Set MRT time. More...
ReturnCode	**st25r500SetStartNoResponseTimer** (uint32_t nrt) Set and Start NRT. More...
void	**st25r500SetGPTime** (uint16_t gpt) Set GPT time. More...
ReturnCode	**st25r500SetStartGPTimer** (uint16_t gpt, uint8_t trigger) Set and Start GPT. More...
void	**st25r500SetNumTxBits** (uint16_t nBits)

Sets the number Tx Bits. More...

uint16_t **st25r500GetNumFIFOBytes** (void)
Get amount of bytes in FIFO. More...

uint8_t **st25r500GetNumFIFOLastBits** (void)
Get amount of bits of the last FIFO byte if incomplete. More...

ReturnCode **st25r500PerformCollisionAvoidance** (uint8_t atThreshold, uint8_t dtThreshold, uint8_t nTRFW)
Perform Collision Avoidance. More...

bool **st25r500CheckChipID** (uint8_t *rev)
Check Identity. More...

ReturnCode **st25r500GetRegsDump** (uint8_t *resRegDump, uint8_t *sizeRegDump)
Register Dump. More...

bool **st25r500IsCmdValid** (uint8_t cmd)
Check if command is valid. More...

ReturnCode **st25r500ExecuteCommandAndGetResult** (uint8_t cmd, uint8_t resReg, uint8_t tOut, uint8_t *result)
Executes a direct command and returns the result. More...

ReturnCode **st25r500GetRSSI** (uint16_t *iRssi, uint16_t *qRssi)
Gets the RSSI values. More...

ReturnCode **st25r500SetAntennaMode** (bool single, bool rfiox)
Set Antenna mode. More...

ReturnCode **st25r500DiagMeasure** (uint8_t meas, uint16_t *res)
Diagnostic Measurement. More...

ReturnCode **st25r500DpocrInitialize** (const **st25r500DpocrConfig** *config)
Initialize dynamic power table. More...

ReturnCode **st25r500DpocrAdjust** (void)
Dynamic power adjust. More...

void **st25r500DpocrSetEnabled** (bool enable)

Dynamic power set enabled state. More...

ReturnCode	**st25r500DpocrConfigWrite** (const **st25r500DpocrConfig** *config) Write dynamic power config. More...
ReturnCode	**st25r500DpocrConfigRead** (**st25r500DpocrConfig** *config) Dynamic power table Read. More...
void	**st25r500DpocrReqAdj** (void) Request DPO adjust. More...
ReturnCode	**st25r500DpocrGetInfo** (**st25r500DpocrInfo** *info) Get Dynamic power information. More...

Detailed Description

Macro Definition Documentation

◆ ST25R500_BR_106_26

```
#define ST25R500_BR_106_26   0x00U
```

ST25R500 Bit Rate 106 kbps (fc/128) / 26 kbps(fc/512)

◆ ST25R500_BR_212_53

```
#define ST25R500_BR_212_53   0x01U
```

ST25R500 Bit Rate 212 kbps (fc/64)

◆ ST25R500_BR_424

```
#define ST25R500_BR_424   0x02U
```

ST25R500 Bit Rate 424 kbps (fc/32) / 53 kbps(fc/256)

◆ ST25R500_BR_848

```
#define ST25R500_BR_848   0x03U
```

ST25R500 Bit Rate 848 kbps (fc/16)

◆ ST25R500_BR_DO_NOT_SET

```
#define ST25R500_BR_DO_NOT_SET   0xFFU
```

Indicates not to change this Bit Rate

◆ ST25R500_CMD_ADJUST_REGULATORS

```
#define ST25R500_CMD_ADJUST_REGULATORS   0x68U
```

Adjust regulators

◆ ST25R500_CMD_CALIBRATE_RC

```
#define ST25R500_CMD_CALIBRATE_RC   0xEEU
```

Calibrate RC

◆ ST25R500_CMD_CALIBRATE_WU

```
#define ST25R500_CMD_CALIBRATE_WU   0x74U
```

Calibrate Wake-up Measurement

◆ ST25R500_CMD_CLEAR_FIFO

```
#define ST25R500_CMD_CLEAR_FIFO   0x64U
```

Clears FIFO, Collision and IRQ status

◆ ST25R500_CMD_CLEAR_RXGAIN

```
#define ST25R500_CMD_CLEAR_RXGAIN   0x66U
```

Clears FIFO, Collision and IRQ status

◆ ST25R500_CMD_CLEAR_WU_CALIB

```
#define ST25R500_CMD_CLEAR_WU_CALIB   0x76U
```

Clear Wake-up Calibratation

◆ ST25R500_CMD_LEN

```
#define ST25R500_CMD_LEN   1U
```

ST25R500 CMD length

◆ ST25R500_CMD_MASK_RECEIVE_DATA

```
#define ST25R500_CMD_MASK_RECEIVE_DATA   0x70U
```

Mask receive data

◆ ST25R500_CMD_MEASURE_IQ

```
#define ST25R500_CMD_MEASURE_IQ   0x7AU
```

Measure I and Q components

◆ ST25R500_CMD_MEASURE_WU

```
#define ST25R500_CMD_MEASURE_WU   0x78U
```

Measure Wake-up I and Q components

◆ ST25R500_CMD_MODE

#define ST25R500_CMD_MODE **ST25R500_WRITE_MODE**

ST25R500 Operation Mode: Direct Command

◆ ST25R500_CMD_NFC_FIELD_ON

#define ST25R500_CMD_NFC_FIELD_ON 0x6EU

Field On

◆ ST25R500_CMD_SENSE_RF

#define ST25R500_CMD_SENSE_RF 0x7CU

Sense RF on RFI pins

◆ ST25R500_CMD_SET_DEFAULT

#define ST25R500_CMD_SET_DEFAULT 0x60U

Puts the chip in default state (same as after power-up)

◆ ST25R500_CMD_START_GP_TIMER

#define ST25R500_CMD_START_GP_TIMER 0xE2U

Start the general purpose timer

◆ ST25R500_CMD_START_MRT

#define ST25R500_CMD_START_MRT 0xE6U

Start the mask-receive timer

◆ ST25R500_CMD_START_NRT

```
#define ST25R500_CMD_START_NRT   0xE8U
```

Start the no-response timer

◆ ST25R500_CMD_START_WUT

```
#define ST25R500_CMD_START_WUT   0xE4U
```

Start the wake-up timer

◆ ST25R500_CMD_STOP

```
#define ST25R500_CMD_STOP   0x62U
```

Stops all activities and clears FIFO

◆ ST25R500_CMD_STOP_NRT

```
#define ST25R500_CMD_STOP_NRT   0xEAU
```

Stop No Response Timer

◆ ST25R500_CMD_TEST_ACCESS

```
#define ST25R500_CMD_TEST_ACCESS   0xFCU
```

Enable R/W access to the test registers

◆ ST25R500_CMD_TRANSMIT

```
#define ST25R500_CMD_TRANSMIT   0x6AU
```

Transmit

◆ ST25R500_CMD_TRANSMIT_EOF

```
#define ST25R500_CMD_TRANSMIT_EOF   0x6CU
```

Transmit ISO15693 EOF

◆ ST25R500_CMD_TRIGGER_DIAG

```
#define ST25R500_CMD_TRIGGER_DIAG   0xF8U
```

Trigger Diagnostic Measurement

◆ ST25R500_CMD_TRIGGER_WU_EV

```
#define ST25R500_CMD_TRIGGER_WU_EV   0x7EU
```

Trigger Wake-up Event

◆ ST25R500_CMD_UNMASK_RECEIVE_DATA

```
#define ST25R500_CMD_UNMASK_RECEIVE_DATA   0x72U
```

Unmask receive data

◆ ST25R500_DIAG_MEAS_CMD

```
#define ST25R500_DIAG_MEAS_CMD   0x01U
```

ST25R500 Diagnostic Measurement cmd size

◆ ST25R500_DIAG_MEAS_CMD_LEN

```
#define ST25R500_DIAG_MEAS_CMD_LEN   0x02U
```

ST25R500 Diagnostic Measurement cmd length

◆ ST25R500_DIAG_MEAS_I_VDD_DR

```
#define ST25R500_DIAG_MEAS_I_VDD_DR   0x01U
```

ST25R500 Diagnostic Measurement: I_VDD_DR

◆ ST25R500_DIAG_MEAS_RES_LEN

```
#define ST25R500_DIAG_MEAS_RES_LEN   0x04U
```

ST25R500 Diagnostic Measurement res length

◆ ST25R500_DIAG_MEAS_VDD_A

```
#define ST25R500_DIAG_MEAS_VDD_A   0x0DU
```

ST25R500 Diagnostic Measurement: VDD_A

◆ ST25R500_DIAG_MEAS_VDD_AGD

```
#define ST25R500_DIAG_MEAS_VDD_AGD   0x12U
```

ST25R500 Diagnostic Measurement: AGD

◆ ST25R500_DIAG_MEAS_VDD_D

```
#define ST25R500_DIAG_MEAS_VDD_D   0x0CU
```

ST25R500 Diagnostic Measurement: VDD_D

◆ ST25R500_DIAG_MEAS_VDD_DR

```
#define ST25R500_DIAG_MEAS_VDD_DR   0x03U
```

ST25R500 Diagnostic Measurement: VDD_DR

◆ ST25R500_DIAG_MEAS_VDD_IO

```
#define ST25R500_DIAG_MEAS_VDD_IO   0x04U
```

ST25R500 Diagnostic Measurement: VDD_IO

◆ ST25R500_DIAG_MEAS_VDD_TX

```
#define ST25R500_DIAG_MEAS_VDD_TX   0x02U
```

ST25R500 Diagnostic Measurement: VDD_TX

◆ ST25R500_DIAG_MEAS_VDD_VDD

```
#define ST25R500_DIAG_MEAS_VDD_VDD   0x11U
```

ST25R500 Diagnostic Measurement: VDD

◆ ST25R500_DPOCR_MAX_ENTRIES

```
#define ST25R500_DPOCR_MAX_ENTRIES   4U
```

Max number of table entries

◆ ST25R500_FIFO_ACCESS

```
#define ST25R500_FIFO_ACCESS   (0x5FU)
```

ST25R500 FIFO Access

◆ ST25R500_FIFO_DEPTH

```
#define ST25R500_FIFO_DEPTH   256U
```

Depth of FIFO

◆ ST25R500_READ_MODE

```
#define ST25R500_READ_MODE   (1U << 7)
```

ST25R500 Operation Mode: Read

◆ ST25R500_REG_DROP_200

```
#define ST25R500_REG_DROP_200   0U
```

ST25R500 target drop for regulator adjustment: 200mV

◆ ST25R500_REG_DROP_250

```
#define ST25R500_REG_DROP_250   1U
```

ST25R500 target drop for regulator adjustment: 250mV

◆ ST25R500_REG_DROP_300

```
#define ST25R500_REG_DROP_300   2U
```

ST25R500 target drop for regulator adjustment: 300mV

◆ ST25R500_REG_DROP_350

```
#define ST25R500_REG_DROP_350   3U
```

ST25R500 target drop for regulator adjustment: 350mV

◆ ST25R500_REG_DROP_400

```
#define ST25R500_REG_DROP_400   4U
```

ST25R500 target drop for regulator adjustment: 400mV

◆ ST25R500_REG_DROP_450

```
#define ST25R500_REG_DROP_450   5U
```

ST25R500 target drop for regulator adjustment: 450mV

◆ ST25R500_REG_DROP_500

```
#define ST25R500_REG_DROP_500   6U
```

ST25R500 target drop for regulator adjustment: 500mV

◆ ST25R500_REG_DROP_550

```
#define ST25R500_REG_DROP_550   7U
```

ST25R500 target drop for regulator adjustment: 550mV

◆ ST25R500_REG_DROP_DO_NOT_SET

```
#define ST25R500_REG_DROP_DO_NOT_SET   0xFFU
```

Indicates not to change this setting (regd)

◆ ST25R500_REG_LEN

```
#define ST25R500_REG_LEN   1U
```

Number of bytes in a ST25R500 register

◆ ST25R500_THRESHOLD_DO_NOT_SET

```
#define ST25R500_THRESHOLD_DO_NOT_SET   0xFFU
```

Indicates not to change this Threshold

◆ ST25R500_TOUT_OSC_STABLE

```
#define ST25R500_TOUT_OSC_STABLE   5U
```

Timeout for Oscillator to get stable

◆ ST25R500_WRITE_MODE

```
#define ST25R500_WRITE_MODE   (0U << 7)
```

ST25R500 Operation Mode: Write

◆ st25r500IsExtFieldOn

```
#define st25r500IsExtFieldOn ( ) st25r500CheckReg( ST25R500_REG_STATUS1, ST25R500_REG_STATUS1_efd_out, ST25R500_REG_STATUS1_efd_out )
```

Checks if External Filed is detected by reading ST25R3916 External Field Detector output

◆ st25r500IsGPTRunning

```
#define st25r500IsGPTRunning ( ) st25r500CheckReg( ST25R500_REG_STATUS2, ST25R500_REG_STATUS2_gpt_on, ST25R500_REG_STATUS2_gpt_on )
```

Checks if General Purpose Timer is still running by reading gpt_on flag

◆ st25r500IsLastFIFOComplete

```
#define st25r500IsLastFIFOComplete ( ) st25r500CheckReg( ST25R500_REG_FIFO_STATUS2, ST25R500_REG_FIFO_STATUS2_fifo_lb_mask, 0 )
```

Checks if last FIFO byte is complete

◆ st25r500IsNRTinEMV

```
#define st25r500IsNRTinEMV ( ) st25r500CheckReg( ST25R500_REG_NRT_GPT_CONF, ST25R500_REG_NRT_GPT_CONF_nrt_emv, ST25R500_REG_NRT_GPT_CONF_nrt_emv_on )
```

Checks if NRT is in EMV mode

◆ st25r500IsOscOn

```
#define st25r500IsOscOn ( ) st25r500CheckReg( ST25R500_REG_OPERATION, ST25R500_REG_OPERATION_en, ST25R500_REG_OPERATION_en )
```

Checks if the Oscillator is enabled

◆ st25r500IsTxEnabled

```
#define st25r500IsTxEnabled ( ) st25r500CheckReg( ST25R500_REG_OPERATION, ST25R500_REG_OPERATION_tx_en, ST25R500_REG_OPERATION_tx_en )
```

Checks if Transmitter is enabled (Field On)

◆ st25r500IsTxOn

```
#define st25r500IsTxOn ( ) st25r500CheckReg( ST25R500_REG_OPERATION, ST25R500_REG_OPERATION_tx_en, ST25R500_REG_OPERATION_tx_en )
```

Checks if Transmitter (Field On) is enabled

◆ st25r500TxOff

```
#define st25r500TxOff ( ) st25r500ClrRegisterBits( ST25R500_REG_OPERATION, ST25R500_REG_OPERATION_tx_en )
```

Disables the Transmitter (Field Off)

◆ st25r500TxRxOff

```
#define st25r500TxRxOff ( ) st25r500ClrRegisterBits( ST25R500_REG_OPERATION, (ST25R500_REG_OPERATION_rx_en | ST25R500_REG_OPERATION_tx_en ) )
```

Disables the Transmitter (Field Off) and Receiver

◆ st25r500TxRxOn

```
#define st25r500TxRxOn ( ) st25r500SetRegisterBits( ST25R500_REG_OPERATION, (ST25R500_REG_OPERATION_rx_en | ST25R500_REG_OPERATION_tx_en ) )
```

Enables the Transmitter (Field On) and Receiver

◆ st25r500VDDDROff

```
#define st25r500VDDDROff ( ) st25r500ClrRegisterBits( ST25R500_REG_OPERATION, (ST25R500_REG_OPERATION_vdddr_en | ST25R500_REG_OPERATION_vdddr_en ) )
```

Disables the Transmitter (Field Off) and Receiver

◆ st25r500VDDDROn

```
#define st25r500VDDDROn ( ) st25r500SetRegisterBits( ST25R500_REG_OPERATION, (ST25R500_REG_OPERATION_vdddr_en | ST25R500_REG_OPERATION_vdddr_en ) )
```

Enables the VDD_DR regulator

Function Documentation

◆ st25r500AdjustRegulators()

```
ReturnCode st25r500AdjustRegulators ( uint8_t drop,
                                      uint16_t * result
                                    )
```

The power level is measured in maximum load conditions and the regulated voltage reference is set below this level.

The regulated voltages will be set to the result of Adjust Regulators

Parameters

[in] **drop** : Targeted drop from supply voltage

[out] **result** : Result of calibration in milliVolts

Returns

RFAL_ERR_IO : Error during communication with ST25R500

RFAL_ERR_NONE : No error

◆ st25r500CalibrateWU()

```
ReturnCode st25r500CalibrateWU ( uint16_t *  resI,
                                 uint16_t *  resQ
                               )
```

This function executes Wake-up Calibration

Parameters

[out] **resI** I channel calibration (unsigned)

[out] **resQ** Q channel calibration (unsigned)

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error

◆ st25r500CheckChipID()

```
bool st25r500CheckChipID ( uint8_t *  rev )
```

Checks if the chip ID is as expected.

5 bit IC type code for ST25R500: 00101 The 3 lsb contain the IC revision code

Parameters

[out] **rev** : the IC revision code

Returns

true when IC type is as expected

false otherwise

◆ st25r500ClearCalibration()

ReturnCode st25r500ClearCalibration (void)

This function clears WU calibration

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error

◆ st25r500Deinitialize()

void st25r500Deinitialize (void)

Calling this function deinitializes the ST25R500 driver.

◆ st25r500DiagMeasure()

ReturnCode st25r500DiagMeasure (uint8_t meas,
uint16_t * res
)

Performs a diagnostic measurement of certain currents and voltages. The function does not work when oscillator is not enabled. For its operation it may temporarily enable the reciever (rx_en=1). Ongoing transmissions/receptions (also in CE mode) may be garbled when this functions is executed.

Parameters

[in] **meas** the measurement function to perform, see ST25R500_DIAG_MEAS_*

[out] **res** 9 bit unsigned value

Returns

RFAL_ERR_WRONG_STATE : Measurement could not be performed (e.g. en=0)

RFAL_ERR_NONE : No error

◆ st25r500DpocrAdjust()

ReturnCode st25r500DpocrAdjust (void)

This function measures the current output and adjusts the power accordingly to the dynamic power table. This method | The adjustment shall be performed when the device is already emiting RF field.

Returns

RFAL_ERR_NONE : No error

RFAL_ERR_PARAM : If configTbl is invalid or parameters are invalid

RFAL_ERR_WRONG_STATE : If the current state is valid for DPO Adjustment

◆ st25r500DpocrConfigRead()

ReturnCode st25r500DpocrConfigRead (**st25r500DpocrConfig** * config)

Read the dynamic power config

Parameters

[out] **config** : location to the config to place the current config

Returns

RFAL_ERR_NONE : No error

RFAL_ERR_PARAM : If configTbl is invalid or parameters are invalid

◆ st25r500DpocrConfigWrite()

ReturnCode st25r500DpocrConfigWrite (const **st25r500DpocrConfig** * config)

Load the dynamic power config

Parameters

[in] **config** : location of power Table to be loaded

Returns

RFAL_ERR_NONE : No error

RFAL_ERR_PARAM : If configTbl is invalid

RFAL_ERR_NOMEM : If the given Table is bigger exceeds the max size

◆ st25r500DpocrGetInfo()

ReturnCode st25r500DpocrGetInfo (**st25r500DpocrInfo** * info)

Get the DPO information|status

Parameters

[out] **info** : pointer where DPO info is to be stored

Returns

RFAL_ERR_NONE : No error

RFAL_ERR_PARAM : Invalid parameter

◆ st25r500DpocrInitialize()

ReturnCode st25r500DpocrInitialize (const **st25r500DpocrConfig** * config)

Initialize the R500 current regulation DPO

Parameters

[in] **config** : pointer to a user provided config or NULL if default one to be used

Returns

RFAL_ERR_NONE : No error

RFAL_ERR_WRONG_STATE : If the current state is valid for DPO operation

◆ st25r500DpocrReqAdj()

void st25r500DpocrReqAdj (void)

Flags the DPO module to perform the adjustment on the next round, even if no change has been perceived from previous measurement|state

◆ st25r500DpocrSetEnabled()

void st25r500DpocrSetEnabled (bool enable)

Parameters

[in] **enable** : new active state

Set state to enable or disable the Dynamic power adjustment

◆ st25r500ExecuteCommandAndGetResult()

```
ReturnCode st25r500ExecuteCommandAndGetResult ( uint8_t    cmd,
                                                uint8_t    resReg,
                                                uint8_t    tOut,
                                                uint8_t *  result
                                              )
```

This function executes the direct command given by *cmd* waits for *sleeptime* for I_dct and returns the result read from register *resreg*. The value of cmd is not checked.

Parameters

- [in] **cmd** : direct command to execute
- [in] **resReg** address of the register containing the result
- [in] **tOut** : time in milliseconds to wait before reading the result
- [out] **result** result

Returns

RFAL_ERR_NONE : No error

◆ st25r500GetCeGain()

```
ReturnCode st25r500GetCeGain ( uint8_t *  result )
```

This function retieves an RFI indicator and stores the result in parameter *result*.

Warning

This method can only be executed in Listen mode

Parameters

- [out] **result** received signal indicator

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error

◆ st25r500GetNoResponseTime()

```
uint32_t st25r500GetNoResponseTime ( void  )
```

This returns the last value set on the NRT

Warning

it does not read chip register, just the sw var that contains the last value set before

Returns

the value of the NRT in 64/fc

◆ st25r500GetNumFIFOBytes()

uint16_t st25r500GetNumFIFOBytes (void)

Gets the number of bytes currently in the FIFO

Returns

the number of bytes currently in the FIFO

◆ st25r500GetNumFIFOLastBits()

uint8_t st25r500GetNumFIFOLastBits (void)

Gets the number of bits of the last FIFO byte if incomplete

Returns

the number of bits of the last FIFO byte if incomplete, 0 if the last byte is complete

◆ st25r500GetRegsDump()

ReturnCode st25r500GetRegsDump (uint8_t * resRegDump,
uint8_t * sizeRegDump
)

Retrieves all internal registers from ST25R500

Parameters

[out]	**resRegDump**	: pointer to the struct/buffer where the reg dump will be written. NULL is allowed and causes only reading to internal buffer
[in,out]	**sizeRegDump**	: number of registers requested and the ones actually written

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error

◆ st25r500GetRSSI()

```
ReturnCode st25r500GetRSSI ( uint16_t *  iRssi,
                             uint16_t *  qRssi
                           )
```

This function gets the RSSI value of the previous reception taking into account the gain reductions that were used. RSSI value for both I and Q channel can be retrieved.

Parameters

[out] **iRssi** the RSSI on the I channel expressed in mV

[out] **qRssi** the RSSI on the Q channel expressed in mV

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error

◆ st25r500Initialize()

```
ReturnCode st25r500Initialize ( void  )
```

This function initialises the ST25R500 driver.

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_HW_MISMATCH : Expected HW do not match or communication error

◆ st25r500IsCmdValid()

```
bool st25r500IsCmdValid ( uint8_t  cmd )
```

Checks if the given command is a valid ST25R500 command

Parameters

[in] **cmd** Command to check

Returns

true if is a valid command

false otherwise

◆ st25r500MeasureAmplitude()

ReturnCode st25r500MeasureAmplitude (uint8_t * result)

This function measured the amplitude on the RFI inputs and stores the result in parameter *result*.

Parameters

[out] **result** result of RF measurement (unsigned)

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error

◆ st25r500MeasureCombinedIQ()

ReturnCode st25r500MeasureCombinedIQ (uint8_t * res)

This function performs an I/Q measurement and returns the vectorial magnitude

Parameters

[out] **res**

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error

◆ st25r500MeasureCurrent()

ReturnCode st25r500MeasureCurrent (uint8_t * res)

This function performs measurement of I_VDD_DR using Diagnostic Measurement Mode For limitations see **st25r500DiagMeasure()**.

Parameters

[out] **res** Raw value - approx 3.2mA/unit

Returns

RFAL_ERR_WRONG_STATE : Measurement could not be performed (e.g. en=0)

RFAL_ERR_NONE : No error

◆ st25r500MeasureI()

ReturnCode st25r500MeasureI (uint8_t * res)

This function performs an I/Q measurement and returns I

Parameters

[out] **res**

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error

◆ st25r500MeasureIQ()

ReturnCode st25r500MeasureIQ (int8_t * resI,
int8_t * resQ
)

This function performs an I/Q measurement The result is stored on the *resI* and *resQ* parameters.

Parameters

[out] **resI** 8 bit long result of the I channel (signed)

[out] **resQ** 8 bit long result of the Q channel (signed)

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error

◆ st25r500MeasurePhase()

ReturnCode st25r500MeasurePhase (uint8_t * res)

This function performs an I/Q measurement and returns the vectorial angle

Parameters

[out] **res**

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error

◆ st25r500MeasureQ()

ReturnCode st25r500MeasureQ (uint8_t * res)

This function performs an I/Q measurement and returns Q

Parameters

[out] **res**

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error

◆ st25r500MeasureWU()

ReturnCode st25r500MeasureWU (uint8_t * resI,
uint8_t * resQ
)

This function performs measuremnt as executed during WU mode

Parameters

[out] **resI** 8 bit long result of the I channel (signed)

[out] **resQ** 8 bit long result of the Q channel (signed)

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error

◆ st25r500OscOn()

ReturnCode st25r500OscOn (void)

This function turn on oscillator and regulator and waits for the oscillator to become stable

Returns

RFAL_ERR_SYSTEM: Unable to verify oscilator enable|stable

RFAL_ERR_NONE : No error

◆ st25r500PerformCollisionAvoidance()

```
ReturnCode st25r500PerformCollisionAvoidance ( uint8_t  atThreshold,
                                               uint8_t  dtThreshold,
                                               uint8_t  nTRFW
                                             )
```

Performs Collision Avoidance with the given threshold and with the
n number of TRFW

Parameters

[in] **atThreshold** : Activation Threshold (ST25R500_REG_EFD_THRESHOLDS_efd_at_xx) 0xff : don't set Threshold (ST25R500_THRESHOLD_DO_NOT_SET)

[in] **dtThreshold** : Deactivation Threshold (ST25R500_REG_EFD_THRESHOLDS_efd_dt_xx) 0xff : don't set Threshold (ST25R500_THRESHOLD_DO_NOT_SET)

[in] **nTRFW** : Number of TRFW

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_RF_COLLISION : Collision detected

RFAL_ERR_NONE : No collision detected

◆ st25r500SetAntennaMode()

```
ReturnCode st25r500SetAntennaMode ( bool  single,
                                    bool  rfiox
                                  )
```

Sets the antenna mode. Differential or single ended antenna mode (RFO1 or RFO2)

Parameters

[in] **single** FALSE differential ; single ended mode

[in] **rfiox** FALSE RFI1/RFO1 ; TRUE RFI2/RFO2

Returns

RFAL_ERR_IO : Internal error

RFAL_ERR_NOTSUPP : Feature not supported

RFAL_ERR_NONE : No error

◆ st25r500SetBitrate()

```
ReturnCode st25r500SetBitrate ( uint8_t  txRate,
                                uint8_t  rxRate
                              )
```

This function sets the bitrates for rx and tx

Parameters

txRate : speed is 2^txrate * 106 kb/s 0xff : don't set txrate (ST25R500_BR_DO_NOT_SET)

rxRate : speed is 2^rxrate * 106 kb/s 0xff : don't set rxrate (ST25R500_BR_DO_NOT_SET)

Returns

RFAL_ERR_PARAM: At least one bit rate was invalid

RFAL_ERR_NONE : No error, both bit rates were set

◆ st25r500SetGPTime()

```
void st25r500SetGPTime ( uint16_t  gpt )
```

This function sets the General Purpose Timer time registers

Parameters

[in] **gpt** : general purpose timer timeout in steps of 8/fc (590ns)

◆ st25r500SetMRT()

```
ReturnCode st25r500SetMRT ( uint32_t  value_fc )
```

This function sets the Mask Receive Time with the given value. It also adapts mrt_step to achieve highest precision.

Parameters

[in] **value_fc** : mask receive time in steps of 1/fc

Returns

RFAL_ERR_PARAM : Invalid parameter (time is too large), max will be set

RFAL_ERR_NONE : No error

◆ st25r500SetNoResponseTime()

ReturnCode st25r500SetNoResponseTime (uint32_t nrt)

This function sets the No Response Time with the given value

Parameters

[in] **nrt** : no response time in steps of 64/fc (4.72us)

Returns

RFAL_ERR_PARAM : Invalid parameter (time is too large)

RFAL_ERR_NONE : No error

◆ st25r500SetNumTxBits()

void st25r500SetNumTxBits (uint16_t nBits)

Sets ST25R500 internal registers with correct number of complete bytes and bits to be sent

Parameters

[in] **nBits** : number of bits to be set/transmitted

◆ st25r500SetRegulators()

ReturnCode st25r500SetRegulators (uint8_t regulation)

Manually sets the regulated voltage setting (rege). The regulated voltages will be set to the manual configuratiom.

Regulation shall be set according to the desired voltage considering the supplied voltage.

Parameters

[in] **regulation** : Regulator setting

Returns

RFAL_ERR_IO : Error during communication with ST25R500

RFAL_ERR_PARAM : Invalid regulator setting

RFAL_ERR_NONE : No error

◆ st25r500SetStartGPTimer()

```
ReturnCode st25r500SetStartGPTimer ( uint16_t gpt,
                                     uint8_t  trigger
                                   )
```

This function sets the General Purpose Timer with the given timeout and immediately starts it ONLY if the trigger source is not set to none.

Parameters

[in] **gpt** : general purpose timer timeout in steps of8/fc (590ns)

[in] **trigger** : no trigger, start of Rx, end of Rx, end of Tx in NFC mode

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error

◆ st25r500SetStartNoResponseTimer()

```
ReturnCode st25r500SetStartNoResponseTimer ( uint32_t nrt )
```

This function sets the No Response Time with the given value and immediately starts it Used when needs to add more time before timeout without performing Tx

Parameters

[in] **nrt** : no response time in steps of 64/fc (4.72us)

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error

st25r500DpocrConfig Struct Reference


```
#include <st25r500_dpocr.h>
```

Data Fields

bool	**enabled**
uint16_t	**target**
uint8_t	**maxRege**
uint8_t	**minRege**
uint8_t	**maxDres**
uint8_t	**minDres**
uint8_t	**regeStepSize**
uint16_t	**currThreshold**

uint16_t	**tableUpThresholds** [**ST25R500_DPOCR_MAX_ENTRIES**]
uint8_t	**numEntries**

Detailed Description

DPO table entry struct, to have link into AC, similar to rfal_dpo levels

Field Documentation

◆ currThreshold

uint16_t st25r500DpocrConfig::currThreshold

If current value is <(target-curr_threshold) or >(target+curr_treshold) regulation will again happen

◆ enabled

bool st25r500DpocrConfig::enabled

DPO Enabled state

◆ maxDres

uint8_t st25r500DpocrConfig::maxDres

Max dres which will be selected

◆ maxRege

uint8_t st25r500DpocrConfig::maxRege

Max regulator which will be selected

◆ minDres

uint8_t st25r500DpocrConfig::minDres

Min dres which will be selected

◆ minRege

uint8_t st25r500DpocrConfig::minRege

Min regulator which will be selected

◆ numEntries

uint8_t st25r500DpocrConfig::numEntries

Number of entries to be used in table

◆ regeStepSize

uint8_t st25r500DpocrConfig::regeStepSize

Step size for rege, 1 gives best accuracy

◆ tableUpThresholds

uint16_t st25r500DpocrConfig::tableUpThresholds[**ST25R500_DPOCR_MAX_ENTRIES**]

Table with rege levels for AC

◆ target

uint16_t st25r500DpocrConfig::target

Target value - electric current as per I_VDD_DR measurement

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

- st25r500_dpocr.h

st25r500DpocrInfo Struct Reference


```
#include <st25r500_dpocr.h>
```

Data Fields

uint8_t	**currentEntryIdx**
uint16_t	**currentElecCurrent**
uint8_t	**currentRege**
uint8_t	**currentDres**

Detailed Description

DPO information struct

Field Documentation

◆ currentDres

uint8_t st25r500DpocrInfo::currentDres

Current ST25R500 dres setting

◆ currentElecCurrent

uint16_t st25r500DpocrInfo::currentElecCurrent

Current DPO measured electrical current

◆ currentEntryIdx

uint8_t st25r500DpocrInfo::currentEntryIdx

Current DPO table Index

◆ currentRege

uint8_t st25r500DpocrInfo::currentRege

Current ST25R500 rege setting

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

- **st25r500_dpocr.h**

RFAL

ST25R500_COM

RFAL ST25R500 Communications. More...

Collaboration diagram for ST25R500_COM:

Macros

#define **ST25R500_FIFO_STATUS_LEN** 2

#define **ST25R500_OBS_MODE_LEN** 2

#define **ST25R500_CEM_A_LEN** 31U

#define **ST25R500_CEM_B_LEN** 0U

#define **ST25R500_CEM_F_LEN** 21U

#define **ST25R500_CEM_LEN** (**ST25R500_CEM_A_LEN** + **ST25R500_CEM_B_LEN** + **ST25R500_CEM_F_LEN**)

#define **ST25R500_REG_OPERATION** 0x00U

#define **ST25R500_REG_GENERAL** 0x01U

```
#define ST25R500_REG_REGULATOR 0x02U
#define ST25R500_REG_DRIVER 0x03U
#define ST25R500_REG_TX_MOD1 0x04U
#define ST25R500_REG_TX_MOD2 0x05U
#define ST25R500_REG_CE_TX_MOD1 0x06U
#define ST25R500_REG_CE_TX_MOD2 0x07U
#define ST25R500_REG_GPIO 0x08U
#define ST25R500_REG_RX_ANA1 0x09U
#define ST25R500_REG_RX_ANA2 0x0AU
#define ST25R500_REG_RX_ANA3 0x0BU
#define ST25R500_REG_RX_ANA4 0x0CU
#define ST25R500_REG_RX_DIG 0x0DU
#define ST25R500_REG_CORR1 0x0EU
#define ST25R500_REG_CORR2 0x0FU
#define ST25R500_REG_CORR3 0x10U
#define ST25R500_REG_CORR4 0x11U
#define ST25R500_REG_CORR5 0x12U
#define ST25R500_REG_CORR6 0x13U
#define ST25R500_REG_PROTOCOL 0x14U
#define ST25R500_REG_PROTOCOL_TX1 0x15U
#define ST25R500_REG_PROTOCOL_TX2 0x16U
```

```
#define ST25R500_REG_PROTOCOL_RX1 0x17U
#define ST25R500_REG_PROTOCOL_RX2 0x18U
#define ST25R500_REG_PROTOCOL_RX3 0x19U
#define ST25R500_REG_EMD1 0x1AU
#define ST25R500_REG_EMD2 0x1BU
#define ST25R500_REG_CE_CONFIG1 0x1CU
#define ST25R500_REG_CE_CONFIG2 0x1DU
#define ST25R500_REG_CE_CONFIG3 0x1EU
#define ST25R500_REG_MRT1 0x1FU
#define ST25R500_REG_MRT2 0x20U
#define ST25R500_REG_SQT 0x21U
#define ST25R500_REG_NRT_GPT_CONF 0x22U
#define ST25R500_REG_NRT1 0x23U
#define ST25R500_REG_NRT2 0x24U
#define ST25R500_REG_GPT1 0x25U
#define ST25R500_REG_GPT2 0x26U
#define ST25R500_REG_WAKEUP_CONF1 0x27U
#define ST25R500_REG_WAKEUP_CONF2 0x28U
#define ST25R500_REG_WAKEUP_CONF3 0x29U
#define ST25R500_REG_WU_I_CONF 0x2AU
#define ST25R500_REG_WU_I_DELTA 0x2BU
```

```
#define ST25R500_REG_WU_I_CAL 0x2CU
#define ST25R500_REG_WU_I_ADC 0x2DU
#define ST25R500_REG_WU_I_REF 0x2EU
#define ST25R500_REG_WU_Q_CONF 0x2FU
#define ST25R500_REG_WU_Q_DELTA 0x30U
#define ST25R500_REG_WU_Q_CAL 0x31U
#define ST25R500_REG_WU_Q_ADC 0x32U
#define ST25R500_REG_WU_Q_REF 0x33U
#define ST25R500_REG_TX_FRAME1 0x34U
#define ST25R500_REG_TX_FRAME2 0x35U
#define ST25R500_REG_FIFO_STATUS1 0x36U
#define ST25R500_REG_FIFO_STATUS2 0x37U
#define ST25R500_REG_COLLISION 0x38U
#define ST25R500_REG_IRQ_MASK1 0x39U
#define ST25R500_REG_IRQ_MASK2 0x3AU
#define ST25R500_REG_IRQ_MASK3 0x3BU
#define ST25R500_REG_IRQ1 0x3CU
#define ST25R500_REG_IRQ2 0x3DU
#define ST25R500_REG_IRQ3 0x3EU
#define ST25R500_REG_IC_ID 0x3FU
#define ST25R500_REG_STATUS1 0x40U
```

```
#define ST25R500_REG_STATUS2 0x41U
#define ST25R500_REG_STATUS_STATIC1 0x42U
#define ST25R500_REG_STATUS_STATIC2 0x43U
#define ST25R500_REG_STATUS_STATIC3 0x44U
#define ST25R500_REG_CE_STATUS1 0x45U
#define ST25R500_REG_CE_STATUS2 0x46U
#define ST25R500_REG_WU_STATUS 0x47U
#define ST25R500_REG_ANA_DISPLAY1 0x48U
#define ST25R500_REG_ANA_DISPLAY2 0x49U
#define ST25R500_REG_RSSI_I 0x4AU
#define ST25R500_REG_RSSI_Q 0x4BU
#define ST25R500_REG_SENSE_DISPLAY 0x4CU
#define ST25R500_REG_AWS_CONFIG1 0x4DU
#define ST25R500_REG_AWS_CONFIG2 0x4EU
#define ST25R500_REG_AWS_TIME1 0x4FU
#define ST25R500_REG_AWS_TIME2 0x50U
#define ST25R500_REG_AWS_TIME3 0x51U
#define ST25R500_REG_AWS_TIME4 0x52U
#define ST25R500_REG_OVERSHOOT_CONF 0x53U
#define ST25R500_REG_UNDERSHOOT_CONF 0x54U
#define ST25R500_REG_EFD_THRESHOLD 0x57U
```

#define **ST25R500_REG_CEM_A** 0x58U

#define **ST25R500_REG_CEM_F** 0x5AU

#define **ST25R500_REG_FIFO** 0x5FU

#define **ST25R500_TEST_REG_ADC2CE** 0x08U

#define **ST25R500_TEST_REG_OSC_TIMING** 0x0EU

#define **ST25R500_TEST_REG_MAN_TIMING** 0x13U

#define **ST25R500_TEST_REG_DIAG_MEAS** 0x14U

#define **ST25R500_TEST_REG_OVERLAP_CONTROL** 0x1EU

#define **ST25R500_TEST_REG_RXS_DISP1** 0x30U

#define **ST25R500_TEST_REG_RXS_DISP2** 0x31U

#define **ST25R500_TEST_REG_RXE_DISP1** 0x32U

#define **ST25R500_TEST_REG_RXE_DISP2** 0x33U

Functions

ReturnCode **st25r500ReadRegister** (uint8_t reg, uint8_t *val)

Returns the content of a register within the ST25R500. More...

ReturnCode **st25r500ReadMultipleRegisters** (uint8_t reg, uint8_t *values, uint16_t length)

Reads from multiple ST25R500 registers. More...

ReturnCode **st25r500WriteRegister** (uint8_t reg, uint8_t val)

Writes a given value to a register within the ST25R500. More...

ReturnCode **st25r500WriteMultipleRegisters** (uint8_t reg, const uint8_t *values, uint16_t length)

Writes multiple values to ST25R500 registers. More...

ReturnCode **st25r500WriteFifo** (const uint8_t *values, uint16_t length)

Writes values to ST25R500 FIFO. More...

ReturnCode	**st25r500ReadFifo** (uint8_t *buf, uint16_t length) Read values from ST25R500 FIFO. More...
ReturnCode	**st25r500ExecuteCommand** (uint8_t cmd) Execute a direct command. More...
ReturnCode	**st25r500ExecuteCommands** (const uint8_t *cmds, uint8_t length) Execute direct commands. More...
ReturnCode	**st25r500ReadTestRegister** (uint8_t reg, uint8_t *val) Read a test register within the ST25R500. More...
ReturnCode	**st25r500WriteTestRegister** (uint8_t reg, uint8_t val) Writes a given value to a test register within the ST25R500. More...
ReturnCode	**st25r500WriteMultipleTestRegister** (uint8_t reg, const uint8_t *values, uint8_t length) Writes a multiple values to test registers within the ST25R500. More...
ReturnCode	**st25r500ClrRegisterBits** (uint8_t reg, uint8_t clr_mask) Cleart bits on Register. More...
ReturnCode	**st25r500SetRegisterBits** (uint8_t reg, uint8_t set_mask) Set bits on Register. More...
ReturnCode	**st25r500ChangeRegisterBits** (uint8_t reg, uint8_t valueMask, uint8_t value) Changes the given bits on a ST25R500 register. More...
ReturnCode	**st25r500ModifyRegister** (uint8_t reg, uint8_t clr_mask, uint8_t set_mask) Modifies a value within a ST25R500 register. More...
ReturnCode	**st25r500ChangeTestRegisterBits** (uint8_t reg, uint8_t valueMask, uint8_t value) Changes the given bits on a ST25R500 Test register. More...
bool	**st25r500CheckReg** (uint8_t reg, uint8_t mask, uint8_t val) Checks if register contains a expected value. More...
bool	**st25r500IsRegValid** (uint8_t reg) Check if register ID is valid. More...

void	**st25r500ledInit** (void) ST25R500 LED Initialize. More...
void	**st25r500ledEvtIrq** (uint32_t irqs) ST25R500 LED Event Interrupt. More...
void	**st25r500ledEvtWrReg** (uint8_t reg, uint8_t val) ST25R500 LED Event Write Register. More...
void	**st25r500ledEvtWrMultiReg** (uint8_t reg, const uint8_t *vals, uint8_t len) ST25R500 LED Event Write Multiple Register. More...
void	**st25r500ledEvtCmd** (uint8_t cmd) ST25R500 LED Event Direct Command. More...
void	**st25r500ledEvtCmds** (const uint8_t *cmds, uint8_t len) ST25R200 LED Event Direct Commands. More...

Detailed Description

Macro Definition Documentation

◆ ST25R500_CEM_A_LEN

```
#define ST25R500_CEM_A_LEN   31U
```

Passive target memory A config length

◆ ST25R500_CEM_B_LEN

```
#define ST25R500_CEM_B_LEN   0U
```

Passive target memory B config length

◆ ST25R500_CEM_F_LEN

```
#define ST25R500_CEM_F_LEN   21U
```

Passive target memory F config length

◆ ST25R500_CEM_LEN

```
#define ST25R500_CEM_LEN   (ST25R500_CEM_A_LEN + ST25R500_CEM_B_LEN + ST25R500_CEM_F_LEN)
```

Full CE memory length

◆ ST25R500_FIFO_STATUS_LEN

```
#define ST25R500_FIFO_STATUS_LEN   2
```

Number of FIFO Status Register

◆ ST25R500_OBS_MODE_LEN

```
#define ST25R500_OBS_MODE_LEN   2
```

Number of Observation Mode Config Registers

◆ ST25R500_REG_ANA_DISPLAY1

```
#define ST25R500_REG_ANA_DISPLAY1   0x48U
```

RO Analog Display Register 1

◆ ST25R500_REG_ANA_DISPLAY2

```
#define ST25R500_REG_ANA_DISPLAY2   0x49U
```

RO Analog Display Register 2

◆ ST25R500_REG_AWS_CONFIG1

```
#define ST25R500_REG_AWS_CONFIG1   0x4DU
```

RW AWS Config Register 1

◆ ST25R500_REG_AWS_CONFIG2

```
#define ST25R500_REG_AWS_CONFIG2   0x4EU
```

RW AWS Config Register 2

◆ ST25R500_REG_AWS_TIME1

```
#define ST25R500_REG_AWS_TIME1   0x4FU
```

RW AWS Time Register 1

◆ ST25R500_REG_AWS_TIME2

```
#define ST25R500_REG_AWS_TIME2   0x50U
```

RW AWS Time Register 2

◆ ST25R500_REG_AWS_TIME3

```
#define ST25R500_REG_AWS_TIME3   0x51U
```

RW AWS Time Register 3

◆ ST25R500_REG_AWS_TIME4

```
#define ST25R500_REG_AWS_TIME4   0x52U
```

RW AWS Time Register 4

◆ ST25R500_REG_CE_CONFIG1

```
#define ST25R500_REG_CE_CONFIG1   0x1CU
```

RW CE Config Register 1

◆ ST25R500_REG_CE_CONFIG2

```
#define ST25R500_REG_CE_CONFIG2   0x1DU
```

RW CE Config Register 2

◆ ST25R500_REG_CE_CONFIG3

```
#define ST25R500_REG_CE_CONFIG3   0x1EU
```

RW CE Config Register 3

◆ ST25R500_REG_CE_STATUS1

```
#define ST25R500_REG_CE_STATUS1   0x45U
```

RO CE Status Register 1

◆ ST25R500_REG_CE_STATUS2

```
#define ST25R500_REG_CE_STATUS2   0x46U
```

RO CE Status Register 2

◆ ST25R500_REG_CE_TX_MOD1

```
#define ST25R500_REG_CE_TX_MOD1   0x06U
```

RW CE TX Modulation Register 1

◆ ST25R500_REG_CE_TX_MOD2

```
#define ST25R500_REG_CE_TX_MOD2   0x07U
```

RW CE TX Modulation Register 2

◆ ST25R500_REG_CEM_A

```
#define ST25R500_REG_CEM_A   0x58U
```

RW CE Memory A

◆ ST25R500_REG_CEM_F

```
#define ST25R500_REG_CEM_F   0x5AU
```

RW CE Memory F Register

◆ ST25R500_REG_COLLISION

```
#define ST25R500_REG_COLLISION   0x38U
```

RO Collision Register 2

◆ ST25R500_REG_CORR1

```
#define ST25R500_REG_CORR1   0x0EU
```

RW Correlator Register 1

◆ ST25R500_REG_CORR2

```
#define ST25R500_REG_CORR2   0x0FU
```

RW Correlator Register 2

◆ ST25R500_REG_CORR3

```
#define ST25R500_REG_CORR3   0x10U
```

RW Correlator Register 3

◆ ST25R500_REG_CORR4

```
#define ST25R500_REG_CORR4   0x11U
```

RW Correlator Register 4

◆ ST25R500_REG_CORR5

```
#define ST25R500_REG_CORR5   0x12U
```

RW Correlator Register 5

◆ ST25R500_REG_CORR6

```
#define ST25R500_REG_CORR6   0x13U
```

RW Correlator Register 6

◆ ST25R500_REG_DRIVER

```
#define ST25R500_REG_DRIVER   0x03U
```

RW TX Driver Register

◆ ST25R500_REG_EFD_THRESHOLD

```
#define ST25R500_REG_EFD_THRESHOLD   0x57U
```

RW EFD Thresholds Register

◆ ST25R500_REG_EMD1

```
#define ST25R500_REG_EMD1   0x1AU
```

RW Protocol EMD Register 1

◆ ST25R500_REG_EMD2

```
#define ST25R500_REG_EMD2   0x1BU
```

RW Protocol EMD Register 2

◆ ST25R500_REG_FIFO

```
#define ST25R500_REG_FIFO   0x5FU
```

RW FIFO address

◆ ST25R500_REG_FIFO_STATUS1

```
#define ST25R500_REG_FIFO_STATUS1   0x36U
```

RO FIFO Status Register 1

◆ ST25R500_REG_FIFO_STATUS2

```
#define ST25R500_REG_FIFO_STATUS2   0x37U
```

RO FIFO Status Register 2

◆ ST25R500_REG_GENERAL

```
#define ST25R500_REG_GENERAL   0x01U
```

RW General Register

◆ ST25R500_REG_GPIO

```
#define ST25R500_REG_GPIO   0x08U
```

RW GPIO control Register

◆ ST25R500_REG_GPT1

```
#define ST25R500_REG_GPT1   0x25U
```

RW General Purpose Timer (MSB) Register 1

◆ ST25R500_REG_GPT2

```
#define ST25R500_REG_GPT2   0x26U
```

RW General Purpose Timer (LSB) Register 2

◆ ST25R500_REG_IC_ID

```
#define ST25R500_REG_IC_ID   0x3FU
```

RO IC Identity Register

◆ ST25R500_REG_IRQ1

```
#define ST25R500_REG_IRQ1   0x3CU
```

RO IRQ Register 1

◆ ST25R500_REG_IRQ2

```
#define ST25R500_REG_IRQ2   0x3DU
```

RO IRQ Register 2

◆ ST25R500_REG_IRQ3

```
#define ST25R500_REG_IRQ3   0x3EU
```

RO IRQ Register 3

◆ ST25R500_REG_IRQ_MASK1

```
#define ST25R500_REG_IRQ_MASK1   0x39U
```

RO IRQ Mask Register 1

◆ ST25R500_REG_IRQ_MASK2

```
#define ST25R500_REG_IRQ_MASK2   0x3AU
```

RO IRQ Mask Register 2

◆ ST25R500_REG_IRQ_MASK3

```
#define ST25R500_REG_IRQ_MASK3   0x3BU
```

RO IRQ Mask Register 3

◆ ST25R500_REG_MRT1

```
#define ST25R500_REG_MRT1   0x1FU
```

RW Mask Receive Timer Configuration Register 1

◆ ST25R500_REG_MRT2

```
#define ST25R500_REG_MRT2   0x20U
```

RW Mask Receive Timer Configuration Register 2

◆ ST25R500_REG_NRT1

```
#define ST25R500_REG_NRT1   0x23U
```

RW No Response Timer (MSB) Register 1

◆ ST25R500_REG_NRT2

```
#define ST25R500_REG_NRT2   0x24U
```

RW No Response Timer (LSB) Register 2

◆ ST25R500_REG_NRT_GPT_CONF

```
#define ST25R500_REG_NRT_GPT_CONF   0x22U
```

RW No Response and Genereal Purpose Timer Configuration Register

◆ ST25R500_REG_OPERATION

```
#define ST25R500_REG_OPERATION   0x00U
```

RW Operation Register

◆ ST25R500_REG_OVERSHOOT_CONF

```
#define ST25R500_REG_OVERSHOOT_CONF   0x53U
```

RW Overshoot Pattern Register

◆ ST25R500_REG_PROTOCOL

```
#define ST25R500_REG_PROTOCOL   0x14U
```

RW Protocol Register

◆ ST25R500_REG_PROTOCOL_RX1

```
#define ST25R500_REG_PROTOCOL_RX1   0x17U
```

RW Protocol RX Register 1

◆ ST25R500_REG_PROTOCOL_RX2

```
#define ST25R500_REG_PROTOCOL_RX2   0x18U
```

RW Protocol RX Register 2

◆ ST25R500_REG_PROTOCOL_RX3

```
#define ST25R500_REG_PROTOCOL_RX3   0x19U
```

RW Protocol RX Register 3

◆ ST25R500_REG_PROTOCOL_TX1

```
#define ST25R500_REG_PROTOCOL_TX1   0x15U
```

RW Protocol TX Register 1

◆ ST25R500_REG_PROTOCOL_TX2

```
#define ST25R500_REG_PROTOCOL_TX2   0x16U
```

RW Protocol TX Register 2

◆ ST25R500_REG_REGULATOR

```
#define ST25R500_REG_REGULATOR   0x02U
```

RW Regulator Register

◆ ST25R500_REG_RSSI_I

```
#define ST25R500_REG_RSSI_I   0x4AU
```

RO I RSSI Display Register

◆ ST25R500_REG_RSSI_Q

```
#define ST25R500_REG_RSSI_Q   0x4BU
```

RO Q RSSI Display Register

◆ ST25R500_REG_RX_ANA1

```
#define ST25R500_REG_RX_ANA1   0x09U
```

RW RX Analog Register 1

◆ ST25R500_REG_RX_ANA2

```
#define ST25R500_REG_RX_ANA2   0x0AU
```

RW RX Analog Register 2

◆ ST25R500_REG_RX_ANA3

```
#define ST25R500_REG_RX_ANA3   0x0BU
```

RW RX Analog Register 3

◆ ST25R500_REG_RX_ANA4

```
#define ST25R500_REG_RX_ANA4   0x0CU
```

RW RX Analog Register 4

◆ ST25R500_REG_RX_DIG

```
#define ST25R500_REG_RX_DIG   0x0DU
```

RW RX Digital Register

◆ ST25R500_REG_SENSE_DISPLAY

```
#define ST25R500_REG_SENSE_DISPLAY   0x4CU
```

RO Display Register 3

◆ ST25R500_REG_SQT

```
#define ST25R500_REG_SQT   0x21U
```

RW Squelch Timer Register

◆ ST25R500_REG_STATUS1

```
#define ST25R500_REG_STATUS1   0x40U
```

RO Status Register 1

◆ ST25R500_REG_STATUS2

```
#define ST25R500_REG_STATUS2   0x41U
```

RO Status Register 2

◆ ST25R500_REG_STATUS_STATIC1

```
#define ST25R500_REG_STATUS_STATIC1   0x42U
```

RO Status Static Register 1

◆ ST25R500_REG_STATUS_STATIC2

```
#define ST25R500_REG_STATUS_STATIC2   0x43U
```

RO Status Static Register 2

◆ ST25R500_REG_STATUS_STATIC3

```
#define ST25R500_REG_STATUS_STATIC3   0x44U
```

RO Status Static Register 3

◆ ST25R500_REG_TX_FRAME1

```
#define ST25R500_REG_TX_FRAME1   0x34U
```

RW TX Frame Register 1

◆ ST25R500_REG_TX_FRAME2

```
#define ST25R500_REG_TX_FRAME2   0x35U
```

RW TX Frame Register 2

◆ ST25R500_REG_TX_MOD1

```
#define ST25R500_REG_TX_MOD1   0x04U
```

RW TX Modulation Register 1

◆ ST25R500_REG_TX_MOD2

```
#define ST25R500_REG_TX_MOD2   0x05U
```

RW TX Modulation Register 2

◆ ST25R500_REG_UNDERSHOOT_CONF

```
#define ST25R500_REG_UNDERSHOOT_CONF   0x54U
```

RW Undershoot Pattern Register

◆ ST25R500_REG_WAKEUP_CONF1

```
#define ST25R500_REG_WAKEUP_CONF1   0x27U
```

RW Wake-up COnfiguration Register 1

◆ ST25R500_REG_WAKEUP_CONF2

```
#define ST25R500_REG_WAKEUP_CONF2   0x28U
```

RW Wake-up COnfiguration Register 2

◆ ST25R500_REG_WAKEUP_CONF3

```
#define ST25R500_REG_WAKEUP_CONF3   0x29U
```

RW Wake-up COnfiguration Register 3

◆ ST25R500_REG_WU_I_ADC

```
#define ST25R500_REG_WU_I_ADC   0x2DU
```

RO Wake-up I-Channel ADC Display Register

◆ ST25R500_REG_WU_I_CAL

```
#define ST25R500_REG_WU_I_CAL   0x2CU
```

RO Wake-up I-Channel Calibration Display Register

◆ ST25R500_REG_WU_I_CONF

```
#define ST25R500_REG_WU_I_CONF   0x2AU
```

RW Wake-up I-Channel Configuration Register

◆ ST25R500_REG_WU_I_DELTA

```
#define ST25R500_REG_WU_I_DELTA   0x2BU
```

RW Wake-up I-Channel Delta Register

◆ ST25R500_REG_WU_I_REF

```
#define ST25R500_REG_WU_I_REF   0x2EU
```

RO Wake-up I-Channel Reference Display Register

◆ ST25R500_REG_WU_Q_ADC

```
#define ST25R500_REG_WU_Q_ADC   0x32U
```

RO Wake-up Q-Channel ADC Display Register

◆ ST25R500_REG_WU_Q_CAL

```
#define ST25R500_REG_WU_Q_CAL   0x31U
```

RO Wake-up Q-Channel Calibration Display Register

◆ ST25R500_REG_WU_Q_CONF

```
#define ST25R500_REG_WU_Q_CONF   0x2FU
```

RW Wake-up Q-Channel Configuration Register

◆ ST25R500_REG_WU_Q_DELTA

```
#define ST25R500_REG_WU_Q_DELTA   0x30U
```

RW Wake-up Q-Channel Delta Register

◆ ST25R500_REG_WU_Q_REF

```
#define ST25R500_REG_WU_Q_REF   0x33U
```

RO Wake-up Q-Channel Reference Display Register

◆ ST25R500_REG_WU_STATUS

```
#define ST25R500_REG_WU_STATUS   0x47U
```

RO WU Status Register

◆ ST25R500_TEST_REG_ADC2CE

```
#define ST25R500_TEST_REG_ADC2CE   0x08U
```

RW ADC to CE memory Register

◆ ST25R500_TEST_REG_DIAG_MEAS

```
#define ST25R500_TEST_REG_DIAG_MEAS   0x14U
```

RW Diagnostic Measurement Register

◆ ST25R500_TEST_REG_MAN_TIMING

```
#define ST25R500_TEST_REG_MAN_TIMING   0x13U
```

RW Manual timing Register

◆ ST25R500_TEST_REG_OSC_TIMING

```
#define ST25R500_TEST_REG_OSC_TIMING   0x0EU
```

RW Oscillator timing control Register

◆ ST25R500_TEST_REG_OVERLAP_CONTROL

```
#define ST25R500_TEST_REG_OVERLAP_CONTROL   0x1EU
```

RW Overlap control Register

◆ ST25R500_TEST_REG_RXE_DISP1

```
#define ST25R500_TEST_REG_RXE_DISP1   0x32U
```

RO Receive end timer display Register 1

◆ ST25R500_TEST_REG_RXE_DISP2

```
#define ST25R500_TEST_REG_RXE_DISP2   0x33U
```

RO Receive end timer display Register 2

◆ ST25R500_TEST_REG_RXS_DISP1

```
#define ST25R500_TEST_REG_RXS_DISP1   0x30U
```

RO Receive start timer display Register 1

◆ ST25R500_TEST_REG_RXS_DISP2

```
#define ST25R500_TEST_REG_RXS_DISP2   0x31U
```

RO Receive start timer display Register 2

Function Documentation

◆ st25r500ChangeRegisterBits()

```
ReturnCode st25r500ChangeRegisterBits ( uint8_t reg,
                                        uint8_t valueMask,
                                        uint8_t value
                                      )
```

This function is used if only a particular bits should be changed within an ST25R500 register.

Parameters

- [in] **reg** Address of the register to change.
- [in] **valueMask** bitmask of bits to be changed
- [in] **value** the bits to be written on the enabled valueMask bits

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r500ChangeTestRegisterBits()

```
ReturnCode st25r500ChangeTestRegisterBits ( uint8_t reg,
                                            uint8_t valueMask,
                                            uint8_t value
                                          )
```

This function is used if only a particular bits should be changed within an ST25R500 register.

Parameters

- [in] **reg** Address of the Test register to change.
- [in] **valueMask** bitmask of bits to be changed
- [in] **value** the bits to be written on the enabled valueMask bits

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r500CheckReg()

```
bool st25r500CheckReg ( uint8_t  reg,
                        uint8_t  mask,
                        uint8_t  val
                      )
```

This function checks if the given reg contains a value that once masked equals the expected value

Parameters

reg : the register to check the value

mask : the mask apply on register value

val : expected value to be compared to

Returns

true when reg contains the expected value | false otherwise

◆ st25r500ClrRegisterBits()

```
ReturnCode st25r500ClrRegisterBits ( uint8_t  reg,
                                     uint8_t  clr_mask
                                   )
```

This function clears the given bitmask on the register

Parameters

[in] **reg** Address of the register clear

[in] **clr_mask** Bitmask of bit to be cleared

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r500ExecuteCommand()

ReturnCode st25r500ExecuteCommand (uint8_t cmd)

This function is used to trigger a direct command. These commands are implemented inside the chip and each command has unique code

Parameters

[in] **cmd** : code of the direct command to be executed

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r500ExecuteCommands()

ReturnCode st25r500ExecuteCommands (const uint8_t * cmds,
uint8_t length
)

This function is used to trigger execution of direct commands. These commands are implemented inside the chip and each command has unique code (see datasheet).

Warning

Commands to be executed must support command chaining

Parameters

[in] **cmds** : code of the direct commands to be executed

[in] **length** : number of direct command to be executed

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r500IsRegValid()

bool st25r500IsRegValid (uint8_t reg)

Checks if the given register ID a valid ST25R500 register

Parameters

[in] **reg** Address of register to check

Returns

true if is a valid register ID

false otherwise

◆ st25r500ledEvtCmd()

```
void st25r500ledEvtCmd ( uint8_t  cmd )
```

This function should be called upon a ST25R500 direct command, providing the event with the command executed

Parameters

[in] **cmd** ST25R500 cmd executed

◆ st25r500ledEvtCmds()

```
void st25r500ledEvtCmds ( const uint8_t *  cmds,
                          uint8_t          len
                        )
```

This function should be called upon the execution of ST25R500 direct commands, providing the event with the commands executed

Parameters

[in] **cmds** ST25R500 cmds executed

[in] **len** : number of commands executed

◆ st25r500ledEvtIrq()

```
void st25r500ledEvtIrq ( uint32_t  irqs )
```

This function should be called upon a ST25R500 Interrupt, providing the interrupt event with ST25R500 irq flags to update LEDs

Parameters

[in] **irqs** ST25R500 irqs mask

◆ st25r500ledEvtWrMultiReg()

```
void st25r500ledEvtWrMultiReg ( uint8_t          reg,
                                const uint8_t *  vals,
                                uint8_t          len
                              )
```

This function should be called upon a ST25R500 Write Multiple Registers, providing the event with the registers and values to update LEDs

Parameters

[in] **reg** : ST25R500 first register written

[in] **vals** pointer to the values written

[in] **len** : number of registers written

◆ st25r500ledEvtWrReg()

```
void st25r500ledEvtWrReg ( uint8_t  reg,
                           uint8_t  val
                         )
```

This function should be called on a ST25R500 Write Register operation providing the event with the register and value to update LEDs

Parameters

[in] **reg** ST25R500 register to be written

[in] **val** value to be written on the register

◆ st25r500ledInit()

```
void st25r500ledInit ( void )
```

This function initializes the LEDs that represent ST25R500 activity

◆ st25r500ModifyRegister()

```
ReturnCode st25r500ModifyRegister ( uint8_t  reg,
                                    uint8_t  clr_mask,
                                    uint8_t  set_mask
                                  )
```

This function is used if only a particular bits should be changed within an ST25R500 register.

Parameters

- [in] **reg** Address of the register to write.
- [in] **clr_mask** bitmask of bits to be cleared to 0.
- [in] **set_mask** bitmask of bits to be set to 1.

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r500ReadFifo()

```
ReturnCode st25r500ReadFifo ( uint8_t *  buf,
                              uint16_t  length
                            )
```

This function needs to be called in order to read from ST25R500 FIFO.

Parameters

- [out] **buf** pointer to a buffer where the FIFO content shall be written to
- [in] **length** Number of bytes to read

Note

: This function doesn't check whether *length* is really the number of available bytes in FIFO

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r500ReadMultipleRegisters()

```
ReturnCode st25r500ReadMultipleRegisters ( uint8_t    reg,
                                           uint8_t *  values,
                                           uint16_t   length
                                          )
```

This function is used to read from multiple registers using the auto-increment feature. That is, after each read the address pointer inside the ST25R500 gets incremented automatically

Parameters

- [in] **reg** Address of the frist register to read from
- [in] **values** pointer to a buffer where the result shall be written to
- [in] **length** Number of registers to be read out

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r500ReadRegister()

```
ReturnCode st25r500ReadRegister ( uint8_t    reg,
                                  uint8_t *  val
                                 )
```

Register bit definitions
DOXYGEN_SUPPRESS

This function is used to read out the content of ST25R500 registers

Parameters

- [in] **reg** Address of register to read
- [out] **val** Returned value

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r500ReadTestRegister()

```
ReturnCode st25r500ReadTestRegister ( uint8_t    reg,
                                      uint8_t *  val
                                    )
```

This function is used to read the content of test address *reg* within the ST25R500

Parameters

[in] **reg** Address of the register to read

[out] **val** Returned read value

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r500SetRegisterBits()

```
ReturnCode st25r500SetRegisterBits ( uint8_t  reg,
                                     uint8_t  set_mask
                                   )
```

This function sets the given bitmask on the register

Parameters

[in] **reg** Address of the register clear

[in] **set_mask** Bitmask of bit to be cleared

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r500WriteFifo()

```
ReturnCode st25r500WriteFifo ( const uint8_t *  values,
                               uint16_t         length
```

```
)
```

This function needs to be called in order to write to the ST25R500 FIFO.

Parameters

[in] **values** pointer to a buffer containing the values to be written to the FIFO

[in] **length** Number of values to be written

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r500WriteMultipleRegisters()

```
ReturnCode st25r500WriteMultipleRegisters ( uint8_t          reg,
                                            const uint8_t *  values,
                                            uint16_t         length
                                          )
```

This function is used to write multiple values to the ST25R500 using the auto-increment feature. That is, after each write the address pointer inside the ST25R500 gets incremented automatically

Parameters

[in] **reg** : Address of the frist register to write

[in] **values** pointer to a buffer containing the values to be written

[in] **length** Number of values to be written

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r500WriteMultipleTestRegister()

```
ReturnCode st25r500WriteMultipleTestRegister ( uint8_t          reg,
                                               const uint8_t *  values,
                                               uint8_t          length
```

```
)
```

This function is used to write *val* to test address *reg* within the ST25R500

Parameters

[in] **reg** : Address of the frist register to write

[in] **values** Pointer to a buffer containing the values to be written

[in] **length** Number of values to be written

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r500WriteRegister()

```
ReturnCode st25r500WriteRegister ( uint8_t reg,
                                   uint8_t val
                                 )
```

This function is used to write *val* to address *reg* within the ST25R500.

Parameters

[in] **reg** Address of the register to write

[in] **val** Value to be written

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

◆ st25r500WriteTestRegister()

```
ReturnCode st25r500WriteTestRegister ( uint8_t reg,
                                       uint8_t val
                                     )
```

This function is used to write *val* to test address *reg* within the ST25R500

Parameters

[in] **reg** Address of the register to write

[in] **val** Value to be written

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_SEND : Transmission error or acknowledge not received

---

ST25R500_IRQ

RFAL ST25R500 IRQ. More...

Collaboration diagram for ST25R500_IRQ:

Macros

#define **ST25R500_IRQ_MASK_ALL** (uint32_t)(0xFFFFFFFFUL)

#define **ST25R500_IRQ_MASK_NONE** (uint32_t)(0x00000000UL)

#define **ST25R500_IRQ_MASK_SUBC_START** (uint32_t)(0x00000080U)

#define **ST25R500_IRQ_MASK_COL** (uint32_t)(0x00000040U)

#define **ST25R500_IRQ_MASK_WL** (uint32_t)(0x00000020U)

#define **ST25R500_IRQ_MASK_RX_REST** (uint32_t)(0x00000010U)

#define **ST25R500_IRQ_MASK_RXE** (uint32_t)(0x00000008U)

#define **ST25R500_IRQ_MASK_RXS** (uint32_t)(0x00000004U)

#define **ST25R500_IRQ_MASK_TXE** (uint32_t)(0x00000002U)

#define **ST25R500_IRQ_MASK_RX_ERR** (uint32_t)(0x00000001U)

#define **ST25R500_IRQ_MASK_GPE** (uint32_t)(0x00008000U)

#define **ST25R500_IRQ_MASK_NRE** (uint32_t)(0x00004000U)

#define **ST25R500_IRQ_MASK_WPT_STOP** (uint32_t)(0x00002000U)

#define **ST25R500_IRQ_MASK_WPT_FOD** (uint32_t)(0x00001000U)

#define **ST25R500_IRQ_MASK_RFU** (uint32_t)(0x00000800U)

#define **ST25R500_IRQ_MASK_CE_SC** (uint32_t)(0x00000400U)

#define **ST25R500_IRQ_MASK_RXE_CE** (uint32_t)(0x00000200U)

#define **ST25R500_IRQ_MASK_NFCT** (uint32_t)(0x00000100U)

#define **ST25R500_IRQ_MASK_WUTME** (uint32_t)(0x00800000U)

#define **ST25R500_IRQ_MASK_EOF** (uint32_t)(0x00400000U)

#define **ST25R500_IRQ_MASK_EON** (uint32_t)(0x00200000U)

#define **ST25R500_IRQ_MASK_DCT** (uint32_t)(0x00100000U)

#define **ST25R500_IRQ_MASK_WUQ** (uint32_t)(0x00080000U)

#define **ST25R500_IRQ_MASK_WUI** (uint32_t)(0x00040000U)

#define **ST25R500_IRQ_MASK_WUT** (uint32_t)(0x00020000U)

#define **ST25R500_IRQ_MASK_OSC** (uint32_t)(0x00010000U)

Functions

uint32_t **st25r500WaitForInterruptsTimed** (uint32_t mask, uint16_t tmo)
Wait until an ST25R500 interrupt occurs. More...

uint32_t **st25r500GetInterrupt** (uint32_t mask)
Get status for the given interrupt. More...

void **st25r500InitInterrupts** (void)
Init the 200 interrupt. More...

void **st25r500ModifyInterrupts** (uint32_t clr_mask, uint32_t set_mask)
Modifies the Interrupt. More...

void **st25r500CheckForReceivedInterrupts** (void)
Checks received interrupts. More...

void **st25r500Isr** (void)
ISR Service routine. More...

void **st25r500EnableInterrupts** (uint32_t mask)
Enable a given ST25R500 Interrupt source. More...

void **st25r500DisableInterrupts** (uint32_t mask)
Disable one or more a given ST25R500 Interrupt sources. More...

void **st25r500ClearInterrupts** (void)
Clear all ST25R500 irq flags. More...

void **st25r500ClearAndEnableInterrupts** (uint32_t mask)
Clears and then enables the given ST25R500 Interrupt sources. More...

void **st25r500IRQCallbackSet** (void(*cb)(void))
Sets IRQ callback for the ST25R500 interrupt. More...

void **st25r500IRQCallbackRestore** (void)

Sets IRQ callback for the ST25R500 interrupt. More...

Detailed Description

Macro Definition Documentation

◆ ST25R500_IRQ_MASK_ALL

```
#define ST25R500_IRQ_MASK_ALL   (uint32_t)(0xFFFFFFFFUL)
```

All ST25R500 interrupt sources

◆ ST25R500_IRQ_MASK_CE_SC

```
#define ST25R500_IRQ_MASK_CE_SC   (uint32_t)(0x00000400U)
```

ST25R500 CE state change interrupt

◆ ST25R500_IRQ_MASK_COL

```
#define ST25R500_IRQ_MASK_COL   (uint32_t)(0x00000040U)
```

ST25R500 bit collision interrupt

◆ ST25R500_IRQ_MASK_DCT

```
#define ST25R500_IRQ_MASK_DCT   (uint32_t)(0x00100000U)
```

ST25R500 termination of direct command interrupt

◆ ST25R500_IRQ_MASK_EOF

```
#define ST25R500_IRQ_MASK_EOF   (uint32_t)(0x00400000U)
```

ST25R500 External Field Off interrupt

◆ ST25R500_IRQ_MASK_EON

```
#define ST25R500_IRQ_MASK_EON   (uint32_t)(0x00200000U)
```

ST25R500 External Field On interrupt

◆ ST25R500_IRQ_MASK_GPE

```
#define ST25R500_IRQ_MASK_GPE   (uint32_t)(0x00008000U)
```

ST25R500 general purpose timer expired interrupt

◆ ST25R500_IRQ_MASK_NFCT

```
#define ST25R500_IRQ_MASK_NFCT   (uint32_t)(0x00000100U)
```

ST25R500 bit rate was recognized interrupt

◆ ST25R500_IRQ_MASK_NONE

```
#define ST25R500_IRQ_MASK_NONE   (uint32_t)(0x00000000UL)
```

No ST25R500 interrupt source

◆ ST25R500_IRQ_MASK_NRE

```
#define ST25R500_IRQ_MASK_NRE   (uint32_t)(0x00004000U)
```

ST25R500 no-response timer expired interrupt

◆ ST25R500_IRQ_MASK_OSC

```
#define ST25R500_IRQ_MASK_OSC   (uint32_t)(0x00010000U)
```

ST25R500 oscillator stable interrupt

◆ ST25R500_IRQ_MASK_RFU

```
#define ST25R500_IRQ_MASK_RFU   (uint32_t)(0x00000800U)
```

ST25R500 RFU

◆ ST25R500_IRQ_MASK_RX_ERR

```
#define ST25R500_IRQ_MASK_RX_ERR   (uint32_t)(0x00000001U)
```

ST25R500 Reception error interrupt

◆ ST25R500_IRQ_MASK_RX_REST

```
#define ST25R500_IRQ_MASK_RX_REST   (uint32_t)(0x00000010U)
```

ST25R500 automatic reception restart interrupt

◆ ST25R500_IRQ_MASK_RXE

```
#define ST25R500_IRQ_MASK_RXE   (uint32_t)(0x00000008U)
```

ST25R500 end of receive interrupt

◆ ST25R500_IRQ_MASK_RXE_CE

```
#define ST25R500_IRQ_MASK_RXE_CE   (uint32_t)(0x00000200U)
```

ST25R500 end of reception in CE interrupt

◆ ST25R500_IRQ_MASK_RXS

```
#define ST25R500_IRQ_MASK_RXS   (uint32_t)(0x00000004U)
```

ST25R500 start of receive interrupt

◆ ST25R500_IRQ_MASK_SUBC_START

```
#define ST25R500_IRQ_MASK_SUBC_START   (uint32_t)(0x00000080U)
```

ST25R500 subcarrier start interrupt

◆ ST25R500_IRQ_MASK_TXE

```
#define ST25R500_IRQ_MASK_TXE   (uint32_t)(0x00000002U)
```

ST25R500 end of transmission interrupt

◆ ST25R500_IRQ_MASK_WL

```
#define ST25R500_IRQ_MASK_WL   (uint32_t)(0x00000020U)
```

ST25R500 FIFO water level interrupt

◆ ST25R500_IRQ_MASK_WPT_FOD

```
#define ST25R500_IRQ_MASK_WPT_FOD   (uint32_t)(0x00001000U)
```

ST25R500 WPT FOD received interrupt

◆ ST25R500_IRQ_MASK_WPT_STOP

```
#define ST25R500_IRQ_MASK_WPT_STOP   (uint32_t)(0x00002000U)
```

ST25R500 WPT stop received interrupt

◆ ST25R500_IRQ_MASK_WUI

```
#define ST25R500_IRQ_MASK_WUI   (uint32_t)(0x00040000U)
```

ST25R500 wake-up I-Channel interrupt

◆ ST25R500_IRQ_MASK_WUQ

```
#define ST25R500_IRQ_MASK_WUQ   (uint32_t)(0x00080000U)
```

ST25R500 wake-up Q-Channel interrupt

◆ ST25R500_IRQ_MASK_WUT

```
#define ST25R500_IRQ_MASK_WUT   (uint32_t)(0x00020000U)
```

ST25R500 wake-up timer interrupt

◆ ST25R500_IRQ_MASK_WUTME

```
#define ST25R500_IRQ_MASK_WUTME   (uint32_t)(0x00800000U)
```

ST25R500 WU Measure event interrupt

Function Documentation

◆ st25r500CheckForReceivedInterrupts()

```
void st25r500CheckForReceivedInterrupts ( void )
```

Checks received interrupts and saves the result into global params

◆ st25r500ClearAndEnableInterrupts()

```
void st25r500ClearAndEnableInterrupts ( uint32_t mask )
```

Parameters

[in] **mask** mask indicating the interrupts to be cleared and enabled

◆ st25r500ClearInterrupts()

```
void st25r500ClearInterrupts ( void )
```

◆ st25r500DisableInterrupts()

void st25r500DisableInterrupts (uint32_t mask)

This function disables all interrupts given by *mask*. 0xff disables all.

Parameters

[in] **mask** mask indicating the interrupts to be disabled.

◆ st25r500EnableInterrupts()

void st25r500EnableInterrupts (uint32_t mask)

This function enables all interrupts given by *mask*, ST25R500_IRQ_MASK_ALL enables all interrupts.

Parameters

[in] **mask** mask indicating the interrupts to be enabled

◆ st25r500GetInterrupt()

uint32_t st25r500GetInterrupt (uint32_t mask)

This function is used to check whether the interrupt given by *mask* has occured. If yes the interrupt gets cleared. This function returns only status bits which are inside *mask*.

Parameters

[in] **mask** : mask indicating the interrupt to check for.

Returns

the mask of the interrupts occurred

◆ st25r500InitInterrupts()

void st25r500InitInterrupts (void)

This function is used to check whether the interrupt given by *mask* has occured.

◆ st25r500IRQCallbackRestore()

void st25r500IRQCallbackRestore (void)

◆ st25r500IRQCallbackSet()

void st25r500IRQCallbackSet (void(*)(void) cb)

Parameters

[in] **cb** pointer to the callback method

◆ st25r500Isr()

void st25r500Isr (void)

This function modiefies the interupt

◆ st25r500ModifyInterrupts()

void st25r500ModifyInterrupts (uint32_t clr_mask,
uint32_t set_mask
)

This function modifies the interrupt

Parameters

[in] **clr_mask** : bit mask to be cleared on the interrupt mask

[in] **set_mask** : bit mask to be set on the interrupt mask

◆ st25r500WaitForInterruptsTimed()

uint32_t st25r500WaitForInterruptsTimed (uint32_t mask,
uint16_t tmo
)

This function is used to access the ST25R500 interrupt flags. Use this to wait for max. *tmo* milliseconds for the **first** interrupt indicated with mask *mask* to occur.

Parameters

[in] **mask** : mask indicating the interrupts to wait for.

[in] **tmo** : time in milliseconds until timeout occurs. If set to 0 the functions waits forever.

Returns

: 0 if timeout occured otherwise a mask indicating the cleared interrupts.

RFAL

ST25R95

RFAL ST25R95 Driver. More...

Collaboration diagram for ST25R95:

Modules

ST25R95_Driver
RFAL ST25R95 Driver.

ST25R95_Com
RFAL ST25R95 Communication.

Detailed Description

ST25R95_Driver

RFAL ST25R95 Driver. More...

Collaboration diagram for ST25R95_Driver:

Macros

#define **st25r95_nIRQ_IN_Pulse**() **st25r95SPI_nIRQ_IN_Pulse**()

#define **st25r95ResetChip**() **st25r95SPIResetChip**()

Functions

ReturnCode **st25r95Initialize** (void)
Initialise ST25R95 driver. More...

void **st25r95Deinitialize** (void)
Deinitialize ST25R95 driver. More...

void	**st25r95SPI_nIRQ_IN_Pulse** (void) Generates an IRQ_IN pulse (SPI intefrace) More...
void	**st25r95UART_nIRQ_IN_Pulse** (void) Generates an IRQ_IN pulse (UART interface) More...
bool	**st25r95CheckChipID** (void) Check Identity. More...
ReturnCode	**st25r95FieldOn** (uint32_t protocol) Turns field on. More...
ReturnCode	**st25r95FieldOff** (void) Turns field off. More...

Detailed Description

Macro Definition Documentation

◆ st25r95_nIRQ_IN_Pulse

#define st25r95_nIRQ_IN_Pulse ( ) **st25r95SPI_nIRQ_IN_Pulse**()

UART/SPI wrapper for st25r95_nIRQ_IN_Pulse

◆ st25r95ResetChip

#define st25r95ResetChip ( ) **st25r95SPIResetChip**()

UART/SPI wrapper for st25r95ResetChip

Function Documentation

◆ st25r95CheckChipID()

bool st25r95CheckChipID (void)

Checks if the chip ID is as expected.

Returns

true when IC type is as expected

◆ st25r95Deinitialize()

void st25r95Deinitialize (void)

Calling this function deinitializes the ST25R95 driver.

◆ st25r95FieldOff()

ReturnCode st25r95FieldOff (void)

This function turns field Off

Returns

RFAL_ERR_NONE : Operation successful

◆ st25r95FieldOn()

ReturnCode st25r95FieldOn (uint32_t protocol)

This function turns field On

Parameters

`[in]` **protocol** ST25R95_PROTOCOL_xxx protocol value.

Returns

RFAL_ERR_NONE : Operation successful

◆ st25r95Initialize()

ReturnCode st25r95Initialize (void)

This function initialises the ST25R95 driver.

Returns

RFAL_ERR_NONE : Operation successful

Other : Init failed

◆ st25r95SPI_nIRQ_IN_Pulse()

void st25r95SPI_nIRQ_IN_Pulse (void)

Generates an IRQ_IN pulse (see CR95HF DS ⍰3.2).

◆ st25r95UART_nIRQ_IN_Pulse()

void st25r95UART_nIRQ_IN_Pulse (void)

Generates an IRQ_IN pulse (see CR95HF DS ⍰3.2).

ST25R95_Com

RFAL » RFAL-HAL » ST25R95

RFAL ST25R95 Communication. More...

Collaboration diagram for ST25R95_Com:

Data Structures

struct **st25r95SPIRxContext**

Macros

#define **ST25R95_CONTROL_SEND** 0x00

#define **ST25R95_CONTROL_RESET** 0x01

#define **ST25R95_CONTROL_READ** 0x02

#define **ST25R95_CONTROL_POLL** 0x03

#define **ST25R95_CONTROL_POLL_TIMEOUT** 100

#define **ST25R95_CONTROL_POLL_NO_TIMEOUT** 0

#define **ST25R95_COMMAND_IDN** 0x01

#define **ST25R95_COMMAND_PROTOCOLSELECT** 0x02

#define **ST25R95_COMMAND_POLLFIED** 0x03

#define **ST25R95_COMMAND_SENDRECV** 0x04

#define **ST25R95_COMMAND_LISTEN** 0x05

#define **ST25R95_COMMAND_SEND** 0x06

#define **ST25R95_COMMAND_IDLE** 0x07

#define **ST25R95_COMMAND_RDREG** 0x08

#define **ST25R95_COMMAND_WRREG** 0x09

#define **ST25R95_COMMAND_BAUDRATE** 0x0A

```
#define ST25R95_COMMAND_ACFILTER 0x0D
#define ST25R95_COMMAND_ECHO 0x55
#define ST25R95_SPI_DUMMY_BYTE 0x00
#define ST25R95_CMD_COMMAND_OFFSET 0x00U
#define ST25R95_CMD_RESULT_OFFSET 0x00U
#define ST25R95_CMD_LENGTH_OFFSET 0x01U
#define ST25R95_CMD_DATA_OFFSET 0x02U
#define ST25R95_COMMUNICATION_BUFFER_SIZE (528 + 2)
#define ST25R95_COMMUNICATION_UART_WDOGTIMER (5000U)
#define ST25R95_PROTOCOL_FIELDOFF 0x00U
#define ST25R95_PROTOCOL_ISO15693 0x01U
#define ST25R95_PROTOCOL_ISO14443A 0x02U
#define ST25R95_PROTOCOL_ISO14443B 0x03U
#define ST25R95_PROTOCOL_ISO18092 0x04U
#define ST25R95_PROTOCOL_CE_ISO14443A 0x05U
#define ST25R95_PROTOCOL_MAX ST25R95_PROTOCOL_CE_ISO14443A
#define ST25R95_FWT_MAX 0x40A8BC0
#define ST25R95_ERRCODE_NONE 0x00
#define ST25R95_ERRCODE_FRAMEOKADDITIONALINFO 0x80
#define ST25R95_ERRCODE_INVALIDCCMDCODE 0x81
#define ST25R95_ERRCODE_INVALIDCMDLENGHT 0x82
```

#define **ST25R95_ERRCODE_INVALIDPROTOCOL** 0x83

#define **ST25R95_ERRCODE_COMERROR** 0x86

#define **ST25R95_ERRCODE_FRAMEWAITTIMEOUT** 0x87

#define **ST25R95_ERRCODE_INVALIDSOF** 0x88

#define **ST25R95_ERRCODE_OVERFLOW** 0x89

#define **ST25R95_ERRCODE_FRAMING** 0x8A

#define **ST25R95_ERRCODE_EGT** 0x8B

#define **ST25R95_ERRCODE_FIELDLENGTH** 0x8C

#define **ST25R95_ERRCODE_CRC** 0x8D

#define **ST25R95_ERRCODE_RECEPTIONLOST** 0x8E

#define **ST25R95_ERRCODE_NOFIELD** 0x8F

#define **ST25R95_ERRCODE_RESULTSRESIDUAL** 0x90

#define **ST25R95_ERRCODE_61_SOF** 0x61

#define **ST25R95_ERRCODE_62_CRC** 0x62

#define **ST25R95_ERRCODE_63_SOF_HIGH** 0x63

#define **ST25R95_ERRCODE_65_SOF_LOW** 0x65

#define **ST25R95_ERRCODE_66_EGT** 0x66

#define **ST25R95_ERRCODE_67_TR1TOOLONG** 0x67

#define **ST25R95_ERRCODE_68_TR1TOOSHORT** 0x68

#define **ST25R95_REG_ARC_B** 0x6801

#define **ST25R95_REG_ACC_A** 0x6804

```
#define ST25R95_REG_TIMERW 0x3A00
#define ST25R95_IDN_RESPONSE_BUFLEN (15 + 2)
#define ST25R95_PROTOCOLSELECT_RESPONSE_BUFLEN (0 + 2)
#define ST25R95_POLLFIELD_RESPONSE_BUFLEN (1 + 2)
#define ST25R95_LISTEN_RESPONSE_BUFLEN (0 + 2)
#define ST25R95_RDREG_RESPONSE_BUFLEN (1 + 2)
#define ST25R95_WRREG_RESPONSE_BUFLEN (0 + 2)
#define ST25R95_ACFILTER_RESPONSE_BUFLEN (1 + 2)
#define ST25R95_IDLE_RESPONSE_BUFLEN (1 + 2)
#define ST25R95_ECHO_RESPONSE_BUFLEN (3)
#define ST25R95_SEND_RESPONSE_BUFLEN (0 + 2)
#define ST25R95_ACSTATE_IDLE 0x00U
#define ST25R95_ACSTATE_READYA 0x01U
#define ST25R95_ACSTATE_ACTIVE 0x04U
#define ST25R95_ACSTATE_HALT 0x80U
#define ST25R95_ACSTATE_READYAX 0x81U
#define ST25R95_ACSTATE_ACTIVEX 0x84U
#define ST25R95_IS_PROT_ISO15693_CRC_ERR(status) (((status) & 0x02U) == 0x02U)
#define ST25R95_IS_PROT_ISO15693_COLLISION_ERR(status) (((status) & 0x01U) == 0x01U)
#define ST25R95_IS_PROT_ISO14443A_COLLISION_ERR(status) (((status) & 0x80U) == 0x80U)
#define ST25R95_IS_PROT_ISO14443A_CRC_ERR(status) (((status) & 0x20U) == 0x20U)
```

#define **ST25R95_IS_PROT_ISO14443A_PARITY_ERR**(status) (((status) & 0x10U) == 0x10U)

#define **ST25R95_IS_PROT_ISO14443B_CRC_ERR**(status) (((status) & 0x02U) == 0x02U)

#define **ST25R95_IS_PROT_ISO18092_CRC_ERR**(status) (((status) & 0x02U) == 0x02U)

#define **ST25R95_PROTOCOLSELECT_BR_OFFSET** (3U)

#define **ST25R95_IDLE_WUPERIOD_OFFSET** (0x09U)

#define **ST25R95_IDLE_DACDATAL_OFFSET** (0x0CU)

#define **ST25R95_IDLE_DACDATAH_OFFSET** (0x0DU)

#define **ST25R95_DACDATA_MAX** (0xFCU)

#define **ST25R95_IDLE_WKUP_TIMEOUT** (0x01U)

#define **ST25R95_IDLE_WKUP_TAGDETECT** (0x02U)

#define **st25r95SendCommandTypeAndLen**(cmd, resp, respBuffLen) **st25r95SPISendCommandTypeAndLen**((cmd), (resp), (respBuffLen))

#define **st25r95CommandEcho**() **st25r95SPICommandEcho**()

#define **st25r95SendData**(buf, bufLen, protocol, flags) **st25r95SPISendData**((buf), (bufLen), (protocol), (flags))

#define **st25r95SendTransmitFlag**(protocol, transmitFlag) **st25r95SPISendTransmitFlag**((protocol), (transmitFlag))

#define **st25r95PollRead**(timeout) **st25r95SPIPollRead**((timeout))

#define **st25r95PrepareRx**(protocol, rxBuf, rxBufLen, rxRcvdLen, flags, additionalRespBytes) **st25r95SPIPrepareRx**((protocol), (rxBuf), (rxBufLen), (rxRcvdLen), (flags), (additionalRespBytes));

#define **st25r95CompleteRx**() **st25r95SPICompleteRx**();

#define **st25r95IsTransmitTxCompleted**() **st25r95SPIIsTransmitCompleted**()

#define **st25r95IsInListen**() **st25r95SPIIsInListen**()

#define **st25r95GetLmState**() **st25r95SPIGetLmState**()

#define **st25r95DeactivateACFilter**() **st25r95SPIDeactivateACFilter**()

#define **st25r95SetACState**(State) **st25r95SPISetACState**(State)

#define **st25r95Idle**(DacDataL, DacDataH, WUPeriod) **st25r95SPIIdle**((DacDataL), (DacDataH), (WUPeriod))

#define **st25r95GetIdleResponse**() **st25r95SPIGetIdleResponse**()

#define **st25r95KillIdle**() **st25r95SPIKillIdle**()

#define **st25r95FlushChipSPIBuffer**() **st25r95SPIRxTx**(**NULL**, **NULL**, **ST25R95_COMMUNICATION_BUFFER_SIZE**)

Functions

void **st25r95SPIResetChip** (void)
Resets the ST25R95 (SPI interface) More...

ReturnCode **st25r95SPISendCommandTypeAndLen** (uint8_t *cmd, uint8_t *resp, uint16_t respBuffLen)
Sends a command to ST25R95 and reads the response (SPI Interface) More...

ReturnCode **st25r95UARTSendCommandTypeAndLen** (uint8_t *cmd, uint8_t *resp, uint16_t respBuffLen)
Sends a command to ST25R95 and reads the response (UART Interface) More...

ReturnCode **st25r95SPICommandEcho** (void)
Sends a echo command to ST25R95 (SPI Interface) More...

ReturnCode **st25r95UARTCommandEcho** (void)
Sends a echo command to ST25R95 (UART Interface) More...

uint8_t **st25r95SPISendReceiveByte** (uint8_t data)
Sends one byte over SPI and returns the byte received. More...

ReturnCode **st25r95ProtocolSelect** (uint8_t protocol)
Sends ProtocolSelect command. More...

ReturnCode **st25r95SetBitRate** (uint8_t protocol, **rfalBitRate** txBR, **rfalBitRate** rxBR)
Set bit rates. More...

void	**st25r95SPISendData** (uint8_t *buf, uint8_t bufLen, uint8_t protocol, uint32_t flags) Sends data (SPI interface) More...
void	**st25r95UARTSendData** (uint8_t *buf, uint8_t bufLen, uint8_t protocol, uint32_t flags) Sends data (UART interface) More...
void	**st25r95SPIPrepareRx** (uint8_t protocol, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rxRcvdLen, uint32_t flags, uint8_t *additionalRespBytes) Prepares Rx (SPI Interface) More...
void	**st25r95UARTPrepareRx** (uint8_t protocol, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rxRcvdLen, uint32_t flags, uint8_t *additionalRespBytes) Prepares Rx (UART Interface) More...
ReturnCode	**st25r95SPIPollRead** (uint32_t timeout) Polls the ST25R95 for incomming data (SPI Interface) More...
ReturnCode	**st25r95UARTPollRead** (uint32_t timeout) Polls the ST25R95 for incomming data (UART Interface) More...
ReturnCode	**st25r95SPIPollSend** (void) Polls the ST25R95 for sending (SPI Interface) More...
ReturnCode	**st25r95SetFWT** (uint8_t protocol, uint32_t fwt) set FWT More...
ReturnCode	**st25r95SetSlotCounter** (uint8_t slots) set slot counter More...
ReturnCode	**st25r95WriteReg** (uint8_t protocol, uint16_t reg, uint8_t value) Write Register. More...
ReturnCode	**st25r95ReadReg** (uint16_t reg, uint8_t *value) Read Register. More...
bool	**st25r95PollField** (void) Poll field (CE mode) More...
ReturnCode	**st25r95SetACFilter** (const **rfalLmConfPA** *confA)

Set Anti-Collision filter (CE mode) More...

ReturnCode	**st25r95Listen** (void) Listen (CE mode) More...
void	**st25r95UartTxCpltCallback** (void) Uart Tx Callback (UART mode) More...
void	**st25r95UartRxCpltCallback** (void) Uart Rx Callback (UART mode) More...
void	**st25r95UartErrorCallback** (void) Uart Error Callback (UART mode) More...
bool	**st25r95SPIIsTransmitCompleted** (void) End of transmit test function (SPI mode) More...
bool	**st25r95UARTIsTransmitTxCompleted** (void) End of transmit test function (UART mode) More...
bool	**st25r95SPIIsInListen** (void) Listen mode test function (CE mode) More...
rfalLmState	**st25r95SPIGetLmState** (void) Get Listen Mode state (CE mode) More...
ReturnCode	**st25r95SPIDeactivateACFilter** (void) Deactivate the Anti-Collision filter (CE mode) More...
ReturnCode	**st25r95SPISetACState** (uint8_t state) Set Anti-Collision filter state (CE mode) More...
void	**st25r95RearmListen** (void) Rearm Listen mode (CE mode) More...
void	**st25r95SPIIdle** (uint8_t dacDataL, uint8_t dacDataH, uint8_t WUPeriod) Send Idle command (SPI mode) More...
void	**st25r95SPIGetIdleResponse** (void)

Flush the Idle response (SPI mode) More...

void **st25r95UARTIdle** (uint8_t dacDataL, uint8_t dacDataH, uint8_t WUPeriod)
Send Idle command (UART mode) More...

void **st25r95UARTGetIdleResponse** (void)
Flush the Idle response (UART mode) More...

void **st25r95SPISendTransmitFlag** (uint8_t protocol, uint8_t transmitFlag)
Send transmit flag (SPI mode) More...

void **st25r95UARTSendTransmitFlag** (uint8_t protocol, uint8_t transmitFlag)
Send transmit flag (UART mode) More...

ReturnCode **st25r95SPICompleteRx** (void)
Process end of Rx (SPI mode) More...

ReturnCode **st25r95UARTCompleteRx** (void)
Process end of Rx (UART mode) More...

void **st25r95SPIKillIdle** (void)
return to ready mode (SPI mode) More...

void **st25r95UARTKillIdle** (void)
return to ready mode (UART mode) More...

uint8_t **st25r95CalibrateTagDetector** (void)
Calibrate Tag Detector. More...

void **st25r95SPIRxTx** (uint8_t *txData, uint8_t *rxData, uint16_t length)
SPI transceive function. More...

Variables

st25r95SPIRxContext **st25r95SPIRxCtx**

Detailed Description

Macro Definition Documentation

◆ ST25R95_ACFILTER_RESPONSE_BUFLEN

```
#define ST25R95_ACFILTER_RESPONSE_BUFLEN   (1 + 2)
```

ACFilter response buffer len

◆ ST25R95_ACSTATE_ACTIVE

```
#define ST25R95_ACSTATE_ACTIVE   0x04U
```

AC Filter state: Active

◆ ST25R95_ACSTATE_ACTIVEX

```
#define ST25R95_ACSTATE_ACTIVEX   0x84U
```

AC Filter state: Active*

◆ ST25R95_ACSTATE_HALT

```
#define ST25R95_ACSTATE_HALT   0x80U
```

AC Filter state: Halt

◆ ST25R95_ACSTATE_IDLE

```
#define ST25R95_ACSTATE_IDLE   0x00U
```

AC Filter state: Idle

◆ ST25R95_ACSTATE_READYA

```
#define ST25R95_ACSTATE_READYA   0x01U
```

AC Filter state: ReadyA

◆ ST25R95_ACSTATE_READYAX

```
#define ST25R95_ACSTATE_READYAX   0x81U
```

AC Filter state: ReadyA*

◆ ST25R95_CMD_COMMAND_OFFSET

```
#define ST25R95_CMD_COMMAND_OFFSET   0x00U
```

CMD Offset. See CR95HF DS ◆ 4.2.1

◆ ST25R95_CMD_DATA_OFFSET

```
#define ST25R95_CMD_DATA_OFFSET   0x02U
```

DATA[0] Offset. See CR95HF DS ◆ 4.2.1

◆ ST25R95_CMD_LENGTH_OFFSET

```
#define ST25R95_CMD_LENGTH_OFFSET   0x01U
```

LEN Offset. See CR95HF DS ◆ 4.2.1

◆ ST25R95_CMD_RESULT_OFFSET

```
#define ST25R95_CMD_RESULT_OFFSET   0x00U
```

Resp Code Offset. See CR95HF DS ◆ 4.2.1

◆ ST25R95_COMMAND_ACFILTER

```
#define ST25R95_COMMAND_ACFILTER   0x0D
```

Enables or disables the anti-collision filter for ISO/IEC 14443 Type A protocol.

◆ ST25R95_COMMAND_BAUDRATE

```
#define ST25R95_COMMAND_BAUDRATE   0x0A
```

Sets the UART baud rate.

◆ ST25R95_COMMAND_ECHO

```
#define ST25R95_COMMAND_ECHO   0x55
```

ST25R95 performs a serial interface ECHO command

◆ ST25R95_COMMAND_IDLE

```
#define ST25R95_COMMAND_IDLE   0x07
```

Switches the ST25R95 into a low consumption

◆ ST25R95_COMMAND_IDN

```
#define ST25R95_COMMAND_IDN   0x01
```

Requests short information about the ST25R95 and its revision.

◆ ST25R95_COMMAND_LISTEN

```
#define ST25R95_COMMAND_LISTEN   0x05
```

Listens for data using previously selected protocol (used in Card Emulation mode).

◆ ST25R95_COMMAND_POLLFIED

```
#define ST25R95_COMMAND_POLLFIED   0x03
```

Returns the current value of the FieldDet flag (used in Card Emulation mode).

◆ ST25R95_COMMAND_PROTOCOLSELECT

```
#define ST25R95_COMMAND_PROTOCOLSELECT   0x02
```

Selects the RF communication protocol and specifies certain protocol-related parameters.

◆ ST25R95_COMMAND_RDREG

```
#define ST25R95_COMMAND_RDREG   0x08
```

Reads Wake-up event register or the Analog Register Configuration (ARC_B) register

◆ ST25R95_COMMAND_SEND

```
#define ST25R95_COMMAND_SEND   0x06
```

Sends data using previously selected protocol (used in Card Emulation mode).

◆ ST25R95_COMMAND_SENDRECV

```
#define ST25R95_COMMAND_SENDRECV   0x04
```

Sends data using the previously selected protocol and receives the tag response.

◆ ST25R95_COMMAND_WRREG

```
#define ST25R95_COMMAND_WRREG   0x09
```

Write register

◆ ST25R95_COMMUNICATION_BUFFER_SIZE

```
#define ST25R95_COMMUNICATION_BUFFER_SIZE   (528 + 2)
```

Max received buffer size

◆ ST25R95_COMMUNICATION_UART_WDOGTIMER

```
#define ST25R95_COMMUNICATION_UART_WDOGTIMER   (5000U)
```

Uart watchdog timer

◆ ST25R95_CONTROL_POLL

```
#define ST25R95_CONTROL_POLL   0x03
```

Poll the ST25R95

◆ ST25R95_CONTROL_POLL_NO_TIMEOUT

```
#define ST25R95_CONTROL_POLL_NO_TIMEOUT   0
```

non blocking polling

◆ ST25R95_CONTROL_POLL_TIMEOUT

```
#define ST25R95_CONTROL_POLL_TIMEOUT   100
```

Polling timeout

◆ ST25R95_CONTROL_READ

```
#define ST25R95_CONTROL_READ   0x02
```

Read data from the ST25R95

◆ ST25R95_CONTROL_RESET

```
#define ST25R95_CONTROL_RESET   0x01
```

Reset the ST25R95

◆ ST25R95_CONTROL_SEND

```
#define ST25R95_CONTROL_SEND   0x00
```

Send command to the ST25R95

◆ ST25R95_DACDATA_MAX

```
#define ST25R95_DACDATA_MAX   (0xFCU)
```

DacData max value (6 bits MSB)

◆ ST25R95_ECHO_RESPONSE_BUFLEN

```
#define ST25R95_ECHO_RESPONSE_BUFLEN   (3)
```

Echo response buffer len

◆ ST25R95_ERRCODE_61_SOF

```
#define ST25R95_ERRCODE_61_SOF   0x61
```

SOF error during the EMD process

◆ ST25R95_ERRCODE_62_CRC

```
#define ST25R95_ERRCODE_62_CRC   0x62
```

CRC error during the EMD process

◆ ST25R95_ERRCODE_63_SOF_HIGH

```
#define ST25R95_ERRCODE_63_SOF_HIGH   0x63
```

SOF error in ISO14443B occurs during high part (duration of 2 to 3 Elementary Unit Time, ETU)

◆ ST25R95_ERRCODE_65_SOF_LOW

```
#define ST25R95_ERRCODE_65_SOF_LOW   0x65
```

SOF error in ISO14443B occurs during low part (duration of 10 to 11 Elementary Unit Time, ETU)

◆ ST25R95_ERRCODE_66_EGT

```
#define ST25R95_ERRCODE_66_EGT   0x66
```

Extra Guard Time (EGT) error in ISO14443B

◆ ST25R95_ERRCODE_67_TR1TOOLONG

```
#define ST25R95_ERRCODE_67_TR1TOOLONG   0x67
```

TR1 set by card too long in case of protocol ISO14443B

◆ ST25R95_ERRCODE_68_TR1TOOSHORT

```
#define ST25R95_ERRCODE_68_TR1TOOSHORT   0x68
```

TTR1 set by card too short in case of protocol ISO14443B

◆ ST25R95_ERRCODE_COMERROR

```
#define ST25R95_ERRCODE_COMERROR   0x86
```

Hardware communication error

◆ ST25R95_ERRCODE_CRC

```
#define ST25R95_ERRCODE_CRC   0x8D
```

CRC error, Valid only for ISO/IEC 18092

◆ ST25R95_ERRCODE_EGT

```
#define ST25R95_ERRCODE_EGT   0x8B
```

EGT time out

◆ ST25R95_ERRCODE_FIELDLENGTH

```
#define ST25R95_ERRCODE_FIELDLENGTH   0x8C
```

Valid for ISO/IEC 18092, if Length <3

◆ ST25R95_ERRCODE_FRAMEOKADDITIONALINFO

```
#define ST25R95_ERRCODE_FRAMEOKADDITIONALINFO   0x80
```

Frame correctly received (additionally see CRC/Parity information)

◆ ST25R95_ERRCODE_FRAMEWAITTIMEOUT

```
#define ST25R95_ERRCODE_FRAMEWAITTIMEOUT   0x87
```

Frame wait time out (no valid reception)

◆ ST25R95_ERRCODE_FRAMING

```
#define ST25R95_ERRCODE_FRAMING   0x8A
```

if start bit = 1 or stop bit = 0

◆ ST25R95_ERRCODE_INVALIDCCMDCODE

```
#define ST25R95_ERRCODE_INVALIDCCMDCODE   0x81
```

Invalid command code

◆ ST25R95_ERRCODE_INVALIDCMDLENGHT

```
#define ST25R95_ERRCODE_INVALIDCMDLENGHT   0x82
```

Invalid command length

◆ ST25R95_ERRCODE_INVALIDPROTOCOL

```
#define ST25R95_ERRCODE_INVALIDPROTOCOL   0x83
```

Invalid protocol

◆ ST25R95_ERRCODE_INVALIDSOF

```
#define ST25R95_ERRCODE_INVALIDSOF   0x88
```

Invalid SOF

◆ ST25R95_ERRCODE_NOFIELD

```
#define ST25R95_ERRCODE_NOFIELD   0x8F
```

When Listen command detects the absence of external field

◆ ST25R95_ERRCODE_NONE

```
#define ST25R95_ERRCODE_NONE   0x00
```

no error occurred

◆ ST25R95_ERRCODE_OVERFLOW

```
#define ST25R95_ERRCODE_OVERFLOW   0x89
```

Too many bytes received and data still arriving

◆ ST25R95_ERRCODE_RECEPTIONLOST

```
#define ST25R95_ERRCODE_RECEPTIONLOST   0x8E
```

When reception is lost without EOF received (or subcarrier was lost)

◆ ST25R95_ERRCODE_RESULTSRESIDUAL

```
#define ST25R95_ERRCODE_RESULTSRESIDUAL   0x90
```

Residual bits in last byte. Useful for ACK/NAK reception of ISO/IEC 14443 Type A.

◆ ST25R95_FWT_MAX

```
#define ST25R95_FWT_MAX   0x40A8BC0
```

Max FWT supported: 5s

◆ ST25R95_IDLE_DACDATAH_OFFSET

```
#define ST25R95_IDLE_DACDATAH_OFFSET   (0x0DU)
```

DacDataH offset in Idle Command

◆ ST25R95_IDLE_DACDATAL_OFFSET

```
#define ST25R95_IDLE_DACDATAL_OFFSET   (0x0CU)
```

DacDataL offset in Idle Command

◆ ST25R95_IDLE_RESPONSE_BUFLEN

```
#define ST25R95_IDLE_RESPONSE_BUFLEN   (1 + 2)
```

Idle response buffer len

◆ ST25R95_IDLE_WKUP_TAGDETECT

```
#define ST25R95_IDLE_WKUP_TAGDETECT   (0x02U)
```

Idle wakeup source: Tag Detected

◆ ST25R95_IDLE_WKUP_TIMEOUT

```
#define ST25R95_IDLE_WKUP_TIMEOUT   (0x01U)
```

Idle wakeup source: timeout

◆ ST25R95_IDLE_WUPERIOD_OFFSET

```
#define ST25R95_IDLE_WUPERIOD_OFFSET   (0x09U)
```

WUPeriod offset in Idle Command

◆ ST25R95_IDN_RESPONSE_BUFLEN

```
#define ST25R95_IDN_RESPONSE_BUFLEN   (15 + 2)
```

IDN response buffer len

◆ ST25R95_IS_PROT_ISO14443A_COLLISION_ERR

```
#define ST25R95_IS_PROT_ISO14443A_COLLISION_ERR (   status )   (((status) & 0x80U) == 0x80U)
```

Test for Colission flag in SendRcv response additional byte for ISO14443A protocol

◆ ST25R95_IS_PROT_ISO14443A_CRC_ERR

```
#define ST25R95_IS_PROT_ISO14443A_CRC_ERR (   status )   (((status) & 0x20U) == 0x20U)
```

Test for CRC flag in SendRcv response additional byte for ISO14443A protocol

◆ ST25R95_IS_PROT_ISO14443A_PARITY_ERR

```
#define ST25R95_IS_PROT_ISO14443A_PARITY_ERR ( status ) (((status) & 0x10U) == 0x10U)
```

Test for Parity flag in SendRcv response additional byte for ISO14443A protocol

◆ ST25R95_IS_PROT_ISO14443B_CRC_ERR

```
#define ST25R95_IS_PROT_ISO14443B_CRC_ERR ( status ) (((status) & 0x02U) == 0x02U)
```

Test for CRC flag in SendRcv response additional byte for ISO14443B protocol

◆ ST25R95_IS_PROT_ISO15693_COLLISION_ERR

```
#define ST25R95_IS_PROT_ISO15693_COLLISION_ERR ( status ) (((status) & 0x01U) == 0x01U)
```

Test for Collision flag in SendRcv response additional byte for ISO15693 protocol

◆ ST25R95_IS_PROT_ISO15693_CRC_ERR

```
#define ST25R95_IS_PROT_ISO15693_CRC_ERR ( status ) (((status) & 0x02U) == 0x02U)
```

Test for CRC flag in SendRcv response additional byte for ISO15693 protocol

◆ ST25R95_IS_PROT_ISO18092_CRC_ERR

```
#define ST25R95_IS_PROT_ISO18092_CRC_ERR ( status ) (((status) & 0x02U) == 0x02U)
```

Test for CRC flag in SendRcv response additional byte for ISO18092 protocol

◆ ST25R95_LISTEN_RESPONSE_BUFLEN

```
#define ST25R95_LISTEN_RESPONSE_BUFLEN (0 + 2)
```

Listen response buffer len

◆ ST25R95_POLLFIELD_RESPONSE_BUFLEN

```
#define ST25R95_POLLFIELD_RESPONSE_BUFLEN   (1 + 2)
```

PollField response buffer len

◆ ST25R95_PROTOCOL_CE_ISO14443A

```
#define ST25R95_PROTOCOL_CE_ISO14443A   0x05U
```

ProtocolSelect protocol code: ISO14443 (Card Emulation)

◆ ST25R95_PROTOCOL_FIELDOFF

```
#define ST25R95_PROTOCOL_FIELDOFF   0x00U
```

ProtocolSelect protocol code: Field OFF

◆ ST25R95_PROTOCOL_ISO14443A

```
#define ST25R95_PROTOCOL_ISO14443A   0x02U
```

ProtocolSelect protocol code: ISO14443A (Reader)

◆ ST25R95_PROTOCOL_ISO14443B

```
#define ST25R95_PROTOCOL_ISO14443B   0x03U
```

ProtocolSelect protocol code: ISO14443B (Reader)

◆ ST25R95_PROTOCOL_ISO15693

```
#define ST25R95_PROTOCOL_ISO15693   0x01U
```

ProtocolSelect protocol code: ISO15693 (Reader)

◆ ST25R95_PROTOCOL_ISO18092

#define ST25R95_PROTOCOL_ISO18092 0x04U

ProtocolSelect protocol code: ISO18092 (Reader)

◆ ST25R95_PROTOCOL_MAX

#define ST25R95_PROTOCOL_MAX **ST25R95_PROTOCOL_CE_ISO14443A**

Max value of protocol field

◆ ST25R95_PROTOCOLSELECT_BR_OFFSET

#define ST25R95_PROTOCOLSELECT_BR_OFFSET (3U)

Bit Rate offset in ProtocolSelect Command

◆ ST25R95_PROTOCOLSELECT_RESPONSE_BUFLEN

#define ST25R95_PROTOCOLSELECT_RESPONSE_BUFLEN (0 + 2)

ProtocolSelect response buffer len

◆ ST25R95_RDREG_RESPONSE_BUFLEN

#define ST25R95_RDREG_RESPONSE_BUFLEN (1 + 2)

ReadReg response buffer len

◆ ST25R95_REG_ACC_A

#define ST25R95_REG_ACC_A 0x6804

ACC_A register address

◆ ST25R95_REG_ARC_B

```
#define ST25R95_REG_ARC_B   0x6801
```

ARC_B register address

◆ ST25R95_REG_TIMERW

```
#define ST25R95_REG_TIMERW   0x3A00
```

TIMER register address

◆ ST25R95_SEND_RESPONSE_BUFLEN

```
#define ST25R95_SEND_RESPONSE_BUFLEN   (0 + 2)
```

Send response buffer len

◆ ST25R95_SPI_DUMMY_BYTE

```
#define ST25R95_SPI_DUMMY_BYTE   0x00
```

Dummy byte when nothing to transmit on SPI

◆ ST25R95_WRREG_RESPONSE_BUFLEN

```
#define ST25R95_WRREG_RESPONSE_BUFLEN   (0 + 2)
```

WriteRead response buffer len

◆ st25r95CommandEcho

#define st25r95CommandEcho ( ) **st25r95SPICommandEcho**()

UART/SPI wrapper for st25r95CommandEcho

◆ st25r95CompleteRx

```
#define st25r95CompleteRx ( ) st25r95SPICompleteRx();
```

UART/SPI wrapper for st25r95CompleteRx

◆ st25r95DeactivateACFilter

```
#define st25r95DeactivateACFilter ( ) st25r95SPIDeactivateACFilter()
```

UART/SPI wrapper for st25r95DeactivateACFilter

◆ st25r95FlushChipSPIBuffer

```
#define st25r95FlushChipSPIBuffer ( ) st25r95SPIRxTx(NULL, NULL, ST25R95_COMMUNICATION_BUFFER_SIZE)
```

st25r95FlushChipSPIBuffer defined as a macro for better readibility

◆ st25r95GetIdleResponse

```
#define st25r95GetIdleResponse ( ) st25r95SPIGetIdleResponse()
```

UART/SPI wrapper for st25r95GetIdleResponse

◆ st25r95GetLmState

```
#define st25r95GetLmState ( ) st25r95SPIGetLmState()
```

UART/SPI wrapper for st25r95GetLmState

◆ st25r95Idle

```
#define st25r95Idle ( DacDataL,
                      DacDataH,
                      WUPeriod
                    ) st25r95SPIIdle((DacDataL), (DacDataH), (WUPeriod))
```

UART/SPI wrapper for st25r95Idle

◆ st25r95IsInListen

#define st25r95IsInListen ( ) **st25r95SPIIsInListen**()

UART/SPI wrapper for st25r95IsInListen

◆ st25r95IsTransmitTxCompleted

#define st25r95IsTransmitTxCompleted ( ) **st25r95SPIIsTransmitCompleted**()

UART/SPI wrapper for st25r95IsTransmitTxCompleted

◆ st25r95KillIdle

#define st25r95KillIdle ( ) **st25r95SPIKillIdle**()

UART/SPI wrapper for st25r95KillIdle

◆ st25r95PollRead

#define st25r95PollRead (timeout) **st25r95SPIPollRead**((timeout))

UART/SPI wrapper for st25r95PollRead

◆ st25r95PrepareRx

```
#define
st25r95PrepareRx        ( protocol,
                          rxBuf,
                          rxBufLen,
                          rxRcvdLen,
                          flags,
                          additionalRespBytes
```

```
st25r95SPIPrepareRx((protocol), (rxBuf), (rxBufLen), (rxRcvdLen), (flags),
) (additionalRespBytes));
```

UART/SPI wrapper for st25r95PrepareRx

◆ st25r95SendCommandTypeAndLen

```
#define st25r95SendCommandTypeAndLen ( cmd,
                                       resp,
                                       respBuffLen
                                     ) st25r95SPISendCommandTypeAndLen((cmd), (resp), (respBuffLen))
```

UART/SPI wrapper for st25r95SendCommandTypeAndLen

◆ st25r95SendData

```
#define st25r95SendData ( buf,
                          bufLen,
                          protocol,
                          flags
                        ) st25r95SPISendData((buf), (bufLen), (protocol), (flags))
```

UART/SPI wrapper for st25r95SendData

◆ st25r95SendTransmitFlag

```
#define st25r95SendTransmitFlag ( protocol,
                                  transmitFlag
                                ) st25r95SPISendTransmitFlag((protocol), (transmitFlag))
```

UART/SPI wrapper for st25r95SendTransmitFlag

◆ st25r95SetACState

```
#define st25r95SetACState ( State ) st25r95SPISetACState(State)
```

UART/SPI wrapper for st25r95SetACState

Function Documentation

◆ st25r95CalibrateTagDetector()

uint8_t st25r95CalibrateTagDetector (void)

This function returns the Tag Detector DAC calibration value.

Returns

uint8_t Any : calibration value

◆ st25r95Listen()

ReturnCode st25r95Listen (void)

This function starts the receive in CE mode.

Returns

RFAL_ERR_NONE : Operation successful

◆ st25r95PollField()

bool st25r95PollField (void)

This function is used to poll field. Only available in CE mode.

Returns

false : field OFF

true : field ON

◆ st25r95ProtocolSelect()

ReturnCode st25r95ProtocolSelect (uint8_t protocol)

This function is used to send ST25R95 ProtocolSelect command

Parameters

[in] **protocol** value of the protocol

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_TIMEOUT : Timeout

RFAL_ERR_PARAM : ProtocolSelect failure

◆ st25r95ReadReg()

```
ReturnCode st25r95ReadReg ( uint16_t  reg,
                            uint8_t *  value
                          )
```

This function is used to read ST25R95 ARC_B register

Parameters

[in] **reg** register (currently only ST25R95_REG_ARC_B register is defined)

[out] **value** register value

Returns

RFAL_ERR_NONE : Operation successful

◆ st25r95RearmListen()

```
void st25r95RearmListen ( void )
```

This function is used to rearm the listen mode. Only available in CE mode.

◆ st25r95SetACFilter()

```
ReturnCode st25r95SetACFilter ( const rfalLmConfPA *  confA )
```

This function is used to program the AC filter. Only available in CE mode.

Parameters

[in] **confA** pointer to Passive A configurations

Returns

RFAL_ERR_NONE : Operation successful

◆ st25r95SetBitRate()

```
ReturnCode st25r95SetBitRate ( uint8_t        protocol,
                               rfalBitRate    txBR,
                               rfalBitRate    rxBR
                             )
```

This function is used to set Tx and Rx bit rates

Parameters

- [in] **protocol** value of the protocol
- [in] **txBR** tx bit rate
- [in] **rxBR** rx bit rate

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_NOT_IMPLEMENTED : unsupported bit rate value

◆ st25r95SetFWT()

```
ReturnCode st25r95SetFWT ( uint8_t    protocol,
                           uint32_t   fwt
                         )
```

This function is used to set the FWT for the next frames

Parameters

- [in] **protocol** protocol index
- [in] **fwt** frame waiting time

Returns

RFAL_ERR_NONE : Operation successful

◆ st25r95SetSlotCounter()

```
ReturnCode st25r95SetSlotCounter ( uint8_t  slots )
```

This function is used to set the slot counter (ISO18092)

Parameters

[in] **slots** number of slots

Returns

RFAL_ERR_NONE : Operation successful

◆ st25r95SPICommandEcho()

ReturnCode st25r95SPICommandEcho (void)

This function is used to send an echo command

Returns

RFAL_ERR_NONE : Operation successful (echo response received)

RFAL_ERR_TIMEOUT : Timeout

◆ st25r95SPICompleteRx()

ReturnCode st25r95SPICompleteRx (void)

This function is used to conclude the Rx request and returns the status of the Rx.

Returns

RFAL_ERR_NONE : Operation successful

◆ st25r95SPIDeactivateACFilter()

ReturnCode st25r95SPIDeactivateACFilter (void)

This function is used to deactivate the AC filter. Only available in CE mode.

Returns

RFAL_ERR_NONE : Operation successful

◆ st25r95SPIGetIdleResponse()

void st25r95SPIGetIdleResponse (void)

This function is used to flush the Idle response.

◆ st25r95SPIGetLmState()

rfalLmState st25r95SPIGetLmState (void)

This function returns the Listen mode state.

Returns

rfalLmState Any : LM State

◆ st25r95SPIIdle()

```
void st25r95SPIIdle ( uint8_t dacDataL,
                      uint8_t dacDataH,
                      uint8_t WUPeriod
                    )
```

This function is used to send the Idle command.

Parameters

[in] **dacDataL** lower comparator value

[in] **dacDataH** higher comparator value

[in] **WUPeriod** time between 2 detections

◆ st25r95SPIIsInListen()

bool st25r95SPIIsInListen (void)

This function is used to test whether the ST25R95 is currently in Listen mode.

Returns

true : Listen mode on going

false : Not in Listen mode

◆ st25r95SPIIsTransmitCompleted()

bool st25r95SPIIsTransmitCompleted (void)

This function is used to test whether the current transmit is finished or not.

Returns

true : Transmit completed

false : Transmit not completed

◆ st25r95SPIKillIdle()

void st25r95SPIKillIdle (void)

This function is used to exit from Idle mode and return to Ready mode.

◆ st25r95SPIPollRead()

ReturnCode st25r95SPIPollRead (uint32_t timeout)

This function is used to poll the ST25R95 for incomming data

Parameters

[in] **timeout** timeout value

Returns

RFAL_ERR_NONE : Data ready to be read

RFAL_ERR_TIMEOUT : No data available

◆ st25r95SPIPollSend()

ReturnCode st25r95SPIPollSend (void)

This function is used to poll the ST25R95 before sending data

Returns

RFAL_ERR_NONE : ST25R95 ready to receive data

RFAL_ERR_TIMEOUT : ST25R95 not ready to receive data

◆ st25r95SPIPrepareRx()

```
void st25r95SPIPrepareRx ( uint8_t     protocol,
                           uint8_t *   rxBuf,
                           uint16_t    rxBufLen,
```

```
                    uint16_t *  rxRcvdLen,
                    uint32_t    flags,
                    uint8_t *   additionalRespBytes
                  )
```

This function is used to send data

Warning

This method should be called only if polling is successful

Parameters

[in]	**protocol**	: protocol index
[in]	**rxBuf**	: buffer to place the response
[in]	**rxBufLen**	: length of rxBuf
[in]	**rxRcvdLen**	: received length
[in]	**flags**	: TransceiveFlags indication special handling
[in]	**additionalRespBytes**	additionnal response bytes

◆ st25r95SPIResetChip()

```
void st25r95SPIResetChip ( void )
```

This function is used to reset the ST25R95 as per CR95HF DS ◆ 4.2.1 .

◆ st25r95SPIRxTx()

```
void st25r95SPIRxTx ( uint8_t *  txData,
                      uint8_t *  rxData,
                      uint16_t   length
                    )
```

This function is used for SPI communication.

Parameters

[in]	**txData**	Tx Data
[out]	**rxData**	Rx Data

[in] **length** Tx/Rx Data buffer len

◆ st25r95SPISendCommandTypeAndLen()

```
ReturnCode st25r95SPISendCommandTypeAndLen ( uint8_t *  cmd,
                                             uint8_t *  resp,
                                             uint16_t   respBuffLen
                                           )
```

This function is used to send a *cmd* and read the *resp*. The *cmd* command must follow the CMD LEN format (i.e. any command except echo)

Parameters

[in] **cmd** Command as per CR95HF DS ◆5.

[out] **resp** buffer for the response.

[in] **respBuffLen** response buffer length

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_TIMEOUT : Timeout

◆ st25r95SPISendData()

```
void st25r95SPISendData ( uint8_t *  buf,
                          uint8_t    bufLen,
                          uint8_t    protocol,
                          uint32_t   flags
                        )
```

This function is used to send data

Parameters

[in] **buf** : data to be sent

[in] **bufLen** : buffer length

[in] **protocol** protocol index

[in] **flags** : TransceiveFlags indication special handling

◆ st25r95SPISendReceiveByte()

uint8_t st25r95SPISendReceiveByte (uint8_t data)

This function is used to send one byte over SPI and returns the byte received

Parameters

[in] **data** byte to be sent over SPI

Returns

RFAL_ERR_NONE : Operation successful (echo response received)

RFAL_ERR_TIMEOUT : Timeout

◆ st25r95SPISendTransmitFlag()

void st25r95SPISendTransmitFlag (uint8_t protocol,
uint8_t transmitFlag
)

This function is used to send the transmit flag during the SendRecv command.

Parameters

[in] **protocol** value of the protocol

[in] **transmitFlag** transmission flags

◆ st25r95SPISetACState()

ReturnCode st25r95SPISetACState (uint8_t state)

This function is used to set the AC filter state. Only available in CE mode.

Parameters

[in] **state** AC state

Returns

RFAL_ERR_NONE : Operation successful

◆ st25r95UARTCommandEcho()

ReturnCode st25r95UARTCommandEcho (void)

This function is used to send an echo command

Returns

RFAL_ERR_NONE : Operation successful (echo response received)

RFAL_ERR_TIMEOUT : Timeout

◆ st25r95UARTCompleteRx()

ReturnCode st25r95UARTCompleteRx (void)

This function is used to conclude the Rx request and returns the status of the Rx.

Returns

RFAL_ERR_NONE : Operation successful

◆ st25r95UartErrorCallback()

void st25r95UartErrorCallback (void)

Uart transceive error interrupt handler callback.

◆ st25r95UARTGetIdleResponse()

void st25r95UARTGetIdleResponse (void)

This function is used to flush the Idle response.

◆ st25r95UARTIdle()

```
void st25r95UARTIdle ( uint8_t  dacDataL,
                       uint8_t  dacDataH,
                       uint8_t  WUPeriod
                     )
```

This function is used to send the Idle command.

Parameters

[in] **dacDataL** lower comparator value

[in] **dacDataH** higher comparator value

[in] **WUPeriod** time between 2 detections

◆ st25r95UARTIsTransmitTxCompleted()

bool st25r95UARTIsTransmitTxCompleted (void)

This function is used to test whether the current transmit is finished or not.

Returns

true : Transmit completed

false : Transmit not completed

◆ st25r95UARTKillIdle()

void st25r95UARTKillIdle (void)

This function is used to exit from Idle mode and return to Ready mode.

◆ st25r95UARTPollRead()

ReturnCode st25r95UARTPollRead (uint32_t timeout)

This function is used to poll the ST25R95 for incomming data

Parameters

[in] **timeout** timeout value

Returns

RFAL_ERR_NONE : Data ready to be read

RFAL_ERR_TIMEOUT : No data available

◆ st25r95UARTPrepareRx()

void st25r95UARTPrepareRx (uint8_t protocol,
uint8_t * rxBuf,

```
                    uint16_t    rxBufLen,
                    uint16_t *  rxRcvdLen,
                    uint32_t    flags,
                    uint8_t *   additionalRespBytes
                  )
```

This function is used to send data

Parameters

[in]	**protocol**	: protocol index
[in]	**rxBuf**	: buffer to place the response
[in]	**rxBufLen**	: length of rxBuf
[in]	**rxRcvdLen**	: received length
[in]	**flags**	: TransceiveFlags indication special handling
[in]	**additionalRespBytes**	additionnal response bytes

◆ st25r95UartRxCpltCallback()

```
void st25r95UartRxCpltCallback ( void )
```

Uart Rx interrupt handler callback.

◆ st25r95UARTSendCommandTypeAndLen()

```
ReturnCode st25r95UARTSendCommandTypeAndLen ( uint8_t *  cmd,
                                              uint8_t *  resp,
                                              uint16_t   respBuffLen
                                            )
```

This function is used to send a *cmd* and read the *resp*. The *cmd* command must follow the CMD LEN format (i.e. any command except echo)

Parameters

[in]	**cmd**	Command as per CR95HF DS ⁄5.
[out]	**resp**	buffer for the response.
[in]	**respBuffLen**	response buffer length

Returns

RFAL_ERR_NONE : Operation successful

RFAL_ERR_TIMEOUT : Timeout

◆ st25r95UARTSendData()

```
void st25r95UARTSendData ( uint8_t *  buf,
                           uint8_t    bufLen,
                           uint8_t    protocol,
                           uint32_t   flags
                         )
```

This function is used to send data

Parameters

[in] **buf** : data to be sent

[in] **bufLen** : buffer length

[in] **protocol** protocol index

[in] **flags** : TransceiveFlags indication special handling

◆ st25r95UARTSendTransmitFlag()

```
void st25r95UARTSendTransmitFlag ( uint8_t  protocol,
                                   uint8_t  transmitFlag
                                 )
```

This function is used to send the transmit flag during the SendRecv command.

Parameters

[in] **protocol** value of the protocol

[in] **transmitFlag** transmission flags

◆ st25r95UartTxCpltCallback()

```
void st25r95UartTxCpltCallback ( void  )
```

Uart Tx interrupt handler callback.

◆ st25r95WriteReg()

```
ReturnCode st25r95WriteReg ( uint8_t   protocol,
                             uint16_t  reg,
                             uint8_t   value
                           )
```

This function is used to write ST25R95 ARC_B register

Parameters

- [in] **protocol** protocol index
- [in] **reg** register (currently only ST25R95_REG_ARC_B register is defined)
- [in] **value** register value

Returns

RFAL_ERR_NONE : Operation successful

Variable Documentation

◆ st25r95SPIRxCtx

st25r95SPIRxContext st25r95SPIRxCtx exter n

Context for SPI transceive

st25r95SPIRxContext Struct Reference


```
#include <st25r95_com.h>
```

Data Fields

bool	**rmvCRC**
bool	**inListen**
bool	**NFCIP1**
rfalLmState	**LmState**
uint16_t	**rxBufLen**
uint16_t *	**rxRcvdLen**
uint8_t *	**rxBuf**
uint8_t *	**additionalRespBytes**
ReturnCode	**retCode**
uint8_t	**BufCRC** [2]
uint8_t	**NFCIP1_SoD** [1]
uint8_t	**protocol**

Detailed Description

SPI transceive context definition

Field Documentation

◆ additionalRespBytes

uint8_t* st25r95SPIRxContext::additionalRespBytes

additionalRespBytes

◆ BufCRC

uint8_t st25r95SPIRxContext::BufCRC[2]

BufCRC

◆ inListen

bool st25r95SPIRxContext::inListen

inListen flags

◆ LmState

rfalLmState st25r95SPIRxContext::LmState

LmState

◆ NFCIP1

bool st25r95SPIRxContext::NFCIP1

NFCIP1 flags

◆ NFCIP1_SoD

uint8_t st25r95SPIRxContext::NFCIP1_SoD[1]

NFCIP1_SoD

◆ protocol

uint8_t st25r95SPIRxContext::protocol

protocol

◆ retCode

ReturnCode st25r95SPIRxContext::retCode

retCode

◆ rmvCRC

bool st25r95SPIRxContext::rmvCRC

Remove CRC flag

◆ rxBuf

uint8_t* st25r95SPIRxContext::rxBuf

rxBuf

◆ rxBufLen

uint16_t st25r95SPIRxContext::rxBufLen

rxBufLen

◆ rxRcvdLen

uint16_t* st25r95SPIRxContext::rxRcvdLen

rxRcvdLen

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

- st25r95_com.h

Generated on Fri Mar 14 2025 11:45:10 for RFAL by doxygen 1.9.5

RFAL

RFAL-HL

RFAL

RFAL Higher Layer. More...

Collaboration diagram for RFAL-HL:

Modules

CD

RFAL Card Detection.

NFC
RFAL NFC Device.

Detailed Description

CD

RFAL » **RFAL-HL**

RFAL Card Detection. More...

Collaboration diagram for CD:

Data Structures

struct **rfalCdRes**

Enumerations

```
enum rfalCdTech {
  RFAL_CD_TECH_NONE = 0x00 , RFAL_CD_TECH_NFCA = 0x01 , RFAL_CD_TECH_NFCB = 0x02 ,
RFAL_CD_TECH_NFCF = 0x04 ,
  RFAL_CD_TECH_NFCV = 0x08 , RFAL_CD_TECH_OTHER = 0x10
}
```

```
enum rfalCdDetType {
  RFAL_CD_NOT_FOUND = 0 , RFAL_CD_SINGLE_DEV = 1 , RFAL_CD_MULTIPLE_DEV = 2 ,
RFAL_CD_MULTIPLE_TECH = 3 ,
  RFAL_CD_CARD_TECH = 4 , RFAL_CD_SINGLE_MULTI_TECH = 5 , RFAL_CD_SINGLE_P2P = 6 ,
RFAL_CD_SINGLE_HB = 7 ,
  RFAL_CD_UNKOWN = 8
}
```

Functions

ReturnCode **rfalCdDetectCard** (**rfalCdRes** *result)
Detect Card. More...

ReturnCode **rfalCdStartDetectCard** (**rfalCdRes** *result)
Start Card Detection. More...

ReturnCode **rfalCdGetDetectCardStatus** (void)
Get Card Detection Status. More...

Detailed Description

Enumeration Type Documentation

◆ rfalCdDetType

enum **rfalCdDetType**

Card Detection result|outcome type

◆ rfalCdTech

enum **rfalCdTech**

Card Detection NFC technology type

Enumerator	
RFAL_CD_TECH_NONE	No NFC Technology
RFAL_CD_TECH_NFCA	NFC Technology NFCB
RFAL_CD_TECH_NFCB	NFC Technology NFCB
RFAL_CD_TECH_NFCF	NFC Technology NFCF
RFAL_CD_TECH_NFCV	NFC Technology NFCV
RFAL_CD_TECH_OTHER	NFC Technology OTHER

Function Documentation

◆ rfalCdDetectCard()

ReturnCode rfalCdDetectCard (**rfalCdRes** * result)

This function checks if a passive NFC card is present in the vicinity

Parameters

[out] **result** : Pointer to detection result|outcome

Returns

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_WRONG_STATE : Incorrect state for this operation

RFAL_ERR_NONE : Detection excuted with no error

RFAL_ERR_RF_COLLISION : RF carrier collision detected

RFAL_ERR_XXXX : Error occurred

◆ rfalCdGetDetectCardStatus()

ReturnCode rfalCdGetDetectCardStatus (void)

This function gets the status of the card detection

Returns

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_WRONG_STATE : Incorrect state for this operation

RFAL_ERR_NONE : Detection excuted with no error

RFAL_ERR_RF_COLLISION : RF carrier collision detected

RFAL_ERR_XXXX : Error occurred

◆ rfalCdStartDetectCard()

ReturnCode rfalCdStartDetectCard (**rfalCdRes** * result)

This function starts the detection for a passive NFC card is present in the vicinity

Parameters

[out] **result** : Pointer to detection result|outcome

Returns

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_WRONG_STATE : Incorrect state for this operation

RFAL_ERR_NONE : Detection will be executed

rfalCdRes Struct Reference


```
#include <rfal_cd.h>
```

Data Fields

bool	**detected**
rfalCdDetType	**detType**

Detailed Description

Card Detection result|outcome

Field Documentation

◆ detected

bool rfalCdRes::detected

Card detected flag

◆ detType

rfalCdDetType rfalCdRes::detType

Card detection type

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

- rfal_cd.h

NFC

RFAL NFC Device. More...

Collaboration diagram for NFC:

Data Structures

struct	**rfalNfcDevice**
struct	**rfalNfcPropCallbacks**
struct	**rfalNfcDiscoverParam**
union	**rfalNfcBuffer**

Macros

```
#define RFAL_NFC_TECH_NONE 0x0000U
#define RFAL_NFC_POLL_TECH_A 0x0001U
#define RFAL_NFC_POLL_TECH_B 0x0002U
#define RFAL_NFC_POLL_TECH_F 0x0004U
#define RFAL_NFC_POLL_TECH_V 0x0008U
#define RFAL_NFC_POLL_TECH_AP2P 0x0010U
#define RFAL_NFC_POLL_TECH_ST25TB 0x0020U
#define RFAL_NFC_POLL_TECH_PROP 0x0040U
#define RFAL_NFC_LISTEN_TECH_A 0x1000U
#define RFAL_NFC_LISTEN_TECH_B 0x2000U
#define RFAL_NFC_LISTEN_TECH_F 0x4000U
#define RFAL_NFC_LISTEN_TECH_AP2P 0x8000U
#define rfalNfcIsDevActivated(st) ( ((st)>= RFAL_NFC_STATE_ACTIVATED) &&
        ((st)<RFAL_NFC_STATE_DEACTIVATION) )
#define rfalNfcIsInDiscovery(st) ( ((st)>= RFAL_NFC_STATE_START_DISCOVERY) &&
        ((st)<RFAL_NFC_STATE_ACTIVATED) )
#define rfalNfcIsRemDevPoller(tp) ( ((tp)>= RFAL_NFC_POLL_TYPE_NFCA) &&
        ((tp)<=RFAL_NFC_POLL_TYPE_AP2P ) )
#define rfalNfcIsRemDevListener(tp) ( ((int16_t)(tp)>= (int16_t)RFAL_NFC_LISTEN_TYPE_NFCA) &&
        ((tp)<=RFAL_NFC_LISTEN_TYPE_AP2P) )
#define rfalNfcDefaultDiscParams(dp)
```

Typedefs

typedef **ReturnCode**(* **rfalNfcPropCallback**) (void)

Enumerations

enum **rfalNfcState** {
RFAL_NFC_STATE_NOTINIT = 0 , **RFAL_NFC_STATE_IDLE** = 1 , **RFAL_NFC_STATE_START_DISCOVERY** = 2 , **RFAL_NFC_STATE_WAKEUP_MODE** = 3 ,
RFAL_NFC_STATE_POLL_TECHDETECT = 10 , **RFAL_NFC_STATE_POLL_COLAVOIDANCE** = 11 , **RFAL_NFC_STATE_POLL_SELECT** = 12 , **RFAL_NFC_STATE_POLL_ACTIVATION** = 13 ,
RFAL_NFC_STATE_LISTEN_TECHDETECT = 20 , **RFAL_NFC_STATE_LISTEN_COLAVOIDANCE** = 21 , **RFAL_NFC_STATE_LISTEN_ACTIVATION** = 22 , **RFAL_NFC_STATE_LISTEN_SLEEP** = 23 ,
RFAL_NFC_STATE_ACTIVATED = 30 , **RFAL_NFC_STATE_DATAEXCHANGE** = 31 , **RFAL_NFC_STATE_DATAEXCHANGE_DONE** = 33 , **RFAL_NFC_STATE_DEACTIVATION** = 34
}

enum **rfalNfcDevType** {
RFAL_NFC_LISTEN_TYPE_NFCA = 0 , **RFAL_NFC_LISTEN_TYPE_NFCB** = 1 , **RFAL_NFC_LISTEN_TYPE_NFCF** = 2 , **RFAL_NFC_LISTEN_TYPE_NFCV** = 3 ,
RFAL_NFC_LISTEN_TYPE_ST25TB = 4 , **RFAL_NFC_LISTEN_TYPE_AP2P** = 5 , **RFAL_NFC_LISTEN_TYPE_PROP** = 6 , **RFAL_NFC_POLL_TYPE_NFCA** = 10 ,
RFAL_NFC_POLL_TYPE_NFCB = 11 , **RFAL_NFC_POLL_TYPE_NFCF** = 12 , **RFAL_NFC_POLL_TYPE_NFCV** = 13 , **RFAL_NFC_POLL_TYPE_AP2P** = 15
}

enum **rfalNfcRfInterface** { **RFAL_NFC_INTERFACE_RF** = 0 , **RFAL_NFC_INTERFACE_ISODEP** = 1 , **RFAL_NFC_INTERFACE_NFCDEP** = 2 }

enum **rfalNfcDeactivateType** { **RFAL_NFC_DEACTIVATE_IDLE** = 0 , **RFAL_NFC_DEACTIVATE_SLEEP** = 1 , **RFAL_NFC_DEACTIVATE_DISCOVERY** = 2 }

Functions

void **rfalNfcWorker** (void)
RFAL NFC Worker. More...

ReturnCode **rfalNfcInitialize** (void)

	RFAL NFC Initialize. More...
ReturnCode	**rfalNfcDiscover** (const **rfalNfcDiscoverParam** *disParams) RFAL NFC Discovery. More...
rfalNfcState	**rfalNfcGetState** (void) RFAL NFC Get State. More...
ReturnCode	**rfalNfcGetDevicesFound** (**rfalNfcDevice** **devList, uint8_t *devCnt) RFAL NFC Get Devices Found. More...
ReturnCode	**rfalNfcGetActiveDevice** (**rfalNfcDevice** **dev) RFAL NFC Get Active Device. More...
ReturnCode	**rfalNfcSelect** (uint8_t devIdx) RFAL NFC Select Device. More...
ReturnCode	**rfalNfcDataExchangeStart** (uint8_t *txData, uint16_t txDataLen, uint8_t **rxData, uint16_t **rvdLen, uint32_t fwt) RFAL NFC Start Data Exchange. More...
ReturnCode	**rfalNfcDataExchangeGetStatus** (void) RFAL NFC Get Data Exchange Status. More...
ReturnCode	**rfalNfcDeactivate** (**rfalNfcDeactivateType** deactType) RFAL NFC Deactivate. More...

Detailed Description

Macro Definition Documentation

◆ RFAL_NFC_LISTEN_TECH_A

```
#define RFAL_NFC_LISTEN_TECH_A   0x1000U
```

Listen NFC-A technology Flag

◆ RFAL_NFC_LISTEN_TECH_AP2P

```
#define RFAL_NFC_LISTEN_TECH_AP2P   0x8000U
```

Listen AP2P technology Flag

◆ RFAL_NFC_LISTEN_TECH_B

```
#define RFAL_NFC_LISTEN_TECH_B   0x2000U
```

Listen NFC-B technology Flag

◆ RFAL_NFC_LISTEN_TECH_F

```
#define RFAL_NFC_LISTEN_TECH_F   0x4000U
```

Listen NFC-F technology Flag

◆ RFAL_NFC_POLL_TECH_A

```
#define RFAL_NFC_POLL_TECH_A   0x0001U
```

Poll NFC-A technology Flag

◆ RFAL_NFC_POLL_TECH_AP2P

```
#define RFAL_NFC_POLL_TECH_AP2P   0x0010U
```

Poll AP2P technology Flag

◆ RFAL_NFC_POLL_TECH_B

```
#define RFAL_NFC_POLL_TECH_B   0x0002U
```

Poll NFC-B technology Flag

◆ RFAL_NFC_POLL_TECH_F

#define RFAL_NFC_POLL_TECH_F 0x0004U

Poll NFC-F technology Flag

◆ RFAL_NFC_POLL_TECH_PROP

#define RFAL_NFC_POLL_TECH_PROP 0x0040U

Poll Proprietary technology Flag

◆ RFAL_NFC_POLL_TECH_ST25TB

#define RFAL_NFC_POLL_TECH_ST25TB 0x0020U

Poll ST25TB technology Flag

◆ RFAL_NFC_POLL_TECH_V

#define RFAL_NFC_POLL_TECH_V 0x0008U

Poll NFC-V technology Flag

◆ RFAL_NFC_TECH_NONE

#define RFAL_NFC_TECH_NONE 0x0000U

No technology

◆ rfalNfcDefaultDiscParams

#define rfalNfcDefaultDiscParams (dp)

Value:

```
if( (dp) != NULL) {
\
RFAL_MEMSET( (dp), 0x00,
sizeof(rfalNfcDiscoverParam) );  \
```

```
                                        ((dp))->compMode                =
RFAL_COMPLIANCE_MODE_NFC; \
                                        ((dp))->devLimit                = 1U;
\
                                        ((dp))->nfcfBR                  = RFAL_BR_212;
\
                                        ((dp))->ap2pBR                  = RFAL_BR_424;
\
                                        ((dp))->maxBR                   = RFAL_BR_KEEP;
\
                                        ((dp))->isoDepFS                =
RFAL_ISODEP_FSXI_256;     \
                                        ((dp))->nfcDepLR                = RFAL_NFCDEP_LR_254;
\
                                        ((dp))->GBLen                   = 0U;
\
                                        ((dp))->p2pNfcaPrio             = false;
\
                                        ((dp))->wakeupEnabled           = false;
\
                                        ((dp))->wakeupConfigDefault     = true;
\
                                        ((dp))->wakeupPollBefore        = false;
\
                                        ((dp))->wakeupNPolls            = 1U;
\
                                        ((dp))->totalDuration           = 1000U;
\
                                        ((dp))->techs2Find              = RFAL_NFC_TECH_NONE;
\
                                        ((dp))->techs2Bail              = RFAL_NFC_TECH_NONE;
\
                                        }
```

Sets the discover parameters to its default values

◆ rfalNfcIsDevActivated

#define rfalNfcIsDevActivated (st) (((st)>= **RFAL_NFC_STATE_ACTIVATED**) && ((st)<**RFAL_NFC_STATE_DEACTIVATION**))

Checks if a device is currently activated

◆ rfalNfcIsInDiscovery

#define rfalNfcIsInDiscovery (st) (((st)>= **RFAL_NFC_STATE_START_DISCOVERY**) && ((st)<**RFAL_NFC_STATE_ACTIVATED**))

Checks if a device is in discovery

◆ rfalNfcIsRemDevListener

```
#define rfalNfcIsRemDevListener ( tp ) ((int16_t)(tp)>= (int16_t)RFAL_NFC_LISTEN_TYPE_NFCA) && ((tp)<=RFAL_NFC_LISTEN_TYPE_AP2P) )
```

Checks if remote device is in Listen mode

◆ rfalNfcIsRemDevPoller

```
#define rfalNfcIsRemDevPoller ( tp ) ((tp)>= RFAL_NFC_POLL_TYPE_NFCA) && ((tp)<=RFAL_NFC_POLL_TYPE_AP2P ) )
```

Checks if remote device is in Poll mode

Typedef Documentation

◆ rfalNfcPropCallback

```
typedef ReturnCode(* rfalNfcPropCallback) (void)
```

Callbacks for Proprietary|Other Technology Activity 2.1 & EMVCo 3.0 9.2

Enumeration Type Documentation

◆ rfalNfcDeactivateType

```
enum rfalNfcDeactivateType
```

Deactivation type

Enumerator	
RFAL_NFC_DEACTIVATE_IDLE	Deactivate and go to IDLE
RFAL_NFC_DEACTIVATE_SLEEP	Deactivate and go to SELECT
RFAL_NFC_DEACTIVATE_DISCOVERY	Deactivate and restart DISCOVERY

◆ rfalNfcDevType

enum **rfalNfcDevType**

Device type

Enumerator	
RFAL_NFC_LISTEN_TYPE_NFCA	NFC-A Listener device type
RFAL_NFC_LISTEN_TYPE_NFCB	NFC-B Listener device type
RFAL_NFC_LISTEN_TYPE_NFCF	NFC-F Listener device type
RFAL_NFC_LISTEN_TYPE_NFCV	NFC-V Listener device type
RFAL_NFC_LISTEN_TYPE_ST25TB	ST25TB Listener device type
RFAL_NFC_LISTEN_TYPE_AP2P	AP2P Listener device type
RFAL_NFC_LISTEN_TYPE_PROP	Proprietary Listen dev type
RFAL_NFC_POLL_TYPE_NFCA	NFC-A Poller device type
RFAL_NFC_POLL_TYPE_NFCB	NFC-B Poller device type
RFAL_NFC_POLL_TYPE_NFCF	NFC-F Poller device type
RFAL_NFC_POLL_TYPE_NFCV	NFC-V Poller device type
RFAL_NFC_POLL_TYPE_AP2P	AP2P Poller device type

◆ rfalNfcRfInterface

enum **rfalNfcRfInterface**

Device interface

Enumerator	
RFAL_NFC_INTERFACE_RF	RF Frame interface
RFAL_NFC_INTERFACE_ISODEP	ISO-DEP interface
RFAL_NFC_INTERFACE_NFCDEP	NFC-DEP interface

◆ rfalNfcState

enum **rfalNfcState**

Main state

Enumerator	
RFAL_NFC_STATE_NOTINIT	Not Initialized state
RFAL_NFC_STATE_IDLE	Initialize state
RFAL_NFC_STATE_START_DISCOVERY	Start Discovery loop state
RFAL_NFC_STATE_WAKEUP_MODE	Wake-Up state
RFAL_NFC_STATE_POLL_TECHDETECT	Technology Detection state
RFAL_NFC_STATE_POLL_COLAVOIDANCE	Collision Avoidance state
RFAL_NFC_STATE_POLL_SELECT	Wait for Selection state
RFAL_NFC_STATE_POLL_ACTIVATION	Activation state
RFAL_NFC_STATE_LISTEN_TECHDETECT	Listen Tech Detect
RFAL_NFC_STATE_LISTEN_COLAVOIDANCE	Listen Collision Avoidance
RFAL_NFC_STATE_LISTEN_ACTIVATION	Listen Activation state
RFAL_NFC_STATE_LISTEN_SLEEP	Listen Sleep state
RFAL_NFC_STATE_ACTIVATED	Activated state
RFAL_NFC_STATE_DATAEXCHANGE	Data Exchange Start state
RFAL_NFC_STATE_DATAEXCHANGE_DONE	Data Exchange terminated
RFAL_NFC_STATE_DEACTIVATION	Deactivation state

Function Documentation

◆ rfalNfcDataExchangeGetStatus()

ReturnCode rfalNfcDataExchangeGetStatus (void)

Gets current Data Exchange status

Returns

RFAL_ERR_NONE : Transceive done with no error

RFAL_ERR_BUSY : Transceive ongoing

RFAL_ERR_AGAIN : received one chaining block, copy received data and continue to call this method to retrieve the remaining blocks

RFAL_ERR_XXXX : Error occurred

RFAL_ERR_TIMEOUT : No response

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PAR : Parity error detected

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_LINK_LOSS : Link Loss - External Field is Off

RFAL_ERR_RF_COLLISION : Collision detected

RFAL_ERR_IO : Internal error

◆ rfalNfcDataExchangeStart()

```
ReturnCode rfalNfcDataExchangeStart ( uint8_t *     txData,
                                      uint16_t      txDataLen,
                                      uint8_t **    rxData,
                                      uint16_t **   rvdLen,
                                      uint32_t      fwt
                                    )
```

After a device has been activated, it starts a data exchange. It handles automatically which interface/protocol to be used and acts accordingly.

In Listen mode the first frame/data shall be sent by the Reader/Initiator therefore this method must be called first with txDataLen set to zero to retrieve the rxData and rcvLen locations.

Parameters

[in]	**txData**	: data to be transmitted
[in]	**txDataLen**	: size of the data to be transmitted (in bits or bytes - see below)
[out]	**rxData**	: location of the received data after operation is completed
[out]	**rvdLen**	: location of the length of the received data (in bits or bytes - see below)
[in]	**fwt**	: FWT to be used in case of RF interface. If ISO-DEP or NFC-DEP interface is used, this will be ignored

Warning

In order to support a wider range of protocols, when RF interface is used the lengths are in number of bits (not bytes). Therefore both input txDataLen and output rvdLen refer to bits. If ISO-DEP or NFC-DEP interface is used those are expressed in number of bytes.

Returns

RFAL_ERR_WRONG_STATE : Incorrect state for this operation

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_NONE : No error

◆ rfalNfcDeactivate()

ReturnCode rfalNfcDeactivate (**rfalNfcDeactivateType** deactType)

It triggers the deactivation procedure to terminate communications with remote device. In case the deactivation type is RFAL_NFC_DEACTIVATE_SLEEP the field is kept On and device selection shall follow. Otherwise the field will be turned Off.

Warning

In case the deactivation type is RFAL_NFC_DEACTIVATE_IDLE the deactivation procedure is executed immediately and in a blocking manner

Parameters

[in] **deactType** : Type of deactivation to be performed

Returns

RFAL_ERR_WRONG_STATE : Incorrect state for this operation

RFAL_ERR_NONE : No error

◆ rfalNfcDiscover()

ReturnCode rfalNfcDiscover (const **rfalNfcDiscoverParam** * disParams)

It set the device in Discovery state. In discovery it will Poll and/or Listen for the technologies configured, and perform Wake-up mode if configured to do so.

The device list passed on disParams must not be empty. The number of devices on the list is indicated by the devLimit and shall be at >= 1.

Parameters

[in] **disParams** : discovery configuration parameters

Returns

RFAL_ERR_WRONG_STATE : Incorrect state for this operation

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_NONE : No error

◆ rfalNfcGetActiveDevice()

```
ReturnCode rfalNfcGetActiveDevice ( rfalNfcDevice ** dev )
```

It returns the location of the device current Active device

Parameters

[out] **dev** : device info location

Returns

RFAL_ERR_WRONG_STATE : Incorrect state for this operation No device activated

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_NONE : No error

◆ rfalNfcGetDevicesFound()

```
ReturnCode rfalNfcGetDevicesFound ( rfalNfcDevice ** devList,
                                    uint8_t *        devCnt
                                  )
```

It returns the location of the device list and the number of devices found.

Parameters

[out] **devList** : device list location

[out] **devCnt** : number of devices found

Returns

RFAL_ERR_WRONG_STATE : Incorrect state for this operation Discovery still ongoing

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_NONE : No error

◆ rfalNfcGetState()

rfalNfcState rfalNfcGetState (void)

It returns the current state

Returns

rfalNfcState : the current state

◆ rfalNfcInitialize()

ReturnCode rfalNfcInitialize (void)

It initializes this module and its dependencies

Returns

RFAL_ERR_WRONG_STATE : Incorrect state for this operation

RFAL_ERR_IO : Generic internal error

RFAL_ERR_NONE : No error

◆ rfalNfcSelect()

ReturnCode rfalNfcSelect (uint8_t devIdx)

It selects the device to be activated. It shall be called when more than one device has been identified to indicate which device shall be actived

Parameters

`[in]` **devIdx** : device index to be activated

Returns

RFAL_ERR_WRONG_STATE : Incorrect state for this operation Not in select state

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_NONE : No error

◆ rfalNfcWorker()

void rfalNfcWorker (void)

It runs the internal state machine and runs the RFAL RF worker.

RFAL

Main Page

Modules

Files

Examples

Data Structures

Data Structure Index

Data Fields

rfalNfcDevice Struct Reference


```
#include <rfal_nfc.h>
```

Collaboration diagram for rfalNfcDevice:

[legend]

Data Fields

rfalNfcDevType	**type**
union { **rfalNfcaListenDevice** **nfca**	
rfalNfcbListenDevice **nfcb**	
rfalNfcfListenDevice **nfcf**	
rfalNfcvListenDevice **nfcv**	
rfalSt25tbListenDevice **st25tb**	
}	**dev**
uint8_t *	**nfcid**
uint8_t	**nfcidLen**
rfalNfcRfInterface	**rfInterface**
union { **rfalIsoDepDevice** **isoDep**	

rfalNfcDepDevice nfcDep	
}	**proto**

Detailed Description

Device struct containing all its details

Field Documentation

◆

union { ... } rfalNfcDevice::dev

Device's instance

◆ isoDep

rfalIsoDepDevice rfalNfcDevice::isoDep

ISO-DEP instance

◆ nfca

rfalNfcaListenDevice rfalNfcDevice::nfca

NFC-A Listen Device instance

◆ nfcb

rfalNfcbListenDevice rfalNfcDevice::nfcb

NFC-B Listen Device instance

◆ nfcDep

rfalNfcDepDevice rfalNfcDevice::nfcDep

NFC-DEP instance

◆ nfcf

rfalNfcfListenDevice rfalNfcDevice::nfcf

NFC-F Listen Device instance

◆ nfcid

uint8_t* rfalNfcDevice::nfcid

Device's NFCID

◆ nfcidLen

uint8_t rfalNfcDevice::nfcidLen

Device's NFCID length

◆ nfcv

rfalNfcvListenDevice rfalNfcDevice::nfcv

NFC-V Listen Device instance

◆

union { ... } rfalNfcDevice::proto

Device's protocol

◆ rfInterface

rfalNfcRfInterface rfalNfcDevice::rfInterface

Device's interface

◆ st25tb

rfalSt25tbListenDevice rfalNfcDevice::st25tb

ST25TB Listen Device instance

◆ type

rfalNfcDevType rfalNfcDevice::type

Device's type

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

- **rfal_nfc.h**

---

rfalNfcPropCallbacks Struct Reference


```
#include <rfal_nfc.h>
```

Data Fields

rfalNfcPropCallback	**rfalNfcpPollerInitialize**
rfalNfcPropCallback	**rfalNfcpPollerTechnologyDetection**
rfalNfcPropCallback	**rfalNfcpPollerStartCollisionResolution**
rfalNfcPropCallback	**rfalNfcpPollerGetCollisionResolutionStatus**
rfalNfcPropCallback	**rfalNfcpStartActivation**
rfalNfcPropCallback	**rfalNfcpGetActivationStatus**

Detailed Description

Struct that holds the Proprietary NFC callbacks

Field Documentation

◆ rfalNfcpGetActivationStatus

rfalNfcPropCallback rfalNfcPropCallbacks::rfalNfcpGetActivationStatus

Prorietary NFC Get Activation status callback

◆ rfalNfcpPollerGetCollisionResolutionStatus

rfalNfcPropCallback rfalNfcPropCallbacks::rfalNfcpPollerGetCollisionResolutionStatus

Prorietary NFC Get Collision Resolution status callback

◆ rfalNfcpPollerInitialize

rfalNfcPropCallback rfalNfcPropCallbacks::rfalNfcpPollerInitialize

Prorietary NFC Initialization callback

◆ rfalNfcpPollerStartCollisionResolution

rfalNfcPropCallback rfalNfcPropCallbacks::rfalNfcpPollerStartCollisionResolution

Prorietary NFC Start Collision Resolution callback

◆ rfalNfcpPollerTechnologyDetection

rfalNfcPropCallback rfalNfcPropCallbacks::rfalNfcpPollerTechnologyDetection

Prorietary NFC Technoly Detection callback

◆ rfalNfcpStartActivation

rfalNfcPropCallback rfalNfcPropCallbacks::rfalNfcpStartActivation

Prorietary NFC Start Activation callback

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

- **rfal_nfc.h**

rfalNfcDiscoverParam Struct Reference


```
#include <rfal_nfc.h>
```

Collaboration diagram for rfalNfcDiscoverParam:

[legend]

Data Fields

rfalComplianceMode compMode

uint16_t	**techs2Find**
uint16_t	**techs2Bail**
uint16_t	**totalDuration**
uint8_t	**devLimit**
rfalBitRate	**maxBR**
rfalBitRate	**nfcfBR**
uint8_t	**nfcid3** [**RFAL_NFCDEP_NFCID3_LEN**]
uint8_t	**GB** [**RFAL_NFCDEP_GB_MAX_LEN**]
uint8_t	**GBLen**
rfalBitRate	**ap2pBR**
bool	**p2pNfcaPrio**
rfalNfcPropCallbacks	**propNfc**
rfalIsoDepFSxI	**isoDepFS**
uint8_t	**nfcDepLR**
rfalLmConfPA	**lmConfigPA**
rfalLmConfPF	**lmConfigPF**
void(*	**notifyCb**)(**rfalNfcState** st)
bool	**wakeupEnabled**
bool	**wakeupConfigDefault**
rfalWakeUpConfig	**wakeupConfig**
bool	**wakeupPollBefore**

uint16_t **wakeupNPolls**

Detailed Description

Discovery parameters

Field Documentation

◆ ap2pBR

rfalBitRate rfalNfcDiscoverParam::ap2pBR

Bit rate to poll for AP2P NCI 2.1 Table 31

◆ compMode

rfalComplianceMode rfalNfcDiscoverParam::compMode

Compliancy mode to be used

◆ devLimit

uint8_t rfalNfcDiscoverParam::devLimit

Max number of devices Activity 2.1 Table 11

◆ GB

uint8_t rfalNfcDiscoverParam::GB[**RFAL_NFCDEP_GB_MAX_LEN**]

General bytes to be used on the ATR-REQ NCI 2.1 Table 29

◆ GBLen

uint8_t rfalNfcDiscoverParam::GBLen

Length of the General Bytes NCI 2.1 Table 29

◆ isoDepFS

rfalIsoDepFSxI rfalNfcDiscoverParam::isoDepFS

ISO-DEP Poller announced maximum frame size Digital 2.2 Table 60

◆ lmConfigPA

rfalLmConfPA rfalNfcDiscoverParam::lmConfigPA

Configuration for Passive Listen mode NFC-A

◆ lmConfigPF

rfalLmConfPF rfalNfcDiscoverParam::lmConfigPF

Configuration for Passive Listen mode NFC-A

◆ maxBR

rfalBitRate rfalNfcDiscoverParam::maxBR

Max Bit rate to be used NCI 2.1 Table 28

◆ nfcDepLR

uint8_t rfalNfcDiscoverParam::nfcDepLR

NFC-DEP Poller & Listener maximum frame size Digital 2.2 Table 90

◆ nfcfBR

rfalBitRate rfalNfcDiscoverParam::nfcfBR

Bit rate to poll for NFC-F NCI 2.1 Table 27

◆ nfcid3

uint8_t rfalNfcDiscoverParam::nfcid3[**RFAL_NFCDEP_NFCID3_LEN**]

NFCID3 to be used on the ATR_REQ/ATR_RES

◆ notifyCb

void(* rfalNfcDiscoverParam::notifyCb) (**rfalNfcState** st)

Callback to Notify upper layer

◆ p2pNfcaPrio

bool rfalNfcDiscoverParam::p2pNfcaPrio

NFC-A P2P (true) or ISO14443-4/T4T (false) priority

◆ propNfc

rfalNfcPropCallbacks rfalNfcDiscoverParam::propNfc

Proprietary Technlogy callbacks

◆ techs2Bail

uint16_t rfalNfcDiscoverParam::techs2Bail

Bail-out after certain NFC technologies

◆ techs2Find

uint16_t rfalNfcDiscoverParam::techs2Find

Technologies to search for

◆ totalDuration

uint16_t rfalNfcDiscoverParam::totalDuration

Duration of a whole Poll + Listen cycle NCI 2.1 Table 46

◆ wakeupConfig

rfalWakeUpConfig rfalNfcDiscoverParam::wakeupConfig

Wake-Up mode configuration

◆ wakeupConfigDefault

bool rfalNfcDiscoverParam::wakeupConfigDefault

Wake-Up mode default configuration

◆ wakeupEnabled

bool rfalNfcDiscoverParam::wakeupEnabled

Enable Wake-Up mode before polling

◆ wakeupNPolls

uint16_t rfalNfcDiscoverParam::wakeupNPolls

Number of polling cycles before|after entering Wake-up

◆ wakeupPollBefore

bool rfalNfcDiscoverParam::wakeupPollBefore

Flag to Poll wakeupNPolls times before entering Wake-up

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

- rfal_nfc.h

rfalNfcBuffer Union Reference


```
#include <rfal_nfc.h>
```

Collaboration diagram for rfalNfcBuffer:

[legend]

Data Fields

uint8_t	**rfBuf** [**RFAL_FEATURE_NFC_RF_BUF_LEN**]
rfalIsoDepApduBufFormat	**isoDepBuf**
rfalNfcDepPduBufFormat	**nfcDepBuf**

Detailed Description

Buffer union, only one interface is used at a time

Field Documentation

◆ isoDepBuf

rfalIsoDepApduBufFormat rfalNfcBuffer::isoDepBuf

ISO-DEP buffer format (with header/prologue)

◆ nfcDepBuf

rfalNfcDepPduBufFormat rfalNfcBuffer::nfcDepBuf

NFC-DEP buffer format (with header/prologue)

◆ rfBuf

uint8_t rfalNfcBuffer::rfBuf[**RFAL_FEATURE_NFC_RF_BUF_LEN**]

RF buffer

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

- **rfal_nfc.h**

RFAL

RFAL-AL

RFAL

RFAL Abstraction Layer. More...

Collaboration diagram for RFAL-AL:

Modules

ISO-DEP
RFAL ISO-DEP Module.

NFC-A
RFAL NFC-A Module.

NFC-B
RFAL NFC-B Module.

NFC-DEP
RFAL NFC-DEP Module.

NFC-F
RFAL NFC-F Module

NFC-V
RFAL NFC-V Module.

ST25TB
RFAL ST25TB Module.

ST25xV
RFAL ST25xV Module.

T1T
RFAL T1T Module.

T2T
RFAL T2T Module.

T4T
RFAL T4T Module.

Detailed Description

RFAL

Main Page

Data Structures

Files

ISO-DEP

RFAL ISO-DEP Module. More...

Collaboration diagram for ISO-DEP:

Data Structures

struct	**rfaIsoDepRats**
struct	**rfaIsoDepAts**
struct	**rfaIsoDepPpsReq**
struct	**rfaIsoDepPpsRes**
struct	**rfaIsoDepAttribCmd**
struct	**rfaIsoDepAttribRes**
struct	**rfaIsoDepSParameter**
union	**rfaIsoDepActivation**
struct	**rfaIsoDepInfo**
struct	**rfaIsoDepDevice**
struct	**rfaIsoDepAttribResParam**
struct	**rfaIsoDepAtsParam**
struct	**rfaIsoDepBufFormat**
struct	**rfaIsoDepApduBufFormat**

struct **rfalIsoDepListenActvParam**

struct **rfalIsoDepTxRxParam**

struct **rfalIsoDepApduTxRxParam**

Macros

#define **RFAL_FEATURE_ISO_DEP_IBLOCK_MAX_LEN** (1U)

#define **RFAL_FEATURE_ISO_DEP_APDU_MAX_LEN** (1U)

#define **RFAL_ISODEP_PROLOGUE_SIZE** (3U)

#define **RFAL_ISODEP_PCB_LEN** (1U)

#define **RFAL_ISODEP_DID_LEN** (1U)

#define **RFAL_ISODEP_NAD_LEN** (1U)

#define **RFAL_ISODEP_NO_DID** (0x10U)

#define **RFAL_ISODEP_NO_NAD** (0xFFU)

#define **RFAL_ISODEP_FWI_MASK** (0xF0U)

#define **RFAL_ISODEP_FWI_SHIFT** (4U)

#define **RFAL_ISODEP_FWI_DEFAULT** (4U)

#define **RFAL_ISODEP_ADV_FEATURE** (0x0FU)

#define **RFAL_ISODEP_DID_MAX** (14U)

#define **RFAL_ISODEP_BRI_MASK** (0x07U)

#define **RFAL_ISODEP_BSI_MASK** (0x70U)

#define **RFAL_ISODEP_SAME_BITRATE_MASK** (0x80U)

#define **RFAL_ISODEP_BITRATE_RFU_MASK** (0x08U)

```
#define RFAL_ISODEP_MAX_FWT ((uint32_t)1U<<26)
#define RFAL_ISODEP_FSDI_DEFAULT RFAL_ISODEP_FSXI_256
#define RFAL_ISODEP_FSX_KEEP (0xFFU)
#define RFAL_ISODEP_DEFAULT_FSCI RFAL_ISODEP_FSXI_256
#define RFAL_ISODEP_DEFAULT_FSC RFAL_ISODEP_FSX_256
#define RFAL_ISODEP_DEFAULT_SFGI (0U)
#define RFAL_ISODEP_DEFAULT_FWI (8U)
#define RFAL_ISODEP_APDU_MAX_LEN RFAL_ISODEP_FSX_1024
#define RFAL_ISODEP_ATTRIB_RES_MBLI_NO_INFO (0x00U)
#define RFAL_ISODEP_ATTRIB_REQ_PARAM1_DEFAULT (0x00U)
#define RFAL_ISODEP_ATTRIB_HLINFO_LEN (32U)
#define RFAL_ISODEP_ATS_HB_MAX_LEN (15U)
#define RFAL_ISODEP_ATTRIB_REQ_MIN_LEN (9U)
#define RFAL_ISODEP_ATTRIB_RES_MIN_LEN (1U)
#define RFAL_ISODEP_SPARAM_VALUES_MAX_LEN (16U)
#define RFAL_ISODEP_SPARAM_TAG_BLOCKINFO (0xA0U)
#define RFAL_ISODEP_SPARAM_TAG_BRREQ (0xA1U)
#define RFAL_ISODEP_SPARAM_TAG_BRIND (0xA2U)
#define RFAL_ISODEP_SPARAM_TAG_BRACT (0xA3U)
#define RFAL_ISODEP_SPARAM_TAG_BRACK (0xA4U)
#define RFAL_ISODEP_SPARAM_TAG_SUP_PCD2PICC (0x80U)
```

#define **RFAL_ISODEP_SPARAM_TAG_SUP_PICC2PCD** (0x81U)

#define **RFAL_ISODEP_SPARAM_TAG_SUP_FRAME** (0x82U)

#define **RFAL_ISODEP_SPARAM_TAG_SEL_PCD2PICC** (0x83U)

#define **RFAL_ISODEP_SPARAM_TAG_SEL_PICC2PCD** (0x84U)

#define **RFAL_ISODEP_SPARAM_TAG_SEL_FRAME** (0x85U)

#define **RFAL_ISODEP_SPARAM_TAG_LEN** (1)

#define **RFAL_ISODEP_SPARAM_TAG_BRREQ_LEN** (0U)

#define **RFAL_ISODEP_SPARAM_TAG_PICC2PCD_LEN** (2U)

#define **RFAL_ISODEP_SPARAM_TAG_PCD2PICC_LEN** (2U)

#define **RFAL_ISODEP_SPARAM_TAG_BRACK_LEN** (0U)

#define **RFAL_ISODEP_ATS_TA_DPL_212** (0x01U)

#define **RFAL_ISODEP_ATS_TA_DPL_424** (0x02U)

#define **RFAL_ISODEP_ATS_TA_DPL_848** (0x04U)

#define **RFAL_ISODEP_ATS_TA_DLP_212** (0x10U)

#define **RFAL_ISODEP_ATS_TA_DLP_424** (0x20U)

#define **RFAL_ISODEP_ATS_TA_DLP_848** (0x40U)

#define **RFAL_ISODEP_ATS_TA_SAME_D** (0x80U)

#define **RFAL_ISODEP_ATS_TB_FWI_MASK** (0xF0U)

#define **RFAL_ISODEP_ATS_TB_SFGI_MASK** (0x0FU)

#define **RFAL_ISODEP_ATS_T0_TA_PRESENCE_MASK** (0x10U)

#define **RFAL_ISODEP_ATS_T0_TB_PRESENCE_MASK** (0x20U)

```
#define RFAL_ISODEP_ATS_T0_TC_PRESENCE_MASK (0x40U)
```

```
#define RFAL_ISODEP_ATS_T0_FSCI_MASK (0x0FU)
```

```
#define RFAL_ISODEP_ATS_T0_OFFSET (0x01U)
```

```
#define RFAL_ISODEP_ATS_TB_FWI_SHIFT (4U)
```

```
#define RFAL_ISODEP_ATS_FWI_MASK (0x0FU)
```

```
#define RFAL_ISODEP_ATS_SFGI_MASK (0x0FU)
```

```
#define RFAL_ISODEP_MAX_I_RETRYS (2U)
```

```
#define RFAL_ISODEP_MAX_R_RETRYS (3U)
```

```
#define RFAL_ISODEP_MAX_WTX_NACK_RETRYS (3U)
```

```
#define RFAL_ISODEP_MAX_WTX_RETRYS (20U)
```

```
#define RFAL_ISODEP_MAX_WTX_RETRYS_ULTD (255U)
```

```
#define RFAL_ISODEP_MAX_DSL_RETRYS (0U)
```

```
#define RFAL_ISODEP_RATS_RETRIES (1U)
```

Enumerations

```
enum rfalIsoDepFSxI {
  RFAL_ISODEP_FSXI_16 = 0 , RFAL_ISODEP_FSXI_24 = 1 , RFAL_ISODEP_FSXI_32 = 2 ,
RFAL_ISODEP_FSXI_40 = 3 ,
  RFAL_ISODEP_FSXI_48 = 4 , RFAL_ISODEP_FSXI_64 = 5 , RFAL_ISODEP_FSXI_96 = 6 ,
RFAL_ISODEP_FSXI_128 = 7 ,
  RFAL_ISODEP_FSXI_256 = 8 , RFAL_ISODEP_FSXI_512 = 9 , RFAL_ISODEP_FSXI_1024 = 10 ,
RFAL_ISODEP_FSXI_2048 = 11 ,
  RFAL_ISODEP_FSXI_4096 = 12
}
```

```
enum rfalIsoDepFSx {
  RFAL_ISODEP_FSX_16 = 16 , RFAL_ISODEP_FSX_24 = 24 , RFAL_ISODEP_FSX_32 = 32 ,
RFAL_ISODEP_FSX_40 = 40 ,
```

RFAL_ISODEP_FSX_48 = 48 , **RFAL_ISODEP_FSX_64** = 64 , **RFAL_ISODEP_FSX_96** = 96 , **RFAL_ISODEP_FSX_128** = 128 , **RFAL_ISODEP_FSX_256** = 256 , **RFAL_ISODEP_FSX_512** = 512 , **RFAL_ISODEP_FSX_1024** = 1024 , **RFAL_ISODEP_FSX_2048** = 2048 , **RFAL_ISODEP_FSX_4096** = 4096
}

Functions

void **rfaIIsoDepInitialize** (void)
Initialize the ISO-DEP protocol. More...

void **rfaIIsoDepInitializeWithParams** (**rfalComplianceMode** compMode, uint8_t maxRetriesR, uint8_t maxRetriesSnWTX, uint8_t maxRetriesSWTX, uint8_t maxRetriesSDSL, uint8_t maxRetriesI, uint8_t maxRetriesRATS)
Initialize the ISO-DEP protocol. More...

uint16_t **rfaIIsoDepFSxI2FSx** (uint8_t FSxI)
FSxI to FSx. More...

uint32_t **rfaIIsoDepFWI2FWT** (uint8_t fwi)
FWI to FWT. More...

bool **rfaIIsoDepIsRats** (const uint8_t *buf, uint8_t bufLen)
Check if the buffer data contains a valid RATS command. More...

bool **rfaIIsoDepIsAttrib** (const uint8_t *buf, uint8_t bufLen)
Check if the buffer data contains a valid ATTRIB command. More...

ReturnCode **rfaIIsoDepListenStartActivation** (**rfaIIsoDepAtsParam** *atsParam, const **rfaIIsoDepAttribResParam** *attribResParam, const uint8_t *buf, uint16_t bufLen, **rfaIIsoDepListenActvParam** actParam)
Start Listen Activation Handling. More...

ReturnCode **rfaIIsoDepListenGetActivationStatus** (void)
Get the current Activation Status. More...

uint16_t **rfaIIsoDepGetMaxInfLen** (void)
Get the ISO-DEP Communication Information. More...

ReturnCode	**rfaIsoDepStartTransceive** (**rfaIsoDepTxRxParam** param) ISO-DEP Start Transceive. More...
ReturnCode	**rfaIsoDepGetTransceiveStatus** (void) Get the Transceive status. More...
ReturnCode	**rfaIsoDepStartApduTransceive** (**rfaIsoDepApduTxRxParam** param) ISO-DEP Start APDU Transceive. More...
ReturnCode	**rfaIsoDepGetApduTransceiveStatus** (void) Get the APDU Transceive status. More...
ReturnCode	**rfaIsoDepRATS** (**rfaIsoDepFSxI** FSDI, uint8_t DID, **rfaIsoDepAts** *ats, uint8_t *atsLen) ISO-DEP Send RATS. More...
ReturnCode	**rfaIsoDepPPS** (uint8_t DID, **rfaBitRate** DSI, **rfaBitRate** DRI, **rfaIsoDepPpsRes** *ppsRes) ISO-DEP Send PPS. More...
ReturnCode	**rfaIsoDepATTRIB** (const uint8_t *nfcid0, uint8_t PARAM1, **rfaBitRate** DSI, **rfaBitRate** DRI, **rfaIsoDepFSxI** FSDI, uint8_t PARAM3, uint8_t DID, const uint8_t *HLInfo, uint8_t HLInfoLen, uint32_t fwt, **rfaIsoDepAttribRes** *attribRes, uint8_t *attribResLen) ISO-DEP Send ATTRIB. More...
ReturnCode	**rfaIsoDepDeselect** (void) Deselect PICC. More...
ReturnCode	**rfaIsoDepStartDeselect** (void) Start Deselect. More...
ReturnCode	**rfaIsoDepGetDeselectStatus** (void) Deselect Get Status. More...
ReturnCode	**rfaIsoDepPollAHandleActivation** (**rfaIsoDepFSxI** FSDI, uint8_t DID, **rfaBitRate** maxBR, **rfaIsoDepDevice** *rfaIsoDepDev) ISO-DEP Poller Handle NFC-A Activation. More...

ReturnCode	**rfalIsoDepPollBHandleActivation** (**rfalIsoDepFSxI** FSDI, uint8_t DID, **rfalBitRate** maxBR, uint8_t PARAM1, const **rfalNfcbListenDevice** *nfcbDev, const uint8_t *HLInfo, uint8_t HLInfoLen, **rfalIsoDepDevice** *rfalIsoDepDev) ISO-DEP Poller Handle NFC-B Activation. More...
ReturnCode	**rfalIsoDepPollHandleSParameters** (**rfalIsoDepDevice** *rfalIsoDepDev, **rfalBitRate** maxTxBR, **rfalBitRate** maxRxBR) ISO-DEP Poller Handle S(Parameters) More...
ReturnCode	**rfalIsoDepPollAStartActivation** (**rfalIsoDepFSxI** FSDI, uint8_t DID, **rfalBitRate** maxBR, **rfalIsoDepDevice** *rfalIsoDepDev) ISO-DEP Poller Start NFC-A Activation. More...
ReturnCode	**rfalIsoDepPollAGetActivationStatus** (void) ISO-DEP Poller Get NFC-A Activation Status. More...
ReturnCode	**rfalIsoDepPollBStartActivation** (**rfalIsoDepFSxI** FSDI, uint8_t DID, **rfalBitRate** maxBR, uint8_t PARAM1, const **rfalNfcbListenDevice** *nfcbDev, const uint8_t *HLInfo, uint8_t HLInfoLen, **rfalIsoDepDevice** *rfalIsoDepDev) ISO-DEP Poller Start NFC-B Activation. More...
ReturnCode	**rfalIsoDepPollBGetActivationStatus** (void) ISO-DEP Poller Get NFC-B Activation Status. More...

Detailed Description

Macro Definition Documentation

◆ RFAL_FEATURE_ISO_DEP_APDU_MAX_LEN

```
#define RFAL_FEATURE_ISO_DEP_APDU_MAX_LEN   (1U)
```

ISO-DEP APDU max length, set to "none"

◆ RFAL_FEATURE_ISO_DEP_IBLOCK_MAX_LEN

#define RFAL_FEATURE_ISO_DEP_IBLOCK_MAX_LEN (1U)

ISO-DEP I-Block max length, set to "none"

◆ RFAL_ISODEP_ADV_FEATURE

#define RFAL_ISODEP_ADV_FEATURE (0x0FU)

Indicate 256 Bytes FSD and Advanc Proto Feature support:NAD & DID

◆ RFAL_ISODEP_APDU_MAX_LEN

#define RFAL_ISODEP_APDU_MAX_LEN **RFAL_ISODEP_FSX_1024**

Max APDU length

◆ RFAL_ISODEP_ATS_FWI_MASK

#define RFAL_ISODEP_ATS_FWI_MASK (0x0FU)

ATS TB's FWI mask Digital 2.0 14.6.2

◆ RFAL_ISODEP_ATS_HB_MAX_LEN

#define RFAL_ISODEP_ATS_HB_MAX_LEN (15U)

Maximum length of Historical Bytes Digital 1.1 13.6.2.23

◆ RFAL_ISODEP_ATS_SFGI_MASK

#define RFAL_ISODEP_ATS_SFGI_MASK (0x0FU)

ATS TB's SFGI mask Digital 2.0 14.6.2

◆ RFAL_ISODEP_ATS_T0_FSCI_MASK

#define RFAL_ISODEP_ATS_T0_FSCI_MASK (0x0FU)

Mask bit for FSCI presence

◆ RFAL_ISODEP_ATS_T0_OFFSET

```
#define RFAL_ISODEP_ATS_T0_OFFSET   (0x01U)
```

Offset of T0 in ATS Response

◆ RFAL_ISODEP_ATS_T0_TA_PRESENCE_MASK

```
#define RFAL_ISODEP_ATS_T0_TA_PRESENCE_MASK   (0x10U)
```

Mask bit for TA presence

◆ RFAL_ISODEP_ATS_T0_TB_PRESENCE_MASK

```
#define RFAL_ISODEP_ATS_T0_TB_PRESENCE_MASK   (0x20U)
```

Mask bit for TB presence

◆ RFAL_ISODEP_ATS_T0_TC_PRESENCE_MASK

```
#define RFAL_ISODEP_ATS_T0_TC_PRESENCE_MASK   (0x40U)
```

Mask bit for TC presence

◆ RFAL_ISODEP_ATS_TA_DLP_212

```
#define RFAL_ISODEP_ATS_TA_DLP_212   (0x10U)
```

ATS TA DSI 212 kbps support bit mask

◆ RFAL_ISODEP_ATS_TA_DLP_424

```
#define RFAL_ISODEP_ATS_TA_DLP_424   (0x20U)
```

ATS TA DRI 424 kbps support bit mask

◆ RFAL_ISODEP_ATS_TA_DLP_848

```
#define RFAL_ISODEP_ATS_TA_DLP_848   (0x40U)
```

ATS TA DRI 848 kbps support bit mask

◆ RFAL_ISODEP_ATS_TA_DPL_212

```
#define RFAL_ISODEP_ATS_TA_DPL_212   (0x01U)
```

ATS TA DSI 212 kbps support bit mask

◆ RFAL_ISODEP_ATS_TA_DPL_424

```
#define RFAL_ISODEP_ATS_TA_DPL_424   (0x02U)
```

ATS TA DSI 424 kbps support bit mask

◆ RFAL_ISODEP_ATS_TA_DPL_848

```
#define RFAL_ISODEP_ATS_TA_DPL_848   (0x04U)
```

ATS TA DSI 848 kbps support bit mask

◆ RFAL_ISODEP_ATS_TA_SAME_D

```
#define RFAL_ISODEP_ATS_TA_SAME_D   (0x80U)
```

ATS TA same bit both directions bit mask

◆ RFAL_ISODEP_ATS_TB_FWI_MASK

```
#define RFAL_ISODEP_ATS_TB_FWI_MASK   (0xF0U)
```

Mask bits for FWI (Frame Waiting Integer) in TB byte

◆ RFAL_ISODEP_ATS_TB_FWI_SHIFT

```
#define RFAL_ISODEP_ATS_TB_FWI_SHIFT   (4U)
```

ATS TB's FWI shift Digital 2.0 14.6.2

◆ RFAL_ISODEP_ATS_TB_SFGI_MASK

```
#define RFAL_ISODEP_ATS_TB_SFGI_MASK   (0x0FU)
```

Mask bits for SFGI (Start-Up Frame Guard Integer) in TB byte

◆ RFAL_ISODEP_ATTRIB_HLINFO_LEN

```
#define RFAL_ISODEP_ATTRIB_HLINFO_LEN   (32U)
```

Maximum Size of Higher Layer Information

◆ RFAL_ISODEP_ATTRIB_REQ_MIN_LEN

```
#define RFAL_ISODEP_ATTRIB_REQ_MIN_LEN   (9U)
```

Minimum Length of ATTRIB_REQ command

◆ RFAL_ISODEP_ATTRIB_REQ_PARAM1_DEFAULT

```
#define RFAL_ISODEP_ATTRIB_REQ_PARAM1_DEFAULT   (0x00U)
```

Default values of Param 1 of ATTRIB_REQ Digital 1.0 12.6.1.3-5

◆ RFAL_ISODEP_ATTRIB_RES_MBLI_NO_INFO

```
#define RFAL_ISODEP_ATTRIB_RES_MBLI_NO_INFO   (0x00U)
```

MBLI indicating no information on its internal input buffer size

◆ RFAL_ISODEP_ATTRIB_RES_MIN_LEN

#define RFAL_ISODEP_ATTRIB_RES_MIN_LEN (1U)

Minimum Length of ATTRIB_RES response

◆ RFAL_ISODEP_BITRATE_RFU_MASK

#define RFAL_ISODEP_BITRATE_RFU_MASK (0x08U)

Mask bit for RFU

◆ RFAL_ISODEP_BRI_MASK

#define RFAL_ISODEP_BRI_MASK (0x07U)

Mask bits for Poll to Listen Send bitrate

◆ RFAL_ISODEP_BSI_MASK

#define RFAL_ISODEP_BSI_MASK (0x70U)

Mask bits for Listen to Poll Send bitrate

◆ RFAL_ISODEP_DEFAULT_FSC

#define RFAL_ISODEP_DEFAULT_FSC **RFAL_ISODEP_FSX_256**

FSC default value (aligned RFAL_ISODEP_DEFAULT_FSCI)

◆ RFAL_ISODEP_DEFAULT_FSCI

#define RFAL_ISODEP_DEFAULT_FSCI **RFAL_ISODEP_FSXI_256**

FSCI default value to be used in Listen Mode

◆ RFAL_ISODEP_DEFAULT_FWI

#define RFAL_ISODEP_DEFAULT_FWI (8U)

Default Listener FWI (Max) Digital 2.0 B7 & B3

◆ RFAL_ISODEP_DEFAULT_SFGI

#define RFAL_ISODEP_DEFAULT_SFGI (0U)

SFGI Default value to be used in Listen Mode

◆ RFAL_ISODEP_DID_LEN

#define RFAL_ISODEP_DID_LEN (1U)

DID length

◆ RFAL_ISODEP_DID_MAX

#define RFAL_ISODEP_DID_MAX (14U)

Maximum DID value

◆ RFAL_ISODEP_FSDI_DEFAULT

#define RFAL_ISODEP_FSDI_DEFAULT **RFAL_ISODEP_FSXI_256**

Default Frame Size Integer in Poll mode

Examples

exampleRfalPoller.c.

◆ RFAL_ISODEP_FSX_KEEP

#define RFAL_ISODEP_FSX_KEEP (0xFFU)

Flag to keep FSX from activation

Examples

exampleRfalPoller.c.

◆ RFAL_ISODEP_FWI_DEFAULT

```
#define RFAL_ISODEP_FWI_DEFAULT   (4U)
```

Default value for FWI Digital 1.0 11.6.2.17

◆ RFAL_ISODEP_FWI_MASK

```
#define RFAL_ISODEP_FWI_MASK   (0xF0U)
```

Mask bits of FWI

◆ RFAL_ISODEP_FWI_SHIFT

```
#define RFAL_ISODEP_FWI_SHIFT   (4U)
```

Shift val of FWI

◆ RFAL_ISODEP_MAX_DSL_RETRYS

```
#define RFAL_ISODEP_MAX_DSL_RETRYS   (0U)
```

Number of retries for a S(DESELECT) Digital 2.0 B9 - nRETRY DESELECT: [0,5]

◆ RFAL_ISODEP_MAX_FWT

```
#define RFAL_ISODEP_MAX_FWT   ((uint32_t)1U<<26)
```

Maximum Frame Waiting Time = ((256 * 16/fc) * 2^FWImax) = ((256*16/fc)*2^14) = (67108864)/fc = 2^26 (1/fc)

◆ RFAL_ISODEP_MAX_I_RETRYS

```
#define RFAL_ISODEP_MAX_I_RETRYS   (2U)
```

Number of retries for a I-Block Digital 2.0 16.2.5.4

◆ RFAL_ISODEP_MAX_R_RETRYS

```
#define RFAL_ISODEP_MAX_R_RETRYS   (3U)
```

Number of retries for a R-Block Digital 2.0 B9 - nRETRY ACK/NAK: [2,5]

◆ RFAL_ISODEP_MAX_WTX_NACK_RETRYS

```
#define RFAL_ISODEP_MAX_WTX_NACK_RETRYS   (3U)
```

Number of S(WTX) replied with NACK Digital 2.0 B9 - nRETRY WTX[2,5]

◆ RFAL_ISODEP_MAX_WTX_RETRYS

```
#define RFAL_ISODEP_MAX_WTX_RETRYS   (20U)
```

Number of overall S(WTX) retries Digital 2.0 16.2.5.2

◆ RFAL_ISODEP_MAX_WTX_RETRYS_ULTD

```
#define RFAL_ISODEP_MAX_WTX_RETRYS_ULTD   (255U)
```

Use unlimited number of overall S(WTX)

◆ RFAL_ISODEP_NAD_LEN

```
#define RFAL_ISODEP_NAD_LEN   (1U)
```

NAD length

◆ RFAL_ISODEP_NO_DID

```
#define RFAL_ISODEP_NO_DID   (0x10U)
```

DID value indicating the ISO-DEP layer not to use DID

Examples

exampleRfalPoller.c.

◆ RFAL_ISODEP_NO_NAD

#define RFAL_ISODEP_NO_NAD (0xFFU)

NAD value indicating the ISO-DEP layer not to use NAD

◆ RFAL_ISODEP_PCB_LEN

#define RFAL_ISODEP_PCB_LEN (1U)

PCB length

◆ RFAL_ISODEP_PROLOGUE_SIZE

#define RFAL_ISODEP_PROLOGUE_SIZE (3U)

Length of Prologue Field for I-Block Format

◆ RFAL_ISODEP_RATS_RETRIES

#define RFAL_ISODEP_RATS_RETRIES (1U)

RATS retries upon fail Digital 2.0 B7 - nRETRY RATS [0,1]

◆ RFAL_ISODEP_SAME_BITRATE_MASK

#define RFAL_ISODEP_SAME_BITRATE_MASK (0x80U)

Mask bit indicate only same bit rate D for both direction support

◆ RFAL_ISODEP_SPARAM_TAG_BLOCKINFO

```
#define RFAL_ISODEP_SPARAM_TAG_BLOCKINFO   (0xA0U)
```

S(PARAMETERS) tag Block information

◆ RFAL_ISODEP_SPARAM_TAG_BRACK

```
#define RFAL_ISODEP_SPARAM_TAG_BRACK   (0xA4U)
```

S(PARAMETERS) tag Bit rates Acknowledgement

◆ RFAL_ISODEP_SPARAM_TAG_BRACK_LEN

```
#define RFAL_ISODEP_SPARAM_TAG_BRACK_LEN   (0U)
```

S(PARAMETERS) tag Bit rates Acknowledgement Length

◆ RFAL_ISODEP_SPARAM_TAG_BRACT

```
#define RFAL_ISODEP_SPARAM_TAG_BRACT   (0xA3U)
```

S(PARAMETERS) tag Bit rates Activation

◆ RFAL_ISODEP_SPARAM_TAG_BRIND

```
#define RFAL_ISODEP_SPARAM_TAG_BRIND   (0xA2U)
```

S(PARAMETERS) tag Bit rates Indication

◆ RFAL_ISODEP_SPARAM_TAG_BRREQ

```
#define RFAL_ISODEP_SPARAM_TAG_BRREQ   (0xA1U)
```

S(PARAMETERS) tag Bit rates Request

◆ RFAL_ISODEP_SPARAM_TAG_BRREQ_LEN

```
#define RFAL_ISODEP_SPARAM_TAG_BRREQ_LEN   (0U)
```

S(PARAMETERS) tag Bit rates Request Length

◆ RFAL_ISODEP_SPARAM_TAG_LEN

```
#define RFAL_ISODEP_SPARAM_TAG_LEN   (1)
```

S(PARAMETERS) Tag Length

◆ RFAL_ISODEP_SPARAM_TAG_PCD2PICC_LEN

```
#define RFAL_ISODEP_SPARAM_TAG_PCD2PICC_LEN   (2U)
```

S(PARAMETERS) bit rates from PICC to PCD Length

◆ RFAL_ISODEP_SPARAM_TAG_PICC2PCD_LEN

```
#define RFAL_ISODEP_SPARAM_TAG_PICC2PCD_LEN   (2U)
```

S(PARAMETERS) bit rates from PCD to PICC Length

◆ RFAL_ISODEP_SPARAM_TAG_SEL_FRAME

```
#define RFAL_ISODEP_SPARAM_TAG_SEL_FRAME   (0x85U)
```

S(PARAMETERS) tag Selected framing options PICC to PCD

◆ RFAL_ISODEP_SPARAM_TAG_SEL_PCD2PICC

```
#define RFAL_ISODEP_SPARAM_TAG_SEL_PCD2PICC   (0x83U)
```

S(PARAMETERS) tag Selected bit rate from PCD to PICC

◆ RFAL_ISODEP_SPARAM_TAG_SEL_PICC2PCD

```
#define RFAL_ISODEP_SPARAM_TAG_SEL_PICC2PCD   (0x84U)
```

S(PARAMETERS) tag Selected bit rate from PICC to PCD

◆ RFAL_ISODEP_SPARAM_TAG_SUP_FRAME

#define RFAL_ISODEP_SPARAM_TAG_SUP_FRAME (0x82U)

S(PARAMETERS) tag Supported framing options PICC to PCD

◆ RFAL_ISODEP_SPARAM_TAG_SUP_PCD2PICC

#define RFAL_ISODEP_SPARAM_TAG_SUP_PCD2PICC (0x80U)

S(PARAMETERS) tag Supported bit rates from PCD to PICC

◆ RFAL_ISODEP_SPARAM_TAG_SUP_PICC2PCD

#define RFAL_ISODEP_SPARAM_TAG_SUP_PICC2PCD (0x81U)

S(PARAMETERS) tag Supported bit rates from PICC to PCD

◆ RFAL_ISODEP_SPARAM_VALUES_MAX_LEN

#define RFAL_ISODEP_SPARAM_VALUES_MAX_LEN (16U)

Maximum Length of the value field on S(PARAMETERS)

Enumeration Type Documentation

◆ rfalIsoDepFSx

enum **rfalIsoDepFSx**

Frame Size for Proximity Card definitions

Enumerator	
RFAL_ISODEP_FSX_16	Frame Size for Proximity Card with 16 bytes
RFAL_ISODEP_FSX_24	Frame Size for Proximity Card with 24 bytes

RFAL_ISODEP_FSX_32	Frame Size for Proximity Card with 32 bytes
RFAL_ISODEP_FSX_40	Frame Size for Proximity Card with 40 bytes
RFAL_ISODEP_FSX_48	Frame Size for Proximity Card with 48 bytes
RFAL_ISODEP_FSX_64	Frame Size for Proximity Card with 64 bytes
RFAL_ISODEP_FSX_96	Frame Size for Proximity Card with 96 bytes
RFAL_ISODEP_FSX_128	Frame Size for Proximity Card with 128 bytes
RFAL_ISODEP_FSX_256	Frame Size for Proximity Card with 256 bytes
RFAL_ISODEP_FSX_512	Frame Size for Proximity Card with 512 bytes ISO14443-3 Amd2 2012
RFAL_ISODEP_FSX_1024	Frame Size for Proximity Card with 1024 bytes ISO14443-3 Amd2 2012
RFAL_ISODEP_FSX_2048	Frame Size for Proximity Card with 2048 bytes ISO14443-3 Amd2 2012
RFAL_ISODEP_FSX_4096	Frame Size for Proximity Card with 4096 bytes ISO14443-3 Amd2 2012

◆ rfalIsoDepFSxI

enum **rfalIsoDepFSxI**

Frame Size for Proximity Card Integer definitions

Enumerator	
RFAL_ISODEP_FSXI_16	Frame Size for Proximity Card Integer with 16 bytes
RFAL_ISODEP_FSXI_24	Frame Size for Proximity Card Integer with 24 bytes
RFAL_ISODEP_FSXI_32	Frame Size for Proximity Card Integer with 32 bytes
RFAL_ISODEP_FSXI_40	Frame Size for Proximity Card Integer with 40 bytes
RFAL_ISODEP_FSXI_48	Frame Size for Proximity Card Integer with 48 bytes
RFAL_ISODEP_FSXI_64	Frame Size for Proximity Card Integer with 64 bytes
RFAL_ISODEP_FSXI_96	Frame Size for Proximity Card Integer with 96 bytes
RFAL_ISODEP_FSXI_128	Frame Size for Proximity Card Integer with 128 bytes
RFAL_ISODEP_FSXI_256	Frame Size for Proximity Card Integer with 256 bytes
RFAL_ISODEP_FSXI_512	Frame Size for Proximity Card Integer with 512 bytes ISO14443-3 Amd2 2012
RFAL_ISODEP_FSXI_1024	Frame Size for Proximity Card Integer with 1024 bytes ISO14443-3 Amd2 2012

RFAL_ISODEP_FSXI_2048	Frame Size for Proximity Card Integer with 2048 bytes ISO14443-3 Amd2 2012
RFAL_ISODEP_FSXI_4096	Frame Size for Proximity Card Integer with 4096 bytes ISO14443-3 Amd2 2012

Function Documentation

◆ rfaIIsoDepATTRIB()

```
ReturnCode rfaIIsoDepATTRIB ( const uint8_t *          nfcid0,
                              uint8_t                  PARAM1,
                              rfaIBitRate              DSI,
                              rfaIBitRate              DRI,
                              rfaIIsoDepFSxI           FSDI,
                              uint8_t                  PARAM3,
                              uint8_t                  DID,
                              const uint8_t *          HLInfo,
                              uint8_t                  HLInfoLen,
                              uint32_t                 fwt,
                              rfaIIsoDepAttribRes *    attribRes,
                              uint8_t *                attribResLen
                            )
```

This sends a ATTRIB to make a NFC-B Listen Device to enter ISO-DEP layer (ISO14443-4) and checks if the received ATTRIB Response is valid

Parameters

- [in] **nfcid0** : NFCID0 to be used for the ATTRIB
- [in] **PARAM1** : ATTRIB PARAM1 byte (communication parameters)
- [in] **DSI** : DSI code the divisor from Listener (PICC) to Poller (PCD)
- [in] **DRI** : DRI code the divisor from Poller (PCD) to Listener (PICC)
- [in] **FSDI** : PCD's Frame Size to be announced on the ATTRIB
- [in] **PARAM3** : ATTRIB PARAM1 byte (protocol type)
- [in] **DID** : Device ID to be used or RFAL_ISODEP_NO_DID for not use DID
- [in] **HLInfo** : pointer to Higher layer INF (NULL if none)

[in] **HLInfoLen** : Length HLInfo

[in] **fwt** : Frame Waiting Time to be used (from SENSB_RES)

[out] **attribRes** : pointer to place the ATTRIB Response

[out] **attribResLen** : pointer to place the ATTRIB Response length

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_NONE : No error, ATTRIB Response received

◆ rfaIsoDepDeselect()

ReturnCode rfaIsoDepDeselect (void)

This function sends a deselect command to PICC and waits for its responce in a blocking way

Returns

RFAL_ERR_NONE : Deselect successfully sent and acknowledged by PICC

RFAL_ERR_PROTO : Protocol error occurred

RFAL_ERR_TIMEOUT : No response rcvd from PICC

Examples

exampleRfalPoller.c.

◆ rfaIsoDepFSxI2FSx()

uint16_t rfaIsoDepFSxI2FSx (uint8_t FSxI)

Convert Frame Size for proximity coupling Device Integer (FSxI) to Frame Size for proximity coupling Device (FSx)

FSD - maximum frame size for NFC Forum Device in Poll Mode FSC - maximum frame size for NFC Forum Device in Listen Mode

FSxI = FSDI or FSCI FSx = FSD or FSC

The FSD/FSC value includes the header and CRC

Parameters

[in] **FSxI** : Frame Size for proximity coupling Device Integer

Returns

fsx : Frame Size for proximity coupling Device (FSD or FSC)

◆ rfaIIsoDepFWI2FWT()

uint32_t rfaIIsoDepFWI2FWT (uint8_t fwi)

Convert Frame Waiting time Integer (FWI) to Frame Waiting Time (FWT) in 1/fc units

Parameters

[in] **fwi** : Frame Waiting time Integer

Returns

fwt : Frame Waiting Time in 1/fc units

◆ rfaIIsoDepGetApduTransceiveStatus()

ReturnCode rfaIIsoDepGetApduTransceiveStatus (void)

Returns

RFAL_ERR_NONE : if Transceive has been completed successfully

RFAL_ERR_BUSY : if Transceive is ongoing

RFAL_ERR_PROTO : if a protocol error occurred

RFAL_ERR_TIMEOUT : if a timeout error occurred

RFAL_ERR_SLEEP_REQ : if Deselect is received and responded

RFAL_ERR_NOMEM : if the received INF does not fit into the receive buffer

RFAL_ERR_LINK_LOSS : if communication is lost because Reader/Writer has turned off its field

◆ rfaIIsoDepGetDeselectStatus()

ReturnCode rfaIIsoDepGetDeselectStatus (void)

This function sends a deselect command to PICC and waits for it`s responce in a blocking way

Returns

RFAL_ERR_NONE : Deselect successfully sent and acknowledged by PICC

RFAL_ERR_PROTO : Protocol error occurred

RFAL_ERR_TIMEOUT : No response rcvd from PICC

◆ rfaIsoDepGetMaxInfLen()

uint16_t rfaIsoDepGetMaxInfLen (void)

Gets the maximum INF length in bytes based on current Frame Size for proximity coupling Device (FSD or FSC) excluding the header and CRC

Returns

maximum INF length in bytes

◆ rfaIsoDepGetTransceiveStatus()

ReturnCode rfaIsoDepGetTransceiveStatus (void)

Returns the status of the ISO-DEP Transceive

Warning

When the other device is performing chaining once a chained block is received the error RFAL_ERR_AGAIN is sent. At this point caller must handle the received data immediately. When RFAL_ERR_AGAIN is returned an ACK has already been sent to the other device and the next block might be incoming. If **rfaIWorker()** is called frequently it will place the next block on the given buffer

Returns

RFAL_ERR_NONE : Transceive has been completed successfully

RFAL_ERR_BUSY : Transceive is ongoing

RFAL_ERR_PROTO : Protocol error occurred

RFAL_ERR_TIMEOUT : Timeout error occurred

RFAL_ERR_SLEEP_REQ : Deselect has been received and responded

RFAL_ERR_NOMEM : The received INF does not fit into the receive buffer

RFAL_ERR_LINK_LOSS : Communication is lost because Reader/Writer has turned off its field

RFAL_ERR_AGAIN : received one chaining block, continue to call this method to retrieve the remaining blocks

Examples

exampleRfalPoller.c.

◆ rfalIsoDepInitialize()

```
void rfalIsoDepInitialize ( void )
```

Initialize the ISO-DEP protocol layer with default config

Examples

exampleRfalPoller.c.

◆ rfalIsoDepInitializeWithParams()

```
void rfalIsoDepInitializeWithParams ( rfalComplianceMode compMode,
                                      uint8_t            maxRetriesR,
                                      uint8_t            maxRetriesSnWTX,
                                      uint8_t            maxRetriesSWTX,
                                      uint8_t            maxRetriesSDSL,
                                      uint8_t            maxRetriesI,
                                      uint8_t            maxRetriesRATS
                                    )
```

Initialize the ISO-DEP protocol layer with additional parameters allowing to customise the protocol layer for specific behaviours

Parameters

[in]	**compMode**	: Compliance mode to be performed
[in]	**maxRetriesR**	: Number of retries for a R-Block Digital 2.0 B9 - nRETRY ACK/NAK: [2,5]
[in]	**maxRetriesSnWTX**	: Number of retries for a S(WTX) (only in case of NAKs) Digital 2.0 B9 - nRETRY WTX[2,5]
[in]	**maxRetriesSWTX**	: Number of overall S(WTX) retries. Use RFAL_ISODEP_MAX_WTX_RETRYS_ULTD for disabling this limit check Digital 2.0 16.2.5.2
[in]	**maxRetriesSDSL**	: Number of retries for a S(DESELECT) Digital 2.0 B9 - nRETRY DESELECT: [0,5]
[in]	**maxRetriesI**	: Number of retries for a I-Block Digital 2.0 16.2.5.4
[in]	**maxRetriesRATS**	: Number of retries for RATS Digital 2.0 B7 - nRETRY RATS [0,1]

◆ rfalIsoDepIsAttrib()

```
bool rfalIsoDepIsAttrib ( const uint8_t *  buf,
                          uint8_t          bufLen
                        )
```

Check if it is a well formed ATTRIB command, but does not check the validity of the information inside

Parameters

[in] **buf** : reference to buffer containing the data to be checked

[in] **bufLen** : length of data in the buffer in bytes

Returns

true if the data indicates a ATTRIB command; false otherwise

◆ rfalIsoDepIsRats()

```
bool rfalIsoDepIsRats ( const uint8_t *  buf,
                        uint8_t          bufLen
                      )
```

Check if it is a well formed RATS command with 2 bytes This function does not check the validity of FSDI and DID

Parameters

[in] **buf** : reference to buffer containing the data to be checked

[in] **bufLen** : length of data in the buffer in bytes

Returns

true if the data indicates a RATS command; false otherwise

◆ rfalIsoDepListenGetActivationStatus()

```
ReturnCode rfalIsoDepListenGetActivationStatus ( void )
```

Returns

RFAL_ERR_NONE if Activation is already completed

RFAL_ERR_BUSY if Activation is ongoing

RFAL_ERR_LINK_LOSS if Remote Field is turned off

◆ rfaIIsoDepListenStartActivation()

```
ReturnCode rfaIIsoDepListenStartActivation ( rfaIIsoDepAtsParam *                atsParam,
                                             const rfaIIsoDepAttribResParam *     attribResParam,
                                             const uint8_t *                      buf,
                                             uint16_t                             bufLen,
                                             rfaIIsoDepListenActvParam            actParam
                                           )
```

Start Listen Activation Handling and setup to receive first I-block which may contain complete or partial APDU after activation is completed

Pass in RATS for T4AT, or ATTRIB for T4BT, to handle ATS or ATTRIB Response respectively The Activation Handling handles ATS and ATTRIB Response; and additionally PPS Response if a PPS is received for T4AT. The method uses the current RFAL state machine to determine if it is expecting RATS or ATTRIB

Activation is completed if PPS Response is sent or if first PDU is received in T4T-A Activation is completed if ATTRIB Response is sent in T4T-B

rfaIIsoDepListenGetActivationStatus provide status if activation is completed. **rfaIIsoDepStartTransceive** shall be called right after activation is completed

Parameters

- [in] **atsParam** : reference to ATS parameters
- [in] **attribResParam** : reference to ATTRIB_RES parameters
- [in] **buf** : reference to buffer containing RATS or ATTRIB
- [in] **bufLen** : length in bytes of the given bufffer
- [in] **actParam** : reference to incoming reception information will be placed

Warning

Once the Activation has been completed the method **rfaIIsoDepGetTransceiveStatus()** must be called. If activation has completed due to reception of a data block (not PPS) the buffer owned by the caller and passed on actParam must still contain this data. The first data will be processed (I-Block or S-DSL) by **rfaIIsoDepGetTransceiveStatus()** inform the caller and then for the next transaction use **rfaIIsoDepStartTransceive()**

Returns

RFAL_ERR_NONE : RATS/ATTRIB is valid and activation has started

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_PROTO : Invalid request

RFAL_ERR_NOTSUPP : Feature not supported

◆ rfaIIsoDepPollAGetActivationStatus()

ReturnCode rfaIIsoDepPollAGetActivationStatus (void)

Returns the activation status started by rfaIIsoDepPollAStartActivation

See also

rfaIIsoDepPollAStartActivation

Returns

RFAL_ERR_BUSY : Operation is ongoing

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_PAR : Parity error detected

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_NONE : No error, activation successful

◆ rfaIIsoDepPollAHandleActivation()

```
ReturnCode rfaIIsoDepPollAHandleActivation ( rfaIIsoDepFSxI        FSDI,
                                             uint8_t               DID,
                                             rfaIBitRate           maxBR,
                                             rfaIIsoDepDevice *    rfaIIsoDepDev
                                           )
```

This performs a NFC-A Activation into ISO-DEP layer (ISO14443-4) with the given parameters. It sends RATS and if the higher bit rates are supported by both devices it additionally sends PPS Once Activated all details of the device are provided on isoDepDev

Parameters

[in]	**FSDI**	: Frame Size Device Integer to be used
[in]	**DID**	: Device ID to be used or RFAL_ISODEP_NO_DID for not use DID
[in]	**maxBR**	: Max bit rate supported by the Poller
[out]	**rfalIsoDepDev**	: ISO-DEP information of the activated Listen device

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_PAR : Parity error detected

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_NONE : No error, activation successful

Examples

exampleRfalPoller.c.

◆ rfalIsoDepPollAStartActivation()

```
ReturnCode rfalIsoDepPollAStartActivation ( rfalIsoDepFSxI        FSDI,
                                            uint8_t               DID,
                                            rfalBitRate           maxBR,
                                            rfalIsoDepDevice *    rfalIsoDepDev
                                          )
```

This starts a NFC-A Activation into ISO-DEP layer (ISO14443-4) with the given parameters. It sends RATS and if the higher bit rates are supported by both devices it additionally sends PPS Once Activated all details of the device are provided on isoDepDev

See also

rfalIsoDepPollAGetActivationStatus

Parameters

[in]	**FSDI**	: Frame Size Device Integer to be used

[in] **DID** : Device ID to be used or RFAL_ISODEP_NO_DID for not use DID

[in] **maxBR** : Max bit rate supported by the Poller

[out] **rfalIsoDepDev** : ISO-DEP information of the activated Listen device

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_PAR : Parity error detected

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_NONE : No error, start of asynchronous operation successful

◆ rfalIsoDepPollBGetActivationStatus()

ReturnCode rfalIsoDepPollBGetActivationStatus (void)

Returns the activation status started by rfalIsoDepPollBStartActivation

See also

rfalIsoDepPollBStartActivation

Returns

RFAL_ERR_BUSY : Operation is ongoing

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_PAR : Parity error detected

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_NONE : No error, activation successful

◆ rfaIsoDepPollBHandleActivation()

```
ReturnCode rfaIsoDepPollBHandleActivation ( rfaIsoDepFSxI                FSDI,
                                            uint8_t                       DID,
                                            rfaBitRate                    maxBR,
                                            uint8_t                       PARAM1,
                                            const rfaNfcbListenDevice *   nfcbDev,
                                            const uint8_t *               HLInfo,
                                            uint8_t                       HLInfoLen,
                                            rfaIsoDepDevice *             rfaIsoDepDev
                                          )
```

This performs a NFC-B Activation into ISO-DEP layer (ISO14443-4) with the given parameters. It sends ATTRIB and calculates supported higher bit rates of both devices and performs activation. Once Activated all details of the device are provided on isoDepDev

Parameters

[in]	**FSDI**	: Frame Size Device Integer to be used
[in]	**DID**	: Device ID to be used or RFAL_ISODEP_NO_DID for not use DID
[in]	**maxBR**	: Max bit rate supported by the Poller
[in]	**PARAM1**	: ATTRIB PARAM1 byte (communication parameters)
[in]	**nfcbDev**	: pointer to the NFC-B Device containing the SENSB_RES
[in]	**HLInfo**	: pointer to Higher layer INF (NULL if none)
[in]	**HLInfoLen**	: Length HLInfo
[out]	**rfaIsoDepDev**	: ISO-DEP information of the activated Listen device

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_PAR : Parity error detected

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_NONE : No error, activation successful

Examples

exampleRfalPoller.c.

◆ rfalIsoDepPollBStartActivation()

```
ReturnCode rfalIsoDepPollBStartActivation ( rfalIsoDepFSxI              FSDI,
                                            uint8_t                     DID,
                                            rfalBitRate                 maxBR,
                                            uint8_t                     PARAM1,
                                            const rfalNfcbListenDevice * nfcbDev,
                                            const uint8_t *             HLInfo,
                                            uint8_t                     HLInfoLen,
                                            rfalIsoDepDevice *          rfalIsoDepDev
                                          )
```

This starts a NFC-B Activation into ISO-DEP layer (ISO14443-4) with the given parameters. It will send ATTRIB and calculate supported higher bit rates of both devices and perform activation. Once Activated all details of the device are provided on isoDepDev

See also

rfalIsoDepPollBGetActivationStatus

Parameters

[in]	**FSDI**	: Frame Size Device Integer to be used
[in]	**DID**	: Device ID to be used or RFAL_ISODEP_NO_DID for not use DID
[in]	**maxBR**	: Max bit rate supported by the Poller
[in]	**PARAM1**	: ATTRIB PARAM1 byte (communication parameters)
[in]	**nfcbDev**	: pointer to the NFC-B Device containing the SENSB_RES
[in]	**HLInfo**	: pointer to Higher layer INF (NULL if none)
[in]	**HLInfoLen**	: Length HLInfo
[out]	**rfalIsoDepDev**	: ISO-DEP information of the activated Listen device

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_PAR : Parity error detected

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_NONE : No error, start of asynchronous operation successful

◆ rfaIIsoDepPollHandleSParameters()

```
ReturnCode rfaIIsoDepPollHandleSParameters ( rfaIIsoDepDevice * rfaIIsoDepDev,
                                             rfaIBitRate        maxTxBR,
                                             rfaIBitRate        maxRxBR
                                           )
```

This checks if PICC supports S(PARAMETERS), retieves PICC's capabilities and sets the Bit Rate at the highest supported by both devices

Parameters

[out] **rfaIIsoDepDev** : ISO-DEP information of the activated Listen device

[in] **maxTxBR** : Maximum Tx bit rate supported by PCD

[in] **maxRxBR** : Maximum Rx bit rate supported by PCD

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_NONE : No error, S(PARAMETERS) selection successful

◆ rfaIIsoDepPPS()

```
ReturnCode rfaIIsoDepPPS ( uint8_t                DID,
```

```
                                rfalBitRate            DSI,
                                rfalBitRate            DRI,
                                rfalIsoDepPpsRes *     ppsRes
                              )
```

This sends a PPS to make a NFC-A Listen Device change the communications bit rate from 106kbps to one of the supported bit rates Additionally checks if the received PPS response is valid

Parameters

- [in] **DID** : Device ID
- [in] **DSI** : DSI code the divisor from Listener (PICC) to Poller (PCD)
- [in] **DRI** : DRI code the divisor from Poller (PCD) to Listener (PICC)
- [out] **ppsRes** : pointer to place the PPS Response

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_PAR : Parity error detected

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_NONE : No error, PPS Response received

◆ rfalIsoDepRATS()

```
ReturnCode rfalIsoDepRATS ( rfalIsoDepFSxI    FSDI,
                            uint8_t           DID,
                            rfalIsoDepAts *   ats,
                            uint8_t *         atsLen
                          )
```

This sends a RATS to make a NFC-A Listen Device to enter ISO-DEP layer (ISO14443-4) and checks if the received ATS is valid

Parameters

[in] **FSDI** : Frame Size Device Integer to be used

[in] **DID** : Device ID to be used or RFAL_ISODEP_NO_DID for not use DID

[out] **ats** : pointer to place the ATS Response

[out] **atsLen** : pointer to place the ATS length

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_PAR : Parity error detected

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_NONE : No error, ATS received

◆ rfaIIsoDepStartApduTransceive()

ReturnCode rfaIIsoDepStartApduTransceive (**rfaIIsoDepApduTxRxParam** param)

This method triggers a ISO-DEP Transceive containing a complete APDU It transmits the given message and handles all protocol retransmitions, error handling and control messages

The txBuf contains a complete APDU to be transmitted The Prologue field will be manipulated by the Transceive

Warning

the txBuf will be modified during the transmission

the maximum RF frame which can be received is limited by param.tmpBuf

Parameters

[in] **param** reference parameters to be used for the Transceive

Returns

RFAL_ERR_PARAM : Bad request

RFAL_ERR_WRONG_STATE : The module is not in a proper state

RFAL_ERR_NONE : The Transceive request has been started

◆ rfaIIsoDepStartDeselect()

ReturnCode rfaIIsoDepStartDeselect (void)

This function starts the exchange to send deselect command to PICC and waits for its response

Returns

RFAL_ERR_NONE : Deselect successfully sent and acknowledged by PICC

RFAL_ERR_PROTO : Protocol error occurred

RFAL_ERR_TIMEOUT : No response rcvd from PICC

◆ rfaIIsoDepStartTransceive()

ReturnCode rfaIIsoDepStartTransceive (**rfaIIsoDepTxRxParam** param)

This method triggers a ISO-DEP Transceive containing a complete or partial APDU It transmits the given message and handles all protocol retransmitions, error handling and control messages

The txBuf contains a complete or partial APDU (INF) to be transmitted The Prologue field will be manipulated by the Transceive

If the buffer contains a partial APDU and is not the last block, then isTxChaining must be set to true

Parameters

`[in]` **param** reference parameters to be used for the Transceive

Returns

RFAL_ERR_PARAM : Bad request

RFAL_ERR_WRONG_STATE : The module is not in a proper state

RFAL_ERR_NONE : The Transceive request has been started

Examples

exampleRfalPoller.c.

rfalIsoDepRats Struct Reference


```
#include <rfal_isoDep.h>
```

Data Fields

uint8_t	**CMD**
uint8_t	**PARAM**

Detailed Description

RATS format Digital 1.1 13.6.1

Field Documentation

◆ CMD

uint8_t rfalIsoDepRats::CMD

RATS command byte: 0xE0

◆ PARAM

uint8_t rfalIsoDepRats::PARAM

Param indicating FSDI and DID

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

- **rfal_isoDep.h**

Generated on Fri Mar 14 2025 11:45:10 for RFAL by doxygen 1.9.5

RFAL

Main Page

Modules

Files

Examples

Data Structures

Data Structure Index

Data Fields

Data Fields

rfalIsoDepAts Struct Reference


```
#include <rfal_isoDep.h>
```

Data Fields

uint8_t	**TL**
uint8_t	**T0**
uint8_t	**TA**
uint8_t	**TB**
uint8_t	**TC**
uint8_t	**HB** [**RFAL_ISODEP_ATS_HB_MAX_LEN**]

Detailed Description

ATS response format Digital 1.1 13.6.2

Field Documentation

◆ HB

uint8_t rfalIsoDepAts::HB[**RFAL_ISODEP_ATS_HB_MAX_LEN**]

Historical Bytes

◆ T0

uint8_t rfalIsoDepAts::T0

Format Byte T0 indicating if TA, TB, TC

◆ TA

uint8_t rfalIsoDepAts::TA

Interface Byte **TA(1)**

◆ TB

uint8_t rfalIsoDepAts::TB

Interface Byte **TB(1)**

◆ TC

uint8_t rfalIsoDepAts::TC

Interface Byte **TC(1)**

◆ TL

uint8_t rfalIsoDepAts::TL

Length Byte, including TL byte itself

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

- **rfal_isoDep.h**

---

rfalIsoDepPpsReq Struct Reference


```
#include <rfal_isoDep.h>
```

Data Fields

uint8_t	**PPSS**
uint8_t	**PPS0**
uint8_t	**PPS1**

Detailed Description

PPS Request format (Protocol and Parameter Selection) ISO14443-4 5.3

Field Documentation

◆ PPS0

uint8_t rfalIsoDepPpsReq::PPS0

Parameter 0:[000b | PPS1[1n] | 0001b]

◆ PPS1

uint8_t rfalIsoDepPpsReq::PPS1

Parameter 1:[0000b | DSI[2b] | DRI[2b]]

◆ PPSS

uint8_t rfalIsoDepPpsReq::PPSS

Start Byte: [1101b | CID[4b]]

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

- **rfal_isoDep.h**

Generated on Fri Mar 14 2025 11:45:10 for RFAL by doxygen 1.9.5

RFAL

rfalIsoDepPpsRes Struct Reference

```
#include <rfal_isoDep.h>
```

Data Fields

uint8_t	**PPSS**

Detailed Description

PPS Response format (Protocol and Parameter Selection) ISO14443-4 5.4

Field Documentation

◆ PPSS

uint8_t rfalIsoDepPpsRes::PPSS

Start Byte: [1101b | CID[4b]]

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

- **rfal_isoDep.h**

rfalIsoDepAttribCmd Struct Reference


```
#include <rfal_isoDep.h>
```

Data Fields

uint8_t	**cmd**
uint8_t	**nfcid0** [**RFAL_NFCB_NFCID0_LEN**]
struct {	
uint8_t	**PARAM1**
uint8_t	**PARAM2**
uint8_t	**PARAM3**
uint8_t	**PARAM4**
}	**Param**
uint8_t	**HLInfo** [**RFAL_ISODEP_ATTRIB_HLINFO_LEN**]

Detailed Description

ATTRIB Command Format Digital 1.1 15.6.1

Field Documentation

◆ cmd

uint8_t rfalIsoDepAttribCmd::cmd

ATTRIB_REQ command byte

◆ HLInfo

uint8_t rfalIsoDepAttribCmd::HLInfo[**RFAL_ISODEP_ATTRIB_HLINFO_LEN**]

Higher Layer Information

◆ nfcid0

uint8_t rfalIsoDepAttribCmd::nfcid0[**RFAL_NFCB_NFCID0_LEN**]

NFCID0 of the card to be selected

◆

struct { ... } rfalIsoDepAttribCmd::Param

Parameter of ATTRIB command

◆ PARAM1

uint8_t rfalIsoDepAttribCmd::PARAM1

PARAM1 of ATTRIB command

◆ PARAM2

uint8_t rfalIsoDepAttribCmd::PARAM2

PARAM2 of ATTRIB command

◆ PARAM3

uint8_t rfalIsoDepAttribCmd::PARAM3

PARAM3 of ATTRIB command

◆ PARAM4

uint8_t rfalIsoDepAttribCmd::PARAM4

PARAM4 of ATTRIB command

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

- **rfal_isoDep.h**

---

rfalIsoDepAttribRes Struct Reference


```
#include <rfal_isoDep.h>
```

Data Fields

uint8_t	**mbliDid**
uint8_t	**HLInfo** [**RFAL_ISODEP_ATTRIB_HLINFO_LEN**]

Detailed Description

ATTRIB Response Format Digital 1.1 15.6.2

Field Documentation

◆ HLInfo

uint8_t rfalIsoDepAttribRes::HLInfo[**RFAL_ISODEP_ATTRIB_HLINFO_LEN**]

Higher Layer Information

◆ mbliDid

uint8_t rfalIsoDepAttribRes::mbliDid

Contains MBLI and DID

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

- **rfal_isoDep.h**

rfalIsoDepSParameter Struct Reference

RFAL » RFAL-AL » ISO-DEP

```
#include <rfal_isoDep.h>
```

Data Fields

uint8_t **tag**

uint8_t **length**

uint8_t **value** [**RFAL_ISODEP_SPARAM_VALUES_MAX_LEN**]

Detailed Description

S(Parameters) Command Format ISO14443-4 (2016) Table 4

Field Documentation

◆ length

uint8_t rfalIsoDepSParameter::length

S(PARAMETERS) Length field

◆ tag

uint8_t rfalIsoDepSParameter::tag

S(PARAMETERS) Tag field

◆ value

uint8_t rfalIsoDepSParameter::value[**RFAL_ISODEP_SPARAM_VALUES_MAX_LEN**]

S(PARAMETERS) Value field

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

- **rfal_isoDep.h**

rfalIsoDepActivation Union Reference


```
#include <rfal_isoDep.h>
```

Collaboration diagram for rfalIsoDepActivation:

[legend]

Data Fields

union {	
struct {	
rfalIsoDepAts **ATS**	
uint8_t **ATSLen**	
} **Listener**	
struct {	
rfalIsoDepRats **RATS**	
} **Poller**	
}	**A**
union {	
struct {	

rfalIsoDepAttribRes **ATTRIB_RES**	
uint8_t **ATTRIB_RESLen**	
} **Listener**	
struct { **rfalIsoDepAttribCmd** **ATTRIB**	
uint8_t **ATTRIBLen**	
} **Poller**	
}	**B**

Detailed Description

Activation info as Poller and Listener for NFC-A and NFC-B

Field Documentation

◆

union { ... } rfalIsoDepActivation::A

NFC-A information

◆ ATS

rfalIsoDepAts rfalIsoDepActivation::ATS

ATS response (Poller mode)

◆ ATSLen

uint8_t rfalIsoDepActivation::ATSLen

ATS response length (Poller mode)

◆ ATTRIB

rfaIsoDepAttribCmd rfaIsoDepActivation::ATTRIB

ATTRIB request (Listener mode)

◆ ATTRIB_RES

rfaIsoDepAttribRes rfaIsoDepActivation::ATTRIB_RES

ATTRIB_RES (Poller mode)

◆ ATTRIB_RESLen

uint8_t rfaIsoDepActivation::ATTRIB_RESLen

ATTRIB_RES length (Poller mode)

◆ ATTRIBLen

uint8_t rfaIsoDepActivation::ATTRIBLen

ATTRIB request length (Listener mode)

◆

union { ... } rfaIsoDepActivation::B

NFC-B information

◆ RATS

rfaIsoDepRats rfaIsoDepActivation::RATS

RATS request (Listener mode)

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

- rfal_isoDep.h

rfaIIsoDepInfo Struct Reference


```
#include <rfal_isoDep.h>
```

Data Fields

uint8_t	**FWI**
uint32_t	**FWT**
uint32_t	**dFWT**
uint32_t	**SFGI**

uint32_t	**SFGT**
uint8_t	**FSxI**
uint16_t	**FSx**
uint32_t	**MBL**
rfalBitRate	**DSI**
rfalBitRate	**DRI**
uint8_t	**DID**
uint8_t	**NAD**
bool	**supDID**
bool	**supNAD**
bool	**supAdFt**

Detailed Description

ISO-DEP device Info

Field Documentation

◆ dFWT

uint32_t rfalIsoDepInfo::dFWT

Delta Frame Waiting Time (1/fc)

◆ DID

uint8_t rfalIsoDepInfo::DID

Device ID

◆ DRI

rfalBitRate rfalIsoDepInfo::DRI

Bit Rate coding from Poller (PCD) to Listener (PICC)

◆ DSI

rfalBitRate rfalIsoDepInfo::DSI

Bit Rate coding from Listener (PICC) to Poller (PCD)

◆ FSx

uint16_t rfalIsoDepInfo::FSx

Frame Size Device/Card (FSD or FSC)

◆ FSxI

uint8_t rfalIsoDepInfo::FSxI

Frame Size Device/Card Integer (FSDI or FSCI)

◆ FWI

uint8_t rfalIsoDepInfo::FWI

Frame Waiting Integer

◆ FWT

uint32_t rfalIsoDepInfo::FWT

Frame Waiting Time (1/fc)

◆ MBL

uint32_t rfalIsoDepInfo::MBL

Maximum Buffer Length (optional for NFC-B)

◆ NAD

uint8_t rfalIsoDepInfo::NAD

Node ADdress

◆ SFGI

uint32_t rfalIsoDepInfo::SFGI

Start-up Frame Guard time Integer

◆ SFGT

uint32_t rfalIsoDepInfo::SFGT

Start-up Frame Guard Time (ms)

◆ supAdFt

bool rfalIsoDepInfo::supAdFt

Advanced Features supported flag

◆ supDID

bool rfalIsoDepInfo::supDID

DID supported flag

◆ supNAD

bool rfalIsoDepInfo::supNAD

NAD supported flag

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

- rfal_isoDep.h

---

rfalIsoDepDevice Struct Reference

RFAL » RFAL-AL » ISO-DEP

```
#include <rfal_isoDep.h>
```

Collaboration diagram for rfalIsoDepDevice:

[legend]

Data Fields

rfaIIsoDepActivation	**activation**
rfaIIsoDepInfo	**info**

Detailed Description

ISO-DEP Device structure

Examples

exampleRfalPoller.c.

Field Documentation

◆ activation

rfaIIsoDepActivation rfaIIsoDepDevice::activation

Activation Info

◆ info

rfaIIsoDepInfo rfaIIsoDepDevice::info

ISO-DEP (ISO14443-4) device Info

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

- **rfal_isoDep.h**

Generated on Fri Mar 14 2025 11:45:10 for RFAL by doxygen 1.9.5

RFAL

rfalIsoDepAttribResParam Struct Reference


```
#include <rfal_isoDep.h>
```

Data Fields

uint8_t	**mbli**
uint8_t	**HLInfo** [**RFAL_ISODEP_ATTRIB_HLINFO_LEN**]
uint8_t	**HLInfoLen**

Detailed Description

ATTRIB Response parameters

Field Documentation

◆ HLInfo

uint8_t rfalIsoDepAttribResParam::HLInfo[**RFAL_ISODEP_ATTRIB_HLINFO_LEN**]

Hi Layer Information

◆ HLInfoLen

uint8_t rfalIsoDepAttribResParam::HLInfoLen

Hi Layer Information Length

◆ mbli

uint8_t rfalIsoDepAttribResParam::mbli

MBLI

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

- **rfal_isoDep.h**

rfalIsoDepAtsParam Struct Reference


```
#include <rfal_isoDep.h>
```

Data Fields

uint8_t	**fsci**
uint8_t	**fwi**
uint8_t	**sfgi**
bool	**didSupport**
uint8_t	**ta**
uint8_t *	**hb**
uint8_t	**hbLen**

Detailed Description

ATS Response parameter

Field Documentation

◆ didSupport

bool rfalIsoDepAtsParam::didSupport

DID Supported

◆ fsci

uint8_t rfalIsoDepAtsParam::fsci

Frame Size of Proximity Card Integer

◆ fwi

uint8_t rfalIsoDepAtsParam::fwi

Frame Waiting Time Integer

◆ hb

uint8_t* rfalIsoDepAtsParam::hb

Historical Bytes data

◆ hbLen

uint8_t rfalIsoDepAtsParam::hbLen

Historical Bytes Length

◆ sfgi

uint8_t rfalIsoDepAtsParam::sfgi

Start-Up Frame Guard Time Integer

◆ ta

uint8_t rfalIsoDepAtsParam::ta

Max supported bitrate both direction

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

- **rfal_isoDep.h**

Generated on Fri Mar 14 2025 11:45:10 for RFAL by doxygen 1.9.5

rfalIsoDepBufFormat Struct Reference

```
#include <rfal_isoDep.h>
```

Data Fields

uint8_t	**prologue** [**RFAL_ISODEP_PROLOGUE_SIZE**]
uint8_t	**inf** [**RFAL_FEATURE_ISO_DEP_IBLOCK_MAX_LEN**]

Detailed Description

Structure of I-Block Buffer format from caller

Examples

exampleRfalPoller.c.

Field Documentation

◆ inf

uint8_t rfalIsoDepBufFormat::inf[**RFAL_FEATURE_ISO_DEP_IBLOCK_MAX_LEN**]

INF/Payload buffer

◆ prologue

uint8_t rfalIsoDepBufFormat::prologue[**RFAL_ISODEP_PROLOGUE_SIZE**]

Prologue/SoD buffer

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

- **rfal_isoDep.h**

RFAL

Main Page

Modules

Files

Examples

Data Structures

Data Structure Index

Data Fields

Data Fields

rfalIsoDepApduBufFormat Struct Reference

RFAL » RFAL-AL » ISO-DEP

```
#include <rfal_isoDep.h>
```

Data Fields

uint8_t	**prologue** [**RFAL_ISODEP_PROLOGUE_SIZE**]
uint8_t	**apdu** [**RFAL_FEATURE_ISO_DEP_APDU_MAX_LEN**]

Detailed Description

Structure of APDU Buffer format from caller

Field Documentation

◆ apdu

uint8_t rfalIsoDepApduBufFormat::apdu[**RFAL_FEATURE_ISO_DEP_APDU_MAX_LEN**]

APDU/Payload buffer

◆ prologue

uint8_t rfalIsoDepApduBufFormat::prologue[**RFAL_ISODEP_PROLOGUE_SIZE**]

Prologue/SoD buffer

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

- **rfal_isoDep.h**

---

rfaIIsoDepListenActvParam Struct Reference

RFAL » RFAL-AL » ISO-DEP

```
#include <rfal_isoDep.h>
```

Collaboration diagram for rfaIIsoDepListenActvParam:

[legend]

Data Fields

rfaIIsoDepBufFormat *	**rxBuf**
uint16_t *	**rxLen**
bool *	**isRxChaining**
rfaIIsoDepDevice *	**isoDepDev**

Detailed Description

Listen Activation Parameters Structure

Field Documentation

◆ isoDepDev

rfalIsoDepDevice* rfalIsoDepListenActvParam::isoDepDev

ISO-DEP device info

◆ isRxChaining

bool* rfalIsoDepListenActvParam::isRxChaining

Received data is not complete

◆ rxBuf

rfalIsoDepBufFormat* rfalIsoDepListenActvParam::rxBuf

Receive Buffer struct reference

◆ rxLen

uint16_t* rfalIsoDepListenActvParam::rxLen

Received INF data length in Bytes

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

- **rfal_isoDep.h**

rfalIsoDepTxRxParam Struct Reference


```
#include <rfal_isoDep.h>
```

Collaboration diagram for rfalIsoDepTxRxParam:

[legend]

Data Fields

rfalIsoDepBufFormat *	txBuf
uint16_t	txBufLen
bool	isTxChaining
rfalIsoDepBufFormat *	rxBuf
uint16_t *	rxLen

bool *	**isRxChaining**
uint32_t	**FWT**
uint32_t	**dFWT**
uint16_t	**ourFSx**
uint16_t	**FSx**
uint8_t	**DID**

Detailed Description

Structure of parameters used on ISO DEP Transceive

Examples

exampleRfalPoller.c.

Field Documentation

◆ dFWT

uint32_t rfalIsoDepTxRxParam::dFWT

Delta FWT to be used

Examples

exampleRfalPoller.c.

◆ DID

uint8_t rfalIsoDepTxRxParam::DID

Device ID (RFAL_ISODEP_NO_DID if no DID)

Examples

exampleRfalPoller.c.

◆ FSx

uint16_t rfalIsoDepTxRxParam::FSx

Other device Frame Size (FSD or FSC)

Examples

exampleRfalPoller.c.

◆ FWT

uint32_t rfalIsoDepTxRxParam::FWT

FWT to be used (ignored in Listen Mode)

Examples

exampleRfalPoller.c.

◆ isRxChaining

bool* rfalIsoDepTxRxParam::isRxChaining

Received data is not complete

Examples

exampleRfalPoller.c.

◆ isTxChaining

bool rfalIsoDepTxRxParam::isTxChaining

Transmit data is not complete

Examples

exampleRfalPoller.c.

◆ ourFSx

uint16_t rfalIsoDepTxRxParam::ourFSx

Our device Frame Size (FSD or FSC)

Examples

exampleRfalPoller.c.

◆ rxBuf

rfalIsoDepBufFormat* rfalIsoDepTxRxParam::rxBuf

Receive Buffer struct reference in Bytes

Examples

exampleRfalPoller.c.

◆ rxLen

uint16_t* rfalIsoDepTxRxParam::rxLen

Received INF data length in Bytes

Examples

exampleRfalPoller.c.

◆ txBuf

rfalIsoDepBufFormat* rfalIsoDepTxRxParam::txBuf

Transmit Buffer struct reference

Examples

exampleRfalPoller.c.

◆ txBufLen

uint16_t rfalIsoDepTxRxParam::txBufLen

Transmit Buffer INF field length in Bytes

Examples

exampleRfalPoller.c.

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

- **rfal_isoDep.h**

---

rfalIsoDepApduTxRxParam Struct Reference

RFAL » **RFAL-AL** » **ISO-DEP**

```
#include <rfal_isoDep.h>
```

Collaboration diagram for rfalIsoDepApduTxRxParam:

[legend]

Data Fields

rfaIIsoDepApduBufFormat *	**txBuf**
uint16_t	**txBufLen**
rfaIIsoDepApduBufFormat *	**rxBuf**
uint16_t *	**rxLen**
rfaIIsoDepBufFormat *	**tmpBuf**
uint32_t	**FWT**
uint32_t	**dFWT**
uint16_t	**FSx**
uint16_t	**ourFSx**
uint8_t	**DID**

Detailed Description

Structure of parameters used on ISO DEP APDU Transceive

Field Documentation

◆ dFWT

uint32_t rfalIsoDepApduTxRxParam::dFWT

Delta FWT to be used

◆ DID

uint8_t rfalIsoDepApduTxRxParam::DID

Device ID (RFAL_ISODEP_NO_DID if no DID)

◆ FSx

uint16_t rfalIsoDepApduTxRxParam::FSx

Other device Frame Size (FSD or FSC)

◆ FWT

uint32_t rfalIsoDepApduTxRxParam::FWT

FWT to be used (ignored in Listen Mode)

◆ ourFSx

uint16_t rfalIsoDepApduTxRxParam::ourFSx

Our device Frame Size (FSD or FSC)

◆ rxBuf

rfalIsoDepApduBufFormat* rfalIsoDepApduTxRxParam::rxBuf

Receive Buffer struct reference in Bytes

◆ rxLen

uint16_t* rfalIsoDepApduTxRxParam::rxLen

Received INF data length in Bytes

◆ tmpBuf

rfalIsoDepBufFormat* rfalIsoDepApduTxRxParam::tmpBuf

Temp buffer for Rx I-Blocks (internal)

◆ txBuf

rfalIsoDepApduBufFormat* rfalIsoDepApduTxRxParam::txBuf

Transmit Buffer struct reference

◆ txBufLen

uint16_t rfalIsoDepApduTxRxParam::txBufLen

Transmit Buffer INF field length in Bytes

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

- **rfal_isoDep.h**

NFC-A

RFAL NFC-A Module. More...

Collaboration diagram for NFC-A:

Data Structures

struct	**rfalNfcaSensRes**
struct	**rfalNfcaSddReq**
struct	**rfalNfcaSddRes**
struct	**rfalNfcaSelReq**
struct	**rfalNfcaSelRes**
struct	**rfalNfcaListenDevice**

Macros

#define	**RFAL_NFCA_CASCADE_1_UID_LEN**	4U
#define	**RFAL_NFCA_CASCADE_2_UID_LEN**	7U
#define	**RFAL_NFCA_CASCADE_3_UID_LEN**	10U
#define	**RFAL_NFCA_SENS_RES_PLATFORM_MASK**	0x0FU
#define	**RFAL_NFCA_SENS_RES_PLATFORM_T1T**	0x0CU

#define **RFAL_NFCA_SEL_RES_CONF_MASK** 0x60U

#define **RFAL_NFCA_SEL_RES_CONF_T2T** 0x00U

#define **RFAL_NFCA_SEL_RES_CONF_T4T** 0x20U

#define **RFAL_NFCA_SEL_RES_CONF_NFCDEP** 0x40U

#define **RFAL_NFCA_SEL_RES_CONF_T4T_NFCDEP** 0x60U

#define **RFAL_NFCA_FDTMIN** 1620U

#define **rfalNfcaIsSensResT1T**(sensRes) ((((**rfalNfcaSensRes***)(sensRes))->platformInfo & **RFAL_NFCA_SENS_RES_PLATFORM_MASK**) == **RFAL_NFCA_SENS_RES_PLATFORM_T1T**)

#define **rfalNfcaIsSelResT2T**(selRes) ((((**rfalNfcaSelRes***)(selRes))->sak & **RFAL_NFCA_SEL_RES_CONF_MASK**) == **RFAL_NFCA_SEL_RES_CONF_T2T**)

#define **rfalNfcaIsSelResT4T**(selRes) ((((**rfalNfcaSelRes***)(selRes))->sak & **RFAL_NFCA_SEL_RES_CONF_MASK**) == **RFAL_NFCA_SEL_RES_CONF_T4T**)

#define **rfalNfcaIsSelResNFCDEP**(selRes) ((((**rfalNfcaSelRes***)(selRes))->sak & **RFAL_NFCA_SEL_RES_CONF_MASK**) == **RFAL_NFCA_SEL_RES_CONF_NFCDEP**)

#define **rfalNfcaIsSelResT4TNFCDEP**(selRes) ((((**rfalNfcaSelRes***)(selRes))->sak & **RFAL_NFCA_SEL_RES_CONF_MASK**) == **RFAL_NFCA_SEL_RES_CONF_T4T_NFCDEP**)

#define **rfalNfcaLisDevIsMultiProto**(lisDev) (((**rfalNfcaListenDevice***)(lisDev))->type == RFAL_NFCA_T4T_NFCDEP)

Enumerations

enum **rfalNfcaListenDeviceType** {
RFAL_NFCA_T1T = 0x01 , **RFAL_NFCA_T2T** = 0x00 , **RFAL_NFCA_T4T** = 0x20 , **RFAL_NFCA_NFCDEP** = 0x40 ,
RFAL_NFCA_T4T_NFCDEP = 0x60
}

Functions

ReturnCode **rfalNfcaPollerInitialize** (void)

Initialize NFC-A Poller mode. More...

ReturnCode **rfalNfcaPollerCheckPresence** (**rfal14443AShortFrameCmd** cmd, **rfalNfcaSensRes** *sensRes)
NFC-A Poller Check Presence. More...

ReturnCode **rfalNfcaPollerSelect** (const uint8_t *nfcid1, uint8_t nfcidLen, **rfalNfcaSelRes** *selRes)
NFC-A Poller Select. More...

ReturnCode **rfalNfcaPollerStartSelect** (const uint8_t *nfcid1, uint8_t nfcidLen, **rfalNfcaSelRes** *selRes)
NFC-A Poller Start Select. More...

ReturnCode **rfalNfcaPollerGetSelectStatus** (void)
NFC-A Poller Get Select Status. More...

ReturnCode **rfalNfcaPollerSleep** (void)
NFC-A Poller Sleep. More...

ReturnCode **rfalNfcaPollerStartSleep** (void)
NFC-A Poller Start Sleep. More...

ReturnCode **rfalNfcaPollerGetSleepStatus** (void)
NFC-A Poller Get Sleep Status. More...

ReturnCode **rfalNfcaPollerTechnologyDetection** (**rfalComplianceMode** compMode, **rfalNfcaSensRes** *sensRes)
NFC-A Technology Detection. More...

ReturnCode **rfalNfcaPollerStartTechnologyDetection** (**rfalComplianceMode** compMode, **rfalNfcaSensRes** *sensRes)
NFC-A Start Technology Detection. More...

ReturnCode **rfalNfcaPollerGetTechnologyDetectionStatus** (void)
NFC-A Get Technology Detection Status. More...

ReturnCode **rfalNfcaPollerSingleCollisionResolution** (uint8_t devLimit, bool *collPending, **rfalNfcaSelRes** *selRes, uint8_t *nfcId1, uint8_t *nfcId1Len)
NFC-A Poller Collision Resolution. More...

ReturnCode **rfalNfcaPollerFullCollisionResolution** (**rfalComplianceMode** compMode, uint8_t devLimit, **rfalNfcaListenDevice** *nfcaDevList, uint8_t *devCnt)
NFC-A Poller Full Collision Resolution. More...

ReturnCode	**rfalNfcaPollerSleepFullCollisionResolution** (uint8_t devLimit, **rfalNfcaListenDevice** *nfcaDevList, uint8_t *devCnt) NFC-A Poller Full Collision Resolution with Sleep. More...
ReturnCode	**rfalNfcaPollerStartFullCollisionResolution** (**rfalComplianceMode** compMode, uint8_t devLimit, **rfalNfcaListenDevice** *nfcaDevList, uint8_t *devCnt) NFC-A Poller Start Full Collision Resolution. More...
ReturnCode	**rfalNfcaPollerGetFullCollisionResolutionStatus** (void) NFC-A Get Full Collision Resolution Status. More...
bool	**rfalNfcaListenerIsSleepReq** (const uint8_t *buf, uint16_t bufLen) NFC-A Listener is SLP_REQ. More...

Detailed Description

Macro Definition Documentation

◆ RFAL_NFCA_CASCADE_1_UID_LEN

```
#define RFAL_NFCA_CASCADE_1_UID_LEN   4U
```

UID length of cascade level 1 only tag

Examples

exampleRfalPoller.c.

◆ RFAL_NFCA_CASCADE_2_UID_LEN

```
#define RFAL_NFCA_CASCADE_2_UID_LEN   7U
```

UID length of cascade level 2 only tag

◆ RFAL_NFCA_CASCADE_3_UID_LEN

```
#define RFAL_NFCA_CASCADE_3_UID_LEN   10U
```

UID length of cascade level 3 only tag

◆ RFAL_NFCA_FDTMIN

```
#define RFAL_NFCA_FDTMIN   1620U
```

NFC-A minimum FDT(listen) = ((n * 128 + (84)) / fc) with n_min = 9 Digital 1.1 6.10.1 = (1236)/fc Relax with 3etu: (3*128)/fc as with multiple NFC-A cards, response may take longer (JCOP cards) = (1236 + 384)/fc = 1620 / fc

◆ RFAL_NFCA_SEL_RES_CONF_MASK

```
#define RFAL_NFCA_SEL_RES_CONF_MASK   0x60U
```

SEL_RES (SAK) platform configuration mask Digital 1.1 Table 19

◆ RFAL_NFCA_SEL_RES_CONF_NFCDEP

```
#define RFAL_NFCA_SEL_RES_CONF_NFCDEP   0x40U
```

SEL_RES (SAK) NFC-DEP configuration Digital 1.1 Table 19

◆ RFAL_NFCA_SEL_RES_CONF_T2T

```
#define RFAL_NFCA_SEL_RES_CONF_T2T   0x00U
```

SEL_RES (SAK) T2T configuration Digital 1.1 Table 19

◆ RFAL_NFCA_SEL_RES_CONF_T4T

```
#define RFAL_NFCA_SEL_RES_CONF_T4T   0x20U
```

SEL_RES (SAK) T4T configuration Digital 1.1 Table 19

◆ RFAL_NFCA_SEL_RES_CONF_T4T_NFCDEP

```
#define RFAL_NFCA_SEL_RES_CONF_T4T_NFCDEP   0x60U
```

SEL_RES (SAK) T4T and NFC-DEP configuration Digital 1.1 Table 19

◆ RFAL_NFCA_SENS_RES_PLATFORM_MASK

```
#define RFAL_NFCA_SENS_RES_PLATFORM_MASK   0x0FU
```

SENS_RES (ATQA) platform configuration mask Digital 1.1 Table 10

◆ RFAL_NFCA_SENS_RES_PLATFORM_T1T

```
#define RFAL_NFCA_SENS_RES_PLATFORM_T1T   0x0CU
```

SENS_RES (ATQA) T1T platform configuration Digital 1.1 Table 10

◆ rfalNfcaIsSelResNFCDEP

```
#define rfalNfcaIsSelResNFCDEP ( selRes ) (((rfalNfcaSelRes*)(selRes))->sak & RFAL_NFCA_SEL_RES_CONF_MASK) == RFAL_NFCA_SEL_RES_CONF_NFCDEP )
```

Checks if device supports NFC-DEP protocol given its SENS_RES

◆ rfalNfcaIsSelResT2T

```
#define rfalNfcaIsSelResT2T ( selRes ) (((rfalNfcaSelRes*)(selRes))->sak & RFAL_NFCA_SEL_RES_CONF_MASK) == RFAL_NFCA_SEL_RES_CONF_T2T )
```

Checks if device is a T2T given its SENS_RES

◆ rfalNfcaIsSelResT4T

```
#define rfalNfcaIsSelResT4T ( selRes ) (((rfalNfcaSelRes*)(selRes))->sak & RFAL_NFCA_SEL_RES_CONF_MASK) == RFAL_NFCA_SEL_RES_CONF_T4T )
```

Checks if device is a T4T given its SENS_RES

◆ rfalNfcaIsSelResT4TNFCDEP

```
#define rfalNfcaIsSelResT4TNFCDEP ( selRes ) ((((rfalNfcaSelRes*)(selRes))->sak & RFAL_NFCA_SEL_RES_CONF_MASK) == RFAL_NFCA_SEL_RES_CONF_T4T_NFCDEP )
```

Checks if device supports ISO-DEP and NFC-DEP protocol given its SENS_RES

◆ rfalNfcaIsSensResT1T

```
#define rfalNfcaIsSensResT1T ( sensRes ) ((((rfalNfcaSensRes*)(sensRes))->platformInfo & RFAL_NFCA_SENS_RES_PLATFORM_MASK) == RFAL_NFCA_SENS_RES_PLATFORM_T1T )
```

Checks if device is a T1T given its SENS_RES

◆ rfalNfcaLisDevIsMultiProto

```
#define rfalNfcaLisDevIsMultiProto ( lisDev ) (((rfalNfcaListenDevice*)(lisDev))->type == RFAL_NFCA_T4T_NFCDEP )
```

Checks if a NFC-A listener device supports multiple protocols (ISO-DEP and NFC-DEP)

Enumeration Type Documentation

◆ rfalNfcaListenDeviceType

```
enum rfalNfcaListenDeviceType
```

NFC-A Listen device types

Function Documentation

◆ rfalNfcaListenerIsSleepReq()

```
bool rfalNfcaListenerIsSleepReq ( const uint8_t * buf,
```

```
                                    uint16_t          bufLen
                                )
```

Checks if the given buffer contains valid NFC-A SLP_REQ (HALT)

Parameters

[in] **buf** buffer containing data

[in] **bufLen** length of the data in buffer to be checked

Returns

true if data in buf contains a SLP_REQ ; false otherwise

◆ rfalNfcaPollerCheckPresence()

```
ReturnCode rfalNfcaPollerCheckPresence ( rfal14443AShortFrameCmd  cmd,
                                         rfalNfcaSensRes *        sensRes
                                       )
```

This method checks if a NFC-A Listen device (PICC) is present on the field by sending an ALL_REQ (WUPA) or SENS_REQ (REQA)

Parameters

[in] **cmd** : Indicate if to send an ALL_REQ or a SENS_REQ

[out] **sensRes** : If received, the SENS_RES

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_RF_COLLISION : Collision detected one or more device in the field

RFAL_ERR_PAR : Parity error detected, one or more device in the field

RFAL_ERR_CRC : CRC error detected, one or more device in the field

RFAL_ERR_FRAMING : Framing error detected, one or more device in the field

RFAL_ERR_PROTO : Protocol error detected, one or more device in the field

RFAL_ERR_TIMEOUT : Timeout error, no listener device detected

RFAL_ERR_NONE : No error, one or more device in the field

Examples

exampleRfalNfca.c, and exampleRfalPoller.c.

◆ rfalNfcaPollerFullCollisionResolution()

```
ReturnCode rfalNfcaPollerFullCollisionResolution ( rfalComplianceMode      compMode,
                                                   uint8_t                 devLimit,
                                                   rfalNfcaListenDevice *  nfcaDevList,
                                                   uint8_t *               devCnt
                                                 )
```

Performs a full Collision resolution as defined in Activity 2.1 9.3.4

Parameters

[in]	**compMode**	: compliance mode to be performed
[in]	**devLimit**	: device limit value, and size nfcaDevList
[out]	**nfcaDevList**	: NFC-A listener device info
[out]	**devCnt**	: Devices found counter

When compMode is set to ISO compliance it assumes that the device is not sleeping and therefore no ALL_REQ (WUPA) is sent at the beginning. When compMode is set to NFC compliance an additional ALL_REQ (WUPA) is sent at the beginning.

When devLimit = 0 it is configured to perform collision detection only. Once a collision is detected the collision resolution is aborted immidiatly. If only one device is found with no collisions, it will properly resolved.

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or mode not set

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_NONE : No error

Examples

exampleRfalNfca.c, and exampleRfalPoller.c.

◆ rfalNfcaPollerGetFullCollisionResolutionStatus()

```
ReturnCode rfalNfcaPollerGetFullCollisionResolutionStatus ( void )
```

Returns the Collision Resolution status

Returns

RFAL_ERR_BUSY : Operation is ongoing

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_PAR : Parity error detected

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_NONE : No error, activation successful

◆ rfalNfcaPollerGetSelectStatus()

ReturnCode rfalNfcaPollerGetSelectStatus (void)

This method gets the selection status

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_PAR : Parity error detected

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_NONE : No error, SEL_RES received

◆ rfalNfcaPollerGetSleepStatus()

ReturnCode rfalNfcaPollerGetSleepStatus (void)

Returns the Sleep status

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_NONE : No error

◆ rfalNfcaPollerGetTechnologyDetectionStatus()

ReturnCode rfalNfcaPollerGetTechnologyDetectionStatus (void)

Returns the Technology Detection status

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_NONE : No error, one or more device in the field

◆ rfalNfcaPollerInitialize()

ReturnCode rfalNfcaPollerInitialize (void)

This methods configures RFAL RF layer to perform as a NFC-A Poller/RW (ISO14443A PCD) including all default timings and bit rate to 106 kbps

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or mode not set

RFAL_ERR_NONE : No error

Examples

exampleRfalNfca.c, and **exampleRfalPoller.c**.

◆ rfalNfcaPollerSelect()

ReturnCode rfalNfcaPollerSelect (const uint8_t * nfcid1,

```
                                uint8_t            nfcidLen,
                                rfalNfcaSelRes *   selRes
                              )
```

This method selects a NFC-A Listener device (PICC)

Parameters

- [in] **nfcid1** : Listener device NFCID1 to be selected
- [in] **nfcidLen** : Length of the NFCID1 to be selected
- [out] **selRes** : pointer to place the SEL_RES

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_PAR : Parity error detected

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_NONE : No error, SEL_RES received

Examples

exampleRfalNfca.c, and **exampleRfalPoller.c**.

◆ rfalNfcaPollerSingleCollisionResolution()

```
ReturnCode rfalNfcaPollerSingleCollisionResolution ( uint8_t            devLimit,
                                                     bool *             collPending,
                                                     rfalNfcaSelRes *   selRes,
                                                     uint8_t *          nfcId1,
                                                     uint8_t *          nfcId1Len
                                                   )
```

Collision resolution for one NFC-A Listener device/card (PICC) as defined in Activity 2.1 9.3.4

This method executes anti collision loop and select the device with higher NFCID1

When devLimit = 0 it is configured to perform collision detection only. Once a collision is detected the collision resolution is aborted immidiatly. If only one device is found with no collisions, it will properly resolved.

Parameters

- [in] **devLimit** : device limit value (CON_DEVICES_LIMIT)
- [out] **collPending** : pointer to collision pending flag (INT_COLL_PEND)
- [out] **selRes** : location to store the last Select Response from listener device (PICC)
- [out] **nfcId1** : location to store the NFCID1 (UID), ensure RFAL_NFCA_CASCADE_3_UID_LEN
- [out] **nfcId1Len** : pointer to length of NFCID1 (UID)

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or mode not set

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_PROTO : Card length invalid

RFAL_ERR_IGNORE : conDevLimit is 0 and there is a collision

RFAL_ERR_NONE : No error

◆ rfalNfcaPollerSleep()

ReturnCode rfalNfcaPollerSleep (void)

This method sends a SLP_REQ (HLTA) No response is expected afterwards Digital 1.1 6.9.2.1

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_NONE : No error

Examples

exampleRfalNfca.c.

◆ rfalNfcaPollerSleepFullCollisionResolution()

ReturnCode rfalNfcaPollerSleepFullCollisionResolution (uint8_t devLimit,
rfalNfcaListenDevice * nfcaDevList,

```
                                                    uint8_t *                devCnt
                                                )
```

Performs a full Collision resolution similar to rfalNfcaPollerFullCollisionResolution but an additional SLP_REQ (HLTA) -> SENS_RES (REQA) is sent regardless if there was a collision. This proprietary behaviour ensures proper activation of certain devices that suffer from influence of Type B commands as foreseen in ISO14443-3 5.2.3 or were somehow not detected by the first round of collision resolution

Parameters

- [in] **devLimit** : device limit value, and size nfcaDevList
- [out] **nfcaDevList** : NFC-A listener device info
- [out] **devCnt** : Devices found counter

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or mode not set

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_NONE : No error

◆ rfalNfcaPollerStartFullCollisionResolution()

```
ReturnCode rfalNfcaPollerStartFullCollisionResolution ( rfalComplianceMode      compMode,
                                                        uint8_t                 devLimit,
                                                        rfalNfcaListenDevice *  nfcaDevList,
                                                        uint8_t *               devCnt
                                                      )
```

This method starts the full Collision resolution as defined in Activity 1.0 or 1.1 9.3.4

Parameters

- [in] **compMode** : compliance mode to be performed
- [in] **devLimit** : device limit value, and size nfcaDevList
- [out] **nfcaDevList** : NFC-A listener device info
- [out] **devCnt** : Devices found counter

When compMode is set to ISO compliance it assumes that the device is not sleeping and therefore no ALL_REQ (WUPA) is sent at the beginning. When compMode is set to NFC compliance an additional ALL_REQ (WUPA) is sent at the beginning.

When devLimit = 0 it is configured to perform collision detection only. Once a collision is detected the collision resolution is aborted immidiatly. If only one device is found with no collisions, it will properly resolved.

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or mode not set

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_NONE : No error

◆ rfalNfcaPollerStartSelect()

```
ReturnCode rfalNfcaPollerStartSelect ( const uint8_t *      nfcid1,
                                       uint8_t              nfcidLen,
                                       rfalNfcaSelRes *     selRes
                                     )
```

This method starts the selection of a NFC-A Listener device (PICC)

Parameters

[in] **nfcid1** : Listener device NFCID1 to be selected

[in] **nfcidLen** : Length of the NFCID1 to be selected

[out] **selRes** : pointer to place the SEL_RES

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_PAR : Parity error detected

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_NONE : No error, SEL_RES received

◆ rfalNfcaPollerStartSleep()

ReturnCode rfalNfcaPollerStartSleep (void)

This method sends a SLP_REQ (HLTA) No response is expected afterwards Digital 1.1 6.9.2.1

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_NONE : No error

◆ rfalNfcaPollerStartTechnologyDetection()

ReturnCode rfalNfcaPollerStartTechnologyDetection (**rfalComplianceMode** compMode, **rfalNfcaSensRes** * sensRes)

This method starts NFC-A Technology Detection as defined in the spec given in the compliance mode

Parameters

[in] **compMode** : compliance mode to be performed

[out] **sensRes** : location to store the SENS_RES, if received

When compMode is set to ISO compliance a SLP_REQ (HLTA) is not sent after detection. When set to EMV a ALL_REQ (WUPA) is sent instead of a SENS_REQ (REQA)

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_NONE : No error, one or more device in the field

◆ rfalNfcaPollerTechnologyDetection()

ReturnCode rfalNfcaPollerTechnologyDetection (**rfalComplianceMode** compMode, **rfalNfcaSensRes** * sensRes)

This method performs NFC-A Technology Detection as defined in the spec given in the compliance mode

Parameters

[in] **compMode** : compliance mode to be performed

[out] **sensRes** : location to store the SENS_RES, if received

When compMode is set to ISO compliance a SLP_REQ (HLTA) is not sent after detection. When set to EMV a ALL_REQ (WUPA) is sent instead of a SENS_REQ (REQA)

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_NONE : No error, one or more device in the field

Examples

exampleRfalNfca.c, and **exampleRfalPoller.c**.

rfalNfcaSensRes Struct Reference


```
#include <rfal_nfca.h>
```

Data Fields

uint8_t	**anticollisionInfo**
uint8_t	**platformInfo**

Detailed Description

SENS_RES (ATQA) format Digital 1.1 6.6.3 & Table 7

Examples

exampleRfalNfca.c, and **exampleRfalPoller.c**.

Field Documentation

◆ anticollisionInfo

uint8_t rfalNfcaSensRes::anticollisionInfo

SENS_RES Anticollision Information

◆ platformInfo

uint8_t rfalNfcaSensRes::platformInfo

SENS_RES Platform Information

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

- **rfal_nfca.h**

RFAL

Main Page

Modules

Files

Examples

Data Structures

Data Structure Index

Data Fields

Data Fields

rfalNfcaSddReq Struct Reference

RFAL » RFAL-AL » NFC-A

```
#include <rfal_nfca.h>
```

Data Fields

uint8_t **selCmd**

uint8_t **selPar**

Detailed Description

SDD_REQ (Anticollision) format Digital 1.1 6.7.1 & Table 11

Field Documentation

◆ selCmd

uint8_t rfalNfcaSddReq::selCmd

SDD_REQ SEL_CMD: cascade Level

◆ selPar

uint8_t rfalNfcaSddReq::selPar

SDD_REQ SEL_PAR: Byte Count[4b] | Bit Count[4b] (NVB: Number of Valid Bits)

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

- **rfal_nfca.h**

---

rfalNfcaSddRes Struct Reference


```
#include <rfal_nfca.h>
```

Data Fields

uint8_t	**nfcid1** [**RFAL_NFCA_CASCADE_1_UID_LEN**]
uint8_t	**bcc**

Detailed Description

SDD_RES (UID CLn) format Digital 1.1 6.7.2 & Table 15

Field Documentation

◆ bcc

uint8_t rfalNfcaSddRes::bcc

BCC Exclusive-OR over first 4 bytes of SDD_RES

◆ nfcid1

uint8_t rfalNfcaSddRes::nfcid1[**RFAL_NFCA_CASCADE_1_UID_LEN**]

NFCID1 cascade level NFCID

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

- **rfal_nfca.h**

RFAL

rfalNfcaSelReq Struct Reference

RFAL » RFAL-AL » NFC-A

```
#include <rfal_nfca.h>
```

Data Fields

uint8_t	**selCmd**
uint8_t	**selPar**
uint8_t	**nfcid1** [**RFAL_NFCA_CASCADE_1_UID_LEN**]
uint8_t	**bcc**

Detailed Description

SEL_REQ (Select) format Digital 1.1 6.8.1 & Table 17

Field Documentation

◆ bcc

uint8_t rfalNfcaSelReq::bcc

Checksum calculated as exclusive-OR over the 4 bytes of NFCID1 CLn

◆ nfcid1

uint8_t rfalNfcaSelReq::nfcid1[**RFAL_NFCA_CASCADE_1_UID_LEN**]

NFCID1 data

◆ selCmd

uint8_t rfalNfcaSelReq::selCmd

SDD_REQ SEL_CMD: cascade Level

◆ selPar

uint8_t rfalNfcaSelReq::selPar

SDD_REQ SEL_PAR: Byte Count[4b] | Bit Count[4b] (NVB: Number of Valid Bits)

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

- **rfal_nfca.h**

rfalNfcaSelRes Struct Reference


```
#include <rfal_nfca.h>
```

Data Fields

uint8_t	**sak**

Detailed Description

SEL_RES (SAK) format Digital 1.1 6.8.2 & Table 19

Examples

exampleRfalNfca.c, and **exampleRfalPoller.c**.

Field Documentation

◆ sak

uint8_t rfalNfcaSelRes::sak

Select Acknowledge

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

- **rfal_nfca.h**

---

rfalNfcaListenDevice Struct Reference

RFAL » **RFAL-AL** » **NFC-A**

```
#include <rfal_nfca.h>
```

Collaboration diagram for rfalNfcaListenDevice:

[legend]

Data Fields

rfalNfcaListenDeviceType	**type**
rfalNfcaSensRes	**sensRes**
rfalNfcaSelRes	**selRes**
uint8_t	**nfcId1Len**
uint8_t	**nfcId1** [**RFAL_NFCA_CASCADE_3_UID_LEN**]
bool	**isSleep**

Detailed Description

NFC-A listener device (PICC) struct

Examples

exampleRfalNfca.c, and **exampleRfalPoller.c**.

Field Documentation

◆ isSleep

bool rfalNfcaListenDevice::isSleep

Device sleeping flag

◆ nfcId1

uint8_t rfalNfcaListenDevice::nfcId1[**RFAL_NFCA_CASCADE_3_UID_LEN**]

NFCID1 (UID)

◆ nfcId1Len

uint8_t rfalNfcaListenDevice::nfcId1Len

NFCID1 Length

◆ selRes

rfalNfcaSelRes rfalNfcaListenDevice::selRes

SEL_RES (SAK)

◆ sensRes

rfalNfcaSensRes rfalNfcaListenDevice::sensRes

SENS_RES (ATQA)

◆ type

rfalNfcaListenDeviceType rfalNfcaListenDevice::type

NFC-A Listen device type

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

- **rfal_nfca.h**

NFC-B

RFAL NFC-B Module. More...

Collaboration diagram for NFC-B:

Data Structures

struct	**rfalNfcbSensbResAppData**
struct	**rfalNfcbSensbResProtocolInfo**
struct	**rfalNfcbSensbRes**
struct	**rfalNfcbListenDevice**

Macros

#define	**RFAL_NFCB_FWTSENSB** 7680U
#define	**RFAL_NFCB_DFWT** 49152U
#define	**RFAL_NFCB_DTPOLL_10** **rfalConvMsTo1fc**(20)
#define	**RFAL_NFCB_DTPOLL_20** **rfalConvMsTo1fc**(17)

```
#define RFAL_NFCB_AFI 0x00U
#define RFAL_NFCB_PARAM 0x00U
#define RFAL_NFCB_CRC_LEN 2U
#define RFAL_NFCB_NFCID0_LEN 4U
#define RFAL_NFCB_CMD_LEN 1U
#define RFAL_NFCB_SENSB_RES_LEN 12U
#define RFAL_NFCB_SENSB_RES_EXT_LEN 13U
#define RFAL_NFCB_SENSB_REQ_ADV_FEATURE 0x20U
#define RFAL_NFCB_SENSB_RES_FSCI_MASK 0x0FU
#define RFAL_NFCB_SENSB_RES_FSCI_SHIFT 4U
#define RFAL_NFCB_SENSB_RES_PROTO_RFU_MASK 0x08U
#define RFAL_NFCB_SENSB_RES_PROTO_TR2_MASK 0x03U
#define RFAL_NFCB_SENSB_RES_PROTO_TR2_SHIFT 1U
#define RFAL_NFCB_SENSB_RES_PROTO_ISO_MASK 0x01U
#define RFAL_NFCB_SENSB_RES_FWI_MASK 0x0FU
#define RFAL_NFCB_SENSB_RES_FWI_SHIFT 4U
#define RFAL_NFCB_SENSB_RES_ADC_MASK 0x0CU
#define RFAL_NFCB_SENSB_RES_ADC_ADV_FEATURE_MASK 0x08U
#define RFAL_NFCB_SENSB_RES_ADC_PROPRIETARY_MASK 0x04U
#define RFAL_NFCB_SENSB_RES_FO_DID_MASK 0x01U
#define RFAL_NFCB_SENSB_RES_FO_NAD_MASK 0x02U
```

#define **RFAL_NFCB_SENSB_RES_FO_MASK** 0x03U

#define **RFAL_NFCB_SENSB_RES_SFGI_MASK** 0x0FU

#define **RFAL_NFCB_SENSB_RES_SFGI_SHIFT** 4U

#define **rfalNfcbGetFSCI**(sensbRes) ((((**rfalNfcbSensbRes***)(sensbRes))->protInfo.FsciProType >> **RFAL_NFCB_SENSB_RES_FSCI_SHIFT**) & **RFAL_NFCB_SENSB_RES_FSCI_MASK**)

#define **rfalNfcbIsIsoDepSupported**(dev) ((((**rfalNfcbListenDevice***)(dev))->sensbRes.protInfo.FsciProType & **RFAL_NFCB_SENSB_RES_PROTO_ISO_MASK**) != 0U)

Enumerations

enum **rfalNfcbSensCmd** { **RFAL_NFCB_SENS_CMD_ALLB_REQ** = 0x08 , **RFAL_NFCB_SENS_CMD_SENSB_REQ** = 0x00 }

enum **rfalNfcbSlots** {
RFAL_NFCB_SLOT_NUM_1 = 0 , **RFAL_NFCB_SLOT_NUM_2** = 1 , **RFAL_NFCB_SLOT_NUM_4** = 2 , **RFAL_NFCB_SLOT_NUM_8** = 3 ,
RFAL_NFCB_SLOT_NUM_16 = 4
}

Functions

ReturnCode **rfalNfcbPollerInitialize** (void)
Initialize NFC-B Poller mode. More...

ReturnCode **rfalNfcbPollerInitializeWithParams** (uint8_t AFI, uint8_t PARAM)
Set NFC-B Poller parameters. More...

ReturnCode **rfalNfcbPollerCheckPresence** (**rfalNfcbSensCmd** cmd, **rfalNfcbSlots** slots, **rfalNfcbSensbRes** *sensbRes, uint8_t *sensbResLen)
NFC-B Poller Check Presence. More...

ReturnCode **rfalNfcbPollerStartCheckPresence** (**rfalNfcbSensCmd** cmd, **rfalNfcbSlots** slots, **rfalNfcbSensbRes** *sensbRes, uint8_t *sensbResLen)
NFC-B Poller Start Check Presence. More...

ReturnCode **rfalNfcbPollerGetCheckPresenceStatus** (void)
NFC-B Poller Get Check Presence Status. More...

ReturnCode **rfalNfcbPollerSleep** (const uint8_t *nfcid0)
NFC-B Poller Sleep. More...

ReturnCode **rfalNfcbPollerSlotMarker** (uint8_t slotCode, **rfalNfcbSensbRes** *sensbRes, uint8_t *sensbResLen)
NFC-B Poller Slot Marker. More...

ReturnCode **rfalNfcbPollerStartSlotMarker** (uint8_t slotCode, **rfalNfcbSensbRes** *sensbRes, uint8_t *sensbResLen)
NFC-B Poller Start Slot Marker. More...

ReturnCode **rfalNfcbPollerGetSlotMarkerStatus** (void)
NFC-B Poller Get Slot Marker Status. More...

ReturnCode **rfalNfcbPollerTechnologyDetection** (**rfalComplianceMode** compMode, **rfalNfcbSensbRes** *sensbRes, uint8_t *sensbResLen)
NFC-B Technology Detection. More...

ReturnCode **rfalNfcbPollerStartTechnologyDetection** (**rfalComplianceMode** compMode, **rfalNfcbSensbRes** *sensbRes, uint8_t *sensbResLen)
NFC-B Start Technology Detection. More...

ReturnCode **rfalNfcbPollerGetTechnologyDetectionStatus** (void)
NFC-B Get Technology Detection Status. More...

ReturnCode **rfalNfcbPollerCollisionResolution** (**rfalComplianceMode** compMode, uint8_t devLimit, **rfalNfcbListenDevice** *nfcbDevList, uint8_t *devCnt)
NFC-B Poller Collision Resolution. More...

ReturnCode **rfalNfcbPollerSlottedCollisionResolution** (**rfalComplianceMode** compMode, uint8_t devLimit, **rfalNfcbSlots** initSlots, **rfalNfcbSlots** endSlots, **rfalNfcbListenDevice** *nfcbDevList, uint8_t *devCnt, bool *colPending)
NFC-B Poller Collision Resolution Slotted. More...

ReturnCode **rfalNfcbPollerStartCollisionResolution** (**rfalComplianceMode** compMode, uint8_t devLimit, **rfalNfcbListenDevice** *nfcbDevList, uint8_t *devCnt)
NFC-B Poller Start Collision Resolution. More...

ReturnCode	**rfalNfcbPollerStartSlottedCollisionResolution** (**rfalComplianceMode** compMode, uint8_t devLimit, **rfalNfcbSlots** initSlots, **rfalNfcbSlots** endSlots, **rfalNfcbListenDevice** *nfcbDevList, uint8_t *devCnt, bool *colPending) NFC-B Poller Start Collision Resolution Slotted. More...
ReturnCode	**rfalNfcbPollerGetCollisionResolutionStatus** (void) NFC-B Get Collision Resolution Status. More...
uint32_t	**rfalNfcbTR2ToFDT** (uint8_t tr2Code) NFC-B TR2 code to FDT. More...

Detailed Description

Macro Definition Documentation

◆ RFAL_NFCB_AFI

```
#define RFAL_NFCB_AFI   0x00U
```

NFC-B default Application Family Digital 1.1 7.6.1.1

◆ RFAL_NFCB_CMD_LEN

```
#define RFAL_NFCB_CMD_LEN   1U
```

Length of NFC-B Command

◆ RFAL_NFCB_CRC_LEN

```
#define RFAL_NFCB_CRC_LEN   2U
```

NFC-B CRC length and CRC_B(AID) Digital 1.1 Table 28

◆ RFAL_NFCB_DFWT

```
#define RFAL_NFCB_DFWT   49152U
```

NFC-B dFWT Delta 2.0 7.9.1.3 & B.3

◆ RFAL_NFCB_DTPOLL_10

```
#define RFAL_NFCB_DTPOLL_10   rfalConvMsTo1fc(20)
```

NFC-B Delta Tb Poll Digital 1.0 A.2

◆ RFAL_NFCB_DTPOLL_20

```
#define RFAL_NFCB_DTPOLL_20   rfalConvMsTo1fc(17)
```

NFC-B Delta Tb Poll Digital 2.1 B.3

◆ RFAL_NFCB_FWTSENSB

```
#define RFAL_NFCB_FWTSENSB   7680U
```

NFC-B FWT(SENSB) Digital 2.0 B.3

◆ RFAL_NFCB_NFCID0_LEN

```
#define RFAL_NFCB_NFCID0_LEN   4U
```

Length of NFC-B NFCID0

◆ RFAL_NFCB_PARAM

```
#define RFAL_NFCB_PARAM   0x00U
```

NFC-B default SENSB_REQ PARAM

◆ RFAL_NFCB_SENSB_REQ_ADV_FEATURE

```
#define RFAL_NFCB_SENSB_REQ_ADV_FEATURE   0x20U
```

Bit mask for Advance Feature in SENSB_REQ

◆ RFAL_NFCB_SENSB_RES_ADC_ADV_FEATURE_MASK

```
#define RFAL_NFCB_SENSB_RES_ADC_ADV_FEATURE_MASK   0x08U
```

Bit mask for ADC.Advanced Proto Features in SENSB_RES

◆ RFAL_NFCB_SENSB_RES_ADC_MASK

```
#define RFAL_NFCB_SENSB_RES_ADC_MASK   0x0CU
```

Bit mask for ADC value in SENSB_RES

◆ RFAL_NFCB_SENSB_RES_ADC_PROPRIETARY_MASK

```
#define RFAL_NFCB_SENSB_RES_ADC_PROPRIETARY_MASK   0x04U
```

Bit mask for ADC.Proprietary Application in SENSB_RES

◆ RFAL_NFCB_SENSB_RES_EXT_LEN

```
#define RFAL_NFCB_SENSB_RES_EXT_LEN   13U
```

Extended length of SENSB_RES with SFGI byte

◆ RFAL_NFCB_SENSB_RES_FO_DID_MASK

```
#define RFAL_NFCB_SENSB_RES_FO_DID_MASK   0x01U
```

Bit mask for DID in SENSB_RES

◆ RFAL_NFCB_SENSB_RES_FO_MASK

```
#define RFAL_NFCB_SENSB_RES_FO_MASK   0x03U
```

Bit mask for FO value in SENSB_RES (NAD and DID)

◆ RFAL_NFCB_SENSB_RES_FO_NAD_MASK

```
#define RFAL_NFCB_SENSB_RES_FO_NAD_MASK   0x02U
```

Bit mask for DID in SENSB_RES

◆ RFAL_NFCB_SENSB_RES_FSCI_MASK

```
#define RFAL_NFCB_SENSB_RES_FSCI_MASK   0x0FU
```

Bit mask for FSCI value in SENSB_RES

◆ RFAL_NFCB_SENSB_RES_FSCI_SHIFT

```
#define RFAL_NFCB_SENSB_RES_FSCI_SHIFT   4U
```

Shift for FSCI value in SENSB_RES

◆ RFAL_NFCB_SENSB_RES_FWI_MASK

```
#define RFAL_NFCB_SENSB_RES_FWI_MASK   0x0FU
```

Bit mask for FWI value in SENSB_RES

◆ RFAL_NFCB_SENSB_RES_FWI_SHIFT

```
#define RFAL_NFCB_SENSB_RES_FWI_SHIFT   4U
```

Bit mask for FWI value in SENSB_RES

◆ RFAL_NFCB_SENSB_RES_LEN

```
#define RFAL_NFCB_SENSB_RES_LEN   12U
```

Standard length of SENSB_RES without SFGI byte

◆ RFAL_NFCB_SENSB_RES_PROTO_ISO_MASK

```
#define RFAL_NFCB_SENSB_RES_PROTO_ISO_MASK   0x01U
```

Bit mask Protocol Type ISO14443 Compliant in SENSB_RES

◆ RFAL_NFCB_SENSB_RES_PROTO_RFU_MASK

```
#define RFAL_NFCB_SENSB_RES_PROTO_RFU_MASK   0x08U
```

Bit mask for Protocol Type RFU in SENSB_RES

◆ RFAL_NFCB_SENSB_RES_PROTO_TR2_MASK

```
#define RFAL_NFCB_SENSB_RES_PROTO_TR2_MASK   0x03U
```

Bit mask for Protocol Type TR2 in SENSB_RES

◆ RFAL_NFCB_SENSB_RES_PROTO_TR2_SHIFT

```
#define RFAL_NFCB_SENSB_RES_PROTO_TR2_SHIFT   1U
```

Shift for Protocol Type TR2 in SENSB_RES

◆ RFAL_NFCB_SENSB_RES_SFGI_MASK

```
#define RFAL_NFCB_SENSB_RES_SFGI_MASK   0x0FU
```

Bit mask for SFGI in SENSB_RES

◆ RFAL_NFCB_SENSB_RES_SFGI_SHIFT

```
#define RFAL_NFCB_SENSB_RES_SFGI_SHIFT   4U
```

Shift for SFGI in SENSB_RES

◆ rfalNfcbGetFSCI

```
#define rfalNfcbGetFSCI ( sensbRes ) ((((rfalNfcbSensbRes*)(sensbRes))->protInfo.FsciProType >> RFAL_NFCB_SENSB_RES_FSCI_SHIFT) & RFAL_NFCB_SENSB_RES_FSCI_MASK)
```

Get device's FSCI given its SENSB_RES Digital 1.1 7.6.2

◆ rfalNfcbIsIsoDepSupported

```
#define rfalNfcbIsIsoDepSupported ( dev ) (((rfalNfcbListenDevice*)(dev))->sensbRes.protInfo.FsciProType & RFAL_NFCB_SENSB_RES_PROTO_ISO_MASK) != 0U )
```

Checks if the given NFC-B device indicates ISO-DEP support

Enumeration Type Documentation

◆ rfalNfcbSensCmd

enum **rfalNfcbSensCmd**

SENSB_REQ and ALLB_REQ param Digital 1.1 7.6.1

Enumerator	
RFAL_NFCB_SENS_CMD_ALLB_REQ	ALLB_REQ (WUPB)
RFAL_NFCB_SENS_CMD_SENSB_REQ	SENSB_REQ (REQB)

◆ rfalNfcbSlots

enum **rfalNfcbSlots**

Number of Slots (NI) codes used for NFC-B anti collision Digital 1.1 Table 26

Enumerator	
RFAL_NFCB_SLOT_NUM_1	N=0 : 1 slot
RFAL_NFCB_SLOT_NUM_2	N=1 : 2 slots

RFAL_NFCB_SLOT_NUM_4	N=2 : 4 slots
RFAL_NFCB_SLOT_NUM_8	N=3 : 8 slots
RFAL_NFCB_SLOT_NUM_16	N=4 : 16 slots

Function Documentation

◆ rfalNfcbPollerCheckPresence()

```
ReturnCode rfalNfcbPollerCheckPresence ( rfalNfcbSensCmd      cmd,
                                         rfalNfcbSlots        slots,
                                         rfalNfcbSensbRes *   sensbRes,
                                         uint8_t *            sensbResLen
                                       )
```

This method checks if a NFC-B Listen device (PICC) is present on the field by sending an ALLB_REQ (WUPB) or SENSB_REQ (REQB)

Parameters

- [in] **cmd** : Indicate if to send an ALLB_REQ or a SENSB_REQ
- [in] **slots** : The number of slots to be announced
- [out] **sensbRes** : If received, the SENSB_RES
- [out] **sensbResLen** : If received, the SENSB_RES length

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_TIMEOUT : Timeout error, no listener device detected

RFAL_ERR_RF_COLLISION : Collision detected one or more device in the field

RFAL_ERR_PAR : Parity error detected, one or more device in the field

RFAL_ERR_CRC : CRC error detected, one or more device in the field

RFAL_ERR_FRAMING : Framing error detected, one or more device in the field

RFAL_ERR_PROTO : Protocol error detected, invalid SENSB_RES received

RFAL_ERR_NONE : No error, SENSB_RES received

Examples

exampleRfalPoller.c.

◆ rfalNfcbPollerCollisionResolution()

```
ReturnCode rfalNfcbPollerCollisionResolution ( rfalComplianceMode      compMode,
                                               uint8_t                 devLimit,
                                               rfalNfcbListenDevice *  nfcbDevList,
                                               uint8_t *               devCnt
                                             )
```

NFC-B Collision resolution Listener device/card (PICC) as defined in Activity 1.1 9.3.5

This function is used to perform collision resolution for detection in case of multiple NFC Forum Devices with Technology B detected. Target with valid SENSB_RES will be stored in nfcbDevList and devCnt incremented.

Parameters

- [in] **compMode** : compliance mode to be performed
- [in] **devLimit** : device limit value, and size nfcbDevList
- [out] **nfcbDevList** : NFC-B listener device info
- [out] **devCnt** : devices found counter

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or mode not set

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_NONE : No error

Examples

exampleRfalPoller.c.

◆ rfalNfcbPollerGetCheckPresenceStatus()

```
ReturnCode rfalNfcbPollerGetCheckPresenceStatus ( void )
```

This method get the presence check status

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_TIMEOUT : Timeout error, no listener device detected

RFAL_ERR_RF_COLLISION : Collision detected one or more device in the field

RFAL_ERR_PAR : Parity error detected, one or more device in the field

RFAL_ERR_CRC : CRC error detected, one or more device in the field

RFAL_ERR_FRAMING : Framing error detected, one or more device in the field

RFAL_ERR_PROTO : Protocol error detected, invalid SENSB_RES received

RFAL_ERR_NONE : No error, SENSB_RES received

◆ rfalNfcbPollerGetCollisionResolutionStatus()

ReturnCode rfalNfcbPollerGetCollisionResolutionStatus (void)

Returns the Collision Resolution status

Returns

RFAL_ERR_BUSY : Operation is ongoing

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_PAR : Parity error detected

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_NONE : No error, activation successful

◆ rfalNfcbPollerGetSlotMarkerStatus()

ReturnCode rfalNfcbPollerGetSlotMarkerStatus (void)

This method gets the status of the NFC-B Slot marker

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_PAR : Parity error detected

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_NONE : No error, SEL_RES received

◆ rfalNfcbPollerGetTechnologyDetectionStatus()

ReturnCode rfalNfcbPollerGetTechnologyDetectionStatus (void)

This method gets the NFC-B Technology Detection status

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_NONE : No error, one or more device in the field

◆ rfalNfcbPollerInitialize()

ReturnCode rfalNfcbPollerInitialize (void)

This methods configures RFAL RF layer to perform as a NFC-B Poller/RW (ISO14443B PCD) including all default timings

It sets NFC-B parameters (AFI, PARAM) to default values

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or mode not set

RFAL_ERR_NONE : No error

Examples

exampleRfalPoller.c.

◆ rfalNfcbPollerInitializeWithParams()

```
ReturnCode rfalNfcbPollerInitializeWithParams ( uint8_t  AFI,
                                                uint8_t  PARAM
                                              )
```

This methods configures RFAL RF layer to perform as a NFCA Poller/RW (ISO14443A PCD) including all default timings

Additionally configures NFC-B specific parameters to be used on the following communications

Parameters

[in] **AFI** : Application Family Identifier to be used

[in] **PARAM** : PARAM to be used, it announces whether Advanced Features or Extended SENSB_RES is supported

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or mode not set

RFAL_ERR_NONE : No error

◆ rfalNfcbPollerSleep()

```
ReturnCode rfalNfcbPollerSleep ( const uint8_t *  nfcid0 )
```

This function is used to send the SLPB_REQ (HLTB) command to put the PICC with the given NFCID0 to state HALT so that they do not reply to further SENSB_REQ commands (only to ALLB_REQ)

Parameters

[in] **nfcid0** : NFCID of the device to be put to Sleep

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_NONE : No error

◆ rfalNfcbPollerSlotMarker()

```
ReturnCode rfalNfcbPollerSlotMarker ( uint8_t                slotCode,
                                      rfalNfcbSensbRes *     sensbRes,
                                      uint8_t *              sensbResLen
                                    )
```

This method sends a NFC-B Slot marker frame

Parameters

- [in] **slotCode** : Slot Code [1-15]
- [out] **sensbRes** : If received, the SENSB_RES
- [out] **sensbResLen** : If received, the SENSB_RES length

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_PAR : Parity error detected

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_NONE : No error, SEL_RES received

◆ rfalNfcbPollerSlottedCollisionResolution()

```
ReturnCode rfalNfcbPollerSlottedCollisionResolution ( rfalComplianceMode      compMode,
                                                      uint8_t                 devLimit,
                                                      rfalNfcbSlots           initSlots,
                                                      rfalNfcbSlots           endSlots,
                                                      rfalNfcbListenDevice *  nfcbDevList,
                                                      uint8_t *               devCnt,
                                                      bool *                  colPending
                                                    )
```

NFC-B Collision resolution Listener device/card (PICC). The sequence can be configured to be according to NFC Forum Activity 1.1 9.3.5, ISO10373 or EMVCo

This function is used to perform collision resolution for detection in case of multiple NFC Forum Devices with Technology B are detected. Target with valid SENSB_RES will be stored in nfcbDevList and devCnt incremented.

This method provides the means to perform a collision resolution loop with specific initial and end number of slots. This allows to user to start the loop already with greater number of slots, and or limit the end number of slots. At the end a flag indicating whether there were collisions pending is returned.

If RFAL_COMPLIANCE_MODE_ISO is used *initSlots* must be set to RFAL_NFCB_SLOT_NUM_1

Parameters

[in]	**compMode**	: compliance mode to be performed
[in]	**devLimit**	: device limit value, and size nfcbDevList
[in]	**initSlots**	: number of slots to open initially
[in]	**endSlots**	: number of slots when to stop collision resolution
[out]	**nfcbDevList**	: NFC-B listener device info
[out]	**devCnt**	: devices found counter
[out]	**colPending**	: flag indicating whether collision are still pending

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or mode not set

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_NONE : No error

◆ rfalNfcbPollerStartCheckPresence()

```
ReturnCode rfalNfcbPollerStartCheckPresence ( rfalNfcbSensCmd      cmd,
                                              rfalNfcbSlots        slots,
                                              rfalNfcbSensbRes *   sensbRes,
                                              uint8_t *            sensbResLen
                                            )
```

This method starts check for a NFC-B Listen device (PICC) presence on the field by sending an ALLB_REQ (WUPB) or SENSB_REQ (REQB)

Parameters

[in] **cmd** : Indicate if to send an ALL_REQ or a SENS_REQ

[in] **slots** : The number of slots to be announced

[out] **sensbRes** : If received, the SENSB_RES

[out] **sensbResLen** : If received, the SENSB_RES length

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_NONE : No error, SENSB_RES received

◆ rfalNfcbPollerStartCollisionResolution()

```
ReturnCode rfalNfcbPollerStartCollisionResolution ( rfalComplianceMode      compMode,
                                                    uint8_t                 devLimit,
                                                    rfalNfcbListenDevice *  nfcbDevList,
                                                    uint8_t *               devCnt
                                                  )
```

It starts the NFC-B Collision resolution Listener device/card (PICC) as defined in Activity 1.1 9.3.5

This function is used to trigger the collision resolution for detection in case of multiple NFC Forum Devices with Technology B detected. Target with valid SENSB_RES will be stored in nfcbDevList and devCnt incremented.

Parameters

[in] **compMode** : compliance mode to be performed

[in] **devLimit** : device limit value, and size nfcbDevList

[out] **nfcbDevList** : NFC-B listener device info

[out] **devCnt** : devices found counter

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or mode not set

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_NONE : No error

◆ rfalNfcbPollerStartSlotMarker()

```
ReturnCode rfalNfcbPollerStartSlotMarker ( uint8_t                slotCode,
                                           rfalNfcbSensbRes *     sensbRes,
                                           uint8_t *              sensbResLen
                                         )
```

This method starts a NFC-B Slot marker

Parameters

[in]	**slotCode**	: Slot Code [1-15]
[out]	**sensbRes**	: If received, the SENSB_RES
[out]	**sensbResLen**	: If received, the SENSB_RES length

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_NONE : No error, SEL_RES received

◆ rfalNfcbPollerStartSlottedCollisionResolution()

```
ReturnCode rfalNfcbPollerStartSlottedCollisionResolution ( rfalComplianceMode      compMode,
                                                           uint8_t                 devLimit,
                                                           rfalNfcbSlots           initSlots,
                                                           rfalNfcbSlots           endSlots,
                                                           rfalNfcbListenDevice *  nfcbDevList,
                                                           uint8_t *               devCnt,
                                                           bool *                  colPending
                                                         )
```

Starts NFC-B Collision resolution Listener device/card (PICC). The sequence can be configured to be according to NFC Forum Activity 1.1 9.3.5, ISO10373 or EMVCo

This function is used to trigger the collision resolution for detection in case of multiple NFC Forum Devices with Technology B are detected. Target with valid SENSB_RES will be stored in nfcbDevList and devCnt incremented.

This method provides the means to perform a collision resolution loop with specific initial and end number of slots. This allows to user to start the loop already with greater number of slots, and or limit the end number of slots. At the end a flag indicating whether there were collisions pending is returned.

If RFAL_COMPLIANCE_MODE_ISO is used *initSlots* must be set to RFAL_NFCB_SLOT_NUM_1

Parameters

- [in] **compMode** : compliance mode to be performed
- [in] **devLimit** : device limit value, and size nfcbDevList
- [in] **initSlots** : number of slots to open initially
- [in] **endSlots** : number of slots when to stop collision resolution
- [out] **nfcbDevList** : NFC-B listener device info
- [out] **devCnt** : devices found counter
- [out] **colPending** : flag indicating whether collision are still pending

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or mode not set

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_NONE : No error

◆ rfalNfcbPollerStartTechnologyDetection()

```
ReturnCode rfalNfcbPollerStartTechnologyDetection ( rfalComplianceMode  compMode,
                                                    rfalNfcbSensbRes *  sensbRes,
                                                    uint8_t *           sensbResLen
                                                  )
```

This method starts the NFC-B Technology Detection as defined in the spec given in the compliance mode

Parameters

- [in] **compMode** : compliance mode to be performed
- [out] **sensbRes** : location to store the SENSB_RES, if received
- [out] **sensbResLen** : length of the SENSB_RES, if received

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_NONE : No error, one or more device in the field

◆ rfalNfcbPollerTechnologyDetection()

```
ReturnCode rfalNfcbPollerTechnologyDetection ( rfalComplianceMode  compMode,
                                               rfalNfcbSensbRes *  sensbRes,
                                               uint8_t *           sensbResLen
                                             )
```

This method performs NFC-B Technology Detection as defined in the spec given in the compliance mode

Parameters

[in] **compMode** : compliance mode to be performed

[out] **sensbRes** : location to store the SENSB_RES, if received

[out] **sensbResLen** : length of the SENSB_RES, if received

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_NONE : No error, one or more device in the field

Examples

exampleRfalPoller.c.

◆ rfalNfcbTR2ToFDT()

```
uint32_t rfalNfcbTR2ToFDT ( uint8_t  tr2Code )
```

Converts the TR2 code as defined in Digital 1.1 Table 33 Minimum TR2 Coding to Frame Delay Time (FDT) in 1/Fc

Parameters

[in] **tr2Code** : TR2 code as defined in Digital 1.1 Table 33

Returns

FDT in 1/Fc

rfalNfcbSensbResAppData Struct Reference


```
#include <rfal_nfcb.h>
```

Data Fields

uint8_t	**AFI**
uint8_t	**CRC_B** [**RFAL_NFCB_CRC_LEN**]
uint8_t	**numApps**

Detailed Description

SENSB_RES (ATQB) Application Data Format Digital 1.1 Table 28

Field Documentation

◆ AFI

uint8_t rfalNfcbSensbResAppData::AFI

Application Family Identifier

◆ CRC_B

uint8_t rfalNfcbSensbResAppData::CRC_B[**RFAL_NFCB_CRC_LEN**]

CRC_B of AID

◆ numApps

uint8_t rfalNfcbSensbResAppData::numApps

Number of Applications

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

- **rfal_nfcb.h**

rfalNfcbSensbResProtocolInfo Struct Reference


```
#include <rfal_nfcb.h>
```

Data Fields

uint8_t	**BRC**
uint8_t	**FsciProType**
uint8_t	**FwiAdcFo**
uint8_t	**SFGI**

Detailed Description

SENSB_RES Protocol Info format Digital 1.1 Table 29

Field Documentation

◆ BRC

uint8_t rfalNfcbSensbResProtocolInfo::BRC

Bit Rate Capability

◆ FsciProType

uint8_t rfalNfcbSensbResProtocolInfo::FsciProType

Frame Size Card Integer [4b] | Protocol Type[4 bits]

◆ FwiAdcFo

uint8_t rfalNfcbSensbResProtocolInfo::FwiAdcFo

Frame Waiting Integer [4b] | Application Data Coding [2b] | Frame Options [2b]

◆ SFGI

uint8_t rfalNfcbSensbResProtocolInfo::SFGI

Optional: Start-Up Frame Guard Time Integer[4b] | RFU [4b]

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

- **rfal_nfcb.h**

---

rfalNfcbSensbRes Struct Reference

RFAL » RFAL-AL » NFC-B

```
#include <rfal_nfcb.h>
```

Collaboration diagram for rfalNfcbSensbRes:

[legend]

Data Fields

uint8_t	**cmd**
uint8_t	**nfcid0** [**RFAL_NFCB_NFCID0_LEN**]
rfalNfcbSensbResAppData	**appData**
rfalNfcbSensbResProtocolInfo	**protInfo**

Detailed Description

SENSB_RES format Digital 1.1 7.6.2

Examples

exampleRfalPoller.c.

Field Documentation

◆ appData

rfalNfcbSensbResAppData rfalNfcbSensbRes::appData

Application Data

◆ cmd

uint8_t rfalNfcbSensbRes::cmd

SENSB_RES: 50h

◆ nfcid0

uint8_t rfalNfcbSensbRes::nfcid0[**RFAL_NFCB_NFCID0_LEN**]

NFC Identifier (PUPI)

◆ protInfo

rfalNfcbSensbResProtocolInfo rfalNfcbSensbRes::protInfo

Protocol Information

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

- **rfal_nfcb.h**

rfalNfcbListenDevice Struct Reference


```
#include <rfal_nfcb.h>
```

Collaboration diagram for rfalNfcbListenDevice:

[legend]

Data Fields

uint8_t	**sensbResLen**
rfalNfcbSensbRes	**sensbRes**
bool	**isSleep**

Detailed Description

NFC-B listener device (PICC) struct

Examples

exampleRfalPoller.c.

Field Documentation

◆ isSleep

bool rfalNfcbListenDevice::isSleep

Device sleeping flag

◆ sensbRes

rfalNfcbSensbRes rfalNfcbListenDevice::sensbRes

SENSB_RES

◆ sensbResLen

uint8_t rfalNfcbListenDevice::sensbResLen

SENSB_RES length

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

- **rfal_nfcb.h**

NFC-DEP

RFAL NFC-DEP Module. More...

Collaboration diagram for NFC-DEP:

Data Structures

struct	**rfalNfcDepConfigs**
struct	**rfalNfcDepAtrReq**
struct	**rfalNfcDepAtrRes**
struct	**rfalNfcDepBufFormat**
struct	**rfalNfcDepPduBufFormat**
union	**rfalNfcDepActivation**
struct	**rfalNfcDepInfo**
struct	**rfalNfcDepDevice**
struct	**rfalNfcDepAtrParam**
struct	**rfalNfcDepListenActvParam**
struct	**rfalNfcDepTargetParam**
struct	**rfalNfcDepTxRxParam**

struct **rfalNfcDepPduTxRxParam**

Macros

#define **RFAL_FEATURE_NFC_DEP_BLOCK_MAX_LEN** 1U

#define **RFAL_FEATURE_NFC_DEP_PDU_MAX_LEN** 1U

#define **RFAL_NFCDEP_FRAME_SIZE_MAX_LEN** 254U

#define **RFAL_NFCDEP_DEPREQ_HEADER_LEN** 5U

#define **RFAL_NFCDEP_DEP_HEADER** (**RFAL_NFCDEP_LEN_LEN** + **RFAL_NFCDEP_CMDTYPE_LEN** + **RFAL_NFCDEP_CMD_LEN** + **RFAL_NFCDEP_DEP_PFB_LEN**)

#define **RFAL_NFCDEP_HEADER** (**RFAL_NFCDEP_CMDTYPE_LEN** + **RFAL_NFCDEP_CMD_LEN**)

#define **RFAL_NFCDEP_SB_LEN** 1U

#define **RFAL_NFCDEP_LEN_LEN** 1U

#define **RFAL_NFCDEP_CMDTYPE_LEN** 1U

#define **RFAL_NFCDEP_CMD_LEN** 1U

#define **RFAL_NFCDEP_DID_LEN** 1U

#define **RFAL_NFCDEP_DEP_PFB_LEN** 1U

#define **RFAL_NFCDEP_DSL_RLS_LEN_NO_DID** (**RFAL_NFCDEP_LEN_LEN** + **RFAL_NFCDEP_CMDTYPE_LEN** + **RFAL_NFCDEP_CMD_LEN**)

#define **RFAL_NFCDEP_DSL_RLS_LEN_DID** (**RFAL_NFCDEP_DSL_RLS_LEN_NO_DID** + **RFAL_NFCDEP_DID_LEN**)

#define **RFAL_NFCDEP_FS_VAL_MIN** 64U

#define **RFAL_NFCDEP_LR_VAL_MASK** 0x03U

#define **RFAL_NFCDEP_PP_LR_MASK** 0x30U

#define **RFAL_NFCDEP_PP_LR_SHIFT** 4U

```
#define RFAL_NFCDEP_DID_MAX 14U
#define RFAL_NFCDEP_DID_KEEP 0xFFU
#define RFAL_NFCDEP_DID_NO 0x00U
#define RFAL_NFCDEP_NAD_NO 0x00U
#define RFAL_NFCDEP_OPER_RTOX_REQ_DIS 0x01U
#define RFAL_NFCDEP_OPER_RTOX_REQ_EN 0x00U
#define RFAL_NFCDEP_OPER_ATN_DIS 0x00U
#define RFAL_NFCDEP_OPER_ATN_EN 0x02U
#define RFAL_NFCDEP_OPER_EMPTY_DEP_DIS 0x04U
#define RFAL_NFCDEP_OPER_EMPTY_DEP_EN 0x00U
#define RFAL_NFCDEP_OPER_FULL_MI_DIS 0x00U
#define RFAL_NFCDEP_OPER_FULL_MI_EN 0x08U
#define RFAL_NFCDEP_BRS_MAINTAIN 0xC0U
#define RFAL_NFCDEP_BRS_Dx_MASK 0x07U
#define RFAL_NFCDEP_BRS_DSI_POS 3U
#define RFAL_NFCDEP_WT_DELTA (16U - RFAL_NFCDEP_WT_DELTA_ADJUST)
#define RFAL_NFCDEP_WT_DELTA_ADJUST 4U
#define RFAL_NFCDEP_ATR_REQ_NFCID3_POS 2U
#define RFAL_NFCDEP_NFCID3_LEN 10U
#define RFAL_NFCDEP_LEN_MIN 3U
#define RFAL_NFCDEP_LEN_MAX 255U
```

```
#define RFAL_NFCDEP_ATRRES_HEADER_LEN 2U
#define RFAL_NFCDEP_ATRRES_MIN_LEN 17U
#define RFAL_NFCDEP_ATRRES_MAX_LEN 64U
#define RFAL_NFCDEP_ATRREQ_MIN_LEN 16U
#define RFAL_NFCDEP_ATRREQ_MAX_LEN RFAL_NFCDEP_ATRRES_MAX_LEN
#define RFAL_NFCDEP_GB_MAX_LEN (RFAL_NFCDEP_ATRREQ_MAX_LEN - RFAL_NFCDEP_ATRREQ_MIN_LEN)
#define RFAL_NFCDEP_WT_INI_DEFAULT RFAL_NFCDEP_WT_INI_MAX
#define RFAL_NFCDEP_WT_INI_MIN 0U
#define RFAL_NFCDEP_WT_INI_MAX 14U
#define RFAL_NFCDEP_RWT_INI_MAX rfalNfcDepWT2RWT( RFAL_NFCDEP_WT_INI_MAX )
#define RFAL_NFCDEP_WT_TRG_MAX_D10 8U
#define RFAL_NFCDEP_WT_TRG_MAX_D11 14U
#define RFAL_NFCDEP_WT_TRG_MAX_L13 10U
#define RFAL_NFCDEP_WT_TRG_MAX RFAL_NFCDEP_WT_TRG_MAX_D11
#define RFAL_NFCDEP_RWT_TRG_MAX rfalNfcDepWT2RWT( RFAL_NFCDEP_WT_TRG_MAX )
#define RFAL_NFCDEP_MAX_FWT ((uint32_t)1U<<20)
#define RFAL_NFCDEP_WT_MASK 0x0FU
#define RFAL_NFCDEP_BR_MASK_106 0x01U
#define RFAL_NFCDEP_BR_MASK_212 0x02U
#define RFAL_NFCDEP_BR_MASK_424 0x04U
#define rfalNfcDepWT2RWT(wt) ( (uint32_t)1U << (((uint32_t)(wt) & RFAL_NFCDEP_WT_MASK) + 12U) )
```

#define **rfalNfcDepDx2BRS**(br) ((((uint8_t)(br) & **RFAL_NFCDEP_BRS_Dx_MASK**) << **RFAL_NFCDEP_BRS_DSI_POS**) | ((uint8_t)(br) & **RFAL_NFCDEP_BRS_Dx_MASK**))

#define **rfalNfcDepBRS2DRI**(brs) (uint8_t)((uint8_t)(brs) & **RFAL_NFCDEP_BRS_Dx_MASK**)

#define **rfalNfcDepBRS2DSI**(brs) (uint8_t)(((uint8_t)(brs) >> **RFAL_NFCDEP_BRS_DSI_POS**) & **RFAL_NFCDEP_BRS_Dx_MASK**)

#define **rfalNfcDepPP2LR**(PPx) (((uint8_t)(PPx) & **RFAL_NFCDEP_PP_LR_MASK**) >> **RFAL_NFCDEP_PP_LR_SHIFT**)

#define **rfalNfcDepLR2PP**(LRx) (((uint8_t)(LRx) << **RFAL_NFCDEP_PP_LR_SHIFT**) & **RFAL_NFCDEP_PP_LR_MASK**)

#define **rfalNfcDepLR2FS**(LRx) (uint16_t)(**RFAL_MIN**((**RFAL_NFCDEP_FS_VAL_MIN** * ((uint16_t)(LRx) + 1U)), **RFAL_NFCDEP_FRAME_SIZE_MAX_LEN**))

#define **rfalNfcDepSetNFCID**(dst, src, len)

Typedefs

typedef bool(* **rfalNfcDepDeactCallback**) (void)

Enumerations

enum {
RFAL_NFCDEP_Bx_NO_HIGH_BR = 0x00 , **RFAL_NFCDEP_Bx_08_848** = 0x01 , **RFAL_NFCDEP_Bx_16_1695** = 0x02 , **RFAL_NFCDEP_Bx_32_3390** = 0x04 ,
RFAL_NFCDEP_Bx_64_6780 = 0x08
}

enum {
RFAL_NFCDEP_Dx_01_106 = RFAL_BR_106 , **RFAL_NFCDEP_Dx_02_212** = RFAL_BR_212 , **RFAL_NFCDEP_Dx_04_424** = RFAL_BR_424 , **RFAL_NFCDEP_Dx_08_848** = RFAL_BR_848 ,
RFAL_NFCDEP_Dx_16_1695 = RFAL_BR_1695 , **RFAL_NFCDEP_Dx_32_3390** = RFAL_BR_3390 , **RFAL_NFCDEP_Dx_64_6780** = RFAL_BR_6780
}

enum { **RFAL_NFCDEP_LR_64** = 0x00 , **RFAL_NFCDEP_LR_128** = 0x01 , **RFAL_NFCDEP_LR_192** = 0x02 , **RFAL_NFCDEP_LR_254** = 0x03 }

enum **rfalNfcDepCommMode** { **RFAL_NFCDEP_COMM_PASSIVE** , **RFAL_NFCDEP_COMM_ACTIVE** }

enum **rfalNfcDepRole** { **RFAL_NFCDEP_ROLE_INITIATOR** , **RFAL_NFCDEP_ROLE_TARGET** }

Functions

void **rfalNfcDepInitialize** (void)
NFCIP Initialize. More...

void **rfalNfcDepSetDeactivatingCallback** (**rfalNfcDepDeactCallback** pFunc)
Set deactivating callback. More...

uint32_t **rfalNfcDepCalculateRWT** (uint8_t wt)
Calculate Response Waiting Time. More...

ReturnCode **rfalNfcDepATR** (const **rfalNfcDepAtrParam** *param, **rfalNfcDepAtrRes** *atrRes, uint8_t *atrResLen)
NFC-DEP Initiator ATR (Attribute Request) More...

ReturnCode **rfalNfcDepPSL** (uint8_t BRS, uint8_t FSL)
NFC-DEP Initiator PSL (Parameter Selection) More...

ReturnCode **rfalNfcDepDSL** (void)
NFC-DEP Initiator DSL (Deselect) More...

ReturnCode **rfalNfcDepRLS** (void)
NFC-DEP Initiator RLS (Release) More...

ReturnCode **rfalNfcDepInitiatorHandleActivation** (**rfalNfcDepAtrParam** *param, **rfalBitRate** desiredBR, **rfalNfcDepDevice** *nfcDepDev)
NFC-DEP Initiator Handle Activation. More...

bool **rfalNfcDepIsAtrReq** (const uint8_t *buf, uint16_t bufLen, uint8_t *nfcid3)
Check if buffer contains valid ATR_REQ. More...

bool **rfalNfcDepTargetRcvdATR** (void)
Check is Target has received ATR. More...

ReturnCode **rfalNfcDepListenStartActivation** (const **rfalNfcDepTargetParam** *param, const uint8_t *atrReq, uint16_t atrReqLength, **rfalNfcDepListenActvParam** rxParam)

NFCDEP Start Listen Activation Handling. More...

ReturnCode	**rfalNfcDepListenGetActivationStatus** (void) Get the current NFC-DEP Activation Status. More...
ReturnCode	**rfalNfcDepStartTransceive** (const **rfalNfcDepTxRxParam** *param) Start Transceive. More...
ReturnCode	**rfalNfcDepGetTransceiveStatus** (void) Return the Transceive status. More...
ReturnCode	**rfalNfcDepStartPduTransceive** (**rfalNfcDepPduTxRxParam** param) Start PDU Transceive. More...
ReturnCode	**rfalNfcDepGetPduTransceiveStatus** (void) Return the PDU Transceive status. More...

Detailed Description

Macro Definition Documentation

◆ RFAL_FEATURE_NFC_DEP_BLOCK_MAX_LEN

```
#define RFAL_FEATURE_NFC_DEP_BLOCK_MAX_LEN   1U
```

NFC-DEP Block/Payload length, set to "none"

◆ RFAL_FEATURE_NFC_DEP_PDU_MAX_LEN

```
#define RFAL_FEATURE_NFC_DEP_PDU_MAX_LEN   1U
```

NFC-DEP PDU length, set to "none"

◆ RFAL_NFCDEP_ATR_REQ_NFCID3_POS

```
#define RFAL_NFCDEP_ATR_REQ_NFCID3_POS   2U
```

NFCID3 offset in ATR_REQ frame

◆ RFAL_NFCDEP_ATRREQ_MAX_LEN

```
#define RFAL_NFCDEP_ATRREQ_MAX_LEN   RFAL_NFCDEP_ATRRES_MAX_LEN
```

Maximum length for an ATR REQ Digital 1.0 14.6.1

◆ RFAL_NFCDEP_ATRREQ_MIN_LEN

```
#define RFAL_NFCDEP_ATRREQ_MIN_LEN   16U
```

Minimum length for an ATR REQ

◆ RFAL_NFCDEP_ATRRES_HEADER_LEN

```
#define RFAL_NFCDEP_ATRRES_HEADER_LEN   2U
```

ATR RES Header Len: CmdType: 0xD5 + Cod: 0x01

◆ RFAL_NFCDEP_ATRRES_MAX_LEN

```
#define RFAL_NFCDEP_ATRRES_MAX_LEN   64U
```

Maximum length for an ATR RES Digital 1.0 14.6.1

◆ RFAL_NFCDEP_ATRRES_MIN_LEN

```
#define RFAL_NFCDEP_ATRRES_MIN_LEN   17U
```

Minimum length for an ATR RES

◆ RFAL_NFCDEP_BR_MASK_106

```
#define RFAL_NFCDEP_BR_MASK_106   0x01U
```

Enable mask bit rate 106

◆ RFAL_NFCDEP_BR_MASK_212

```
#define RFAL_NFCDEP_BR_MASK_212   0x02U
```

Enable mask bit rate 242

◆ RFAL_NFCDEP_BR_MASK_424

```
#define RFAL_NFCDEP_BR_MASK_424   0x04U
```

Enable mask bit rate 424

◆ RFAL_NFCDEP_BRS_DSI_POS

```
#define RFAL_NFCDEP_BRS_DSI_POS   3U
```

Value signalling that BR is to be maintained (no PSL)

◆ RFAL_NFCDEP_BRS_Dx_MASK

```
#define RFAL_NFCDEP_BRS_Dx_MASK   0x07U
```

Value signalling that BR is to be maintained (no PSL)

◆ RFAL_NFCDEP_BRS_MAINTAIN

```
#define RFAL_NFCDEP_BRS_MAINTAIN   0xC0U
```

Value signalling that BR is to be maintained (no PSL)

◆ RFAL_NFCDEP_CMD_LEN

```
#define RFAL_NFCDEP_CMD_LEN   1U
```

Length of the cmd on NFCIP frame

◆ RFAL_NFCDEP_CMDTYPE_LEN

```
#define RFAL_NFCDEP_CMDTYPE_LEN   1U
```

Length of the cmd type (REQ | RES) on NFCIP frame

◆ RFAL_NFCDEP_DEP_HEADER

```
#define RFAL_NFCDEP_DEP_HEADER   ( RFAL_NFCDEP_LEN_LEN + RFAL_NFCDEP_CMDTYPE_LEN +
RFAL_NFCDEP_CMD_LEN + RFAL_NFCDEP_DEP_PFB_LEN )
```

Length NFCIP DEP REQ or RES header (incl LEN)

◆ RFAL_NFCDEP_DEP_PFB_LEN

```
#define RFAL_NFCDEP_DEP_PFB_LEN   1U
```

Length of the PFB field on NFCIP frame

◆ RFAL_NFCDEP_DEPREQ_HEADER_LEN

```
#define RFAL_NFCDEP_DEPREQ_HEADER_LEN   5U
```

DEP_REQ header length: CMD_TYPE + CMD_CMD + PBF + DID + NAD

◆ RFAL_NFCDEP_DID_KEEP

```
#define RFAL_NFCDEP_DID_KEEP   0xFFU
```

Keep DID value already configured

Examples

exampleRfalPoller.c.

◆ RFAL_NFCDEP_DID_LEN

```
#define RFAL_NFCDEP_DID_LEN   1U
```

Length of did on NFCIP frame

◆ RFAL_NFCDEP_DID_MAX

```
#define RFAL_NFCDEP_DID_MAX   14U
```

Max DID value Digital 14.6.2.3

◆ RFAL_NFCDEP_DID_NO

```
#define RFAL_NFCDEP_DID_NO   0x00U
```

No DID shall be used

Examples

exampleRfalPoller.c.

◆ RFAL_NFCDEP_DSL_RLS_LEN_DID

#define RFAL_NFCDEP_DSL_RLS_LEN_DID (**RFAL_NFCDEP_DSL_RLS_LEN_NO_DID** + **RFAL_NFCDEP_DID_LEN**)

Length of DSL_REQ and RLS_REQ with DID

◆ RFAL_NFCDEP_DSL_RLS_LEN_NO_DID

#define RFAL_NFCDEP_DSL_RLS_LEN_NO_DID (**RFAL_NFCDEP_LEN_LEN** + **RFAL_NFCDEP_CMDTYPE_LEN** + **RFAL_NFCDEP_CMD_LEN**)

Length of DSL_REQ and RLS_REQ with no DID

◆ RFAL_NFCDEP_FRAME_SIZE_MAX_LEN

```
#define RFAL_NFCDEP_FRAME_SIZE_MAX_LEN   254U
```

Maximum Frame Size Digital 2.0 Table 90

◆ RFAL_NFCDEP_FS_VAL_MIN

```
#define RFAL_NFCDEP_FS_VAL_MIN   64U
```

Minimum LR value

◆ RFAL_NFCDEP_GB_MAX_LEN

```
#define RFAL_NFCDEP_GB_MAX_LEN   (RFAL_NFCDEP_ATRREQ_MAX_LEN - RFAL_NFCDEP_ATRREQ_MIN_LEN)
```

Maximum length the General Bytes on ATR Digital 1.1 16.6.3

◆ RFAL_NFCDEP_HEADER

```
#define RFAL_NFCDEP_HEADER   ( RFAL_NFCDEP_CMDTYPE_LEN + RFAL_NFCDEP_CMD_LEN )
```

NFCIP header length

◆ RFAL_NFCDEP_LEN_LEN

```
#define RFAL_NFCDEP_LEN_LEN   1U
```

LEN length on NFCIP frame

◆ RFAL_NFCDEP_LEN_MAX

```
#define RFAL_NFCDEP_LEN_MAX   255U
```

Maximum length byte LEN value

◆ RFAL_NFCDEP_LEN_MIN

```
#define RFAL_NFCDEP_LEN_MIN   3U
```

Minimum length byte LEN value

◆ RFAL_NFCDEP_LR_VAL_MASK

```
#define RFAL_NFCDEP_LR_VAL_MASK   0x03U
```

Bit mask for a LR value

◆ RFAL_NFCDEP_MAX_FWT

```
#define RFAL_NFCDEP_MAX_FWT   ((uint32_t)1U<<20)
```

Maximum Frame Waiting Time = ((256 * 16/fc)*2^FWImax) = ((256*16/fc)*2^14) = (1048576 / 64)/fc = (100000h*64)/fc

◆ RFAL_NFCDEP_NAD_NO

```
#define RFAL_NFCDEP_NAD_NO   0x00U
```

No NAD shall be used

Examples

exampleRfalPoller.c.

◆ RFAL_NFCDEP_NFCID3_LEN

```
#define RFAL_NFCDEP_NFCID3_LEN   10U
```

NFCID3 Length

Examples

exampleRfalPoller.c.

◆ RFAL_NFCDEP_OPER_ATN_DIS

```
#define RFAL_NFCDEP_OPER_ATN_DIS   0x00U
```

Operation config: ATN disable

◆ RFAL_NFCDEP_OPER_ATN_EN

```
#define RFAL_NFCDEP_OPER_ATN_EN   0x02U
```

Operation config: ATN enable

Examples

exampleRfalPoller.c.

◆ RFAL_NFCDEP_OPER_EMPTY_DEP_DIS

```
#define RFAL_NFCDEP_OPER_EMPTY_DEP_DIS   0x04U
```

Operation config: empty DEPs disable

Examples

exampleRfalPoller.c.

◆ RFAL_NFCDEP_OPER_EMPTY_DEP_EN

```
#define RFAL_NFCDEP_OPER_EMPTY_DEP_EN   0x00U
```

Operation config: empty DEPs enable

◆ RFAL_NFCDEP_OPER_FULL_MI_DIS

```
#define RFAL_NFCDEP_OPER_FULL_MI_DIS   0x00U
```

Operation config: full chaining DEPs disable

◆ RFAL_NFCDEP_OPER_FULL_MI_EN

```
#define RFAL_NFCDEP_OPER_FULL_MI_EN   0x08U
```

Operation config: full chaining DEPs enable

Examples

exampleRfalPoller.c.

◆ RFAL_NFCDEP_OPER_RTOX_REQ_DIS

```
#define RFAL_NFCDEP_OPER_RTOX_REQ_DIS   0x01U
```

Operation config: RTOX REQ disable

◆ RFAL_NFCDEP_OPER_RTOX_REQ_EN

```
#define RFAL_NFCDEP_OPER_RTOX_REQ_EN   0x00U
```

Operation config: RTOX REQ enable

Examples

exampleRfalPoller.c.

◆ RFAL_NFCDEP_PP_LR_MASK

```
#define RFAL_NFCDEP_PP_LR_MASK   0x30U
```

Bit mask for LR value in PP byte on a ATR REQ/RES

◆ RFAL_NFCDEP_PP_LR_SHIFT

```
#define RFAL_NFCDEP_PP_LR_SHIFT   4U
```

Position of LR value in PP byte on a ATR REQ/RES

◆ RFAL_NFCDEP_RWT_INI_MAX

#define RFAL_NFCDEP_RWT_INI_MAX **rfalNfcDepWT2RWT**(**RFAL_NFCDEP_WT_INI_MAX**)

RWT Initiator maximum value

◆ RFAL_NFCDEP_RWT_TRG_MAX

#define RFAL_NFCDEP_RWT_TRG_MAX **rfalNfcDepWT2RWT**(**RFAL_NFCDEP_WT_TRG_MAX**)

RWT Initiator maximum value

◆ RFAL_NFCDEP_SB_LEN

```
#define RFAL_NFCDEP_SB_LEN   1U
```

SB length on NFCIP fram for NFC-A

◆ RFAL_NFCDEP_WT_DELTA

```
#define RFAL_NFCDEP_WT_DELTA   (16U - RFAL_NFCDEP_WT_DELTA_ADJUST)
```

NFC-DEP dWRT (adjusted) Digital 2.0 B.10

◆ RFAL_NFCDEP_WT_DELTA_ADJUST

```
#define RFAL_NFCDEP_WT_DELTA_ADJUST   4U
```

dWRT value adjustment

◆ RFAL_NFCDEP_WT_INI_DEFAULT

```
#define RFAL_NFCDEP_WT_INI_DEFAULT   RFAL_NFCDEP_WT_INI_MAX
```

WT Initiator default value Digital 1.0 14.6.3.8

◆ RFAL_NFCDEP_WT_INI_MAX

```
#define RFAL_NFCDEP_WT_INI_MAX   14U
```

WT Initiator maximum value Digital 1.0 14.6.3.8 A.10

◆ RFAL_NFCDEP_WT_INI_MIN

```
#define RFAL_NFCDEP_WT_INI_MIN   0U
```

WT Initiator minimum value Digital 1.0 14.6.3.8

◆ RFAL_NFCDEP_WT_MASK

#define RFAL_NFCDEP_WT_MASK 0x0FU

Bit mask for the Wait Time value

◆ RFAL_NFCDEP_WT_TRG_MAX

#define RFAL_NFCDEP_WT_TRG_MAX **RFAL_NFCDEP_WT_TRG_MAX_D11**

WT target max Digital x.x | LLCP x.x

◆ RFAL_NFCDEP_WT_TRG_MAX_D10

#define RFAL_NFCDEP_WT_TRG_MAX_D10 8U

WT target max Digital 1.0 14.6.3.8 A.10

◆ RFAL_NFCDEP_WT_TRG_MAX_D11

#define RFAL_NFCDEP_WT_TRG_MAX_D11 14U

WT target max Digital 1.1 16.6.3.9 A.9

◆ RFAL_NFCDEP_WT_TRG_MAX_L13

#define RFAL_NFCDEP_WT_TRG_MAX_L13 10U

WT target max [LLCP] 1.3 6.2.1

◆ rfalNfcDepBRS2DRI

#define rfalNfcDepBRS2DRI (brs) (uint8_t)((uint8_t)(brs) & **RFAL_NFCDEP_BRS_Dx_MASK**)

Returns the DRI value from the given BRS byte

◆ rfalNfcDepBRS2DSI

```
#define rfalNfcDepBRS2DSI ( brs ) (uint8_t)( ((uint8_t)(brs) >> RFAL_NFCDEP_BRS_DSI_POS) & RFAL_NFCDEP_BRS_Dx_MASK )
```

Returns the DSI value from the given BRS byte

◆ rfalNfcDepDx2BRS

```
#define rfalNfcDepDx2BRS ( br ) (((uint8_t)(br) & RFAL_NFCDEP_BRS_Dx_MASK) << RFAL_NFCDEP_BRS_DSI_POS) | ((uint8_t)(br) & RFAL_NFCDEP_BRS_Dx_MASK) )
```

Returns the BRS value from the given bit rate

◆ rfalNfcDepLR2FS

```
#define rfalNfcDepLR2FS ( LRx ) (uint16_t)(RFAL_MIN( (RFAL_NFCDEP_FS_VAL_MIN * ((uint16_t)(LRx) + 1U) ), RFAL_NFCDEP_FRAME_SIZE_MAX_LEN ))
```

Returns the Frame size value from the given LRx value

Examples

exampleRfalPoller.c.

◆ rfalNfcDepLR2PP

```
#define rfalNfcDepLR2PP ( LRx ) ((uint8_t)(LRx) << RFAL_NFCDEP_PP_LR_SHIFT) & RFAL_NFCDEP_PP_LR_MASK)
```

Returns the PP byte with the given LRx value

◆ rfalNfcDepPP2LR

```
#define rfalNfcDepPP2LR ( PPx ) ((uint8_t)(PPx) & RFAL_NFCDEP_PP_LR_MASK ) >> RFAL_NFCDEP_PP_LR_SHIFT)
```

Returns the LR value from the given PPx byte

Examples

exampleRfalPoller.c.

◆ rfalNfcDepSetNFCID

#define rfalNfcDepSetNFCID (dst, src, len)

Value:

```
RFAL_MEMSET( (dst), 0x00, RFAL_NFCDEP_NFCID3_LEN ); \
if( (len) > 0U ) {RFAL_MEMCPY( (dst), (src), (len) );}
```

Despite DIGITAL 1.0 14.6.2.1 stating that the last two bytes may filled with any value, some devices (Samsung Google Nexus) only accept when these are 0

◆ rfalNfcDepWT2RWT

#define rfalNfcDepWT2RWT (wt) ((uint32_t)1U << (((uint32_t)(wt) & **RFAL_NFCDEP_WT_MASK**) + 12U))

Converts WT value to RWT (1/fc)

Typedef Documentation

◆ rfalNfcDepDeactCallback

typedef bool(* rfalNfcDepDeactCallback) (void)

NFC-DEP callback to check if upper layer has deactivation pending

Enumeration Type Documentation

◆ anonymous enum

anonymous enum

Enumeration of NFC-DEP bit rate Dividor in PSL Digital 1.0 Table 100

Enumerator	
RFAL_NFCDEP_Dx_01_106	Divisor D = 1 : bit rate = 106
RFAL_NFCDEP_Dx_02_212	Divisor D = 2 : bit rate = 212
RFAL_NFCDEP_Dx_04_424	Divisor D = 4 : bit rate = 424
RFAL_NFCDEP_Dx_08_848	Divisor D = 8 : bit rate = 848
RFAL_NFCDEP_Dx_16_1695	Divisor D = 16 : bit rate = 1695
RFAL_NFCDEP_Dx_32_3390	Divisor D = 32 : bit rate = 3390
RFAL_NFCDEP_Dx_64_6780	Divisor D = 64 : bit rate = 6780

◆ anonymous enum

anonymous enum

Enumeration of NFC-DEP Length Reduction (LR) Digital 1.0 Table 91

Enumerator	
RFAL_NFCDEP_LR_64	Maximum payload size is 64 bytes
RFAL_NFCDEP_LR_128	Maximum payload size is 128 bytes
RFAL_NFCDEP_LR_192	Maximum payload size is 192 bytes
RFAL_NFCDEP_LR_254	Maximum payload size is 254 bytes

◆ anonymous enum

anonymous enum

Enumeration of NFC-DEP bit rate in ATR Digital 1.0 Table 93 and 94

Enumerator	
RFAL_NFCDEP_Bx_NO_HIGH_BR	Peer supports no high bit rates
RFAL_NFCDEP_Bx_08_848	Peer also supports 848

RFAL_NFCDEP_Bx_16_1695	Peer also supports 1695
RFAL_NFCDEP_Bx_32_3390	Peer also supports 3390
RFAL_NFCDEP_Bx_64_6780	Peer also supports 6780

◆ rfalNfcDepCommMode

enum **rfalNfcDepCommMode**

Enumeration of the nfcip communication modes

Enumerator	
RFAL_NFCDEP_COMM_PASSIVE	Passive communication mode
RFAL_NFCDEP_COMM_ACTIVE	Active communication mode

◆ rfalNfcDepRole

enum **rfalNfcDepRole**

Enumeration of the nfcip roles

Enumerator	
RFAL_NFCDEP_ROLE_INITIATOR	Perform as Initiator
RFAL_NFCDEP_ROLE_TARGET	Perform as Target

Function Documentation

◆ rfalNfcDepATR()

```
ReturnCode rfalNfcDepATR ( const rfalNfcDepAtrParam *  param,
                           rfalNfcDepAtrRes *          atrRes,
                           uint8_t *                   atrResLen
                         )
```

This method configures the NFC-DEP layer with given parameters and then sends an ATR to the Target with and checks for a valid response response

Parameters

[in] **param** : parameters to initialize and compose the ATR

[out] **atrRes** : location to store the ATR_RES

[out] **atrResLen** : length of the ATR_RES received

Returns

RFAL_ERR_NONE : No error

RFAL_ERR_TIMEOUT : Timeout occurred

RFAL_ERR_PROTO : Protocol error occurred

◆ rfalNfcDepCalculateRWT()

uint32_t rfalNfcDepCalculateRWT (uint8_t wt)

Calculates the Response Waiting Time (RWT) from the given Waiting Time (WT)

Parameters

[in] **wt** : the WT value to calculate RWT

Returns

RWT value in 1/fc

◆ rfalNfcDepDSL()

ReturnCode rfalNfcDepDSL (void)

This method checks if the NFCIP module is configured as initiator and if so sends a DSL REQ, waits the target's response and checks it

In case of performing as target no action is taken

Returns

RFAL_ERR_NONE : No error

RFAL_ERR_TIMEOUT : Timeout occurred

RFAL_ERR_MAX_RERUNS : Timeout occurred

RFAL_ERR_PROTO : Protocol error occurred

◆ rfalNfcDepGetPduTransceiveStatus()

ReturnCode rfalNfcDepGetPduTransceiveStatus (void)

Returns the status of the NFC-DEP PDU Transceive

Returns

RFAL_ERR_NONE : Transceive has been completed successfully

RFAL_ERR_BUSY : Transceive is ongoing

RFAL_ERR_PROTO : Protocol error occurred

RFAL_ERR_TIMEOUT : Timeout error occurred

RFAL_ERR_SLEEP_REQ : Deselect has been received and responded

RFAL_ERR_NOMEM : The received I-PDU does not fit into the receive buffer

RFAL_ERR_LINK_LOSS : Communication is lost because Reader/Writer has turned off its field

◆ rfalNfcDepGetTransceiveStatus()

ReturnCode rfalNfcDepGetTransceiveStatus (void)

Returns the status of the NFC-DEP Transceive

Warning

When the other device is performing chaining once a chained block is received the error RFAL_ERR_AGAIN is sent. At this point caller must handle the received data immediately. When RFAL_ERR_AGAIN is returned an ACK has already been sent to the other device and the next block might be incoming. If **rfalWorker()** is called frequently it will place the next block on the given buffer

Returns

RFAL_ERR_NONE : Transceive has been completed successfully

RFAL_ERR_BUSY : Transceive is ongoing

RFAL_ERR_PROTO : Protocol error occurred

RFAL_ERR_TIMEOUT : Timeout error occurred

RFAL_ERR_SLEEP_REQ : Deselect has been received and responded

RFAL_ERR_NOMEM : The received I-PDU does not fit into the receive buffer

RFAL_ERR_LINK_LOSS : Communication is lost because Reader/Writer has turned off its field

RFAL_ERR_AGAIN : received one chaining block, continue to call this method to retrieve the remaining blocks

Examples

exampleRfalPoller.c.

◆ rfalNfcDepInitialize()

void rfalNfcDepInitialize (void)

This method resets all NFC-DEP inner states, counters and context and sets default values

◆ rfalNfcDepInitiatorHandleActivation()

```
ReturnCode rfalNfcDepInitiatorHandleActivation ( rfalNfcDepAtrParam *  param,
                                                 rfalBitRate           desiredBR,
                                                 rfalNfcDepDevice *    nfcDepDev
                                               )
```

This performs a Activation into NFC-DEP layer with the given parameters. It sends ATR_REQ and if the higher bit rates are supported by both devices it additionally sends PSL Once Activated all details of the device are provided on nfcDepDev

Parameters

- [in] **param** : required parameters to initialize and send ATR_REQ
- [in] **desiredBR** : Desired bit rate supported by the Poller
- [out] **nfcDepDev** : NFC-DEP information of the activated Listen device

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_PAR : Parity error detected

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_NONE : No error, activation successful

Examples

exampleRfalPoller.c.

◆ rfalNfcDepIsAtrReq()

```
bool rfalNfcDepIsAtrReq ( const uint8_t *  buf,
                          uint16_t         bufLen,
                          uint8_t *        nfcid3
                        )
```

This method checks if the given ATR_REQ is valid

Parameters

- [in] **buf** : buffer holding Initiator's received request
- [in] **bufLen** : size of the msg contained on the buf in Bytes
- [out] **nfcid3** : pointer to where the NFCID3 may be outputed, nfcid3 has NFCF_SENSF_NFCID3_LEN as length Pass NULL if output parameter not desired

Returns

true : Valid ATR_REQ received, the ATR_RES has been computed in txBuf

false : Invalid protocol request

◆ rfalNfcDepListenGetActivationStatus()

```
ReturnCode rfalNfcDepListenGetActivationStatus ( void )
```

Returns

RFAL_ERR_NONE : Activation has completed successfully

RFAL_ERR_BUSY : Activation is ongoing

RFAL_ERR_LINK_LOSS : Remote Field was turned off

◆ rfalNfcDepListenStartActivation()

```
ReturnCode rfalNfcDepListenStartActivation ( const rfalNfcDepTargetParam *  param,
                                             const uint8_t *                 atrReq,
                                             uint16_t                        atrReqLength,
                                             rfalNfcDepListenActvParam       rxParam
                                           )
```

Start Activation Handling and setup to receive first frame which may contain complete or partial DEP-REQ after activation is completed

Pass in ATR_REQ for NFC-DEP to handle ATR_RES. The Activation Handling handles ATR_RES and PSL_RES if a PSL_REQ is received

Activation is completed if PSL_RES is sent or if first I-PDU is received

rfalNfcDepListenGetActivationStatus() provide status of the ongoing activation

Warning

nfcDepGetTransceiveStatus() shall be called right after activation is completed (i.e. **rfalNfcDepListenGetActivationStatus()** return RFAL_ERR_NONE) to check for first received frame.

Parameters

[in]	**param**	: Target parameters to be used
[in]	**atrReq**	: reference to buffer containing ATR_REQ
[in]	**atrReqLength**	Length of ATR_REQ
[out]	**rxParam**	: references to buffer, length and chaining indication for first complete LLCP to be received

Returns

RFAL_ERR_NONE : ATR_REQ is valid and activation ongoing

RFAL_ERR_PARAM : ATR_REQ or other params are invalid

RFAL_ERR_LINK_LOSS : Remote Field is turned off

◆ rfalNfcDepPSL()

```
ReturnCode rfalNfcDepPSL ( uint8_t  BRS,
                           uint8_t  FSL
                         )
```

This method sends a PSL to the Target with the given parameters and checks for a valid response response

The parameters must be coded according to Digital 1.1 16.7.1

Parameters

[in] **BRS** : the selected Bit Rates for Initiator and Target

[in] **FSL** : the maximum length of Commands and Responses

Returns

RFAL_ERR_NONE : No error

RFAL_ERR_TIMEOUT : Timeout occurred

RFAL_ERR_PROTO : Protocol error occurred

◆ rfalNfcDepRLS()

ReturnCode rfalNfcDepRLS (void)

This method checks if the NFCIP module is configured as initiator and if so sends a RLS REQ, waits target's response and checks it

In case of performing as target no action is taken

Returns

RFAL_ERR_NONE : No error

RFAL_ERR_TIMEOUT : Timeout occurred

RFAL_ERR_MAX_RERUNS : Timeout occurred

RFAL_ERR_PROTO : Protocol error occurred

Examples

exampleRfalPoller.c.

◆ rfalNfcDepSetDeactivatingCallback()

void rfalNfcDepSetDeactivatingCallback (**rfalNfcDepDeactCallback** pFunc)

Sets the deactivating callback so that nfcip layer can check if upper layer has a deactivation pending, and not perform error recovery upon specific errors

Parameters

`[in]` **pFunc** : method pointer to deactivation flag check

◆ rfalNfcDepStartPduTransceive()

ReturnCode rfalNfcDepStartPduTransceive (**rfalNfcDepPduTxRxParam** param)

This method triggers a NFC-DEP Transceive containing a complete PDU It transmits the given message and handles all protocol retransmitions, error handling and control messages

The txBuf contains a complete PDU to be transmitted The Prologue field will be manipulated by the Transceive

Warning

the txBuf will be modified during the transmission

the maximum RF frame which can be received is limited by param.tmpBuf

Parameters

[in] **param** reference parameters to be used for the Transceive

Returns

RFAL_ERR_PARAM : Bad request

RFAL_ERR_WRONG_STATE : The module is not in a proper state

RFAL_ERR_NONE : The Transceive request has been started

◆ rfalNfcDepStartTransceive()

ReturnCode rfalNfcDepStartTransceive (const **rfalNfcDepTxRxParam** * param)

Transceives a complete or partial DEP block

The txBuf contains complete or partial of DEP to be transmitted. The Prologue field of the I-PDU is handled internally

If the buffer contains partial LLCP and is not the last block, then isTxChaining must be set to true

Parameters

[in] **param** reference parameters to be used for the Transceive

Returns

RFAL_ERR_PARAM : Bad request

RFAL_ERR_WRONG_STATE : The module is not in a proper state

RFAL_ERR_NONE : The Transceive request has been started

Examples

exampleRfalPoller.c.

◆ rfalNfcDepTargetRcvdATR()

bool rfalNfcDepTargetRcvdATR (void)

This method checks if the NFCIP module is configured as target and if a ATR REQ has been received (whether is in activation or in data exchange)

Returns

true : a ATR has already been received

false : no ATR has been received

rfalNfcDepConfigs Struct Reference


```
#include <rfal_nfcDep.h>
```

Data Fields

rfalNfcDepRole	role
rfalNfcDepCommMode	commMode
uint8_t	oper
uint8_t	did

uint8_t	**nad**
uint8_t	**bs**
uint8_t	**br**
uint8_t	**nfcid** [**RFAL_NFCDEP_NFCID3_LEN**]
uint8_t	**nfcidLen**
uint8_t	**gb** [**RFAL_NFCDEP_GB_MAX_LEN**]
uint8_t	**gbLen**
uint8_t	**lr**
uint8_t	**to**
uint32_t	**fwt**
uint32_t	**dFwt**

Detailed Description

Struct that holds all NFCIP configs

Field Documentation

◆ br

uint8_t rfalNfcDepConfigs::br

Bit rate in Receiving Direction

◆ bs

uint8_t rfalNfcDepConfigs::bs

Bit rate in Sending Direction

◆ commMode

rfalNfcDepCommMode rfalNfcDepConfigs::commMode

Current NFCIP communication mode

◆ dFwt

uint32_t rfalNfcDepConfigs::dFwt

Delta Frame Waiting Time (dFWT) to be used

◆ did

uint8_t rfalNfcDepConfigs::did

Current Device ID (DID)

◆ fwt

uint32_t rfalNfcDepConfigs::fwt

Frame Waiting Time (FWT) to be used

◆ gb

uint8_t rfalNfcDepConfigs::gb[**RFAL_NFCDEP_GB_MAX_LEN**]

Pointer General Bytes (GB) to be used

◆ gbLen

uint8_t rfalNfcDepConfigs::gbLen

Length of the given GB in gb

◆ lr

uint8_t rfalNfcDepConfigs::lr

Length Reduction (LR) to be used

◆ nad

uint8_t rfalNfcDepConfigs::nad

Current Node Addressing (NAD)

◆ nfcid

uint8_t rfalNfcDepConfigs::nfcid[**RFAL_NFCDEP_NFCID3_LEN**]

Pointer to the NFCID to be used

◆ nfcidLen

uint8_t rfalNfcDepConfigs::nfcidLen

Length of the given NFCID in nfcid

◆ oper

uint8_t rfalNfcDepConfigs::oper

Operation config similar to NCI 1.0 Table 81

◆ role

rfalNfcDepRole rfalNfcDepConfigs::role

Current NFCIP role

◆ to

uint8_t rfalNfcDepConfigs::to

Timeout (TO) to be used

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

- **rfal_nfcDep.h**

---

rfalNfcDepAtrReq Struct Reference


```
#include <rfal_nfcDep.h>
```

Data Fields

uint8_t	**CMD1**
uint8_t	**CMD2**
uint8_t	**NFCID3** [**RFAL_NFCDEP_NFCID3_LEN**]
uint8_t	**DID**
uint8_t	**BSi**
uint8_t	**BRi**
uint8_t	**PPi**
uint8_t	**GBi** [**RFAL_NFCDEP_GB_MAX_LEN**]

Detailed Description

ATR_REQ command Digital 1.1 16.6.2

Field Documentation

◆ BRi

uint8_t rfalNfcDepAtrReq::BRi

Receiving Bitrate for Initiator

◆ BSi

uint8_t rfalNfcDepAtrReq::BSi

Sending Bitrate for Initiator

◆ CMD1

uint8_t rfalNfcDepAtrReq::CMD1

Command format 0xD4

◆ CMD2

uint8_t rfalNfcDepAtrReq::CMD2

Command Value

◆ DID

uint8_t rfalNfcDepAtrReq::DID

DID

◆ GBi

uint8_t rfalNfcDepAtrReq::GBi[**RFAL_NFCDEP_GB_MAX_LEN**]

General Bytes

◆ NFCID3

uint8_t rfalNfcDepAtrReq::NFCID3[**RFAL_NFCDEP_NFCID3_LEN**]

NFCID3 value

◆ PPi

uint8_t rfalNfcDepAtrReq::PPi

Optional Parameters presence indicator

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

Generated on Fri Mar 14 2025 11:45:10 for RFAL by doxygen 1.9.5

RFAL

rfalNfcDepAtrRes Struct Reference

RFAL » RFAL-AL » NFC-DEP

```
#include <rfal_nfcDep.h>
```

Data Fields

uint8_t	CMD1
uint8_t	CMD2
uint8_t	NFCID3 [RFAL_NFCDEP_NFCID3_LEN]
uint8_t	DID
uint8_t	BSt

uint8_t	**BRt**
uint8_t	**TO**
uint8_t	**PPt**
uint8_t	**GBt** [**RFAL_NFCDEP_GB_MAX_LEN**]

Detailed Description

ATR_RES response Digital 1.1 16.6.3

Field Documentation

◆ BRt

uint8_t rfalNfcDepAtrRes::BRt

Receiving Bitrate for Initiator

◆ BSt

uint8_t rfalNfcDepAtrRes::BSt

Sending Bitrate for Initiator

◆ CMD1

uint8_t rfalNfcDepAtrRes::CMD1

Response Byte 0xD5

◆ CMD2

uint8_t rfalNfcDepAtrRes::CMD2

Command Value

◆ DID

uint8_t rfalNfcDepAtrRes::DID

DID

◆ GBt

uint8_t rfalNfcDepAtrRes::GBt[**RFAL_NFCDEP_GB_MAX_LEN**]

General Bytes

◆ NFCID3

uint8_t rfalNfcDepAtrRes::NFCID3[**RFAL_NFCDEP_NFCID3_LEN**]

NFCID3 value

◆ PPt

uint8_t rfalNfcDepAtrRes::PPt

Optional Parameters presence indicator

◆ TO

uint8_t rfalNfcDepAtrRes::TO

Timeout

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

- **rfal_nfcDep.h**

rfalNfcDepBufFormat Struct Reference

RFAL » RFAL-AL » NFC-DEP

```
#include <rfal_nfcDep.h>
```

Data Fields

uint8_t	**prologue** [**RFAL_NFCDEP_DEPREQ_HEADER_LEN**]
uint8_t	**inf** [**RFAL_FEATURE_NFC_DEP_BLOCK_MAX_LEN**]

Detailed Description

Structure of transmit I-PDU Buffer format from caller

Examples

exampleRfalPoller.c.

Field Documentation

◆ inf

uint8_t rfalNfcDepBufFormat::inf[**RFAL_FEATURE_NFC_DEP_BLOCK_MAX_LEN**]

INF | Data area of the buffer

◆ prologue

uint8_t rfalNfcDepBufFormat::prologue[**RFAL_NFCDEP_DEPREQ_HEADER_LEN**]

Prologue space for NFC-DEP header

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

- **rfal_nfcDep.h**

---

rfalNfcDepPduBufFormat Struct Reference


```
#include <rfal_nfcDep.h>
```

Data Fields

uint8_t	**prologue** [**RFAL_NFCDEP_DEPREQ_HEADER_LEN**]
uint8_t	**pdu** [**RFAL_FEATURE_NFC_DEP_PDU_MAX_LEN**]

Detailed Description

Structure of APDU Buffer format from caller

Field Documentation

◆ pdu

uint8_t rfalNfcDepPduBufFormat::pdu[**RFAL_FEATURE_NFC_DEP_PDU_MAX_LEN**]

Complete PDU/Payload buffer

◆ prologue

uint8_t rfalNfcDepPduBufFormat::prologue[**RFAL_NFCDEP_DEPREQ_HEADER_LEN**]

Prologue/SoD buffer

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

- **rfal_nfcDep.h**

rfalNfcDepActivation Union Reference

RFAL » RFAL-AL » NFC-DEP

```
#include <rfal_nfcDep.h>
```

Collaboration diagram for rfalNfcDepActivation:

[legend]

Data Fields

struct {

rfalNfcDepAtrRes	ATR_RES
uint8_t	ATR_RESLen

}	**Target**
struct {	
rfalNfcDepAtrReq ATR_REQ	
uint8_t **ATR_REQLen**	
}	**Initiator**

Detailed Description

Activation info as Initiator and Target

Field Documentation

◆ ATR_REQ

rfalNfcDepAtrReq rfalNfcDepActivation::ATR_REQ

ATR REQ (Target mode)

◆ ATR_REQLen

uint8_t rfalNfcDepActivation::ATR_REQLen

ATR REQ length (Target mode)

◆ ATR_RES

rfalNfcDepAtrRes rfalNfcDepActivation::ATR_RES

ATR RES (Initiator mode)

◆ ATR_RESLen

uint8_t rfalNfcDepActivation::ATR_RESLen

ATR RES length (Initiator mode)

◆

struct { ... } rfalNfcDepActivation::Initiator

Initiator

◆

struct { ... } rfalNfcDepActivation::Target

Target

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

- **rfal_nfcDep.h**

---

rfalNfcDepInfo Struct Reference


```
#include <rfal_nfcDep.h>
```

Data Fields

uint8_t	**GBLen**
uint8_t	**WT**
uint32_t	**FWT**
uint32_t	**dFWT**
uint8_t	**LR**
uint16_t	**FS**
rfalBitRate	**DSI**
rfalBitRate	**DRI**
uint8_t	**DID**
uint8_t	**NAD**

Detailed Description

NFC-DEP device Info

Field Documentation

◆ dFWT

uint32_t rfalNfcDepInfo::dFWT

Delta FWT to be used (1/fc)

◆ DID

uint8_t rfalNfcDepInfo::DID

Device ID (RFAL_NFCDEP_DID_NO if no DID)

◆ DRI

rfalBitRate rfalNfcDepInfo::DRI

Bit Rate coding from Target to Initiator

◆ DSI

rfalBitRate rfalNfcDepInfo::DSI

Bit Rate coding from Initiator to Target

◆ FS

uint16_t rfalNfcDepInfo::FS

Frame Size

◆ FWT

uint32_t rfalNfcDepInfo::FWT

FWT to be used (1/fc)(ignored Listen Mode)

◆ GBLen

uint8_t rfalNfcDepInfo::GBLen

General Bytes length

◆ LR

uint8_t rfalNfcDepInfo::LR

Length Reduction coding the max payload

◆ NAD

uint8_t rfalNfcDepInfo::NAD

Node ADdress (RFAL_NFCDEP_NAD_NO if no NAD)

◆ WT

uint8_t rfalNfcDepInfo::WT

WT to be used (ignored in Listen Mode)

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

- **rfal_nfcDep.h**

rfalNfcDepDevice Struct Reference


```
#include <rfal_nfcDep.h>
```

Collaboration diagram for rfalNfcDepDevice:

[legend]

Data Fields

rfalNfcDepActivation	**activation**
rfalNfcDepInfo	**info**

Detailed Description

NFC-DEP Device structure

Examples

exampleRfalPoller.c.

Field Documentation

◆ activation

rfalNfcDepActivation rfalNfcDepDevice::activation

Activation Info

◆ info

rfalNfcDepInfo rfalNfcDepDevice::info

NFC-DEP device Info

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

- **rfal_nfcDep.h**

---

rfalNfcDepAtrParam Struct Reference


```
#include <rfal_nfcDep.h>
```

Data Fields

rfalNfcDepCommMode	commMode
uint8_t	operParam
uint8_t *	nfcid
uint8_t	nfcidLen
uint8_t	DID
uint8_t	NAD
uint8_t	BS
uint8_t	BR
uint8_t	LR
uint8_t *	GB
uint8_t	GBLen

Detailed Description

NFCIP Protocol structure for P2P Target

operParam : derives from NFC-Forum NCI NFC-DEP Operation Parameter NCI 1.1 Table 86: NFC-DEP Operation Parameter and it's a bit mask composed as: [0000b | Chain SHALL use max. Transport Data Byte[1b] | I-PDU with no Transport Data SHALL NOT be sent [1b] | NFC-DEP Target SHALL NOT send RTOX request [1b]]

Examples

exampleRfalPoller.c.

Field Documentation

◆ BR

uint8_t rfalNfcDepAtrParam::BR

Initiator's Bit Rates supported in Rx

Examples

exampleRfalPoller.c.

◆ BS

uint8_t rfalNfcDepAtrParam::BS

Initiator's Bit Rates supported in Tx

Examples

exampleRfalPoller.c.

◆ commMode

rfalNfcDepCommMode rfalNfcDepAtrParam::commMode

Initiator in Active P2P or Passive P2P

Examples

exampleRfalPoller.c.

◆ DID

uint8_t rfalNfcDepAtrParam::DID

Initiator's Device ID DID

Examples

exampleRfalPoller.c.

◆ GB

uint8_t* rfalNfcDepAtrParam::GB

Initiator's General Bytes (Gi)

Examples

exampleRfalPoller.c.

◆ GBLen

uint8_t rfalNfcDepAtrParam::GBLen

Initiator's General Bytes length

Examples

exampleRfalPoller.c.

◆ LR

uint8_t rfalNfcDepAtrParam::LR

Initiator's Length reduction

Examples

exampleRfalPoller.c.

◆ NAD

uint8_t rfalNfcDepAtrParam::NAD

Initiator's Node ID NAD

Examples

exampleRfalPoller.c.

◆ nfcid

uint8_t* rfalNfcDepAtrParam::nfcid

Initiator's NFCID2 or NFCID3

Examples

exampleRfalPoller.c.

◆ nfcidLen

uint8_t rfalNfcDepAtrParam::nfcidLen

Initiator's NFCID length (NFCID2/3)

Examples

exampleRfalPoller.c.

◆ operParam

uint8_t rfalNfcDepAtrParam::operParam

NFC-DEP Operation Parameter

Examples

exampleRfalPoller.c.

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

- **rfal_nfcDep.h**

rfalNfcDepListenActvParam Struct Reference


```
#include <rfal_nfcDep.h>
```

Collaboration diagram for rfalNfcDepListenActvParam:

[legend]

Data Fields

rfalNfcDepBufFormat *	**rxBuf**
uint16_t *	**rxLen**
bool *	**isRxChaining**
rfalNfcDepDevice *	**nfcDepDev**

Detailed Description

Structure of parameters to be passed in for nfcDepListenStartActivation

Field Documentation

◆ isRxChaining

bool* rfalNfcDepListenActvParam::isRxChaining

Received data is not complete

◆ nfcDepDev

rfalNfcDepDevice* rfalNfcDepListenActvParam::nfcDepDev

NFC-DEP device info

◆ rxBuf

rfalNfcDepBufFormat* rfalNfcDepListenActvParam::rxBuf

Receive Buffer struct reference

◆ rxLen

uint16_t* rfalNfcDepListenActvParam::rxLen

Receive INF data length in bytes

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

- **rfal_nfcDep.h**

---

rfalNfcDepTargetParam Struct Reference

RFAL » **RFAL-AL** » **NFC-DEP**

```
#include <rfal_nfcDep.h>
```

Data Fields

rfalNfcDepCommMode	**commMode**
uint8_t	**nfcid3** [**RFAL_NFCDEP_NFCID3_LEN**]
uint8_t	**bst**
uint8_t	**brt**
uint8_t	**to**
uint8_t	**ppt**
uint8_t	**GBt** [**RFAL_NFCDEP_GB_MAX_LEN**]
uint8_t	**GBtLen**
uint8_t	**operParam**

Detailed Description

NFCIP Protocol structure for P2P Target

operParam : derives from NFC-Forum NCI NFC-DEP Operation Parameter NCI 1.1 Table 86: NFC-DEP Operation Parameter and it's a bit mask composed as: [0000b | Chain SHALL use max. Transport Data Byte[1b] | I-PDU with no Transport Data SHALL NOT be sent [1b] | NFC-DEP Target SHALL NOT send RTOX request [1b]]

Field Documentation

◆ brt

uint8_t rfalNfcDepTargetParam::brt

Target's Bit Rates supported in Rx

◆ bst

uint8_t rfalNfcDepTargetParam::bst

Target's Bit Rates supported in Tx

◆ commMode

rfalNfcDepCommMode rfalNfcDepTargetParam::commMode

Target in Active P2P or Passive P2P

◆ GBt

uint8_t rfalNfcDepTargetParam::GBt[**RFAL_NFCDEP_GB_MAX_LEN**]

Target's General Bytes (Gt)

◆ GBtLen

uint8_t rfalNfcDepTargetParam::GBtLen

Target's General Bytes length

◆ nfcid3

uint8_t rfalNfcDepTargetParam::nfcid3[**RFAL_NFCDEP_NFCID3_LEN**]

Target's NFCID3

◆ operParam

uint8_t rfalNfcDepTargetParam::operParam

NFC-DEP Operation Parameter

◆ ppt

uint8_t rfalNfcDepTargetParam::ppt

Target's Presence optional Params(PPt)

◆ to

uint8_t rfalNfcDepTargetParam::to

Target's timeout (TO) value

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

- **rfal_nfcDep.h**

---

rfalNfcDepTxRxParam Struct Reference


```
#include <rfal_nfcDep.h>
```

Collaboration diagram for rfalNfcDepTxRxParam:

[legend]

Data Fields

rfalNfcDepBufFormat *	**txBuf**
uint16_t	**txBufLen**
bool	**isTxChaining**
rfalNfcDepBufFormat *	**rxBuf**
uint16_t *	**rxLen**
bool *	**isRxChaining**
uint32_t	**FWT**
uint32_t	**dFWT**
uint16_t	**FSx**
uint8_t	**DID**

Detailed Description

Structure of parameters to be passed in for nfcDepStartIpduTransceive

Examples

exampleRfalPoller.c.

Field Documentation

◆ dFWT

uint32_t rfalNfcDepTxRxParam::dFWT

Delta FWT to be used

Examples

exampleRfalPoller.c.

◆ DID

uint8_t rfalNfcDepTxRxParam::DID

Device ID (RFAL_ISODEP_NO_DID if no DID)

Examples

exampleRfalPoller.c.

◆ FSx

uint16_t rfalNfcDepTxRxParam::FSx

Other device Frame Size (FSD or FSC)

Examples

exampleRfalPoller.c.

◆ FWT

uint32_t rfalNfcDepTxRxParam::FWT

FWT to be used (ignored in Listen Mode)

Examples

exampleRfalPoller.c.

◆ isRxChaining

bool* rfalNfcDepTxRxParam::isRxChaining

Received data is not complete

Examples

exampleRfalPoller.c.

◆ isTxChaining

bool rfalNfcDepTxRxParam::isTxChaining

Transmit data is not complete

Examples

exampleRfalPoller.c.

◆ rxBuf

rfalNfcDepBufFormat* rfalNfcDepTxRxParam::rxBuf

Receive Buffer struct reference

Examples

exampleRfalPoller.c.

◆ rxLen

uint16_t* rfalNfcDepTxRxParam::rxLen

Receive INF data length

Examples

exampleRfalPoller.c.

◆ txBuf

rfalNfcDepBufFormat* rfalNfcDepTxRxParam::txBuf

Transmit Buffer struct reference

Examples

exampleRfalPoller.c.

◆ txBufLen

uint16_t rfalNfcDepTxRxParam::txBufLen

Transmit Buffer INF field length in bytes

Examples

exampleRfalPoller.c.

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

- rfal_nfcDep.h

---

rfalNfcDepPduTxRxParam Struct Reference

```
#include <rfal_nfcDep.h>
```

Collaboration diagram for rfalNfcDepPduTxRxParam:

[legend]

Data Fields

rfalNfcDepPduBufFormat *	**txBuf**
uint16_t	**txBufLen**
rfalNfcDepPduBufFormat *	**rxBuf**
uint16_t *	**rxLen**
rfalNfcDepBufFormat *	**tmpBuf**
uint32_t	**FWT**
uint32_t	**dFWT**
uint16_t	**FSx**
uint8_t	**DID**

Detailed Description

Structure of parameters used on NFC DEP PDU Transceive

Field Documentation

◆ dFWT

uint32_t rfalNfcDepPduTxRxParam::dFWT

Delta FWT to be used

◆ DID

uint8_t rfalNfcDepPduTxRxParam::DID

Device ID (RFAL_ISODEP_NO_DID if no DID)

◆ FSx

uint16_t rfalNfcDepPduTxRxParam::FSx

Other device Frame Size (FSD or FSC)

◆ FWT

uint32_t rfalNfcDepPduTxRxParam::FWT

FWT to be used (ignored in Listen Mode)

◆ rxBuf

rfalNfcDepPduBufFormat* rfalNfcDepPduTxRxParam::rxBuf

Receive Buffer struct reference in Bytes

◆ rxLen

uint16_t* rfalNfcDepPduTxRxParam::rxLen

Received INF data length in Bytes

◆ tmpBuf

rfalNfcDepBufFormat* rfalNfcDepPduTxRxParam::tmpBuf

Temp buffer for single PDUs (internal)

◆ txBuf

rfalNfcDepPduBufFormat* rfalNfcDepPduTxRxParam::txBuf

Transmit Buffer struct reference

◆ txBufLen

uint16_t rfalNfcDepPduTxRxParam::txBufLen

Transmit Buffer INF field length in Bytes

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

- **rfal_nfcDep.h**

NFC-F

RFAL NFC-F Module

More...

Collaboration diagram for NFC-F:

Data Structures

struct	**rfalNfcfSensfRes**
struct	**rfalNfcfPollDevice**
struct	**rfalNfcfListenDevice**
struct	**rfalNfcfBlockListElem**
struct	**rfalNfcfServBlockListParam**

Macros

#define	**RFAL_NFCF_NFCID2_LEN**	8U
#define	**RFAL_NFCF_SENSF_RES_LEN_MIN**	16U
#define	**RFAL_NFCF_SENSF_RES_LEN_MAX**	18U
#define	**RFAL_NFCF_SENSF_RES_PAD0_LEN**	2U
#define	**RFAL_NFCF_SENSF_RES_PAD1_LEN**	3U
#define	**RFAL_NFCF_SENSF_RES_RD_LEN**	2U
#define	**RFAL_NFCF_SENSF_RES_BYTE1**	1U
#define	**RFAL_NFCF_SENSF_SC_LEN**	2U
#define	**RFAL_NFCF_SENSF_PARAMS_SC1_POS**	0U

```
#define RFAL_NFCF_SENSF_PARAMS_SC2_POS 1U
#define RFAL_NFCF_SENSF_PARAMS_RC_POS 2U
#define RFAL_NFCF_SENSF_PARAMS_TSN_POS 3U
#define RFAL_NFCF_POLL_MAXCARDS 16U
#define RFAL_NFCF_CMD_POS 0U
#define RFAL_NFCF_CMD_LEN 1U
#define RFAL_NFCF_LENGTH_LEN 1U
#define RFAL_NFCF_HEADER_LEN (RFAL_NFCF_LENGTH_LEN + RFAL_NFCF_CMD_LEN)
#define RFAL_NFCF_NOS_LEN 1U
#define RFAL_NFCF_NOB_LEN 1U
#define RFAL_NFCF_SENSF_NFCID2_BYTE1_POS 0U
#define RFAL_NFCF_SENSF_NFCID2_BYTE2_POS 1U
#define RFAL_NFCF_SENSF_NFCID2_PROT_TYPE_LEN 2U
#define RFAL_NFCF_SENSF_NFCID2_BYTE1_NFCDEP 0x01U
#define RFAL_NFCF_SENSF_NFCID2_BYTE2_NFCDEP 0xFEU
#define RFAL_NFCF_SYSTEMCODE 0xFFFFU
#define RFAL_NFCF_SYSTEMCODE_LEN 2U
#define RFAL_NFCF_BLOCK_LEN 16U
#define RFAL_NFCF_CHECKUPDATE_RES_ST1_POS 9U
#define RFAL_NFCF_CHECKUPDATE_RES_ST2_POS 10U
#define RFAL_NFCF_CHECKUPDATE_RES_NOB_POS 11U
```

#define **RFAL_NFCF_STATUS_FLAG_SUCCESS** 0x00U

#define **RFAL_NFCF_STATUS_FLAG_ERROR** 0xFFU

#define **RFAL_NFCF_BLOCKLISTELEM_MAX_LEN** 3U

#define **RFAL_NFCF_BLOCKLISTELEM_LEN_BIT** 0x80U

#define **RFAL_NFCF_SERVICECODE_RDONLY** 0x000BU

#define **RFAL_NFCF_SERVICECODE_RDWR** 0x0009U

#define **RFAL_NFCF_TEST_LB_CMD0** 0xD8U

#define **RFAL_NFCF_TEST_LB_CMD1** 0x00U

#define **rfalNfcfIsNfcDepSupported**(dev)

Typedefs

typedef uint16_t **rfalNfcfServ**

Enumerations

enum {
RFAL_NFCF_CMD_POLLING = 0x00 , **RFAL_NFCF_CMD_POLLING_RES** = 0x01 ,
RFAL_NFCF_CMD_REQUEST_SERVICE = 0x02 , **RFAL_NFCF_CMD_REQUEST_RESPONSE** = 0x04 ,
RFAL_NFCF_CMD_READ_WITHOUT_ENCRYPTION = 0x06 ,
RFAL_NFCF_CMD_READ_WITHOUT_ENCRYPTION_RES = 0x07 ,
RFAL_NFCF_CMD_WRITE_WITHOUT_ENCRYPTION = 0x08 ,
RFAL_NFCF_CMD_WRITE_WITHOUT_ENCRYPTION_RES = 0x09 ,
RFAL_NFCF_CMD_REQUEST_SYSTEM_CODE = 0x0c , **RFAL_NFCF_CMD_AUTHENTICATION1** = 0x10 ,
RFAL_NFCF_CMD_AUTHENTICATION2 = 0x12 , **RFAL_NFCF_CMD_READ** = 0x14 ,
RFAL_NFCF_CMD_WRITE = 0x16
}

Functions

ReturnCode **rfalNfcfPollerInitialize** (**rfalBitRate** bitRate)

Initialize NFC-F Poller mode. More...

ReturnCode **rfalNfcfPollerCheckPresence** (void)
NFC-F Poller Check Presence. More...

ReturnCode **rfalNfcfPollerStartCheckPresence** (void)
NFC-F Poller Start Check Presence. More...

ReturnCode **rfalNfcfPollerGetCheckPresenceStatus** (void)
NFC-F Poller Get Check Presence Status. More...

ReturnCode **rfalNfcfPollerPoll** (**rfalFeliCaPollSlots** slots, uint16_t sysCode, uint8_t reqCode, **rfalFeliCaPollRes** *cardList, uint8_t *devCnt, uint8_t *collisions)
NFC-F Poller Poll. More...

ReturnCode **rfalNfcfPollerCollisionResolution** (**rfalComplianceMode** compMode, uint8_t devLimit, **rfalNfcfListenDevice** *nfcfDevList, uint8_t *devCnt)
NFC-F Poller Full Collision Resolution. More...

ReturnCode **rfalNfcfPollerStartCollisionResolution** (**rfalComplianceMode** compMode, uint8_t devLimit, **rfalNfcfListenDevice** *nfcfDevList, uint8_t *devCnt)
NFC-F Start Poller Collision Resolution. More...

ReturnCode **rfalNfcfPollerGetCollisionResolutionStatus** (void)
NFC-F Poller Get Collision Resolution Status. More...

ReturnCode **rfalNfcfPollerCheck** (const uint8_t *nfcid2, const **rfalNfcfServBlockListParam** *servBlock, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvdLen)
NFC-F Poller Check/Read. More...

ReturnCode **rfalNfcfPollerUpdate** (const uint8_t *nfcid2, const **rfalNfcfServBlockListParam** *servBlock, uint8_t *txBuf, uint16_t txBufLen, const uint8_t *blockData, uint8_t *rxBuf, uint16_t rxBufLen)
NFC-F Poller Update/Write. More...

bool **rfalNfcfListenerIsT3TReq** (const uint8_t *buf, uint16_t bufLen, uint8_t *nfcid2)
NFC-F Listener is T3T Request
More...

Detailed Description

Macro Definition Documentation

◆ RFAL_NFCF_BLOCK_LEN

```
#define RFAL_NFCF_BLOCK_LEN   16U
```

NFCF T3T Block size T3T 1.0 4.1

◆ RFAL_NFCF_BLOCKLISTELEM_LEN_BIT

```
#define RFAL_NFCF_BLOCKLISTELEM_LEN_BIT   0x80U
```

Block List Element Length bit (2|3 bytes) T3T 1.0 5.6.1

◆ RFAL_NFCF_BLOCKLISTELEM_MAX_LEN

```
#define RFAL_NFCF_BLOCKLISTELEM_MAX_LEN   3U
```

Block List Element max Length (3 bytes) T3T 1.0 5.6.1

◆ RFAL_NFCF_CHECKUPDATE_RES_NOB_POS

```
#define RFAL_NFCF_CHECKUPDATE_RES_NOB_POS   11U
```

Check|Update Res Number of Blocks position T3T 1.0 Table 8

◆ RFAL_NFCF_CHECKUPDATE_RES_ST1_POS

```
#define RFAL_NFCF_CHECKUPDATE_RES_ST1_POS   9U
```

Check|Update Res Status Flag 1 position T3T 1.0 Table 8

◆ RFAL_NFCF_CHECKUPDATE_RES_ST2_POS

```
#define RFAL_NFCF_CHECKUPDATE_RES_ST2_POS   10U
```

Check|Update Res Status Flag 2 position T3T 1.0 Table 8

◆ RFAL_NFCF_CMD_LEN

```
#define RFAL_NFCF_CMD_LEN   1U
```

Command/Responce code length

◆ RFAL_NFCF_CMD_POS

```
#define RFAL_NFCF_CMD_POS   0U
```

Command/Responce code length

◆ RFAL_NFCF_HEADER_LEN

```
#define RFAL_NFCF_HEADER_LEN   (RFAL_NFCF_LENGTH_LEN + RFAL_NFCF_CMD_LEN)
```

Header length

◆ RFAL_NFCF_LENGTH_LEN

```
#define RFAL_NFCF_LENGTH_LEN   1U
```

LEN field length

◆ RFAL_NFCF_NFCID2_LEN

```
#define RFAL_NFCF_NFCID2_LEN   8U
```

NFCID2 (FeliCa IDm) length

Examples

exampleRfalPoller.c.

◆ RFAL_NFCF_NOB_LEN

```
#define RFAL_NFCF_NOB_LEN   1U
```

Number of Blocks length

◆ RFAL_NFCF_NOS_LEN

```
#define RFAL_NFCF_NOS_LEN   1U
```

Number of Services length

◆ RFAL_NFCF_POLL_MAXCARDS

```
#define RFAL_NFCF_POLL_MAXCARDS   16U
```

Max number slots/cards 16

◆ RFAL_NFCF_SENSF_NFCID2_BYTE1_NFCDEP

```
#define RFAL_NFCF_SENSF_NFCID2_BYTE1_NFCDEP   0x01U
```

NFCID2 byte1 NFC-DEP support Digital 1.0 Table 44

◆ RFAL_NFCF_SENSF_NFCID2_BYTE1_POS

```
#define RFAL_NFCF_SENSF_NFCID2_BYTE1_POS   0U
```

NFCID2 byte1 position

◆ RFAL_NFCF_SENSF_NFCID2_BYTE2_NFCDEP

```
#define RFAL_NFCF_SENSF_NFCID2_BYTE2_NFCDEP   0xFEU
```

NFCID2 byte2 NFC-DEP support Digital 1.0 Table 44

◆ RFAL_NFCF_SENSF_NFCID2_BYTE2_POS

```
#define RFAL_NFCF_SENSF_NFCID2_BYTE2_POS   1U
```

NFCID2 byte2 position

◆ RFAL_NFCF_SENSF_NFCID2_PROT_TYPE_LEN

```
#define RFAL_NFCF_SENSF_NFCID2_PROT_TYPE_LEN   2U
```

NFCID2 length for byte 1 and byte 2 indicating NFC-DEP or T3T support

◆ RFAL_NFCF_SENSF_PARAMS_RC_POS

```
#define RFAL_NFCF_SENSF_PARAMS_RC_POS   2U
```

Request Code position in the SENSF_REQ

◆ RFAL_NFCF_SENSF_PARAMS_SC1_POS

```
#define RFAL_NFCF_SENSF_PARAMS_SC1_POS   0U
```

System Code byte1 position in the SENSF_REQ

◆ RFAL_NFCF_SENSF_PARAMS_SC2_POS

```
#define RFAL_NFCF_SENSF_PARAMS_SC2_POS   1U
```

System Code byte2 position in the SENSF_REQ

◆ RFAL_NFCF_SENSF_PARAMS_TSN_POS

```
#define RFAL_NFCF_SENSF_PARAMS_TSN_POS   3U
```

Time Slot Number position in the SENSF_REQ

◆ RFAL_NFCF_SENSF_RES_BYTE1

```
#define RFAL_NFCF_SENSF_RES_BYTE1   1U
```

SENSF_RES first byte value

◆ RFAL_NFCF_SENSF_RES_LEN_MAX

```
#define RFAL_NFCF_SENSF_RES_LEN_MAX   18U
```

SENSF_RES maximum length

◆ RFAL_NFCF_SENSF_RES_LEN_MIN

```
#define RFAL_NFCF_SENSF_RES_LEN_MIN   16U
```

SENSF_RES minimum length

◆ RFAL_NFCF_SENSF_RES_PAD0_LEN

```
#define RFAL_NFCF_SENSF_RES_PAD0_LEN   2U
```

SENSF_RES PAD0 length

◆ RFAL_NFCF_SENSF_RES_PAD1_LEN

```
#define RFAL_NFCF_SENSF_RES_PAD1_LEN   3U
```

SENSF_RES PAD1 length

◆ RFAL_NFCF_SENSF_RES_RD_LEN

```
#define RFAL_NFCF_SENSF_RES_RD_LEN   2U
```

SENSF_RES Request Data length

◆ RFAL_NFCF_SENSF_SC_LEN

```
#define RFAL_NFCF_SENSF_SC_LEN   2U
```

Felica SENSF_REQ System Code length

◆ RFAL_NFCF_SERVICECODE_RDONLY

```
#define RFAL_NFCF_SERVICECODE_RDONLY   0x000BU
```

NDEF Service Code as Read-Only T3T 1.0 7.2.1

◆ RFAL_NFCF_SERVICECODE_RDWR

```
#define RFAL_NFCF_SERVICECODE_RDWR   0x0009U
```

NDEF Service Code as Read and Write T3T 1.0 7.2.1

◆ RFAL_NFCF_STATUS_FLAG_ERROR

```
#define RFAL_NFCF_STATUS_FLAG_ERROR   0xFFU
```

Check response Number of Blocks position T3T 1.0 Table 11

◆ RFAL_NFCF_STATUS_FLAG_SUCCESS

```
#define RFAL_NFCF_STATUS_FLAG_SUCCESS   0x00U
```

Check response Number of Blocks position T3T 1.0 Table 11

◆ RFAL_NFCF_SYSTEMCODE

```
#define RFAL_NFCF_SYSTEMCODE   0xFFFFU
```

SENSF_RES Default System Code Digital 2.3 8.6.1.5

◆ RFAL_NFCF_SYSTEMCODE_LEN

#define RFAL_NFCF_SYSTEMCODE_LEN 2U

SENSF_RES System Code length Digital 2.3 8.6.1

◆ RFAL_NFCF_TEST_LB_CMD0

#define RFAL_NFCF_TEST_LB_CMD0 0xD8U

T3T loopback CMD0 ETSI TS 102 695-1 5.6.4.4.2

◆ RFAL_NFCF_TEST_LB_CMD1

#define RFAL_NFCF_TEST_LB_CMD1 0x00U

T3T loopback CMD1 ETSI TS 102 695-1 5.6.4.4.2

◆ rfalNfcfIsNfcDepSupported

#define rfalNfcfIsNfcDepSupported (dev)

Value:

```
( (((rfalNfcfListenDevice*)(dev))->sensfRes.NFCID2[RFAL_NFCF_SENSF_NFCID2_BYTE1_POS] == RFAL_NFCF_SENSF_NFCID2_BYTE1_NFCDEP) && \
(((rfalNfcfListenDevice*)(dev))->sensfRes.NFCID2[RFAL_NFCF_SENSF_NFCID2_BYTE2_POS] == RFAL_NFCF_SENSF_NFCID2_BYTE2_NFCDEP) )
```

Checks if the given NFC-F device indicates NFC-DEP support

Examples

exampleRfalPoller.c.

Typedef Documentation

◆ rfalNfcfServ

typedef uint16_t **rfaINfcfServ**

NFC-F Service Code

Enumeration Type Documentation

◆ anonymous enum

anonymous enum

NFC-F Felica command set JIS X6319-4 9.1

Enumerator	
RFAL_NFCF_CMD_POLLING	SENSF_REQ (Felica Poll/REQC command to identify a card)
RFAL_NFCF_CMD_POLLING_RES	SENSF_RES (Felica Poll/REQC command response)
RFAL_NFCF_CMD_REQUEST_SERVICE	verify the existence of Area and Service
RFAL_NFCF_CMD_REQUEST_RESPONSE	verify the existence of a card
RFAL_NFCF_CMD_READ_WITHOUT_ENCRYPTION	read Block Data from a Service that requires no authentication
RFAL_NFCF_CMD_READ_WITHOUT_ENCRYPTION_RES	read Block Data response from a Service with no authentication
RFAL_NFCF_CMD_WRITE_WITHOUT_ENCRYPTION	write Block Data to a Service that requires no authentication
RFAL_NFCF_CMD_WRITE_WITHOUT_ENCRYPTION_RES	write Block Data response to a Service with no authentication
RFAL_NFCF_CMD_REQUEST_SYSTEM_CODE	acquire the System Code registered to a card
RFAL_NFCF_CMD_AUTHENTICATION1	authenticate a card
RFAL_NFCF_CMD_AUTHENTICATION2	allow a card to authenticate a Reader/Writer
RFAL_NFCF_CMD_READ	read Block Data from a Service that requires authentication
RFAL_NFCF_CMD_WRITE	write Block Data to a Service that requires authentication

Function Documentation

◆ rfalNfcfListenerIsT3TReq()

```
bool rfalNfcfListenerIsT3TReq ( const uint8_t *  buf,
                                uint16_t         bufLen,
                                uint8_t *        nfcid2
                              )
```

This method checks if the given data is a valid T3T command (Read or Write) and in case a valid request has been received it may output the request's NFCID2

Parameters

- [in] **buf** : buffer holding Initiator's received command
- [in] **bufLen** : length of received command in bytes
- [out] **nfcid2** : pointer to where the NFCID2 may be outputed, nfcid2 has NFCF_SENSF_NFCID2_LEN as length Pass NULL if output parameter not desired

Returns

true : Valid T3T command (Read or Write) received

false : Invalid protocol request

◆ rfalNfcfPollerCheck()

```
ReturnCode rfalNfcfPollerCheck ( const uint8_t *                      nfcid2,
                                 const rfalNfcfServBlockListParam *   servBlock,
                                 uint8_t *                            rxBuf,
                                 uint16_t                             rxBufLen,
                                 uint16_t *                           rcvdLen
                               )
```

It computes a Check / Read command accoring to T3T 1.0 and JIS X6319-4 and sends it to PICC. If sucessfully, the rxBuf will contain the the number of blocks in the first byte followed by the blocks data.

Parameters

- [in] **nfcid2** : nfcid2 of the device
- [in] **servBlock** : parameter containing the list of Services and Blocks to be addressed by this command

[out] **rxBuf** : buffer to place check/read data

[in] **rxBufLen** : size of the rxBuf

[out] **rcvdLen** : length of data placed in rxBuf

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or mode not set

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_REQUEST : The request was executed with error

RFAL_ERR_NONE : No error

◆ rfalNfcfPollerCheckPresence()

ReturnCode rfalNfcfPollerCheckPresence (void)

This function sends a Poll/SENSF command according to NFC Activity spec It detects if a NCF-F device is within range

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error, no listener device detected

RFAL_ERR_NONE : No error and some NFC-F device was detected

Examples

exampleRfalPoller.c.

◆ rfalNfcfPollerCollisionResolution()

```
ReturnCode rfalNfcfPollerCollisionResolution ( rfalComplianceMode    compMode,
                                               uint8_t               devLimit,
                                               rfalNfcfListenDevice * nfcfDevList,
                                               uint8_t *             devCnt
```

)

Performs a full Collision resolution as defined in Activity 1.1 9.3.4

Parameters

- [in] **compMode** : compliance mode to be performed
- [in] **devLimit** : device limit value, and size nfcaDevList
- [out] **nfcfDevList** : NFC-F listener devices list
- [out] **devCnt** : Devices found counter

NFC-F Collision Resolution uses information collected at Technlology dectection (if provided on nfcfDevList). Technology Detection shall be performed using 4 slots, therefore at least 4 slots/devLimit shall be allocated on nfcfDevList.

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or mode not set

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_NONE : No error

Examples

exampleRfalPoller.c.

◆ rfalNfcfPollerGetCheckPresenceStatus()

ReturnCode rfalNfcfPollerGetCheckPresenceStatus (void)

This function gets the status of the Check Presense operation triggered by **rfalNfcfPollerStartCheckPresence()**

Returns

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error, no listener device detected

RFAL_ERR_NONE : No error and some NFC-F device was detected

◆ rfalNfcfPollerGetCollisionResolutionStatus()

ReturnCode rfalNfcfPollerGetCollisionResolutionStatus (void)

This function gets the status of the Collision Resolution operation triggered by **rfalNfcfPollerStartCollisionResolution()**

Returns

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error, no listener device detected

RFAL_ERR_NONE : No error and some NFC-F device was detected

◆ rfalNfcfPollerInitialize()

ReturnCode rfalNfcfPollerInitialize (**rfalBitRate** bitRate)

This methods configures RFAL RF layer to perform as a NFC-F Poller/RW (FeliCa PCD) including all default timings

Parameters

[in] **bitRate** : NFC-F bitrate to be initialize (212 or 424)

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or mode not set

RFAL_ERR_PARAM : Incorrect bitrate

RFAL_ERR_NONE : No error

Examples

exampleRfalPoller.c.

◆ rfalNfcfPollerPoll()

```
ReturnCode rfalNfcfPollerPoll ( rfalFeliCaPollSlots  slots,
                                uint16_t             sysCode,
                                uint8_t              reqCode,
                                rfalFeliCaPollRes *  cardList,
```

```
                                uint8_t *          devCnt,
                                uint8_t *          collisions
                              )
```

This function sends to all PICCs in field the POLL command with the given number of slots.

Parameters

- [in] **slots** : the number of slots to be performed
- [in] **sysCode** : as given in FeliCa poll command
- [in] **reqCode** : FeliCa communication parameters
- [out] **cardList** : Parameter of type rfalFeliCaPollRes which will hold the cards found
- [out] **devCnt** : actual number of cards found
- [out] **collisions** : number of collisions encountered

Warning

the list cardList has to be as big as the number of slots for the Poll

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error, no listener device detected

RFAL_ERR_NONE : No error and some NFC-F device was detected

◆ rfalNfcfPollerStartCheckPresence()

ReturnCode rfalNfcfPollerStartCheckPresence (void)

This function triggers a Poll/SENSF command according to NFC Activity spec It detects if a NCF-F device is within range

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_NONE : No error and some NFC-F device was detected

◆ rfalNfcfPollerStartCollisionResolution()

```
ReturnCode rfalNfcfPollerStartCollisionResolution ( rfalComplianceMode      compMode,
                                                    uint8_t                 devLimit,
                                                    rfalNfcfListenDevice *  nfcfDevList,
                                                    uint8_t *               devCnt
                                                  )
```

Triggers a Collision Resolution as defined in Activity 2.1 9.3.6

NFC-F Collision Resolution uses information collected at Technlology dectection (if provided on nfcfDevList). Technology Detection shall be performed using 4 slots, therefore at least 4 slots/devLimit shall be allocated on nfcfDevList.

Parameters

- [in] **compMode** : compliance mode to be performed
- [in] **devLimit** : device limit value, and size nfcaDevList
- [out] **nfcfDevList** : NFC-F listener devices list
- [out] **devCnt** : Devices found counter

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or mode not set

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_NONE : No error

◆ rfalNfcfPollerUpdate()

```
ReturnCode rfalNfcfPollerUpdate ( const uint8_t *                    nfcid2,
                                  const rfalNfcfServBlockListParam * servBlock,
                                  uint8_t *                          txBuf,
                                  uint16_t                           txBufLen,
                                  const uint8_t *                    blockData,
                                  uint8_t *                          rxBuf,
                                  uint16_t                           rxBufLen
                                )
```

It computes a Update / Write command accoring to T3T 1.0 and JIS X6319-4 and sends it to PICC.

Parameters

[in]	**nfcid2**	: nfcid2 of the device
[in]	**servBlock**	: parameter containing the list of Services and Blocks to be addressed by this command
[in]	**txBuf**	: buffer where the request will be composed
[in]	**txBufLen**	: size of txBuf
[in]	**blockData**	: data to written on the given block(s)
[out]	**rxBuf**	: buffer to place check/read data
[in]	**rxBufLen**	: size of the rxBuf

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or mode not set

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_REQUEST : The request was executed with error

RFAL_ERR_NONE : No error

RFAL

Main Page

Modules

Files

Examples

Data Structures

Data Structure Index

Data Fields

rfalNfcfSensfRes Struct Reference


```
#include <rfal_nfcf.h>
```

Data Fields

uint8_t	**CMD**
uint8_t	**NFCID2** [**RFAL_NFCF_NFCID2_LEN**]
uint8_t	**PAD0** [**RFAL_NFCF_SENSF_RES_PAD0_LEN**]
uint8_t	**PAD1** [**RFAL_NFCF_SENSF_RES_PAD1_LEN**]
uint8_t	**MRTIcheck**
uint8_t	**MRTIupdate**
uint8_t	**PAD2**
uint8_t	**RD** [**RFAL_NFCF_SENSF_RES_RD_LEN**]

Detailed Description

NFC-F SENSF_RES format Digital 1.1 8.6.2

Field Documentation

◆ CMD

uint8_t rfalNfcfSensfRes::CMD

Command Code: 01h

◆ MRTIcheck

uint8_t rfalNfcfSensfRes::MRTIcheck

MRTIcheck

◆ MRTIupdate

uint8_t rfalNfcfSensfRes::MRTIupdate

MRTIupdate

◆ NFCID2

uint8_t rfalNfcfSensfRes::NFCID2[**RFAL_NFCF_NFCID2_LEN**]

NFCID2

◆ PAD0

uint8_t rfalNfcfSensfRes::PAD0[**RFAL_NFCF_SENSF_RES_PAD0_LEN**]

PAD0

◆ PAD1

uint8_t rfalNfcfSensfRes::PAD1[**RFAL_NFCF_SENSF_RES_PAD1_LEN**]

PAD1

◆ PAD2

uint8_t rfalNfcfSensfRes::PAD2

PAD2

◆ RD

uint8_t rfalNfcfSensfRes::RD[**RFAL_NFCF_SENSF_RES_RD_LEN**]

Request Data

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

- **rfal_nfcf.h**

RFAL

rfalNfcfPollDevice Struct Reference


```
#include <rfal_nfcf.h>
```

Data Fields

uint8_t **NFCID2** [**RFAL_NFCF_NFCID2_LEN**]

Detailed Description

NFC-F poller device (PCD) struct

Field Documentation

◆ NFCID2

uint8_t rfalNfcfPollDevice::NFCID2[**RFAL_NFCF_NFCID2_LEN**]

NFCID2

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

- **rfal_nfcf.h**

rfalNfcfListenDevice Struct Reference

RFAL » RFAL-AL » NFC-F

```
#include <rfal_nfcf.h>
```

Collaboration diagram for rfalNfcfListenDevice:

[legend]

Data Fields

uint8_t	**sensfResLen**
rfalNfcfSensfRes	**sensfRes**

Detailed Description

NFC-F listener device (PICC) struct

Examples

exampleRfalPoller.c.

Field Documentation

◆ sensfRes

rfalNfcfSensfRes rfalNfcfListenDevice::sensfRes

SENF_RES

◆ sensfResLen

uint8_t rfalNfcfListenDevice::sensfResLen

SENF_RES length

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

- rfal_nfcf.h

Generated on Fri Mar 14 2025 11:45:10 for RFAL by doxygen 1.9.5

RFAL

Main Page

Modules

Files

Examples

Data Structures

Data Structure Index

Data Fields

Data Fields

rfalNfcfBlockListElem Struct Reference


```
#include <rfal_nfcf.h>
```

Data Fields

uint8_t	**conf**
uint16_t	**blockNum**

Detailed Description

NFC-F Block List Element (2 or 3 bytes element) T3T 1.0 5.6.1

Field Documentation

◆ blockNum

uint16_t rfalNfcfBlockListElem::blockNum

Block Number

◆ conf

uint8_t rfalNfcfBlockListElem::conf

Access Mode | Serv Code List Order

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

- **rfal_nfcf.h**

rfalNfcfServBlockListParam Struct Reference


```
#include <rfal_nfcf.h>
```

Collaboration diagram for rfalNfcfServBlockListParam:

[legend]

Data Fields

uint8_t	**numServ**
rfalNfcfServ *	**servList**
uint8_t	**numBlock**
rfalNfcfBlockListElem *	**blockList**

Detailed Description

Check Update Service list and Block list parameter

Field Documentation

◆ blockList

rfalNfcfBlockListElem* rfalNfcfServBlockListParam::blockList

Block Number List

◆ numBlock

uint8_t rfalNfcfServBlockListParam::numBlock

Number of Blocks

◆ numServ

uint8_t rfalNfcfServBlockListParam::numServ

Number of Services

◆ servList

rfalNfcfServ* rfalNfcfServBlockListParam::servList

Service Code List

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

- **rfal_nfcf.h**

NFC-V

RFAL NFC-V Module. More...

Collaboration diagram for NFC-V:

Data Structures

struct	**rfalNfcvInventoryRes**
struct	**rfalNfcvGenericReq**
struct	**rfalNfcvGenericRes**
struct	**rfalNfcvListenDevice**

Macros

#define	**RFAL_NFCV_UID_LEN** 8U
#define	**RFAL_NFCV_MAX_BLOCK_LEN** 32U
#define	**RFAL_NFCV_BNO_LEN** 1U
#define	**RFAL_NFCV_CRC_LEN** 2U
#define	**RFAL_NFCV_MAX_GEN_DATA_LEN** (**RFAL_NFCV_MAX_BLOCK_LEN** + **RFAL_NFCV_BNO_LEN** + **RFAL_NFCV_UID_LEN**)

```
#define RFAL_NFCV_BLOCKNUM_LEN 1U
```

```
#define RFAL_NFCV_BLOCKNUM_EXTENDED_LEN 2U
```

```
#define RFAL_NFCV_PARAM_SKIP 0U
```

```
#define RFAL_NFCV_ST_IC_MFG_CODE 0x02U
```

Enumerations

```
enum {
  RFAL_NFCV_REQ_FLAG_DEFAULT = 0x02U , RFAL_NFCV_REQ_FLAG_SUB_CARRIER = 0x01U ,
RFAL_NFCV_REQ_FLAG_DATA_RATE = 0x02U , RFAL_NFCV_REQ_FLAG_INVENTORY = 0x04U ,
  RFAL_NFCV_REQ_FLAG_PROTOCOL_EXT = 0x08U , RFAL_NFCV_REQ_FLAG_SELECT = 0x10U ,
RFAL_NFCV_REQ_FLAG_ADDRESS = 0x20U , RFAL_NFCV_REQ_FLAG_OPTION = 0x40U ,
  RFAL_NFCV_REQ_FLAG_RFU = 0x80U , RFAL_NFCV_REQ_FLAG_AFI = 0x10U ,
RFAL_NFCV_REQ_FLAG_NB_SLOTS = 0x20U
}
```

```
enum {
  RFAL_NFCV_RES_FLAG_ERROR = 0x01U , RFAL_NFCV_RES_FLAG_RFU1 = 0x02U ,
RFAL_NFCV_RES_FLAG_RFU2 = 0x04U , RFAL_NFCV_RES_FLAG_EXTENSION = 0x08U ,
  RFAL_NFCV_RES_FLAG_RFU3 = 0x10U , RFAL_NFCV_RES_FLAG_RFU4 = 0x20U ,
RFAL_NFCV_RES_FLAG_RFU5 = 0x40U , RFAL_NFCV_RES_FLAG_RFU6 = 0x80U
}
```

```
enum {
  RFAL_NFCV_ERROR_CMD_NOT_SUPPORTED = 0x01U , RFAL_NFCV_ERROR_CMD_NOT_RECOGNIZED =
0x02U , RFAL_NFCV_ERROR_OPTION_NOT_SUPPORTED = 0x03U , RFAL_NFCV_ERROR_UNKNOWN =
0x0FU ,
  RFAL_NFCV_ERROR_BLOCK_NOT_AVALIABLE = 0x10U , RFAL_NFCV_ERROR_BLOCK_ALREDY_LOCKED
= 0x11U , RFAL_NFCV_ERROR_BLOCK_LOCKED = 0x12U , RFAL_NFCV_ERROR_WRITE_FAILED = 0x13U ,
  RFAL_NFCV_ERROR_BLOCK_FAILED = 0x14U
}
```

```
enum {
  RFAL_NFCV_CMD_INVENTORY = 0x01U , RFAL_NFCV_CMD_SLPV = 0x02U ,
RFAL_NFCV_CMD_READ_SINGLE_BLOCK = 0x20U , RFAL_NFCV_CMD_WRITE_SINGLE_BLOCK = 0x21U ,
  RFAL_NFCV_CMD_LOCK_BLOCK = 0x22U , RFAL_NFCV_CMD_READ_MULTIPLE_BLOCKS = 0x23U ,
```

RFAL_NFCV_CMD_WRITE_MULTIPLE_BLOCKS = 0x24U , RFAL_NFCV_CMD_SELECT = 0x25U ,
RFAL_NFCV_CMD_RESET_TO_READY = 0x26U , RFAL_NFCV_CMD_GET_SYS_INFO = 0x2BU ,
RFAL_NFCV_CMD_EXTENDED_READ_SINGLE_BLOCK = 0x30U ,
RFAL_NFCV_CMD_EXTENDED_WRITE_SINGLE_BLOCK = 0x31U ,
RFAL_NFCV_CMD_EXTENDED_LOCK_SINGLE_BLOCK = 0x32U ,
RFAL_NFCV_CMD_EXTENDED_READ_MULTIPLE_BLOCK = 0x33U ,
RFAL_NFCV_CMD_EXTENDED_WRITE_MULTIPLE_BLOCK = 0x34U ,
RFAL_NFCV_CMD_EXTENDED_GET_SYS_INFO = 0x3BU
}

enum {
RFAL_NFCV_CMD_READ_CONFIGURATION = 0xA0U , RFAL_NFCV_CMD_WRITE_CONFIGURATION = 0xA1U , RFAL_NFCV_CMD_SET_EAS = 0xA2U , RFAL_NFCV_CMD_RESET_EAS = 0xA3U ,
RFAL_NFCV_CMD_LOCK_EAS = 0xA4U , RFAL_NFCV_CMD_ENABLE_EAS = 0xA5U ,
RFAL_NFCV_CMD_KILL = 0xA6U , RFAL_NFCV_CMD_WRITE_EAS_ID = 0xA7U ,
RFAL_NFCV_CMD_WRITE_EAS_CONFIG = 0xA8U , RFAL_NFCV_CMD_MANAGE_GPO = 0xA9U ,
RFAL_NFCV_CMD_WRITE_MESSAGE = 0xAAU , RFAL_NFCV_CMD_READ_MESSAGE_LENGTH = 0xABU ,
RFAL_NFCV_CMD_READ_MESSAGE = 0xACU , RFAL_NFCV_CMD_READ_DYN_CONFIGURATION = 0xADU ,
RFAL_NFCV_CMD_WRITE_DYN_CONFIGURATION = 0xAEU , RFAL_NFCV_CMD_WRITE_PASSWORD = 0xB1U ,
RFAL_NFCV_CMD_LOCK_KILL = 0xB2U , RFAL_NFCV_CMD_PRESENT_PASSWORD = 0xB3U ,
RFAL_NFCV_CMD_GET_RANDOM_NUMBER = 0xB4U , RFAL_NFCV_CMD_FAST_READ_SINGLE_BLOCK = 0xC0U ,
RFAL_NFCV_CMD_FAST_READ_MULTIPLE_BLOCKS = 0xC3U ,
RFAL_NFCV_CMD_FAST_EXTENDED_READ_SINGLE_BLOCK = 0xC4U ,
RFAL_NFCV_CMD_FAST_EXTENDED_READ_MULTIPLE_BLOCKS = 0xC5U ,
RFAL_NFCV_CMD_FAST_WRITE_MESSAGE = 0xCAU ,
RFAL_NFCV_CMD_FAST_READ_MESSAGE_LENGTH = 0xCBU , RFAL_NFCV_CMD_FAST_READ_MESSAGE = 0xCCU , RFAL_NFCV_CMD_FAST_READ_DYN_CONFIGURATION = 0xCDU ,
RFAL_NFCV_CMD_FAST_WRITE_DYN_CONFIGURATION = 0xCEU
}

enum {
RFAL_NFCV_SYSINFO_DFSID = 0x01U , RFAL_NFCV_SYSINFO_AFI = 0x02U ,
RFAL_NFCV_SYSINFO_MEMSIZE = 0x04U , RFAL_NFCV_SYSINFO_ICREF = 0x08U ,
RFAL_NFCV_SYSINFO_MOI = 0x10U , RFAL_NFCV_SYSINFO_CMDLIST = 0x20U ,
RFAL_NFCV_SYSINFO_CSI = 0x40U , RFAL_NFCV_SYSINFO_REQ_ALL = 0x7FU
}

enum **rfalNfcvNumSlots** { **RFAL_NFCV_NUM_SLOTS_1** = 0x20 , **RFAL_NFCV_NUM_SLOTS_16** = 0x00 }

Functions

ReturnCode **rfalNfcvPollerInitialize** (void)
Initialize NFC-V Poller mode. More...

ReturnCode **rfalNfcvPollerCheckPresence** (**rfalNfcvInventoryRes** *invRes)
NFC-V Poller Check Presence. More...

ReturnCode **rfalNfcvPollerInventory** (**rfalNfcvNumSlots** nSlots, uint8_t maskLen, const uint8_t *maskVal, **rfalNfcvInventoryRes** *invRes, uint16_t *rcvdLen)
NFC-V Poller Poll. More...

ReturnCode **rfalNfcvPollerCollisionResolution** (**rfalComplianceMode** compMode, uint8_t devLimit, **rfalNfcvListenDevice** *nfcvDevList, uint8_t *devCnt)
NFC-V Poller Full Collision Resolution. More...

ReturnCode **rfalNfcvPollerSleepCollisionResolution** (uint8_t devLimit, **rfalNfcvListenDevice** *nfcvDevList, uint8_t *devCnt)
NFC-V Poller Full Collision Resolution With Sleep. More...

ReturnCode **rfalNfcvPollerSleep** (uint8_t flags, const uint8_t *uid)
NFC-V Poller Sleep. More...

ReturnCode **rfalNfcvPollerSelect** (uint8_t flags, const uint8_t *uid)
NFC-V Poller Select. More...

ReturnCode **rfalNfcvPollerReadSingleBlock** (uint8_t flags, const uint8_t *uid, uint8_t blockNum, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen)
NFC-V Poller Read Single Block. More...

ReturnCode **rfalNfcvPollerWriteSingleBlock** (uint8_t flags, const uint8_t *uid, uint8_t blockNum, const uint8_t *wrData, uint8_t blockLen)
NFC-V Poller Write Single Block. More...

ReturnCode **rfalNfcvPollerReadMultipleBlocks** (uint8_t flags, const uint8_t *uid, uint8_t firstBlockNum, uint8_t numOfBlocks, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen)

NFC-V Poller Read Multiple Blocks. More...

ReturnCode **rfaINfcvPollerWriteMultipleBlocks** (uint8_t flags, const uint8_t *uid, uint8_t firstBlockNum, uint8_t numOfBlocks, uint8_t *txBuf, uint16_t txBufLen, uint8_t blockLen, const uint8_t *wrData, uint16_t wrDataLen)
NFC-V Poller Write Multiple Blocks. More...

ReturnCode **rfaINfcvPollerLockBlock** (uint8_t flags, const uint8_t *uid, uint8_t blockNum)
NFC-V Poller Lock Single Block. More...

ReturnCode **rfaINfcvPollerExtendedLockSingleBlock** (uint8_t flags, const uint8_t *uid, uint16_t blockNum)
NFC-V Poller Extended Lock Single Block. More...

ReturnCode **rfaINfcvPollerExtendedReadSingleBlock** (uint8_t flags, const uint8_t *uid, uint16_t blockNum, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen)
NFC-V Poller Extended Read Single Block. More...

ReturnCode **rfaINfcvPollerExtendedWriteSingleBlock** (uint8_t flags, const uint8_t *uid, uint16_t blockNum, const uint8_t *wrData, uint8_t blockLen)
NFC-V Poller Extended Write Single Block. More...

ReturnCode **rfaINfcvPollerExtendedReadMultipleBlocks** (uint8_t flags, const uint8_t *uid, uint16_t firstBlockNum, uint16_t numOfBlocks, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen)
NFC-V Poller Extended Read Multiple Blocks. More...

ReturnCode **rfaINfcvPollerExtendedWriteMultipleBlocks** (uint8_t flags, const uint8_t *uid, uint16_t firstBlockNum, uint16_t numOfBlocks, uint8_t *txBuf, uint16_t txBufLen, uint8_t blockLen, const uint8_t *wrData, uint16_t wrDataLen)
NFC-V Poller Extended Write Multiple Blocks. More...

ReturnCode **rfaINfcvPollerGetSystemInformation** (uint8_t flags, const uint8_t *uid, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen)
NFC-V Get System Information. More...

ReturnCode **rfaINfcvPollerExtendedGetSystemInformation** (uint8_t flags, const uint8_t *uid, uint8_t requestField, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen)
NFC-V Extended Get System Information. More...

ReturnCode **rfaINfcvPollerTransceiveReq** (uint8_t cmd, uint8_t flags, uint8_t param, const uint8_t *uid, const uint8_t *data, uint16_t dataLen, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen)

NFC-V Transceive Request. More...

Detailed Description

Macro Definition Documentation

◆ RFAL_NFCV_BLOCKNUM_EXTENDED_LEN

```
#define RFAL_NFCV_BLOCKNUM_EXTENDED_LEN   2U
```

Block Number length on extended commands: 16 bits

◆ RFAL_NFCV_BLOCKNUM_LEN

```
#define RFAL_NFCV_BLOCKNUM_LEN   1U
```

Block Number length on normal commands: 8 bits

◆ RFAL_NFCV_BNO_LEN

```
#define RFAL_NFCV_BNO_LEN   1U
```

NFC-V Block Number length

◆ RFAL_NFCV_CRC_LEN

```
#define RFAL_NFCV_CRC_LEN   2U
```

NFC-V CRC length

◆ RFAL_NFCV_MAX_BLOCK_LEN

```
#define RFAL_NFCV_MAX_BLOCK_LEN   32U
```

Max Block size: can be of up to 256 bits ISO 15693 2000 5

◆ RFAL_NFCV_MAX_GEN_DATA_LEN

#define RFAL_NFCV_MAX_GEN_DATA_LEN (**RFAL_NFCV_MAX_BLOCK_LEN** + **RFAL_NFCV_BNO_LEN** + **RFAL_NFCV_UID_LEN**)

Max data

◆ RFAL_NFCV_PARAM_SKIP

#define RFAL_NFCV_PARAM_SKIP 0U

Skip proprietary Param Request

◆ RFAL_NFCV_ST_IC_MFG_CODE

#define RFAL_NFCV_ST_IC_MFG_CODE 0x02U

ST IC Mfg code (used for custom commands)

◆ RFAL_NFCV_UID_LEN

#define RFAL_NFCV_UID_LEN 8U

NFC-V UID length

Enumeration Type Documentation

◆ anonymous enum

anonymous enum

NFC-V RequestFlags ISO15693 2000 7.3.1

Enumerator

RFAL_NFCV_REQ_FLAG_DEFAULT	Default Request Flags
RFAL_NFCV_REQ_FLAG_SUB_CARRIER	Sub Carrier flag
RFAL_NFCV_REQ_FLAG_DATA_RATE	Data Rate flag
RFAL_NFCV_REQ_FLAG_INVENTORY	Inventory flag
RFAL_NFCV_REQ_FLAG_PROTOCOL_EXT	Protocol Extension flag
RFAL_NFCV_REQ_FLAG_SELECT	Select flag
RFAL_NFCV_REQ_FLAG_ADDRESS	Address flag
RFAL_NFCV_REQ_FLAG_OPTION	Option flag
RFAL_NFCV_REQ_FLAG_RFU	RFU flag
RFAL_NFCV_REQ_FLAG_AFI	AFI flag
RFAL_NFCV_REQ_FLAG_NB_SLOTS	Number of Slots flag

◆ anonymous enum

anonymous enum

NFC-V Response Flags ISO15693 2000 7.4.1

Enumerator	
RFAL_NFCV_RES_FLAG_ERROR	Error flag
RFAL_NFCV_RES_FLAG_RFU1	RFU flag
RFAL_NFCV_RES_FLAG_RFU2	RFU flag
RFAL_NFCV_RES_FLAG_EXTENSION	Extension flag
RFAL_NFCV_RES_FLAG_RFU3	RFU flag
RFAL_NFCV_RES_FLAG_RFU4	RFU flag
RFAL_NFCV_RES_FLAG_RFU5	RFU flag
RFAL_NFCV_RES_FLAG_RFU6	RFU flag

◆ anonymous enum

anonymous enum

NFC-V Error code ISO15693 2000 7.4.2

Enumerator	
RFAL_NFCV_ERROR_CMD_NOT_SUPPORTED	The command is not supported, code is not recognised
RFAL_NFCV_ERROR_CMD_NOT_RECOGNIZED	The command is not recognised, format error occurred
RFAL_NFCV_ERROR_OPTION_NOT_SUPPORTED	The option is not supported
RFAL_NFCV_ERROR_UNKNOWN	Unknown error
RFAL_NFCV_ERROR_BLOCK_NOT_AVALIABLE	The specified block is not available
RFAL_NFCV_ERROR_BLOCK_ALREDY_LOCKED	The specified block is already locked
RFAL_NFCV_ERROR_BLOCK_LOCKED	The specified block is locked
RFAL_NFCV_ERROR_WRITE_FAILED	The specified block was not successfully programmed
RFAL_NFCV_ERROR_BLOCK_FAILED	The specified block was not successfully locked

◆ anonymous enum

anonymous enum

NFC-V command set ISO15693 2000 9.1

Enumerator	
RFAL_NFCV_CMD_INVENTORY	INVENTORY_REQ (Inventory) command
RFAL_NFCV_CMD_SLPV	SLPV_REQ (Stay quiet) command
RFAL_NFCV_CMD_READ_SINGLE_BLOCK	Read single block command
RFAL_NFCV_CMD_WRITE_SINGLE_BLOCK	Write single block command
RFAL_NFCV_CMD_LOCK_BLOCK	Lock block command
RFAL_NFCV_CMD_READ_MULTIPLE_BLOCKS	Read multiple blocks command
RFAL_NFCV_CMD_WRITE_MULTIPLE_BLOCKS	Write multiple blocks command
RFAL_NFCV_CMD_SELECT	Select command
RFAL_NFCV_CMD_RESET_TO_READY	Reset To Ready command
RFAL_NFCV_CMD_GET_SYS_INFO	Get System Information command
RFAL_NFCV_CMD_EXTENDED_READ_SINGLE_BLOCK	Extended read single block command

RFAL_NFCV_CMD_EXTENDED_WRITE_SINGLE_BLOCK	Extended write single block command
RFAL_NFCV_CMD_EXTENDED_LOCK_SINGLE_BLOCK	Extended lock single block command
RFAL_NFCV_CMD_EXTENDED_READ_MULTIPLE_BLOCK	Extended read multiple block command
RFAL_NFCV_CMD_EXTENDED_WRITE_MULTIPLE_BLOCK	Extended read multiple block command
RFAL_NFCV_CMD_EXTENDED_GET_SYS_INFO	Extended Get System Information command

◆ anonymous enum

anonymous enum

ST25TV/ST25DV command set

Enumerator	
RFAL_NFCV_CMD_READ_CONFIGURATION	Read configuration command
RFAL_NFCV_CMD_WRITE_CONFIGURATION	Write configuration command
RFAL_NFCV_CMD_SET_EAS	Set EAS command
RFAL_NFCV_CMD_RESET_EAS	Reset EAS command
RFAL_NFCV_CMD_LOCK_EAS	Lock EAS command
RFAL_NFCV_CMD_ENABLE_EAS	Enable EAS command
RFAL_NFCV_CMD_KILL	Kill command
RFAL_NFCV_CMD_WRITE_EAS_ID	Write EAS ID command
RFAL_NFCV_CMD_WRITE_EAS_CONFIG	Write EAS CONFIG command
RFAL_NFCV_CMD_MANAGE_GPO	Manage GPO command
RFAL_NFCV_CMD_WRITE_MESSAGE	Write Message command
RFAL_NFCV_CMD_READ_MESSAGE_LENGTH	Read Message Length command
RFAL_NFCV_CMD_READ_MESSAGE	Read Message command
RFAL_NFCV_CMD_READ_DYN_CONFIGURATION	Read Dynamic Configuration command
RFAL_NFCV_CMD_WRITE_DYN_CONFIGURATION	Write Dynamic Configuration command
RFAL_NFCV_CMD_WRITE_PASSWORD	Write Kill Password / Write Password command
RFAL_NFCV_CMD_LOCK_KILL	Lock Kill command

RFAL_NFCV_CMD_PRESENT_PASSWORD	Present Password command
RFAL_NFCV_CMD_GET_RANDOM_NUMBER	Get Random Number command
RFAL_NFCV_CMD_FAST_READ_SINGLE_BLOCK	Fast Read single block command
RFAL_NFCV_CMD_FAST_READ_MULTIPLE_BLOCKS	Fast Read multiple blocks command
RFAL_NFCV_CMD_FAST_EXTENDED_READ_SINGLE_BLOCK	Fast Extended Read single block command
RFAL_NFCV_CMD_FAST_EXTENDED_READ_MULTIPLE_BLOCKS	Fast Extended Read multiple blocks command
RFAL_NFCV_CMD_FAST_WRITE_MESSAGE	Fast Write Message
RFAL_NFCV_CMD_FAST_READ_MESSAGE_LENGTH	Fast Read Message Length
RFAL_NFCV_CMD_FAST_READ_MESSAGE	Fast Read Message
RFAL_NFCV_CMD_FAST_READ_DYN_CONFIGURATION	Fast Read Dynamic configuration
RFAL_NFCV_CMD_FAST_WRITE_DYN_CONFIGURATION	Fast Write Dynamic Configuration

◆ anonymous enum

anonymous enum

ISO 15693 Get System info parameter request field ISO15693 2018 Table 94

Enumerator	
RFAL_NFCV_SYSINFO_DFSID	Get System info DFSID flag
RFAL_NFCV_SYSINFO_AFI	Get System info AFI flag
RFAL_NFCV_SYSINFO_MEMSIZE	Get System info MEMSIZE flag
RFAL_NFCV_SYSINFO_ICREF	Get System info ICREF flag
RFAL_NFCV_SYSINFO_MOI	Get System info MOI flag
RFAL_NFCV_SYSINFO_CMDLIST	Get System info CMDLIST flag
RFAL_NFCV_SYSINFO_CSI	Get System info CSI flag
RFAL_NFCV_SYSINFO_REQ_ALL	Get System info request of all parameters

◆ rfalNfcvNumSlots

enum **rfalNfcvNumSlots**

NFC-V Number of slots Digital 2.0 9.6.1

Enumerator	
RFAL_NFCV_NUM_SLOTS_1	Number of slots: 1
RFAL_NFCV_NUM_SLOTS_16	Number of slots: 16

Function Documentation

◆ rfalNfcvPollerCheckPresence()

ReturnCode rfalNfcvPollerCheckPresence (**rfalNfcvInventoryRes** * invRes)

This method checks if a NFC-V Listen device (VICC) is present on the field by sending an Inventory (INVENTORY_REQ)

Parameters

[out] **invRes** : If received, the INVENTORY_RES. Please note that received invRes may contain errors and is not guaranteed to contain valid data.

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_TIMEOUT : Timeout error, no listener device detectedd

RFAL_ERR_NONE : No error, one or more device in the field

Examples

exampleRfalPoller.c.

◆ rfalNfcvPollerCollisionResolution()

```
ReturnCode rfalNfcvPollerCollisionResolution ( rfalComplianceMode      compMode,
                                               uint8_t                 devLimit,
                                               rfalNfcvListenDevice *  nfcvDevList,
                                               uint8_t *               devCnt
                                             )
```

Performs a full Collision resolution as defined in Activity 2.0 9.3.7 Once done, the devCnt will indicate how many (if any) devices have been identified and their details are contained on nfcvDevList

Parameters

- [in] **compMode** : compliance mode to be performed
- [in] **devLimit** : device limit value, and size nfcaDevList
- [out] **nfcvDevList** : NFC-V listener devices list
- [out] **devCnt** : Devices found counter

When compMode is set to ISO the function immediately goes to 16 slots improving chances to detect more than only one strong card.

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or mode not set

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_NONE : No error

Examples

exampleRfalPoller.c.

◆ rfalNfcvPollerExtendedGetSystemInformation()

```
ReturnCode rfalNfcvPollerExtendedGetSystemInformation ( uint8_t          flags,
                                                        const uint8_t *  uid,
                                                        uint8_t          requestField,
                                                        uint8_t *        rxBuf,
                                                        uint16_t         rxBufLen,
                                                        uint16_t *       rcvLen
                                                      )
```

Sends Extended Get System Information command

Parameters

- [in] **flags** : Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT
- [in] **uid** : UID of the device to be put to be read if UID is provided Addressed mode will be used

[in] **requestField** : Get System info parameter request field

[out] **rxBuf** : buffer to store response (also with RES_FLAGS)

[in] **rxBufLen** : length of rxBuf

[out] **rcvLen** : number of bytes received

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

◆ rfalNfcvPollerExtendedLockSingleBlock()

```
ReturnCode rfalNfcvPollerExtendedLockSingleBlock ( uint8_t          flags,
                                                   const uint8_t *  uid,
                                                   uint16_t         blockNum
                                                 )
```

Locks a Single Block from a device (VICC) supporting extended commands

Parameters

[in] **flags** : Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT

[in] **uid** : UID of the device if UID is provided Addressed mode will be used

[in] **blockNum** : Number of the block to be locked (16 bits)

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

◆ rfalNfcvPollerExtendedReadMultipleBlocks()

```
ReturnCode rfalNfcvPollerExtendedReadMultipleBlocks ( uint8_t          flags,
                                                      const uint8_t *   uid,
                                                      uint16_t          firstBlockNum,
                                                      uint16_t          numOfBlocks,
                                                      uint8_t *         rxBuf,
                                                      uint16_t          rxBufLen,
                                                      uint16_t *        rcvLen
                                                    )
```

Reads Multiple Blocks from a device (VICC) supporting extended commands

Parameters

[in]	**flags**	: Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT
[in]	**uid**	: UID of the device to be put to be read if UID is provided Addressed mode will be used
[in]	**firstBlockNum**	: first block to be read (16 bits)
[in]	**numOfBlocks**	: number of consecutive blocks to read (16 bits)
[out]	**rxBuf**	: buffer to store response (also with RES_FLAGS)
[in]	**rxBufLen**	: length of rxBuf
[out]	**rcvLen**	: number of bytes received

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

◆ rfalNfcvPollerExtendedReadSingleBlock()

```
ReturnCode rfalNfcvPollerExtendedReadSingleBlock ( uint8_t          flags,
                                                   const uint8_t *  uid,
                                                   uint16_t         blockNum,
                                                   uint8_t *        rxBuf,
                                                   uint16_t         rxBufLen,
                                                   uint16_t *       rcvLen
                                                 )
```

Reads a Single Block from a device (VICC) supporting extended commands

Parameters

[in]	**flags**	: Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT
[in]	**uid**	: UID of the device to be put to be read if UID is provided Addressed mode will be used
[in]	**blockNum**	: Number of the block to read (16 bits)
[out]	**rxBuf**	: buffer to store response (also with RES_FLAGS)
[in]	**rxBufLen**	: length of rxBuf
[out]	**rcvLen**	: number of bytes received

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

◆ rfalNfcvPollerExtendedWriteMultipleBlocks()

```
ReturnCode rfalNfcvPollerExtendedWriteMultipleBlocks ( uint8_t          flags,
                                                       const uint8_t *  uid,
                                                       uint16_t         firstBlockNum,
                                                       uint16_t         numOfBlocks,
                                                       uint8_t *        txBuf,
                                                       uint16_t         txBufLen,
                                                       uint8_t          blockLen,
                                                       const uint8_t *  wrData,
                                                       uint16_t         wrDataLen
                                                     )
```

Writes Multiple Blocks from a device (VICC) supporting extended commands In order to not limit the length of the Write multiple command, a buffer must be provided where the request will be composed and then sent.

Parameters

- [in] **flags** : Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT
- [in] **uid** : UID of the device to be put to be read if UID is provided Addressed mode will be used
- [in] **firstBlockNum** : first block to be write (16 bits)
- [in] **numOfBlocks** : number of consecutive blocks to write (16 bits)
- [in] **txBuf** : buffer where the request will be composed
- [in] **txBufLen** : length of txBuf
- [in] **blockLen** : number of bytes of a block
- [in] **wrData** : data to be written
- [in] **wrDataLen** : length of the data do be written. Must be aligned with number of blocks to write and the size of a block

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

◆ rfalNfcvPollerExtendedWriteSingleBlock()

```
ReturnCode rfalNfcvPollerExtendedWriteSingleBlock ( uint8_t          flags,
                                                    const uint8_t *  uid,
                                                    uint16_t         blockNum,
                                                    const uint8_t *  wrData,
                                                    uint8_t          blockLen
                                                  )
```

Writes a Single Block from a device (VICC) supporting extended commands

Parameters

- [in] **flags** : Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT
- [in] **uid** : UID of the device if UID is provided Addressed mode will be used
- [in] **blockNum** : Number of the block to write (16 bits)
- [in] **wrData** : data to be written on the given block
- [in] **blockLen** : number of bytes of a block

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

◆ rfalNfcvPollerGetSystemInformation()

```
ReturnCode rfalNfcvPollerGetSystemInformation ( uint8_t          flags,
                                                const uint8_t *  uid,
```

```
                                        uint8_t *       rxBuf,
                                        uint16_t        rxBufLen,
                                        uint16_t *      rcvLen
                                      )
```

Sends Get System Information command

Parameters

- [in] **flags** : Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT
- [in] **uid** : UID of the device to be put to be read if UID is provided Addressed mode will be used
- [out] **rxBuf** : buffer to store response (also with RES_FLAGS)
- [in] **rxBufLen** : length of rxBuf
- [out] **rcvLen** : number of bytes received

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

◆ rfalNfcvPollerInitialize()

ReturnCode rfalNfcvPollerInitialize (void)

This methods configures RFAL RF layer to perform as a NFC-V Poller/RW (ISO15693) including all default timings

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or mode not set

RFAL_ERR_PARAM : Incorrect bitrate

RFAL_ERR_NONE : No error

Examples

exampleRfalPoller.c.

◆ rfalNfcvPollerInventory()

```
ReturnCode rfalNfcvPollerInventory ( rfalNfcvNumSlots          nSlots,
                                     uint8_t                   maskLen,
                                     const uint8_t *           maskVal,
                                     rfalNfcvInventoryRes *    invRes,
                                     uint16_t *                rcvdLen
                                   )
```

This function sends to all VICCs in field the INVENTORY command with the given number of slots

If more than one slot is used the following EOF need to be handled by the caller using **rfalISO15693TransceiveEOFAnticollision()**

Parameters

- [in] **nSlots** : Number of Slots to be sent (1 or 16)
- [in] **maskLen** : Number bits on the Mask value
- [in] **maskVal** : location of the Mask value
- [out] **invRes** : location to place the INVENTORY_RES
- [out] **rcvdLen** : number of bits received (without collision)

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_RF_COLLISION : Collision detected

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_NONE : No error

◆ rfalNfcvPollerLockBlock()

```
ReturnCode rfalNfcvPollerLockBlock ( uint8_t          flags,
```

```
                                    const uint8_t *  uid,
                                    uint8_t          blockNum
                                  )
```

Locks a Single Block from a device (VICC) supporting extended commands

Parameters

- [in] **flags** : Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT
- [in] **uid** : UID of the device if UID is provided Addressed mode will be used
- [in] **blockNum** : Number of the block to be locked

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

◆ rfalNfcvPollerReadMultipleBlocks()

```
ReturnCode rfalNfcvPollerReadMultipleBlocks ( uint8_t          flags,
                                              const uint8_t *  uid,
                                              uint8_t          firstBlockNum,
                                              uint8_t          numOfBlocks,
                                              uint8_t *        rxBuf,
                                              uint16_t         rxBufLen,
                                              uint16_t *       rcvLen
                                            )
```

Reads Multiple Blocks from a device (VICC)

Parameters

[in] **flags** : Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT

[in] **uid** : UID of the device to be put to be read if UID is provided Addressed mode will be used

[in] **firstBlockNum** : first block to be read

[in] **numOfBlocks** : number of block to read

[out] **rxBuf** : buffer to store response (also with RES_FLAGS)

[in] **rxBufLen** : length of rxBuf

[out] **rcvLen** : number of bytes received

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

◆ rfalNfcvPollerReadSingleBlock()

```
ReturnCode rfalNfcvPollerReadSingleBlock ( uint8_t          flags,
                                           const uint8_t *  uid,
                                           uint8_t          blockNum,
                                           uint8_t *        rxBuf,
                                           uint16_t         rxBufLen,
                                           uint16_t *       rcvLen
                                         )
```

Reads a Single Block from a device (VICC)

Parameters

[in] **flags** : Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT

[in] **uid** : UID of the device to be put to be read if UID is provided Addressed mode will be used

[in] **blockNum** : Number of the block to read

[out] **rxBuf** : buffer to store response (also with RES_FLAGS)

[in] **rxBufLen** : length of rxBuf

[out] **rcvLen** : number of bytes received

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

◆ rfalNfcvPollerSelect()

```
ReturnCode rfalNfcvPollerSelect ( uint8_t          flags,
                                  const uint8_t *  uid
                                )
```

Selects a device (VICC) by its UID

Parameters

[in] **flags** : Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT

[in] **uid** : UID of the device to be put to be Selected

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

◆ rfalNfcvPollerSleep()

```
ReturnCode rfalNfcvPollerSleep ( uint8_t          flags,
                                 const uint8_t *  uid
                               )
```

This function is used to send the SLPV_REQ (Stay Quiet) command to put the VICC with the given UID to state QUIET so that they do not reply to more Inventory

Parameters

[in] **flags** : Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT

[in] **uid** : UID of the device to be put to Sleep

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_NONE : No error

◆ rfalNfcvPollerSleepCollisionResolution()

```
ReturnCode rfalNfcvPollerSleepCollisionResolution ( uint8_t                  devLimit,
                                                    rfalNfcvListenDevice *   nfcvDevList,
                                                    uint8_t *                devCnt
                                                  )
```

Performs a full Collision resolution which is different from Activity 2.0 9.3.7. The implementation uses SLPV (StayQuiet) command to make sure all cards are found. Once done, the devCnt will indicate how many (if any) devices have been identified and their details are contained on nfcvDevList

Parameters

[in] **devLimit** : device limit value, and size nfcaDevList

[out] **nfcvDevList** : NFC-V listener devices list

[out] **devCnt** : Devices found counter

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or mode not set

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_NONE : No error

◆ rfalNfcvPollerTransceiveReq()

```
ReturnCode rfalNfcvPollerTransceiveReq ( uint8_t          cmd,
                                         uint8_t          flags,
                                         uint8_t          param,
                                         const uint8_t *  uid,
                                         const uint8_t *  data,
                                         uint16_t         dataLen,
                                         uint8_t *        rxBuf,
                                         uint16_t         rxBufLen,
                                         uint16_t *       rcvLen
                                       )
```

Performs a generic transceive with an ISO15693 tag

Parameters

[in]	**cmd**	: NFC-V command
[in]	**flags**	: Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT
[in]	**param**	: Prepend parameter on certain proprietary requests For default commands skip: RFAL_NFCV_PARAM_SKIP
[in]	**uid**	: UID of the device to be put to be read if UID is provided Addressed mode will be used
[in]	**data**	: command parameters append after UID
[in]	**dataLen**	: command parameters Len
[out]	**rxBuf**	: buffer to store response (also with RES_FLAGS)
[in]	**rxBufLen**	: length of rxBuf
[out]	**rcvLen**	: number of bytes received

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

◆ rfalNfcvPollerWriteMultipleBlocks()

```
ReturnCode rfalNfcvPollerWriteMultipleBlocks ( uint8_t          flags,
                                               const uint8_t *  uid,
                                               uint8_t          firstBlockNum,
                                               uint8_t          numOfBlocks,
                                               uint8_t *        txBuf,
                                               uint16_t         txBufLen,
                                               uint8_t          blockLen,
                                               const uint8_t *  wrData,
                                               uint16_t         wrDataLen
                                             )
```

Writes Multiple Blocks from a device (VICC) In order to not limit the length of the Write multiple command, a buffer must be provided where the request will be composed and then sent.

Parameters

[in]	**flags**	: Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT
[in]	**uid**	: UID of the device to be put to be read if UID is provided Addressed mode will be used
[in]	**firstBlockNum**	: first block to be written
[in]	**numOfBlocks**	: number of consecutive blocks to write
[in]	**txBuf**	: buffer where the request will be composed
[in]	**txBufLen**	: length of txBuf
[in]	**blockLen**	: number of bytes of a block
[in]	**wrData**	: data to be written

[in] **wrDataLen** : length of the data do be written. Must be aligned with number of blocks to write and the size of a block

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

◆ rfalNfcvPollerWriteSingleBlock()

```
ReturnCode rfalNfcvPollerWriteSingleBlock ( uint8_t          flags,
                                            const uint8_t *  uid,
                                            uint8_t          blockNum,
                                            const uint8_t *  wrData,
                                            uint8_t          blockLen
                                          )
```

Writes a Single Block from a device (VICC)

Parameters

[in] **flags** : Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT

[in] **uid** : UID of the device to be put to be written if UID is provided Addressed mode will be used

[in] **blockNum** : Number of the block to write

[in] **wrData** : data to be written on the given block

[in] **blockLen** : number of bytes of a block

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

rfalNfcvInventoryRes Struct Reference


```
#include <rfal_nfcv.h>
```

Data Fields

uint8_t **RES_FLAG**

uint8_t **DSFID**

uint8_t	**UID** [**RFAL_NFCV_UID_LEN**]
uint8_t	**crc** [**RFAL_CRC_LEN**]

Detailed Description

NFC-V INVENTORY_RES format Digital 2.0 9.6.2

Examples

exampleRfalPoller.c.

Field Documentation

◆ crc

uint8_t rfalNfcvInventoryRes::crc[**RFAL_CRC_LEN**]

CRC

◆ DSFID

uint8_t rfalNfcvInventoryRes::DSFID

Data Storage Format Identifier

◆ RES_FLAG

uint8_t rfalNfcvInventoryRes::RES_FLAG

Response Flags

◆ UID

uint8_t rfalNfcvInventoryRes::UID[**RFAL_NFCV_UID_LEN**]

NFC-V device UID

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

- **rfal_nfcv.h**

Generated on Fri Mar 14 2025 11:45:10 for RFAL by doxygen 1.9.5

RFAL

rfalNfcvGenericReq Struct Reference


```
#include <rfal_nfcv.h>
```

Data Fields

uint8_t	**REQ_FLAG**
uint8_t	**CMD**

union {

uint8_t	**UID** [**RFAL_NFCV_UID_LEN**]
uint8_t	**data** [**RFAL_NFCV_MAX_GEN_DATA_LEN**]
}	**payload**

Detailed Description

NFC-V Generic Req format

Field Documentation

◆ CMD

uint8_t rfalNfcvGenericReq::CMD

Command code

◆ data

uint8_t rfalNfcvGenericReq::data[**RFAL_NFCV_MAX_GEN_DATA_LEN**]

Data

◆

union { ... } rfalNfcvGenericReq::payload

Payload

◆ REQ_FLAG

uint8_t rfalNfcvGenericReq::REQ_FLAG

Request Flags

◆ UID

uint8_t rfalNfcvGenericReq::UID[**RFAL_NFCV_UID_LEN**]

Mask Value

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

- **rfal_nfcv.h**

Generated on Fri Mar 14 2025 11:45:10 for RFAL by doxygen 1.9.5

RFAL

rfalNfcvGenericRes Struct Reference


```
#include <rfal_nfcv.h>
```

Data Fields

uint8_t	**RES_FLAG**
uint8_t	**data** [**RFAL_NFCV_MAX_GEN_DATA_LEN**]

Detailed Description

NFC-V Generic Response format

Field Documentation

◆ data

uint8_t rfalNfcvGenericRes::data[**RFAL_NFCV_MAX_GEN_DATA_LEN**]

Data

◆ RES_FLAG

uint8_t rfalNfcvGenericRes::RES_FLAG

Response Flags

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

- **rfal_nfcv.h**

rfalNfcvListenDevice Struct Reference


```
#include <rfal_nfcv.h>
```

Collaboration diagram for rfalNfcvListenDevice:

[legend]

Data Fields

rfalNfcvInventoryRes	**InvRes**
bool	**isSleep**

Detailed Description

NFC-V listener device (VICC) struct

Examples

exampleRfalPoller.c.

Field Documentation

◆ InvRes

rfalNfcvInventoryRes rfalNfcvListenDevice::InvRes

INVENTORY_RES

◆ isSleep

bool rfalNfcvListenDevice::isSleep

Device sleeping flag

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

- **rfal_nfcv.h**

RFAL

ST25TB

RFAL » **RFAL-AL**

RFAL ST25TB Module. More...

Collaboration diagram for ST25TB:

Data Structures

struct **rfalSt25tbListenDevice**

Macros

#define **RFAL_ST25TB_CHIP_ID_LEN** 1U

#define **RFAL_ST25TB_CRC_LEN** 2U

#define **RFAL_ST25TB_UID_LEN** 8U

#define **RFAL_ST25TB_BLOCK_LEN** 4U

Typedefs

typedef uint8_t **rfalSt25tbUID**[**RFAL_ST25TB_UID_LEN**]

typedef uint8_t **rfalSt25tbBlock**[**RFAL_ST25TB_BLOCK_LEN**]

Functions

ReturnCode **rfalSt25tbPollerInitialize** (void)
Initialize ST25TB Poller mode. More...

ReturnCode **rfalSt25tbPollerCheckPresence** (uint8_t *chipId)
ST25TB Poller Check Presence. More...

ReturnCode **rfalSt25tbPollerCollisionResolution** (uint8_t devLimit, **rfalSt25tbListenDevice** *st25tbDevList, uint8_t *devCnt)
ST25TB Poller Collision Resolution. More...

ReturnCode **rfalSt25tbPollerInitiate** (uint8_t *chipId)
ST25TB Poller Initiate. More...

ReturnCode	**rfalSt25tbPollerPcall** (uint8_t *chipId) ST25TB Poller Pcall. More...
ReturnCode	**rfalSt25tbPollerSlotMarker** (uint8_t slotNum, uint8_t *chipIdRes) ST25TB Poller Slot Marker. More...
ReturnCode	**rfalSt25tbPollerSelect** (uint8_t chipId) ST25TB Poller Select. More...
ReturnCode	**rfalSt25tbPollerGetUID** (**rfalSt25tbUID** *UID) ST25TB Get UID. More...
ReturnCode	**rfalSt25tbPollerReadBlock** (uint8_t blockAddress, **rfalSt25tbBlock** *blockData) ST25TB Poller Read Block. More...
ReturnCode	**rfalSt25tbPollerWriteBlock** (uint8_t blockAddress, const **rfalSt25tbBlock** *blockData) ST25TB Poller Write Block. More...
ReturnCode	**rfalSt25tbPollerCompletion** (void) ST25TB Poller Completion. More...
ReturnCode	**rfalSt25tbPollerResetToInventory** (void) ST25TB Poller Reset to Inventory. More...

Detailed Description

Macro Definition Documentation

◆ RFAL_ST25TB_BLOCK_LEN

```
#define RFAL_ST25TB_BLOCK_LEN   4U
```

ST25TB Data Block length

◆ RFAL_ST25TB_CHIP_ID_LEN

```
#define RFAL_ST25TB_CHIP_ID_LEN   1U
```

ST25TB chip ID length

◆ RFAL_ST25TB_CRC_LEN

```
#define RFAL_ST25TB_CRC_LEN   2U
```

ST25TB CRC length

◆ RFAL_ST25TB_UID_LEN

```
#define RFAL_ST25TB_UID_LEN   8U
```

ST25TB Unique ID length

Typedef Documentation

◆ rfalSt25tbBlock

typedef uint8_t rfalSt25tbBlock[**RFAL_ST25TB_BLOCK_LEN**]

ST25TB Block type

◆ rfalSt25tbUID

typedef uint8_t rfalSt25tbUID[**RFAL_ST25TB_UID_LEN**]

ST25TB UID type

Function Documentation

◆ rfalSt25tbPollerCheckPresence()

ReturnCode rfalSt25tbPollerCheckPresence (uint8_t * chipId)

This method checks if a ST25TB Listen device (PICC) is present on the field by sending an Initiate command

Parameters

[out] **chipId** : if successfully retrieved, the device's chip ID

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_TIMEOUT : Timeout error, no listener device detected

RFAL_ERR_RF_COLLISION : Collision detected one or more device in the field

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_NONE : No error

◆ rfalSt25tbPollerCollisionResolution()

```
ReturnCode rfalSt25tbPollerCollisionResolution ( uint8_t                    devLimit,
                                                 rfalSt25tbListenDevice *   st25tbDevList,
                                                 uint8_t *                  devCnt
                                               )
```

This method performs ST25TB Collision resolution, selects the each device, retrieves its UID and then deselects. In case only one device is identified the ST25TB device is left in select state.

Parameters

[in] **devLimit** : device limit value, and size st25tbDevList

[out] **st25tbDevList** : ST35TB listener device info

[out] **devCnt** : Devices found counter

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_TIMEOUT : Timeout error, no listener device detected

RFAL_ERR_RF_COLLISION : Collision detected one or more device in the field

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_NONE : No error

◆ rfalSt25tbPollerCompletion()

ReturnCode rfalSt25tbPollerCompletion (void)

This method sends a completion command to the ST25TB. After the completion the card no longer will reply to any command.

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_TIMEOUT : Timeout error, no listener device detected

RFAL_ERR_PROTO : Protocol error detected, invalid SENSB_RES received

RFAL_ERR_NONE : No error

◆ rfalSt25tbPollerGetUID()

ReturnCode rfalSt25tbPollerGetUID (**rfalSt25tbUID** * UID)

This method sends a Get_UID command

If a single device responds the chip UID will be retrieved

Parameters

`[out]` **UID** : UID of the found device

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_TIMEOUT : Timeout error, no listener device detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_NONE : No error

◆ rfalSt25tbPollerInitialize()

ReturnCode rfalSt25tbPollerInitialize (void)

This methods configures RFAL RF layer to perform as a ST25TB Poller/RW including all default timings

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or mode not set

RFAL_ERR_NONE : No error

◆ rfalSt25tbPollerInitiate()

ReturnCode rfalSt25tbPollerInitiate (uint8_t * chipId)

This method sends an Initiate command

If a single device responds the chip ID will be retrieved

Parameters

[out] **chipId** : chip ID of the device

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_TIMEOUT : Timeout error, no listener device detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_NONE : No error

◆ rfalSt25tbPollerPcall()

ReturnCode rfalSt25tbPollerPcall (uint8_t * chipId)

This method sends a Pcall command If successful the device's chip ID will be retrieved

Parameters

[out] **chipId** : Chip ID of the device

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_TIMEOUT : Timeout error, no listener device detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_NONE : No error

◆ rfalSt25tbPollerReadBlock()

```
ReturnCode rfalSt25tbPollerReadBlock ( uint8_t            blockAddress,
                                       rfalSt25tbBlock *  blockData
                                     )
```

This method reads a block of the ST25TB

Parameters

[in] **blockAddress** : address of the block to be read

[out] **blockData** : location to place the data read from block

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_TIMEOUT : Timeout error, no listener device detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_NONE : No error

◆ rfalSt25tbPollerResetToInventory()

```
ReturnCode rfalSt25tbPollerResetToInventory ( void )
```

This method sends a Reset to Inventory command to the ST25TB.

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_TIMEOUT : Timeout error, no listener device detected

RFAL_ERR_PROTO : Protocol error detected, invalid SENSB_RES received

RFAL_ERR_NONE : No error

◆ rfalSt25tbPollerSelect()

ReturnCode rfalSt25tbPollerSelect (uint8_t chipId)

This method sends a ST25TB Select command with the given chip ID.

If the device is already in Selected state and receives an incorrect chip ID, it goes into Deselected state

Parameters

[in] **chipId** : chip ID of the device to be selected

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_TIMEOUT : Timeout error, no listener device detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_NONE : No error

◆ rfalSt25tbPollerSlotMarker()

ReturnCode rfalSt25tbPollerSlotMarker (uint8_t slotNum,
uint8_t * chipIdRes
)

This method sends a Slot Marker

If a single device responds the chip ID will be retrieved

Parameters

[in] **slotNum** : Slot Number

[out] **chipIdRes** : Chip ID of the device

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_TIMEOUT : Timeout error, no listener device detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_NONE : No error

◆ rfalSt25tbPollerWriteBlock()

```
ReturnCode rfalSt25tbPollerWriteBlock ( uint8_t                    blockAddress,
                                        const rfalSt25tbBlock *    blockData
                                      )
```

This method writes a block of the ST25TB

Parameters

[in] **blockAddress** : address of the block to be written

[in] **blockData** : data to be written on the block

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_TIMEOUT : Timeout error, no listener device detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_NONE : No error

rfalSt25tbListenDevice Struct Reference


```
#include <rfal_st25tb.h>
```

Data Fields

uint8_t	**chipID**
rfalSt25tbUID	**UID**
bool	**isDeselected**

Detailed Description

ST25TB listener device (PICC) struct

Field Documentation

◆ chipID

uint8_t rfalSt25tbListenDevice::chipID

Device's session Chip ID

◆ isDeselected

bool rfalSt25tbListenDevice::isDeselected

Device deselect flag

◆ UID

rfalSt25tbUID rfalSt25tbListenDevice::UID

Device's UID

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

- **rfal_st25tb.h**

Generated on Fri Mar 14 2025 11:45:10 for RFAL by doxygen 1.9.5

RFAL

ST25xV

RFAL » **RFAL-AL**

RFAL ST25xV Module. More...

Collaboration diagram for ST25xV:

Macros

#define **RFAL_NFCV_BLOCKNUM_M24LR_LEN** 2U

Functions

ReturnCode **rfalST25xVPollerM24LRReadSingleBlock** (uint8_t flags, const uint8_t *uid, uint16_t blockNum, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen)
NFC-V Poller Read Single Block (M24LR) More...

ReturnCode **rfalST25xVPollerM24LRFastReadSingleBlock** (uint8_t flags, const uint8_t *uid, uint16_t blockNum, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen)
NFC-V Poller Fast Read Single Block (M24LR) More...

ReturnCode **rfalST25xVPollerM24LRWriteSingleBlock** (uint8_t flags, const uint8_t *uid, uint16_t blockNum, const uint8_t *wrData, uint8_t blockLen)
NFC-V Poller Write Single Block (M24LR) More...

ReturnCode **rfalST25xVPollerM24LRReadMultipleBlocks** (uint8_t flags, const uint8_t *uid, uint16_t firstBlockNum, uint8_t numOfBlocks, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen)
NFC-V Poller Read Multiple Blocks (M24LR) More...

ReturnCode **rfalST25xVPollerM24LRFastReadMultipleBlocks** (uint8_t flags, const uint8_t *uid, uint16_t firstBlockNum, uint8_t numOfBlocks, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen)
NFC-V Poller Fast Read Multiple Blocks (M24LR) More...

ReturnCode **rfalST25xVPollerFastReadSingleBlock** (uint8_t flags, const uint8_t *uid, uint8_t blockNum, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen)
NFC-V Poller Fast Read Single Block. More...

ReturnCode **rfalST25xVPollerFastReadMultipleBlocks** (uint8_t flags, const uint8_t *uid, uint8_t firstBlockNum, uint8_t numOfBlocks, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen)
NFC-V Poller Fast Read Multiple Blocks. More...

ReturnCode **rfalST25xVPollerFastExtendedReadSingleBlock** (uint8_t flags, const uint8_t *uid, uint16_t blockNum, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen)

NFC-V Poller Fast Extended Read Single Block. More...

ReturnCode **rfalST25xVPollerFastExtReadMultipleBlocks** (uint8_t flags, const uint8_t *uid, uint16_t firstBlockNum, uint16_t numOfBlocks, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen)
NFC-V Poller Fast Extended Read Multiple Blocks. More...

ReturnCode **rfalST25xVPollerReadConfiguration** (uint8_t flags, const uint8_t *uid, uint8_t pointer, uint8_t *regValue)
NFC-V Poller Read Configuration. More...

ReturnCode **rfalST25xVPollerWriteConfiguration** (uint8_t flags, const uint8_t *uid, uint8_t pointer, uint8_t regValue)
NFC-V Poller Write Configuration. More...

ReturnCode **rfalST25xVPollerReadDynamicConfiguration** (uint8_t flags, const uint8_t *uid, uint8_t pointer, uint8_t *regValue)
NFC-V Poller Read Dynamic Configuration. More...

ReturnCode **rfalST25xVPollerWriteDynamicConfiguration** (uint8_t flags, const uint8_t *uid, uint8_t pointer, uint8_t regValue)
NFC-V Poller Write Dynamic Configuration. More...

ReturnCode **rfalST25xVPollerFastReadDynamicConfiguration** (uint8_t flags, const uint8_t *uid, uint8_t pointer, uint8_t *regValue)
NFC-V Poller Fast Read Dynamic Configuration. More...

ReturnCode **rfalST25xVPollerFastWriteDynamicConfiguration** (uint8_t flags, const uint8_t *uid, uint8_t pointer, uint8_t regValue)
NFC-V Poller Fast Write Dynamic Configuration. More...

ReturnCode **rfalST25xVPollerPresentPassword** (uint8_t flags, const uint8_t *uid, uint8_t pwdNum, const uint8_t *pwd, uint8_t pwdLen)
NFC-V Poller Present Password. More...

ReturnCode **rfalST25xVPollerWritePassword** (uint8_t flags, const uint8_t *uid, uint8_t pwdNum, const uint8_t *pwd, uint8_t pwdLen)
NFC-V Poller Write Password. More...

ReturnCode **rfalST25xVPollerGetRandomNumber** (uint8_t flags, const uint8_t *uid, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen)
NFC-V Poller Get Random Number. More...

ReturnCode	**rfalST25xVPollerReadMessageLength** (uint8_t flags, const uint8_t *uid, uint8_t *msgLen) NFC-V Poller Read Message length. More...
ReturnCode	**rfalST25xVPollerFastReadMsgLength** (uint8_t flags, const uint8_t *uid, uint8_t *msgLen) NFC-V Poller Fast Read Message length. More...
ReturnCode	**rfalST25xVPollerReadMessage** (uint8_t flags, const uint8_t *uid, uint8_t mbPointer, uint8_t numBytes, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen) NFC-V Poller Read Message. More...
ReturnCode	**rfalST25xVPollerFastReadMessage** (uint8_t flags, const uint8_t *uid, uint8_t mbPointer, uint8_t numBytes, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen) NFC-V Poller Fast Read Message. More...
ReturnCode	**rfalST25xVPollerWriteMessage** (uint8_t flags, const uint8_t *uid, uint8_t msgLen, const uint8_t *msgData, uint8_t *txBuf, uint16_t txBufLen) NFC-V Poller Write Message. More...
ReturnCode	**rfalST25xVPollerFastWriteMessage** (uint8_t flags, const uint8_t *uid, uint8_t msgLen, const uint8_t *msgData, uint8_t *txBuf, uint16_t txBufLen) NFC-V Poller Fast Write Message. More...
ReturnCode	**rfalST25TV512CPollerReadConfiguration** (uint8_t flags, const uint8_t *uid, uint8_t fid, uint8_t pid, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen) NFC-V Poller Read Configuration (ST25TV02KC ST25TV512C version) More...
ReturnCode	**rfalST25TV512CPollerWriteConfiguration** (uint8_t flags, const uint8_t *uid, uint8_t fid, uint8_t pid, const uint8_t *data, uint8_t dataLen) NFC-V Poller Write Configuration (ST25TV02KC ST25TV512C version) More...

Detailed Description

Macro Definition Documentation

◆ RFAL_NFCV_BLOCKNUM_M24LR_LEN

```
#define RFAL_NFCV_BLOCKNUM_M24LR_LEN   2U
```

Block Number length of MR24LR tags: 16 bits

Function Documentation

◆ rfalST25TV512CPollerReadConfiguration()

```
ReturnCode rfalST25TV512CPollerReadConfiguration ( uint8_t          flags,
                                                   const uint8_t *  uid,
                                                   uint8_t          fid,
                                                   uint8_t          pid,
                                                   uint8_t *        rxBuf,
                                                   uint16_t         rxBufLen,
                                                   uint16_t *       rcvLen
                                                 )
```

Reads static configuration registers at the Pointer address

Parameters

- [in] **flags** : Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT
- [in] **uid** : UID of the device to be put to be read if not provided Select mode will be used
- [in] **fid** : FID
- [in] **pid** : PID
- [out] **rxBuf** : buffer to store response (also with RES_FLAGS)
- [in] **rxBufLen** : length of rxBuf
- [out] **rcvLen** : number of bytes received

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

◆ rfalST25TV512CPollerWriteConfiguration()

```
ReturnCode rfalST25TV512CPollerWriteConfiguration ( uint8_t          flags,
                                                    const uint8_t *  uid,
                                                    uint8_t          fid,
                                                    uint8_t          pid,
                                                    const uint8_t *  data,
                                                    uint8_t          dataLen
                                                  )
```

Writes static configuration registers at the Pointer address

Parameters

[in] **flags** : Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT

[in] **uid** : UID of the device to be put to be read if not provided Select mode will be used

[in] **fid** : FID

[in] **pid** : PID

[in] **data** : Register value

[in] **dataLen** : size of register value

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

◆ rfalST25xVPollerFastExtendedReadSingleBlock()

```
ReturnCode rfalST25xVPollerFastExtendedReadSingleBlock ( uint8_t          flags,
                                                         const uint8_t *  uid,
                                                         uint16_t         blockNum,
                                                         uint8_t *        rxBuf,
                                                         uint16_t         rxBufLen,
                                                         uint16_t *       rcvLen
                                                       )
```

Reads a Single Block from a device (VICC) supporting extended commands using ST Fast mode

Parameters

[in]	**flags**	: Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT
[in]	**uid**	: UID of the device to be put to be read if not provided Select mode will be used
[in]	**blockNum**	: Number of the block to read (16 bits)
[out]	**rxBuf**	: buffer to store response (also with RES_FLAGS)
[in]	**rxBufLen**	: length of rxBuf
[out]	**rcvLen**	: number of bytes received

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

◆ rfalST25xVPollerFastExtReadMultipleBlocks()

```
ReturnCode rfalST25xVPollerFastExtReadMultipleBlocks ( uint8_t          flags,
                                                       const uint8_t *  uid,
                                                       uint16_t         firstBlockNum,
                                                       uint16_t         numOfBlocks,
                                                       uint8_t *        rxBuf,
                                                       uint16_t         rxBufLen,
                                                       uint16_t *       rcvLen
                                                     )
```

Reads Multiple Blocks from a device (VICC) supporting extended commands using ST Fast mode

Parameters

[in]	**flags**	: Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT
[in]	**uid**	: UID of the device to be put to be read if not provided Select mode will be used
[in]	**firstBlockNum**	: first block to be read (16 bits)
[in]	**numOfBlocks**	: number of consecutive blocks to read (16 bits)
[out]	**rxBuf**	: buffer to store response (also with RES_FLAGS)
[in]	**rxBufLen**	: length of rxBuf
[out]	**rcvLen**	: number of bytes received

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

◆ rfalST25xVPollerFastReadDynamicConfiguration()

```
ReturnCode rfalST25xVPollerFastReadDynamicConfiguration ( uint8_t          flags,
                                                          const uint8_t *  uid,
```

```
                                                    uint8_t        pointer,
                                                    uint8_t *      regValue
                                                  )
```

Reads dynamic registers at the Pointer address using ST Fast mode

Parameters

- [in] **flags** : Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT
- [in] **uid** : UID of the device to be put to be read if not provided Select mode will be used
- [in] **pointer** : Pointer address
- [out] **regValue** : Register value

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

◆ rfalST25xVPollerFastReadMessage()

```
ReturnCode rfalST25xVPollerFastReadMessage ( uint8_t          flags,
                                             const uint8_t *  uid,
                                             uint8_t          mbPointer,
                                             uint8_t          numBytes,
                                             uint8_t *        rxBuf,
                                             uint16_t         rxBufLen,
                                             uint16_t *       rcvLen
                                           )
```

Reads up to 256 bytes in the Mailbox from the location specified by MBpointer and sends back their value in the rxBuf response using ST Fast mode. First MailBox location is '00'. When Number of bytes is set to 00h and MBPointer is equals to 00h, the MB_LEN bytes of the full message are returned. Otherwise, Read Message command returns (Number of Bytes + 1) bytes (i.e. 01h returns 2 bytes, FFh returns 256 bytes). An error is reported if (Pointer + Nb of bytes + 1) is greater than the message length. RF Reading of the last byte of the mailbox message automatically clears b1 of MB_CTRL_Dyn HOST_PUT_MSG, and allows RF to put a new message.

Parameters

[in]	**flags**	: Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT
[in]	**uid**	: UID of the device to be put to be read if not provided Select mode will be used
[in]	**mbPointer**	: MPpointer
[in]	**numBytes**	: number of bytes
[out]	**rxBuf**	: buffer to store response (also with RES_FLAGS)
[in]	**rxBufLen**	: length of rxBuf
[out]	**rcvLen**	: number of bytes received

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

◆ rfalST25xVPollerFastReadMsgLength()

```
ReturnCode rfalST25xVPollerFastReadMsgLength ( uint8_t          flags,
                                               const uint8_t *  uid,
                                               uint8_t *        msgLen
                                             )
```

Sends a Fast Read Message Length message to retrieve the value of MB_LEN_Dyn using ST Fast mode.

Parameters

[in] **flags** : Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT

[in] **uid** : UID of the device to be put to be read if not provided Select mode will be used

[out] **msgLen** : Message Length

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

◆ rfalST25xVPollerFastReadMultipleBlocks()

```
ReturnCode rfalST25xVPollerFastReadMultipleBlocks ( uint8_t          flags,
                                                    const uint8_t *  uid,
                                                    uint8_t          firstBlockNum,
                                                    uint8_t          numOfBlocks,
                                                    uint8_t *        rxBuf,
                                                    uint16_t         rxBufLen,
                                                    uint16_t *       rcvLen
                                                  )
```

Reads Multiple Blocks from a device (VICC) using ST Fast mode

Parameters

[in] **flags** : Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT

[in] **uid** : UID of the device to be put to be read if not provided Select mode will be used

[in] **firstBlockNum** : first block to be read

[in] **numOfBlocks** : number of block to read

[out] **rxBuf** : buffer to store response (also with RES_FLAGS)

[in] **rxBufLen** : length of rxBuf

[out] **rcvLen** : number of bytes received

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

◆ rfalST25xVPollerFastReadSingleBlock()

```
ReturnCode rfalST25xVPollerFastReadSingleBlock ( uint8_t          flags,
                                                 const uint8_t *  uid,
                                                 uint8_t          blockNum,
                                                 uint8_t *        rxBuf,
                                                 uint16_t         rxBufLen,
                                                 uint16_t *       rcvLen
                                               )
```

Reads a Single Block from a device (VICC) using ST Fast mode

Parameters

[in] **flags** : Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT

[in] **uid** : UID of the device to be put to be read if not provided Select mode will be used

[in] **blockNum** : Number of the block to read

[out] **rxBuf** : buffer to store response (also with RES_FLAGS)

[in] **rxBufLen** : length of rxBuf

[out] **rcvLen** : number of bytes received

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

◆ rfalST25xVPollerFastWriteDynamicConfiguration()

```
ReturnCode rfalST25xVPollerFastWriteDynamicConfiguration ( uint8_t          flags,
                                                           const uint8_t *   uid,
                                                           uint8_t          pointer,
                                                           uint8_t          regValue
                                                         )
```

Writes dynamic registers at the Pointer address using ST Fast mode

Parameters

[in] **flags** : Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT

[in] **uid** : UID of the device to be put to be read if not provided Select mode will be used

[in] **pointer** : Pointer address

[in] **regValue** : Register value

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

◆ rfalST25xVPollerFastWriteMessage()

```
ReturnCode rfalST25xVPollerFastWriteMessage ( uint8_t          flags,
                                              const uint8_t *  uid,
                                              uint8_t          msgLen,
                                              const uint8_t *  msgData,
                                              uint8_t *        txBuf,
                                              uint16_t         txBufLen
                                            )
```

Sends Fast Write message Command using ST Fast mode

On receiving the Write Message command, the ST25DVxxx puts the data contained in the request into the Mailbox buffer, update the MB_LEN_Dyn register, and set bit RF_PUT_MSG in MB_CTRL_Dyn register. It then reports if the write operation was successful in the response. The ST25DVxxx Mailbox contains up to 256 data bytes which are filled from the first location '00'. MSGlength parameter of the command is the number of Data bytes minus 1 (00 for 1 byte of data, FFh for 256 bytes of data). Write Message could be executed only when Mailbox is accessible by RF.

Parameters

- [in] **flags** : Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT
- [in] **uid** : UID of the device to be put to be read if not provided Select mode will be used
- [in] **msgLen** : MSGLen number of Data bytes minus 1
- [in] **msgData** : Message Data
- [out] **txBuf** : buffer to used to build the Write Message command
- [in] **txBufLen** : length of txBuf

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

◆ rfalST25xVPollerGetRandomNumber()

```
ReturnCode rfalST25xVPollerGetRandomNumber ( uint8_t          flags,
                                             const uint8_t *  uid,
                                             uint8_t *        rxBuf,
                                             uint16_t         rxBufLen,
                                             uint16_t *       rcvLen
                                           )
```

Returns a 16 bit random number

Parameters

[in]	**flags**	: Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT
[in]	**uid**	: UID of the device to be put to be read if not provided Select mode will be used
[out]	**rxBuf**	: buffer to store response (also with RES_FLAGS)
[in]	**rxBufLen**	: length of rxBuf
[out]	**rcvLen**	: number of bytes received

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

◆ rfalST25xVPollerM24LRFastReadMultipleBlocks()

```
ReturnCode rfalST25xVPollerM24LRFastReadMultipleBlocks ( uint8_t          flags,
                                                         const uint8_t *  uid,
```

```
                                                    uint16_t        firstBlockNum,
                                                    uint8_t         numOfBlocks,
                                                    uint8_t *       rxBuf,
                                                    uint16_t        rxBufLen,
                                                    uint16_t *      rcvLen
                                                  )
```

Reads Multiple Blocks from a device from a M24LR tag which has the number of blocks bigger than 256 (M24LR16 ; M24LR64) using ST Fast mode

Parameters

[in]	**flags**	: Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT
[in]	**uid**	: UID of the device to be put to be read if not provided Select mode will be used
[in]	**firstBlockNum**	: first block to be read (16 bits)
[in]	**numOfBlocks**	: number of block to read
[out]	**rxBuf**	: buffer to store response (also with RES_FLAGS)
[in]	**rxBufLen**	: length of rxBuf
[out]	**rcvLen**	: number of bytes received

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

◆ rfalST25xVPollerM24LRFastReadSingleBlock()

```
ReturnCode rfalST25xVPollerM24LRFastReadSingleBlock ( uint8_t          flags,
                                                      const uint8_t *  uid,
                                                      uint16_t         blockNum,
```

```
                                                    uint8_t *        rxBuf,
                                                    uint16_t         rxBufLen,
                                                    uint16_t *       rcvLen
                                                  )
```

Reads a Single Block from a M24LR tag which has the number of blocks bigger than 256 (M24LR16 ; M24LR64) using ST Fast mode

Parameters

- [in] **flags** : Flags to be used: Sub-carrier; Data_rate; Option default: RFAL_NFCV_REQ_FLAG_DEFAULT
- [in] **uid** : UID of the device to be put to be read if not provided Select mode will be used
- [in] **blockNum** : Number of the block to read (16 bits)
- [out] **rxBuf** : buffer to store response (also with RES_FLAGS)
- [in] **rxBufLen** : length of rxBuf
- [out] **rcvLen** : number of bytes received

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

◆ rfalST25xVPollerM24LRReadMultipleBlocks()

```
ReturnCode rfalST25xVPollerM24LRReadMultipleBlocks ( uint8_t          flags,
                                                     const uint8_t *  uid,
                                                     uint16_t         firstBlockNum,
                                                     uint8_t          numOfBlocks,
                                                     uint8_t *        rxBuf,
                                                     uint16_t         rxBufLen,
                                                     uint16_t *       rcvLen
```

)

Reads Multiple Blocks from a device from a M24LR tag which has the number of blocks bigger than 256 (M24LR16 ; M24LR64)

Parameters

[in]	**flags**	: Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT
[in]	**uid**	: UID of the device to be put to be read if not provided Select mode will be used
[in]	**firstBlockNum**	: first block to be read (16 bits)
[in]	**numOfBlocks**	: number of block to read
[out]	**rxBuf**	: buffer to store response (also with RES_FLAGS)
[in]	**rxBufLen**	: length of rxBuf
[out]	**rcvLen**	: number of bytes received

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

◆ rfalST25xVPollerM24LRReadSingleBlock()

```
ReturnCode rfalST25xVPollerM24LRReadSingleBlock ( uint8_t          flags,
                                                  const uint8_t *  uid,
                                                  uint16_t         blockNum,
                                                  uint8_t *        rxBuf,
                                                  uint16_t         rxBufLen,
                                                  uint16_t *       rcvLen
                                                )
```

Reads a Single Block from a M24LR tag which has the number of blocks bigger than 256 (M24LR16 ; M24LR64)

Parameters

[in]	**flags**	: Flags to be used: Sub-carrier; Data_rate; Option default: RFAL_NFCV_REQ_FLAG_DEFAULT
[in]	**uid**	: UID of the device to be put to be read if not provided Select mode will be used
[in]	**blockNum**	: Number of the block to read (16 bits)
[out]	**rxBuf**	: buffer to store response (also with RES_FLAGS)
[in]	**rxBufLen**	: length of rxBuf
[out]	**rcvLen**	: number of bytes received

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

◆ rfalST25xVPollerM24LRWriteSingleBlock()

```
ReturnCode rfalST25xVPollerM24LRWriteSingleBlock ( uint8_t          flags,
                                                   const uint8_t *  uid,
                                                   uint16_t         blockNum,
                                                   const uint8_t *  wrData,
                                                   uint8_t          blockLen
                                                 )
```

Writes a Single Block from a M24LR tag which has the number of blocks bigger than 256 (M24LR16 ; M24LR64)

Parameters

[in]	**flags**	: Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT
[in]	**uid**	: UID of the device to be put to be written if not provided Select mode will be used

[in] **blockNum** : Number of the block to write (16 bits)

[in] **wrData** : data to be written on the given block

[in] **blockLen** : number of bytes of a block

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

◆ rfalST25xVPollerPresentPassword()

```
ReturnCode rfalST25xVPollerPresentPassword ( uint8_t          flags,
                                             const uint8_t *  uid,
                                             uint8_t          pwdNum,
                                             const uint8_t *  pwd,
                                             uint8_t          pwdLen
                                           )
```

Sends the Present Password command

Parameters

[in] **flags** : Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT

[in] **uid** : UID of the device to be put to be read if not provided Select mode will be used

[in] **pwdNum** : Password number

[in] **pwd** : Password

[in] **pwdLen** : Password length

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

◆ rfalST25xVPollerReadConfiguration()

```
ReturnCode rfalST25xVPollerReadConfiguration ( uint8_t          flags,
                                               const uint8_t *  uid,
                                               uint8_t          pointer,
                                               uint8_t *        regValue
                                             )
```

Reads static configuration registers at the Pointer address

Parameters

[in]	**flags**	: Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT
[in]	**uid**	: UID of the device to be put to be read if not provided Select mode will be used
[in]	**pointer**	: Pointer address
[out]	**regValue**	: Register value

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

◆ rfalST25xVPollerReadDynamicConfiguration()

```
ReturnCode rfalST25xVPollerReadDynamicConfiguration ( uint8_t        flags,
                                                      const uint8_t * uid,
                                                      uint8_t        pointer,
                                                      uint8_t *      regValue
                                                    )
```

Reads dynamic registers at the Pointer address

Parameters

- [in] **flags** : Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT
- [in] **uid** : UID of the device to be put to be read if not provided Select mode will be used
- [in] **pointer** : Pointer address
- [out] **regValue** : Register value

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

◆ rfalST25xVPollerReadMessage()

```
ReturnCode rfalST25xVPollerReadMessage ( uint8_t        flags,
                                         const uint8_t * uid,
                                         uint8_t        mbPointer,
                                         uint8_t        numBytes,
                                         uint8_t *      rxBuf,
                                         uint16_t       rxBufLen,
```

```
                                        uint16_t *        rcvLen
                                      )
```

Reads up to 256 bytes in the Mailbox from the location specified by MBpointer and sends back their value in the rxBuf response. First MailBox location is '00'. When Number of bytes is set to 00h and MBPointer is equals to 00h, the MB_LEN bytes of the full message are returned. Otherwise, Read Message command returns (Number of Bytes + 1) bytes (i.e. 01h returns 2 bytes, FFh returns 256 bytes). An error is reported if (Pointer + Nb of bytes + 1) is greater than the message length. RF Reading of the last byte of the mailbox message automatically clears b1 of MB_CTRL_Dyn HOST_PUT_MSG, and allows RF to put a new message.

Parameters

[in]	**flags**	: Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT
[in]	**uid**	: UID of the device to be put to be read if not provided Select mode will be used
[in]	**mbPointer**	: MPpointer
[in]	**numBytes**	: number of bytes
[out]	**rxBuf**	: buffer to store response (also with RES_FLAGS)
[in]	**rxBufLen**	: length of rxBuf
[out]	**rcvLen**	: number of bytes received

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

◆ rfalST25xVPollerReadMessageLength()

```
ReturnCode rfalST25xVPollerReadMessageLength ( uint8_t          flags,
                                               const uint8_t *  uid,
                                               uint8_t *        msgLen
                                             )
```

Sends a Read Message Length message to retrieve the value of MB_LEN_Dyn

Parameters

[in] **flags** : Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT

[in] **uid** : UID of the device to be put to be read if not provided Select mode will be used

[out] **msgLen** : Message Length

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

◆ rfalST25xVPollerWriteConfiguration()

```
ReturnCode rfalST25xVPollerWriteConfiguration ( uint8_t        flags,
                                                const uint8_t * uid,
                                                uint8_t        pointer,
                                                uint8_t        regValue
                                              )
```

Writes static configuration registers at the Pointer address

Parameters

[in] **flags** : Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT

[in] **uid** : UID of the device to be put to be read if not provided Select mode will be used

[in] **pointer** : Pointer address

[in] **regValue** : Register value

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

◆ rfalST25xVPollerWriteDynamicConfiguration()

```
ReturnCode rfalST25xVPollerWriteDynamicConfiguration ( uint8_t        flags,
                                                       const uint8_t * uid,
                                                       uint8_t        pointer,
                                                       uint8_t        regValue
                                                     )
```

Writes dynamic registers at the Pointer address

Parameters

- [in] **flags** : Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT
- [in] **uid** : UID of the device to be put to be read if not provided Select mode will be used
- [in] **pointer** : Pointer address
- [in] **regValue** : Register value

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

◆ rfalST25xVPollerWriteMessage()

```
ReturnCode rfalST25xVPollerWriteMessage ( uint8_t          flags,
                                          const uint8_t *  uid,
                                          uint8_t          msgLen,
                                          const uint8_t *  msgData,
                                          uint8_t *        txBuf,
                                          uint16_t         txBufLen
                                        )
```

Sends Write message Command

On receiving the Write Message command, the ST25DVxxx puts the data contained in the request into the Mailbox buffer, update the MB_LEN_Dyn register, and set bit RF_PUT_MSG in MB_CTRL_Dyn register. It then reports if the write operation was successful in the response. The ST25DVxxx Mailbox contains up to 256 data bytes which are filled from the first location '00'. MSGlength parameter of the command is the number of Data bytes minus 1 (00 for 1 byte of data, FFh for 256 bytes of data). Write Message could be executed only when Mailbox is accessible by RF.

Parameters

[in]	**flags**	: Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT
[in]	**uid**	: UID of the device to be put to be read if not provided Select mode will be used
[in]	**msgLen**	: MSGLen number of Data bytes minus 1
[in]	**msgData**	: Message Data
[out]	**txBuf**	: buffer to used to build the Write Message command
[in]	**txBufLen**	: length of txBuf

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

◆ rfalST25xVPollerWritePassword()

```
ReturnCode rfalST25xVPollerWritePassword ( uint8_t          flags,
                                           const uint8_t *  uid,
                                           uint8_t          pwdNum,
                                           const uint8_t *  pwd,
                                           uint8_t          pwdLen
                                         )
```

Sends the Write Password command

Parameters

- [in] **flags** : Flags to be used: Sub-carrier; Data_rate; Option for NFC-Forum use: RFAL_NFCV_REQ_FLAG_DEFAULT
- [in] **uid** : UID of the device to be put to be read if not provided Select mode will be used
- [in] **pwdNum** : Password number
- [in] **pwd** : Password
- [in] **pwdLen** : Password length

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or incorrect mode

RFAL_ERR_PARAM : Invalid parameters

RFAL_ERR_IO : Generic internal error

RFAL_ERR_CRC : CRC error detected

RFAL_ERR_FRAMING : Framing error detected

RFAL_ERR_PROTO : Protocol error detected

RFAL_ERR_TIMEOUT : Timeout error

RFAL_ERR_NONE : No error

T1T

RFAL T1T Module. More...

Collaboration diagram for T1T:

Data Structures

struct **rfalT1TRidRes**

Macros

#define **RFAL_T1T_UID_LEN** 4

#define **RFAL_T1T_HR_LENGTH** 2

#define **RFAL_T1T_HR0_NDEF_MASK** 0xF0

#define **RFAL_T1T_HR0_NDEF_SUPPORT** 0x10

Enumerations

enum **rfalT1Tcmds** {
RFAL_T1T_CMD_RID = 0x78 , **RFAL_T1T_CMD_RALL** = 0x00 , **RFAL_T1T_CMD_READ** = 0x01 ,
RFAL_T1T_CMD_WRITE_E = 0x53 ,
RFAL_T1T_CMD_WRITE_NE = 0x1A
}

Functions

ReturnCode	**rfalT1TPollerInitialize** (void) Initialize NFC-A T1T Poller mode. More...
ReturnCode	**rfalT1TPollerRid** (**rfalT1TRidRes** *ridRes) NFC-A T1T Poller RID. More...
ReturnCode	**rfalT1TPollerRall** (const uint8_t *uid, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rxRcvdLen) NFC-A T1T Poller RALL. More...
ReturnCode	**rfalT1TPollerWrite** (const uint8_t *uid, uint8_t address, uint8_t data) NFC-A T1T Poller Write. More...

Detailed Description

Macro Definition Documentation

◆ RFAL_T1T_HR0_NDEF_MASK

```
#define RFAL_T1T_HR0_NDEF_MASK   0xF0
```

T1T HR0 NDEF capability mask T1T 1.2 2.2.2

◆ RFAL_T1T_HR0_NDEF_SUPPORT

```
#define RFAL_T1T_HR0_NDEF_SUPPORT   0x10
```

T1T HR0 NDEF capable value T1T 1.2 2.2.2

◆ RFAL_T1T_HR_LENGTH

```
#define RFAL_T1T_HR_LENGTH   2
```

T1T HR(Header ROM) length

◆ RFAL_T1T_UID_LEN

#define RFAL_T1T_UID_LEN 4

T1T UID length of cascade level 1 only tag

Enumeration Type Documentation

◆ rfalT1Tcmds

enum **rfalT1Tcmds**

NFC-A T1T (Topaz) command set

Enumerator	
RFAL_T1T_CMD_RID	T1T Read UID
RFAL_T1T_CMD_RALL	T1T Read All
RFAL_T1T_CMD_READ	T1T Read
RFAL_T1T_CMD_WRITE_E	T1T Write with erase (single byte)
RFAL_T1T_CMD_WRITE_NE	T1T Write with no erase (single byte)

Function Documentation

◆ rfalT1TPollerInitialize()

ReturnCode rfalT1TPollerInitialize (void)

This methods configures RFAL RF layer to perform as a NFC-A T1T Poller/RW (Topaz) including all default timings

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or mode not set

RFAL_ERR_NONE : No error

◆ rfalT1TPollerRall()

```
ReturnCode rfalT1TPollerRall ( const uint8_t *  uid,
                               uint8_t *        rxBuf,
                               uint16_t         rxBufLen,
                               uint16_t *       rxRcvdLen
                             )
```

This method send a Read All command to a NFC-A T1T Listener device

Parameters

- [in] **uid** : the UID of the device to read data
- [out] **rxBuf** : pointer to place the read data
- [in] **rxBufLen** : size of rxBuf
- [out] **rxRcvdLen** : actual received data

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or mode not set

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error

◆ rfalT1TPollerRid()

```
ReturnCode rfalT1TPollerRid ( rfalT1TRidRes *  ridRes )
```

This method reads the UID of a NFC-A T1T Listener device

Parameters

- [out] **ridRes** : pointer to place the RID_RES

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or mode not set

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error

◆ rfalT1TPollerWrite()

```
ReturnCode rfalT1TPollerWrite ( const uint8_t *  uid,
```

```
                                uint8_t          address,
                                uint8_t          data
                              )
```

This method writes the given data on the address of a NFC-A T1T Listener device

Parameters

- [in] **uid** : the UID of the device to read data
- [in] **address** : address to write the data
- [in] **data** : the data to be written

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or mode not set

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_NONE : No error

rfalT1TRidRes Struct Reference

```
#include <rfal_t1t.h>
```

Data Fields

uint8_t **hr0**

uint8_t **hr1**

uint8_t **uid** [**RFAL_T1T_UID_LEN**]

Detailed Description

NFC-A T1T (Topaz) RID_RES Digital 1.1 10.6.2 & Table 50

Field Documentation

◆ hr0

uint8_t rfalT1TRidRes::hr0

T1T Header ROM: HR0

◆ hr1

uint8_t rfalT1TRidRes::hr1

T1T Header ROM: HR1

◆ uid

uint8_t rfalT1TRidRes::uid[**RFAL_T1T_UID_LEN**]

T1T UID

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

- **rfal_t1t.h**

T2T

RFAL T2T Module. More...

Collaboration diagram for T2T:

Macros

#define **RFAL_T2T_BLOCK_LEN** 4U

#define **RFAL_T2T_READ_DATA_LEN** (4U * **RFAL_T2T_BLOCK_LEN**)

#define **RFAL_T2T_WRITE_DATA_LEN** **RFAL_T2T_BLOCK_LEN**

Functions

ReturnCode **rfalT2TPollerRead** (uint8_t blockNum, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen)

	NFC-A T2T Poller Read. More...
ReturnCode	**rfalT2TPollerWrite** (uint8_t blockNum, const uint8_t *wrData) NFC-A T2T Poller Write. More...
ReturnCode	**rfalT2TPollerSectorSelect** (uint8_t sectorNum) NFC-A T2T Poller Sector Select. More...

Detailed Description

Macro Definition Documentation

◆ RFAL_T2T_BLOCK_LEN

```
#define RFAL_T2T_BLOCK_LEN   4U
```

T2T block length

◆ RFAL_T2T_READ_DATA_LEN

```
#define RFAL_T2T_READ_DATA_LEN   (4U * RFAL_T2T_BLOCK_LEN)
```

T2T READ data length

◆ RFAL_T2T_WRITE_DATA_LEN

```
#define RFAL_T2T_WRITE_DATA_LEN   RFAL_T2T_BLOCK_LEN
```

T2T WRITE data length

Function Documentation

◆ rfalT2TPollerRead()

```
ReturnCode rfalT2TPollerRead ( uint8_t     blockNum,
                               uint8_t *   rxBuf,
                               uint16_t    rxBufLen,
                               uint16_t *  rcvLen
                             )
```

This method sends a Read command to a NFC-A T2T Listener device

Parameters

- [in] **blockNum** : Number of the block to read
- [out] **rxBuf** : pointer to place the read data
- [in] **rxBufLen** : size of rxBuf (RFAL_T2T_READ_DATA_LEN)
- [out] **rcvLen** : actual received data

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or mode not set

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_PROTO : Protocol error

RFAL_ERR_NONE : No error

◆ rfalT2TPollerSectorSelect()

```
ReturnCode rfalT2TPollerSectorSelect ( uint8_t  sectorNum )
```

This method sends a Sector Select commands to a NFC-A T2T Listener device

Parameters

- [in] **sectorNum** : Sector Number

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or mode not set

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_PROTO : Protocol error

RFAL_ERR_NONE : No error

◆ rfalT2TPollerWrite()

```
ReturnCode rfalT2TPollerWrite ( uint8_t          blockNum,
                                const uint8_t *  wrData
                              )
```

This method sends a Write command to a NFC-A T2T Listener device

Parameters

[in] **blockNum** : Number of the block to write

[in] **wrData** : data to be written on the given block size must be of RFAL_T2T_WRITE_DATA_LEN

Returns

RFAL_ERR_WRONG_STATE : RFAL not initialized or mode not set

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_PROTO : Protocol error

RFAL_ERR_NONE : No error

RFAL

Main Page

Data Structures

Files

Examples

Data Structures | Macros | Enumerations | Functions

T4T

RFAL » **RFAL-AL**

RFAL T4T Module. More...

Collaboration diagram for T4T:

Data Structures

struct **rfalT4tCApduParam**

struct **rfalT4tRApduParam**

Macros

#define **RFAL_T4T_MAX_CAPDU_PROLOGUE_LEN** 4U

#define **RFAL_T4T_LE_LEN** 1U

#define **RFAL_T4T_LC_LEN** 1U

#define **RFAL_T4T_MAX_RAPDU_SW1SW2_LEN** 2U

#define **RFAL_T4T_CLA** 0x00U

#define **RFAL_T4T_ISO7816_P1_SELECT_BY_DF_NAME** 0x04U

#define **RFAL_T4T_ISO7816_P1_SELECT_BY_FILEID** 0x00U

#define **RFAL_T4T_ISO7816_P2_SELECT_FIRST_OR_ONLY_OCCURENCE** 0x00U

#define **RFAL_T4T_ISO7816_P2_SELECT_RETURN_FCI_TEMPLATE** 0x00U

#define **RFAL_T4T_ISO7816_P2_SELECT_NO_RESPONSE_DATA** 0x0CU

#define **RFAL_T4T_ISO7816_STATUS_COMPLETE** 0x9000U

Enumerations

enum **rfalT4tCmds** {
RFAL_T4T_INS_SELECT = 0xA4U , **RFAL_T4T_INS_READBINARY** = 0xB0U ,
RFAL_T4T_INS_UPDATEBINARY = 0xD6U , **RFAL_T4T_INS_READBINARY_ODO** = 0xB1U ,
RFAL_T4T_INS_UPDATEBINARY_ODO = 0xD7U
}

Functions

ReturnCode **rfalT4TPollerComposeCAPDU** (const **rfalT4tCApduParam** *apduParam)
T4T Compose APDU. More...

ReturnCode **rfalT4TPollerParseRAPDU** (**rfalT4tRApduParam** *apduParam)
T4T Parse R-APDU. More...

ReturnCode **rfalT4TPollerComposeSelectAppl** (**rfalIsoDepApduBufFormat** *cApduBuf, const uint8_t *aid, uint8_t aidLen, uint16_t *cApduLen)
T4T Compose Select Application APDU. More...

ReturnCode **rfalT4TPollerComposeSelectFile** (**rfalIsoDepApduBufFormat** *cApduBuf, const uint8_t *fid, uint8_t fidLen, uint16_t *cApduLen)
T4T Compose Select File APDU. More...

ReturnCode **rfalT4TPollerComposeSelectFileV1Mapping** (**rfalIsoDepApduBufFormat** *cApduBuf, const uint8_t *fid, uint8_t fidLen, uint16_t *cApduLen)
T4T Compose Select File APDU for Mapping Version 1. More...

ReturnCode **rfalT4TPollerComposeReadData** (**rfalIsoDepApduBufFormat** *cApduBuf, uint16_t offset, uint8_t expLen, uint16_t *cApduLen)
T4T Compose Read Data APDU. More...

ReturnCode **rfalT4TPollerComposeReadDataODO** (**rfalIsoDepApduBufFormat** *cApduBuf, uint32_t offset, uint8_t expLen, uint16_t *cApduLen)
T4T Compose Read Data ODO APDU. More...

ReturnCode **rfalT4TPollerComposeWriteData** (**rfalIsoDepApduBufFormat** *cApduBuf, uint16_t offset, const uint8_t *data, uint8_t dataLen, uint16_t *cApduLen)
T4T Compose Write Data APDU. More...

ReturnCode **rfalT4TPollerComposeWriteDataODO** (**rfalIsoDepApduBufFormat** *cApduBuf, uint32_t offset, const uint8_t *data, uint8_t dataLen, uint16_t *cApduLen)
T4T Compose Write Data ODO APDU. More...

Detailed Description

Macro Definition Documentation

◆ RFAL_T4T_CLA

```
#define RFAL_T4T_CLA   0x00U
```

Class byte (contains 00h because secure message are not used)

◆ RFAL_T4T_ISO7816_P1_SELECT_BY_DF_NAME

```
#define RFAL_T4T_ISO7816_P1_SELECT_BY_DF_NAME   0x04U
```

P1 value for Select by name

◆ RFAL_T4T_ISO7816_P1_SELECT_BY_FILEID

```
#define RFAL_T4T_ISO7816_P1_SELECT_BY_FILEID   0x00U
```

P1 value for Select by file identifier

◆ RFAL_T4T_ISO7816_P2_SELECT_FIRST_OR_ONLY_OCCURENCE

```
#define RFAL_T4T_ISO7816_P2_SELECT_FIRST_OR_ONLY_OCCURENCE   0x00U
```

b2b1 P2 value for First or only occurence

◆ RFAL_T4T_ISO7816_P2_SELECT_NO_RESPONSE_DATA

```
#define RFAL_T4T_ISO7816_P2_SELECT_NO_RESPONSE_DATA   0x0CU
```

b4b3 P2 value for No responce data

◆ RFAL_T4T_ISO7816_P2_SELECT_RETURN_FCI_TEMPLATE

```
#define RFAL_T4T_ISO7816_P2_SELECT_RETURN_FCI_TEMPLATE   0x00U
```

b4b3 P2 value for Return FCI template

◆ RFAL_T4T_ISO7816_STATUS_COMPLETE

```
#define RFAL_T4T_ISO7816_STATUS_COMPLETE   0x9000U
```

Command completed \ Normal processing - No further qualification

◆ RFAL_T4T_LC_LEN

```
#define RFAL_T4T_LC_LEN   1U
```

Lc Data field length (short field coding)

◆ RFAL_T4T_LE_LEN

```
#define RFAL_T4T_LE_LEN   1U
```

Le Expected Response Length (short field coding)

◆ RFAL_T4T_MAX_CAPDU_PROLOGUE_LEN

```
#define RFAL_T4T_MAX_CAPDU_PROLOGUE_LEN   4U
```

Command-APDU prologue length (CLA INS P1 P2)

◆ RFAL_T4T_MAX_RAPDU_SW1SW2_LEN

```
#define RFAL_T4T_MAX_RAPDU_SW1SW2_LEN   2U
```

SW1 SW2 length

Enumeration Type Documentation

◆ rfalT4tCmds

enum **rfalT4tCmds**

NFC-A T4T command set T4T 1.0 & ISO7816-4 2013 Table 4

Enumerator	
RFAL_T4T_INS_SELECT	T4T Select
RFAL_T4T_INS_READBINARY	T4T ReadBinary
RFAL_T4T_INS_UPDATEBINARY	T4T UpdateBinay
RFAL_T4T_INS_READBINARY_ODO	T4T ReadBinary using ODO
RFAL_T4T_INS_UPDATEBINARY_ODO	T4T UpdateBinay using ODO

Function Documentation

◆ rfalT4TPollerComposeCAPDU()

ReturnCode rfalT4TPollerComposeCAPDU (const **rfalT4tCApduParam** * apduParam)

This method computes a C-APDU according to NFC Forum T4T and ISO7816-4.

If C-APDU contains data to be sent, it must be placed inside the buffer rfalT4tTxRxApduParam.txRx.cApduBuf.apdu and signaled by Lc

To transceive the formed APDU the ISO-DEP layer shall be used

See also

rfalIsoDepStartApduTransceive()

rfalIsoDepGetApduTransceiveStatus()

rfalT4TPollerParseRAPDU()

Warning

The ISO-DEP module is used to perform the tranceive. Usually activation has been done via ISO-DEP activatiavtion. If not please call **rfalIsoDepInitialize()** before.

Parameters

`[in,out]` **apduParam** : APDU parameters apduParam.cApduLen will contain the APDU length

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_PROTO : Protocol error

RFAL_ERR_NONE : No error

◆ rfalT4TPollerComposeReadData()

```
ReturnCode rfalT4TPollerComposeReadData ( rfalIsoDepApduBufFormat *  cApduBuf,
                                          uint16_t                   offset,
                                          uint8_t                    expLen,
                                          uint16_t *                 cApduLen
                                        )
```

This method computes a Read Data APDU according to NFC Forum T4T

To transceive the formed APDU the ISO-DEP layer shall be used

See also

rfalIsoDepStartApduTransceive()

rfalIsoDepGetApduTransceiveStatus()

Parameters

[out] **cApduBuf** : buffer where the C-APDU will be placed

[in] **offset** : File offset

[in] **expLen** : Expected length (Le)

[out] **cApduLen** : Composed C-APDU length

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_PROTO : Protocol error

RFAL_ERR_NONE : No error

◆ rfalT4TPollerComposeReadDataODO()

```
ReturnCode rfalT4TPollerComposeReadDataODO ( rfalIsoDepApduBufFormat *  cApduBuf,
                                             uint32_t                    offset,
                                             uint8_t                     expLen,
```

```
                                                uint16_t *                    cApduLen
                                              )
```

This method computes a Read Data ODO APDU according to NFC Forum T4T

To transceive the formed APDU the ISO-DEP layer shall be used

See also

rfalIsoDepStartApduTransceive()

rfalIsoDepGetApduTransceiveStatus()

Parameters

[out] **cApduBuf** : buffer where the C-APDU will be placed

[in] **offset** : File offset

[in] **expLen** : Expected length (Le)

[out] **cApduLen** : Composed C-APDU length

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_PROTO : Protocol error

RFAL_ERR_NONE : No error

◆ rfalT4TPollerComposeSelectAppl()

```
ReturnCode rfalT4TPollerComposeSelectAppl ( rfalIsoDepApduBufFormat *  cApduBuf,
                                            const uint8_t *             aid,
                                            uint8_t                     aidLen,
                                            uint16_t *                  cApduLen
                                          )
```

This method computes a Select Application APDU according to NFC Forum T4T

To transceive the formed APDU the ISO-DEP layer shall be used

See also

rfalIsoDepStartApduTransceive()

rfalIsoDepGetApduTransceiveStatus()

Parameters

[out] **cApduBuf** : buffer where the C-APDU will be placed

[in] **aid** : Application ID to be used

[in] **aidLen** : Application ID length

[out] **cApduLen** : Composed C-APDU length

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_PROTO : Protocol error

RFAL_ERR_NONE : No error

◆ rfalT4TPollerComposeSelectFile()

```
ReturnCode rfalT4TPollerComposeSelectFile ( rfalIsoDepApduBufFormat *  cApduBuf,
                                            const uint8_t *             fid,
                                            uint8_t                     fidLen,
                                            uint16_t *                  cApduLen
                                          )
```

This method computes a Select File APDU according to NFC Forum T4T

To transceive the formed APDU the ISO-DEP layer shall be used

See also

rfalIsoDepStartApduTransceive()

rfalIsoDepGetApduTransceiveStatus()

Parameters

[out] **cApduBuf** : buffer where the C-APDU will be placed

[in] **fid** : File ID to be used

[in] **fidLen** : File ID length

[out] **cApduLen** : Composed C-APDU length

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_PROTO : Protocol error

RFAL_ERR_NONE : No error

◆ rfalT4TPollerComposeSelectFileV1Mapping()

```
ReturnCode rfalT4TPollerComposeSelectFileV1Mapping ( rfalIsoDepApduBufFormat *  cApduBuf,
                                                     const uint8_t *            fid,
                                                     uint8_t                    fidLen,
                                                     uint16_t *                 cApduLen
                                                   )
```

This method computes a Select File APDU according to NFC Forum T4TOP_v1.0

To transceive the formed APDU the ISO-DEP layer shall be used

See also

rfalIsoDepStartApduTransceive()

rfalIsoDepGetApduTransceiveStatus()

Parameters

[out] **cApduBuf** : buffer where the C-APDU will be placed

[in] **fid** : File ID to be used

[in] **fidLen** : File ID length

[out] **cApduLen** : Composed C-APDU length

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_PROTO : Protocol error

RFAL_ERR_NONE : No error

◆ rfalT4TPollerComposeWriteData()

```
ReturnCode rfalT4TPollerComposeWriteData ( rfalIsoDepApduBufFormat *  cApduBuf,
                                           uint16_t                    offset,
                                           const uint8_t *             data,
                                           uint8_t                     dataLen,
                                           uint16_t *                  cApduLen
                                         )
```

This method computes a Write Data APDU according to NFC Forum T4T

To transceive the formed APDU the ISO-DEP layer shall be used

See also

rfaIsoDepStartApduTransceive()

rfaIsoDepGetApduTransceiveStatus()

Parameters

[out] **cApduBuf** : buffer where the C-APDU will be placed

[in] **offset** : File offset

[in] **data** : Data to be written

[in] **dataLen** : Data length to be written (Lc)

[out] **cApduLen** : Composed C-APDU length

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_PROTO : Protocol error

RFAL_ERR_NONE : No error

◆ rfalT4TPollerComposeWriteDataODO()

```
ReturnCode rfalT4TPollerComposeWriteDataODO ( rfaIsoDepApduBufFormat *  cApduBuf,
                                              uint32_t                   offset,
                                              const uint8_t *            data,
                                              uint8_t                    dataLen,
                                              uint16_t *                 cApduLen
                                            )
```

This method computes a Write Data ODO sAPDU according to NFC Forum T4T

To transceive the formed APDU the ISO-DEP layer shall be used

See also

rfaIsoDepStartApduTransceive()

rfaIsoDepGetApduTransceiveStatus()

Parameters

[out] **cApduBuf** : buffer where the C-APDU will be placed

[in] **offset** : File offset

[in] **data** : Data to be written

[in] **dataLen** : Data length to be written (Lc)

[out] **cApduLen** : Composed C-APDU length

Returns

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_PROTO : Protocol error

RFAL_ERR_NONE : No error

◆ rfalT4TPollerParseRAPDU()

ReturnCode rfalT4TPollerParseRAPDU (**rfalT4tRApduParam** * apduParam)

This method parses a R-APDU according to NFC Forum T4T and ISO7816-4. It will extract the data length and check if the Satus word is expected.

Parameters

[in,out] **apduParam** : APDU parameters apduParam.rApduBodyLen will contain the data length
apduParam.statusWord will contain the SW1 and SW2

Returns

RFAL_ERR_REQUEST : Status word (SW1 SW2) different from 9000

RFAL_ERR_PARAM : Invalid parameter

RFAL_ERR_PROTO : Protocol error

RFAL_ERR_NONE : No error

rfalT4tCApduParam Struct Reference


```
#include <rfal_t4t.h>
```

Collaboration diagram for rfalT4tCApduParam:

[legend]

Data Fields

uint8_t	**CLA**
uint8_t	**INS**
uint8_t	**P1**
uint8_t	**P2**
uint8_t	**Lc**
bool	**LcFlag**
uint8_t	**Le**
bool	**LeFlag**
rfalIsoDepApduBufFormat *	**cApduBuf**

uint16_t *	**cApduLen**

Detailed Description

NFC-A T4T Command-APDU structure

Field Documentation

◆ cApduBuf

rfaIsoDepApduBufFormat* rfalT4tCApduParam::cApduBuf

Command-APDU buffer (Tx)

◆ cApduLen

uint16_t* rfalT4tCApduParam::cApduLen

Command-APDU Length

◆ CLA

uint8_t rfalT4tCApduParam::CLA

Class byte

◆ INS

uint8_t rfalT4tCApduParam::INS

Instruction byte

◆ Lc

uint8_t rfalT4tCApduParam::Lc

Data field length

◆ LcFlag

bool rfalT4tCApduParam::LcFlag

Lc flag (append Lc when true)

◆ Le

uint8_t rfalT4tCApduParam::Le

Expected Response Length

◆ LeFlag

bool rfalT4tCApduParam::LeFlag

Le flag (append Le when true)

◆ P1

uint8_t rfalT4tCApduParam::P1

Parameter byte 1

◆ P2

uint8_t rfalT4tCApduParam::P2

Parameter byte 2

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

rfalT4tRApduParam Struct Reference


```
#include <rfal_t4t.h>
```

Collaboration diagram for rfalT4tRApduParam:

[legend]

Data Fields

rfalIsoDepApduBufFormat * rApduBuf

uint16_t	**rcvdLen**
uint16_t	**rApduBodyLen**
uint16_t	**statusWord**

Detailed Description

NFC-A T4T Response-APDU structure

Field Documentation

◆ rApduBodyLen

uint16_t rfalT4tRApduParam::rApduBodyLen

Response body length

◆ rApduBuf

rfalIsoDepApduBufFormat* rfalT4tRApduParam::rApduBuf

Response-APDU buffer (Rx)

◆ rcvdLen

uint16_t rfalT4tRApduParam::rcvdLen

Full response length

◆ statusWord

uint16_t rfalT4tRApduParam::statusWord

R-APDU Status Word SW1|SW2

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

- rfal_t4t.h

Data Structures

Here are the data structures with brief descriptions:

- iso15693StreamConfig
- rfalAnalogConfig
- rfalAnalogConfigRegAddrMaskVal
- rfalCdRes
- rfalDlmaEntry
- rfalDlmaInfo
- rfalDpoEntry
- rfalDpoInfo
- rfalIso15693PhyConfig_t
- rfalIsoDepActivation

CrfaIIsoDepApduBufFormat

CrfaIIsoDepApduTxRxParam

CrfaIIsoDepAts

CrfaIIsoDepAtsParam

CrfaIIsoDepAttribCmd

CrfaIIsoDepAttribRes

CrfaIIsoDepAttribResParam

CrfaIIsoDepBufFormat

CrfaIIsoDepDevice

CrfaIIsoDepInfo

CrfaIIsoDepListenActvParam

CrfaIIsoDepPpsReq

CrfaIIsoDepPpsRes

CrfaIIsoDepRats

CrfaIIsoDepSParameter

CrfaIIsoDepTxRxParam

CrfaLmConfPA

CrfaLmConfPB

CrfaLmConfPF

CrfaINfcaListenDevice

CrfaINfcaSddReq

CrfaINfcaSddRes

CrfaINfcaSelReq

CrfaINfcaSelRes

CrfaINfcaSensRes

CrfaINfcbListenDevice

CrfaINfcbSensbRes

CrfaINfcbSensbResAppData

CrfaINfcbSensbResProtocolInfo

CrfaINfcBuffer

CrfaINfcDepActivation

CrfaINfcDepAtrParam

CrfaINfcDepAtrReq

CrfaINfcDepAtrRes

CrfalNfcDepBufFormat

CrfalNfcDepConfigs

CrfalNfcDepDevice

CrfalNfcDepInfo

CrfalNfcDepListenActvParam

CrfalNfcDepPduBufFormat

CrfalNfcDepPduTxRxParam

CrfalNfcDepTargetParam

CrfalNfcDepTxRxParam

CrfalNfcDevice

CrfalNfcDiscoverParam

CrfalNfcfBlockListElem

CrfalNfcfListenDevice

CrfalNfcfPollDevice

CrfalNfcfSensfRes

CrfalNfcfServBlockListParam

CrfalNfcPropCallbacks

CrfalNfcvGenericReq

CrfalNfcvGenericRes

CrfalNfcvInventoryRes

CrfalNfcvListenDevice

CrfalSt25tbListenDevice

CrfalT1TRidRes

CrfalT4tCApduParam

CrfalT4tRApduParam

CrfalTransceiveContext

CrfalWakeUpConfig

CrfalWakeUpInfo

CrfalWumMeasChannel

Cst25r3911StreamConfig

Cst25r3916AatTuneParams

Cst25r3916AatTuneResult

Cst25r3916StreamConfig

Cst25r500DpocrConfig

st25r500DpocrInfo
st25r95SPIRxContext
t_st25r3916Regs

iso15693StreamConfig Struct Reference

```
#include <rfal_iso15693_2.h>
```

Data Fields

uint8_t	**useBPSK**
uint8_t	**din**
uint8_t	**dout**
uint8_t	**report_period_length**

Detailed Description

Parameters how the stream mode should work

Field Documentation

◆ din

uint8_t iso15693StreamConfig::din

the divider for the in subcarrier frequency: fc/2^din

◆ dout

uint8_t iso15693StreamConfig::dout

the divider for the in subcarrier frequency fc/2^dout

◆ report_period_length

uint8_t iso15693StreamConfig::report_period_length

the length of the reporting period 2^report_period_length

◆ useBPSK

uint8_t iso15693StreamConfig::useBPSK

0: subcarrier, 1:BPSK

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

- **rfal_iso15693_2.h**

RFAL

Main Page

Modules

Files

Examples

Data Structures

Data Structure Index

Data Fields

rfaIIso15693PhyConfig_t Struct Reference

```
#include <rfal_iso15693_2.h>
```

Data Fields

rfaIIso15693VcdCoding_t	**coding**
uint32_t	**speedMode**

Detailed Description

Enum holding possible VICC datarates

Configuration parameter used by rfaIIso15693PhyConfigure

Field Documentation

◆ coding

rfaIIso15693VcdCoding_t rfaIIso15693PhyConfig_t::coding

desired VCD coding

◆ speedMode

uint32_t rfaIIso15693PhyConfig_t::speedMode

0: normal mode, 1: 2^1 = x2 Fast mode, 2 : 2^2 = x4 mode, 3 : 2^3 = x8 mode - all rx pulse numbers and times are divided by 1,2,4,8

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

- **rfal_iso15693_2.h**

---

rfalWakeUpConfig Struct Reference

```
#include <rfal_features.h>
```

Collaboration diagram for rfalWakeUpConfig:

[legend]

Data Fields

rfalWumPeriod	**period**
bool	**irqTout**
bool	**autoAvg**
bool	**skipCal**
bool	**skipReCal**
bool	**delCal**
bool	**delRef**
rfalWumMeasDuration	**measDur**
rfalWumMeasFilter	**measFil**
rfalWumMeasChannel	**I**
rfalWumMeasChannel	**Q**

struct {

bool **enabled**	
rfalWumPeriod **refDelay**	
}	**refWU**
struct {	
bool **enabled**	
uint8_t **delta**	
uint8_t **reference**	
bool **autoAvg**	
bool **aaInclMeas**	
rfalWumAAWeight **aaWeight**	
}	**indAmp**
struct {	
bool **enabled**	
uint8_t **delta**	
uint8_t **reference**	
bool **autoAvg**	
bool **aaInclMeas**	
rfalWumAAWeight **aaWeight**	
}	**indPha**
struct {	
bool **enabled**	
uint8_t **delta**	
uint8_t **reference**	

bool **autoAvg**	
bool **aaInclMeas**	
rfalWumAAWeight aaWeight	
}	**cap**
bool	**swTagDetect**
struct {	
bool **enabled**	
rfalWumPeriod refDelay	
}	**refWU**
struct {	
bool **enabled**	
uint8_t **delta**	
uint8_t **fracDelta**	
uint16_t **reference**	
bool **autoAvg**	
bool **aaInclMeas**	
rfalWumAAWeight aaWeight	
}	**indAmp**
struct {	
bool **enabled**	
uint8_t **delta**	
uint8_t **fracDelta**	

uint16_t **reference**	
bool **autoAvg**	
bool **aaInclMeas**	
rfalWumAAWeight **aaWeight**	
}	**indPha**
struct {	
bool **enabled**	
uint8_t **delta**	
uint16_t **reference**	
bool **autoAvg**	
bool **aaInclMeas**	
rfalWumAAWeight **aaWeight**	
}	**cap**
struct {	
bool **enabled**	
uint8_t **delta**	
uint16_t **reference**	
}	**indAmp**

Detailed Description

RFAL Wake-Up Mode configuration

Field Documentation

◆ aaInclMeas

bool rfalWakeUpConfig::aaInclMeas

When AutoAvg is enabled, include IRQ measurement

◆ aaWeight

rfalWumAAWeight rfalWakeUpConfig::aaWeight

When AutoAvg is enabled, last measure weight

◆ autoAvg

bool rfalWakeUpConfig::autoAvg

Use the HW Auto Averaging feature on the enabled channel(s)

Use the HW Auto Averaging feature

◆ [1/2]

struct { ... } rfalWakeUpConfig::cap

Capacitive Configuration

◆ [2/2]

struct { ... } rfalWakeUpConfig::cap

Capacitive Configuration

◆ delCal

bool rfalWakeUpConfig::delCal

Delay calibration step starting WU mode

◆ delRef

bool rfalWakeUpConfig::delRef

Delay reference step starting WU mode

◆ delta

uint8_t rfalWakeUpConfig::delta

Delta between the reference and measurement to wake-up

◆ enabled

bool rfalWakeUpConfig::enabled

Reference from WU mode enabled

Inductive Amplitude measurement enabled

Inductive Phase measurement enabled

Capacitive measurement enabled

◆ fracDelta

uint8_t rfalWakeUpConfig::fracDelta

Fractional part of the delta [0;3] 0.25 steps (SW TD only)

◆ I

rfalWumMeasChannel rfalWakeUpConfig::I

I channel Configuration

◆ [1/3]

struct { ... } rfalWakeUpConfig::indAmp

Inductive Amplitude Configuration

◆ [2/3]

struct { ... } rfalWakeUpConfig::indAmp

Inductive Amplitude Configuration

◆ [3/3]

struct { ... } rfalWakeUpConfig::indAmp

Inductive Amplitude Configuration

◆ [1/2]

struct { ... } rfalWakeUpConfig::indPha

Inductive Phase Configuration

◆ [2/2]

struct { ... } rfalWakeUpConfig::indPha

Inductive Phase Configuration

◆ irqTout

bool rfalWakeUpConfig::irqTout

IRQ at every timeout will refresh the measurement(s)

IRQ due to Wake-up timer and measurement event (wutme)

◆ measDur

rfalWumMeasDuration rfalWakeUpConfig::measDur

Wake-up measurement duration config

◆ measFil

rfalWumMeasFilter rfalWakeUpConfig::measFil

Wake-up measurement filter config

◆ period

rfalWumPeriod rfalWakeUpConfig::period

Wake-Up Timer period;how often measurement(s) is performed

◆ Q

rfalWumMeasChannel rfalWakeUpConfig::Q

Q channel Configuration

◆ refDelay

rfalWumPeriod rfalWakeUpConfig::refDelay

Obtain reference from WU after delay time

◆ reference [1/2]

uint8_t rfalWakeUpConfig::reference

Reference to be used;RFAL_WUM_REFERENCE_AUTO sets it auto

◆ reference [2/2]

uint16_t rfalWakeUpConfig::reference

Reference to be used;RFAL_WUM_REFERENCE_AUTO sets it auto

◆ [1/2]

struct { ... } rfalWakeUpConfig::refWU

Reference obtained from PD|WU mode

◆ [2/2]

struct { ... } rfalWakeUpConfig::refWU

Reference obtained from PD|WU mode

◆ skipCal

bool rfalWakeUpConfig::skipCal

Do not preform calibration starting WU mode

◆ skipReCal

bool rfalWakeUpConfig::skipReCal

Do not preform recalibration during WU mode

◆ swTagDetect

bool rfalWakeUpConfig::swTagDetect

Use SW Tag Detection instead of HW Wake-Up mode

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

- st25r200/rfal_features.h
- st25r3911/rfal_features.h
- st25r3916/rfal_features.h
- st25r500/rfal_features.h
- st25r95/rfal_features.h

rfalWakeUpInfo Struct Reference

```
#include <rfal_features.h>
```

Data Fields

bool	irqWut
uint8_t	status

struct {

uint8_t **lastMeas**	
uint8_t **reference**	
uint8_t **calib**	
bool **irqWu**	
}	**I**
struct {	
uint8_t **lastMeas**	
uint8_t **reference**	
uint8_t **calib**	
bool **irqWu**	
}	**Q**
struct {	
uint8_t **lastMeas**	
uint8_t **reference**	
bool **irqWu**	
}	**indAmp**
struct {	
uint8_t **lastMeas**	
uint8_t **reference**	
bool **irqWu**	
}	**indPha**
struct {	
uint8_t **lastMeas**	

uint8_t **reference**	
bool **irqWu**	
}	**cap**
struct {	
uint8_t **lastMeas**	
uint16_t **reference**	
bool **irqWu**	
}	**indAmp**
struct {	
uint8_t **lastMeas**	
uint16_t **reference**	
bool **irqWu**	
}	**indPha**
struct {	
uint8_t **lastMeas**	
uint16_t **reference**	
bool **irqWu**	
}	**cap**
struct {	
uint8_t **lastMeas**	
uint8_t **reference**	
uint8_t **calib**	
bool **irqWu**	

}	**I**
struct {	
uint8_t **lastMeas**	
uint8_t **reference**	
uint8_t **calib**	
bool **irqWu**	
}	**Q**
struct {	
bool **irqWptFod**	
bool **irqWptStop**	
}	**WLC**
uint8_t **dummy**	

Detailed Description

RFAL Wake-Up Mode information

Field Documentation

◆ calib

uint8_t rfalWakeUpInfo::calib

Current calibration value

◆ [1/2]

struct { ... } rfalWakeUpInfo::cap

Capacitive

◆ [2/2]

struct { ... } rfalWakeUpInfo::cap

Capacitive

◆ dummy

uint8_t rfalWakeUpInfo::dummy

dummy member to avoid empty structs

◆ [1/2]

struct { ... } rfalWakeUpInfo::I

I channel Information

◆ [2/2]

struct { ... } rfalWakeUpInfo::I

I channel Information

◆ [1/2]

struct { ... } rfalWakeUpInfo::indAmp

Inductive Amplitude

◆ [2/2]

struct { ... } rfalWakeUpInfo::indAmp

Inductive Amplitude

◆ [1/2]

struct { ... } rfalWakeUpInfo::indPha

Inductive Phase

◆ [2/2]

struct { ... } rfalWakeUpInfo::indPha

Inductive Phase

◆ irqWptFod

bool rfalWakeUpInfo::irqWptFod

WLC WPT FOD detected

◆ irqWptStop

bool rfalWakeUpInfo::irqWptStop

WLC WPT Stop condition detected

◆ irqWu

bool rfalWakeUpInfo::irqWu

Amplitude WU IRQ received (cleared upon read)

Phase WU IRQ received (cleared upon read)

Capacitive WU IRQ received (cleared upon read)

◆ irqWut

bool rfalWakeUpInfo::irqWut

Wake-Up Timer IRQ received (cleared upon read)

Wake-Up timer and measurement event IRQ (cleared upon read)

◆ lastMeas

uint8_t rfalWakeUpInfo::lastMeas

Value of the latest measurement

◆ [1/2]

struct { ... } rfalWakeUpInfo::Q

Q channel Information

◆ [2/2]

struct { ... } rfalWakeUpInfo::Q

Q channel Information

◆ reference [1/2]

uint8_t rfalWakeUpInfo::reference

Current reference value

◆ reference [2/2]

uint16_t rfalWakeUpInfo::reference

Current reference value (TD format if SW TD enabled)

Current reference value

◆ status

uint8_t rfalWakeUpInfo::status

Wake-Up status

◆

struct { ... } rfalWakeUpInfo::WLC

WLC Wireless Power Transfer phase

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

- st25r200/rfal_features.h
- st25r3911/rfal_features.h
- st25r3916/rfal_features.h
- st25r500/rfal_features.h
- st25r95/rfal_features.h

---

rfalWumMeasChannel Struct Reference

```
#include <rfal_features.h>
```

Data Fields

bool	**enabled**
uint8_t	**delta**
uint8_t	**reference**
uint8_t	**threshold**
bool	**aaInclMeas**
rfalWumAAWeight	**aaWeight**

Detailed Description

RFAL Wake-Up channel configuration

Field Documentation

◆ aaInclMeas

bool rfalWumMeasChannel::aaInclMeas

When AutoAvg is enabled, include IRQ measurement

◆ aaWeight

rfalWumAAWeight rfalWumMeasChannel::aaWeight

When AutoAvg is enabled, last measure weight

◆ delta

uint8_t rfalWumMeasChannel::delta

Delta between the reference and measurement to wake-up

◆ enabled

bool rfalWumMeasChannel::enabled

Inductive Amplitude measurement enabled

◆ reference

uint8_t rfalWumMeasChannel::reference

Reference to be used;RFAL_WUM_REFERENCE_AUTO sets it auto

◆ threshold

uint8_t rfalWumMeasChannel::threshold

Wake-Up trigger treshold bitmask

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

- st25r200/rfal_features.h
- st25r500/rfal_features.h

st25r3916AatTuneParams Struct Reference

```
#include <st25r3916_aat.h>
```

Data Fields

uint8_t	**aat_a_min**
uint8_t	**aat_a_max**
uint8_t	**aat_a_start**
uint8_t	**aat_a_stepWidth**
uint8_t	**aat_b_min**
uint8_t	**aat_b_max**
uint8_t	**aat_b_start**
uint8_t	**aat_b_stepWidth**
uint8_t	**phaTarget**
uint8_t	**phaWeight**
uint8_t	**ampTarget**

uint8_t	**ampWeight**
bool	**doDynamicSteps**
uint8_t	**measureLimit**

Detailed Description

struct representing input parameters for the antenna tuning

Field Documentation

◆ aat_a_max

uint8_t st25r3916AatTuneParams::aat_a_max

max value of A cap

◆ aat_a_min

uint8_t st25r3916AatTuneParams::aat_a_min

min value of A cap

◆ aat_a_start

uint8_t st25r3916AatTuneParams::aat_a_start

start value of A cap

◆ aat_a_stepWidth

uint8_t st25r3916AatTuneParams::aat_a_stepWidth

increment stepWidth for A cap

◆ aat_b_max

uint8_t st25r3916AatTuneParams::aat_b_max

max value of B cap

◆ aat_b_min

uint8_t st25r3916AatTuneParams::aat_b_min

min value of B cap

◆ aat_b_start

uint8_t st25r3916AatTuneParams::aat_b_start

start value of B cap

◆ aat_b_stepWidth

uint8_t st25r3916AatTuneParams::aat_b_stepWidth

increment stepWidth for B cap

◆ ampTarget

uint8_t st25r3916AatTuneParams::ampTarget

target amplitude

◆ ampWeight

uint8_t st25r3916AatTuneParams::ampWeight

weight of target amplitude

◆ doDynamicSteps

bool st25r3916AatTuneParams::doDynamicSteps

dynamically reduce step size in algo

◆ measureLimit

uint8_t st25r3916AatTuneParams::measureLimit

max number of allowed steps/measurements

◆ phaTarget

uint8_t st25r3916AatTuneParams::phaTarget

target phase

◆ phaWeight

uint8_t st25r3916AatTuneParams::phaWeight

weight of target phase

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

- **st25r3916_aat.h**

st25r3916AatTuneResult Struct Reference

```
#include <st25r3916_aat.h>
```

Data Fields

uint8_t	**aat_a**
uint8_t	**aat_b**
uint8_t	**pha**
uint8_t	**amp**
uint16_t	**measureCnt**

Detailed Description

struct representing out parameters for the antenna tuning

Field Documentation

◆ aat_a

uint8_t st25r3916AatTuneResult::aat_a

serial cap after tuning

◆ aat_b

uint8_t st25r3916AatTuneResult::aat_b

parallel cap after tuning

◆ amp

uint8_t st25r3916AatTuneResult::amp

amplitude after tuning

◆ measureCnt

uint16_t st25r3916AatTuneResult::measureCnt

number of measures performed

◆ pha

uint8_t st25r3916AatTuneResult::pha

phase after tuning

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

- **st25r3916_aat.h**

Data Structure Index

I | R | S | T

I

iso15693StreamConfig

R

rfalAnalogConfigrfalAnalogConfigRegAddrMaskValrfalCdResrfalDlmaEntryrfalDlmaInforfalDpoEntryrfalDpoInforfalIso15693PhyConfig_trfalIsoDepActivationrfalIsoDepApduBufFormatrfalIsoDepApduTxRxParamrfalIsoDepAtsrfalIsoDepAtsParamrfalIsoDepAttribCmdrfalIsoDepAttribResrfalIsoDepAttribResParamrfalIsoDepBufFormatrfalIsoDepDevicerfalIsoDepInforfalIsoDepListenActvParamrfalIsoDepPpsReqrfalIsoDepPpsResrfalIsoDepRatsrfalIsoDepSParameterrfalIsoDepTxRxParamrfalLmConfPArfalLmConfPBrfalLmConfPFrfalNfcaListenDevicerfalNfcaSddReqrfalNfcaSddResrfalNfcaSelReqrfalNfcaSelResrfalNfcaSensResrfalNfcbListenDevicerfalNfcbSensbResrfalNfcbSensbResAppDatarfalNfcbSensbResProtocolInforfalNfcBufferrfalNfcDepActivationrfalNfcDepAtrParamrfalNfcDepAtrReqrfalNfcDepAtrResrfalNfcDepBufFormatrfalNfcDepConfigsrfalNfcDepDevicerfalNfcDepInforfalNfcDepListenActvParamrfalNfcDepPduBufFormatrfalNfcDepPduTxRxParamrfalNfcDepTargetParamrfalNfcDepTxRxParamrfalNfcDevicerfalNfcDiscoverParamrfalNfcfBlockListElemrfalNfcfListenDevicerfalNfcfPollDevicerfalNfcfSensfResrfalNfcfServBlockListParamrfalNfcPropCallbacksrfalNfcvGenericReqrfalNfcvGenericResrfalNfcvInventoryResrfalNfcvListenDevicerfalSt25tbListenDevicerfalT1TRidResrfalT4tCApduParamrfalT4tRApduParamrfalTransceiveContextrfalWakeUpConfigrfalWakeUpInforfalWumMeasChannel

S

st25r3911StreamConfigst25r3916AatTuneParamsst25r3916AatTuneResultst25r3916StreamConfigst25r500DpocrConfigst25r500DpocrInfost25r95SPIRxContext

T

Generated on Fri Mar 14 2025 11:45:11 for RFAL by doxygen 1.9.5

RFAL

Main Page

Modules

Files

Examples

Data Structures

Data Structure Index

Variables

b

c

d

e

f

g

h

i

l

m

n

o

p

q

r

s

t

u

v

w

Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- a -

- A : **rfalIsoDepActivation**
- aaInclMeas : **rfalWakeUpConfig**, **rfalWumMeasChannel**
- aat_a : **st25r3916AatTuneResult**
- aat_a_max : **st25r3916AatTuneParams**
- aat_a_min : **st25r3916AatTuneParams**
- aat_a_start : **st25r3916AatTuneParams**
- aat_a_stepWidth : **st25r3916AatTuneParams**
- aat_b : **st25r3916AatTuneResult**
- aat_b_max : **st25r3916AatTuneParams**
- aat_b_min : **st25r3916AatTuneParams**
- aat_b_start : **st25r3916AatTuneParams**
- aat_b_stepWidth : **st25r3916AatTuneParams**
- aaWeight : **rfalWakeUpConfig**, **rfalWumMeasChannel**
- activation : **rfalIsoDepDevice**, **rfalNfcDepDevice**
- additionalRespBytes : **st25r95SPIRxContext**

- addr : **rfalAnalogConfigRegAddrMaskVal**
- adjustCallback : **rfalDlmaInfo**, **rfalDpoInfo**
- AFI : **rfalNfcbSensbResAppData**
- amp : **st25r3916AatTuneResult**
- ampTarget : **st25r3916AatTuneParams**
- ampWeight : **st25r3916AatTuneParams**
- anticollisionInfo : **rfalNfcaSensRes**
- ap2pBR : **rfalNfcDiscoverParam**
- apdu : **rfalIsoDepApduBufFormat**
- appData : **rfalNfcbSensbRes**
- ATR_REQ : **rfalNfcDepActivation**
- ATR_REQLen : **rfalNfcDepActivation**
- ATR_RES : **rfalNfcDepActivation**
- ATR_RESLen : **rfalNfcDepActivation**
- ATS : **rfalIsoDepActivation**
- ATSLen : **rfalIsoDepActivation**
- ATTRIB : **rfalIsoDepActivation**
- ATTRIB_RES : **rfalIsoDepActivation**
- ATTRIB_RESLen : **rfalIsoDepActivation**
- ATTRIBLen : **rfalIsoDepActivation**
- autoAvg : **rfalWakeUpConfig**

RFAL

Variables

a

c

d

e

f

g

h

i

l

m

n

o

p

q

r

s

t

u

v

w

Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- b -

- B : **rfaIsoDepActivation**
- bcc : **rfaINfcaSddRes**, **rfaINfcaSelReq**
- blockList : **rfaINfcfServBlockListParam**
- blockNum : **rfaINfcfBlockListElem**
- BR : **rfaINfcDepAtrParam**
- br : **rfaINfcDepConfigs**
- BRC : **rfaINfcbSensbResProtocolInfo**
- BRi : **rfaINfcDepAtrReq**
- BRt : **rfaINfcDepAtrRes**
- brt : **rfaINfcDepTargetParam**
- BS : **rfaINfcDepAtrParam**
- bs : **rfaINfcDepConfigs**
- BSi : **rfaINfcDepAtrReq**
- BSt : **rfaINfcDepAtrRes**
- bst : **rfaINfcDepTargetParam**
- BufCRC : **st25r95SPIRxContext**

Variables

a

b

d

e

f

g

h

i

l

m

n

o

p

q

r

s

t

u

v

w

Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- c -

- calib : **rfalWakeUpInfo**
- cap : **rfalWakeUpConfig**, **rfalWakeUpInfo**
- cApduBuf : **rfalT4tCApduParam**
- cApduLen : **rfalT4tCApduParam**
- chipID : **rfalSt25tbListenDevice**
- CLA : **rfalT4tCApduParam**
- cmd : **rfalIsoDepAttribCmd**
- CMD : **rfalIsoDepRats**
- cmd : **rfalNfcbSensbRes**
- CMD : **rfalNfcfSensfRes**, **rfalNfcvGenericReq**
- CMD1 : **rfalNfcDepAtrReq**, **rfalNfcDepAtrRes**
- CMD2 : **rfalNfcDepAtrReq**, **rfalNfcDepAtrRes**
- coding : **rfalIso15693PhyConfig_t**
- commMode : **rfalNfcDepAtrParam**, **rfalNfcDepConfigs**, **rfalNfcDepTargetParam**
- compMode : **rfalNfcDiscoverParam**
- conf : **rfalNfcfBlockListElem**
- crc : **rfalNfcvInventoryRes**
- CRC_B : **rfalNfcbSensbResAppData**
- currentDres : **st25r500DpocrInfo**
- currentElecCurrent : **st25r500DpocrInfo**
- currentEntryIdx : **st25r500DpocrInfo**
- currentRege : **st25r500DpocrInfo**
- currThreshold : **st25r500DpocrConfig**

RFAL

Main Page

Modules

Files

Examples

Data Structures

Data Structure Index

Variables

a

b

c

e

f

g

h

i

l

m

n

o

p

q

r

s

t

u

v

w

Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- d -

- data : **rfalNfcvGenericReq**, **rfalNfcvGenericRes**
- dec : **rfalDlmaEntry**, **rfalDpoEntry**
- delCal : **rfalWakeUpConfig**
- delRef : **rfalWakeUpConfig**
- delta : **rfalWakeUpConfig**, **rfalWumMeasChannel**
- detected : **rfalCdRes**
- detType : **rfalCdRes**
- dev : **rfalNfcDevice**
- devLimit : **rfalNfcDiscoverParam**
- dFWT : **rfalIsoDepApduTxRxParam**, **rfalIsoDepInfo**, **rfalIsoDepTxRxParam**
- dFwt : **rfalNfcDepConfigs**
- dFWT : **rfalNfcDepInfo**, **rfalNfcDepPduTxRxParam**, **rfalNfcDepTxRxParam**
- DID : **rfalIsoDepApduTxRxParam**, **rfalIsoDepInfo**, **rfalIsoDepTxRxParam**, **rfalNfcDepAtrParam**, **rfalNfcDepAtrReq**, **rfalNfcDepAtrRes**
- did : **rfalNfcDepConfigs**
- DID : **rfalNfcDepInfo**, **rfalNfcDepPduTxRxParam**, **rfalNfcDepTxRxParam**
- didSupport : **rfalIsoDepAtsParam**
- din : **iso15693StreamConfig**, **st25r3911StreamConfig**, **st25r3916StreamConfig**
- doDynamicSteps : **st25r3916AatTuneParams**
- dout : **iso15693StreamConfig**, **st25r3911StreamConfig**, **st25r3916StreamConfig**
- DRI : **rfalIsoDepInfo**, **rfalNfcDepInfo**
- DSFID : **rfalNfcvInventoryRes**
- DSI : **rfalIsoDepInfo**, **rfalNfcDepInfo**
- dummy : **rfalWakeUpInfo**

RFAL

Main Page

Modules

Files

Examples

Data Structures

Data Structure Index

Variables

a

b

c

d

f

g

h

i

l

m

n

o

p

q

r

s

t

u

v

w

Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- e -

- enabled : **rfalDlmaInfo**, **rfalDpoInfo**, **rfalWakeUpConfig**, **rfalWumMeasChannel**, **st25r500DpocrConfig**

Data Structure Index

Variables

a

b

c

d

e

g

h

i

l

m

n

o

p

q

r

s

t

u

v

Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- f -

- flags : **rfalTransceiveContext**
- fracDelta : **rfalWakeUpConfig**
- FS : **rfalNfcDepInfo**
- fsci : **rfalIsoDepAtsParam**
- FsciProType : **rfalNfcbSensbResProtocolInfo**
- FSx : **rfalIsoDepApduTxRxParam**, **rfalIsoDepInfo**, **rfalIsoDepTxRxParam**, **rfalNfcDepPduTxRxParam**, **rfalNfcDepTxRxParam**
- FSxI : **rfalIsoDepInfo**
- fwi : **rfalIsoDepAtsParam**
- FWI : **rfalIsoDepInfo**
- FwiAdcFo : **rfalNfcbSensbResProtocolInfo**
- FWT : **rfalIsoDepApduTxRxParam**, **rfalIsoDepInfo**, **rfalIsoDepTxRxParam**
- fwt : **rfalNfcDepConfigs**
- FWT : **rfalNfcDepInfo**, **rfalNfcDepPduTxRxParam**, **rfalNfcDepTxRxParam**
- fwt : **rfalTransceiveContext**

Variables

a

b

c

d

e

f

h

i

l

m

n

o

p

q

r

s

t

u

v

w

Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- g -

- GB : **rfalNfcDepAtrParam**
- gb : **rfalNfcDepConfigs**
- GB : **rfalNfcDiscoverParam**
- GBi : **rfalNfcDepAtrReq**
- GBLen : **rfalNfcDepAtrParam**
- gbLen : **rfalNfcDepConfigs**
- GBLen : **rfalNfcDepInfo**, **rfalNfcDiscoverParam**
- GBt : **rfalNfcDepAtrRes**, **rfalNfcDepTargetParam**
- GBtLen : **rfalNfcDepTargetParam**

---

c

d

e

f

g

i

l

m

n

o

p

q

r

s

t

u

v

w

Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- h -

- HB : **rfaIIsoDepAts**
- hb : **rfaIIsoDepAtsParam**
- hbLen : **rfaIIsoDepAtsParam**
- HLInfo : **rfaIIsoDepAttribCmd**, **rfaIIsoDepAttribRes**, **rfaIIsoDepAttribResParam**

- HLInfoLen : **rfaIsoDepAttribResParam**
- hr0 : **rfalT1TRidRes**
- hr1 : **rfalT1TRidRes**

Generated on Fri Mar 14 2025 11:45:11 for RFAL by doxygen 1.9.5

RFAL

Main Page

Modules

Files

Examples

Data Structures

Data Structure Index

Variables

a

b

c

d

e

f

g

h

l

m

n

o

p

q

r

s

t

u

v

w

Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- i -

- I : **rfalWakeUpConfig**, **rfalWakeUpInfo**
- id : **rfalAnalogConfig**
- inc : **rfalDlmaEntry**, **rfalDpoEntry**
- indAmp : **rfalWakeUpConfig**, **rfalWakeUpInfo**
- indPha : **rfalWakeUpConfig**, **rfalWakeUpInfo**
- inf : **rfalIsoDepBufFormat**, **rfalNfcDepBufFormat**
- info : **rfalIsoDepDevice**, **rfalNfcDepDevice**
- Initiator : **rfalNfcDepActivation**
- inListen : **st25r95SPIRxContext**
- INS : **rfalT4tCApduParam**
- InvRes : **rfalNfcvListenDevice**
- irqTout : **rfalWakeUpConfig**
- irqWptFod : **rfalWakeUpInfo**
- irqWptStop : **rfalWakeUpInfo**

- irqWu : **rfalWakeUpInfo**
- irqWut : **rfalWakeUpInfo**
- isDeselected : **rfalSt25tbListenDevice**
- isoDep : **rfalNfcDevice**
- isoDepBuf : **rfalNfcBuffer**
- isoDepDev : **rfalIsoDepListenActvParam**
- isoDepFS : **rfalNfcDiscoverParam**
- isRxChaining : **rfalIsoDepListenActvParam**, **rfalIsoDepTxRxParam**, **rfalNfcDepListenActvParam**, **rfalNfcDepTxRxParam**
- isSleep : **rfalNfcaListenDevice**, **rfalNfcbListenDevice**, **rfalNfcvListenDevice**
- isTxChaining : **rfalIsoDepTxRxParam**, **rfalNfcDepTxRxParam**

RFAL

Main Page

Modules

Files

Examples

Data Structures

Data Structure Index

Variables

a

b

c

d

e

f

g

h

i

m

n

o

p

q

r

s

t

u

v

w

Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- l -

- lastMeas : **rfalWakeUpInfo**
- Lc : **rfalT4tCApduParam**
- LcFlag : **rfalT4tCApduParam**
- Le : **rfalT4tCApduParam**
- LeFlag : **rfalT4tCApduParam**

- length : **rfaIsoDepSParameter**
- lmConfigPA : **rfaINfcDiscoverParam**
- lmConfigPF : **rfaINfcDiscoverParam**
- LmState : **st25r95SPIRxContext**
- LR : **rfaINfcDepAtrParam**
- lr : **rfaINfcDepConfigs**
- LR : **rfaINfcDepInfo**

---

f

g

h

i

l

n

o

p

q

r

s

t

u

v

w

Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- m -

- mask : **rfalAnalogConfigRegAddrMaskVal**
- maxBR : **rfalNfcDiscoverParam**
- maxDres : **st25r500DpocrConfig**
- maxRege : **st25r500DpocrConfig**
- MBL : **rfalIsoDepInfo**
- mbli : **rfalIsoDepAttribResParam**
- mbliDid : **rfalIsoDepAttribRes**
- measDur : **rfalWakeUpConfig**
- measFil : **rfalWakeUpConfig**

- measureCallback : **rfalDlmaInfo**, **rfalDpoInfo**
- measureCnt : **st25r3916AatTuneResult**
- measureLimit : **st25r3916AatTuneParams**
- minDres : **st25r500DpocrConfig**
- minRege : **st25r500DpocrConfig**
- modRes : **rfalDlmaEntry**
- MRTIcheck : **rfalNfcfSensfRes**
- MRTIupdate : **rfalNfcfSensfRes**

Generated on Fri Mar 14 2025 11:45:11 for RFAL by doxygen 1.9.5

RFAL

Main Page

Modules

Files

Examples

Data Structures

Data Structure Index

Variables

a

b

c

d

e

Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- n -

- NAD : **rfalIsoDepInfo**, **rfalNfcDepAtrParam**
- nad : **rfalNfcDepConfigs**
- NAD : **rfalNfcDepInfo**
- nfca : **rfalNfcDevice**
- nfcb : **rfalNfcDevice**
- nfcDep : **rfalNfcDevice**
- nfcDepBuf : **rfalNfcBuffer**
- nfcDepDev : **rfalNfcDepListenActvParam**
- nfcDepLR : **rfalNfcDiscoverParam**

- nfcf : **rfalNfcDevice**
- nfcfBR : **rfalNfcDiscoverParam**
- nfcid : **rfalLmConfPA**, **rfalNfcDepAtrParam**, **rfalNfcDepConfigs**, **rfalNfcDevice**
- nfcid0 : **rfalIsoDepAttribCmd**, **rfalNfcbSensbRes**
- nfcId1 : **rfalNfcaListenDevice**
- nfcid1 : **rfalNfcaSddRes**, **rfalNfcaSelReq**
- nfcId1Len : **rfalNfcaListenDevice**
- NFCID2 : **rfalNfcfPollDevice**, **rfalNfcfSensfRes**
- NFCID3 : **rfalNfcDepAtrReq**, **rfalNfcDepAtrRes**
- nfcid3 : **rfalNfcDepTargetParam**, **rfalNfcDiscoverParam**
- nfcidLen : **rfalLmConfPA**, **rfalNfcDepAtrParam**, **rfalNfcDepConfigs**, **rfalNfcDevice**
- NFCIP1 : **st25r95SPIRxContext**
- NFCIP1_SoD : **st25r95SPIRxContext**
- nfcv : **rfalNfcDevice**
- notifyCb : **rfalNfcDiscoverParam**
- num : **rfalAnalogConfig**
- numApps : **rfalNfcbSensbResAppData**
- numBlock : **rfalNfcfServBlockListParam**
- numEntries : **st25r500DpocrConfig**
- numServ : **rfalNfcfServBlockListParam**

RFAL

Main Page

Modules

Files

Examples

Data Structures

Data Structure Index

Variables

a

b

c

d

e

f

g

h

i

l

m

n

p

q

r

s

t

u

v

w

Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- o -

- oper : **rfalNfcDepConfigs**
- operParam : **rfalNfcDepAtrParam**, **rfalNfcDepTargetParam**
- ourFSx : **rfalIsoDepApduTxRxParam**, **rfalIsoDepTxRxParam**

Generated on Fri Mar 14 2025 11:45:11 for RFAL by doxygen 1.9.5

RFAL

f

g

h

i

l

m

n

o

q

r

s

t

u

v

w

Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- p -

- P1 : **rfalT4tCApduParam**
- P2 : **rfalT4tCApduParam**
- p2pNfcaPrio : **rfalNfcDiscoverParam**
- PAD0 : **rfalNfcfSensfRes**
- PAD1 : **rfalNfcfSensfRes**
- PAD2 : **rfalNfcfSensfRes**
- Param : **rfalIsoDepAttribCmd**
- PARAM : **rfalIsoDepRats**
- PARAM1 : **rfalIsoDepAttribCmd**

- PARAM2 : **rfaIIsoDepAttribCmd**
- PARAM3 : **rfaIIsoDepAttribCmd**
- PARAM4 : **rfaIIsoDepAttribCmd**
- payload : **rfaINfcvGenericReq**
- pdu : **rfaINfcDepPduBufFormat**
- period : **rfaIWakeUpConfig**
- pha : **st25r3916AatTuneResult**
- phaTarget : **st25r3916AatTuneParams**
- phaWeight : **st25r3916AatTuneParams**
- platformInfo : **rfaINfcaSensRes**
- PPi : **rfaINfcDepAtrReq**
- PPS0 : **rfaIIsoDepPpsReq**
- PPS1 : **rfaIIsoDepPpsReq**
- PPSS : **rfaIIsoDepPpsReq**, **rfaIIsoDepPpsRes**
- PPt : **rfaINfcDepAtrRes**
- ppt : **rfaINfcDepTargetParam**
- prologue : **rfaIIsoDepApduBufFormat**, **rfaIIsoDepBufFormat**, **rfaINfcDepBufFormat**, **rfaINfcDepPduBufFormat**
- propNfc : **rfaINfcDiscoverParam**
- protInfo : **rfaINfcbSensbRes**
- proto : **rfaINfcDevice**
- protocol : **st25r95SPIRxContext**

RFAL

Main Page

Modules

Files

Examples

Data Structures

Data Structure Index

Variables

a

b

c

d

e

f

g

h

i

l

m

n

o

p

r

s

t

u

v

w

Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- q -

- Q : **rfalWakeUpConfig**, **rfalWakeUpInfo**

Generated on Fri Mar 14 2025 11:45:11 for RFAL by doxygen 1.9.5

RFAL

Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- r -

- rApduBodyLen : **rfalT4tRApduParam**
- rApduBuf : **rfalT4tRApduParam**
- RATS : **rfalIsoDepActivation**
- rcvdLen : **rfalT4tRApduParam**
- RD : **rfalNfcfSensfRes**
- refDelay : **rfalWakeUpConfig**
- reference : **rfalWakeUpConfig**, **rfalWakeUpInfo**, **rfalWumMeasChannel**
- refMeasurement : **rfalDlmaInfo**, **rfalDpoInfo**
- refWU : **rfalWakeUpConfig**
- regeStepSize : **st25r500DpocrConfig**
- regSet : **rfalAnalogConfig**
- report_period_length : **iso15693StreamConfig**, **st25r3911StreamConfig**, **st25r3916StreamConfig**

- REQ_FLAG : **rfalNfcvGenericReq**
- RES_FLAG : **rfalNfcvGenericRes**, **rfalNfcvInventoryRes**
- retCode : **st25r95SPIRxContext**
- rfalNfcpGetActivationStatus : **rfalNfcPropCallbacks**
- rfalNfcpPollerGetCollisionResolutionStatus : **rfalNfcPropCallbacks**
- rfalNfcpPollerInitialize : **rfalNfcPropCallbacks**
- rfalNfcpPollerStartCollisionResolution : **rfalNfcPropCallbacks**
- rfalNfcpPollerTechnologyDetection : **rfalNfcPropCallbacks**
- rfalNfcpStartActivation : **rfalNfcPropCallbacks**
- rfBuf : **rfalNfcBuffer**
- rfInterface : **rfalNfcDevice**
- rfoRes : **rfalDpoEntry**
- rmvCRC : **st25r95SPIRxContext**
- role : **rfalNfcDepConfigs**
- RsA : **t_st25r3916Regs**
- RsB : **t_st25r3916Regs**
- rxBuf : **rfalIsoDepApduTxRxParam**, **rfalIsoDepListenActvParam**, **rfalIsoDepTxRxParam**, **rfalNfcDepListenActvParam**, **rfalNfcDepPduTxRxParam**, **rfalNfcDepTxRxParam**, **rfalTransceiveContext**, **st25r95SPIRxContext**
- rxBufLen : **rfalTransceiveContext**, **st25r95SPIRxContext**
- rxLen : **rfalIsoDepApduTxRxParam**, **rfalIsoDepListenActvParam**, **rfalIsoDepTxRxParam**, **rfalNfcDepListenActvParam**, **rfalNfcDepPduTxRxParam**, **rfalNfcDepTxRxParam**
- rxRcvdLen : **rfalTransceiveContext**, **st25r95SPIRxContext**

RFAL

Main Page

Modules

Files

Examples

Data Structures

Data Structure Index

Variables

a

b

c

d

e

f

g

h

i

l

m

n

o

p

q

r

t

u

v

w

Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- s -

- sak : **rfalNfcaSelRes**
- SC : **rfalLmConfPF**
- SEL_RES : **rfalLmConfPA**
- selCmd : **rfalNfcaSddReq**, **rfalNfcaSelReq**
- selPar : **rfalNfcaSddReq**, **rfalNfcaSelReq**
- selRes : **rfalNfcaListenDevice**
- SENS_RES : **rfalLmConfPA**
- SENSB_RES : **rfalLmConfPB**
- sensbRes : **rfalNfcbListenDevice**
- sensbResLen : **rfalNfcbListenDevice**
- SENSF_RES : **rfalLmConfPF**
- sensfRes : **rfalNfcfListenDevice**
- sensfResLen : **rfalNfcfListenDevice**
- sensRes : **rfalNfcaListenDevice**
- servList : **rfalNfcfServBlockListParam**
- sfgi : **rfalIsoDepAtsParam**
- SFGI : **rfalIsoDepInfo**, **rfalNfcbSensbResProtocolInfo**
- SFGT : **rfalIsoDepInfo**
- skipCal : **rfalWakeUpConfig**
- skipReCal : **rfalWakeUpConfig**
- speedMode : **rfalIso15693PhyConfig_t**
- st25tb : **rfalNfcDevice**
- status : **rfalWakeUpInfo**
- statusWord : **rfalT4tRApduParam**
- supAdFt : **rfalIsoDepInfo**
- supDID : **rfalIsoDepInfo**
- supNAD : **rfalIsoDepInfo**
- swTagDetect : **rfalWakeUpConfig**

RFAL

Variables

a

b

c

d

e

f

g

h

i

l

m

Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- t -

- T0 : **rfalIsoDepAts**
- TA : **rfalIsoDepAts**
- ta : **rfalIsoDepAtsParam**
- tableEntries : **rfalDlmaInfo**, **rfalDpoInfo**
- tableEntry : **rfalDlmaInfo**, **rfalDpoInfo**
- tableUpThresholds : **st25r500DpocrConfig**
- tag : **rfalIsoDepSParameter**
- Target : **rfalNfcDepActivation**
- target : **st25r500DpocrConfig**
- TB : **rfalIsoDepAts**
- TC : **rfalIsoDepAts**
- techs2Bail : **rfalNfcDiscoverParam**
- techs2Find : **rfalNfcDiscoverParam**
- threshold : **rfalWumMeasChannel**
- TL : **rfalIsoDepAts**
- tmpBuf : **rfalIsoDepApduTxRxParam**, **rfalNfcDepPduTxRxParam**
- TO : **rfalNfcDepAtrRes**
- to : **rfalNfcDepConfigs**, **rfalNfcDepTargetParam**

- totalDuration : **rfalNfcDiscoverParam**
- txBuf : **rfalIsoDepApduTxRxParam**, **rfalIsoDepTxRxParam**, **rfalNfcDepPduTxRxParam**, **rfalNfcDepTxRxParam**, **rfalTransceiveContext**
- txBufLen : **rfalIsoDepApduTxRxParam**, **rfalIsoDepTxRxParam**, **rfalNfcDepPduTxRxParam**, **rfalNfcDepTxRxParam**, **rfalTransceiveContext**
- type : **rfalNfcaListenDevice**, **rfalNfcDevice**

Generated on Fri Mar 14 2025 11:45:11 for RFAL by doxygen 1.9.5

RFAL

Main Page

Modules

Files

Examples

Data Structures

Data Structure Index

Variables

a

b

c

d

e

f

g

h

i

l

m

n

o

p

q

r

s

t

v

w

Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- u -

- UID : **rfalNfcvGenericReq**, **rfalNfcvInventoryRes**, **rfalSt25tbListenDevice**
- uid : **rfalT1TRidRes**
- unmodRes : **rfalDlmaEntry**
- useBPSK : **iso15693StreamConfig**, **st25r3911StreamConfig**, **st25r3916StreamConfig**

Main Page

Modules

Files

Examples

Data Structures

Data Structure Index

Variables

a

b

c

d

e

f

g

h

i

l

m

n

o

p

q

r

s

t

u

w

Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- v -

- val : **rfalAnalogConfigRegAddrMaskVal**
- value : **rfalIsoDepSParameter**

RFAL

Variables

a

b

c

d

e

f

g

h

i

l

m

n

o

p

q

r

s

t

u

v

Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- w -

- wakeupConfig : **rfalNfcDiscoverParam**
- wakeupConfigDefault : **rfalNfcDiscoverParam**
- wakeupEnabled : **rfalNfcDiscoverParam**
- wakeupNPolls : **rfalNfcDiscoverParam**
- wakeupPollBefore : **rfalNfcDiscoverParam**
- WLC : **rfalWakeUpInfo**
- WT : **rfalNfcDepInfo**

RFAL

f

g

h

i

l

m

n

o

p

q

r

s

t

u

v

w

- a -

- A : **rfaIIsoDepActivation**
- aaInclMeas : **rfaIWakeUpConfig**, **rfaIWumMeasChannel**
- aat_a : **st25r3916AatTuneResult**
- aat_a_max : **st25r3916AatTuneParams**
- aat_a_min : **st25r3916AatTuneParams**
- aat_a_start : **st25r3916AatTuneParams**
- aat_a_stepWidth : **st25r3916AatTuneParams**
- aat_b : **st25r3916AatTuneResult**
- aat_b_max : **st25r3916AatTuneParams**

- aat_b_min : **st25r3916AatTuneParams**
- aat_b_start : **st25r3916AatTuneParams**
- aat_b_stepWidth : **st25r3916AatTuneParams**
- aaWeight : **rfalWakeUpConfig**, **rfalWumMeasChannel**
- activation : **rfalIsoDepDevice**, **rfalNfcDepDevice**
- additionalRespBytes : **st25r95SPIRxContext**
- addr : **rfalAnalogConfigRegAddrMaskVal**
- adjustCallback : **rfalDImaInfo**, **rfalDpoInfo**
- AFI : **rfalNfcbSensbResAppData**
- amp : **st25r3916AatTuneResult**
- ampTarget : **st25r3916AatTuneParams**
- ampWeight : **st25r3916AatTuneParams**
- anticollisionInfo : **rfalNfcaSensRes**
- ap2pBR : **rfalNfcDiscoverParam**
- apdu : **rfalIsoDepApduBufFormat**
- appData : **rfalNfcbSensbRes**
- ATR_REQ : **rfalNfcDepActivation**
- ATR_REQLen : **rfalNfcDepActivation**
- ATR_RES : **rfalNfcDepActivation**
- ATR_RESLen : **rfalNfcDepActivation**
- ATS : **rfalIsoDepActivation**
- ATSLen : **rfalIsoDepActivation**
- ATTRIB : **rfalIsoDepActivation**
- ATTRIB_RES : **rfalIsoDepActivation**
- ATTRIB_RESLen : **rfalIsoDepActivation**
- ATTRIBLen : **rfalIsoDepActivation**
- autoAvg : **rfalWakeUpConfig**

Files

Examples

Data Structures

Data Structure Index

All

a

c

d

e

f

g

h

i

l

m

n

o

p

q

r

s

t

u

v

w

- b -

- B : **rfaIsoDepActivation**
- bcc : **rfaINfcaSddRes**, **rfaINfcaSelReq**
- blockList : **rfaINfcfServBlockListParam**
- blockNum : **rfaINfcfBlockListElem**
- BR : **rfaINfcDepAtrParam**
- br : **rfaINfcDepConfigs**
- BRC : **rfaINfcbSensbResProtocolInfo**
- BRi : **rfaINfcDepAtrReq**
- BRt : **rfaINfcDepAtrRes**
- brt : **rfaINfcDepTargetParam**
- BS : **rfaINfcDepAtrParam**
- bs : **rfaINfcDepConfigs**
- BSi : **rfaINfcDepAtrReq**
- BSt : **rfaINfcDepAtrRes**
- bst : **rfaINfcDepTargetParam**
- BufCRC : **st25r95SPIRxContext**

Files

Examples

Data Structures

Data Structure Index

All

a

b

d

e

f

g

h

i

l

m

n

o

p

q

r

s

t

u

v

w

- c -

- calib : **rfalWakeUpInfo**
- cap : **rfalWakeUpConfig**, **rfalWakeUpInfo**
- cApduBuf : **rfalT4tCApduParam**
- cApduLen : **rfalT4tCApduParam**
- chipID : **rfalSt25tbListenDevice**
- CLA : **rfalT4tCApduParam**
- cmd : **rfalIsoDepAttribCmd**
- CMD : **rfalIsoDepRats**
- cmd : **rfalNfcbSensbRes**
- CMD : **rfalNfcfSensfRes**, **rfalNfcvGenericReq**
- CMD1 : **rfalNfcDepAtrReq**, **rfalNfcDepAtrRes**
- CMD2 : **rfalNfcDepAtrReq**, **rfalNfcDepAtrRes**
- coding : **rfalIso15693PhyConfig_t**
- commMode : **rfalNfcDepAtrParam**, **rfalNfcDepConfigs**, **rfalNfcDepTargetParam**
- compMode : **rfalNfcDiscoverParam**
- conf : **rfalNfcfBlockListElem**
- crc : **rfalNfcvInventoryRes**
- CRC_B : **rfalNfcbSensbResAppData**
- currentDres : **st25r500DpocrInfo**
- currentElecCurrent : **st25r500DpocrInfo**
- currentEntryIdx : **st25r500DpocrInfo**
- currentRege : **st25r500DpocrInfo**
- currThreshold : **st25r500DpocrConfig**

RFAL

Main Page

Modules

Files

Examples

Data Structures

Data Structure Index

All

a

b

c

e

f

g

h

i

l

m

n

o

p

q

r

s

t

u

v

w

- d -

- data : **rfalNfcvGenericReq**, **rfalNfcvGenericRes**
- dec : **rfalDlmaEntry**, **rfalDpoEntry**
- delCal : **rfalWakeUpConfig**
- delRef : **rfalWakeUpConfig**
- delta : **rfalWakeUpConfig**, **rfalWumMeasChannel**
- detected : **rfalCdRes**
- detType : **rfalCdRes**
- dev : **rfalNfcDevice**
- devLimit : **rfalNfcDiscoverParam**
- dFWT : **rfalIsoDepApduTxRxParam**, **rfalIsoDepInfo**, **rfalIsoDepTxRxParam**
- dFwt : **rfalNfcDepConfigs**
- dFWT : **rfalNfcDepInfo**, **rfalNfcDepPduTxRxParam**, **rfalNfcDepTxRxParam**
- DID : **rfalIsoDepApduTxRxParam**, **rfalIsoDepInfo**, **rfalIsoDepTxRxParam**, **rfalNfcDepAtrParam**, **rfalNfcDepAtrReq**, **rfalNfcDepAtrRes**
- did : **rfalNfcDepConfigs**
- DID : **rfalNfcDepInfo**, **rfalNfcDepPduTxRxParam**, **rfalNfcDepTxRxParam**
- didSupport : **rfalIsoDepAtsParam**
- din : **iso15693StreamConfig**, **st25r3911StreamConfig**, **st25r3916StreamConfig**

- doDynamicSteps : **st25r3916AatTuneParams**
- dout : **iso15693StreamConfig**, **st25r3911StreamConfig**, **st25r3916StreamConfig**
- DRI : **rfalIsoDepInfo**, **rfalNfcDepInfo**
- DSFID : **rfalNfcvInventoryRes**
- DSI : **rfalIsoDepInfo**, **rfalNfcDepInfo**
- dummy : **rfalWakeUpInfo**

Generated on Fri Mar 14 2025 11:45:11 for RFAL by doxygen 1.9.5

RFAL

Main Page

Modules

Files

Examples

Data Structures

Data Structure Index

All

a

b

c

d

f

g

h

i

l

m

n

o

p

q

r

s

t

u

v

w

- e -

- enabled : **rfalDIMAInfo**, **rfalDPOInfo**, **rfalWakeUpConfig**, **rfalWUMMeasChannel**, **st25r500DpocrConfig**

Modules

Files

Examples

Data Structures

Data Structure Index

All

a

b

c

d

e

g

h

i

l

m

n

o

p

q

r

s

t

u

v

w

- f -

- flags : **rfalTransceiveContext**
- fracDelta : **rfalWakeUpConfig**
- FS : **rfalNfcDepInfo**
- fsci : **rfalIsoDepAtsParam**
- FsciProType : **rfalNfcbSensbResProtocolInfo**
- FSx : **rfalIsoDepApduTxRxParam**, **rfalIsoDepInfo**, **rfalIsoDepTxRxParam**, **rfalNfcDepPduTxRxParam**, **rfalNfcDepTxRxParam**
- FSxI : **rfalIsoDepInfo**
- fwi : **rfalIsoDepAtsParam**
- FWI : **rfalIsoDepInfo**
- FwiAdcFo : **rfalNfcbSensbResProtocolInfo**
- FWT : **rfalIsoDepApduTxRxParam**, **rfalIsoDepInfo**, **rfalIsoDepTxRxParam**
- fwt : **rfalNfcDepConfigs**
- FWT : **rfalNfcDepInfo**, **rfalNfcDepPduTxRxParam**, **rfalNfcDepTxRxParam**
- fwt : **rfalTransceiveContext**

Files

Examples

Data Structures

Data Structure Index

All

a

b

c

d

e

f

h

i

l

m

n

o

p

q

r

s

t

u

v

w

- g -

- GB : **rfalNfcDepAtrParam**
- gb : **rfalNfcDepConfigs**
- GB : **rfalNfcDiscoverParam**
- GBi : **rfalNfcDepAtrReq**
- GBLen : **rfalNfcDepAtrParam**
- gbLen : **rfalNfcDepConfigs**
- GBLen : **rfalNfcDepInfo**, **rfalNfcDiscoverParam**
- GBt : **rfalNfcDepAtrRes**, **rfalNfcDepTargetParam**
- GBtLen : **rfalNfcDepTargetParam**

RFAL

All

a

b

c

d

e

f

g

i

l

m

n

o

p

q

r

s

t

u

v

w

- h -

- HB : **rfaIIsoDepAts**
- hb : **rfaIIsoDepAtsParam**
- hbLen : **rfaIIsoDepAtsParam**
- HLInfo : **rfaIIsoDepAttribCmd**, **rfaIIsoDepAttribRes**, **rfaIIsoDepAttribResParam**
- HLInfoLen : **rfaIIsoDepAttribResParam**
- hr0 : **rfaIT1TRidRes**
- hr1 : **rfaIT1TRidRes**

RFAL

d

e

f

g

h

l

m

n

o

p

q

r

s

t

u

v

w

- i -

- I : **rfalWakeUpConfig**, **rfalWakeUpInfo**
- id : **rfalAnalogConfig**
- inc : **rfalDlmaEntry**, **rfalDpoEntry**
- indAmp : **rfalWakeUpConfig**, **rfalWakeUpInfo**
- indPha : **rfalWakeUpConfig**, **rfalWakeUpInfo**

- inf : **rfalIsoDepBufFormat**, **rfalNfcDepBufFormat**
- info : **rfalIsoDepDevice**, **rfalNfcDepDevice**
- Initiator : **rfalNfcDepActivation**
- inListen : **st25r95SPIRxContext**
- INS : **rfalT4tCApduParam**
- InvRes : **rfalNfcvListenDevice**
- irqTout : **rfalWakeUpConfig**
- irqWptFod : **rfalWakeUpInfo**
- irqWptStop : **rfalWakeUpInfo**
- irqWu : **rfalWakeUpInfo**
- irqWut : **rfalWakeUpInfo**
- isDeselected : **rfalSt25tbListenDevice**
- isoDep : **rfalNfcDevice**
- isoDepBuf : **rfalNfcBuffer**
- isoDepDev : **rfalIsoDepListenActvParam**
- isoDepFS : **rfalNfcDiscoverParam**
- isRxChaining : **rfalIsoDepListenActvParam**, **rfalIsoDepTxRxParam**, **rfalNfcDepListenActvParam**, **rfalNfcDepTxRxParam**
- isSleep : **rfalNfcaListenDevice**, **rfalNfcbListenDevice**, **rfalNfcvListenDevice**
- isTxChaining : **rfalIsoDepTxRxParam**, **rfalNfcDepTxRxParam**

RFAL

All

a

b

c

d

e

f

g

h

i

m

n

o

p

q

r

s

t

u

v

w

- l -

- lastMeas : **rfalWakeUpInfo**
- Lc : **rfalT4tCApduParam**
- LcFlag : **rfalT4tCApduParam**
- Le : **rfalT4tCApduParam**
- LeFlag : **rfalT4tCApduParam**
- length : **rfalIsoDepSParameter**
- lmConfigPA : **rfalNfcDiscoverParam**
- lmConfigPF : **rfalNfcDiscoverParam**
- LmState : **st25r95SPIRxContext**
- LR : **rfalNfcDepAtrParam**
- lr : **rfalNfcDepConfigs**
- LR : **rfalNfcDepInfo**

Generated on Fri Mar 14 2025 11:45:12 for RFAL by doxygen 1.9.5

RFAL

a

b

c

d

e

f

g

h

i

l

n

o

p

q

r

s

t

u

v

w

- m -

- mask : **rfalAnalogConfigRegAddrMaskVal**
- maxBR : **rfalNfcDiscoverParam**
- maxDres : **st25r500DpocrConfig**
- maxRege : **st25r500DpocrConfig**
- MBL : **rfalIsoDepInfo**
- mbli : **rfalIsoDepAttribResParam**
- mbliDid : **rfalIsoDepAttribRes**
- measDur : **rfalWakeUpConfig**
- measFil : **rfalWakeUpConfig**
- measureCallback : **rfalDlmaInfo**, **rfalDpoInfo**
- measureCnt : **st25r3916AatTuneResult**
- measureLimit : **st25r3916AatTuneParams**
- minDres : **st25r500DpocrConfig**
- minRege : **st25r500DpocrConfig**
- modRes : **rfalDlmaEntry**
- MRTIcheck : **rfalNfcfSensfRes**
- MRTIupdate : **rfalNfcfSensfRes**

RFAL

a

b

c

d

e

f

g

h

i

l

m

o

p

q

r

s

t

u

v

w

- n -

- NAD : **rfalIsoDepInfo**, **rfalNfcDepAtrParam**
- nad : **rfalNfcDepConfigs**
- NAD : **rfalNfcDepInfo**
- nfca : **rfalNfcDevice**
- nfcb : **rfalNfcDevice**
- nfcDep : **rfalNfcDevice**
- nfcDepBuf : **rfalNfcBuffer**
- nfcDepDev : **rfalNfcDepListenActvParam**
- nfcDepLR : **rfalNfcDiscoverParam**
- nfcf : **rfalNfcDevice**
- nfcfBR : **rfalNfcDiscoverParam**
- nfcid : **rfalLmConfPA**, **rfalNfcDepAtrParam**, **rfalNfcDepConfigs**, **rfalNfcDevice**
- nfcid0 : **rfalIsoDepAttribCmd**, **rfalNfcbSensbRes**
- nfcId1 : **rfalNfcaListenDevice**
- nfcid1 : **rfalNfcaSddRes**, **rfalNfcaSelReq**
- nfcId1Len : **rfalNfcaListenDevice**
- NFCID2 : **rfalNfcfPollDevice**, **rfalNfcfSensfRes**
- NFCID3 : **rfalNfcDepAtrReq**, **rfalNfcDepAtrRes**
- nfcid3 : **rfalNfcDepTargetParam**, **rfalNfcDiscoverParam**
- nfcidLen : **rfalLmConfPA**, **rfalNfcDepAtrParam**, **rfalNfcDepConfigs**, **rfalNfcDevice**
- NFCIP1 : **st25r95SPIRxContext**
- NFCIP1_SoD : **st25r95SPIRxContext**
- nfcv : **rfalNfcDevice**
- notifyCb : **rfalNfcDiscoverParam**
- num : **rfalAnalogConfig**
- numApps : **rfalNfcbSensbResAppData**
- numBlock : **rfalNfcfServBlockListParam**
- numEntries : **st25r500DpocrConfig**
- numServ : **rfalNfcfServBlockListParam**

Files

Examples

Data Structures

Data Structure Index

All

a

b

c

d

e

f

g

h

i

l

m

n

p

q

r

s

t

u

v

w

- o -

- oper : **rfalNfcDepConfigs**
- operParam : **rfalNfcDepAtrParam**, **rfalNfcDepTargetParam**
- ourFSx : **rfalIsoDepApduTxRxParam**, **rfalIsoDepTxRxParam**

---

a

b

c

d

e

f

g

h

i

l

m

n

o

q

r

s

t

u

v

w

- p -

- P1 : **rfalT4tCApduParam**
- P2 : **rfalT4tCApduParam**
- p2pNfcaPrio : **rfalNfcDiscoverParam**
- PAD0 : **rfalNfcfSensfRes**
- PAD1 : **rfalNfcfSensfRes**
- PAD2 : **rfalNfcfSensfRes**
- Param : **rfalIsoDepAttribCmd**
- PARAM : **rfalIsoDepRats**
- PARAM1 : **rfalIsoDepAttribCmd**
- PARAM2 : **rfalIsoDepAttribCmd**
- PARAM3 : **rfalIsoDepAttribCmd**
- PARAM4 : **rfalIsoDepAttribCmd**
- payload : **rfalNfcvGenericReq**
- pdu : **rfalNfcDepPduBufFormat**
- period : **rfalWakeUpConfig**
- pha : **st25r3916AatTuneResult**
- phaTarget : **st25r3916AatTuneParams**
- phaWeight : **st25r3916AatTuneParams**
- platformInfo : **rfalNfcaSensRes**
- PPi : **rfalNfcDepAtrReq**
- PPS0 : **rfalIsoDepPpsReq**
- PPS1 : **rfalIsoDepPpsReq**
- PPSS : **rfalIsoDepPpsReq**, **rfalIsoDepPpsRes**
- PPt : **rfalNfcDepAtrRes**
- ppt : **rfalNfcDepTargetParam**
- prologue : **rfalIsoDepApduBufFormat**, **rfalIsoDepBufFormat**, **rfalNfcDepBufFormat**, **rfalNfcDepPduBufFormat**
- propNfc : **rfalNfcDiscoverParam**
- protInfo : **rfalNfcbSensbRes**
- proto : **rfalNfcDevice**
- protocol : **st25r95SPIRxContext**

Modules

Files

Examples

Data Structures

Data Structure Index

All

a

b

c

d

e

f

g

h

i

l

m

n

o

p

r

s

t

u

v

w

- q -

- Q : **rfalWakeUpConfig**, **rfalWakeUpInfo**

a

b

c

d

e

f

g

h

i

l

m

n

o

p

q

s

t

u

v

w

- r -

- rApduBodyLen : **rfalT4tRApduParam**
- rApduBuf : **rfalT4tRApduParam**
- RATS : **rfalIsoDepActivation**
- rcvdLen : **rfalT4tRApduParam**
- RD : **rfalNfcfSensfRes**
- refDelay : **rfalWakeUpConfig**
- reference : **rfalWakeUpConfig**, **rfalWakeUpInfo**, **rfalWumMeasChannel**
- refMeasurement : **rfalDlmaInfo**, **rfalDpoInfo**
- refWU : **rfalWakeUpConfig**
- regeStepSize : **st25r500DpocrConfig**
- regSet : **rfalAnalogConfig**
- report_period_length : **iso15693StreamConfig**, **st25r3911StreamConfig**, **st25r3916StreamConfig**
- REQ_FLAG : **rfalNfcvGenericReq**
- RES_FLAG : **rfalNfcvGenericRes**, **rfalNfcvInventoryRes**
- retCode : **st25r95SPIRxContext**
- rfalNfcpGetActivationStatus : **rfalNfcPropCallbacks**
- rfalNfcpPollerGetCollisionResolutionStatus : **rfalNfcPropCallbacks**
- rfalNfcpPollerInitialize : **rfalNfcPropCallbacks**
- rfalNfcpPollerStartCollisionResolution : **rfalNfcPropCallbacks**
- rfalNfcpPollerTechnologyDetection : **rfalNfcPropCallbacks**
- rfalNfcpStartActivation : **rfalNfcPropCallbacks**
- rfBuf : **rfalNfcBuffer**
- rfInterface : **rfalNfcDevice**
- rfoRes : **rfalDpoEntry**
- rmvCRC : **st25r95SPIRxContext**
- role : **rfalNfcDepConfigs**
- RsA : **t_st25r3916Regs**
- RsB : **t_st25r3916Regs**
- rxBuf : **rfalIsoDepApduTxRxParam**, **rfalIsoDepListenActvParam**, **rfalIsoDepTxRxParam**, **rfalNfcDepListenActvParam**, **rfalNfcDepPduTxRxParam**, **rfalNfcDepTxRxParam**, **rfalTransceiveContext**, **st25r95SPIRxContext**
- rxBufLen : **rfalTransceiveContext**, **st25r95SPIRxContext**
- rxLen : **rfalIsoDepApduTxRxParam**, **rfalIsoDepListenActvParam**, **rfalIsoDepTxRxParam**, **rfalNfcDepListenActvParam**, **rfalNfcDepPduTxRxParam**, **rfalNfcDepTxRxParam**
- rxRcvdLen : **rfalTransceiveContext**, **st25r95SPIRxContext**

RFAL

Main Page

Modules

Files

Examples

Data Structures

Data Structure Index

All

a

b

c

d

e

f

g

h

i

l

m

n

o

p

q

r

t

u

v

w

- s -

- sak : **rfaINfcaSelRes**
- SC : **rfaILmConfPF**
- SEL_RES : **rfaILmConfPA**
- selCmd : **rfaINfcaSddReq**, **rfaINfcaSelReq**
- selPar : **rfaINfcaSddReq**, **rfaINfcaSelReq**
- selRes : **rfaINfcaListenDevice**
- SENS_RES : **rfaILmConfPA**
- SENSB_RES : **rfaILmConfPB**
- sensbRes : **rfaINfcbListenDevice**
- sensbResLen : **rfaINfcbListenDevice**
- SENSF_RES : **rfaILmConfPF**
- sensfRes : **rfaINfcfListenDevice**
- sensfResLen : **rfaINfcfListenDevice**
- sensRes : **rfaINfcaListenDevice**
- servList : **rfaINfcfServBlockListParam**
- sfgi : **rfaIIsoDepAtsParam**
- SFGI : **rfaIIsoDepInfo**, **rfaINfcbSensbResProtocolInfo**
- SFGT : **rfaIIsoDepInfo**

- skipCal : **rfalWakeUpConfig**
- skipReCal : **rfalWakeUpConfig**
- speedMode : **rfalIso15693PhyConfig_t**
- st25tb : **rfalNfcDevice**
- status : **rfalWakeUpInfo**
- statusWord : **rfalT4tRApduParam**
- supAdFt : **rfalIsoDepInfo**
- supDID : **rfalIsoDepInfo**
- supNAD : **rfalIsoDepInfo**
- swTagDetect : **rfalWakeUpConfig**

Generated on Fri Mar 14 2025 11:45:12 for RFAL by doxygen 1.9.5

RFAL

Main Page

Modules

Files

Examples

Data Structures

Data Structure Index

All

a

b

c

d

e

f

g

h

i

l

m

n

o

p

q

r

s

u

v

w

- t -

- T0 : **rfalIsoDepAts**
- TA : **rfalIsoDepAts**
- ta : **rfalIsoDepAtsParam**
- tableEntries : **rfalDlmaInfo**, **rfalDpoInfo**
- tableEntry : **rfalDlmaInfo**, **rfalDpoInfo**

- tableUpThresholds : **st25r500DpocrConfig**
- tag : **rfaIIsoDepSParameter**
- Target : **rfaINfcDepActivation**
- target : **st25r500DpocrConfig**
- TB : **rfaIIsoDepAts**
- TC : **rfaIIsoDepAts**
- techs2Bail : **rfaINfcDiscoverParam**
- techs2Find : **rfaINfcDiscoverParam**
- threshold : **rfaIWumMeasChannel**
- TL : **rfaIIsoDepAts**
- tmpBuf : **rfaIIsoDepApduTxRxParam**, **rfaINfcDepPduTxRxParam**
- TO : **rfaINfcDepAtrRes**
- to : **rfaINfcDepConfigs**, **rfaINfcDepTargetParam**
- totalDuration : **rfaINfcDiscoverParam**
- txBuf : **rfaIIsoDepApduTxRxParam**, **rfaIIsoDepTxRxParam**, **rfaINfcDepPduTxRxParam**, **rfaINfcDepTxRxParam**, **rfaITransceiveContext**
- txBufLen : **rfaIIsoDepApduTxRxParam**, **rfaIIsoDepTxRxParam**, **rfaINfcDepPduTxRxParam**, **rfaINfcDepTxRxParam**, **rfaITransceiveContext**
- type : **rfaINfcaListenDevice**, **rfaINfcDevice**

RFAL

Main Page

Modules

Files

Examples

Data Structures

Data Structure Index

All

a

b

c

d

e

f

g

h

i

l

m

n

o

p

q

r

s

t

v

w

- u -

- UID : **rfalNfcvGenericReq**, **rfalNfcvInventoryRes**, **rfalSt25tbListenDevice**
- uid : **rfalT1TRidRes**
- unmodRes : **rfalDlmaEntry**
- useBPSK : **iso15693StreamConfig**, **st25r3911StreamConfig**, **st25r3916StreamConfig**

RFAL

g

h

i

l

m

n

o

p

q

r

s

t

u

w

- v -

- val : **rfalAnalogConfigRegAddrMaskVal**
- value : **rfalIsoDepSParameter**

Modules

Files

Examples

Data Structures

Data Structure Index

All

a

b

c

d

e

f

g

h

i

l

m

n

o

p

q

r

s

t

u

v

- w -

- wakeupConfig : **rfalNfcDiscoverParam**
- wakeupConfigDefault : **rfalNfcDiscoverParam**
- wakeupEnabled : **rfalNfcDiscoverParam**
- wakeupNPolls : **rfalNfcDiscoverParam**
- wakeupPollBefore : **rfalNfcDiscoverParam**
- WLC : **rfalWakeUpInfo**
- WT : **rfalNfcDepInfo**

File List

Here is a list of all documented files with brief descriptions:

[detail level 123]

▼	**include**	
	rfal_analogConfig.h	RF Chip Analog Configuration Settings
	rfal_cd.h	Implementation of a Card Detection Algorithm
	rfal_chip.h	RF Chip specific Layer
	rfal_defConfig.h	RF Abstraction Layer (RFAL) default Config file
	rfal_dlma.h	Dynamic load modulation adjustment
	rfal_dpo.h	Dynamic Power Output
	rfal_isoDep.h	Implementation of ISO-DEP protocol
	rfal_nfc.h	RFAL NFC device
	rfal_nfca.h	Provides several NFC-A convenience methods and definitions
	rfal_nfcb.h	Implementation of NFC-B (ISO14443B) helpers
	rfal_nfcDep.h	Implementation of NFC-DEP protocol
	rfal_nfcf.h	Implementation of NFC-F Poller (FeliCa PCD) device
	rfal_nfcv.h	Implementation of NFC-V Poller (ISO15693) device
	rfal_rf.h	RF Abstraction Layer (RFAL)
	rfal_st25tb.h	Implementation of ST25TB interface
	rfal_st25xv.h	NFC-V ST25 NFC-V Tag specific features
	rfal_t1t.h	Provides NFC-A T1T convenience methods and definitions
	rfal_t2t.h	Provides NFC-A T2T convenience methods and definitions
	rfal_t4t.h	Provides convenience methods and definitions for T4T (ISO7816-4)
	rfal_utils.h	RF Abstraction Layer (RFAL) Utils
▼	**source**	
▼	**st25r200**	
	st25r200/rfal_features.h	RFAL Features/Capabilities Definition for ST25R200
	st25r200.h	ST25R200 high level interface
	st25r200_com.h	ST25R200 communication declaration file
	st25r200_irq.h	ST25R200 Interrupt handling
▼	**st25r3911**	
	st25r3911/rfal_features.h	RFAL Features/Capabilities Definition for ST25R3911
	st25r3911.h	ST25R3911 declaration file

	st25r3911_com.h	ST25R3911 communication declaration file
	st25r3911_interrupt.h	ST25R3911 Interrupt header file
▼	st25r3916	
	st25r3916/rfal_dlmaTbl.h	RF Dynamic LMA Table default values
	st25r3916/rfal_dpoTbl.h	RF Dynamic Power Table default values
	st25r3916/rfal_features.h	RFAL Features/Capabilities Definition for ST25R3916
	st25r3916.h	ST25R3916 high level interface
	st25r3916_aat.h	ST25R3916 Antenna Tuning
	st25r3916_com.h	ST25R3916 communication declaration file
	st25r3916_irq.h	ST25R3916 Interrupt handling
	st25r3916_led.h	ST25R3916 LEDs handling
▼	st25r500	
	st25r500/rfal_dlmaTbl.h	RF Dynamic LMA Table default values
	st25r500/rfal_features.h	RFAL Features/Capabilities Definition for ST25R500
	st25r500.h	ST25R500 high level interface
	st25r500_com.h	ST25R500 communication declaration file
	st25r500_dpocr.h	Dynamic Power Output adjustment via Current Regulation
	st25r500_irq.h	ST25R500 Interrupt handling
▼	st25r95	
	st25r95/rfal_features.h	RFAL Features/Capabilities Definition for ST25R95
	st25r95.h	ST25R95 declaration file
	st25r95_com.h	ST25R95 communication declaration file
	rfal_crc.h	CRC calculation module
	rfal_iso15693_2.h	Implementation of ISO-15693-2
	rfal_platform.h	Platform specific definition layer

RFAL

include Directory Reference

Files

file **rfal_analogConfig.h**
RF Chip Analog Configuration Settings.

file **rfal_cd.h**
Implementation of a Card Detection Algorithm.

file **rfal_chip.h**
RF Chip specific Layer.

file **rfal_defConfig.h**
RF Abstraction Layer (RFAL) default Config file.

file **rfal_dlma.h**
Dynamic load modulation adjustment.

file **rfal_dpo.h**
Dynamic Power Output.

file **rfal_isoDep.h**
Implementation of ISO-DEP protocol.

file **rfal_nfc.h**
RFAL NFC device.

file **rfal_nfca.h**
Provides several NFC-A convenience methods and definitions.

file **rfal_nfcb.h**
Implementation of NFC-B (ISO14443B) helpers.

file **rfal_nfcDep.h**
Implementation of NFC-DEP protocol.

file **rfal_nfcf.h**
Implementation of NFC-F Poller (FeliCa PCD) device.

file **rfal_nfcv.h**
Implementation of NFC-V Poller (ISO15693) device.

file **rfal_rf.h**
RF Abstraction Layer (RFAL)

file **rfal_st25tb.h**
Implementation of ST25TB interface.

file **rfal_st25xv.h**
NFC-V ST25 NFC-V Tag specific features.

file **rfal_t1t.h**
Provides NFC-A T1T convenience methods and definitions.

file **rfal_t2t.h**
Provides NFC-A T2T convenience methods and definitions.

file **rfal_t4t.h**
Provides convenience methods and definitions for T4T (ISO7816-4)

file **rfal_utils.h**
RF Abstraction Layer (RFAL) Utils.

rfal_analogConfig.h File Reference

RF Chip Analog Configuration Settings. More...

```
#include "rfal_platform.h"
#include "rfal_utils.h"
#include "rfal_rf.h"
```

Include dependency graph for rfal_analogConfig.h:

Data Structures

struct **rfalAnalogConfigRegAddrMaskVal**

struct **rfalAnalogConfig**

Macros

#define **RFAL_ANALOG_CONFIG_TBL_SIZE** (1024U)

#define **RFAL_ANALOG_CONFIG_LUT_SIZE** (87U)

#define **RFAL_ANALOG_CONFIG_LUT_NOT_FOUND** (0xFFU)

#define **RFAL_ANALOG_CONFIG_POLL_LISTEN_MODE_MASK** (0x8000U)

#define **RFAL_ANALOG_CONFIG_TECH_MASK** (0x7F00U)

#define **RFAL_ANALOG_CONFIG_BITRATE_MASK** (0x00F0U)

#define **RFAL_ANALOG_CONFIG_DIRECTION_MASK** (0x000FU)

#define **RFAL_ANALOG_CONFIG_CHIP_SPECIFIC_MASK** (0x00FFU)

#define **RFAL_ANALOG_CONFIG_POLL_LISTEN_MODE_SHIFT** (15U)

#define **RFAL_ANALOG_CONFIG_TECH_SHIFT** (8U)

#define **RFAL_ANALOG_CONFIG_BITRATE_SHIFT** (4U)

#define **RFAL_ANALOG_CONFIG_DIRECTION_SHIFT** (0U)

```
#define RFAL_ANALOG_CONFIG_POLL (0x0000U)
```

```
#define RFAL_ANALOG_CONFIG_LISTEN (0x8000U)
```

```
#define RFAL_ANALOG_CONFIG_TECH_CHIP (0x0000U)
```

```
#define RFAL_ANALOG_CONFIG_TECH_NFCA (0x0100U)
```

```
#define RFAL_ANALOG_CONFIG_TECH_NFCB (0x0200U)
```

```
#define RFAL_ANALOG_CONFIG_TECH_NFCF (0x0400U)
```

```
#define RFAL_ANALOG_CONFIG_TECH_AP2P (0x0800U)
```

```
#define RFAL_ANALOG_CONFIG_TECH_NFCV (0x1000U)
```

```
#define RFAL_ANALOG_CONFIG_TECH_RFU (0x2000U)
```

```
#define RFAL_ANALOG_CONFIG_TECH_RFU2 (0x4000U)
```

```
#define RFAL_ANALOG_CONFIG_BITRATE_COMMON (0x0000U)
```

```
#define RFAL_ANALOG_CONFIG_BITRATE_106 (0x0010U)
```

```
#define RFAL_ANALOG_CONFIG_BITRATE_212 (0x0020U)
```

```
#define RFAL_ANALOG_CONFIG_BITRATE_424 (0x0030U)
```

#define
RFAL_ANALOG_CONFIG_BITRATE_848 (0x0040U)

#define
RFAL_ANALOG_CONFIG_BITRATE_1695 (0x0050U)

#define
RFAL_ANALOG_CONFIG_BITRATE_3390 (0x0060U)

#define
RFAL_ANALOG_CONFIG_BITRATE_6780 (0x0070U)

#define
RFAL_ANALOG_CONFIG_BITRATE_211p88 (0x0090U)

#define
RFAL_ANALOG_CONFIG_BITRATE_105p94 (0x00A0U)

#define
RFAL_ANALOG_CONFIG_BITRATE_53 (0x00B0U)

#define
RFAL_ANALOG_CONFIG_BITRATE_26 (0x00C0U)

#define
RFAL_ANALOG_CONFIG_BITRATE_1p6 (0x00D0U)

#define
RFAL_ANALOG_CONFIG_BITRATE_RFU (0x00E0U)

#define
RFAL_ANALOG_CONFIG_BITRATE_RFU2 (0x00F0U)

#define
RFAL_ANALOG_CONFIG_NO_DIRECTION (0x0000U)

#define
RFAL_ANALOG_CONFIG_TX (0x0001U)

#define
RFAL_ANALOG_CONFIG_RX (0x0002U)

#define
RFAL_ANALOG_CONFIG_ANTICOL (0x0003U)

#define
RFAL_ANALOG_CONFIG_DPO (0x0004U)

#define
RFAL_ANALOG_CONFIG_DLMA (0x0005U)

#define
RFAL_ANALOG_CONFIG_DIRECTION_RFU2 (0x0006U)

#define
RFAL_ANALOG_CONFIG_DIRECTION_RFU3 (0x0007U)

#define
RFAL_ANALOG_CONFIG_DIRECTION_RFU4 (0x0008U)

#define
RFAL_ANALOG_CONFIG_DIRECTION_RFU5 (0x0009U)

#define
RFAL_ANALOG_CONFIG_DIRECTION_RFU6 (0x000AU)

#define
RFAL_ANALOG_CONFIG_DIRECTION_RFU7 (0x000BU)

#define
RFAL_ANALOG_CONFIG_DIRECTION_RFU8 (0x000CU)

#define
RFAL_ANALOG_CONFIG_DIRECTION_RFU9 (0x000DU)

#define
RFAL_ANALOG_CONFIG_DIRECTION_RFU10 (0x000EU)

#define
RFAL_ANALOG_CONFIG_DIRECTION_RFU11 (0x000FU)

```
#define RFAL_ANALOG_CONFIG_CHIP_INIT (0x0000U)
```

```
#define RFAL_ANALOG_CONFIG_CHIP_DEINIT (0x0001U)
```

```
#define RFAL_ANALOG_CONFIG_CHIP_FIELD_ON (0x0002U)
```

```
#define RFAL_ANALOG_CONFIG_CHIP_FIELD_OFF (0x0003U)
```

```
#define RFAL_ANALOG_CONFIG_CHIP_WAKEUP_ON (0x0004U)
```

```
#define RFAL_ANALOG_CONFIG_CHIP_WAKEUP_OFF (0x0005U)
```

```
#define RFAL_ANALOG_CONFIG_CHIP_LISTEN_ON (0x0006U)
```

```
#define RFAL_ANALOG_CONFIG_CHIP_LISTEN_OFF (0x0007U)
```

```
#define RFAL_ANALOG_CONFIG_CHIP_POLL_COMMON (0x0008U)
```

```
#define RFAL_ANALOG_CONFIG_CHIP_LISTEN_COMMON (0x0009U)
```

```
#define RFAL_ANALOG_CONFIG_CHIP_LOWPOWER_ON (0x000AU)
```

```
#define RFAL_ANALOG_CONFIG_CHIP_LOWPOWER_OFF (0x000BU)
```

```
#define RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_00 (0x0010U)
```

```
#define RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_01 (0x0011U)
```

```
#define RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_02 (0x0012U)
```

```
#define RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_03 (0x0013U)
```

```
#define RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_04 (0x0014U)
```

```
#define RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_05 (0x0015U)
```

```
#define RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_06 (0x0016U)
```

```
#define RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_07 (0x0017U)
```

```
#define RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_08 (0x0018U)
```

```
#define RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_09 (0x0019U)
```

```
#define RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_10 (0x001AU)
```

```
#define RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_11 (0x001BU)
```

```
#define RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_12 (0x001CU)
```

```
#define RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_13 (0x001DU)
```

```
#define RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_14 (0x001EU)
```

```
#define RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_15 (0x001FU)
```

```
#define RFAL_ANALOG_CONFIG_UPDATE_LAST (0x00U)
```

```
#define RFAL_ANALOG_CONFIG_UPDATE_MORE (0x01U)
```

```
#define RFAL_ANALOG_CONFIG_ID_GET_POLL_LISTEN(id) (RFAL_ANALOG_CONFIG_POLL_LISTEN_MODE_MASK & (id))
```

```
#define RFAL_ANALOG_CONFIG_ID_GET_TECH(id) (RFAL_ANALOG_CONFIG_TECH_MASK & (id))
```

```
#define RFAL_ANALOG_CONFIG_ID_IS_CHIP(id) (RFAL_ANALOG_CONFIG_TECH_MASK & (id))
```

```
#define RFAL_ANALOG_CONFIG_ID_IS_NFCA(id) (RFAL_ANALOG_CONFIG_TECH_NFCA & (id))
```

```
#define RFAL_ANALOG_CONFIG_ID_IS_NFCB(id) (RFAL_ANALOG_CONFIG_TECH_NFCB & (id))
```

```
#define RFAL_ANALOG_CONFIG_ID_IS_NFCF(id) (RFAL_ANALOG_CONFIG_TECH_NFCF & (id))
```

```
#define RFAL_ANALOG_CONFIG_ID_IS_AP2P(id) (RFAL_ANALOG_CONFIG_TECH_AP2P & (id))
```

```
#define RFAL_ANALOG_CONFIG_ID_IS_NFCV(id) (RFAL_ANALOG_CONFIG_TECH_NFCV & (id))
```

```
#define RFAL_ANALOG_CONFIG_ID_GET_BITRATE(id) (RFAL_ANALOG_CONFIG_BITRATE_MASK & (id))
```

```
#define RFAL_ANALOG_CONFIG_ID_IS_COMMON(id) (RFAL_ANALOG_CONFIG_BITRATE_MASK & (id))
```

```
#define RFAL_ANALOG_CONFIG_ID_IS_106(id)   (RFAL_ANALOG_CONFIG_BITRATE_106 & (id))
```

```
#define RFAL_ANALOG_CONFIG_ID_IS_212(id)   (RFAL_ANALOG_CONFIG_BITRATE_212 & (id))
```

```
#define RFAL_ANALOG_CONFIG_ID_IS_424(id)   (RFAL_ANALOG_CONFIG_BITRATE_424 & (id))
```

```
#define RFAL_ANALOG_CONFIG_ID_IS_848(id)   (RFAL_ANALOG_CONFIG_BITRATE_848 & (id))
```

```
#define RFAL_ANALOG_CONFIG_ID_IS_1695(id)   (RFAL_ANALOG_CONFIG_BITRATE_1695 & (id))
```

```
#define RFAL_ANALOG_CONFIG_ID_IS_3390(id)   (RFAL_ANALOG_CONFIG_BITRATE_3390 & (id))
```

```
#define RFAL_ANALOG_CONFIG_ID_IS_6780(id)   (RFAL_ANALOG_CONFIG_BITRATE_6780 & (id))
```

```
#define RFAL_ANALOG_CONFIG_ID_IS_26(id)   (RFAL_ANALOG_CONFIG_BITRATE_26 & (id))
```

```
#define RFAL_ANALOG_CONFIG_ID_IS_1p6(id)   (RFAL_ANALOG_CONFIG_BITRATE_1p6 & (id))
```

```
#define RFAL_ANALOG_CONFIG_ID_GET_DIRECTION(id)   (RFAL_ANALOG_CONFIG_DIRECTION_MASK & (id))
```

```
#define RFAL_ANALOG_CONFIG_ID_IS_TX(id)   (RFAL_ANALOG_CONFIG_TX & (id))
```

```
#define RFAL_ANALOG_CONFIG_ID_IS_RX(id)   (RFAL_ANALOG_CONFIG_RX & (id))
```

```
#define RFAL_ANALOG_CONFIG_CONFIG_NUM(x)   (sizeof(x)/sizeof((x)[0]))
```

```
#define RFAL_ANALOG_CONFIG_ID_SET(mode, tech, br, direction)
```

Typedefs

typedef uint8_t	**rfalAnalogConfigMode**
typedef uint8_t	**rfalAnalogConfigTech**
typedef uint8_t	**rfalAnalogConfigBitrate**
typedef uint8_t	**rfalAnalogConfigDirection**
typedef uint8_t	**rfalAnalogConfigRegAddr**[2]
typedef uint8_t	**rfalAnalogConfigRegMask**
typedef uint8_t	**rfalAnalogConfigRegVal**
typedef uint16_t	**rfalAnalogConfigId**
typedef uint16_t	**rfalAnalogConfigOffset**
typedef uint8_t	**rfalAnalogConfigNum**

Functions

void	**rfalAnalogConfigInitialize** (void) Initialize the Analog Configuration. More...
bool	**rfalAnalogConfigIsReady** (void) Indicate if the current Analog Configuration Table is complete and ready to be used. More...
ReturnCode	**rfalAnalogConfigListWriteRaw** (const uint8_t *configTbl, uint16_t configTblSize) Write the whole Analog Configuration table in raw format. More...
ReturnCode	**rfalAnalogConfigListWrite** (uint8_t more, const **rfalAnalogConfig** *config) Write the Analog Configuration table with new analog settings. More...
ReturnCode	**rfalAnalogConfigListReadRaw** (uint8_t *tblBuf, uint16_t tblBufLen, uint16_t *configTblSize) Read the whole Analog Configuration table in raw format. More...

ReturnCode	**rfalAnalogConfigListRead** (**rfalAnalogConfigOffset** *configOffset, uint8_t *more, **rfalAnalogConfig** *config, **rfalAnalogConfigNum** numConfig) Read the Analog Configuration table. More...
ReturnCode	**rfalSetAnalogConfig** (**rfalAnalogConfigId** configId) Set the Analog settings of indicated Configuration ID. More...
uint16_t	**rfalAnalogConfigGenModeID** (**rfalMode** md, **rfalBitRate** br, uint16_t dir) Generates Analog Config mode ID. More...

Detailed Description

ST25R95 Analog Configuration Settings.

ST25R500 Analog Configuration Settings.

ST25R3916 Analog Configuration Settings.

ST25R3911 Analog Configuration Settings.

ST25R200 Analog Configuration Settings.

Author

bkam

Author

Gustavo Patricio

Author

bkam

Author

rfal_cd.h File Reference

Implementation of a Card Detection Algorithm. More...

```
#include "rfal_platform.h"
#include "rfal_utils.h"
```

Include dependency graph for rfal_cd.h:

Data Structures

struct **rfalCdRes**

Enumerations

enum **rfalCdTech** {
RFAL_CD_TECH_NONE = 0x00 , **RFAL_CD_TECH_NFCA** = 0x01 , **RFAL_CD_TECH_NFCB** = 0x02 , **RFAL_CD_TECH_NFCF** = 0x04 ,

RFAL_CD_TECH_NFCV = 0x08 , RFAL_CD_TECH_OTHER = 0x10
}

enum rfalCdDetType {
RFAL_CD_NOT_FOUND = 0 , RFAL_CD_SINGLE_DEV = 1 , RFAL_CD_MULTIPLE_DEV = 2 , RFAL_CD_MULTIPLE_TECH = 3 ,
RFAL_CD_CARD_TECH = 4 , RFAL_CD_SINGLE_MULTI_TECH = 5 , RFAL_CD_SINGLE_P2P = 6 , RFAL_CD_SINGLE_HB = 7 ,
RFAL_CD_UNKOWN = 8
}

Functions

ReturnCode **rfalCdDetectCard** (**rfalCdRes** *result)
Detect Card. More...

ReturnCode **rfalCdStartDetectCard** (**rfalCdRes** *result)
Start Card Detection. More...

ReturnCode **rfalCdGetDetectCardStatus** (void)
Get Card Detection Status. More...

Detailed Description

Author

rfal_chip.h File Reference

RF Chip specific Layer. More...

```
#include "rfal_platform.h"
#include "rfal_utils.h"
#include "rfal_rf.h"
```

Include dependency graph for rfal_chip.h:

Functions

ReturnCode	**rfalChipWriteReg** (uint16_t reg, const uint8_t *values, uint8_t len) Writes a register on the RF Chip. More...
ReturnCode	**rfalChipReadReg** (uint16_t reg, uint8_t *values, uint8_t len) Reads a register on the RF Chip. More...
ReturnCode	**rfalChipChangeRegBits** (uint16_t reg, uint8_t valueMask, uint8_t value) Change a register on the RF Chip. More...

ReturnCode **rfalChipWriteTestReg** (uint16_t reg, uint8_t value)
Writes a Test register on the RF Chip. More...

ReturnCode **rfalChipReadTestReg** (uint16_t reg, uint8_t *value)
Reads a Test register on the RF Chip. More...

ReturnCode **rfalChipChangeTestRegBits** (uint16_t reg, uint8_t valueMask, uint8_t value)
Change a Test register on the RF Chip. More...

ReturnCode **rfalChipExecCmd** (uint16_t cmd)
Execute command on the RF Chip. More...

ReturnCode **rfalChipSetRFO** (uint8_t rfo)
Set RFO. More...

ReturnCode **rfalChipGetRFO** (uint8_t *result)
Get RFO. More...

ReturnCode **rfalChipGetLmFieldInd** (uint8_t *result)
Get LM Field Indicator. More...

ReturnCode **rfalChipSetLMMod** (uint8_t mod, uint8_t unmod)
Set Listen Mode Modulation. More...

ReturnCode **rfalChipGetLMMod** (uint8_t *mod, uint8_t *unmod)
Get Listen Mode Modulation. More...

ReturnCode **rfalChipMeasureAmplitude** (uint8_t *result)
Measure Amplitude. More...

ReturnCode **rfalChipMeasurePhase** (uint8_t *result)
Measure Phase. More...

ReturnCode **rfalChipMeasureCapacitance** (uint8_t *result)
Measure Capacitance. More...

ReturnCode **rfalChipMeasurePowerSupply** (uint8_t param, uint8_t *result)
Measure Power Supply. More...

ReturnCode	**rfalChipMeasureIQ** (int8_t *resI, int8_t *resQ) Measure I and Q. More...
ReturnCode	**rfalChipMeasureCombinedIQ** (uint8_t *result) Measure combined I and Q. More...
ReturnCode	**rfalChipMeasureI** (uint8_t *result) Measure I channel. More...
ReturnCode	**rfalChipMeasureQ** (uint8_t *result) Measure Q channel. More...
ReturnCode	**rfalChipMeasureCurrent** (uint8_t *result) Measure Current. More...
ReturnCode	**rfalChipSetAntennaMode** (bool single, bool rfiox) Set Antenna mode. More...

Detailed Description

Author

Gustavo Patricio

Warning

This layer, which provides direct access to RF chip, should only be used for debug purposes and/or advanced features

rfal_defConfig.h File Reference

RF Abstraction Layer (RFAL) default Config file. More...

```
#include "rfal_features.h"
```

Include dependency graph for rfal_defConfig.h:

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

Macros

#define **RFAL_FEATURE_WAKEUP_MODE** true

#define **RFAL_FEATURE_LOWPOWER_MODE** false

#define **RFAL_FEATURE_NFC_RF_BUF_LEN** 258U

#define **RFAL_FEATURE_ST25xV** false

#define **RFAL_FEATURE_DYNAMIC_ANALOG_CONFIG** false

#define **RFAL_FEATURE_DPO** false

#define **RFAL_FEATURE_DLMA** false

#define **platformProtectST25RIrqStatus**()

#define **platformUnprotectST25RIrqStatus**()

#define **platformProtectWorker**()

#define **platformUnprotectWorker**()

#define **platformIrqST25RPinInitialize**()

#define **platformIrqST25RSetCallback**(cb)

#define **platformLedsInitialize**()

#define **platformLedOff**(port, pin)

#define **platformLedOn**(port, pin)

#define **platformLedToggle**(port, pin)

#define **platformGetSysTick**()

#define **platformTimerGetRemaining**(timer) (0U)

#define **platformTimerDestroy**(timer)

#define **platformLog**(...)

#define **platformAssert**(exp)

#define **platformErrorHandle**()

#define **platformI2CTx**(txBuf, len, last, txOnly)

#define **platformI2CRx**(txBuf, len)

#define **platformI2CStart**()

#define **platformI2CStop**()

#define **platformI2CRepeatStart**()

#define **platformI2CSlaveAddrWR**(add)

#define **platformI2CSlaveAddrRD**(add)

Detailed Description

Author

Gustavo Patricio

This file contains a base/default configuration for the RFAL library. Users can and shall define their on configuration acording to their speficic system needs on **rfal_platform.h**.

rfal_dlma.h File Reference

Dynamic load modulation adjustment. More...

```
#include "rfal_platform.h"
#include "rfal_utils.h"
```

Include dependency graph for rfal_dlma.h:

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

Data Structures

struct **rfalDlmaEntry**

struct **rfalDlmaInfo**

Macros

#define **RFAL_DLMA_TABLE_MAX_ENTRIES** 4U

#define **RFAL_DLMA_TABLE_PARAM_LEN** sizeof(**rfalDlmaEntry**)

#define **RFAL_DLMA_TABLE_SIZE_MAX** (**RFAL_DLMA_TABLE_MAX_ENTRIES** * **RFAL_DLMA_TABLE_PARAM_LEN**)

Typedefs

typedef **ReturnCode**(* **rfalDlmaMeasureFunc**) (uint8_t *res)

typedef **ReturnCode**(* **rfalDlmaAdjustFunc**) (uint8_t mod, uint8_t unmod)

Functions

void	**rfalDlmaInitialize** (void) Initialize dynamic power table. More...
void	**rfalDlmaSetMeasureCallback** (**rfalDlmaMeasureFunc** pFunc) Set the measurement methode. More...
void	**rfalDlmaSetAdjustCallback** (**rfalDlmaAdjustFunc** pFunc) Set the measurement methode. More...
ReturnCode	**rfalDlmaTableWrite** (const **rfalDlmaEntry** *powerTbl, uint8_t powerTblEntries) Dynamic LMA table write. More...
ReturnCode	**rfalDlmaTableRead** (**rfalDlmaEntry** *tblBuf, uint8_t tblBufEntries, uint8_t *tableEntries) Dynamic LMA table Read. More...
ReturnCode	**rfalDlmaAdjust** (void) Dynamic LMA adjust. More...
void	**rfalDlmaSetEnabled** (bool enable) Dynamic LMA set enabled state. More...
bool	**rfalDlmaIsEnabled** (void) Get the Dynamic LMA enabled state. More...
ReturnCode	**rfalDlmaGetInfo** (**rfalDlmaInfo** *info) Get DLMA information. More...

Detailed Description

This module provides an interface to perform the load modulation dynamically

rfal_dpo.h File Reference

Dynamic Power Output. More...

```
#include "rfal_platform.h"
#include "rfal_utils.h"
```

Include dependency graph for rfal_dpo.h:

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

Data Structures

struct **rfalDpoEntry**

struct **rfalDpoInfo**

Macros

#define **RFAL_DPO_TABLE_MAX_ENTRIES** 4U

#define **RFAL_DPO_TABLE_PARAM_LEN** sizeof(**rfalDpoEntry**)

#define **RFAL_DPO_TABLE_SIZE_MAX** (**RFAL_DPO_TABLE_MAX_ENTRIES** * **RFAL_DPO_TABLE_PARAM_LEN**)

Typedefs

typedef **ReturnCode**(* **rfalDpoMeasureFunc**) (uint8_t *res)

typedef **ReturnCode**(* **rfalDpoAdjustFunc**) (uint8_t res)

Functions

void **rfalDpoInitialize** (void)

Initialize dynamic power table. More...

void **rfalDpoSetMeasureCallback** (**rfalDpoMeasureFunc** pFunc)

Set the Measurement method. More...

void **rfalDpoSetAdjustCallback** (**rfalDpoAdjustFunc** pFunc)

Set the Adjust method. More...

ReturnCode **rfalDpoTableWrite** (const **rfalDpoEntry** *powerTbl, uint8_t powerTblEntries)

Write dynamic power table. More...

ReturnCode **rfalDpoTableRead** (**rfalDpoEntry** *tblBuf, uint8_t tblBufEntries, uint8_t *tableEntries)

Dynamic power table Read. More...

ReturnCode **rfalDpoAdjust** (void)

	Dynamic power adjust. More...
void	**rfalDpoSetEnabled** (bool enable) Dynamic power set enabled state. More...
bool	**rfalDpoIsEnabled** (void) Get the Dynamic power enabled state. More...
void	**rfalDpoReqAdj** (void) Request DPO adjust. More...
ReturnCode	**rfalDpoGetInfo** (**rfalDpoInfo** *info) Get DPO information. More...

Detailed Description

RF Dynamic Power Table default values.

Author

Martin Zechleitner

This module provides an interface to perform the power adjustment dynamically

Author

Gustavo Patricio

rfal_isoDep.h File Reference

Implementation of ISO-DEP protocol. More...

```
#include "rfal_platform.h"
#include "rfal_nfcb.h"
```

Include dependency graph for rfal_isoDep.h:

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

Data Structures

struct	**rfalIsoDepRats**
struct	**rfalIsoDepAts**

struct	**rfalIsoDepPpsReq**
struct	**rfalIsoDepPpsRes**
struct	**rfalIsoDepAttribCmd**
struct	**rfalIsoDepAttribRes**
struct	**rfalIsoDepSParameter**
union	**rfalIsoDepActivation**
struct	**rfalIsoDepInfo**
struct	**rfalIsoDepDevice**
struct	**rfalIsoDepAttribResParam**
struct	**rfalIsoDepAtsParam**
struct	**rfalIsoDepBufFormat**
struct	**rfalIsoDepApduBufFormat**
struct	**rfalIsoDepListenActvParam**
struct	**rfalIsoDepTxRxParam**
struct	**rfalIsoDepApduTxRxParam**

Macros

#define	**RFAL_FEATURE_ISO_DEP_IBLOCK_MAX_LEN** (1U)
#define	**RFAL_FEATURE_ISO_DEP_APDU_MAX_LEN** (1U)
#define	**RFAL_ISODEP_PROLOGUE_SIZE** (3U)
#define	**RFAL_ISODEP_PCB_LEN** (1U)
#define	**RFAL_ISODEP_DID_LEN** (1U)

```
#define RFAL_ISODEP_NAD_LEN (1U)
#define RFAL_ISODEP_NO_DID (0x10U)
#define RFAL_ISODEP_NO_NAD (0xFFU)
#define RFAL_ISODEP_FWI_MASK (0xF0U)
#define RFAL_ISODEP_FWI_SHIFT (4U)
#define RFAL_ISODEP_FWI_DEFAULT (4U)
#define RFAL_ISODEP_ADV_FEATURE (0x0FU)
#define RFAL_ISODEP_DID_MAX (14U)
#define RFAL_ISODEP_BRI_MASK (0x07U)
#define RFAL_ISODEP_BSI_MASK (0x70U)
#define RFAL_ISODEP_SAME_BITRATE_MASK (0x80U)
#define RFAL_ISODEP_BITRATE_RFU_MASK (0x08U)
#define RFAL_ISODEP_MAX_FWT ((uint32_t)1U<<26)
#define RFAL_ISODEP_FSDI_DEFAULT RFAL_ISODEP_FSXI_256
#define RFAL_ISODEP_FSX_KEEP (0xFFU)
#define RFAL_ISODEP_DEFAULT_FSCI RFAL_ISODEP_FSXI_256
#define RFAL_ISODEP_DEFAULT_FSC RFAL_ISODEP_FSX_256
#define RFAL_ISODEP_DEFAULT_SFGI (0U)
#define RFAL_ISODEP_DEFAULT_FWI (8U)
#define RFAL_ISODEP_APDU_MAX_LEN RFAL_ISODEP_FSX_1024
#define RFAL_ISODEP_ATTRIB_RES_MBLI_NO_INFO (0x00U)
```

#define **RFAL_ISODEP_ATTRIB_REQ_PARAM1_DEFAULT** (0x00U)

#define **RFAL_ISODEP_ATTRIB_HLINFO_LEN** (32U)

#define **RFAL_ISODEP_ATS_HB_MAX_LEN** (15U)

#define **RFAL_ISODEP_ATTRIB_REQ_MIN_LEN** (9U)

#define **RFAL_ISODEP_ATTRIB_RES_MIN_LEN** (1U)

#define **RFAL_ISODEP_SPARAM_VALUES_MAX_LEN** (16U)

#define **RFAL_ISODEP_SPARAM_TAG_BLOCKINFO** (0xA0U)

#define **RFAL_ISODEP_SPARAM_TAG_BRREQ** (0xA1U)

#define **RFAL_ISODEP_SPARAM_TAG_BRIND** (0xA2U)

#define **RFAL_ISODEP_SPARAM_TAG_BRACT** (0xA3U)

#define **RFAL_ISODEP_SPARAM_TAG_BRACK** (0xA4U)

#define **RFAL_ISODEP_SPARAM_TAG_SUP_PCD2PICC** (0x80U)

#define **RFAL_ISODEP_SPARAM_TAG_SUP_PICC2PCD** (0x81U)

#define **RFAL_ISODEP_SPARAM_TAG_SUP_FRAME** (0x82U)

#define **RFAL_ISODEP_SPARAM_TAG_SEL_PCD2PICC** (0x83U)

#define **RFAL_ISODEP_SPARAM_TAG_SEL_PICC2PCD** (0x84U)

#define **RFAL_ISODEP_SPARAM_TAG_SEL_FRAME** (0x85U)

#define **RFAL_ISODEP_SPARAM_TAG_LEN** (1)

#define **RFAL_ISODEP_SPARAM_TAG_BRREQ_LEN** (0U)

#define **RFAL_ISODEP_SPARAM_TAG_PICC2PCD_LEN** (2U)

#define **RFAL_ISODEP_SPARAM_TAG_PCD2PICC_LEN** (2U)

#define **RFAL_ISODEP_SPARAM_TAG_BRACK_LEN** (0U)

#define **RFAL_ISODEP_ATS_TA_DPL_212** (0x01U)

#define **RFAL_ISODEP_ATS_TA_DPL_424** (0x02U)

#define **RFAL_ISODEP_ATS_TA_DPL_848** (0x04U)

#define **RFAL_ISODEP_ATS_TA_DLP_212** (0x10U)

#define **RFAL_ISODEP_ATS_TA_DLP_424** (0x20U)

#define **RFAL_ISODEP_ATS_TA_DLP_848** (0x40U)

#define **RFAL_ISODEP_ATS_TA_SAME_D** (0x80U)

#define **RFAL_ISODEP_ATS_TB_FWI_MASK** (0xF0U)

#define **RFAL_ISODEP_ATS_TB_SFGI_MASK** (0x0FU)

#define **RFAL_ISODEP_ATS_T0_TA_PRESENCE_MASK** (0x10U)

#define **RFAL_ISODEP_ATS_T0_TB_PRESENCE_MASK** (0x20U)

#define **RFAL_ISODEP_ATS_T0_TC_PRESENCE_MASK** (0x40U)

#define **RFAL_ISODEP_ATS_T0_FSCI_MASK** (0x0FU)

#define **RFAL_ISODEP_ATS_T0_OFFSET** (0x01U)

#define **RFAL_ISODEP_ATS_TB_FWI_SHIFT** (4U)

#define **RFAL_ISODEP_ATS_FWI_MASK** (0x0FU)

#define **RFAL_ISODEP_ATS_SFGI_MASK** (0x0FU)

#define **RFAL_ISODEP_MAX_I_RETRYS** (2U)

#define **RFAL_ISODEP_MAX_R_RETRYS** (3U)

#define **RFAL_ISODEP_MAX_WTX_NACK_RETRYS** (3U)

#define **RFAL_ISODEP_MAX_WTX_RETRYS** (20U)

#define **RFAL_ISODEP_MAX_WTX_RETRYS_ULTD** (255U)

#define **RFAL_ISODEP_MAX_DSL_RETRYS** (0U)

#define **RFAL_ISODEP_RATS_RETRIES** (1U)

Enumerations

enum **rfalIsoDepFSxI** {
RFAL_ISODEP_FSXI_16 = 0 , **RFAL_ISODEP_FSXI_24** = 1 , **RFAL_ISODEP_FSXI_32** = 2 , **RFAL_ISODEP_FSXI_40** = 3 ,
RFAL_ISODEP_FSXI_48 = 4 , **RFAL_ISODEP_FSXI_64** = 5 , **RFAL_ISODEP_FSXI_96** = 6 , **RFAL_ISODEP_FSXI_128** = 7 ,
RFAL_ISODEP_FSXI_256 = 8 , **RFAL_ISODEP_FSXI_512** = 9 , **RFAL_ISODEP_FSXI_1024** = 10 , **RFAL_ISODEP_FSXI_2048** = 11 ,
RFAL_ISODEP_FSXI_4096 = 12
}

enum **rfalIsoDepFSx** {
RFAL_ISODEP_FSX_16 = 16 , **RFAL_ISODEP_FSX_24** = 24 , **RFAL_ISODEP_FSX_32** = 32 , **RFAL_ISODEP_FSX_40** = 40 ,
RFAL_ISODEP_FSX_48 = 48 , **RFAL_ISODEP_FSX_64** = 64 , **RFAL_ISODEP_FSX_96** = 96 , **RFAL_ISODEP_FSX_128** = 128 ,
RFAL_ISODEP_FSX_256 = 256 , **RFAL_ISODEP_FSX_512** = 512 , **RFAL_ISODEP_FSX_1024** = 1024 , **RFAL_ISODEP_FSX_2048** = 2048 ,
RFAL_ISODEP_FSX_4096 = 4096
}

Functions

void **rfalIsoDepInitialize** (void)
Initialize the ISO-DEP protocol. More...

void **rfalIsoDepInitializeWithParams** (**rfalComplianceMode** compMode, uint8_t maxRetriesR, uint8_t maxRetriesSnWTX, uint8_t maxRetriesSWTX, uint8_t maxRetriesSDSL, uint8_t maxRetriesI, uint8_t maxRetriesRATS)
Initialize the ISO-DEP protocol. More...

uint16_t	**rfaIsoDepFSxI2FSx** (uint8_t FSxI) FSxI to FSx. More...
uint32_t	**rfaIsoDepFWI2FWT** (uint8_t fwi) FWI to FWT. More...
bool	**rfaIsoDepIsRats** (const uint8_t *buf, uint8_t bufLen) Check if the buffer data contains a valid RATS command. More...
bool	**rfaIsoDepIsAttrib** (const uint8_t *buf, uint8_t bufLen) Check if the buffer data contains a valid ATTRIB command. More...
ReturnCode	**rfaIsoDepListenStartActivation** (**rfaIsoDepAtsParam** *atsParam, const **rfaIsoDepAttribResParam** *attribResParam, const uint8_t *buf, uint16_t bufLen, **rfaIsoDepListenActvParam** actParam) Start Listen Activation Handling. More...
ReturnCode	**rfaIsoDepListenGetActivationStatus** (void) Get the current Activation Status. More...
uint16_t	**rfaIsoDepGetMaxInfLen** (void) Get the ISO-DEP Communication Information. More...
ReturnCode	**rfaIsoDepStartTransceive** (**rfaIsoDepTxRxParam** param) ISO-DEP Start Transceive. More...
ReturnCode	**rfaIsoDepGetTransceiveStatus** (void) Get the Transceive status. More...
ReturnCode	**rfaIsoDepStartApduTransceive** (**rfaIsoDepApduTxRxParam** param) ISO-DEP Start APDU Transceive. More...
ReturnCode	**rfaIsoDepGetApduTransceiveStatus** (void) Get the APDU Transceive status. More...
ReturnCode	**rfaIsoDepRATS** (**rfaIsoDepFSxI** FSDI, uint8_t DID, **rfaIsoDepAts** *ats, uint8_t *atsLen) ISO-DEP Send RATS. More...
ReturnCode	**rfaIsoDepPPS** (uint8_t DID, **rfalBitRate** DSI, **rfalBitRate** DRI, **rfaIsoDepPpsRes** *ppsRes)

ISO-DEP Send PPS. More...

ReturnCode **rfaIsoDepATTRIB** (const uint8_t *nfcid0, uint8_t PARAM1, **rfalBitRate** DSI, **rfalBitRate** DRI, **rfaIsoDepFSxI** FSDI, uint8_t PARAM3, uint8_t DID, const uint8_t *HLInfo, uint8_t HLInfoLen, uint32_t fwt, **rfaIsoDepAttribRes** *attribRes, uint8_t *attribResLen)

ISO-DEP Send ATTRIB. More...

ReturnCode **rfaIsoDepDeselect** (void)

Deselect PICC. More...

ReturnCode **rfaIsoDepStartDeselect** (void)

Start Deselect. More...

ReturnCode **rfaIsoDepGetDeselectStatus** (void)

Deselect Get Status. More...

ReturnCode **rfaIsoDepPollAHandleActivation** (**rfaIsoDepFSxI** FSDI, uint8_t DID, **rfalBitRate** maxBR, **rfaIsoDepDevice** *rfaIsoDepDev)

ISO-DEP Poller Handle NFC-A Activation. More...

ReturnCode **rfaIsoDepPollBHandleActivation** (**rfaIsoDepFSxI** FSDI, uint8_t DID, **rfalBitRate** maxBR, uint8_t PARAM1, const **rfalNfcbListenDevice** *nfcbDev, const uint8_t *HLInfo, uint8_t HLInfoLen, **rfaIsoDepDevice** *rfaIsoDepDev)

ISO-DEP Poller Handle NFC-B Activation. More...

ReturnCode **rfaIsoDepPollHandleSParameters** (**rfaIsoDepDevice** *rfaIsoDepDev, **rfalBitRate** maxTxBR, **rfalBitRate** maxRxBR)

ISO-DEP Poller Handle S(Parameters) More...

ReturnCode **rfaIsoDepPollAStartActivation** (**rfaIsoDepFSxI** FSDI, uint8_t DID, **rfalBitRate** maxBR, **rfaIsoDepDevice** *rfaIsoDepDev)

ISO-DEP Poller Start NFC-A Activation. More...

ReturnCode **rfaIsoDepPollAGetActivationStatus** (void)

ISO-DEP Poller Get NFC-A Activation Status. More...

ReturnCode **rfaIsoDepPollBStartActivation** (**rfaIsoDepFSxI** FSDI, uint8_t DID, **rfalBitRate** maxBR, uint8_t PARAM1, const **rfalNfcbListenDevice** *nfcbDev, const uint8_t *HLInfo, uint8_t HLInfoLen, **rfaIsoDepDevice** *rfaIsoDepDev)

	ISO-DEP Poller Start NFC-B Activation. More...
ReturnCode	**rfaIsoDepPollBGetActivationStatus** (void) ISO-DEP Poller Get NFC-B Activation Status. More...

Detailed Description

Author

Gustavo Patricio

This implementation was based on the following specs:

- ISO/IEC 14443-4 2nd Edition 2008-07-15
- NFC Forum Digital Protocol 1.1 2014-01-14

rfal_nfc.h File Reference

RFAL NFC device. More...

```
#include "rfal_platform.h"
#include "rfal_utils.h"
#include "rfal_rf.h"
#include "rfal_nfca.h"
#include "rfal_nfcb.h"
#include "rfal_nfcf.h"
#include "rfal_nfcv.h"
#include "rfal_st25tb.h"
#include "rfal_nfcDep.h"
#include "rfal_isoDep.h"
```

Include dependency graph for rfal_nfc.h:

Data Structures

struct	**rfalNfcDevice**
struct	**rfalNfcPropCallbacks**
struct	**rfalNfcDiscoverParam**
union	**rfalNfcBuffer**

Macros

#define	**RFAL_NFC_TECH_NONE**	0x0000U

#define **RFAL_NFC_POLL_TECH_A** 0x0001U

#define **RFAL_NFC_POLL_TECH_B** 0x0002U

#define **RFAL_NFC_POLL_TECH_F** 0x0004U

#define **RFAL_NFC_POLL_TECH_V** 0x0008U

#define **RFAL_NFC_POLL_TECH_AP2P** 0x0010U

#define **RFAL_NFC_POLL_TECH_ST25TB** 0x0020U

#define **RFAL_NFC_POLL_TECH_PROP** 0x0040U

#define **RFAL_NFC_LISTEN_TECH_A** 0x1000U

#define **RFAL_NFC_LISTEN_TECH_B** 0x2000U

#define **RFAL_NFC_LISTEN_TECH_F** 0x4000U

#define **RFAL_NFC_LISTEN_TECH_AP2P** 0x8000U

#define **rfalNfcIsDevActivated**(st) (((st)>= **RFAL_NFC_STATE_ACTIVATED**) && ((st)<**RFAL_NFC_STATE_DEACTIVATION**))

#define **rfalNfcIsInDiscovery**(st) (((st)>= **RFAL_NFC_STATE_START_DISCOVERY**) && ((st)<**RFAL_NFC_STATE_ACTIVATED**))

#define **rfalNfcIsRemDevPoller**(tp) (((tp)>= **RFAL_NFC_POLL_TYPE_NFCA**) && ((tp)<=**RFAL_NFC_POLL_TYPE_AP2P**))

#define **rfalNfcIsRemDevListener**(tp) (((int16_t)(tp)>= (int16_t)**RFAL_NFC_LISTEN_TYPE_NFCA**) && ((tp)<=**RFAL_NFC_LISTEN_TYPE_AP2P**))

#define **rfalNfcDefaultDiscParams**(dp)

Typedefs

typedef **ReturnCode**(* **rfalNfcPropCallback**) (void)

Enumerations

enum **rfalNfcState** {
RFAL_NFC_STATE_NOTINIT = 0 , **RFAL_NFC_STATE_IDLE** = 1 , **RFAL_NFC_STATE_START_DISCOVERY** = 2 , **RFAL_NFC_STATE_WAKEUP_MODE** = 3 ,
RFAL_NFC_STATE_POLL_TECHDETECT = 10 , **RFAL_NFC_STATE_POLL_COLAVOIDANCE** = 11 , **RFAL_NFC_STATE_POLL_SELECT** = 12 , **RFAL_NFC_STATE_POLL_ACTIVATION** = 13 ,
RFAL_NFC_STATE_LISTEN_TECHDETECT = 20 , **RFAL_NFC_STATE_LISTEN_COLAVOIDANCE** = 21 , **RFAL_NFC_STATE_LISTEN_ACTIVATION** = 22 , **RFAL_NFC_STATE_LISTEN_SLEEP** = 23 ,
RFAL_NFC_STATE_ACTIVATED = 30 , **RFAL_NFC_STATE_DATAEXCHANGE** = 31 , **RFAL_NFC_STATE_DATAEXCHANGE_DONE** = 33 , **RFAL_NFC_STATE_DEACTIVATION** = 34
}

enum **rfalNfcDevType** {
RFAL_NFC_LISTEN_TYPE_NFCA = 0 , **RFAL_NFC_LISTEN_TYPE_NFCB** = 1 , **RFAL_NFC_LISTEN_TYPE_NFCF** = 2 , **RFAL_NFC_LISTEN_TYPE_NFCV** = 3 ,
RFAL_NFC_LISTEN_TYPE_ST25TB = 4 , **RFAL_NFC_LISTEN_TYPE_AP2P** = 5 , **RFAL_NFC_LISTEN_TYPE_PROP** = 6 , **RFAL_NFC_POLL_TYPE_NFCA** = 10 ,
RFAL_NFC_POLL_TYPE_NFCB = 11 , **RFAL_NFC_POLL_TYPE_NFCF** = 12 , **RFAL_NFC_POLL_TYPE_NFCV** = 13 , **RFAL_NFC_POLL_TYPE_AP2P** = 15
}

enum **rfalNfcRfInterface** { **RFAL_NFC_INTERFACE_RF** = 0 , **RFAL_NFC_INTERFACE_ISODEP** = 1 , **RFAL_NFC_INTERFACE_NFCDEP** = 2 }

enum **rfalNfcDeactivateType** { **RFAL_NFC_DEACTIVATE_IDLE** = 0 , **RFAL_NFC_DEACTIVATE_SLEEP** = 1 , **RFAL_NFC_DEACTIVATE_DISCOVERY** = 2 }

Functions

void **rfalNfcWorker** (void)
RFAL NFC Worker. More...

ReturnCode **rfalNfcInitialize** (void)
RFAL NFC Initialize. More...

ReturnCode **rfalNfcDiscover** (const **rfalNfcDiscoverParam** *disParams)
RFAL NFC Discovery. More...

rfalNfcState	**rfalNfcGetState** (void) RFAL NFC Get State. More...
ReturnCode	**rfalNfcGetDevicesFound** (**rfalNfcDevice** **devList, uint8_t *devCnt) RFAL NFC Get Devices Found. More...
ReturnCode	**rfalNfcGetActiveDevice** (**rfalNfcDevice** **dev) RFAL NFC Get Active Device. More...
ReturnCode	**rfalNfcSelect** (uint8_t devIdx) RFAL NFC Select Device. More...
ReturnCode	**rfalNfcDataExchangeStart** (uint8_t *txData, uint16_t txDataLen, uint8_t **rxData, uint16_t **rvdLen, uint32_t fwt) RFAL NFC Start Data Exchange. More...
ReturnCode	**rfalNfcDataExchangeGetStatus** (void) RFAL NFC Get Data Exchange Status. More...
ReturnCode	**rfalNfcDeactivate** (**rfalNfcDeactivateType** deactType) RFAL NFC Deactivate. More...

Detailed Description

This module provides the required features to behave as an NFC Poller or Listener device. It grants an easy to use interface for the following activities: Technology Detection, Collision Resolution, Activation, Data Exchange, and Deactivation

This layer is influenced by (but not fully aligned with) the NFC Forum specifications, in particular: Activity 2.0 and NCI 2.0

Modules

Data Structures

Examples

File List

Globals

include

rfal_nfca.h File Reference

Provides several NFC-A convenience methods and definitions. More...

```
#include "rfal_platform.h"
#include "rfal_utils.h"
#include "rfal_rf.h"
#include "rfal_t1t.h"
```

Include dependency graph for rfal_nfca.h:

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

Data Structures

struct **rfalNfcaSensRes**

struct **rfalNfcaSddReq**

struct **rfalNfcaSddRes**

struct **rfalNfcaSelReq**

struct **rfalNfcaSelRes**

struct **rfalNfcaListenDevice**

Macros

#define **RFAL_NFCA_CASCADE_1_UID_LEN** 4U

#define **RFAL_NFCA_CASCADE_2_UID_LEN** 7U

#define **RFAL_NFCA_CASCADE_3_UID_LEN** 10U

#define **RFAL_NFCA_SENS_RES_PLATFORM_MASK** 0x0FU

#define **RFAL_NFCA_SENS_RES_PLATFORM_T1T** 0x0CU

#define **RFAL_NFCA_SEL_RES_CONF_MASK** 0x60U

#define **RFAL_NFCA_SEL_RES_CONF_T2T** 0x00U

#define **RFAL_NFCA_SEL_RES_CONF_T4T** 0x20U

#define **RFAL_NFCA_SEL_RES_CONF_NFCDEP** 0x40U

#define **RFAL_NFCA_SEL_RES_CONF_T4T_NFCDEP** 0x60U

#define **RFAL_NFCA_FDTMIN** 1620U

#define **rfalNfcaIsSensResT1T**(sensRes) ((((**rfalNfcaSensRes***)(sensRes))->platformInfo & **RFAL_NFCA_SENS_RES_PLATFORM_MASK**) == **RFAL_NFCA_SENS_RES_PLATFORM_T1T**)

#define **rfalNfcaIsSelResT2T**(selRes) ((((**rfalNfcaSelRes***)(selRes))->sak & **RFAL_NFCA_SEL_RES_CONF_MASK**) == **RFAL_NFCA_SEL_RES_CONF_T2T**)

#define **rfalNfcaIsSelResT4T**(selRes) ((((**rfalNfcaSelRes***)(selRes))->sak & **RFAL_NFCA_SEL_RES_CONF_MASK**) == **RFAL_NFCA_SEL_RES_CONF_T4T**)

#define **rfalNfcaIsSelResNFCDEP**(selRes) ((((**rfalNfcaSelRes***)(selRes))->sak & **RFAL_NFCA_SEL_RES_CONF_MASK**) == **RFAL_NFCA_SEL_RES_CONF_NFCDEP**)

#define **rfalNfcaIsSelResT4TNFCDEP**(selRes) ((((**rfalNfcaSelRes***)(selRes))->sak & **RFAL_NFCA_SEL_RES_CONF_MASK**) == **RFAL_NFCA_SEL_RES_CONF_T4T_NFCDEP**)

#define **rfalNfcaLisDevIsMultiProto**(lisDev) (((**rfalNfcaListenDevice***)(lisDev))->type == RFAL_NFCA_T4T_NFCDEP)

Enumerations

enum **rfalNfcaListenDeviceType** {
RFAL_NFCA_T1T = 0x01 , **RFAL_NFCA_T2T** = 0x00 , **RFAL_NFCA_T4T** = 0x20 , **RFAL_NFCA_NFCDEP** = 0x40 ,
RFAL_NFCA_T4T_NFCDEP = 0x60
}

Functions

ReturnCode **rfalNfcaPollerInitialize** (void)
Initialize NFC-A Poller mode. More...

ReturnCode **rfalNfcaPollerCheckPresence** (**rfal14443AShortFrameCmd** cmd, **rfalNfcaSensRes** *sensRes)
NFC-A Poller Check Presence. More...

ReturnCode **rfalNfcaPollerSelect** (const uint8_t *nfcid1, uint8_t nfcidLen, **rfalNfcaSelRes** *selRes)
NFC-A Poller Select. More...

ReturnCode **rfalNfcaPollerStartSelect** (const uint8_t *nfcid1, uint8_t nfcidLen, **rfalNfcaSelRes** *selRes)

NFC-A Poller Start Select. More...

ReturnCode **rfalNfcaPollerGetSelectStatus** (void)
NFC-A Poller Get Select Status. More...

ReturnCode **rfalNfcaPollerSleep** (void)
NFC-A Poller Sleep. More...

ReturnCode **rfalNfcaPollerStartSleep** (void)
NFC-A Poller Start Sleep. More...

ReturnCode **rfalNfcaPollerGetSleepStatus** (void)
NFC-A Poller Get Sleep Status. More...

ReturnCode **rfalNfcaPollerTechnologyDetection** (**rfalComplianceMode** compMode, **rfalNfcaSensRes** *sensRes)
NFC-A Technology Detection. More...

ReturnCode **rfalNfcaPollerStartTechnologyDetection** (**rfalComplianceMode** compMode, **rfalNfcaSensRes** *sensRes)
NFC-A Start Technology Detection. More...

ReturnCode **rfalNfcaPollerGetTechnologyDetectionStatus** (void)
NFC-A Get Technology Detection Status. More...

ReturnCode **rfalNfcaPollerSingleCollisionResolution** (uint8_t devLimit, bool *collPending, **rfalNfcaSelRes** *selRes, uint8_t *nfcId1, uint8_t *nfcId1Len)
NFC-A Poller Collision Resolution. More...

ReturnCode **rfalNfcaPollerFullCollisionResolution** (**rfalComplianceMode** compMode, uint8_t devLimit, **rfalNfcaListenDevice** *nfcaDevList, uint8_t *devCnt)
NFC-A Poller Full Collision Resolution. More...

ReturnCode **rfalNfcaPollerSleepFullCollisionResolution** (uint8_t devLimit, **rfalNfcaListenDevice** *nfcaDevList, uint8_t *devCnt)
NFC-A Poller Full Collision Resolution with Sleep. More...

ReturnCode **rfalNfcaPollerStartFullCollisionResolution** (**rfalComplianceMode** compMode, uint8_t devLimit, **rfalNfcaListenDevice** *nfcaDevList, uint8_t *devCnt)
NFC-A Poller Start Full Collision Resolution. More...

ReturnCode	**rfalNfcaPollerGetFullCollisionResolutionStatus** (void) NFC-A Get Full Collision Resolution Status. More...
bool	**rfalNfcaListenerIsSleepReq** (const uint8_t *buf, uint16_t bufLen) NFC-A Listener is SLP_REQ. More...

Detailed Description

Author

Gustavo Patricio

It provides a Poller (ISO14443A PCD) interface as well as some NFC-A Listener (ISO14443A PICC) helpers.

The definitions and helper methods provided by this module are only up to ISO14443-3 layer

An usage example is provided here: **exampleRfalNfca.c**

rfal_nfcb.h File Reference

Implementation of NFC-B (ISO14443B) helpers. More...

```
#include "rfal_platform.h"
#include "rfal_utils.h"
#include "rfal_rf.h"
```

Include dependency graph for rfal_nfcb.h:

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

Data Structures

struct	**rfalNfcbSensbResAppData**
struct	**rfalNfcbSensbResProtocolInfo**
struct	**rfalNfcbSensbRes**
struct	**rfalNfcbListenDevice**

Macros

#define **RFAL_NFCB_FWTSENSB** 7680U

#define **RFAL_NFCB_DFWT** 49152U

#define **RFAL_NFCB_DTPOLL_10** **rfalConvMsTo1fc**(20)

#define **RFAL_NFCB_DTPOLL_20** **rfalConvMsTo1fc**(17)

#define **RFAL_NFCB_AFI** 0x00U

#define **RFAL_NFCB_PARAM** 0x00U

#define **RFAL_NFCB_CRC_LEN** 2U

#define **RFAL_NFCB_NFCID0_LEN** 4U

#define **RFAL_NFCB_CMD_LEN** 1U

#define **RFAL_NFCB_SENSB_RES_LEN** 12U

#define **RFAL_NFCB_SENSB_RES_EXT_LEN** 13U

#define **RFAL_NFCB_SENSB_REQ_ADV_FEATURE** 0x20U

#define **RFAL_NFCB_SENSB_RES_FSCI_MASK** 0x0FU

#define **RFAL_NFCB_SENSB_RES_FSCI_SHIFT** 4U

#define **RFAL_NFCB_SENSB_RES_PROTO_RFU_MASK** 0x08U

#define **RFAL_NFCB_SENSB_RES_PROTO_TR2_MASK** 0x03U

#define **RFAL_NFCB_SENSB_RES_PROTO_TR2_SHIFT** 1U

#define **RFAL_NFCB_SENSB_RES_PROTO_ISO_MASK** 0x01U

#define **RFAL_NFCB_SENSB_RES_FWI_MASK** 0x0FU

#define **RFAL_NFCB_SENSB_RES_FWI_SHIFT** 4U

#define **RFAL_NFCB_SENSB_RES_ADC_MASK** 0x0CU

#define **RFAL_NFCB_SENSB_RES_ADC_ADV_FEATURE_MASK** 0x08U

#define **RFAL_NFCB_SENSB_RES_ADC_PROPRIETARY_MASK** 0x04U

#define **RFAL_NFCB_SENSB_RES_FO_DID_MASK** 0x01U

#define **RFAL_NFCB_SENSB_RES_FO_NAD_MASK** 0x02U

#define **RFAL_NFCB_SENSB_RES_FO_MASK** 0x03U

#define **RFAL_NFCB_SENSB_RES_SFGI_MASK** 0x0FU

#define **RFAL_NFCB_SENSB_RES_SFGI_SHIFT** 4U

#define **rfalNfcbGetFSCI**(sensbRes) ((((**rfalNfcbSensbRes***)(sensbRes))->protInfo.FsciProType >> **RFAL_NFCB_SENSB_RES_FSCI_SHIFT**) & **RFAL_NFCB_SENSB_RES_FSCI_MASK**)

#define **rfalNfcbIsIsoDepSupported**(dev) ((((**rfalNfcbListenDevice***)(dev))->sensbRes.protInfo.FsciProType & **RFAL_NFCB_SENSB_RES_PROTO_ISO_MASK**) != 0U)

Enumerations

enum **rfalNfcbSensCmd** { **RFAL_NFCB_SENS_CMD_ALLB_REQ** = 0x08 , **RFAL_NFCB_SENS_CMD_SENSB_REQ** = 0x00 }

enum **rfalNfcbSlots** {
RFAL_NFCB_SLOT_NUM_1 = 0 , **RFAL_NFCB_SLOT_NUM_2** = 1 , **RFAL_NFCB_SLOT_NUM_4** = 2 , **RFAL_NFCB_SLOT_NUM_8** = 3 ,
RFAL_NFCB_SLOT_NUM_16 = 4
}

Functions

ReturnCode **rfalNfcbPollerInitialize** (void)
Initialize NFC-B Poller mode. More...

ReturnCode **rfalNfcbPollerInitializeWithParams** (uint8_t AFI, uint8_t PARAM)
Set NFC-B Poller parameters. More...

ReturnCode	**rfalNfcbPollerCheckPresence** (**rfalNfcbSensCmd** cmd, **rfalNfcbSlots** slots, **rfalNfcbSensbRes** *sensbRes, uint8_t *sensbResLen) NFC-B Poller Check Presence. More...
ReturnCode	**rfalNfcbPollerStartCheckPresence** (**rfalNfcbSensCmd** cmd, **rfalNfcbSlots** slots, **rfalNfcbSensbRes** *sensbRes, uint8_t *sensbResLen) NFC-B Poller Start Check Presence. More...
ReturnCode	**rfalNfcbPollerGetCheckPresenceStatus** (void) NFC-B Poller Get Check Presence Status. More...
ReturnCode	**rfalNfcbPollerSleep** (const uint8_t *nfcid0) NFC-B Poller Sleep. More...
ReturnCode	**rfalNfcbPollerSlotMarker** (uint8_t slotCode, **rfalNfcbSensbRes** *sensbRes, uint8_t *sensbResLen) NFC-B Poller Slot Marker. More...
ReturnCode	**rfalNfcbPollerStartSlotMarker** (uint8_t slotCode, **rfalNfcbSensbRes** *sensbRes, uint8_t *sensbResLen) NFC-B Poller Start Slot Marker. More...
ReturnCode	**rfalNfcbPollerGetSlotMarkerStatus** (void) NFC-B Poller Get Slot Marker Status. More...
ReturnCode	**rfalNfcbPollerTechnologyDetection** (**rfalComplianceMode** compMode, **rfalNfcbSensbRes** *sensbRes, uint8_t *sensbResLen) NFC-B Technology Detection. More...
ReturnCode	**rfalNfcbPollerStartTechnologyDetection** (**rfalComplianceMode** compMode, **rfalNfcbSensbRes** *sensbRes, uint8_t *sensbResLen) NFC-B Start Technology Detection. More...
ReturnCode	**rfalNfcbPollerGetTechnologyDetectionStatus** (void) NFC-B Get Technology Detection Status. More...
ReturnCode	**rfalNfcbPollerCollisionResolution** (**rfalComplianceMode** compMode, uint8_t devLimit, **rfalNfcbListenDevice** *nfcbDevList, uint8_t *devCnt) NFC-B Poller Collision Resolution. More...

ReturnCode	**rfalNfcbPollerSlottedCollisionResolution** (**rfalComplianceMode** compMode, uint8_t devLimit, **rfalNfcbSlots** initSlots, **rfalNfcbSlots** endSlots, **rfalNfcbListenDevice** *nfcbDevList, uint8_t *devCnt, bool *colPending) NFC-B Poller Collision Resolution Slotted. More...
ReturnCode	**rfalNfcbPollerStartCollisionResolution** (**rfalComplianceMode** compMode, uint8_t devLimit, **rfalNfcbListenDevice** *nfcbDevList, uint8_t *devCnt) NFC-B Poller Start Collision Resolution. More...
ReturnCode	**rfalNfcbPollerStartSlottedCollisionResolution** (**rfalComplianceMode** compMode, uint8_t devLimit, **rfalNfcbSlots** initSlots, **rfalNfcbSlots** endSlots, **rfalNfcbListenDevice** *nfcbDevList, uint8_t *devCnt, bool *colPending) NFC-B Poller Start Collision Resolution Slotted. More...
ReturnCode	**rfalNfcbPollerGetCollisionResolutionStatus** (void) NFC-B Get Collision Resolution Status. More...
uint32_t	**rfalNfcbTR2ToFDT** (uint8_t tr2Code) NFC-B TR2 code to FDT. More...

Detailed Description

Author

Gustavo Patricio

It provides a NFC-B Poller (ISO14443B PCD) interface and also provides some NFC-B Listener (ISO14443B PICC) helpers

The definitions and helpers methods provided by this module are only up to ISO14443-3 layer (excluding ATTRIB)

rfal_nfcDep.h File Reference

Implementation of NFC-DEP protocol. More...

```
#include "rfal_platform.h"
#include "rfal_utils.h"
#include "rfal_rf.h"
```

Include dependency graph for rfal_nfcDep.h:

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

Data Structures

struct	**rfalNfcDepConfigs**
struct	**rfalNfcDepAtrReq**
struct	**rfalNfcDepAtrRes**
struct	**rfalNfcDepBufFormat**
struct	**rfalNfcDepPduBufFormat**
union	**rfalNfcDepActivation**
struct	**rfalNfcDepInfo**
struct	**rfalNfcDepDevice**
struct	**rfalNfcDepAtrParam**
struct	**rfalNfcDepListenActvParam**
struct	**rfalNfcDepTargetParam**
struct	**rfalNfcDepTxRxParam**
struct	**rfalNfcDepPduTxRxParam**

Macros

#define	**RFAL_FEATURE_NFC_DEP_BLOCK_MAX_LEN** 1U
#define	**RFAL_FEATURE_NFC_DEP_PDU_MAX_LEN** 1U
#define	**RFAL_NFCDEP_FRAME_SIZE_MAX_LEN** 254U
#define	**RFAL_NFCDEP_DEPREQ_HEADER_LEN** 5U
#define	**RFAL_NFCDEP_DEP_HEADER** (**RFAL_NFCDEP_LEN_LEN** + **RFAL_NFCDEP_CMDTYPE_LEN** + **RFAL_NFCDEP_CMD_LEN** + **RFAL_NFCDEP_DEP_PFB_LEN**)

```
#define RFAL_NFCDEP_HEADER ( RFAL_NFCDEP_CMDTYPE_LEN + RFAL_NFCDEP_CMD_LEN )
#define RFAL_NFCDEP_SB_LEN 1U
#define RFAL_NFCDEP_LEN_LEN 1U
#define RFAL_NFCDEP_CMDTYPE_LEN 1U
#define RFAL_NFCDEP_CMD_LEN 1U
#define RFAL_NFCDEP_DID_LEN 1U
#define RFAL_NFCDEP_DEP_PFB_LEN 1U
#define RFAL_NFCDEP_DSL_RLS_LEN_NO_DID (RFAL_NFCDEP_LEN_LEN + RFAL_NFCDEP_CMDTYPE_LEN +
        RFAL_NFCDEP_CMD_LEN)
#define RFAL_NFCDEP_DSL_RLS_LEN_DID (RFAL_NFCDEP_DSL_RLS_LEN_NO_DID + RFAL_NFCDEP_DID_LEN)
#define RFAL_NFCDEP_FS_VAL_MIN 64U
#define RFAL_NFCDEP_LR_VAL_MASK 0x03U
#define RFAL_NFCDEP_PP_LR_MASK 0x30U
#define RFAL_NFCDEP_PP_LR_SHIFT 4U
#define RFAL_NFCDEP_DID_MAX 14U
#define RFAL_NFCDEP_DID_KEEP 0xFFU
#define RFAL_NFCDEP_DID_NO 0x00U
#define RFAL_NFCDEP_NAD_NO 0x00U
#define RFAL_NFCDEP_OPER_RTOX_REQ_DIS 0x01U
#define RFAL_NFCDEP_OPER_RTOX_REQ_EN 0x00U
#define RFAL_NFCDEP_OPER_ATN_DIS 0x00U
#define RFAL_NFCDEP_OPER_ATN_EN 0x02U
```

#define **RFAL_NFCDEP_OPER_EMPTY_DEP_DIS** 0x04U

#define **RFAL_NFCDEP_OPER_EMPTY_DEP_EN** 0x00U

#define **RFAL_NFCDEP_OPER_FULL_MI_DIS** 0x00U

#define **RFAL_NFCDEP_OPER_FULL_MI_EN** 0x08U

#define **RFAL_NFCDEP_BRS_MAINTAIN** 0xC0U

#define **RFAL_NFCDEP_BRS_Dx_MASK** 0x07U

#define **RFAL_NFCDEP_BRS_DSI_POS** 3U

#define **RFAL_NFCDEP_WT_DELTA** (16U - **RFAL_NFCDEP_WT_DELTA_ADJUST**)

#define **RFAL_NFCDEP_WT_DELTA_ADJUST** 4U

#define **RFAL_NFCDEP_ATR_REQ_NFCID3_POS** 2U

#define **RFAL_NFCDEP_NFCID3_LEN** 10U

#define **RFAL_NFCDEP_LEN_MIN** 3U

#define **RFAL_NFCDEP_LEN_MAX** 255U

#define **RFAL_NFCDEP_ATRRES_HEADER_LEN** 2U

#define **RFAL_NFCDEP_ATRRES_MIN_LEN** 17U

#define **RFAL_NFCDEP_ATRRES_MAX_LEN** 64U

#define **RFAL_NFCDEP_ATRREQ_MIN_LEN** 16U

#define **RFAL_NFCDEP_ATRREQ_MAX_LEN** **RFAL_NFCDEP_ATRRES_MAX_LEN**

#define **RFAL_NFCDEP_GB_MAX_LEN** (**RFAL_NFCDEP_ATRREQ_MAX_LEN** - **RFAL_NFCDEP_ATRREQ_MIN_LEN**)

#define **RFAL_NFCDEP_WT_INI_DEFAULT** **RFAL_NFCDEP_WT_INI_MAX**

#define **RFAL_NFCDEP_WT_INI_MIN** 0U

```
#define RFAL_NFCDEP_WT_INI_MAX 14U
#define RFAL_NFCDEP_RWT_INI_MAX rfalNfcDepWT2RWT( RFAL_NFCDEP_WT_INI_MAX )
#define RFAL_NFCDEP_WT_TRG_MAX_D10 8U
#define RFAL_NFCDEP_WT_TRG_MAX_D11 14U
#define RFAL_NFCDEP_WT_TRG_MAX_L13 10U
#define RFAL_NFCDEP_WT_TRG_MAX RFAL_NFCDEP_WT_TRG_MAX_D11
#define RFAL_NFCDEP_RWT_TRG_MAX rfalNfcDepWT2RWT( RFAL_NFCDEP_WT_TRG_MAX )
#define RFAL_NFCDEP_MAX_FWT ((uint32_t)1U<<20)
#define RFAL_NFCDEP_WT_MASK 0x0FU
#define RFAL_NFCDEP_BR_MASK_106 0x01U
#define RFAL_NFCDEP_BR_MASK_212 0x02U
#define RFAL_NFCDEP_BR_MASK_424 0x04U
#define rfalNfcDepWT2RWT(wt) ( (uint32_t)1U << (((uint32_t)(wt) & RFAL_NFCDEP_WT_MASK) + 12U) )
#define rfalNfcDepDx2BRS(br) ( (((uint8_t)(br) & RFAL_NFCDEP_BRS_Dx_MASK) << RFAL_NFCDEP_BRS_DSI_POS)
        | ((uint8_t)(br) & RFAL_NFCDEP_BRS_Dx_MASK) )
#define rfalNfcDepBRS2DRI(brs) (uint8_t)( (uint8_t)(brs) & RFAL_NFCDEP_BRS_Dx_MASK )
#define rfalNfcDepBRS2DSI(brs) (uint8_t)( ((uint8_t)(brs) >> RFAL_NFCDEP_BRS_DSI_POS) &
        RFAL_NFCDEP_BRS_Dx_MASK )
#define rfalNfcDepPP2LR(PPx) ( ((uint8_t)(PPx) & RFAL_NFCDEP_PP_LR_MASK ) >>
        RFAL_NFCDEP_PP_LR_SHIFT)
#define rfalNfcDepLR2PP(LRx) ( ((uint8_t)(LRx) << RFAL_NFCDEP_PP_LR_SHIFT) & RFAL_NFCDEP_PP_LR_MASK)
#define rfalNfcDepLR2FS(LRx) (uint16_t)(RFAL_MIN( (RFAL_NFCDEP_FS_VAL_MIN * ((uint16_t)(LRx) + 1U) ),
        RFAL_NFCDEP_FRAME_SIZE_MAX_LEN ))
```

#define **rfalNfcDepSetNFCID**(dst, src, len)

Typedefs

typedef bool(* **rfalNfcDepDeactCallback**) (void)

Enumerations

enum {
RFAL_NFCDEP_Bx_NO_HIGH_BR = 0x00 , **RFAL_NFCDEP_Bx_08_848** = 0x01 , **RFAL_NFCDEP_Bx_16_1695** = 0x02 , **RFAL_NFCDEP_Bx_32_3390** = 0x04 ,
RFAL_NFCDEP_Bx_64_6780 = 0x08
}

enum {
RFAL_NFCDEP_Dx_01_106 = RFAL_BR_106 , **RFAL_NFCDEP_Dx_02_212** = RFAL_BR_212 , **RFAL_NFCDEP_Dx_04_424** = RFAL_BR_424 , **RFAL_NFCDEP_Dx_08_848** = RFAL_BR_848 ,
RFAL_NFCDEP_Dx_16_1695 = RFAL_BR_1695 , **RFAL_NFCDEP_Dx_32_3390** = RFAL_BR_3390 , **RFAL_NFCDEP_Dx_64_6780** = RFAL_BR_6780
}

enum { **RFAL_NFCDEP_LR_64** = 0x00 , **RFAL_NFCDEP_LR_128** = 0x01 , **RFAL_NFCDEP_LR_192** = 0x02 , **RFAL_NFCDEP_LR_254** = 0x03 }

enum **rfalNfcDepCommMode** { **RFAL_NFCDEP_COMM_PASSIVE** , **RFAL_NFCDEP_COMM_ACTIVE** }

enum **rfalNfcDepRole** { **RFAL_NFCDEP_ROLE_INITIATOR** , **RFAL_NFCDEP_ROLE_TARGET** }

Functions

void **rfalNfcDepInitialize** (void)
NFCIP Initialize. More...

void **rfalNfcDepSetDeactivatingCallback** (**rfalNfcDepDeactCallback** pFunc)
Set deactivating callback. More...

uint32_t **rfalNfcDepCalculateRWT** (uint8_t wt)
Calculate Response Waiting Time. More...

ReturnCode	**rfalNfcDepATR** (const **rfalNfcDepAtrParam** *param, **rfalNfcDepAtrRes** *atrRes, uint8_t *atrResLen) NFC-DEP Initiator ATR (Attribute Request) More...
ReturnCode	**rfalNfcDepPSL** (uint8_t BRS, uint8_t FSL) NFC-DEP Initiator PSL (Parameter Selection) More...
ReturnCode	**rfalNfcDepDSL** (void) NFC-DEP Initiator DSL (Deselect) More...
ReturnCode	**rfalNfcDepRLS** (void) NFC-DEP Initiator RLS (Release) More...
ReturnCode	**rfalNfcDepInitiatorHandleActivation** (**rfalNfcDepAtrParam** *param, **rfalBitRate** desiredBR, **rfalNfcDepDevice** *nfcDepDev) NFC-DEP Initiator Handle Activation. More...
bool	**rfalNfcDepIsAtrReq** (const uint8_t *buf, uint16_t bufLen, uint8_t *nfcid3) Check if buffer contains valid ATR_REQ. More...
bool	**rfalNfcDepTargetRcvdATR** (void) Check is Target has received ATR. More...
ReturnCode	**rfalNfcDepListenStartActivation** (const **rfalNfcDepTargetParam** *param, const uint8_t *atrReq, uint16_t atrReqLength, **rfalNfcDepListenActvParam** rxParam) NFCDEP Start Listen Activation Handling. More...
ReturnCode	**rfalNfcDepListenGetActivationStatus** (void) Get the current NFC-DEP Activation Status. More...
ReturnCode	**rfalNfcDepStartTransceive** (const **rfalNfcDepTxRxParam** *param) Start Transceive. More...
ReturnCode	**rfalNfcDepGetTransceiveStatus** (void) Return the Transceive status. More...
ReturnCode	**rfalNfcDepStartPduTransceive** (**rfalNfcDepPduTxRxParam** param) Start PDU Transceive. More...

ReturnCode	**rfalNfcDepGetPduTransceiveStatus** (void) Return the PDU Transceive status. More...

Detailed Description

Author

Gustavo Patricio

NFC-DEP is also known as NFCIP - Near Field Communication Interface and Protocol

This implementation was based on the following specs:

- NFC Forum Digital 1.1
- ECMA 340 3rd Edition 2013

rfal_nfcf.h File Reference

Implementation of NFC-F Poller (FeliCa PCD) device. More...

```
#include "rfal_platform.h"
#include "rfal_utils.h"
#include "rfal_rf.h"
```

Include dependency graph for rfal_nfcf.h:

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

Data Structures

struct	**rfalNfcfSensfRes**
struct	**rfalNfcfPollDevice**
struct	**rfalNfcfListenDevice**
struct	**rfalNfcfBlockListElem**
struct	**rfalNfcfServBlockListParam**

Macros

#define	**RFAL_NFCF_NFCID2_LEN**	8U
#define	**RFAL_NFCF_SENSF_RES_LEN_MIN**	16U

```
#define RFAL_NFCF_SENSF_RES_LEN_MAX 18U
#define RFAL_NFCF_SENSF_RES_PAD0_LEN 2U
#define RFAL_NFCF_SENSF_RES_PAD1_LEN 3U
#define RFAL_NFCF_SENSF_RES_RD_LEN 2U
#define RFAL_NFCF_SENSF_RES_BYTE1 1U
#define RFAL_NFCF_SENSF_SC_LEN 2U
#define RFAL_NFCF_SENSF_PARAMS_SC1_POS 0U
#define RFAL_NFCF_SENSF_PARAMS_SC2_POS 1U
#define RFAL_NFCF_SENSF_PARAMS_RC_POS 2U
#define RFAL_NFCF_SENSF_PARAMS_TSN_POS 3U
#define RFAL_NFCF_POLL_MAXCARDS 16U
#define RFAL_NFCF_CMD_POS 0U
#define RFAL_NFCF_CMD_LEN 1U
#define RFAL_NFCF_LENGTH_LEN 1U
#define RFAL_NFCF_HEADER_LEN (RFAL_NFCF_LENGTH_LEN + RFAL_NFCF_CMD_LEN)
#define RFAL_NFCF_NOS_LEN 1U
#define RFAL_NFCF_NOB_LEN 1U
#define RFAL_NFCF_SENSF_NFCID2_BYTE1_POS 0U
#define RFAL_NFCF_SENSF_NFCID2_BYTE2_POS 1U
#define RFAL_NFCF_SENSF_NFCID2_PROT_TYPE_LEN 2U
#define RFAL_NFCF_SENSF_NFCID2_BYTE1_NFCDEP 0x01U
```

#define **RFAL_NFCF_SENSF_NFCID2_BYTE2_NFCDEP** 0xFEU

#define **RFAL_NFCF_SYSTEMCODE** 0xFFFFU

#define **RFAL_NFCF_SYSTEMCODE_LEN** 2U

#define **RFAL_NFCF_BLOCK_LEN** 16U

#define **RFAL_NFCF_CHECKUPDATE_RES_ST1_POS** 9U

#define **RFAL_NFCF_CHECKUPDATE_RES_ST2_POS** 10U

#define **RFAL_NFCF_CHECKUPDATE_RES_NOB_POS** 11U

#define **RFAL_NFCF_STATUS_FLAG_SUCCESS** 0x00U

#define **RFAL_NFCF_STATUS_FLAG_ERROR** 0xFFU

#define **RFAL_NFCF_BLOCKLISTELEM_MAX_LEN** 3U

#define **RFAL_NFCF_BLOCKLISTELEM_LEN_BIT** 0x80U

#define **RFAL_NFCF_SERVICECODE_RDONLY** 0x000BU

#define **RFAL_NFCF_SERVICECODE_RDWR** 0x0009U

#define **RFAL_NFCF_TEST_LB_CMD0** 0xD8U

#define **RFAL_NFCF_TEST_LB_CMD1** 0x00U

#define **rfalNfcfIsNfcDepSupported**(dev)

Typedefs

typedef uint16_t **rfalNfcfServ**

Enumerations

enum {
RFAL_NFCF_CMD_POLLING = 0x00 , **RFAL_NFCF_CMD_POLLING_RES** = 0x01 ,
RFAL_NFCF_CMD_REQUEST_SERVICE = 0x02 , **RFAL_NFCF_CMD_REQUEST_RESPONSE** = 0x04 ,

```
  RFAL_NFCF_CMD_READ_WITHOUT_ENCRYPTION = 0x06 ,
RFAL_NFCF_CMD_READ_WITHOUT_ENCRYPTION_RES = 0x07 ,
RFAL_NFCF_CMD_WRITE_WITHOUT_ENCRYPTION = 0x08 ,
RFAL_NFCF_CMD_WRITE_WITHOUT_ENCRYPTION_RES = 0x09 ,
  RFAL_NFCF_CMD_REQUEST_SYSTEM_CODE = 0x0c , RFAL_NFCF_CMD_AUTHENTICATION1 = 0x10 ,
RFAL_NFCF_CMD_AUTHENTICATION2 = 0x12 , RFAL_NFCF_CMD_READ = 0x14 ,
  RFAL_NFCF_CMD_WRITE = 0x16
}
```

Functions

ReturnCode **rfalNfcfPollerInitialize** (**rfalBitRate** bitRate)
Initialize NFC-F Poller mode. More...

ReturnCode **rfalNfcfPollerCheckPresence** (void)
NFC-F Poller Check Presence. More...

ReturnCode **rfalNfcfPollerStartCheckPresence** (void)
NFC-F Poller Start Check Presence. More...

ReturnCode **rfalNfcfPollerGetCheckPresenceStatus** (void)
NFC-F Poller Get Check Presence Status. More...

ReturnCode **rfalNfcfPollerPoll** (**rfalFeliCaPollSlots** slots, uint16_t sysCode, uint8_t reqCode, **rfalFeliCaPollRes** *cardList, uint8_t *devCnt, uint8_t *collisions)
NFC-F Poller Poll. More...

ReturnCode **rfalNfcfPollerCollisionResolution** (**rfalComplianceMode** compMode, uint8_t devLimit, **rfalNfcfListenDevice** *nfcfDevList, uint8_t *devCnt)
NFC-F Poller Full Collision Resolution. More...

ReturnCode **rfalNfcfPollerStartCollisionResolution** (**rfalComplianceMode** compMode, uint8_t devLimit, **rfalNfcfListenDevice** *nfcfDevList, uint8_t *devCnt)
NFC-F Start Poller Collision Resolution. More...

ReturnCode **rfalNfcfPollerGetCollisionResolutionStatus** (void)
NFC-F Poller Get Collision Resolution Status. More...

ReturnCode	**rfalNfcfPollerCheck** (const uint8_t *nfcid2, const **rfalNfcfServBlockListParam** *servBlock, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvdLen) NFC-F Poller Check/Read. More...
ReturnCode	**rfalNfcfPollerUpdate** (const uint8_t *nfcid2, const **rfalNfcfServBlockListParam** *servBlock, uint8_t *txBuf, uint16_t txBufLen, const uint8_t *blockData, uint8_t *rxBuf, uint16_t rxBufLen) NFC-F Poller Update/Write. More...
bool	**rfalNfcfListenerIsT3TReq** (const uint8_t *buf, uint16_t bufLen, uint8_t *nfcid2) NFC-F Listener is T3T Request More...

Detailed Description

Author

Gustavo Patricio

The definitions and helpers methods provided by this module are aligned with NFC-F (FeliCa - JIS X6319-4)

rfal_nfcv.h File Reference

Implementation of NFC-V Poller (ISO15693) device. More...

```
#include "rfal_platform.h"
#include "rfal_utils.h"
#include "rfal_rf.h"
```

Include dependency graph for rfal_nfcv.h:

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

Data Structures

struct **rfalNfcvInventoryRes**

struct **rfalNfcvGenericReq**

struct **rfalNfcvGenericRes**

struct **rfalNfcvListenDevice**

Macros

#define **RFAL_NFCV_UID_LEN** 8U

```
#define RFAL_NFCV_MAX_BLOCK_LEN 32U
```

```
#define RFAL_NFCV_BNO_LEN 1U
```

```
#define RFAL_NFCV_CRC_LEN 2U
```

```
#define RFAL_NFCV_MAX_GEN_DATA_LEN (RFAL_NFCV_MAX_BLOCK_LEN + RFAL_NFCV_BNO_LEN +
RFAL_NFCV_UID_LEN)
```

```
#define RFAL_NFCV_BLOCKNUM_LEN 1U
```

```
#define RFAL_NFCV_BLOCKNUM_EXTENDED_LEN 2U
```

```
#define RFAL_NFCV_PARAM_SKIP 0U
```

```
#define RFAL_NFCV_ST_IC_MFG_CODE 0x02U
```

Enumerations

```
enum {
  RFAL_NFCV_REQ_FLAG_DEFAULT = 0x02U , RFAL_NFCV_REQ_FLAG_SUB_CARRIER = 0x01U ,
RFAL_NFCV_REQ_FLAG_DATA_RATE = 0x02U , RFAL_NFCV_REQ_FLAG_INVENTORY = 0x04U ,
  RFAL_NFCV_REQ_FLAG_PROTOCOL_EXT = 0x08U , RFAL_NFCV_REQ_FLAG_SELECT = 0x10U ,
RFAL_NFCV_REQ_FLAG_ADDRESS = 0x20U , RFAL_NFCV_REQ_FLAG_OPTION = 0x40U ,
  RFAL_NFCV_REQ_FLAG_RFU = 0x80U , RFAL_NFCV_REQ_FLAG_AFI = 0x10U ,
RFAL_NFCV_REQ_FLAG_NB_SLOTS = 0x20U
}
```

```
enum {
  RFAL_NFCV_RES_FLAG_ERROR = 0x01U , RFAL_NFCV_RES_FLAG_RFU1 = 0x02U ,
RFAL_NFCV_RES_FLAG_RFU2 = 0x04U , RFAL_NFCV_RES_FLAG_EXTENSION = 0x08U ,
  RFAL_NFCV_RES_FLAG_RFU3 = 0x10U , RFAL_NFCV_RES_FLAG_RFU4 = 0x20U ,
RFAL_NFCV_RES_FLAG_RFU5 = 0x40U , RFAL_NFCV_RES_FLAG_RFU6 = 0x80U
}
```

```
enum {
  RFAL_NFCV_ERROR_CMD_NOT_SUPPORTED = 0x01U , RFAL_NFCV_ERROR_CMD_NOT_RECOGNIZED =
0x02U , RFAL_NFCV_ERROR_OPTION_NOT_SUPPORTED = 0x03U , RFAL_NFCV_ERROR_UNKNOWN =
0x0FU ,
```

```
 RFAL_NFCV_ERROR_BLOCK_NOT_AVALIABLE = 0x10U , RFAL_NFCV_ERROR_BLOCK_ALREDY_LOCKED
= 0x11U , RFAL_NFCV_ERROR_BLOCK_LOCKED = 0x12U , RFAL_NFCV_ERROR_WRITE_FAILED = 0x13U ,
 RFAL_NFCV_ERROR_BLOCK_FAILED = 0x14U
}
```

```
enum {
 RFAL_NFCV_CMD_INVENTORY = 0x01U , RFAL_NFCV_CMD_SLPV = 0x02U ,
RFAL_NFCV_CMD_READ_SINGLE_BLOCK = 0x20U , RFAL_NFCV_CMD_WRITE_SINGLE_BLOCK = 0x21U ,
 RFAL_NFCV_CMD_LOCK_BLOCK = 0x22U , RFAL_NFCV_CMD_READ_MULTIPLE_BLOCKS = 0x23U ,
RFAL_NFCV_CMD_WRITE_MULTIPLE_BLOCKS = 0x24U , RFAL_NFCV_CMD_SELECT = 0x25U ,
 RFAL_NFCV_CMD_RESET_TO_READY = 0x26U , RFAL_NFCV_CMD_GET_SYS_INFO = 0x2BU ,
RFAL_NFCV_CMD_EXTENDED_READ_SINGLE_BLOCK = 0x30U ,
RFAL_NFCV_CMD_EXTENDED_WRITE_SINGLE_BLOCK = 0x31U ,
 RFAL_NFCV_CMD_EXTENDED_LOCK_SINGLE_BLOCK = 0x32U ,
RFAL_NFCV_CMD_EXTENDED_READ_MULTIPLE_BLOCK = 0x33U ,
RFAL_NFCV_CMD_EXTENDED_WRITE_MULTIPLE_BLOCK = 0x34U ,
RFAL_NFCV_CMD_EXTENDED_GET_SYS_INFO = 0x3BU
}
```

```
enum {
 RFAL_NFCV_CMD_READ_CONFIGURATION = 0xA0U , RFAL_NFCV_CMD_WRITE_CONFIGURATION =
0xA1U , RFAL_NFCV_CMD_SET_EAS = 0xA2U , RFAL_NFCV_CMD_RESET_EAS = 0xA3U ,
 RFAL_NFCV_CMD_LOCK_EAS = 0xA4U , RFAL_NFCV_CMD_ENABLE_EAS = 0xA5U ,
RFAL_NFCV_CMD_KILL = 0xA6U , RFAL_NFCV_CMD_WRITE_EAS_ID = 0xA7U ,
 RFAL_NFCV_CMD_WRITE_EAS_CONFIG = 0xA8U , RFAL_NFCV_CMD_MANAGE_GPO = 0xA9U ,
RFAL_NFCV_CMD_WRITE_MESSAGE = 0xAAU , RFAL_NFCV_CMD_READ_MESSAGE_LENGTH = 0xABU ,
 RFAL_NFCV_CMD_READ_MESSAGE = 0xACU , RFAL_NFCV_CMD_READ_DYN_CONFIGURATION = 0xADU ,
RFAL_NFCV_CMD_WRITE_DYN_CONFIGURATION = 0xAEU , RFAL_NFCV_CMD_WRITE_PASSWORD =
0xB1U ,
 RFAL_NFCV_CMD_LOCK_KILL = 0xB2U , RFAL_NFCV_CMD_PRESENT_PASSWORD = 0xB3U ,
RFAL_NFCV_CMD_GET_RANDOM_NUMBER = 0xB4U , RFAL_NFCV_CMD_FAST_READ_SINGLE_BLOCK =
0xC0U ,
 RFAL_NFCV_CMD_FAST_READ_MULTIPLE_BLOCKS = 0xC3U ,
RFAL_NFCV_CMD_FAST_EXTENDED_READ_SINGLE_BLOCK = 0xC4U ,
RFAL_NFCV_CMD_FAST_EXTENDED_READ_MULTIPLE_BLOCKS = 0xC5U ,
RFAL_NFCV_CMD_FAST_WRITE_MESSAGE = 0xCAU ,
 RFAL_NFCV_CMD_FAST_READ_MESSAGE_LENGTH = 0xCBU , RFAL_NFCV_CMD_FAST_READ_MESSAGE
= 0xCCU , RFAL_NFCV_CMD_FAST_READ_DYN_CONFIGURATION = 0xCDU ,
```

RFAL_NFCV_CMD_FAST_WRITE_DYN_CONFIGURATION = 0xCEU
}

enum {
RFAL_NFCV_SYSINFO_DFSID = 0x01U , **RFAL_NFCV_SYSINFO_AFI** = 0x02U , **RFAL_NFCV_SYSINFO_MEMSIZE** = 0x04U , **RFAL_NFCV_SYSINFO_ICREF** = 0x08U , **RFAL_NFCV_SYSINFO_MOI** = 0x10U , **RFAL_NFCV_SYSINFO_CMDLIST** = 0x20U , **RFAL_NFCV_SYSINFO_CSI** = 0x40U , **RFAL_NFCV_SYSINFO_REQ_ALL** = 0x7FU
}

enum **rfalNfcvNumSlots** { **RFAL_NFCV_NUM_SLOTS_1** = 0x20 , **RFAL_NFCV_NUM_SLOTS_16** = 0x00 }

Functions

ReturnCode **rfalNfcvPollerInitialize** (void)
Initialize NFC-V Poller mode. More...

ReturnCode **rfalNfcvPollerCheckPresence** (**rfalNfcvInventoryRes** *invRes)
NFC-V Poller Check Presence. More...

ReturnCode **rfalNfcvPollerInventory** (**rfalNfcvNumSlots** nSlots, uint8_t maskLen, const uint8_t *maskVal, **rfalNfcvInventoryRes** *invRes, uint16_t *rcvdLen)
NFC-V Poller Poll. More...

ReturnCode **rfalNfcvPollerCollisionResolution** (**rfalComplianceMode** compMode, uint8_t devLimit, **rfalNfcvListenDevice** *nfcvDevList, uint8_t *devCnt)
NFC-V Poller Full Collision Resolution. More...

ReturnCode **rfalNfcvPollerSleepCollisionResolution** (uint8_t devLimit, **rfalNfcvListenDevice** *nfcvDevList, uint8_t *devCnt)
NFC-V Poller Full Collision Resolution With Sleep. More...

ReturnCode **rfalNfcvPollerSleep** (uint8_t flags, const uint8_t *uid)
NFC-V Poller Sleep. More...

ReturnCode **rfalNfcvPollerSelect** (uint8_t flags, const uint8_t *uid)
NFC-V Poller Select. More...

ReturnCode	**rfalNfcvPollerReadSingleBlock** (uint8_t flags, const uint8_t *uid, uint8_t blockNum, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen) NFC-V Poller Read Single Block. More...
ReturnCode	**rfalNfcvPollerWriteSingleBlock** (uint8_t flags, const uint8_t *uid, uint8_t blockNum, const uint8_t *wrData, uint8_t blockLen) NFC-V Poller Write Single Block. More...
ReturnCode	**rfalNfcvPollerReadMultipleBlocks** (uint8_t flags, const uint8_t *uid, uint8_t firstBlockNum, uint8_t numOfBlocks, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen) NFC-V Poller Read Multiple Blocks. More...
ReturnCode	**rfalNfcvPollerWriteMultipleBlocks** (uint8_t flags, const uint8_t *uid, uint8_t firstBlockNum, uint8_t numOfBlocks, uint8_t *txBuf, uint16_t txBufLen, uint8_t blockLen, const uint8_t *wrData, uint16_t wrDataLen) NFC-V Poller Write Multiple Blocks. More...
ReturnCode	**rfalNfcvPollerLockBlock** (uint8_t flags, const uint8_t *uid, uint8_t blockNum) NFC-V Poller Lock Single Block. More...
ReturnCode	**rfalNfcvPollerExtendedLockSingleBlock** (uint8_t flags, const uint8_t *uid, uint16_t blockNum) NFC-V Poller Extended Lock Single Block. More...
ReturnCode	**rfalNfcvPollerExtendedReadSingleBlock** (uint8_t flags, const uint8_t *uid, uint16_t blockNum, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen) NFC-V Poller Extended Read Single Block. More...
ReturnCode	**rfalNfcvPollerExtendedWriteSingleBlock** (uint8_t flags, const uint8_t *uid, uint16_t blockNum, const uint8_t *wrData, uint8_t blockLen) NFC-V Poller Extended Write Single Block. More...
ReturnCode	**rfalNfcvPollerExtendedReadMultipleBlocks** (uint8_t flags, const uint8_t *uid, uint16_t firstBlockNum, uint16_t numOfBlocks, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen) NFC-V Poller Extended Read Multiple Blocks. More...
ReturnCode	**rfalNfcvPollerExtendedWriteMultipleBlocks** (uint8_t flags, const uint8_t *uid, uint16_t firstBlockNum, uint16_t numOfBlocks, uint8_t *txBuf, uint16_t txBufLen, uint8_t blockLen, const uint8_t *wrData, uint16_t wrDataLen) NFC-V Poller Extended Write Multiple Blocks. More...

ReturnCode	**rfalNfcvPollerGetSystemInformation** (uint8_t flags, const uint8_t *uid, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen) NFC-V Get System Information. More...
ReturnCode	**rfalNfcvPollerExtendedGetSystemInformation** (uint8_t flags, const uint8_t *uid, uint8_t requestField, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen) NFC-V Extended Get System Information. More...
ReturnCode	**rfalNfcvPollerTransceiveReq** (uint8_t cmd, uint8_t flags, uint8_t param, const uint8_t *uid, const uint8_t *data, uint16_t dataLen, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen) NFC-V Transceive Request. More...

Detailed Description

Author

Gustavo Patricio

The definitions and helpers methods provided by this module are aligned with NFC-V Digital 2.1

rfal_rf.h File Reference

RF Abstraction Layer (RFAL) More...

```
#include "rfal_platform.h"
#include "rfal_utils.h"
#include "rfal_features.h"
```

Include dependency graph for rfal_rf.h:

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

Data Structures

struct **rfalTransceiveContext**

struct **rfalLmConfPA**

struct **rfalLmConfPB**

struct **rfalLmConfPF**

Macros

#define **RFAL_VERSION** 0x040000U

```
#define RFAL_FWT_NONE 0xFFFFFFFFU
#define RFAL_GT_NONE RFAL_TIMING_NONE
#define RFAL_TIMING_NONE 0x00U
#define RFAL_1FC_IN_4096FC (uint32_t)4096U
#define RFAL_1FC_IN_2048FC (uint32_t)2048U
#define RFAL_1FC_IN_512FC (uint32_t)512U
#define RFAL_1FC_IN_64FC (uint32_t)64U
#define RFAL_1FC_IN_8FC (uint32_t)8U
#define RFAL_US_IN_MS (uint32_t)1000U
#define RFAL_1MS_IN_1FC (uint32_t)13560U
#define RFAL_BITS_IN_BYTE (uint16_t)8U
#define RFAL_CRC_LEN 2U
#define RFAL_TXRX_FLAGS_DEFAULT ( (uint32_t)RFAL_TXRX_FLAGS_CRC_TX_AUTO |
        (uint32_t)RFAL_TXRX_FLAGS_CRC_RX_REMV | (uint32_t)RFAL_TXRX_FLAGS_NFCIP1_OFF |
        (uint32_t)RFAL_TXRX_FLAGS_AGC_ON | (uint32_t)RFAL_TXRX_FLAGS_PAR_RX_REMV |
        (uint32_t)RFAL_TXRX_FLAGS_PAR_TX_AUTO | (uint32_t)RFAL_TXRX_FLAGS_NFCV_FLAG_AUTO)
#define RFAL_LM_MASK_NFCA ((uint32_t)1U<<(uint8_t)RFAL_MODE_LISTEN_NFCA)
#define RFAL_LM_MASK_NFCB ((uint32_t)1U<<(uint8_t)RFAL_MODE_LISTEN_NFCB)
#define RFAL_LM_MASK_NFCF ((uint32_t)1U<<(uint8_t)RFAL_MODE_LISTEN_NFCF)
#define RFAL_LM_MASK_ACTIVE_P2P ((uint32_t)1U<<(uint8_t)RFAL_MODE_LISTEN_ACTIVE_P2P)
#define RFAL_LM_SENS_RES_LEN 2U
#define RFAL_LM_SENSB_RES_LEN 13U
#define RFAL_LM_SENSF_RES_LEN 19U
```

```
#define RFAL_LM_SENSF_SC_LEN 2U
```

```
#define RFAL_NFCID3_LEN 10U
```

```
#define RFAL_NFCID2_LEN 8U
```

```
#define RFAL_NFCID1_TRIPLE_LEN 10U
```

```
#define RFAL_NFCID1_DOUBLE_LEN 7U
```

```
#define RFAL_NFCID1_SINGLE_LEN 4U
```

```
#define rfalGetMaxBrRW() ( ((RFAL_SUPPORT_BR_RW_6780) ? RFAL_BR_6780 : ((RFAL_SUPPORT_BR_RW_3390)
? RFAL_BR_3390 : ((RFAL_SUPPORT_BR_RW_1695) ? RFAL_BR_1695 : ((RFAL_SUPPORT_BR_RW_848) ?
RFAL_BR_848 : ((RFAL_SUPPORT_BR_RW_424) ? RFAL_BR_424 : ((RFAL_SUPPORT_BR_RW_212) ?
RFAL_BR_212 : RFAL_BR_106 ) ) ) ) ) ) ) )
```

```
#define rfalGetMaxBrAP2P() ( ((RFAL_SUPPORT_BR_AP2P_848) ? RFAL_BR_848 :
((RFAL_SUPPORT_BR_AP2P_424) ? RFAL_BR_424 : ((RFAL_SUPPORT_BR_AP2P_212) ? RFAL_BR_212 :
RFAL_BR_106 ) ) ) ) )
```

```
#define rfalGetMaxBrCEA() ( ((RFAL_SUPPORT_BR_CE_A_848) ? RFAL_BR_848 :
((RFAL_SUPPORT_BR_CE_A_424) ? RFAL_BR_424 : ((RFAL_SUPPORT_BR_CE_A_212) ? RFAL_BR_212 :
RFAL_BR_106 ) ) ) ) )
```

```
#define rfalGetMaxBrCEB() ( ((RFAL_SUPPORT_BR_CE_B_848) ? RFAL_BR_848 :
((RFAL_SUPPORT_BR_CE_B_424) ? RFAL_BR_424 : ((RFAL_SUPPORT_BR_CE_B_212) ? RFAL_BR_212 :
RFAL_BR_106 ) ) ) ) )
```

```
#define rfalGetMaxBrCEF() ( ((RFAL_SUPPORT_BR_CE_F_424) ? RFAL_BR_424 : RFAL_BR_212 ) )
```

```
#define rfalIsModeActiveComm(md) ( ((md) == RFAL_MODE_POLL_ACTIVE_P2P) || ((md) ==
RFAL_MODE_LISTEN_ACTIVE_P2P) )
```

```
#define rfalIsModePassiveComm(md) ( !rfalIsModeActiveComm(md) )
```

```
#define rfalIsModePassiveListen(md) ( ((md) == RFAL_MODE_LISTEN_NFCA) || ((md) ==
RFAL_MODE_LISTEN_NFCB) || ((md) == RFAL_MODE_LISTEN_NFCF) )
```

```
#define rfalIsModePassivePoll(md) ( rfalIsModePassiveComm(md) && (!rfalIsModePassiveListen(md)) )
```

```
#define rfalConv1fcTo8fc(t)  (uint32_t)( (uint32_t)(t) / RFAL_1FC_IN_8FC )
#define rfalConv8fcTo1fc(t)  (uint32_t)( (uint32_t)(t) * RFAL_1FC_IN_8FC )
#define rfalConv1fcTo64fc(t)  (uint32_t)( (uint32_t)(t) / RFAL_1FC_IN_64FC )
#define rfalConv64fcTo1fc(t)  (uint32_t)( (uint32_t)(t) * RFAL_1FC_IN_64FC )
#define rfalConv1fcTo512fc(t)  (uint32_t)( (uint32_t)(t) / RFAL_1FC_IN_512FC )
#define rfalConv512fcTo1fc(t)  (uint32_t)( (uint32_t)(t) * RFAL_1FC_IN_512FC )
#define rfalConv1fcTo2018fc(t)  (uint32_t)( (uint32_t)(t) / RFAL_1FC_IN_2048FC )
#define rfalConv2048fcTo1fc(t)  (uint32_t)( (uint32_t)(t) * RFAL_1FC_IN_2048FC )
#define rfalConv1fcTo4096fc(t)  (uint32_t)( (uint32_t)(t) / RFAL_1FC_IN_4096FC )
#define rfalConv4096fcTo1fc(t)  (uint32_t)( (uint32_t)(t) * RFAL_1FC_IN_4096FC )
#define rfalConv1fcToMs(t)  (uint32_t)( (uint32_t)(t) / RFAL_1MS_IN_1FC )
#define rfalConvMsTo1fc(t)  (uint32_t)( (uint32_t)(t) * RFAL_1MS_IN_1FC )
#define rfalConv1fcToUs(t)  (uint32_t)( ((uint32_t)(t) * RFAL_US_IN_MS) / RFAL_1MS_IN_1FC)
#define rfalConvUsTo1fc(t)  (uint32_t)( ((uint32_t)(t) * RFAL_1MS_IN_1FC) / RFAL_US_IN_MS)
#define rfalConv64fcToMs(t)  (uint32_t)( (uint32_t)(t) / (RFAL_1MS_IN_1FC / RFAL_1FC_IN_64FC) )
#define rfalConvMsTo64fc(t)  (uint32_t)( (uint32_t)(t) * (RFAL_1MS_IN_1FC / RFAL_1FC_IN_64FC) )
#define rfalConvBitsToBytes(n)  (uint16_t)( ((uint16_t)(n)+(RFAL_BITS_IN_BYTE-1U)) / (RFAL_BITS_IN_BYTE) )
#define rfalConvBytesToBits(n)  (uint32_t)( (uint32_t)(n) * (RFAL_BITS_IN_BYTE) )
#define rfalRunBlocking(e, fn)  do{ (e)=(fn); rfalWorker(); }while( (e) == RFAL_ERR_BUSY )
#define rfalCreateByteTxRxContext(ctx, tB, tBL, rB, rBL, rdL, t)
#define rfalCreateByteFlagsTxRxContext(ctx, tB, tBL, rB, rBL, rdL, fl, t)
```

```
#define rfalLogE(...) platformLog(__VA_ARGS__)
#define rfalLogW(...) platformLog(__VA_ARGS__)
#define rfalLogI(...) platformLog(__VA_ARGS__)
#define rfalLogD(...) platformLog(__VA_ARGS__)
#define RFAL_GT_NFCA rfalConvMsTo1fc(5U)
#define RFAL_GT_NFCB rfalConvMsTo1fc(5U)
#define RFAL_GT_NFCF rfalConvMsTo1fc(20U)
#define RFAL_GT_NFCV rfalConvMsTo1fc(5U)
#define RFAL_GT_PICOPASS rfalConvMsTo1fc(1U)
#define RFAL_GT_AP2P rfalConvMsTo1fc(5U)
#define RFAL_GT_AP2P_ADJUSTED rfalConvMsTo1fc(5U+25U)
#define RFAL_FDT_LISTEN_NFCA_POLLER 1172U
#define RFAL_FDT_LISTEN_NFCB_POLLER 1008U
#define RFAL_FDT_LISTEN_NFCF_POLLER 2672U
#define RFAL_FDT_LISTEN_NFCV_POLLER 4310U
#define RFAL_FDT_LISTEN_PICOPASS_POLLER 3400U
#define RFAL_FDT_LISTEN_AP2P_POLLER 64U
#define RFAL_FDT_LISTEN_NFCA_LISTENER 1172U
#define RFAL_FDT_LISTEN_NFCB_LISTENER 1024U
#define RFAL_FDT_LISTEN_NFCF_LISTENER 2688U
#define RFAL_FDT_LISTEN_AP2P_LISTENER 64U
```

#define **RFAL_FDT_POLL_NFCA_POLLER** 6780U

#define **RFAL_FDT_POLL_NFCA_T1T_POLLER** 384U

#define **RFAL_FDT_POLL_NFCB_POLLER** 6780U

#define **RFAL_FDT_POLL_NFCF_POLLER** 6800U

#define **RFAL_FDT_POLL_NFCV_POLLER** 4192U

#define **RFAL_FDT_POLL_PICOPASS_POLLER** 1790U

#define **RFAL_FDT_POLL_AP2P_POLLER** 6800U

#define **RFAL_FELICA_LEN_LEN** 1U

#define **RFAL_FELICA_POLL_REQ_LEN** (**RFAL_FELICA_LEN_LEN** + 1U + 2U + 1U + 1U)

#define **RFAL_FELICA_POLL_RES_LEN** (**RFAL_FELICA_LEN_LEN** + 1U + 8U + 8U + 2U)

#define **RFAL_FELICA_POLL_MAX_SLOTS** 16U

#define **RFAL_WUM_REFERENCE_AUTO** 0xFFU

Typedefs

typedef void(* **rfalUpperLayerCallback**) (void)

typedef void(* **rfalPreTxRxCallback**) (void)

typedef void(* **rfalPostTxRxCallback**) (void)

typedef bool(* **rfalSyncTxRxCallback**) (void)

typedef void(* **rfalLmEonCallback**) (void)

typedef uint8_t **rfalFeliCaPollRes**[**RFAL_FELICA_POLL_RES_LEN**]

Enumerations

```
enum rfalMode {
  RFAL_MODE_NONE = 0 , RFAL_MODE_POLL_NFCA = 1 , RFAL_MODE_POLL_NFCA_T1T = 2 ,
RFAL_MODE_POLL_NFCB = 3 ,
  RFAL_MODE_POLL_B_PRIME = 4 , RFAL_MODE_POLL_B_CTS = 5 , RFAL_MODE_POLL_NFCF = 6 ,
RFAL_MODE_POLL_NFCV = 7 ,
  RFAL_MODE_POLL_PICOPASS = 8 , RFAL_MODE_POLL_ACTIVE_P2P = 9 , RFAL_MODE_LISTEN_NFCA =
10 , RFAL_MODE_LISTEN_NFCB = 11 ,
  RFAL_MODE_LISTEN_NFCF = 12 , RFAL_MODE_LISTEN_ACTIVE_P2P = 13
}
```

```
enum rfalBitRate {
  RFAL_BR_106 = 0 , RFAL_BR_212 = 1 , RFAL_BR_424 = 2 , RFAL_BR_848 = 3 ,
  RFAL_BR_1695 = 4 , RFAL_BR_3390 = 5 , RFAL_BR_6780 = 6 , RFAL_BR_13560 = 7 ,
  RFAL_BR_211p88 = 0xE9 , RFAL_BR_105p94 = 0xEA , RFAL_BR_52p97 = 0xEB , RFAL_BR_26p48 = 0xEC ,
  RFAL_BR_1p66 = 0xED , RFAL_BR_KEEP = 0xFF
}
```

```
enum rfalComplianceMode { RFAL_COMPLIANCE_MODE_NFC , RFAL_COMPLIANCE_MODE_EMV ,
RFAL_COMPLIANCE_MODE_ISO }
```

```
enum rfalState {
  RFAL_STATE_IDLE = 0 , RFAL_STATE_INIT = 1 , RFAL_STATE_MODE_SET = 2 , RFAL_STATE_TXRX = 3 ,
  RFAL_STATE_LM = 4 , RFAL_STATE_WUM = 5
}
```

```
enum rfalTransceiveState {
  RFAL_TXRX_STATE_IDLE = 0 , RFAL_TXRX_STATE_INIT = 1 , RFAL_TXRX_STATE_START = 2 ,
RFAL_TXRX_STATE_TX_IDLE = 11 ,
  RFAL_TXRX_STATE_TX_WAIT_GT = 12 , RFAL_TXRX_STATE_TX_WAIT_FDT = 13 ,
RFAL_TXRX_STATE_TX_PREP_TX = 14 , RFAL_TXRX_STATE_TX_TRANSMIT = 15 ,
  RFAL_TXRX_STATE_TX_WAIT_WL = 16 , RFAL_TXRX_STATE_TX_RELOAD_FIFO = 17 ,
RFAL_TXRX_STATE_TX_WAIT_TXE = 18 , RFAL_TXRX_STATE_TX_DONE = 19 ,
  RFAL_TXRX_STATE_TX_FAIL = 20 , RFAL_TXRX_STATE_RX_IDLE = 81 ,
RFAL_TXRX_STATE_RX_WAIT_EON = 82 , RFAL_TXRX_STATE_RX_WAIT_RXS = 83 ,
  RFAL_TXRX_STATE_RX_WAIT_RXE = 84 , RFAL_TXRX_STATE_RX_READ_FIFO = 85 ,
RFAL_TXRX_STATE_RX_ERR_CHECK = 86 , RFAL_TXRX_STATE_RX_READ_DATA = 87 ,
  RFAL_TXRX_STATE_RX_WAIT_EOF = 88 , RFAL_TXRX_STATE_RX_DONE = 89 ,
RFAL_TXRX_STATE_RX_FAIL = 90
}
```

```
enum {
  RFAL_TXRX_FLAGS_CRC_TX_AUTO = (0U<<0) , RFAL_TXRX_FLAGS_CRC_TX_MANUAL = (1U<<0) ,
RFAL_TXRX_FLAGS_CRC_RX_KEEP = (1U<<1) , RFAL_TXRX_FLAGS_CRC_RX_REMV = (0U<<1) ,
  RFAL_TXRX_FLAGS_NFCIP1_ON = (1U<<2) , RFAL_TXRX_FLAGS_NFCIP1_OFF = (0U<<2) ,
RFAL_TXRX_FLAGS_AGC_OFF = (1U<<3) , RFAL_TXRX_FLAGS_AGC_ON = (0U<<3) ,
  RFAL_TXRX_FLAGS_PAR_RX_KEEP = (1U<<4) , RFAL_TXRX_FLAGS_PAR_RX_REMV = (0U<<4) ,
RFAL_TXRX_FLAGS_PAR_TX_NONE = (1U<<5) , RFAL_TXRX_FLAGS_PAR_TX_AUTO = (0U<<5) ,
  RFAL_TXRX_FLAGS_NFCV_FLAG_MANUAL = (1U<<6) , RFAL_TXRX_FLAGS_NFCV_FLAG_AUTO = (0U<<6)
, RFAL_TXRX_FLAGS_CRC_RX_MANUAL = (1U<<7) , RFAL_TXRX_FLAGS_CRC_RX_AUTO = (0U<<7)
}
```

```
enum rfalEHandling { RFAL_ERRORHANDLING_NONE = 0 , RFAL_ERRORHANDLING_EMD = 1 }
```

```
enum rfal14443AShortFrameCmd { RFAL_14443A_SHORTFRAME_CMD_WUPA = 0x52 ,
RFAL_14443A_SHORTFRAME_CMD_REQA = 0x26 }
```

```
enum { RFAL_FELICA_POLL_RC_NO_REQUEST = 0x00U , RFAL_FELICA_POLL_RC_SYSTEM_CODE = 0x01U ,
RFAL_FELICA_POLL_RC_COM_PERFORMANCE = 0x02U }
```

```
enum rfalFeliCaPollSlots {
  RFAL_FELICA_1_SLOT = 0 , RFAL_FELICA_2_SLOTS = 1 , RFAL_FELICA_4_SLOTS = 3 ,
RFAL_FELICA_8_SLOTS = 7 ,
  RFAL_FELICA_16_SLOTS = 15
}
```

```
enum rfalLmNfcidLen { RFAL_LM_NFCID_LEN_04 = RFAL_NFCID1_SINGLE_LEN , RFAL_LM_NFCID_LEN_07 =
RFAL_NFCID1_DOUBLE_LEN , RFAL_LM_NFCID_LEN_10 = RFAL_NFCID1_TRIPLE_LEN }
```

```
enum rfalLmState {
  RFAL_LM_STATE_NOT_INIT = 0x00 , RFAL_LM_STATE_POWER_OFF = 0x01 , RFAL_LM_STATE_IDLE =
0x02 , RFAL_LM_STATE_READY_A = 0x03 ,
  RFAL_LM_STATE_READY_B = 0x04 , RFAL_LM_STATE_READY_F = 0x05 , RFAL_LM_STATE_ACTIVE_A =
0x06 , RFAL_LM_STATE_CARDEMU_4A = 0x07 ,
  RFAL_LM_STATE_CARDEMU_4B = 0x08 , RFAL_LM_STATE_CARDEMU_3 = 0x09 ,
RFAL_LM_STATE_TARGET_A = 0x0A , RFAL_LM_STATE_TARGET_F = 0x0B ,
  RFAL_LM_STATE_SLEEP_A = 0x0C , RFAL_LM_STATE_SLEEP_B = 0x0D , RFAL_LM_STATE_READY_Ax =
0x0E , RFAL_LM_STATE_ACTIVE_Ax = 0x0F ,
  RFAL_LM_STATE_SLEEP_AF = 0x10
}
```

enum **rfalLpMode** { **RFAL_LP_MODE_PD** = 0 , **RFAL_LP_MODE_HR** = 1 }

enum **rfalWumState** { **RFAL_WUM_STATE_NOT_INIT** = 0x00 , **RFAL_WUM_STATE_INITIALIZING** = 0x01 , **RFAL_WUM_STATE_ENABLED** = 0x02 , **RFAL_WUM_STATE_ENABLED_WOKE** = 0x03 }

Functions

ReturnCode **rfalInitialize** (void)
RFAL Initialize. More...

ReturnCode **rfalCalibrate** (void)
RFAL Calibrate. More...

ReturnCode **rfalAdjustRegulators** (uint16_t *result)
RFAL Adjust Regulators. More...

ReturnCode **rfalSetRegulators** (uint8_t regulation)
RFAL Set Regulators. More...

void **rfalSetUpperLayerCallback** (**rfalUpperLayerCallback** pFunc)
RFAL Set System Callback. More...

void **rfalSetPreTxRxCallback** (**rfalPreTxRxCallback** pFunc)
RFAL Set Pre Tx Callback. More...

void **rfalSetSyncTxRxCallback** (**rfalSyncTxRxCallback** pFunc)
RFAL Sync Pre Tx Callback. More...

void **rfalSetPostTxRxCallback** (**rfalPostTxRxCallback** pFunc)
RFAL Set Post Tx Callback. More...

void **rfalSetLmEonCallback** (**rfalLmEonCallback** pFunc)
RFAL Set LM EON Callback. More...

ReturnCode **rfalDeinitialize** (void)
RFAL Deinitialize. More...

ReturnCode **rfalSetMode** (**rfalMode** mode, **rfalBitRate** txBR, **rfalBitRate** rxBR)

RFAL Set Mode. More...

rfalMode	**rfalGetMode** (void) RFAL Get Mode. More...
ReturnCode	**rfalSetBitRate** (**rfalBitRate** txBR, **rfalBitRate** rxBR) RFAL Set Bit Rate. More...
ReturnCode	**rfalGetBitRate** (**rfalBitRate** *txBR, **rfalBitRate** *rxBR) RFAL Get Bit Rate. More...
void	**rfalSetErrorHandling** (**rfalEHandling** eHandling) Set Error Handling Mode. More...
rfalEHandling	**rfalGetErrorHandling** (void) Get Error Handling Mode. More...
void	**rfalSetObsvMode** (uint32_t txMode, uint32_t rxMode) Set Observation Mode. More...
void	**rfalGetObsvMode** (uint8_t *txMode, uint8_t *rxMode) Get Observation Mode. More...
void	**rfalDisableObsvMode** (void) Disable Observation Mode. More...
void	**rfalSetFDTPoll** (uint32_t FDTPoll) RFAL Set FDT Poll. More...
uint32_t	**rfalGetFDTPoll** (void) RFAL Set FDT Poll. More...
void	**rfalSetFDTListen** (uint32_t FDTListen) RFAL Set FDT Listen. More...
uint32_t	**rfalGetFDTListen** (void) RFAL Set FDT Listen. More...
uint32_t	**rfalGetGT** (void)

RFAL Get GT. More...

void **rfalSetGT** (uint32_t GT)
RFAL Set GT. More...

bool **rfalIsGTExpired** (void)
RFAL Is GT expired. More...

ReturnCode **rfalFieldOnAndStartGT** (void)
RFAL Turn Field On and Start GT. More...

ReturnCode **rfalFieldOff** (void)
RFAL Turn Field Off. More...

ReturnCode **rfalStartTransceive** (const **rfalTransceiveContext** *ctx)
RFAL Set transceive context. More...

rfalTransceiveState **rfalGetTransceiveState** (void)
Get Transceive State. More...

ReturnCode **rfalGetTransceiveStatus** (void)
Get Transceive Status. More...

bool **rfalIsTransceiveInTx** (void)
Is Transceive in Tx. More...

bool **rfalIsTransceiveInRx** (void)
Is Transceive in Rx. More...

ReturnCode **rfalGetTransceiveRSSI** (uint16_t *rssi)
Get Transceive RSSI. More...

bool **rfalIsTransceiveSubcDetected** (void)
Is Transceive Subcarrier Detected. More...

void **rfalWorker** (void)
RFAL Worker. More...

ReturnCode **rfalISO14443ATransceiveShortFrame** (**rfal14443AShortFrameCmd** txCmd, uint8_t *rxBuf, uint8_t rxBufLen, uint16_t *rxRcvdLen, uint32_t fwt)

Transceives an ISO14443A ShortFrame

More...

ReturnCode **rfalISO14443ATransceiveAnticollisionFrame** (uint8_t *buf, uint8_t *bytesToSend, uint8_t *bitsToSend, uint16_t *rxLength, uint32_t fwt)

Sends an ISO14443A Anticollision Frame. More...

ReturnCode **rfalISO14443AStartTransceiveAnticollisionFrame** (uint8_t *buf, uint8_t *bytesToSend, uint8_t *bitsToSend, uint16_t *rxLength, uint32_t fwt)

Start ISO14443A Anticollision Frame transceive. More...

ReturnCode **rfalISO14443AGetTransceiveAnticollisionFrameStatus** (void)

Get ISO14443A Anticollision Frame Status. More...

ReturnCode **rfalFeliCaPoll** (**rfalFeliCaPollSlots** slots, uint16_t sysCode, uint8_t reqCode, **rfalFeliCaPollRes** *pollResList, uint8_t pollResListSize, uint8_t *devicesDetected, uint8_t *collisionsDetected)

FeliCa Poll. More...

ReturnCode **rfalStartFeliCaPoll** (**rfalFeliCaPollSlots** slots, uint16_t sysCode, uint8_t reqCode, **rfalFeliCaPollRes** *pollResList, uint8_t pollResListSize, uint8_t *devicesDetected, uint8_t *collisionsDetected)

Start FeliCa Poll. More...

ReturnCode **rfalGetFeliCaPollStatus** (void)

Get FeliCa Poll Status. More...

ReturnCode **rfalISO15693TransceiveAnticollisionFrame** (uint8_t *txBuf, uint8_t txBufLen, uint8_t *rxBuf, uint8_t rxBufLen, uint16_t *actLen)

Sends an ISO15693 Anticollision Frame. More...

ReturnCode **rfalISO15693TransceiveEOFAnticollision** (uint8_t *rxBuf, uint8_t rxBufLen, uint16_t *actLen)

Sends an ISO15693 Anticollision EOF. More...

ReturnCode **rfalISO15693TransceiveEOF** (uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *actLen)

Sends an ISO15693 EOF. More...

ReturnCode **rfalTransceiveBlockingTx** (uint8_t *txBuf, uint16_t txBufLen, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *actLen, uint32_t flags, uint32_t fwt)

Transceive Blocking Tx. More...

ReturnCode **rfalTransceiveBlockingRx** (void)
Transceive Blocking Rx. More...

ReturnCode **rfalTransceiveBlockingTxRx** (uint8_t *txBuf, uint16_t txBufLen, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *actLen, uint32_t flags, uint32_t fwt)
Transceive Blocking. More...

bool **rfalIsExtFieldOn** (void)
Is external Field On. More...

ReturnCode **rfalListenStart** (uint32_t lmMask, const **rfalLmConfPA** *confA, const **rfalLmConfPB** *confB, const **rfalLmConfPF** *confF, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rxLen)
Listen Mode start. More...

ReturnCode **rfalListenSleepStart** (**rfalLmState** sleepSt, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rxLen)
Listen Mode start Sleeping. More...

ReturnCode **rfalListenStop** (void)
Listen Mode Stop. More...

rfalLmState **rfalListenGetState** (bool *dataFlag, **rfalBitRate** *lastBR)
Listen Mode get state. More...

ReturnCode **rfalListenSetState** (**rfalLmState** newSt)
Listen Mode set state. More...

ReturnCode **rfalWakeUpModeStart** (const **rfalWakeUpConfig** *config)
Wake-Up Mode Start. More...

bool **rfalWakeUpModeHasWoke** (void)
Wake-Up has Woke. More...

bool **rfalWakeUpModeIsEnabled** (void)
Wake-Up is Enabled. More...

ReturnCode **rfalWakeUpModeGetInfo** (bool force, **rfalWakeUpInfo** *info)
Wake-Up Get Info. More...

ReturnCode	**rfalWakeUpModeStop** (void) Wake-Up Mode Stop. More...
ReturnCode	**rfalWlcPWptMonitorStart** (const **rfalWakeUpConfig** *config) WLC-P WPT Monitor Start. More...
ReturnCode	**rfalWlcPWptMonitorStop** (void) WLC-P WPT Monitor Start Stop. More...
bool	**rfalWlcPWptIsFodDetected** (void) WLC-P WPT FOD is Detected. More...
bool	**rfalWlcPWptIsStopDetected** (void) WLC-P WPT Stop is Detected. More...
ReturnCode	**rfalLowPowerModeStart** (**rfalLpMode** mode) Low Power Mode Start. More...
ReturnCode	**rfalLowPowerModeStop** (void) Low Power Mode Stop. More...

Detailed Description

Author

Gustavo Patricio

RFAL (RF Abstraction Layer) provides several functionalities required to perform RF/NFC communications.
The RFAL encapsulates the different ST25 RF ICs (ST25R3911, ST25R3916, etc) into a common and easy to use interface.

It provides interfaces to configure the RF IC, set/get timings, modes, bit rates, specific handlings, execute listen mode, etc.

Furthermore it provides a common interface to perform a Transceive operations. The Transceive can be executed in a blocking or non blocking way.
Additionally few specific Transceive methods are available to cope with the specifics of these particular operations.

The most common interfaces are:

rfalInitialize()

rfalSetFDTPoll()
rfalSetFDTListen()
rfalSetGT()
rfalSetBitRate()
rfalSetMode()
rfalFieldOnAndStartGT()
rfalFieldOff()
rfalStartTransceive()
rfalGetTransceiveStatus()
rfalTransceiveBlockingTxRx()

An usage example is provided here: **exampleRfalPoller.c**

Generated on Fri Mar 14 2025 11:45:08 for RFAL by doxygen 1.9.5

RFAL

rfal_st25tb.h File Reference

Implementation of ST25TB interface. More...

```
#include "rfal_platform.h"
#include "rfal_utils.h"
```

```
#include "rfal_rf.h"
#include "rfal_nfcb.h"
```

Include dependency graph for rfal_st25tb.h:

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

Data Structures

struct **rfalSt25tbListenDevice**

Macros

#define **RFAL_ST25TB_CHIP_ID_LEN** 1U

#define **RFAL_ST25TB_CRC_LEN** 2U

#define **RFAL_ST25TB_UID_LEN** 8U

#define **RFAL_ST25TB_BLOCK_LEN** 4U

Typedefs

typedef uint8_t **rfalSt25tbUID**[**RFAL_ST25TB_UID_LEN**]

typedef uint8_t **rfalSt25tbBlock**[**RFAL_ST25TB_BLOCK_LEN**]

Functions

ReturnCode **rfalSt25tbPollerInitialize** (void)
Initialize ST25TB Poller mode. More...

ReturnCode **rfalSt25tbPollerCheckPresence** (uint8_t *chipId)
ST25TB Poller Check Presence. More...

ReturnCode **rfalSt25tbPollerCollisionResolution** (uint8_t devLimit, **rfalSt25tbListenDevice** *st25tbDevList, uint8_t *devCnt)
ST25TB Poller Collision Resolution. More...

ReturnCode **rfalSt25tbPollerInitiate** (uint8_t *chipId)
ST25TB Poller Initiate. More...

ReturnCode **rfalSt25tbPollerPcall** (uint8_t *chipId)
ST25TB Poller Pcall. More...

ReturnCode **rfalSt25tbPollerSlotMarker** (uint8_t slotNum, uint8_t *chipIdRes)
ST25TB Poller Slot Marker. More...

ReturnCode **rfalSt25tbPollerSelect** (uint8_t chipId)
ST25TB Poller Select. More...

ReturnCode **rfalSt25tbPollerGetUID** (**rfalSt25tbUID** *UID)
ST25TB Get UID. More...

ReturnCode **rfalSt25tbPollerReadBlock** (uint8_t blockAddress, **rfalSt25tbBlock** *blockData)
ST25TB Poller Read Block. More...

ReturnCode **rfalSt25tbPollerWriteBlock** (uint8_t blockAddress, const **rfalSt25tbBlock** *blockData)
ST25TB Poller Write Block. More...

ReturnCode **rfalSt25tbPollerCompletion** (void)
ST25TB Poller Completion. More...

ReturnCode	**rfalSt25tbPollerResetToInventory** (void) ST25TB Poller Reset to Inventory. More...

Detailed Description

Author

Gustavo Patricio

Macros | Functions

rfal_st25xv.h File Reference

NFC-V ST25 NFC-V Tag specific features. More...

```
#include "rfal_platform.h"
#include "rfal_utils.h"
#include "rfal_rf.h"
```

Include dependency graph for rfal_st25xv.h:

Macros

#define **RFAL_NFCV_BLOCKNUM_M24LR_LEN** 2U

Functions

ReturnCode **rfalST25xVPollerM24LRReadSingleBlock** (uint8_t flags, const uint8_t *uid, uint16_t blockNum, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen)

NFC-V Poller Read Single Block (M24LR) More...

ReturnCode **rfalST25xVPollerM24LRFastReadSingleBlock** (uint8_t flags, const uint8_t *uid, uint16_t blockNum, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen)

NFC-V Poller Fast Read Single Block (M24LR) More...

ReturnCode **rfalST25xVPollerM24LRWriteSingleBlock** (uint8_t flags, const uint8_t *uid, uint16_t blockNum, const uint8_t *wrData, uint8_t blockLen)

NFC-V Poller Write Single Block (M24LR) More...

ReturnCode **rfalST25xVPollerM24LRReadMultipleBlocks** (uint8_t flags, const uint8_t *uid, uint16_t firstBlockNum, uint8_t numOfBlocks, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen)

NFC-V Poller Read Multiple Blocks (M24LR) More...

ReturnCode **rfalST25xVPollerM24LRFastReadMultipleBlocks** (uint8_t flags, const uint8_t *uid, uint16_t firstBlockNum, uint8_t numOfBlocks, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen)

NFC-V Poller Fast Read Multiple Blocks (M24LR) More...

ReturnCode **rfalST25xVPollerFastReadSingleBlock** (uint8_t flags, const uint8_t *uid, uint8_t blockNum, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen)

NFC-V Poller Fast Read Single Block. More...

ReturnCode	**rfalST25xVPollerFastReadMultipleBlocks** (uint8_t flags, const uint8_t *uid, uint8_t firstBlockNum, uint8_t numOfBlocks, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen) NFC-V Poller Fast Read Multiple Blocks. More...
ReturnCode	**rfalST25xVPollerFastExtendedReadSingleBlock** (uint8_t flags, const uint8_t *uid, uint16_t blockNum, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen) NFC-V Poller Fast Extended Read Single Block. More...
ReturnCode	**rfalST25xVPollerFastExtReadMultipleBlocks** (uint8_t flags, const uint8_t *uid, uint16_t firstBlockNum, uint16_t numOfBlocks, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen) NFC-V Poller Fast Extended Read Multiple Blocks. More...
ReturnCode	**rfalST25xVPollerReadConfiguration** (uint8_t flags, const uint8_t *uid, uint8_t pointer, uint8_t *regValue) NFC-V Poller Read Configuration. More...
ReturnCode	**rfalST25xVPollerWriteConfiguration** (uint8_t flags, const uint8_t *uid, uint8_t pointer, uint8_t regValue) NFC-V Poller Write Configuration. More...
ReturnCode	**rfalST25xVPollerReadDynamicConfiguration** (uint8_t flags, const uint8_t *uid, uint8_t pointer, uint8_t *regValue) NFC-V Poller Read Dynamic Configuration. More...
ReturnCode	**rfalST25xVPollerWriteDynamicConfiguration** (uint8_t flags, const uint8_t *uid, uint8_t pointer, uint8_t regValue) NFC-V Poller Write Dynamic Configuration. More...
ReturnCode	**rfalST25xVPollerFastReadDynamicConfiguration** (uint8_t flags, const uint8_t *uid, uint8_t pointer, uint8_t *regValue) NFC-V Poller Fast Read Dynamic Configuration. More...
ReturnCode	**rfalST25xVPollerFastWriteDynamicConfiguration** (uint8_t flags, const uint8_t *uid, uint8_t pointer, uint8_t regValue) NFC-V Poller Fast Write Dynamic Configuration. More...
ReturnCode	**rfalST25xVPollerPresentPassword** (uint8_t flags, const uint8_t *uid, uint8_t pwdNum, const uint8_t *pwd, uint8_t pwdLen) NFC-V Poller Present Password. More...

ReturnCode	**rfalST25xVPollerWritePassword** (uint8_t flags, const uint8_t *uid, uint8_t pwdNum, const uint8_t *pwd, uint8_t pwdLen) NFC-V Poller Write Password. More...
ReturnCode	**rfalST25xVPollerGetRandomNumber** (uint8_t flags, const uint8_t *uid, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen) NFC-V Poller Get Random Number. More...
ReturnCode	**rfalST25xVPollerReadMessageLength** (uint8_t flags, const uint8_t *uid, uint8_t *msgLen) NFC-V Poller Read Message length. More...
ReturnCode	**rfalST25xVPollerFastReadMsgLength** (uint8_t flags, const uint8_t *uid, uint8_t *msgLen) NFC-V Poller Fast Read Message length. More...
ReturnCode	**rfalST25xVPollerReadMessage** (uint8_t flags, const uint8_t *uid, uint8_t mbPointer, uint8_t numBytes, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen) NFC-V Poller Read Message. More...
ReturnCode	**rfalST25xVPollerFastReadMessage** (uint8_t flags, const uint8_t *uid, uint8_t mbPointer, uint8_t numBytes, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen) NFC-V Poller Fast Read Message. More...
ReturnCode	**rfalST25xVPollerWriteMessage** (uint8_t flags, const uint8_t *uid, uint8_t msgLen, const uint8_t *msgData, uint8_t *txBuf, uint16_t txBufLen) NFC-V Poller Write Message. More...
ReturnCode	**rfalST25xVPollerFastWriteMessage** (uint8_t flags, const uint8_t *uid, uint8_t msgLen, const uint8_t *msgData, uint8_t *txBuf, uint16_t txBufLen) NFC-V Poller Fast Write Message. More...
ReturnCode	**rfalST25TV512CPollerReadConfiguration** (uint8_t flags, const uint8_t *uid, uint8_t fid, uint8_t pid, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen) NFC-V Poller Read Configuration (ST25TV02KC ST25TV512C version) More...
ReturnCode	**rfalST25TV512CPollerWriteConfiguration** (uint8_t flags, const uint8_t *uid, uint8_t fid, uint8_t pid, const uint8_t *data, uint8_t dataLen) NFC-V Poller Write Configuration (ST25TV02KC ST25TV512C version) More...

Detailed Description

Author

Gustavo Patricio

This module provides support for ST's specific features available on NFC-V (ISO15693) tag families: ST25D, ST25TV, M24LR

rfal_t1t.h File Reference

Provides NFC-A T1T convenience methods and definitions. More...

```
#include "rfal_platform.h"
#include "rfal_utils.h"
#include "rfal_rf.h"
```

Include dependency graph for rfal_t1t.h:

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

Data Structures

struct **rfalT1TRidRes**

Macros

#define **RFAL_T1T_UID_LEN** 4

#define **RFAL_T1T_HR_LENGTH** 2

#define **RFAL_T1T_HR0_NDEF_MASK** 0xF0

#define **RFAL_T1T_HR0_NDEF_SUPPORT** 0x10

Enumerations

enum **rfalT1Tcmds** {
RFAL_T1T_CMD_RID = 0x78 , **RFAL_T1T_CMD_RALL** = 0x00 , **RFAL_T1T_CMD_READ** = 0x01 ,
RFAL_T1T_CMD_WRITE_E = 0x53 ,

```
    RFAL_T1T_CMD_WRITE_NE = 0x1A
}
```

Functions

ReturnCode	**rfalT1TPollerInitialize** (void) Initialize NFC-A T1T Poller mode. More...
ReturnCode	**rfalT1TPollerRid** (**rfalT1TRidRes** *ridRes) NFC-A T1T Poller RID. More...
ReturnCode	**rfalT1TPollerRall** (const uint8_t *uid, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rxRcvdLen) NFC-A T1T Poller RALL. More...
ReturnCode	**rfalT1TPollerWrite** (const uint8_t *uid, uint8_t address, uint8_t data) NFC-A T1T Poller Write. More...

Detailed Description

Author

Gustavo Patricio

This module provides an interface to perform as a NFC-A Reader/Writer to handle a Type 1 Tag T1T (Topaz)

rfal_t2t.h File Reference

Provides NFC-A T2T convenience methods and definitions. More...

```
#include "rfal_platform.h"
#include "rfal_utils.h"
#include "rfal_rf.h"
```

Include dependency graph for rfal_t2t.h:

Macros

#define **RFAL_T2T_BLOCK_LEN** 4U

#define **RFAL_T2T_READ_DATA_LEN** (4U * **RFAL_T2T_BLOCK_LEN**)

#define **RFAL_T2T_WRITE_DATA_LEN** **RFAL_T2T_BLOCK_LEN**

Functions

ReturnCode **rfalT2TPollerRead** (uint8_t blockNum, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rcvLen)

NFC-A T2T Poller Read. More...

ReturnCode	**rfalT2TPollerWrite** (uint8_t blockNum, const uint8_t *wrData) NFC-A T2T Poller Write. More...

ReturnCode	**rfalT2TPollerSectorSelect** (uint8_t sectorNum) NFC-A T2T Poller Sector Select. More...

Detailed Description

Author

Gustavo Patricio

This module provides an interface to perform as a NFC-A Reader/Writer to handle a Type 2 Tag T2T

Generated on Fri Mar 14 2025 11:45:08 for RFAL by doxygen 1.9.5

Data Structures | Macros | Enumerations | Functions

rfal_t4t.h File Reference

Provides convenience methods and definitions for T4T (ISO7816-4) More...

```
#include "rfal_platform.h"
#include "rfal_utils.h"
#include "rfal_rf.h"
#include "rfal_isoDep.h"
```

Include dependency graph for rfal_t4t.h:

Data Structures

struct	**rfalT4tCApduParam**
struct	**rfalT4tRApduParam**

Macros

#define	**RFAL_T4T_MAX_CAPDU_PROLOGUE_LEN**	4U
#define	**RFAL_T4T_LE_LEN**	1U
#define	**RFAL_T4T_LC_LEN**	1U
#define	**RFAL_T4T_MAX_RAPDU_SW1SW2_LEN**	2U
#define	**RFAL_T4T_CLA**	0x00U
#define	**RFAL_T4T_ISO7816_P1_SELECT_BY_DF_NAME**	0x04U

#define **RFAL_T4T_ISO7816_P1_SELECT_BY_FILEID** 0x00U

#define **RFAL_T4T_ISO7816_P2_SELECT_FIRST_OR_ONLY_OCCURENCE** 0x00U

#define **RFAL_T4T_ISO7816_P2_SELECT_RETURN_FCI_TEMPLATE** 0x00U

#define **RFAL_T4T_ISO7816_P2_SELECT_NO_RESPONSE_DATA** 0x0CU

#define **RFAL_T4T_ISO7816_STATUS_COMPLETE** 0x9000U

Enumerations

enum **rfalT4tCmds** {
RFAL_T4T_INS_SELECT = 0xA4U , **RFAL_T4T_INS_READBINARY** = 0xB0U ,
RFAL_T4T_INS_UPDATEBINARY = 0xD6U , **RFAL_T4T_INS_READBINARY_ODO** = 0xB1U ,
RFAL_T4T_INS_UPDATEBINARY_ODO = 0xD7U
}

Functions

ReturnCode **rfalT4TPollerComposeCAPDU** (const **rfalT4tCApduParam** *apduParam)
T4T Compose APDU. More...

ReturnCode **rfalT4TPollerParseRAPDU** (**rfalT4tRApduParam** *apduParam)
T4T Parse R-APDU. More...

ReturnCode **rfalT4TPollerComposeSelectAppl** (**rfalIsoDepApduBufFormat** *cApduBuf, const uint8_t *aid, uint8_t aidLen, uint16_t *cApduLen)
T4T Compose Select Application APDU. More...

ReturnCode **rfalT4TPollerComposeSelectFile** (**rfalIsoDepApduBufFormat** *cApduBuf, const uint8_t *fid, uint8_t fidLen, uint16_t *cApduLen)
T4T Compose Select File APDU. More...

ReturnCode **rfalT4TPollerComposeSelectFileV1Mapping** (**rfalIsoDepApduBufFormat** *cApduBuf, const uint8_t *fid, uint8_t fidLen, uint16_t *cApduLen)
T4T Compose Select File APDU for Mapping Version 1. More...

ReturnCode	**rfalT4TPollerComposeReadData** (**rfalIsoDepApduBufFormat** *cApduBuf, uint16_t offset, uint8_t expLen, uint16_t *cApduLen) T4T Compose Read Data APDU. More...
ReturnCode	**rfalT4TPollerComposeReadDataODO** (**rfalIsoDepApduBufFormat** *cApduBuf, uint32_t offset, uint8_t expLen, uint16_t *cApduLen) T4T Compose Read Data ODO APDU. More...
ReturnCode	**rfalT4TPollerComposeWriteData** (**rfalIsoDepApduBufFormat** *cApduBuf, uint16_t offset, const uint8_t *data, uint8_t dataLen, uint16_t *cApduLen) T4T Compose Write Data APDU. More...
ReturnCode	**rfalT4TPollerComposeWriteDataODO** (**rfalIsoDepApduBufFormat** *cApduBuf, uint32_t offset, const uint8_t *data, uint8_t dataLen, uint16_t *cApduLen) T4T Compose Write Data ODO APDU. More...

Detailed Description

Author

Gustavo Patricio

This module provides an interface to exchange T4T APDUs according to NFC Forum T4T and ISO7816-4

This implementation was based on the following specs:

- ISO/IEC 7816-4 3rd Edition 2013-04-15
- NFC Forum T4T Technical Specification 1.0 2017-08-28

rfal_utils.h File Reference

RF Abstraction Layer (RFAL) Utils. More...

```
#include <stdint.h>
#include <string.h>
```

Include dependency graph for rfal_utils.h:

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

Macros

#define **RFAL_ERR_NONE** ((**ReturnCode**)0U)

#define **RFAL_ERR_NOMEM** ((**ReturnCode**)1U)

#define **RFAL_ERR_BUSY** ((**ReturnCode**)2U)

#define **RFAL_ERR_IO** ((**ReturnCode**)3U)

#define **RFAL_ERR_TIMEOUT** ((**ReturnCode**)4U)

```
#define RFAL_ERR_REQUEST ((ReturnCode)5U)
#define RFAL_ERR_NOMSG ((ReturnCode)6U)
#define RFAL_ERR_PARAM ((ReturnCode)7U)
#define RFAL_ERR_SYSTEM ((ReturnCode)8U)
#define RFAL_ERR_FRAMING ((ReturnCode)9U)
#define RFAL_ERR_OVERRUN ((ReturnCode)10U)
#define RFAL_ERR_PROTO ((ReturnCode)11U)
#define RFAL_ERR_INTERNAL ((ReturnCode)12U)
#define RFAL_ERR_AGAIN ((ReturnCode)13U)
#define RFAL_ERR_MEM_CORRUPT ((ReturnCode)14U)
#define RFAL_ERR_NOT_IMPLEMENTED ((ReturnCode)15U)
#define RFAL_ERR_PC_CORRUPT ((ReturnCode)16U)
#define RFAL_ERR_SEND ((ReturnCode)17U)
#define RFAL_ERR_IGNORE ((ReturnCode)18U)
#define RFAL_ERR_SEMANTIC ((ReturnCode)19U)
#define RFAL_ERR_SYNTAX ((ReturnCode)20U)
#define RFAL_ERR_CRC ((ReturnCode)21U)
#define RFAL_ERR_NOTFOUND ((ReturnCode)22U)
#define RFAL_ERR_NOTUNIQUE ((ReturnCode)23U)
#define RFAL_ERR_NOTSUPP ((ReturnCode)24U)
#define RFAL_ERR_WRITE ((ReturnCode)25U)
```

```
#define RFAL_ERR_FIFO ((ReturnCode)26U)
#define RFAL_ERR_PAR ((ReturnCode)27U)
#define RFAL_ERR_DONE ((ReturnCode)28U)
#define RFAL_ERR_RF_COLLISION ((ReturnCode)29U)
#define RFAL_ERR_HW_OVERRUN ((ReturnCode)30U)
#define RFAL_ERR_RELEASE_REQ ((ReturnCode)31U)
#define RFAL_ERR_SLEEP_REQ ((ReturnCode)32U)
#define RFAL_ERR_WRONG_STATE ((ReturnCode)33U)
#define RFAL_ERR_MAX_RERUNS ((ReturnCode)34U)
#define RFAL_ERR_DISABLED ((ReturnCode)35U)
#define RFAL_ERR_HW_MISMATCH ((ReturnCode)36U)
#define RFAL_ERR_LINK_LOSS ((ReturnCode)37U)
#define RFAL_ERR_INVALID_HANDLE ((ReturnCode)38U)
#define RFAL_ERR_INCOMPLETE_BYTE ((ReturnCode)40U)
#define RFAL_ERR_INCOMPLETE_BYTE_01 ((ReturnCode)41U)
#define RFAL_ERR_INCOMPLETE_BYTE_02 ((ReturnCode)42U)
#define RFAL_ERR_INCOMPLETE_BYTE_03 ((ReturnCode)43U)
#define RFAL_ERR_INCOMPLETE_BYTE_04 ((ReturnCode)44U)
#define RFAL_ERR_INCOMPLETE_BYTE_05 ((ReturnCode)45U)
#define RFAL_ERR_INCOMPLETE_BYTE_06 ((ReturnCode)46U)
#define RFAL_ERR_INCOMPLETE_BYTE_07 ((ReturnCode)47U)
```

#define **RFAL_EXIT_ON_ERR**(r, f)

#define **RFAL_EXIT_ON_BUSY**(r, f)

#define **RFAL_SIZEOF_ARRAY**(a) (sizeof(a) / sizeof((a)[0]))

#define **RFAL_MAX**(a, b) (((a) > (b)) ? (a) : (b))

#define **RFAL_MIN**(a, b) (((a) < (b)) ? (a) : (b))

#define **RFAL_GETU16**(a) (((uint16_t)(a)[0] << 8) | (uint16_t)(a)[1])

#define **RFAL_GETU32**(a) (((uint32_t)(a)[0] << 24) | ((uint32_t)(a)[1] << 16) | ((uint32_t)(a)[2] << 8) | ((uint32_t)(a)[3]))

#define **RFAL_MEMMOVE** memmove

#define **RFAL_MEMCPY** memcpy

#define **RFAL_MEMSET** memset

#define **RFAL_BYTECMP** memcmp

#define **RFAL_NO_WARNING**(v) ((void) (v))

#define **NULL** (void*)0

Typedefs

typedef uint16_t **ReturnCode**

Detailed Description

Author

Gustavo Patricio

source Directory Reference

Directory dependency graph for source:

Directories

directory	st25r200
directory	st25r3911
directory	st25r3916
directory	st25r500
directory	st25r95

Files

file **rfal_crc.h**
CRC calculation module.

file **rfal_iso15693_2.h**
Implementation of ISO-15693-2.

st25r200 Directory Reference

Directory dependency graph for st25r200:

Files

file **st25r200/rfal_features.h**
RFAL Features/Capabilities Definition for ST25R200.

file **st25r200.h**
ST25R200 high level interface.

file **st25r200_com.h**
ST25R200 communication declaration file.

file **st25r200_irq.h**
ST25R200 Interrupt handling.

RFAL

Main Page

Modules

Data Structures

Examples

File List

Globals

st25r200/rfal_features.h File Reference

RFAL Features/Capabilities Definition for ST25R200. More...

```
#include "rfal_platform.h"
```

Include dependency graph for st25r200/rfal_features.h:

Data Structures

struct **rfalWumMeasChannel**

struct **rfalWakeUpConfig**

struct **rfalWakeUpInfo**

Macros

#define **RFAL_SUPPORT_MODE_POLL_NFCA** true

#define **RFAL_SUPPORT_MODE_POLL_NFCB** true

#define **RFAL_SUPPORT_MODE_POLL_NFCF** false

#define **RFAL_SUPPORT_MODE_POLL_NFCV** true

#define **RFAL_SUPPORT_MODE_POLL_ACTIVE_P2P** false

```
#define RFAL_SUPPORT_MODE_LISTEN_NFCA false
#define RFAL_SUPPORT_MODE_LISTEN_NFCB false
#define RFAL_SUPPORT_MODE_LISTEN_NFCF false
#define RFAL_SUPPORT_MODE_LISTEN_ACTIVE_P2P false
#define RFAL_SUPPORT_CE ( RFAL_SUPPORT_MODE_LISTEN_NFCA || RFAL_SUPPORT_MODE_LISTEN_NFCB ||
        RFAL_SUPPORT_MODE_LISTEN_NFCF )
#define RFAL_SUPPORT_RW ( RFAL_SUPPORT_MODE_POLL_NFCA || RFAL_SUPPORT_MODE_POLL_NFCB ||
        RFAL_SUPPORT_MODE_POLL_NFCF || RFAL_SUPPORT_MODE_POLL_NFCV )
#define RFAL_SUPPORT_AP2P ( RFAL_SUPPORT_MODE_POLL_ACTIVE_P2P ||
        RFAL_SUPPORT_MODE_LISTEN_ACTIVE_P2P )
#define RFAL_SUPPORT_BR_RW_26 true
#define RFAL_SUPPORT_BR_RW_53 true
#define RFAL_SUPPORT_BR_RW_106 true
#define RFAL_SUPPORT_BR_RW_212 false
#define RFAL_SUPPORT_BR_RW_424 false
#define RFAL_SUPPORT_BR_RW_848 false
#define RFAL_SUPPORT_BR_RW_1695 false
#define RFAL_SUPPORT_BR_RW_3390 false
#define RFAL_SUPPORT_BR_RW_6780 false
#define RFAL_SUPPORT_BR_RW_13560 false
#define RFAL_SUPPORT_BR_AP2P_106 false
#define RFAL_SUPPORT_BR_AP2P_212 false
#define RFAL_SUPPORT_BR_AP2P_424 false
```

#define **RFAL_SUPPORT_BR_AP2P_848** false

#define **RFAL_SUPPORT_BR_CE_A_106** false

#define **RFAL_SUPPORT_BR_CE_A_212** false

#define **RFAL_SUPPORT_BR_CE_A_424** false

#define **RFAL_SUPPORT_BR_CE_A_848** false

#define **RFAL_SUPPORT_BR_CE_B_106** false

#define **RFAL_SUPPORT_BR_CE_B_212** false

#define **RFAL_SUPPORT_BR_CE_B_424** false

#define **RFAL_SUPPORT_BR_CE_B_848** false

#define **RFAL_SUPPORT_BR_CE_F_212** false

#define **RFAL_SUPPORT_BR_CE_F_424** false

Typedefs

typedef struct **rfalWakeUpConfig** **rfalWakeUpConfig**

Enumerations

enum **rfalWumPeriod** {
RFAL_WUM_PERIOD_10MS = 0x00 , **RFAL_WUM_PERIOD_15MS** = 0x01 , **RFAL_WUM_PERIOD_20MS** = 0x02 , **RFAL_WUM_PERIOD_25MS** = 0x03 ,
RFAL_WUM_PERIOD_40MS = 0x04 , **RFAL_WUM_PERIOD_55MS** = 0x05 , **RFAL_WUM_PERIOD_80MS** = 0x06 , **RFAL_WUM_PERIOD_105MS** = 0x07 ,
RFAL_WUM_PERIOD_155MS = 0x08 , **RFAL_WUM_PERIOD_215MS** = 0x09 , **RFAL_WUM_PERIOD_310MS** = 0x0A , **RFAL_WUM_PERIOD_425MS** = 0x0B ,
RFAL_WUM_PERIOD_620MS = 0x0C , **RFAL_WUM_PERIOD_850MS** = 0x0D , **RFAL_WUM_PERIOD_1240MS** = 0x0E , **RFAL_WUM_PERIOD_1700MS** = 0x0F ,
RFAL_WUM_PERIOD_10MS = 0x00 , **RFAL_WUM_PERIOD_20MS** = 0x01 , **RFAL_WUM_PERIOD_30MS** = 0x02 , **RFAL_WUM_PERIOD_40MS** = 0x03 ,

RFAL_WUM_PERIOD_50MS = 0x04 , RFAL_WUM_PERIOD_60MS = 0x05 , RFAL_WUM_PERIOD_70MS = 0x06 , RFAL_WUM_PERIOD_80MS = 0x07 ,
RFAL_WUM_PERIOD_100MS = 0x10 , RFAL_WUM_PERIOD_200MS = 0x11 , RFAL_WUM_PERIOD_300MS = 0x12 , RFAL_WUM_PERIOD_400MS = 0x13 ,
RFAL_WUM_PERIOD_500MS = 0x14 , RFAL_WUM_PERIOD_600MS = 0x15 , RFAL_WUM_PERIOD_700MS = 0x16 , RFAL_WUM_PERIOD_800MS = 0x17 ,
RFAL_WUM_PERIOD_10MS = 0x00 , RFAL_WUM_PERIOD_20MS = 0x01 , RFAL_WUM_PERIOD_30MS = 0x02 , RFAL_WUM_PERIOD_40MS = 0x03 ,
RFAL_WUM_PERIOD_50MS = 0x04 , RFAL_WUM_PERIOD_60MS = 0x05 , RFAL_WUM_PERIOD_70MS = 0x06 , RFAL_WUM_PERIOD_80MS = 0x07 ,
RFAL_WUM_PERIOD_100MS = 0x10 , RFAL_WUM_PERIOD_200MS = 0x11 , RFAL_WUM_PERIOD_300MS = 0x12 , RFAL_WUM_PERIOD_400MS = 0x13 ,
RFAL_WUM_PERIOD_500MS = 0x14 , RFAL_WUM_PERIOD_600MS = 0x15 , RFAL_WUM_PERIOD_700MS = 0x16 , RFAL_WUM_PERIOD_800MS = 0x17 ,
RFAL_WUM_PERIOD_12MS = 0x00 , RFAL_WUM_PERIOD_25MS = 0x01 , RFAL_WUM_PERIOD_50MS = 0x02 , RFAL_WUM_PERIOD_75MS = 0x03 ,
RFAL_WUM_PERIOD_100MS = 0x04 , RFAL_WUM_PERIOD_125MS = 0x05 , RFAL_WUM_PERIOD_150MS = 0x06 , RFAL_WUM_PERIOD_200MS = 0x07 ,
RFAL_WUM_PERIOD_250MS = 0x08 , RFAL_WUM_PERIOD_300MS = 0x09 , RFAL_WUM_PERIOD_350MS = 0x0A , RFAL_WUM_PERIOD_400MS = 0x0B ,
RFAL_WUM_PERIOD_500MS = 0x0C , RFAL_WUM_PERIOD_800MS = 0x0D , RFAL_WUM_PERIOD_1000MS = 0x0E , RFAL_WUM_PERIOD_INF = 0x0F ,
RFAL_WUM_PERIOD_300MS = 0x12
}

enum **rfalWumAAWeight** {
RFAL_WUM_AA_WEIGHT_4 = 0x00 , RFAL_WUM_AA_WEIGHT_8 = 0x01 , RFAL_WUM_AA_WEIGHT_16 = 0x02 , RFAL_WUM_AA_WEIGHT_32 = 0x03 ,
RFAL_WUM_AA_WEIGHT_4 = 0x00 , RFAL_WUM_AA_WEIGHT_8 = 0x01 , RFAL_WUM_AA_WEIGHT_16 = 0x02 , RFAL_WUM_AA_WEIGHT_32 = 0x03 ,
RFAL_WUM_AA_WEIGHT_4 = 0x00 , RFAL_WUM_AA_WEIGHT_8 = 0x01 , RFAL_WUM_AA_WEIGHT_16 = 0x02 , RFAL_WUM_AA_WEIGHT_32 = 0x03 ,
RFAL_WUM_AA_WEIGHT_4 = 0x00 , RFAL_WUM_AA_WEIGHT_8 = 0x01 , RFAL_WUM_AA_WEIGHT_16 = 0x02 , RFAL_WUM_AA_WEIGHT_32 = 0x03 ,
RFAL_WUM_AA_WEIGHT_50 = 0x04 , RFAL_WUM_AA_WEIGHT_64 = 0x05 , RFAL_WUM_AA_WEIGHT_100 = 0x06 , RFAL_WUM_AA_WEIGHT_128 = 0x07
}

enum **rfalWumMeasDuration** { **RFAL_WUM_MEAS_DUR_26_10** = 0 , **RFAL_WUM_MEAS_DUR_30_14** = 1 , **RFAL_WUM_MEAS_DUR_34_19** = 2 , **RFAL_WUM_MEAS_DUR_44_28** = 3 , **RFAL_WUM_MEAS_DUR_26_10** = 0 , **RFAL_WUM_MEAS_DUR_30_14** = 1 , **RFAL_WUM_MEAS_DUR_34_19** = 2 , **RFAL_WUM_MEAS_DUR_44_28** = 3 }

enum **rfalWumMeasFilter** { **RFAL_WUM_MEAS_FIL_SLOW** = false , **RFAL_WUM_MEAS_FIL_FAST** = true , **RFAL_WUM_MEAS_FIL_SLOW** = false , **RFAL_WUM_MEAS_FIL_FAST** = true }

enum { **RFAL_WUM_TRE_ABOVE** = (1U<<2) , **RFAL_WUM_TRE_BETWEEN** = (1U<<1) , **RFAL_WUM_TRE_BELOW** = (1U<<0) }

Detailed Description

Author

Gustavo Patricio

Macro Definition Documentation

◆ RFAL_SUPPORT_AP2P

#define RFAL_SUPPORT_AP2P (**RFAL_SUPPORT_MODE_POLL_ACTIVE_P2P** || **RFAL_SUPPORT_MODE_LISTEN_ACTIVE_P2P**)

RFAL support for Active P2P (AP2P)

◆ RFAL_SUPPORT_BR_AP2P_106

#define RFAL_SUPPORT_BR_AP2P_106 false

RFAL AP2P 106 Bit Rate support switch

◆ RFAL_SUPPORT_BR_AP2P_212

```
#define RFAL_SUPPORT_BR_AP2P_212   false
```

RFAL AP2P 212 Bit Rate support switch

◆ RFAL_SUPPORT_BR_AP2P_424

```
#define RFAL_SUPPORT_BR_AP2P_424   false
```

RFAL AP2P 424 Bit Rate support switch

◆ RFAL_SUPPORT_BR_AP2P_848

```
#define RFAL_SUPPORT_BR_AP2P_848   false
```

RFAL AP2P 848 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_A_106

```
#define RFAL_SUPPORT_BR_CE_A_106   false
```

RFAL CE A 106 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_A_212

```
#define RFAL_SUPPORT_BR_CE_A_212   false
```

RFAL CE A 212 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_A_424

```
#define RFAL_SUPPORT_BR_CE_A_424   false
```

RFAL CE A 424 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_A_848

```
#define RFAL_SUPPORT_BR_CE_A_848   false
```

RFAL CE A 848 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_B_106

```
#define RFAL_SUPPORT_BR_CE_B_106   false
```

RFAL CE B 106 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_B_212

```
#define RFAL_SUPPORT_BR_CE_B_212   false
```

RFAL CE B 212 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_B_424

```
#define RFAL_SUPPORT_BR_CE_B_424   false
```

RFAL CE B 424 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_B_848

```
#define RFAL_SUPPORT_BR_CE_B_848   false
```

RFAL CE B 848 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_F_212

```
#define RFAL_SUPPORT_BR_CE_F_212   false
```

RFAL CE F 212 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_F_424

```
#define RFAL_SUPPORT_BR_CE_F_424   false
```

RFAL CE F 424 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_106

```
#define RFAL_SUPPORT_BR_RW_106   true
```

RFAL RW 106 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_13560

```
#define RFAL_SUPPORT_BR_RW_13560   false
```

RFAL RW 6780 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_1695

```
#define RFAL_SUPPORT_BR_RW_1695   false
```

RFAL RW 1695 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_212

```
#define RFAL_SUPPORT_BR_RW_212   false
```

RFAL RW 212 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_26

```
#define RFAL_SUPPORT_BR_RW_26   true
```

RFAL RW 26 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_3390

```
#define RFAL_SUPPORT_BR_RW_3390   false
```

RFAL RW 3390 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_424

#define RFAL_SUPPORT_BR_RW_424 false

RFAL RW 424 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_53

#define RFAL_SUPPORT_BR_RW_53 true

RFAL RW 53 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_6780

#define RFAL_SUPPORT_BR_RW_6780 false

RFAL RW 6780 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_848

#define RFAL_SUPPORT_BR_RW_848 false

RFAL RW 848 Bit Rate support switch

◆ RFAL_SUPPORT_CE

#define RFAL_SUPPORT_CE (**RFAL_SUPPORT_MODE_LISTEN_NFCA** || **RFAL_SUPPORT_MODE_LISTEN_NFCB** || **RFAL_SUPPORT_MODE_LISTEN_NFCF**)

RFAL supported Card Emulation (CE)

◆ RFAL_SUPPORT_MODE_LISTEN_ACTIVE_P2P

#define RFAL_SUPPORT_MODE_LISTEN_ACTIVE_P2P false

RFAL Listen AP2P mode support switch

◆ RFAL_SUPPORT_MODE_LISTEN_NFCA

```
#define RFAL_SUPPORT_MODE_LISTEN_NFCA   false
```

RFAL Listen NFCA mode support switch

◆ RFAL_SUPPORT_MODE_LISTEN_NFCB

```
#define RFAL_SUPPORT_MODE_LISTEN_NFCB   false
```

RFAL Listen NFCB mode support switch

◆ RFAL_SUPPORT_MODE_LISTEN_NFCF

```
#define RFAL_SUPPORT_MODE_LISTEN_NFCF   false
```

RFAL Listen NFCF mode support switch

◆ RFAL_SUPPORT_MODE_POLL_ACTIVE_P2P

```
#define RFAL_SUPPORT_MODE_POLL_ACTIVE_P2P   false
```

RFAL Poll AP2P mode support switch

◆ RFAL_SUPPORT_MODE_POLL_NFCA

```
#define RFAL_SUPPORT_MODE_POLL_NFCA   true
```

RFAL Poll NFCA mode support switch

◆ RFAL_SUPPORT_MODE_POLL_NFCB

```
#define RFAL_SUPPORT_MODE_POLL_NFCB   true
```

RFAL Poll NFCB mode support switch

◆ RFAL_SUPPORT_MODE_POLL_NFCF

#define RFAL_SUPPORT_MODE_POLL_NFCF false

RFAL Poll NFCF mode support switch

◆ RFAL_SUPPORT_MODE_POLL_NFCV

#define RFAL_SUPPORT_MODE_POLL_NFCV true

RFAL Poll NFCV mode support switch

◆ RFAL_SUPPORT_RW

#define RFAL_SUPPORT_RW (**RFAL_SUPPORT_MODE_POLL_NFCA** || **RFAL_SUPPORT_MODE_POLL_NFCB** || **RFAL_SUPPORT_MODE_POLL_NFCF** || **RFAL_SUPPORT_MODE_POLL_NFCV**)

RFAL supported Reader/Writer (RW)

Typedef Documentation

◆ rfalWakeUpConfig

typedef struct **rfalWakeUpConfig rfalWakeUpConfig**

RFAL Wake-Up Mode configuration

Enumeration Type Documentation

◆ anonymous enum

anonymous enum

RFAL Wake-Up trigger tresholds for threshold bitmask config

Enumerator	
RFAL_WUM_TRE_ABOVE	Wake-up trigger threshold: above upper limit
RFAL_WUM_TRE_BETWEEN	Wake-up trigger threshold: between upper and lower limit
RFAL_WUM_TRE_BELOW	Wake-up trigger threshold: below lower limit

◆ rfalWumAAWeight

enum **rfalWumAAWeight**

RFAL Wake-Up Period/Timer

Enumerator	
RFAL_WUM_AA_WEIGHT_4	Wake-Up Auto Average Weight 4
RFAL_WUM_AA_WEIGHT_8	Wake-Up Auto Average Weight 8
RFAL_WUM_AA_WEIGHT_16	Wake-Up Auto Average Weight 16
RFAL_WUM_AA_WEIGHT_32	Wake-Up Auto Average Weight 32
RFAL_WUM_AA_WEIGHT_4	Wake-Up Auto Average Weight 4
RFAL_WUM_AA_WEIGHT_8	Wake-Up Auto Average Weight 8
RFAL_WUM_AA_WEIGHT_16	Wake-Up Auto Average Weight 16
RFAL_WUM_AA_WEIGHT_32	Wake-Up Auto Average Weight 32
RFAL_WUM_AA_WEIGHT_4	Wake-Up Auto Average Weight 4
RFAL_WUM_AA_WEIGHT_8	Wake-Up Auto Average Weight 8
RFAL_WUM_AA_WEIGHT_16	Wake-Up Auto Average Weight 16
RFAL_WUM_AA_WEIGHT_32	Wake-Up Auto Average Weight 32
RFAL_WUM_AA_WEIGHT_4	Wake-Up Auto Average Weight 4
RFAL_WUM_AA_WEIGHT_8	Wake-Up Auto Average Weight 8
RFAL_WUM_AA_WEIGHT_16	Wake-Up Auto Average Weight 16
RFAL_WUM_AA_WEIGHT_32	Wake-Up Auto Average Weight 32
RFAL_WUM_AA_WEIGHT_50	Wake-Up Auto Average Weight 50
RFAL_WUM_AA_WEIGHT_64	Wake-Up Auto Average Weight 64

RFAL_WUM_AA_WEIGHT_100	Wake-Up Auto Average Weight 100
RFAL_WUM_AA_WEIGHT_128	Wake-Up Auto Average Weight 128

◆ rfalWumMeasDuration

enum **rfalWumMeasDuration**

RFAL Wake-Up mesurement duration

Enumerator	
RFAL_WUM_MEAS_DUR_26_10	WU measurement duration: 26.0us (slow) / 10.6us (fast)
RFAL_WUM_MEAS_DUR_30_14	WU measurement duration: 29.5us (slow) / 14.2us (fast)
RFAL_WUM_MEAS_DUR_34_19	WU measurement duration: 34.2us (slow) / 18.9us (fast)
RFAL_WUM_MEAS_DUR_44_28	WU measurement duration: 43.7us (slow) / 28.3us (fast)
RFAL_WUM_MEAS_DUR_26_10	WU measurement duration: 26.0us (slow) / 10.6us (fast)
RFAL_WUM_MEAS_DUR_30_14	WU measurement duration: 29.5us (slow) / 14.2us (fast)
RFAL_WUM_MEAS_DUR_34_19	WU measurement duration: 34.2us (slow) / 18.9us (fast)
RFAL_WUM_MEAS_DUR_44_28	WU measurement duration: 43.7us (slow) / 28.3us (fast)

◆ rfalWumMeasFilter

enum **rfalWumMeasFilter**

RFAL Wake-Up mesurement filter

Enumerator	
RFAL_WUM_MEAS_FIL_SLOW	Wake-Up measurement slow filter
RFAL_WUM_MEAS_FIL_FAST	Wake-up measurement fast filter
RFAL_WUM_MEAS_FIL_SLOW	Wake-Up measurement slow filter
RFAL_WUM_MEAS_FIL_FAST	Wake-up measurement fast filter

◆ rfalWumPeriod

enum **rfalWumPeriod**

RFAL Wake-Up Period/Timer

Enumerator	
RFAL_WUM_PERIOD_10MS	Wake-Up timer ~9.7ms
RFAL_WUM_PERIOD_15MS	Wake-Up timer ~13.3ms
RFAL_WUM_PERIOD_20MS	Wake-Up timer ~19.3ms
RFAL_WUM_PERIOD_25MS	Wake-Up timer ~26.6ms
RFAL_WUM_PERIOD_40MS	Wake-Up timer ~38.7ms
RFAL_WUM_PERIOD_55MS	Wake-Up timer ~53.2ms
RFAL_WUM_PERIOD_80MS	Wake-Up timer ~77.3ms
RFAL_WUM_PERIOD_105MS	Wake-Up timer ~106.3ms
RFAL_WUM_PERIOD_155MS	Wake-Up timer ~154.7ms
RFAL_WUM_PERIOD_215MS	Wake-Up timer ~212.7ms
RFAL_WUM_PERIOD_310MS	Wake-Up timer ~309.3ms
RFAL_WUM_PERIOD_425MS	Wake-Up timer ~425.3ms
RFAL_WUM_PERIOD_620MS	Wake-Up timer ~618.6ms
RFAL_WUM_PERIOD_850MS	Wake-Up timer ~850.6ms
RFAL_WUM_PERIOD_1240MS	Wake-Up timer ~1237.3ms
RFAL_WUM_PERIOD_1700MS	Wake-Up timer ~1701.2ms
RFAL_WUM_PERIOD_10MS	Wake-Up timer 10ms
RFAL_WUM_PERIOD_20MS	Wake-Up timer 20ms
RFAL_WUM_PERIOD_30MS	Wake-Up timer 30ms
RFAL_WUM_PERIOD_40MS	Wake-Up timer 40ms
RFAL_WUM_PERIOD_50MS	Wake-Up timer 50ms
RFAL_WUM_PERIOD_60MS	Wake-Up timer 60ms
RFAL_WUM_PERIOD_70MS	Wake-Up timer 70ms
RFAL_WUM_PERIOD_80MS	Wake-Up timer 80ms
RFAL_WUM_PERIOD_100MS	Wake-Up timer 100ms
RFAL_WUM_PERIOD_200MS	Wake-Up timer 200ms

RFAL_WUM_PERIOD_300MS	Wake-Up timer 300ms
RFAL_WUM_PERIOD_400MS	Wake-Up timer 400ms
RFAL_WUM_PERIOD_500MS	Wake-Up timer 500ms
RFAL_WUM_PERIOD_600MS	Wake-Up timer 600ms
RFAL_WUM_PERIOD_700MS	Wake-Up timer 700ms
RFAL_WUM_PERIOD_800MS	Wake-Up timer 800ms
RFAL_WUM_PERIOD_10MS	Wake-Up timer 10ms
RFAL_WUM_PERIOD_20MS	Wake-Up timer 20ms
RFAL_WUM_PERIOD_30MS	Wake-Up timer 30ms
RFAL_WUM_PERIOD_40MS	Wake-Up timer 40ms
RFAL_WUM_PERIOD_50MS	Wake-Up timer 50ms
RFAL_WUM_PERIOD_60MS	Wake-Up timer 60ms
RFAL_WUM_PERIOD_70MS	Wake-Up timer 70ms
RFAL_WUM_PERIOD_80MS	Wake-Up timer 80ms
RFAL_WUM_PERIOD_100MS	Wake-Up timer 100ms
RFAL_WUM_PERIOD_200MS	Wake-Up timer 200ms
RFAL_WUM_PERIOD_300MS	Wake-Up timer 300ms
RFAL_WUM_PERIOD_400MS	Wake-Up timer 400ms
RFAL_WUM_PERIOD_500MS	Wake-Up timer 500ms
RFAL_WUM_PERIOD_600MS	Wake-Up timer 600ms
RFAL_WUM_PERIOD_700MS	Wake-Up timer 700ms
RFAL_WUM_PERIOD_800MS	Wake-Up timer 800ms
RFAL_WUM_PERIOD_12MS	Wake-Up timer ~12.5ms
RFAL_WUM_PERIOD_25MS	Wake-Up timer ~25ms
RFAL_WUM_PERIOD_50MS	Wake-Up timer ~50ms
RFAL_WUM_PERIOD_75MS	Wake-Up timer ~75ms
RFAL_WUM_PERIOD_100MS	Wake-Up timer ~100ms
RFAL_WUM_PERIOD_125MS	Wake-Up timer ~125ms

RFAL_WUM_PERIOD_150MS	Wake-Up timer ~150ms
RFAL_WUM_PERIOD_200MS	Wake-Up timer ~200ms
RFAL_WUM_PERIOD_250MS	Wake-Up timer ~250ms
RFAL_WUM_PERIOD_300MS	Wake-Up timer ~300ms
RFAL_WUM_PERIOD_350MS	Wake-Up timer ~350ms
RFAL_WUM_PERIOD_400MS	Wake-Up timer ~400ms
RFAL_WUM_PERIOD_500MS	Wake-Up timer ~600ms
RFAL_WUM_PERIOD_800MS	Wake-Up timer ~800ms
RFAL_WUM_PERIOD_1000MS	Wake-Up timer ~1000ms
RFAL_WUM_PERIOD_INF	Wake-Up timer Infinite
RFAL_WUM_PERIOD_300MS	Wake-Up timer 300ms

st25r3911/rfal_features.h File Reference

RFAL Features/Capabilities Definition for ST25R3911. More...

```
#include "rfal_platform.h"
```

Include dependency graph for st25r3911/rfal_features.h:

Data Structures

struct	**rfalWakeUpConfig**
struct	**rfalWakeUpInfo**

Macros

#define	**RFAL_SUPPORT_MODE_POLL_NFCA**	true
#define	**RFAL_SUPPORT_MODE_POLL_NFCB**	true
#define	**RFAL_SUPPORT_MODE_POLL_NFCF**	true
#define	**RFAL_SUPPORT_MODE_POLL_NFCV**	true
#define	**RFAL_SUPPORT_MODE_POLL_ACTIVE_P2P**	true
#define	**RFAL_SUPPORT_MODE_LISTEN_NFCA**	false
#define	**RFAL_SUPPORT_MODE_LISTEN_NFCB**	false
#define	**RFAL_SUPPORT_MODE_LISTEN_NFCF**	false
#define	**RFAL_SUPPORT_MODE_LISTEN_ACTIVE_P2P**	true

```
#define RFAL_SUPPORT_CE ( RFAL_SUPPORT_MODE_LISTEN_NFCA || RFAL_SUPPORT_MODE_LISTEN_NFCB ||
        RFAL_SUPPORT_MODE_LISTEN_NFCF )
```

```
#define RFAL_SUPPORT_RW ( RFAL_SUPPORT_MODE_POLL_NFCA || RFAL_SUPPORT_MODE_POLL_NFCB ||
        RFAL_SUPPORT_MODE_POLL_NFCF || RFAL_SUPPORT_MODE_POLL_NFCV )
```

```
#define RFAL_SUPPORT_AP2P ( RFAL_SUPPORT_MODE_POLL_ACTIVE_P2P ||
        RFAL_SUPPORT_MODE_LISTEN_ACTIVE_P2P )
```

```
#define RFAL_SUPPORT_BR_RW_106 true
```

```
#define RFAL_SUPPORT_BR_RW_212 true
```

```
#define RFAL_SUPPORT_BR_RW_424 true
```

```
#define RFAL_SUPPORT_BR_RW_848 true
```

```
#define RFAL_SUPPORT_BR_RW_1695 true
```

```
#define RFAL_SUPPORT_BR_RW_3390 true
```

```
#define RFAL_SUPPORT_BR_RW_6780 false
```

```
#define RFAL_SUPPORT_BR_RW_13560 false
```

```
#define RFAL_SUPPORT_BR_AP2P_106 true
```

```
#define RFAL_SUPPORT_BR_AP2P_212 true
```

```
#define RFAL_SUPPORT_BR_AP2P_424 true
```

```
#define RFAL_SUPPORT_BR_AP2P_848 false
```

```
#define RFAL_SUPPORT_BR_AP2P_1695 false
```

```
#define RFAL_SUPPORT_BR_AP2P_3390 false
```

```
#define RFAL_SUPPORT_BR_AP2P_6780 false
```

```
#define RFAL_SUPPORT_BR_CE_A_106 false
```

```
#define RFAL_SUPPORT_BR_CE_A_212 false
```

```
#define RFAL_SUPPORT_BR_CE_A_424 false
```

```
#define RFAL_SUPPORT_BR_CE_A_848 false
```

```
#define RFAL_SUPPORT_BR_CE_B_106 false
```

```
#define RFAL_SUPPORT_BR_CE_B_212 false
```

```
#define RFAL_SUPPORT_BR_CE_B_424 false
```

```
#define RFAL_SUPPORT_BR_CE_B_848 false
```

```
#define RFAL_SUPPORT_BR_CE_F_212 false
```

```
#define RFAL_SUPPORT_BR_CE_F_424 false
```

Enumerations

```
enum rfalWumPeriod {
  RFAL_WUM_PERIOD_10MS = 0x00 , RFAL_WUM_PERIOD_15MS = 0x01 , RFAL_WUM_PERIOD_20MS = 0x02
, RFAL_WUM_PERIOD_25MS = 0x03 ,
  RFAL_WUM_PERIOD_40MS = 0x04 , RFAL_WUM_PERIOD_55MS = 0x05 , RFAL_WUM_PERIOD_80MS = 0x06
, RFAL_WUM_PERIOD_105MS = 0x07 ,
  RFAL_WUM_PERIOD_155MS = 0x08 , RFAL_WUM_PERIOD_215MS = 0x09 , RFAL_WUM_PERIOD_310MS =
0x0A , RFAL_WUM_PERIOD_425MS = 0x0B ,
  RFAL_WUM_PERIOD_620MS = 0x0C , RFAL_WUM_PERIOD_850MS = 0x0D , RFAL_WUM_PERIOD_1240MS =
0x0E , RFAL_WUM_PERIOD_1700MS = 0x0F ,
  RFAL_WUM_PERIOD_10MS = 0x00 , RFAL_WUM_PERIOD_20MS = 0x01 , RFAL_WUM_PERIOD_30MS = 0x02
, RFAL_WUM_PERIOD_40MS = 0x03 ,
  RFAL_WUM_PERIOD_50MS = 0x04 , RFAL_WUM_PERIOD_60MS = 0x05 , RFAL_WUM_PERIOD_70MS = 0x06
, RFAL_WUM_PERIOD_80MS = 0x07 ,
  RFAL_WUM_PERIOD_100MS = 0x10 , RFAL_WUM_PERIOD_200MS = 0x11 , RFAL_WUM_PERIOD_300MS =
0x12 , RFAL_WUM_PERIOD_400MS = 0x13 ,
  RFAL_WUM_PERIOD_500MS = 0x14 , RFAL_WUM_PERIOD_600MS = 0x15 , RFAL_WUM_PERIOD_700MS =
0x16 , RFAL_WUM_PERIOD_800MS = 0x17 ,
  RFAL_WUM_PERIOD_10MS = 0x00 , RFAL_WUM_PERIOD_20MS = 0x01 , RFAL_WUM_PERIOD_30MS = 0x02
, RFAL_WUM_PERIOD_40MS = 0x03 ,
  RFAL_WUM_PERIOD_50MS = 0x04 , RFAL_WUM_PERIOD_60MS = 0x05 , RFAL_WUM_PERIOD_70MS = 0x06
, RFAL_WUM_PERIOD_80MS = 0x07 ,
```

```
RFAL_WUM_PERIOD_100MS = 0x10 , RFAL_WUM_PERIOD_200MS = 0x11 , RFAL_WUM_PERIOD_300MS = 0x12 , RFAL_WUM_PERIOD_400MS = 0x13 ,
RFAL_WUM_PERIOD_500MS = 0x14 , RFAL_WUM_PERIOD_600MS = 0x15 , RFAL_WUM_PERIOD_700MS = 0x16 , RFAL_WUM_PERIOD_800MS = 0x17 ,
RFAL_WUM_PERIOD_12MS = 0x00 , RFAL_WUM_PERIOD_25MS = 0x01 , RFAL_WUM_PERIOD_50MS = 0x02 , RFAL_WUM_PERIOD_75MS = 0x03 ,
RFAL_WUM_PERIOD_100MS = 0x04 , RFAL_WUM_PERIOD_125MS = 0x05 , RFAL_WUM_PERIOD_150MS = 0x06 , RFAL_WUM_PERIOD_200MS = 0x07 ,
RFAL_WUM_PERIOD_250MS = 0x08 , RFAL_WUM_PERIOD_300MS = 0x09 , RFAL_WUM_PERIOD_350MS = 0x0A , RFAL_WUM_PERIOD_400MS = 0x0B ,
RFAL_WUM_PERIOD_500MS = 0x0C , RFAL_WUM_PERIOD_800MS = 0x0D , RFAL_WUM_PERIOD_1000MS = 0x0E , RFAL_WUM_PERIOD_INF = 0x0F ,
RFAL_WUM_PERIOD_300MS = 0x12
}
```

```
enum rfalWumAAWeight {
RFAL_WUM_AA_WEIGHT_4 = 0x00 , RFAL_WUM_AA_WEIGHT_8 = 0x01 , RFAL_WUM_AA_WEIGHT_16 = 0x02 , RFAL_WUM_AA_WEIGHT_32 = 0x03 ,
RFAL_WUM_AA_WEIGHT_4 = 0x00 , RFAL_WUM_AA_WEIGHT_8 = 0x01 , RFAL_WUM_AA_WEIGHT_16 = 0x02 , RFAL_WUM_AA_WEIGHT_32 = 0x03 ,
RFAL_WUM_AA_WEIGHT_4 = 0x00 , RFAL_WUM_AA_WEIGHT_8 = 0x01 , RFAL_WUM_AA_WEIGHT_16 = 0x02 , RFAL_WUM_AA_WEIGHT_32 = 0x03 ,
RFAL_WUM_AA_WEIGHT_4 = 0x00 , RFAL_WUM_AA_WEIGHT_8 = 0x01 , RFAL_WUM_AA_WEIGHT_16 = 0x02 , RFAL_WUM_AA_WEIGHT_32 = 0x03 ,
RFAL_WUM_AA_WEIGHT_50 = 0x04 , RFAL_WUM_AA_WEIGHT_64 = 0x05 , RFAL_WUM_AA_WEIGHT_100 = 0x06 , RFAL_WUM_AA_WEIGHT_128 = 0x07
}
```

Detailed Description

Author

Gustavo Patricio

Macro Definition Documentation

◆ RFAL_SUPPORT_AP2P

```
#define RFAL_SUPPORT_AP2P ( RFAL_SUPPORT_MODE_POLL_ACTIVE_P2P || RFAL_SUPPORT_MODE_LISTEN_ACTIVE_P2P )
```

RFAL support for Active P2P (AP2P)

◆ RFAL_SUPPORT_BR_AP2P_106

```
#define RFAL_SUPPORT_BR_AP2P_106 true
```

RFAL AP2P 106 Bit Rate support switch

◆ RFAL_SUPPORT_BR_AP2P_1695

```
#define RFAL_SUPPORT_BR_AP2P_1695 false
```

RFAL AP2P 1695 Bit Rate support switch

◆ RFAL_SUPPORT_BR_AP2P_212

```
#define RFAL_SUPPORT_BR_AP2P_212 true
```

RFAL AP2P 212 Bit Rate support switch

◆ RFAL_SUPPORT_BR_AP2P_3390

```
#define RFAL_SUPPORT_BR_AP2P_3390 false
```

RFAL AP2P 3390 Bit Rate support switch

◆ RFAL_SUPPORT_BR_AP2P_424

```
#define RFAL_SUPPORT_BR_AP2P_424 true
```

RFAL AP2P 424 Bit Rate support switch

◆ RFAL_SUPPORT_BR_AP2P_6780

```
#define RFAL_SUPPORT_BR_AP2P_6780   false
```

RFAL AP2P 6780 Bit Rate support switch

◆ RFAL_SUPPORT_BR_AP2P_848

```
#define RFAL_SUPPORT_BR_AP2P_848   false
```

RFAL AP2P 848 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_A_106

```
#define RFAL_SUPPORT_BR_CE_A_106   false
```

RFAL CE A 106 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_A_212

```
#define RFAL_SUPPORT_BR_CE_A_212   false
```

RFAL CE A 212 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_A_424

```
#define RFAL_SUPPORT_BR_CE_A_424   false
```

RFAL CE A 424 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_A_848

```
#define RFAL_SUPPORT_BR_CE_A_848   false
```

RFAL CE A 848 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_B_106

```
#define RFAL_SUPPORT_BR_CE_B_106   false
```

RFAL CE B 106 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_B_212

```
#define RFAL_SUPPORT_BR_CE_B_212   false
```

RFAL CE B 212 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_B_424

```
#define RFAL_SUPPORT_BR_CE_B_424   false
```

RFAL CE B 424 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_B_848

```
#define RFAL_SUPPORT_BR_CE_B_848   false
```

RFAL CE B 848 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_F_212

```
#define RFAL_SUPPORT_BR_CE_F_212   false
```

RFAL CE F 212 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_F_424

```
#define RFAL_SUPPORT_BR_CE_F_424   false
```

RFAL CE F 424 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_106

```
#define RFAL_SUPPORT_BR_RW_106   true
```

RFAL RW 106 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_13560

```
#define RFAL_SUPPORT_BR_RW_13560   false
```

RFAL RW 6780 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_1695

```
#define RFAL_SUPPORT_BR_RW_1695   true
```

RFAL RW 1695 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_212

```
#define RFAL_SUPPORT_BR_RW_212   true
```

RFAL RW 212 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_3390

```
#define RFAL_SUPPORT_BR_RW_3390   true
```

RFAL RW 3390 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_424

```
#define RFAL_SUPPORT_BR_RW_424   true
```

RFAL RW 424 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_6780

#define RFAL_SUPPORT_BR_RW_6780 false

RFAL RW 6780 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_848

#define RFAL_SUPPORT_BR_RW_848 true

RFAL RW 848 Bit Rate support switch

◆ RFAL_SUPPORT_CE

#define RFAL_SUPPORT_CE (**RFAL_SUPPORT_MODE_LISTEN_NFCA** || **RFAL_SUPPORT_MODE_LISTEN_NFCB** || **RFAL_SUPPORT_MODE_LISTEN_NFCF**)

RFAL supported Card Emulation (CE)

◆ RFAL_SUPPORT_MODE_LISTEN_ACTIVE_P2P

#define RFAL_SUPPORT_MODE_LISTEN_ACTIVE_P2P true

RFAL Listen AP2P mode support switch

◆ RFAL_SUPPORT_MODE_LISTEN_NFCA

#define RFAL_SUPPORT_MODE_LISTEN_NFCA false

RFAL Listen NFCA mode support switch

◆ RFAL_SUPPORT_MODE_LISTEN_NFCB

#define RFAL_SUPPORT_MODE_LISTEN_NFCB false

RFAL Listen NFCB mode support switch

◆ RFAL_SUPPORT_MODE_LISTEN_NFCF

```
#define RFAL_SUPPORT_MODE_LISTEN_NFCF   false
```

RFAL Listen NFCF mode support switch

◆ RFAL_SUPPORT_MODE_POLL_ACTIVE_P2P

```
#define RFAL_SUPPORT_MODE_POLL_ACTIVE_P2P   true
```

RFAL Poll AP2P mode support switch

◆ RFAL_SUPPORT_MODE_POLL_NFCA

```
#define RFAL_SUPPORT_MODE_POLL_NFCA   true
```

RFAL Poll NFCA mode support switch

◆ RFAL_SUPPORT_MODE_POLL_NFCB

```
#define RFAL_SUPPORT_MODE_POLL_NFCB   true
```

RFAL Poll NFCB mode support switch

◆ RFAL_SUPPORT_MODE_POLL_NFCF

```
#define RFAL_SUPPORT_MODE_POLL_NFCF   true
```

RFAL Poll NFCF mode support switch

◆ RFAL_SUPPORT_MODE_POLL_NFCV

```
#define RFAL_SUPPORT_MODE_POLL_NFCV   true
```

RFAL Poll NFCV mode support switch

◆ RFAL_SUPPORT_RW

#define RFAL_SUPPORT_RW (**RFAL_SUPPORT_MODE_POLL_NFCA** || **RFAL_SUPPORT_MODE_POLL_NFCB** || **RFAL_SUPPORT_MODE_POLL_NFCF** || **RFAL_SUPPORT_MODE_POLL_NFCV**)

RFAL supported Reader/Writer (RW)

Enumeration Type Documentation

◆ rfalWumAAWeight

enum **rfalWumAAWeight**

RFAL Wake-Up Period/Timer

Enumerator	
RFAL_WUM_AA_WEIGHT_4	Wake-Up Auto Average Weight 4
RFAL_WUM_AA_WEIGHT_8	Wake-Up Auto Average Weight 8
RFAL_WUM_AA_WEIGHT_16	Wake-Up Auto Average Weight 16
RFAL_WUM_AA_WEIGHT_32	Wake-Up Auto Average Weight 32
RFAL_WUM_AA_WEIGHT_4	Wake-Up Auto Average Weight 4
RFAL_WUM_AA_WEIGHT_8	Wake-Up Auto Average Weight 8
RFAL_WUM_AA_WEIGHT_16	Wake-Up Auto Average Weight 16
RFAL_WUM_AA_WEIGHT_32	Wake-Up Auto Average Weight 32
RFAL_WUM_AA_WEIGHT_4	Wake-Up Auto Average Weight 4
RFAL_WUM_AA_WEIGHT_8	Wake-Up Auto Average Weight 8
RFAL_WUM_AA_WEIGHT_16	Wake-Up Auto Average Weight 16
RFAL_WUM_AA_WEIGHT_32	Wake-Up Auto Average Weight 32
RFAL_WUM_AA_WEIGHT_4	Wake-Up Auto Average Weight 4
RFAL_WUM_AA_WEIGHT_8	Wake-Up Auto Average Weight 8
RFAL_WUM_AA_WEIGHT_16	Wake-Up Auto Average Weight 16

RFAL_WUM_AA_WEIGHT_32	Wake-Up Auto Average Weight 32
RFAL_WUM_AA_WEIGHT_50	Wake-Up Auto Average Weight 50
RFAL_WUM_AA_WEIGHT_64	Wake-Up Auto Average Weight 64
RFAL_WUM_AA_WEIGHT_100	Wake-Up Auto Average Weight 100
RFAL_WUM_AA_WEIGHT_128	Wake-Up Auto Average Weight 128

◆ rfalWumPeriod

enum **rfalWumPeriod**

RFAL Wake-Up Period/Timer

Enumerator	
RFAL_WUM_PERIOD_10MS	Wake-Up timer ~9.7ms
RFAL_WUM_PERIOD_15MS	Wake-Up timer ~13.3ms
RFAL_WUM_PERIOD_20MS	Wake-Up timer ~19.3ms
RFAL_WUM_PERIOD_25MS	Wake-Up timer ~26.6ms
RFAL_WUM_PERIOD_40MS	Wake-Up timer ~38.7ms
RFAL_WUM_PERIOD_55MS	Wake-Up timer ~53.2ms
RFAL_WUM_PERIOD_80MS	Wake-Up timer ~77.3ms
RFAL_WUM_PERIOD_105MS	Wake-Up timer ~106.3ms
RFAL_WUM_PERIOD_155MS	Wake-Up timer ~154.7ms
RFAL_WUM_PERIOD_215MS	Wake-Up timer ~212.7ms
RFAL_WUM_PERIOD_310MS	Wake-Up timer ~309.3ms
RFAL_WUM_PERIOD_425MS	Wake-Up timer ~425.3ms
RFAL_WUM_PERIOD_620MS	Wake-Up timer ~618.6ms
RFAL_WUM_PERIOD_850MS	Wake-Up timer ~850.6ms
RFAL_WUM_PERIOD_1240MS	Wake-Up timer ~1237.3ms
RFAL_WUM_PERIOD_1700MS	Wake-Up timer ~1701.2ms
RFAL_WUM_PERIOD_10MS	Wake-Up timer 10ms

RFAL_WUM_PERIOD_20MS	Wake-Up timer 20ms
RFAL_WUM_PERIOD_30MS	Wake-Up timer 30ms
RFAL_WUM_PERIOD_40MS	Wake-Up timer 40ms
RFAL_WUM_PERIOD_50MS	Wake-Up timer 50ms
RFAL_WUM_PERIOD_60MS	Wake-Up timer 60ms
RFAL_WUM_PERIOD_70MS	Wake-Up timer 70ms
RFAL_WUM_PERIOD_80MS	Wake-Up timer 80ms
RFAL_WUM_PERIOD_100MS	Wake-Up timer 100ms
RFAL_WUM_PERIOD_200MS	Wake-Up timer 200ms
RFAL_WUM_PERIOD_300MS	Wake-Up timer 300ms
RFAL_WUM_PERIOD_400MS	Wake-Up timer 400ms
RFAL_WUM_PERIOD_500MS	Wake-Up timer 500ms
RFAL_WUM_PERIOD_600MS	Wake-Up timer 600ms
RFAL_WUM_PERIOD_700MS	Wake-Up timer 700ms
RFAL_WUM_PERIOD_800MS	Wake-Up timer 800ms
RFAL_WUM_PERIOD_10MS	Wake-Up timer 10ms
RFAL_WUM_PERIOD_20MS	Wake-Up timer 20ms
RFAL_WUM_PERIOD_30MS	Wake-Up timer 30ms
RFAL_WUM_PERIOD_40MS	Wake-Up timer 40ms
RFAL_WUM_PERIOD_50MS	Wake-Up timer 50ms
RFAL_WUM_PERIOD_60MS	Wake-Up timer 60ms
RFAL_WUM_PERIOD_70MS	Wake-Up timer 70ms
RFAL_WUM_PERIOD_80MS	Wake-Up timer 80ms
RFAL_WUM_PERIOD_100MS	Wake-Up timer 100ms
RFAL_WUM_PERIOD_200MS	Wake-Up timer 200ms
RFAL_WUM_PERIOD_300MS	Wake-Up timer 300ms
RFAL_WUM_PERIOD_400MS	Wake-Up timer 400ms
RFAL_WUM_PERIOD_500MS	Wake-Up timer 500ms

RFAL_WUM_PERIOD_600MS	Wake-Up timer 600ms
RFAL_WUM_PERIOD_700MS	Wake-Up timer 700ms
RFAL_WUM_PERIOD_800MS	Wake-Up timer 800ms
RFAL_WUM_PERIOD_12MS	Wake-Up timer ~12.5ms
RFAL_WUM_PERIOD_25MS	Wake-Up timer ~25ms
RFAL_WUM_PERIOD_50MS	Wake-Up timer ~50ms
RFAL_WUM_PERIOD_75MS	Wake-Up timer ~75ms
RFAL_WUM_PERIOD_100MS	Wake-Up timer ~100ms
RFAL_WUM_PERIOD_125MS	Wake-Up timer ~125ms
RFAL_WUM_PERIOD_150MS	Wake-Up timer ~150ms
RFAL_WUM_PERIOD_200MS	Wake-Up timer ~200ms
RFAL_WUM_PERIOD_250MS	Wake-Up timer ~250ms
RFAL_WUM_PERIOD_300MS	Wake-Up timer ~300ms
RFAL_WUM_PERIOD_350MS	Wake-Up timer ~350ms
RFAL_WUM_PERIOD_400MS	Wake-Up timer ~400ms
RFAL_WUM_PERIOD_500MS	Wake-Up timer ~600ms
RFAL_WUM_PERIOD_800MS	Wake-Up timer ~800ms
RFAL_WUM_PERIOD_1000MS	Wake-Up timer ~1000ms
RFAL_WUM_PERIOD_INF	Wake-Up timer Infinite
RFAL_WUM_PERIOD_300MS	Wake-Up timer 300ms

st25r3916/rfal_features.h File Reference

RFAL Features/Capabilities Definition for ST25R3916. More...

```
#include "rfal_platform.h"
```

Include dependency graph for st25r3916/rfal_features.h:

Data Structures

struct **rfalWakeUpConfig**

struct **rfalWakeUpInfo**

Macros

#define **RFAL_SUPPORT_MODE_POLL_NFCA** true

#define **RFAL_SUPPORT_MODE_POLL_NFCB** true

#define **RFAL_SUPPORT_MODE_POLL_NFCF** true

```
#define RFAL_SUPPORT_MODE_POLL_NFCV true
#define RFAL_SUPPORT_MODE_POLL_ACTIVE_P2P true
#define RFAL_SUPPORT_MODE_LISTEN_NFCA true
#define RFAL_SUPPORT_MODE_LISTEN_NFCB false
#define RFAL_SUPPORT_MODE_LISTEN_NFCF true
#define RFAL_SUPPORT_MODE_LISTEN_ACTIVE_P2P true
#define RFAL_SUPPORT_CE ( RFAL_SUPPORT_MODE_LISTEN_NFCA || RFAL_SUPPORT_MODE_LISTEN_NFCB ||
        RFAL_SUPPORT_MODE_LISTEN_NFCF )
#define RFAL_SUPPORT_RW ( RFAL_SUPPORT_MODE_POLL_NFCA || RFAL_SUPPORT_MODE_POLL_NFCB ||
        RFAL_SUPPORT_MODE_POLL_NFCF || RFAL_SUPPORT_MODE_POLL_NFCV )
#define RFAL_SUPPORT_AP2P ( RFAL_SUPPORT_MODE_POLL_ACTIVE_P2P ||
        RFAL_SUPPORT_MODE_LISTEN_ACTIVE_P2P )
#define RFAL_SUPPORT_BR_RW_106 true
#define RFAL_SUPPORT_BR_RW_212 true
#define RFAL_SUPPORT_BR_RW_424 true
#define RFAL_SUPPORT_BR_RW_848 true
#define RFAL_SUPPORT_BR_RW_1695 false
#define RFAL_SUPPORT_BR_RW_3390 false
#define RFAL_SUPPORT_BR_RW_6780 false
#define RFAL_SUPPORT_BR_RW_13560 false
#define RFAL_SUPPORT_BR_AP2P_106 true
#define RFAL_SUPPORT_BR_AP2P_212 true
```

```
#define RFAL_SUPPORT_BR_AP2P_424 true
#define RFAL_SUPPORT_BR_AP2P_848 false
#define RFAL_SUPPORT_BR_CE_A_106 true
#define RFAL_SUPPORT_BR_CE_A_212 false
#define RFAL_SUPPORT_BR_CE_A_424 false
#define RFAL_SUPPORT_BR_CE_A_848 false
#define RFAL_SUPPORT_BR_CE_B_106 false
#define RFAL_SUPPORT_BR_CE_B_212 false
#define RFAL_SUPPORT_BR_CE_B_424 false
#define RFAL_SUPPORT_BR_CE_B_848 false
#define RFAL_SUPPORT_BR_CE_F_212 true
#define RFAL_SUPPORT_BR_CE_F_424 true
```

Enumerations

```
enum rfalWumPeriod {
  RFAL_WUM_PERIOD_10MS = 0x00 , RFAL_WUM_PERIOD_15MS = 0x01 , RFAL_WUM_PERIOD_20MS = 0x02
, RFAL_WUM_PERIOD_25MS = 0x03 ,
  RFAL_WUM_PERIOD_40MS = 0x04 , RFAL_WUM_PERIOD_55MS = 0x05 , RFAL_WUM_PERIOD_80MS = 0x06
, RFAL_WUM_PERIOD_105MS = 0x07 ,
  RFAL_WUM_PERIOD_155MS = 0x08 , RFAL_WUM_PERIOD_215MS = 0x09 , RFAL_WUM_PERIOD_310MS =
0x0A , RFAL_WUM_PERIOD_425MS = 0x0B ,
  RFAL_WUM_PERIOD_620MS = 0x0C , RFAL_WUM_PERIOD_850MS = 0x0D , RFAL_WUM_PERIOD_1240MS =
0x0E , RFAL_WUM_PERIOD_1700MS = 0x0F ,
  RFAL_WUM_PERIOD_10MS = 0x00 , RFAL_WUM_PERIOD_20MS = 0x01 , RFAL_WUM_PERIOD_30MS = 0x02
, RFAL_WUM_PERIOD_40MS = 0x03 ,
  RFAL_WUM_PERIOD_50MS = 0x04 , RFAL_WUM_PERIOD_60MS = 0x05 , RFAL_WUM_PERIOD_70MS = 0x06
, RFAL_WUM_PERIOD_80MS = 0x07 ,
  RFAL_WUM_PERIOD_100MS = 0x10 , RFAL_WUM_PERIOD_200MS = 0x11 , RFAL_WUM_PERIOD_300MS =
```

```
0x12 , RFAL_WUM_PERIOD_400MS = 0x13 ,
 RFAL_WUM_PERIOD_500MS = 0x14 , RFAL_WUM_PERIOD_600MS = 0x15 , RFAL_WUM_PERIOD_700MS =
0x16 , RFAL_WUM_PERIOD_800MS = 0x17 ,
 RFAL_WUM_PERIOD_10MS = 0x00 , RFAL_WUM_PERIOD_20MS = 0x01 , RFAL_WUM_PERIOD_30MS = 0x02
, RFAL_WUM_PERIOD_40MS = 0x03 ,
 RFAL_WUM_PERIOD_50MS = 0x04 , RFAL_WUM_PERIOD_60MS = 0x05 , RFAL_WUM_PERIOD_70MS = 0x06
, RFAL_WUM_PERIOD_80MS = 0x07 ,
 RFAL_WUM_PERIOD_100MS = 0x10 , RFAL_WUM_PERIOD_200MS = 0x11 , RFAL_WUM_PERIOD_300MS =
0x12 , RFAL_WUM_PERIOD_400MS = 0x13 ,
 RFAL_WUM_PERIOD_500MS = 0x14 , RFAL_WUM_PERIOD_600MS = 0x15 , RFAL_WUM_PERIOD_700MS =
0x16 , RFAL_WUM_PERIOD_800MS = 0x17 ,
 RFAL_WUM_PERIOD_12MS = 0x00 , RFAL_WUM_PERIOD_25MS = 0x01 , RFAL_WUM_PERIOD_50MS = 0x02
, RFAL_WUM_PERIOD_75MS = 0x03 ,
 RFAL_WUM_PERIOD_100MS = 0x04 , RFAL_WUM_PERIOD_125MS = 0x05 , RFAL_WUM_PERIOD_150MS =
0x06 , RFAL_WUM_PERIOD_200MS = 0x07 ,
 RFAL_WUM_PERIOD_250MS = 0x08 , RFAL_WUM_PERIOD_300MS = 0x09 , RFAL_WUM_PERIOD_350MS =
0x0A , RFAL_WUM_PERIOD_400MS = 0x0B ,
 RFAL_WUM_PERIOD_500MS = 0x0C , RFAL_WUM_PERIOD_800MS = 0x0D , RFAL_WUM_PERIOD_1000MS =
0x0E , RFAL_WUM_PERIOD_INF = 0x0F ,
 RFAL_WUM_PERIOD_300MS = 0x12
}
```

```
enum rfalWumAAWeight {
 RFAL_WUM_AA_WEIGHT_4 = 0x00 , RFAL_WUM_AA_WEIGHT_8 = 0x01 , RFAL_WUM_AA_WEIGHT_16 =
0x02 , RFAL_WUM_AA_WEIGHT_32 = 0x03 ,
 RFAL_WUM_AA_WEIGHT_4 = 0x00 , RFAL_WUM_AA_WEIGHT_8 = 0x01 , RFAL_WUM_AA_WEIGHT_16 =
0x02 , RFAL_WUM_AA_WEIGHT_32 = 0x03 ,
 RFAL_WUM_AA_WEIGHT_4 = 0x00 , RFAL_WUM_AA_WEIGHT_8 = 0x01 , RFAL_WUM_AA_WEIGHT_16 =
0x02 , RFAL_WUM_AA_WEIGHT_32 = 0x03 ,
 RFAL_WUM_AA_WEIGHT_4 = 0x00 , RFAL_WUM_AA_WEIGHT_8 = 0x01 , RFAL_WUM_AA_WEIGHT_16 =
0x02 , RFAL_WUM_AA_WEIGHT_32 = 0x03 ,
 RFAL_WUM_AA_WEIGHT_50 = 0x04 , RFAL_WUM_AA_WEIGHT_64 = 0x05 , RFAL_WUM_AA_WEIGHT_100 =
0x06 , RFAL_WUM_AA_WEIGHT_128 = 0x07
}
```

Detailed Description

Author

Gustavo Patricio

Macro Definition Documentation

◆ RFAL_SUPPORT_AP2P

```
#define RFAL_SUPPORT_AP2P ( RFAL_SUPPORT_MODE_POLL_ACTIVE_P2P || RFAL_SUPPORT_MODE_LISTEN_ACTIVE_P2P )
```

RFAL support for Active P2P (AP2P)

◆ RFAL_SUPPORT_BR_AP2P_106

```
#define RFAL_SUPPORT_BR_AP2P_106 true
```

RFAL AP2P 106 Bit Rate support switch

◆ RFAL_SUPPORT_BR_AP2P_212

```
#define RFAL_SUPPORT_BR_AP2P_212 true
```

RFAL AP2P 212 Bit Rate support switch

◆ RFAL_SUPPORT_BR_AP2P_424

```
#define RFAL_SUPPORT_BR_AP2P_424 true
```

RFAL AP2P 424 Bit Rate support switch

◆ RFAL_SUPPORT_BR_AP2P_848

```
#define RFAL_SUPPORT_BR_AP2P_848 false
```

RFAL AP2P 848 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_A_106

```
#define RFAL_SUPPORT_BR_CE_A_106   true
```

RFAL CE A 106 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_A_212

```
#define RFAL_SUPPORT_BR_CE_A_212   false
```

RFAL CE A 212 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_A_424

```
#define RFAL_SUPPORT_BR_CE_A_424   false
```

RFAL CE A 424 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_A_848

```
#define RFAL_SUPPORT_BR_CE_A_848   false
```

RFAL CE A 848 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_B_106

```
#define RFAL_SUPPORT_BR_CE_B_106   false
```

RFAL CE B 106 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_B_212

```
#define RFAL_SUPPORT_BR_CE_B_212   false
```

RFAL CE B 212 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_B_424

```
#define RFAL_SUPPORT_BR_CE_B_424   false
```

RFAL CE B 424 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_B_848

```
#define RFAL_SUPPORT_BR_CE_B_848   false
```

RFAL CE B 848 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_F_212

```
#define RFAL_SUPPORT_BR_CE_F_212   true
```

RFAL CE F 212 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_F_424

```
#define RFAL_SUPPORT_BR_CE_F_424   true
```

RFAL CE F 424 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_106

```
#define RFAL_SUPPORT_BR_RW_106   true
```

RFAL RW 106 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_13560

```
#define RFAL_SUPPORT_BR_RW_13560   false
```

RFAL RW 6780 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_1695

```
#define RFAL_SUPPORT_BR_RW_1695   false
```

RFAL RW 1695 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_212

```
#define RFAL_SUPPORT_BR_RW_212   true
```

RFAL RW 212 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_3390

```
#define RFAL_SUPPORT_BR_RW_3390   false
```

RFAL RW 3390 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_424

```
#define RFAL_SUPPORT_BR_RW_424   true
```

RFAL RW 424 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_6780

```
#define RFAL_SUPPORT_BR_RW_6780   false
```

RFAL RW 6780 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_848

```
#define RFAL_SUPPORT_BR_RW_848   true
```

RFAL RW 848 Bit Rate support switch

◆ RFAL_SUPPORT_CE

#define RFAL_SUPPORT_CE (**RFAL_SUPPORT_MODE_LISTEN_NFCA** || **RFAL_SUPPORT_MODE_LISTEN_NFCB** || **RFAL_SUPPORT_MODE_LISTEN_NFCF**)

RFAL supported Card Emulation (CE)

◆ RFAL_SUPPORT_MODE_LISTEN_ACTIVE_P2P

#define RFAL_SUPPORT_MODE_LISTEN_ACTIVE_P2P true

RFAL Listen AP2P mode support switch

◆ RFAL_SUPPORT_MODE_LISTEN_NFCA

#define RFAL_SUPPORT_MODE_LISTEN_NFCA true

RFAL Listen NFCA mode support switch

◆ RFAL_SUPPORT_MODE_LISTEN_NFCB

#define RFAL_SUPPORT_MODE_LISTEN_NFCB false

RFAL Listen NFCB mode support switch

◆ RFAL_SUPPORT_MODE_LISTEN_NFCF

#define RFAL_SUPPORT_MODE_LISTEN_NFCF true

RFAL Listen NFCF mode support switch

◆ RFAL_SUPPORT_MODE_POLL_ACTIVE_P2P

#define RFAL_SUPPORT_MODE_POLL_ACTIVE_P2P true

RFAL Poll AP2P mode support switch

◆ RFAL_SUPPORT_MODE_POLL_NFCA

#define RFAL_SUPPORT_MODE_POLL_NFCA true

RFAL Poll NFCA mode support switch

◆ RFAL_SUPPORT_MODE_POLL_NFCB

#define RFAL_SUPPORT_MODE_POLL_NFCB true

RFAL Poll NFCB mode support switch

◆ RFAL_SUPPORT_MODE_POLL_NFCF

#define RFAL_SUPPORT_MODE_POLL_NFCF true

RFAL Poll NFCF mode support switch

◆ RFAL_SUPPORT_MODE_POLL_NFCV

#define RFAL_SUPPORT_MODE_POLL_NFCV true

RFAL Poll NFCV mode support switch

◆ RFAL_SUPPORT_RW

#define RFAL_SUPPORT_RW (**RFAL_SUPPORT_MODE_POLL_NFCA** || **RFAL_SUPPORT_MODE_POLL_NFCB** || **RFAL_SUPPORT_MODE_POLL_NFCF** || **RFAL_SUPPORT_MODE_POLL_NFCV**)

RFAL supported Reader/Writer (RW)

Enumeration Type Documentation

◆ rfalWumAAWeight

enum **rfalWumAAWeight**

RFAL Wake-Up Period/Timer

Enumerator	
RFAL_WUM_AA_WEIGHT_4	Wake-Up Auto Average Weight 4
RFAL_WUM_AA_WEIGHT_8	Wake-Up Auto Average Weight 8
RFAL_WUM_AA_WEIGHT_16	Wake-Up Auto Average Weight 16
RFAL_WUM_AA_WEIGHT_32	Wake-Up Auto Average Weight 32
RFAL_WUM_AA_WEIGHT_4	Wake-Up Auto Average Weight 4
RFAL_WUM_AA_WEIGHT_8	Wake-Up Auto Average Weight 8
RFAL_WUM_AA_WEIGHT_16	Wake-Up Auto Average Weight 16
RFAL_WUM_AA_WEIGHT_32	Wake-Up Auto Average Weight 32
RFAL_WUM_AA_WEIGHT_4	Wake-Up Auto Average Weight 4
RFAL_WUM_AA_WEIGHT_8	Wake-Up Auto Average Weight 8
RFAL_WUM_AA_WEIGHT_16	Wake-Up Auto Average Weight 16
RFAL_WUM_AA_WEIGHT_32	Wake-Up Auto Average Weight 32
RFAL_WUM_AA_WEIGHT_4	Wake-Up Auto Average Weight 4
RFAL_WUM_AA_WEIGHT_8	Wake-Up Auto Average Weight 8
RFAL_WUM_AA_WEIGHT_16	Wake-Up Auto Average Weight 16
RFAL_WUM_AA_WEIGHT_32	Wake-Up Auto Average Weight 32
RFAL_WUM_AA_WEIGHT_50	Wake-Up Auto Average Weight 50
RFAL_WUM_AA_WEIGHT_64	Wake-Up Auto Average Weight 64
RFAL_WUM_AA_WEIGHT_100	Wake-Up Auto Average Weight 100
RFAL_WUM_AA_WEIGHT_128	Wake-Up Auto Average Weight 128

◆ rfalWumPeriod

enum **rfalWumPeriod**

RFAL Wake-Up Period/Timer

Enumerator

RFAL_WUM_PERIOD_10MS	Wake-Up timer ~9.7ms
RFAL_WUM_PERIOD_15MS	Wake-Up timer ~13.3ms
RFAL_WUM_PERIOD_20MS	Wake-Up timer ~19.3ms
RFAL_WUM_PERIOD_25MS	Wake-Up timer ~26.6ms
RFAL_WUM_PERIOD_40MS	Wake-Up timer ~38.7ms
RFAL_WUM_PERIOD_55MS	Wake-Up timer ~53.2ms
RFAL_WUM_PERIOD_80MS	Wake-Up timer ~77.3ms
RFAL_WUM_PERIOD_105MS	Wake-Up timer ~106.3ms
RFAL_WUM_PERIOD_155MS	Wake-Up timer ~154.7ms
RFAL_WUM_PERIOD_215MS	Wake-Up timer ~212.7ms
RFAL_WUM_PERIOD_310MS	Wake-Up timer ~309.3ms
RFAL_WUM_PERIOD_425MS	Wake-Up timer ~425.3ms
RFAL_WUM_PERIOD_620MS	Wake-Up timer ~618.6ms
RFAL_WUM_PERIOD_850MS	Wake-Up timer ~850.6ms
RFAL_WUM_PERIOD_1240MS	Wake-Up timer ~1237.3ms
RFAL_WUM_PERIOD_1700MS	Wake-Up timer ~1701.2ms
RFAL_WUM_PERIOD_10MS	Wake-Up timer 10ms
RFAL_WUM_PERIOD_20MS	Wake-Up timer 20ms
RFAL_WUM_PERIOD_30MS	Wake-Up timer 30ms
RFAL_WUM_PERIOD_40MS	Wake-Up timer 40ms
RFAL_WUM_PERIOD_50MS	Wake-Up timer 50ms
RFAL_WUM_PERIOD_60MS	Wake-Up timer 60ms
RFAL_WUM_PERIOD_70MS	Wake-Up timer 70ms
RFAL_WUM_PERIOD_80MS	Wake-Up timer 80ms
RFAL_WUM_PERIOD_100MS	Wake-Up timer 100ms
RFAL_WUM_PERIOD_200MS	Wake-Up timer 200ms
RFAL_WUM_PERIOD_300MS	Wake-Up timer 300ms
RFAL_WUM_PERIOD_400MS	Wake-Up timer 400ms

RFAL_WUM_PERIOD_500MS	Wake-Up timer 500ms
RFAL_WUM_PERIOD_600MS	Wake-Up timer 600ms
RFAL_WUM_PERIOD_700MS	Wake-Up timer 700ms
RFAL_WUM_PERIOD_800MS	Wake-Up timer 800ms
RFAL_WUM_PERIOD_10MS	Wake-Up timer 10ms
RFAL_WUM_PERIOD_20MS	Wake-Up timer 20ms
RFAL_WUM_PERIOD_30MS	Wake-Up timer 30ms
RFAL_WUM_PERIOD_40MS	Wake-Up timer 40ms
RFAL_WUM_PERIOD_50MS	Wake-Up timer 50ms
RFAL_WUM_PERIOD_60MS	Wake-Up timer 60ms
RFAL_WUM_PERIOD_70MS	Wake-Up timer 70ms
RFAL_WUM_PERIOD_80MS	Wake-Up timer 80ms
RFAL_WUM_PERIOD_100MS	Wake-Up timer 100ms
RFAL_WUM_PERIOD_200MS	Wake-Up timer 200ms
RFAL_WUM_PERIOD_300MS	Wake-Up timer 300ms
RFAL_WUM_PERIOD_400MS	Wake-Up timer 400ms
RFAL_WUM_PERIOD_500MS	Wake-Up timer 500ms
RFAL_WUM_PERIOD_600MS	Wake-Up timer 600ms
RFAL_WUM_PERIOD_700MS	Wake-Up timer 700ms
RFAL_WUM_PERIOD_800MS	Wake-Up timer 800ms
RFAL_WUM_PERIOD_12MS	Wake-Up timer ~12.5ms
RFAL_WUM_PERIOD_25MS	Wake-Up timer ~25ms
RFAL_WUM_PERIOD_50MS	Wake-Up timer ~50ms
RFAL_WUM_PERIOD_75MS	Wake-Up timer ~75ms
RFAL_WUM_PERIOD_100MS	Wake-Up timer ~100ms
RFAL_WUM_PERIOD_125MS	Wake-Up timer ~125ms
RFAL_WUM_PERIOD_150MS	Wake-Up timer ~150ms
RFAL_WUM_PERIOD_200MS	Wake-Up timer ~200ms

RFAL_WUM_PERIOD_250MS	Wake-Up timer ~250ms
RFAL_WUM_PERIOD_300MS	Wake-Up timer ~300ms
RFAL_WUM_PERIOD_350MS	Wake-Up timer ~350ms
RFAL_WUM_PERIOD_400MS	Wake-Up timer ~400ms
RFAL_WUM_PERIOD_500MS	Wake-Up timer ~600ms
RFAL_WUM_PERIOD_800MS	Wake-Up timer ~800ms
RFAL_WUM_PERIOD_1000MS	Wake-Up timer ~1000ms
RFAL_WUM_PERIOD_INF	Wake-Up timer Infinite
RFAL_WUM_PERIOD_300MS	Wake-Up timer 300ms

st25r500/rfal_features.h File Reference

RFAL Features/Capabilities Definition for ST25R500. More...

```
#include "rfal_platform.h"
```

Include dependency graph for st25r500/rfal_features.h:

Data Structures

struct **rfalWumMeasChannel**

struct **rfalWakeUpConfig**

struct **rfalWakeUpInfo**

Macros

#define **RFAL_SUPPORT_MODE_POLL_NFCA** true

#define **RFAL_SUPPORT_MODE_POLL_NFCB** true

#define **RFAL_SUPPORT_MODE_POLL_NFCF** true

#define **RFAL_SUPPORT_MODE_POLL_NFCV** true

#define **RFAL_SUPPORT_MODE_POLL_ACTIVE_P2P** false

#define **RFAL_SUPPORT_MODE_LISTEN_NFCA** true

#define **RFAL_SUPPORT_MODE_LISTEN_NFCB** false

#define **RFAL_SUPPORT_MODE_LISTEN_NFCF** true

#define **RFAL_SUPPORT_MODE_LISTEN_ACTIVE_P2P** false

#define **RFAL_SUPPORT_CE** (**RFAL_SUPPORT_MODE_LISTEN_NFCA** || **RFAL_SUPPORT_MODE_LISTEN_NFCB** || **RFAL_SUPPORT_MODE_LISTEN_NFCF**)

```
#define RFAL_SUPPORT_RW ( RFAL_SUPPORT_MODE_POLL_NFCA || RFAL_SUPPORT_MODE_POLL_NFCB ||
        RFAL_SUPPORT_MODE_POLL_NFCF || RFAL_SUPPORT_MODE_POLL_NFCV )
```

```
#define RFAL_SUPPORT_AP2P ( RFAL_SUPPORT_MODE_POLL_ACTIVE_P2P ||
        RFAL_SUPPORT_MODE_LISTEN_ACTIVE_P2P )
```

```
#define RFAL_SUPPORT_BR_RW_26 true
```

```
#define RFAL_SUPPORT_BR_RW_53 true
```

```
#define RFAL_SUPPORT_BR_RW_106 true
```

```
#define RFAL_SUPPORT_BR_RW_212 true
```

```
#define RFAL_SUPPORT_BR_RW_424 true
```

```
#define RFAL_SUPPORT_BR_RW_848 true
```

```
#define RFAL_SUPPORT_BR_RW_1695 false
```

```
#define RFAL_SUPPORT_BR_RW_3390 false
```

```
#define RFAL_SUPPORT_BR_RW_6780 false
```

```
#define RFAL_SUPPORT_BR_RW_13560 false
```

```
#define RFAL_SUPPORT_BR_AP2P_106 false
```

```
#define RFAL_SUPPORT_BR_AP2P_212 false
```

```
#define RFAL_SUPPORT_BR_AP2P_424 false
```

```
#define RFAL_SUPPORT_BR_AP2P_848 false
```

```
#define RFAL_SUPPORT_BR_CE_A_106 true
```

```
#define RFAL_SUPPORT_BR_CE_A_212 false
```

```
#define RFAL_SUPPORT_BR_CE_A_424 false
```

```
#define RFAL_SUPPORT_BR_CE_A_848 false
```

```
#define RFAL_SUPPORT_BR_CE_B_106 false
```

```
#define RFAL_SUPPORT_BR_CE_B_212 false
```

```
#define RFAL_SUPPORT_BR_CE_B_424 false
```

```
#define RFAL_SUPPORT_BR_CE_B_848 false
```

```
#define RFAL_SUPPORT_BR_CE_F_212 true
```

```
#define RFAL_SUPPORT_BR_CE_F_424 true
```

Enumerations

```
enum rfalWumPeriod {
  RFAL_WUM_PERIOD_10MS = 0x00 , RFAL_WUM_PERIOD_15MS = 0x01 , RFAL_WUM_PERIOD_20MS = 0x02
, RFAL_WUM_PERIOD_25MS = 0x03 ,
  RFAL_WUM_PERIOD_40MS = 0x04 , RFAL_WUM_PERIOD_55MS = 0x05 , RFAL_WUM_PERIOD_80MS = 0x06
, RFAL_WUM_PERIOD_105MS = 0x07 ,
  RFAL_WUM_PERIOD_155MS = 0x08 , RFAL_WUM_PERIOD_215MS = 0x09 , RFAL_WUM_PERIOD_310MS =
0x0A , RFAL_WUM_PERIOD_425MS = 0x0B ,
  RFAL_WUM_PERIOD_620MS = 0x0C , RFAL_WUM_PERIOD_850MS = 0x0D , RFAL_WUM_PERIOD_1240MS =
0x0E , RFAL_WUM_PERIOD_1700MS = 0x0F ,
  RFAL_WUM_PERIOD_10MS = 0x00 , RFAL_WUM_PERIOD_20MS = 0x01 , RFAL_WUM_PERIOD_30MS = 0x02
, RFAL_WUM_PERIOD_40MS = 0x03 ,
  RFAL_WUM_PERIOD_50MS = 0x04 , RFAL_WUM_PERIOD_60MS = 0x05 , RFAL_WUM_PERIOD_70MS = 0x06
, RFAL_WUM_PERIOD_80MS = 0x07 ,
  RFAL_WUM_PERIOD_100MS = 0x10 , RFAL_WUM_PERIOD_200MS = 0x11 , RFAL_WUM_PERIOD_300MS =
0x12 , RFAL_WUM_PERIOD_400MS = 0x13 ,
  RFAL_WUM_PERIOD_500MS = 0x14 , RFAL_WUM_PERIOD_600MS = 0x15 , RFAL_WUM_PERIOD_700MS =
0x16 , RFAL_WUM_PERIOD_800MS = 0x17 ,
  RFAL_WUM_PERIOD_10MS = 0x00 , RFAL_WUM_PERIOD_20MS = 0x01 , RFAL_WUM_PERIOD_30MS = 0x02
, RFAL_WUM_PERIOD_40MS = 0x03 ,
  RFAL_WUM_PERIOD_50MS = 0x04 , RFAL_WUM_PERIOD_60MS = 0x05 , RFAL_WUM_PERIOD_70MS = 0x06
, RFAL_WUM_PERIOD_80MS = 0x07 ,
  RFAL_WUM_PERIOD_100MS = 0x10 , RFAL_WUM_PERIOD_200MS = 0x11 , RFAL_WUM_PERIOD_300MS =
0x12 , RFAL_WUM_PERIOD_400MS = 0x13 ,
  RFAL_WUM_PERIOD_500MS = 0x14 , RFAL_WUM_PERIOD_600MS = 0x15 , RFAL_WUM_PERIOD_700MS =
0x16 , RFAL_WUM_PERIOD_800MS = 0x17 ,
```

RFAL_WUM_PERIOD_12MS = 0x00 , RFAL_WUM_PERIOD_25MS = 0x01 , RFAL_WUM_PERIOD_50MS = 0x02 , RFAL_WUM_PERIOD_75MS = 0x03 ,
RFAL_WUM_PERIOD_100MS = 0x04 , RFAL_WUM_PERIOD_125MS = 0x05 , RFAL_WUM_PERIOD_150MS = 0x06 , RFAL_WUM_PERIOD_200MS = 0x07 ,
RFAL_WUM_PERIOD_250MS = 0x08 , RFAL_WUM_PERIOD_300MS = 0x09 , RFAL_WUM_PERIOD_350MS = 0x0A , RFAL_WUM_PERIOD_400MS = 0x0B ,
RFAL_WUM_PERIOD_500MS = 0x0C , RFAL_WUM_PERIOD_800MS = 0x0D , RFAL_WUM_PERIOD_1000MS = 0x0E , RFAL_WUM_PERIOD_INF = 0x0F ,
RFAL_WUM_PERIOD_300MS = 0x12
}

enum **rfalWumAAWeight** {
RFAL_WUM_AA_WEIGHT_4 = 0x00 , RFAL_WUM_AA_WEIGHT_8 = 0x01 , RFAL_WUM_AA_WEIGHT_16 = 0x02 , RFAL_WUM_AA_WEIGHT_32 = 0x03 ,
RFAL_WUM_AA_WEIGHT_4 = 0x00 , RFAL_WUM_AA_WEIGHT_8 = 0x01 , RFAL_WUM_AA_WEIGHT_16 = 0x02 , RFAL_WUM_AA_WEIGHT_32 = 0x03 ,
RFAL_WUM_AA_WEIGHT_4 = 0x00 , RFAL_WUM_AA_WEIGHT_8 = 0x01 , RFAL_WUM_AA_WEIGHT_16 = 0x02 , RFAL_WUM_AA_WEIGHT_32 = 0x03 ,
RFAL_WUM_AA_WEIGHT_4 = 0x00 , RFAL_WUM_AA_WEIGHT_8 = 0x01 , RFAL_WUM_AA_WEIGHT_16 = 0x02 , RFAL_WUM_AA_WEIGHT_32 = 0x03 ,
RFAL_WUM_AA_WEIGHT_50 = 0x04 , RFAL_WUM_AA_WEIGHT_64 = 0x05 , RFAL_WUM_AA_WEIGHT_100 = 0x06 , RFAL_WUM_AA_WEIGHT_128 = 0x07
}

enum **rfalWumMeasDuration** {
RFAL_WUM_MEAS_DUR_26_10 = 0 , RFAL_WUM_MEAS_DUR_30_14 = 1 , RFAL_WUM_MEAS_DUR_34_19 = 2 , RFAL_WUM_MEAS_DUR_44_28 = 3 ,
RFAL_WUM_MEAS_DUR_26_10 = 0 , RFAL_WUM_MEAS_DUR_30_14 = 1 , RFAL_WUM_MEAS_DUR_34_19 = 2 , RFAL_WUM_MEAS_DUR_44_28 = 3
}

enum **rfalWumMeasFilter** { RFAL_WUM_MEAS_FIL_SLOW = false , RFAL_WUM_MEAS_FIL_FAST = true , RFAL_WUM_MEAS_FIL_SLOW = false , RFAL_WUM_MEAS_FIL_FAST = true }

enum { RFAL_WUM_TRE_ABOVE = (1U<<2) , RFAL_WUM_TRE_BETWEEN = (1U<<1) , RFAL_WUM_TRE_BELOW = (1U<<0) }

Detailed Description

Author

Gustavo Patricio

Macro Definition Documentation

◆ RFAL_SUPPORT_AP2P

#define RFAL_SUPPORT_AP2P (**RFAL_SUPPORT_MODE_POLL_ACTIVE_P2P** || **RFAL_SUPPORT_MODE_LISTEN_ACTIVE_P2P**)

RFAL support for Active P2P (AP2P)

◆ RFAL_SUPPORT_BR_AP2P_106

#define RFAL_SUPPORT_BR_AP2P_106 false

RFAL AP2P 106 Bit Rate support switch

◆ RFAL_SUPPORT_BR_AP2P_212

#define RFAL_SUPPORT_BR_AP2P_212 false

RFAL AP2P 212 Bit Rate support switch

◆ RFAL_SUPPORT_BR_AP2P_424

#define RFAL_SUPPORT_BR_AP2P_424 false

RFAL AP2P 424 Bit Rate support switch

◆ RFAL_SUPPORT_BR_AP2P_848

#define RFAL_SUPPORT_BR_AP2P_848 false

RFAL AP2P 848 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_A_106

```
#define RFAL_SUPPORT_BR_CE_A_106   true
```

RFAL CE A 106 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_A_212

```
#define RFAL_SUPPORT_BR_CE_A_212   false
```

RFAL CE A 212 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_A_424

```
#define RFAL_SUPPORT_BR_CE_A_424   false
```

RFAL CE A 424 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_A_848

```
#define RFAL_SUPPORT_BR_CE_A_848   false
```

RFAL CE A 848 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_B_106

```
#define RFAL_SUPPORT_BR_CE_B_106   false
```

RFAL CE B 106 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_B_212

```
#define RFAL_SUPPORT_BR_CE_B_212   false
```

RFAL CE B 212 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_B_424

```
#define RFAL_SUPPORT_BR_CE_B_424   false
```

RFAL CE B 424 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_B_848

```
#define RFAL_SUPPORT_BR_CE_B_848   false
```

RFAL CE B 848 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_F_212

```
#define RFAL_SUPPORT_BR_CE_F_212   true
```

RFAL CE F 212 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_F_424

```
#define RFAL_SUPPORT_BR_CE_F_424   true
```

RFAL CE F 424 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_106

```
#define RFAL_SUPPORT_BR_RW_106   true
```

RFAL RW 106 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_13560

```
#define RFAL_SUPPORT_BR_RW_13560   false
```

RFAL RW 6780 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_1695

```
#define RFAL_SUPPORT_BR_RW_1695   false
```

RFAL RW 1695 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_212

```
#define RFAL_SUPPORT_BR_RW_212   true
```

RFAL RW 212 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_26

```
#define RFAL_SUPPORT_BR_RW_26   true
```

RFAL RW 26 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_3390

```
#define RFAL_SUPPORT_BR_RW_3390   false
```

RFAL RW 3390 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_424

```
#define RFAL_SUPPORT_BR_RW_424   true
```

RFAL RW 424 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_53

```
#define RFAL_SUPPORT_BR_RW_53   true
```

RFAL RW 53 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_6780

#define RFAL_SUPPORT_BR_RW_6780 false

RFAL RW 6780 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_848

#define RFAL_SUPPORT_BR_RW_848 true

RFAL RW 848 Bit Rate support switch

◆ RFAL_SUPPORT_CE

#define RFAL_SUPPORT_CE (**RFAL_SUPPORT_MODE_LISTEN_NFCA** || **RFAL_SUPPORT_MODE_LISTEN_NFCB** || **RFAL_SUPPORT_MODE_LISTEN_NFCF**)

RFAL supported Card Emulation (CE)

◆ RFAL_SUPPORT_MODE_LISTEN_ACTIVE_P2P

#define RFAL_SUPPORT_MODE_LISTEN_ACTIVE_P2P false

RFAL Listen AP2P mode support switch

◆ RFAL_SUPPORT_MODE_LISTEN_NFCA

#define RFAL_SUPPORT_MODE_LISTEN_NFCA true

RFAL Listen NFCA mode support switch

◆ RFAL_SUPPORT_MODE_LISTEN_NFCB

#define RFAL_SUPPORT_MODE_LISTEN_NFCB false

RFAL Listen NFCB mode support switch

◆ RFAL_SUPPORT_MODE_LISTEN_NFCF

```
#define RFAL_SUPPORT_MODE_LISTEN_NFCF   true
```

RFAL Listen NFCF mode support switch

◆ RFAL_SUPPORT_MODE_POLL_ACTIVE_P2P

```
#define RFAL_SUPPORT_MODE_POLL_ACTIVE_P2P   false
```

RFAL Poll AP2P mode support switch

◆ RFAL_SUPPORT_MODE_POLL_NFCA

```
#define RFAL_SUPPORT_MODE_POLL_NFCA   true
```

RFAL Poll NFCA mode support switch

◆ RFAL_SUPPORT_MODE_POLL_NFCB

```
#define RFAL_SUPPORT_MODE_POLL_NFCB   true
```

RFAL Poll NFCB mode support switch

◆ RFAL_SUPPORT_MODE_POLL_NFCF

```
#define RFAL_SUPPORT_MODE_POLL_NFCF   true
```

RFAL Poll NFCF mode support switch

◆ RFAL_SUPPORT_MODE_POLL_NFCV

```
#define RFAL_SUPPORT_MODE_POLL_NFCV   true
```

RFAL Poll NFCV mode support switch

◆ RFAL_SUPPORT_RW

#define RFAL_SUPPORT_RW (**RFAL_SUPPORT_MODE_POLL_NFCA** || **RFAL_SUPPORT_MODE_POLL_NFCB** || **RFAL_SUPPORT_MODE_POLL_NFCF** || **RFAL_SUPPORT_MODE_POLL_NFCV**)

RFAL supported Reader/Writer (RW)

Enumeration Type Documentation

◆ anonymous enum

anonymous enum

RFAL Wake-Up trigger tresholds for threshold bitmask config

Enumerator	
RFAL_WUM_TRE_ABOVE	Wake-up trigger threshold: above upper limit
RFAL_WUM_TRE_BETWEEN	Wake-up trigger threshold: between upper and lower limit
RFAL_WUM_TRE_BELOW	Wake-up trigger threshold: below lower limit

◆ rfalWumAAWeight

enum **rfalWumAAWeight**

RFAL Wake-Up weight of last measurement result for auto- averaging

Enumerator	
RFAL_WUM_AA_WEIGHT_4	Wake-Up Auto Average Weight 4
RFAL_WUM_AA_WEIGHT_8	Wake-Up Auto Average Weight 8
RFAL_WUM_AA_WEIGHT_16	Wake-Up Auto Average Weight 16
RFAL_WUM_AA_WEIGHT_32	Wake-Up Auto Average Weight 32
RFAL_WUM_AA_WEIGHT_4	Wake-Up Auto Average Weight 4
RFAL_WUM_AA_WEIGHT_8	Wake-Up Auto Average Weight 8
RFAL_WUM_AA_WEIGHT_16	Wake-Up Auto Average Weight 16

RFAL_WUM_AA_WEIGHT_32	Wake-Up Auto Average Weight 32
RFAL_WUM_AA_WEIGHT_4	Wake-Up Auto Average Weight 4
RFAL_WUM_AA_WEIGHT_8	Wake-Up Auto Average Weight 8
RFAL_WUM_AA_WEIGHT_16	Wake-Up Auto Average Weight 16
RFAL_WUM_AA_WEIGHT_32	Wake-Up Auto Average Weight 32
RFAL_WUM_AA_WEIGHT_4	Wake-Up Auto Average Weight 4
RFAL_WUM_AA_WEIGHT_8	Wake-Up Auto Average Weight 8
RFAL_WUM_AA_WEIGHT_16	Wake-Up Auto Average Weight 16
RFAL_WUM_AA_WEIGHT_32	Wake-Up Auto Average Weight 32
RFAL_WUM_AA_WEIGHT_50	Wake-Up Auto Average Weight 50
RFAL_WUM_AA_WEIGHT_64	Wake-Up Auto Average Weight 64
RFAL_WUM_AA_WEIGHT_100	Wake-Up Auto Average Weight 100
RFAL_WUM_AA_WEIGHT_128	Wake-Up Auto Average Weight 128

◆ rfalWumMeasDuration

enum **rfalWumMeasDuration**

RFAL Wake-Up mesurement duration

Enumerator	
RFAL_WUM_MEAS_DUR_26_10	WU measurement duration: 26.0us (slow) / 10.6us (fast)
RFAL_WUM_MEAS_DUR_30_14	WU measurement duration: 29.5us (slow) / 14.2us (fast)
RFAL_WUM_MEAS_DUR_34_19	WU measurement duration: 34.2us (slow) / 18.9us (fast)
RFAL_WUM_MEAS_DUR_44_28	WU measurement duration: 43.7us (slow) / 28.3us (fast)
RFAL_WUM_MEAS_DUR_26_10	WU measurement duration: 26.0us (slow) / 10.6us (fast)
RFAL_WUM_MEAS_DUR_30_14	WU measurement duration: 29.5us (slow) / 14.2us (fast)
RFAL_WUM_MEAS_DUR_34_19	WU measurement duration: 34.2us (slow) / 18.9us (fast)
RFAL_WUM_MEAS_DUR_44_28	WU measurement duration: 43.7us (slow) / 28.3us (fast)

◆ rfalWumMeasFilter

enum **rfalWumMeasFilter**

RFAL Wake-Up mesurement filter

Enumerator	
RFAL_WUM_MEAS_FIL_SLOW	Wake-Up measurement slow filter
RFAL_WUM_MEAS_FIL_FAST	Wake-up measurement fast filter
RFAL_WUM_MEAS_FIL_SLOW	Wake-Up measurement slow filter
RFAL_WUM_MEAS_FIL_FAST	Wake-up measurement fast filter

◆ rfalWumPeriod

enum **rfalWumPeriod**

RFAL Wake-Up Period/Timer

Enumerator	
RFAL_WUM_PERIOD_10MS	Wake-Up timer ~9.7ms
RFAL_WUM_PERIOD_15MS	Wake-Up timer ~13.3ms
RFAL_WUM_PERIOD_20MS	Wake-Up timer ~19.3ms
RFAL_WUM_PERIOD_25MS	Wake-Up timer ~26.6ms
RFAL_WUM_PERIOD_40MS	Wake-Up timer ~38.7ms
RFAL_WUM_PERIOD_55MS	Wake-Up timer ~53.2ms
RFAL_WUM_PERIOD_80MS	Wake-Up timer ~77.3ms
RFAL_WUM_PERIOD_105MS	Wake-Up timer ~106.3ms
RFAL_WUM_PERIOD_155MS	Wake-Up timer ~154.7ms
RFAL_WUM_PERIOD_215MS	Wake-Up timer ~212.7ms
RFAL_WUM_PERIOD_310MS	Wake-Up timer ~309.3ms
RFAL_WUM_PERIOD_425MS	Wake-Up timer ~425.3ms
RFAL_WUM_PERIOD_620MS	Wake-Up timer ~618.6ms

RFAL_WUM_PERIOD_850MS	Wake-Up timer ~850.6ms
RFAL_WUM_PERIOD_1240MS	Wake-Up timer ~1237.3ms
RFAL_WUM_PERIOD_1700MS	Wake-Up timer ~1701.2ms
RFAL_WUM_PERIOD_10MS	Wake-Up timer 10ms
RFAL_WUM_PERIOD_20MS	Wake-Up timer 20ms
RFAL_WUM_PERIOD_30MS	Wake-Up timer 30ms
RFAL_WUM_PERIOD_40MS	Wake-Up timer 40ms
RFAL_WUM_PERIOD_50MS	Wake-Up timer 50ms
RFAL_WUM_PERIOD_60MS	Wake-Up timer 60ms
RFAL_WUM_PERIOD_70MS	Wake-Up timer 70ms
RFAL_WUM_PERIOD_80MS	Wake-Up timer 80ms
RFAL_WUM_PERIOD_100MS	Wake-Up timer 100ms
RFAL_WUM_PERIOD_200MS	Wake-Up timer 200ms
RFAL_WUM_PERIOD_300MS	Wake-Up timer 300ms
RFAL_WUM_PERIOD_400MS	Wake-Up timer 400ms
RFAL_WUM_PERIOD_500MS	Wake-Up timer 500ms
RFAL_WUM_PERIOD_600MS	Wake-Up timer 600ms
RFAL_WUM_PERIOD_700MS	Wake-Up timer 700ms
RFAL_WUM_PERIOD_800MS	Wake-Up timer 800ms
RFAL_WUM_PERIOD_10MS	Wake-Up timer 10ms
RFAL_WUM_PERIOD_20MS	Wake-Up timer 20ms
RFAL_WUM_PERIOD_30MS	Wake-Up timer 30ms
RFAL_WUM_PERIOD_40MS	Wake-Up timer 40ms
RFAL_WUM_PERIOD_50MS	Wake-Up timer 50ms
RFAL_WUM_PERIOD_60MS	Wake-Up timer 60ms
RFAL_WUM_PERIOD_70MS	Wake-Up timer 70ms
RFAL_WUM_PERIOD_80MS	Wake-Up timer 80ms
RFAL_WUM_PERIOD_100MS	Wake-Up timer 100ms

RFAL_WUM_PERIOD_200MS	Wake-Up timer 200ms
RFAL_WUM_PERIOD_300MS	Wake-Up timer 300ms
RFAL_WUM_PERIOD_400MS	Wake-Up timer 400ms
RFAL_WUM_PERIOD_500MS	Wake-Up timer 500ms
RFAL_WUM_PERIOD_600MS	Wake-Up timer 600ms
RFAL_WUM_PERIOD_700MS	Wake-Up timer 700ms
RFAL_WUM_PERIOD_800MS	Wake-Up timer 800ms
RFAL_WUM_PERIOD_12MS	Wake-Up timer ~12.5ms
RFAL_WUM_PERIOD_25MS	Wake-Up timer ~25ms
RFAL_WUM_PERIOD_50MS	Wake-Up timer ~50ms
RFAL_WUM_PERIOD_75MS	Wake-Up timer ~75ms
RFAL_WUM_PERIOD_100MS	Wake-Up timer ~100ms
RFAL_WUM_PERIOD_125MS	Wake-Up timer ~125ms
RFAL_WUM_PERIOD_150MS	Wake-Up timer ~150ms
RFAL_WUM_PERIOD_200MS	Wake-Up timer ~200ms
RFAL_WUM_PERIOD_250MS	Wake-Up timer ~250ms
RFAL_WUM_PERIOD_300MS	Wake-Up timer ~300ms
RFAL_WUM_PERIOD_350MS	Wake-Up timer ~350ms
RFAL_WUM_PERIOD_400MS	Wake-Up timer ~400ms
RFAL_WUM_PERIOD_500MS	Wake-Up timer ~600ms
RFAL_WUM_PERIOD_800MS	Wake-Up timer ~800ms
RFAL_WUM_PERIOD_1000MS	Wake-Up timer ~1000ms
RFAL_WUM_PERIOD_INF	Wake-Up timer Infinite
RFAL_WUM_PERIOD_300MS	Wake-Up timer 300ms

st25r95/rfal_features.h File Reference

RFAL Features/Capabilities Definition for ST25R95. More...

```
#include "rfal_platform.h"
```

Include dependency graph for st25r95/rfal_features.h:

Data Structures

struct	**rfalWakeUpConfig**
struct	**rfalWakeUpInfo**

Macros

```
#define RFAL_SUPPORT_MODE_POLL_NFCA true
#define RFAL_SUPPORT_MODE_POLL_NFCB true
#define RFAL_SUPPORT_MODE_POLL_NFCF true
#define RFAL_SUPPORT_MODE_POLL_NFCV true
#define RFAL_SUPPORT_MODE_POLL_ACTIVE_P2P false
#define RFAL_SUPPORT_MODE_LISTEN_NFCA false
#define RFAL_SUPPORT_MODE_LISTEN_NFCB false
#define RFAL_SUPPORT_MODE_LISTEN_NFCF false
#define RFAL_SUPPORT_MODE_LISTEN_ACTIVE_P2P false
#define RFAL_SUPPORT_CE ( RFAL_SUPPORT_MODE_LISTEN_NFCA || RFAL_SUPPORT_MODE_LISTEN_NFCB ||
        RFAL_SUPPORT_MODE_LISTEN_NFCF )
#define RFAL_SUPPORT_RW ( RFAL_SUPPORT_MODE_POLL_NFCA || RFAL_SUPPORT_MODE_POLL_NFCB ||
        RFAL_SUPPORT_MODE_POLL_NFCF || RFAL_SUPPORT_MODE_POLL_NFCV )
#define RFAL_SUPPORT_AP2P ( RFAL_SUPPORT_MODE_POLL_ACTIVE_P2P ||
        RFAL_SUPPORT_MODE_LISTEN_ACTIVE_P2P )
#define RFAL_SUPPORT_BR_RW_106 true
#define RFAL_SUPPORT_BR_RW_212 true
#define RFAL_SUPPORT_BR_RW_424 true
#define RFAL_SUPPORT_BR_RW_848 true
#define RFAL_SUPPORT_BR_RW_1695 false
#define RFAL_SUPPORT_BR_RW_3390 false
#define RFAL_SUPPORT_BR_RW_6780 false
#define RFAL_SUPPORT_BR_AP2P_106 false
```

#define **RFAL_SUPPORT_BR_AP2P_212** false

#define **RFAL_SUPPORT_BR_AP2P_424** false

#define **RFAL_SUPPORT_BR_AP2P_848** false

#define **RFAL_SUPPORT_BR_AP2P_1695** false

#define **RFAL_SUPPORT_BR_AP2P_3390** false

#define **RFAL_SUPPORT_BR_AP2P_6780** false

#define **RFAL_SUPPORT_BR_CE_A_106** true

#define **RFAL_SUPPORT_BR_CE_A_212** false

#define **RFAL_SUPPORT_BR_CE_A_424** false

#define **RFAL_SUPPORT_BR_CE_A_848** false

#define **RFAL_SUPPORT_BR_CE_B_106** false

#define **RFAL_SUPPORT_BR_CE_B_212** false

#define **RFAL_SUPPORT_BR_CE_B_424** false

#define **RFAL_SUPPORT_BR_CE_B_848** false

#define **RFAL_SUPPORT_BR_CE_F_212** false

#define **RFAL_SUPPORT_BR_CE_F_424** false

Enumerations

enum **rfalWumPeriod** {
RFAL_WUM_PERIOD_10MS = 0x00 , **RFAL_WUM_PERIOD_15MS** = 0x01 , **RFAL_WUM_PERIOD_20MS** = 0x02 , **RFAL_WUM_PERIOD_25MS** = 0x03 ,
RFAL_WUM_PERIOD_40MS = 0x04 , **RFAL_WUM_PERIOD_55MS** = 0x05 , **RFAL_WUM_PERIOD_80MS** = 0x06 , **RFAL_WUM_PERIOD_105MS** = 0x07 ,
RFAL_WUM_PERIOD_155MS = 0x08 , **RFAL_WUM_PERIOD_215MS** = 0x09 , **RFAL_WUM_PERIOD_310MS** =

```
0x0A , RFAL_WUM_PERIOD_425MS = 0x0B ,
 RFAL_WUM_PERIOD_620MS = 0x0C , RFAL_WUM_PERIOD_850MS = 0x0D , RFAL_WUM_PERIOD_1240MS =
0x0E , RFAL_WUM_PERIOD_1700MS = 0x0F ,
 RFAL_WUM_PERIOD_10MS = 0x00 , RFAL_WUM_PERIOD_20MS = 0x01 , RFAL_WUM_PERIOD_30MS = 0x02
, RFAL_WUM_PERIOD_40MS = 0x03 ,
 RFAL_WUM_PERIOD_50MS = 0x04 , RFAL_WUM_PERIOD_60MS = 0x05 , RFAL_WUM_PERIOD_70MS = 0x06
, RFAL_WUM_PERIOD_80MS = 0x07 ,
 RFAL_WUM_PERIOD_100MS = 0x10 , RFAL_WUM_PERIOD_200MS = 0x11 , RFAL_WUM_PERIOD_300MS =
0x12 , RFAL_WUM_PERIOD_400MS = 0x13 ,
 RFAL_WUM_PERIOD_500MS = 0x14 , RFAL_WUM_PERIOD_600MS = 0x15 , RFAL_WUM_PERIOD_700MS =
0x16 , RFAL_WUM_PERIOD_800MS = 0x17 ,
 RFAL_WUM_PERIOD_10MS = 0x00 , RFAL_WUM_PERIOD_20MS = 0x01 , RFAL_WUM_PERIOD_30MS = 0x02
, RFAL_WUM_PERIOD_40MS = 0x03 ,
 RFAL_WUM_PERIOD_50MS = 0x04 , RFAL_WUM_PERIOD_60MS = 0x05 , RFAL_WUM_PERIOD_70MS = 0x06
, RFAL_WUM_PERIOD_80MS = 0x07 ,
 RFAL_WUM_PERIOD_100MS = 0x10 , RFAL_WUM_PERIOD_200MS = 0x11 , RFAL_WUM_PERIOD_300MS =
0x12 , RFAL_WUM_PERIOD_400MS = 0x13 ,
 RFAL_WUM_PERIOD_500MS = 0x14 , RFAL_WUM_PERIOD_600MS = 0x15 , RFAL_WUM_PERIOD_700MS =
0x16 , RFAL_WUM_PERIOD_800MS = 0x17 ,
 RFAL_WUM_PERIOD_12MS = 0x00 , RFAL_WUM_PERIOD_25MS = 0x01 , RFAL_WUM_PERIOD_50MS = 0x02
, RFAL_WUM_PERIOD_75MS = 0x03 ,
 RFAL_WUM_PERIOD_100MS = 0x04 , RFAL_WUM_PERIOD_125MS = 0x05 , RFAL_WUM_PERIOD_150MS =
0x06 , RFAL_WUM_PERIOD_200MS = 0x07 ,
 RFAL_WUM_PERIOD_250MS = 0x08 , RFAL_WUM_PERIOD_300MS = 0x09 , RFAL_WUM_PERIOD_350MS =
0x0A , RFAL_WUM_PERIOD_400MS = 0x0B ,
 RFAL_WUM_PERIOD_500MS = 0x0C , RFAL_WUM_PERIOD_800MS = 0x0D , RFAL_WUM_PERIOD_1000MS =
0x0E , RFAL_WUM_PERIOD_INF = 0x0F ,
 RFAL_WUM_PERIOD_300MS = 0x12
}
```

Detailed Description

Author

Macro Definition Documentation

◆ RFAL_SUPPORT_AP2P

```
#define RFAL_SUPPORT_AP2P ( RFAL_SUPPORT_MODE_POLL_ACTIVE_P2P || RFAL_SUPPORT_MODE_LISTEN_ACTIVE_P2P )
```

RFAL support for Active P2P (AP2P)

◆ RFAL_SUPPORT_BR_AP2P_106

```
#define RFAL_SUPPORT_BR_AP2P_106 false
```

RFAL AP2P 106 Bit Rate support switch

◆ RFAL_SUPPORT_BR_AP2P_1695

```
#define RFAL_SUPPORT_BR_AP2P_1695 false
```

RFAL AP2P 1695 Bit Rate support switch

◆ RFAL_SUPPORT_BR_AP2P_212

```
#define RFAL_SUPPORT_BR_AP2P_212 false
```

RFAL AP2P 212 Bit Rate support switch

◆ RFAL_SUPPORT_BR_AP2P_3390

```
#define RFAL_SUPPORT_BR_AP2P_3390 false
```

RFAL AP2P 3390 Bit Rate support switch

◆ RFAL_SUPPORT_BR_AP2P_424

```
#define RFAL_SUPPORT_BR_AP2P_424 false
```

RFAL AP2P 424 Bit Rate support switch

◆ RFAL_SUPPORT_BR_AP2P_6780

```
#define RFAL_SUPPORT_BR_AP2P_6780   false
```

RFAL AP2P 6780 Bit Rate support switch

◆ RFAL_SUPPORT_BR_AP2P_848

```
#define RFAL_SUPPORT_BR_AP2P_848   false
```

RFAL AP2P 848 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_A_106

```
#define RFAL_SUPPORT_BR_CE_A_106   true
```

RFAL CE A 106 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_A_212

```
#define RFAL_SUPPORT_BR_CE_A_212   false
```

RFAL CE A 212 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_A_424

```
#define RFAL_SUPPORT_BR_CE_A_424   false
```

RFAL CE A 424 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_A_848

```
#define RFAL_SUPPORT_BR_CE_A_848   false
```

RFAL CE A 848 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_B_106

```
#define RFAL_SUPPORT_BR_CE_B_106   false
```

RFAL CE B 106 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_B_212

```
#define RFAL_SUPPORT_BR_CE_B_212   false
```

RFAL CE B 212 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_B_424

```
#define RFAL_SUPPORT_BR_CE_B_424   false
```

RFAL CE B 424 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_B_848

```
#define RFAL_SUPPORT_BR_CE_B_848   false
```

RFAL CE B 848 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_F_212

```
#define RFAL_SUPPORT_BR_CE_F_212   false
```

RFAL CE F 212 Bit Rate support switch

◆ RFAL_SUPPORT_BR_CE_F_424

```
#define RFAL_SUPPORT_BR_CE_F_424   false
```

RFAL CE F 424 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_106

```
#define RFAL_SUPPORT_BR_RW_106   true
```

RFAL RW 106 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_1695

```
#define RFAL_SUPPORT_BR_RW_1695   false
```

RFAL RW 1695 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_212

```
#define RFAL_SUPPORT_BR_RW_212   true
```

RFAL RW 212 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_3390

```
#define RFAL_SUPPORT_BR_RW_3390   false
```

RFAL RW 3390 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_424

```
#define RFAL_SUPPORT_BR_RW_424   true
```

RFAL RW 424 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_6780

```
#define RFAL_SUPPORT_BR_RW_6780   false
```

RFAL RW 6780 Bit Rate support switch

◆ RFAL_SUPPORT_BR_RW_848

```
#define RFAL_SUPPORT_BR_RW_848   true
```

RFAL RW 848 Bit Rate support switch

◆ RFAL_SUPPORT_CE

```
#define RFAL_SUPPORT_CE   ( RFAL_SUPPORT_MODE_LISTEN_NFCA || RFAL_SUPPORT_MODE_LISTEN_NFCB ||
RFAL_SUPPORT_MODE_LISTEN_NFCF )
```

RFAL supported Card Emulation (CE)

◆ RFAL_SUPPORT_MODE_LISTEN_ACTIVE_P2P

```
#define RFAL_SUPPORT_MODE_LISTEN_ACTIVE_P2P   false
```

RFAL Listen AP2P mode support switch

◆ RFAL_SUPPORT_MODE_LISTEN_NFCA

```
#define RFAL_SUPPORT_MODE_LISTEN_NFCA   false
```

RFAL Listen NFCA mode support switch

◆ RFAL_SUPPORT_MODE_LISTEN_NFCB

```
#define RFAL_SUPPORT_MODE_LISTEN_NFCB   false
```

RFAL Listen NFCB mode support switch

◆ RFAL_SUPPORT_MODE_LISTEN_NFCF

```
#define RFAL_SUPPORT_MODE_LISTEN_NFCF   false
```

RFAL Listen NFCF mode support switch

◆ RFAL_SUPPORT_MODE_POLL_ACTIVE_P2P

```
#define RFAL_SUPPORT_MODE_POLL_ACTIVE_P2P   false
```

RFAL Poll AP2P mode support switch

◆ RFAL_SUPPORT_MODE_POLL_NFCA

```
#define RFAL_SUPPORT_MODE_POLL_NFCA   true
```

RFAL Poll NFCA mode support switch

◆ RFAL_SUPPORT_MODE_POLL_NFCB

```
#define RFAL_SUPPORT_MODE_POLL_NFCB   true
```

RFAL Poll NFCB mode support switch

◆ RFAL_SUPPORT_MODE_POLL_NFCF

```
#define RFAL_SUPPORT_MODE_POLL_NFCF   true
```

RFAL Poll NFCF mode support switch

◆ RFAL_SUPPORT_MODE_POLL_NFCV

```
#define RFAL_SUPPORT_MODE_POLL_NFCV   true
```

RFAL Poll NFCV mode support switch

◆ RFAL_SUPPORT_RW

#define RFAL_SUPPORT_RW (**RFAL_SUPPORT_MODE_POLL_NFCA** || **RFAL_SUPPORT_MODE_POLL_NFCB** || **RFAL_SUPPORT_MODE_POLL_NFCF** || **RFAL_SUPPORT_MODE_POLL_NFCV**)

RFAL supported Reader/Writer (RW)

Enumeration Type Documentation

◆ rfalWumPeriod

enum **rfalWumPeriod**

RFAL Wake-Up Period/Timer

Enumerator	
RFAL_WUM_PERIOD_10MS	Wake-Up timer ~9.7ms
RFAL_WUM_PERIOD_15MS	Wake-Up timer ~13.3ms
RFAL_WUM_PERIOD_20MS	Wake-Up timer ~19.3ms
RFAL_WUM_PERIOD_25MS	Wake-Up timer ~26.6ms
RFAL_WUM_PERIOD_40MS	Wake-Up timer ~38.7ms
RFAL_WUM_PERIOD_55MS	Wake-Up timer ~53.2ms
RFAL_WUM_PERIOD_80MS	Wake-Up timer ~77.3ms
RFAL_WUM_PERIOD_105MS	Wake-Up timer ~106.3ms
RFAL_WUM_PERIOD_155MS	Wake-Up timer ~154.7ms
RFAL_WUM_PERIOD_215MS	Wake-Up timer ~212.7ms
RFAL_WUM_PERIOD_310MS	Wake-Up timer ~309.3ms
RFAL_WUM_PERIOD_425MS	Wake-Up timer ~425.3ms
RFAL_WUM_PERIOD_620MS	Wake-Up timer ~618.6ms
RFAL_WUM_PERIOD_850MS	Wake-Up timer ~850.6ms
RFAL_WUM_PERIOD_1240MS	Wake-Up timer ~1237.3ms
RFAL_WUM_PERIOD_1700MS	Wake-Up timer ~1701.2ms
RFAL_WUM_PERIOD_10MS	Wake-Up timer 10ms
RFAL_WUM_PERIOD_20MS	Wake-Up timer 20ms
RFAL_WUM_PERIOD_30MS	Wake-Up timer 30ms
RFAL_WUM_PERIOD_40MS	Wake-Up timer 40ms
RFAL_WUM_PERIOD_50MS	Wake-Up timer 50ms

RFAL_WUM_PERIOD_60MS	Wake-Up timer 60ms
RFAL_WUM_PERIOD_70MS	Wake-Up timer 70ms
RFAL_WUM_PERIOD_80MS	Wake-Up timer 80ms
RFAL_WUM_PERIOD_100MS	Wake-Up timer 100ms
RFAL_WUM_PERIOD_200MS	Wake-Up timer 200ms
RFAL_WUM_PERIOD_300MS	Wake-Up timer 300ms
RFAL_WUM_PERIOD_400MS	Wake-Up timer 400ms
RFAL_WUM_PERIOD_500MS	Wake-Up timer 500ms
RFAL_WUM_PERIOD_600MS	Wake-Up timer 600ms
RFAL_WUM_PERIOD_700MS	Wake-Up timer 700ms
RFAL_WUM_PERIOD_800MS	Wake-Up timer 800ms
RFAL_WUM_PERIOD_10MS	Wake-Up timer 10ms
RFAL_WUM_PERIOD_20MS	Wake-Up timer 20ms
RFAL_WUM_PERIOD_30MS	Wake-Up timer 30ms
RFAL_WUM_PERIOD_40MS	Wake-Up timer 40ms
RFAL_WUM_PERIOD_50MS	Wake-Up timer 50ms
RFAL_WUM_PERIOD_60MS	Wake-Up timer 60ms
RFAL_WUM_PERIOD_70MS	Wake-Up timer 70ms
RFAL_WUM_PERIOD_80MS	Wake-Up timer 80ms
RFAL_WUM_PERIOD_100MS	Wake-Up timer 100ms
RFAL_WUM_PERIOD_200MS	Wake-Up timer 200ms
RFAL_WUM_PERIOD_300MS	Wake-Up timer 300ms
RFAL_WUM_PERIOD_400MS	Wake-Up timer 400ms
RFAL_WUM_PERIOD_500MS	Wake-Up timer 500ms
RFAL_WUM_PERIOD_600MS	Wake-Up timer 600ms
RFAL_WUM_PERIOD_700MS	Wake-Up timer 700ms
RFAL_WUM_PERIOD_800MS	Wake-Up timer 800ms
RFAL_WUM_PERIOD_12MS	Wake-Up timer ~12.5ms

RFAL_WUM_PERIOD_25MS	Wake-Up timer ~25ms
RFAL_WUM_PERIOD_50MS	Wake-Up timer ~50ms
RFAL_WUM_PERIOD_75MS	Wake-Up timer ~75ms
RFAL_WUM_PERIOD_100MS	Wake-Up timer ~100ms
RFAL_WUM_PERIOD_125MS	Wake-Up timer ~125ms
RFAL_WUM_PERIOD_150MS	Wake-Up timer ~150ms
RFAL_WUM_PERIOD_200MS	Wake-Up timer ~200ms
RFAL_WUM_PERIOD_250MS	Wake-Up timer ~250ms
RFAL_WUM_PERIOD_300MS	Wake-Up timer ~300ms
RFAL_WUM_PERIOD_350MS	Wake-Up timer ~350ms
RFAL_WUM_PERIOD_400MS	Wake-Up timer ~400ms
RFAL_WUM_PERIOD_500MS	Wake-Up timer ~600ms
RFAL_WUM_PERIOD_800MS	Wake-Up timer ~800ms
RFAL_WUM_PERIOD_1000MS	Wake-Up timer ~1000ms
RFAL_WUM_PERIOD_INF	Wake-Up timer Infinite
RFAL_WUM_PERIOD_300MS	Wake-Up timer 300ms

RFAL

Main Page

Modules

Data Structures

Examples

File List

st25r200.h File Reference

ST25R200 high level interface. More...

```
#include "rfal_platform.h"
#include "rfal_utils.h"
#include "st25r200_com.h"
```

Include dependency graph for st25r200.h:

Macros

#define **ST25R200_CMD_SET_DEFAULT** 0x60U

#define **ST25R200_CMD_STOP** 0x62U

#define **ST25R200_CMD_CLEAR_FIFO** 0x64U

#define **ST25R200_CMD_CLEAR_RXGAIN** 0x66U

#define **ST25R200_CMD_ADJUST_REGULATORS** 0x68U

#define **ST25R200_CMD_TRANSMIT** 0x6AU

#define **ST25R200_CMD_TRANSMIT_EOF** 0x6CU

```
#define ST25R200_CMD_MASK_RECEIVE_DATA 0x70U
#define ST25R200_CMD_UNMASK_RECEIVE_DATA 0x72U
#define ST25R200_CMD_CALIBRATE_WU 0x74U
#define ST25R200_CMD_CLEAR_WU_CALIB 0x76U
#define ST25R200_CMD_MEASURE_WU 0x78U
#define ST25R200_CMD_MEASURE_IQ 0x7AU
#define ST25R200_CMD_SENSE_RF 0x7CU
#define ST25R200_CMD_TRANSPARENT_MODE 0xE0U
#define ST25R200_CMD_START_GP_TIMER 0xE2U
#define ST25R200_CMD_START_WUT 0xE4U
#define ST25R200_CMD_START_MRT 0xE6U
#define ST25R200_CMD_START_NRT 0xE8U
#define ST25R200_CMD_STOP_NRT 0xEAU
#define ST25R200_CMD_TEST_ACCESS 0xFCU
#define ST25R200_BR_DO_NOT_SET 0xFFU
#define ST25R200_BR_106_26 0x00U
#define ST25R200_BR_212 0x01U
#define ST25R200_BR_424_53 0x02U
#define ST25R200_BR_848 0x03U
#define ST25R200_REG_DROP_200 0U
#define ST25R200_REG_DROP_250 1U
```

#define **ST25R200_REG_DROP_300** 2U

#define **ST25R200_REG_DROP_350** 3U

#define **ST25R200_REG_DROP_400** 4U

#define **ST25R200_REG_DROP_450** 5U

#define **ST25R200_REG_DROP_500** 6U

#define **ST25R200_REG_DROP_550** 7U

#define **ST25R200_REG_DROP_DO_NOT_SET** 0xFFU

#define **ST25R200_REG_LEN** 1U

#define **ST25R200_CMD_LEN** 1U

#define **ST25R200_FIFO_DEPTH** 256U

#define **ST25R200_WRITE_MODE** (0U << 7)

#define **ST25R200_READ_MODE** (1U << 7)

#define **ST25R200_CMD_MODE** **ST25R200_WRITE_MODE**

#define **ST25R200_FIFO_ACCESS** (0x5FU)

#define **st25r200TxRxOn**() **st25r200SetRegisterBits**(**ST25R200_REG_OPERATION**, (ST25R200_REG_OPERATION_rx_en | ST25R200_REG_OPERATION_tx_en))

#define **st25r200TxRxOff**() **st25r200ClrRegisterBits**(**ST25R200_REG_OPERATION**, (ST25R200_REG_OPERATION_rx_en | ST25R200_REG_OPERATION_tx_en))

#define **st25r200TxOff**() **st25r200ClrRegisterBits**(**ST25R200_REG_OPERATION**, ST25R200_REG_OPERATION_tx_en)

#define **st25r200IsGPTRunning**() **st25r200CheckReg**(**ST25R200_REG_STATUS**, ST25R200_REG_STATUS_gpt_on, ST25R200_REG_STATUS_gpt_on)

#define **st25r200IsTxEnabled**() **st25r200CheckReg**(**ST25R200_REG_OPERATION**, ST25R200_REG_OPERATION_tx_en, ST25R200_REG_OPERATION_tx_en)

#define **st25r200IsNRTinEMV**() **st25r200CheckReg**(**ST25R200_REG_NRT_GPT_CONF**, ST25R200_REG_NRT_GPT_CONF_nrt_emd, ST25R200_REG_NRT_GPT_CONF_nrt_emd_on)

#define **st25r200IsLastFIFOComplete**() **st25r200CheckReg**(**ST25R200_REG_FIFO_STATUS2**, ST25R200_REG_FIFO_STATUS2_fifo_lb_mask, 0)

#define **st25r200IsOscOn**() **st25r200CheckReg**(**ST25R200_REG_OPERATION**, ST25R200_REG_OPERATION_en, ST25R200_REG_OPERATION_en)

#define **st25r200IsTxOn**() **st25r200CheckReg**(**ST25R200_REG_OPERATION**, ST25R200_REG_OPERATION_tx_en, ST25R200_REG_OPERATION_tx_en)

Functions

ReturnCode **st25r200Initialize** (void)
Initialise ST25R200 driver. More...

void **st25r200Deinitialize** (void)
Deinitialize ST25R200 driver. More...

ReturnCode **st25r200OscOn** (void)
Turn on Oscillator and Regulator. More...

ReturnCode **st25r200SetBitrate** (uint8_t txRate, uint8_t rxRate)
Sets the bitrate. More...

ReturnCode **st25r200AdjustRegulators** (uint8_t drop, uint16_t *result)
Adjusts supply regulators according to the current supply voltage. More...

ReturnCode **st25r200SetRegulators** (uint8_t regulation)
Sets supply regulators. More...

ReturnCode **st25r200MeasureIQ** (int8_t *resI, int8_t *resQ)
Measure I/Q. More...

ReturnCode **st25r200MeasureCombinedIQ** (uint8_t *res)
Measure Combined I/Q. More...

ReturnCode	**st25r200MeasureI** (uint8_t *res) Measure I. More...
ReturnCode	**st25r200MeasureQ** (uint8_t *res) Measure Q. More...
ReturnCode	**st25r200CalibrateWU** (uint8_t *resI, uint8_t *resQ) Calibrate WU. More...
ReturnCode	**st25r200ClearCalibration** (void) Clear WU Calibration. More...
ReturnCode	**st25r200MeasureWU** (uint8_t *resI, uint8_t *resQ) Measure WU. More...
bool	**st25r200IsExtFieldOn** (void) Check External Filed. More...
uint32_t	**st25r200GetNoResponseTime** (void) Get NRT time. More...
ReturnCode	**st25r200SetNoResponseTime** (uint32_t nrt) Set NRT time. More...
ReturnCode	**st25r200SetStartNoResponseTimer** (uint32_t nrt) Set and Start NRT. More...
void	**st25r200SetGPTime** (uint16_t gpt) Set GPT time. More...
ReturnCode	**st25r200SetStartGPTimer** (uint16_t gpt, uint8_t trigger) Set and Start GPT. More...
void	**st25r200SetNumTxBits** (uint16_t nBits) Sets the number Tx Bits. More...
uint16_t	**st25r200GetNumFIFOBytes** (void) Get amount of bytes in FIFO. More...

uint8_t	**st25r200GetNumFIFOLastBits** (void) Get amount of bits of the last FIFO byte if incomplete. More...
ReturnCode	**st25r200PerformCollisionAvoidance** (void) Perform Collision Avoidance. More...
bool	**st25r200CheckChipID** (uint8_t *rev) Check Identity. More...
ReturnCode	**st25r200GetRegsDump** (uint8_t *resRegDump, uint8_t *sizeRegDump) Register Dump. More...
bool	**st25r200IsCmdValid** (uint8_t cmd) Check if command is valid. More...
ReturnCode	**st25r200ExecuteCommandAndGetResult** (uint8_t cmd, uint8_t resReg, uint8_t tOut, uint8_t *result) Executes a direct command and returns the result. More...
ReturnCode	**st25r200GetRSSI** (uint16_t *iRssi, uint16_t *qRssi) Gets the RSSI values. More...
ReturnCode	**st25r200SetAntennaMode** (bool single, bool rfiox) Set Antenna mode. More...

Detailed Description

Author

Gustavo Patricio

st25r200_com.h File Reference

ST25R200 communication declaration file. More...

```
#include "rfal_platform.h"
#include "rfal_utils.h"
```

Include dependency graph for st25r200_com.h:

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

Macros

#define **ST25R200_FIFO_STATUS_LEN** 2

```
#define ST25R200_OBS_MODE_LEN 2
#define ST25R200_REG_OPERATION 0x00U
#define ST25R200_REG_GENERAL 0x01U
#define ST25R200_REG_REGULATOR 0x02U
#define ST25R200_REG_TX_DRIVER 0x03U
#define ST25R200_REG_TX_MOD 0x04U
#define ST25R200_REG_TX_RES_MOD 0x05U
#define ST25R200_REG_RX_ANA1 0x06U
#define ST25R200_REG_RX_ANA2 0x07U
#define ST25R200_REG_RX_DIG 0x08U
#define ST25R200_REG_CORR1 0x09U
#define ST25R200_REG_CORR2 0x0AU
#define ST25R200_REG_CORR3 0x0BU
#define ST25R200_REG_CORR4 0x0CU
#define ST25R200_REG_CORR5 0x0DU
#define ST25R200_REG_CORR6 0x0EU
#define ST25R200_REG_DISPLAY1 0x0FU
#define ST25R200_REG_DISPLAY2 0x10U
#define ST25R200_REG_STATUS 0x11U
#define ST25R200_REG_PROTOCOL 0x12U
#define ST25R200_REG_PROTOCOL_TX1 0x13U
```

```
#define ST25R200_REG_PROTOCOL_TX2 0x14U
#define ST25R200_REG_PROTOCOL_TX3 0x15U
#define ST25R200_REG_PROTOCOL_RX1 0x16U
#define ST25R200_REG_PROTOCOL_RX2 0x17U
#define ST25R200_REG_PROTOCOL_RX3 0x18U
#define ST25R200_REG_EMD1 0x19U
#define ST25R200_REG_EMD2 0x1AU
#define ST25R200_REG_MRT_SQT_CONF 0x1BU
#define ST25R200_REG_MRT 0x1CU
#define ST25R200_REG_SQT 0x1DU
#define ST25R200_REG_NRT_GPT_CONF 0x1EU
#define ST25R200_REG_NRT1 0x1FU
#define ST25R200_REG_NRT2 0x20U
#define ST25R200_REG_GPT1 0x21U
#define ST25R200_REG_GPT2 0x22U
#define ST25R200_REG_DISPLAY3 0x23U
#define ST25R200_REG_DISPLAY4 0x24U
#define ST25R200_REG_OVUNSHOOT_CONF 0x25U
#define ST25R200_REG_OVERSHOOT_CONF 0x26U
#define ST25R200_REG_UNDERSHOOT_CONF 0x27U
#define ST25R200_REG_WAKEUP_CONF1 0x28U
```

```
#define ST25R200_REG_WAKEUP_CONF2 0x29U
#define ST25R200_REG_WU_I_CONF 0x2AU
#define ST25R200_REG_WU_I_DELTA 0x2BU
#define ST25R200_REG_WU_I_ADC 0x2CU
#define ST25R200_REG_WU_I_REF 0x2DU
#define ST25R200_REG_WU_I_CAL 0x2EU
#define ST25R200_REG_WU_Q_CONF 0x2FU
#define ST25R200_REG_WU_Q_DELTA 0x30U
#define ST25R200_REG_WU_Q_ADC 0x31U
#define ST25R200_REG_WU_Q_REF 0x32U
#define ST25R200_REG_WU_Q_CAL 0x33U
#define ST25R200_REG_TX_FRAME1 0x34U
#define ST25R200_REG_TX_FRAME2 0x35U
#define ST25R200_REG_FIFO_STATUS1 0x36U
#define ST25R200_REG_FIFO_STATUS2 0x37U
#define ST25R200_REG_COLLISION 0x38U
#define ST25R200_REG_IRQ_MASK1 0x39U
#define ST25R200_REG_IRQ_MASK2 0x3AU
#define ST25R200_REG_IRQ_MASK3 0x3BU
#define ST25R200_REG_IRQ1 0x3CU
#define ST25R200_REG_IRQ2 0x3DU
```

#define **ST25R200_REG_IRQ3** 0x3EU

#define **ST25R200_REG_IC_ID** 0x3FU

Functions

ReturnCode **st25r200ReadRegister** (uint8_t reg, uint8_t *val)
Returns the content of a register within the ST25R200. More...

ReturnCode **st25r200ReadMultipleRegisters** (uint8_t reg, uint8_t *values, uint16_t length)
Reads from multiple ST25R200 registers. More...

ReturnCode **st25r200WriteRegister** (uint8_t reg, uint8_t val)
Writes a given value to a register within the ST25R200. More...

ReturnCode **st25r200WriteMultipleRegisters** (uint8_t reg, const uint8_t *values, uint16_t length)
Writes multiple values to ST25R200 registers. More...

ReturnCode **st25r200WriteFifo** (const uint8_t *values, uint16_t length)
Writes values to ST25R200 FIFO. More...

ReturnCode **st25r200ReadFifo** (uint8_t *buf, uint16_t length)
Read values from ST25R200 FIFO. More...

ReturnCode **st25r200ExecuteCommand** (uint8_t cmd)
Execute a direct command. More...

ReturnCode **st25r200ExecuteCommands** (const uint8_t *cmds, uint8_t length)
Execute direct commands. More...

ReturnCode **st25r200ReadTestRegister** (uint8_t reg, uint8_t *val)
Read a test register within the ST25R200. More...

ReturnCode **st25r200WriteTestRegister** (uint8_t reg, uint8_t val)
Writes a given value to a test register within the ST25R200. More...

ReturnCode **st25r200WriteMultipleTestRegister** (uint8_t reg, const uint8_t *values, uint8_t length)
Writes a multiple values to test registers within the ST25R200. More...

ReturnCode	**st25r200ClrRegisterBits** (uint8_t reg, uint8_t clr_mask) Cleart bits on Register. More...
ReturnCode	**st25r200SetRegisterBits** (uint8_t reg, uint8_t set_mask) Set bits on Register. More...
ReturnCode	**st25r200ChangeRegisterBits** (uint8_t reg, uint8_t valueMask, uint8_t value) Changes the given bits on a ST25R200 register. More...
ReturnCode	**st25r200ModifyRegister** (uint8_t reg, uint8_t clr_mask, uint8_t set_mask) Modifies a value within a ST25R200 register. More...
ReturnCode	**st25r200ChangeTestRegisterBits** (uint8_t reg, uint8_t valueMask, uint8_t value) Changes the given bits on a ST25R200 Test register. More...
bool	**st25r200CheckReg** (uint8_t reg, uint8_t mask, uint8_t val) Checks if register contains a expected value. More...
bool	**st25r200IsRegValid** (uint8_t reg) Check if register ID is valid. More...
void	**st25r200ledInit** (void) ST25R200 LED Initialize. More...
void	**st25r200ledEvtIrq** (uint32_t irqs) ST25R200 LED Event Interrupt. More...
void	**st25r200ledEvtWrReg** (uint8_t reg, uint8_t val) ST25R200 LED Event Write Register. More...
void	**st25r200ledEvtWrMultiReg** (uint8_t reg, const uint8_t *vals, uint8_t len) ST25R200 LED Event Write Multiple Register. More...
void	**st25r200ledEvtCmd** (uint8_t cmd) ST25R200 LED Event Direct Command. More...
void	**st25r200ledEvtCmds** (const uint8_t *cmds, uint8_t len) ST25R200 LED Event Direct Commands. More...

Detailed Description

Author

Gustavo Patricio

This driver provides basic abstraction for communication with the ST25R200

st25r200_irq.h File Reference

ST25R200 Interrupt handling. More...

```
#include "rfal_platform.h"
```

Include dependency graph for st25r200_irq.h:

Macros

#define **ST25R200_IRQ_MASK_ALL** (uint32_t)(0xFFFFFFFFUL)

#define **ST25R200_IRQ_MASK_NONE** (uint32_t)(0x00000000UL)

#define **ST25R200_IRQ_MASK_SUBC_START** (uint32_t)(0x00000080U)

#define **ST25R200_IRQ_MASK_COL** (uint32_t)(0x00000040U)

#define **ST25R200_IRQ_MASK_WL** (uint32_t)(0x00000020U)

#define **ST25R200_IRQ_MASK_RX_REST** (uint32_t)(0x00000010U)

#define **ST25R200_IRQ_MASK_RXE** (uint32_t)(0x00000008U)

#define **ST25R200_IRQ_MASK_RXS** (uint32_t)(0x00000004U)

#define **ST25R200_IRQ_MASK_TXE** (uint32_t)(0x00000002U)

#define **ST25R200_IRQ_MASK_RFU1** (uint32_t)(0x00000001U)

#define **ST25R200_IRQ_MASK_GPE** (uint32_t)(0x00008000U)

#define **ST25R200_IRQ_MASK_NRE** (uint32_t)(0x00004000U)

#define **ST25R200_IRQ_MASK_RFU2** (uint32_t)(0x00002000U)

#define **ST25R200_IRQ_MASK_RFU3** (uint32_t)(0x00001000U)

#define **ST25R200_IRQ_MASK_CRC** (uint32_t)(0x00000800U)

#define **ST25R200_IRQ_MASK_PAR** (uint32_t)(0x00000400U)

#define **ST25R200_IRQ_MASK_HFE** (uint32_t)(0x00000200U)

#define **ST25R200_IRQ_MASK_SFE** (uint32_t)(0x00000100U)

#define **ST25R200_IRQ_MASK_RFU4** (uint32_t)(0x00800000U)

#define **ST25R200_IRQ_MASK_RFU5** (uint32_t)(0x00400000U)

#define **ST25R200_IRQ_MASK_RFU6** (uint32_t)(0x00200000U)

#define **ST25R200_IRQ_MASK_DCT** (uint32_t)(0x00100000U)

#define **ST25R200_IRQ_MASK_WUQ** (uint32_t)(0x00080000U)

#define **ST25R200_IRQ_MASK_WUI** (uint32_t)(0x00040000U)

#define **ST25R200_IRQ_MASK_WUT** (uint32_t)(0x00020000U)

#define **ST25R200_IRQ_MASK_OSC** (uint32_t)(0x00010000U)

Functions

uint32_t **st25r200WaitForInterruptsTimed** (uint32_t mask, uint16_t tmo)

Wait until an ST25R200 interrupt occurs. More...

uint32_t **st25r200GetInterrupt** (uint32_t mask)

Get status for the given interrupt. More...

void **st25r200InitInterrupts** (void)

Init the 200 interrupt. More...

void **st25r200ModifyInterrupts** (uint32_t clr_mask, uint32_t set_mask)

Modifies the Interrupt. More...

void **st25r200CheckForReceivedInterrupts** (void)

Checks received interrupts. More...

void **st25r200Isr** (void)

ISR Service routine. More...

void **st25r200EnableInterrupts** (uint32_t mask)
Enable a given ST25R200 Interrupt source. More...

void **st25r200DisableInterrupts** (uint32_t mask)
Disable one or more a given ST25R200 Interrupt sources. More...

void **st25r200ClearInterrupts** (void)
Clear all ST25R200 irq flags. More...

void **st25r200ClearAndEnableInterrupts** (uint32_t mask)
Clears and then enables the given ST25R200 Interrupt sources. More...

void **st25r200IRQCallbackSet** (void(*cb)(void))
Sets IRQ callback for the ST25R200 interrupt. More...

void **st25r200IRQCallbackRestore** (void)
Sets IRQ callback for the ST25R200 interrupt. More...

Detailed Description

Author

Gustavo Patricio

st25r3911 Directory Reference

Directory dependency graph for st25r3911:

Files

file **st25r3911/rfal_features.h**
RFAL Features/Capabilities Definition for ST25R3911.

file **st25r3911.h**
ST25R3911 declaration file.

file **st25r3911_com.h**
ST25R3911 communication declaration file.

file **st25r3911_interrupt.h**
ST25R3911 Interrupt header file.

st25r3911.h File Reference

ST25R3911 declaration file. More...

```
#include "rfal_platform.h"
#include "rfal_utils.h"
```

Include dependency graph for st25r3911.h:

Data Structures

struct **st25r3911StreamConfig**

Macros

#define **ST25R3911_FDT_NONE** 0x00U

```
#define MS_TO_64FCS(A)  ((A) * 212U)
#define MS_FROM_64FCS(A)  ((A) / 212U)
#define ST25R3911_CMD_SET_DEFAULT  0xC1U
#define ST25R3911_CMD_CLEAR_FIFO  0xC2U
#define ST25R3911_CMD_TRANSMIT_WITH_CRC  0xC4U
#define ST25R3911_CMD_TRANSMIT_WITHOUT_CRC  0xC5U
#define ST25R3911_CMD_TRANSMIT_REQA  0xC6U
#define ST25R3911_CMD_TRANSMIT_WUPA  0xC7U
#define ST25R3911_CMD_INITIAL_RF_COLLISION  0xC8U
#define ST25R3911_CMD_RESPONSE_RF_COLLISION_N  0xC9U
#define ST25R3911_CMD_RESPONSE_RF_COLLISION_0  0xCAU
#define ST25R3911_CMD_NORMAL_NFC_MODE  0xCBU
#define ST25R3911_CMD_ANALOG_PRESET  0xCCU
#define ST25R3911_CMD_MASK_RECEIVE_DATA  0xD0U
#define ST25R3911_CMD_UNMASK_RECEIVE_DATA  0xD1U
#define ST25R3911_CMD_MEASURE_AMPLITUDE  0xD3U
#define ST25R3911_CMD_SQUELCH  0xD4U
#define ST25R3911_CMD_CLEAR_SQUELCH  0xD5U
#define ST25R3911_CMD_ADJUST_REGULATORS  0xD6U
#define ST25R3911_CMD_CALIBRATE_MODULATION  0xD7U
#define ST25R3911_CMD_CALIBRATE_ANTENNA  0xD8U
```

```
#define ST25R3911_CMD_MEASURE_PHASE 0xD9U
#define ST25R3911_CMD_CLEAR_RSSI 0xDAU
#define ST25R3911_CMD_TRANSPARENT_MODE 0xDCU
#define ST25R3911_CMD_CALIBRATE_C_SENSOR 0xDDU
#define ST25R3911_CMD_MEASURE_CAPACITANCE 0xDEU
#define ST25R3911_CMD_MEASURE_VDD 0xDFU
#define ST25R3911_CMD_START_GP_TIMER 0xE0U
#define ST25R3911_CMD_START_WUP_TIMER 0xE1U
#define ST25R3911_CMD_START_MASK_RECEIVE_TIMER 0xE2U
#define ST25R3911_CMD_START_NO_RESPONSE_TIMER 0xE3U
#define ST25R3911_CMD_TEST_CLEARA 0xFAU
#define ST25R3911_CMD_TEST_CLEARB 0xFBU
#define ST25R3911_CMD_TEST_ACCESS 0xFCU
#define ST25R3911_CMD_LOAD_PPROM 0xFDU
#define ST25R3911_CMD_FUSE_PPROM 0xFEU
#define ST25R3911_FIFO_DEPTH 96U
#define ST25R3911_THRESHOLD_DO_NOT_SET 0xFFU
#define ST25R3911_BR_DO_NOT_SET 0xFFU
#define ST25R3911_BR_106 0x00U
#define ST25R3911_BR_212 0x01U
#define ST25R3911_BR_424 0x02U
```

#define **ST25R3911_BR_848** 0x03U

#define **ST25R3911_BR_1695** 0x04U

#define **ST25R3911_BR_3390** 0x05U

#define **ST25R3911_BR_6780** 0x06U

#define **st25r3911IsGPTRunning**() (**st25r3911CheckReg**(**ST25R3911_REG_REGULATOR_RESULT**, ST25R3911_REG_REGULATOR_RESULT_gpt_on, ST25R3911_REG_REGULATOR_RESULT_gpt_on))

#define **st25r3911IsCRCinFIFO**() (**st25r3911CheckReg**(**ST25R3911_REG_AUX**, ST25R3911_REG_AUX_crc_2_fifo, ST25R3911_REG_AUX_crc_2_fifo))

#define **st25r3911IsExtFieldOn**() (**st25r3911CheckReg**(**ST25R3911_REG_AUX_DISPLAY**, ST25R3911_REG_AUX_DISPLAY_efd_o, ST25R3911_REG_AUX_DISPLAY_efd_o))

#define **st25r3911IsTxEnabled**() (**st25r3911CheckReg**(**ST25R3911_REG_OP_CONTROL**, ST25R3911_REG_OP_CONTROL_tx_en, ST25R3911_REG_OP_CONTROL_tx_en))

#define **st25r3911TxOff**() **st25r3911ClrRegisterBits**(**ST25R3911_REG_OP_CONTROL**, ST25R3911_REG_OP_CONTROL_tx_en);

#define **st25r3911IsLastFIFOComplete**() **st25r3911CheckReg**(**ST25R3911_REG_FIFO_RX_STATUS2**, ST25R3911_REG_FIFO_RX_STATUS2_mask_fifo_lb, 0)

#define **st25r3911IsOscOn**() **st25r3911CheckReg**(**ST25R3911_REG_OP_CONTROL**, ST25R3911_REG_OP_CONTROL_en, ST25R3911_REG_OP_CONTROL_en)

Functions

ReturnCode **st25r3911OscOn** (void)

Turn on Oscillator and Regulator. More...

void **st25r3911TxRxOn** (void)

Turn On Tx and Rx. More...

void **st25r3911TxRxOff** (void)

Turn Off Tx and Rx. More...

ReturnCode	**st25r3911Initialize** (void) Initialise ST25R3911 driver. More...
void	**st25r3911Deinitialize** (void) Deinitialize ST25R3911 driver. More...
ReturnCode	**st25r3911SetBitrate** (uint8_t txRate, uint8_t rxRate) Sets the bitrate registers. More...
ReturnCode	**st25r3911AdjustRegulators** (uint16_t *result_mV) Adjusts supply regulators according to the current supply voltage. More...
ReturnCode	**st25r3911SetRegulators** (uint8_t regulation) Sets supply regulators. More...
void	**st25r3911MeasureAmplitude** (uint8_t *result) Measure Amplitude. More...
void	**st25r3911MeasureCapacitance** (uint8_t *result) Measure Capacitance. More...
uint8_t	**st25r3911MeasurePowerSupply** (uint8_t mpsv) Measure Voltage. More...
uint16_t	**st25r3911MeasureVoltage** (uint8_t mpsv) Measure Voltage. More...
void	**st25r3911CalibrateAntenna** (uint8_t *result) Calibrate antenna. More...
void	**st25r3911MeasurePhase** (uint8_t *result) Measure Phase. More...
void	**st25r3911CalibrateModulationDepth** (uint8_t *result) Calibrate modulation depth. More...
ReturnCode	**st25r3911CalibrateCapacitiveSensor** (uint8_t *result) Calibrate Capacitive Sensor. More...

ReturnCode	**st25r3911SetNoResponseTime_64fcs** (uint32_t nrt_64fcs) set no response time More...
ReturnCode	**st25r3911SetStartNoResponseTime_64fcs** (uint32_t nrt_64fcs) set no response time More...
ReturnCode	**st25r3911PerformCollisionAvoidance** (uint8_t FieldONCmd, uint8_t pdThreshold, uint8_t caThreshold, uint8_t nTRFW) Perform Collision Avoidance. More...
uint8_t	**st25r3911GetNumFIFOLastBits** (void) Get amount of bits of the last FIFO byte if incomplete. More...
uint32_t	**st25r3911GetNoResponseTime_64fcs** (void) Get NRT time. More...
void	**st25r3911SetGPTime_8fcs** (uint16_t gpt_8fcs) set general purpose timer timeout More...
void	**st25r3911StartGPTimer_8fcs** (uint16_t gpt_8fcs, uint8_t trigger_source) Starts GPT with given timeout. More...
bool	**st25r3911CheckReg** (uint8_t reg, uint8_t mask, uint8_t value) Checks if register contains a expected value. More...
void	**st25r3911SetNumTxBits** (uint32_t nBits) Sets the number Tx Bits. More...
bool	**st25r3911CheckChipID** (uint8_t *rev) Check Identity. More...
bool	**st25r3911IsCmdValid** (uint8_t cmd) Check if command is valid. More...
ReturnCode	**st25r3911StreamConfigure** (const struct **st25r3911StreamConfig** *config) Configure the stream mode of ST25R3911. More...
ReturnCode	**st25r3911GetRegsDump** (uint8_t *resRegDump, uint8_t *sizeRegDump)

	Register Dump. More...
ReturnCode	**st25r3911GetRSSI** (uint16_t *amRssi, uint16_t *pmRssi) Gets the RSSI values. More...
ReturnCode	**st25r3911SetAntennaMode** (bool single, bool rfiox) Set Antenna mode. More...

Detailed Description

Author

Ulrich Herrmann

API:

- Initialize ST25R3911 driver: **st25r3911Initialize**
- Deinitialize ST25R3911 driver: **st25r3911Deinitialize**

st25r3911_com.h File Reference

ST25R3911 communication declaration file. More...

```
#include "rfal_platform.h"
#include "rfal_utils.h"
```

Include dependency graph for st25r3911_com.h:

Macros

#define **ST25R3911_FIFO_STATUS_LEN** 2

#define **ST25R3911_REG_IO_CONF1** 0x00U

#define **ST25R3911_REG_IO_CONF2** 0x01U

#define **ST25R3911_REG_OP_CONTROL** 0x02U

#define **ST25R3911_REG_MODE** 0x03U

#define **ST25R3911_REG_BIT_RATE** 0x04U

#define **ST25R3911_REG_ISO14443A_NFC** 0x05U

#define **ST25R3911_REG_ISO14443B_1** 0x06U

#define **ST25R3911_REG_ISO14443B_2** 0x07U

#define **ST25R3911_REG_STREAM_MODE** 0x08U

#define **ST25R3911_REG_AUX** 0x09U

```
#define ST25R3911_REG_RX_CONF1 0x0AU
#define ST25R3911_REG_RX_CONF2 0x0BU
#define ST25R3911_REG_RX_CONF3 0x0CU
#define ST25R3911_REG_RX_CONF4 0x0DU
#define ST25R3911_REG_MASK_RX_TIMER 0x0EU
#define ST25R3911_REG_NO_RESPONSE_TIMER1 0x0FU
#define ST25R3911_REG_NO_RESPONSE_TIMER2 0x10U
#define ST25R3911_REG_GPT_CONTROL 0x11U
#define ST25R3911_REG_GPT1 0x12U
#define ST25R3911_REG_GPT2 0x13U
#define ST25R3911_REG_IRQ_MASK_MAIN 0x14U
#define ST25R3911_REG_IRQ_MASK_TIMER_NFC 0x15U
#define ST25R3911_REG_IRQ_MASK_ERROR_WUP 0x16U
#define ST25R3911_REG_IRQ_MAIN 0x17U
#define ST25R3911_REG_IRQ_TIMER_NFC 0x18U
#define ST25R3911_REG_IRQ_ERROR_WUP 0x19U
#define ST25R3911_REG_FIFO_RX_STATUS1 0x1AU
#define ST25R3911_REG_FIFO_RX_STATUS2 0x1BU
#define ST25R3911_REG_COLLISION_STATUS 0x1CU
#define ST25R3911_REG_NUM_TX_BYTES1 0x1DU
#define ST25R3911_REG_NUM_TX_BYTES2 0x1EU
```

```
#define ST25R3911_REG_NFCIP1_BIT_RATE 0x1FU
#define ST25R3911_REG_AD_RESULT 0x20U
#define ST25R3911_REG_ANT_CAL_CONTROL 0x21U
#define ST25R3911_REG_ANT_CAL_TARGET 0x22U
#define ST25R3911_REG_ANT_CAL_RESULT 0x23U
#define ST25R3911_REG_AM_MOD_DEPTH_CONTROL 0x24U
#define ST25R3911_REG_AM_MOD_DEPTH_RESULT 0x25U
#define ST25R3911_REG_RFO_AM_ON_LEVEL 0x26U
#define ST25R3911_REG_RFO_AM_OFF_LEVEL 0x27U
#define ST25R3911_REG_FIELD_THRESHOLD 0x29U
#define ST25R3911_REG_REGULATOR_CONTROL 0x2AU
#define ST25R3911_REG_REGULATOR_RESULT 0x2BU
#define ST25R3911_REG_RSSI_RESULT 0x2CU
#define ST25R3911_REG_GAIN_RED_STATE 0x2DU
#define ST25R3911_REG_CAP_SENSOR_CONTROL 0x2EU
#define ST25R3911_REG_CAP_SENSOR_RESULT 0x2FU
#define ST25R3911_REG_AUX_DISPLAY 0x30U
#define ST25R3911_REG_WUP_TIMER_CONTROL 0x31U
#define ST25R3911_REG_AMPLITUDE_MEASURE_CONF 0x32U
#define ST25R3911_REG_AMPLITUDE_MEASURE_REF 0x33U
#define ST25R3911_REG_AMPLITUDE_MEASURE_AA_RESULT 0x34U
```

#define **ST25R3911_REG_AMPLITUDE_MEASURE_RESULT** 0x35U

#define **ST25R3911_REG_PHASE_MEASURE_CONF** 0x36U

#define **ST25R3911_REG_PHASE_MEASURE_REF** 0x37U

#define **ST25R3911_REG_PHASE_MEASURE_AA_RESULT** 0x38U

#define **ST25R3911_REG_PHASE_MEASURE_RESULT** 0x39U

#define **ST25R3911_REG_CAPACITANCE_MEASURE_CONF** 0x3AU

#define **ST25R3911_REG_CAPACITANCE_MEASURE_REF** 0x3BU

#define **ST25R3911_REG_CAPACITANCE_MEASURE_AA_RESULT** 0x3CU

#define **ST25R3911_REG_CAPACITANCE_MEASURE_RESULT** 0x3DU

#define **ST25R3911_REG_IC_IDENTITY** 0x3FU

Functions

void **st25r3911ReadRegister** (uint8_t reg, uint8_t *value)

Returns the content of a register within the ST25R3911. More...

void **st25r3911WriteRegister** (uint8_t reg, uint8_t value)

Writes a given value to a register within the ST25R3911. More...

void **st25r3911ClrRegisterBits** (uint8_t reg, uint8_t clr_mask)

Cleart bits on Register. More...

void **st25r3911SetRegisterBits** (uint8_t reg, uint8_t set_mask)

Set bits on Register. More...

void **st25r3911ChangeRegisterBits** (uint8_t reg, uint8_t valueMask, uint8_t value)

Changes the given bits on a ST25R3911 register. More...

void **st25r3911ReadTestRegister** (uint8_t reg, uint8_t *value)

Read a test register within the ST25R3911. More...

void **st25r3911WriteTestRegister** (uint8_t reg, uint8_t value)
Writes a given value to a test register within the ST25R3911. More...

void **st25r3911ModifyRegister** (uint8_t reg, uint8_t clr_mask, uint8_t set_mask)
Modifies a value within a ST25R3911 register. More...

void **st25r3911ChangeTestRegisterBits** (uint8_t reg, uint8_t valueMask, uint8_t value)
Changes the given bits on a ST25R3911 Test register. More...

void **st25r3911WriteMultipleRegisters** (uint8_t reg, const uint8_t *values, uint8_t length)
Writes multiple values to ST25R3911 registers. More...

void **st25r3911ReadMultipleRegisters** (uint8_t reg, uint8_t *values, uint8_t length)
Reads from multiple ST25R3911 registers. More...

void **st25r3911WriteFifo** (const uint8_t *values, uint8_t length)
Writes values to ST25R3911 FIFO. More...

void **st25r3911ReadFifo** (uint8_t *buf, uint8_t length)
Read values from ST25R3911 FIFO. More...

void **st25r3911ExecuteCommand** (uint8_t cmd)
Execute a direct command. More...

void **st25r3911ExecuteCommands** (const uint8_t *cmds, uint8_t length)
Execute several direct commands. More...

bool **st25r3911IsRegValid** (uint8_t reg)
Check if register ID is valid. More...

Detailed Description

Author

Ulrich Herrmann

RFAL

Main Page

Modules

Data Structures

Examples

File List

Globals

source

st25r3911

Macros | Functions

st25r3911_interrupt.h File Reference

ST25R3911 Interrupt header file. More...

```
#include "rfal_platform.h"
```

Include dependency graph for st25r3911_interrupt.h:

Macros

#define **ST25R3911_IRQ_MASK_ALL** (uint32_t)(0xFFFFFFU)

#define **ST25R3911_IRQ_MASK_NONE** (uint32_t)(0U)

#define **ST25R3911_IRQ_MASK_OSC** (uint32_t)(0x80U)

#define **ST25R3911_IRQ_MASK_FWL** (uint32_t)(0x40U)

#define **ST25R3911_IRQ_MASK_RXS** (uint32_t)(0x20U)

#define **ST25R3911_IRQ_MASK_RXE** (uint32_t)(0x10U)

#define **ST25R3911_IRQ_MASK_TXE** (uint32_t)(0x08U)

#define **ST25R3911_IRQ_MASK_COL** (uint32_t)(0x04U)

#define **ST25R3911_IRQ_MASK_DCT** (uint32_t)(0x8000U)

#define **ST25R3911_IRQ_MASK_NRE** (uint32_t)(0x4000U)

#define **ST25R3911_IRQ_MASK_GPE** (uint32_t)(0x2000U)

#define **ST25R3911_IRQ_MASK_EON** (uint32_t)(0x1000U)

#define **ST25R3911_IRQ_MASK_EOF** (uint32_t)(0x0800U)

#define **ST25R3911_IRQ_MASK_CAC** (uint32_t)(0x0400U)

#define **ST25R3911_IRQ_MASK_CAT** (uint32_t)(0x0200U)

#define **ST25R3911_IRQ_MASK_NFCT** (uint32_t)(0x0100U)

#define **ST25R3911_IRQ_MASK_CRC** (uint32_t)(0x800000U)

#define **ST25R3911_IRQ_MASK_PAR** (uint32_t)(0x400000U)

#define **ST25R3911_IRQ_MASK_ERR2** (uint32_t)(0x200000U)

#define **ST25R3911_IRQ_MASK_ERR1** (uint32_t)(0x100000U)

#define **ST25R3911_IRQ_MASK_WT** (uint32_t)(0x080000U)

#define **ST25R3911_IRQ_MASK_WAM** (uint32_t)(0x040000U)

#define **ST25R3911_IRQ_MASK_WPH** (uint32_t)(0x020000U)

#define **ST25R3911_IRQ_MASK_WCAP** (uint32_t)(0x010000U)

#define **ST25R3911_IRQ_MASK_TIM** (0x02U)

#define **ST25R3911_IRQ_MASK_ERR** (0x01U)

Functions

uint32_t **st25r3911WaitForInterruptsTimed** (uint32_t mask, uint16_t tmo)
Wait until an ST25R3911 interrupt occurs. More...

uint32_t **st25r3911GetInterrupt** (uint32_t mask)
Get status for the given interrupt. More...

void **st25r3911InitInterrupts** (void)
Init the 3911 interrupt. More...

void **st25r3911ModifyInterrupts** (uint32_t clr_mask, uint32_t set_mask)
Modifies the Interrupt. More...

void **st25r3911CheckForReceivedInterrupts** (void)
Checks received interrupts. More...

void **st25r3911Isr** (void)
ISR Service routine. More...

void **st25r3911EnableInterrupts** (uint32_t mask)
Enable a given ST25R3911 Interrupt source. More...

void **st25r3911DisableInterrupts** (uint32_t mask)
Disable one or more a given ST25R3911 Interrupt sources. More...

void **st25r3911ClearInterrupts** (void)
Clear all st25r3911 irq flags. More...

void **st25r3911IRQCallbackSet** (void(*cb)(void))
Sets IRQ callback for the ST25R3911 interrupt. More...

void **st25r3911IRQCallbackRestore** (void)

Sets IRQ callback for the ST25R3911 interrupt. More...

Detailed Description

Author

Ulrich Herrmann

st25r3916 Directory Reference

Directory dependency graph for st25r3916:

Files

file **st25r3916/rfal_dlmaTbl.h**
RF Dynamic LMA Table default values.

file **st25r3916/rfal_dpoTbl.h**
RF Dynamic Power Table default values.

file **st25r3916/rfal_features.h**
RFAL Features/Capabilities Definition for ST25R3916.

file **st25r3916.h**
ST25R3916 high level interface.

file **st25r3916_aat.h**
ST25R3916 Antenna Tuning.

file **st25r3916_com.h**
ST25R3916 communication declaration file.

file **st25r3916_irq.h**
ST25R3916 Interrupt handling.

file **st25r3916_led.h**
ST25R3916 LEDs handling.

st25r3916/rfal_dlmaTbl.h File Reference

RF Dynamic LMA Table default values. More...

```
#include "rfal_dlma.h"
```

Include dependency graph for st25r3916/rfal_dlmaTbl.h:

Variables

const **rfalDlmaEntry** **rfalDlmaDefaultSettings** []

Detailed Description

Author

Gustavo Patricio

Variable Documentation

◆ rfalDlmaDefaultSettings

const **rfalDlmaEntry** rfalDlmaDefaultSettings[]

Initial value:

```
= {
    {0x0F, 0x01,  39,   0},
    {0x0F, 0x03,  69,  40},
    {0x0F, 0x04,  79,  70},
    {0x0F, 0x05, 255,  80}
}
```

Default DLMA table

RFAL

st25r3916/rfal_dpoTbl.h File Reference

RF Dynamic Power Table default values. More...

```
#include "rfal_dpo.h"
```

Include dependency graph for st25r3916/rfal_dpoTbl.h:

Detailed Description

Author

Martin Zechleitner

st25r3916.h File Reference

ST25R3916 high level interface. More...

```
#include "rfal_platform.h"
#include "rfal_utils.h"
#include "st25r3916_com.h"
```

Include dependency graph for st25r3916.h:

Data Structures

struct **t_st25r3916Regs**

struct **st25r3916StreamConfig**

Macros

#define **ST25R3916_CMD_SET_DEFAULT** 0xC1U

#define **ST25R3916_CMD_STOP** 0xC2U

#define **ST25R3916_CMD_TRANSMIT_WITH_CRC** 0xC4U

#define **ST25R3916_CMD_TRANSMIT_WITHOUT_CRC** 0xC5U

#define **ST25R3916_CMD_TRANSMIT_REQA** 0xC6U

#define **ST25R3916_CMD_TRANSMIT_WUPA** 0xC7U

#define **ST25R3916_CMD_INITIAL_RF_COLLISION** 0xC8U

#define **ST25R3916_CMD_RESPONSE_RF_COLLISION_N** 0xC9U

#define **ST25R3916_CMD_GOTO_SENSE** 0xCDU

#define **ST25R3916_CMD_GOTO_SLEEP** 0xCEU

#define **ST25R3916_CMD_MASK_RECEIVE_DATA** 0xD0U

#define **ST25R3916_CMD_UNMASK_RECEIVE_DATA** 0xD1U

#define **ST25R3916_CMD_AM_MOD_STATE_CHANGE** 0xD2U

#define **ST25R3916_CMD_MEASURE_AMPLITUDE** 0xD3U

#define **ST25R3916_CMD_RESET_RXGAIN** 0xD5U

#define **ST25R3916_CMD_ADJUST_REGULATORS** 0xD6U

#define **ST25R3916_CMD_CALIBRATE_DRIVER_TIMING** 0xD8U

#define **ST25R3916_CMD_MEASURE_PHASE** 0xD9U

#define **ST25R3916_CMD_CLEAR_RSSI** 0xDAU

#define **ST25R3916_CMD_CLEAR_FIFO** 0xDBU

#define **ST25R3916_CMD_TRANSPARENT_MODE** 0xDCU

#define **ST25R3916_CMD_MEASURE_VDD** 0xDFU

```
#define ST25R3916_CMD_START_GP_TIMER 0xE0U
#define ST25R3916_CMD_START_WUP_TIMER 0xE1U
#define ST25R3916_CMD_START_MASK_RECEIVE_TIMER 0xE2U
#define ST25R3916_CMD_START_NO_RESPONSE_TIMER 0xE3U
#define ST25R3916_CMD_START_PPON2_TIMER 0xE4U
#define ST25R3916_CMD_STOP_NRT 0xE8U
#define ST25R3916_CMD_SPACE_B_ACCESS 0xFBU
#define ST25R3916_CMD_TEST_ACCESS 0xFCU
#define ST25R3916_THRESHOLD_DO_NOT_SET 0xFFU
#define ST25R3916_BR_DO_NOT_SET 0xFFU
#define ST25R3916_BR_106 0x00U
#define ST25R3916_BR_212 0x01U
#define ST25R3916_BR_424 0x02U
#define ST25R3916_BR_848 0x03U
#define ST25R3916_BR_1695 0x04U
#define ST25R3916_BR_3390 0x05U
#define ST25R3916_BR_6780 0x07U
#define ST25R3916_FIFO_DEPTH 512U
#define ST25R3916_TOUT_OSC_STABLE 10U
#define st25r3916TxRxOn() st25r3916SetRegisterBits( ST25R3916_REG_OP_CONTROL,
        (ST25R3916_REG_OP_CONTROL_rx_en | ST25R3916_REG_OP_CONTROL_tx_en ) )
```

#define **st25r3916TxRxOff**() **st25r3916ClrRegisterBits**(**ST25R3916_REG_OP_CONTROL**, (ST25R3916_REG_OP_CONTROL_rx_en | ST25R3916_REG_OP_CONTROL_tx_en))

#define **st25r3916TxOff**() **st25r3916ClrRegisterBits**(**ST25R3916_REG_OP_CONTROL**, ST25R3916_REG_OP_CONTROL_tx_en)

#define **st25r3916IsGPTRunning**() **st25r3916CheckReg**(**ST25R3916_REG_NFCIP1_BIT_RATE**, ST25R3916_REG_NFCIP1_BIT_RATE_gpt_on, ST25R3916_REG_NFCIP1_BIT_RATE_gpt_on)

#define **st25r3916IsExtFieldOn**() **st25r3916CheckReg**(**ST25R3916_REG_AUX_DISPLAY**, ST25R3916_REG_AUX_DISPLAY_efd_o, ST25R3916_REG_AUX_DISPLAY_efd_o)

#define **st25r3916IsTxEnabled**() **st25r3916CheckReg**(**ST25R3916_REG_OP_CONTROL**, ST25R3916_REG_OP_CONTROL_tx_en, ST25R3916_REG_OP_CONTROL_tx_en)

#define **st25r3916IsNRTinEMV**() **st25r3916CheckReg**(**ST25R3916_REG_TIMER_EMV_CONTROL**, ST25R3916_REG_TIMER_EMV_CONTROL_nrt_emv, ST25R3916_REG_TIMER_EMV_CONTROL_nrt_emv_on)

#define **st25r3916IsLastFIFOComplete**() **st25r3916CheckReg**(**ST25R3916_REG_FIFO_STATUS2**, ST25R3916_REG_FIFO_STATUS2_fifo_lb_mask, 0)

#define **st25r3916IsOscOn**() **st25r3916CheckReg**(**ST25R3916_REG_OP_CONTROL**, ST25R3916_REG_OP_CONTROL_en, ST25R3916_REG_OP_CONTROL_en)

#define **st25r3916IsAATOn**() **st25r3916CheckReg**(**ST25R3916_REG_IO_CONF2**, ST25R3916_REG_IO_CONF2_aat_en, ST25R3916_REG_IO_CONF2_aat_en)

Functions

ReturnCode **st25r3916Initialize** (void)

Initialise ST25R3916 driver. More...

void **st25r3916Deinitialize** (void)

Deinitialize ST25R3916 driver. More...

ReturnCode **st25r3916OscOn** (void)

Turn on Oscillator and Regulator. More...

ReturnCode **st25r3916SetBitrate** (uint8_t txrate, uint8_t rxrate)

Sets the bitrate. More...

ReturnCode	**st25r3916AdjustRegulators** (uint16_t *result_mV) Adjusts supply regulators according to the current supply voltage. More...
ReturnCode	**st25r3916SetRegulators** (uint8_t regulation) Sets supply regulators. More...
ReturnCode	**st25r3916MeasureAmplitude** (uint8_t *result) Measure Amplitude. More...
uint8_t	**st25r3916MeasurePowerSupply** (uint8_t mpsv) Measure Power Supply. More...
uint16_t	**st25r3916MeasureVoltage** (uint8_t mpsv) Measure Voltage. More...
ReturnCode	**st25r3916MeasurePhase** (uint8_t *result) Measure Phase. More...
ReturnCode	**st25r3916MeasureCapacitance** (uint8_t *result) Measure Capacitance. More...
ReturnCode	**st25r3916CalibrateCapacitiveSensor** (uint8_t *result) Calibrates Capacitive Sensor. More...
uint32_t	**st25r3916GetNoResponseTime** (void) Get NRT time. More...
ReturnCode	**st25r3916SetNoResponseTime** (uint32_t nrt_64fcs) Set NRT time. More...
ReturnCode	**st25r3916SetStartNoResponseTimer** (uint32_t nrt_64fcs) Set and Start NRT. More...
void	**st25r3916SetGPTime** (uint16_t gpt_8fcs) Set GPT time. More...
ReturnCode	**st25r3916SetStartGPTimer** (uint16_t gpt_8fcs, uint8_t trigger_source) Set and Start GPT. More...

void	**st25r3916SetNumTxBits** (uint16_t nBits) Sets the number Tx Bits. More...
uint16_t	**st25r3916GetNumFIFOBytes** (void) Get amount of bytes in FIFO. More...
uint8_t	**st25r3916GetNumFIFOLastBits** (void) Get amount of bits of the last FIFO byte if incomplete. More...
ReturnCode	**st25r3916PerformCollisionAvoidance** (uint8_t FieldONCmd, uint8_t pdThreshold, uint8_t caThreshold, uint8_t nTRFW) Perform Collision Avoidance. More...
bool	**st25r3916CheckChipID** (uint8_t *rev) Check Identity. More...
ReturnCode	**st25r3916GetRegsDump** (**t_st25r3916Regs** *regDump) Retrieves all internal registers from ST25R3916. More...
bool	**st25r3916IsCmdValid** (uint8_t cmd) Check if command is valid. More...
ReturnCode	**st25r3916StreamConfigure** (const struct **st25r3916StreamConfig** *config) Configure the stream mode of ST25R3916. More...
ReturnCode	**st25r3916ExecuteCommandAndGetResult** (uint8_t cmd, uint8_t resReg, uint8_t tOut, uint8_t *result) Executes a direct command and returns the result. More...
ReturnCode	**st25r3916GetRSSI** (uint16_t *amRssi, uint16_t *pmRssi) Gets the RSSI values. More...
ReturnCode	**st25r3916SetAntennaMode** (bool single, bool rfiox) Set Antenna mode. More...

Detailed Description

Author

Gustavo Patricio

st25r3916_aat.h File Reference

ST25R3916 Antenna Tuning. More...

```
#include "rfal_platform.h"
#include "rfal_utils.h"
```

Include dependency graph for st25r3916_aat.h:

Data Structures

struct	**st25r3916AatTuneParams**
struct	**st25r3916AatTuneResult**

Functions

ReturnCode	**st25r3916AatTune** (const struct **st25r3916AatTuneParams** *tuningParams, struct **st25r3916AatTuneResult** *tuningStatus) Perform antenna tuning. More...

Detailed Description

Author

The antenna tuning algorithm tries to find the optimal settings for the AAT_A and AAT_B registers, which are connected to variable capacitors to tune the antenna matching.

Function Documentation

◆ st25r3916AatTune()

```
ReturnCode st25r3916AatTune ( const struct st25r3916AatTuneParams *  tuningParams,
                              struct st25r3916AatTuneResult *        tuningStatus
                            )
```

This function starts an antenna tuning procedure by modifying the serial and parallel capacitors of the antenna matching circuit via the AAT_A and AAT_B registers. This function is best run if the field is already turned on. When used on ST25R3916B with new rgs_am=1 it is necessary to turn on the field before running this procedure or to set rgs_txonoff=0.

Parameters

[in] **tuningParams** : Input parameters for the tuning algorithm. If NULL default values will be used.

[out] **tuningStatus** : Result information of performed tuning. If NULL no further information is returned, only registers ST25R3916 (AAT_A,B) will be adapted.

Returns

RFAL_ERR_IO : Error during communication.

RFAL_ERR_PARAM : Invalid input parameters

RFAL_ERR_NONE : No error.

Generated on Fri Mar 14 2025 11:45:08 for RFAL by doxygen 1.9.5

RFAL

st25r3916_com.h File Reference

ST25R3916 communication declaration file. More...

```
#include "rfal_platform.h"
#include "rfal_utils.h"
```

Include dependency graph for st25r3916_com.h:

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

Macros

#define **ST25R3916_SPACE_B** 0x40U

#define **ST25R3916_SPACE_B_REG_LEN** 16U

#define **ST25R3916_FIFO_STATUS_LEN** 2

#define **ST25R3916_PTM_A_LEN** 15U

#define **ST25R3916_PTM_B_LEN** 0U

#define **ST25R3916_PTM_F_LEN** 21U

#define **ST25R3916_PTM_TSN_LEN** 12U

#define **ST25R3916_PTM_LEN** (**ST25R3916_PTM_A_LEN** + **ST25R3916_PTM_B_LEN** + **ST25R3916_PTM_F_LEN** + **ST25R3916_PTM_TSN_LEN**)

#define **ST25R3916_REG_IO_CONF1** 0x00U

#define **ST25R3916_REG_IO_CONF2** 0x01U

#define **ST25R3916_REG_OP_CONTROL** 0x02U

#define **ST25R3916_REG_MODE** 0x03U

#define **ST25R3916_REG_BIT_RATE** 0x04U

#define **ST25R3916_REG_ISO14443A_NFC** 0x05U

#define **ST25R3916_REG_EMD_SUP_CONF** (**ST25R3916_SPACE_B**|0x05U)

#define **ST25R3916_REG_ISO14443B_1** 0x06U

#define **ST25R3916_REG_SUBC_START_TIME** (**ST25R3916_SPACE_B**|0x06U)

#define **ST25R3916_REG_ISO14443B_2** 0x07U

#define **ST25R3916_REG_PASSIVE_TARGET** 0x08U

#define **ST25R3916_REG_STREAM_MODE** 0x09U

#define **ST25R3916_REG_AUX** 0x0AU

#define **ST25R3916_REG_RX_CONF1** 0x0BU

#define **ST25R3916_REG_RX_CONF2** 0x0CU

#define **ST25R3916_REG_RX_CONF3** 0x0DU

#define **ST25R3916_REG_RX_CONF4** 0x0EU

#define **ST25R3916_REG_P2P_RX_CONF** (**ST25R3916_SPACE_B**|0x0BU)

#define **ST25R3916_REG_CORR_CONF1** (**ST25R3916_SPACE_B**|0x0CU)

#define **ST25R3916_REG_CORR_CONF2** (**ST25R3916_SPACE_B**|0x0DU)

#define **ST25R3916_REG_MASK_RX_TIMER** 0x0FU

#define **ST25R3916_REG_NO_RESPONSE_TIMER1** 0x10U

#define **ST25R3916_REG_NO_RESPONSE_TIMER2** 0x11U

#define **ST25R3916_REG_TIMER_EMV_CONTROL** 0x12U

#define **ST25R3916_REG_GPT1** 0x13U

```
#define ST25R3916_REG_GPT2 0x14U
#define ST25R3916_REG_PPON2 0x15U
#define ST25R3916_REG_SQUELCH_TIMER (ST25R3916_SPACE_B|0x0FU)
#define ST25R3916_REG_FIELD_ON_GT (ST25R3916_SPACE_B|0x15U)
#define ST25R3916_REG_IRQ_MASK_MAIN 0x16U
#define ST25R3916_REG_IRQ_MASK_TIMER_NFC 0x17U
#define ST25R3916_REG_IRQ_MASK_ERROR_WUP 0x18U
#define ST25R3916_REG_IRQ_MASK_TARGET 0x19U
#define ST25R3916_REG_IRQ_MAIN 0x1AU
#define ST25R3916_REG_IRQ_TIMER_NFC 0x1BU
#define ST25R3916_REG_IRQ_ERROR_WUP 0x1CU
#define ST25R3916_REG_IRQ_TARGET 0x1DU
#define ST25R3916_REG_FIFO_STATUS1 0x1EU
#define ST25R3916_REG_FIFO_STATUS2 0x1FU
#define ST25R3916_REG_COLLISION_STATUS 0x20U
#define ST25R3916_REG_PASSIVE_TARGET_STATUS 0x21U
#define ST25R3916_REG_NUM_TX_BYTES1 0x22U
#define ST25R3916_REG_NUM_TX_BYTES2 0x23U
#define ST25R3916_REG_NFCIP1_BIT_RATE 0x24U
#define ST25R3916_REG_AD_RESULT 0x25U
#define ST25R3916_REG_ANT_TUNE_A 0x26U
```

```
#define ST25R3916_REG_ANT_TUNE_B 0x27U
#define ST25R3916_REG_TX_DRIVER 0x28U
#define ST25R3916_REG_PT_MOD 0x29U
#define ST25R3916_REG_AUX_MOD (ST25R3916_SPACE_B|0x28U)
#define ST25R3916_REG_TX_DRIVER_TIMING (ST25R3916_SPACE_B|0x29U)
#define ST25R3916_REG_RES_AM_MOD (ST25R3916_SPACE_B|0x2AU)
#define ST25R3916_REG_TX_DRIVER_STATUS (ST25R3916_SPACE_B|0x2BU)
#define ST25R3916_REG_FIELD_THRESHOLD_ACTV 0x2AU
#define ST25R3916_REG_FIELD_THRESHOLD_DEACTV 0x2BU
#define ST25R3916_REG_REGULATOR_CONTROL 0x2CU
#define ST25R3916_REG_REGULATOR_RESULT (ST25R3916_SPACE_B|0x2CU)
#define ST25R3916_REG_RSSI_RESULT 0x2DU
#define ST25R3916_REG_GAIN_RED_STATE 0x2EU
#define ST25R3916_REG_CAP_SENSOR_CONTROL 0x2FU
#define ST25R3916_REG_CAP_SENSOR_RESULT 0x30U
#define ST25R3916_REG_AUX_DISPLAY 0x31U
#define ST25R3916_REG_OVERSHOOT_CONF1 (ST25R3916_SPACE_B|0x30U)
#define ST25R3916_REG_OVERSHOOT_CONF2 (ST25R3916_SPACE_B|0x31U)
#define ST25R3916_REG_UNDERSHOOT_CONF1 (ST25R3916_SPACE_B|0x32U)
#define ST25R3916_REG_UNDERSHOOT_CONF2 (ST25R3916_SPACE_B|0x33U)
#define ST25R3916_REG_WUP_TIMER_CONTROL 0x32U
```

#define **ST25R3916_REG_AMPLITUDE_MEASURE_CONF** 0x33U

#define **ST25R3916_REG_AMPLITUDE_MEASURE_REF** 0x34U

#define **ST25R3916_REG_AMPLITUDE_MEASURE_AA_RESULT** 0x35U

#define **ST25R3916_REG_AMPLITUDE_MEASURE_RESULT** 0x36U

#define **ST25R3916_REG_PHASE_MEASURE_CONF** 0x37U

#define **ST25R3916_REG_PHASE_MEASURE_REF** 0x38U

#define **ST25R3916_REG_PHASE_MEASURE_AA_RESULT** 0x39U

#define **ST25R3916_REG_PHASE_MEASURE_RESULT** 0x3AU

#define **ST25R3916_REG_IC_IDENTITY** 0x3FU

Functions

ReturnCode **st25r3916ReadRegister** (uint8_t reg, uint8_t *val)

Returns the content of a register within the ST25R3916. More...

ReturnCode **st25r3916ReadMultipleRegisters** (uint8_t reg, uint8_t *values, uint8_t length)

Reads from multiple ST25R3916 registers. More...

ReturnCode **st25r3916WriteRegister** (uint8_t reg, uint8_t val)

Writes a given value to a register within the ST25R3916. More...

ReturnCode **st25r3916WriteMultipleRegisters** (uint8_t reg, const uint8_t *values, uint8_t length)

Writes multiple values to ST25R3916 registers. More...

ReturnCode **st25r3916WriteFifo** (const uint8_t *values, uint16_t length)

Writes values to ST25R3916 FIFO. More...

ReturnCode **st25r3916ReadFifo** (uint8_t *buf, uint16_t length)

Read values from ST25R3916 FIFO. More...

ReturnCode **st25r3916WritePTMem** (const uint8_t *values, uint16_t length)

Writes values to ST25R3916 PTM. More...

ReturnCode	**st25r3916ReadPTMem** (uint8_t *values, uint16_t length) Reads the ST25R3916 PTM. More...
ReturnCode	**st25r3916WritePTMemF** (const uint8_t *values, uint16_t length) Writes values to ST25R3916 PTM F config. More...
ReturnCode	**st25r3916WritePTMemTSN** (const uint8_t *values, uint16_t length) Writes values to ST25R3916 PTM TSN Data. More...
ReturnCode	**st25r3916ExecuteCommand** (uint8_t cmd) Execute a direct command. More...
ReturnCode	**st25r3916ExecuteCommands** (const uint8_t *cmds, uint8_t length) Execute direct commands. More...
ReturnCode	**st25r3916ReadTestRegister** (uint8_t reg, uint8_t *val) Read a test register within the ST25R3916. More...
ReturnCode	**st25r3916WriteTestRegister** (uint8_t reg, uint8_t val) Writes a given value to a test register within the ST25R3916. More...
ReturnCode	**st25r3916ClrRegisterBits** (uint8_t reg, uint8_t clr_mask) Cleart bits on Register. More...
ReturnCode	**st25r3916SetRegisterBits** (uint8_t reg, uint8_t set_mask) Set bits on Register. More...
ReturnCode	**st25r3916ChangeRegisterBits** (uint8_t reg, uint8_t valueMask, uint8_t value) Changes the given bits on a ST25R3916 register. More...
ReturnCode	**st25r3916ModifyRegister** (uint8_t reg, uint8_t clr_mask, uint8_t set_mask) Modifies a value within a ST25R3916 register. More...
ReturnCode	**st25r3916ChangeTestRegisterBits** (uint8_t reg, uint8_t valueMask, uint8_t value) Changes the given bits on a ST25R3916 Test register. More...
bool	**st25r3916CheckReg** (uint8_t reg, uint8_t mask, uint8_t val) Checks if register contains a expected value. More...

bool	**st25r3916IsRegValid** (uint8_t reg) Check if register ID is valid. More...

Detailed Description

Author

Gustavo Patricio

This driver provides basic abstraction for communication with the ST25R3916

st25r3916_irq.h File Reference

ST25R3916 Interrupt handling. More...

```
#include "rfal_platform.h"
```

Include dependency graph for st25r3916_irq.h:

Macros

#define **ST25R3916_IRQ_MASK_ALL** (uint32_t)(0xFFFFFFFFUL)

#define **ST25R3916_IRQ_MASK_NONE** (uint32_t)(0x00000000UL)

#define **ST25R3916_IRQ_MASK_OSC** (uint32_t)(0x00000080U)

#define **ST25R3916_IRQ_MASK_FWL** (uint32_t)(0x00000040U)

#define **ST25R3916_IRQ_MASK_RXS** (uint32_t)(0x00000020U)

#define **ST25R3916_IRQ_MASK_RXE** (uint32_t)(0x00000010U)

#define **ST25R3916_IRQ_MASK_TXE** (uint32_t)(0x00000008U)

#define **ST25R3916_IRQ_MASK_COL** (uint32_t)(0x00000004U)

#define **ST25R3916_IRQ_MASK_RX_REST** (uint32_t)(0x00000002U)

#define **ST25R3916_IRQ_MASK_RFU** (uint32_t)(0x00000001U)

#define **ST25R3916_IRQ_MASK_DCT** (uint32_t)(0x00008000U)

#define **ST25R3916_IRQ_MASK_NRE** (uint32_t)(0x00004000U)

#define **ST25R3916_IRQ_MASK_GPE** (uint32_t)(0x00002000U)

#define **ST25R3916_IRQ_MASK_EON** (uint32_t)(0x00001000U)

#define **ST25R3916_IRQ_MASK_EOF** (uint32_t)(0x00000800U)

#define **ST25R3916_IRQ_MASK_CAC** (uint32_t)(0x00000400U)

#define **ST25R3916_IRQ_MASK_CAT** (uint32_t)(0x00000200U)

#define **ST25R3916_IRQ_MASK_NFCT** (uint32_t)(0x00000100U)

#define **ST25R3916_IRQ_MASK_CRC** (uint32_t)(0x00800000U)

#define **ST25R3916_IRQ_MASK_PAR** (uint32_t)(0x00400000U)

#define **ST25R3916_IRQ_MASK_ERR2** (uint32_t)(0x00200000U)

#define **ST25R3916_IRQ_MASK_ERR1** (uint32_t)(0x00100000U)

#define **ST25R3916_IRQ_MASK_WT** (uint32_t)(0x00080000U)

#define **ST25R3916_IRQ_MASK_WAM** (uint32_t)(0x00040000U)

#define **ST25R3916_IRQ_MASK_WPH** (uint32_t)(0x00020000U)

#define **ST25R3916_IRQ_MASK_PPON2** (uint32_t)(0x80000000U)

#define **ST25R3916_IRQ_MASK_SL_WL** (uint32_t)(0x40000000U)

#define **ST25R3916_IRQ_MASK_APON** (uint32_t)(0x20000000U)

#define **ST25R3916_IRQ_MASK_RXE_PTA** (uint32_t)(0x10000000U)

#define **ST25R3916_IRQ_MASK_WU_F** (uint32_t)(0x08000000U)

#define **ST25R3916_IRQ_MASK_RFU2** (uint32_t)(0x04000000U)

#define **ST25R3916_IRQ_MASK_WU_A_X** (uint32_t)(0x02000000U)

#define **ST25R3916_IRQ_MASK_WU_A** (uint32_t)(0x01000000U)

Functions

uint32_t **st25r3916WaitForInterruptsTimed** (uint32_t mask, uint16_t tmo)

Wait until an ST25R3916 interrupt occurs. More...

uint32_t	**st25r3916GetInterrupt** (uint32_t mask) Get status for the given interrupt. More...
void	**st25r3916InitInterrupts** (void) Init the 3916 interrupt. More...
void	**st25r3916ModifyInterrupts** (uint32_t clr_mask, uint32_t set_mask) Modifies the Interrupt. More...
void	**st25r3916CheckForReceivedInterrupts** (void) Checks received interrupts. More...
void	**st25r3916Isr** (void) ISR Service routine. More...
void	**st25r3916EnableInterrupts** (uint32_t mask) Enable a given ST25R3916 Interrupt source. More...
void	**st25r3916DisableInterrupts** (uint32_t mask) Disable one or more a given ST25R3916 Interrupt sources. More...
void	**st25r3916ClearInterrupts** (void) Clear all ST25R3916 irq flags. More...
void	**st25r3916ClearAndEnableInterrupts** (uint32_t mask) Clears and then enables the given ST25R3916 Interrupt sources. More...
void	**st25r3916IRQCallbackSet** (void(*cb)(void)) Sets IRQ callback for the ST25R3916 interrupt. More...
void	**st25r3916IRQCallbackRestore** (void) Sets IRQ callback for the ST25R3916 interrupt. More...

Detailed Description

Author

Gustavo Patricio

st25r3916_led.h File Reference

ST25R3916 LEDs handling. More...

```
#include "rfal_platform.h"
```

Include dependency graph for st25r3916_led.h:

Functions

void **st25r3916ledInit** (void)
ST25R3916 LED Initialize. More...

void **st25r3916ledEvtIrq** (uint32_t irqs)
ST25R3916 LED Event Interrupt. More...

void **st25r3916ledEvtWrReg** (uint8_t reg, uint8_t val)
ST25R3916 LED Event Write Register. More...

void **st25r3916ledEvtWrMultiReg** (uint8_t reg, const uint8_t *vals, uint8_t len)
ST25R3916 LED Event Write Multiple Register. More...

void **st25r3916ledEvtCmd** (uint8_t cmd)
ST25R3916 LED Event Direct Command. More...

void **st25r3916ledEvtCmds** (const uint8_t *cmds, uint8_t len)
ST25R3916 LED Event Direct Commands. More...

Detailed Description

Author

Gustavo Patricio

st25r500 Directory Reference

Directory dependency graph for st25r500:

Files

file **st25r500/rfal_dlmaTbl.h**
RF Dynamic LMA Table default values.

file **st25r500/rfal_features.h**
RFAL Features/Capabilities Definition for ST25R500.

file **st25r500.h**
ST25R500 high level interface.

file **st25r500_com.h**
ST25R500 communication declaration file.

file **st25r500_dpocr.h**
Dynamic Power Output adjustment via Current Regulation.

file **st25r500_irq.h**
ST25R500 Interrupt handling.

RFAL

Main Page

Modules

Data Structures

Examples

File List

Globals

source

st25r500

Variables

st25r500/rfal_dlmaTbl.h File Reference

RF Dynamic LMA Table default values. More...

```
#include "rfal_dlma.h"
```

Include dependency graph for st25r500/rfal_dlmaTbl.h:

Variables

const **rfalDlmaEntry** **rfalDlmaDefaultSettings** []

Detailed Description

Author

Variable Documentation

◆ rfalDlmaDefaultSettings

const **rfalDlmaEntry** rfalDlmaDefaultSettings[]

Initial value:

```
= {
    {0x05, 0x0F,  3,  0},
    {0x05, 0x0D,  7,  3},
    {0x05, 0x0B, 11,  7},
    {0x05, 0x09, 15, 11}
}
```

Default DLMA table

RFAL

Main Page

Modules

Data Structures

Examples

File List

st25r500.h File Reference

ST25R500 high level interface. More...

```
#include "rfal_platform.h"
#include "rfal_utils.h"
#include "st25r500_com.h"
```

Include dependency graph for st25r500.h:

Macros

#define **ST25R500_CMD_SET_DEFAULT** 0x60U

#define **ST25R500_CMD_STOP** 0x62U

#define **ST25R500_CMD_CLEAR_FIFO** 0x64U

#define **ST25R500_CMD_CLEAR_RXGAIN** 0x66U

#define **ST25R500_CMD_ADJUST_REGULATORS** 0x68U

#define **ST25R500_CMD_TRANSMIT** 0x6AU

#define **ST25R500_CMD_TRANSMIT_EOF** 0x6CU

```
#define ST25R500_CMD_NFC_FIELD_ON 0x6EU
#define ST25R500_CMD_MASK_RECEIVE_DATA 0x70U
#define ST25R500_CMD_UNMASK_RECEIVE_DATA 0x72U
#define ST25R500_CMD_CALIBRATE_WU 0x74U
#define ST25R500_CMD_CLEAR_WU_CALIB 0x76U
#define ST25R500_CMD_MEASURE_WU 0x78U
#define ST25R500_CMD_MEASURE_IQ 0x7AU
#define ST25R500_CMD_SENSE_RF 0x7CU
#define ST25R500_CMD_TRIGGER_WU_EV 0x7EU
#define ST25R500_CMD_START_GP_TIMER 0xE2U
#define ST25R500_CMD_START_WUT 0xE4U
#define ST25R500_CMD_START_MRT 0xE6U
#define ST25R500_CMD_START_NRT 0xE8U
#define ST25R500_CMD_STOP_NRT 0xEAU
#define ST25R500_CMD_CALIBRATE_RC 0xEEU
#define ST25R500_CMD_TRIGGER_DIAG 0xF8U
#define ST25R500_CMD_TEST_ACCESS 0xFCU
#define ST25R500_BR_DO_NOT_SET 0xFFU
#define ST25R500_BR_106_26 0x00U
#define ST25R500_BR_212_53 0x01U
#define ST25R500_BR_424 0x02U
```

```
#define ST25R500_BR_848 0x03U
#define ST25R500_REG_DROP_200 0U
#define ST25R500_REG_DROP_250 1U
#define ST25R500_REG_DROP_300 2U
#define ST25R500_REG_DROP_350 3U
#define ST25R500_REG_DROP_400 4U
#define ST25R500_REG_DROP_450 5U
#define ST25R500_REG_DROP_500 6U
#define ST25R500_REG_DROP_550 7U
#define ST25R500_REG_DROP_DO_NOT_SET 0xFFU
#define ST25R500_THRESHOLD_DO_NOT_SET 0xFFU
#define ST25R500_REG_LEN 1U
#define ST25R500_CMD_LEN 1U
#define ST25R500_FIFO_DEPTH 256U
#define ST25R500_TOUT_OSC_STABLE 5U
#define ST25R500_WRITE_MODE (0U << 7)
#define ST25R500_READ_MODE (1U << 7)
#define ST25R500_CMD_MODE ST25R500_WRITE_MODE
#define ST25R500_FIFO_ACCESS (0x5FU)
#define ST25R500_DIAG_MEAS_CMD 0x01U
#define ST25R500_DIAG_MEAS_CMD_LEN 0x02U
```

```
#define ST25R500_DIAG_MEAS_RES_LEN 0x04U
#define ST25R500_DIAG_MEAS_I_VDD_DR 0x01U
#define ST25R500_DIAG_MEAS_VDD_TX 0x02U
#define ST25R500_DIAG_MEAS_VDD_DR 0x03U
#define ST25R500_DIAG_MEAS_VDD_IO 0x04U
#define ST25R500_DIAG_MEAS_VDD_D 0x0CU
#define ST25R500_DIAG_MEAS_VDD_A 0x0DU
#define ST25R500_DIAG_MEAS_VDD_VDD 0x11U
#define ST25R500_DIAG_MEAS_VDD_AGD 0x12U
#define st25r500TxRxOn() st25r500SetRegisterBits( ST25R500_REG_OPERATION,
        (ST25R500_REG_OPERATION_rx_en | ST25R500_REG_OPERATION_tx_en ) )
#define st25r500TxRxOff() st25r500ClrRegisterBits( ST25R500_REG_OPERATION,
        (ST25R500_REG_OPERATION_rx_en | ST25R500_REG_OPERATION_tx_en ) )
#define st25r500VDDDROn() st25r500SetRegisterBits( ST25R500_REG_OPERATION,
        (ST25R500_REG_OPERATION_vdddr_en | ST25R500_REG_OPERATION_vdddr_en ) )
#define st25r500VDDDROff() st25r500ClrRegisterBits( ST25R500_REG_OPERATION,
        (ST25R500_REG_OPERATION_vdddr_en | ST25R500_REG_OPERATION_vdddr_en ) )
#define st25r500TxOff() st25r500ClrRegisterBits( ST25R500_REG_OPERATION, ST25R500_REG_OPERATION_tx_en
        )
#define st25r500IsGPTRunning() st25r500CheckReg( ST25R500_REG_STATUS2, ST25R500_REG_STATUS2_gpt_on,
        ST25R500_REG_STATUS2_gpt_on )
#define st25r500IsExtFieldOn() st25r500CheckReg( ST25R500_REG_STATUS1, ST25R500_REG_STATUS1_efd_out,
        ST25R500_REG_STATUS1_efd_out )
#define st25r500IsTxEnabled() st25r500CheckReg( ST25R500_REG_OPERATION,
        ST25R500_REG_OPERATION_tx_en, ST25R500_REG_OPERATION_tx_en )
```

#define **st25r500IsNRTinEMV**() **st25r500CheckReg**(**ST25R500_REG_NRT_GPT_CONF**, ST25R500_REG_NRT_GPT_CONF_nrt_emv, ST25R500_REG_NRT_GPT_CONF_nrt_emv_on)

#define **st25r500IsLastFIFOComplete**() **st25r500CheckReg**(**ST25R500_REG_FIFO_STATUS2**, ST25R500_REG_FIFO_STATUS2_fifo_lb_mask, 0)

#define **st25r500IsOscOn**() **st25r500CheckReg**(**ST25R500_REG_OPERATION**, ST25R500_REG_OPERATION_en, ST25R500_REG_OPERATION_en)

#define **st25r500IsTxOn**() **st25r500CheckReg**(**ST25R500_REG_OPERATION**, ST25R500_REG_OPERATION_tx_en, ST25R500_REG_OPERATION_tx_en)

Functions

ReturnCode **st25r500Initialize** (void)
Initialise ST25R500 driver. More...

void **st25r500Deinitialize** (void)
Deinitialize ST25R500 driver. More...

ReturnCode **st25r500OscOn** (void)
Turn on Oscillator and Regulator. More...

ReturnCode **st25r500SetBitrate** (uint8_t txRate, uint8_t rxRate)
Sets the bitrate. More...

ReturnCode **st25r500AdjustRegulators** (uint8_t drop, uint16_t *result)
Adjusts supply regulators according to the current supply voltage. More...

ReturnCode **st25r500SetRegulators** (uint8_t regulation)
Sets supply regulators. More...

ReturnCode **st25r500MeasureAmplitude** (uint8_t *result)
Measure Amplitude. More...

ReturnCode **st25r500GetCeGain** (uint8_t *result)
Get Ce gain. More...

ReturnCode **st25r500MeasureIQ** (int8_t *resI, int8_t *resQ)

Measure I/Q. More...

ReturnCode	**st25r500MeasureCombinedIQ** (uint8_t *res) Measure Combined I/Q. More...
ReturnCode	**st25r500MeasureI** (uint8_t *res) Measure I. More...
ReturnCode	**st25r500MeasureQ** (uint8_t *res) Measure Q. More...
ReturnCode	**st25r500MeasurePhase** (uint8_t *res) Measure Phase. More...
ReturnCode	**st25r500MeasureCurrent** (uint8_t *res) Measure electrical current. More...
ReturnCode	**st25r500CalibrateWU** (uint16_t *resI, uint16_t *resQ) Calibrate WU. More...
ReturnCode	**st25r500ClearCalibration** (void) Clear WU Calibration. More...
ReturnCode	**st25r500MeasureWU** (uint8_t *resI, uint8_t *resQ) Measure WU. More...
uint32_t	**st25r500GetNoResponseTime** (void) Get NRT time. More...
ReturnCode	**st25r500SetNoResponseTime** (uint32_t nrt) Set NRT time. More...
ReturnCode	**st25r500SetMRT** (uint32_t value_fc) Set MRT time. More...
ReturnCode	**st25r500SetStartNoResponseTimer** (uint32_t nrt) Set and Start NRT. More...
void	**st25r500SetGPTime** (uint16_t gpt)

Set GPT time. More...

ReturnCode	**st25r500SetStartGPTimer** (uint16_t gpt, uint8_t trigger) Set and Start GPT. More...
void	**st25r500SetNumTxBits** (uint16_t nBits) Sets the number Tx Bits. More...
uint16_t	**st25r500GetNumFIFOBytes** (void) Get amount of bytes in FIFO. More...
uint8_t	**st25r500GetNumFIFOLastBits** (void) Get amount of bits of the last FIFO byte if incomplete. More...
ReturnCode	**st25r500PerformCollisionAvoidance** (uint8_t atThreshold, uint8_t dtThreshold, uint8_t nTRFW) Perform Collision Avoidance. More...
bool	**st25r500CheckChipID** (uint8_t *rev) Check Identity. More...
ReturnCode	**st25r500GetRegsDump** (uint8_t *resRegDump, uint8_t *sizeRegDump) Register Dump. More...
bool	**st25r500IsCmdValid** (uint8_t cmd) Check if command is valid. More...
ReturnCode	**st25r500ExecuteCommandAndGetResult** (uint8_t cmd, uint8_t resReg, uint8_t tOut, uint8_t *result) Executes a direct command and returns the result. More...
ReturnCode	**st25r500GetRSSI** (uint16_t *iRssi, uint16_t *qRssi) Gets the RSSI values. More...
ReturnCode	**st25r500SetAntennaMode** (bool single, bool rfiox) Set Antenna mode. More...
ReturnCode	**st25r500DiagMeasure** (uint8_t meas, uint16_t *res) Diagnostic Measurement. More...

Detailed Description

Author

Gustavo Patricio

st25r500_com.h File Reference

ST25R500 communication declaration file. More...

```
#include "rfal_platform.h"
#include "rfal_utils.h"
```

Include dependency graph for st25r500_com.h:

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

Macros

#define **ST25R500_FIFO_STATUS_LEN** 2

#define **ST25R500_OBS_MODE_LEN** 2

#define **ST25R500_CEM_A_LEN** 31U

#define **ST25R500_CEM_B_LEN** 0U

#define **ST25R500_CEM_F_LEN** 21U

#define **ST25R500_CEM_LEN** (**ST25R500_CEM_A_LEN** + **ST25R500_CEM_B_LEN** + **ST25R500_CEM_F_LEN**)

#define **ST25R500_REG_OPERATION** 0x00U

#define **ST25R500_REG_GENERAL** 0x01U

#define **ST25R500_REG_REGULATOR** 0x02U

#define **ST25R500_REG_DRIVER** 0x03U

#define **ST25R500_REG_TX_MOD1** 0x04U

#define **ST25R500_REG_TX_MOD2** 0x05U

```
#define ST25R500_REG_CE_TX_MOD1 0x06U
#define ST25R500_REG_CE_TX_MOD2 0x07U
#define ST25R500_REG_GPIO 0x08U
#define ST25R500_REG_RX_ANA1 0x09U
#define ST25R500_REG_RX_ANA2 0x0AU
#define ST25R500_REG_RX_ANA3 0x0BU
#define ST25R500_REG_RX_ANA4 0x0CU
#define ST25R500_REG_RX_DIG 0x0DU
#define ST25R500_REG_CORR1 0x0EU
#define ST25R500_REG_CORR2 0x0FU
#define ST25R500_REG_CORR3 0x10U
#define ST25R500_REG_CORR4 0x11U
#define ST25R500_REG_CORR5 0x12U
#define ST25R500_REG_CORR6 0x13U
#define ST25R500_REG_PROTOCOL 0x14U
#define ST25R500_REG_PROTOCOL_TX1 0x15U
#define ST25R500_REG_PROTOCOL_TX2 0x16U
#define ST25R500_REG_PROTOCOL_RX1 0x17U
#define ST25R500_REG_PROTOCOL_RX2 0x18U
#define ST25R500_REG_PROTOCOL_RX3 0x19U
#define ST25R500_REG_EMD1 0x1AU
```

```
#define ST25R500_REG_EMD2 0x1BU
#define ST25R500_REG_CE_CONFIG1 0x1CU
#define ST25R500_REG_CE_CONFIG2 0x1DU
#define ST25R500_REG_CE_CONFIG3 0x1EU
#define ST25R500_REG_MRT1 0x1FU
#define ST25R500_REG_MRT2 0x20U
#define ST25R500_REG_SQT 0x21U
#define ST25R500_REG_NRT_GPT_CONF 0x22U
#define ST25R500_REG_NRT1 0x23U
#define ST25R500_REG_NRT2 0x24U
#define ST25R500_REG_GPT1 0x25U
#define ST25R500_REG_GPT2 0x26U
#define ST25R500_REG_WAKEUP_CONF1 0x27U
#define ST25R500_REG_WAKEUP_CONF2 0x28U
#define ST25R500_REG_WAKEUP_CONF3 0x29U
#define ST25R500_REG_WU_I_CONF 0x2AU
#define ST25R500_REG_WU_I_DELTA 0x2BU
#define ST25R500_REG_WU_I_CAL 0x2CU
#define ST25R500_REG_WU_I_ADC 0x2DU
#define ST25R500_REG_WU_I_REF 0x2EU
#define ST25R500_REG_WU_Q_CONF 0x2FU
```

```
#define ST25R500_REG_WU_Q_DELTA 0x30U
#define ST25R500_REG_WU_Q_CAL 0x31U
#define ST25R500_REG_WU_Q_ADC 0x32U
#define ST25R500_REG_WU_Q_REF 0x33U
#define ST25R500_REG_TX_FRAME1 0x34U
#define ST25R500_REG_TX_FRAME2 0x35U
#define ST25R500_REG_FIFO_STATUS1 0x36U
#define ST25R500_REG_FIFO_STATUS2 0x37U
#define ST25R500_REG_COLLISION 0x38U
#define ST25R500_REG_IRQ_MASK1 0x39U
#define ST25R500_REG_IRQ_MASK2 0x3AU
#define ST25R500_REG_IRQ_MASK3 0x3BU
#define ST25R500_REG_IRQ1 0x3CU
#define ST25R500_REG_IRQ2 0x3DU
#define ST25R500_REG_IRQ3 0x3EU
#define ST25R500_REG_IC_ID 0x3FU
#define ST25R500_REG_STATUS1 0x40U
#define ST25R500_REG_STATUS2 0x41U
#define ST25R500_REG_STATUS_STATIC1 0x42U
#define ST25R500_REG_STATUS_STATIC2 0x43U
#define ST25R500_REG_STATUS_STATIC3 0x44U
```

```
#define ST25R500_REG_CE_STATUS1 0x45U
#define ST25R500_REG_CE_STATUS2 0x46U
#define ST25R500_REG_WU_STATUS 0x47U
#define ST25R500_REG_ANA_DISPLAY1 0x48U
#define ST25R500_REG_ANA_DISPLAY2 0x49U
#define ST25R500_REG_RSSI_I 0x4AU
#define ST25R500_REG_RSSI_Q 0x4BU
#define ST25R500_REG_SENSE_DISPLAY 0x4CU
#define ST25R500_REG_AWS_CONFIG1 0x4DU
#define ST25R500_REG_AWS_CONFIG2 0x4EU
#define ST25R500_REG_AWS_TIME1 0x4FU
#define ST25R500_REG_AWS_TIME2 0x50U
#define ST25R500_REG_AWS_TIME3 0x51U
#define ST25R500_REG_AWS_TIME4 0x52U
#define ST25R500_REG_OVERSHOOT_CONF 0x53U
#define ST25R500_REG_UNDERSHOOT_CONF 0x54U
#define ST25R500_REG_EFD_THRESHOLD 0x57U
#define ST25R500_REG_CEM_A 0x58U
#define ST25R500_REG_CEM_F 0x5AU
#define ST25R500_REG_FIFO 0x5FU
#define ST25R500_TEST_REG_ADC2CE 0x08U
```

#define **ST25R500_TEST_REG_OSC_TIMING** 0x0EU

#define **ST25R500_TEST_REG_MAN_TIMING** 0x13U

#define **ST25R500_TEST_REG_DIAG_MEAS** 0x14U

#define **ST25R500_TEST_REG_OVERLAP_CONTROL** 0x1EU

#define **ST25R500_TEST_REG_RXS_DISP1** 0x30U

#define **ST25R500_TEST_REG_RXS_DISP2** 0x31U

#define **ST25R500_TEST_REG_RXE_DISP1** 0x32U

#define **ST25R500_TEST_REG_RXE_DISP2** 0x33U

Functions

ReturnCode **st25r500ReadRegister** (uint8_t reg, uint8_t *val)

Returns the content of a register within the ST25R500. More...

ReturnCode **st25r500ReadMultipleRegisters** (uint8_t reg, uint8_t *values, uint16_t length)

Reads from multiple ST25R500 registers. More...

ReturnCode **st25r500WriteRegister** (uint8_t reg, uint8_t val)

Writes a given value to a register within the ST25R500. More...

ReturnCode **st25r500WriteMultipleRegisters** (uint8_t reg, const uint8_t *values, uint16_t length)

Writes multiple values to ST25R500 registers. More...

ReturnCode **st25r500WriteFifo** (const uint8_t *values, uint16_t length)

Writes values to ST25R500 FIFO. More...

ReturnCode **st25r500ReadFifo** (uint8_t *buf, uint16_t length)

Read values from ST25R500 FIFO. More...

ReturnCode **st25r500ExecuteCommand** (uint8_t cmd)

Execute a direct command. More...

ReturnCode **st25r500ExecuteCommands** (const uint8_t *cmds, uint8_t length)

Execute direct commands. More...

ReturnCode	**st25r500ReadTestRegister** (uint8_t reg, uint8_t *val) Read a test register within the ST25R500. More...
ReturnCode	**st25r500WriteTestRegister** (uint8_t reg, uint8_t val) Writes a given value to a test register within the ST25R500. More...
ReturnCode	**st25r500WriteMultipleTestRegister** (uint8_t reg, const uint8_t *values, uint8_t length) Writes a multiple values to test registers within the ST25R500. More...
ReturnCode	**st25r500ClrRegisterBits** (uint8_t reg, uint8_t clr_mask) Cleart bits on Register. More...
ReturnCode	**st25r500SetRegisterBits** (uint8_t reg, uint8_t set_mask) Set bits on Register. More...
ReturnCode	**st25r500ChangeRegisterBits** (uint8_t reg, uint8_t valueMask, uint8_t value) Changes the given bits on a ST25R500 register. More...
ReturnCode	**st25r500ModifyRegister** (uint8_t reg, uint8_t clr_mask, uint8_t set_mask) Modifies a value within a ST25R500 register. More...
ReturnCode	**st25r500ChangeTestRegisterBits** (uint8_t reg, uint8_t valueMask, uint8_t value) Changes the given bits on a ST25R500 Test register. More...
bool	**st25r500CheckReg** (uint8_t reg, uint8_t mask, uint8_t val) Checks if register contains a expected value. More...
bool	**st25r500IsRegValid** (uint8_t reg) Check if register ID is valid. More...
void	**st25r500ledInit** (void) ST25R500 LED Initialize. More...
void	**st25r500ledEvtIrq** (uint32_t irqs) ST25R500 LED Event Interrupt. More...
void	**st25r500ledEvtWrReg** (uint8_t reg, uint8_t val)

ST25R500 LED Event Write Register. More...

void **st25r500ledEvtWrMultiReg** (uint8_t reg, const uint8_t *vals, uint8_t len)
ST25R500 LED Event Write Multiple Register. More...

void **st25r500ledEvtCmd** (uint8_t cmd)
ST25R500 LED Event Direct Command. More...

void **st25r500ledEvtCmds** (const uint8_t *cmds, uint8_t len)
ST25R200 LED Event Direct Commands. More...

Detailed Description

Author

Gustavo Patricio

This driver provides basic abstraction for communication with the ST25R500

st25r500_dpocr.h File Reference

Dynamic Power Output adjustment via Current Regulation. More...

```
#include "rfal_platform.h"
#include "rfal_utils.h"
```

Include dependency graph for st25r500_dpocr.h:

Data Structures

struct **st25r500DpocrConfig**

struct **st25r500DpocrInfo**

Macros

#define **ST25R500_DPOCR_MAX_ENTRIES** 4U

Functions

ReturnCode **st25r500DpocrInitialize** (const **st25r500DpocrConfig** *config)

Initialize dynamic power table. More...

ReturnCode **st25r500DpocrAdjust** (void)

Dynamic power adjust. More...

void **st25r500DpocrSetEnabled** (bool enable)

Dynamic power set enabled state. More...

ReturnCode	**st25r500DpocrConfigWrite** (const **st25r500DpocrConfig** *config) Write dynamic power config. More...
ReturnCode	**st25r500DpocrConfigRead** (**st25r500DpocrConfig** *config) Dynamic power table Read. More...
void	**st25r500DpocrReqAdj** (void) Request DPO adjust. More...
ReturnCode	**st25r500DpocrGetInfo** (**st25r500DpocrInfo** *info) Get Dynamic power information. More...

Detailed Description

Author

Ulrich Herrmann

This module provides an interface to perform the power adjustment dynamically using a current regulation loop with a target value (ST25R500 only)

st25r500_irq.h File Reference

ST25R500 Interrupt handling. More...

```
#include "rfal_platform.h"
```

Include dependency graph for st25r500_irq.h:

Macros

#define **ST25R500_IRQ_MASK_ALL** (uint32_t)(0xFFFFFFFFUL)

#define **ST25R500_IRQ_MASK_NONE** (uint32_t)(0x00000000UL)

#define **ST25R500_IRQ_MASK_SUBC_START** (uint32_t)(0x00000080U)

#define **ST25R500_IRQ_MASK_COL** (uint32_t)(0x00000040U)

#define **ST25R500_IRQ_MASK_WL** (uint32_t)(0x00000020U)

#define **ST25R500_IRQ_MASK_RX_REST** (uint32_t)(0x00000010U)

#define **ST25R500_IRQ_MASK_RXE** (uint32_t)(0x00000008U)

#define **ST25R500_IRQ_MASK_RXS** (uint32_t)(0x00000004U)

#define **ST25R500_IRQ_MASK_TXE** (uint32_t)(0x00000002U)

#define **ST25R500_IRQ_MASK_RX_ERR** (uint32_t)(0x00000001U)

#define **ST25R500_IRQ_MASK_GPE** (uint32_t)(0x00008000U)

#define **ST25R500_IRQ_MASK_NRE** (uint32_t)(0x00004000U)

#define **ST25R500_IRQ_MASK_WPT_STOP** (uint32_t)(0x00002000U)

#define **ST25R500_IRQ_MASK_WPT_FOD** (uint32_t)(0x00001000U)

#define **ST25R500_IRQ_MASK_RFU** (uint32_t)(0x00000800U)

#define **ST25R500_IRQ_MASK_CE_SC** (uint32_t)(0x00000400U)

#define **ST25R500_IRQ_MASK_RXE_CE** (uint32_t)(0x00000200U)

#define **ST25R500_IRQ_MASK_NFCT** (uint32_t)(0x00000100U)

#define **ST25R500_IRQ_MASK_WUTME** (uint32_t)(0x00800000U)

#define **ST25R500_IRQ_MASK_EOF** (uint32_t)(0x00400000U)

#define **ST25R500_IRQ_MASK_EON** (uint32_t)(0x00200000U)

#define **ST25R500_IRQ_MASK_DCT** (uint32_t)(0x00100000U)

#define **ST25R500_IRQ_MASK_WUQ** (uint32_t)(0x00080000U)

#define **ST25R500_IRQ_MASK_WUI** (uint32_t)(0x00040000U)

#define **ST25R500_IRQ_MASK_WUT** (uint32_t)(0x00020000U)

#define **ST25R500_IRQ_MASK_OSC** (uint32_t)(0x00010000U)

Functions

uint32_t **st25r500WaitForInterruptsTimed** (uint32_t mask, uint16_t tmo)

Wait until an ST25R500 interrupt occurs. More...

uint32_t **st25r500GetInterrupt** (uint32_t mask)

Get status for the given interrupt. More...

void **st25r500InitInterrupts** (void)
Init the 200 interrupt. More...

void **st25r500ModifyInterrupts** (uint32_t clr_mask, uint32_t set_mask)
Modifies the Interrupt. More...

void **st25r500CheckForReceivedInterrupts** (void)
Checks received interrupts. More...

void **st25r500Isr** (void)
ISR Service routine. More...

void **st25r500EnableInterrupts** (uint32_t mask)
Enable a given ST25R500 Interrupt source. More...

void **st25r500DisableInterrupts** (uint32_t mask)
Disable one or more a given ST25R500 Interrupt sources. More...

void **st25r500ClearInterrupts** (void)
Clear all ST25R500 irq flags. More...

void **st25r500ClearAndEnableInterrupts** (uint32_t mask)
Clears and then enables the given ST25R500 Interrupt sources. More...

void **st25r500IRQCallbackSet** (void(*cb)(void))
Sets IRQ callback for the ST25R500 interrupt. More...

void **st25r500IRQCallbackRestore** (void)
Sets IRQ callback for the ST25R500 interrupt. More...

Detailed Description

Author

Gustavo Patricio

RFAL

st25r95 Directory Reference

Directory dependency graph for st25r95:

Files

file **st25r95/rfal_features.h**
RFAL Features/Capabilities Definition for ST25R95.

file **st25r95.h**
ST25R95 declaration file.

file **st25r95_com.h**
ST25R95 communication declaration file.

st25r95.h File Reference

ST25R95 declaration file. More...

```
#include "rfal_platform.h"
#include "rfal_utils.h"
```

Include dependency graph for st25r95.h:

Macros

#define **st25r95_nIRQ_IN_Pulse**() **st25r95SPI_nIRQ_IN_Pulse**()

#define **st25r95ResetChip**() **st25r95SPIResetChip**()

Functions

ReturnCode **st25r95Initialize** (void)
Initialise ST25R95 driver. More...

void **st25r95Deinitialize** (void)
Deinitialize ST25R95 driver. More...

void **st25r95SPI_nIRQ_IN_Pulse** (void)
Generates an IRQ_IN pulse (SPI intefrace) More...

void **st25r95UART_nIRQ_IN_Pulse** (void)
Generates an IRQ_IN pulse (UART interface) More...

bool **st25r95CheckChipID** (void)
Check Identity. More...

ReturnCode **st25r95FieldOn** (uint32_t protocol)
Turns field on. More...

ReturnCode **st25r95FieldOff** (void)
Turns field off. More...

Detailed Description

Author

API:

- Initialize ST25R95 driver: **st25r95Initialize**
- Deinitialize ST25R95 driver: **st25r95Deinitialize**

---

st25r95_com.h File Reference

ST25R95 communication declaration file. More...

```
#include "rfal_platform.h"
#include "rfal_utils.h"
#include "rfal_rf.h"
```

Include dependency graph for st25r95_com.h:

Data Structures

struct **st25r95SPIRxContext**

Macros

#define **ST25R95_CONTROL_SEND** 0x00

#define **ST25R95_CONTROL_RESET** 0x01

#define **ST25R95_CONTROL_READ** 0x02

#define **ST25R95_CONTROL_POLL** 0x03

#define **ST25R95_CONTROL_POLL_TIMEOUT** 100

#define **ST25R95_CONTROL_POLL_NO_TIMEOUT** 0

#define **ST25R95_COMMAND_IDN** 0x01

#define **ST25R95_COMMAND_PROTOCOLSELECT** 0x02

#define **ST25R95_COMMAND_POLLFIED** 0x03

#define **ST25R95_COMMAND_SENDRECV** 0x04

#define **ST25R95_COMMAND_LISTEN** 0x05

#define **ST25R95_COMMAND_SEND** 0x06

```
#define ST25R95_COMMAND_IDLE 0x07
#define ST25R95_COMMAND_RDREG 0x08
#define ST25R95_COMMAND_WRREG 0x09
#define ST25R95_COMMAND_BAUDRATE 0x0A
#define ST25R95_COMMAND_ACFILTER 0x0D
#define ST25R95_COMMAND_ECHO 0x55
#define ST25R95_SPI_DUMMY_BYTE 0x00
#define ST25R95_CMD_COMMAND_OFFSET 0x00U
#define ST25R95_CMD_RESULT_OFFSET 0x00U
#define ST25R95_CMD_LENGTH_OFFSET 0x01U
#define ST25R95_CMD_DATA_OFFSET 0x02U
#define ST25R95_COMMUNICATION_BUFFER_SIZE (528 + 2)
#define ST25R95_COMMUNICATION_UART_WDOGTIMER (5000U)
#define ST25R95_PROTOCOL_FIELDOFF 0x00U
#define ST25R95_PROTOCOL_ISO15693 0x01U
#define ST25R95_PROTOCOL_ISO14443A 0x02U
#define ST25R95_PROTOCOL_ISO14443B 0x03U
#define ST25R95_PROTOCOL_ISO18092 0x04U
#define ST25R95_PROTOCOL_CE_ISO14443A 0x05U
#define ST25R95_PROTOCOL_MAX ST25R95_PROTOCOL_CE_ISO14443A
#define ST25R95_FWT_MAX 0x40A8BC0
```

```
#define ST25R95_ERRCODE_NONE 0x00
#define ST25R95_ERRCODE_FRAMEOKADDITIONALINFO 0x80
#define ST25R95_ERRCODE_INVALIDCCMDCODE 0x81
#define ST25R95_ERRCODE_INVALIDCMDLENGHT 0x82
#define ST25R95_ERRCODE_INVALIDPROTOCOL 0x83
#define ST25R95_ERRCODE_COMERROR 0x86
#define ST25R95_ERRCODE_FRAMEWAITTIMEOUT 0x87
#define ST25R95_ERRCODE_INVALIDSOF 0x88
#define ST25R95_ERRCODE_OVERFLOW 0x89
#define ST25R95_ERRCODE_FRAMING 0x8A
#define ST25R95_ERRCODE_EGT 0x8B
#define ST25R95_ERRCODE_FIELDLENGTH 0x8C
#define ST25R95_ERRCODE_CRC 0x8D
#define ST25R95_ERRCODE_RECEPTIONLOST 0x8E
#define ST25R95_ERRCODE_NOFIELD 0x8F
#define ST25R95_ERRCODE_RESULTSRESIDUAL 0x90
#define ST25R95_ERRCODE_61_SOF 0x61
#define ST25R95_ERRCODE_62_CRC 0x62
#define ST25R95_ERRCODE_63_SOF_HIGH 0x63
#define ST25R95_ERRCODE_65_SOF_LOW 0x65
#define ST25R95_ERRCODE_66_EGT 0x66
```

```
#define ST25R95_ERRCODE_67_TR1TOOLONG 0x67
#define ST25R95_ERRCODE_68_TR1TOOSHORT 0x68
#define ST25R95_REG_ARC_B 0x6801
#define ST25R95_REG_ACC_A 0x6804
#define ST25R95_REG_TIMERW 0x3A00
#define ST25R95_IDN_RESPONSE_BUFLEN (15 + 2)
#define ST25R95_PROTOCOLSELECT_RESPONSE_BUFLEN (0 + 2)
#define ST25R95_POLLFIELD_RESPONSE_BUFLEN (1 + 2)
#define ST25R95_LISTEN_RESPONSE_BUFLEN (0 + 2)
#define ST25R95_RDREG_RESPONSE_BUFLEN (1 + 2)
#define ST25R95_WRREG_RESPONSE_BUFLEN (0 + 2)
#define ST25R95_ACFILTER_RESPONSE_BUFLEN (1 + 2)
#define ST25R95_IDLE_RESPONSE_BUFLEN (1 + 2)
#define ST25R95_ECHO_RESPONSE_BUFLEN (3)
#define ST25R95_SEND_RESPONSE_BUFLEN (0 + 2)
#define ST25R95_ACSTATE_IDLE 0x00U
#define ST25R95_ACSTATE_READYA 0x01U
#define ST25R95_ACSTATE_ACTIVE 0x04U
#define ST25R95_ACSTATE_HALT 0x80U
#define ST25R95_ACSTATE_READYAX 0x81U
#define ST25R95_ACSTATE_ACTIVEX 0x84U
```

#define **ST25R95_IS_PROT_ISO15693_CRC_ERR**(status) (((status) & 0x02U) == 0x02U)

#define **ST25R95_IS_PROT_ISO15693_COLLISION_ERR**(status) (((status) & 0x01U) == 0x01U)

#define **ST25R95_IS_PROT_ISO14443A_COLLISION_ERR**(status) (((status) & 0x80U) == 0x80U)

#define **ST25R95_IS_PROT_ISO14443A_CRC_ERR**(status) (((status) & 0x20U) == 0x20U)

#define **ST25R95_IS_PROT_ISO14443A_PARITY_ERR**(status) (((status) & 0x10U) == 0x10U)

#define **ST25R95_IS_PROT_ISO14443B_CRC_ERR**(status) (((status) & 0x02U) == 0x02U)

#define **ST25R95_IS_PROT_ISO18092_CRC_ERR**(status) (((status) & 0x02U) == 0x02U)

#define **ST25R95_PROTOCOLSELECT_BR_OFFSET** (3U)

#define **ST25R95_IDLE_WUPERIOD_OFFSET** (0x09U)

#define **ST25R95_IDLE_DACDATAL_OFFSET** (0x0CU)

#define **ST25R95_IDLE_DACDATAH_OFFSET** (0x0DU)

#define **ST25R95_DACDATA_MAX** (0xFCU)

#define **ST25R95_IDLE_WKUP_TIMEOUT** (0x01U)

#define **ST25R95_IDLE_WKUP_TAGDETECT** (0x02U)

#define **st25r95SendCommandTypeAndLen**(cmd, resp, respBuffLen) **st25r95SPISendCommandTypeAndLen**((cmd), (resp), (respBuffLen))

#define **st25r95CommandEcho**() **st25r95SPICommandEcho**()

#define **st25r95SendData**(buf, bufLen, protocol, flags) **st25r95SPISendData**((buf), (bufLen), (protocol), (flags))

#define **st25r95SendTransmitFlag**(protocol, transmitFlag) **st25r95SPISendTransmitFlag**((protocol), (transmitFlag))

#define **st25r95PollRead**(timeout) **st25r95SPIPollRead**((timeout))

#define **st25r95PrepareRx**(protocol, rxBuf, rxBufLen, rxRcvdLen, flags, additionalRespBytes) **st25r95SPIPrepareRx**((protocol), (rxBuf), (rxBufLen), (rxRcvdLen), (flags), (additionalRespBytes));

#define **st25r95CompleteRx**() **st25r95SPICompleteRx**();

#define **st25r95IsTransmitTxCompleted**() **st25r95SPIIsTransmitCompleted**()

#define **st25r95IsInListen**() **st25r95SPIIsInListen**()

#define **st25r95GetLmState**() **st25r95SPIGetLmState**()

#define **st25r95DeactivateACFilter**() **st25r95SPIDeactivateACFilter**()

#define **st25r95SetACState**(State) **st25r95SPISetACState**(State)

#define **st25r95Idle**(DacDataL, DacDataH, WUPeriod) **st25r95SPIIdle**((DacDataL), (DacDataH), (WUPeriod))

#define **st25r95GetIdleResponse**() **st25r95SPIGetIdleResponse**()

#define **st25r95KillIdle**() **st25r95SPIKillIdle**()

#define **st25r95FlushChipSPIBuffer**() **st25r95SPIRxTx**(**NULL**, **NULL**, **ST25R95_COMMUNICATION_BUFFER_SIZE**)

Functions

void **st25r95SPIResetChip** (void)
Resets the ST25R95 (SPI interface) More...

ReturnCode **st25r95SPISendCommandTypeAndLen** (uint8_t *cmd, uint8_t *resp, uint16_t respBuffLen)
Sends a command to ST25R95 and reads the response (SPI Interface) More...

ReturnCode **st25r95UARTSendCommandTypeAndLen** (uint8_t *cmd, uint8_t *resp, uint16_t respBuffLen)
Sends a command to ST25R95 and reads the response (UART Interface) More...

ReturnCode **st25r95SPICommandEcho** (void)
Sends a echo command to ST25R95 (SPI Interface) More...

ReturnCode **st25r95UARTCommandEcho** (void)
Sends a echo command to ST25R95 (UART Interface) More...

uint8_t **st25r95SPISendReceiveByte** (uint8_t data)
Sends one byte over SPI and returns the byte received. More...

ReturnCode	**st25r95ProtocolSelect** (uint8_t protocol) Sends ProtocolSelect command. More...
ReturnCode	**st25r95SetBitRate** (uint8_t protocol, **rfalBitRate** txBR, **rfalBitRate** rxBR) Set bit rates. More...
void	**st25r95SPISendData** (uint8_t *buf, uint8_t bufLen, uint8_t protocol, uint32_t flags) Sends data (SPI interface) More...
void	**st25r95UARTSendData** (uint8_t *buf, uint8_t bufLen, uint8_t protocol, uint32_t flags) Sends data (UART interface) More...
void	**st25r95SPIPrepareRx** (uint8_t protocol, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rxRcvdLen, uint32_t flags, uint8_t *additionalRespBytes) Prepares Rx (SPI Interface) More...
void	**st25r95UARTPrepareRx** (uint8_t protocol, uint8_t *rxBuf, uint16_t rxBufLen, uint16_t *rxRcvdLen, uint32_t flags, uint8_t *additionalRespBytes) Prepares Rx (UART Interface) More...
ReturnCode	**st25r95SPIPollRead** (uint32_t timeout) Polls the ST25R95 for incomming data (SPI Interface) More...
ReturnCode	**st25r95UARTPollRead** (uint32_t timeout) Polls the ST25R95 for incomming data (UART Interface) More...
ReturnCode	**st25r95SPIPollSend** (void) Polls the ST25R95 for sending (SPI Interface) More...
ReturnCode	**st25r95SetFWT** (uint8_t protocol, uint32_t fwt) set FWT More...
ReturnCode	**st25r95SetSlotCounter** (uint8_t slots) set slot counter More...
ReturnCode	**st25r95WriteReg** (uint8_t protocol, uint16_t reg, uint8_t value) Write Register. More...

ReturnCode	**st25r95ReadReg** (uint16_t reg, uint8_t *value) Read Register. More...
bool	**st25r95PollField** (void) Poll field (CE mode) More...
ReturnCode	**st25r95SetACFilter** (const **rfalLmConfPA** *confA) Set Anti-Collision filter (CE mode) More...
ReturnCode	**st25r95Listen** (void) Listen (CE mode) More...
void	**st25r95UartTxCpltCallback** (void) Uart Tx Callback (UART mode) More...
void	**st25r95UartRxCpltCallback** (void) Uart Rx Callback (UART mode) More...
void	**st25r95UartErrorCallback** (void) Uart Error Callback (UART mode) More...
bool	**st25r95SPIIsTransmitCompleted** (void) End of transmit test function (SPI mode) More...
bool	**st25r95UARTIsTransmitTxCompleted** (void) End of transmit test function (UART mode) More...
bool	**st25r95SPIIsInListen** (void) Listen mode test function (CE mode) More...
rfalLmState	**st25r95SPIGetLmState** (void) Get Listen Mode state (CE mode) More...
ReturnCode	**st25r95SPIDeactivateACFilter** (void) Deactivate the Anti-Collision filter (CE mode) More...
ReturnCode	**st25r95SPISetACState** (uint8_t state) Set Anti-Collision filter state (CE mode) More...

void	**st25r95RearmListen** (void) Rearm Listen mode (CE mode) More...
void	**st25r95SPIIdle** (uint8_t dacDataL, uint8_t dacDataH, uint8_t WUPeriod) Send Idle command (SPI mode) More...
void	**st25r95SPIGetIdleResponse** (void) Flush the Idle response (SPI mode) More...
void	**st25r95UARTIdle** (uint8_t dacDataL, uint8_t dacDataH, uint8_t WUPeriod) Send Idle command (UART mode) More...
void	**st25r95UARTGetIdleResponse** (void) Flush the Idle response (UART mode) More...
void	**st25r95SPISendTransmitFlag** (uint8_t protocol, uint8_t transmitFlag) Send transmit flag (SPI mode) More...
void	**st25r95UARTSendTransmitFlag** (uint8_t protocol, uint8_t transmitFlag) Send transmit flag (UART mode) More...
ReturnCode	**st25r95SPICompleteRx** (void) Process end of Rx (SPI mode) More...
ReturnCode	**st25r95UARTCompleteRx** (void) Process end of Rx (UART mode) More...
void	**st25r95SPIKillIdle** (void) return to ready mode (SPI mode) More...
void	**st25r95UARTKillIdle** (void) return to ready mode (UART mode) More...
uint8_t	**st25r95CalibrateTagDetector** (void) Calibrate Tag Detector. More...
void	**st25r95SPIRxTx** (uint8_t *txData, uint8_t *rxData, uint16_t length) SPI transceive function. More...

Variables

st25r95SPIRxContext	**st25r95SPIRxCtx**

Detailed Description

Author

RFAL

source

Functions

rfal_crc.h File Reference

CRC calculation module. More...

```
#include "rfal_platform.h"
```

Include dependency graph for rfal_crc.h:

Functions

uint16_t **rfalCrcCalculateCcitt** (uint16_t preloadValue, const uint8_t *buf, uint16_t length)
Calculate CRC according to CCITT standard. More...

Detailed Description

Author

Ulrich Herrmann

Function Documentation

◆ rfalCrcCalculateCcitt()

```
uint16_t rfalCrcCalculateCcitt ( uint16_t          preloadValue,
                                 const uint8_t *   buf,
                                 uint16_t          length
                               )
```

This function takes *length* bytes from *buf* and calculates the CRC for this data. The result is returned.

Note

This implementation calculates the CRC with LSB first, i.e. all bytes are "read" from right to left.

Parameters

[in] **preloadValue** : Initial value of CRC calculation.

[in] **buf** : buffer to calculate the CRC for.

[in] **length** : size of the buffer.

Returns

16 bit long crc value.

rfal_iso15693_2.h File Reference

Implementation of ISO-15693-2. More...

```
#include "rfal_platform.h"
#include "rfal_utils.h"
```

Include dependency graph for rfal_iso15693_2.h:

Data Structures

struct **rfalIso15693PhyConfig_t**

struct **iso15693StreamConfig**

Macros

#define **ISO15693_REQ_FLAG_TWO_SUBCARRIERS** 0x01U

#define **ISO15693_REQ_FLAG_HIGH_DATARATE** 0x02U

#define **ISO15693_MASK_FDT_LISTEN** (65)

#define **ISO15693_FWT** (69 + 12)

Enumerations

enum **rfalIso15693VcdCoding_t** { **ISO15693_VCD_CODING_1_4** , **ISO15693_VCD_CODING_1_256** }

Functions

ReturnCode **rfalIso15693PhyConfigure** (const **rfalIso15693PhyConfig_t** *config, const struct **iso15693StreamConfig** **needed_stream_config)

Initialize the ISO15693 phy. More...

ReturnCode **rfalIso15693PhyGetConfiguration** (**rfalIso15693PhyConfig_t** *config)

Return current phy configuration. More...

ReturnCode	**rfalIso15693VCDCode** (uint8_t *buffer, uint16_t length, bool sendCrc, bool sendFlags, bool picopassMode, uint16_t *subbit_total_length, uint16_t *offset, uint8_t *outbuf, uint16_t outBufSize, uint16_t *actOutBufSize) Code an ISO15693 compatible frame. More...
ReturnCode	**rfalIso15693VICCDecode** (const uint8_t *inBuf, uint16_t inBufLen, uint8_t *outBuf, uint16_t outBufLen, uint16_t *outBufPos, uint16_t *bitsBeforeCol, uint16_t ignoreBits, bool picopassMode) Receive an ISO15693 compatible frame. More...

Detailed Description

Author

Ulrich Herrmann

Macro Definition Documentation

◆ ISO15693_FWT

```
#define ISO15693_FWT   (69 + 12)
```

t1max = 323,3us = 4384/fc = 68.5 * 64/fc 12 = 768/fc unmodulated time of single subcarrior SoF

◆ ISO15693_MASK_FDT_LISTEN

```
#define ISO15693_MASK_FDT_LISTEN   (65)
```

t1min = 308,2us = 4192/fc = 65.5 * 64/fc

◆ ISO15693_REQ_FLAG_HIGH_DATARATE

```
#define ISO15693_REQ_FLAG_HIGH_DATARATE   0x02U
```

Flag indication that communication uses high bitrate

◆ ISO15693_REQ_FLAG_TWO_SUBCARRIERS

```
#define ISO15693_REQ_FLAG_TWO_SUBCARRIERS   0x01U
```

Flag indication that communication uses two subcarriers

Enumeration Type Documentation

◆ rfaIIso15693VcdCoding_t

enum **rfaIIso15693VcdCoding_t**

Enum holding possible VCD codings

Function Documentation

◆ rfaIIso15693PhyConfigure()

```
ReturnCode rfaIIso15693PhyConfigure ( const rfaIIso15693PhyConfig_t *          config,
                                      const struct iso15693StreamConfig **     needed_stream_config
                                    )
```

Parameters

[in]	**config**	: ISO15693 phy related configuration (See **rfaIIso15693PhyConfig_t**)
[out]	**needed_stream_config**	: return a pointer to the stream config needed for this iso15693 config. To be used for configure RF chip.

Returns

RFAL_ERR_IO : Error during communication.

RFAL_ERR_NONE : No error.

◆ rfaIIso15693PhyGetConfiguration()

```
ReturnCode rfaIIso15693PhyGetConfiguration ( rfaIIso15693PhyConfig_t *  config )
```

This function returns current Phy configuration previously set by rfaIIso15693PhyConfigure

Parameters

[out] **config** : ISO15693 phy configuration.

Returns

RFAL_ERR_NONE : No error.

◆ rfaIIso15693VCDCode()

```
ReturnCode rfaIIso15693VCDCode ( uint8_t *    buffer,
                                 uint16_t     length,
                                 bool         sendCrc,
                                 bool         sendFlags,
                                 bool         picopassMode,
                                 uint16_t *   subbit_total_length,
                                 uint16_t *   offset,
                                 uint8_t *    outbuf,
                                 uint16_t     outBufSize,
                                 uint16_t *   actOutBufSize
                               )
```

This function takes *length* bytes from *buffer*, perform proper encoding and sends out the frame to the ST25R391x.

Parameters

[in]	**buffer**	: data to send, modified to adapt flags.
[in]	**length**	: number of bytes to send.
[in]	**sendCrc**	: If set to true, CRC is appended to the frame
[in]	**sendFlags**	: If set to true, flag field is sent according to ISO15693.
[in]	**picopassMode**	: If set to true, the coding will be according to Picopass
[out]	**subbit_total_length**	: Return the complete bytes which need to be send for the current coding
[in,out]	**offset**	: Set to 0 for first transfer, function will update it to point to next byte to be coded
[out]	**outbuf**	: buffer where the function will store the coded subbit stream
[out]	**outBufSize**	: the size of the output buffer
[out]	**actOutBufSize**	: the amount of data stored into the buffer at this call

Returns

RFAL_ERR_IO : Error during communication.

RFAL_ERR_AGAIN : Data was not coded all the way. Call function again with a new/emptied buffer

RFAL_ERR_NO_MEM : In case outBuf is not big enough. Needs to have at least 5 bytes for 1of4 coding and 65 bytes for 1of256 coding

RFAL_ERR_NONE : No error

◆ rfaIIso15693VICCDecode()

```
ReturnCode rfaIIso15693VICCDecode ( const uint8_t *  inBuf,
                                    uint16_t         inBufLen,
                                    uint8_t *        outBuf,
                                    uint16_t         outBufLen,
                                    uint16_t *       outBufPos,
                                    uint16_t *       bitsBeforeCol,
                                    uint16_t         ignoreBits,
                                    bool             picopassMode
                                  )
```

This function receives an ISO15693 frame from the ST25R391x, decodes the frame and writes the raw data to *buffer*.

Note

Buffer needs to be big enough to hold CRC also (+2 bytes)

Parameters

[in]	**inBuf**	: buffer with the hamming coded stream to be decoded
[in]	**inBufLen**	: number of bytes to decode (=length of buffer).
[out]	**outBuf**	: buffer where received data shall be written to.
[in]	**outBufLen**	: Length of output buffer, should be approx twice the size of inBuf
[out]	**outBufPos**	: The number of decoded bytes. Could be used in extended implementation to allow multiple calls
[out]	**bitsBeforeCol**	: in case of RFAL_ERR_RF_COLLISION this value holds the number of bits in the current byte where the collision happened
[in]	**ignoreBits**	: number of bits in the beginning where collisions will be ignored
[in]	**picopassMode**	: if set to true, the decoding will be according to Picopass

Returns

RFAL_ERR_RF_COLLISION : collision occured, data uncorrect

RFAL_ERR_CRC : CRC error, data uncorrect

RFAL_ERR_TIMEOUT : timeout waiting for data.

RFAL_ERR_NONE : No error

rfal_platform.h File Reference

Platform specific definition layer

More...

```
#include <stdint.h>
#include <stdlib.h>
#include <stdbool.h>
#include <limits.h>
#include "stm32l4xx_hal.h"
#include "spi.h"
#include "timer.h"
#include "main.h"
#include "rfal_defConfig.h"
```

Include dependency graph for rfal_platform.h:

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

Macros

#define **ST25R_COM_SINGLETXRX**

#define **ST25R_SS_PIN** SPI1_CS_Pin

#define **ST25R_SS_PORT** SPI1_CS_GPIO_Port

#define **ST25R_INT_PIN** IRQ_391x_Pin

#define **ST25R_INT_PORT** IRQ_391x_GPIO_Port

#define **ST25R_RESET_PIN** RST_Pin

#define **ST25R_RESET_PORT** RST_GPIO_Port

#define **platformProtectST25RComm**() do{ **globalCommProtectCnt**++; __DSB();NVIC_DisableIRQ(EXTI0_IRQn);__DSB();__ISB();}while(0)

#define **platformUnprotectST25RComm**() do{ if (--**globalCommProtectCnt**==0) {NVIC_EnableIRQ(EXTI0_IRQn);} }while(0)

#define **platformProtectST25RIrqStatus**() **platformProtectST25RComm**()

#define **platformUnprotectST25RIrqStatus**() **platformUnprotectST25RComm**()

#define **platformLedOff**(port, pin) **platformGpioClear**((port), (pin))

#define **platformLedOn**(port, pin) **platformGpioSet**((port), (pin))

#define **platformLedToggle**(port, pin) **platformGpioToggle**((port), (pin))

```
#define platformGpioSet(port, pin)  HAL_GPIO_WritePin(port, pin, GPIO_PIN_SET)
#define platformGpioClear(port, pin)  HAL_GPIO_WritePin(port, pin, GPIO_PIN_RESET)
#define platformGpioToggle(port, pin)  HAL_GPIO_TogglePin(port, pin)
#define platformGpioIsHigh(port, pin)  (HAL_GPIO_ReadPin(port, pin) == GPIO_PIN_SET)
#define platformGpioIsLow(port, pin)  (!platformGpioIsHigh(port, pin))
#define platformTimerCreate(t)  timerCalculateTimer(t)
#define platformTimerIsExpired(timer)  timerIsExpired(timer)
#define platformTimerGetRemaining(timer)  timerGetRemaining(timer)
#define platformDelay(t)  HAL_Delay( t )
#define platformGetSysTick()  HAL_GetTick()
#define platformAssert(exp)  assert_param( exp )
#define platformErrorHandle()  Error_Handler()
#define platformSpiSelect()  platformGpioClear( ST25R_SS_PORT, ST25R_SS_PIN )
#define platformSpiDeselect()  platformGpioSet( ST25R_SS_PORT, ST25R_SS_PIN )
#define platformSpiTxRx(txBuf, rxBuf, len)  spiTxRx(txBuf, rxBuf, len)
#define platformI2CTx(txBuf, len, last, txOnly)
#define platformI2CRx(txBuf, len)
#define platformI2CStart()
#define platformI2CStop()
#define platformI2CRepeatStart()
#define platformI2CSlaveAddrWR(add)
```

#define **platformI2CSlaveAddrRD**(add)

#define **platformLog**(...)

#define **RFAL_FEATURE_LISTEN_MODE** true

#define **RFAL_FEATURE_WAKEUP_MODE** true

#define **RFAL_FEATURE_LOWPOWER_MODE** true

#define **RFAL_FEATURE_NFCA** true

#define **RFAL_FEATURE_NFCB** true

#define **RFAL_FEATURE_NFCF** true

#define **RFAL_FEATURE_NFCV** true

#define **RFAL_FEATURE_T1T** true

#define **RFAL_FEATURE_T2T** true

#define **RFAL_FEATURE_T4T** true

#define **RFAL_FEATURE_ST25TB** true

#define **RFAL_FEATURE_ST25xV** true

#define **RFAL_FEATURE_DYNAMIC_ANALOG_CONFIG** false

#define **RFAL_FEATURE_DPO** false

#define **RFAL_FEATURE_DLMA** false

#define **RFAL_FEATURE_ISO_DEP** true

#define **RFAL_FEATURE_ISO_DEP_POLL** true

#define **RFAL_FEATURE_ISO_DEP_LISTEN** true

#define **RFAL_FEATURE_NFC_DEP** true

#define **RFAL_FEATURE_ISO_DEP_IBLOCK_MAX_LEN** 256U

#define **RFAL_FEATURE_NFC_DEP_BLOCK_MAX_LEN** 254U

#define **RFAL_FEATURE_NFC_RF_BUF_LEN** 256U

#define **RFAL_FEATURE_ISO_DEP_APDU_MAX_LEN** 512U

#define **RFAL_FEATURE_NFC_DEP_PDU_MAX_LEN** 512U

Variables

uint8_t **globalCommProtectCnt**

Detailed Description

Author

This should contain all platform and hardware specifics such as GPIO assignment, system resources, locks, IRQs, etc

Each distinct platform/system/board must provide this definitions for all SW layers to use

Macro Definition Documentation

◆ platformAssert

#define platformAssert (exp) assert_param(exp)

Asserts whether the given expression is true

◆ platformDelay

#define platformDelay (t) HAL_Delay(t)

Performs a delay for the given time (ms)

Examples

exampleRfalNfca.c, and **exampleRfalPoller.c**.

◆ platformErrorHandle

```
#define platformErrorHandle ( ) Error_Handler()
```

Global error handle/trap

◆ platformGetSysTick

```
#define platformGetSysTick ( ) HAL_GetTick()
```

Get System Tick (1 tick = 1 ms)

◆ platformGpioClear

```
#define platformGpioClear ( port,
                            pin
                          ) HAL_GPIO_WritePin(port, pin, GPIO_PIN_RESET)
```

Turns the given GPIO Low

◆ platformGpioIsHigh

```
#define platformGpioIsHigh ( port,
                             pin
                           ) (HAL_GPIO_ReadPin(port, pin) == GPIO_PIN_SET)
```

Checks if the given LED is High

◆ platformGpioIsLow

```
#define platformGpioIsLow ( port,
                            pin
                          ) (!platformGpioIsHigh(port, pin))
```

Checks if the given LED is Low

◆ platformGpioSet

```
#define platformGpioSet (   port,
                            pin
                        )   HAL_GPIO_WritePin(port, pin, GPIO_PIN_SET)
```

Turns the given GPIO High

◆ platformGpioToggle

```
#define platformGpioToggle (   port,
                               pin
                           )   HAL_GPIO_TogglePin(port, pin)
```

Toggles the given GPIO

◆ platformI2CRepeatStart

```
#define platformI2CRepeatStart (  )
```

I2C Repeat Start

◆ platformI2CRx

```
#define platformI2CRx (   txBuf,
                          len
                      )
```

I2C Receive

◆ platformI2CSlaveAddrRD

```
#define platformI2CSlaveAddrRD (   add )
```

I2C Slave address for Read operation

◆ platformI2CSlaveAddrWR

```
#define platformI2CSlaveAddrWR (   add )
```

I2C Slave address for Write operation

◆ platformI2CStart

```
#define platformI2CStart (  )
```

I2C Start condition

◆ platformI2CStop

```
#define platformI2CStop (  )
```

I2C Stop condition

◆ platformI2CTx

```
#define platformI2CTx (   txBuf,
                          len,
                          last,
                          txOnly
                       )
```

I2C Transmit

◆ platformLedOff

```
#define platformLedOff (   port,
                           pin
                       )    platformGpioClear((port), (pin))
```

Turns the given LED Off

◆ platformLedOn

```
#define platformLedOn (  port,
                         pin
                       )   platformGpioSet((port), (pin))
```

Turns the given LED On

◆ platformLedToggle

```
#define platformLedToggle (  port,
                             pin
                           )   platformGpioToggle((port), (pin))
```

Toggles the given LED

◆ platformLog

```
#define platformLog (  ... )
```

Log method

◆ platformProtectST25RComm

```
#define                                do{ globalCommProtectCnt++;
platformProtectST25RComm      ( ) __DSB();NVIC_DisableIRQ(EXTI0_IRQn);__DSB();__ISB();}while(0)
```

Protect unique access to ST25R communication channel - IRQ disable on single thread environment (MCU) ; Mutex lock on a multi thread environment

◆ platformProtectST25RIrqStatus

```
#define platformProtectST25RIrqStatus (  )   platformProtectST25RComm()
```

Protect unique access to IRQ status var - IRQ disable on single thread environment (MCU) ; Mutex lock on a multi thread environment

◆ platformSpiDeselect

```
#define platformSpiDeselect ( ) platformGpioSet( ST25R_SS_PORT, ST25R_SS_PIN )
```

SPI SS/CS: Chip|Slave Deselect

◆ platformSpiSelect

```
#define platformSpiSelect ( ) platformGpioClear( ST25R_SS_PORT, ST25R_SS_PIN )
```

SPI SS/CS: Chip|Slave Select

◆ platformSpiTxRx

```
#define platformSpiTxRx ( txBuf,
                          rxBuf,
                          len
                        ) spiTxRx(txBuf, rxBuf, len)
```

SPI transceive

◆ platformTimerCreate

```
#define platformTimerCreate ( t ) timerCalculateTimer(t)
```

Creates a timer with the given time (ms)

◆ platformTimerGetRemaining

```
#define platformTimerGetRemaining ( timer ) timerGetRemaining(timer)
```

Gets the remaining time until expiration

◆ platformTimerIsExpired

```
#define platformTimerIsExpired ( timer ) timerIsExpired(timer)
```

Checks if the given timer is expired

◆ platformUnprotectST25RComm

```
#define platformUnprotectST25RComm ( ) do{ if (--globalCommProtectCnt==0) {NVIC_EnableIRQ(EXTI0_IRQn);} }while(0)
```

Unprotect unique access to ST25R communication channel - IRQ enable on a single thread environment (MCU) ; Mutex unlock on a multi thread environment

◆ platformUnprotectST25RIrqStatus

```
#define platformUnprotectST25RIrqStatus ( ) platformUnprotectST25RComm()
```

Unprotect the IRQ status var - IRQ enable on a single thread environment (MCU) ; Mutex unlock on a multi thread environment

◆ RFAL_FEATURE_DLMA

```
#define RFAL_FEATURE_DLMA false
```

Enable/Disable RFAL Dynamic LMA support

◆ RFAL_FEATURE_DPO

```
#define RFAL_FEATURE_DPO false
```

Enable/Disable RFAL Dynamic Power Output support

◆ RFAL_FEATURE_DYNAMIC_ANALOG_CONFIG

```
#define RFAL_FEATURE_DYNAMIC_ANALOG_CONFIG false
```

Enable/Disable Analog Configs to be dynamically updated (RAM)

◆ RFAL_FEATURE_ISO_DEP

```
#define RFAL_FEATURE_ISO_DEP   true
```

Enable/Disable RFAL support for ISO-DEP (ISO14443-4)

◆ RFAL_FEATURE_ISO_DEP_APDU_MAX_LEN

```
#define RFAL_FEATURE_ISO_DEP_APDU_MAX_LEN   512U
```

ISO-DEP APDU max length.

◆ RFAL_FEATURE_ISO_DEP_IBLOCK_MAX_LEN

```
#define RFAL_FEATURE_ISO_DEP_IBLOCK_MAX_LEN   256U
```

ISO-DEP I-Block max length. Please use values as defined by rfalIsoDepFSx

◆ RFAL_FEATURE_ISO_DEP_LISTEN

```
#define RFAL_FEATURE_ISO_DEP_LISTEN   true
```

Enable/Disable RFAL support for Listen mode (PICC) ISO-DEP (ISO14443-4)

◆ RFAL_FEATURE_ISO_DEP_POLL

```
#define RFAL_FEATURE_ISO_DEP_POLL   true
```

Enable/Disable RFAL support for Poller mode (PCD) ISO-DEP (ISO14443-4)

◆ RFAL_FEATURE_LISTEN_MODE

```
#define RFAL_FEATURE_LISTEN_MODE   true
```

Enable/Disable RFAL support for Listen Mode

◆ RFAL_FEATURE_LOWPOWER_MODE

```
#define RFAL_FEATURE_LOWPOWER_MODE   true
```

Enable/Disable RFAL support for the Low Power mode

◆ RFAL_FEATURE_NFC_DEP

```
#define RFAL_FEATURE_NFC_DEP   true
```

Enable/Disable RFAL support for NFC-DEP (NFCIP1/P2P)

◆ RFAL_FEATURE_NFC_DEP_BLOCK_MAX_LEN

```
#define RFAL_FEATURE_NFC_DEP_BLOCK_MAX_LEN   254U
```

NFC-DEP Block/Payload length. Allowed values: 64, 128, 192, 254

◆ RFAL_FEATURE_NFC_DEP_PDU_MAX_LEN

```
#define RFAL_FEATURE_NFC_DEP_PDU_MAX_LEN   512U
```

NFC-DEP PDU max length.

◆ RFAL_FEATURE_NFC_RF_BUF_LEN

```
#define RFAL_FEATURE_NFC_RF_BUF_LEN   256U
```

RF buffer length used by RFAL NFC layer

◆ RFAL_FEATURE_NFCA

```
#define RFAL_FEATURE_NFCA   true
```

Enable/Disable RFAL support for NFC-A (ISO14443A)

◆ RFAL_FEATURE_NFCB

```
#define RFAL_FEATURE_NFCB   true
```

Enable/Disable RFAL support for NFC-B (ISO14443B)

◆ RFAL_FEATURE_NFCF

```
#define RFAL_FEATURE_NFCF   true
```

Enable/Disable RFAL support for NFC-F (FeliCa)

◆ RFAL_FEATURE_NFCV

```
#define RFAL_FEATURE_NFCV   true
```

Enable/Disable RFAL support for NFC-V (ISO15693)

◆ RFAL_FEATURE_ST25TB

```
#define RFAL_FEATURE_ST25TB   true
```

Enable/Disable RFAL support for ST25TB

◆ RFAL_FEATURE_ST25xV

```
#define RFAL_FEATURE_ST25xV   true
```

Enable/Disable RFAL support for ST25TV/ST25DV

◆ RFAL_FEATURE_T1T

```
#define RFAL_FEATURE_T1T   true
```

Enable/Disable RFAL support for T1T (Topaz)

◆ RFAL_FEATURE_T2T

```
#define RFAL_FEATURE_T2T   true
```

Enable/Disable RFAL support for T2T

◆ RFAL_FEATURE_T4T

```
#define RFAL_FEATURE_T4T   true
```

Enable/Disable RFAL support for T4T

◆ RFAL_FEATURE_WAKEUP_MODE

```
#define RFAL_FEATURE_WAKEUP_MODE   true
```

Enable/Disable RFAL support for the Wake-Up mode

◆ ST25R_COM_SINGLETXRX

```
#define ST25R_COM_SINGLETXRX
```

Use single Transceive

◆ ST25R_INT_PIN

```
#define ST25R_INT_PIN   IRQ_391x_Pin
```

GPIO pin used for ST25R External Interrupt

◆ ST25R_INT_PORT

```
#define ST25R_INT_PORT   IRQ_391x_GPIO_Port
```

GPIO port used for ST25R External Interrupt

◆ ST25R_RESET_PIN

#define ST25R_RESET_PIN RST_Pin

GPIO pin used for ST25R Reset

◆ ST25R_RESET_PORT

#define ST25R_RESET_PORT RST_GPIO_Port

GPIO port used for ST25R Reset

◆ ST25R_SS_PIN

#define ST25R_SS_PIN SPI1_CS_Pin

GPIO pin used for ST25R SPI SS

◆ ST25R_SS_PORT

#define ST25R_SS_PORT SPI1_CS_GPIO_Port

GPIO port used for ST25RSPI SS port

Variable Documentation

◆ globalCommProtectCnt

uint8_t globalCommProtectCnt exter n

Global Protection Counter provided per platform - instantiated in main.c

RFAL

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

- g -

- globalCommProtectCnt : **rfal_platform.h**

Generated on Fri Mar 14 2025 11:45:12 for RFAL by doxygen 1.9.5

RFAL

p

r

s

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

- i -

- ISO15693_FWT : **rfal_iso15693_2.h**
- ISO15693_MASK_FDT_LISTEN : **rfal_iso15693_2.h**
- ISO15693_REQ_FLAG_HIGH_DATARATE : **rfal_iso15693_2.h**
- ISO15693_REQ_FLAG_TWO_SUBCARRIERS : **rfal_iso15693_2.h**

RFAL

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

- m -

- MS_FROM_64FCS : **st25r3911.h**
- MS_TO_64FCS : **st25r3911.h**

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

- n -

- NULL : **rfal_utils.h**

Data Structures

Examples

File List

Functions

Variables

Typedefs

Enumerations

Enumerator

Macros

g

i

m

n

r

s

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

- p -

- platformAssert : **rfal_defConfig.h**, **rfal_platform.h**
- platformDelay : **rfal_platform.h**
- platformErrorHandle : **rfal_defConfig.h**, **rfal_platform.h**
- platformGetSysTick : **rfal_defConfig.h**, **rfal_platform.h**

- platformGpioClear : **rfal_platform.h**
- platformGpioIsHigh : **rfal_platform.h**
- platformGpioIsLow : **rfal_platform.h**
- platformGpioSet : **rfal_platform.h**
- platformGpioToggle : **rfal_platform.h**
- platformI2CRepeatStart : **rfal_defConfig.h**, **rfal_platform.h**
- platformI2CRx : **rfal_defConfig.h**, **rfal_platform.h**
- platformI2CSlaveAddrRD : **rfal_defConfig.h**, **rfal_platform.h**
- platformI2CSlaveAddrWR : **rfal_defConfig.h**, **rfal_platform.h**
- platformI2CStart : **rfal_defConfig.h**, **rfal_platform.h**
- platformI2CStop : **rfal_defConfig.h**, **rfal_platform.h**
- platformI2CTx : **rfal_defConfig.h**, **rfal_platform.h**
- platformIrqST25RPinInitialize : **rfal_defConfig.h**
- platformIrqST25RSetCallback : **rfal_defConfig.h**
- platformLedOff : **rfal_defConfig.h**, **rfal_platform.h**
- platformLedOn : **rfal_defConfig.h**, **rfal_platform.h**
- platformLedsInitialize : **rfal_defConfig.h**
- platformLedToggle : **rfal_defConfig.h**, **rfal_platform.h**
- platformLog : **rfal_defConfig.h**, **rfal_platform.h**
- platformProtectST25RComm : **rfal_platform.h**
- platformProtectST25RIrqStatus : **rfal_defConfig.h**, **rfal_platform.h**
- platformProtectWorker : **rfal_defConfig.h**
- platformSpiDeselect : **rfal_platform.h**
- platformSpiSelect : **rfal_platform.h**
- platformSpiTxRx : **rfal_platform.h**
- platformTimerCreate : **rfal_platform.h**
- platformTimerDestroy : **rfal_defConfig.h**
- platformTimerGetRemaining : **rfal_defConfig.h**, **rfal_platform.h**
- platformTimerIsExpired : **rfal_platform.h**
- platformUnprotectST25RComm : **rfal_platform.h**
- platformUnprotectST25RIrqStatus : **rfal_defConfig.h**, **rfal_platform.h**
- platformUnprotectWorker : **rfal_defConfig.h**

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

- r -

- ReturnCode : **rfal_utils.h**
- rfal14443AShortFrameCmd : **rfal_rf.h**
- RFAL_14443A_SHORTFRAME_CMD_REQA : **rfal_rf.h**
- RFAL_14443A_SHORTFRAME_CMD_WUPA : **rfal_rf.h**
- RFAL_1FC_IN_2048FC : **rfal_rf.h**
- RFAL_1FC_IN_4096FC : **rfal_rf.h**
- RFAL_1FC_IN_512FC : **rfal_rf.h**
- RFAL_1FC_IN_64FC : **rfal_rf.h**
- RFAL_1FC_IN_8FC : **rfal_rf.h**
- RFAL_1MS_IN_1FC : **rfal_rf.h**
- RFAL_ANALOG_CONFIG_ANTICOL : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_BITRATE_105p94 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_BITRATE_106 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_BITRATE_1695 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_BITRATE_1p6 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_BITRATE_211p88 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_BITRATE_212 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_BITRATE_26 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_BITRATE_3390 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_BITRATE_424 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_BITRATE_53 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_BITRATE_6780 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_BITRATE_848 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_BITRATE_COMMON : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_BITRATE_MASK : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_BITRATE_RFU : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_BITRATE_RFU2 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_BITRATE_SHIFT : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_DEINIT : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_FIELD_OFF : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_FIELD_ON : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_INIT : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_LISTEN_COMMON : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_LISTEN_OFF : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_LISTEN_ON : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_LOWPOWER_OFF : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_LOWPOWER_ON : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_POLL_COMMON : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_00 : **rfal_analogConfig.h**

- RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_01 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_02 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_03 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_04 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_05 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_06 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_07 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_08 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_09 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_10 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_11 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_12 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_13 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_14 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_15 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_SPECIFIC_MASK : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_WAKEUP_OFF : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_WAKEUP_ON : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CONFIG_NUM : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_DIRECTION_MASK : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_DIRECTION_RFU10 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_DIRECTION_RFU11 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_DIRECTION_RFU2 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_DIRECTION_RFU3 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_DIRECTION_RFU4 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_DIRECTION_RFU5 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_DIRECTION_RFU6 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_DIRECTION_RFU7 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_DIRECTION_RFU8 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_DIRECTION_RFU9 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_DIRECTION_SHIFT : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_DLMA : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_DPO : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_GET_BITRATE : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_GET_DIRECTION : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_GET_POLL_LISTEN : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_GET_TECH : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_IS_106 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_IS_1695 : **rfal_analogConfig.h**

- RFAL_ANALOG_CONFIG_ID_IS_1p6 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_IS_212 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_IS_26 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_IS_3390 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_IS_424 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_IS_6780 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_IS_848 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_IS_AP2P : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_IS_CHIP : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_IS_COMMON : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_IS_NFCA : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_IS_NFCB : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_IS_NFCF : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_IS_NFCV : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_IS_RX : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_IS_TX : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_SET : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_LISTEN : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_LUT_NOT_FOUND : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_LUT_SIZE : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_NO_DIRECTION : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_POLL : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_POLL_LISTEN_MODE_MASK : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_POLL_LISTEN_MODE_SHIFT : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_RX : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_TBL_SIZE : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_TECH_AP2P : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_TECH_CHIP : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_TECH_MASK : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_TECH_NFCA : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_TECH_NFCB : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_TECH_NFCF : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_TECH_NFCV : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_TECH_RFU : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_TECH_RFU2 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_TECH_SHIFT : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_TX : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_UPDATE_LAST : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_UPDATE_MORE : **rfal_analogConfig.h**

- RFAL_BITS_IN_BYTE : **rfal_rf.h**
- RFAL_BR_105p94 : **rfal_rf.h**
- RFAL_BR_106 : **rfal_rf.h**
- RFAL_BR_13560 : **rfal_rf.h**
- RFAL_BR_1695 : **rfal_rf.h**
- RFAL_BR_1p66 : **rfal_rf.h**
- RFAL_BR_211p88 : **rfal_rf.h**
- RFAL_BR_212 : **rfal_rf.h**
- RFAL_BR_26p48 : **rfal_rf.h**
- RFAL_BR_3390 : **rfal_rf.h**
- RFAL_BR_424 : **rfal_rf.h**
- RFAL_BR_52p97 : **rfal_rf.h**
- RFAL_BR_6780 : **rfal_rf.h**
- RFAL_BR_848 : **rfal_rf.h**
- RFAL_BR_KEEP : **rfal_rf.h**
- RFAL_BYTECMP : **rfal_utils.h**
- RFAL_CD_TECH_NFCA : **rfal_cd.h**
- RFAL_CD_TECH_NFCB : **rfal_cd.h**
- RFAL_CD_TECH_NFCF : **rfal_cd.h**
- RFAL_CD_TECH_NFCV : **rfal_cd.h**
- RFAL_CD_TECH_NONE : **rfal_cd.h**
- RFAL_CD_TECH_OTHER : **rfal_cd.h**
- RFAL_COMPLIANCE_MODE_EMV : **rfal_rf.h**
- RFAL_COMPLIANCE_MODE_ISO : **rfal_rf.h**
- RFAL_COMPLIANCE_MODE_NFC : **rfal_rf.h**
- RFAL_CRC_LEN : **rfal_rf.h**
- RFAL_DLMA_TABLE_MAX_ENTRIES : **rfal_dlma.h**
- RFAL_DLMA_TABLE_PARAM_LEN : **rfal_dlma.h**
- RFAL_DLMA_TABLE_SIZE_MAX : **rfal_dlma.h**
- RFAL_DPO_TABLE_MAX_ENTRIES : **rfal_dpo.h**
- RFAL_DPO_TABLE_PARAM_LEN : **rfal_dpo.h**
- RFAL_DPO_TABLE_SIZE_MAX : **rfal_dpo.h**
- RFAL_ERR_AGAIN : **rfal_utils.h**
- RFAL_ERR_BUSY : **rfal_utils.h**
- RFAL_ERR_CRC : **rfal_utils.h**
- RFAL_ERR_DISABLED : **rfal_utils.h**
- RFAL_ERR_DONE : **rfal_utils.h**
- RFAL_ERR_FIFO : **rfal_utils.h**
- RFAL_ERR_FRAMING : **rfal_utils.h**

- RFAL_ERR_HW_MISMATCH : **rfal_utils.h**
- RFAL_ERR_HW_OVERRUN : **rfal_utils.h**
- RFAL_ERR_IGNORE : **rfal_utils.h**
- RFAL_ERR_INCOMPLETE_BYTE : **rfal_utils.h**
- RFAL_ERR_INCOMPLETE_BYTE_01 : **rfal_utils.h**
- RFAL_ERR_INCOMPLETE_BYTE_02 : **rfal_utils.h**
- RFAL_ERR_INCOMPLETE_BYTE_03 : **rfal_utils.h**
- RFAL_ERR_INCOMPLETE_BYTE_04 : **rfal_utils.h**
- RFAL_ERR_INCOMPLETE_BYTE_05 : **rfal_utils.h**
- RFAL_ERR_INCOMPLETE_BYTE_06 : **rfal_utils.h**
- RFAL_ERR_INCOMPLETE_BYTE_07 : **rfal_utils.h**
- RFAL_ERR_INTERNAL : **rfal_utils.h**
- RFAL_ERR_INVALID_HANDLE : **rfal_utils.h**
- RFAL_ERR_IO : **rfal_utils.h**
- RFAL_ERR_LINK_LOSS : **rfal_utils.h**
- RFAL_ERR_MAX_RERUNS : **rfal_utils.h**
- RFAL_ERR_MEM_CORRUPT : **rfal_utils.h**
- RFAL_ERR_NOMEM : **rfal_utils.h**
- RFAL_ERR_NOMSG : **rfal_utils.h**
- RFAL_ERR_NONE : **rfal_utils.h**
- RFAL_ERR_NOT_IMPLEMENTED : **rfal_utils.h**
- RFAL_ERR_NOTFOUND : **rfal_utils.h**
- RFAL_ERR_NOTSUPP : **rfal_utils.h**
- RFAL_ERR_NOTUNIQUE : **rfal_utils.h**
- RFAL_ERR_OVERRUN : **rfal_utils.h**
- RFAL_ERR_PAR : **rfal_utils.h**
- RFAL_ERR_PARAM : **rfal_utils.h**
- RFAL_ERR_PC_CORRUPT : **rfal_utils.h**
- RFAL_ERR_PROTO : **rfal_utils.h**
- RFAL_ERR_RELEASE_REQ : **rfal_utils.h**
- RFAL_ERR_REQUEST : **rfal_utils.h**
- RFAL_ERR_RF_COLLISION : **rfal_utils.h**
- RFAL_ERR_SEMANTIC : **rfal_utils.h**
- RFAL_ERR_SEND : **rfal_utils.h**
- RFAL_ERR_SLEEP_REQ : **rfal_utils.h**
- RFAL_ERR_SYNTAX : **rfal_utils.h**
- RFAL_ERR_SYSTEM : **rfal_utils.h**
- RFAL_ERR_TIMEOUT : **rfal_utils.h**
- RFAL_ERR_WRITE : **rfal_utils.h**

- RFAL_ERR_WRONG_STATE : **rfal_utils.h**
- RFAL_ERRORHANDLING_EMD : **rfal_rf.h**
- RFAL_ERRORHANDLING_NONE : **rfal_rf.h**
- RFAL_EXIT_ON_BUSY : **rfal_utils.h**
- RFAL_EXIT_ON_ERR : **rfal_utils.h**
- RFAL_FDT_LISTEN_AP2P_LISTENER : **rfal_rf.h**
- RFAL_FDT_LISTEN_AP2P_POLLER : **rfal_rf.h**
- RFAL_FDT_LISTEN_NFCA_LISTENER : **rfal_rf.h**
- RFAL_FDT_LISTEN_NFCA_POLLER : **rfal_rf.h**
- RFAL_FDT_LISTEN_NFCB_LISTENER : **rfal_rf.h**
- RFAL_FDT_LISTEN_NFCB_POLLER : **rfal_rf.h**
- RFAL_FDT_LISTEN_NFCF_LISTENER : **rfal_rf.h**
- RFAL_FDT_LISTEN_NFCF_POLLER : **rfal_rf.h**
- RFAL_FDT_LISTEN_NFCV_POLLER : **rfal_rf.h**
- RFAL_FDT_LISTEN_PICOPASS_POLLER : **rfal_rf.h**
- RFAL_FDT_POLL_AP2P_POLLER : **rfal_rf.h**
- RFAL_FDT_POLL_NFCA_POLLER : **rfal_rf.h**
- RFAL_FDT_POLL_NFCA_T1T_POLLER : **rfal_rf.h**
- RFAL_FDT_POLL_NFCB_POLLER : **rfal_rf.h**
- RFAL_FDT_POLL_NFCF_POLLER : **rfal_rf.h**
- RFAL_FDT_POLL_NFCV_POLLER : **rfal_rf.h**
- RFAL_FDT_POLL_PICOPASS_POLLER : **rfal_rf.h**
- RFAL_FEATURE_DLMA : **rfal_defConfig.h**, **rfal_platform.h**
- RFAL_FEATURE_DPO : **rfal_defConfig.h**, **rfal_platform.h**
- RFAL_FEATURE_DYNAMIC_ANALOG_CONFIG : **rfal_defConfig.h**, **rfal_platform.h**
- RFAL_FEATURE_ISO_DEP : **rfal_platform.h**
- RFAL_FEATURE_ISO_DEP_APDU_MAX_LEN : **rfal_isoDep.h**, **rfal_platform.h**
- RFAL_FEATURE_ISO_DEP_IBLOCK_MAX_LEN : **rfal_isoDep.h**, **rfal_platform.h**
- RFAL_FEATURE_ISO_DEP_LISTEN : **rfal_platform.h**
- RFAL_FEATURE_ISO_DEP_POLL : **rfal_platform.h**
- RFAL_FEATURE_LISTEN_MODE : **rfal_platform.h**
- RFAL_FEATURE_LOWPOWER_MODE : **rfal_defConfig.h**, **rfal_platform.h**
- RFAL_FEATURE_NFC_DEP : **rfal_platform.h**
- RFAL_FEATURE_NFC_DEP_BLOCK_MAX_LEN : **rfal_nfcDep.h**, **rfal_platform.h**
- RFAL_FEATURE_NFC_DEP_PDU_MAX_LEN : **rfal_nfcDep.h**, **rfal_platform.h**
- RFAL_FEATURE_NFC_RF_BUF_LEN : **rfal_defConfig.h**, **rfal_platform.h**
- RFAL_FEATURE_NFCA : **rfal_platform.h**
- RFAL_FEATURE_NFCB : **rfal_platform.h**
- RFAL_FEATURE_NFCF : **rfal_platform.h**

- RFAL_FEATURE_NFCV : **rfal_platform.h**
- RFAL_FEATURE_ST25TB : **rfal_platform.h**
- RFAL_FEATURE_ST25xV : **rfal_defConfig.h**, **rfal_platform.h**
- RFAL_FEATURE_T1T : **rfal_platform.h**
- RFAL_FEATURE_T2T : **rfal_platform.h**
- RFAL_FEATURE_T4T : **rfal_platform.h**
- RFAL_FEATURE_WAKEUP_MODE : **rfal_defConfig.h**, **rfal_platform.h**
- RFAL_FELICA_16_SLOTS : **rfal_rf.h**
- RFAL_FELICA_1_SLOT : **rfal_rf.h**
- RFAL_FELICA_2_SLOTS : **rfal_rf.h**
- RFAL_FELICA_4_SLOTS : **rfal_rf.h**
- RFAL_FELICA_8_SLOTS : **rfal_rf.h**
- RFAL_FELICA_LEN_LEN : **rfal_rf.h**
- RFAL_FELICA_POLL_MAX_SLOTS : **rfal_rf.h**
- RFAL_FELICA_POLL_RC_COM_PERFORMANCE : **rfal_rf.h**
- RFAL_FELICA_POLL_RC_NO_REQUEST : **rfal_rf.h**
- RFAL_FELICA_POLL_RC_SYSTEM_CODE : **rfal_rf.h**
- RFAL_FELICA_POLL_REQ_LEN : **rfal_rf.h**
- RFAL_FELICA_POLL_RES_LEN : **rfal_rf.h**
- RFAL_FWT_NONE : **rfal_rf.h**
- RFAL_GETU16 : **rfal_utils.h**
- RFAL_GETU32 : **rfal_utils.h**
- RFAL_GT_AP2P : **rfal_rf.h**
- RFAL_GT_AP2P_ADJUSTED : **rfal_rf.h**
- RFAL_GT_NFCA : **rfal_rf.h**
- RFAL_GT_NFCB : **rfal_rf.h**
- RFAL_GT_NFCF : **rfal_rf.h**
- RFAL_GT_NFCV : **rfal_rf.h**
- RFAL_GT_NONE : **rfal_rf.h**
- RFAL_GT_PICOPASS : **rfal_rf.h**
- RFAL_ISODEP_ADV_FEATURE : **rfal_isoDep.h**
- RFAL_ISODEP_APDU_MAX_LEN : **rfal_isoDep.h**
- RFAL_ISODEP_ATS_FWI_MASK : **rfal_isoDep.h**
- RFAL_ISODEP_ATS_HB_MAX_LEN : **rfal_isoDep.h**
- RFAL_ISODEP_ATS_SFGI_MASK : **rfal_isoDep.h**
- RFAL_ISODEP_ATS_T0_FSCI_MASK : **rfal_isoDep.h**
- RFAL_ISODEP_ATS_T0_OFFSET : **rfal_isoDep.h**
- RFAL_ISODEP_ATS_T0_TA_PRESENCE_MASK : **rfal_isoDep.h**
- RFAL_ISODEP_ATS_T0_TB_PRESENCE_MASK : **rfal_isoDep.h**

- RFAL_ISODEP_ATS_T0_TC_PRESENCE_MASK : **rfal_isoDep.h**
- RFAL_ISODEP_ATS_TA_DLP_212 : **rfal_isoDep.h**
- RFAL_ISODEP_ATS_TA_DLP_424 : **rfal_isoDep.h**
- RFAL_ISODEP_ATS_TA_DLP_848 : **rfal_isoDep.h**
- RFAL_ISODEP_ATS_TA_DPL_212 : **rfal_isoDep.h**
- RFAL_ISODEP_ATS_TA_DPL_424 : **rfal_isoDep.h**
- RFAL_ISODEP_ATS_TA_DPL_848 : **rfal_isoDep.h**
- RFAL_ISODEP_ATS_TA_SAME_D : **rfal_isoDep.h**
- RFAL_ISODEP_ATS_TB_FWI_MASK : **rfal_isoDep.h**
- RFAL_ISODEP_ATS_TB_FWI_SHIFT : **rfal_isoDep.h**
- RFAL_ISODEP_ATS_TB_SFGI_MASK : **rfal_isoDep.h**
- RFAL_ISODEP_ATTRIB_HLINFO_LEN : **rfal_isoDep.h**
- RFAL_ISODEP_ATTRIB_REQ_MIN_LEN : **rfal_isoDep.h**
- RFAL_ISODEP_ATTRIB_REQ_PARAM1_DEFAULT : **rfal_isoDep.h**
- RFAL_ISODEP_ATTRIB_RES_MBLI_NO_INFO : **rfal_isoDep.h**
- RFAL_ISODEP_ATTRIB_RES_MIN_LEN : **rfal_isoDep.h**
- RFAL_ISODEP_BITRATE_RFU_MASK : **rfal_isoDep.h**
- RFAL_ISODEP_BRI_MASK : **rfal_isoDep.h**
- RFAL_ISODEP_BSI_MASK : **rfal_isoDep.h**
- RFAL_ISODEP_DEFAULT_FSC : **rfal_isoDep.h**
- RFAL_ISODEP_DEFAULT_FSCI : **rfal_isoDep.h**
- RFAL_ISODEP_DEFAULT_FWI : **rfal_isoDep.h**
- RFAL_ISODEP_DEFAULT_SFGI : **rfal_isoDep.h**
- RFAL_ISODEP_DID_LEN : **rfal_isoDep.h**
- RFAL_ISODEP_DID_MAX : **rfal_isoDep.h**
- RFAL_ISODEP_FSDI_DEFAULT : **rfal_isoDep.h**
- RFAL_ISODEP_FSX_1024 : **rfal_isoDep.h**
- RFAL_ISODEP_FSX_128 : **rfal_isoDep.h**
- RFAL_ISODEP_FSX_16 : **rfal_isoDep.h**
- RFAL_ISODEP_FSX_2048 : **rfal_isoDep.h**
- RFAL_ISODEP_FSX_24 : **rfal_isoDep.h**
- RFAL_ISODEP_FSX_256 : **rfal_isoDep.h**
- RFAL_ISODEP_FSX_32 : **rfal_isoDep.h**
- RFAL_ISODEP_FSX_40 : **rfal_isoDep.h**
- RFAL_ISODEP_FSX_4096 : **rfal_isoDep.h**
- RFAL_ISODEP_FSX_48 : **rfal_isoDep.h**
- RFAL_ISODEP_FSX_512 : **rfal_isoDep.h**
- RFAL_ISODEP_FSX_64 : **rfal_isoDep.h**
- RFAL_ISODEP_FSX_96 : **rfal_isoDep.h**

- RFAL_ISODEP_FSX_KEEP : **rfal_isoDep.h**
- RFAL_ISODEP_FSXI_1024 : **rfal_isoDep.h**
- RFAL_ISODEP_FSXI_128 : **rfal_isoDep.h**
- RFAL_ISODEP_FSXI_16 : **rfal_isoDep.h**
- RFAL_ISODEP_FSXI_2048 : **rfal_isoDep.h**
- RFAL_ISODEP_FSXI_24 : **rfal_isoDep.h**
- RFAL_ISODEP_FSXI_256 : **rfal_isoDep.h**
- RFAL_ISODEP_FSXI_32 : **rfal_isoDep.h**
- RFAL_ISODEP_FSXI_40 : **rfal_isoDep.h**
- RFAL_ISODEP_FSXI_4096 : **rfal_isoDep.h**
- RFAL_ISODEP_FSXI_48 : **rfal_isoDep.h**
- RFAL_ISODEP_FSXI_512 : **rfal_isoDep.h**
- RFAL_ISODEP_FSXI_64 : **rfal_isoDep.h**
- RFAL_ISODEP_FSXI_96 : **rfal_isoDep.h**
- RFAL_ISODEP_FWI_DEFAULT : **rfal_isoDep.h**
- RFAL_ISODEP_FWI_MASK : **rfal_isoDep.h**
- RFAL_ISODEP_FWI_SHIFT : **rfal_isoDep.h**
- RFAL_ISODEP_MAX_DSL_RETRYS : **rfal_isoDep.h**
- RFAL_ISODEP_MAX_FWT : **rfal_isoDep.h**
- RFAL_ISODEP_MAX_I_RETRYS : **rfal_isoDep.h**
- RFAL_ISODEP_MAX_R_RETRYS : **rfal_isoDep.h**
- RFAL_ISODEP_MAX_WTX_NACK_RETRYS : **rfal_isoDep.h**
- RFAL_ISODEP_MAX_WTX_RETRYS : **rfal_isoDep.h**
- RFAL_ISODEP_MAX_WTX_RETRYS_ULTD : **rfal_isoDep.h**
- RFAL_ISODEP_NAD_LEN : **rfal_isoDep.h**
- RFAL_ISODEP_NO_DID : **rfal_isoDep.h**
- RFAL_ISODEP_NO_NAD : **rfal_isoDep.h**
- RFAL_ISODEP_PCB_LEN : **rfal_isoDep.h**
- RFAL_ISODEP_PROLOGUE_SIZE : **rfal_isoDep.h**
- RFAL_ISODEP_RATS_RETRIES : **rfal_isoDep.h**
- RFAL_ISODEP_SAME_BITRATE_MASK : **rfal_isoDep.h**
- RFAL_ISODEP_SPARAM_TAG_BLOCKINFO : **rfal_isoDep.h**
- RFAL_ISODEP_SPARAM_TAG_BRACK : **rfal_isoDep.h**
- RFAL_ISODEP_SPARAM_TAG_BRACK_LEN : **rfal_isoDep.h**
- RFAL_ISODEP_SPARAM_TAG_BRACT : **rfal_isoDep.h**
- RFAL_ISODEP_SPARAM_TAG_BRIND : **rfal_isoDep.h**
- RFAL_ISODEP_SPARAM_TAG_BRREQ : **rfal_isoDep.h**
- RFAL_ISODEP_SPARAM_TAG_BRREQ_LEN : **rfal_isoDep.h**
- RFAL_ISODEP_SPARAM_TAG_LEN : **rfal_isoDep.h**

- RFAL_ISODEP_SPARAM_TAG_PCD2PICC_LEN : **rfal_isoDep.h**
- RFAL_ISODEP_SPARAM_TAG_PICC2PCD_LEN : **rfal_isoDep.h**
- RFAL_ISODEP_SPARAM_TAG_SEL_FRAME : **rfal_isoDep.h**
- RFAL_ISODEP_SPARAM_TAG_SEL_PCD2PICC : **rfal_isoDep.h**
- RFAL_ISODEP_SPARAM_TAG_SEL_PICC2PCD : **rfal_isoDep.h**
- RFAL_ISODEP_SPARAM_TAG_SUP_FRAME : **rfal_isoDep.h**
- RFAL_ISODEP_SPARAM_TAG_SUP_PCD2PICC : **rfal_isoDep.h**
- RFAL_ISODEP_SPARAM_TAG_SUP_PICC2PCD : **rfal_isoDep.h**
- RFAL_ISODEP_SPARAM_VALUES_MAX_LEN : **rfal_isoDep.h**
- RFAL_LM_MASK_ACTIVE_P2P : **rfal_rf.h**
- RFAL_LM_MASK_NFCA : **rfal_rf.h**
- RFAL_LM_MASK_NFCB : **rfal_rf.h**
- RFAL_LM_MASK_NFCF : **rfal_rf.h**
- RFAL_LM_NFCID_LEN_04 : **rfal_rf.h**
- RFAL_LM_NFCID_LEN_07 : **rfal_rf.h**
- RFAL_LM_NFCID_LEN_10 : **rfal_rf.h**
- RFAL_LM_SENS_RES_LEN : **rfal_rf.h**
- RFAL_LM_SENSB_RES_LEN : **rfal_rf.h**
- RFAL_LM_SENSF_RES_LEN : **rfal_rf.h**
- RFAL_LM_SENSF_SC_LEN : **rfal_rf.h**
- RFAL_LM_STATE_ACTIVE_A : **rfal_rf.h**
- RFAL_LM_STATE_ACTIVE_Ax : **rfal_rf.h**
- RFAL_LM_STATE_CARDEMU_3 : **rfal_rf.h**
- RFAL_LM_STATE_CARDEMU_4A : **rfal_rf.h**
- RFAL_LM_STATE_CARDEMU_4B : **rfal_rf.h**
- RFAL_LM_STATE_IDLE : **rfal_rf.h**
- RFAL_LM_STATE_NOT_INIT : **rfal_rf.h**
- RFAL_LM_STATE_POWER_OFF : **rfal_rf.h**
- RFAL_LM_STATE_READY_A : **rfal_rf.h**
- RFAL_LM_STATE_READY_Ax : **rfal_rf.h**
- RFAL_LM_STATE_READY_B : **rfal_rf.h**
- RFAL_LM_STATE_READY_F : **rfal_rf.h**
- RFAL_LM_STATE_SLEEP_A : **rfal_rf.h**
- RFAL_LM_STATE_SLEEP_AF : **rfal_rf.h**
- RFAL_LM_STATE_SLEEP_B : **rfal_rf.h**
- RFAL_LM_STATE_TARGET_A : **rfal_rf.h**
- RFAL_LM_STATE_TARGET_F : **rfal_rf.h**
- RFAL_LP_MODE_HR : **rfal_rf.h**
- RFAL_LP_MODE_PD : **rfal_rf.h**

- RFAL_MAX : **rfal_utils.h**
- RFAL_MEMCPY : **rfal_utils.h**
- RFAL_MEMMOVE : **rfal_utils.h**
- RFAL_MEMSET : **rfal_utils.h**
- RFAL_MIN : **rfal_utils.h**
- RFAL_MODE_LISTEN_ACTIVE_P2P : **rfal_rf.h**
- RFAL_MODE_LISTEN_NFCA : **rfal_rf.h**
- RFAL_MODE_LISTEN_NFCB : **rfal_rf.h**
- RFAL_MODE_LISTEN_NFCF : **rfal_rf.h**
- RFAL_MODE_NONE : **rfal_rf.h**
- RFAL_MODE_POLL_ACTIVE_P2P : **rfal_rf.h**
- RFAL_MODE_POLL_B_CTS : **rfal_rf.h**
- RFAL_MODE_POLL_B_PRIME : **rfal_rf.h**
- RFAL_MODE_POLL_NFCA : **rfal_rf.h**
- RFAL_MODE_POLL_NFCA_T1T : **rfal_rf.h**
- RFAL_MODE_POLL_NFCB : **rfal_rf.h**
- RFAL_MODE_POLL_NFCF : **rfal_rf.h**
- RFAL_MODE_POLL_NFCV : **rfal_rf.h**
- RFAL_MODE_POLL_PICOPASS : **rfal_rf.h**
- RFAL_NFC_DEACTIVATE_DISCOVERY : **rfal_nfc.h**
- RFAL_NFC_DEACTIVATE_IDLE : **rfal_nfc.h**
- RFAL_NFC_DEACTIVATE_SLEEP : **rfal_nfc.h**
- RFAL_NFC_INTERFACE_ISODEP : **rfal_nfc.h**
- RFAL_NFC_INTERFACE_NFCDEP : **rfal_nfc.h**
- RFAL_NFC_INTERFACE_RF : **rfal_nfc.h**
- RFAL_NFC_LISTEN_TECH_A : **rfal_nfc.h**
- RFAL_NFC_LISTEN_TECH_AP2P : **rfal_nfc.h**
- RFAL_NFC_LISTEN_TECH_B : **rfal_nfc.h**
- RFAL_NFC_LISTEN_TECH_F : **rfal_nfc.h**
- RFAL_NFC_LISTEN_TYPE_AP2P : **rfal_nfc.h**
- RFAL_NFC_LISTEN_TYPE_NFCA : **rfal_nfc.h**
- RFAL_NFC_LISTEN_TYPE_NFCB : **rfal_nfc.h**
- RFAL_NFC_LISTEN_TYPE_NFCF : **rfal_nfc.h**
- RFAL_NFC_LISTEN_TYPE_NFCV : **rfal_nfc.h**
- RFAL_NFC_LISTEN_TYPE_PROP : **rfal_nfc.h**
- RFAL_NFC_LISTEN_TYPE_ST25TB : **rfal_nfc.h**
- RFAL_NFC_POLL_TECH_A : **rfal_nfc.h**
- RFAL_NFC_POLL_TECH_AP2P : **rfal_nfc.h**
- RFAL_NFC_POLL_TECH_B : **rfal_nfc.h**

- RFAL_NFC_POLL_TECH_F : **rfal_nfc.h**
- RFAL_NFC_POLL_TECH_PROP : **rfal_nfc.h**
- RFAL_NFC_POLL_TECH_ST25TB : **rfal_nfc.h**
- RFAL_NFC_POLL_TECH_V : **rfal_nfc.h**
- RFAL_NFC_POLL_TYPE_AP2P : **rfal_nfc.h**
- RFAL_NFC_POLL_TYPE_NFCA : **rfal_nfc.h**
- RFAL_NFC_POLL_TYPE_NFCB : **rfal_nfc.h**
- RFAL_NFC_POLL_TYPE_NFCF : **rfal_nfc.h**
- RFAL_NFC_POLL_TYPE_NFCV : **rfal_nfc.h**
- RFAL_NFC_STATE_ACTIVATED : **rfal_nfc.h**
- RFAL_NFC_STATE_DATAEXCHANGE : **rfal_nfc.h**
- RFAL_NFC_STATE_DATAEXCHANGE_DONE : **rfal_nfc.h**
- RFAL_NFC_STATE_DEACTIVATION : **rfal_nfc.h**
- RFAL_NFC_STATE_IDLE : **rfal_nfc.h**
- RFAL_NFC_STATE_LISTEN_ACTIVATION : **rfal_nfc.h**
- RFAL_NFC_STATE_LISTEN_COLAVOIDANCE : **rfal_nfc.h**
- RFAL_NFC_STATE_LISTEN_SLEEP : **rfal_nfc.h**
- RFAL_NFC_STATE_LISTEN_TECHDETECT : **rfal_nfc.h**
- RFAL_NFC_STATE_NOTINIT : **rfal_nfc.h**
- RFAL_NFC_STATE_POLL_ACTIVATION : **rfal_nfc.h**
- RFAL_NFC_STATE_POLL_COLAVOIDANCE : **rfal_nfc.h**
- RFAL_NFC_STATE_POLL_SELECT : **rfal_nfc.h**
- RFAL_NFC_STATE_POLL_TECHDETECT : **rfal_nfc.h**
- RFAL_NFC_STATE_START_DISCOVERY : **rfal_nfc.h**
- RFAL_NFC_STATE_WAKEUP_MODE : **rfal_nfc.h**
- RFAL_NFC_TECH_NONE : **rfal_nfc.h**
- RFAL_NFCA_CASCADE_1_UID_LEN : **rfal_nfca.h**
- RFAL_NFCA_CASCADE_2_UID_LEN : **rfal_nfca.h**
- RFAL_NFCA_CASCADE_3_UID_LEN : **rfal_nfca.h**
- RFAL_NFCA_FDTMIN : **rfal_nfca.h**
- RFAL_NFCA_SEL_RES_CONF_MASK : **rfal_nfca.h**
- RFAL_NFCA_SEL_RES_CONF_NFCDEP : **rfal_nfca.h**
- RFAL_NFCA_SEL_RES_CONF_T2T : **rfal_nfca.h**
- RFAL_NFCA_SEL_RES_CONF_T4T : **rfal_nfca.h**
- RFAL_NFCA_SEL_RES_CONF_T4T_NFCDEP : **rfal_nfca.h**
- RFAL_NFCA_SENS_RES_PLATFORM_MASK : **rfal_nfca.h**
- RFAL_NFCA_SENS_RES_PLATFORM_T1T : **rfal_nfca.h**
- RFAL_NFCB_AFI : **rfal_nfcb.h**
- RFAL_NFCB_CMD_LEN : **rfal_nfcb.h**

- RFAL_NFCB_CRC_LEN : **rfal_nfcb.h**
- RFAL_NFCB_DFWT : **rfal_nfcb.h**
- RFAL_NFCB_DTPOLL_10 : **rfal_nfcb.h**
- RFAL_NFCB_DTPOLL_20 : **rfal_nfcb.h**
- RFAL_NFCB_FWTSENSB : **rfal_nfcb.h**
- RFAL_NFCB_NFCID0_LEN : **rfal_nfcb.h**
- RFAL_NFCB_PARAM : **rfal_nfcb.h**
- RFAL_NFCB_SENS_CMD_ALLB_REQ : **rfal_nfcb.h**
- RFAL_NFCB_SENS_CMD_SENSB_REQ : **rfal_nfcb.h**
- RFAL_NFCB_SENSB_REQ_ADV_FEATURE : **rfal_nfcb.h**
- RFAL_NFCB_SENSB_RES_ADC_ADV_FEATURE_MASK : **rfal_nfcb.h**
- RFAL_NFCB_SENSB_RES_ADC_MASK : **rfal_nfcb.h**
- RFAL_NFCB_SENSB_RES_ADC_PROPRIETARY_MASK : **rfal_nfcb.h**
- RFAL_NFCB_SENSB_RES_EXT_LEN : **rfal_nfcb.h**
- RFAL_NFCB_SENSB_RES_FO_DID_MASK : **rfal_nfcb.h**
- RFAL_NFCB_SENSB_RES_FO_MASK : **rfal_nfcb.h**
- RFAL_NFCB_SENSB_RES_FO_NAD_MASK : **rfal_nfcb.h**
- RFAL_NFCB_SENSB_RES_FSCI_MASK : **rfal_nfcb.h**
- RFAL_NFCB_SENSB_RES_FSCI_SHIFT : **rfal_nfcb.h**
- RFAL_NFCB_SENSB_RES_FWI_MASK : **rfal_nfcb.h**
- RFAL_NFCB_SENSB_RES_FWI_SHIFT : **rfal_nfcb.h**
- RFAL_NFCB_SENSB_RES_LEN : **rfal_nfcb.h**
- RFAL_NFCB_SENSB_RES_PROTO_ISO_MASK : **rfal_nfcb.h**
- RFAL_NFCB_SENSB_RES_PROTO_RFU_MASK : **rfal_nfcb.h**
- RFAL_NFCB_SENSB_RES_PROTO_TR2_MASK : **rfal_nfcb.h**
- RFAL_NFCB_SENSB_RES_PROTO_TR2_SHIFT : **rfal_nfcb.h**
- RFAL_NFCB_SENSB_RES_SFGI_MASK : **rfal_nfcb.h**
- RFAL_NFCB_SENSB_RES_SFGI_SHIFT : **rfal_nfcb.h**
- RFAL_NFCB_SLOT_NUM_1 : **rfal_nfcb.h**
- RFAL_NFCB_SLOT_NUM_16 : **rfal_nfcb.h**
- RFAL_NFCB_SLOT_NUM_2 : **rfal_nfcb.h**
- RFAL_NFCB_SLOT_NUM_4 : **rfal_nfcb.h**
- RFAL_NFCB_SLOT_NUM_8 : **rfal_nfcb.h**
- RFAL_NFCDEP_ATR_REQ_NFCID3_POS : **rfal_nfcDep.h**
- RFAL_NFCDEP_ATRREQ_MAX_LEN : **rfal_nfcDep.h**
- RFAL_NFCDEP_ATRREQ_MIN_LEN : **rfal_nfcDep.h**
- RFAL_NFCDEP_ATRRES_HEADER_LEN : **rfal_nfcDep.h**
- RFAL_NFCDEP_ATRRES_MAX_LEN : **rfal_nfcDep.h**
- RFAL_NFCDEP_ATRRES_MIN_LEN : **rfal_nfcDep.h**

- RFAL_NFCDEP_BR_MASK_106 : **rfal_nfcDep.h**
- RFAL_NFCDEP_BR_MASK_212 : **rfal_nfcDep.h**
- RFAL_NFCDEP_BR_MASK_424 : **rfal_nfcDep.h**
- RFAL_NFCDEP_BRS_DSI_POS : **rfal_nfcDep.h**
- RFAL_NFCDEP_BRS_Dx_MASK : **rfal_nfcDep.h**
- RFAL_NFCDEP_BRS_MAINTAIN : **rfal_nfcDep.h**
- RFAL_NFCDEP_Bx_08_848 : **rfal_nfcDep.h**
- RFAL_NFCDEP_Bx_16_1695 : **rfal_nfcDep.h**
- RFAL_NFCDEP_Bx_32_3390 : **rfal_nfcDep.h**
- RFAL_NFCDEP_Bx_64_6780 : **rfal_nfcDep.h**
- RFAL_NFCDEP_Bx_NO_HIGH_BR : **rfal_nfcDep.h**
- RFAL_NFCDEP_CMD_LEN : **rfal_nfcDep.h**
- RFAL_NFCDEP_CMDTYPE_LEN : **rfal_nfcDep.h**
- RFAL_NFCDEP_COMM_ACTIVE : **rfal_nfcDep.h**
- RFAL_NFCDEP_COMM_PASSIVE : **rfal_nfcDep.h**
- RFAL_NFCDEP_DEP_HEADER : **rfal_nfcDep.h**
- RFAL_NFCDEP_DEP_PFB_LEN : **rfal_nfcDep.h**
- RFAL_NFCDEP_DEPREQ_HEADER_LEN : **rfal_nfcDep.h**
- RFAL_NFCDEP_DID_KEEP : **rfal_nfcDep.h**
- RFAL_NFCDEP_DID_LEN : **rfal_nfcDep.h**
- RFAL_NFCDEP_DID_MAX : **rfal_nfcDep.h**
- RFAL_NFCDEP_DID_NO : **rfal_nfcDep.h**
- RFAL_NFCDEP_DSL_RLS_LEN_DID : **rfal_nfcDep.h**
- RFAL_NFCDEP_DSL_RLS_LEN_NO_DID : **rfal_nfcDep.h**
- RFAL_NFCDEP_Dx_01_106 : **rfal_nfcDep.h**
- RFAL_NFCDEP_Dx_02_212 : **rfal_nfcDep.h**
- RFAL_NFCDEP_Dx_04_424 : **rfal_nfcDep.h**
- RFAL_NFCDEP_Dx_08_848 : **rfal_nfcDep.h**
- RFAL_NFCDEP_Dx_16_1695 : **rfal_nfcDep.h**
- RFAL_NFCDEP_Dx_32_3390 : **rfal_nfcDep.h**
- RFAL_NFCDEP_Dx_64_6780 : **rfal_nfcDep.h**
- RFAL_NFCDEP_FRAME_SIZE_MAX_LEN : **rfal_nfcDep.h**
- RFAL_NFCDEP_FS_VAL_MIN : **rfal_nfcDep.h**
- RFAL_NFCDEP_GB_MAX_LEN : **rfal_nfcDep.h**
- RFAL_NFCDEP_HEADER : **rfal_nfcDep.h**
- RFAL_NFCDEP_LEN_LEN : **rfal_nfcDep.h**
- RFAL_NFCDEP_LEN_MAX : **rfal_nfcDep.h**
- RFAL_NFCDEP_LEN_MIN : **rfal_nfcDep.h**
- RFAL_NFCDEP_LR_128 : **rfal_nfcDep.h**

- RFAL_NFCDEP_LR_192 : **rfal_nfcDep.h**
- RFAL_NFCDEP_LR_254 : **rfal_nfcDep.h**
- RFAL_NFCDEP_LR_64 : **rfal_nfcDep.h**
- RFAL_NFCDEP_LR_VAL_MASK : **rfal_nfcDep.h**
- RFAL_NFCDEP_MAX_FWT : **rfal_nfcDep.h**
- RFAL_NFCDEP_NAD_NO : **rfal_nfcDep.h**
- RFAL_NFCDEP_NFCID3_LEN : **rfal_nfcDep.h**
- RFAL_NFCDEP_OPER_ATN_DIS : **rfal_nfcDep.h**
- RFAL_NFCDEP_OPER_ATN_EN : **rfal_nfcDep.h**
- RFAL_NFCDEP_OPER_EMPTY_DEP_DIS : **rfal_nfcDep.h**
- RFAL_NFCDEP_OPER_EMPTY_DEP_EN : **rfal_nfcDep.h**
- RFAL_NFCDEP_OPER_FULL_MI_DIS : **rfal_nfcDep.h**
- RFAL_NFCDEP_OPER_FULL_MI_EN : **rfal_nfcDep.h**
- RFAL_NFCDEP_OPER_RTOX_REQ_DIS : **rfal_nfcDep.h**
- RFAL_NFCDEP_OPER_RTOX_REQ_EN : **rfal_nfcDep.h**
- RFAL_NFCDEP_PP_LR_MASK : **rfal_nfcDep.h**
- RFAL_NFCDEP_PP_LR_SHIFT : **rfal_nfcDep.h**
- RFAL_NFCDEP_ROLE_INITIATOR : **rfal_nfcDep.h**
- RFAL_NFCDEP_ROLE_TARGET : **rfal_nfcDep.h**
- RFAL_NFCDEP_RWT_INI_MAX : **rfal_nfcDep.h**
- RFAL_NFCDEP_RWT_TRG_MAX : **rfal_nfcDep.h**
- RFAL_NFCDEP_SB_LEN : **rfal_nfcDep.h**
- RFAL_NFCDEP_WT_DELTA : **rfal_nfcDep.h**
- RFAL_NFCDEP_WT_DELTA_ADJUST : **rfal_nfcDep.h**
- RFAL_NFCDEP_WT_INI_DEFAULT : **rfal_nfcDep.h**
- RFAL_NFCDEP_WT_INI_MAX : **rfal_nfcDep.h**
- RFAL_NFCDEP_WT_INI_MIN : **rfal_nfcDep.h**
- RFAL_NFCDEP_WT_MASK : **rfal_nfcDep.h**
- RFAL_NFCDEP_WT_TRG_MAX : **rfal_nfcDep.h**
- RFAL_NFCDEP_WT_TRG_MAX_D10 : **rfal_nfcDep.h**
- RFAL_NFCDEP_WT_TRG_MAX_D11 : **rfal_nfcDep.h**
- RFAL_NFCDEP_WT_TRG_MAX_L13 : **rfal_nfcDep.h**
- RFAL_NFCF_BLOCK_LEN : **rfal_nfcf.h**
- RFAL_NFCF_BLOCKLISTELEM_LEN_BIT : **rfal_nfcf.h**
- RFAL_NFCF_BLOCKLISTELEM_MAX_LEN : **rfal_nfcf.h**
- RFAL_NFCF_CHECKUPDATE_RES_NOB_POS : **rfal_nfcf.h**
- RFAL_NFCF_CHECKUPDATE_RES_ST1_POS : **rfal_nfcf.h**
- RFAL_NFCF_CHECKUPDATE_RES_ST2_POS : **rfal_nfcf.h**
- RFAL_NFCF_CMD_AUTHENTICATION1 : **rfal_nfcf.h**

- RFAL_NFCF_CMD_AUTHENTICATION2 : **rfal_nfcf.h**
- RFAL_NFCF_CMD_LEN : **rfal_nfcf.h**
- RFAL_NFCF_CMD_POLLING : **rfal_nfcf.h**
- RFAL_NFCF_CMD_POLLING_RES : **rfal_nfcf.h**
- RFAL_NFCF_CMD_POS : **rfal_nfcf.h**
- RFAL_NFCF_CMD_READ : **rfal_nfcf.h**
- RFAL_NFCF_CMD_READ_WITHOUT_ENCRYPTION : **rfal_nfcf.h**
- RFAL_NFCF_CMD_READ_WITHOUT_ENCRYPTION_RES : **rfal_nfcf.h**
- RFAL_NFCF_CMD_REQUEST_RESPONSE : **rfal_nfcf.h**
- RFAL_NFCF_CMD_REQUEST_SERVICE : **rfal_nfcf.h**
- RFAL_NFCF_CMD_REQUEST_SYSTEM_CODE : **rfal_nfcf.h**
- RFAL_NFCF_CMD_WRITE : **rfal_nfcf.h**
- RFAL_NFCF_CMD_WRITE_WITHOUT_ENCRYPTION : **rfal_nfcf.h**
- RFAL_NFCF_CMD_WRITE_WITHOUT_ENCRYPTION_RES : **rfal_nfcf.h**
- RFAL_NFCF_HEADER_LEN : **rfal_nfcf.h**
- RFAL_NFCF_LENGTH_LEN : **rfal_nfcf.h**
- RFAL_NFCF_NFCID2_LEN : **rfal_nfcf.h**
- RFAL_NFCF_NOB_LEN : **rfal_nfcf.h**
- RFAL_NFCF_NOS_LEN : **rfal_nfcf.h**
- RFAL_NFCF_POLL_MAXCARDS : **rfal_nfcf.h**
- RFAL_NFCF_SENSF_NFCID2_BYTE1_NFCDEP : **rfal_nfcf.h**
- RFAL_NFCF_SENSF_NFCID2_BYTE1_POS : **rfal_nfcf.h**
- RFAL_NFCF_SENSF_NFCID2_BYTE2_NFCDEP : **rfal_nfcf.h**
- RFAL_NFCF_SENSF_NFCID2_BYTE2_POS : **rfal_nfcf.h**
- RFAL_NFCF_SENSF_NFCID2_PROT_TYPE_LEN : **rfal_nfcf.h**
- RFAL_NFCF_SENSF_PARAMS_RC_POS : **rfal_nfcf.h**
- RFAL_NFCF_SENSF_PARAMS_SC1_POS : **rfal_nfcf.h**
- RFAL_NFCF_SENSF_PARAMS_SC2_POS : **rfal_nfcf.h**
- RFAL_NFCF_SENSF_PARAMS_TSN_POS : **rfal_nfcf.h**
- RFAL_NFCF_SENSF_RES_BYTE1 : **rfal_nfcf.h**
- RFAL_NFCF_SENSF_RES_LEN_MAX : **rfal_nfcf.h**
- RFAL_NFCF_SENSF_RES_LEN_MIN : **rfal_nfcf.h**
- RFAL_NFCF_SENSF_RES_PAD0_LEN : **rfal_nfcf.h**
- RFAL_NFCF_SENSF_RES_PAD1_LEN : **rfal_nfcf.h**
- RFAL_NFCF_SENSF_RES_RD_LEN : **rfal_nfcf.h**
- RFAL_NFCF_SENSF_SC_LEN : **rfal_nfcf.h**
- RFAL_NFCF_SERVICECODE_RDONLY : **rfal_nfcf.h**
- RFAL_NFCF_SERVICECODE_RDWR : **rfal_nfcf.h**
- RFAL_NFCF_STATUS_FLAG_ERROR : **rfal_nfcf.h**

- RFAL_NFCF_STATUS_FLAG_SUCCESS : **rfal_nfcf.h**
- RFAL_NFCF_SYSTEMCODE : **rfal_nfcf.h**
- RFAL_NFCF_SYSTEMCODE_LEN : **rfal_nfcf.h**
- RFAL_NFCF_TEST_LB_CMD0 : **rfal_nfcf.h**
- RFAL_NFCF_TEST_LB_CMD1 : **rfal_nfcf.h**
- RFAL_NFCID1_DOUBLE_LEN : **rfal_rf.h**
- RFAL_NFCID1_SINGLE_LEN : **rfal_rf.h**
- RFAL_NFCID1_TRIPLE_LEN : **rfal_rf.h**
- RFAL_NFCID2_LEN : **rfal_rf.h**
- RFAL_NFCID3_LEN : **rfal_rf.h**
- RFAL_NFCV_BLOCKNUM_EXTENDED_LEN : **rfal_nfcv.h**
- RFAL_NFCV_BLOCKNUM_LEN : **rfal_nfcv.h**
- RFAL_NFCV_BLOCKNUM_M24LR_LEN : **rfal_st25xv.h**
- RFAL_NFCV_BNO_LEN : **rfal_nfcv.h**
- RFAL_NFCV_CMD_ENABLE_EAS : **rfal_nfcv.h**
- RFAL_NFCV_CMD_EXTENDED_GET_SYS_INFO : **rfal_nfcv.h**
- RFAL_NFCV_CMD_EXTENDED_LOCK_SINGLE_BLOCK : **rfal_nfcv.h**
- RFAL_NFCV_CMD_EXTENDED_READ_MULTIPLE_BLOCK : **rfal_nfcv.h**
- RFAL_NFCV_CMD_EXTENDED_READ_SINGLE_BLOCK : **rfal_nfcv.h**
- RFAL_NFCV_CMD_EXTENDED_WRITE_MULTIPLE_BLOCK : **rfal_nfcv.h**
- RFAL_NFCV_CMD_EXTENDED_WRITE_SINGLE_BLOCK : **rfal_nfcv.h**
- RFAL_NFCV_CMD_FAST_EXTENDED_READ_MULTIPLE_BLOCKS : **rfal_nfcv.h**
- RFAL_NFCV_CMD_FAST_EXTENDED_READ_SINGLE_BLOCK : **rfal_nfcv.h**
- RFAL_NFCV_CMD_FAST_READ_DYN_CONFIGURATION : **rfal_nfcv.h**
- RFAL_NFCV_CMD_FAST_READ_MESSAGE : **rfal_nfcv.h**
- RFAL_NFCV_CMD_FAST_READ_MESSAGE_LENGTH : **rfal_nfcv.h**
- RFAL_NFCV_CMD_FAST_READ_MULTIPLE_BLOCKS : **rfal_nfcv.h**
- RFAL_NFCV_CMD_FAST_READ_SINGLE_BLOCK : **rfal_nfcv.h**
- RFAL_NFCV_CMD_FAST_WRITE_DYN_CONFIGURATION : **rfal_nfcv.h**
- RFAL_NFCV_CMD_FAST_WRITE_MESSAGE : **rfal_nfcv.h**
- RFAL_NFCV_CMD_GET_RANDOM_NUMBER : **rfal_nfcv.h**
- RFAL_NFCV_CMD_GET_SYS_INFO : **rfal_nfcv.h**
- RFAL_NFCV_CMD_INVENTORY : **rfal_nfcv.h**
- RFAL_NFCV_CMD_KILL : **rfal_nfcv.h**
- RFAL_NFCV_CMD_LOCK_BLOCK : **rfal_nfcv.h**
- RFAL_NFCV_CMD_LOCK_EAS : **rfal_nfcv.h**
- RFAL_NFCV_CMD_LOCK_KILL : **rfal_nfcv.h**
- RFAL_NFCV_CMD_MANAGE_GPO : **rfal_nfcv.h**
- RFAL_NFCV_CMD_PRESENT_PASSWORD : **rfal_nfcv.h**

- RFAL_NFCV_CMD_READ_CONFIGURATION : **rfal_nfcv.h**
- RFAL_NFCV_CMD_READ_DYN_CONFIGURATION : **rfal_nfcv.h**
- RFAL_NFCV_CMD_READ_MESSAGE : **rfal_nfcv.h**
- RFAL_NFCV_CMD_READ_MESSAGE_LENGTH : **rfal_nfcv.h**
- RFAL_NFCV_CMD_READ_MULTIPLE_BLOCKS : **rfal_nfcv.h**
- RFAL_NFCV_CMD_READ_SINGLE_BLOCK : **rfal_nfcv.h**
- RFAL_NFCV_CMD_RESET_EAS : **rfal_nfcv.h**
- RFAL_NFCV_CMD_RESET_TO_READY : **rfal_nfcv.h**
- RFAL_NFCV_CMD_SELECT : **rfal_nfcv.h**
- RFAL_NFCV_CMD_SET_EAS : **rfal_nfcv.h**
- RFAL_NFCV_CMD_SLPV : **rfal_nfcv.h**
- RFAL_NFCV_CMD_WRITE_CONFIGURATION : **rfal_nfcv.h**
- RFAL_NFCV_CMD_WRITE_DYN_CONFIGURATION : **rfal_nfcv.h**
- RFAL_NFCV_CMD_WRITE_EAS_CONFIG : **rfal_nfcv.h**
- RFAL_NFCV_CMD_WRITE_EAS_ID : **rfal_nfcv.h**
- RFAL_NFCV_CMD_WRITE_MESSAGE : **rfal_nfcv.h**
- RFAL_NFCV_CMD_WRITE_MULTIPLE_BLOCKS : **rfal_nfcv.h**
- RFAL_NFCV_CMD_WRITE_PASSWORD : **rfal_nfcv.h**
- RFAL_NFCV_CMD_WRITE_SINGLE_BLOCK : **rfal_nfcv.h**
- RFAL_NFCV_CRC_LEN : **rfal_nfcv.h**
- RFAL_NFCV_ERROR_BLOCK_ALREDY_LOCKED : **rfal_nfcv.h**
- RFAL_NFCV_ERROR_BLOCK_FAILED : **rfal_nfcv.h**
- RFAL_NFCV_ERROR_BLOCK_LOCKED : **rfal_nfcv.h**
- RFAL_NFCV_ERROR_BLOCK_NOT_AVALIABLE : **rfal_nfcv.h**
- RFAL_NFCV_ERROR_CMD_NOT_RECOGNIZED : **rfal_nfcv.h**
- RFAL_NFCV_ERROR_CMD_NOT_SUPPORTED : **rfal_nfcv.h**
- RFAL_NFCV_ERROR_OPTION_NOT_SUPPORTED : **rfal_nfcv.h**
- RFAL_NFCV_ERROR_UNKNOWN : **rfal_nfcv.h**
- RFAL_NFCV_ERROR_WRITE_FAILED : **rfal_nfcv.h**
- RFAL_NFCV_MAX_BLOCK_LEN : **rfal_nfcv.h**
- RFAL_NFCV_MAX_GEN_DATA_LEN : **rfal_nfcv.h**
- RFAL_NFCV_NUM_SLOTS_1 : **rfal_nfcv.h**
- RFAL_NFCV_NUM_SLOTS_16 : **rfal_nfcv.h**
- RFAL_NFCV_PARAM_SKIP : **rfal_nfcv.h**
- RFAL_NFCV_REQ_FLAG_ADDRESS : **rfal_nfcv.h**
- RFAL_NFCV_REQ_FLAG_AFI : **rfal_nfcv.h**
- RFAL_NFCV_REQ_FLAG_DATA_RATE : **rfal_nfcv.h**
- RFAL_NFCV_REQ_FLAG_DEFAULT : **rfal_nfcv.h**
- RFAL_NFCV_REQ_FLAG_INVENTORY : **rfal_nfcv.h**

- RFAL_NFCV_REQ_FLAG_NB_SLOTS : **rfal_nfcv.h**
- RFAL_NFCV_REQ_FLAG_OPTION : **rfal_nfcv.h**
- RFAL_NFCV_REQ_FLAG_PROTOCOL_EXT : **rfal_nfcv.h**
- RFAL_NFCV_REQ_FLAG_RFU : **rfal_nfcv.h**
- RFAL_NFCV_REQ_FLAG_SELECT : **rfal_nfcv.h**
- RFAL_NFCV_REQ_FLAG_SUB_CARRIER : **rfal_nfcv.h**
- RFAL_NFCV_RES_FLAG_ERROR : **rfal_nfcv.h**
- RFAL_NFCV_RES_FLAG_EXTENSION : **rfal_nfcv.h**
- RFAL_NFCV_RES_FLAG_RFU1 : **rfal_nfcv.h**
- RFAL_NFCV_RES_FLAG_RFU2 : **rfal_nfcv.h**
- RFAL_NFCV_RES_FLAG_RFU3 : **rfal_nfcv.h**
- RFAL_NFCV_RES_FLAG_RFU4 : **rfal_nfcv.h**
- RFAL_NFCV_RES_FLAG_RFU5 : **rfal_nfcv.h**
- RFAL_NFCV_RES_FLAG_RFU6 : **rfal_nfcv.h**
- RFAL_NFCV_ST_IC_MFG_CODE : **rfal_nfcv.h**
- RFAL_NFCV_SYSINFO_AFI : **rfal_nfcv.h**
- RFAL_NFCV_SYSINFO_CMDLIST : **rfal_nfcv.h**
- RFAL_NFCV_SYSINFO_CSI : **rfal_nfcv.h**
- RFAL_NFCV_SYSINFO_DFSID : **rfal_nfcv.h**
- RFAL_NFCV_SYSINFO_ICREF : **rfal_nfcv.h**
- RFAL_NFCV_SYSINFO_MEMSIZE : **rfal_nfcv.h**
- RFAL_NFCV_SYSINFO_MOI : **rfal_nfcv.h**
- RFAL_NFCV_SYSINFO_REQ_ALL : **rfal_nfcv.h**
- RFAL_NFCV_UID_LEN : **rfal_nfcv.h**
- RFAL_NO_WARNING : **rfal_utils.h**
- RFAL_SIZEOF_ARRAY : **rfal_utils.h**
- RFAL_ST25TB_BLOCK_LEN : **rfal_st25tb.h**
- RFAL_ST25TB_CHIP_ID_LEN : **rfal_st25tb.h**
- RFAL_ST25TB_CRC_LEN : **rfal_st25tb.h**
- RFAL_ST25TB_UID_LEN : **rfal_st25tb.h**
- RFAL_SUPPORT_AP2P : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_AP2P_106 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_AP2P_1695 : **st25r3911/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_AP2P_212 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_AP2P_3390 : **st25r3911/rfal_features.h**, **st25r95/rfal_features.h**

- RFAL_SUPPORT_BR_AP2P_424 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_AP2P_6780 : **st25r3911/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_AP2P_848 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_CE_A_106 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_CE_A_212 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_CE_A_424 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_CE_A_848 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_CE_B_106 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_CE_B_212 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_CE_B_424 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_CE_B_848 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_CE_F_212 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_CE_F_424 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_RW_106 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_RW_13560 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**
- RFAL_SUPPORT_BR_RW_1695 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_RW_212 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_RW_26 : **st25r200/rfal_features.h**, **st25r500/rfal_features.h**
- RFAL_SUPPORT_BR_RW_3390 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_RW_424 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_RW_53 : **st25r200/rfal_features.h**, **st25r500/rfal_features.h**

- RFAL_SUPPORT_BR_RW_6780 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_RW_848 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_CE : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_MODE_LISTEN_ACTIVE_P2P : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_MODE_LISTEN_NFCA : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_MODE_LISTEN_NFCB : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_MODE_LISTEN_NFCF : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_MODE_POLL_ACTIVE_P2P : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_MODE_POLL_NFCA : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_MODE_POLL_NFCB : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_MODE_POLL_NFCF : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_MODE_POLL_NFCV : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_RW : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_T1T_CMD_RALL : **rfal_t1t.h**
- RFAL_T1T_CMD_READ : **rfal_t1t.h**
- RFAL_T1T_CMD_RID : **rfal_t1t.h**
- RFAL_T1T_CMD_WRITE_E : **rfal_t1t.h**
- RFAL_T1T_CMD_WRITE_NE : **rfal_t1t.h**
- RFAL_T1T_HR0_NDEF_MASK : **rfal_t1t.h**
- RFAL_T1T_HR0_NDEF_SUPPORT : **rfal_t1t.h**
- RFAL_T1T_HR_LENGTH : **rfal_t1t.h**
- RFAL_T1T_UID_LEN : **rfal_t1t.h**
- RFAL_T2T_BLOCK_LEN : **rfal_t2t.h**
- RFAL_T2T_READ_DATA_LEN : **rfal_t2t.h**
- RFAL_T2T_WRITE_DATA_LEN : **rfal_t2t.h**
- RFAL_T4T_CLA : **rfal_t4t.h**

- RFAL_T4T_INS_READBINARY : **rfal_t4t.h**
- RFAL_T4T_INS_READBINARY_ODO : **rfal_t4t.h**
- RFAL_T4T_INS_SELECT : **rfal_t4t.h**
- RFAL_T4T_INS_UPDATEBINARY : **rfal_t4t.h**
- RFAL_T4T_INS_UPDATEBINARY_ODO : **rfal_t4t.h**
- RFAL_T4T_ISO7816_P1_SELECT_BY_DF_NAME : **rfal_t4t.h**
- RFAL_T4T_ISO7816_P1_SELECT_BY_FILEID : **rfal_t4t.h**
- RFAL_T4T_ISO7816_P2_SELECT_FIRST_OR_ONLY_OCCURENCE : **rfal_t4t.h**
- RFAL_T4T_ISO7816_P2_SELECT_NO_RESPONSE_DATA : **rfal_t4t.h**
- RFAL_T4T_ISO7816_P2_SELECT_RETURN_FCI_TEMPLATE : **rfal_t4t.h**
- RFAL_T4T_ISO7816_STATUS_COMPLETE : **rfal_t4t.h**
- RFAL_T4T_LC_LEN : **rfal_t4t.h**
- RFAL_T4T_LE_LEN : **rfal_t4t.h**
- RFAL_T4T_MAX_CAPDU_PROLOGUE_LEN : **rfal_t4t.h**
- RFAL_T4T_MAX_RAPDU_SW1SW2_LEN : **rfal_t4t.h**
- RFAL_TIMING_NONE : **rfal_rf.h**
- RFAL_TXRX_FLAGS_AGC_OFF : **rfal_rf.h**
- RFAL_TXRX_FLAGS_AGC_ON : **rfal_rf.h**
- RFAL_TXRX_FLAGS_CRC_RX_AUTO : **rfal_rf.h**
- RFAL_TXRX_FLAGS_CRC_RX_KEEP : **rfal_rf.h**
- RFAL_TXRX_FLAGS_CRC_RX_MANUAL : **rfal_rf.h**
- RFAL_TXRX_FLAGS_CRC_RX_REMV : **rfal_rf.h**
- RFAL_TXRX_FLAGS_CRC_TX_AUTO : **rfal_rf.h**
- RFAL_TXRX_FLAGS_CRC_TX_MANUAL : **rfal_rf.h**
- RFAL_TXRX_FLAGS_DEFAULT : **rfal_rf.h**
- RFAL_TXRX_FLAGS_NFCIP1_OFF : **rfal_rf.h**
- RFAL_TXRX_FLAGS_NFCIP1_ON : **rfal_rf.h**
- RFAL_TXRX_FLAGS_NFCV_FLAG_AUTO : **rfal_rf.h**
- RFAL_TXRX_FLAGS_NFCV_FLAG_MANUAL : **rfal_rf.h**
- RFAL_TXRX_FLAGS_PAR_RX_KEEP : **rfal_rf.h**
- RFAL_TXRX_FLAGS_PAR_RX_REMV : **rfal_rf.h**
- RFAL_TXRX_FLAGS_PAR_TX_AUTO : **rfal_rf.h**
- RFAL_TXRX_FLAGS_PAR_TX_NONE : **rfal_rf.h**
- RFAL_US_IN_MS : **rfal_rf.h**
- RFAL_VERSION : **rfal_rf.h**
- RFAL_WUM_AA_WEIGHT_100 : **st25r500/rfal_features.h**
- RFAL_WUM_AA_WEIGHT_128 : **st25r500/rfal_features.h**
- RFAL_WUM_AA_WEIGHT_16 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**

- RFAL_WUM_AA_WEIGHT_32 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**
- RFAL_WUM_AA_WEIGHT_4 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**
- RFAL_WUM_AA_WEIGHT_50 : **st25r500/rfal_features.h**
- RFAL_WUM_AA_WEIGHT_64 : **st25r500/rfal_features.h**
- RFAL_WUM_AA_WEIGHT_8 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**
- RFAL_WUM_MEAS_DUR_26_10 : **st25r200/rfal_features.h**, **st25r500/rfal_features.h**
- RFAL_WUM_MEAS_DUR_30_14 : **st25r200/rfal_features.h**, **st25r500/rfal_features.h**
- RFAL_WUM_MEAS_DUR_34_19 : **st25r200/rfal_features.h**, **st25r500/rfal_features.h**
- RFAL_WUM_MEAS_DUR_44_28 : **st25r200/rfal_features.h**, **st25r500/rfal_features.h**
- RFAL_WUM_MEAS_FIL_FAST : **st25r200/rfal_features.h**, **st25r500/rfal_features.h**
- RFAL_WUM_MEAS_FIL_SLOW : **st25r200/rfal_features.h**, **st25r500/rfal_features.h**
- RFAL_WUM_PERIOD_1000MS : **st25r500/rfal_features.h**
- RFAL_WUM_PERIOD_100MS : **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**
- RFAL_WUM_PERIOD_105MS : **st25r200/rfal_features.h**
- RFAL_WUM_PERIOD_10MS : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**
- RFAL_WUM_PERIOD_1240MS : **st25r200/rfal_features.h**
- RFAL_WUM_PERIOD_125MS : **st25r500/rfal_features.h**
- RFAL_WUM_PERIOD_12MS : **st25r500/rfal_features.h**
- RFAL_WUM_PERIOD_150MS : **st25r500/rfal_features.h**
- RFAL_WUM_PERIOD_155MS : **st25r200/rfal_features.h**
- RFAL_WUM_PERIOD_15MS : **st25r200/rfal_features.h**
- RFAL_WUM_PERIOD_1700MS : **st25r200/rfal_features.h**
- RFAL_WUM_PERIOD_200MS : **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**
- RFAL_WUM_PERIOD_20MS : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**
- RFAL_WUM_PERIOD_215MS : **st25r200/rfal_features.h**
- RFAL_WUM_PERIOD_250MS : **st25r500/rfal_features.h**
- RFAL_WUM_PERIOD_25MS : **st25r200/rfal_features.h**, **st25r500/rfal_features.h**
- RFAL_WUM_PERIOD_300MS : **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_WUM_PERIOD_30MS : **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**
- RFAL_WUM_PERIOD_310MS : **st25r200/rfal_features.h**
- RFAL_WUM_PERIOD_350MS : **st25r500/rfal_features.h**
- RFAL_WUM_PERIOD_400MS : **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**
- RFAL_WUM_PERIOD_40MS : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**
- RFAL_WUM_PERIOD_425MS : **st25r200/rfal_features.h**
- RFAL_WUM_PERIOD_500MS : **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**

- RFAL_WUM_PERIOD_50MS : **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**
- RFAL_WUM_PERIOD_55MS : **st25r200/rfal_features.h**
- RFAL_WUM_PERIOD_600MS : **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**
- RFAL_WUM_PERIOD_60MS : **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**
- RFAL_WUM_PERIOD_620MS : **st25r200/rfal_features.h**
- RFAL_WUM_PERIOD_700MS : **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**
- RFAL_WUM_PERIOD_70MS : **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**
- RFAL_WUM_PERIOD_75MS : **st25r500/rfal_features.h**
- RFAL_WUM_PERIOD_800MS : **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**
- RFAL_WUM_PERIOD_80MS : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**
- RFAL_WUM_PERIOD_850MS : **st25r200/rfal_features.h**
- RFAL_WUM_PERIOD_INF : **st25r500/rfal_features.h**
- RFAL_WUM_REFERENCE_AUTO : **rfal_rf.h**
- RFAL_WUM_STATE_ENABLED : **rfal_rf.h**
- RFAL_WUM_STATE_ENABLED_WOKE : **rfal_rf.h**
- RFAL_WUM_STATE_INITIALIZING : **rfal_rf.h**
- RFAL_WUM_STATE_NOT_INIT : **rfal_rf.h**
- RFAL_WUM_TRE_ABOVE : **st25r200/rfal_features.h**, **st25r500/rfal_features.h**
- RFAL_WUM_TRE_BELOW : **st25r200/rfal_features.h**, **st25r500/rfal_features.h**
- RFAL_WUM_TRE_BETWEEN : **st25r200/rfal_features.h**, **st25r500/rfal_features.h**
- rfalAdjustRegulators() : **rfal_rf.h**
- rfalAnalogConfigBitrate : **rfal_analogConfig.h**
- rfalAnalogConfigDirection : **rfal_analogConfig.h**
- rfalAnalogConfigGenModeID() : **rfal_analogConfig.h**
- rfalAnalogConfigId : **rfal_analogConfig.h**
- rfalAnalogConfigInitialize() : **rfal_analogConfig.h**
- rfalAnalogConfigIsReady() : **rfal_analogConfig.h**
- rfalAnalogConfigListRead() : **rfal_analogConfig.h**
- rfalAnalogConfigListReadRaw() : **rfal_analogConfig.h**
- rfalAnalogConfigListWrite() : **rfal_analogConfig.h**
- rfalAnalogConfigListWriteRaw() : **rfal_analogConfig.h**
- rfalAnalogConfigMode : **rfal_analogConfig.h**
- rfalAnalogConfigNum : **rfal_analogConfig.h**
- rfalAnalogConfigOffset : **rfal_analogConfig.h**
- rfalAnalogConfigRegAddr : **rfal_analogConfig.h**
- rfalAnalogConfigRegMask : **rfal_analogConfig.h**
- rfalAnalogConfigRegVal : **rfal_analogConfig.h**
- rfalAnalogConfigTech : **rfal_analogConfig.h**
- rfalBitRate : **rfal_rf.h**

- rfalCalibrate() : **rfal_rf.h**
- rfalCdDetectCard() : **rfal_cd.h**
- rfalCdDetType : **rfal_cd.h**
- rfalCdGetDetectCardStatus() : **rfal_cd.h**
- rfalCdStartDetectCard() : **rfal_cd.h**
- rfalCdTech : **rfal_cd.h**
- rfalChipChangeRegBits() : **rfal_chip.h**
- rfalChipChangeTestRegBits() : **rfal_chip.h**
- rfalChipExecCmd() : **rfal_chip.h**
- rfalChipGetLmFieldInd() : **rfal_chip.h**
- rfalChipGetLMMod() : **rfal_chip.h**
- rfalChipGetRFO() : **rfal_chip.h**
- rfalChipMeasureAmplitude() : **rfal_chip.h**
- rfalChipMeasureCapacitance() : **rfal_chip.h**
- rfalChipMeasureCombinedIQ() : **rfal_chip.h**
- rfalChipMeasureCurrent() : **rfal_chip.h**
- rfalChipMeasureI() : **rfal_chip.h**
- rfalChipMeasureIQ() : **rfal_chip.h**
- rfalChipMeasurePhase() : **rfal_chip.h**
- rfalChipMeasurePowerSupply() : **rfal_chip.h**
- rfalChipMeasureQ() : **rfal_chip.h**
- rfalChipReadReg() : **rfal_chip.h**
- rfalChipReadTestReg() : **rfal_chip.h**
- rfalChipSetAntennaMode() : **rfal_chip.h**
- rfalChipSetLMMod() : **rfal_chip.h**
- rfalChipSetRFO() : **rfal_chip.h**
- rfalChipWriteReg() : **rfal_chip.h**
- rfalChipWriteTestReg() : **rfal_chip.h**
- rfalComplianceMode : **rfal_rf.h**
- rfalConv1fcTo2018fc : **rfal_rf.h**
- rfalConv1fcTo4096fc : **rfal_rf.h**
- rfalConv1fcTo512fc : **rfal_rf.h**
- rfalConv1fcTo64fc : **rfal_rf.h**
- rfalConv1fcTo8fc : **rfal_rf.h**
- rfalConv1fcToMs : **rfal_rf.h**
- rfalConv1fcToUs : **rfal_rf.h**
- rfalConv2048fcTo1fc : **rfal_rf.h**
- rfalConv4096fcTo1fc : **rfal_rf.h**
- rfalConv512fcTo1fc : **rfal_rf.h**

- rfalConv64fcTo1fc : **rfal_rf.h**
- rfalConv64fcToMs : **rfal_rf.h**
- rfalConv8fcTo1fc : **rfal_rf.h**
- rfalConvBitsToBytes : **rfal_rf.h**
- rfalConvBytesToBits : **rfal_rf.h**
- rfalConvMsTo1fc : **rfal_rf.h**
- rfalConvMsTo64fc : **rfal_rf.h**
- rfalConvUsTo1fc : **rfal_rf.h**
- rfalCrcCalculateCcitt() : **rfal_crc.h**
- rfalCreateByteFlagsTxRxContext : **rfal_rf.h**
- rfalCreateByteTxRxContext : **rfal_rf.h**
- rfalDeinitialize() : **rfal_rf.h**
- rfalDisableObsvMode() : **rfal_rf.h**
- rfalDlmaAdjust() : **rfal_dlma.h**
- rfalDlmaAdjustFunc : **rfal_dlma.h**
- rfalDlmaDefaultSettings : **st25r3916/rfal_dlmaTbl.h**, **st25r500/rfal_dlmaTbl.h**
- rfalDlmaGetInfo() : **rfal_dlma.h**
- rfalDlmaInitialize() : **rfal_dlma.h**
- rfalDlmaIsEnabled() : **rfal_dlma.h**
- rfalDlmaMeasureFunc : **rfal_dlma.h**
- rfalDlmaSetAdjustCallback() : **rfal_dlma.h**
- rfalDlmaSetEnabled() : **rfal_dlma.h**
- rfalDlmaSetMeasureCallback() : **rfal_dlma.h**
- rfalDlmaTableRead() : **rfal_dlma.h**
- rfalDlmaTableWrite() : **rfal_dlma.h**
- rfalDpoAdjust() : **rfal_dpo.h**
- rfalDpoAdjustFunc : **rfal_dpo.h**
- rfalDpoGetInfo() : **rfal_dpo.h**
- rfalDpoInitialize() : **rfal_dpo.h**
- rfalDpoIsEnabled() : **rfal_dpo.h**
- rfalDpoMeasureFunc : **rfal_dpo.h**
- rfalDpoReqAdj() : **rfal_dpo.h**
- rfalDpoSetAdjustCallback() : **rfal_dpo.h**
- rfalDpoSetEnabled() : **rfal_dpo.h**
- rfalDpoSetMeasureCallback() : **rfal_dpo.h**
- rfalDpoTableRead() : **rfal_dpo.h**
- rfalDpoTableWrite() : **rfal_dpo.h**
- rfalEHandling : **rfal_rf.h**
- rfalFeliCaPoll() : **rfal_rf.h**

- rfalFeliCaPollRes : **rfal_rf.h**
- rfalFeliCaPollSlots : **rfal_rf.h**
- rfalFieldOff() : **rfal_rf.h**
- rfalFieldOnAndStartGT() : **rfal_rf.h**
- rfalGetBitRate() : **rfal_rf.h**
- rfalGetErrorHandling() : **rfal_rf.h**
- rfalGetFDTListen() : **rfal_rf.h**
- rfalGetFDTPoll() : **rfal_rf.h**
- rfalGetFeliCaPollStatus() : **rfal_rf.h**
- rfalGetGT() : **rfal_rf.h**
- rfalGetMaxBrAP2P : **rfal_rf.h**
- rfalGetMaxBrCEA : **rfal_rf.h**
- rfalGetMaxBrCEB : **rfal_rf.h**
- rfalGetMaxBrCEF : **rfal_rf.h**
- rfalGetMaxBrRW : **rfal_rf.h**
- rfalGetMode() : **rfal_rf.h**
- rfalGetObsvMode() : **rfal_rf.h**
- rfalGetTransceiveRSSI() : **rfal_rf.h**
- rfalGetTransceiveState() : **rfal_rf.h**
- rfalGetTransceiveStatus() : **rfal_rf.h**
- rfalInitialize() : **rfal_rf.h**
- rfalIsExtFieldOn() : **rfal_rf.h**
- rfalIsGTExpired() : **rfal_rf.h**
- rfalIsModeActiveComm : **rfal_rf.h**
- rfalIsModePassiveComm : **rfal_rf.h**
- rfalIsModePassiveListen : **rfal_rf.h**
- rfalIsModePassivePoll : **rfal_rf.h**
- rfalISO14443AGetTransceiveAnticollisionFrameStatus() : **rfal_rf.h**
- rfalISO14443AStartTransceiveAnticollisionFrame() : **rfal_rf.h**
- rfalISO14443ATransceiveAnticollisionFrame() : **rfal_rf.h**
- rfalISO14443ATransceiveShortFrame() : **rfal_rf.h**
- rfalIso15693PhyConfigure() : **rfal_iso15693_2.h**
- rfalIso15693PhyGetConfiguration() : **rfal_iso15693_2.h**
- rfalISO15693TransceiveAnticollisionFrame() : **rfal_rf.h**
- rfalISO15693TransceiveEOF() : **rfal_rf.h**
- rfalISO15693TransceiveEOFAnticollision() : **rfal_rf.h**
- rfalIso15693VCDCode() : **rfal_iso15693_2.h**
- rfalIso15693VcdCoding_t : **rfal_iso15693_2.h**
- rfalIso15693VICCDecode() : **rfal_iso15693_2.h**

- rfalIsoDepATTRIB() : **rfal_isoDep.h**
- rfalIsoDepDeselect() : **rfal_isoDep.h**
- rfalIsoDepFSx : **rfal_isoDep.h**
- rfalIsoDepFSxI : **rfal_isoDep.h**
- rfalIsoDepFSxI2FSx() : **rfal_isoDep.h**
- rfalIsoDepFWI2FWT() : **rfal_isoDep.h**
- rfalIsoDepGetApduTransceiveStatus() : **rfal_isoDep.h**
- rfalIsoDepGetDeselectStatus() : **rfal_isoDep.h**
- rfalIsoDepGetMaxInfLen() : **rfal_isoDep.h**
- rfalIsoDepGetTransceiveStatus() : **rfal_isoDep.h**
- rfalIsoDepInitialize() : **rfal_isoDep.h**
- rfalIsoDepInitializeWithParams() : **rfal_isoDep.h**
- rfalIsoDepIsAttrib() : **rfal_isoDep.h**
- rfalIsoDepIsRats() : **rfal_isoDep.h**
- rfalIsoDepListenGetActivationStatus() : **rfal_isoDep.h**
- rfalIsoDepListenStartActivation() : **rfal_isoDep.h**
- rfalIsoDepPollAGetActivationStatus() : **rfal_isoDep.h**
- rfalIsoDepPollAHandleActivation() : **rfal_isoDep.h**
- rfalIsoDepPollAStartActivation() : **rfal_isoDep.h**
- rfalIsoDepPollBGetActivationStatus() : **rfal_isoDep.h**
- rfalIsoDepPollBHandleActivation() : **rfal_isoDep.h**
- rfalIsoDepPollBStartActivation() : **rfal_isoDep.h**
- rfalIsoDepPollHandleSParameters() : **rfal_isoDep.h**
- rfalIsoDepPPS() : **rfal_isoDep.h**
- rfalIsoDepRATS() : **rfal_isoDep.h**
- rfalIsoDepStartApduTransceive() : **rfal_isoDep.h**
- rfalIsoDepStartDeselect() : **rfal_isoDep.h**
- rfalIsoDepStartTransceive() : **rfal_isoDep.h**
- rfalIsTransceiveInRx() : **rfal_rf.h**
- rfalIsTransceiveInTx() : **rfal_rf.h**
- rfalIsTransceiveSubcDetected() : **rfal_rf.h**
- rfalListenGetState() : **rfal_rf.h**
- rfalListenSetState() : **rfal_rf.h**
- rfalListenSleepStart() : **rfal_rf.h**
- rfalListenStart() : **rfal_rf.h**
- rfalListenStop() : **rfal_rf.h**
- rfalLmEonCallback : **rfal_rf.h**
- rfalLmNfcidLen : **rfal_rf.h**
- rfalLmState : **rfal_rf.h**

- rfalLogD : **rfal_rf.h**
- rfalLogE : **rfal_rf.h**
- rfalLogI : **rfal_rf.h**
- rfalLogW : **rfal_rf.h**
- rfalLowPowerModeStart() : **rfal_rf.h**
- rfalLowPowerModeStop() : **rfal_rf.h**
- rfalLpMode : **rfal_rf.h**
- rfalMode : **rfal_rf.h**
- rfalNfcaIsSelResNFCDEP : **rfal_nfca.h**
- rfalNfcaIsSelResT2T : **rfal_nfca.h**
- rfalNfcaIsSelResT4T : **rfal_nfca.h**
- rfalNfcaIsSelResT4TNFCDEP : **rfal_nfca.h**
- rfalNfcaIsSensResT1T : **rfal_nfca.h**
- rfalNfcaLisDevIsMultiProto : **rfal_nfca.h**
- rfalNfcaListenDeviceType : **rfal_nfca.h**
- rfalNfcaListenerIsSleepReq() : **rfal_nfca.h**
- rfalNfcaPollerCheckPresence() : **rfal_nfca.h**
- rfalNfcaPollerFullCollisionResolution() : **rfal_nfca.h**
- rfalNfcaPollerGetFullCollisionResolutionStatus() : **rfal_nfca.h**
- rfalNfcaPollerGetSelectStatus() : **rfal_nfca.h**
- rfalNfcaPollerGetSleepStatus() : **rfal_nfca.h**
- rfalNfcaPollerGetTechnologyDetectionStatus() : **rfal_nfca.h**
- rfalNfcaPollerInitialize() : **rfal_nfca.h**
- rfalNfcaPollerSelect() : **rfal_nfca.h**
- rfalNfcaPollerSingleCollisionResolution() : **rfal_nfca.h**
- rfalNfcaPollerSleep() : **rfal_nfca.h**
- rfalNfcaPollerSleepFullCollisionResolution() : **rfal_nfca.h**
- rfalNfcaPollerStartFullCollisionResolution() : **rfal_nfca.h**
- rfalNfcaPollerStartSelect() : **rfal_nfca.h**
- rfalNfcaPollerStartSleep() : **rfal_nfca.h**
- rfalNfcaPollerStartTechnologyDetection() : **rfal_nfca.h**
- rfalNfcaPollerTechnologyDetection() : **rfal_nfca.h**
- rfalNfcbGetFSCI : **rfal_nfcb.h**
- rfalNfcbIsIsoDepSupported : **rfal_nfcb.h**
- rfalNfcbPollerCheckPresence() : **rfal_nfcb.h**
- rfalNfcbPollerCollisionResolution() : **rfal_nfcb.h**
- rfalNfcbPollerGetCheckPresenceStatus() : **rfal_nfcb.h**
- rfalNfcbPollerGetCollisionResolutionStatus() : **rfal_nfcb.h**
- rfalNfcbPollerGetSlotMarkerStatus() : **rfal_nfcb.h**

- rfalNfcbPollerGetTechnologyDetectionStatus() : **rfal_nfcb.h**
- rfalNfcbPollerInitialize() : **rfal_nfcb.h**
- rfalNfcbPollerInitializeWithParams() : **rfal_nfcb.h**
- rfalNfcbPollerSleep() : **rfal_nfcb.h**
- rfalNfcbPollerSlotMarker() : **rfal_nfcb.h**
- rfalNfcbPollerSlottedCollisionResolution() : **rfal_nfcb.h**
- rfalNfcbPollerStartCheckPresence() : **rfal_nfcb.h**
- rfalNfcbPollerStartCollisionResolution() : **rfal_nfcb.h**
- rfalNfcbPollerStartSlotMarker() : **rfal_nfcb.h**
- rfalNfcbPollerStartSlottedCollisionResolution() : **rfal_nfcb.h**
- rfalNfcbPollerStartTechnologyDetection() : **rfal_nfcb.h**
- rfalNfcbPollerTechnologyDetection() : **rfal_nfcb.h**
- rfalNfcbSensCmd : **rfal_nfcb.h**
- rfalNfcbSlots : **rfal_nfcb.h**
- rfalNfcbTR2ToFDT() : **rfal_nfcb.h**
- rfalNfcDataExchangeGetStatus() : **rfal_nfc.h**
- rfalNfcDataExchangeStart() : **rfal_nfc.h**
- rfalNfcDeactivate() : **rfal_nfc.h**
- rfalNfcDeactivateType : **rfal_nfc.h**
- rfalNfcDefaultDiscParams : **rfal_nfc.h**
- rfalNfcDepATR() : **rfal_nfcDep.h**
- rfalNfcDepBRS2DRI : **rfal_nfcDep.h**
- rfalNfcDepBRS2DSI : **rfal_nfcDep.h**
- rfalNfcDepCalculateRWT() : **rfal_nfcDep.h**
- rfalNfcDepCommMode : **rfal_nfcDep.h**
- rfalNfcDepDeactCallback : **rfal_nfcDep.h**
- rfalNfcDepDSL() : **rfal_nfcDep.h**
- rfalNfcDepDx2BRS : **rfal_nfcDep.h**
- rfalNfcDepGetPduTransceiveStatus() : **rfal_nfcDep.h**
- rfalNfcDepGetTransceiveStatus() : **rfal_nfcDep.h**
- rfalNfcDepInitialize() : **rfal_nfcDep.h**
- rfalNfcDepInitiatorHandleActivation() : **rfal_nfcDep.h**
- rfalNfcDepIsAtrReq() : **rfal_nfcDep.h**
- rfalNfcDepListenGetActivationStatus() : **rfal_nfcDep.h**
- rfalNfcDepListenStartActivation() : **rfal_nfcDep.h**
- rfalNfcDepLR2FS : **rfal_nfcDep.h**
- rfalNfcDepLR2PP : **rfal_nfcDep.h**
- rfalNfcDepPP2LR : **rfal_nfcDep.h**
- rfalNfcDepPSL() : **rfal_nfcDep.h**

- rfalNfcDepRLS() : **rfal_nfcDep.h**
- rfalNfcDepRole : **rfal_nfcDep.h**
- rfalNfcDepSetDeactivatingCallback() : **rfal_nfcDep.h**
- rfalNfcDepSetNFCID : **rfal_nfcDep.h**
- rfalNfcDepStartPduTransceive() : **rfal_nfcDep.h**
- rfalNfcDepStartTransceive() : **rfal_nfcDep.h**
- rfalNfcDepTargetRcvdATR() : **rfal_nfcDep.h**
- rfalNfcDepWT2RWT : **rfal_nfcDep.h**
- rfalNfcDevType : **rfal_nfc.h**
- rfalNfcDiscover() : **rfal_nfc.h**
- rfalNfcfIsNfcDepSupported : **rfal_nfcf.h**
- rfalNfcfListenerIsT3TReq() : **rfal_nfcf.h**
- rfalNfcfPollerCheck() : **rfal_nfcf.h**
- rfalNfcfPollerCheckPresence() : **rfal_nfcf.h**
- rfalNfcfPollerCollisionResolution() : **rfal_nfcf.h**
- rfalNfcfPollerGetCheckPresenceStatus() : **rfal_nfcf.h**
- rfalNfcfPollerGetCollisionResolutionStatus() : **rfal_nfcf.h**
- rfalNfcfPollerInitialize() : **rfal_nfcf.h**
- rfalNfcfPollerPoll() : **rfal_nfcf.h**
- rfalNfcfPollerStartCheckPresence() : **rfal_nfcf.h**
- rfalNfcfPollerStartCollisionResolution() : **rfal_nfcf.h**
- rfalNfcfPollerUpdate() : **rfal_nfcf.h**
- rfalNfcfServ : **rfal_nfcf.h**
- rfalNfcGetActiveDevice() : **rfal_nfc.h**
- rfalNfcGetDevicesFound() : **rfal_nfc.h**
- rfalNfcGetState() : **rfal_nfc.h**
- rfalNfcInitialize() : **rfal_nfc.h**
- rfalNfcIsDevActivated : **rfal_nfc.h**
- rfalNfcIsInDiscovery : **rfal_nfc.h**
- rfalNfcIsRemDevListener : **rfal_nfc.h**
- rfalNfcIsRemDevPoller : **rfal_nfc.h**
- rfalNfcPropCallback : **rfal_nfc.h**
- rfalNfcRfInterface : **rfal_nfc.h**
- rfalNfcSelect() : **rfal_nfc.h**
- rfalNfcState : **rfal_nfc.h**
- rfalNfcvNumSlots : **rfal_nfcv.h**
- rfalNfcvPollerCheckPresence() : **rfal_nfcv.h**
- rfalNfcvPollerCollisionResolution() : **rfal_nfcv.h**
- rfalNfcvPollerExtendedGetSystemInformation() : **rfal_nfcv.h**

- rfalNfcvPollerExtendedLockSingleBlock() : **rfal_nfcv.h**
- rfalNfcvPollerExtendedReadMultipleBlocks() : **rfal_nfcv.h**
- rfalNfcvPollerExtendedReadSingleBlock() : **rfal_nfcv.h**
- rfalNfcvPollerExtendedWriteMultipleBlocks() : **rfal_nfcv.h**
- rfalNfcvPollerExtendedWriteSingleBlock() : **rfal_nfcv.h**
- rfalNfcvPollerGetSystemInformation() : **rfal_nfcv.h**
- rfalNfcvPollerInitialize() : **rfal_nfcv.h**
- rfalNfcvPollerInventory() : **rfal_nfcv.h**
- rfalNfcvPollerLockBlock() : **rfal_nfcv.h**
- rfalNfcvPollerReadMultipleBlocks() : **rfal_nfcv.h**
- rfalNfcvPollerReadSingleBlock() : **rfal_nfcv.h**
- rfalNfcvPollerSelect() : **rfal_nfcv.h**
- rfalNfcvPollerSleep() : **rfal_nfcv.h**
- rfalNfcvPollerSleepCollisionResolution() : **rfal_nfcv.h**
- rfalNfcvPollerTransceiveReq() : **rfal_nfcv.h**
- rfalNfcvPollerWriteMultipleBlocks() : **rfal_nfcv.h**
- rfalNfcvPollerWriteSingleBlock() : **rfal_nfcv.h**
- rfalNfcWorker() : **rfal_nfc.h**
- rfalPostTxRxCallback : **rfal_rf.h**
- rfalPreTxRxCallback : **rfal_rf.h**
- rfalRunBlocking : **rfal_rf.h**
- rfalSetAnalogConfig() : **rfal_analogConfig.h**
- rfalSetBitRate() : **rfal_rf.h**
- rfalSetErrorHandling() : **rfal_rf.h**
- rfalSetFDTListen() : **rfal_rf.h**
- rfalSetFDTPoll() : **rfal_rf.h**
- rfalSetGT() : **rfal_rf.h**
- rfalSetLmEonCallback() : **rfal_rf.h**
- rfalSetMode() : **rfal_rf.h**
- rfalSetObsvMode() : **rfal_rf.h**
- rfalSetPostTxRxCallback() : **rfal_rf.h**
- rfalSetPreTxRxCallback() : **rfal_rf.h**
- rfalSetRegulators() : **rfal_rf.h**
- rfalSetSyncTxRxCallback() : **rfal_rf.h**
- rfalSetUpperLayerCallback() : **rfal_rf.h**
- rfalSt25tbBlock : **rfal_st25tb.h**
- rfalSt25tbPollerCheckPresence() : **rfal_st25tb.h**
- rfalSt25tbPollerCollisionResolution() : **rfal_st25tb.h**
- rfalSt25tbPollerCompletion() : **rfal_st25tb.h**

- rfalSt25tbPollerGetUID() : **rfal_st25tb.h**
- rfalSt25tbPollerInitialize() : **rfal_st25tb.h**
- rfalSt25tbPollerInitiate() : **rfal_st25tb.h**
- rfalSt25tbPollerPcall() : **rfal_st25tb.h**
- rfalSt25tbPollerReadBlock() : **rfal_st25tb.h**
- rfalSt25tbPollerResetToInventory() : **rfal_st25tb.h**
- rfalSt25tbPollerSelect() : **rfal_st25tb.h**
- rfalSt25tbPollerSlotMarker() : **rfal_st25tb.h**
- rfalSt25tbPollerWriteBlock() : **rfal_st25tb.h**
- rfalSt25tbUID : **rfal_st25tb.h**
- rfalST25TV512CPollerReadConfiguration() : **rfal_st25xv.h**
- rfalST25TV512CPollerWriteConfiguration() : **rfal_st25xv.h**
- rfalST25xVPollerFastExtendedReadSingleBlock() : **rfal_st25xv.h**
- rfalST25xVPollerFastExtReadMultipleBlocks() : **rfal_st25xv.h**
- rfalST25xVPollerFastReadDynamicConfiguration() : **rfal_st25xv.h**
- rfalST25xVPollerFastReadMessage() : **rfal_st25xv.h**
- rfalST25xVPollerFastReadMsgLength() : **rfal_st25xv.h**
- rfalST25xVPollerFastReadMultipleBlocks() : **rfal_st25xv.h**
- rfalST25xVPollerFastReadSingleBlock() : **rfal_st25xv.h**
- rfalST25xVPollerFastWriteDynamicConfiguration() : **rfal_st25xv.h**
- rfalST25xVPollerFastWriteMessage() : **rfal_st25xv.h**
- rfalST25xVPollerGetRandomNumber() : **rfal_st25xv.h**
- rfalST25xVPollerM24LRFastReadMultipleBlocks() : **rfal_st25xv.h**
- rfalST25xVPollerM24LRFastReadSingleBlock() : **rfal_st25xv.h**
- rfalST25xVPollerM24LRReadMultipleBlocks() : **rfal_st25xv.h**
- rfalST25xVPollerM24LRReadSingleBlock() : **rfal_st25xv.h**
- rfalST25xVPollerM24LRWriteSingleBlock() : **rfal_st25xv.h**
- rfalST25xVPollerPresentPassword() : **rfal_st25xv.h**
- rfalST25xVPollerReadConfiguration() : **rfal_st25xv.h**
- rfalST25xVPollerReadDynamicConfiguration() : **rfal_st25xv.h**
- rfalST25xVPollerReadMessage() : **rfal_st25xv.h**
- rfalST25xVPollerReadMessageLength() : **rfal_st25xv.h**
- rfalST25xVPollerWriteConfiguration() : **rfal_st25xv.h**
- rfalST25xVPollerWriteDynamicConfiguration() : **rfal_st25xv.h**
- rfalST25xVPollerWriteMessage() : **rfal_st25xv.h**
- rfalST25xVPollerWritePassword() : **rfal_st25xv.h**
- rfalStartFeliCaPoll() : **rfal_rf.h**
- rfalStartTransceive() : **rfal_rf.h**
- rfalState : **rfal_rf.h**

- rfalSyncTxRxCallback : **rfal_rf.h**
- rfalT1Tcmds : **rfal_t1t.h**
- rfalT1TPollerInitialize() : **rfal_t1t.h**
- rfalT1TPollerRall() : **rfal_t1t.h**
- rfalT1TPollerRid() : **rfal_t1t.h**
- rfalT1TPollerWrite() : **rfal_t1t.h**
- rfalT2TPollerRead() : **rfal_t2t.h**
- rfalT2TPollerSectorSelect() : **rfal_t2t.h**
- rfalT2TPollerWrite() : **rfal_t2t.h**
- rfalT4tCmds : **rfal_t4t.h**
- rfalT4TPollerComposeCAPDU() : **rfal_t4t.h**
- rfalT4TPollerComposeReadData() : **rfal_t4t.h**
- rfalT4TPollerComposeReadDataODO() : **rfal_t4t.h**
- rfalT4TPollerComposeSelectAppl() : **rfal_t4t.h**
- rfalT4TPollerComposeSelectFile() : **rfal_t4t.h**
- rfalT4TPollerComposeSelectFileV1Mapping() : **rfal_t4t.h**
- rfalT4TPollerComposeWriteData() : **rfal_t4t.h**
- rfalT4TPollerComposeWriteDataODO() : **rfal_t4t.h**
- rfalT4TPollerParseRAPDU() : **rfal_t4t.h**
- rfalTransceiveBlockingRx() : **rfal_rf.h**
- rfalTransceiveBlockingTx() : **rfal_rf.h**
- rfalTransceiveBlockingTxRx() : **rfal_rf.h**
- rfalTransceiveState : **rfal_rf.h**
- rfalUpperLayerCallback : **rfal_rf.h**
- rfalWakeUpConfig : **st25r200/rfal_features.h**
- rfalWakeUpModeGetInfo() : **rfal_rf.h**
- rfalWakeUpModeHasWoke() : **rfal_rf.h**
- rfalWakeUpModeIsEnabled() : **rfal_rf.h**
- rfalWakeUpModeStart() : **rfal_rf.h**
- rfalWakeUpModeStop() : **rfal_rf.h**
- rfalWlcPWptIsFodDetected() : **rfal_rf.h**
- rfalWlcPWptIsStopDetected() : **rfal_rf.h**
- rfalWlcPWptMonitorStart() : **rfal_rf.h**
- rfalWlcPWptMonitorStop() : **rfal_rf.h**
- rfalWorker() : **rfal_rf.h**
- rfalWumAAWeight : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**
- rfalWumMeasDuration : **st25r200/rfal_features.h**, **st25r500/rfal_features.h**
- rfalWumMeasFilter : **st25r200/rfal_features.h**, **st25r500/rfal_features.h**

- rfalWumPeriod : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- rfalWumState : **rfal_rf.h**

Generated on Fri Mar 14 2025 11:45:12 for RFAL by doxygen 1.9.5

RFAL

Main Page

Modules

Data Structures

Examples

File List

Functions

Variables

Typedefs

Enumerations

Enumerator

Macros

g

i

m

n

p

r

Here is a list of all documented functions, variables, defines, enums, and typedefs with links to the documentation:

- s -

- ST25R200_BR_106_26 : **st25r200.h**
- ST25R200_BR_212 : **st25r200.h**
- ST25R200_BR_424_53 : **st25r200.h**
- ST25R200_BR_848 : **st25r200.h**
- ST25R200_BR_DO_NOT_SET : **st25r200.h**
- ST25R200_CMD_ADJUST_REGULATORS : **st25r200.h**
- ST25R200_CMD_CALIBRATE_WU : **st25r200.h**
- ST25R200_CMD_CLEAR_FIFO : **st25r200.h**
- ST25R200_CMD_CLEAR_RXGAIN : **st25r200.h**
- ST25R200_CMD_CLEAR_WU_CALIB : **st25r200.h**
- ST25R200_CMD_LEN : **st25r200.h**
- ST25R200_CMD_MASK_RECEIVE_DATA : **st25r200.h**
- ST25R200_CMD_MEASURE_IQ : **st25r200.h**
- ST25R200_CMD_MEASURE_WU : **st25r200.h**
- ST25R200_CMD_MODE : **st25r200.h**
- ST25R200_CMD_SENSE_RF : **st25r200.h**
- ST25R200_CMD_SET_DEFAULT : **st25r200.h**
- ST25R200_CMD_START_GP_TIMER : **st25r200.h**
- ST25R200_CMD_START_MRT : **st25r200.h**
- ST25R200_CMD_START_NRT : **st25r200.h**
- ST25R200_CMD_START_WUT : **st25r200.h**
- ST25R200_CMD_STOP : **st25r200.h**
- ST25R200_CMD_STOP_NRT : **st25r200.h**
- ST25R200_CMD_TEST_ACCESS : **st25r200.h**
- ST25R200_CMD_TRANSMIT : **st25r200.h**
- ST25R200_CMD_TRANSMIT_EOF : **st25r200.h**
- ST25R200_CMD_TRANSPARENT_MODE : **st25r200.h**
- ST25R200_CMD_UNMASK_RECEIVE_DATA : **st25r200.h**
- ST25R200_FIFO_ACCESS : **st25r200.h**
- ST25R200_FIFO_DEPTH : **st25r200.h**

- ST25R200_FIFO_STATUS_LEN : **st25r200_com.h**
- ST25R200_IRQ_MASK_ALL : **st25r200_irq.h**
- ST25R200_IRQ_MASK_COL : **st25r200_irq.h**
- ST25R200_IRQ_MASK_CRC : **st25r200_irq.h**
- ST25R200_IRQ_MASK_DCT : **st25r200_irq.h**
- ST25R200_IRQ_MASK_GPE : **st25r200_irq.h**
- ST25R200_IRQ_MASK_HFE : **st25r200_irq.h**
- ST25R200_IRQ_MASK_NONE : **st25r200_irq.h**
- ST25R200_IRQ_MASK_NRE : **st25r200_irq.h**
- ST25R200_IRQ_MASK_OSC : **st25r200_irq.h**
- ST25R200_IRQ_MASK_PAR : **st25r200_irq.h**
- ST25R200_IRQ_MASK_RFU1 : **st25r200_irq.h**
- ST25R200_IRQ_MASK_RFU2 : **st25r200_irq.h**
- ST25R200_IRQ_MASK_RFU3 : **st25r200_irq.h**
- ST25R200_IRQ_MASK_RFU4 : **st25r200_irq.h**
- ST25R200_IRQ_MASK_RFU5 : **st25r200_irq.h**
- ST25R200_IRQ_MASK_RFU6 : **st25r200_irq.h**
- ST25R200_IRQ_MASK_RX_REST : **st25r200_irq.h**
- ST25R200_IRQ_MASK_RXE : **st25r200_irq.h**
- ST25R200_IRQ_MASK_RXS : **st25r200_irq.h**
- ST25R200_IRQ_MASK_SFE : **st25r200_irq.h**
- ST25R200_IRQ_MASK_SUBC_START : **st25r200_irq.h**
- ST25R200_IRQ_MASK_TXE : **st25r200_irq.h**
- ST25R200_IRQ_MASK_WL : **st25r200_irq.h**
- ST25R200_IRQ_MASK_WUI : **st25r200_irq.h**
- ST25R200_IRQ_MASK_WUQ : **st25r200_irq.h**
- ST25R200_IRQ_MASK_WUT : **st25r200_irq.h**
- ST25R200_OBS_MODE_LEN : **st25r200_com.h**
- ST25R200_READ_MODE : **st25r200.h**
- ST25R200_REG_COLLISION : **st25r200_com.h**
- ST25R200_REG_CORR1 : **st25r200_com.h**
- ST25R200_REG_CORR2 : **st25r200_com.h**
- ST25R200_REG_CORR3 : **st25r200_com.h**
- ST25R200_REG_CORR4 : **st25r200_com.h**
- ST25R200_REG_CORR5 : **st25r200_com.h**
- ST25R200_REG_CORR6 : **st25r200_com.h**
- ST25R200_REG_DISPLAY1 : **st25r200_com.h**
- ST25R200_REG_DISPLAY2 : **st25r200_com.h**
- ST25R200_REG_DISPLAY3 : **st25r200_com.h**

- ST25R200_REG_DISPLAY4 : **st25r200_com.h**
- ST25R200_REG_DROP_200 : **st25r200.h**
- ST25R200_REG_DROP_250 : **st25r200.h**
- ST25R200_REG_DROP_300 : **st25r200.h**
- ST25R200_REG_DROP_350 : **st25r200.h**
- ST25R200_REG_DROP_400 : **st25r200.h**
- ST25R200_REG_DROP_450 : **st25r200.h**
- ST25R200_REG_DROP_500 : **st25r200.h**
- ST25R200_REG_DROP_550 : **st25r200.h**
- ST25R200_REG_DROP_DO_NOT_SET : **st25r200.h**
- ST25R200_REG_EMD1 : **st25r200_com.h**
- ST25R200_REG_EMD2 : **st25r200_com.h**
- ST25R200_REG_FIFO_STATUS1 : **st25r200_com.h**
- ST25R200_REG_FIFO_STATUS2 : **st25r200_com.h**
- ST25R200_REG_GENERAL : **st25r200_com.h**
- ST25R200_REG_GPT1 : **st25r200_com.h**
- ST25R200_REG_GPT2 : **st25r200_com.h**
- ST25R200_REG_IC_ID : **st25r200_com.h**
- ST25R200_REG_IRQ1 : **st25r200_com.h**
- ST25R200_REG_IRQ2 : **st25r200_com.h**
- ST25R200_REG_IRQ3 : **st25r200_com.h**
- ST25R200_REG_IRQ_MASK1 : **st25r200_com.h**
- ST25R200_REG_IRQ_MASK2 : **st25r200_com.h**
- ST25R200_REG_IRQ_MASK3 : **st25r200_com.h**
- ST25R200_REG_LEN : **st25r200.h**
- ST25R200_REG_MRT : **st25r200_com.h**
- ST25R200_REG_MRT_SQT_CONF : **st25r200_com.h**
- ST25R200_REG_NRT1 : **st25r200_com.h**
- ST25R200_REG_NRT2 : **st25r200_com.h**
- ST25R200_REG_NRT_GPT_CONF : **st25r200_com.h**
- ST25R200_REG_OPERATION : **st25r200_com.h**
- ST25R200_REG_OVERSHOOT_CONF : **st25r200_com.h**
- ST25R200_REG_OVUNSHOOT_CONF : **st25r200_com.h**
- ST25R200_REG_PROTOCOL : **st25r200_com.h**
- ST25R200_REG_PROTOCOL_RX1 : **st25r200_com.h**
- ST25R200_REG_PROTOCOL_RX2 : **st25r200_com.h**
- ST25R200_REG_PROTOCOL_RX3 : **st25r200_com.h**
- ST25R200_REG_PROTOCOL_TX1 : **st25r200_com.h**
- ST25R200_REG_PROTOCOL_TX2 : **st25r200_com.h**

- ST25R200_REG_PROTOCOL_TX3 : **st25r200_com.h**
- ST25R200_REG_REGULATOR : **st25r200_com.h**
- ST25R200_REG_RX_ANA1 : **st25r200_com.h**
- ST25R200_REG_RX_ANA2 : **st25r200_com.h**
- ST25R200_REG_RX_DIG : **st25r200_com.h**
- ST25R200_REG_SQT : **st25r200_com.h**
- ST25R200_REG_STATUS : **st25r200_com.h**
- ST25R200_REG_TX_DRIVER : **st25r200_com.h**
- ST25R200_REG_TX_FRAME1 : **st25r200_com.h**
- ST25R200_REG_TX_FRAME2 : **st25r200_com.h**
- ST25R200_REG_TX_MOD : **st25r200_com.h**
- ST25R200_REG_TX_RES_MOD : **st25r200_com.h**
- ST25R200_REG_UNDERSHOOT_CONF : **st25r200_com.h**
- ST25R200_REG_WAKEUP_CONF1 : **st25r200_com.h**
- ST25R200_REG_WAKEUP_CONF2 : **st25r200_com.h**
- ST25R200_REG_WU_I_ADC : **st25r200_com.h**
- ST25R200_REG_WU_I_CAL : **st25r200_com.h**
- ST25R200_REG_WU_I_CONF : **st25r200_com.h**
- ST25R200_REG_WU_I_DELTA : **st25r200_com.h**
- ST25R200_REG_WU_I_REF : **st25r200_com.h**
- ST25R200_REG_WU_Q_ADC : **st25r200_com.h**
- ST25R200_REG_WU_Q_CAL : **st25r200_com.h**
- ST25R200_REG_WU_Q_CONF : **st25r200_com.h**
- ST25R200_REG_WU_Q_DELTA : **st25r200_com.h**
- ST25R200_REG_WU_Q_REF : **st25r200_com.h**
- ST25R200_WRITE_MODE : **st25r200.h**
- st25r200AdjustRegulators() : **st25r200.h**
- st25r200CalibrateWU() : **st25r200.h**
- st25r200ChangeRegisterBits() : **st25r200_com.h**
- st25r200ChangeTestRegisterBits() : **st25r200_com.h**
- st25r200CheckChipID() : **st25r200.h**
- st25r200CheckForReceivedInterrupts() : **st25r200_irq.h**
- st25r200CheckReg() : **st25r200_com.h**
- st25r200ClearAndEnableInterrupts() : **st25r200_irq.h**
- st25r200ClearCalibration() : **st25r200.h**
- st25r200ClearInterrupts() : **st25r200_irq.h**
- st25r200ClrRegisterBits() : **st25r200_com.h**
- st25r200Deinitialize() : **st25r200.h**
- st25r200DisableInterrupts() : **st25r200_irq.h**

- st25r200EnableInterrupts() : **st25r200_irq.h**
- st25r200ExecuteCommand() : **st25r200_com.h**
- st25r200ExecuteCommandAndGetResult() : **st25r200.h**
- st25r200ExecuteCommands() : **st25r200_com.h**
- st25r200GetInterrupt() : **st25r200_irq.h**
- st25r200GetNoResponseTime() : **st25r200.h**
- st25r200GetNumFIFOBytes() : **st25r200.h**
- st25r200GetNumFIFOLastBits() : **st25r200.h**
- st25r200GetRegsDump() : **st25r200.h**
- st25r200GetRSSI() : **st25r200.h**
- st25r200Initialize() : **st25r200.h**
- st25r200InitInterrupts() : **st25r200_irq.h**
- st25r200IRQCallbackRestore() : **st25r200_irq.h**
- st25r200IRQCallbackSet() : **st25r200_irq.h**
- st25r200IsCmdValid() : **st25r200.h**
- st25r200IsExtFieldOn() : **st25r200.h**
- st25r200IsGPTRunning : **st25r200.h**
- st25r200IsLastFIFOComplete : **st25r200.h**
- st25r200IsNRTinEMV : **st25r200.h**
- st25r200IsOscOn : **st25r200.h**
- st25r200Isr() : **st25r200_irq.h**
- st25r200IsRegValid() : **st25r200_com.h**
- st25r200IsTxEnabled : **st25r200.h**
- st25r200IsTxOn : **st25r200.h**
- st25r200ledEvtCmd() : **st25r200_com.h**
- st25r200ledEvtCmds() : **st25r200_com.h**
- st25r200ledEvtIrq() : **st25r200_com.h**
- st25r200ledEvtWrMultiReg() : **st25r200_com.h**
- st25r200ledEvtWrReg() : **st25r200_com.h**
- st25r200ledInit() : **st25r200_com.h**
- st25r200MeasureCombinedIQ() : **st25r200.h**
- st25r200MeasureI() : **st25r200.h**
- st25r200MeasureIQ() : **st25r200.h**
- st25r200MeasureQ() : **st25r200.h**
- st25r200MeasureWU() : **st25r200.h**
- st25r200ModifyInterrupts() : **st25r200_irq.h**
- st25r200ModifyRegister() : **st25r200_com.h**
- st25r200OscOn() : **st25r200.h**
- st25r200PerformCollisionAvoidance() : **st25r200.h**

- st25r200ReadFifo() : **st25r200_com.h**
- st25r200ReadMultipleRegisters() : **st25r200_com.h**
- st25r200ReadRegister() : **st25r200_com.h**
- st25r200ReadTestRegister() : **st25r200_com.h**
- st25r200SetAntennaMode() : **st25r200.h**
- st25r200SetBitrate() : **st25r200.h**
- st25r200SetGPTime() : **st25r200.h**
- st25r200SetNoResponseTime() : **st25r200.h**
- st25r200SetNumTxBits() : **st25r200.h**
- st25r200SetRegisterBits() : **st25r200_com.h**
- st25r200SetRegulators() : **st25r200.h**
- st25r200SetStartGPTimer() : **st25r200.h**
- st25r200SetStartNoResponseTimer() : **st25r200.h**
- st25r200TxOff : **st25r200.h**
- st25r200TxRxOff : **st25r200.h**
- st25r200TxRxOn : **st25r200.h**
- st25r200WaitForInterruptsTimed() : **st25r200_irq.h**
- st25r200WriteFifo() : **st25r200_com.h**
- st25r200WriteMultipleRegisters() : **st25r200_com.h**
- st25r200WriteMultipleTestRegister() : **st25r200_com.h**
- st25r200WriteRegister() : **st25r200_com.h**
- st25r200WriteTestRegister() : **st25r200_com.h**
- ST25R3911_BR_106 : **st25r3911.h**
- ST25R3911_BR_1695 : **st25r3911.h**
- ST25R3911_BR_212 : **st25r3911.h**
- ST25R3911_BR_3390 : **st25r3911.h**
- ST25R3911_BR_424 : **st25r3911.h**
- ST25R3911_BR_6780 : **st25r3911.h**
- ST25R3911_BR_848 : **st25r3911.h**
- ST25R3911_BR_DO_NOT_SET : **st25r3911.h**
- ST25R3911_CMD_ADJUST_REGULATORS : **st25r3911.h**
- ST25R3911_CMD_ANALOG_PRESET : **st25r3911.h**
- ST25R3911_CMD_CALIBRATE_ANTENNA : **st25r3911.h**
- ST25R3911_CMD_CALIBRATE_C_SENSOR : **st25r3911.h**
- ST25R3911_CMD_CALIBRATE_MODULATION : **st25r3911.h**
- ST25R3911_CMD_CLEAR_FIFO : **st25r3911.h**
- ST25R3911_CMD_CLEAR_RSSI : **st25r3911.h**
- ST25R3911_CMD_CLEAR_SQUELCH : **st25r3911.h**
- ST25R3911_CMD_FUSE_PPROM : **st25r3911.h**

- ST25R3911_CMD_INITIAL_RF_COLLISION : **st25r3911.h**
- ST25R3911_CMD_LOAD_PPROM : **st25r3911.h**
- ST25R3911_CMD_MASK_RECEIVE_DATA : **st25r3911.h**
- ST25R3911_CMD_MEASURE_AMPLITUDE : **st25r3911.h**
- ST25R3911_CMD_MEASURE_CAPACITANCE : **st25r3911.h**
- ST25R3911_CMD_MEASURE_PHASE : **st25r3911.h**
- ST25R3911_CMD_MEASURE_VDD : **st25r3911.h**
- ST25R3911_CMD_NORMAL_NFC_MODE : **st25r3911.h**
- ST25R3911_CMD_RESPONSE_RF_COLLISION_0 : **st25r3911.h**
- ST25R3911_CMD_RESPONSE_RF_COLLISION_N : **st25r3911.h**
- ST25R3911_CMD_SET_DEFAULT : **st25r3911.h**
- ST25R3911_CMD_SQUELCH : **st25r3911.h**
- ST25R3911_CMD_START_GP_TIMER : **st25r3911.h**
- ST25R3911_CMD_START_MASK_RECEIVE_TIMER : **st25r3911.h**
- ST25R3911_CMD_START_NO_RESPONSE_TIMER : **st25r3911.h**
- ST25R3911_CMD_START_WUP_TIMER : **st25r3911.h**
- ST25R3911_CMD_TEST_ACCESS : **st25r3911.h**
- ST25R3911_CMD_TEST_CLEARA : **st25r3911.h**
- ST25R3911_CMD_TEST_CLEARB : **st25r3911.h**
- ST25R3911_CMD_TRANSMIT_REQA : **st25r3911.h**
- ST25R3911_CMD_TRANSMIT_WITH_CRC : **st25r3911.h**
- ST25R3911_CMD_TRANSMIT_WITHOUT_CRC : **st25r3911.h**
- ST25R3911_CMD_TRANSMIT_WUPA : **st25r3911.h**
- ST25R3911_CMD_TRANSPARENT_MODE : **st25r3911.h**
- ST25R3911_CMD_UNMASK_RECEIVE_DATA : **st25r3911.h**
- ST25R3911_FDT_NONE : **st25r3911.h**
- ST25R3911_FIFO_DEPTH : **st25r3911.h**
- ST25R3911_FIFO_STATUS_LEN : **st25r3911_com.h**
- ST25R3911_IRQ_MASK_ALL : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_CAC : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_CAT : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_COL : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_CRC : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_DCT : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_EOF : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_EON : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_ERR : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_ERR1 : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_ERR2 : **st25r3911_interrupt.h**

- ST25R3911_IRQ_MASK_FWL : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_GPE : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_NFCT : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_NONE : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_NRE : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_OSC : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_PAR : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_RXE : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_RXS : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_TIM : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_TXE : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_WAM : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_WCAP : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_WPH : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_WT : **st25r3911_interrupt.h**
- ST25R3911_REG_AD_RESULT : **st25r3911_com.h**
- ST25R3911_REG_AM_MOD_DEPTH_CONTROL : **st25r3911_com.h**
- ST25R3911_REG_AM_MOD_DEPTH_RESULT : **st25r3911_com.h**
- ST25R3911_REG_AMPLITUDE_MEASURE_AA_RESULT : **st25r3911_com.h**
- ST25R3911_REG_AMPLITUDE_MEASURE_CONF : **st25r3911_com.h**
- ST25R3911_REG_AMPLITUDE_MEASURE_REF : **st25r3911_com.h**
- ST25R3911_REG_AMPLITUDE_MEASURE_RESULT : **st25r3911_com.h**
- ST25R3911_REG_ANT_CAL_CONTROL : **st25r3911_com.h**
- ST25R3911_REG_ANT_CAL_RESULT : **st25r3911_com.h**
- ST25R3911_REG_ANT_CAL_TARGET : **st25r3911_com.h**
- ST25R3911_REG_AUX : **st25r3911_com.h**
- ST25R3911_REG_AUX_DISPLAY : **st25r3911_com.h**
- ST25R3911_REG_BIT_RATE : **st25r3911_com.h**
- ST25R3911_REG_CAP_SENSOR_CONTROL : **st25r3911_com.h**
- ST25R3911_REG_CAP_SENSOR_RESULT : **st25r3911_com.h**
- ST25R3911_REG_CAPACITANCE_MEASURE_AA_RESULT : **st25r3911_com.h**
- ST25R3911_REG_CAPACITANCE_MEASURE_CONF : **st25r3911_com.h**
- ST25R3911_REG_CAPACITANCE_MEASURE_REF : **st25r3911_com.h**
- ST25R3911_REG_CAPACITANCE_MEASURE_RESULT : **st25r3911_com.h**
- ST25R3911_REG_COLLISION_STATUS : **st25r3911_com.h**
- ST25R3911_REG_FIELD_THRESHOLD : **st25r3911_com.h**
- ST25R3911_REG_FIFO_RX_STATUS1 : **st25r3911_com.h**
- ST25R3911_REG_FIFO_RX_STATUS2 : **st25r3911_com.h**
- ST25R3911_REG_GAIN_RED_STATE : **st25r3911_com.h**

- ST25R3911_REG_GPT1 : **st25r3911_com.h**
- ST25R3911_REG_GPT2 : **st25r3911_com.h**
- ST25R3911_REG_GPT_CONTROL : **st25r3911_com.h**
- ST25R3911_REG_IC_IDENTITY : **st25r3911_com.h**
- ST25R3911_REG_IO_CONF1 : **st25r3911_com.h**
- ST25R3911_REG_IO_CONF2 : **st25r3911_com.h**
- ST25R3911_REG_IRQ_ERROR_WUP : **st25r3911_com.h**
- ST25R3911_REG_IRQ_MAIN : **st25r3911_com.h**
- ST25R3911_REG_IRQ_MASK_ERROR_WUP : **st25r3911_com.h**
- ST25R3911_REG_IRQ_MASK_MAIN : **st25r3911_com.h**
- ST25R3911_REG_IRQ_MASK_TIMER_NFC : **st25r3911_com.h**
- ST25R3911_REG_IRQ_TIMER_NFC : **st25r3911_com.h**
- ST25R3911_REG_ISO14443A_NFC : **st25r3911_com.h**
- ST25R3911_REG_ISO14443B_1 : **st25r3911_com.h**
- ST25R3911_REG_ISO14443B_2 : **st25r3911_com.h**
- ST25R3911_REG_MASK_RX_TIMER : **st25r3911_com.h**
- ST25R3911_REG_MODE : **st25r3911_com.h**
- ST25R3911_REG_NFCIP1_BIT_RATE : **st25r3911_com.h**
- ST25R3911_REG_NO_RESPONSE_TIMER1 : **st25r3911_com.h**
- ST25R3911_REG_NO_RESPONSE_TIMER2 : **st25r3911_com.h**
- ST25R3911_REG_NUM_TX_BYTES1 : **st25r3911_com.h**
- ST25R3911_REG_NUM_TX_BYTES2 : **st25r3911_com.h**
- ST25R3911_REG_OP_CONTROL : **st25r3911_com.h**
- ST25R3911_REG_PHASE_MEASURE_AA_RESULT : **st25r3911_com.h**
- ST25R3911_REG_PHASE_MEASURE_CONF : **st25r3911_com.h**
- ST25R3911_REG_PHASE_MEASURE_REF : **st25r3911_com.h**
- ST25R3911_REG_PHASE_MEASURE_RESULT : **st25r3911_com.h**
- ST25R3911_REG_REGULATOR_CONTROL : **st25r3911_com.h**
- ST25R3911_REG_REGULATOR_RESULT : **st25r3911_com.h**
- ST25R3911_REG_RFO_AM_OFF_LEVEL : **st25r3911_com.h**
- ST25R3911_REG_RFO_AM_ON_LEVEL : **st25r3911_com.h**
- ST25R3911_REG_RSSI_RESULT : **st25r3911_com.h**
- ST25R3911_REG_RX_CONF1 : **st25r3911_com.h**
- ST25R3911_REG_RX_CONF2 : **st25r3911_com.h**
- ST25R3911_REG_RX_CONF3 : **st25r3911_com.h**
- ST25R3911_REG_RX_CONF4 : **st25r3911_com.h**
- ST25R3911_REG_STREAM_MODE : **st25r3911_com.h**
- ST25R3911_REG_WUP_TIMER_CONTROL : **st25r3911_com.h**
- ST25R3911_THRESHOLD_DO_NOT_SET : **st25r3911.h**

- st25r3911AdjustRegulators() : **st25r3911.h**
- st25r3911CalibrateAntenna() : **st25r3911.h**
- st25r3911CalibrateCapacitiveSensor() : **st25r3911.h**
- st25r3911CalibrateModulationDepth() : **st25r3911.h**
- st25r3911ChangeRegisterBits() : **st25r3911_com.h**
- st25r3911ChangeTestRegisterBits() : **st25r3911_com.h**
- st25r3911CheckChipID() : **st25r3911.h**
- st25r3911CheckForReceivedInterrupts() : **st25r3911_interrupt.h**
- st25r3911CheckReg() : **st25r3911.h**
- st25r3911ClearInterrupts() : **st25r3911_interrupt.h**
- st25r3911ClrRegisterBits() : **st25r3911_com.h**
- st25r3911Deinitialize() : **st25r3911.h**
- st25r3911DisableInterrupts() : **st25r3911_interrupt.h**
- st25r3911EnableInterrupts() : **st25r3911_interrupt.h**
- st25r3911ExecuteCommand() : **st25r3911_com.h**
- st25r3911ExecuteCommands() : **st25r3911_com.h**
- st25r3911GetInterrupt() : **st25r3911_interrupt.h**
- st25r3911GetNoResponseTime_64fcs() : **st25r3911.h**
- st25r3911GetNumFIFOLastBits() : **st25r3911.h**
- st25r3911GetRegsDump() : **st25r3911.h**
- st25r3911GetRSSI() : **st25r3911.h**
- st25r3911Initialize() : **st25r3911.h**
- st25r3911InitInterrupts() : **st25r3911_interrupt.h**
- st25r3911IRQCallbackRestore() : **st25r3911_interrupt.h**
- st25r3911IRQCallbackSet() : **st25r3911_interrupt.h**
- st25r3911IsCmdValid() : **st25r3911.h**
- st25r3911IsCRCinFIFO : **st25r3911.h**
- st25r3911IsExtFieldOn : **st25r3911.h**
- st25r3911IsGPTRunning : **st25r3911.h**
- st25r3911IsLastFIFOComplete : **st25r3911.h**
- st25r3911IsOscOn : **st25r3911.h**
- st25r3911Isr() : **st25r3911_interrupt.h**
- st25r3911IsRegValid() : **st25r3911_com.h**
- st25r3911IsTxEnabled : **st25r3911.h**
- st25r3911MeasureAmplitude() : **st25r3911.h**
- st25r3911MeasureCapacitance() : **st25r3911.h**
- st25r3911MeasurePhase() : **st25r3911.h**
- st25r3911MeasurePowerSupply() : **st25r3911.h**
- st25r3911MeasureVoltage() : **st25r3911.h**

- st25r3911ModifyInterrupts() : **st25r3911_interrupt.h**
- st25r3911ModifyRegister() : **st25r3911_com.h**
- st25r3911OscOn() : **st25r3911.h**
- st25r3911PerformCollisionAvoidance() : **st25r3911.h**
- st25r3911ReadFifo() : **st25r3911_com.h**
- st25r3911ReadMultipleRegisters() : **st25r3911_com.h**
- st25r3911ReadRegister() : **st25r3911_com.h**
- st25r3911ReadTestRegister() : **st25r3911_com.h**
- st25r3911SetAntennaMode() : **st25r3911.h**
- st25r3911SetBitrate() : **st25r3911.h**
- st25r3911SetGPTime_8fcs() : **st25r3911.h**
- st25r3911SetNoResponseTime_64fcs() : **st25r3911.h**
- st25r3911SetNumTxBits() : **st25r3911.h**
- st25r3911SetRegisterBits() : **st25r3911_com.h**
- st25r3911SetRegulators() : **st25r3911.h**
- st25r3911SetStartNoResponseTime_64fcs() : **st25r3911.h**
- st25r3911StartGPTimer_8fcs() : **st25r3911.h**
- st25r3911StreamConfigure() : **st25r3911.h**
- st25r3911TxOff : **st25r3911.h**
- st25r3911TxRxOff() : **st25r3911.h**
- st25r3911TxRxOn() : **st25r3911.h**
- st25r3911WaitForInterruptsTimed() : **st25r3911_interrupt.h**
- st25r3911WriteFifo() : **st25r3911_com.h**
- st25r3911WriteMultipleRegisters() : **st25r3911_com.h**
- st25r3911WriteRegister() : **st25r3911_com.h**
- st25r3911WriteTestRegister() : **st25r3911_com.h**
- ST25R3916_BR_106 : **st25r3916.h**
- ST25R3916_BR_1695 : **st25r3916.h**
- ST25R3916_BR_212 : **st25r3916.h**
- ST25R3916_BR_3390 : **st25r3916.h**
- ST25R3916_BR_424 : **st25r3916.h**
- ST25R3916_BR_6780 : **st25r3916.h**
- ST25R3916_BR_848 : **st25r3916.h**
- ST25R3916_BR_DO_NOT_SET : **st25r3916.h**
- ST25R3916_CMD_ADJUST_REGULATORS : **st25r3916.h**
- ST25R3916_CMD_AM_MOD_STATE_CHANGE : **st25r3916.h**
- ST25R3916_CMD_CALIBRATE_DRIVER_TIMING : **st25r3916.h**
- ST25R3916_CMD_CLEAR_FIFO : **st25r3916.h**
- ST25R3916_CMD_CLEAR_RSSI : **st25r3916.h**

- ST25R3916_CMD_GOTO_SENSE : **st25r3916.h**
- ST25R3916_CMD_GOTO_SLEEP : **st25r3916.h**
- ST25R3916_CMD_INITIAL_RF_COLLISION : **st25r3916.h**
- ST25R3916_CMD_MASK_RECEIVE_DATA : **st25r3916.h**
- ST25R3916_CMD_MEASURE_AMPLITUDE : **st25r3916.h**
- ST25R3916_CMD_MEASURE_PHASE : **st25r3916.h**
- ST25R3916_CMD_MEASURE_VDD : **st25r3916.h**
- ST25R3916_CMD_RESET_RXGAIN : **st25r3916.h**
- ST25R3916_CMD_RESPONSE_RF_COLLISION_N : **st25r3916.h**
- ST25R3916_CMD_SET_DEFAULT : **st25r3916.h**
- ST25R3916_CMD_SPACE_B_ACCESS : **st25r3916.h**
- ST25R3916_CMD_START_GP_TIMER : **st25r3916.h**
- ST25R3916_CMD_START_MASK_RECEIVE_TIMER : **st25r3916.h**
- ST25R3916_CMD_START_NO_RESPONSE_TIMER : **st25r3916.h**
- ST25R3916_CMD_START_PPON2_TIMER : **st25r3916.h**
- ST25R3916_CMD_START_WUP_TIMER : **st25r3916.h**
- ST25R3916_CMD_STOP : **st25r3916.h**
- ST25R3916_CMD_STOP_NRT : **st25r3916.h**
- ST25R3916_CMD_TEST_ACCESS : **st25r3916.h**
- ST25R3916_CMD_TRANSMIT_REQA : **st25r3916.h**
- ST25R3916_CMD_TRANSMIT_WITH_CRC : **st25r3916.h**
- ST25R3916_CMD_TRANSMIT_WITHOUT_CRC : **st25r3916.h**
- ST25R3916_CMD_TRANSMIT_WUPA : **st25r3916.h**
- ST25R3916_CMD_TRANSPARENT_MODE : **st25r3916.h**
- ST25R3916_CMD_UNMASK_RECEIVE_DATA : **st25r3916.h**
- ST25R3916_FIFO_DEPTH : **st25r3916.h**
- ST25R3916_FIFO_STATUS_LEN : **st25r3916_com.h**
- ST25R3916_IRQ_MASK_ALL : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_APON : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_CAC : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_CAT : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_COL : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_CRC : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_DCT : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_EOF : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_EON : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_ERR1 : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_ERR2 : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_FWL : **st25r3916_irq.h**

- ST25R3916_IRQ_MASK_GPE : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_NFCT : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_NONE : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_NRE : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_OSC : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_PAR : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_PPON2 : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_RFU : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_RFU2 : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_RX_REST : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_RXE : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_RXE_PTA : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_RXS : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_SL_WL : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_TXE : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_WAM : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_WPH : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_WT : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_WU_A : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_WU_A_X : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_WU_F : **st25r3916_irq.h**
- ST25R3916_PTM_A_LEN : **st25r3916_com.h**
- ST25R3916_PTM_B_LEN : **st25r3916_com.h**
- ST25R3916_PTM_F_LEN : **st25r3916_com.h**
- ST25R3916_PTM_LEN : **st25r3916_com.h**
- ST25R3916_PTM_TSN_LEN : **st25r3916_com.h**
- ST25R3916_REG_AD_RESULT : **st25r3916_com.h**
- ST25R3916_REG_AMPLITUDE_MEASURE_AA_RESULT : **st25r3916_com.h**
- ST25R3916_REG_AMPLITUDE_MEASURE_CONF : **st25r3916_com.h**
- ST25R3916_REG_AMPLITUDE_MEASURE_REF : **st25r3916_com.h**
- ST25R3916_REG_AMPLITUDE_MEASURE_RESULT : **st25r3916_com.h**
- ST25R3916_REG_ANT_TUNE_A : **st25r3916_com.h**
- ST25R3916_REG_ANT_TUNE_B : **st25r3916_com.h**
- ST25R3916_REG_AUX : **st25r3916_com.h**
- ST25R3916_REG_AUX_DISPLAY : **st25r3916_com.h**
- ST25R3916_REG_AUX_MOD : **st25r3916_com.h**
- ST25R3916_REG_BIT_RATE : **st25r3916_com.h**
- ST25R3916_REG_CAP_SENSOR_CONTROL : **st25r3916_com.h**
- ST25R3916_REG_CAP_SENSOR_RESULT : **st25r3916_com.h**

- ST25R3916_REG_COLLISION_STATUS : **st25r3916_com.h**
- ST25R3916_REG_CORR_CONF1 : **st25r3916_com.h**
- ST25R3916_REG_CORR_CONF2 : **st25r3916_com.h**
- ST25R3916_REG_EMD_SUP_CONF : **st25r3916_com.h**
- ST25R3916_REG_FIELD_ON_GT : **st25r3916_com.h**
- ST25R3916_REG_FIELD_THRESHOLD_ACTV : **st25r3916_com.h**
- ST25R3916_REG_FIELD_THRESHOLD_DEACTV : **st25r3916_com.h**
- ST25R3916_REG_FIFO_STATUS1 : **st25r3916_com.h**
- ST25R3916_REG_FIFO_STATUS2 : **st25r3916_com.h**
- ST25R3916_REG_GAIN_RED_STATE : **st25r3916_com.h**
- ST25R3916_REG_GPT1 : **st25r3916_com.h**
- ST25R3916_REG_GPT2 : **st25r3916_com.h**
- ST25R3916_REG_IC_IDENTITY : **st25r3916_com.h**
- ST25R3916_REG_IO_CONF1 : **st25r3916_com.h**
- ST25R3916_REG_IO_CONF2 : **st25r3916_com.h**
- ST25R3916_REG_IRQ_ERROR_WUP : **st25r3916_com.h**
- ST25R3916_REG_IRQ_MAIN : **st25r3916_com.h**
- ST25R3916_REG_IRQ_MASK_ERROR_WUP : **st25r3916_com.h**
- ST25R3916_REG_IRQ_MASK_MAIN : **st25r3916_com.h**
- ST25R3916_REG_IRQ_MASK_TARGET : **st25r3916_com.h**
- ST25R3916_REG_IRQ_MASK_TIMER_NFC : **st25r3916_com.h**
- ST25R3916_REG_IRQ_TARGET : **st25r3916_com.h**
- ST25R3916_REG_IRQ_TIMER_NFC : **st25r3916_com.h**
- ST25R3916_REG_ISO14443A_NFC : **st25r3916_com.h**
- ST25R3916_REG_ISO14443B_1 : **st25r3916_com.h**
- ST25R3916_REG_ISO14443B_2 : **st25r3916_com.h**
- ST25R3916_REG_MASK_RX_TIMER : **st25r3916_com.h**
- ST25R3916_REG_MODE : **st25r3916_com.h**
- ST25R3916_REG_NFCIP1_BIT_RATE : **st25r3916_com.h**
- ST25R3916_REG_NO_RESPONSE_TIMER1 : **st25r3916_com.h**
- ST25R3916_REG_NO_RESPONSE_TIMER2 : **st25r3916_com.h**
- ST25R3916_REG_NUM_TX_BYTES1 : **st25r3916_com.h**
- ST25R3916_REG_NUM_TX_BYTES2 : **st25r3916_com.h**
- ST25R3916_REG_OP_CONTROL : **st25r3916_com.h**
- ST25R3916_REG_OVERSHOOT_CONF1 : **st25r3916_com.h**
- ST25R3916_REG_OVERSHOOT_CONF2 : **st25r3916_com.h**
- ST25R3916_REG_P2P_RX_CONF : **st25r3916_com.h**
- ST25R3916_REG_PASSIVE_TARGET : **st25r3916_com.h**
- ST25R3916_REG_PASSIVE_TARGET_STATUS : **st25r3916_com.h**

- ST25R3916_REG_PHASE_MEASURE_AA_RESULT : **st25r3916_com.h**
- ST25R3916_REG_PHASE_MEASURE_CONF : **st25r3916_com.h**
- ST25R3916_REG_PHASE_MEASURE_REF : **st25r3916_com.h**
- ST25R3916_REG_PHASE_MEASURE_RESULT : **st25r3916_com.h**
- ST25R3916_REG_PPON2 : **st25r3916_com.h**
- ST25R3916_REG_PT_MOD : **st25r3916_com.h**
- ST25R3916_REG_REGULATOR_CONTROL : **st25r3916_com.h**
- ST25R3916_REG_REGULATOR_RESULT : **st25r3916_com.h**
- ST25R3916_REG_RES_AM_MOD : **st25r3916_com.h**
- ST25R3916_REG_RSSI_RESULT : **st25r3916_com.h**
- ST25R3916_REG_RX_CONF1 : **st25r3916_com.h**
- ST25R3916_REG_RX_CONF2 : **st25r3916_com.h**
- ST25R3916_REG_RX_CONF3 : **st25r3916_com.h**
- ST25R3916_REG_RX_CONF4 : **st25r3916_com.h**
- ST25R3916_REG_SQUELCH_TIMER : **st25r3916_com.h**
- ST25R3916_REG_STREAM_MODE : **st25r3916_com.h**
- ST25R3916_REG_SUBC_START_TIME : **st25r3916_com.h**
- ST25R3916_REG_TIMER_EMV_CONTROL : **st25r3916_com.h**
- ST25R3916_REG_TX_DRIVER : **st25r3916_com.h**
- ST25R3916_REG_TX_DRIVER_STATUS : **st25r3916_com.h**
- ST25R3916_REG_TX_DRIVER_TIMING : **st25r3916_com.h**
- ST25R3916_REG_UNDERSHOOT_CONF1 : **st25r3916_com.h**
- ST25R3916_REG_UNDERSHOOT_CONF2 : **st25r3916_com.h**
- ST25R3916_REG_WUP_TIMER_CONTROL : **st25r3916_com.h**
- ST25R3916_SPACE_B : **st25r3916_com.h**
- ST25R3916_SPACE_B_REG_LEN : **st25r3916_com.h**
- ST25R3916_THRESHOLD_DO_NOT_SET : **st25r3916.h**
- ST25R3916_TOUT_OSC_STABLE : **st25r3916.h**
- st25r3916AatTune() : **st25r3916_aat.h**
- st25r3916AdjustRegulators() : **st25r3916.h**
- st25r3916CalibrateCapacitiveSensor() : **st25r3916.h**
- st25r3916ChangeRegisterBits() : **st25r3916_com.h**
- st25r3916ChangeTestRegisterBits() : **st25r3916_com.h**
- st25r3916CheckChipID() : **st25r3916.h**
- st25r3916CheckForReceivedInterrupts() : **st25r3916_irq.h**
- st25r3916CheckReg() : **st25r3916_com.h**
- st25r3916ClearAndEnableInterrupts() : **st25r3916_irq.h**
- st25r3916ClearInterrupts() : **st25r3916_irq.h**
- st25r3916ClrRegisterBits() : **st25r3916_com.h**

- st25r3916Deinitialize() : **st25r3916.h**
- st25r3916DisableInterrupts() : **st25r3916_irq.h**
- st25r3916EnableInterrupts() : **st25r3916_irq.h**
- st25r3916ExecuteCommand() : **st25r3916_com.h**
- st25r3916ExecuteCommandAndGetResult() : **st25r3916.h**
- st25r3916ExecuteCommands() : **st25r3916_com.h**
- st25r3916GetInterrupt() : **st25r3916_irq.h**
- st25r3916GetNoResponseTime() : **st25r3916.h**
- st25r3916GetNumFIFOBytes() : **st25r3916.h**
- st25r3916GetNumFIFOLastBits() : **st25r3916.h**
- st25r3916GetRegsDump() : **st25r3916.h**
- st25r3916GetRSSI() : **st25r3916.h**
- st25r3916Initialize() : **st25r3916.h**
- st25r3916InitInterrupts() : **st25r3916_irq.h**
- st25r3916IRQCallbackRestore() : **st25r3916_irq.h**
- st25r3916IRQCallbackSet() : **st25r3916_irq.h**
- st25r3916IsAATOn : **st25r3916.h**
- st25r3916IsCmdValid() : **st25r3916.h**
- st25r3916IsExtFieldOn : **st25r3916.h**
- st25r3916IsGPTRunning : **st25r3916.h**
- st25r3916IsLastFIFOComplete : **st25r3916.h**
- st25r3916IsNRTinEMV : **st25r3916.h**
- st25r3916IsOscOn : **st25r3916.h**
- st25r3916Isr() : **st25r3916_irq.h**
- st25r3916IsRegValid() : **st25r3916_com.h**
- st25r3916IsTxEnabled : **st25r3916.h**
- st25r3916ledEvtCmd() : **st25r3916_led.h**
- st25r3916ledEvtCmds() : **st25r3916_led.h**
- st25r3916ledEvtIrq() : **st25r3916_led.h**
- st25r3916ledEvtWrMultiReg() : **st25r3916_led.h**
- st25r3916ledEvtWrReg() : **st25r3916_led.h**
- st25r3916ledInit() : **st25r3916_led.h**
- st25r3916MeasureAmplitude() : **st25r3916.h**
- st25r3916MeasureCapacitance() : **st25r3916.h**
- st25r3916MeasurePhase() : **st25r3916.h**
- st25r3916MeasurePowerSupply() : **st25r3916.h**
- st25r3916MeasureVoltage() : **st25r3916.h**
- st25r3916ModifyInterrupts() : **st25r3916_irq.h**
- st25r3916ModifyRegister() : **st25r3916_com.h**

- st25r3916OscOn() : **st25r3916.h**
- st25r3916PerformCollisionAvoidance() : **st25r3916.h**
- st25r3916ReadFifo() : **st25r3916_com.h**
- st25r3916ReadMultipleRegisters() : **st25r3916_com.h**
- st25r3916ReadPTMem() : **st25r3916_com.h**
- st25r3916ReadRegister() : **st25r3916_com.h**
- st25r3916ReadTestRegister() : **st25r3916_com.h**
- st25r3916SetAntennaMode() : **st25r3916.h**
- st25r3916SetBitrate() : **st25r3916.h**
- st25r3916SetGPTime() : **st25r3916.h**
- st25r3916SetNoResponseTime() : **st25r3916.h**
- st25r3916SetNumTxBits() : **st25r3916.h**
- st25r3916SetRegisterBits() : **st25r3916_com.h**
- st25r3916SetRegulators() : **st25r3916.h**
- st25r3916SetStartGPTimer() : **st25r3916.h**
- st25r3916SetStartNoResponseTimer() : **st25r3916.h**
- st25r3916StreamConfigure() : **st25r3916.h**
- st25r3916TxOff : **st25r3916.h**
- st25r3916TxRxOff : **st25r3916.h**
- st25r3916TxRxOn : **st25r3916.h**
- st25r3916WaitForInterruptsTimed() : **st25r3916_irq.h**
- st25r3916WriteFifo() : **st25r3916_com.h**
- st25r3916WriteMultipleRegisters() : **st25r3916_com.h**
- st25r3916WritePTMem() : **st25r3916_com.h**
- st25r3916WritePTMemF() : **st25r3916_com.h**
- st25r3916WritePTMemTSN() : **st25r3916_com.h**
- st25r3916WriteRegister() : **st25r3916_com.h**
- st25r3916WriteTestRegister() : **st25r3916_com.h**
- ST25R500_BR_106_26 : **st25r500.h**
- ST25R500_BR_212_53 : **st25r500.h**
- ST25R500_BR_424 : **st25r500.h**
- ST25R500_BR_848 : **st25r500.h**
- ST25R500_BR_DO_NOT_SET : **st25r500.h**
- ST25R500_CEM_A_LEN : **st25r500_com.h**
- ST25R500_CEM_B_LEN : **st25r500_com.h**
- ST25R500_CEM_F_LEN : **st25r500_com.h**
- ST25R500_CEM_LEN : **st25r500_com.h**
- ST25R500_CMD_ADJUST_REGULATORS : **st25r500.h**
- ST25R500_CMD_CALIBRATE_RC : **st25r500.h**

- ST25R500_CMD_CALIBRATE_WU : **st25r500.h**
- ST25R500_CMD_CLEAR_FIFO : **st25r500.h**
- ST25R500_CMD_CLEAR_RXGAIN : **st25r500.h**
- ST25R500_CMD_CLEAR_WU_CALIB : **st25r500.h**
- ST25R500_CMD_LEN : **st25r500.h**
- ST25R500_CMD_MASK_RECEIVE_DATA : **st25r500.h**
- ST25R500_CMD_MEASURE_IQ : **st25r500.h**
- ST25R500_CMD_MEASURE_WU : **st25r500.h**
- ST25R500_CMD_MODE : **st25r500.h**
- ST25R500_CMD_NFC_FIELD_ON : **st25r500.h**
- ST25R500_CMD_SENSE_RF : **st25r500.h**
- ST25R500_CMD_SET_DEFAULT : **st25r500.h**
- ST25R500_CMD_START_GP_TIMER : **st25r500.h**
- ST25R500_CMD_START_MRT : **st25r500.h**
- ST25R500_CMD_START_NRT : **st25r500.h**
- ST25R500_CMD_START_WUT : **st25r500.h**
- ST25R500_CMD_STOP : **st25r500.h**
- ST25R500_CMD_STOP_NRT : **st25r500.h**
- ST25R500_CMD_TEST_ACCESS : **st25r500.h**
- ST25R500_CMD_TRANSMIT : **st25r500.h**
- ST25R500_CMD_TRANSMIT_EOF : **st25r500.h**
- ST25R500_CMD_TRIGGER_DIAG : **st25r500.h**
- ST25R500_CMD_TRIGGER_WU_EV : **st25r500.h**
- ST25R500_CMD_UNMASK_RECEIVE_DATA : **st25r500.h**
- ST25R500_DIAG_MEAS_CMD : **st25r500.h**
- ST25R500_DIAG_MEAS_CMD_LEN : **st25r500.h**
- ST25R500_DIAG_MEAS_I_VDD_DR : **st25r500.h**
- ST25R500_DIAG_MEAS_RES_LEN : **st25r500.h**
- ST25R500_DIAG_MEAS_VDD_A : **st25r500.h**
- ST25R500_DIAG_MEAS_VDD_AGD : **st25r500.h**
- ST25R500_DIAG_MEAS_VDD_D : **st25r500.h**
- ST25R500_DIAG_MEAS_VDD_DR : **st25r500.h**
- ST25R500_DIAG_MEAS_VDD_IO : **st25r500.h**
- ST25R500_DIAG_MEAS_VDD_TX : **st25r500.h**
- ST25R500_DIAG_MEAS_VDD_VDD : **st25r500.h**
- ST25R500_DPOCR_MAX_ENTRIES : **st25r500_dpocr.h**
- ST25R500_FIFO_ACCESS : **st25r500.h**
- ST25R500_FIFO_DEPTH : **st25r500.h**
- ST25R500_FIFO_STATUS_LEN : **st25r500_com.h**

- ST25R500_IRQ_MASK_ALL : **st25r500_irq.h**
- ST25R500_IRQ_MASK_CE_SC : **st25r500_irq.h**
- ST25R500_IRQ_MASK_COL : **st25r500_irq.h**
- ST25R500_IRQ_MASK_DCT : **st25r500_irq.h**
- ST25R500_IRQ_MASK_EOF : **st25r500_irq.h**
- ST25R500_IRQ_MASK_EON : **st25r500_irq.h**
- ST25R500_IRQ_MASK_GPE : **st25r500_irq.h**
- ST25R500_IRQ_MASK_NFCT : **st25r500_irq.h**
- ST25R500_IRQ_MASK_NONE : **st25r500_irq.h**
- ST25R500_IRQ_MASK_NRE : **st25r500_irq.h**
- ST25R500_IRQ_MASK_OSC : **st25r500_irq.h**
- ST25R500_IRQ_MASK_RFU : **st25r500_irq.h**
- ST25R500_IRQ_MASK_RX_ERR : **st25r500_irq.h**
- ST25R500_IRQ_MASK_RX_REST : **st25r500_irq.h**
- ST25R500_IRQ_MASK_RXE : **st25r500_irq.h**
- ST25R500_IRQ_MASK_RXE_CE : **st25r500_irq.h**
- ST25R500_IRQ_MASK_RXS : **st25r500_irq.h**
- ST25R500_IRQ_MASK_SUBC_START : **st25r500_irq.h**
- ST25R500_IRQ_MASK_TXE : **st25r500_irq.h**
- ST25R500_IRQ_MASK_WL : **st25r500_irq.h**
- ST25R500_IRQ_MASK_WPT_FOD : **st25r500_irq.h**
- ST25R500_IRQ_MASK_WPT_STOP : **st25r500_irq.h**
- ST25R500_IRQ_MASK_WUI : **st25r500_irq.h**
- ST25R500_IRQ_MASK_WUQ : **st25r500_irq.h**
- ST25R500_IRQ_MASK_WUT : **st25r500_irq.h**
- ST25R500_IRQ_MASK_WUTME : **st25r500_irq.h**
- ST25R500_OBS_MODE_LEN : **st25r500_com.h**
- ST25R500_READ_MODE : **st25r500.h**
- ST25R500_REG_ANA_DISPLAY1 : **st25r500_com.h**
- ST25R500_REG_ANA_DISPLAY2 : **st25r500_com.h**
- ST25R500_REG_AWS_CONFIG1 : **st25r500_com.h**
- ST25R500_REG_AWS_CONFIG2 : **st25r500_com.h**
- ST25R500_REG_AWS_TIME1 : **st25r500_com.h**
- ST25R500_REG_AWS_TIME2 : **st25r500_com.h**
- ST25R500_REG_AWS_TIME3 : **st25r500_com.h**
- ST25R500_REG_AWS_TIME4 : **st25r500_com.h**
- ST25R500_REG_CE_CONFIG1 : **st25r500_com.h**
- ST25R500_REG_CE_CONFIG2 : **st25r500_com.h**
- ST25R500_REG_CE_CONFIG3 : **st25r500_com.h**

- ST25R500_REG_CE_STATUS1 : **st25r500_com.h**
- ST25R500_REG_CE_STATUS2 : **st25r500_com.h**
- ST25R500_REG_CE_TX_MOD1 : **st25r500_com.h**
- ST25R500_REG_CE_TX_MOD2 : **st25r500_com.h**
- ST25R500_REG_CEM_A : **st25r500_com.h**
- ST25R500_REG_CEM_F : **st25r500_com.h**
- ST25R500_REG_COLLISION : **st25r500_com.h**
- ST25R500_REG_CORR1 : **st25r500_com.h**
- ST25R500_REG_CORR2 : **st25r500_com.h**
- ST25R500_REG_CORR3 : **st25r500_com.h**
- ST25R500_REG_CORR4 : **st25r500_com.h**
- ST25R500_REG_CORR5 : **st25r500_com.h**
- ST25R500_REG_CORR6 : **st25r500_com.h**
- ST25R500_REG_DRIVER : **st25r500_com.h**
- ST25R500_REG_DROP_200 : **st25r500.h**
- ST25R500_REG_DROP_250 : **st25r500.h**
- ST25R500_REG_DROP_300 : **st25r500.h**
- ST25R500_REG_DROP_350 : **st25r500.h**
- ST25R500_REG_DROP_400 : **st25r500.h**
- ST25R500_REG_DROP_450 : **st25r500.h**
- ST25R500_REG_DROP_500 : **st25r500.h**
- ST25R500_REG_DROP_550 : **st25r500.h**
- ST25R500_REG_DROP_DO_NOT_SET : **st25r500.h**
- ST25R500_REG_EFD_THRESHOLD : **st25r500_com.h**
- ST25R500_REG_EMD1 : **st25r500_com.h**
- ST25R500_REG_EMD2 : **st25r500_com.h**
- ST25R500_REG_FIFO : **st25r500_com.h**
- ST25R500_REG_FIFO_STATUS1 : **st25r500_com.h**
- ST25R500_REG_FIFO_STATUS2 : **st25r500_com.h**
- ST25R500_REG_GENERAL : **st25r500_com.h**
- ST25R500_REG_GPIO : **st25r500_com.h**
- ST25R500_REG_GPT1 : **st25r500_com.h**
- ST25R500_REG_GPT2 : **st25r500_com.h**
- ST25R500_REG_IC_ID : **st25r500_com.h**
- ST25R500_REG_IRQ1 : **st25r500_com.h**
- ST25R500_REG_IRQ2 : **st25r500_com.h**
- ST25R500_REG_IRQ3 : **st25r500_com.h**
- ST25R500_REG_IRQ_MASK1 : **st25r500_com.h**
- ST25R500_REG_IRQ_MASK2 : **st25r500_com.h**

- ST25R500_REG_IRQ_MASK3 : **st25r500_com.h**
- ST25R500_REG_LEN : **st25r500.h**
- ST25R500_REG_MRT1 : **st25r500_com.h**
- ST25R500_REG_MRT2 : **st25r500_com.h**
- ST25R500_REG_NRT1 : **st25r500_com.h**
- ST25R500_REG_NRT2 : **st25r500_com.h**
- ST25R500_REG_NRT_GPT_CONF : **st25r500_com.h**
- ST25R500_REG_OPERATION : **st25r500_com.h**
- ST25R500_REG_OVERSHOOT_CONF : **st25r500_com.h**
- ST25R500_REG_PROTOCOL : **st25r500_com.h**
- ST25R500_REG_PROTOCOL_RX1 : **st25r500_com.h**
- ST25R500_REG_PROTOCOL_RX2 : **st25r500_com.h**
- ST25R500_REG_PROTOCOL_RX3 : **st25r500_com.h**
- ST25R500_REG_PROTOCOL_TX1 : **st25r500_com.h**
- ST25R500_REG_PROTOCOL_TX2 : **st25r500_com.h**
- ST25R500_REG_REGULATOR : **st25r500_com.h**
- ST25R500_REG_RSSI_I : **st25r500_com.h**
- ST25R500_REG_RSSI_Q : **st25r500_com.h**
- ST25R500_REG_RX_ANA1 : **st25r500_com.h**
- ST25R500_REG_RX_ANA2 : **st25r500_com.h**
- ST25R500_REG_RX_ANA3 : **st25r500_com.h**
- ST25R500_REG_RX_ANA4 : **st25r500_com.h**
- ST25R500_REG_RX_DIG : **st25r500_com.h**
- ST25R500_REG_SENSE_DISPLAY : **st25r500_com.h**
- ST25R500_REG_SQT : **st25r500_com.h**
- ST25R500_REG_STATUS1 : **st25r500_com.h**
- ST25R500_REG_STATUS2 : **st25r500_com.h**
- ST25R500_REG_STATUS_STATIC1 : **st25r500_com.h**
- ST25R500_REG_STATUS_STATIC2 : **st25r500_com.h**
- ST25R500_REG_STATUS_STATIC3 : **st25r500_com.h**
- ST25R500_REG_TX_FRAME1 : **st25r500_com.h**
- ST25R500_REG_TX_FRAME2 : **st25r500_com.h**
- ST25R500_REG_TX_MOD1 : **st25r500_com.h**
- ST25R500_REG_TX_MOD2 : **st25r500_com.h**
- ST25R500_REG_UNDERSHOOT_CONF : **st25r500_com.h**
- ST25R500_REG_WAKEUP_CONF1 : **st25r500_com.h**
- ST25R500_REG_WAKEUP_CONF2 : **st25r500_com.h**
- ST25R500_REG_WAKEUP_CONF3 : **st25r500_com.h**
- ST25R500_REG_WU_I_ADC : **st25r500_com.h**

- ST25R500_REG_WU_I_CAL : **st25r500_com.h**
- ST25R500_REG_WU_I_CONF : **st25r500_com.h**
- ST25R500_REG_WU_I_DELTA : **st25r500_com.h**
- ST25R500_REG_WU_I_REF : **st25r500_com.h**
- ST25R500_REG_WU_Q_ADC : **st25r500_com.h**
- ST25R500_REG_WU_Q_CAL : **st25r500_com.h**
- ST25R500_REG_WU_Q_CONF : **st25r500_com.h**
- ST25R500_REG_WU_Q_DELTA : **st25r500_com.h**
- ST25R500_REG_WU_Q_REF : **st25r500_com.h**
- ST25R500_REG_WU_STATUS : **st25r500_com.h**
- ST25R500_TEST_REG_ADC2CE : **st25r500_com.h**
- ST25R500_TEST_REG_DIAG_MEAS : **st25r500_com.h**
- ST25R500_TEST_REG_MAN_TIMING : **st25r500_com.h**
- ST25R500_TEST_REG_OSC_TIMING : **st25r500_com.h**
- ST25R500_TEST_REG_OVERLAP_CONTROL : **st25r500_com.h**
- ST25R500_TEST_REG_RXE_DISP1 : **st25r500_com.h**
- ST25R500_TEST_REG_RXE_DISP2 : **st25r500_com.h**
- ST25R500_TEST_REG_RXS_DISP1 : **st25r500_com.h**
- ST25R500_TEST_REG_RXS_DISP2 : **st25r500_com.h**
- ST25R500_THRESHOLD_DO_NOT_SET : **st25r500.h**
- ST25R500_TOUT_OSC_STABLE : **st25r500.h**
- ST25R500_WRITE_MODE : **st25r500.h**
- st25r500AdjustRegulators() : **st25r500.h**
- st25r500CalibrateWU() : **st25r500.h**
- st25r500ChangeRegisterBits() : **st25r500_com.h**
- st25r500ChangeTestRegisterBits() : **st25r500_com.h**
- st25r500CheckChipID() : **st25r500.h**
- st25r500CheckForReceivedInterrupts() : **st25r500_irq.h**
- st25r500CheckReg() : **st25r500_com.h**
- st25r500ClearAndEnableInterrupts() : **st25r500_irq.h**
- st25r500ClearCalibration() : **st25r500.h**
- st25r500ClearInterrupts() : **st25r500_irq.h**
- st25r500ClrRegisterBits() : **st25r500_com.h**
- st25r500Deinitialize() : **st25r500.h**
- st25r500DiagMeasure() : **st25r500.h**
- st25r500DisableInterrupts() : **st25r500_irq.h**
- st25r500DpocrAdjust() : **st25r500_dpocr.h**
- st25r500DpocrConfigRead() : **st25r500_dpocr.h**
- st25r500DpocrConfigWrite() : **st25r500_dpocr.h**

- st25r500DpocrGetInfo() : **st25r500_dpocr.h**
- st25r500DpocrInitialize() : **st25r500_dpocr.h**
- st25r500DpocrReqAdj() : **st25r500_dpocr.h**
- st25r500DpocrSetEnabled() : **st25r500_dpocr.h**
- st25r500EnableInterrupts() : **st25r500_irq.h**
- st25r500ExecuteCommand() : **st25r500_com.h**
- st25r500ExecuteCommandAndGetResult() : **st25r500.h**
- st25r500ExecuteCommands() : **st25r500_com.h**
- st25r500GetCeGain() : **st25r500.h**
- st25r500GetInterrupt() : **st25r500_irq.h**
- st25r500GetNoResponseTime() : **st25r500.h**
- st25r500GetNumFIFOBytes() : **st25r500.h**
- st25r500GetNumFIFOLastBits() : **st25r500.h**
- st25r500GetRegsDump() : **st25r500.h**
- st25r500GetRSSI() : **st25r500.h**
- st25r500Initialize() : **st25r500.h**
- st25r500InitInterrupts() : **st25r500_irq.h**
- st25r500IRQCallbackRestore() : **st25r500_irq.h**
- st25r500IRQCallbackSet() : **st25r500_irq.h**
- st25r500IsCmdValid() : **st25r500.h**
- st25r500IsExtFieldOn : **st25r500.h**
- st25r500IsGPTRunning : **st25r500.h**
- st25r500IsLastFIFOComplete : **st25r500.h**
- st25r500IsNRTinEMV : **st25r500.h**
- st25r500IsOscOn : **st25r500.h**
- st25r500Isr() : **st25r500_irq.h**
- st25r500IsRegValid() : **st25r500_com.h**
- st25r500IsTxEnabled : **st25r500.h**
- st25r500IsTxOn : **st25r500.h**
- st25r500ledEvtCmd() : **st25r500_com.h**
- st25r500ledEvtCmds() : **st25r500_com.h**
- st25r500ledEvtIrq() : **st25r500_com.h**
- st25r500ledEvtWrMultiReg() : **st25r500_com.h**
- st25r500ledEvtWrReg() : **st25r500_com.h**
- st25r500ledInit() : **st25r500_com.h**
- st25r500MeasureAmplitude() : **st25r500.h**
- st25r500MeasureCombinedIQ() : **st25r500.h**
- st25r500MeasureCurrent() : **st25r500.h**
- st25r500MeasureI() : **st25r500.h**

- st25r500MeasureIQ() : **st25r500.h**
- st25r500MeasurePhase() : **st25r500.h**
- st25r500MeasureQ() : **st25r500.h**
- st25r500MeasureWU() : **st25r500.h**
- st25r500ModifyInterrupts() : **st25r500_irq.h**
- st25r500ModifyRegister() : **st25r500_com.h**
- st25r500OscOn() : **st25r500.h**
- st25r500PerformCollisionAvoidance() : **st25r500.h**
- st25r500ReadFifo() : **st25r500_com.h**
- st25r500ReadMultipleRegisters() : **st25r500_com.h**
- st25r500ReadRegister() : **st25r500_com.h**
- st25r500ReadTestRegister() : **st25r500_com.h**
- st25r500SetAntennaMode() : **st25r500.h**
- st25r500SetBitrate() : **st25r500.h**
- st25r500SetGPTime() : **st25r500.h**
- st25r500SetMRT() : **st25r500.h**
- st25r500SetNoResponseTime() : **st25r500.h**
- st25r500SetNumTxBits() : **st25r500.h**
- st25r500SetRegisterBits() : **st25r500_com.h**
- st25r500SetRegulators() : **st25r500.h**
- st25r500SetStartGPTimer() : **st25r500.h**
- st25r500SetStartNoResponseTimer() : **st25r500.h**
- st25r500TxOff : **st25r500.h**
- st25r500TxRxOff : **st25r500.h**
- st25r500TxRxOn : **st25r500.h**
- st25r500VDDDROff : **st25r500.h**
- st25r500VDDDROn : **st25r500.h**
- st25r500WaitForInterruptsTimed() : **st25r500_irq.h**
- st25r500WriteFifo() : **st25r500_com.h**
- st25r500WriteMultipleRegisters() : **st25r500_com.h**
- st25r500WriteMultipleTestRegister() : **st25r500_com.h**
- st25r500WriteRegister() : **st25r500_com.h**
- st25r500WriteTestRegister() : **st25r500_com.h**
- ST25R95_ACFILTER_RESPONSE_BUFLEN : **st25r95_com.h**
- ST25R95_ACSTATE_ACTIVE : **st25r95_com.h**
- ST25R95_ACSTATE_ACTIVEX : **st25r95_com.h**
- ST25R95_ACSTATE_HALT : **st25r95_com.h**
- ST25R95_ACSTATE_IDLE : **st25r95_com.h**
- ST25R95_ACSTATE_READYA : **st25r95_com.h**

- ST25R95_ACSTATE_READYAX : **st25r95_com.h**
- ST25R95_CMD_COMMAND_OFFSET : **st25r95_com.h**
- ST25R95_CMD_DATA_OFFSET : **st25r95_com.h**
- ST25R95_CMD_LENGTH_OFFSET : **st25r95_com.h**
- ST25R95_CMD_RESULT_OFFSET : **st25r95_com.h**
- ST25R95_COMMAND_ACFILTER : **st25r95_com.h**
- ST25R95_COMMAND_BAUDRATE : **st25r95_com.h**
- ST25R95_COMMAND_ECHO : **st25r95_com.h**
- ST25R95_COMMAND_IDLE : **st25r95_com.h**
- ST25R95_COMMAND_IDN : **st25r95_com.h**
- ST25R95_COMMAND_LISTEN : **st25r95_com.h**
- ST25R95_COMMAND_POLLFIED : **st25r95_com.h**
- ST25R95_COMMAND_PROTOCOLSELECT : **st25r95_com.h**
- ST25R95_COMMAND_RDREG : **st25r95_com.h**
- ST25R95_COMMAND_SEND : **st25r95_com.h**
- ST25R95_COMMAND_SENDRECV : **st25r95_com.h**
- ST25R95_COMMAND_WRREG : **st25r95_com.h**
- ST25R95_COMMUNICATION_BUFFER_SIZE : **st25r95_com.h**
- ST25R95_COMMUNICATION_UART_WDOGTIMER : **st25r95_com.h**
- ST25R95_CONTROL_POLL : **st25r95_com.h**
- ST25R95_CONTROL_POLL_NO_TIMEOUT : **st25r95_com.h**
- ST25R95_CONTROL_POLL_TIMEOUT : **st25r95_com.h**
- ST25R95_CONTROL_READ : **st25r95_com.h**
- ST25R95_CONTROL_RESET : **st25r95_com.h**
- ST25R95_CONTROL_SEND : **st25r95_com.h**
- ST25R95_DACDATA_MAX : **st25r95_com.h**
- ST25R95_ECHO_RESPONSE_BUFLEN : **st25r95_com.h**
- ST25R95_ERRCODE_61_SOF : **st25r95_com.h**
- ST25R95_ERRCODE_62_CRC : **st25r95_com.h**
- ST25R95_ERRCODE_63_SOF_HIGH : **st25r95_com.h**
- ST25R95_ERRCODE_65_SOF_LOW : **st25r95_com.h**
- ST25R95_ERRCODE_66_EGT : **st25r95_com.h**
- ST25R95_ERRCODE_67_TR1TOOLONG : **st25r95_com.h**
- ST25R95_ERRCODE_68_TR1TOOSHORT : **st25r95_com.h**
- ST25R95_ERRCODE_COMERROR : **st25r95_com.h**
- ST25R95_ERRCODE_CRC : **st25r95_com.h**
- ST25R95_ERRCODE_EGT : **st25r95_com.h**
- ST25R95_ERRCODE_FIELDLENGTH : **st25r95_com.h**
- ST25R95_ERRCODE_FRAMEOKADDITIONALINFO : **st25r95_com.h**

- ST25R95_ERRCODE_FRAMEWAITTIMEOUT : **st25r95_com.h**
- ST25R95_ERRCODE_FRAMING : **st25r95_com.h**
- ST25R95_ERRCODE_INVALIDCCMDCODE : **st25r95_com.h**
- ST25R95_ERRCODE_INVALIDCMDLENGHT : **st25r95_com.h**
- ST25R95_ERRCODE_INVALIDPROTOCOL : **st25r95_com.h**
- ST25R95_ERRCODE_INVALIDSOF : **st25r95_com.h**
- ST25R95_ERRCODE_NOFIELD : **st25r95_com.h**
- ST25R95_ERRCODE_NONE : **st25r95_com.h**
- ST25R95_ERRCODE_OVERFLOW : **st25r95_com.h**
- ST25R95_ERRCODE_RECEPTIONLOST : **st25r95_com.h**
- ST25R95_ERRCODE_RESULTSRESIDUAL : **st25r95_com.h**
- ST25R95_FWT_MAX : **st25r95_com.h**
- ST25R95_IDLE_DACDATAH_OFFSET : **st25r95_com.h**
- ST25R95_IDLE_DACDATAL_OFFSET : **st25r95_com.h**
- ST25R95_IDLE_RESPONSE_BUFLEN : **st25r95_com.h**
- ST25R95_IDLE_WKUP_TAGDETECT : **st25r95_com.h**
- ST25R95_IDLE_WKUP_TIMEOUT : **st25r95_com.h**
- ST25R95_IDLE_WUPERIOD_OFFSET : **st25r95_com.h**
- ST25R95_IDN_RESPONSE_BUFLEN : **st25r95_com.h**
- ST25R95_IS_PROT_ISO14443A_COLLISION_ERR : **st25r95_com.h**
- ST25R95_IS_PROT_ISO14443A_CRC_ERR : **st25r95_com.h**
- ST25R95_IS_PROT_ISO14443A_PARITY_ERR : **st25r95_com.h**
- ST25R95_IS_PROT_ISO14443B_CRC_ERR : **st25r95_com.h**
- ST25R95_IS_PROT_ISO15693_COLLISION_ERR : **st25r95_com.h**
- ST25R95_IS_PROT_ISO15693_CRC_ERR : **st25r95_com.h**
- ST25R95_IS_PROT_ISO18092_CRC_ERR : **st25r95_com.h**
- ST25R95_LISTEN_RESPONSE_BUFLEN : **st25r95_com.h**
- st25r95_nIRQ_IN_Pulse : **st25r95.h**
- ST25R95_POLLFIELD_RESPONSE_BUFLEN : **st25r95_com.h**
- ST25R95_PROTOCOL_CE_ISO14443A : **st25r95_com.h**
- ST25R95_PROTOCOL_FIELDOFF : **st25r95_com.h**
- ST25R95_PROTOCOL_ISO14443A : **st25r95_com.h**
- ST25R95_PROTOCOL_ISO14443B : **st25r95_com.h**
- ST25R95_PROTOCOL_ISO15693 : **st25r95_com.h**
- ST25R95_PROTOCOL_ISO18092 : **st25r95_com.h**
- ST25R95_PROTOCOL_MAX : **st25r95_com.h**
- ST25R95_PROTOCOLSELECT_BR_OFFSET : **st25r95_com.h**
- ST25R95_PROTOCOLSELECT_RESPONSE_BUFLEN : **st25r95_com.h**
- ST25R95_RDREG_RESPONSE_BUFLEN : **st25r95_com.h**

- ST25R95_REG_ACC_A : **st25r95_com.h**
- ST25R95_REG_ARC_B : **st25r95_com.h**
- ST25R95_REG_TIMERW : **st25r95_com.h**
- ST25R95_SEND_RESPONSE_BUFLEN : **st25r95_com.h**
- ST25R95_SPI_DUMMY_BYTE : **st25r95_com.h**
- ST25R95_WRREG_RESPONSE_BUFLEN : **st25r95_com.h**
- st25r95CalibrateTagDetector() : **st25r95_com.h**
- st25r95CheckChipID() : **st25r95.h**
- st25r95CommandEcho : **st25r95_com.h**
- st25r95CompleteRx : **st25r95_com.h**
- st25r95DeactivateACFilter : **st25r95_com.h**
- st25r95Deinitialize() : **st25r95.h**
- st25r95FieldOff() : **st25r95.h**
- st25r95FieldOn() : **st25r95.h**
- st25r95FlushChipSPIBuffer : **st25r95_com.h**
- st25r95GetIdleResponse : **st25r95_com.h**
- st25r95GetLmState : **st25r95_com.h**
- st25r95Idle : **st25r95_com.h**
- st25r95Initialize() : **st25r95.h**
- st25r95IsInListen : **st25r95_com.h**
- st25r95IsTransmitTxCompleted : **st25r95_com.h**
- st25r95KillIdle : **st25r95_com.h**
- st25r95Listen() : **st25r95_com.h**
- st25r95PollField() : **st25r95_com.h**
- st25r95PollRead : **st25r95_com.h**
- st25r95PrepareRx : **st25r95_com.h**
- st25r95ProtocolSelect() : **st25r95_com.h**
- st25r95ReadReg() : **st25r95_com.h**
- st25r95RearmListen() : **st25r95_com.h**
- st25r95ResetChip : **st25r95.h**
- st25r95SendCommandTypeAndLen : **st25r95_com.h**
- st25r95SendData : **st25r95_com.h**
- st25r95SendTransmitFlag : **st25r95_com.h**
- st25r95SetACFilter() : **st25r95_com.h**
- st25r95SetACState : **st25r95_com.h**
- st25r95SetBitRate() : **st25r95_com.h**
- st25r95SetFWT() : **st25r95_com.h**
- st25r95SetSlotCounter() : **st25r95_com.h**
- st25r95SPI_nIRQ_IN_Pulse() : **st25r95.h**

- st25r95SPICommandEcho() : **st25r95_com.h**
- st25r95SPICompleteRx() : **st25r95_com.h**
- st25r95SPIDeactivateACFilter() : **st25r95_com.h**
- st25r95SPIGetIdleResponse() : **st25r95_com.h**
- st25r95SPIGetLmState() : **st25r95_com.h**
- st25r95SPIIdle() : **st25r95_com.h**
- st25r95SPIIsInListen() : **st25r95_com.h**
- st25r95SPIIsTransmitCompleted() : **st25r95_com.h**
- st25r95SPIKillIdle() : **st25r95_com.h**
- st25r95SPIPollRead() : **st25r95_com.h**
- st25r95SPIPollSend() : **st25r95_com.h**
- st25r95SPIPrepareRx() : **st25r95_com.h**
- st25r95SPIResetChip() : **st25r95_com.h**
- st25r95SPIRxCtx : **st25r95_com.h**
- st25r95SPIRxTx() : **st25r95_com.h**
- st25r95SPISendCommandTypeAndLen() : **st25r95_com.h**
- st25r95SPISendData() : **st25r95_com.h**
- st25r95SPISendReceiveByte() : **st25r95_com.h**
- st25r95SPISendTransmitFlag() : **st25r95_com.h**
- st25r95SPISetACState() : **st25r95_com.h**
- st25r95UART_nIRQ_IN_Pulse() : **st25r95.h**
- st25r95UARTCommandEcho() : **st25r95_com.h**
- st25r95UARTCompleteRx() : **st25r95_com.h**
- st25r95UartErrorCallback() : **st25r95_com.h**
- st25r95UARTGetIdleResponse() : **st25r95_com.h**
- st25r95UARTIdle() : **st25r95_com.h**
- st25r95UARTIsTransmitTxCompleted() : **st25r95_com.h**
- st25r95UARTKillIdle() : **st25r95_com.h**
- st25r95UARTPollRead() : **st25r95_com.h**
- st25r95UARTPrepareRx() : **st25r95_com.h**
- st25r95UartRxCpltCallback() : **st25r95_com.h**
- st25r95UARTSendCommandTypeAndLen() : **st25r95_com.h**
- st25r95UARTSendData() : **st25r95_com.h**
- st25r95UARTSendTransmitFlag() : **st25r95_com.h**
- st25r95UartTxCpltCallback() : **st25r95_com.h**
- st25r95WriteReg() : **st25r95_com.h**
- ST25R_COM_SINGLETXRX : **rfal_platform.h**
- ST25R_INT_PIN : **rfal_platform.h**
- ST25R_INT_PORT : **rfal_platform.h**

- ST25R_RESET_PIN : **rfal_platform.h**
- ST25R_RESET_PORT : **rfal_platform.h**
- ST25R_SS_PIN : **rfal_platform.h**
- ST25R_SS_PORT : **rfal_platform.h**

Generated on Fri Mar 14 2025 11:45:12 for RFAL by doxygen 1.9.5

RFAL

Main Page

Modules

Data Structures

Examples

File List

All

Variables

Typedefs

Enumerations

Enumerator

Macros

s

- r -

- rfalAdjustRegulators() : **rfal_rf.h**
- rfalAnalogConfigGenModeID() : **rfal_analogConfig.h**
- rfalAnalogConfigInitialize() : **rfal_analogConfig.h**
- rfalAnalogConfigIsReady() : **rfal_analogConfig.h**
- rfalAnalogConfigListRead() : **rfal_analogConfig.h**
- rfalAnalogConfigListReadRaw() : **rfal_analogConfig.h**
- rfalAnalogConfigListWrite() : **rfal_analogConfig.h**
- rfalAnalogConfigListWriteRaw() : **rfal_analogConfig.h**
- rfalCalibrate() : **rfal_rf.h**
- rfalCdDetectCard() : **rfal_cd.h**
- rfalCdGetDetectCardStatus() : **rfal_cd.h**
- rfalCdStartDetectCard() : **rfal_cd.h**
- rfalChipChangeRegBits() : **rfal_chip.h**
- rfalChipChangeTestRegBits() : **rfal_chip.h**
- rfalChipExecCmd() : **rfal_chip.h**
- rfalChipGetLmFieldInd() : **rfal_chip.h**
- rfalChipGetLMMod() : **rfal_chip.h**
- rfalChipGetRFO() : **rfal_chip.h**
- rfalChipMeasureAmplitude() : **rfal_chip.h**
- rfalChipMeasureCapacitance() : **rfal_chip.h**
- rfalChipMeasureCombinedIQ() : **rfal_chip.h**
- rfalChipMeasureCurrent() : **rfal_chip.h**
- rfalChipMeasureI() : **rfal_chip.h**
- rfalChipMeasureIQ() : **rfal_chip.h**
- rfalChipMeasurePhase() : **rfal_chip.h**
- rfalChipMeasurePowerSupply() : **rfal_chip.h**
- rfalChipMeasureQ() : **rfal_chip.h**
- rfalChipReadReg() : **rfal_chip.h**
- rfalChipReadTestReg() : **rfal_chip.h**
- rfalChipSetAntennaMode() : **rfal_chip.h**
- rfalChipSetLMMod() : **rfal_chip.h**
- rfalChipSetRFO() : **rfal_chip.h**
- rfalChipWriteReg() : **rfal_chip.h**
- rfalChipWriteTestReg() : **rfal_chip.h**
- rfalCrcCalculateCcitt() : **rfal_crc.h**
- rfalDeinitialize() : **rfal_rf.h**
- rfalDisableObsvMode() : **rfal_rf.h**

- rfalDlmaAdjust() : **rfal_dlma.h**
- rfalDlmaGetInfo() : **rfal_dlma.h**
- rfalDlmaInitialize() : **rfal_dlma.h**
- rfalDlmaIsEnabled() : **rfal_dlma.h**
- rfalDlmaSetAdjustCallback() : **rfal_dlma.h**
- rfalDlmaSetEnabled() : **rfal_dlma.h**
- rfalDlmaSetMeasureCallback() : **rfal_dlma.h**
- rfalDlmaTableRead() : **rfal_dlma.h**
- rfalDlmaTableWrite() : **rfal_dlma.h**
- rfalDpoAdjust() : **rfal_dpo.h**
- rfalDpoGetInfo() : **rfal_dpo.h**
- rfalDpoInitialize() : **rfal_dpo.h**
- rfalDpoIsEnabled() : **rfal_dpo.h**
- rfalDpoReqAdj() : **rfal_dpo.h**
- rfalDpoSetAdjustCallback() : **rfal_dpo.h**
- rfalDpoSetEnabled() : **rfal_dpo.h**
- rfalDpoSetMeasureCallback() : **rfal_dpo.h**
- rfalDpoTableRead() : **rfal_dpo.h**
- rfalDpoTableWrite() : **rfal_dpo.h**
- rfalFeliCaPoll() : **rfal_rf.h**
- rfalFieldOff() : **rfal_rf.h**
- rfalFieldOnAndStartGT() : **rfal_rf.h**
- rfalGetBitRate() : **rfal_rf.h**
- rfalGetErrorHandling() : **rfal_rf.h**
- rfalGetFDTListen() : **rfal_rf.h**
- rfalGetFDTPoll() : **rfal_rf.h**
- rfalGetFeliCaPollStatus() : **rfal_rf.h**
- rfalGetGT() : **rfal_rf.h**
- rfalGetMode() : **rfal_rf.h**
- rfalGetObsvMode() : **rfal_rf.h**
- rfalGetTransceiveRSSI() : **rfal_rf.h**
- rfalGetTransceiveState() : **rfal_rf.h**
- rfalGetTransceiveStatus() : **rfal_rf.h**
- rfalInitialize() : **rfal_rf.h**
- rfalIsExtFieldOn() : **rfal_rf.h**
- rfalIsGTExpired() : **rfal_rf.h**
- rfalISO14443AGetTransceiveAnticollisionFrameStatus() : **rfal_rf.h**
- rfalISO14443AStartTransceiveAnticollisionFrame() : **rfal_rf.h**
- rfalISO14443ATransceiveAnticollisionFrame() : **rfal_rf.h**

- rfalISO14443ATransceiveShortFrame() : **rfal_rf.h**
- rfalIso15693PhyConfigure() : **rfal_iso15693_2.h**
- rfalIso15693PhyGetConfiguration() : **rfal_iso15693_2.h**
- rfalISO15693TransceiveAnticollisionFrame() : **rfal_rf.h**
- rfalISO15693TransceiveEOF() : **rfal_rf.h**
- rfalISO15693TransceiveEOFAnticollision() : **rfal_rf.h**
- rfalIso15693VCDCode() : **rfal_iso15693_2.h**
- rfalIso15693VICCDecode() : **rfal_iso15693_2.h**
- rfalIsoDepATTRIB() : **rfal_isoDep.h**
- rfalIsoDepDeselect() : **rfal_isoDep.h**
- rfalIsoDepFSxI2FSx() : **rfal_isoDep.h**
- rfalIsoDepFWI2FWT() : **rfal_isoDep.h**
- rfalIsoDepGetApduTransceiveStatus() : **rfal_isoDep.h**
- rfalIsoDepGetDeselectStatus() : **rfal_isoDep.h**
- rfalIsoDepGetMaxInfLen() : **rfal_isoDep.h**
- rfalIsoDepGetTransceiveStatus() : **rfal_isoDep.h**
- rfalIsoDepInitialize() : **rfal_isoDep.h**
- rfalIsoDepInitializeWithParams() : **rfal_isoDep.h**
- rfalIsoDepIsAttrib() : **rfal_isoDep.h**
- rfalIsoDepIsRats() : **rfal_isoDep.h**
- rfalIsoDepListenGetActivationStatus() : **rfal_isoDep.h**
- rfalIsoDepListenStartActivation() : **rfal_isoDep.h**
- rfalIsoDepPollAGetActivationStatus() : **rfal_isoDep.h**
- rfalIsoDepPollAHandleActivation() : **rfal_isoDep.h**
- rfalIsoDepPollAStartActivation() : **rfal_isoDep.h**
- rfalIsoDepPollBGetActivationStatus() : **rfal_isoDep.h**
- rfalIsoDepPollBHandleActivation() : **rfal_isoDep.h**
- rfalIsoDepPollBStartActivation() : **rfal_isoDep.h**
- rfalIsoDepPollHandleSParameters() : **rfal_isoDep.h**
- rfalIsoDepPPS() : **rfal_isoDep.h**
- rfalIsoDepRATS() : **rfal_isoDep.h**
- rfalIsoDepStartApduTransceive() : **rfal_isoDep.h**
- rfalIsoDepStartDeselect() : **rfal_isoDep.h**
- rfalIsoDepStartTransceive() : **rfal_isoDep.h**
- rfalIsTransceiveInRx() : **rfal_rf.h**
- rfalIsTransceiveInTx() : **rfal_rf.h**
- rfalIsTransceiveSubcDetected() : **rfal_rf.h**
- rfalListenGetState() : **rfal_rf.h**
- rfalListenSetState() : **rfal_rf.h**

- rfalListenSleepStart() : **rfal_rf.h**
- rfalListenStart() : **rfal_rf.h**
- rfalListenStop() : **rfal_rf.h**
- rfalLowPowerModeStart() : **rfal_rf.h**
- rfalLowPowerModeStop() : **rfal_rf.h**
- rfalNfcaListenerIsSleepReq() : **rfal_nfca.h**
- rfalNfcaPollerCheckPresence() : **rfal_nfca.h**
- rfalNfcaPollerFullCollisionResolution() : **rfal_nfca.h**
- rfalNfcaPollerGetFullCollisionResolutionStatus() : **rfal_nfca.h**
- rfalNfcaPollerGetSelectStatus() : **rfal_nfca.h**
- rfalNfcaPollerGetSleepStatus() : **rfal_nfca.h**
- rfalNfcaPollerGetTechnologyDetectionStatus() : **rfal_nfca.h**
- rfalNfcaPollerInitialize() : **rfal_nfca.h**
- rfalNfcaPollerSelect() : **rfal_nfca.h**
- rfalNfcaPollerSingleCollisionResolution() : **rfal_nfca.h**
- rfalNfcaPollerSleep() : **rfal_nfca.h**
- rfalNfcaPollerSleepFullCollisionResolution() : **rfal_nfca.h**
- rfalNfcaPollerStartFullCollisionResolution() : **rfal_nfca.h**
- rfalNfcaPollerStartSelect() : **rfal_nfca.h**
- rfalNfcaPollerStartSleep() : **rfal_nfca.h**
- rfalNfcaPollerStartTechnologyDetection() : **rfal_nfca.h**
- rfalNfcaPollerTechnologyDetection() : **rfal_nfca.h**
- rfalNfcbPollerCheckPresence() : **rfal_nfcb.h**
- rfalNfcbPollerCollisionResolution() : **rfal_nfcb.h**
- rfalNfcbPollerGetCheckPresenceStatus() : **rfal_nfcb.h**
- rfalNfcbPollerGetCollisionResolutionStatus() : **rfal_nfcb.h**
- rfalNfcbPollerGetSlotMarkerStatus() : **rfal_nfcb.h**
- rfalNfcbPollerGetTechnologyDetectionStatus() : **rfal_nfcb.h**
- rfalNfcbPollerInitialize() : **rfal_nfcb.h**
- rfalNfcbPollerInitializeWithParams() : **rfal_nfcb.h**
- rfalNfcbPollerSleep() : **rfal_nfcb.h**
- rfalNfcbPollerSlotMarker() : **rfal_nfcb.h**
- rfalNfcbPollerSlottedCollisionResolution() : **rfal_nfcb.h**
- rfalNfcbPollerStartCheckPresence() : **rfal_nfcb.h**
- rfalNfcbPollerStartCollisionResolution() : **rfal_nfcb.h**
- rfalNfcbPollerStartSlotMarker() : **rfal_nfcb.h**
- rfalNfcbPollerStartSlottedCollisionResolution() : **rfal_nfcb.h**
- rfalNfcbPollerStartTechnologyDetection() : **rfal_nfcb.h**
- rfalNfcbPollerTechnologyDetection() : **rfal_nfcb.h**

- rfalNfcbTR2ToFDT() : **rfal_nfcb.h**
- rfalNfcDataExchangeGetStatus() : **rfal_nfc.h**
- rfalNfcDataExchangeStart() : **rfal_nfc.h**
- rfalNfcDeactivate() : **rfal_nfc.h**
- rfalNfcDepATR() : **rfal_nfcDep.h**
- rfalNfcDepCalculateRWT() : **rfal_nfcDep.h**
- rfalNfcDepDSL() : **rfal_nfcDep.h**
- rfalNfcDepGetPduTransceiveStatus() : **rfal_nfcDep.h**
- rfalNfcDepGetTransceiveStatus() : **rfal_nfcDep.h**
- rfalNfcDepInitialize() : **rfal_nfcDep.h**
- rfalNfcDepInitiatorHandleActivation() : **rfal_nfcDep.h**
- rfalNfcDepIsAtrReq() : **rfal_nfcDep.h**
- rfalNfcDepListenGetActivationStatus() : **rfal_nfcDep.h**
- rfalNfcDepListenStartActivation() : **rfal_nfcDep.h**
- rfalNfcDepPSL() : **rfal_nfcDep.h**
- rfalNfcDepRLS() : **rfal_nfcDep.h**
- rfalNfcDepSetDeactivatingCallback() : **rfal_nfcDep.h**
- rfalNfcDepStartPduTransceive() : **rfal_nfcDep.h**
- rfalNfcDepStartTransceive() : **rfal_nfcDep.h**
- rfalNfcDepTargetRcvdATR() : **rfal_nfcDep.h**
- rfalNfcDiscover() : **rfal_nfc.h**
- rfalNfcfListenerIsT3TReq() : **rfal_nfcf.h**
- rfalNfcfPollerCheck() : **rfal_nfcf.h**
- rfalNfcfPollerCheckPresence() : **rfal_nfcf.h**
- rfalNfcfPollerCollisionResolution() : **rfal_nfcf.h**
- rfalNfcfPollerGetCheckPresenceStatus() : **rfal_nfcf.h**
- rfalNfcfPollerGetCollisionResolutionStatus() : **rfal_nfcf.h**
- rfalNfcfPollerInitialize() : **rfal_nfcf.h**
- rfalNfcfPollerPoll() : **rfal_nfcf.h**
- rfalNfcfPollerStartCheckPresence() : **rfal_nfcf.h**
- rfalNfcfPollerStartCollisionResolution() : **rfal_nfcf.h**
- rfalNfcfPollerUpdate() : **rfal_nfcf.h**
- rfalNfcGetActiveDevice() : **rfal_nfc.h**
- rfalNfcGetDevicesFound() : **rfal_nfc.h**
- rfalNfcGetState() : **rfal_nfc.h**
- rfalNfcInitialize() : **rfal_nfc.h**
- rfalNfcSelect() : **rfal_nfc.h**
- rfalNfcvPollerCheckPresence() : **rfal_nfcv.h**
- rfalNfcvPollerCollisionResolution() : **rfal_nfcv.h**

- rfalNfcvPollerExtendedGetSystemInformation() : **rfal_nfcv.h**
- rfalNfcvPollerExtendedLockSingleBlock() : **rfal_nfcv.h**
- rfalNfcvPollerExtendedReadMultipleBlocks() : **rfal_nfcv.h**
- rfalNfcvPollerExtendedReadSingleBlock() : **rfal_nfcv.h**
- rfalNfcvPollerExtendedWriteMultipleBlocks() : **rfal_nfcv.h**
- rfalNfcvPollerExtendedWriteSingleBlock() : **rfal_nfcv.h**
- rfalNfcvPollerGetSystemInformation() : **rfal_nfcv.h**
- rfalNfcvPollerInitialize() : **rfal_nfcv.h**
- rfalNfcvPollerInventory() : **rfal_nfcv.h**
- rfalNfcvPollerLockBlock() : **rfal_nfcv.h**
- rfalNfcvPollerReadMultipleBlocks() : **rfal_nfcv.h**
- rfalNfcvPollerReadSingleBlock() : **rfal_nfcv.h**
- rfalNfcvPollerSelect() : **rfal_nfcv.h**
- rfalNfcvPollerSleep() : **rfal_nfcv.h**
- rfalNfcvPollerSleepCollisionResolution() : **rfal_nfcv.h**
- rfalNfcvPollerTransceiveReq() : **rfal_nfcv.h**
- rfalNfcvPollerWriteMultipleBlocks() : **rfal_nfcv.h**
- rfalNfcvPollerWriteSingleBlock() : **rfal_nfcv.h**
- rfalNfcWorker() : **rfal_nfc.h**
- rfalSetAnalogConfig() : **rfal_analogConfig.h**
- rfalSetBitRate() : **rfal_rf.h**
- rfalSetErrorHandling() : **rfal_rf.h**
- rfalSetFDTListen() : **rfal_rf.h**
- rfalSetFDTPoll() : **rfal_rf.h**
- rfalSetGT() : **rfal_rf.h**
- rfalSetLmEonCallback() : **rfal_rf.h**
- rfalSetMode() : **rfal_rf.h**
- rfalSetObsvMode() : **rfal_rf.h**
- rfalSetPostTxRxCallback() : **rfal_rf.h**
- rfalSetPreTxRxCallback() : **rfal_rf.h**
- rfalSetRegulators() : **rfal_rf.h**
- rfalSetSyncTxRxCallback() : **rfal_rf.h**
- rfalSetUpperLayerCallback() : **rfal_rf.h**
- rfalSt25tbPollerCheckPresence() : **rfal_st25tb.h**
- rfalSt25tbPollerCollisionResolution() : **rfal_st25tb.h**
- rfalSt25tbPollerCompletion() : **rfal_st25tb.h**
- rfalSt25tbPollerGetUID() : **rfal_st25tb.h**
- rfalSt25tbPollerInitialize() : **rfal_st25tb.h**
- rfalSt25tbPollerInitiate() : **rfal_st25tb.h**

- rfalSt25tbPollerPcall() : **rfal_st25tb.h**
- rfalSt25tbPollerReadBlock() : **rfal_st25tb.h**
- rfalSt25tbPollerResetToInventory() : **rfal_st25tb.h**
- rfalSt25tbPollerSelect() : **rfal_st25tb.h**
- rfalSt25tbPollerSlotMarker() : **rfal_st25tb.h**
- rfalSt25tbPollerWriteBlock() : **rfal_st25tb.h**
- rfalST25TV512CPollerReadConfiguration() : **rfal_st25xv.h**
- rfalST25TV512CPollerWriteConfiguration() : **rfal_st25xv.h**
- rfalST25xVPollerFastExtendedReadSingleBlock() : **rfal_st25xv.h**
- rfalST25xVPollerFastExtReadMultipleBlocks() : **rfal_st25xv.h**
- rfalST25xVPollerFastReadDynamicConfiguration() : **rfal_st25xv.h**
- rfalST25xVPollerFastReadMessage() : **rfal_st25xv.h**
- rfalST25xVPollerFastReadMsgLength() : **rfal_st25xv.h**
- rfalST25xVPollerFastReadMultipleBlocks() : **rfal_st25xv.h**
- rfalST25xVPollerFastReadSingleBlock() : **rfal_st25xv.h**
- rfalST25xVPollerFastWriteDynamicConfiguration() : **rfal_st25xv.h**
- rfalST25xVPollerFastWriteMessage() : **rfal_st25xv.h**
- rfalST25xVPollerGetRandomNumber() : **rfal_st25xv.h**
- rfalST25xVPollerM24LRFastReadMultipleBlocks() : **rfal_st25xv.h**
- rfalST25xVPollerM24LRFastReadSingleBlock() : **rfal_st25xv.h**
- rfalST25xVPollerM24LRReadMultipleBlocks() : **rfal_st25xv.h**
- rfalST25xVPollerM24LRReadSingleBlock() : **rfal_st25xv.h**
- rfalST25xVPollerM24LRWriteSingleBlock() : **rfal_st25xv.h**
- rfalST25xVPollerPresentPassword() : **rfal_st25xv.h**
- rfalST25xVPollerReadConfiguration() : **rfal_st25xv.h**
- rfalST25xVPollerReadDynamicConfiguration() : **rfal_st25xv.h**
- rfalST25xVPollerReadMessage() : **rfal_st25xv.h**
- rfalST25xVPollerReadMessageLength() : **rfal_st25xv.h**
- rfalST25xVPollerWriteConfiguration() : **rfal_st25xv.h**
- rfalST25xVPollerWriteDynamicConfiguration() : **rfal_st25xv.h**
- rfalST25xVPollerWriteMessage() : **rfal_st25xv.h**
- rfalST25xVPollerWritePassword() : **rfal_st25xv.h**
- rfalStartFeliCaPoll() : **rfal_rf.h**
- rfalStartTransceive() : **rfal_rf.h**
- rfalT1TPollerInitialize() : **rfal_t1t.h**
- rfalT1TPollerRall() : **rfal_t1t.h**
- rfalT1TPollerRid() : **rfal_t1t.h**
- rfalT1TPollerWrite() : **rfal_t1t.h**
- rfalT2TPollerRead() : **rfal_t2t.h**

- rfalT2TPollerSectorSelect() : **rfal_t2t.h**
- rfalT2TPollerWrite() : **rfal_t2t.h**
- rfalT4TPollerComposeCAPDU() : **rfal_t4t.h**
- rfalT4TPollerComposeReadData() : **rfal_t4t.h**
- rfalT4TPollerComposeReadDataODO() : **rfal_t4t.h**
- rfalT4TPollerComposeSelectAppl() : **rfal_t4t.h**
- rfalT4TPollerComposeSelectFile() : **rfal_t4t.h**
- rfalT4TPollerComposeSelectFileV1Mapping() : **rfal_t4t.h**
- rfalT4TPollerComposeWriteData() : **rfal_t4t.h**
- rfalT4TPollerComposeWriteDataODO() : **rfal_t4t.h**
- rfalT4TPollerParseRAPDU() : **rfal_t4t.h**
- rfalTransceiveBlockingRx() : **rfal_rf.h**
- rfalTransceiveBlockingTx() : **rfal_rf.h**
- rfalTransceiveBlockingTxRx() : **rfal_rf.h**
- rfalWakeUpModeGetInfo() : **rfal_rf.h**
- rfalWakeUpModeHasWoke() : **rfal_rf.h**
- rfalWakeUpModeIsEnabled() : **rfal_rf.h**
- rfalWakeUpModeStart() : **rfal_rf.h**
- rfalWakeUpModeStop() : **rfal_rf.h**
- rfalWlcPWptIsFodDetected() : **rfal_rf.h**
- rfalWlcPWptIsStopDetected() : **rfal_rf.h**
- rfalWlcPWptMonitorStart() : **rfal_rf.h**
- rfalWlcPWptMonitorStop() : **rfal_rf.h**
- rfalWorker() : **rfal_rf.h**

RFAL

Main Page

Modules

Data Structures

Examples

- s -

- st25r200AdjustRegulators() : **st25r200.h**
- st25r200CalibrateWU() : **st25r200.h**
- st25r200ChangeRegisterBits() : **st25r200_com.h**
- st25r200ChangeTestRegisterBits() : **st25r200_com.h**
- st25r200CheckChipID() : **st25r200.h**
- st25r200CheckForReceivedInterrupts() : **st25r200_irq.h**
- st25r200CheckReg() : **st25r200_com.h**
- st25r200ClearAndEnableInterrupts() : **st25r200_irq.h**
- st25r200ClearCalibration() : **st25r200.h**
- st25r200ClearInterrupts() : **st25r200_irq.h**
- st25r200ClrRegisterBits() : **st25r200_com.h**
- st25r200Deinitialize() : **st25r200.h**
- st25r200DisableInterrupts() : **st25r200_irq.h**
- st25r200EnableInterrupts() : **st25r200_irq.h**
- st25r200ExecuteCommand() : **st25r200_com.h**
- st25r200ExecuteCommandAndGetResult() : **st25r200.h**
- st25r200ExecuteCommands() : **st25r200_com.h**

- st25r200GetInterrupt() : **st25r200_irq.h**
- st25r200GetNoResponseTime() : **st25r200.h**
- st25r200GetNumFIFOBytes() : **st25r200.h**
- st25r200GetNumFIFOLastBits() : **st25r200.h**
- st25r200GetRegsDump() : **st25r200.h**
- st25r200GetRSSI() : **st25r200.h**
- st25r200Initialize() : **st25r200.h**
- st25r200InitInterrupts() : **st25r200_irq.h**
- st25r200IRQCallbackRestore() : **st25r200_irq.h**
- st25r200IRQCallbackSet() : **st25r200_irq.h**
- st25r200IsCmdValid() : **st25r200.h**
- st25r200IsExtFieldOn() : **st25r200.h**
- st25r200Isr() : **st25r200_irq.h**
- st25r200IsRegValid() : **st25r200_com.h**
- st25r200ledEvtCmd() : **st25r200_com.h**
- st25r200ledEvtCmds() : **st25r200_com.h**
- st25r200ledEvtIrq() : **st25r200_com.h**
- st25r200ledEvtWrMultiReg() : **st25r200_com.h**
- st25r200ledEvtWrReg() : **st25r200_com.h**
- st25r200ledInit() : **st25r200_com.h**
- st25r200MeasureCombinedIQ() : **st25r200.h**
- st25r200MeasureI() : **st25r200.h**
- st25r200MeasureIQ() : **st25r200.h**
- st25r200MeasureQ() : **st25r200.h**
- st25r200MeasureWU() : **st25r200.h**
- st25r200ModifyInterrupts() : **st25r200_irq.h**
- st25r200ModifyRegister() : **st25r200_com.h**
- st25r200OscOn() : **st25r200.h**
- st25r200PerformCollisionAvoidance() : **st25r200.h**
- st25r200ReadFifo() : **st25r200_com.h**
- st25r200ReadMultipleRegisters() : **st25r200_com.h**
- st25r200ReadRegister() : **st25r200_com.h**
- st25r200ReadTestRegister() : **st25r200_com.h**
- st25r200SetAntennaMode() : **st25r200.h**
- st25r200SetBitrate() : **st25r200.h**
- st25r200SetGPTime() : **st25r200.h**
- st25r200SetNoResponseTime() : **st25r200.h**
- st25r200SetNumTxBits() : **st25r200.h**
- st25r200SetRegisterBits() : **st25r200_com.h**

- st25r200SetRegulators() : **st25r200.h**
- st25r200SetStartGPTimer() : **st25r200.h**
- st25r200SetStartNoResponseTimer() : **st25r200.h**
- st25r200WaitForInterruptsTimed() : **st25r200_irq.h**
- st25r200WriteFifo() : **st25r200_com.h**
- st25r200WriteMultipleRegisters() : **st25r200_com.h**
- st25r200WriteMultipleTestRegister() : **st25r200_com.h**
- st25r200WriteRegister() : **st25r200_com.h**
- st25r200WriteTestRegister() : **st25r200_com.h**
- st25r3911AdjustRegulators() : **st25r3911.h**
- st25r3911CalibrateAntenna() : **st25r3911.h**
- st25r3911CalibrateCapacitiveSensor() : **st25r3911.h**
- st25r3911CalibrateModulationDepth() : **st25r3911.h**
- st25r3911ChangeRegisterBits() : **st25r3911_com.h**
- st25r3911ChangeTestRegisterBits() : **st25r3911_com.h**
- st25r3911CheckChipID() : **st25r3911.h**
- st25r3911CheckForReceivedInterrupts() : **st25r3911_interrupt.h**
- st25r3911CheckReg() : **st25r3911.h**
- st25r3911ClearInterrupts() : **st25r3911_interrupt.h**
- st25r3911ClrRegisterBits() : **st25r3911_com.h**
- st25r3911Deinitialize() : **st25r3911.h**
- st25r3911DisableInterrupts() : **st25r3911_interrupt.h**
- st25r3911EnableInterrupts() : **st25r3911_interrupt.h**
- st25r3911ExecuteCommand() : **st25r3911_com.h**
- st25r3911ExecuteCommands() : **st25r3911_com.h**
- st25r3911GetInterrupt() : **st25r3911_interrupt.h**
- st25r3911GetNoResponseTime_64fcs() : **st25r3911.h**
- st25r3911GetNumFIFOLastBits() : **st25r3911.h**
- st25r3911GetRegsDump() : **st25r3911.h**
- st25r3911GetRSSI() : **st25r3911.h**
- st25r3911Initialize() : **st25r3911.h**
- st25r3911InitInterrupts() : **st25r3911_interrupt.h**
- st25r3911IRQCallbackRestore() : **st25r3911_interrupt.h**
- st25r3911IRQCallbackSet() : **st25r3911_interrupt.h**
- st25r3911IsCmdValid() : **st25r3911.h**
- st25r3911Isr() : **st25r3911_interrupt.h**
- st25r3911IsRegValid() : **st25r3911_com.h**
- st25r3911MeasureAmplitude() : **st25r3911.h**
- st25r3911MeasureCapacitance() : **st25r3911.h**

- st25r3911MeasurePhase() : **st25r3911.h**
- st25r3911MeasurePowerSupply() : **st25r3911.h**
- st25r3911MeasureVoltage() : **st25r3911.h**
- st25r3911ModifyInterrupts() : **st25r3911_interrupt.h**
- st25r3911ModifyRegister() : **st25r3911_com.h**
- st25r3911OscOn() : **st25r3911.h**
- st25r3911PerformCollisionAvoidance() : **st25r3911.h**
- st25r3911ReadFifo() : **st25r3911_com.h**
- st25r3911ReadMultipleRegisters() : **st25r3911_com.h**
- st25r3911ReadRegister() : **st25r3911_com.h**
- st25r3911ReadTestRegister() : **st25r3911_com.h**
- st25r3911SetAntennaMode() : **st25r3911.h**
- st25r3911SetBitrate() : **st25r3911.h**
- st25r3911SetGPTime_8fcs() : **st25r3911.h**
- st25r3911SetNoResponseTime_64fcs() : **st25r3911.h**
- st25r3911SetNumTxBits() : **st25r3911.h**
- st25r3911SetRegisterBits() : **st25r3911_com.h**
- st25r3911SetRegulators() : **st25r3911.h**
- st25r3911SetStartNoResponseTime_64fcs() : **st25r3911.h**
- st25r3911StartGPTimer_8fcs() : **st25r3911.h**
- st25r3911StreamConfigure() : **st25r3911.h**
- st25r3911TxRxOff() : **st25r3911.h**
- st25r3911TxRxOn() : **st25r3911.h**
- st25r3911WaitForInterruptsTimed() : **st25r3911_interrupt.h**
- st25r3911WriteFifo() : **st25r3911_com.h**
- st25r3911WriteMultipleRegisters() : **st25r3911_com.h**
- st25r3911WriteRegister() : **st25r3911_com.h**
- st25r3911WriteTestRegister() : **st25r3911_com.h**
- st25r3916AatTune() : **st25r3916_aat.h**
- st25r3916AdjustRegulators() : **st25r3916.h**
- st25r3916CalibrateCapacitiveSensor() : **st25r3916.h**
- st25r3916ChangeRegisterBits() : **st25r3916_com.h**
- st25r3916ChangeTestRegisterBits() : **st25r3916_com.h**
- st25r3916CheckChipID() : **st25r3916.h**
- st25r3916CheckForReceivedInterrupts() : **st25r3916_irq.h**
- st25r3916CheckReg() : **st25r3916_com.h**
- st25r3916ClearAndEnableInterrupts() : **st25r3916_irq.h**
- st25r3916ClearInterrupts() : **st25r3916_irq.h**
- st25r3916ClrRegisterBits() : **st25r3916_com.h**

- st25r3916Deinitialize() : **st25r3916.h**
- st25r3916DisableInterrupts() : **st25r3916_irq.h**
- st25r3916EnableInterrupts() : **st25r3916_irq.h**
- st25r3916ExecuteCommand() : **st25r3916_com.h**
- st25r3916ExecuteCommandAndGetResult() : **st25r3916.h**
- st25r3916ExecuteCommands() : **st25r3916_com.h**
- st25r3916GetInterrupt() : **st25r3916_irq.h**
- st25r3916GetNoResponseTime() : **st25r3916.h**
- st25r3916GetNumFIFOBytes() : **st25r3916.h**
- st25r3916GetNumFIFOLastBits() : **st25r3916.h**
- st25r3916GetRegsDump() : **st25r3916.h**
- st25r3916GetRSSI() : **st25r3916.h**
- st25r3916Initialize() : **st25r3916.h**
- st25r3916InitInterrupts() : **st25r3916_irq.h**
- st25r3916IRQCallbackRestore() : **st25r3916_irq.h**
- st25r3916IRQCallbackSet() : **st25r3916_irq.h**
- st25r3916IsCmdValid() : **st25r3916.h**
- st25r3916Isr() : **st25r3916_irq.h**
- st25r3916IsRegValid() : **st25r3916_com.h**
- st25r3916ledEvtCmd() : **st25r3916_led.h**
- st25r3916ledEvtCmds() : **st25r3916_led.h**
- st25r3916ledEvtIrq() : **st25r3916_led.h**
- st25r3916ledEvtWrMultiReg() : **st25r3916_led.h**
- st25r3916ledEvtWrReg() : **st25r3916_led.h**
- st25r3916ledInit() : **st25r3916_led.h**
- st25r3916MeasureAmplitude() : **st25r3916.h**
- st25r3916MeasureCapacitance() : **st25r3916.h**
- st25r3916MeasurePhase() : **st25r3916.h**
- st25r3916MeasurePowerSupply() : **st25r3916.h**
- st25r3916MeasureVoltage() : **st25r3916.h**
- st25r3916ModifyInterrupts() : **st25r3916_irq.h**
- st25r3916ModifyRegister() : **st25r3916_com.h**
- st25r3916OscOn() : **st25r3916.h**
- st25r3916PerformCollisionAvoidance() : **st25r3916.h**
- st25r3916ReadFifo() : **st25r3916_com.h**
- st25r3916ReadMultipleRegisters() : **st25r3916_com.h**
- st25r3916ReadPTMem() : **st25r3916_com.h**
- st25r3916ReadRegister() : **st25r3916_com.h**
- st25r3916ReadTestRegister() : **st25r3916_com.h**

- st25r3916SetAntennaMode() : **st25r3916.h**
- st25r3916SetBitrate() : **st25r3916.h**
- st25r3916SetGPTime() : **st25r3916.h**
- st25r3916SetNoResponseTime() : **st25r3916.h**
- st25r3916SetNumTxBits() : **st25r3916.h**
- st25r3916SetRegisterBits() : **st25r3916_com.h**
- st25r3916SetRegulators() : **st25r3916.h**
- st25r3916SetStartGPTimer() : **st25r3916.h**
- st25r3916SetStartNoResponseTimer() : **st25r3916.h**
- st25r3916StreamConfigure() : **st25r3916.h**
- st25r3916WaitForInterruptsTimed() : **st25r3916_irq.h**
- st25r3916WriteFifo() : **st25r3916_com.h**
- st25r3916WriteMultipleRegisters() : **st25r3916_com.h**
- st25r3916WritePTMem() : **st25r3916_com.h**
- st25r3916WritePTMemF() : **st25r3916_com.h**
- st25r3916WritePTMemTSN() : **st25r3916_com.h**
- st25r3916WriteRegister() : **st25r3916_com.h**
- st25r3916WriteTestRegister() : **st25r3916_com.h**
- st25r500AdjustRegulators() : **st25r500.h**
- st25r500CalibrateWU() : **st25r500.h**
- st25r500ChangeRegisterBits() : **st25r500_com.h**
- st25r500ChangeTestRegisterBits() : **st25r500_com.h**
- st25r500CheckChipID() : **st25r500.h**
- st25r500CheckForReceivedInterrupts() : **st25r500_irq.h**
- st25r500CheckReg() : **st25r500_com.h**
- st25r500ClearAndEnableInterrupts() : **st25r500_irq.h**
- st25r500ClearCalibration() : **st25r500.h**
- st25r500ClearInterrupts() : **st25r500_irq.h**
- st25r500ClrRegisterBits() : **st25r500_com.h**
- st25r500Deinitialize() : **st25r500.h**
- st25r500DiagMeasure() : **st25r500.h**
- st25r500DisableInterrupts() : **st25r500_irq.h**
- st25r500DpocrAdjust() : **st25r500_dpocr.h**
- st25r500DpocrConfigRead() : **st25r500_dpocr.h**
- st25r500DpocrConfigWrite() : **st25r500_dpocr.h**
- st25r500DpocrGetInfo() : **st25r500_dpocr.h**
- st25r500DpocrInitialize() : **st25r500_dpocr.h**
- st25r500DpocrReqAdj() : **st25r500_dpocr.h**
- st25r500DpocrSetEnabled() : **st25r500_dpocr.h**

- st25r500EnableInterrupts() : **st25r500_irq.h**
- st25r500ExecuteCommand() : **st25r500_com.h**
- st25r500ExecuteCommandAndGetResult() : **st25r500.h**
- st25r500ExecuteCommands() : **st25r500_com.h**
- st25r500GetCeGain() : **st25r500.h**
- st25r500GetInterrupt() : **st25r500_irq.h**
- st25r500GetNoResponseTime() : **st25r500.h**
- st25r500GetNumFIFOBytes() : **st25r500.h**
- st25r500GetNumFIFOLastBits() : **st25r500.h**
- st25r500GetRegsDump() : **st25r500.h**
- st25r500GetRSSI() : **st25r500.h**
- st25r500Initialize() : **st25r500.h**
- st25r500InitInterrupts() : **st25r500_irq.h**
- st25r500IRQCallbackRestore() : **st25r500_irq.h**
- st25r500IRQCallbackSet() : **st25r500_irq.h**
- st25r500IsCmdValid() : **st25r500.h**
- st25r500Isr() : **st25r500_irq.h**
- st25r500IsRegValid() : **st25r500_com.h**
- st25r500ledEvtCmd() : **st25r500_com.h**
- st25r500ledEvtCmds() : **st25r500_com.h**
- st25r500ledEvtIrq() : **st25r500_com.h**
- st25r500ledEvtWrMultiReg() : **st25r500_com.h**
- st25r500ledEvtWrReg() : **st25r500_com.h**
- st25r500ledInit() : **st25r500_com.h**
- st25r500MeasureAmplitude() : **st25r500.h**
- st25r500MeasureCombinedIQ() : **st25r500.h**
- st25r500MeasureCurrent() : **st25r500.h**
- st25r500MeasureI() : **st25r500.h**
- st25r500MeasureIQ() : **st25r500.h**
- st25r500MeasurePhase() : **st25r500.h**
- st25r500MeasureQ() : **st25r500.h**
- st25r500MeasureWU() : **st25r500.h**
- st25r500ModifyInterrupts() : **st25r500_irq.h**
- st25r500ModifyRegister() : **st25r500_com.h**
- st25r500OscOn() : **st25r500.h**
- st25r500PerformCollisionAvoidance() : **st25r500.h**
- st25r500ReadFifo() : **st25r500_com.h**
- st25r500ReadMultipleRegisters() : **st25r500_com.h**
- st25r500ReadRegister() : **st25r500_com.h**

- st25r500ReadTestRegister() : **st25r500_com.h**
- st25r500SetAntennaMode() : **st25r500.h**
- st25r500SetBitrate() : **st25r500.h**
- st25r500SetGPTime() : **st25r500.h**
- st25r500SetMRT() : **st25r500.h**
- st25r500SetNoResponseTime() : **st25r500.h**
- st25r500SetNumTxBits() : **st25r500.h**
- st25r500SetRegisterBits() : **st25r500_com.h**
- st25r500SetRegulators() : **st25r500.h**
- st25r500SetStartGPTimer() : **st25r500.h**
- st25r500SetStartNoResponseTimer() : **st25r500.h**
- st25r500WaitForInterruptsTimed() : **st25r500_irq.h**
- st25r500WriteFifo() : **st25r500_com.h**
- st25r500WriteMultipleRegisters() : **st25r500_com.h**
- st25r500WriteMultipleTestRegister() : **st25r500_com.h**
- st25r500WriteRegister() : **st25r500_com.h**
- st25r500WriteTestRegister() : **st25r500_com.h**
- st25r95CalibrateTagDetector() : **st25r95_com.h**
- st25r95CheckChipID() : **st25r95.h**
- st25r95Deinitialize() : **st25r95.h**
- st25r95FieldOff() : **st25r95.h**
- st25r95FieldOn() : **st25r95.h**
- st25r95Initialize() : **st25r95.h**
- st25r95Listen() : **st25r95_com.h**
- st25r95PollField() : **st25r95_com.h**
- st25r95ProtocolSelect() : **st25r95_com.h**
- st25r95ReadReg() : **st25r95_com.h**
- st25r95RearmListen() : **st25r95_com.h**
- st25r95SetACFilter() : **st25r95_com.h**
- st25r95SetBitRate() : **st25r95_com.h**
- st25r95SetFWT() : **st25r95_com.h**
- st25r95SetSlotCounter() : **st25r95_com.h**
- st25r95SPI_nIRQ_IN_Pulse() : **st25r95.h**
- st25r95SPICommandEcho() : **st25r95_com.h**
- st25r95SPICompleteRx() : **st25r95_com.h**
- st25r95SPIDeactivateACFilter() : **st25r95_com.h**
- st25r95SPIGetIdleResponse() : **st25r95_com.h**
- st25r95SPIGetLmState() : **st25r95_com.h**
- st25r95SPIIdle() : **st25r95_com.h**

- st25r95SPIIsInListen() : **st25r95_com.h**
- st25r95SPIIsTransmitCompleted() : **st25r95_com.h**
- st25r95SPIKillIdle() : **st25r95_com.h**
- st25r95SPIPollRead() : **st25r95_com.h**
- st25r95SPIPollSend() : **st25r95_com.h**
- st25r95SPIPrepareRx() : **st25r95_com.h**
- st25r95SPIResetChip() : **st25r95_com.h**
- st25r95SPIRxTx() : **st25r95_com.h**
- st25r95SPISendCommandTypeAndLen() : **st25r95_com.h**
- st25r95SPISendData() : **st25r95_com.h**
- st25r95SPISendReceiveByte() : **st25r95_com.h**
- st25r95SPISendTransmitFlag() : **st25r95_com.h**
- st25r95SPISetACState() : **st25r95_com.h**
- st25r95UART_nIRQ_IN_Pulse() : **st25r95.h**
- st25r95UARTCommandEcho() : **st25r95_com.h**
- st25r95UARTCompleteRx() : **st25r95_com.h**
- st25r95UartErrorCallback() : **st25r95_com.h**
- st25r95UARTGetIdleResponse() : **st25r95_com.h**
- st25r95UARTIdle() : **st25r95_com.h**
- st25r95UARTIsTransmitTxCompleted() : **st25r95_com.h**
- st25r95UARTKillIdle() : **st25r95_com.h**
- st25r95UARTPollRead() : **st25r95_com.h**
- st25r95UARTPrepareRx() : **st25r95_com.h**
- st25r95UartRxCpltCallback() : **st25r95_com.h**
- st25r95UARTSendCommandTypeAndLen() : **st25r95_com.h**
- st25r95UARTSendData() : **st25r95_com.h**
- st25r95UARTSendTransmitFlag() : **st25r95_com.h**
- st25r95UartTxCpltCallback() : **st25r95_com.h**
- st25r95WriteReg() : **st25r95_com.h**

Data Structures

Examples

File List

All

Functions

Typedefs

Enumerations

Enumerator

Macros

- globalCommProtectCnt : **rfal_platform.h**
- rfalDlmaDefaultSettings : **st25r3916/rfal_dlmaTbl.h**, **st25r500/rfal_dlmaTbl.h**
- st25r95SPIRxCtx : **st25r95_com.h**

RFAL

Main Page

Modules

Data Structures

Examples

File List

All

Functions

Variables

Enumerations

Enumerator

Macros

- ReturnCode : **rfal_utils.h**
- rfalAnalogConfigBitrate : **rfal_analogConfig.h**
- rfalAnalogConfigDirection : **rfal_analogConfig.h**
- rfalAnalogConfigId : **rfal_analogConfig.h**
- rfalAnalogConfigMode : **rfal_analogConfig.h**
- rfalAnalogConfigNum : **rfal_analogConfig.h**
- rfalAnalogConfigOffset : **rfal_analogConfig.h**
- rfalAnalogConfigRegAddr : **rfal_analogConfig.h**
- rfalAnalogConfigRegMask : **rfal_analogConfig.h**
- rfalAnalogConfigRegVal : **rfal_analogConfig.h**
- rfalAnalogConfigTech : **rfal_analogConfig.h**
- rfalDlmaAdjustFunc : **rfal_dlma.h**
- rfalDlmaMeasureFunc : **rfal_dlma.h**
- rfalDpoAdjustFunc : **rfal_dpo.h**
- rfalDpoMeasureFunc : **rfal_dpo.h**
- rfalFeliCaPollRes : **rfal_rf.h**
- rfalLmEonCallback : **rfal_rf.h**
- rfalNfcDepDeactCallback : **rfal_nfcDep.h**
- rfalNfcfServ : **rfal_nfcf.h**
- rfalNfcPropCallback : **rfal_nfc.h**

- rfalPostTxRxCallback : **rfal_rf.h**
- rfalPreTxRxCallback : **rfal_rf.h**
- rfalSt25tbBlock : **rfal_st25tb.h**
- rfalSt25tbUID : **rfal_st25tb.h**
- rfalSyncTxRxCallback : **rfal_rf.h**
- rfalUpperLayerCallback : **rfal_rf.h**
- rfalWakeUpConfig : **st25r200/rfal_features.h**

Generated on Fri Mar 14 2025 11:45:12 for RFAL by doxygen 1.9.5

RFAL

Main Page

Modules

Data Structures

Examples

File List

All

Functions

Variables

Typedefs

Enumerator

Macros

- r -

- rfal14443AShortFrameCmd : **rfal_rf.h**
- rfalBitRate : **rfal_rf.h**
- rfalCdDetType : **rfal_cd.h**
- rfalCdTech : **rfal_cd.h**
- rfalComplianceMode : **rfal_rf.h**
- rfalEHandling : **rfal_rf.h**
- rfalFeliCaPollSlots : **rfal_rf.h**
- rfalIso15693VcdCoding_t : **rfal_iso15693_2.h**
- rfalIsoDepFSx : **rfal_isoDep.h**
- rfalIsoDepFSxI : **rfal_isoDep.h**
- rfalLmNfcidLen : **rfal_rf.h**
- rfalLmState : **rfal_rf.h**
- rfalLpMode : **rfal_rf.h**
- rfalMode : **rfal_rf.h**
- rfalNfcaListenDeviceType : **rfal_nfca.h**
- rfalNfcbSensCmd : **rfal_nfcb.h**
- rfalNfcbSlots : **rfal_nfcb.h**
- rfalNfcDeactivateType : **rfal_nfc.h**
- rfalNfcDepCommMode : **rfal_nfcDep.h**
- rfalNfcDepRole : **rfal_nfcDep.h**
- rfalNfcDevType : **rfal_nfc.h**
- rfalNfcRfInterface : **rfal_nfc.h**
- rfalNfcState : **rfal_nfc.h**
- rfalNfcvNumSlots : **rfal_nfcv.h**
- rfalState : **rfal_rf.h**
- rfalT1Tcmds : **rfal_t1t.h**
- rfalT4tCmds : **rfal_t4t.h**
- rfalTransceiveState : **rfal_rf.h**
- rfalWumAAWeight : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**
- rfalWumMeasDuration : **st25r200/rfal_features.h**, **st25r500/rfal_features.h**
- rfalWumMeasFilter : **st25r200/rfal_features.h**, **st25r500/rfal_features.h**
- rfalWumPeriod : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- rfalWumState : **rfal_rf.h**

RFAL

Main Page

Modules

Data Structures

Examples

File List

All

Functions

Variables

Typedefs

Enumerations

Macros

- r -

- RFAL_14443A_SHORTFRAME_CMD_REQA : **rfal_rf.h**
- RFAL_14443A_SHORTFRAME_CMD_WUPA : **rfal_rf.h**
- RFAL_BR_105p94 : **rfal_rf.h**
- RFAL_BR_106 : **rfal_rf.h**

- RFAL_BR_13560 : **rfal_rf.h**
- RFAL_BR_1695 : **rfal_rf.h**
- RFAL_BR_1p66 : **rfal_rf.h**
- RFAL_BR_211p88 : **rfal_rf.h**
- RFAL_BR_212 : **rfal_rf.h**
- RFAL_BR_26p48 : **rfal_rf.h**
- RFAL_BR_3390 : **rfal_rf.h**
- RFAL_BR_424 : **rfal_rf.h**
- RFAL_BR_52p97 : **rfal_rf.h**
- RFAL_BR_6780 : **rfal_rf.h**
- RFAL_BR_848 : **rfal_rf.h**
- RFAL_BR_KEEP : **rfal_rf.h**
- RFAL_CD_TECH_NFCA : **rfal_cd.h**
- RFAL_CD_TECH_NFCB : **rfal_cd.h**
- RFAL_CD_TECH_NFCF : **rfal_cd.h**
- RFAL_CD_TECH_NFCV : **rfal_cd.h**
- RFAL_CD_TECH_NONE : **rfal_cd.h**
- RFAL_CD_TECH_OTHER : **rfal_cd.h**
- RFAL_COMPLIANCE_MODE_EMV : **rfal_rf.h**
- RFAL_COMPLIANCE_MODE_ISO : **rfal_rf.h**
- RFAL_COMPLIANCE_MODE_NFC : **rfal_rf.h**
- RFAL_ERRORHANDLING_EMD : **rfal_rf.h**
- RFAL_ERRORHANDLING_NONE : **rfal_rf.h**
- RFAL_FELICA_16_SLOTS : **rfal_rf.h**
- RFAL_FELICA_1_SLOT : **rfal_rf.h**
- RFAL_FELICA_2_SLOTS : **rfal_rf.h**
- RFAL_FELICA_4_SLOTS : **rfal_rf.h**
- RFAL_FELICA_8_SLOTS : **rfal_rf.h**
- RFAL_FELICA_POLL_RC_COM_PERFORMANCE : **rfal_rf.h**
- RFAL_FELICA_POLL_RC_NO_REQUEST : **rfal_rf.h**
- RFAL_FELICA_POLL_RC_SYSTEM_CODE : **rfal_rf.h**
- RFAL_ISODEP_FSX_1024 : **rfal_isoDep.h**
- RFAL_ISODEP_FSX_128 : **rfal_isoDep.h**
- RFAL_ISODEP_FSX_16 : **rfal_isoDep.h**
- RFAL_ISODEP_FSX_2048 : **rfal_isoDep.h**
- RFAL_ISODEP_FSX_24 : **rfal_isoDep.h**
- RFAL_ISODEP_FSX_256 : **rfal_isoDep.h**
- RFAL_ISODEP_FSX_32 : **rfal_isoDep.h**
- RFAL_ISODEP_FSX_40 : **rfal_isoDep.h**

- RFAL_ISODEP_FSX_4096 : **rfal_isoDep.h**
- RFAL_ISODEP_FSX_48 : **rfal_isoDep.h**
- RFAL_ISODEP_FSX_512 : **rfal_isoDep.h**
- RFAL_ISODEP_FSX_64 : **rfal_isoDep.h**
- RFAL_ISODEP_FSX_96 : **rfal_isoDep.h**
- RFAL_ISODEP_FSXI_1024 : **rfal_isoDep.h**
- RFAL_ISODEP_FSXI_128 : **rfal_isoDep.h**
- RFAL_ISODEP_FSXI_16 : **rfal_isoDep.h**
- RFAL_ISODEP_FSXI_2048 : **rfal_isoDep.h**
- RFAL_ISODEP_FSXI_24 : **rfal_isoDep.h**
- RFAL_ISODEP_FSXI_256 : **rfal_isoDep.h**
- RFAL_ISODEP_FSXI_32 : **rfal_isoDep.h**
- RFAL_ISODEP_FSXI_40 : **rfal_isoDep.h**
- RFAL_ISODEP_FSXI_4096 : **rfal_isoDep.h**
- RFAL_ISODEP_FSXI_48 : **rfal_isoDep.h**
- RFAL_ISODEP_FSXI_512 : **rfal_isoDep.h**
- RFAL_ISODEP_FSXI_64 : **rfal_isoDep.h**
- RFAL_ISODEP_FSXI_96 : **rfal_isoDep.h**
- RFAL_LM_NFCID_LEN_04 : **rfal_rf.h**
- RFAL_LM_NFCID_LEN_07 : **rfal_rf.h**
- RFAL_LM_NFCID_LEN_10 : **rfal_rf.h**
- RFAL_LM_STATE_ACTIVE_A : **rfal_rf.h**
- RFAL_LM_STATE_ACTIVE_Ax : **rfal_rf.h**
- RFAL_LM_STATE_CARDEMU_3 : **rfal_rf.h**
- RFAL_LM_STATE_CARDEMU_4A : **rfal_rf.h**
- RFAL_LM_STATE_CARDEMU_4B : **rfal_rf.h**
- RFAL_LM_STATE_IDLE : **rfal_rf.h**
- RFAL_LM_STATE_NOT_INIT : **rfal_rf.h**
- RFAL_LM_STATE_POWER_OFF : **rfal_rf.h**
- RFAL_LM_STATE_READY_A : **rfal_rf.h**
- RFAL_LM_STATE_READY_Ax : **rfal_rf.h**
- RFAL_LM_STATE_READY_B : **rfal_rf.h**
- RFAL_LM_STATE_READY_F : **rfal_rf.h**
- RFAL_LM_STATE_SLEEP_A : **rfal_rf.h**
- RFAL_LM_STATE_SLEEP_AF : **rfal_rf.h**
- RFAL_LM_STATE_SLEEP_B : **rfal_rf.h**
- RFAL_LM_STATE_TARGET_A : **rfal_rf.h**
- RFAL_LM_STATE_TARGET_F : **rfal_rf.h**
- RFAL_LP_MODE_HR : **rfal_rf.h**

- RFAL_LP_MODE_PD : **rfal_rf.h**
- RFAL_MODE_LISTEN_ACTIVE_P2P : **rfal_rf.h**
- RFAL_MODE_LISTEN_NFCA : **rfal_rf.h**
- RFAL_MODE_LISTEN_NFCB : **rfal_rf.h**
- RFAL_MODE_LISTEN_NFCF : **rfal_rf.h**
- RFAL_MODE_NONE : **rfal_rf.h**
- RFAL_MODE_POLL_ACTIVE_P2P : **rfal_rf.h**
- RFAL_MODE_POLL_B_CTS : **rfal_rf.h**
- RFAL_MODE_POLL_B_PRIME : **rfal_rf.h**
- RFAL_MODE_POLL_NFCA : **rfal_rf.h**
- RFAL_MODE_POLL_NFCA_T1T : **rfal_rf.h**
- RFAL_MODE_POLL_NFCB : **rfal_rf.h**
- RFAL_MODE_POLL_NFCF : **rfal_rf.h**
- RFAL_MODE_POLL_NFCV : **rfal_rf.h**
- RFAL_MODE_POLL_PICOPASS : **rfal_rf.h**
- RFAL_NFC_DEACTIVATE_DISCOVERY : **rfal_nfc.h**
- RFAL_NFC_DEACTIVATE_IDLE : **rfal_nfc.h**
- RFAL_NFC_DEACTIVATE_SLEEP : **rfal_nfc.h**
- RFAL_NFC_INTERFACE_ISODEP : **rfal_nfc.h**
- RFAL_NFC_INTERFACE_NFCDEP : **rfal_nfc.h**
- RFAL_NFC_INTERFACE_RF : **rfal_nfc.h**
- RFAL_NFC_LISTEN_TYPE_AP2P : **rfal_nfc.h**
- RFAL_NFC_LISTEN_TYPE_NFCA : **rfal_nfc.h**
- RFAL_NFC_LISTEN_TYPE_NFCB : **rfal_nfc.h**
- RFAL_NFC_LISTEN_TYPE_NFCF : **rfal_nfc.h**
- RFAL_NFC_LISTEN_TYPE_NFCV : **rfal_nfc.h**
- RFAL_NFC_LISTEN_TYPE_PROP : **rfal_nfc.h**
- RFAL_NFC_LISTEN_TYPE_ST25TB : **rfal_nfc.h**
- RFAL_NFC_POLL_TYPE_AP2P : **rfal_nfc.h**
- RFAL_NFC_POLL_TYPE_NFCA : **rfal_nfc.h**
- RFAL_NFC_POLL_TYPE_NFCB : **rfal_nfc.h**
- RFAL_NFC_POLL_TYPE_NFCF : **rfal_nfc.h**
- RFAL_NFC_POLL_TYPE_NFCV : **rfal_nfc.h**
- RFAL_NFC_STATE_ACTIVATED : **rfal_nfc.h**
- RFAL_NFC_STATE_DATAEXCHANGE : **rfal_nfc.h**
- RFAL_NFC_STATE_DATAEXCHANGE_DONE : **rfal_nfc.h**
- RFAL_NFC_STATE_DEACTIVATION : **rfal_nfc.h**
- RFAL_NFC_STATE_IDLE : **rfal_nfc.h**
- RFAL_NFC_STATE_LISTEN_ACTIVATION : **rfal_nfc.h**

- RFAL_NFC_STATE_LISTEN_COLAVOIDANCE : **rfal_nfc.h**
- RFAL_NFC_STATE_LISTEN_SLEEP : **rfal_nfc.h**
- RFAL_NFC_STATE_LISTEN_TECHDETECT : **rfal_nfc.h**
- RFAL_NFC_STATE_NOTINIT : **rfal_nfc.h**
- RFAL_NFC_STATE_POLL_ACTIVATION : **rfal_nfc.h**
- RFAL_NFC_STATE_POLL_COLAVOIDANCE : **rfal_nfc.h**
- RFAL_NFC_STATE_POLL_SELECT : **rfal_nfc.h**
- RFAL_NFC_STATE_POLL_TECHDETECT : **rfal_nfc.h**
- RFAL_NFC_STATE_START_DISCOVERY : **rfal_nfc.h**
- RFAL_NFC_STATE_WAKEUP_MODE : **rfal_nfc.h**
- RFAL_NFCB_SENS_CMD_ALLB_REQ : **rfal_nfcb.h**
- RFAL_NFCB_SENS_CMD_SENSB_REQ : **rfal_nfcb.h**
- RFAL_NFCB_SLOT_NUM_1 : **rfal_nfcb.h**
- RFAL_NFCB_SLOT_NUM_16 : **rfal_nfcb.h**
- RFAL_NFCB_SLOT_NUM_2 : **rfal_nfcb.h**
- RFAL_NFCB_SLOT_NUM_4 : **rfal_nfcb.h**
- RFAL_NFCB_SLOT_NUM_8 : **rfal_nfcb.h**
- RFAL_NFCDEP_Bx_08_848 : **rfal_nfcDep.h**
- RFAL_NFCDEP_Bx_16_1695 : **rfal_nfcDep.h**
- RFAL_NFCDEP_Bx_32_3390 : **rfal_nfcDep.h**
- RFAL_NFCDEP_Bx_64_6780 : **rfal_nfcDep.h**
- RFAL_NFCDEP_Bx_NO_HIGH_BR : **rfal_nfcDep.h**
- RFAL_NFCDEP_COMM_ACTIVE : **rfal_nfcDep.h**
- RFAL_NFCDEP_COMM_PASSIVE : **rfal_nfcDep.h**
- RFAL_NFCDEP_Dx_01_106 : **rfal_nfcDep.h**
- RFAL_NFCDEP_Dx_02_212 : **rfal_nfcDep.h**
- RFAL_NFCDEP_Dx_04_424 : **rfal_nfcDep.h**
- RFAL_NFCDEP_Dx_08_848 : **rfal_nfcDep.h**
- RFAL_NFCDEP_Dx_16_1695 : **rfal_nfcDep.h**
- RFAL_NFCDEP_Dx_32_3390 : **rfal_nfcDep.h**
- RFAL_NFCDEP_Dx_64_6780 : **rfal_nfcDep.h**
- RFAL_NFCDEP_LR_128 : **rfal_nfcDep.h**
- RFAL_NFCDEP_LR_192 : **rfal_nfcDep.h**
- RFAL_NFCDEP_LR_254 : **rfal_nfcDep.h**
- RFAL_NFCDEP_LR_64 : **rfal_nfcDep.h**
- RFAL_NFCDEP_ROLE_INITIATOR : **rfal_nfcDep.h**
- RFAL_NFCDEP_ROLE_TARGET : **rfal_nfcDep.h**
- RFAL_NFCF_CMD_AUTHENTICATION1 : **rfal_nfcf.h**
- RFAL_NFCF_CMD_AUTHENTICATION2 : **rfal_nfcf.h**

- RFAL_NFCF_CMD_POLLING : **rfal_nfcf.h**
- RFAL_NFCF_CMD_POLLING_RES : **rfal_nfcf.h**
- RFAL_NFCF_CMD_READ : **rfal_nfcf.h**
- RFAL_NFCF_CMD_READ_WITHOUT_ENCRYPTION : **rfal_nfcf.h**
- RFAL_NFCF_CMD_READ_WITHOUT_ENCRYPTION_RES : **rfal_nfcf.h**
- RFAL_NFCF_CMD_REQUEST_RESPONSE : **rfal_nfcf.h**
- RFAL_NFCF_CMD_REQUEST_SERVICE : **rfal_nfcf.h**
- RFAL_NFCF_CMD_REQUEST_SYSTEM_CODE : **rfal_nfcf.h**
- RFAL_NFCF_CMD_WRITE : **rfal_nfcf.h**
- RFAL_NFCF_CMD_WRITE_WITHOUT_ENCRYPTION : **rfal_nfcf.h**
- RFAL_NFCF_CMD_WRITE_WITHOUT_ENCRYPTION_RES : **rfal_nfcf.h**
- RFAL_NFCV_CMD_ENABLE_EAS : **rfal_nfcv.h**
- RFAL_NFCV_CMD_EXTENDED_GET_SYS_INFO : **rfal_nfcv.h**
- RFAL_NFCV_CMD_EXTENDED_LOCK_SINGLE_BLOCK : **rfal_nfcv.h**
- RFAL_NFCV_CMD_EXTENDED_READ_MULTIPLE_BLOCK : **rfal_nfcv.h**
- RFAL_NFCV_CMD_EXTENDED_READ_SINGLE_BLOCK : **rfal_nfcv.h**
- RFAL_NFCV_CMD_EXTENDED_WRITE_MULTIPLE_BLOCK : **rfal_nfcv.h**
- RFAL_NFCV_CMD_EXTENDED_WRITE_SINGLE_BLOCK : **rfal_nfcv.h**
- RFAL_NFCV_CMD_FAST_EXTENDED_READ_MULTIPLE_BLOCKS : **rfal_nfcv.h**
- RFAL_NFCV_CMD_FAST_EXTENDED_READ_SINGLE_BLOCK : **rfal_nfcv.h**
- RFAL_NFCV_CMD_FAST_READ_DYN_CONFIGURATION : **rfal_nfcv.h**
- RFAL_NFCV_CMD_FAST_READ_MESSAGE : **rfal_nfcv.h**
- RFAL_NFCV_CMD_FAST_READ_MESSAGE_LENGTH : **rfal_nfcv.h**
- RFAL_NFCV_CMD_FAST_READ_MULTIPLE_BLOCKS : **rfal_nfcv.h**
- RFAL_NFCV_CMD_FAST_READ_SINGLE_BLOCK : **rfal_nfcv.h**
- RFAL_NFCV_CMD_FAST_WRITE_DYN_CONFIGURATION : **rfal_nfcv.h**
- RFAL_NFCV_CMD_FAST_WRITE_MESSAGE : **rfal_nfcv.h**
- RFAL_NFCV_CMD_GET_RANDOM_NUMBER : **rfal_nfcv.h**
- RFAL_NFCV_CMD_GET_SYS_INFO : **rfal_nfcv.h**
- RFAL_NFCV_CMD_INVENTORY : **rfal_nfcv.h**
- RFAL_NFCV_CMD_KILL : **rfal_nfcv.h**
- RFAL_NFCV_CMD_LOCK_BLOCK : **rfal_nfcv.h**
- RFAL_NFCV_CMD_LOCK_EAS : **rfal_nfcv.h**
- RFAL_NFCV_CMD_LOCK_KILL : **rfal_nfcv.h**
- RFAL_NFCV_CMD_MANAGE_GPO : **rfal_nfcv.h**
- RFAL_NFCV_CMD_PRESENT_PASSWORD : **rfal_nfcv.h**
- RFAL_NFCV_CMD_READ_CONFIGURATION : **rfal_nfcv.h**
- RFAL_NFCV_CMD_READ_DYN_CONFIGURATION : **rfal_nfcv.h**
- RFAL_NFCV_CMD_READ_MESSAGE : **rfal_nfcv.h**

- RFAL_NFCV_CMD_READ_MESSAGE_LENGTH : **rfal_nfcv.h**
- RFAL_NFCV_CMD_READ_MULTIPLE_BLOCKS : **rfal_nfcv.h**
- RFAL_NFCV_CMD_READ_SINGLE_BLOCK : **rfal_nfcv.h**
- RFAL_NFCV_CMD_RESET_EAS : **rfal_nfcv.h**
- RFAL_NFCV_CMD_RESET_TO_READY : **rfal_nfcv.h**
- RFAL_NFCV_CMD_SELECT : **rfal_nfcv.h**
- RFAL_NFCV_CMD_SET_EAS : **rfal_nfcv.h**
- RFAL_NFCV_CMD_SLPV : **rfal_nfcv.h**
- RFAL_NFCV_CMD_WRITE_CONFIGURATION : **rfal_nfcv.h**
- RFAL_NFCV_CMD_WRITE_DYN_CONFIGURATION : **rfal_nfcv.h**
- RFAL_NFCV_CMD_WRITE_EAS_CONFIG : **rfal_nfcv.h**
- RFAL_NFCV_CMD_WRITE_EAS_ID : **rfal_nfcv.h**
- RFAL_NFCV_CMD_WRITE_MESSAGE : **rfal_nfcv.h**
- RFAL_NFCV_CMD_WRITE_MULTIPLE_BLOCKS : **rfal_nfcv.h**
- RFAL_NFCV_CMD_WRITE_PASSWORD : **rfal_nfcv.h**
- RFAL_NFCV_CMD_WRITE_SINGLE_BLOCK : **rfal_nfcv.h**
- RFAL_NFCV_ERROR_BLOCK_ALREDY_LOCKED : **rfal_nfcv.h**
- RFAL_NFCV_ERROR_BLOCK_FAILED : **rfal_nfcv.h**
- RFAL_NFCV_ERROR_BLOCK_LOCKED : **rfal_nfcv.h**
- RFAL_NFCV_ERROR_BLOCK_NOT_AVALIABLE : **rfal_nfcv.h**
- RFAL_NFCV_ERROR_CMD_NOT_RECOGNIZED : **rfal_nfcv.h**
- RFAL_NFCV_ERROR_CMD_NOT_SUPPORTED : **rfal_nfcv.h**
- RFAL_NFCV_ERROR_OPTION_NOT_SUPPORTED : **rfal_nfcv.h**
- RFAL_NFCV_ERROR_UNKNOWN : **rfal_nfcv.h**
- RFAL_NFCV_ERROR_WRITE_FAILED : **rfal_nfcv.h**
- RFAL_NFCV_NUM_SLOTS_1 : **rfal_nfcv.h**
- RFAL_NFCV_NUM_SLOTS_16 : **rfal_nfcv.h**
- RFAL_NFCV_REQ_FLAG_ADDRESS : **rfal_nfcv.h**
- RFAL_NFCV_REQ_FLAG_AFI : **rfal_nfcv.h**
- RFAL_NFCV_REQ_FLAG_DATA_RATE : **rfal_nfcv.h**
- RFAL_NFCV_REQ_FLAG_DEFAULT : **rfal_nfcv.h**
- RFAL_NFCV_REQ_FLAG_INVENTORY : **rfal_nfcv.h**
- RFAL_NFCV_REQ_FLAG_NB_SLOTS : **rfal_nfcv.h**
- RFAL_NFCV_REQ_FLAG_OPTION : **rfal_nfcv.h**
- RFAL_NFCV_REQ_FLAG_PROTOCOL_EXT : **rfal_nfcv.h**
- RFAL_NFCV_REQ_FLAG_RFU : **rfal_nfcv.h**
- RFAL_NFCV_REQ_FLAG_SELECT : **rfal_nfcv.h**
- RFAL_NFCV_REQ_FLAG_SUB_CARRIER : **rfal_nfcv.h**
- RFAL_NFCV_RES_FLAG_ERROR : **rfal_nfcv.h**

- RFAL_NFCV_RES_FLAG_EXTENSION : **rfal_nfcv.h**
- RFAL_NFCV_RES_FLAG_RFU1 : **rfal_nfcv.h**
- RFAL_NFCV_RES_FLAG_RFU2 : **rfal_nfcv.h**
- RFAL_NFCV_RES_FLAG_RFU3 : **rfal_nfcv.h**
- RFAL_NFCV_RES_FLAG_RFU4 : **rfal_nfcv.h**
- RFAL_NFCV_RES_FLAG_RFU5 : **rfal_nfcv.h**
- RFAL_NFCV_RES_FLAG_RFU6 : **rfal_nfcv.h**
- RFAL_NFCV_SYSINFO_AFI : **rfal_nfcv.h**
- RFAL_NFCV_SYSINFO_CMDLIST : **rfal_nfcv.h**
- RFAL_NFCV_SYSINFO_CSI : **rfal_nfcv.h**
- RFAL_NFCV_SYSINFO_DFSID : **rfal_nfcv.h**
- RFAL_NFCV_SYSINFO_ICREF : **rfal_nfcv.h**
- RFAL_NFCV_SYSINFO_MEMSIZE : **rfal_nfcv.h**
- RFAL_NFCV_SYSINFO_MOI : **rfal_nfcv.h**
- RFAL_NFCV_SYSINFO_REQ_ALL : **rfal_nfcv.h**
- RFAL_T1T_CMD_RALL : **rfal_t1t.h**
- RFAL_T1T_CMD_READ : **rfal_t1t.h**
- RFAL_T1T_CMD_RID : **rfal_t1t.h**
- RFAL_T1T_CMD_WRITE_E : **rfal_t1t.h**
- RFAL_T1T_CMD_WRITE_NE : **rfal_t1t.h**
- RFAL_T4T_INS_READBINARY : **rfal_t4t.h**
- RFAL_T4T_INS_READBINARY_ODO : **rfal_t4t.h**
- RFAL_T4T_INS_SELECT : **rfal_t4t.h**
- RFAL_T4T_INS_UPDATEBINARY : **rfal_t4t.h**
- RFAL_T4T_INS_UPDATEBINARY_ODO : **rfal_t4t.h**
- RFAL_TXRX_FLAGS_AGC_OFF : **rfal_rf.h**
- RFAL_TXRX_FLAGS_AGC_ON : **rfal_rf.h**
- RFAL_TXRX_FLAGS_CRC_RX_AUTO : **rfal_rf.h**
- RFAL_TXRX_FLAGS_CRC_RX_KEEP : **rfal_rf.h**
- RFAL_TXRX_FLAGS_CRC_RX_MANUAL : **rfal_rf.h**
- RFAL_TXRX_FLAGS_CRC_RX_REMV : **rfal_rf.h**
- RFAL_TXRX_FLAGS_CRC_TX_AUTO : **rfal_rf.h**
- RFAL_TXRX_FLAGS_CRC_TX_MANUAL : **rfal_rf.h**
- RFAL_TXRX_FLAGS_NFCIP1_OFF : **rfal_rf.h**
- RFAL_TXRX_FLAGS_NFCIP1_ON : **rfal_rf.h**
- RFAL_TXRX_FLAGS_NFCV_FLAG_AUTO : **rfal_rf.h**
- RFAL_TXRX_FLAGS_NFCV_FLAG_MANUAL : **rfal_rf.h**
- RFAL_TXRX_FLAGS_PAR_RX_KEEP : **rfal_rf.h**
- RFAL_TXRX_FLAGS_PAR_RX_REMV : **rfal_rf.h**

- RFAL_TXRX_FLAGS_PAR_TX_AUTO : **rfal_rf.h**
- RFAL_TXRX_FLAGS_PAR_TX_NONE : **rfal_rf.h**
- RFAL_WUM_AA_WEIGHT_100 : **st25r500/rfal_features.h**
- RFAL_WUM_AA_WEIGHT_128 : **st25r500/rfal_features.h**
- RFAL_WUM_AA_WEIGHT_16 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**
- RFAL_WUM_AA_WEIGHT_32 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**
- RFAL_WUM_AA_WEIGHT_4 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**
- RFAL_WUM_AA_WEIGHT_50 : **st25r500/rfal_features.h**
- RFAL_WUM_AA_WEIGHT_64 : **st25r500/rfal_features.h**
- RFAL_WUM_AA_WEIGHT_8 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**
- RFAL_WUM_MEAS_DUR_26_10 : **st25r200/rfal_features.h**, **st25r500/rfal_features.h**
- RFAL_WUM_MEAS_DUR_30_14 : **st25r200/rfal_features.h**, **st25r500/rfal_features.h**
- RFAL_WUM_MEAS_DUR_34_19 : **st25r200/rfal_features.h**, **st25r500/rfal_features.h**
- RFAL_WUM_MEAS_DUR_44_28 : **st25r200/rfal_features.h**, **st25r500/rfal_features.h**
- RFAL_WUM_MEAS_FIL_FAST : **st25r200/rfal_features.h**, **st25r500/rfal_features.h**
- RFAL_WUM_MEAS_FIL_SLOW : **st25r200/rfal_features.h**, **st25r500/rfal_features.h**
- RFAL_WUM_PERIOD_1000MS : **st25r500/rfal_features.h**
- RFAL_WUM_PERIOD_100MS : **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**
- RFAL_WUM_PERIOD_105MS : **st25r200/rfal_features.h**
- RFAL_WUM_PERIOD_10MS : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**
- RFAL_WUM_PERIOD_1240MS : **st25r200/rfal_features.h**
- RFAL_WUM_PERIOD_125MS : **st25r500/rfal_features.h**
- RFAL_WUM_PERIOD_12MS : **st25r500/rfal_features.h**
- RFAL_WUM_PERIOD_150MS : **st25r500/rfal_features.h**
- RFAL_WUM_PERIOD_155MS : **st25r200/rfal_features.h**
- RFAL_WUM_PERIOD_15MS : **st25r200/rfal_features.h**
- RFAL_WUM_PERIOD_1700MS : **st25r200/rfal_features.h**
- RFAL_WUM_PERIOD_200MS : **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**
- RFAL_WUM_PERIOD_20MS : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**
- RFAL_WUM_PERIOD_215MS : **st25r200/rfal_features.h**
- RFAL_WUM_PERIOD_250MS : **st25r500/rfal_features.h**
- RFAL_WUM_PERIOD_25MS : **st25r200/rfal_features.h**, **st25r500/rfal_features.h**
- RFAL_WUM_PERIOD_300MS : **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_WUM_PERIOD_30MS : **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**

- RFAL_WUM_PERIOD_310MS : st25r200/rfal_features.h
- RFAL_WUM_PERIOD_350MS : st25r500/rfal_features.h
- RFAL_WUM_PERIOD_400MS : st25r3911/rfal_features.h, st25r3916/rfal_features.h, st25r500/rfal_features.h
- RFAL_WUM_PERIOD_40MS : st25r200/rfal_features.h, st25r3911/rfal_features.h, st25r3916/rfal_features.h
- RFAL_WUM_PERIOD_425MS : st25r200/rfal_features.h
- RFAL_WUM_PERIOD_500MS : st25r3911/rfal_features.h, st25r3916/rfal_features.h, st25r500/rfal_features.h
- RFAL_WUM_PERIOD_50MS : st25r3911/rfal_features.h, st25r3916/rfal_features.h, st25r500/rfal_features.h
- RFAL_WUM_PERIOD_55MS : st25r200/rfal_features.h
- RFAL_WUM_PERIOD_600MS : st25r3911/rfal_features.h, st25r3916/rfal_features.h
- RFAL_WUM_PERIOD_60MS : st25r3911/rfal_features.h, st25r3916/rfal_features.h
- RFAL_WUM_PERIOD_620MS : st25r200/rfal_features.h
- RFAL_WUM_PERIOD_700MS : st25r3911/rfal_features.h, st25r3916/rfal_features.h
- RFAL_WUM_PERIOD_70MS : st25r3911/rfal_features.h, st25r3916/rfal_features.h
- RFAL_WUM_PERIOD_75MS : st25r500/rfal_features.h
- RFAL_WUM_PERIOD_800MS : st25r3911/rfal_features.h, st25r3916/rfal_features.h, st25r500/rfal_features.h
- RFAL_WUM_PERIOD_80MS : st25r200/rfal_features.h, st25r3911/rfal_features.h, st25r3916/rfal_features.h
- RFAL_WUM_PERIOD_850MS : st25r200/rfal_features.h
- RFAL_WUM_PERIOD_INF : st25r500/rfal_features.h
- RFAL_WUM_STATE_ENABLED : rfal_rf.h
- RFAL_WUM_STATE_ENABLED_WOKE : rfal_rf.h
- RFAL_WUM_STATE_INITIALIZING : rfal_rf.h
- RFAL_WUM_STATE_NOT_INIT : rfal_rf.h
- RFAL_WUM_TRE_ABOVE : st25r200/rfal_features.h, st25r500/rfal_features.h
- RFAL_WUM_TRE_BELOW : st25r200/rfal_features.h, st25r500/rfal_features.h
- RFAL_WUM_TRE_BETWEEN : st25r200/rfal_features.h, st25r500/rfal_features.h

Examples

File List

All

Functions

Variables

Typedefs

Enumerations

Enumerator

m

n

p

r

s

- i -

- ISO15693_FWT : **rfal_iso15693_2.h**
- ISO15693_MASK_FDT_LISTEN : **rfal_iso15693_2.h**
- ISO15693_REQ_FLAG_HIGH_DATARATE : **rfal_iso15693_2.h**
- ISO15693_REQ_FLAG_TWO_SUBCARRIERS : **rfal_iso15693_2.h**

RFAL

Main Page

Modules

Data Structures

Examples

File List

All

Functions

Variables

Typedefs

Enumerations

Enumerator

i

n

p

r

s

- m -

- MS_FROM_64FCS : **st25r3911.h**
- MS_TO_64FCS : **st25r3911.h**

Generated on Fri Mar 14 2025 11:45:12 for RFAL by doxygen 1.9.5

RFAL

p

r

s

- n -

- NULL : **rfal_utils.h**

RFAL

Main Page

Modules

Data Structures

Examples

File List

All

Functions

Variables

Typedefs

Enumerations

Enumerator

i

m

n

r

s

- p -

- platformAssert : **rfal_defConfig.h**, **rfal_platform.h**
- platformDelay : **rfal_platform.h**
- platformErrorHandle : **rfal_defConfig.h**, **rfal_platform.h**
- platformGetSysTick : **rfal_defConfig.h**, **rfal_platform.h**
- platformGpioClear : **rfal_platform.h**
- platformGpioIsHigh : **rfal_platform.h**
- platformGpioIsLow : **rfal_platform.h**
- platformGpioSet : **rfal_platform.h**
- platformGpioToggle : **rfal_platform.h**
- platformI2CRepeatStart : **rfal_defConfig.h**, **rfal_platform.h**
- platformI2CRx : **rfal_defConfig.h**, **rfal_platform.h**
- platformI2CSlaveAddrRD : **rfal_defConfig.h**, **rfal_platform.h**
- platformI2CSlaveAddrWR : **rfal_defConfig.h**, **rfal_platform.h**
- platformI2CStart : **rfal_defConfig.h**, **rfal_platform.h**
- platformI2CStop : **rfal_defConfig.h**, **rfal_platform.h**
- platformI2CTx : **rfal_defConfig.h**, **rfal_platform.h**
- platformIrqST25RPinInitialize : **rfal_defConfig.h**
- platformIrqST25RSetCallback : **rfal_defConfig.h**
- platformLedOff : **rfal_defConfig.h**, **rfal_platform.h**
- platformLedOn : **rfal_defConfig.h**, **rfal_platform.h**
- platformLedsInitialize : **rfal_defConfig.h**
- platformLedToggle : **rfal_defConfig.h**, **rfal_platform.h**
- platformLog : **rfal_defConfig.h**, **rfal_platform.h**

- platformProtectST25RComm : **rfal_platform.h**
- platformProtectST25RIrqStatus : **rfal_defConfig.h**, **rfal_platform.h**
- platformProtectWorker : **rfal_defConfig.h**
- platformSpiDeselect : **rfal_platform.h**
- platformSpiSelect : **rfal_platform.h**
- platformSpiTxRx : **rfal_platform.h**
- platformTimerCreate : **rfal_platform.h**
- platformTimerDestroy : **rfal_defConfig.h**
- platformTimerGetRemaining : **rfal_defConfig.h**, **rfal_platform.h**
- platformTimerIsExpired : **rfal_platform.h**
- platformUnprotectST25RComm : **rfal_platform.h**
- platformUnprotectST25RIrqStatus : **rfal_defConfig.h**, **rfal_platform.h**
- platformUnprotectWorker : **rfal_defConfig.h**

Generated on Fri Mar 14 2025 11:45:12 for RFAL by doxygen 1.9.5

RFAL

Main Page

Modules

Data Structures

Examples

File List

All

Functions

Variables

Typedefs

Enumerations

Enumerator

i

m

n

p

s

- r -

- RFAL_1FC_IN_2048FC : **rfal_rf.h**
- RFAL_1FC_IN_4096FC : **rfal_rf.h**
- RFAL_1FC_IN_512FC : **rfal_rf.h**
- RFAL_1FC_IN_64FC : **rfal_rf.h**
- RFAL_1FC_IN_8FC : **rfal_rf.h**
- RFAL_1MS_IN_1FC : **rfal_rf.h**
- RFAL_ANALOG_CONFIG_ANTICOL : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_BITRATE_105p94 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_BITRATE_106 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_BITRATE_1695 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_BITRATE_1p6 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_BITRATE_211p88 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_BITRATE_212 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_BITRATE_26 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_BITRATE_3390 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_BITRATE_424 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_BITRATE_53 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_BITRATE_6780 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_BITRATE_848 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_BITRATE_COMMON : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_BITRATE_MASK : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_BITRATE_RFU : **rfal_analogConfig.h**

- RFAL_ANALOG_CONFIG_BITRATE_RFU2 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_BITRATE_SHIFT : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_DEINIT : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_FIELD_OFF : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_FIELD_ON : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_INIT : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_LISTEN_COMMON : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_LISTEN_OFF : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_LISTEN_ON : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_LOWPOWER_OFF : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_LOWPOWER_ON : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_POLL_COMMON : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_00 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_01 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_02 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_03 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_04 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_05 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_06 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_07 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_08 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_09 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_10 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_11 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_12 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_13 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_14 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_POWER_LVL_15 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_SPECIFIC_MASK : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_WAKEUP_OFF : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CHIP_WAKEUP_ON : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_CONFIG_NUM : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_DIRECTION_MASK : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_DIRECTION_RFU10 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_DIRECTION_RFU11 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_DIRECTION_RFU2 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_DIRECTION_RFU3 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_DIRECTION_RFU4 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_DIRECTION_RFU5 : **rfal_analogConfig.h**

- RFAL_ANALOG_CONFIG_DIRECTION_RFU6 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_DIRECTION_RFU7 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_DIRECTION_RFU8 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_DIRECTION_RFU9 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_DIRECTION_SHIFT : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_DLMA : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_DPO : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_GET_BITRATE : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_GET_DIRECTION : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_GET_POLL_LISTEN : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_GET_TECH : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_IS_106 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_IS_1695 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_IS_1p6 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_IS_212 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_IS_26 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_IS_3390 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_IS_424 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_IS_6780 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_IS_848 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_IS_AP2P : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_IS_CHIP : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_IS_COMMON : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_IS_NFCA : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_IS_NFCB : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_IS_NFCF : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_IS_NFCV : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_IS_RX : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_IS_TX : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_ID_SET : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_LISTEN : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_LUT_NOT_FOUND : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_LUT_SIZE : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_NO_DIRECTION : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_POLL : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_POLL_LISTEN_MODE_MASK : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_POLL_LISTEN_MODE_SHIFT : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_RX : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_TBL_SIZE : **rfal_analogConfig.h**

- RFAL_ANALOG_CONFIG_TECH_AP2P : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_TECH_CHIP : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_TECH_MASK : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_TECH_NFCA : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_TECH_NFCB : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_TECH_NFCF : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_TECH_NFCV : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_TECH_RFU : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_TECH_RFU2 : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_TECH_SHIFT : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_TX : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_UPDATE_LAST : **rfal_analogConfig.h**
- RFAL_ANALOG_CONFIG_UPDATE_MORE : **rfal_analogConfig.h**
- RFAL_BITS_IN_BYTE : **rfal_rf.h**
- RFAL_BYTECMP : **rfal_utils.h**
- RFAL_CRC_LEN : **rfal_rf.h**
- RFAL_DLMA_TABLE_MAX_ENTRIES : **rfal_dlma.h**
- RFAL_DLMA_TABLE_PARAM_LEN : **rfal_dlma.h**
- RFAL_DLMA_TABLE_SIZE_MAX : **rfal_dlma.h**
- RFAL_DPO_TABLE_MAX_ENTRIES : **rfal_dpo.h**
- RFAL_DPO_TABLE_PARAM_LEN : **rfal_dpo.h**
- RFAL_DPO_TABLE_SIZE_MAX : **rfal_dpo.h**
- RFAL_ERR_AGAIN : **rfal_utils.h**
- RFAL_ERR_BUSY : **rfal_utils.h**
- RFAL_ERR_CRC : **rfal_utils.h**
- RFAL_ERR_DISABLED : **rfal_utils.h**
- RFAL_ERR_DONE : **rfal_utils.h**
- RFAL_ERR_FIFO : **rfal_utils.h**
- RFAL_ERR_FRAMING : **rfal_utils.h**
- RFAL_ERR_HW_MISMATCH : **rfal_utils.h**
- RFAL_ERR_HW_OVERRUN : **rfal_utils.h**
- RFAL_ERR_IGNORE : **rfal_utils.h**
- RFAL_ERR_INCOMPLETE_BYTE : **rfal_utils.h**
- RFAL_ERR_INCOMPLETE_BYTE_01 : **rfal_utils.h**
- RFAL_ERR_INCOMPLETE_BYTE_02 : **rfal_utils.h**
- RFAL_ERR_INCOMPLETE_BYTE_03 : **rfal_utils.h**
- RFAL_ERR_INCOMPLETE_BYTE_04 : **rfal_utils.h**
- RFAL_ERR_INCOMPLETE_BYTE_05 : **rfal_utils.h**
- RFAL_ERR_INCOMPLETE_BYTE_06 : **rfal_utils.h**

- RFAL_ERR_INCOMPLETE_BYTE_07 : **rfal_utils.h**
- RFAL_ERR_INTERNAL : **rfal_utils.h**
- RFAL_ERR_INVALID_HANDLE : **rfal_utils.h**
- RFAL_ERR_IO : **rfal_utils.h**
- RFAL_ERR_LINK_LOSS : **rfal_utils.h**
- RFAL_ERR_MAX_RERUNS : **rfal_utils.h**
- RFAL_ERR_MEM_CORRUPT : **rfal_utils.h**
- RFAL_ERR_NOMEM : **rfal_utils.h**
- RFAL_ERR_NOMSG : **rfal_utils.h**
- RFAL_ERR_NONE : **rfal_utils.h**
- RFAL_ERR_NOT_IMPLEMENTED : **rfal_utils.h**
- RFAL_ERR_NOTFOUND : **rfal_utils.h**
- RFAL_ERR_NOTSUPP : **rfal_utils.h**
- RFAL_ERR_NOTUNIQUE : **rfal_utils.h**
- RFAL_ERR_OVERRUN : **rfal_utils.h**
- RFAL_ERR_PAR : **rfal_utils.h**
- RFAL_ERR_PARAM : **rfal_utils.h**
- RFAL_ERR_PC_CORRUPT : **rfal_utils.h**
- RFAL_ERR_PROTO : **rfal_utils.h**
- RFAL_ERR_RELEASE_REQ : **rfal_utils.h**
- RFAL_ERR_REQUEST : **rfal_utils.h**
- RFAL_ERR_RF_COLLISION : **rfal_utils.h**
- RFAL_ERR_SEMANTIC : **rfal_utils.h**
- RFAL_ERR_SEND : **rfal_utils.h**
- RFAL_ERR_SLEEP_REQ : **rfal_utils.h**
- RFAL_ERR_SYNTAX : **rfal_utils.h**
- RFAL_ERR_SYSTEM : **rfal_utils.h**
- RFAL_ERR_TIMEOUT : **rfal_utils.h**
- RFAL_ERR_WRITE : **rfal_utils.h**
- RFAL_ERR_WRONG_STATE : **rfal_utils.h**
- RFAL_EXIT_ON_BUSY : **rfal_utils.h**
- RFAL_EXIT_ON_ERR : **rfal_utils.h**
- RFAL_FDT_LISTEN_AP2P_LISTENER : **rfal_rf.h**
- RFAL_FDT_LISTEN_AP2P_POLLER : **rfal_rf.h**
- RFAL_FDT_LISTEN_NFCA_LISTENER : **rfal_rf.h**
- RFAL_FDT_LISTEN_NFCA_POLLER : **rfal_rf.h**
- RFAL_FDT_LISTEN_NFCB_LISTENER : **rfal_rf.h**
- RFAL_FDT_LISTEN_NFCB_POLLER : **rfal_rf.h**
- RFAL_FDT_LISTEN_NFCF_LISTENER : **rfal_rf.h**

- RFAL_FDT_LISTEN_NFCF_POLLER : **rfal_rf.h**
- RFAL_FDT_LISTEN_NFCV_POLLER : **rfal_rf.h**
- RFAL_FDT_LISTEN_PICOPASS_POLLER : **rfal_rf.h**
- RFAL_FDT_POLL_AP2P_POLLER : **rfal_rf.h**
- RFAL_FDT_POLL_NFCA_POLLER : **rfal_rf.h**
- RFAL_FDT_POLL_NFCA_T1T_POLLER : **rfal_rf.h**
- RFAL_FDT_POLL_NFCB_POLLER : **rfal_rf.h**
- RFAL_FDT_POLL_NFCF_POLLER : **rfal_rf.h**
- RFAL_FDT_POLL_NFCV_POLLER : **rfal_rf.h**
- RFAL_FDT_POLL_PICOPASS_POLLER : **rfal_rf.h**
- RFAL_FEATURE_DLMA : **rfal_defConfig.h**, **rfal_platform.h**
- RFAL_FEATURE_DPO : **rfal_defConfig.h**, **rfal_platform.h**
- RFAL_FEATURE_DYNAMIC_ANALOG_CONFIG : **rfal_defConfig.h**, **rfal_platform.h**
- RFAL_FEATURE_ISO_DEP : **rfal_platform.h**
- RFAL_FEATURE_ISO_DEP_APDU_MAX_LEN : **rfal_isoDep.h**, **rfal_platform.h**
- RFAL_FEATURE_ISO_DEP_IBLOCK_MAX_LEN : **rfal_isoDep.h**, **rfal_platform.h**
- RFAL_FEATURE_ISO_DEP_LISTEN : **rfal_platform.h**
- RFAL_FEATURE_ISO_DEP_POLL : **rfal_platform.h**
- RFAL_FEATURE_LISTEN_MODE : **rfal_platform.h**
- RFAL_FEATURE_LOWPOWER_MODE : **rfal_defConfig.h**, **rfal_platform.h**
- RFAL_FEATURE_NFC_DEP : **rfal_platform.h**
- RFAL_FEATURE_NFC_DEP_BLOCK_MAX_LEN : **rfal_nfcDep.h**, **rfal_platform.h**
- RFAL_FEATURE_NFC_DEP_PDU_MAX_LEN : **rfal_nfcDep.h**, **rfal_platform.h**
- RFAL_FEATURE_NFC_RF_BUF_LEN : **rfal_defConfig.h**, **rfal_platform.h**
- RFAL_FEATURE_NFCA : **rfal_platform.h**
- RFAL_FEATURE_NFCB : **rfal_platform.h**
- RFAL_FEATURE_NFCF : **rfal_platform.h**
- RFAL_FEATURE_NFCV : **rfal_platform.h**
- RFAL_FEATURE_ST25TB : **rfal_platform.h**
- RFAL_FEATURE_ST25xV : **rfal_defConfig.h**, **rfal_platform.h**
- RFAL_FEATURE_T1T : **rfal_platform.h**
- RFAL_FEATURE_T2T : **rfal_platform.h**
- RFAL_FEATURE_T4T : **rfal_platform.h**
- RFAL_FEATURE_WAKEUP_MODE : **rfal_defConfig.h**, **rfal_platform.h**
- RFAL_FELICA_LEN_LEN : **rfal_rf.h**
- RFAL_FELICA_POLL_MAX_SLOTS : **rfal_rf.h**
- RFAL_FELICA_POLL_REQ_LEN : **rfal_rf.h**
- RFAL_FELICA_POLL_RES_LEN : **rfal_rf.h**
- RFAL_FWT_NONE : **rfal_rf.h**

- RFAL_GETU16 : **rfal_utils.h**
- RFAL_GETU32 : **rfal_utils.h**
- RFAL_GT_AP2P : **rfal_rf.h**
- RFAL_GT_AP2P_ADJUSTED : **rfal_rf.h**
- RFAL_GT_NFCA : **rfal_rf.h**
- RFAL_GT_NFCB : **rfal_rf.h**
- RFAL_GT_NFCF : **rfal_rf.h**
- RFAL_GT_NFCV : **rfal_rf.h**
- RFAL_GT_NONE : **rfal_rf.h**
- RFAL_GT_PICOPASS : **rfal_rf.h**
- RFAL_ISODEP_ADV_FEATURE : **rfal_isoDep.h**
- RFAL_ISODEP_APDU_MAX_LEN : **rfal_isoDep.h**
- RFAL_ISODEP_ATS_FWI_MASK : **rfal_isoDep.h**
- RFAL_ISODEP_ATS_HB_MAX_LEN : **rfal_isoDep.h**
- RFAL_ISODEP_ATS_SFGI_MASK : **rfal_isoDep.h**
- RFAL_ISODEP_ATS_T0_FSCI_MASK : **rfal_isoDep.h**
- RFAL_ISODEP_ATS_T0_OFFSET : **rfal_isoDep.h**
- RFAL_ISODEP_ATS_T0_TA_PRESENCE_MASK : **rfal_isoDep.h**
- RFAL_ISODEP_ATS_T0_TB_PRESENCE_MASK : **rfal_isoDep.h**
- RFAL_ISODEP_ATS_T0_TC_PRESENCE_MASK : **rfal_isoDep.h**
- RFAL_ISODEP_ATS_TA_DLP_212 : **rfal_isoDep.h**
- RFAL_ISODEP_ATS_TA_DLP_424 : **rfal_isoDep.h**
- RFAL_ISODEP_ATS_TA_DLP_848 : **rfal_isoDep.h**
- RFAL_ISODEP_ATS_TA_DPL_212 : **rfal_isoDep.h**
- RFAL_ISODEP_ATS_TA_DPL_424 : **rfal_isoDep.h**
- RFAL_ISODEP_ATS_TA_DPL_848 : **rfal_isoDep.h**
- RFAL_ISODEP_ATS_TA_SAME_D : **rfal_isoDep.h**
- RFAL_ISODEP_ATS_TB_FWI_MASK : **rfal_isoDep.h**
- RFAL_ISODEP_ATS_TB_FWI_SHIFT : **rfal_isoDep.h**
- RFAL_ISODEP_ATS_TB_SFGI_MASK : **rfal_isoDep.h**
- RFAL_ISODEP_ATTRIB_HLINFO_LEN : **rfal_isoDep.h**
- RFAL_ISODEP_ATTRIB_REQ_MIN_LEN : **rfal_isoDep.h**
- RFAL_ISODEP_ATTRIB_REQ_PARAM1_DEFAULT : **rfal_isoDep.h**
- RFAL_ISODEP_ATTRIB_RES_MBLI_NO_INFO : **rfal_isoDep.h**
- RFAL_ISODEP_ATTRIB_RES_MIN_LEN : **rfal_isoDep.h**
- RFAL_ISODEP_BITRATE_RFU_MASK : **rfal_isoDep.h**
- RFAL_ISODEP_BRI_MASK : **rfal_isoDep.h**
- RFAL_ISODEP_BSI_MASK : **rfal_isoDep.h**
- RFAL_ISODEP_DEFAULT_FSC : **rfal_isoDep.h**

- RFAL_ISODEP_DEFAULT_FSCI : **rfal_isoDep.h**
- RFAL_ISODEP_DEFAULT_FWI : **rfal_isoDep.h**
- RFAL_ISODEP_DEFAULT_SFGI : **rfal_isoDep.h**
- RFAL_ISODEP_DID_LEN : **rfal_isoDep.h**
- RFAL_ISODEP_DID_MAX : **rfal_isoDep.h**
- RFAL_ISODEP_FSDI_DEFAULT : **rfal_isoDep.h**
- RFAL_ISODEP_FSX_KEEP : **rfal_isoDep.h**
- RFAL_ISODEP_FWI_DEFAULT : **rfal_isoDep.h**
- RFAL_ISODEP_FWI_MASK : **rfal_isoDep.h**
- RFAL_ISODEP_FWI_SHIFT : **rfal_isoDep.h**
- RFAL_ISODEP_MAX_DSL_RETRYS : **rfal_isoDep.h**
- RFAL_ISODEP_MAX_FWT : **rfal_isoDep.h**
- RFAL_ISODEP_MAX_I_RETRYS : **rfal_isoDep.h**
- RFAL_ISODEP_MAX_R_RETRYS : **rfal_isoDep.h**
- RFAL_ISODEP_MAX_WTX_NACK_RETRYS : **rfal_isoDep.h**
- RFAL_ISODEP_MAX_WTX_RETRYS : **rfal_isoDep.h**
- RFAL_ISODEP_MAX_WTX_RETRYS_ULTD : **rfal_isoDep.h**
- RFAL_ISODEP_NAD_LEN : **rfal_isoDep.h**
- RFAL_ISODEP_NO_DID : **rfal_isoDep.h**
- RFAL_ISODEP_NO_NAD : **rfal_isoDep.h**
- RFAL_ISODEP_PCB_LEN : **rfal_isoDep.h**
- RFAL_ISODEP_PROLOGUE_SIZE : **rfal_isoDep.h**
- RFAL_ISODEP_RATS_RETRIES : **rfal_isoDep.h**
- RFAL_ISODEP_SAME_BITRATE_MASK : **rfal_isoDep.h**
- RFAL_ISODEP_SPARAM_TAG_BLOCKINFO : **rfal_isoDep.h**
- RFAL_ISODEP_SPARAM_TAG_BRACK : **rfal_isoDep.h**
- RFAL_ISODEP_SPARAM_TAG_BRACK_LEN : **rfal_isoDep.h**
- RFAL_ISODEP_SPARAM_TAG_BRACT : **rfal_isoDep.h**
- RFAL_ISODEP_SPARAM_TAG_BRIND : **rfal_isoDep.h**
- RFAL_ISODEP_SPARAM_TAG_BRREQ : **rfal_isoDep.h**
- RFAL_ISODEP_SPARAM_TAG_BRREQ_LEN : **rfal_isoDep.h**
- RFAL_ISODEP_SPARAM_TAG_LEN : **rfal_isoDep.h**
- RFAL_ISODEP_SPARAM_TAG_PCD2PICC_LEN : **rfal_isoDep.h**
- RFAL_ISODEP_SPARAM_TAG_PICC2PCD_LEN : **rfal_isoDep.h**
- RFAL_ISODEP_SPARAM_TAG_SEL_FRAME : **rfal_isoDep.h**
- RFAL_ISODEP_SPARAM_TAG_SEL_PCD2PICC : **rfal_isoDep.h**
- RFAL_ISODEP_SPARAM_TAG_SEL_PICC2PCD : **rfal_isoDep.h**
- RFAL_ISODEP_SPARAM_TAG_SUP_FRAME : **rfal_isoDep.h**
- RFAL_ISODEP_SPARAM_TAG_SUP_PCD2PICC : **rfal_isoDep.h**

- RFAL_ISODEP_SPARAM_TAG_SUP_PICC2PCD : **rfal_isoDep.h**
- RFAL_ISODEP_SPARAM_VALUES_MAX_LEN : **rfal_isoDep.h**
- RFAL_LM_MASK_ACTIVE_P2P : **rfal_rf.h**
- RFAL_LM_MASK_NFCA : **rfal_rf.h**
- RFAL_LM_MASK_NFCB : **rfal_rf.h**
- RFAL_LM_MASK_NFCF : **rfal_rf.h**
- RFAL_LM_SENS_RES_LEN : **rfal_rf.h**
- RFAL_LM_SENSB_RES_LEN : **rfal_rf.h**
- RFAL_LM_SENSF_RES_LEN : **rfal_rf.h**
- RFAL_LM_SENSF_SC_LEN : **rfal_rf.h**
- RFAL_MAX : **rfal_utils.h**
- RFAL_MEMCPY : **rfal_utils.h**
- RFAL_MEMMOVE : **rfal_utils.h**
- RFAL_MEMSET : **rfal_utils.h**
- RFAL_MIN : **rfal_utils.h**
- RFAL_NFC_LISTEN_TECH_A : **rfal_nfc.h**
- RFAL_NFC_LISTEN_TECH_AP2P : **rfal_nfc.h**
- RFAL_NFC_LISTEN_TECH_B : **rfal_nfc.h**
- RFAL_NFC_LISTEN_TECH_F : **rfal_nfc.h**
- RFAL_NFC_POLL_TECH_A : **rfal_nfc.h**
- RFAL_NFC_POLL_TECH_AP2P : **rfal_nfc.h**
- RFAL_NFC_POLL_TECH_B : **rfal_nfc.h**
- RFAL_NFC_POLL_TECH_F : **rfal_nfc.h**
- RFAL_NFC_POLL_TECH_PROP : **rfal_nfc.h**
- RFAL_NFC_POLL_TECH_ST25TB : **rfal_nfc.h**
- RFAL_NFC_POLL_TECH_V : **rfal_nfc.h**
- RFAL_NFC_TECH_NONE : **rfal_nfc.h**
- RFAL_NFCA_CASCADE_1_UID_LEN : **rfal_nfca.h**
- RFAL_NFCA_CASCADE_2_UID_LEN : **rfal_nfca.h**
- RFAL_NFCA_CASCADE_3_UID_LEN : **rfal_nfca.h**
- RFAL_NFCA_FDTMIN : **rfal_nfca.h**
- RFAL_NFCA_SEL_RES_CONF_MASK : **rfal_nfca.h**
- RFAL_NFCA_SEL_RES_CONF_NFCDEP : **rfal_nfca.h**
- RFAL_NFCA_SEL_RES_CONF_T2T : **rfal_nfca.h**
- RFAL_NFCA_SEL_RES_CONF_T4T : **rfal_nfca.h**
- RFAL_NFCA_SEL_RES_CONF_T4T_NFCDEP : **rfal_nfca.h**
- RFAL_NFCA_SENS_RES_PLATFORM_MASK : **rfal_nfca.h**
- RFAL_NFCA_SENS_RES_PLATFORM_T1T : **rfal_nfca.h**
- RFAL_NFCB_AFI : **rfal_nfcb.h**

- RFAL_NFCB_CMD_LEN : **rfal_nfcb.h**
- RFAL_NFCB_CRC_LEN : **rfal_nfcb.h**
- RFAL_NFCB_DFWT : **rfal_nfcb.h**
- RFAL_NFCB_DTPOLL_10 : **rfal_nfcb.h**
- RFAL_NFCB_DTPOLL_20 : **rfal_nfcb.h**
- RFAL_NFCB_FWTSENSB : **rfal_nfcb.h**
- RFAL_NFCB_NFCID0_LEN : **rfal_nfcb.h**
- RFAL_NFCB_PARAM : **rfal_nfcb.h**
- RFAL_NFCB_SENSB_REQ_ADV_FEATURE : **rfal_nfcb.h**
- RFAL_NFCB_SENSB_RES_ADC_ADV_FEATURE_MASK : **rfal_nfcb.h**
- RFAL_NFCB_SENSB_RES_ADC_MASK : **rfal_nfcb.h**
- RFAL_NFCB_SENSB_RES_ADC_PROPRIETARY_MASK : **rfal_nfcb.h**
- RFAL_NFCB_SENSB_RES_EXT_LEN : **rfal_nfcb.h**
- RFAL_NFCB_SENSB_RES_FO_DID_MASK : **rfal_nfcb.h**
- RFAL_NFCB_SENSB_RES_FO_MASK : **rfal_nfcb.h**
- RFAL_NFCB_SENSB_RES_FO_NAD_MASK : **rfal_nfcb.h**
- RFAL_NFCB_SENSB_RES_FSCI_MASK : **rfal_nfcb.h**
- RFAL_NFCB_SENSB_RES_FSCI_SHIFT : **rfal_nfcb.h**
- RFAL_NFCB_SENSB_RES_FWI_MASK : **rfal_nfcb.h**
- RFAL_NFCB_SENSB_RES_FWI_SHIFT : **rfal_nfcb.h**
- RFAL_NFCB_SENSB_RES_LEN : **rfal_nfcb.h**
- RFAL_NFCB_SENSB_RES_PROTO_ISO_MASK : **rfal_nfcb.h**
- RFAL_NFCB_SENSB_RES_PROTO_RFU_MASK : **rfal_nfcb.h**
- RFAL_NFCB_SENSB_RES_PROTO_TR2_MASK : **rfal_nfcb.h**
- RFAL_NFCB_SENSB_RES_PROTO_TR2_SHIFT : **rfal_nfcb.h**
- RFAL_NFCB_SENSB_RES_SFGI_MASK : **rfal_nfcb.h**
- RFAL_NFCB_SENSB_RES_SFGI_SHIFT : **rfal_nfcb.h**
- RFAL_NFCDEP_ATR_REQ_NFCID3_POS : **rfal_nfcDep.h**
- RFAL_NFCDEP_ATRREQ_MAX_LEN : **rfal_nfcDep.h**
- RFAL_NFCDEP_ATRREQ_MIN_LEN : **rfal_nfcDep.h**
- RFAL_NFCDEP_ATRRES_HEADER_LEN : **rfal_nfcDep.h**
- RFAL_NFCDEP_ATRRES_MAX_LEN : **rfal_nfcDep.h**
- RFAL_NFCDEP_ATRRES_MIN_LEN : **rfal_nfcDep.h**
- RFAL_NFCDEP_BR_MASK_106 : **rfal_nfcDep.h**
- RFAL_NFCDEP_BR_MASK_212 : **rfal_nfcDep.h**
- RFAL_NFCDEP_BR_MASK_424 : **rfal_nfcDep.h**
- RFAL_NFCDEP_BRS_DSI_POS : **rfal_nfcDep.h**
- RFAL_NFCDEP_BRS_Dx_MASK : **rfal_nfcDep.h**
- RFAL_NFCDEP_BRS_MAINTAIN : **rfal_nfcDep.h**

- RFAL_NFCDEP_CMD_LEN : **rfal_nfcDep.h**
- RFAL_NFCDEP_CMDTYPE_LEN : **rfal_nfcDep.h**
- RFAL_NFCDEP_DEP_HEADER : **rfal_nfcDep.h**
- RFAL_NFCDEP_DEP_PFB_LEN : **rfal_nfcDep.h**
- RFAL_NFCDEP_DEPREQ_HEADER_LEN : **rfal_nfcDep.h**
- RFAL_NFCDEP_DID_KEEP : **rfal_nfcDep.h**
- RFAL_NFCDEP_DID_LEN : **rfal_nfcDep.h**
- RFAL_NFCDEP_DID_MAX : **rfal_nfcDep.h**
- RFAL_NFCDEP_DID_NO : **rfal_nfcDep.h**
- RFAL_NFCDEP_DSL_RLS_LEN_DID : **rfal_nfcDep.h**
- RFAL_NFCDEP_DSL_RLS_LEN_NO_DID : **rfal_nfcDep.h**
- RFAL_NFCDEP_FRAME_SIZE_MAX_LEN : **rfal_nfcDep.h**
- RFAL_NFCDEP_FS_VAL_MIN : **rfal_nfcDep.h**
- RFAL_NFCDEP_GB_MAX_LEN : **rfal_nfcDep.h**
- RFAL_NFCDEP_HEADER : **rfal_nfcDep.h**
- RFAL_NFCDEP_LEN_LEN : **rfal_nfcDep.h**
- RFAL_NFCDEP_LEN_MAX : **rfal_nfcDep.h**
- RFAL_NFCDEP_LEN_MIN : **rfal_nfcDep.h**
- RFAL_NFCDEP_LR_VAL_MASK : **rfal_nfcDep.h**
- RFAL_NFCDEP_MAX_FWT : **rfal_nfcDep.h**
- RFAL_NFCDEP_NAD_NO : **rfal_nfcDep.h**
- RFAL_NFCDEP_NFCID3_LEN : **rfal_nfcDep.h**
- RFAL_NFCDEP_OPER_ATN_DIS : **rfal_nfcDep.h**
- RFAL_NFCDEP_OPER_ATN_EN : **rfal_nfcDep.h**
- RFAL_NFCDEP_OPER_EMPTY_DEP_DIS : **rfal_nfcDep.h**
- RFAL_NFCDEP_OPER_EMPTY_DEP_EN : **rfal_nfcDep.h**
- RFAL_NFCDEP_OPER_FULL_MI_DIS : **rfal_nfcDep.h**
- RFAL_NFCDEP_OPER_FULL_MI_EN : **rfal_nfcDep.h**
- RFAL_NFCDEP_OPER_RTOX_REQ_DIS : **rfal_nfcDep.h**
- RFAL_NFCDEP_OPER_RTOX_REQ_EN : **rfal_nfcDep.h**
- RFAL_NFCDEP_PP_LR_MASK : **rfal_nfcDep.h**
- RFAL_NFCDEP_PP_LR_SHIFT : **rfal_nfcDep.h**
- RFAL_NFCDEP_RWT_INI_MAX : **rfal_nfcDep.h**
- RFAL_NFCDEP_RWT_TRG_MAX : **rfal_nfcDep.h**
- RFAL_NFCDEP_SB_LEN : **rfal_nfcDep.h**
- RFAL_NFCDEP_WT_DELTA : **rfal_nfcDep.h**
- RFAL_NFCDEP_WT_DELTA_ADJUST : **rfal_nfcDep.h**
- RFAL_NFCDEP_WT_INI_DEFAULT : **rfal_nfcDep.h**
- RFAL_NFCDEP_WT_INI_MAX : **rfal_nfcDep.h**

- RFAL_NFCDEP_WT_INI_MIN : **rfal_nfcDep.h**
- RFAL_NFCDEP_WT_MASK : **rfal_nfcDep.h**
- RFAL_NFCDEP_WT_TRG_MAX : **rfal_nfcDep.h**
- RFAL_NFCDEP_WT_TRG_MAX_D10 : **rfal_nfcDep.h**
- RFAL_NFCDEP_WT_TRG_MAX_D11 : **rfal_nfcDep.h**
- RFAL_NFCDEP_WT_TRG_MAX_L13 : **rfal_nfcDep.h**
- RFAL_NFCF_BLOCK_LEN : **rfal_nfcf.h**
- RFAL_NFCF_BLOCKLISTELEM_LEN_BIT : **rfal_nfcf.h**
- RFAL_NFCF_BLOCKLISTELEM_MAX_LEN : **rfal_nfcf.h**
- RFAL_NFCF_CHECKUPDATE_RES_NOB_POS : **rfal_nfcf.h**
- RFAL_NFCF_CHECKUPDATE_RES_ST1_POS : **rfal_nfcf.h**
- RFAL_NFCF_CHECKUPDATE_RES_ST2_POS : **rfal_nfcf.h**
- RFAL_NFCF_CMD_LEN : **rfal_nfcf.h**
- RFAL_NFCF_CMD_POS : **rfal_nfcf.h**
- RFAL_NFCF_HEADER_LEN : **rfal_nfcf.h**
- RFAL_NFCF_LENGTH_LEN : **rfal_nfcf.h**
- RFAL_NFCF_NFCID2_LEN : **rfal_nfcf.h**
- RFAL_NFCF_NOB_LEN : **rfal_nfcf.h**
- RFAL_NFCF_NOS_LEN : **rfal_nfcf.h**
- RFAL_NFCF_POLL_MAXCARDS : **rfal_nfcf.h**
- RFAL_NFCF_SENSF_NFCID2_BYTE1_NFCDEP : **rfal_nfcf.h**
- RFAL_NFCF_SENSF_NFCID2_BYTE1_POS : **rfal_nfcf.h**
- RFAL_NFCF_SENSF_NFCID2_BYTE2_NFCDEP : **rfal_nfcf.h**
- RFAL_NFCF_SENSF_NFCID2_BYTE2_POS : **rfal_nfcf.h**
- RFAL_NFCF_SENSF_NFCID2_PROT_TYPE_LEN : **rfal_nfcf.h**
- RFAL_NFCF_SENSF_PARAMS_RC_POS : **rfal_nfcf.h**
- RFAL_NFCF_SENSF_PARAMS_SC1_POS : **rfal_nfcf.h**
- RFAL_NFCF_SENSF_PARAMS_SC2_POS : **rfal_nfcf.h**
- RFAL_NFCF_SENSF_PARAMS_TSN_POS : **rfal_nfcf.h**
- RFAL_NFCF_SENSF_RES_BYTE1 : **rfal_nfcf.h**
- RFAL_NFCF_SENSF_RES_LEN_MAX : **rfal_nfcf.h**
- RFAL_NFCF_SENSF_RES_LEN_MIN : **rfal_nfcf.h**
- RFAL_NFCF_SENSF_RES_PAD0_LEN : **rfal_nfcf.h**
- RFAL_NFCF_SENSF_RES_PAD1_LEN : **rfal_nfcf.h**
- RFAL_NFCF_SENSF_RES_RD_LEN : **rfal_nfcf.h**
- RFAL_NFCF_SENSF_SC_LEN : **rfal_nfcf.h**
- RFAL_NFCF_SERVICECODE_RDONLY : **rfal_nfcf.h**
- RFAL_NFCF_SERVICECODE_RDWR : **rfal_nfcf.h**
- RFAL_NFCF_STATUS_FLAG_ERROR : **rfal_nfcf.h**

- RFAL_NFCF_STATUS_FLAG_SUCCESS : **rfal_nfcf.h**
- RFAL_NFCF_SYSTEMCODE : **rfal_nfcf.h**
- RFAL_NFCF_SYSTEMCODE_LEN : **rfal_nfcf.h**
- RFAL_NFCF_TEST_LB_CMD0 : **rfal_nfcf.h**
- RFAL_NFCF_TEST_LB_CMD1 : **rfal_nfcf.h**
- RFAL_NFCID1_DOUBLE_LEN : **rfal_rf.h**
- RFAL_NFCID1_SINGLE_LEN : **rfal_rf.h**
- RFAL_NFCID1_TRIPLE_LEN : **rfal_rf.h**
- RFAL_NFCID2_LEN : **rfal_rf.h**
- RFAL_NFCID3_LEN : **rfal_rf.h**
- RFAL_NFCV_BLOCKNUM_EXTENDED_LEN : **rfal_nfcv.h**
- RFAL_NFCV_BLOCKNUM_LEN : **rfal_nfcv.h**
- RFAL_NFCV_BLOCKNUM_M24LR_LEN : **rfal_st25xv.h**
- RFAL_NFCV_BNO_LEN : **rfal_nfcv.h**
- RFAL_NFCV_CRC_LEN : **rfal_nfcv.h**
- RFAL_NFCV_MAX_BLOCK_LEN : **rfal_nfcv.h**
- RFAL_NFCV_MAX_GEN_DATA_LEN : **rfal_nfcv.h**
- RFAL_NFCV_PARAM_SKIP : **rfal_nfcv.h**
- RFAL_NFCV_ST_IC_MFG_CODE : **rfal_nfcv.h**
- RFAL_NFCV_UID_LEN : **rfal_nfcv.h**
- RFAL_NO_WARNING : **rfal_utils.h**
- RFAL_SIZEOF_ARRAY : **rfal_utils.h**
- RFAL_ST25TB_BLOCK_LEN : **rfal_st25tb.h**
- RFAL_ST25TB_CHIP_ID_LEN : **rfal_st25tb.h**
- RFAL_ST25TB_CRC_LEN : **rfal_st25tb.h**
- RFAL_ST25TB_UID_LEN : **rfal_st25tb.h**
- RFAL_SUPPORT_AP2P : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_AP2P_106 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_AP2P_1695 : **st25r3911/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_AP2P_212 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_AP2P_3390 : **st25r3911/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_AP2P_424 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_AP2P_6780 : **st25r3911/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_AP2P_848 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**

- RFAL_SUPPORT_BR_CE_A_106 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_CE_A_212 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_CE_A_424 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_CE_A_848 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_CE_B_106 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_CE_B_212 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_CE_B_424 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_CE_B_848 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_CE_F_212 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_CE_F_424 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_RW_106 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_RW_13560 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**
- RFAL_SUPPORT_BR_RW_1695 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_RW_212 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_RW_26 : **st25r200/rfal_features.h**, **st25r500/rfal_features.h**
- RFAL_SUPPORT_BR_RW_3390 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_RW_424 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_RW_53 : **st25r200/rfal_features.h**, **st25r500/rfal_features.h**
- RFAL_SUPPORT_BR_RW_6780 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_BR_RW_848 : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**

- RFAL_SUPPORT_CE : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_MODE_LISTEN_ACTIVE_P2P : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_MODE_LISTEN_NFCA : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_MODE_LISTEN_NFCB : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_MODE_LISTEN_NFCF : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_MODE_POLL_ACTIVE_P2P : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_MODE_POLL_NFCA : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_MODE_POLL_NFCB : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_MODE_POLL_NFCF : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_MODE_POLL_NFCV : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_SUPPORT_RW : **st25r200/rfal_features.h**, **st25r3911/rfal_features.h**, **st25r3916/rfal_features.h**, **st25r500/rfal_features.h**, **st25r95/rfal_features.h**
- RFAL_T1T_HR0_NDEF_MASK : **rfal_t1t.h**
- RFAL_T1T_HR0_NDEF_SUPPORT : **rfal_t1t.h**
- RFAL_T1T_HR_LENGTH : **rfal_t1t.h**
- RFAL_T1T_UID_LEN : **rfal_t1t.h**
- RFAL_T2T_BLOCK_LEN : **rfal_t2t.h**
- RFAL_T2T_READ_DATA_LEN : **rfal_t2t.h**
- RFAL_T2T_WRITE_DATA_LEN : **rfal_t2t.h**
- RFAL_T4T_CLA : **rfal_t4t.h**
- RFAL_T4T_ISO7816_P1_SELECT_BY_DF_NAME : **rfal_t4t.h**
- RFAL_T4T_ISO7816_P1_SELECT_BY_FILEID : **rfal_t4t.h**
- RFAL_T4T_ISO7816_P2_SELECT_FIRST_OR_ONLY_OCCURENCE : **rfal_t4t.h**
- RFAL_T4T_ISO7816_P2_SELECT_NO_RESPONSE_DATA : **rfal_t4t.h**
- RFAL_T4T_ISO7816_P2_SELECT_RETURN_FCI_TEMPLATE : **rfal_t4t.h**
- RFAL_T4T_ISO7816_STATUS_COMPLETE : **rfal_t4t.h**
- RFAL_T4T_LC_LEN : **rfal_t4t.h**
- RFAL_T4T_LE_LEN : **rfal_t4t.h**
- RFAL_T4T_MAX_CAPDU_PROLOGUE_LEN : **rfal_t4t.h**

- RFAL_T4T_MAX_RAPDU_SW1SW2_LEN : **rfal_t4t.h**
- RFAL_TIMING_NONE : **rfal_rf.h**
- RFAL_TXRX_FLAGS_DEFAULT : **rfal_rf.h**
- RFAL_US_IN_MS : **rfal_rf.h**
- RFAL_VERSION : **rfal_rf.h**
- RFAL_WUM_REFERENCE_AUTO : **rfal_rf.h**
- rfalConv1fcTo2018fc : **rfal_rf.h**
- rfalConv1fcTo4096fc : **rfal_rf.h**
- rfalConv1fcTo512fc : **rfal_rf.h**
- rfalConv1fcTo64fc : **rfal_rf.h**
- rfalConv1fcTo8fc : **rfal_rf.h**
- rfalConv1fcToMs : **rfal_rf.h**
- rfalConv1fcToUs : **rfal_rf.h**
- rfalConv2048fcTo1fc : **rfal_rf.h**
- rfalConv4096fcTo1fc : **rfal_rf.h**
- rfalConv512fcTo1fc : **rfal_rf.h**
- rfalConv64fcTo1fc : **rfal_rf.h**
- rfalConv64fcToMs : **rfal_rf.h**
- rfalConv8fcTo1fc : **rfal_rf.h**
- rfalConvBitsToBytes : **rfal_rf.h**
- rfalConvBytesToBits : **rfal_rf.h**
- rfalConvMsTo1fc : **rfal_rf.h**
- rfalConvMsTo64fc : **rfal_rf.h**
- rfalConvUsTo1fc : **rfal_rf.h**
- rfalCreateByteFlagsTxRxContext : **rfal_rf.h**
- rfalCreateByteTxRxContext : **rfal_rf.h**
- rfalGetMaxBrAP2P : **rfal_rf.h**
- rfalGetMaxBrCEA : **rfal_rf.h**
- rfalGetMaxBrCEB : **rfal_rf.h**
- rfalGetMaxBrCEF : **rfal_rf.h**
- rfalGetMaxBrRW : **rfal_rf.h**
- rfalIsModeActiveComm : **rfal_rf.h**
- rfalIsModePassiveComm : **rfal_rf.h**
- rfalIsModePassiveListen : **rfal_rf.h**
- rfalIsModePassivePoll : **rfal_rf.h**
- rfalLogD : **rfal_rf.h**
- rfalLogE : **rfal_rf.h**
- rfalLogI : **rfal_rf.h**
- rfalLogW : **rfal_rf.h**

- rfalNfcaIsSelResNFCDEP : **rfal_nfca.h**
- rfalNfcaIsSelResT2T : **rfal_nfca.h**
- rfalNfcaIsSelResT4T : **rfal_nfca.h**
- rfalNfcaIsSelResT4TNFCDEP : **rfal_nfca.h**
- rfalNfcaIsSensResT1T : **rfal_nfca.h**
- rfalNfcaLisDevIsMultiProto : **rfal_nfca.h**
- rfalNfcbGetFSCI : **rfal_nfcb.h**
- rfalNfcbIsIsoDepSupported : **rfal_nfcb.h**
- rfalNfcDefaultDiscParams : **rfal_nfc.h**
- rfalNfcDepBRS2DRI : **rfal_nfcDep.h**
- rfalNfcDepBRS2DSI : **rfal_nfcDep.h**
- rfalNfcDepDx2BRS : **rfal_nfcDep.h**
- rfalNfcDepLR2FS : **rfal_nfcDep.h**
- rfalNfcDepLR2PP : **rfal_nfcDep.h**
- rfalNfcDepPP2LR : **rfal_nfcDep.h**
- rfalNfcDepSetNFCID : **rfal_nfcDep.h**
- rfalNfcDepWT2RWT : **rfal_nfcDep.h**
- rfalNfcfIsNfcDepSupported : **rfal_nfcf.h**
- rfalNfcIsDevActivated : **rfal_nfc.h**
- rfalNfcIsInDiscovery : **rfal_nfc.h**
- rfalNfcIsRemDevListener : **rfal_nfc.h**
- rfalNfcIsRemDevPoller : **rfal_nfc.h**
- rfalRunBlocking : **rfal_rf.h**

RFAL

Main Page

Modules

Data Structures

Examples

- s -

- ST25R200_BR_106_26 : **st25r200.h**
- ST25R200_BR_212 : **st25r200.h**
- ST25R200_BR_424_53 : **st25r200.h**
- ST25R200_BR_848 : **st25r200.h**
- ST25R200_BR_DO_NOT_SET : **st25r200.h**
- ST25R200_CMD_ADJUST_REGULATORS : **st25r200.h**
- ST25R200_CMD_CALIBRATE_WU : **st25r200.h**
- ST25R200_CMD_CLEAR_FIFO : **st25r200.h**
- ST25R200_CMD_CLEAR_RXGAIN : **st25r200.h**
- ST25R200_CMD_CLEAR_WU_CALIB : **st25r200.h**

- ST25R200_CMD_LEN : **st25r200.h**
- ST25R200_CMD_MASK_RECEIVE_DATA : **st25r200.h**
- ST25R200_CMD_MEASURE_IQ : **st25r200.h**
- ST25R200_CMD_MEASURE_WU : **st25r200.h**
- ST25R200_CMD_MODE : **st25r200.h**
- ST25R200_CMD_SENSE_RF : **st25r200.h**
- ST25R200_CMD_SET_DEFAULT : **st25r200.h**
- ST25R200_CMD_START_GP_TIMER : **st25r200.h**
- ST25R200_CMD_START_MRT : **st25r200.h**
- ST25R200_CMD_START_NRT : **st25r200.h**
- ST25R200_CMD_START_WUT : **st25r200.h**
- ST25R200_CMD_STOP : **st25r200.h**
- ST25R200_CMD_STOP_NRT : **st25r200.h**
- ST25R200_CMD_TEST_ACCESS : **st25r200.h**
- ST25R200_CMD_TRANSMIT : **st25r200.h**
- ST25R200_CMD_TRANSMIT_EOF : **st25r200.h**
- ST25R200_CMD_TRANSPARENT_MODE : **st25r200.h**
- ST25R200_CMD_UNMASK_RECEIVE_DATA : **st25r200.h**
- ST25R200_FIFO_ACCESS : **st25r200.h**
- ST25R200_FIFO_DEPTH : **st25r200.h**
- ST25R200_FIFO_STATUS_LEN : **st25r200_com.h**
- ST25R200_IRQ_MASK_ALL : **st25r200_irq.h**
- ST25R200_IRQ_MASK_COL : **st25r200_irq.h**
- ST25R200_IRQ_MASK_CRC : **st25r200_irq.h**
- ST25R200_IRQ_MASK_DCT : **st25r200_irq.h**
- ST25R200_IRQ_MASK_GPE : **st25r200_irq.h**
- ST25R200_IRQ_MASK_HFE : **st25r200_irq.h**
- ST25R200_IRQ_MASK_NONE : **st25r200_irq.h**
- ST25R200_IRQ_MASK_NRE : **st25r200_irq.h**
- ST25R200_IRQ_MASK_OSC : **st25r200_irq.h**
- ST25R200_IRQ_MASK_PAR : **st25r200_irq.h**
- ST25R200_IRQ_MASK_RFU1 : **st25r200_irq.h**
- ST25R200_IRQ_MASK_RFU2 : **st25r200_irq.h**
- ST25R200_IRQ_MASK_RFU3 : **st25r200_irq.h**
- ST25R200_IRQ_MASK_RFU4 : **st25r200_irq.h**
- ST25R200_IRQ_MASK_RFU5 : **st25r200_irq.h**
- ST25R200_IRQ_MASK_RFU6 : **st25r200_irq.h**
- ST25R200_IRQ_MASK_RX_REST : **st25r200_irq.h**
- ST25R200_IRQ_MASK_RXE : **st25r200_irq.h**

- ST25R200_IRQ_MASK_RXS : **st25r200_irq.h**
- ST25R200_IRQ_MASK_SFE : **st25r200_irq.h**
- ST25R200_IRQ_MASK_SUBC_START : **st25r200_irq.h**
- ST25R200_IRQ_MASK_TXE : **st25r200_irq.h**
- ST25R200_IRQ_MASK_WL : **st25r200_irq.h**
- ST25R200_IRQ_MASK_WUI : **st25r200_irq.h**
- ST25R200_IRQ_MASK_WUQ : **st25r200_irq.h**
- ST25R200_IRQ_MASK_WUT : **st25r200_irq.h**
- ST25R200_OBS_MODE_LEN : **st25r200_com.h**
- ST25R200_READ_MODE : **st25r200.h**
- ST25R200_REG_COLLISION : **st25r200_com.h**
- ST25R200_REG_CORR1 : **st25r200_com.h**
- ST25R200_REG_CORR2 : **st25r200_com.h**
- ST25R200_REG_CORR3 : **st25r200_com.h**
- ST25R200_REG_CORR4 : **st25r200_com.h**
- ST25R200_REG_CORR5 : **st25r200_com.h**
- ST25R200_REG_CORR6 : **st25r200_com.h**
- ST25R200_REG_DISPLAY1 : **st25r200_com.h**
- ST25R200_REG_DISPLAY2 : **st25r200_com.h**
- ST25R200_REG_DISPLAY3 : **st25r200_com.h**
- ST25R200_REG_DISPLAY4 : **st25r200_com.h**
- ST25R200_REG_DROP_200 : **st25r200.h**
- ST25R200_REG_DROP_250 : **st25r200.h**
- ST25R200_REG_DROP_300 : **st25r200.h**
- ST25R200_REG_DROP_350 : **st25r200.h**
- ST25R200_REG_DROP_400 : **st25r200.h**
- ST25R200_REG_DROP_450 : **st25r200.h**
- ST25R200_REG_DROP_500 : **st25r200.h**
- ST25R200_REG_DROP_550 : **st25r200.h**
- ST25R200_REG_DROP_DO_NOT_SET : **st25r200.h**
- ST25R200_REG_EMD1 : **st25r200_com.h**
- ST25R200_REG_EMD2 : **st25r200_com.h**
- ST25R200_REG_FIFO_STATUS1 : **st25r200_com.h**
- ST25R200_REG_FIFO_STATUS2 : **st25r200_com.h**
- ST25R200_REG_GENERAL : **st25r200_com.h**
- ST25R200_REG_GPT1 : **st25r200_com.h**
- ST25R200_REG_GPT2 : **st25r200_com.h**
- ST25R200_REG_IC_ID : **st25r200_com.h**
- ST25R200_REG_IRQ1 : **st25r200_com.h**

- ST25R200_REG_IRQ2 : **st25r200_com.h**
- ST25R200_REG_IRQ3 : **st25r200_com.h**
- ST25R200_REG_IRQ_MASK1 : **st25r200_com.h**
- ST25R200_REG_IRQ_MASK2 : **st25r200_com.h**
- ST25R200_REG_IRQ_MASK3 : **st25r200_com.h**
- ST25R200_REG_LEN : **st25r200.h**
- ST25R200_REG_MRT : **st25r200_com.h**
- ST25R200_REG_MRT_SQT_CONF : **st25r200_com.h**
- ST25R200_REG_NRT1 : **st25r200_com.h**
- ST25R200_REG_NRT2 : **st25r200_com.h**
- ST25R200_REG_NRT_GPT_CONF : **st25r200_com.h**
- ST25R200_REG_OPERATION : **st25r200_com.h**
- ST25R200_REG_OVERSHOOT_CONF : **st25r200_com.h**
- ST25R200_REG_OVUNSHOOT_CONF : **st25r200_com.h**
- ST25R200_REG_PROTOCOL : **st25r200_com.h**
- ST25R200_REG_PROTOCOL_RX1 : **st25r200_com.h**
- ST25R200_REG_PROTOCOL_RX2 : **st25r200_com.h**
- ST25R200_REG_PROTOCOL_RX3 : **st25r200_com.h**
- ST25R200_REG_PROTOCOL_TX1 : **st25r200_com.h**
- ST25R200_REG_PROTOCOL_TX2 : **st25r200_com.h**
- ST25R200_REG_PROTOCOL_TX3 : **st25r200_com.h**
- ST25R200_REG_REGULATOR : **st25r200_com.h**
- ST25R200_REG_RX_ANA1 : **st25r200_com.h**
- ST25R200_REG_RX_ANA2 : **st25r200_com.h**
- ST25R200_REG_RX_DIG : **st25r200_com.h**
- ST25R200_REG_SQT : **st25r200_com.h**
- ST25R200_REG_STATUS : **st25r200_com.h**
- ST25R200_REG_TX_DRIVER : **st25r200_com.h**
- ST25R200_REG_TX_FRAME1 : **st25r200_com.h**
- ST25R200_REG_TX_FRAME2 : **st25r200_com.h**
- ST25R200_REG_TX_MOD : **st25r200_com.h**
- ST25R200_REG_TX_RES_MOD : **st25r200_com.h**
- ST25R200_REG_UNDERSHOOT_CONF : **st25r200_com.h**
- ST25R200_REG_WAKEUP_CONF1 : **st25r200_com.h**
- ST25R200_REG_WAKEUP_CONF2 : **st25r200_com.h**
- ST25R200_REG_WU_I_ADC : **st25r200_com.h**
- ST25R200_REG_WU_I_CAL : **st25r200_com.h**
- ST25R200_REG_WU_I_CONF : **st25r200_com.h**
- ST25R200_REG_WU_I_DELTA : **st25r200_com.h**

- ST25R200_REG_WU_I_REF : **st25r200_com.h**
- ST25R200_REG_WU_Q_ADC : **st25r200_com.h**
- ST25R200_REG_WU_Q_CAL : **st25r200_com.h**
- ST25R200_REG_WU_Q_CONF : **st25r200_com.h**
- ST25R200_REG_WU_Q_DELTA : **st25r200_com.h**
- ST25R200_REG_WU_Q_REF : **st25r200_com.h**
- ST25R200_WRITE_MODE : **st25r200.h**
- st25r200IsGPTRunning : **st25r200.h**
- st25r200IsLastFIFOComplete : **st25r200.h**
- st25r200IsNRTinEMV : **st25r200.h**
- st25r200IsOscOn : **st25r200.h**
- st25r200IsTxEnabled : **st25r200.h**
- st25r200IsTxOn : **st25r200.h**
- st25r200TxOff : **st25r200.h**
- st25r200TxRxOff : **st25r200.h**
- st25r200TxRxOn : **st25r200.h**
- ST25R3911_BR_106 : **st25r3911.h**
- ST25R3911_BR_1695 : **st25r3911.h**
- ST25R3911_BR_212 : **st25r3911.h**
- ST25R3911_BR_3390 : **st25r3911.h**
- ST25R3911_BR_424 : **st25r3911.h**
- ST25R3911_BR_6780 : **st25r3911.h**
- ST25R3911_BR_848 : **st25r3911.h**
- ST25R3911_BR_DO_NOT_SET : **st25r3911.h**
- ST25R3911_CMD_ADJUST_REGULATORS : **st25r3911.h**
- ST25R3911_CMD_ANALOG_PRESET : **st25r3911.h**
- ST25R3911_CMD_CALIBRATE_ANTENNA : **st25r3911.h**
- ST25R3911_CMD_CALIBRATE_C_SENSOR : **st25r3911.h**
- ST25R3911_CMD_CALIBRATE_MODULATION : **st25r3911.h**
- ST25R3911_CMD_CLEAR_FIFO : **st25r3911.h**
- ST25R3911_CMD_CLEAR_RSSI : **st25r3911.h**
- ST25R3911_CMD_CLEAR_SQUELCH : **st25r3911.h**
- ST25R3911_CMD_FUSE_PPROM : **st25r3911.h**
- ST25R3911_CMD_INITIAL_RF_COLLISION : **st25r3911.h**
- ST25R3911_CMD_LOAD_PPROM : **st25r3911.h**
- ST25R3911_CMD_MASK_RECEIVE_DATA : **st25r3911.h**
- ST25R3911_CMD_MEASURE_AMPLITUDE : **st25r3911.h**
- ST25R3911_CMD_MEASURE_CAPACITANCE : **st25r3911.h**
- ST25R3911_CMD_MEASURE_PHASE : **st25r3911.h**

- ST25R3911_CMD_MEASURE_VDD : **st25r3911.h**
- ST25R3911_CMD_NORMAL_NFC_MODE : **st25r3911.h**
- ST25R3911_CMD_RESPONSE_RF_COLLISION_0 : **st25r3911.h**
- ST25R3911_CMD_RESPONSE_RF_COLLISION_N : **st25r3911.h**
- ST25R3911_CMD_SET_DEFAULT : **st25r3911.h**
- ST25R3911_CMD_SQUELCH : **st25r3911.h**
- ST25R3911_CMD_START_GP_TIMER : **st25r3911.h**
- ST25R3911_CMD_START_MASK_RECEIVE_TIMER : **st25r3911.h**
- ST25R3911_CMD_START_NO_RESPONSE_TIMER : **st25r3911.h**
- ST25R3911_CMD_START_WUP_TIMER : **st25r3911.h**
- ST25R3911_CMD_TEST_ACCESS : **st25r3911.h**
- ST25R3911_CMD_TEST_CLEARA : **st25r3911.h**
- ST25R3911_CMD_TEST_CLEARB : **st25r3911.h**
- ST25R3911_CMD_TRANSMIT_REQA : **st25r3911.h**
- ST25R3911_CMD_TRANSMIT_WITH_CRC : **st25r3911.h**
- ST25R3911_CMD_TRANSMIT_WITHOUT_CRC : **st25r3911.h**
- ST25R3911_CMD_TRANSMIT_WUPA : **st25r3911.h**
- ST25R3911_CMD_TRANSPARENT_MODE : **st25r3911.h**
- ST25R3911_CMD_UNMASK_RECEIVE_DATA : **st25r3911.h**
- ST25R3911_FDT_NONE : **st25r3911.h**
- ST25R3911_FIFO_DEPTH : **st25r3911.h**
- ST25R3911_FIFO_STATUS_LEN : **st25r3911_com.h**
- ST25R3911_IRQ_MASK_ALL : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_CAC : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_CAT : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_COL : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_CRC : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_DCT : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_EOF : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_EON : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_ERR : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_ERR1 : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_ERR2 : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_FWL : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_GPE : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_NFCT : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_NONE : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_NRE : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_OSC : **st25r3911_interrupt.h**

- ST25R3911_IRQ_MASK_PAR : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_RXE : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_RXS : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_TIM : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_TXE : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_WAM : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_WCAP : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_WPH : **st25r3911_interrupt.h**
- ST25R3911_IRQ_MASK_WT : **st25r3911_interrupt.h**
- ST25R3911_REG_AD_RESULT : **st25r3911_com.h**
- ST25R3911_REG_AM_MOD_DEPTH_CONTROL : **st25r3911_com.h**
- ST25R3911_REG_AM_MOD_DEPTH_RESULT : **st25r3911_com.h**
- ST25R3911_REG_AMPLITUDE_MEASURE_AA_RESULT : **st25r3911_com.h**
- ST25R3911_REG_AMPLITUDE_MEASURE_CONF : **st25r3911_com.h**
- ST25R3911_REG_AMPLITUDE_MEASURE_REF : **st25r3911_com.h**
- ST25R3911_REG_AMPLITUDE_MEASURE_RESULT : **st25r3911_com.h**
- ST25R3911_REG_ANT_CAL_CONTROL : **st25r3911_com.h**
- ST25R3911_REG_ANT_CAL_RESULT : **st25r3911_com.h**
- ST25R3911_REG_ANT_CAL_TARGET : **st25r3911_com.h**
- ST25R3911_REG_AUX : **st25r3911_com.h**
- ST25R3911_REG_AUX_DISPLAY : **st25r3911_com.h**
- ST25R3911_REG_BIT_RATE : **st25r3911_com.h**
- ST25R3911_REG_CAP_SENSOR_CONTROL : **st25r3911_com.h**
- ST25R3911_REG_CAP_SENSOR_RESULT : **st25r3911_com.h**
- ST25R3911_REG_CAPACITANCE_MEASURE_AA_RESULT : **st25r3911_com.h**
- ST25R3911_REG_CAPACITANCE_MEASURE_CONF : **st25r3911_com.h**
- ST25R3911_REG_CAPACITANCE_MEASURE_REF : **st25r3911_com.h**
- ST25R3911_REG_CAPACITANCE_MEASURE_RESULT : **st25r3911_com.h**
- ST25R3911_REG_COLLISION_STATUS : **st25r3911_com.h**
- ST25R3911_REG_FIELD_THRESHOLD : **st25r3911_com.h**
- ST25R3911_REG_FIFO_RX_STATUS1 : **st25r3911_com.h**
- ST25R3911_REG_FIFO_RX_STATUS2 : **st25r3911_com.h**
- ST25R3911_REG_GAIN_RED_STATE : **st25r3911_com.h**
- ST25R3911_REG_GPT1 : **st25r3911_com.h**
- ST25R3911_REG_GPT2 : **st25r3911_com.h**
- ST25R3911_REG_GPT_CONTROL : **st25r3911_com.h**
- ST25R3911_REG_IC_IDENTITY : **st25r3911_com.h**
- ST25R3911_REG_IO_CONF1 : **st25r3911_com.h**
- ST25R3911_REG_IO_CONF2 : **st25r3911_com.h**

- ST25R3911_REG_IRQ_ERROR_WUP : **st25r3911_com.h**
- ST25R3911_REG_IRQ_MAIN : **st25r3911_com.h**
- ST25R3911_REG_IRQ_MASK_ERROR_WUP : **st25r3911_com.h**
- ST25R3911_REG_IRQ_MASK_MAIN : **st25r3911_com.h**
- ST25R3911_REG_IRQ_MASK_TIMER_NFC : **st25r3911_com.h**
- ST25R3911_REG_IRQ_TIMER_NFC : **st25r3911_com.h**
- ST25R3911_REG_ISO14443A_NFC : **st25r3911_com.h**
- ST25R3911_REG_ISO14443B_1 : **st25r3911_com.h**
- ST25R3911_REG_ISO14443B_2 : **st25r3911_com.h**
- ST25R3911_REG_MASK_RX_TIMER : **st25r3911_com.h**
- ST25R3911_REG_MODE : **st25r3911_com.h**
- ST25R3911_REG_NFCIP1_BIT_RATE : **st25r3911_com.h**
- ST25R3911_REG_NO_RESPONSE_TIMER1 : **st25r3911_com.h**
- ST25R3911_REG_NO_RESPONSE_TIMER2 : **st25r3911_com.h**
- ST25R3911_REG_NUM_TX_BYTES1 : **st25r3911_com.h**
- ST25R3911_REG_NUM_TX_BYTES2 : **st25r3911_com.h**
- ST25R3911_REG_OP_CONTROL : **st25r3911_com.h**
- ST25R3911_REG_PHASE_MEASURE_AA_RESULT : **st25r3911_com.h**
- ST25R3911_REG_PHASE_MEASURE_CONF : **st25r3911_com.h**
- ST25R3911_REG_PHASE_MEASURE_REF : **st25r3911_com.h**
- ST25R3911_REG_PHASE_MEASURE_RESULT : **st25r3911_com.h**
- ST25R3911_REG_REGULATOR_CONTROL : **st25r3911_com.h**
- ST25R3911_REG_REGULATOR_RESULT : **st25r3911_com.h**
- ST25R3911_REG_RFO_AM_OFF_LEVEL : **st25r3911_com.h**
- ST25R3911_REG_RFO_AM_ON_LEVEL : **st25r3911_com.h**
- ST25R3911_REG_RSSI_RESULT : **st25r3911_com.h**
- ST25R3911_REG_RX_CONF1 : **st25r3911_com.h**
- ST25R3911_REG_RX_CONF2 : **st25r3911_com.h**
- ST25R3911_REG_RX_CONF3 : **st25r3911_com.h**
- ST25R3911_REG_RX_CONF4 : **st25r3911_com.h**
- ST25R3911_REG_STREAM_MODE : **st25r3911_com.h**
- ST25R3911_REG_WUP_TIMER_CONTROL : **st25r3911_com.h**
- ST25R3911_THRESHOLD_DO_NOT_SET : **st25r3911.h**
- st25r3911IsCRCinFIFO : **st25r3911.h**
- st25r3911IsExtFieldOn : **st25r3911.h**
- st25r3911IsGPTRunning : **st25r3911.h**
- st25r3911IsLastFIFOComplete : **st25r3911.h**
- st25r3911IsOscOn : **st25r3911.h**
- st25r3911IsTxEnabled : **st25r3911.h**

- st25r3911TxOff : **st25r3911.h**
- ST25R3916_BR_106 : **st25r3916.h**
- ST25R3916_BR_1695 : **st25r3916.h**
- ST25R3916_BR_212 : **st25r3916.h**
- ST25R3916_BR_3390 : **st25r3916.h**
- ST25R3916_BR_424 : **st25r3916.h**
- ST25R3916_BR_6780 : **st25r3916.h**
- ST25R3916_BR_848 : **st25r3916.h**
- ST25R3916_BR_DO_NOT_SET : **st25r3916.h**
- ST25R3916_CMD_ADJUST_REGULATORS : **st25r3916.h**
- ST25R3916_CMD_AM_MOD_STATE_CHANGE : **st25r3916.h**
- ST25R3916_CMD_CALIBRATE_DRIVER_TIMING : **st25r3916.h**
- ST25R3916_CMD_CLEAR_FIFO : **st25r3916.h**
- ST25R3916_CMD_CLEAR_RSSI : **st25r3916.h**
- ST25R3916_CMD_GOTO_SENSE : **st25r3916.h**
- ST25R3916_CMD_GOTO_SLEEP : **st25r3916.h**
- ST25R3916_CMD_INITIAL_RF_COLLISION : **st25r3916.h**
- ST25R3916_CMD_MASK_RECEIVE_DATA : **st25r3916.h**
- ST25R3916_CMD_MEASURE_AMPLITUDE : **st25r3916.h**
- ST25R3916_CMD_MEASURE_PHASE : **st25r3916.h**
- ST25R3916_CMD_MEASURE_VDD : **st25r3916.h**
- ST25R3916_CMD_RESET_RXGAIN : **st25r3916.h**
- ST25R3916_CMD_RESPONSE_RF_COLLISION_N : **st25r3916.h**
- ST25R3916_CMD_SET_DEFAULT : **st25r3916.h**
- ST25R3916_CMD_SPACE_B_ACCESS : **st25r3916.h**
- ST25R3916_CMD_START_GP_TIMER : **st25r3916.h**
- ST25R3916_CMD_START_MASK_RECEIVE_TIMER : **st25r3916.h**
- ST25R3916_CMD_START_NO_RESPONSE_TIMER : **st25r3916.h**
- ST25R3916_CMD_START_PPON2_TIMER : **st25r3916.h**
- ST25R3916_CMD_START_WUP_TIMER : **st25r3916.h**
- ST25R3916_CMD_STOP : **st25r3916.h**
- ST25R3916_CMD_STOP_NRT : **st25r3916.h**
- ST25R3916_CMD_TEST_ACCESS : **st25r3916.h**
- ST25R3916_CMD_TRANSMIT_REQA : **st25r3916.h**
- ST25R3916_CMD_TRANSMIT_WITH_CRC : **st25r3916.h**
- ST25R3916_CMD_TRANSMIT_WITHOUT_CRC : **st25r3916.h**
- ST25R3916_CMD_TRANSMIT_WUPA : **st25r3916.h**
- ST25R3916_CMD_TRANSPARENT_MODE : **st25r3916.h**
- ST25R3916_CMD_UNMASK_RECEIVE_DATA : **st25r3916.h**

- ST25R3916_FIFO_DEPTH : **st25r3916.h**
- ST25R3916_FIFO_STATUS_LEN : **st25r3916_com.h**
- ST25R3916_IRQ_MASK_ALL : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_APON : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_CAC : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_CAT : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_COL : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_CRC : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_DCT : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_EOF : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_EON : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_ERR1 : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_ERR2 : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_FWL : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_GPE : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_NFCT : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_NONE : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_NRE : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_OSC : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_PAR : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_PPON2 : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_RFU : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_RFU2 : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_RX_REST : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_RXE : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_RXE_PTA : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_RXS : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_SL_WL : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_TXE : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_WAM : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_WPH : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_WT : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_WU_A : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_WU_A_X : **st25r3916_irq.h**
- ST25R3916_IRQ_MASK_WU_F : **st25r3916_irq.h**
- ST25R3916_PTM_A_LEN : **st25r3916_com.h**
- ST25R3916_PTM_B_LEN : **st25r3916_com.h**
- ST25R3916_PTM_F_LEN : **st25r3916_com.h**
- ST25R3916_PTM_LEN : **st25r3916_com.h**

- ST25R3916_PTM_TSN_LEN : **st25r3916_com.h**
- ST25R3916_REG_AD_RESULT : **st25r3916_com.h**
- ST25R3916_REG_AMPLITUDE_MEASURE_AA_RESULT : **st25r3916_com.h**
- ST25R3916_REG_AMPLITUDE_MEASURE_CONF : **st25r3916_com.h**
- ST25R3916_REG_AMPLITUDE_MEASURE_REF : **st25r3916_com.h**
- ST25R3916_REG_AMPLITUDE_MEASURE_RESULT : **st25r3916_com.h**
- ST25R3916_REG_ANT_TUNE_A : **st25r3916_com.h**
- ST25R3916_REG_ANT_TUNE_B : **st25r3916_com.h**
- ST25R3916_REG_AUX : **st25r3916_com.h**
- ST25R3916_REG_AUX_DISPLAY : **st25r3916_com.h**
- ST25R3916_REG_AUX_MOD : **st25r3916_com.h**
- ST25R3916_REG_BIT_RATE : **st25r3916_com.h**
- ST25R3916_REG_CAP_SENSOR_CONTROL : **st25r3916_com.h**
- ST25R3916_REG_CAP_SENSOR_RESULT : **st25r3916_com.h**
- ST25R3916_REG_COLLISION_STATUS : **st25r3916_com.h**
- ST25R3916_REG_CORR_CONF1 : **st25r3916_com.h**
- ST25R3916_REG_CORR_CONF2 : **st25r3916_com.h**
- ST25R3916_REG_EMD_SUP_CONF : **st25r3916_com.h**
- ST25R3916_REG_FIELD_ON_GT : **st25r3916_com.h**
- ST25R3916_REG_FIELD_THRESHOLD_ACTV : **st25r3916_com.h**
- ST25R3916_REG_FIELD_THRESHOLD_DEACTV : **st25r3916_com.h**
- ST25R3916_REG_FIFO_STATUS1 : **st25r3916_com.h**
- ST25R3916_REG_FIFO_STATUS2 : **st25r3916_com.h**
- ST25R3916_REG_GAIN_RED_STATE : **st25r3916_com.h**
- ST25R3916_REG_GPT1 : **st25r3916_com.h**
- ST25R3916_REG_GPT2 : **st25r3916_com.h**
- ST25R3916_REG_IC_IDENTITY : **st25r3916_com.h**
- ST25R3916_REG_IO_CONF1 : **st25r3916_com.h**
- ST25R3916_REG_IO_CONF2 : **st25r3916_com.h**
- ST25R3916_REG_IRQ_ERROR_WUP : **st25r3916_com.h**
- ST25R3916_REG_IRQ_MAIN : **st25r3916_com.h**
- ST25R3916_REG_IRQ_MASK_ERROR_WUP : **st25r3916_com.h**
- ST25R3916_REG_IRQ_MASK_MAIN : **st25r3916_com.h**
- ST25R3916_REG_IRQ_MASK_TARGET : **st25r3916_com.h**
- ST25R3916_REG_IRQ_MASK_TIMER_NFC : **st25r3916_com.h**
- ST25R3916_REG_IRQ_TARGET : **st25r3916_com.h**
- ST25R3916_REG_IRQ_TIMER_NFC : **st25r3916_com.h**
- ST25R3916_REG_ISO14443A_NFC : **st25r3916_com.h**
- ST25R3916_REG_ISO14443B_1 : **st25r3916_com.h**

- ST25R3916_REG_ISO14443B_2 : **st25r3916_com.h**
- ST25R3916_REG_MASK_RX_TIMER : **st25r3916_com.h**
- ST25R3916_REG_MODE : **st25r3916_com.h**
- ST25R3916_REG_NFCIP1_BIT_RATE : **st25r3916_com.h**
- ST25R3916_REG_NO_RESPONSE_TIMER1 : **st25r3916_com.h**
- ST25R3916_REG_NO_RESPONSE_TIMER2 : **st25r3916_com.h**
- ST25R3916_REG_NUM_TX_BYTES1 : **st25r3916_com.h**
- ST25R3916_REG_NUM_TX_BYTES2 : **st25r3916_com.h**
- ST25R3916_REG_OP_CONTROL : **st25r3916_com.h**
- ST25R3916_REG_OVERSHOOT_CONF1 : **st25r3916_com.h**
- ST25R3916_REG_OVERSHOOT_CONF2 : **st25r3916_com.h**
- ST25R3916_REG_P2P_RX_CONF : **st25r3916_com.h**
- ST25R3916_REG_PASSIVE_TARGET : **st25r3916_com.h**
- ST25R3916_REG_PASSIVE_TARGET_STATUS : **st25r3916_com.h**
- ST25R3916_REG_PHASE_MEASURE_AA_RESULT : **st25r3916_com.h**
- ST25R3916_REG_PHASE_MEASURE_CONF : **st25r3916_com.h**
- ST25R3916_REG_PHASE_MEASURE_REF : **st25r3916_com.h**
- ST25R3916_REG_PHASE_MEASURE_RESULT : **st25r3916_com.h**
- ST25R3916_REG_PPON2 : **st25r3916_com.h**
- ST25R3916_REG_PT_MOD : **st25r3916_com.h**
- ST25R3916_REG_REGULATOR_CONTROL : **st25r3916_com.h**
- ST25R3916_REG_REGULATOR_RESULT : **st25r3916_com.h**
- ST25R3916_REG_RES_AM_MOD : **st25r3916_com.h**
- ST25R3916_REG_RSSI_RESULT : **st25r3916_com.h**
- ST25R3916_REG_RX_CONF1 : **st25r3916_com.h**
- ST25R3916_REG_RX_CONF2 : **st25r3916_com.h**
- ST25R3916_REG_RX_CONF3 : **st25r3916_com.h**
- ST25R3916_REG_RX_CONF4 : **st25r3916_com.h**
- ST25R3916_REG_SQUELCH_TIMER : **st25r3916_com.h**
- ST25R3916_REG_STREAM_MODE : **st25r3916_com.h**
- ST25R3916_REG_SUBC_START_TIME : **st25r3916_com.h**
- ST25R3916_REG_TIMER_EMV_CONTROL : **st25r3916_com.h**
- ST25R3916_REG_TX_DRIVER : **st25r3916_com.h**
- ST25R3916_REG_TX_DRIVER_STATUS : **st25r3916_com.h**
- ST25R3916_REG_TX_DRIVER_TIMING : **st25r3916_com.h**
- ST25R3916_REG_UNDERSHOOT_CONF1 : **st25r3916_com.h**
- ST25R3916_REG_UNDERSHOOT_CONF2 : **st25r3916_com.h**
- ST25R3916_REG_WUP_TIMER_CONTROL : **st25r3916_com.h**
- ST25R3916_SPACE_B : **st25r3916_com.h**

- ST25R3916_SPACE_B_REG_LEN : **st25r3916_com.h**
- ST25R3916_THRESHOLD_DO_NOT_SET : **st25r3916.h**
- ST25R3916_TOUT_OSC_STABLE : **st25r3916.h**
- st25r3916IsAATOn : **st25r3916.h**
- st25r3916IsExtFieldOn : **st25r3916.h**
- st25r3916IsGPTRunning : **st25r3916.h**
- st25r3916IsLastFIFOComplete : **st25r3916.h**
- st25r3916IsNRTinEMV : **st25r3916.h**
- st25r3916IsOscOn : **st25r3916.h**
- st25r3916IsTxEnabled : **st25r3916.h**
- st25r3916TxOff : **st25r3916.h**
- st25r3916TxRxOff : **st25r3916.h**
- st25r3916TxRxOn : **st25r3916.h**
- ST25R500_BR_106_26 : **st25r500.h**
- ST25R500_BR_212_53 : **st25r500.h**
- ST25R500_BR_424 : **st25r500.h**
- ST25R500_BR_848 : **st25r500.h**
- ST25R500_BR_DO_NOT_SET : **st25r500.h**
- ST25R500_CEM_A_LEN : **st25r500_com.h**
- ST25R500_CEM_B_LEN : **st25r500_com.h**
- ST25R500_CEM_F_LEN : **st25r500_com.h**
- ST25R500_CEM_LEN : **st25r500_com.h**
- ST25R500_CMD_ADJUST_REGULATORS : **st25r500.h**
- ST25R500_CMD_CALIBRATE_RC : **st25r500.h**
- ST25R500_CMD_CALIBRATE_WU : **st25r500.h**
- ST25R500_CMD_CLEAR_FIFO : **st25r500.h**
- ST25R500_CMD_CLEAR_RXGAIN : **st25r500.h**
- ST25R500_CMD_CLEAR_WU_CALIB : **st25r500.h**
- ST25R500_CMD_LEN : **st25r500.h**
- ST25R500_CMD_MASK_RECEIVE_DATA : **st25r500.h**
- ST25R500_CMD_MEASURE_IQ : **st25r500.h**
- ST25R500_CMD_MEASURE_WU : **st25r500.h**
- ST25R500_CMD_MODE : **st25r500.h**
- ST25R500_CMD_NFC_FIELD_ON : **st25r500.h**
- ST25R500_CMD_SENSE_RF : **st25r500.h**
- ST25R500_CMD_SET_DEFAULT : **st25r500.h**
- ST25R500_CMD_START_GP_TIMER : **st25r500.h**
- ST25R500_CMD_START_MRT : **st25r500.h**
- ST25R500_CMD_START_NRT : **st25r500.h**

- ST25R500_CMD_START_WUT : **st25r500.h**
- ST25R500_CMD_STOP : **st25r500.h**
- ST25R500_CMD_STOP_NRT : **st25r500.h**
- ST25R500_CMD_TEST_ACCESS : **st25r500.h**
- ST25R500_CMD_TRANSMIT : **st25r500.h**
- ST25R500_CMD_TRANSMIT_EOF : **st25r500.h**
- ST25R500_CMD_TRIGGER_DIAG : **st25r500.h**
- ST25R500_CMD_TRIGGER_WU_EV : **st25r500.h**
- ST25R500_CMD_UNMASK_RECEIVE_DATA : **st25r500.h**
- ST25R500_DIAG_MEAS_CMD : **st25r500.h**
- ST25R500_DIAG_MEAS_CMD_LEN : **st25r500.h**
- ST25R500_DIAG_MEAS_I_VDD_DR : **st25r500.h**
- ST25R500_DIAG_MEAS_RES_LEN : **st25r500.h**
- ST25R500_DIAG_MEAS_VDD_A : **st25r500.h**
- ST25R500_DIAG_MEAS_VDD_AGD : **st25r500.h**
- ST25R500_DIAG_MEAS_VDD_D : **st25r500.h**
- ST25R500_DIAG_MEAS_VDD_DR : **st25r500.h**
- ST25R500_DIAG_MEAS_VDD_IO : **st25r500.h**
- ST25R500_DIAG_MEAS_VDD_TX : **st25r500.h**
- ST25R500_DIAG_MEAS_VDD_VDD : **st25r500.h**
- ST25R500_DPOCR_MAX_ENTRIES : **st25r500_dpocr.h**
- ST25R500_FIFO_ACCESS : **st25r500.h**
- ST25R500_FIFO_DEPTH : **st25r500.h**
- ST25R500_FIFO_STATUS_LEN : **st25r500_com.h**
- ST25R500_IRQ_MASK_ALL : **st25r500_irq.h**
- ST25R500_IRQ_MASK_CE_SC : **st25r500_irq.h**
- ST25R500_IRQ_MASK_COL : **st25r500_irq.h**
- ST25R500_IRQ_MASK_DCT : **st25r500_irq.h**
- ST25R500_IRQ_MASK_EOF : **st25r500_irq.h**
- ST25R500_IRQ_MASK_EON : **st25r500_irq.h**
- ST25R500_IRQ_MASK_GPE : **st25r500_irq.h**
- ST25R500_IRQ_MASK_NFCT : **st25r500_irq.h**
- ST25R500_IRQ_MASK_NONE : **st25r500_irq.h**
- ST25R500_IRQ_MASK_NRE : **st25r500_irq.h**
- ST25R500_IRQ_MASK_OSC : **st25r500_irq.h**
- ST25R500_IRQ_MASK_RFU : **st25r500_irq.h**
- ST25R500_IRQ_MASK_RX_ERR : **st25r500_irq.h**
- ST25R500_IRQ_MASK_RX_REST : **st25r500_irq.h**
- ST25R500_IRQ_MASK_RXE : **st25r500_irq.h**

- ST25R500_IRQ_MASK_RXE_CE : **st25r500_irq.h**
- ST25R500_IRQ_MASK_RXS : **st25r500_irq.h**
- ST25R500_IRQ_MASK_SUBC_START : **st25r500_irq.h**
- ST25R500_IRQ_MASK_TXE : **st25r500_irq.h**
- ST25R500_IRQ_MASK_WL : **st25r500_irq.h**
- ST25R500_IRQ_MASK_WPT_FOD : **st25r500_irq.h**
- ST25R500_IRQ_MASK_WPT_STOP : **st25r500_irq.h**
- ST25R500_IRQ_MASK_WUI : **st25r500_irq.h**
- ST25R500_IRQ_MASK_WUQ : **st25r500_irq.h**
- ST25R500_IRQ_MASK_WUT : **st25r500_irq.h**
- ST25R500_IRQ_MASK_WUTME : **st25r500_irq.h**
- ST25R500_OBS_MODE_LEN : **st25r500_com.h**
- ST25R500_READ_MODE : **st25r500.h**
- ST25R500_REG_ANA_DISPLAY1 : **st25r500_com.h**
- ST25R500_REG_ANA_DISPLAY2 : **st25r500_com.h**
- ST25R500_REG_AWS_CONFIG1 : **st25r500_com.h**
- ST25R500_REG_AWS_CONFIG2 : **st25r500_com.h**
- ST25R500_REG_AWS_TIME1 : **st25r500_com.h**
- ST25R500_REG_AWS_TIME2 : **st25r500_com.h**
- ST25R500_REG_AWS_TIME3 : **st25r500_com.h**
- ST25R500_REG_AWS_TIME4 : **st25r500_com.h**
- ST25R500_REG_CE_CONFIG1 : **st25r500_com.h**
- ST25R500_REG_CE_CONFIG2 : **st25r500_com.h**
- ST25R500_REG_CE_CONFIG3 : **st25r500_com.h**
- ST25R500_REG_CE_STATUS1 : **st25r500_com.h**
- ST25R500_REG_CE_STATUS2 : **st25r500_com.h**
- ST25R500_REG_CE_TX_MOD1 : **st25r500_com.h**
- ST25R500_REG_CE_TX_MOD2 : **st25r500_com.h**
- ST25R500_REG_CEM_A : **st25r500_com.h**
- ST25R500_REG_CEM_F : **st25r500_com.h**
- ST25R500_REG_COLLISION : **st25r500_com.h**
- ST25R500_REG_CORR1 : **st25r500_com.h**
- ST25R500_REG_CORR2 : **st25r500_com.h**
- ST25R500_REG_CORR3 : **st25r500_com.h**
- ST25R500_REG_CORR4 : **st25r500_com.h**
- ST25R500_REG_CORR5 : **st25r500_com.h**
- ST25R500_REG_CORR6 : **st25r500_com.h**
- ST25R500_REG_DRIVER : **st25r500_com.h**
- ST25R500_REG_DROP_200 : **st25r500.h**

- ST25R500_REG_DROP_250 : **st25r500.h**
- ST25R500_REG_DROP_300 : **st25r500.h**
- ST25R500_REG_DROP_350 : **st25r500.h**
- ST25R500_REG_DROP_400 : **st25r500.h**
- ST25R500_REG_DROP_450 : **st25r500.h**
- ST25R500_REG_DROP_500 : **st25r500.h**
- ST25R500_REG_DROP_550 : **st25r500.h**
- ST25R500_REG_DROP_DO_NOT_SET : **st25r500.h**
- ST25R500_REG_EFD_THRESHOLD : **st25r500_com.h**
- ST25R500_REG_EMD1 : **st25r500_com.h**
- ST25R500_REG_EMD2 : **st25r500_com.h**
- ST25R500_REG_FIFO : **st25r500_com.h**
- ST25R500_REG_FIFO_STATUS1 : **st25r500_com.h**
- ST25R500_REG_FIFO_STATUS2 : **st25r500_com.h**
- ST25R500_REG_GENERAL : **st25r500_com.h**
- ST25R500_REG_GPIO : **st25r500_com.h**
- ST25R500_REG_GPT1 : **st25r500_com.h**
- ST25R500_REG_GPT2 : **st25r500_com.h**
- ST25R500_REG_IC_ID : **st25r500_com.h**
- ST25R500_REG_IRQ1 : **st25r500_com.h**
- ST25R500_REG_IRQ2 : **st25r500_com.h**
- ST25R500_REG_IRQ3 : **st25r500_com.h**
- ST25R500_REG_IRQ_MASK1 : **st25r500_com.h**
- ST25R500_REG_IRQ_MASK2 : **st25r500_com.h**
- ST25R500_REG_IRQ_MASK3 : **st25r500_com.h**
- ST25R500_REG_LEN : **st25r500.h**
- ST25R500_REG_MRT1 : **st25r500_com.h**
- ST25R500_REG_MRT2 : **st25r500_com.h**
- ST25R500_REG_NRT1 : **st25r500_com.h**
- ST25R500_REG_NRT2 : **st25r500_com.h**
- ST25R500_REG_NRT_GPT_CONF : **st25r500_com.h**
- ST25R500_REG_OPERATION : **st25r500_com.h**
- ST25R500_REG_OVERSHOOT_CONF : **st25r500_com.h**
- ST25R500_REG_PROTOCOL : **st25r500_com.h**
- ST25R500_REG_PROTOCOL_RX1 : **st25r500_com.h**
- ST25R500_REG_PROTOCOL_RX2 : **st25r500_com.h**
- ST25R500_REG_PROTOCOL_RX3 : **st25r500_com.h**
- ST25R500_REG_PROTOCOL_TX1 : **st25r500_com.h**
- ST25R500_REG_PROTOCOL_TX2 : **st25r500_com.h**

- ST25R500_REG_REGULATOR : **st25r500_com.h**
- ST25R500_REG_RSSI_I : **st25r500_com.h**
- ST25R500_REG_RSSI_Q : **st25r500_com.h**
- ST25R500_REG_RX_ANA1 : **st25r500_com.h**
- ST25R500_REG_RX_ANA2 : **st25r500_com.h**
- ST25R500_REG_RX_ANA3 : **st25r500_com.h**
- ST25R500_REG_RX_ANA4 : **st25r500_com.h**
- ST25R500_REG_RX_DIG : **st25r500_com.h**
- ST25R500_REG_SENSE_DISPLAY : **st25r500_com.h**
- ST25R500_REG_SQT : **st25r500_com.h**
- ST25R500_REG_STATUS1 : **st25r500_com.h**
- ST25R500_REG_STATUS2 : **st25r500_com.h**
- ST25R500_REG_STATUS_STATIC1 : **st25r500_com.h**
- ST25R500_REG_STATUS_STATIC2 : **st25r500_com.h**
- ST25R500_REG_STATUS_STATIC3 : **st25r500_com.h**
- ST25R500_REG_TX_FRAME1 : **st25r500_com.h**
- ST25R500_REG_TX_FRAME2 : **st25r500_com.h**
- ST25R500_REG_TX_MOD1 : **st25r500_com.h**
- ST25R500_REG_TX_MOD2 : **st25r500_com.h**
- ST25R500_REG_UNDERSHOOT_CONF : **st25r500_com.h**
- ST25R500_REG_WAKEUP_CONF1 : **st25r500_com.h**
- ST25R500_REG_WAKEUP_CONF2 : **st25r500_com.h**
- ST25R500_REG_WAKEUP_CONF3 : **st25r500_com.h**
- ST25R500_REG_WU_I_ADC : **st25r500_com.h**
- ST25R500_REG_WU_I_CAL : **st25r500_com.h**
- ST25R500_REG_WU_I_CONF : **st25r500_com.h**
- ST25R500_REG_WU_I_DELTA : **st25r500_com.h**
- ST25R500_REG_WU_I_REF : **st25r500_com.h**
- ST25R500_REG_WU_Q_ADC : **st25r500_com.h**
- ST25R500_REG_WU_Q_CAL : **st25r500_com.h**
- ST25R500_REG_WU_Q_CONF : **st25r500_com.h**
- ST25R500_REG_WU_Q_DELTA : **st25r500_com.h**
- ST25R500_REG_WU_Q_REF : **st25r500_com.h**
- ST25R500_REG_WU_STATUS : **st25r500_com.h**
- ST25R500_TEST_REG_ADC2CE : **st25r500_com.h**
- ST25R500_TEST_REG_DIAG_MEAS : **st25r500_com.h**
- ST25R500_TEST_REG_MAN_TIMING : **st25r500_com.h**
- ST25R500_TEST_REG_OSC_TIMING : **st25r500_com.h**
- ST25R500_TEST_REG_OVERLAP_CONTROL : **st25r500_com.h**

- ST25R500_TEST_REG_RXE_DISP1 : **st25r500_com.h**
- ST25R500_TEST_REG_RXE_DISP2 : **st25r500_com.h**
- ST25R500_TEST_REG_RXS_DISP1 : **st25r500_com.h**
- ST25R500_TEST_REG_RXS_DISP2 : **st25r500_com.h**
- ST25R500_THRESHOLD_DO_NOT_SET : **st25r500.h**
- ST25R500_TOUT_OSC_STABLE : **st25r500.h**
- ST25R500_WRITE_MODE : **st25r500.h**
- st25r500IsExtFieldOn : **st25r500.h**
- st25r500IsGPTRunning : **st25r500.h**
- st25r500IsLastFIFOComplete : **st25r500.h**
- st25r500IsNRTinEMV : **st25r500.h**
- st25r500IsOscOn : **st25r500.h**
- st25r500IsTxEnabled : **st25r500.h**
- st25r500IsTxOn : **st25r500.h**
- st25r500TxOff : **st25r500.h**
- st25r500TxRxOff : **st25r500.h**
- st25r500TxRxOn : **st25r500.h**
- st25r500VDDDROff : **st25r500.h**
- st25r500VDDDROn : **st25r500.h**
- ST25R95_ACFILTER_RESPONSE_BUFLEN : **st25r95_com.h**
- ST25R95_ACSTATE_ACTIVE : **st25r95_com.h**
- ST25R95_ACSTATE_ACTIVEX : **st25r95_com.h**
- ST25R95_ACSTATE_HALT : **st25r95_com.h**
- ST25R95_ACSTATE_IDLE : **st25r95_com.h**
- ST25R95_ACSTATE_READYA : **st25r95_com.h**
- ST25R95_ACSTATE_READYAX : **st25r95_com.h**
- ST25R95_CMD_COMMAND_OFFSET : **st25r95_com.h**
- ST25R95_CMD_DATA_OFFSET : **st25r95_com.h**
- ST25R95_CMD_LENGTH_OFFSET : **st25r95_com.h**
- ST25R95_CMD_RESULT_OFFSET : **st25r95_com.h**
- ST25R95_COMMAND_ACFILTER : **st25r95_com.h**
- ST25R95_COMMAND_BAUDRATE : **st25r95_com.h**
- ST25R95_COMMAND_ECHO : **st25r95_com.h**
- ST25R95_COMMAND_IDLE : **st25r95_com.h**
- ST25R95_COMMAND_IDN : **st25r95_com.h**
- ST25R95_COMMAND_LISTEN : **st25r95_com.h**
- ST25R95_COMMAND_POLLFIED : **st25r95_com.h**
- ST25R95_COMMAND_PROTOCOLSELECT : **st25r95_com.h**
- ST25R95_COMMAND_RDREG : **st25r95_com.h**

- ST25R95_COMMAND_SEND : **st25r95_com.h**
- ST25R95_COMMAND_SENDRECV : **st25r95_com.h**
- ST25R95_COMMAND_WRREG : **st25r95_com.h**
- ST25R95_COMMUNICATION_BUFFER_SIZE : **st25r95_com.h**
- ST25R95_COMMUNICATION_UART_WDOGTIMER : **st25r95_com.h**
- ST25R95_CONTROL_POLL : **st25r95_com.h**
- ST25R95_CONTROL_POLL_NO_TIMEOUT : **st25r95_com.h**
- ST25R95_CONTROL_POLL_TIMEOUT : **st25r95_com.h**
- ST25R95_CONTROL_READ : **st25r95_com.h**
- ST25R95_CONTROL_RESET : **st25r95_com.h**
- ST25R95_CONTROL_SEND : **st25r95_com.h**
- ST25R95_DACDATA_MAX : **st25r95_com.h**
- ST25R95_ECHO_RESPONSE_BUFLEN : **st25r95_com.h**
- ST25R95_ERRCODE_61_SOF : **st25r95_com.h**
- ST25R95_ERRCODE_62_CRC : **st25r95_com.h**
- ST25R95_ERRCODE_63_SOF_HIGH : **st25r95_com.h**
- ST25R95_ERRCODE_65_SOF_LOW : **st25r95_com.h**
- ST25R95_ERRCODE_66_EGT : **st25r95_com.h**
- ST25R95_ERRCODE_67_TR1TOOLONG : **st25r95_com.h**
- ST25R95_ERRCODE_68_TR1TOOSHORT : **st25r95_com.h**
- ST25R95_ERRCODE_COMERROR : **st25r95_com.h**
- ST25R95_ERRCODE_CRC : **st25r95_com.h**
- ST25R95_ERRCODE_EGT : **st25r95_com.h**
- ST25R95_ERRCODE_FIELDLENGTH : **st25r95_com.h**
- ST25R95_ERRCODE_FRAMEOKADDITIONALINFO : **st25r95_com.h**
- ST25R95_ERRCODE_FRAMEWAITTIMEOUT : **st25r95_com.h**
- ST25R95_ERRCODE_FRAMING : **st25r95_com.h**
- ST25R95_ERRCODE_INVALIDCCMDCODE : **st25r95_com.h**
- ST25R95_ERRCODE_INVALIDCMDLENGHT : **st25r95_com.h**
- ST25R95_ERRCODE_INVALIDPROTOCOL : **st25r95_com.h**
- ST25R95_ERRCODE_INVALIDSOF : **st25r95_com.h**
- ST25R95_ERRCODE_NOFIELD : **st25r95_com.h**
- ST25R95_ERRCODE_NONE : **st25r95_com.h**
- ST25R95_ERRCODE_OVERFLOW : **st25r95_com.h**
- ST25R95_ERRCODE_RECEPTIONLOST : **st25r95_com.h**
- ST25R95_ERRCODE_RESULTSRESIDUAL : **st25r95_com.h**
- ST25R95_FWT_MAX : **st25r95_com.h**
- ST25R95_IDLE_DACDATAH_OFFSET : **st25r95_com.h**
- ST25R95_IDLE_DACDATAL_OFFSET : **st25r95_com.h**

- ST25R95_IDLE_RESPONSE_BUFLEN : **st25r95_com.h**
- ST25R95_IDLE_WKUP_TAGDETECT : **st25r95_com.h**
- ST25R95_IDLE_WKUP_TIMEOUT : **st25r95_com.h**
- ST25R95_IDLE_WUPERIOD_OFFSET : **st25r95_com.h**
- ST25R95_IDN_RESPONSE_BUFLEN : **st25r95_com.h**
- ST25R95_IS_PROT_ISO14443A_COLLISION_ERR : **st25r95_com.h**
- ST25R95_IS_PROT_ISO14443A_CRC_ERR : **st25r95_com.h**
- ST25R95_IS_PROT_ISO14443A_PARITY_ERR : **st25r95_com.h**
- ST25R95_IS_PROT_ISO14443B_CRC_ERR : **st25r95_com.h**
- ST25R95_IS_PROT_ISO15693_COLLISION_ERR : **st25r95_com.h**
- ST25R95_IS_PROT_ISO15693_CRC_ERR : **st25r95_com.h**
- ST25R95_IS_PROT_ISO18092_CRC_ERR : **st25r95_com.h**
- ST25R95_LISTEN_RESPONSE_BUFLEN : **st25r95_com.h**
- st25r95_nIRQ_IN_Pulse : **st25r95.h**
- ST25R95_POLLFIELD_RESPONSE_BUFLEN : **st25r95_com.h**
- ST25R95_PROTOCOL_CE_ISO14443A : **st25r95_com.h**
- ST25R95_PROTOCOL_FIELDOFF : **st25r95_com.h**
- ST25R95_PROTOCOL_ISO14443A : **st25r95_com.h**
- ST25R95_PROTOCOL_ISO14443B : **st25r95_com.h**
- ST25R95_PROTOCOL_ISO15693 : **st25r95_com.h**
- ST25R95_PROTOCOL_ISO18092 : **st25r95_com.h**
- ST25R95_PROTOCOL_MAX : **st25r95_com.h**
- ST25R95_PROTOCOLSELECT_BR_OFFSET : **st25r95_com.h**
- ST25R95_PROTOCOLSELECT_RESPONSE_BUFLEN : **st25r95_com.h**
- ST25R95_RDREG_RESPONSE_BUFLEN : **st25r95_com.h**
- ST25R95_REG_ACC_A : **st25r95_com.h**
- ST25R95_REG_ARC_B : **st25r95_com.h**
- ST25R95_REG_TIMERW : **st25r95_com.h**
- ST25R95_SEND_RESPONSE_BUFLEN : **st25r95_com.h**
- ST25R95_SPI_DUMMY_BYTE : **st25r95_com.h**
- ST25R95_WRREG_RESPONSE_BUFLEN : **st25r95_com.h**
- st25r95CommandEcho : **st25r95_com.h**
- st25r95CompleteRx : **st25r95_com.h**
- st25r95DeactivateACFilter : **st25r95_com.h**
- st25r95FlushChipSPIBuffer : **st25r95_com.h**
- st25r95GetIdleResponse : **st25r95_com.h**
- st25r95GetLmState : **st25r95_com.h**
- st25r95Idle : **st25r95_com.h**
- st25r95IsInListen : **st25r95_com.h**

- st25r95IsTransmitTxCompleted : **st25r95_com.h**
- st25r95KillIdle : **st25r95_com.h**
- st25r95PollRead : **st25r95_com.h**
- st25r95PrepareRx : **st25r95_com.h**
- st25r95ResetChip : **st25r95.h**
- st25r95SendCommandTypeAndLen : **st25r95_com.h**
- st25r95SendData : **st25r95_com.h**
- st25r95SendTransmitFlag : **st25r95_com.h**
- st25r95SetACState : **st25r95_com.h**
- ST25R_COM_SINGLETXRX : **rfal_platform.h**
- ST25R_INT_PIN : **rfal_platform.h**
- ST25R_INT_PORT : **rfal_platform.h**
- ST25R_RESET_PIN : **rfal_platform.h**
- ST25R_RESET_PORT : **rfal_platform.h**
- ST25R_SS_PIN : **rfal_platform.h**
- ST25R_SS_PORT : **rfal_platform.h**

Generated on Fri Mar 14 2025 11:45:12 for RFAL by doxygen 1.9.5

RFAL

Examples

Here is a list of all examples:

- **exampleRfalNfca.c**
- **exampleRfalPoller.c**

RFAL

exampleRfalNfca.c

```
/******************************************************************************
  * @attention
  *
  * COPYRIGHT 2016 STMicroelectronics, all rights reserved
  *
  * Unless required by applicable law or agreed to in writing, software
  * distributed under the License is distributed on an "AS IS" BASIS,
  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied,
  * AND SPECIFICALLY DISCLAIMING THE IMPLIED WARRANTIES OF MERCHANTABILITY,
  * FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT.
  * See the License for the specific language governing permissions and
  * limitations under the License.
  *
******************************************************************************/



/*
 *      PROJECT:   ST25R firmware
 *      LANGUAGE:  ISO C99
 */


/*  \file exampleRfalNfca.c
 *
 *  \example exampleRfalNfca.c
 *
 *  \author Gustavo Patricio
 *
```

```
 *  \brief Simple Passive NFCA Poll example
 *
 *   This example shows how to use the NFC-A module to detect and
 *   determine a NFC-A device
 *
 */


/*
 ******************************************************************************
 * INCLUDES
 ******************************************************************************
 */


#include "rfal_utils.h"
#include "rfal_platform.h"
#include "rfal_nfca.h"



#define EXAMPLE_NFCA_DEVICES     10


/*
******************************************************************************
* GLOBAL FUNCTIONS
******************************************************************************
*/


extern void exampleNFCARun( void )
{
    ReturnCode           err;
    rfalNfcaSensRes      sensRes;
    rfalNfcaSelRes       selRes;
    rfalNfcaListenDevice nfcaDevList[EXAMPLE_NFCA_DEVICES];
    uint8_t              devCnt;
    uint8_t              devIt;


    rfalInitialize();


    for(;;)
    {
        rfalFieldOff();                                                         /*
      Turn the Field Off */
        platformDelay(500);


        rfalNfcaPollerInitialize();                                             /*
      Initialize RFAL for NFC-A */
        rfalFieldOnAndStartGT();                                                /*
      Turns the Field On and starts GT timer */
```

```
 /****************************************************************************/
 /* Perform NFC-A Technology detection                                       */
 err = rfalNfcaPollerTechnologyDetection( RFAL_COMPLIANCE_MODE_NFC, &sensRes ); /* Poll
for nearby NFC-A devices */
 if(err == RFAL_ERR_NONE)
/* NFC-A type card found */
 {


     /****************************************************************************/
     /* Perform NFC-A Collision Resolution                                       */
     err = rfalNfcaPollerFullCollisionResolution( RFAL_COMPLIANCE_MODE_NFC,
EXAMPLE_NFCA_DEVICES, nfcaDevList, &devCnt );  /* Perform collision avoidance */


     if ( (err == RFAL_ERR_NONE) && (devCnt > 0) )
     {
         platformLog("NFC-A device(s) found %d\r\n", devCnt );


         devIt = 0;                                                          /* Use
the first device on the list */



/****************************************************************************/
         /* Check if desired device is in Sleep
*/
         if( nfcaDevList[devIt].isSleep )
         {
             err = rfalNfcaPollerCheckPresence( RFAL_14443A_SHORTFRAME_CMD_WUPA, &sensRes
); /* Wake up all cards  */
             if( err != RFAL_ERR_NONE )
             {
                 continue;
             }


             err = rfalNfcaPollerSelect( nfcaDevList[devIt].nfcId1,
nfcaDevList[devIt].nfcId1Len, &selRes ); /* Select specific device  */
             if( err != RFAL_ERR_NONE )
             {
                 continue;
             }
         }



/****************************************************************************/
         /* Perform protocol specific activation
*/
         switch( nfcaDevList[devIt].type )
         {


           case RFAL_NFCA_T1T:
```

```
                /* No further activation needed for a T1T (RID already performed)*/
                platformLog("NFC-A T1T device found \r\n");               /* NFC-A T1T
device found, NFCID/UID is contained in: t1tRidRes.uid */


                /* Following communications shall be performed using:
                 *   - Non blocking: rfalStartTransceive() + rfalGetTransceiveState()
                 *   -     Blocking: rfalTransceiveBlockingTx() +
rfalTransceiveBlockingRx() or rfalTransceiveBlockingTxRx()    */
                break;



          case RFAL_NFCA_T2T:


                /* No specific activation needed for a T2T */
                platformLog("NFC-A T2T device found \r\n");                   /* NFC-A
T2T device found, NFCID/UID is contained in: nfcaDev.nfcid */


                /* Following communications shall be performed using:
                 *   - Non blocking: rfalStartTransceive() + rfalGetTransceiveState()
                 *   -     Blocking: rfalTransceiveBlockingTx() +
rfalTransceiveBlockingRx() or rfalTransceiveBlockingTxRx()    */
                break;



          case RFAL_NFCA_T4T:


                platformLog("NFC-A T4T (ISO-DEP) device found \r\n");         /* NFC-A
T4T device found, NFCID/UID is contained in: nfcaDev.nfcid */


                /* Activation should continue using rfalIsoDepPollAHandleActivation(),
see exampleRfalPoller.c */
                break;



          case RFAL_NFCA_T4T_NFCDEP:                                          /* Device
supports T4T and NFC-DEP */
          case RFAL_NFCA_NFCDEP:                                              /* Device
supports NFC-DEP */


                platformLog("NFC-A P2P (NFC-DEP) device found \r\n");         /* NFC-A
P2P device found, NFCID/UID is contained in: nfcaDev.nfcid */
```

```
                        /* Activation should continue using
        rfalNfcDepInitiatorHandleActivation(), see exampleRfalPoller.c */
                        break;
                }


                rfalNfcaPollerSleep();                                          /* Put
        device to sleep / HLTA (useless as the field will be turned off anyhow) */
            }
          }
      }
}
```


exampleRfalPoller.c

```
/******************************************************************************
  * @attention
  *
  * COPYRIGHT 2016 STMicroelectronics, all rights reserved
  *
  * Unless required by applicable law or agreed to in writing, software
  * distributed under the License is distributed on an "AS IS" BASIS,
  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied,
  * AND SPECIFICALLY DISCLAIMING THE IMPLIED WARRANTIES OF MERCHANTABILITY,
  * FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT.
  * See the License for the specific language governing permissions and
  * limitations under the License.
  *
```

```
******************************************************************************/



/*
 *      PROJECT:   ST25R firmware
 *      LANGUAGE:  ISO C99
 */


/*  \file exampleRfalPoller.c
 *
 *  \example exampleRfalPoller.c
 *
 *  \author Gustavo Patricio
 *
 *  \brief  NFC Poller/Reader device (PCD) example
 *
 *   This example shows how to use the different RFAL modules to perform all
 *   procedures required as a NFC Poller / Reader device (PCD).
 *
 *   It performs all the necessary steps to detect and activate a nearby device
 *   (Tags/Cards and P2P enabled devices such as phones). Once the device has
 *   been activated a presence check is performed in a loop until the device
 *   has been removed.
 *
 *   This example follows the guidelines described on NFC Forum Activity spec
 *   but it does not fully implement all its requirements.
 *
 *
 */


/*
 ******************************************************************************
 * INCLUDES
 ******************************************************************************
 */


#include <stdint.h>
#include <stdbool.h>
#include "rfal_utils.h"
#include "rfal_utils.h"
#include "rfal_platform.h"
#include "rfal_nfca.h"
#include "rfal_nfcb.h"
#include "rfal_nfcf.h"
#include "rfal_nfcv.h"
#include "rfal_isoDep.h"
#include "rfal_nfcDep.h"
#include "rfal_analogConfig.h"
```

```
/*
******************************************************************************
* GLOBAL DEFINES
******************************************************************************
*/
#define EXAMPLE_RFAL_POLLER_DEVICES      10    /* Number of devices supported */
#define EXAMPLE_RFAL_POLLER_RF_BUF_LEN   255   /* RF buffer length            */


#define EXAMPLE_RFAL_POLLER_FOUND_NONE   0x00  /* No device found Flag        */
#define EXAMPLE_RFAL_POLLER_FOUND_A      0x01  /* NFC-A device found Flag     */
#define EXAMPLE_RFAL_POLLER_FOUND_B      0x02  /* NFC-B device found Flag     */
#define EXAMPLE_RFAL_POLLER_FOUND_F      0x04  /* NFC-F device found Flag     */
#define EXAMPLE_RFAL_POLLER_FOUND_V      0x08  /* NFC-V device Flag           */



/*
******************************************************************************
* GLOBAL TYPES
******************************************************************************
*/


typedef enum{
    EXAMPLE_RFAL_POLLER_STATE_INIT                =  0,  /* Initialize state            */
    EXAMPLE_RFAL_POLLER_STATE_TECHDETECT          =  1,  /* Technology Detection state  */
    EXAMPLE_RFAL_POLLER_STATE_COLAVOIDANCE        =  2,  /* Collision Avoidance state   */
    EXAMPLE_RFAL_POLLER_STATE_ACTIVATION          =  3,  /* Activation state            */
    EXAMPLE_RFAL_POLLER_STATE_DATAEXCHANGE_START  =  4,  /* Data Exchange Start state   */
    EXAMPLE_RFAL_POLLER_STATE_DATAEXCHANGE_CHECK  =  5,  /* Data Exchange Check state   */
    EXAMPLE_RFAL_POLLER_STATE_DEACTIVATION        =  9   /* Deactivation state          */
}exampleRfalPollerState;



typedef enum{
    EXAMPLE_RFAL_POLLER_TYPE_NFCA  =  0,                 /* NFC-A device type           */
    EXAMPLE_RFAL_POLLER_TYPE_NFCB  =  1,                 /* NFC-B device type           */
    EXAMPLE_RFAL_POLLER_TYPE_NFCF  =  2,                 /* NFC-F device type           */
    EXAMPLE_RFAL_POLLER_TYPE_NFCV  =  3                  /* NFC-V device type           */
}exampleRfalPollerDevType;



typedef enum{
    EXAMPLE_RFAL_POLLER_INTERFACE_RF     = 0,            /* RF Frame interface          */
    EXAMPLE_RFAL_POLLER_INTERFACE_ISODEP = 1,            /* ISO-DEP interface           */
    EXAMPLE_RFAL_POLLER_INTERFACE_NFCDEP = 2             /* NFC-DEP interface           */
}exampleRfalPollerRfInterface;
```

```
typedef struct{
    exampleRfalPollerDevType type;                      /* Device's type                */
    union{
        rfalNfcaListenDevice nfca;                      /* NFC-A Listen Device instance */
        rfalNfcbListenDevice nfcb;                      /* NFC-B Listen Device instance */
        rfalNfcfListenDevice nfcf;                      /* NFC-F Listen Device instance */
        rfalNfcvListenDevice nfcv;                      /* NFC-V Listen Device instance */
    }dev;                                               /* Device's instance            */


    exampleRfalPollerRfInterface rfInterface;           /* Device's interface           */
    union{
        rfalIsoDepDevice isoDep;                        /* ISO-DEP instance             */
        rfalNfcDepDevice nfcDep;                        /* NFC-DEP instance             */
    }proto;                                             /* Device's protocol            */


}exampleRfalPollerDevice;



/*
 ******************************************************************************
 * LOCAL VARIABLES
 ******************************************************************************
 */
static uint8_t                  t1tReadReq[]    = { 0x01, 0x00, 0x00, 0x11, 0x22, 0x33, 0x44 };
      /* T1T READ Block:0 Byte:0 */
static uint8_t                  t2tReadReq[]    = { 0x30, 0x00 };
      /* T2T READ Block:0 */
static uint8_t                  t3tCheckReq[]   = { 0x06, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66,
      0x77, 0x88, 0x01, 0x09, 0x00, 0x01, 0x80, 0x00 };   /* T3T Check/Read command */
static uint8_t                  t4tSelectReq[]  = { 0x00, 0xA4, 0x00, 0x00, 0x00 };
      /* T4T Select MF, DF or EF APDU  */
static uint8_t                  t5tSysInfoReq[] = { 0x02, 0x2B };
      /* NFC-V Get SYstem Information command*/
static uint8_t                  nfcbReq[]       = { 0x00 };
      /* NFC-B proprietary command */
static uint8_t                  llcpSymm[]      = { 0x00, 0x00 };
      /* LLCP SYMM command */


static uint8_t                  gNfcid3[]       = {0x01, 0xFE, 0x03, 0x04, 0x05, 0x06, 0x07,
      0x08, 0x09, 0x0A };                                 /* NFCID3 used for ATR_REQ */
static uint8_t                  gGenBytes[]     = { 0x46, 0x66, 0x6d, 0x01, 0x01, 0x11, 0x02,
      0x02, 0x07, 0x80, 0x03, 0x02, 0x00, 0x03, 0x04, 0x01, 0x32, 0x07, 0x01, 0x03 }; /* P2P
      General Bytes: LCCP Connect */


/******************************************************************************/


static uint8_t                  gDevCnt;                                    /* Number of devices
      found                         */
```

```
static exampleRfalPollerDevice gDevList[EXAMPLE_RFAL_POLLER_DEVICES];   /* Device List
      */
static exampleRfalPollerState  gState;                                  /* Main state
      */
static uint8_t                 gTechsFound;                             /* Technologies found
      bitmask                      */
exampleRfalPollerDevice        *gActiveDev;                             /* Active device pointer
      */
static uint16_t                gRcvLen;                                 /* Received length
      */
static bool                    gRxChaining;                             /* Rx chaining flag
      */


static union{
    uint8_t                 rfTxBuf[EXAMPLE_RFAL_POLLER_RF_BUF_LEN];    /* RF Tx buffer (not
       used on this demo)            */
    rfalIsoDepBufFormat     isoDepTxBuf;                                /* ISO-DEP Tx buffer
       format (with header/prologue) */
    rfalNfcDepBufFormat     nfcDepTxBuf;                                /* NFC-DEP Rx buffer
       format (with header/prologue) */
}gTxBuf;




static union {
    uint8_t                 rfRxBuf[EXAMPLE_RFAL_POLLER_RF_BUF_LEN];    /* RF Rx buffer
       */
    rfalIsoDepBufFormat     isoDepRxBuf;                                /* ISO-DEP Rx buffer
       format (with header/prologue) */
    rfalNfcDepBufFormat     nfcDepRxBuf;                                /* NFC-DEP Rx buffer
       format (with header/prologue) */
}gRxBuf;



/*
******************************************************************************
* LOCAL FUNCTION PROTOTYPES
******************************************************************************
*/
static bool exampleRfalPollerTechDetetection( void );
static bool exampleRfalPollerCollResolution( void );
static bool exampleRfalPollerActivation( uint8_t devIt );
static bool exampleRfalPollerNfcDepActivate( exampleRfalPollerDevice *device );
static ReturnCode exampleRfalPollerDataExchange( void );
static bool exampleRfalPollerDeactivate( void );



/*
******************************************************************************
* GLOBAL FUNCTIONS
******************************************************************************
```

```
*/

extern void exampleRfalPollerRun( void )
{
    ReturnCode err;

    rfalAnalogConfigInitialize();                                                   /*
       Initialize RFAL's Analog Configs */
    rfalInitialize();                                                               /*
       Initialize RFAL */
    platformLog("\n\rExample RFAL Poller started \r\n");

    for(;;)
    {
        rfalWorker();                                                               /* Execute
       RFAL process */

        switch( gState )
        {
            /*****************************************************************************/
            case EXAMPLE_RFAL_POLLER_STATE_INIT:

                gTechsFound = EXAMPLE_RFAL_POLLER_FOUND_NONE;
                gActiveDev  = NULL;
                gDevCnt     = 0;

                gState = EXAMPLE_RFAL_POLLER_STATE_TECHDETECT;
                break;



            /*****************************************************************************/
            case EXAMPLE_RFAL_POLLER_STATE_TECHDETECT:

                if( !exampleRfalPollerTechDetetection() )                           /* Poll
       for nearby devices in different technologies */
                {
                    gState = EXAMPLE_RFAL_POLLER_STATE_DEACTIVATION;                /* If no
       device was found, restart loop */
                    break;
                }

                gState = EXAMPLE_RFAL_POLLER_STATE_COLAVOIDANCE;                    /* One or
       more devices found, go to Collision Avoidance */
                break;
```

```
    /******************************************************************************/
    case EXAMPLE_RFAL_POLLER_STATE_COLAVOIDANCE:


        if( !exampleRfalPollerCollResolution() )                                /* Resolve
any eventual collision */
        {
            gState = EXAMPLE_RFAL_POLLER_STATE_DEACTIVATION;                    /* If
Collision Resolution was unable to retrieve any device, restart loop */
            break;
        }


        platformLog("Device(s) found: %d \r\n", gDevCnt);
        gState = EXAMPLE_RFAL_POLLER_STATE_ACTIVATION;                          /*
Device(s) have been identified, go to Activation */
        break;




    /******************************************************************************/
    case EXAMPLE_RFAL_POLLER_STATE_ACTIVATION:


        if( !exampleRfalPollerActivation( 0 ) )                                 /* Any
device previous identified can be Activated, on this example will select the firt on the
list */
        {
            gState = EXAMPLE_RFAL_POLLER_STATE_DEACTIVATION;                    /* If
Activation failed, restart loop */
            break;
        }


        gState = EXAMPLE_RFAL_POLLER_STATE_DATAEXCHANGE_START;                  /* Device
has been properly activated, go to Data Exchange */
        break;




    /******************************************************************************/
    case EXAMPLE_RFAL_POLLER_STATE_DATAEXCHANGE_START:
    case EXAMPLE_RFAL_POLLER_STATE_DATAEXCHANGE_CHECK:


        err = exampleRfalPollerDataExchange();                                  /* Perform
Data Exchange, in this example a simple transfer will executed in order to do device's
presence check */
        switch( err )
        {
            case RFAL_ERR_NONE:                                                     /*
Data exchange successful  */
                platformDelay(300);                                             /* Wait a
bit */
```

```
                    gState = EXAMPLE_RFAL_POLLER_STATE_DATAEXCHANGE_START;        /* Trigger
new exchange with device */
                    break;


                case RFAL_ERR_BUSY:                                                   /*
Data exchange ongoing  */
                    gState = EXAMPLE_RFAL_POLLER_STATE_DATAEXCHANGE_CHECK;        /* Once
triggered/started the Data Exchange only do check until is completed */
                    break;


                default:                                                          /* Data
exchange not successful, card removed or other transmission error */
                    platformLog("Data exchange terminated with error: %d \r\n", err);
                    gState = EXAMPLE_RFAL_POLLER_STATE_DEACTIVATION;              /* Restart
loop */
                    break;
            }
            break;




        /****************************************************************************/
        case EXAMPLE_RFAL_POLLER_STATE_DEACTIVATION:


            exampleRfalPollerDeactivate();                                        /* If a
card has been activated, properly deactivate the device */


            rfalFieldOff();                                                       /* Turn
the Field Off powering down any device nearby */
            platformDelay( 500 );                                                 /* Remain
a certain period with field off */


            gState = EXAMPLE_RFAL_POLLER_STATE_INIT;                              /* Restart
the loop */
            break;




        /****************************************************************************/
        default:
            return;
    }
  }
}



static bool exampleRfalPollerTechDetetection( void )
{
```

```
ReturnCode           err;
rfalNfcaSensRes      sensRes;
rfalNfcbSensbRes     sensbRes;
rfalNfcvInventoryRes invRes;
uint8_t              sensbResLen;


gTechsFound = EXAMPLE_RFAL_POLLER_FOUND_NONE;


/******************************************************************************/
/* NFC-A Technology Detection                                                 */
/******************************************************************************/


rfalNfcaPollerInitialize();                                                    /*
   Initialize RFAL for NFC-A */
rfalFieldOnAndStartGT();                                                       /* Turns
   the Field On and starts GT timer */


err = rfalNfcaPollerTechnologyDetection( RFAL_COMPLIANCE_MODE_NFC, &sensRes ); /* Poll for
   NFC-A devices */
if( err == RFAL_ERR_NONE )
{
    gTechsFound |= EXAMPLE_RFAL_POLLER_FOUND_A;
}




/******************************************************************************/
/* NFC-B Technology Detection                                                 */
/******************************************************************************/


rfalNfcbPollerInitialize();                                                    /*
   Initialize RFAL for NFC-B */
rfalFieldOnAndStartGT();                                                       /* As
   field is already On only starts GT timer */


err = rfalNfcbPollerTechnologyDetection( RFAL_COMPLIANCE_MODE_NFC, &sensbRes, &sensbResLen
   ); /* Poll for NFC-B devices */
if( err == RFAL_ERR_NONE )
{
    gTechsFound |= EXAMPLE_RFAL_POLLER_FOUND_B;
}




/******************************************************************************/
/* NFC-F Technology Detection                                                 */
/******************************************************************************/
```

```
    rfalNfcfPollerInitialize( RFAL_BR_212 );                                    /*
       Initialize RFAL for NFC-F */
    rfalFieldOnAndStartGT();                                                    /* As
       field is already On only starts GT timer */


    err = rfalNfcfPollerCheckPresence();                                        /* Poll
       for NFC-F devices */
    if( err == RFAL_ERR_NONE )
    {
        gTechsFound |= EXAMPLE_RFAL_POLLER_FOUND_F;
    }




    /******************************************************************************/
    /* NFC-V Technology Detection                                                 */
    /******************************************************************************/


    rfalNfcvPollerInitialize();                                                 /*
       Initialize RFAL for NFC-V */
    rfalFieldOnAndStartGT();                                                    /* As
       field is already On only starts GT timer */


    err = rfalNfcvPollerCheckPresence( &invRes );                               /* Poll
       for NFC-V devices */
    if( err == RFAL_ERR_NONE )
    {
        gTechsFound |= EXAMPLE_RFAL_POLLER_FOUND_V;
    }


    return (gTechsFound != EXAMPLE_RFAL_POLLER_FOUND_NONE);
}


static bool exampleRfalPollerCollResolution( void )
{
    uint8_t    i;
    uint8_t    devCnt;
    ReturnCode err;




    /******************************************************************************/
    /* NFC-A Collision Resolution                                                 */
    /******************************************************************************/
    if( gTechsFound & EXAMPLE_RFAL_POLLER_FOUND_A )                             /* If a
       NFC-A device was found/detected, perform Collision Resolution */
    {
        rfalNfcaListenDevice nfcaDevList[EXAMPLE_RFAL_POLLER_DEVICES];
```

```
    rfalNfcaPollerInitialize();
    rfalFieldOnAndStartGT();                                                    /* Ensure
   GT again as other technologies have also been polled */
    err = rfalNfcaPollerFullCollisionResolution( RFAL_COMPLIANCE_MODE_NFC,
   (EXAMPLE_RFAL_POLLER_DEVICES - gDevCnt), nfcaDevList, &devCnt );
    if( (err == RFAL_ERR_NONE) && (devCnt != 0) )
    {
        for( i=0; i<devCnt; i++ )                                               /* Copy
   devices found form local Nfca list into global device list */
        {
            gDevList[gDevCnt].type     = EXAMPLE_RFAL_POLLER_TYPE_NFCA;
            gDevList[gDevCnt].dev.nfca = nfcaDevList[i];
            gDevCnt++;
        }
    }
}


/****************************************************************************/
/* NFC-B Collision Resolution                                               */
/****************************************************************************/
if( gTechsFound & EXAMPLE_RFAL_POLLER_FOUND_B )                             /* If a
   NFC-A device was found/detected, perform Collision Resolution */
{
    rfalNfcbListenDevice nfcbDevList[EXAMPLE_RFAL_POLLER_DEVICES];


    rfalNfcbPollerInitialize();
    rfalFieldOnAndStartGT();                                                    /* Ensure
   GT again as other technologies have also been polled */
    err = rfalNfcbPollerCollisionResolution( RFAL_COMPLIANCE_MODE_NFC,
   (EXAMPLE_RFAL_POLLER_DEVICES - gDevCnt), nfcbDevList, &devCnt );
    if( (err == RFAL_ERR_NONE) && (devCnt != 0) )
    {
        for( i=0; i<devCnt; i++ )                                               /* Copy
   devices found form local Nfcb list into global device list */
        {
            gDevList[gDevCnt].type     = EXAMPLE_RFAL_POLLER_TYPE_NFCB;
            gDevList[gDevCnt].dev.nfcb = nfcbDevList[i];
            gDevCnt++;
        }
    }
}


/****************************************************************************/
/* NFC-F Collision Resolution                                               */
/****************************************************************************/
if( gTechsFound & EXAMPLE_RFAL_POLLER_FOUND_F )                             /* If a
   NFC-F device was found/detected, perform Collision Resolution */
{
    rfalNfcfListenDevice nfcfDevList[EXAMPLE_RFAL_POLLER_DEVICES];


    rfalNfcfPollerInitialize( RFAL_BR_212 );
    rfalFieldOnAndStartGT();                                                    /* Ensure
   GT again as other technologies have also been polled */
```

```
        err = rfalNfcfPollerCollisionResolution( RFAL_COMPLIANCE_MODE_NFC,
       (EXAMPLE_RFAL_POLLER_DEVICES - gDevCnt), nfcfDevList, &devCnt );
        if( (err == RFAL_ERR_NONE) && (devCnt != 0) )
        {
            for( i=0; i<devCnt; i++ )                                          /* Copy
       devices found form local Nfcf list into global device list */
            {
                gDevList[gDevCnt].type     = EXAMPLE_RFAL_POLLER_TYPE_NFCF;
                gDevList[gDevCnt].dev.nfcf = nfcfDevList[i];
                gDevCnt++;
            }
        }
    }


    /****************************************************************************/
    /* NFC-V Collision Resolution                                               */
    /****************************************************************************/
    if( gTechsFound & EXAMPLE_RFAL_POLLER_FOUND_V )                            /* If a
       NFC-F device was found/detected, perform Collision Resolution */
    {
        rfalNfcvListenDevice nfcvDevList[EXAMPLE_RFAL_POLLER_DEVICES];


        rfalNfcvPollerInitialize();
        rfalFieldOnAndStartGT();                                               /* Ensure
       GT again as other technologies have also been polled */
        err = rfalNfcvPollerCollisionResolution( (EXAMPLE_RFAL_POLLER_DEVICES - gDevCnt),
       nfcvDevList, &devCnt );
        if( (err == RFAL_ERR_NONE) && (devCnt != 0) )
        {
            for( i=0; i<devCnt; i++ )                                          /* Copy
       devices found form local Nfcf list into global device list */
            {
                gDevList[gDevCnt].type     = EXAMPLE_RFAL_POLLER_TYPE_NFCV;
                gDevList[gDevCnt].dev.nfcv = nfcvDevList[i];
                gDevCnt++;
            }
        }
    }


    return (gDevCnt > 0);
}



static bool exampleRfalPollerActivation( uint8_t devIt )
{
    ReturnCode          err;
    rfalNfcaSensRes     sensRes;
    rfalNfcaSelRes      selRes;
    rfalNfcbSensbRes    sensbRes;
    uint8_t             sensbResLen;
```

```
if( devIt > gDevCnt )
{
    return false;
}


switch( gDevList[devIt].type )
{
    /******************************************************************************/
    /* NFC-A Activation                                                           */
    /******************************************************************************/
    case EXAMPLE_RFAL_POLLER_TYPE_NFCA:


        rfalNfcaPollerInitialize();
        if( gDevList[devIt].dev.nfca.isSleep )                                  /* Check
  if desired device is in Sleep      */
        {
            err = rfalNfcaPollerCheckPresence( RFAL_14443A_SHORTFRAME_CMD_WUPA, &sensRes );
  /* Wake up all cards  */
            if( err != RFAL_ERR_NONE )
            {
                return false;
            }


            err = rfalNfcaPollerSelect( gDevList[devIt].dev.nfca.nfcId1,
  gDevList[devIt].dev.nfca.nfcId1Len, &selRes ); /* Select specific device  */
            if( err != RFAL_ERR_NONE )
            {
                return false;
            }
        }


        /******************************************************************************/
        /* Perform protocol specific activation                                       */
        switch( gDevList[devIt].dev.nfca.type )
        {

  /******************************************************************************/
            case RFAL_NFCA_T1T:


                /* No further activation needed for a T1T (RID already performed)*/
                platformLog("NFC-A T1T device activated \r\n");                 /* NFC-A
  T1T device activated */


                gDevList[devIt].rfInterface = EXAMPLE_RFAL_POLLER_INTERFACE_RF;
                break;




  /******************************************************************************/
```

```
        case RFAL_NFCA_T2T:

            /* No specific activation needed for a T2T */
            platformLog("NFC-A T2T device activated \r\n");                    /* NFC-A
T2T device activated */

            gDevList[devIt].rfInterface = EXAMPLE_RFAL_POLLER_INTERFACE_RF;
            break;




/*****************************************************************************/
        case RFAL_NFCA_T4T:

            /* Perform ISO-DEP (ISO14443-4) activation: RATS and PPS if supported */
            rfalIsoDepInitialize();
            err = rfalIsoDepPollAHandleActivation(
(rfalIsoDepFSxI)RFAL_ISODEP_FSDI_DEFAULT, RFAL_ISODEP_NO_DID, RFAL_BR_424,
&gDevList[devIt].proto.isoDep );
            if( err != RFAL_ERR_NONE )
            {
                return false;
            }

            platformLog("NFC-A T4T (ISO-DEP) device activated \r\n");          /* NFC-A
T4T device activated */

            gDevList[devIt].rfInterface = EXAMPLE_RFAL_POLLER_INTERFACE_ISODEP;
            break;




/*****************************************************************************/
        case RFAL_NFCA_T4T_NFCDEP:                                              /*
Device supports both T4T and NFC-DEP */
        case RFAL_NFCA_NFCDEP:                                                  /*
Device supports NFC-DEP */

            /* Perform NFC-DEP (P2P) activation: ATR and PSL if supported */
            if( !exampleRfalPollerNfcDepActivate( &gDevList[devIt] ) )
            {
              return false;
            }

            platformLog("NFC-A P2P (NFC-DEP) device activated \r\n");          /* NFC-A
P2P device activated */
```

```
            gDevList[devIt].rfInterface = EXAMPLE_RFAL_POLLER_INTERFACE_NFCDEP;
            break;
    }


    break;


 /****************************************************************************/
 /* NFC-B Activation                                                         */
 /****************************************************************************/
 case EXAMPLE_RFAL_POLLER_TYPE_NFCB:


    rfalNfcbPollerInitialize();
    if( gDevList[devIt].dev.nfcb.isSleep )                                  /* Check
if desired device is in Sleep */
    {
        /* Wake up all cards. SENSB_RES may return collision but the NFCID0 is available
to explicitly select NFC-B card via ATTRIB; so error will be ignored here */
        rfalNfcbPollerCheckPresence( RFAL_NFCB_SENS_CMD_ALLB_REQ, RFAL_NFCB_SLOT_NUM_1,
&sensbRes, &sensbResLen );
    }




    /****************************************************************************/
    /* Perform ISO-DEP (ISO14443-4) activation: RATS and PPS if supported       */
    rfalIsoDepInitialize();
    err = rfalIsoDepPollBHandleActivation( (rfalIsoDepFSxI)RFAL_ISODEP_FSDI_DEFAULT,
RFAL_ISODEP_NO_DID, RFAL_BR_424, 0x00, &gDevList[devIt].dev.nfcb, NULL, 0,
&gDevList[devIt].proto.isoDep );
    if( err == RFAL_ERR_NONE )
    {
        platformLog("NFC-B T4T (ISO-DEP) device activated \r\n");           /* NFC-B
T4T device activated */


        gDevList[devIt].rfInterface = EXAMPLE_RFAL_POLLER_INTERFACE_ISODEP ;
        break;
    }


    platformLog("NFC-B device activated \r\n");                             /* NFC-B
device activated */
    gDevList[devIt].rfInterface =  EXAMPLE_RFAL_POLLER_INTERFACE_RF;
    break;


 /****************************************************************************/
 /* NFC-F Activation                                                         */
 /****************************************************************************/
 case EXAMPLE_RFAL_POLLER_TYPE_NFCF:


    rfalNfcfPollerInitialize( RFAL_BR_212 );
```

```
            if( rfalNfcfIsNfcDepSupported( &gDevList[devIt].dev.nfcf ) )
            {
                /* Perform NFC-DEP (P2P) activation: ATR and PSL if supported */
                if( !exampleRfalPollerNfcDepActivate( &gDevList[devIt] ) )
                {
                    return false;
                }


                platformLog("NFC-F P2P (NFC-DEP) device activated \r\n");             /* NFC-A
      P2P device activated */


                gDevList[devIt].rfInterface = EXAMPLE_RFAL_POLLER_INTERFACE_NFCDEP;
                break;
            }


            platformLog("NFC-F T3T device activated \r\n");                           /* NFC-F
      T3T device activated */
            gDevList[devIt].rfInterface = EXAMPLE_RFAL_POLLER_INTERFACE_RF;
            break;


        /****************************************************************************/
        /* NFC-V Activation                                                         */
        /****************************************************************************/
        case EXAMPLE_RFAL_POLLER_TYPE_NFCV:


            rfalNfcvPollerInitialize();


            /* No specific activation needed for a T5T */
            platformLog("NFC-V T5T device activated \r\n");                           /* NFC-V
      T5T device activated */


            gDevList[devIt].rfInterface = EXAMPLE_RFAL_POLLER_INTERFACE_RF;
            break;


        /****************************************************************************/
        default:
            return false;
    }


    gActiveDev = &gDevList[devIt];                                                    /* Assign
      active device to be used further on */
    return true;
}



static bool exampleRfalPollerNfcDepActivate( exampleRfalPollerDevice *device )
```

```
{
    rfalNfcDepAtrParam   param;


    /******************************************************************************/
    /* If Passive F use the NFCID2 retrieved from SENSF                           */
    if( device->type == EXAMPLE_RFAL_POLLER_TYPE_NFCF )
    {
        param.nfcid    = device->dev.nfcf.sensfRes.NFCID2;
        param.nfcidLen = RFAL_NFCF_NFCID2_LEN;
    }
    else
    {
        param.nfcid    = gNfcid3;
        param.nfcidLen = RFAL_NFCDEP_NFCID3_LEN;
    }


    param.BS    = RFAL_NFCDEP_Bx_NO_HIGH_BR;
    param.BR    = RFAL_NFCDEP_Bx_NO_HIGH_BR;
    param.DID   = RFAL_NFCDEP_DID_NO;
    param.NAD   = RFAL_NFCDEP_NAD_NO;
    param.LR    = RFAL_NFCDEP_LR_254;
    param.GB    = gGenBytes;
    param.GBLen = sizeof(gGenBytes);
    param.commMode  = RFAL_NFCDEP_COMM_PASSIVE;
    param.operParam = (RFAL_NFCDEP_OPER_FULL_MI_EN | RFAL_NFCDEP_OPER_EMPTY_DEP_DIS |
       RFAL_NFCDEP_OPER_ATN_EN | RFAL_NFCDEP_OPER_RTOX_REQ_EN);


    /* Perform NFC-DEP (P2P) activation: ATR and PSL if supported */
    return (rfalNfcDepInitiatorHandleActivation( &param, RFAL_BR_424, &device->proto.nfcDep ) ==
       RFAL_ERR_NONE);
}




static ReturnCode exampleRfalPollerDataExchange( void )
{
    rfalTransceiveContext ctx;
    ReturnCode            err;
    rfalIsoDepTxRxParam   isoDepTxRx;
    rfalNfcDepTxRxParam   nfcDepTxRx;
    uint8_t               *txBuf;
    uint16_t              txBufLen;



    /******************************************************************************/
    /* The Data Exchange is divided in two different moments, the trigger/Start of *
     *  the transfer followed by the check until its completion                   */
    if( gState == EXAMPLE_RFAL_POLLER_STATE_DATAEXCHANGE_START )                      /*
       Trigger/Start the data exchange */
    {
```

```
 switch( gActiveDev->rfInterface )                                          /* Check
which RF interface shall be used/has been activated */
 {
     /*****************************************************************************/
     case EXAMPLE_RFAL_POLLER_INTERFACE_RF:


         switch( gActiveDev->type )                                         /* Over RF
interface no specific protocol is selected, each device supports a different protocol */
         {

/*****************************************************************************/
             case EXAMPLE_RFAL_POLLER_TYPE_NFCA:
                 switch( gActiveDev->dev.nfca.type )
                 {

/*****************************************************************************/
                     case RFAL_NFCA_T1T:


                         /* To perform presence check, on this example a T1T Read command
is used */
                         RFAL_MEMCPY( &t1tReadReq[3], gActiveDev->dev.nfca.nfcId1,
RFAL_NFCA_CASCADE_1_UID_LEN );  /* Assign device's NFCID for read command */


                         txBuf    = t1tReadReq;
                         txBufLen = sizeof(t1tReadReq);
                         break;



/*****************************************************************************/
                     case RFAL_NFCA_T2T:


                         /* To perform presence check, on this example a T2T Read command
is used */
                         txBuf    = t2tReadReq;
                         txBufLen = sizeof(t2tReadReq);
                         break;



/*****************************************************************************/
                     default:
                         return RFAL_ERR_REQUEST;
                 }
                 break;




/*****************************************************************************/
             case EXAMPLE_RFAL_POLLER_TYPE_NFCB:
```

```
                /* To perform presence check, no specific command is used */
                txBuf    = nfcbReq;
                txBufLen = sizeof(nfcbReq);
                break;




/*****************************************************************************/
            case EXAMPLE_RFAL_POLLER_TYPE_NFCF:


                /* To perform presence check, on this example a T3T Check/Read command
is used */
                RFAL_MEMCPY( &t3tCheckReq[1], gActiveDev->dev.nfcf.sensfRes.NFCID2,
RFAL_NFCF_NFCID2_LEN );  /* Assign device's NFCID for Check command */


                txBuf    = t3tCheckReq;
                txBufLen = sizeof(t3tCheckReq);
                break;




/*****************************************************************************/
            case EXAMPLE_RFAL_POLLER_TYPE_NFCV:


                /* To perform presence check, on this example a Get System Information
command is used */
                txBuf    = t5tSysInfoReq;
                txBufLen = sizeof(t5tSysInfoReq);
                break;




/*****************************************************************************/
            default:
                return RFAL_ERR_REQUEST;
        }



/*****************************************************************************/
        /* Trigger a RFAL Transceive using the previous defined frames
*/
        rfalCreateByteFlagsTxRxContext( ctx, txBuf, txBufLen, gRxBuf.rfRxBuf,
sizeof(gRxBuf.rfRxBuf), &gRcvLen, RFAL_TXRX_FLAGS_DEFAULT, rfalConvMsTo1fc(20) );
        return (((err = rfalStartTransceive( &ctx )) == RFAL_ERR_NONE) ? RFAL_ERR_BUSY :
err);     /* Signal RFAL_ERR_BUSY as Data Exchange has been started and is ongoing */
```

```
    case EXAMPLE_RFAL_POLLER_INTERFACE_ISODEP:

        RFAL_MEMCPY( gTxBuf.isoDepTxBuf.inf, t4tSelectReq, sizeof(t4tSelectReq) );

        isoDepTxRx.DID          = RFAL_ISODEP_NO_DID;
        isoDepTxRx.ourFSx       = RFAL_ISODEP_FSX_KEEP;
        isoDepTxRx.FSx          = gActiveDev->proto.isoDep.info.FSx;
        isoDepTxRx.dFWT         = gActiveDev->proto.isoDep.info.dFWT;
        isoDepTxRx.FWT          = gActiveDev->proto.isoDep.info.FWT;
        isoDepTxRx.txBuf        = &gTxBuf.isoDepTxBuf;
        isoDepTxRx.txBufLen     = sizeof(t4tSelectReq);
        isoDepTxRx.isTxChaining = false;
        isoDepTxRx.rxBuf        = &gRxBuf.isoDepRxBuf;
        isoDepTxRx.rxLen        = &gRcvLen;
        isoDepTxRx.isRxChaining = &gRxChaining;


/******************************************************************************/
        /* Trigger a RFAL ISO-DEP Transceive
*/
        return (((err = rfalIsoDepStartTransceive( isoDepTxRx )) == RFAL_ERR_NONE) ?
RFAL_ERR_BUSY : err); /* Signal RFAL_ERR_BUSY as Data Exchange has been started and is
ongoing */



    case EXAMPLE_RFAL_POLLER_INTERFACE_NFCDEP:

        RFAL_MEMCPY( gTxBuf.nfcDepTxBuf.inf, llcpSymm, sizeof(llcpSymm) );

        nfcDepTxRx.DID          = RFAL_NFCDEP_DID_KEEP;
        nfcDepTxRx.FSx          = rfalNfcDepLR2FS( rfalNfcDepPP2LR( gActiveDev-
>proto.nfcDep.activation.Target.ATR_RES.PPt ) );
        nfcDepTxRx.dFWT         = gActiveDev->proto.nfcDep.info.dFWT;
        nfcDepTxRx.FWT          = gActiveDev->proto.nfcDep.info.FWT;
        nfcDepTxRx.txBuf        = &gTxBuf.nfcDepTxBuf;
        nfcDepTxRx.txBufLen     = sizeof(llcpSymm);
        nfcDepTxRx.isTxChaining = false;
        nfcDepTxRx.rxBuf        = &gRxBuf.nfcDepRxBuf;
        nfcDepTxRx.rxLen        = &gRcvLen;
        nfcDepTxRx.isRxChaining = &gRxChaining;


/******************************************************************************/
        /* Trigger a RFAL NFC-DEP Transceive
*/
        return (((err = rfalNfcDepStartTransceive( &nfcDepTxRx )) == RFAL_ERR_NONE) ?
RFAL_ERR_BUSY : err);  /* Signal RFAL_ERR_BUSY as Data Exchange has been started and is
ongoing */
```

```
            default:
                break;
        }
    }
    /******************************************************************************/
    /* The Data Exchange has been started, wait until completed                   */
    else if( gState == EXAMPLE_RFAL_POLLER_STATE_DATAEXCHANGE_CHECK )
    {
        switch( gActiveDev->rfInterface )
        {
            /******************************************************************************/
            case EXAMPLE_RFAL_POLLER_INTERFACE_RF:
                return rfalGetTransceiveStatus();


            /******************************************************************************/
            case EXAMPLE_RFAL_POLLER_INTERFACE_ISODEP:
                return rfalIsoDepGetTransceiveStatus();


            /******************************************************************************/
            case EXAMPLE_RFAL_POLLER_INTERFACE_NFCDEP:
                return rfalNfcDepGetTransceiveStatus();


            /******************************************************************************/
            default:
                return RFAL_ERR_PARAM;
        }
    }
    return RFAL_ERR_REQUEST;
}



static bool exampleRfalPollerDeactivate( void )
{
    if( gActiveDev != NULL )                                                    /* Check
       if a device has been activated */
    {
        switch( gActiveDev->rfInterface )
        {
            /******************************************************************************/
            case EXAMPLE_RFAL_POLLER_INTERFACE_RF:
                break;                                                          /* No
       specific deactivation to be performed */


            /******************************************************************************/
            case EXAMPLE_RFAL_POLLER_INTERFACE_ISODEP:
                rfalIsoDepDeselect();                                           /* Send a
       Deselect to device */
                break;
```

```
            /*****************************************************************************/
            case EXAMPLE_RFAL_POLLER_INTERFACE_NFCDEP:
                rfalNfcDepRLS();                                                    /* Send a
       Release to device */
                break;


            default:
                return false;
        }
        platformLog("Device deactivated \r\n");
    }


    return true;
}
```


RFAL

Main Page

Modules

Data Structures

Files

Examples

Graph Legend

This page explains how to interpret the graphs that are generated by doxygen.

Consider the following example:

```
/*! Invisible class because of truncation */
class Invisible { };

/*! Truncated class, inheritance relation is hidden */
class Truncated : public Invisible { };


/* Class not documented with doxygen comments */
class Undocumented { };
```

```
/*! Class that is inherited using public inheritance */
class PublicBase : public Truncated { };

/*! A template class */
template<class T> class Templ { };

/*! Class that is inherited using protected inheritance */
class ProtectedBase { };

/*! Class that is inherited using private inheritance */
class PrivateBase { };

/*! Class that is used by the Inherited class */
class Used { };

/*! Super class that inherits a number of other classes */
class Inherited : public PublicBase,
                  protected ProtectedBase,
                  private PrivateBase,
                  public Undocumented,
                  public Templ<int>
{
  private:
    Used *m_usedClass;
};
```

This will result in the following graph:

The boxes in the above graph have the following meaning:

- A filled gray box represents the struct or class for which the graph is generated.
- A box with a black border denotes a documented struct or class.
- A box with a gray border denotes an undocumented struct or class.
- A box with a red border denotes a documented struct or class forwhich not all inheritance/containment relations are shown. A graph is truncated if it does not fit within the specified boundaries.

The arrows have the following meaning:

- A blue arrow is used to visualize a public inheritance relation between two classes.

- A dark green arrow is used for protected inheritance.
- A dark red arrow is used for private inheritance.
- A purple dashed arrow is used if a class is contained or used by another class. The arrow is labelled with the variable(s) through which the pointed class or struct is accessible.
- A yellow dashed arrow denotes a relation between a template instance and the template class it was instantiated from. The arrow is labelled with the template parameters of the instance.