

OrangeForRN2483

Version
8/11/2017 3:08:00 PM

Table of Contents

OrangeForRN2483	2
Class Index	3
File Index	4
Class Documentation	5
DownlinkMessage	5
LpwaOrangeEncoderClass	7
OrangeForRN2483Class	11
RadioCmdsClass	26
RnRequestClass	33
SysCmdsClass	36
File Documentation	40
ConstOrangeForRN2483.h	40
DownlinkMessage.cpp	46
DownlinkMessage.h	47
InternalConstForRN2483.h	48
LpwaOrangeEncoder.cpp	51
LpwaOrangeEncoder.h	52
OrangeForRN2483.cpp	53
OrangeForRN2483.h	54
RadioCmds.cpp	55
RadioCmds.h	56
RnRequest.cpp	58
RnRequest.h	59
SysCmds.cpp	62
SysCmds.h	63
Index	65

OrangeForRN2483

Version:

1.0.5

Author:

Karim Baali

Class Index

Class List

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

<u>DownlinkMessage</u>	5
<u>LpwaOrangeEncoderClass</u>	7
<u>OrangeForRN2483Class</u>	11
<u>RadioCmdsClass</u>	26
<u>RnRequestClass</u>	33
<u>SysCmdsClass</u>	36

File Index

File List

Here is a list of all files with brief descriptions:

<u>ConstOrangeForRN2483.h</u> (Defines all the constant values used in the project)	40
<u>DownlinkMessage.cpp</u>	46
<u>DownlinkMessage.h</u> (User's interface to access to the received data)	47
<u>InternalConstForRN2483.h</u>	48
<u>LpwaOrangeEncoder.cpp</u>	51
<u>LpwaOrangeEncoder.h</u> (Helpful methods to create, update or decode payloads)	52
<u>OrangeForRN2483.cpp</u>	53
<u>OrangeForRN2483.h</u> (User's main interface to use the module)	54
<u>RadioCmds.cpp</u>	55
<u>RadioCmds.h</u> (Contains all the RADIO commands)	56
<u>RnRequest.cpp</u>	58
<u>RnRequest.h</u> (Main methods used by the library to communicate with the module)	59
<u>SysCmds.cpp</u>	62
<u>SysCmds.h</u> (Contains all the SYS commands)	63

Class Documentation

DownlinkMessage Class Reference

```
#include <DownlinkMessage.h>
```

Public Member Functions

- [DownlinkMessage](#) ()
Constructor for the [DownlinkMessage](#) class.
- [~DownlinkMessage](#) ()
Destructor for the [DownlinkMessage](#) class.
- int8_t [getPort](#) ()
Getter for the port class attribute.
- const String [getMessage](#) ()
Getter for the receiveBuffer class attribute as a string value.
- const int8_t * [getMessageByteArray](#) (int8_t *len)
Getter for the receiveBuffer class attribute as a byte array.

Protected Member Functions

- void [setPort](#) (int8_t port)
- void [setResponseMessage](#) (int8_t *message)

Friends

- class [OrangeForRN2483Class](#)

Constructor & Destructor Documentation

DownlinkMessage::DownlinkMessage ()

Constructor for the [DownlinkMessage](#) class.

Used to instantiate a new [DownlinkMessage](#) object

DownlinkMessage::~~DownlinkMessage ()

Destructor for the [DownlinkMessage](#) class.

Used to delete an [DownlinkMessage](#) instance

Member Function Documentation

const String DownlinkMessage::getMessage ()

Getter for the *receiveBuffer* class attribute as a string value.

This function allows the user to have access to the receiveBuffer attribute corresponding to the data sent by the server as a string value

Returns:

String value corresponding to the `receiveBuffer` attribute value

const int8_t * DownlinkMessage::getMessageByteArray (int8_t * *len*)

Getter for the *receiveBuffer* class attribute as a byte array.

This function allows the user to have access to the `receiveBuffer` attribute corresponding to the data sent by the server as a byte array

Parameters:

<i>len</i>	Pointer on an <code>uint8_t</code> value to receive the message length value
------------	--

Returns:

Byte array corresponding to the `receiveBuffer` attribute value

int8_t DownlinkMessage::getPort ()

Getter for the *port* class attribute.

This function allows the user to have access to the port attribute

Returns:

Decimal number corresponding to the port attribute value

void DownlinkMessage::setPort (int8_t *port*) [protected]

void DownlinkMessage::setResponseMessage (int8_t * *message*) [protected]

Friends And Related Function Documentation

friend class [OrangeForRN2483Class](#) [friend]

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

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

LpwaOrangeEncoderClass Class Reference

```
#include <LpwaOrangeEncoder.h>
```

Public Member Functions

- [LpwaOrangeEncoderClass \(\)](#)
Constructor for the [LpwaOrangeEncoderClass](#) class.
- virtual [~LpwaOrangeEncoderClass \(\)](#)
Destructor for the [LpwaOrangeEncoderClass](#) class.
- void [flush \(\)](#)
Flushing the payload and counter attributes of the class.
- int8_t * [getFramePayload](#) (int8_t *len)
Getter for a payload and its length.
- bool [addBool](#) (bool value)
Quickly add a boolean value to a payload.
- bool [addByte](#) (int8_t value)
Quickly add a byte to a payload.
- bool [addShort](#) (int16_t value)
Quickly add a short to a payload.
- bool [addInt](#) (int32_t value)
Quickly add an integer to a payload.
- bool [addLong](#) (int64_t value)
Quickly add a long to a payload.
- bool [addFloat](#) (float value)
Quickly add a float to a payload.
- bool [addUByte](#) (uint8_t value)
Quickly add an unsigned byte to a payload.
- bool [addUShort](#) (uint16_t value)
Quickly add an unsigned short to a payload.
- bool [addUInt](#) (uint32_t value)
Quickly add an unsigned integer to a payload.
- bool [addULong](#) (uint64_t value)
Quickly add an unsigned long to a payload.

Constructor & Destructor Documentation

LpwaOrangeEncoderClass::LpwaOrangeEncoderClass ()

Constructor for the [LpwaOrangeEncoderClass](#) class.

Used to instantiate a new [LpwaOrangeEncoderClass](#) object

LpwaOrangeEncoderClass::~~LpwaOrangeEncoderClass () [virtual]

Destructor for the [LpwaOrangeEncoderClass](#) class.

Used to delete an [LpwaOrangeEncoderClass](#) instance

Member Function Documentation

bool LpwaOrangeEncoderClass::addBool (bool *value*)

Quickly add a boolean value to a payload.

This function allows the user to quickly add a boolean value to a payload if it doesn't exceed the maximal length

Parameters:

<i>value</i>	Boolean value to add to the payload
--------------	-------------------------------------

Returns:

A boolean value, true if everything is ok, false if the add was impossible

bool LpwaOrangeEncoderClass::addByte (int8_t *value*)

Quickly add a byte to a payload.

This function allows the user to quickly add a byte to a payload if it doesn't exceed the maximal length

Parameters:

<i>value</i>	Byte to add to the payload
--------------	----------------------------

Returns:

A boolean value, true if everything is ok, false if the add was impossible

bool LpwaOrangeEncoderClass::addFloat (float *value*)

Quickly add a float to a payload.

This function allows the user to quickly add a float to a payload if it doesn't exceed the maximal length

Parameters:

<i>value</i>	Float to add to the payload
--------------	-----------------------------

Returns:

A boolean value, true if everything is ok, false if the add was impossible

bool LpwaOrangeEncoderClass::addInt (int32_t *value*)

Quickly add an integer to a payload.

This function allows the user to quickly add an integer to a payload if it doesn't exceed the maximal length

Parameters:

<i>value</i>	Integer to add to the payload
--------------	-------------------------------

Returns:

A boolean value, true if everything is ok, false if the add was impossible

bool LpwaOrangeEncoderClass::addLong (int64_t *value*)

Quickly add a long to a payload.

This function allows the user to quickly add a long to a payload if it doesn't exceed the maximal length

Parameters:

<i>value</i>	Long to add to the payload
--------------	----------------------------

Returns:

A boolean value, true if everything is ok, false if the add was impossible

bool LpwaOrangeEncoderClass::addShort (int16_t *value*)

Quickly add a short to a payload.

This function allows the user to quickly add a short to a payload if it doesn't exceed the maximal length

Parameters:

<i>value</i>	Short to add to the payload
--------------	-----------------------------

Returns:

A boolean value, true if everything is ok, false if the add was impossible

bool LpwaOrangeEncoderClass::addUByte (uint8_t *value*)

Quickly add an unsigned byte to a payload.

This function allows the user to quickly add an unsigned byte to a payload if it doesn't exceed the maximal length

Parameters:

<i>value</i>	Unsigned byte to add to the payload
--------------	-------------------------------------

Returns:

A boolean value, true if everything is ok, false if the add was impossible

bool LpwaOrangeEncoderClass::addUInt (uint32_t *value*)

Quickly add an unsigned integer to a payload.

This function allows the user to quickly add an unsigned integer to a payload if it doesn't exceed the maximal length

Parameters:

<i>value</i>	Unsigned integer to add to the payload
--------------	--

Returns:

A boolean value, true if everything is ok, false if the add was impossible

bool LpwaOrangeEncoderClass::addULong (uint64_t *value*)

Quickly add an unsigned long to a payload.

This function allows the user to quickly add an unsigned long to a payload if it doesn't exceed the maximal length

Parameters:

<i>value</i>	Unsigned long to add to the payload
--------------	-------------------------------------

Returns:

A boolean value, true if everything is ok, false if the add was impossible

bool LpwaOrangeEncoderClass::addUShort (uint16_t *value*)

Quickly add an unsigned short to a payload.

This function allows the user to quickly add an unsigned short to a payload if it doesn't exceed the maximal length

Parameters:

<i>value</i>	Unsigned short to add to the payload
--------------	--------------------------------------

Returns:

A boolean value, true if everything is ok, false if the add was impossible

void LpwaOrangeEncoderClass::flush ()

Flushing the payload and counter attributes of the class.

This function is used to reset the counter to 0 and and clear the framePayload attribute

int8_t * LpwaOrangeEncoderClass::getFramePayload (int8_t * *len*)

Getter for a payload and its length.

This function allows the user to get a pointer on a frame payload to be able to manipulate it easily and to receive its length in the argument variable

Parameters:

<i>len</i>	Pointer on an uint8_t value to receive the payload length value
------------	---

Returns:

An uint8_t pointer on the frame payload

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

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

OrangeForRN2483Class Class Reference

```
#include <OrangeForRN2483.h>
```

Public Member Functions

- [OrangeForRN2483Class \(\)](#)
Constructor for the [OrangeForRN2483Class](#) class.
- virtual [~OrangeForRN2483Class \(\)](#)
Destructor for the [OrangeForRN2483Class](#) class.
- void [init \(\)](#)
Initializing the lorastream attribute.
- bool [getJoinState \(\)](#)
Getter for the isNetworkJoined class attribute.
- bool [setAbpKeys](#) (const int8_t *nwksKey, const int8_t *appSKey)
Setting the mandatory keys for a join request.
- bool [joinNetwork](#) (const int8_t *devEUI, const int8_t *appEUI, const int8_t *appKey)
Sending a join request to the server.
- bool [joinNetwork](#) (const int8_t *appEUI, const int8_t *appKey)
Sending a join request to the server with HWEUI by default.
- bool [sendMessage](#) ([eTypeMessage](#) typeMessage, int8_t *data, uint8_t size, int8_t port)
Sending data to the server.
- bool [sendMessage](#) (int8_t *data, uint8_t size, int8_t port)
Sending data to the server.
- [DownlinkMessage](#) * [getDownlinkMessage \(\)](#)
Getter for [DownlinkMessage](#) attribute.
- [RadioCmdsClass](#) * [getRadioCmds \(\)](#)
Getter for RadioCmds attribute.
- [SysCmdsClass](#) * [getSysCmds \(\)](#)
Getter for SysCmds attribute.
- [eErrorType](#) [getLastError \(\)](#)
Getter for the last saved error.
- String [getDevAddr \(\)](#)
Getter on the device address.
- String [getDevEUI \(\)](#)
Getter on the device EUI.
- String [getAppEUI \(\)](#)
Getter on the application EUI.
- [eBoolean](#) [isAdr \(\)](#)
Getter on the adaptive data rate mechanism.
- bool [enableAdr](#) (bool adr=true)
Setter for the adaptive data rate value.
- bool [getStatus](#) (uint32_t &status)
Getter on the current status of the module.
- short [getSync \(\)](#)
Getter on the synchronization word.
- String [getAutoReply \(\)](#)
Getter on the state of the automatic reply.
- [eDataRate](#) [getDataRate \(\)](#)

Getter on the data rate of the module.

- [ePowerIdx getPwrIdxValue \(\)](#)
Getter on the output power index value.
- `int16_t` [getBand \(\)](#)
Getter on the frequency band.
- `int16_t` [getRetransNb \(\)](#)
Getter on the number of retransmissions.
- `int16_t` [getDemodMargin \(\)](#)
Getter on the demodulation margin.
- `int16_t` [getGatewayNb \(\)](#)
Getter on the number of gateways.
- `int16_t` [getRx2](#) (`uint16_t freqBand`)
Getter on the data rate configured for the second receive window.
- `int32_t` [getRxdelay1 \(\)](#)
Getter on the interval for rxdelay1.
- `int32_t` [getRxdelay2 \(\)](#)
Getter on the interval for rxdelay2.
- `int32_t` [getDCyclePs \(\)](#)
Getter on the duty cycle prescaler value.
- `int64_t` [getUpctr \(\)](#)
Getter on the uplink frame counter.
- `int64_t` [getDwnctr \(\)](#)
Getter on the downlink frame counter.
- `bool` [setDevAddr](#) (`const int8_t *devAddr`)
Setter for the device address.
- `bool` [setDevEUI](#) (`const int8_t *devEUI`)
Setter for the device EUI.
- `bool` [setAppEUI](#) (`const int8_t *appEUI`)
Setter for the application EUI.
- `bool` [setNwkSKey](#) (`const int8_t *nwkSKey`)
Setter for the network session key.
- `bool` [setAppSKey](#) (`const int8_t *appSKey`)
Setter for the application session key.
- `bool` [setAppKey](#) (`const int8_t *appKey`)
Setter for the application key.
- `bool` [setPwrIdx](#) (`uint8_t pwrIdx`)
Setter for the power index value.
- `bool` [setDataRate](#) (`eDataRate dataRate`)
Setter for the data rate value.
- `bool` [setBatLvl](#) (`uint8_t lvl`)
Setter for the battery level value.
- `bool` [setRetx](#) (`uint8_t retx`)
Setter for the number of retransmissions for an uplink confirmed packet.
- `bool` [setLinkCheck](#) (`uint16_t linkCheck`)
Setter for the link check process interval.
- `bool` [setRxDelay1](#) (`uint16_t rxDelay1`)
Setter for the delay between the transmission and the first reception window.
- `bool` [setAutoReply](#) (`String autoRep`)
Setter for the auto reply state.

- bool [setRx2](#) ([eDataRate](#) dataRate, uint32_t frequency)
Setter for the data rate and frequency used for the second receive window.
- bool [setSync](#) (int8_t syncWord)
Setter for the synchronization word.
- bool [setUpctr](#) (uint32_t upctr)
- bool [setDwnctr](#) (uint32_t dwnctr)
Setter for the downlink frame counter.
- bool [join](#) ()
Send a request to authenticate with the network.
- bool [save](#) ()
Save configuration parameters to the user EEPROM.
- bool [pause](#) ()
Pause the LoRaWAN stack functionality.
- bool [resume](#) ()
Resume the LoRaWAN stack functionality.

Protected Member Functions

- bool [isStreamInit](#) ()
- int8_t * [tx](#) ([eTypeMessage](#) typeMessage, int8_t *data, uint8_t size, int8_t port)
- void [setLastError](#) ([eErrorType](#) errorType)
- bool [setOtaKeys](#) (const int8_t *devEui, const int8_t *appEui, const int8_t *appKey)
- void [resetDevice](#) ()

Protected Attributes

- [RadioCmdsClass](#) [RadioCmds](#)
- [SysCmdsClass](#) [SysCmds](#)
- [DownlinkMessage](#) [downlinkMessage](#)
- Stream * [diagStream](#)
- bool [isNetworkJoined](#)
- const char * [params](#) [[COUNT_PARAM_MAC](#)]
- const char * [commandType](#) [[COUNT_TYPE](#)] = { "mac", "sys", "radio" }
- const char * [possibleResponses](#) [[LORA_COUNT_ERRORS](#)]

Constructor & Destructor Documentation

OrangeForRN2483Class::OrangeForRN2483Class ()

Constructor for the [OrangeForRN2483Class](#) class.

Used to instantiate a new [OrangeForRN2483Class](#) object

OrangeForRN2483Class::~~OrangeForRN2483Class () [virtual]

Destructor for the [OrangeForRN2483Class](#) class.

Used to delete an [OrangeForRN2483Class](#) instance

Member Function Documentation

bool OrangeForRN2483Class::enableAdr (bool *adr* = true)

Setter for the adaptive data rate value.

This function allows the user to set or update the **adaptive data rate** (ADR) by executing a "mac set adr <adr>" command on the module

Parameters:

<i>adr</i>	Boolean value enabling or not the adaptive data rate
------------	--

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

String OrangeForRN2483Class::getAppEUI ()

Getter on the application EUI.

This function allows the user to have access to the **application EUI** identifier by executing a "mac get appeui" command on the module

Returns:

The received application EUI as a *string* value

String OrangeForRN2483Class::getAutoReply ()

Getter on the state of the automatic reply.

This function allows the user to have access to the **state** of the **automatic reply** by executing a "mac get ar" command on the module

Returns:

The received value as a *string* value, either **on** or **off**

int16_t OrangeForRN2483Class::getBand ()

Getter on the frequency band.

This function allows the user to have access to the **frequency band** by executing a "mac get band" command on the module

Returns:

The received value as a *decimal number*, either 868 or 433

[eDataRate](#) OrangeForRN2483Class::getDataRate ()

Getter on the data rate of the module.

This function allows the user to have access to the **data rate** by executing a "mac get dr" command on the module

Returns:

The received value as an *eDataRate* value (see [constOrangeForRn2483.h](#) for more information)

int32_t OrangeForRN2483Class::getDCyclePs ()

Getter on the duty cycle prescaler value.

This function allows the user to have access to the duty cycle **prescaler value** by executing a "mac get dcycleps" command on the module

Returns:

The received value as a *decimal number* , from 0 to 65535

int16_t OrangeForRN2483Class::getDemodMargin ()

Getter on the demodulation margin.

This function allows the user to have access to the **demodulation margin** by executing a "mac get mrgn" command on the module

Returns:

The received value as a *decimal number* , from 0 to 255

String OrangeForRN2483Class::getDevAddr ()

Getter on the device address.

This function allows the user to have access to the **device address** by executing a "mac get devaddr" command on the module

Returns:

The received device address as a *string* value

String OrangeForRN2483Class::getDevEUI ()

Getter on the device EUI.

This function allows the user to have access to the unique **device EUI** identifier by executing a "mac get deveui" command on the module

Returns:

The received device EUI as a *string* value

[DownlinkMessage](#) * OrangeForRN2483Class::getDownlinkMessage ()

Getter for [DownlinkMessage](#) attribute.

This function allows the user to have access to the methods available in the [DownlinkMessage](#) object

Returns:

Pointer on the attribute

int64_t OrangeForRN2483Class::getDwnctr ()

Getter on the downlink frame counter.

This function allows the user to have access to the **downlink frame counter** by executing a "mac get dnctr" command on the module

Returns:

The received value as a *decimal number* , from 0 to 4294967295

int16_t OrangeForRN2483Class::getGatewayNb ()

Getter on the number of gateways.

This function allows the user to have access to the **number of gateways** by executing a "mac get gwnb" command on the module

Returns:

The received value as a *decimal number* , from 0 to 255

bool OrangeForRN2483Class::getJoinState ()

Getter for the *isNetworkJoined* class attribute.

This function allows the user to have access to the isNetworkJoined attribute

Returns:

Boolean corresponding to the attribute value

eErrorType OrangeForRN2483Class::getLastError ()

Getter for the last saved error.

This function allows the user to have access to the value of the last saved error

Returns:

eErrorType value corresponding to the last error found (see [constOrangeForRn2483.h](#) for more information)

ePowerIdx OrangeForRN2483Class::getPwrIdxValue ()

Getter on the output power index value.

This function allows the user to have access to the **output power index value** by executing a "mac get pwrIdx" command on the module

Returns:

The received value as an *ePowerIdx* value (see [constOrangeForRn2483.h](#) for more information)

RadioCmdsClass * OrangeForRN2483Class::getRadioCmds ()

Getter for RadioCmds attribute.

This function allows the user to have access to the radio commands available in the RadioCmds object

Returns:

Pointer on the attribute

int16_t OrangeForRN2483Class::getRetransNb ()

Getter on the number of retransmissions.

This function allows the user to have access to the **number of retransmissions** by executing a "mac get retx" command on the module

Returns:

The received value as a *decimal number* , from 0 to 255

int16_t OrangeForRN2483Class::getRx2 (uint16_t freqBand)

Getter on the data rate configured for the second receive window.

This function allows the user to have access to the **data rate** for the second receive window by executing a "mac get rx2 <freqBand>" command on the module

Parameters:

<i>freqBand</i>	Decimal number representing the frequency band
-----------------	--

Returns:

The received value as a *decimal number*

int32_t OrangeForRN2483Class::getRxdelay1 ()

Getter on the interval for rxdelay1.

This function allows the user to have access to the **interval for rxdelay1** by executing a "mac get rxdelay1" command on the module

Returns:

The received value as a *decimal number* , from 0 to 65535

int32_t OrangeForRN2483Class::getRxdelay2 ()

Getter on the interval for rxdelay2.

This function allows the user to have access to the **interval for rxdelay2** by executing a "mac get rxdelay2" command on the module

Returns:

The received value as a *decimal number* , from 0 to 65535

bool OrangeForRN2483Class::getStatus (uint32_t & status)

Getter on the current status of the module.

This function allows the user to have access to the **module status** by executing a "mac get status" command on the module

Parameters:

<i>status</i>	uint32_t variable to store the received status from the module
---------------	--

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

short OrangeForRN2483Class::getSync ()

Getter on the synchronization word.

This function allows the user to have access to the **synchronization word** by executing a "mac get sync" command on the module

Returns:

The received value as a *short value*

SysCmdsClass * OrangeForRN2483Class::getSysCmds ()

Getter for SysCmds attribute.

This function allows the user to have access to the system commands available in the SysCmds object

Returns:

Pointer on the attribute

int64_t OrangeForRN2483Class::getUpctr ()

Getter on the uplink frame counter.

This function allows the user to have access to the **uplink frame counter** by executing a "mac get upctr" command on the module

Returns:

The received value as a *decimal number* , from 0 to 4294967295

void OrangeForRN2483Class::init ()

Initializing the lorastream attribute.

This function is used to initialize the lorastream with UART stream

eBoolean OrangeForRN2483Class::isAdr ()

Getter on the adaptive data rate mechanism.

This function allows the user to have access to the **adaptive data rate** by executing a "mac get adr" command on the module

Returns:

The received value as an *eBoolean* value (see [constOrangeForRn2483.h](#) for more information)

bool OrangeForRN2483Class::isStreamInit () [protected]

bool OrangeForRN2483Class::join ()

Send a request to authenticate with the network.

This function allows the user to send a **join request** to the server by executing a "mac join <mode>" command on the module

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

bool OrangeForRN2483Class::joinNetwork (const int8_t * *devEUI*, const int8_t * *appEUI*, const int8_t * *appKey*)

Sending a join request to the server.

This function allows the user to send a *join request* to the server

Parameters:

<i>devEUI</i>	Hexadecimal number represented by an array whose size is 8 int8_t
<i>appEUI</i>	Hexadecimal number represented by an array whose size is 8 int8_t
<i>appKey</i>	Hexadecimal number represented by an array whose size is 16 int8_t

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

bool OrangeForRN2483Class::joinNetwork (const int8_t * *appEUI*, const int8_t * *appKey*)

Sending a join request to the server with HWEUI by default.

This function allows the user to send a *join request* to the server with the hardware devEUI (HWEUI) used by default

Parameters:

<i>appEUI</i>	Hexadecimal number represented by an array whose size is 8 int8_t
<i>appKey</i>	Hexadecimal number represented by an array whose size is 16 int8_t

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

bool OrangeForRN2483Class::pause ()

Pause the LoRaWAN stack functionality.

This function allows the user to pause the **LoRaWAN stack functionality** to allow transceiver configuration by executing a "mac pause" command on the module

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

void OrangeForRN2483Class::resetDevice () [protected]

bool OrangeForRN2483Class::resume ()

Resume the LoRaWAN stack functionality.

This function allows the user to resume the **LoRaWAN stack functionality** after being paused by executing a "mac resume" command on the module

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

bool OrangeForRN2483Class::save ()

Save configuration parameters to the user EEPROM.

This function allows the user to save the **configuration parameters** to the user EEPROM, allowing the module to be initialized with the last saved information avec a reset by executing a "mac save" command on the module

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

bool OrangeForRN2483Class::sendMessage ([eTypeMessage](#) *typeMessage*, int8_t * *data*, uint8_t *size*, int8_t *port*)

Sending data to the server.

This function allows the user to **send data** to the server by giving the data, the size the port to use

Parameters:

<i>typeMessage</i>	eTypeMessage value representing the uplink payload type (<i>CONFIRMED_MESSAGE</i> or <i>UNCONFIRMED_MESSAGE</i>)
--------------------	--

<i>data</i>	Hexadecimal value representing the data sent to the server
<i>size</i>	Unsigned integer value representing the size of the transmitted data only
<i>port</i>	Integer value representing the port to use

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

bool OrangeForRN2483Class::sendMessage (int8_t * *data*, uint8_t *size*, int8_t *port*)

Sending data to the server.

This function allows the user to **send *data*** to the server by giving the data, the size the port to use. An unconfirmed message is sent by default by calling previous method

Parameters:

<i>data</i>	Hexadecimal value representing the data sent to the server
<i>size</i>	Unsigned integer value representing the size of the transmitted data only
<i>port</i>	Integer value representing the port to use

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

bool OrangeForRN2483Class::setAbpKeys (const int8_t * *nwkSkey*, const int8_t * *appSKey*)

Setting the mandatory keys for a join request.

This function allows the user to set the different kind of **keys needed** to be able to send an ABP join request to the server.

Parameters:

<i>nwkSkey</i>	Hexadecimal number represented by an array whose size is 16 int8_t
<i>appSKey</i>	Hexadecimal number represented by an array whose size is 16 int8_t

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

bool OrangeForRN2483Class::setAppEUI (const int8_t * *appEUI*)

Setter for the application EUI.

This function allows the user to set or update the **application EUI identifier** by executing a "mac set appeui <appEUI>" command on the module

Parameters:

<i>appEUI</i>	Hexadecimal number represented by an array whose size is 8 int8_t
---------------	--

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

bool OrangeForRN2483Class::setAppKey (const int8_t * *appKey*)

Setter for the application key.

This function allows the user to set or update the **application key** by executing a "mac set appkey <appKey>" command on the module

Parameters:

<i>appKey</i>	Hexadecimal number represented by an array whose size is 16 int8_t
---------------	---

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

bool OrangeForRN2483Class::setAppSKey (const int8_t * *appSKey*)

Setter for the application session key.

This function allows the user to set or update the **application session key** by executing a "mac set appskey <appSessionKey>" command on the module

Parameters:

<i>appSKey</i>	Hexadecimal number represented by an array whose size is 16 int8_t
----------------	---

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

bool OrangeForRN2483Class::setAutoReply (String *autoRep*)

Setter for the auto reply state.

This function allows the user to set or update the **state of the automatic reply** by executing a "mac set ar <autoRep>" command on the module

Parameters:

<i>autoRep</i>	String value enabling or not the automatic reply (<i>on</i> or <i>off</i>)
----------------	--

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

bool OrangeForRN2483Class::setBatLvl (uint8_t *lvl*)

Setter for the battery level value.

This function allows the user to set or update the **battery level value** by executing a "mac set bat <lvl>" command on the module

Parameters:

<i>lvl</i>	Decimal number representing the battery level, from 0 to 255
------------	--

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

bool OrangeForRN2483Class::setDataRate ([eDataRate](#) *dataRate*)

Setter for the data rate value.

This function allows the user to set or update the **data rate value** by executing a "mac set dr <dataRate>" command on the module

Parameters:

<i>dataRate</i>	eDataRate value representing the data rate value (see constOrangeForRn2483.h for more information)
-----------------	--

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

bool OrangeForRN2483Class::setDevAddr (const int8_t * *devAddr*)

Setter for the device address.

This function allows the user to set or update the **device address** by executing a "mac set devaddr <devAddr>" command on the module

Parameters:

<i>devAddr</i>	Hexadecimal number represented by an array whose size is 4 int8_t
----------------	--

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

bool OrangeForRN2483Class::setDevEUI (const int8_t * devEUI)

Setter for the device EUI.

This function allows the user to set or update the **device EUI identifier** by executing a "mac set deveui <devEUI>" command on the module

Parameters:

<i>devEUI</i>	Hexadecimal number represented by an array whose size is 8 int8_t
---------------	--

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

bool OrangeForRN2483Class::setDwnctr (uint32_t dwnctr)

Setter for the downlink frame counter.

This function allows the user to set or update the **downlink frame counter** used for the next uplink transmission by executing a "mac set dnctr <dwnctr>" command on the module

Parameters:

<i>dwnctr</i>	Decimal number representing the downlink frame counter, from 0 to 4294967295
---------------	--

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

void OrangeForRN2483Class::setLastError ([eErrorType](#) errorType) [protected]

bool OrangeForRN2483Class::setLinkCheck (uint16_t linkCheck)

Setter for the link check process interval.

This function allows the user to set or update the link check process **interval** by executing a "mac set linkchk <linkCheck>" command on the module

Parameters:

<i>linkCheck</i>	Decimal number representing the interval for the link check process, from 0 to 65535
------------------	--

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

bool OrangeForRN2483Class::setNwkSKey (const int8_t * nwkSKey)

Setter for the network session key.

This function allows the user to set or update the **network session key** by executing a "mac set nwkskey <nwkSKey>" command on the module

Parameters:

<i>nwkSKey</i>	Hexadecimal number represented by an array whose size is 16 int8_t
----------------	---

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

bool OrangeForRN2483Class::setOttaKeys (const int8_t * *devEui*, const int8_t * *appEui*, const int8_t * *appKey*) [protected]

bool OrangeForRN2483Class::setPwrIdx (uint8_t *pwrIdx*)

Setter for the power index value.

This function allows the user to set or update the **power index value** by executing a "mac set pwrIdx <pwrIdx>" command on the module

Parameters:

<i>pwrIdx</i>	Decimal number representing the index value for the output power
---------------	--

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

bool OrangeForRN2483Class::setRetx (uint8_t *retx*)

Setter for the number of retransmissions for an uplink confirmed packet.

This function allows the user to set or update the **number of retransmissions** for an uplink confirmed packet by executing a "mac set retx <retx>" command on the module

Parameters:

<i>retx</i>	Decimal number representing the number of retransmissions, from 0 to 255
-------------	--

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

bool OrangeForRN2483Class::setRx2 ([eDataRate](#) *dataRate*, uint32_t *frequency*)

Setter for the data rate and frequency used for the second receive window.

This function allows the user to set or update the **data rate** and the **frequency** used for the second receive window by executing a "mac set rx2 <dataRate> <frequency>" command on the module

Parameters:

<i>dataRate</i>	eDataRate value representing the data rate value (see constOrangeForRn2483.h for more information)
<i>frequency</i>	Decimal number representing the frequency value, 863000000 to 870000000 or from 433050000 to 434790000

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

bool OrangeForRN2483Class::setRxDelay1 (uint16_t *rxDelay1*)

Setter for the delay between the transmission and the first reception window.

This function allows the user to set or update the **delay** between the transmission and the first reception window by executing a "mac set rxdelay1 <rxDelay1>" command on the module

Parameters:

<i>rxDelay1</i>	Decimal number representing the delay in milliseconds, from 0 to 65535
-----------------	--

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

bool OrangeForRN2483Class::setSync (int8_t *syncWord*)

Setter for the synchronization word.

This function allows the user to set or update the **synchronization word** for the LoRaWAN communication by executing a "mac set sync <syncWord>" command on the module

Parameters:

<i>syncWord</i>	Decimal number representing the synchronization word
-----------------	--

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

bool OrangeForRN2483Class::setUpctr (uint32_t *upctr*)

for the uplink frame counter

This function allows the user to set or update the **uplink frame counter** used for the next uplink transmission by executing a "mac set upctr <upctr>" command on the module

Parameters:

<i>upctr</i>	Decimal number representing the uplink frame counter, from 0 to 4294967295
--------------	--

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

int8_t * OrangeForRN2483Class::tx ([eTypeMessage](#) *typeMessage*, int8_t * *data*, uint8_t *size*, int8_t *port*) [protected]

Member Data Documentation

const char* OrangeForRN2483Class::commandType[\[COUNT_TYPE\]](#) = { "mac", "sys", "radio" } [protected]

Stream* OrangeForRN2483Class::diagStream [protected]

[DownlinkMessage](#) **OrangeForRN2483Class::downlinkMessage** [protected]

bool OrangeForRN2483Class::isNetworkJoined [protected]

const char* OrangeForRN2483Class::params[\[COUNT_PARAM_MAC\]](#) [protected]

const char* OrangeForRN2483Class::possibleResponses[\[LORA_COUNT_ERRORS\]](#) [protected]

```
Initial value:= { "ok"
                  "invalid_param",
                  "keys_not_init",
                  "no_free_ch",
                  "silent",
                  "busy",
                  "mac paused",
                  "denied",
                  "invalid_data_len",
```

```
"frame_counter_err_rejoin_needed"}
```

[RadioCmdsClass](#) OrangeForRN2483Class::RadioCmds [protected]

[SysCmdsClass](#) OrangeForRN2483Class::SysCmds [protected]

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

- [OrangeForRN2483.h](#)
- [OrangeForRN2483.cpp](#)

RadioCmdsClass Class Reference

```
#include <RadioCmds.h>
```

Public Member Functions

- [eBT getBt \(\)](#)
Getter on the data shaping FSK configuration.
- `bool setBt (eBT bt)`
Setter for the data shaping value of the module.
- [eModulation getModulation \(\)](#)
Getter on the current mode of operation of the module.
- `bool setModulation (eModulation mode)`
Setter for the modulation method of the module.
- [eSpreadingFactor getSF \(\)](#)
Getter on the current spreading factor of the module.
- `bool setSF (eSpreadingFactor spreadingFactor)`
Setter for the spreading factor used during transmission.
- [eBoolean getCrc \(\)](#)
Getter on the status of the CRC header.
- [eBoolean getIqInversion \(\)](#)
Getter on the status of the Invert IQ fonctionnality.
- [eCodingRate getCodingRate \(\)](#)
Getter on the current value settings used for the coding rate.
- `short getSync ()`
Getter on the synchronization word used for radio communication.
- `float getAutoFreqCorrBw ()`
Getter on the status of the Automatic Frequency Correction Bandwidth.
- `float getReceiveBw ()`
Getter on the signal bandwidth used for receiving.
- `bool getOutputPower (int8_t &outputPower)`
Getter on the current power level setting used in operation.
- `int16_t getBandWidth ()`
Getter on the current operating radio bandwidth.
- `int16_t getSigNoiseRation ()`
Getter on the value of the signal to noise ratio (SNR)
- `int32_t getBitRate ()`
Getter on the configured bit rate for FSK communication.
- `int32_t getFreqDeviation ()`
Getter on the frequency deviation setting on the transceiver.
- `int32_t getPreambleLength ()`
Getter on the current preamble length used for communication.
- `int32_t getFrequency ()`
Getter on the current operation frequency of the module.
- `bool setFrequency (int32_t frequency)`
Setter for the communication frequency of the radio transceiver.
- `bool getWatchdog (uint64_t &watchdog)`
Getter on the length used for the watchdog time-out.
- `bool setOutputPower (int8_t pwrout)`

Setter for the transceiver output power.

- bool [setAutoFreqBand](#) (String autoFreqBand)
Setter for the automatic frequency correction bandwidth.

Protected Attributes

- const char * [params](#) [[COUNT](#) [PARAM](#) [RAD](#)]
-

Member Function Documentation

float RadioCmdsClass::getAutoFreqCorrBw ()

Getter on the status of the Automatic Frequency Correction Bandwidth.

This function allows the user to have access to the status of the **automatic frequency correction bandwidth** by executing a "radio get afcbw" command on the module

Returns:

The received value as a *float* value

int16_t RadioCmdsClass::getBandWidth ()

Getter on the current operating radio bandwidth.

This function allows the user to have access to the current **operating radio bandwidth** by executing a "radio get bw" command on the module

Returns:

The received value as a *decimal* value

int32_t RadioCmdsClass::getBitRate ()

Getter on the configured bit rate for FSK communication.

This function allows the user to have access to the configured **bit rate** for FSK communication by executing a "radio get bitrate" command on the module

Returns:

The received value as a *signed decimal* value

[eBT](#) RadioCmdsClass::getBt ()

Getter on the data shaping FSK configuration.

[constOrangeForRn2483](#)

This function allows the user to have access to the **data shaping FSK configuration** by executing a "radio get bt" command on the module

Returns:

The received value as an *eBT* value (see [constOrangeForRN2483.h](#) for more information)

[eCodingRate](#) RadioCmdsClass::getCodingRate ()

Getter on the current value settings used for the coding rate.

This function allows the user to have access to the current value settings used for the **coding rate** during communication by executing a "radio get cr" command on the module

Returns:

The received value as an *eCodingRate* value (see [constOrangeForRN2483.h](#) for more information)

eBoolean RadioCmdsClass::getCrc ()

Getter on the status of the CRC header.

This function allows the user to have access to the status of the **CRC header** by executing a "radio get crc" command on the module

Returns:

The received value as an *eBoolean* value (see [constOrangeForRN2483.h](#) for more information)

int32_t RadioCmdsClass::getFreqDeviation ()

Getter on the frequency deviation setting on the transceiver.

This function allows the user to have access to the **frequency deviation setting** on the transceiver by executing a "radio get fdev" command on the module

Returns:

The received value as a *signed decimal* value

int32_t RadioCmdsClass::getFrequency ()

Getter on the current operation frequency of the module.

This function allows the user to have access to the current **operation frequency** of the module by executing a "radio get freq" command on the module

Returns:

The received value as a *decimal* value

eBoolean RadioCmdsClass::getIqInversion ()

Getter on the status of the Invert IQ functionality.

This function allows the user to have access to the status of the **Invert IQ functionality** by executing a "radio get iq" command on the module

Returns:

The received value as an *eBoolean* value (see [constOrangeForRN2483.h](#) for more information)

eModulation RadioCmdsClass::getModulation ()

Getter on the current mode of operation of the module.

This function allows the user to have access to the current **mode of operation** of the module by executing a "radio get mod" command on the module

Returns:

The received value as an *eModulation* value (see [constOrangeForRN2483.h](#) for more information)

bool RadioCmdsClass::getOutputPower (int8_t & outputPower)

Getter on the current power level setting used in operation.

This function allows the user to have access to the current **power level** setting used in operation by executing a "radio get pwr" command on the module

Parameters:

<i>outputPower</i>	uint8_t variable to store the received power from the module
--------------------	--

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

int32_t RadioCmdsClass::getPreambleLength ()

Getter on the current preamble length used for communication.

This function allows the user to have access to the current **preamble length** used for communication by executing a "radio get prlen" command on the module

Returns:

The received value as a *signed decimal* value

float RadioCmdsClass::getReceiveBw ()

Getter on the signal bandwidth used for receiving.

This function allows the user to have access to the **signal bandwidth** used for receiving by executing a "radio get rxbw" command on the module

Returns:

The received value as a *float* value

[eSpreadingFactor](#) RadioCmdsClass::getSF ()

Getter on the current spreading factor of the module.

This function allows the user to have access to the current **spreading factor** used by the transceiver by executing a "radio get sf" command on the module

Returns:

The received value as an *eSpreadingFactor* value (see [constOrangeForRN2483.h](#) for more information)

int16_t RadioCmdsClass::getSigNoiseRation ()

Getter on the value of the signal to noise ratio (SNR)

This function allows the user to have access to the value of the **signal to noise ratio** for the last received packet by executing a "radio get snr" command on the module

Returns:

The received value as a *signed decimal* value

short RadioCmdsClass::getSync ()

Getter on the synchronization word used for radio communication.

This function allows the user to have access to the configured synchronization word used for radio communication during communication by executing a "radio get sync" command on the module

Returns:

The received value as a *short* value

bool RadioCmdsClass::getWatchdog (uint64_t & watchdog)

Getter on the length used for the watchdog time-out.

This function allows the user to have access to the **length** used for the **watchdog time-out** by executing a "radio get wdt" command on the module

Parameters:

<i>watchdog</i>	uint64_t variable to store the received value from the module
-----------------	---

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

bool RadioCmdsClass::setAutoFreqBand (String autoFreqBand)

Setter for the automatic frequency correction bandwidth.

This function allows the user to set or update the **automatic frequency correction bandwidth** for receiving/transmitting by executing a "radio set afcbw <autoFreqBand>" command on the module

Parameters:

<i>autoFreqBand</i>	String representing the automatic frequency correction
---------------------	--

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

bool RadioCmdsClass::setBt ([eBT](#) bt)

Setter for the data shaping value of the module.

This function allows the user to set or update the **data shaping value** applied to FSK transmissions by executing a "radio set bt <gfBT>" command on the module

Parameters:

<i>bt</i>	eBT value representing the Gaussian baseband data shaping (see constOrangeForRN2483.h for more information)
-----------	---

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

bool RadioCmdsClass::setFrequency (int32_t frequency)

Setter for the communication frequency of the radio transceiver.

This function allows the user to set or update the **communication frequency** of the radio transceiver by executing a "radio set freq <frequency>" command on the module

Parameters:

<i>frequency</i>	Decimal number representing the communication frequency, from 433050000 to 434790000 or from 863000000 to 870000000, in Hz.
------------------	---

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

bool RadioCmdsClass::setModulation ([eModulation](#) mode)

Setter for the modulation method of the module.

This function allows the user to set or update the **modulation method** used by the module by executing a "radio set mod <mode>" command on the module

Parameters:

<i>mode</i>	eModulation value representing the modulation mode (see constOrangeForRN2483.h for more information)
-------------	--

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

bool RadioCmdsClass::setOutputPower (int8_t pwrout)

Setter for the transceiver output power.

This function allows the user to set or update the **transceiver output power** by executing a "radio set pwr <pwrout>" command on the module

Parameters:

<i>pwrout</i>	Decimal number representing the transceiver output power, from -3 to 15
---------------	---

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

bool RadioCmdsClass::setSF ([eSpreadingFactor](#) spreadingFactor)

Setter for the spreading factor used during transmission.

This function allows the user to set or update the **spreading factor** used during transmission by executing a "radio set sf <spreadingFactor>" command on the module

Parameters:

<i>spreadingFactor</i>	eSpreadingFactor value representing the spreading factor (see constOrangeForRN2483.h for more information)
------------------------	--

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

Member Data Documentation**const char* RadioCmdsClass::params[\[COUNT_PARAM_RAD\]](#)[\[protected\]](#)**

```
Initial value:= {
    "rx",
    "tx",
    "cw",
    "bt",
    "mod",
    "freq",
    "pwr",
    "sf",
    "afcbw",
```

```
        "rxbw",  
        "bitrate",  
        "fdev",  
        "prlen",  
        "crc",  
        "iqi",  
        "cr",  
        "wdt",  
        "bw",  
        "snr",  
        "sync",  
    }  
}
```

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

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

RnRequestClass Class Reference

```
#include <RnRequest.h>
```

Public Member Functions

- [RnRequestClass](#) ()
Constructor for the [RnRequestClass](#) class.
- virtual [~RnRequestClass](#) ()
Destructor for the [RnRequestClass](#) class.

Protected Member Functions

- uint16_t [getReceivedData](#) ()
- uint16_t [readLn](#) (int8_t *buffer, uint16_t size)
- [eErrorType](#) [checkErrors](#) (int8_t *resp)
- bool [isStreamInit](#) ()
- void [init](#) ()
- bool [writeHexString](#) (const int8_t *paramValue, uint8_t lenParamValue)
- bool [cmdRequest](#) (uint8_t type, const char *command, const char *paramName)
- int8_t * [rnRequest](#) (uint8_t type, const char *command, const char *paramName, const int8_t *paramValue, uint8_t lenParamValue)
- int8_t * [rnRequest](#) (uint8_t type, const char *command, const char *paramName, const int8_t *paramValue, uint8_t lenParamValue, int8_t port)
- int8_t * [rnRequest](#) (uint8_t type, const char *command, const char *paramName=NULL, const char *paramValues=NULL)
- int8_t * [getResponse](#) (uint32_t timeout=[DEFAULT_TIMEOUT](#))
- [eSuccessType](#) [getLastSuccess](#) ()
- [eErrorType](#) [getLastError](#) ()
- void [setLastError](#) ([eErrorType](#) errorType)
- [eSuccessType](#) [checkSuccess](#) (int8_t *resp)

Protected Attributes

- [SerialType](#) * [loraStream](#)
- int8_t [receiveBuffer](#) [[DEFAULT_INPUT_BUFFER_SIZE](#)]
- [eSuccessType](#) [successType](#)
- [eErrorType](#) [errorType](#)
- const char * [params](#) [[COUNT_PARAM_MAC](#)]
- const char * [commandType](#) [[COUNT_TYPE](#)] = { "mac", "sys", "radio" }
- const char * [successResponses](#) [[LORA_COUNT_SUCCESS](#)]
- const char * [possibleResponses](#) [[LORA_COUNT_ERRORS](#)]

Friends

- class [OrangeForRN2483Class](#)
- class [RadioCmdsClass](#)
- class [SysCmdsClass](#)

Constructor & Destructor Documentation

RnRequestClass::RnRequestClass ()

Constructor for the [RnRequestClass](#) class.

Used to instantiate a new [RnRequestClass](#) object

RnRequestClass::~~RnRequestClass () [virtual]

Destructor for the [RnRequestClass](#) class.

Used to delete an [RnRequestClass](#) instance

Member Function Documentation

[eErrorType](#) RnRequestClass::checkErrors (int8_t * *resp*) [protected]

[eSuccessType](#) RnRequestClass::checkSuccess (int8_t * *resp*) [protected]

bool RnRequestClass::cmdRequest (uint8_t *type*, const char * *command*, const char * *paramName*) [protected]

[eErrorType](#) RnRequestClass::getLastError () [protected]

[eSuccessType](#) RnRequestClass::getLastSuccess () [protected]

uint16_t RnRequestClass::getReceivedData () [protected]

int8_t * RnRequestClass::getResponse (uint32_t *timeout* = [DEFAULT_TIMEOUT](#)) [protected]

void RnRequestClass::init () [protected]

bool RnRequestClass::isStreamInit () [protected]

uint16_t RnRequestClass::readLn (int8_t * *buffer*, uint16_t *size*) [protected]

int8_t * RnRequestClass::rnRequest (uint8_t *type*, const char * *command*, const char * *paramName*, const int8_t * *paramValue*, uint8_t *lenParamValue*) [protected]

int8_t * RnRequestClass::rnRequest (uint8_t *type*, const char * *command*, const char * *paramName*, const int8_t * *paramValue*, uint8_t *lenParamValue*, int8_t *port*) [protected]

int8_t * RnRequestClass::rnRequest (uint8_t *type*, const char * *command*, const char * *paramName* = NULL, const char * *paramValues* = NULL) [protected]

void RnRequestClass::setLastError ([eErrorType](#) *errorType*) [protected]

bool RnRequestClass::writeHexString (const int8_t * *paramValue*, uint8_t *lenParamValue*) [protected]

Friends And Related Function Documentation

friend class [OrangeForRN2483Class](#) [friend]

friend class [RadioCmdsClass](#) [friend]

friend class [SysCmdsClass](#) [friend]

Member Data Documentation

const char* RnRequestClass::commandType[[COUNT_TYPE](#)] = { "mac", "sys", "radio" } [protected]

[eErrorType](#) RnRequestClass::errorType [protected]

[SerialType](#)* RnRequestClass::loraStream [protected]

const char* RnRequestClass::params[[COUNT_PARAM_MAC](#)] [protected]

const char*
RnRequestClass::possibleResponses[[LORA_COUNT_ERRORS](#)] [protected]

```
Initial value:= {"invalid_param",  
                "keys not init",  
                "no_free_ch",  
                "silent",  
                "busy",  
                "mac paused",  
                "denied",  
                "invalid_data_len",  
                "frame_counter_err_rejoin_needed"  
                }
```

int8_t RnRequestClass::receiveBuffer[[DEFAULT_INPUT_BUFFER_SIZE](#)] [protected]

const char*
RnRequestClass::successResponses[[LORA_COUNT_SUCCESS](#)] [protected]

```
Initial value:= { "ok",  
                 "mac_tx_ok",  
                 "accepted",  
                 "mac_rx"  
                 }
```

[eSuccessType](#) RnRequestClass::successType [protected]

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

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

SysCmdsClass Class Reference

```
#include <SysCmds.h>
```

Public Member Functions

- String [getVersion](#) ()
Getter on the firmware version of the device.
- String [getNvm](#) (int8_t address[2])
Getter on the data stored in the user EEPROM of the module.
- bool [setNvm](#) (uint8_t address[2], uint8_t data[1])
Setter for the data stored in the user EEPROM of the module.
- String [getHardwareDevEUI](#) ()
Getter on the preprogrammed EUI address of the device.
- String [getPindig](#) (String pinname)
Getter on the state of a specified digital input.
- bool [setPinDig](#) (String pinname, String pinstate)
Setter for the state of a specified pin.
- String [getPinana](#) (String pinname)
Getter on the state of a specified analog input.
- int16_t [getVdd](#) ()
Getter on the Vdd value measured for the module.
- bool [setPinMode](#) (String pinname, String pinfunc)
Setter for the function of a specified pin.
- bool [sleep](#) (uint32_t delay)
Put the system to sleep mode for a specified number of milliseconds.
- String [reset](#) ()
Reset and restart the RN2483 module and return firmware version.

Protected Attributes

- const char * [params](#) [[COUNT_PARAM_SYS](#)]

Member Function Documentation

String SysCmdsClass::getHardwareDevEUI ()

Getter on the preprogrammed EUI address of the device.

This function allows the user to have access to the **preprogrammed EUI address** of the device by executing a "sys get hweui" command on the module

Returns:

The received DevEUI address as a *string* value

String SysCmdsClass::getNvm (int8_t address[2])

Getter on the data stored in the user EEPROM of the module.

This function allows the user to have access to the **data stored** in the **user EEPROM** of the device by executing a "sys get vnm <address>" command on the module

Parameters:

<i>address</i>	Hexadecimal number represented by an array whose size is 2 <code>int8_t</code>
----------------	--

Returns:

The received hexadecimal data stored at the address as a *string* value

String SysCmdsClass::getPinana (String *pinname*)

Getter on the state of a specified analog input.

This function allows the user to have access to the **state of a analog input** of the device by executing a "sys get pinana <pinname>" command on the module

Parameters:

<i>pinname</i>	String value representing the name of the analog input
----------------	--

Returns:

The received state as a *string* value, from 0 to 1023

String SysCmdsClass::getPindig (String *pinname*)

Getter on the state of a specified digital input.

This function allows the user to have access to the **state of a digital input** of the device by executing a "sys get pindig <pinname>" command on the module

Parameters:

<i>pinname</i>	String value representing the name of the digital input
----------------	---

Returns:

The received state as a *string* value, either 0 or 1

int16_t SysCmdsClass::getVdd ()

Getter on the Vdd value measured for the module.

This function allows the user to have access to an **ADC converted value** of Vdd measure by executing a "sys get vdd" command on the module

Returns:

The received converted value as a *decimal* value

String SysCmdsClass::getVersion ()

Getter on the firmware version of the device.

This function allows the user to have access to the **firmware version** of the device by executing a "sys get ver" command on the module

Returns:

The received firmware version as a *string* value

String SysCmdsClass::reset ()

Reset and restart the RN2483 module and return firmware version.

This function allows the user to **reset** and **restart** the RN2483 module and returns the current firmware information.

Returns:

String value corresponding to the received firmware information

bool SysCmdsClass::setNvm (uint8_t *address*[2], uint8_t *data*[1])

Setter for the data stored in the user EEPROM of the module.

This function allows the user to set or update the **data stored in the user EEPROM** used for by executing a "sys set nvm <address> <data>" command on the module

Parameters:

<i>address</i>	Hexadecimal number representing the user EEPROM address, from 300 to 3FF
<i>data</i>	Hexadecimal number representing the data to be stored at this address, from 00 to FF

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

bool SysCmdsClass::setPinDig (String *pinname*, String *pinstate*)

Setter for the state of a specified pin.

This function allows the user to set or update the **state** of a pin by executing a "sys set pinstate <pinname> <pinstate>" command on the module

Parameters:

<i>pinname</i>	String value representing the pin name ("GPIO0", "GPIO13", "UART_CTS", "UART_RTS")
<i>pinstate</i>	Decimal number representing the state (0 or 1)

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

bool SysCmdsClass::setPinMode (String *pinname*, String *pinfunc*)

Setter for the function of a specified pin.

This function allows the user to set or update the **function** of a pin by executing a "sys set pinmode <pinname> <pinfunc>" command on the module

Parameters:

<i>pinname</i>	String value representing the pin name ("GPIO0", "GPIO13", "UART_CTS", "UART_RTS")
<i>pinfunc</i>	String value representing the function of a pin ("digin", "digout", "ana")

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

bool SysCmdsClass::sleep (uint32_t *delay*)

Put the system to sleep mode for a specified number of milliseconds.

This function allows the user to put the module in a **sleep mode** for a specified number of milliseconds by executing a "sys sleep <delay>" command on the module

Parameters:

<i>delay</i>	Decimal number representing the number of milliseconds
--------------	--

Returns:

Boolean value, true if everything is ok, false if there was a problem during the execution

Member Data Documentation

const char* SysCmdsClass::params[COUNT_PARAM_SYS](#)**[protected]**

```
Initial value:= {  
    "ver",  
    "nvm",  
    "vdd",  
    "pindig",  
    "pinana",  
    "pinmode",  
    "hweui",  
    "sleep",  
    "reset",  
}
```

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

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

File Documentation

ConstOrangeForRN2483.h File Reference

Defines all the constant values used in the project.

Typedefs

- typedef enum [_typeMessage](#) [eTypeMessage](#)
*Different values for the **confirmer message** parameter.*
- typedef enum [_dataRate](#) [eDataRate](#)
*Different values for the **datarate** attribute.*
- typedef enum [_ePowerIdx](#) [ePowerIdx](#)
*Different values for the **power index** attribute.*
- typedef enum [_eBT](#) [eBT](#)
*Different values for the **data shaping value** applied to FSK transmissions.*
- typedef enum [_eBoolean](#) [eBoolean](#)
*Different values for a **boolean** value.*
- typedef enum [_eModulation](#) [eModulation](#)
*Different values for the **modulation type** attribute.*
- typedef enum [_eCodingRate](#) [eCodingRate](#)
*Different values for the **coding rate** attribute.*
- typedef enum [_eSpreadingFactor](#) [eSpreadingFactor](#)
*Different values for the **spreading factor** attribute.*
- typedef enum [_eErrorType](#) [eErrorType](#)
Different kind of error which could be encountered.

Enumerations

- enum [_typeMessage](#) { [UNCONFIRMED_MESSAGE](#) = 0, [CONFIRMED_MESSAGE](#) } *Different values for the **confirmer message** parameter.*
- enum [_dataRate](#) { [DATA_RATE_0](#) = 0, [DATA_RATE_1](#), [DATA_RATE_2](#), [DATA_RATE_3](#), [DATA_RATE_4](#), [DATA_RATE_5](#), [DATA_RATE_6](#), [DATA_RATE_7](#), [COUNT_DATA_RATE](#), [DATA_RATE_ERROR](#) = -1 } *Different values for the **datarate** attribute.*
- enum [_ePowerIdx](#) { [POWER_0](#) = 0, [POWER_1](#), [POWER_2](#), [POWER_3](#), [POWER_4](#), [POWER_5](#), [COUNT_POWER](#), [POWER_ERROR](#) = -1 } *Different values for the **power index** attribute.*
- enum [_eBT](#) { [BT_NONE](#) = 0, [BT_1_0](#), [BT_0_5](#), [BT_0_3](#), [BT_COUNT](#), [BT_ERROR](#) = -1 } *Different values for the **data shaping value** applied to FSK transmissions.*
- enum [_eBoolean](#) { [BOOL_FALSE](#) = 0, [BOOL_TRUE](#), [BOOL_ERROR](#) = -1 } *Different values for a **boolean** value.*
- enum [_eModulation](#) { [FSK_MODULATION](#) = 0, [LORA_MODULATION](#), [GET_MODULATION_ERROR](#) = -1 } *Different values for the **modulation type** attribute.*
- enum [_eCodingRate](#) { [CR_4_5](#) = 0, [CR_4_6](#), [CR_4_7](#), [CR_4_8](#), [CR_ERROR](#) = -1 } *Different values for the **coding rate** attribute.*
- enum [_eSpreadingFactor](#) { [SF7](#) = 7, [SF8](#), [SF9](#), [SF10](#), [SF11](#), [SF12](#), [SF_COUNT](#), [SF_ERROR](#) = -1 } *Different values for the **spreading factor** attribute.*
- enum [_eErrorType](#) { [LORA_SUCCESS](#) = 0, [LORA_INVALID_PARAM](#), [LORA_KEYS_NOT_INIT](#), [LORA_NO_FREE_CH](#), [LORA_SILENT](#), [LORA_BUSY](#), [LORA_MAC_PAUSED](#), [LORA_JOIN_DENIED](#), [LORA_INVALID_DATA_LEN](#), [LORA_ERR_FRAME_CNTR_ERR_REJOIN_NEEDED](#), [LORA_COUNT_ERRORS](#), [LORA_NOT_INIT](#), [LORA_NETWORK_NOT_JOINED](#), [LORA_TIMEOUT](#) } *Different kind of error which could be encountered.*

Detailed Description

Defines all the constant values used in the project.

This file defines all the constant variables and functions used in this project

Typedef Documentation

typedef enum [eBoolean](#) [eBoolean](#)

Different values for a **boolean** value.

Each of these values is used to guide the user when trying to set an attribute to true or false

typedef enum [eBT](#) [eBT](#)

Different values for the **data shaping value** applied to FSK transmissions.

Each of these values is used to guide the user when trying to set the data shaping value applied to FSK transmissions

typedef enum [eCodingRate](#) [eCodingRate](#)

Different values for the **coding rate** attribute.

Each of these values is used to guide the user when trying to set the coding rate value

typedef enum [dataRate](#) [eDataRate](#)

Different values for the **datarate** attribute.

Each of these values is used to guide the user when trying to set the datarate value

typedef enum [eErrorType](#) [eErrorType](#)

Different kind of error which could be encountered.

Each of these values is used to find the correct string value in the *possibleResponses* attribute of the OrangeForRn2483 class and other error cases

typedef enum [eModulation](#) [eModulation](#)

Different values for the **modulation type** attribute.

Each of these values is used to guide the user when trying to set the modulation type value

typedef enum [ePowerIdx](#) [ePowerIdx](#)

Different values for the **power index** attribute.

Each of these values is used to guide the user when trying to set the power index value

typedef enum [_eSpreadingFactor](#) eSpreadingFactor

Different values for the **spreading factor** attribute.

Each of these values is used to guide the user when trying to set the spreading factor value

typedef enum [_typeMessage](#) eTypeMessage

Different values for the **confirmer message** parameter.

Each of these values is used to guide the user when trying to send a message with a confirmed or unconfirmed message

Enumeration Type Documentation

enum [_dataRate](#)

Different values for the **datarate** attribute.

Each of these values is used to guide the user when trying to set the datarate value

Enumerator:

DATA_RATE_0	
DATA_RATE_1	
DATA_RATE_2	
DATA_RATE_3	
DATA_RATE_4	
DATA_RATE_5	
DATA_RATE_6	
DATA_RATE_7	
COUNT_DATA_RATE	
DATA_RATE_ERROR	

enum [_eBoolean](#)

Different values for a **boolean** value.

Each of these values is used to guide the user when trying to set an attribute to true or false

Enumerator:

BOOL_FALSE	
BOOL_TRUE	
BOOL_ERROR	

enum [_eBT](#)

Different values for the **data shaping value** applied to FSK transmissions.

Each of these values is used to guide the user when trying to set the data shaping value applied to FSK transmissions

Enumerator:

BT_NONE	
BT_1_0	
BT_0_5	
BT_0_3	
BT_COUNT	
BT_ERROR	

enum [_eCodingRate](#)

Different values for the **coding rate** attribute.

Each of these values is used to guide the user when trying to set the coding rate value

Enumerator:

CR_4_5	
CR_4_6	
CR_4_7	
CR_4_8	
CR_ERROR	

enum [_eErrorType](#)

Different kind of error which could be encountered.

Each of these values is used to find the correct string value in the *possibleResponses* attribute of the OrangeForRn2483 class and other error cases

Enumerator:

LORA_SUCCESS	
LORA_INVALID _PARAM	
LORA_KEYS_N OT_INIT	
LORA_NO_FREE _CH	
LORA_SILENT	
LORA_BUSY	
LORA_MAC_PA USED	
LORA_JOIN_DE NIED	
LORA_INVALID _DATA_LEN	
LORA_ERR_FRA ME_CNTR_ERR_ REJOIN_NEEDE D	
LORA_COUNT_ ERRORS	
LORA_NOT_INI T	

LORA_NETWORK_NOT_JOINED	
LORA_TIMEOUT	

enum [eModulation](#)

Different values for the **modulation type** attribute.

Each of these values is used to guide the user when trying to set the modulation type value

Enumerator:

FSK_MODULATION	
LORA_MODULATION	
GET_MODULATION_ERROR	

enum [ePowerIdx](#)

Different values for the **power index** attribute.

Each of these values is used to guide the user when trying to set the power index value

Enumerator:

POWER_0	
POWER_1	
POWER_2	
POWER_3	
POWER_4	
POWER_5	
COUNT_POWER	
POWER_ERROR	

enum [eSpreadingFactor](#)

Different values for the **spreading factor** attribute.

Each of these values is used to guide the user when trying to set the spreading factor value

Enumerator:

SF7	
SF8	
SF9	
SF10	
SF11	
SF12	
SF_COUNT	
SF_ERROR	

enum [typeMessage](#)

Different values for the **confirmer message** parameter.

Each of these values is used to guide the user when trying to send a message with a confirmed or unconfirmed message

Enumerator:

UNCONFIRMED _MESSAGE	
CONFIRMED_M ESSAGE	

DownlinkMessage.cpp File Reference

```
#include "DownlinkMessage.h"
```

DownlinkMessage.h File Reference

User's interface to access to the received data.

```
#include "InternalConstForRN2483.h"  
#include <Arduino.h>
```

Classes

- class [DownlinkMessage](#)
-

Detailed Description

User's interface to access to the received data.

This class contains all the necessary methods to be able to use and manipulate the data sent by the server after an uplink transmission.

InternalConstForRN2483.h File Reference

Macros

- #define [DEBUG](#)
 - #define [debugPrintLn](#)(X) SerialUSB.println((char*)X)
 - #define [debugPrint](#)(X) SerialUSB.print((char*)X)
 - #define [debugInt](#)(X) SerialUSB.print((int)X, HEX)
 - #define [debugIntLn](#)(X) SerialUSB.println((int)X, HEX)
 - #define [HEX_CHAR_TO_HIGH_NIBBLE](#)(X) (((X >= 'A') ? X - 'A' + 10 : X - '0') << 4)
 - #define [HEX_CHAR_TO_LOW_NIBBLE](#)(X) ((X >= 'A') ? X - 'A' + 10 : X - '0')
 - #define [NIBBLE_TO_HEX_CHAR](#)(i) ((i <= 9) ? ('0' + i) : ('A' - 10 + i))
 - #define [HIGH_NIBBLE](#)(i) ((i >> 4) & 0x0F)
 - #define [LOW_NIBBLE](#)(i) (i & 0x0F)
 - #define [IS_ON](#)(X) X.equals("on") ? [BOOL_TRUE](#) : [BOOL_FALSE](#)
 - #define [DEFAULT_TIMEOUT](#) 4000
 - #define [DEFAULT_TIMEOUT_2](#) ([DEFAULT_TIMEOUT](#) + 3000)
 - #define [JOIN_TIMEOUT_1](#) 5000
 - #define [JOIN_TIMEOUT_2](#) 10000
 - #define [DEFAULT_INPUT_BUFFER_SIZE](#) 64
 - #define [SEPARATOR](#) ((char*)" ")
 - #define [STR_OTAA](#) "otaa"
 - #define [STR_ABP](#) "abp"
 - #define [GET](#) "get"
 - #define [SET](#) "set"
 - #define [STR_MAC_RX](#) "mac_rx"
 - #define [STR_OK](#) "ok"
 - #define [STR_ON](#) "on"
 - #define [STR_OFF](#) "off"
 - #define [STR_CNF](#) "cnf"
 - #define [STR_UNCNF](#) "uncnf"
 - #define [CRLF](#) "\r\n"
 - #define [STR_ERROR_EMPTY_BUFFER](#) "Error : receiving buffer is empty"
 - #define [STR_ERROR_LORASTR_NOT_INIT](#) "Error : lorastream is not initialized"
 - #define [STR_ERROR_NETWORK_NOT_JOINED](#) "Error : network isn't joined yet"
 - #define [INT_ERROR_FAILED](#) -1
 - #define [FLT_ERROR_FAILED](#) -1.0
-

Macro Definition Documentation

#define CRLF `"\r\n"`

#define DEBUG

#define debugInt(X) `SerialUSB.print((int)X, HEX)`

#define debugIntLn(X) `SerialUSB.println((int)X, HEX)`

#define debugPrint(X) `SerialUSB.print((char*)X)`

#define debugPrintLn(X) `SerialUSB.println((char*)X)`

#define DEFAULT_INPUT_BUFFER_SIZE `64`

#define DEFAULT_TIMEOUT `4000`

#define DEFAULT_TIMEOUT_2 `(DEFAULT_TIMEOUT + 3000)`

#define FLT_ERROR_FAILED `-1.0`

#define GET `"get"`

#define HEX_CHAR_TO_HIGH_NIBBLE(X) `((X >= 'A') ? X - 'A' + 10 : X - '0') << 4)`

#define HEX_CHAR_TO_LOW_NIBBLE(X) `((X >= 'A') ? X - 'A' + 10 : X - '0')`

#define HIGH_NIBBLE(i) `((i >> 4) & 0x0F)`

#define INT_ERROR_FAILED `-1`

#define iS_ON(X) `X.equals("on") ? BOOL_TRUE : BOOL_FALSE`

#define JOIN_TIMEOUT_1 `5000`

#define JOIN_TIMEOUT_2 `10000`

#define LOW_NIBBLE(i) `(i & 0x0F)`

#define NIBBLE_TO_HEX_CHAR(i) `((i <= 9) ? ('0' + i) : ('A' - 10 + i))`

#define SEPARATOR `((char*)" ")`

#define SET `"set"`

#define STR_ABP `"abp"`

#define STR_CNF `"cnf"`

#define STR_ERROR_EMPTY_BUFFER `"Error : receiving buffer is empty"`

```
#define STR_ERROR_LORASTR_NOT_INIT "Error : lorastream is not initialized"

#define STR_ERROR_NETWORK_NOT_JOINED "Error : network isn't joined yet"

#define STR_MAC_RX "mac_rx"

#define STR_OFF "off"

#define STR_OK "ok"

#define STR_ON "on"

#define STR_OTAA "otaa"

#define STR_UNCNF "uncnf"
```

LpwaOrangeEncoder.cpp File Reference

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

Macros

- #define [MAX_LEN_PAYLOAD](#) 64
- #define [CHECK_COUNTER\(X\)](#) if((counter + (X - 1)) >= [MAX_LEN_PAYLOAD](#)) return false;

Variables

- [LpwaOrangeEncoderClass](#) [LpwaOrangeEncoder](#)
-

Macro Definition Documentation

```
#define CHECK_COUNTER( X)  if((counter + (X - 1)) >= MAX\_LEN\_PAYLOAD) return  
false;
```

```
#define MAX_LEN_PAYLOAD 64
```

Variable Documentation

[LpwaOrangeEncoderClass](#) [LpwaOrangeEncoder](#)

LpwaOrangeEncoder.h File Reference

Helpful methods to create, update or decode payloads.

Classes

- class [LpwaOrangeEncoderClass](#)

Variables

- [LpwaOrangeEncoderClass](#) [LpwaOrangeEncoder](#)
-

Detailed Description

Helpful methods to create, update or decode payloads.

This class contains all the necessary methods to be able to easily interact with a payload and modify it if necessary

Variable Documentation

[LpwaOrangeEncoderClass](#) [LpwaOrangeEncoder](#)

OrangeForRN2483.cpp File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include "OrangeForRN2483.h"
```

Variables

- [OrangeForRN2483Class](#) [OrangeForRN2483](#)
-

Variable Documentation

[OrangeForRN2483Class](#) OrangeForRN2483

OrangeForRN2483.h File Reference

User's main interface to use the module.

```
#include <Arduino.h>
#include "InternalConstForRN2483.h"
#include "ConstOrangeForRN2483.h"
#include "RadioCmds.h"
#include "SysCmds.h"
#include "LpwaOrangeEncoder.h"
#include "RnRequest.h"
#include "DownlinkMessage.h"
```

Classes

- class [OrangeForRN2483Class](#)

Variables

- [OrangeForRN2483Class](#) [OrangeForRN2483](#)
-

Detailed Description

User's main interface to use the module.

This class contains all the necessary methods to be able to easily use the Sdaq RN2483 module

Variable Documentation

[OrangeForRN2483Class](#) [OrangeForRN2483](#)

RadioCmds.cpp File Reference

```
#include "RadioCmds.h"  
#include "RnRequest.h"
```

RadioCmds.h File Reference

Contains all the RADIO commands.

```
#include "WProgram.h"
```

```
#include "ConstOrangeForRN2483.h"
```

Classes

- class [RadioCmdsClass](#)

Typedefs

- typedef enum [_paramRad](#) [eParamRad](#)
Different kind of RADIO commands.

Enumerations

- enum [_paramRad](#) { [RX](#) = 0, [TX_RADIO](#), [CW](#), [BT](#), [MOD](#), [FREQ](#), [PWR](#), [SPR_FACTOR](#), [AUTO_FREQ_CORR_BW](#), [RECEIVE_BW](#), [BIT_RATE](#), [FREQ_DEVIATION](#), [PREAMBLE_LENGTH](#), [CRC](#), [IQ_INVERS](#), [CODING_RATE](#), [WATCHDOG_TIMER](#), [BANDWIDTH](#), [SIG_NOISE_RATIO](#), [SYNC_RADIO](#), [COUNT_PARAM_RAD](#) } *Different kind of RADIO commands.*

Detailed Description

Contains all the RADIO commands.

This class defines all the RADIO commands available for the user, such as getters and setters

Typedef Documentation

typedef enum [_paramRad](#) [eParamRad](#)

Different kind of RADIO commands.

Each of these values is used to find the correct string value in the *params* attribute of the class

Enumeration Type Documentation

enum [_paramRad](#)

Different kind of RADIO commands.

Each of these values is used to find the correct string value in the *params* attribute of the class

Enumerator:

RX	
TX_RADIO	
CW	
BT	

MOD	
FREQ	
PWR	
SPR_FACTOR	
AUTO_FREQ_CO RR_BW	
RECEIVE_BW	
BIT_RATE	
FREQ_DEVIATI ON	
PREAMBLE_LE NGTH	
CRC	
IQ_INVERS	
CODING_RATE	
WATCHDOG_TI MER	
BANDWIDTH	
SIG_NOISE_RAT IO	
SYNC_RADIO	
COUNT_PARAM _RAD	

RnRequest.cpp File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include "RnRequest.h"
```

Variables

- [RnRequestClass](#) [RnRequest](#)
-

Variable Documentation

[RnRequestClass](#) [RnRequest](#)

RnRequest.h File Reference

Main methods used by the library to communicate with the module.

```
#include <Arduino.h>
#include "InternalConstForRN2483.h"
#include "ConstOrangeForRN2483.h"
```

Classes

- class [RnRequestClass](#)

Typedefs

- typedef Stream [SerialType](#)
- typedef enum [_typecmd](#) [eTypeCommand](#)
Different kind of commands which could be sent to the Rn2483 module.
- typedef enum [_paramMac](#) [eParamMac](#)
Different kind of MAC commands.
- typedef enum [_eSuccessType](#) [eSuccessType](#)
Different kind of success message.

Enumerations

- enum [_typecmd](#) { [MAC](#) = 0, [SYS](#), [RADIO](#), [COUNT_TYPE](#) } *Different kind of commands which could be sent to the Rn2483 module.*
- enum [_paramMac](#) { [DEVADDR](#) = 0, [DEVEUI](#), [APPEUI](#), [BAND](#), [DATARATE](#), [PWR_IND_VAL](#), [ADR](#), [RETRANS_NB](#), [RX_DELAY_1](#), [RX_DELAY_2](#), [AUTO_REPLY](#), [RX2_D_CYCLE_PS](#), [DEMOD_MARGIN](#), [GATEWAY_NB](#), [STATUS](#), [SYNC](#), [UP_CTR](#), [DWN_CTR](#), [NWKS_KEY](#), [APPS_KEY](#), [APP_KEY](#), [JOIN](#), [TX_MAC](#), [BAT_LVL](#), [LINK_CHECK](#), [SAVE](#), [PAUSE](#), [RESUME](#), [COUNT_PARAM_MAC](#) } *Different kind of MAC commands.*
- enum [_eSuccessType](#) { [LORA_OK](#) = 0, [LORA_MAC_TX_OK](#), [LORA_ACCEPTED](#), [LORA_RX](#), [LORA_COUNT_SUCCESS](#), [LORA_FAILED](#) } *Different kind of success message.*

Variables

- [RnRequestClass](#) [RnRequest](#)

Detailed Description

Main methods used by the library to communicate with the module.

This class contains all the necessary methods to be able to easily communicate the Sodaq RN2483 module

Typedef Documentation

typedef enum [_paramMac](#) [eParamMac](#)

Different kind of MAC commands.

Each of these values is used to find the correct string value in the *params* attribute of the class

typedef enum [_eSuccessType](#) eSuccessType

Different kind of success message.

Each of these values is used to find the correct string value in the *successResponses* attribute of the class

typedef enum [_typecmd](#) eTypeCommand

Different kind of commands which could be sent to the Rn2483 module.

Each of these values is used to find the correct string value in the *commandType* attribute of the class

typedef Stream [SerialType](#)

Enumeration Type Documentation

enum [_eSuccessType](#)

Different kind of success message.

Each of these values is used to find the correct string value in the *successResponses* attribute of the class

Enumerator:

LORA_OK	
LORA_MAC_TX _OK	
LORA_ACCEPTE D	
LORA_RX	
LORA_COUNT_S UCCESS	
LORA_FAILED	

enum [_paramMac](#)

Different kind of MAC commands.

Each of these values is used to find the correct string value in the *params* attribute of the class

Enumerator:

DEVADDR	
DEVEUI	
APPEUI	
BAND	
DATARATE	
PWR_IND_VAL	
ADR	
RETRANS_NB	
RX_DELAY_1	

RX_DELAY_2	
AUTO_REPLY	
RX2	
D_CYCLE_PS	
DEMOD_MARGIN	
GATEWAY_NB	
STATUS	
SYNC	
UP_CTR	
DWN_CTR	
NWKS_KEY	
APPS_KEY	
APP_KEY	
JOIN	
TX_MAC	
BAT_LVL	
LINK_CHECK	
SAVE	
PAUSE	
RESUME	
COUNT_PARAM_MAC	

enum [typecmd](#)

Different kind of commands which could be sent to the Rn2483 module.

Each of these values is used to find the correct string value in the *commandType* attribute of the class

Enumerator:

MAC	
SYS	
RADIO	
COUNT_TYPE	

Variable Documentation

[RnRequestClass](#) RnRequest

SysCmds.cpp File Reference

```
#include "SysCmds.h"  
#include "RnRequest.h"
```

SysCmds.h File Reference

Contains all the SYS commands.
`#include "WProgram.h"`

Classes

- class [SysCmdsClass](#)

Typedefs

- typedef enum [_paramSys](#) [eParamSys](#)
Different kind of SYS commands.

Enumerations

- enum [_paramSys](#) { [VERSION](#) = 0, [NVM](#), [VDD](#), [PIN_DIG](#), [PIN_ANA](#), [PIN_MODE](#), [HWEUI](#), [SLEEP](#), [RESET](#), [COUNT_PARAM_SYS](#) } *Different kind of SYS commands.*

Detailed Description

Contains all the SYS commands.

This class defines all the SYS commands available for the user : getters, setters or other specific commands such as sleep.

Typedef Documentation

typedef enum [_paramSys](#) [eParamSys](#)

Different kind of SYS commands.

Each of these values is used to find the correct string value in the *params* attribute of the class

Enumeration Type Documentation

enum [_paramSys](#)

Different kind of SYS commands.

Each of these values is used to find the correct string value in the *params* attribute of the class

Enumerator:

VERSION	
NVM	
VDD	
PIN_DIG	
PIN_ANA	
PIN_MODE	
HWEUI	

SLEEP	
RESET	
COUNT_PARAM _SYS	

Index

- `_dataRate`
 - `ConstOrangeForRN2483.h`, 42
- `_eBoolean`
 - `ConstOrangeForRN2483.h`, 42
- `_eBT`
 - `ConstOrangeForRN2483.h`, 42
- `_eCodingRate`
 - `ConstOrangeForRN2483.h`, 43
- `_eErrorType`
 - `ConstOrangeForRN2483.h`, 43
- `_eModulation`
 - `ConstOrangeForRN2483.h`, 44
- `_ePowerIdx`
 - `ConstOrangeForRN2483.h`, 44
- `_eSpreadingFactor`
 - `ConstOrangeForRN2483.h`, 44
- `_eSuccessType`
 - `RnRequest.h`, 60
- `_paramMac`
 - `RnRequest.h`, 60
- `_paramRad`
 - `RadioCmds.h`, 56
- `_paramSys`
 - `SysCmds.h`, 63
- `_typecmd`
 - `RnRequest.h`, 61
- `_typeMessage`
 - `ConstOrangeForRN2483.h`, 44
- `~DownlinkMessage`
 - `DownlinkMessage`, 5
- `~LpwaOrangeEncoderClass`
 - `LpwaOrangeEncoderClass`, 7
- `~OrangeForRN2483Class`
 - `OrangeForRN2483Class`, 13
- `~RnRequestClass`
 - `RnRequestClass`, 34
- `addBool`
 - `LpwaOrangeEncoderClass`, 8
- `addByte`
 - `LpwaOrangeEncoderClass`, 8
- `addFloat`
 - `LpwaOrangeEncoderClass`, 8
- `addInt`
 - `LpwaOrangeEncoderClass`, 8
- `addLong`
 - `LpwaOrangeEncoderClass`, 8
- `addShort`
 - `LpwaOrangeEncoderClass`, 9
- `addUByte`
 - `LpwaOrangeEncoderClass`, 9
- `addUInt`
 - `LpwaOrangeEncoderClass`, 9
- `addULong`
 - `LpwaOrangeEncoderClass`, 9
- `addUShort`
 - `LpwaOrangeEncoderClass`, 9

- `ADR`
 - `RnRequest.h`, 60
- `APP_KEY`
 - `RnRequest.h`, 61
- `APPEUI`
 - `RnRequest.h`, 60
- `APPS_KEY`
 - `RnRequest.h`, 61
- `AUTO_FREQ_CORR_BW`
 - `RadioCmds.h`, 57
- `AUTO_REPLY`
 - `RnRequest.h`, 61
- `BAND`
 - `RnRequest.h`, 60
- `BANDWIDTH`
 - `RadioCmds.h`, 57
- `BAT_LVL`
 - `RnRequest.h`, 61
- `BIT_RATE`
 - `RadioCmds.h`, 57
- `BOOL_ERROR`
 - `ConstOrangeForRN2483.h`, 42
- `BOOL_FALSE`
 - `ConstOrangeForRN2483.h`, 42
- `BOOL_TRUE`
 - `ConstOrangeForRN2483.h`, 42
- `BT`
 - `RadioCmds.h`, 56
- `BT_0_3`
 - `ConstOrangeForRN2483.h`, 43
- `BT_0_5`
 - `ConstOrangeForRN2483.h`, 43
- `BT_1_0`
 - `ConstOrangeForRN2483.h`, 43
- `BT_COUNT`
 - `ConstOrangeForRN2483.h`, 43
- `BT_ERROR`
 - `ConstOrangeForRN2483.h`, 43
- `BT_NONE`
 - `ConstOrangeForRN2483.h`, 43
- `CHECK_COUNTER`
 - `LpwaOrangeEncoder.cpp`, 51
- `checkErrors`
 - `RnRequestClass`, 34
- `checkSuccess`
 - `RnRequestClass`, 34
- `cmdRequest`
 - `RnRequestClass`, 34
- `CODING_RATE`
 - `RadioCmds.h`, 57
- `commandType`
 - `OrangeForRN2483Class`, 24
 - `RnRequestClass`, 35
- `CONFIRMED_MESSAGE`
 - `ConstOrangeForRN2483.h`, 45
- `ConstOrangeForRN2483.h`, 40

_dataRate, 42
 _eBoolean, 42
 _eBT, 42
 _eCodingRate, 43
 _eErrorType, 43
 _eModulation, 44
 _ePowerIdx, 44
 _eSpreadingFactor, 44
 _typeMessage, 44
 BOOL_ERROR, 42
 BOOL_FALSE, 42
 BOOL_TRUE, 42
 BT_0_3, 43
 BT_0_5, 43
 BT_1_0, 43
 BT_COUNT, 43
 BT_ERROR, 43
 BT_NONE, 43
 CONFIRMED_MESSAGE, 45
 COUNT_DATA_RATE
 ConstOrangeForRN2483.h, 42
 COUNT_PARAM_MAC
 RnRequest.h, 61
 COUNT_PARAM_RAD
 RadioCmds.h, 57
 COUNT_PARAM_SYS
 SysCmds.h, 64
 COUNT_POWER
 ConstOrangeForRN2483.h, 44
 COUNT_TYPE
 RnRequest.h, 61
 CR_4_5
 ConstOrangeForRN2483.h, 43
 CR_4_6
 ConstOrangeForRN2483.h, 43
 CR_4_7
 ConstOrangeForRN2483.h, 43
 CR_4_8
 ConstOrangeForRN2483.h, 43
 CR_ERROR
 ConstOrangeForRN2483.h, 43
 CRC
 RadioCmds.h, 57
 CRLF
 InternalConstForRN2483.h, 49
 CW
 RadioCmds.h, 56
 D_CYCLE_PS
 RnRequest.h, 61
 DATA_RATE_0
 ConstOrangeForRN2483.h, 42
 DATA_RATE_1
 ConstOrangeForRN2483.h, 42
 DATA_RATE_2
 ConstOrangeForRN2483.h, 42
 DATA_RATE_3
 ConstOrangeForRN2483.h, 42
 DATA_RATE_4
 ConstOrangeForRN2483.h, 42
 DATA_RATE_5
 ConstOrangeForRN2483.h, 42
 LORA_SUCCESS, 43
 LORA_TIMEOUT, 44
 POWER_0, 44
 POWER_1, 44
 POWER_2, 44
 POWER_3, 44
 POWER_4, 44
 POWER_5, 44
 POWER_ERROR, 44
 SF_COUNT, 44
 SF_ERROR, 44
 SF10, 44
 SF11, 44
 SF12, 44
 SF7, 44
 SF8, 44
 SF9, 44
 UNCONFIRMED_MESSAGE, 45
 COUNT_DATA_RATE
 ConstOrangeForRN2483.h, 42
 COUNT_PARAM_MAC
 RnRequest.h, 61
 COUNT_PARAM_RAD
 RadioCmds.h, 57
 COUNT_PARAM_SYS
 SysCmds.h, 64
 COUNT_POWER
 ConstOrangeForRN2483.h, 44
 COUNT_TYPE
 RnRequest.h, 61
 CR_4_5
 ConstOrangeForRN2483.h, 43
 CR_4_6
 ConstOrangeForRN2483.h, 43
 CR_4_7
 ConstOrangeForRN2483.h, 43
 CR_4_8
 ConstOrangeForRN2483.h, 43
 CR_ERROR
 ConstOrangeForRN2483.h, 43
 CRC
 RadioCmds.h, 57
 CRLF
 InternalConstForRN2483.h, 49
 CW
 RadioCmds.h, 56
 D_CYCLE_PS
 RnRequest.h, 61
 DATA_RATE_0
 ConstOrangeForRN2483.h, 42
 DATA_RATE_1
 ConstOrangeForRN2483.h, 42
 DATA_RATE_2
 ConstOrangeForRN2483.h, 42
 DATA_RATE_3
 ConstOrangeForRN2483.h, 42
 DATA_RATE_4
 ConstOrangeForRN2483.h, 42
 DATA_RATE_5
 ConstOrangeForRN2483.h, 42

- DATA_RATE_6
 - ConstOrangeForRN2483.h, 42
- DATA_RATE_7
 - ConstOrangeForRN2483.h, 42
- DATA_RATE_ERROR
 - ConstOrangeForRN2483.h, 42
- DATARATE
 - RnRequest.h, 60
- DEBUG
 - InternalConstForRN2483.h, 49
- debugInt
 - InternalConstForRN2483.h, 49
- debugIntLn
 - InternalConstForRN2483.h, 49
- debugPrint
 - InternalConstForRN2483.h, 49
- debugPrintLn
 - InternalConstForRN2483.h, 49
- DEFAULT_INPUT_BUFFER_SIZE
 - InternalConstForRN2483.h, 49
- DEFAULT_TIMEOUT
 - InternalConstForRN2483.h, 49
- DEFAULT_TIMEOUT_2
 - InternalConstForRN2483.h, 49
- DEMOD_MARGIN
 - RnRequest.h, 61
- DEVADDR
 - RnRequest.h, 60
- DEVEUI
 - RnRequest.h, 60
- diagStream
 - OrangeForRN2483Class, 24
- downlinkMessage
 - OrangeForRN2483Class, 24
- DownlinkMessage, 5
 - ~DownlinkMessage, 5
 - DownlinkMessage, 5
 - getMessage, 5
 - getMessageByteArray, 6
 - getPort, 6
 - OrangeForRN2483Class, 6
 - setPort, 6
 - setResponseMessage, 6
- DownlinkMessage.cpp, 46
- DownlinkMessage.h, 47
- DWN_CTR
 - RnRequest.h, 61
- eBoolean
 - ConstOrangeForRN2483.h, 41
- eBT
 - ConstOrangeForRN2483.h, 41
- eCodingRate
 - ConstOrangeForRN2483.h, 41
- eDataRate
 - ConstOrangeForRN2483.h, 41
- eErrorType
 - ConstOrangeForRN2483.h, 41
- eModulation
 - ConstOrangeForRN2483.h, 41
- enableAdr
 - OrangeForRN2483Class, 14
- eParamMac
 - RnRequest.h, 59
- eParamRad
 - RadioCmds.h, 56
- eParamSys
 - SysCmds.h, 63
- ePowerIdx
 - ConstOrangeForRN2483.h, 41
- errorType
 - RnRequestClass, 35
- eSpreadingFactor
 - ConstOrangeForRN2483.h, 42
- eSuccessType
 - RnRequest.h, 60
- eTypeCommand
 - RnRequest.h, 60
- eTypeMessage
 - ConstOrangeForRN2483.h, 42
- FLT_ERROR_FAILED
 - InternalConstForRN2483.h, 49
- flush
 - LpwaOrangeEncoderClass, 10
- FREQ
 - RadioCmds.h, 57
- FREQ_DEVIATION
 - RadioCmds.h, 57
- FSK_MODULATION
 - ConstOrangeForRN2483.h, 44
- GATEWAY_NB
 - RnRequest.h, 61
- GET
 - InternalConstForRN2483.h, 49
- GET_MODULATION_ERROR
 - ConstOrangeForRN2483.h, 44
- getAppEUI
 - OrangeForRN2483Class, 14
- getAutoFreqCorrBw
 - RadioCmdsClass, 27
- getAutoReply
 - OrangeForRN2483Class, 14
- getBand
 - OrangeForRN2483Class, 14
- getBandWidth
 - RadioCmdsClass, 27
- getBitRate
 - RadioCmdsClass, 27
- getBt
 - RadioCmdsClass, 27
- getCodingRate
 - RadioCmdsClass, 27
- getCrc
 - RadioCmdsClass, 28
- getDataRate
 - OrangeForRN2483Class, 14
- getDCyclePs
 - OrangeForRN2483Class, 14
- getDemodMargin
 - OrangeForRN2483Class, 15
- getDevAddr

- OrangeForRN2483Class, 15
- getDevEUI
 - OrangeForRN2483Class, 15
- getDownlinkMessage
 - OrangeForRN2483Class, 15
- getDwnctr
 - OrangeForRN2483Class, 15
- getFramePayload
 - LpwaOrangeEncoderClass, 10
- getFreqDeviation
 - RadioCmdsClass, 28
- getFrequency
 - RadioCmdsClass, 28
- getGatewayNb
 - OrangeForRN2483Class, 16
- getHardwareDevEUI
 - SysCmdsClass, 36
- getLqInversion
 - RadioCmdsClass, 28
- getJoinState
 - OrangeForRN2483Class, 16
- getLastError
 - OrangeForRN2483Class, 16
 - RnRequestClass, 34
- getLastSuccess
 - RnRequestClass, 34
- getMessage
 - DownlinkMessage, 5
- getMessageByteArray
 - DownlinkMessage, 6
- getModulation
 - RadioCmdsClass, 28
- getNvm
 - SysCmdsClass, 36
- getOutputPower
 - RadioCmdsClass, 29
- getPinana
 - SysCmdsClass, 37
- getPindig
 - SysCmdsClass, 37
- getPort
 - DownlinkMessage, 6
- getPreambleLength
 - RadioCmdsClass, 29
- getPwrIdxValue
 - OrangeForRN2483Class, 16
- getRadioCmds
 - OrangeForRN2483Class, 16
- getReceiveBw
 - RadioCmdsClass, 29
- getReceivedData
 - RnRequestClass, 34
- getResponse
 - RnRequestClass, 34
- getRetransNb
 - OrangeForRN2483Class, 16
- getRx2
 - OrangeForRN2483Class, 17
- getRxdelay1
 - OrangeForRN2483Class, 17
- getRxdelay2
 - OrangeForRN2483Class, 17
- getSF
 - RadioCmdsClass, 29
- getSigNoiseRation
 - RadioCmdsClass, 29
- getStatus
 - OrangeForRN2483Class, 17
- getSync
 - OrangeForRN2483Class, 17
 - RadioCmdsClass, 30
- getSysCmds
 - OrangeForRN2483Class, 17
- getUpctr
 - OrangeForRN2483Class, 18
- getVdd
 - SysCmdsClass, 37
- getVersion
 - SysCmdsClass, 37
- getWatchdog
 - RadioCmdsClass, 30
- HEX_CHAR_TO_HIGH_NIBBLE
 - InternalConstForRN2483.h, 49
- HEX_CHAR_TO_LOW_NIBBLE
 - InternalConstForRN2483.h, 49
- HIGH_NIBBLE
 - InternalConstForRN2483.h, 49
- HWEUI
 - SysCmds.h, 63
- init
 - OrangeForRN2483Class, 18
 - RnRequestClass, 34
- INT_ERROR_FAILED
 - InternalConstForRN2483.h, 49
- InternalConstForRN2483.h, 48
 - CRLF, 49
 - DEBUG, 49
 - debugInt, 49
 - debugIntLn, 49
 - debugPrint, 49
 - debugPrintLn, 49
 - DEFAULT_INPUT_BUFFER_SIZE, 49
 - DEFAULT_TIMEOUT, 49
 - DEFAULT_TIMEOUT_2, 49
 - FLT_ERROR_FAILED, 49
 - GET, 49
 - HEX_CHAR_TO_HIGH_NIBBLE, 49
 - HEX_CHAR_TO_LOW_NIBBLE, 49
 - HIGH_NIBBLE, 49
 - INT_ERROR_FAILED, 49
 - iS_ON, 49
 - JOIN_TIMEOUT_1, 49
 - JOIN_TIMEOUT_2, 49
 - LOW_NIBBLE, 49
 - NIBBLE_TO_HEX_CHAR, 49
 - SEPARATOR, 49
 - SET, 49
 - STR_ABP, 49
 - STR_CNF, 49
 - STR_ERROR_EMPTY_BUFFER, 49

- STR_ERROR_LORASTR_NOT_INIT, 50
- STR_ERROR_NETWORK_NOT_JOINED, 50
- STR_MAC_RX, 50
- STR_OFF, 50
- STR_OK, 50
- STR_ON, 50
- STR_OTAA, 50
- STR_UNCNF, 50
- IQ_INVERS
 - RadioCmds.h, 57
- is_ON
 - InternalConstForRN2483.h, 49
- isAdr
 - OrangeForRN2483Class, 18
- isNetworkJoined
 - OrangeForRN2483Class, 24
- isStreamInit
 - OrangeForRN2483Class, 18
 - RnRequestClass, 34
- join
 - OrangeForRN2483Class, 18
- JOIN
 - RnRequest.h, 61
- JOIN_TIMEOUT_1
 - InternalConstForRN2483.h, 49
- JOIN_TIMEOUT_2
 - InternalConstForRN2483.h, 49
- joinNetwork
 - OrangeForRN2483Class, 18, 19
- LINK_CHECK
 - RnRequest.h, 61
- LORA_ACCEPTED
 - RnRequest.h, 60
- LORA_BUSY
 - ConstOrangeForRN2483.h, 43
- LORA_COUNT_ERRORS
 - ConstOrangeForRN2483.h, 43
- LORA_COUNT_SUCCESS
 - RnRequest.h, 60
- LORA_ERR_FRAME_CNTR_ERR_REJOIN_NEED
 - ConstOrangeForRN2483.h, 43
- LORA_FAILED
 - RnRequest.h, 60
- LORA_INVALID_DATA_LEN
 - ConstOrangeForRN2483.h, 43
- LORA_INVALID_PARAM
 - ConstOrangeForRN2483.h, 43
- LORA_JOIN_DENIED
 - ConstOrangeForRN2483.h, 43
- LORA_KEYS_NOT_INIT
 - ConstOrangeForRN2483.h, 43
- LORA_MAC_PAUSED
 - ConstOrangeForRN2483.h, 43
- LORA_MAC_TX_OK
 - RnRequest.h, 60
- LORA_MODULATION
 - ConstOrangeForRN2483.h, 44
- LORA_NETWORK_NOT_JOINED
 - ConstOrangeForRN2483.h, 44
- LORA_NO_FREE_CH
 - ConstOrangeForRN2483.h, 43
- LORA_NOT_INIT
 - ConstOrangeForRN2483.h, 43
- LORA_OK
 - RnRequest.h, 60
- LORA_RX
 - RnRequest.h, 60
- LORA_SILENT
 - ConstOrangeForRN2483.h, 43
- LORA_SUCCESS
 - ConstOrangeForRN2483.h, 43
- LORA_TIMEOUT
 - ConstOrangeForRN2483.h, 44
- loraStream
 - RnRequestClass, 35
- LOW_NIBBLE
 - InternalConstForRN2483.h, 49
- LpwaOrangeEncoder
 - LpwaOrangeEncoder.cpp, 51
 - LpwaOrangeEncoder.h, 52
- LpwaOrangeEncoder.cpp, 51
 - CHECK_COUNTER, 51
 - LpwaOrangeEncoder, 51
 - MAX_LEN_PAYLOAD, 51
- LpwaOrangeEncoder.h, 52
 - LpwaOrangeEncoder, 52
- LpwaOrangeEncoderClass, 7
 - ~LpwaOrangeEncoderClass, 7
 - addBool, 8
 - addByte, 8
 - addFloat, 8
 - addInt, 8
 - addLong, 8
 - addShort, 9
 - addUByte, 9
 - addUInt, 9
 - addULong, 9
 - addUShort, 9
 - flush, 10
 - getFramePayload, 10
 - LpwaOrangeEncoderClass, 7
- MAC
 - RnRequest.h, 61
- MAX_LEN_PAYLOAD
 - LpwaOrangeEncoder.cpp, 51
- MOD
 - RadioCmds.h, 57
- NIBBLE_TO_HEX_CHAR
 - InternalConstForRN2483.h, 49
- NVM
 - SysCmds.h, 63
- NWKS_KEY
 - RnRequest.h, 61
- OrangeForRN2483
 - OrangeForRN2483.cpp, 53
 - OrangeForRN2483.h, 54
 - OrangeForRN2483.cpp, 53
 - OrangeForRN2483, 53

- OrangeForRN2483.h, 54
 - OrangeForRN2483, 54
- OrangeForRN2483Class, 11
 - ~OrangeForRN2483Class, 13
 - commandType, 24
 - diagStream, 24
 - downlinkMessage, 24
 - DownlinkMessage, 6
 - enableAdr, 14
 - getAppEUI, 14
 - getAutoReply, 14
 - getBand, 14
 - getDataRate, 14
 - getDCyclePs, 14
 - getDemodMargin, 15
 - getDevAddr, 15
 - getDevEUI, 15
 - getDownlinkMessage, 15
 - getDwnctr, 15
 - getGatewayNb, 16
 - getJoinState, 16
 - getLastError, 16
 - getPwrIdxValue, 16
 - getRadioCmds, 16
 - getRetransNb, 16
 - getRx2, 17
 - getRxdelay1, 17
 - getRxdelay2, 17
 - getStatus, 17
 - getSync, 17
 - getSysCmds, 17
 - getUpctr, 18
 - init, 18
 - isAdr, 18
 - isNetworkJoined, 24
 - isStreamInit, 18
 - join, 18
 - joinNetwork, 18, 19
 - OrangeForRN2483Class, 13
 - params, 24
 - pause, 19
 - possibleResponses, 24
 - RadioCmds, 25
 - resetDevice, 19
 - resume, 19
 - RnRequestClass, 35
 - save, 19
 - sendMessage, 19, 20
 - setAbpKeys, 20
 - setAppEUI, 20
 - setAppKey, 20
 - setAppSKey, 21
 - setAutoReply, 21
 - setBatLvl, 21
 - setDataRate, 21
 - setDevAddr, 21
 - setDevEUI, 22
 - setDwnctr, 22
 - setLastError, 22
 - setLinkCheck, 22
 - setNwkSKey, 22
 - setOttaKeys, 23
 - setPwrIdx, 23
 - setRetx, 23
 - setRx2, 23
 - setRxDelay1, 23
 - setSync, 24
 - setUpctr, 24
 - SysCmds, 25
 - tx, 24
- params
 - OrangeForRN2483Class, 24
 - RadioCmdsClass, 31
 - RnRequestClass, 35
 - SysCmdsClass, 39
- pause
 - OrangeForRN2483Class, 19
- PAUSE
 - RnRequest.h, 61
- PIN_ANA
 - SysCmds.h, 63
- PIN_DIG
 - SysCmds.h, 63
- PIN_MODE
 - SysCmds.h, 63
- possibleResponses
 - OrangeForRN2483Class, 24
 - RnRequestClass, 35
- POWER_0
 - ConstOrangeForRN2483.h, 44
- POWER_1
 - ConstOrangeForRN2483.h, 44
- POWER_2
 - ConstOrangeForRN2483.h, 44
- POWER_3
 - ConstOrangeForRN2483.h, 44
- POWER_4
 - ConstOrangeForRN2483.h, 44
- POWER_5
 - ConstOrangeForRN2483.h, 44
- POWER_ERROR
 - ConstOrangeForRN2483.h, 44
- PREAMBLE_LENGTH
 - RadioCmds.h, 57
- PWR
 - RadioCmds.h, 57
- PWR_IND_VAL
 - RnRequest.h, 60
- RADIO
 - RnRequest.h, 61
- RadioCmds
 - OrangeForRN2483Class, 25
- RadioCmds.cpp, 55
- RadioCmds.h, 56
 - _paramRad, 56
 - AUTO_FREQ_CORR_BW, 57
 - BANDWIDTH, 57
 - BIT_RATE, 57
 - BT, 56
 - CODING_RATE, 57

- COUNT_PARAM_RAD, 57
- CRC, 57
- CW, 56
- eParamRad, 56
- FREQ, 57
- FREQ_DEVIATION, 57
- IQ_INVERS, 57
- MOD, 57
- PREAMBLE_LENGTH, 57
- PWR, 57
- RECEIVE_BW, 57
- RX, 56
- SIG_NOISE_RATIO, 57
- SPR_FACTOR, 57
- SYNC_RADIO, 57
- TX_RADIO, 56
- WATCHDOG_TIMER, 57
- RadioCmdsClass, 26
 - getAutoFreqCorrBw, 27
 - getBandWidth, 27
 - getBitRate, 27
 - getBt, 27
 - getCodingRate, 27
 - getCrc, 28
 - getFreqDeviation, 28
 - getFrequency, 28
 - getIqInversion, 28
 - getModulation, 28
 - getOutputPower, 29
 - getPreambleLength, 29
 - getReceiveBw, 29
 - getSF, 29
 - getSigNoiseRation, 29
 - getSync, 30
 - getWatchdog, 30
 - params, 31
 - RnRequestClass, 35
 - setAutoFreqBand, 30
 - setBt, 30
 - setFrequency, 30
 - setModulation, 31
 - setOutputPower, 31
 - setSF, 31
- readLn
 - RnRequestClass, 34
- RECEIVE_BW
 - RadioCmds.h, 57
- receiveBuffer
 - RnRequestClass, 35
- reset
 - SysCmdsClass, 37
- RESET
 - SysCmds.h, 64
- resetDevice
 - OrangeForRN2483Class, 19
- resume
 - OrangeForRN2483Class, 19
- RESUME
 - RnRequest.h, 61
- RETRANS_NB
 - RnRequest.h, 60
- rnRequest
 - RnRequestClass, 34
- RnRequest
 - RnRequest.cpp, 58
 - RnRequest.h, 61
- RnRequest.cpp, 58
 - RnRequest, 58
- RnRequest.h, 59
 - _eSuccessType, 60
 - _paramMac, 60
 - _typecmd, 61
 - ADR, 60
 - APP_KEY, 61
 - APPEUI, 60
 - APPS_KEY, 61
 - AUTO_REPLY, 61
 - BAND, 60
 - BAT_LVL, 61
 - COUNT_PARAM_MAC, 61
 - COUNT_TYPE, 61
 - D_CYCLE_PS, 61
 - DATARATE, 60
 - DEMOD_MARGIN, 61
 - DEVADDR, 60
 - DEVEUI, 60
 - DWN_CTR, 61
 - eParamMac, 59
 - eSuccessType, 60
 - eTypeCommand, 60
 - GATEWAY_NB, 61
 - JOIN, 61
 - LINK_CHECK, 61
 - LORA_ACCEPTED, 60
 - LORA_COUNT_SUCCESS, 60
 - LORA_FAILED, 60
 - LORA_MAC_TX_OK, 60
 - LORA_OK, 60
 - LORA_RX, 60
 - MAC, 61
 - NWKS_KEY, 61
 - PAUSE, 61
 - PWR_IND_VAL, 60
 - RADIO, 61
 - RESUME, 61
 - RETRANS_NB, 60
 - RnRequest, 61
 - RX_DELAY_1, 60
 - RX_DELAY_2, 61
 - RX2, 61
 - SAVE, 61
 - SerialType, 60
 - STATUS, 61
 - SYNC, 61
 - SYS, 61
 - TX_MAC, 61
 - UP_CTR, 61
- RnRequestClass, 33
 - ~RnRequestClass, 34
 - checkErrors, 34

- checkSuccess, 34
- cmdRequest, 34
- commandType, 35
- errorType, 35
- getLastError, 34
- getLastSuccess, 34
- getReceivedData, 34
- getResponse, 34
- init, 34
- isStreamInit, 34
- loraStream, 35
- OrangeForRN2483Class, 35
- params, 35
- possibleResponses, 35
- RadioCmdsClass, 35
- readLn, 34
- receiveBuffer, 35
- rnRequest, 34
- RnRequestClass, 33
- setLastError, 34
- successResponses, 35
- successType, 35
- SysCmdsClass, 35
- writeHexString, 34
- RX
 - RadioCmds.h, 56
- RX_DELAY_1
 - RnRequest.h, 60
- RX_DELAY_2
 - RnRequest.h, 61
- RX2
 - RnRequest.h, 61
- save
 - OrangeForRN2483Class, 19
- SAVE
 - RnRequest.h, 61
- sendMessage
 - OrangeForRN2483Class, 19, 20
- SEPARATOR
 - InternalConstForRN2483.h, 49
- SerialType
 - RnRequest.h, 60
- SET
 - InternalConstForRN2483.h, 49
- setAbpKeys
 - OrangeForRN2483Class, 20
- setAppEUI
 - OrangeForRN2483Class, 20
- setAppKey
 - OrangeForRN2483Class, 20
- setAppSKey
 - OrangeForRN2483Class, 21
- setAutoFreqBand
 - RadioCmdsClass, 30
- setAutoReply
 - OrangeForRN2483Class, 21
- setBatLvl
 - OrangeForRN2483Class, 21
- setBt
 - RadioCmdsClass, 30
- setDataRate
 - OrangeForRN2483Class, 21
- setDevAddr
 - OrangeForRN2483Class, 21
- setDevEUI
 - OrangeForRN2483Class, 22
- setDwnctr
 - OrangeForRN2483Class, 22
- setFrequency
 - RadioCmdsClass, 30
- setLastError
 - OrangeForRN2483Class, 22
 - RnRequestClass, 34
- setLinkCheck
 - OrangeForRN2483Class, 22
- setModulation
 - RadioCmdsClass, 31
- setNvm
 - SysCmdsClass, 38
- setNwkSKey
 - OrangeForRN2483Class, 22
- setOttaKeys
 - OrangeForRN2483Class, 23
- setOutputPower
 - RadioCmdsClass, 31
- setPinDig
 - SysCmdsClass, 38
- setPinMode
 - SysCmdsClass, 38
- setPort
 - DownlinkMessage, 6
- setPwrIdx
 - OrangeForRN2483Class, 23
- setResponseMessage
 - DownlinkMessage, 6
- setRetx
 - OrangeForRN2483Class, 23
- setRx2
 - OrangeForRN2483Class, 23
- setRxDelay1
 - OrangeForRN2483Class, 23
- setSF
 - RadioCmdsClass, 31
- setSync
 - OrangeForRN2483Class, 24
- setUpctr
 - OrangeForRN2483Class, 24
- SF_COUNT
 - ConstOrangeForRN2483.h, 44
- SF_ERROR
 - ConstOrangeForRN2483.h, 44
- SF10
 - ConstOrangeForRN2483.h, 44
- SF11
 - ConstOrangeForRN2483.h, 44
- SF12
 - ConstOrangeForRN2483.h, 44
- SF7
 - ConstOrangeForRN2483.h, 44
- SF8
 - ConstOrangeForRN2483.h, 44

- ConstOrangeForRN2483.h, 44
- SF9
 - ConstOrangeForRN2483.h, 44
- SIG_NOISE_RATIO
 - RadioCmds.h, 57
- sleep
 - SysCmdsClass, 38
- SLEEP
 - SysCmds.h, 64
- SPR_FACTOR
 - RadioCmds.h, 57
- STATUS
 - RnRequest.h, 61
- STR_ABP
 - InternalConstForRN2483.h, 49
- STR_CNF
 - InternalConstForRN2483.h, 49
- STR_ERROR_EMPTY_BUFFER
 - InternalConstForRN2483.h, 49
- STR_ERROR_LORASTR_NOT_INIT
 - InternalConstForRN2483.h, 50
- STR_ERROR_NETWORK_NOT_JOINED
 - InternalConstForRN2483.h, 50
- STR_MAC_RX
 - InternalConstForRN2483.h, 50
- STR_OFF
 - InternalConstForRN2483.h, 50
- STR_OK
 - InternalConstForRN2483.h, 50
- STR_ON
 - InternalConstForRN2483.h, 50
- STR_OTAA
 - InternalConstForRN2483.h, 50
- STR_UNCNF
 - InternalConstForRN2483.h, 50
- successResponses
 - RnRequestClass, 35
- successType
 - RnRequestClass, 35
- SYNC
 - RnRequest.h, 61
- SYNC_RADIO
 - RadioCmds.h, 57
- SYS
 - RnRequest.h, 61
- SysCmds
 - OrangeForRN2483Class, 25
- SysCmds.cpp, 62
- SysCmds.h, 63
 - _paramSys, 63
 - COUNT_PARAM_SYS, 64
 - eParamSys, 63
 - HWEUI, 63
 - NVM, 63
 - PIN_ANA, 63
 - PIN_DIG, 63
 - PIN_MODE, 63
 - RESET, 64
 - SLEEP, 64
 - VDD, 63
 - VERSION, 63
- SysCmdsClass, 36
 - getHardwareDevEUI, 36
 - getNvm, 36
 - getPinana, 37
 - getPindig, 37
 - getVdd, 37
 - getVersion, 37
 - params, 39
 - reset, 37
 - RnRequestClass, 35
 - setNvm, 38
 - setPinDig, 38
 - setPinMode, 38
 - sleep, 38
- tx
 - OrangeForRN2483Class, 24
- TX_MAC
 - RnRequest.h, 61
- TX_RADIO
 - RadioCmds.h, 56
- UNCONFIRMED_MESSAGE
 - ConstOrangeForRN2483.h, 45
- UP_CTR
 - RnRequest.h, 61
- VDD
 - SysCmds.h, 63
- VERSION
 - SysCmds.h, 63
- WATCHDOG_TIMER
 - RadioCmds.h, 57
- writeHexString
 - RnRequestClass, 34