# OrangeForRN2483

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

# Table of Contents

# OrangeForRN2483

**Version:**
1.0.5

**Author:**
Karim Baali

# Class Index

## Class List

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

# File Index

## File List

Here is a list of all files with brief descriptions:

# 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 instanciate 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:**

| | |
|---|---|
| *len* | Pointer on an uint8_t 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 instanciate 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:**

| | |
|---|---|
| *value* | 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:**

| | |
|---|---|
| *value* | 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:**

| | |
|---|---|
| *value* | 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:**

| | |
|---|---|
| *value* | 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:**

| | |
|---|---|
| *value* | 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:**

| | |
|---|---|
| *value* | 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:**

| | |
|---|---|
| *value* | 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:**

| | |
|---|---|
| *value* | 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:**

| | |
|---|---|
| *value* | 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:**

| | |
|---|---|
| *value* | 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:**

| | |
|---|---|
| *len* | 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 setOttaKeys (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 instanciate 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:**

| | |
|---|---|
| *adr* | 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:**

| | |
|---|---|
| *freqBand* | 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:**

| | |
|---|---|
| *status* | 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:**

| | |
|---|---|
| *devEUI* | Hexadecimal number represented by an array whose size is **8** int8_t |
| *appEUI* | Hexadecimal number represented by an array whose size is **8** int8_t |
| *appKey* | 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:**

| | |
|---|---|
| *appEUI* | Hexadecimal number represented by an array whose size is **8** int8_t |
| *appKey* | 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:**

| | |
|---|---|
| *typeMessage* | eTypeMessage value representing the uplink payload type (*CONFIRMED_MESSAGE* or *UNCONFIRMED_MESSAGE* ) |

| | |
|---|---|
| *data* | Hexadecimal value representing the data sent to the server |
| *size* | Unsigned integer value representing the size of the transmitted data only |
| *port* | 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:**

| | |
|---|---|
| *data* | Hexadecimal value representing the data sent to the server |
| *size* | Unsigned integer value representing the size of the transmitted data only |
| *port* | 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:**

| | |
|---|---|
| *nwkSkey* | Hexadecimal number represented by an array whose size is **16** int8_t |
| *appSKey* | 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:**

| | |
|---|---|
| *appEUI* | 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:**

| | |
|---|---|
| *appKey* | 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:**

| | |
|---|---|
| *appSKey* | 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:**

| | |
|---|---|
| *autoRep* | String value enabling or not the automatic reply (*on* or *off* ) |

**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:**

| | |
|---|---|
| *lvl* | 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:**

| | |
|---|---|
| *dataRate* | 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:**

| | |
|---|---|
| *devAddr* | 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:**

| | |
|---|---|
| *devEUI* | 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:**

| | |
|---|---|
| *dwnctr* | 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:**

| | |
|---|---|
| *linkCheck* | 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:**

| | |
|---|---|
| *nwkSKey* | 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:**

| | |
|---|---|
| *pwrIdx* | 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:**

| | |
|---|---|
| *retx* | 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:**

| | |
|---|---|
| *dataRate* | eDataRate value representing the data rate value (see constOrangeForRn2483.h for more information) |
| *frequency* | 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:**

| | |
|---|---|
| *rxDelay1* | 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:**

| | |
|---|---|
| *syncWord* | 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:**

| | |
|---|---|
| *upctr* | 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 functionnality.*
- 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 functionnality.

This function allows the user to have access to the status of the **Invert IQ functionnality** by executing a "radio get iqi" 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:**

| | |
|---|---|
| *outputPower* | 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:**

| | |
|---|---|
| *watchdog* | 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:**

| | |
|---|---|
| *autoFreqBand* | 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:**

| | |
|---|---|
| *bt* | 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:**

| | |
|---|---|
| *frequency* | 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:**

| | |
|---|---|
| *mode* | 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:**

| | |
|---|---|
| *pwrout* | 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:**

| | |
|---|---|
| *spreadingFactor* | 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 instanciate 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:**

| | |
|---|---|
| *address* | Hexadecimal number represented by an array whose size is **2** int8_t |

**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:**

| | |
|---|---|
| *pinname* | 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:**

| | |
|---|---|
| *pinname* | 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:**

| | |
|---|---|
| *address* | Hexadecimal number representing the user EEPROM address, from 300 to 3FF |
| *data* | 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:**

| | |
|---|---|
| *pinname* | String value representing the pin name ("GPIO0", "GPIO13", "UART_CTS", "UART_RTS") |
| *pinstate* | 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:**

| | |
|---|---|
| *pinname* | String value representing the pin name ("GPIO0", "GPIO13", "UART_CTS", "UART_RTS") |
| *pinfunc* | 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:**

| | |
|---|---|
| *delay* | 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_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 | |

**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_MESSAGE | |

## 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 Sodaq 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_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 | |

# 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_ACCEPTED | |
| LORA_RX | |
| LORA_COUNT_SUCCESS | |
| 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