ENOCEAN SYSTEM SPECIFICATION

Remote Management 2.0

March 06, 2013

EnOcean GmbH
Kolpingring 18a

Phone +49.89.67 34 689-0
Fax +49.89.67 34 689-50

Subject to modifications

EnOcean Standard

REVISION HISTORY

The following major modifications and improvements have been made to the first version of this document:

No	Major Changes	Date	Who
1.1	Added Smart Acknowledge RPCs		
1.2	Corrected Smart Acknowledge RPCs for simple remote management		
1.3	Migrated to the System Specification Document All the answer commands are defined as UNICAST Corrected EEP definition – there is no such thing as default EEP so EEP mask bits were introduced Set correct Manufacturer ID in answer telegrams Adjusted return codes of answer telegrams, introduced new return codes		
1.4	Corrected EEP mask bit definition, added information about repeating		
1.5	Modified security behavior, if code is set there is no 30min unlock period		
1.6	Exported from system spec	15.10.2010	ASt
1.7	Moved RPC description to the EEP2.1 specification	14.12.2010	ASt
1.8	Major review, improved text and structure	16.12.2010	ASt

1.9	Inserted ERP2 information	16.08.2012	AP
2.0	Manufacturer ID Clarification	06.03.2014	BE

Published by EnOcean GmbH, Kolpingring 18a, 82041 Oberhaching, Germany

www.enocean.com, info@enocean.com, phone ++49 (89) 6734 6890

Important!

This information describes the type of component and shall not be considered as assured characteristics. No responsibility is assumed for possible omissions or inaccuracies. Circuitry and specifications are subject to change without notice. For the latest product specifications, refer to the EnOcean website: http://www.enocean.com.

As far as patents or other rights of third parties are concerned, liability is only assumed for modules, not for the described applications, processes and circuits.

EnOcean does not assume responsibility for use of modules described and limits its liability to the replacement of modules determined to be defective due to workmanship. Devices or systems containing RF components must meet the essential requirements of the local legal authorities.

The modules must not be used in any relation with equipment that supports, directly or indirectly, human health or life or with applications that can result in danger for people, animals or real value.

Components of the modules are considered and should be disposed of as hazardous waste. Local government regulations are to be observed.

Packing: Please use the recycling operators known to you. By agreement we will take packing material back if it is sorted. You must bear the costs of transport. For packing material that is returned to us unsorted or that we are not obliged to accept, we shall have to invoice you for any costs incurred.

INDEX

Annotation

Sys_ex telegram – is a telegram that is sent through radio or serial interface and is built according to the sys_ex telegram specification.

Message – is information that is sent through radio or serial interface. It consists of one or several sys_ex telegrams.

Command – is a request from Remote Manager to perform a specified reaction.

EEP – EnOcean Equipment Profile

Remote Management – is a network module, which allows Remote Devices to be configured and maintained over the air or via serial interface from a Remote Manager.

Remote Device – is a device that supports Remote Management. In the network model it is the client. The device that handles requests, sends answers and executes functions. It is the managed module.

Remote Manager – is a device that supports Remote Management. In network model it is the manager. The device sends request to execute functions and processes answers. It is the manager module. Usually the actor uses Remote Manager to communicate with remote devices.

1 Introduction

This document describes the functionality of Remote Management. Remote Management allows EnOcean devices to be configured and maintained over the air or serial interface using radio or serial telegrams. Thanks to Remote Management, sensors or switches IDs, for instance, can be stored or deleted from already installed actuators or gateways which are hard to access.

Remote Management also allows querying debug information from the Remote Device and calling some manufacturer implemented functions.

Remote Management is supposed to be used with current and future products, so it has to ensure back compatibility with devices and be extendible for future use. The software of current devices has to be changed in order to support Remote Management. Remote Management is platform independent.

2 Functional description

Remote Management is performed by the Remote Manager, operated by the actor, on the managed Remote Device (Sensor, Gateway). The management is done through a series of commands and responding answers. Actor sends the commands to the Remote Device. Remote Device sends answers to the actor. The commands indicate the Remote Device what to do. Remote Device answers if requested by the command. The commands belong to one of the main use case categories, which are:

- Security
- Locate / indentify remote device
- Get status
- Extended function.

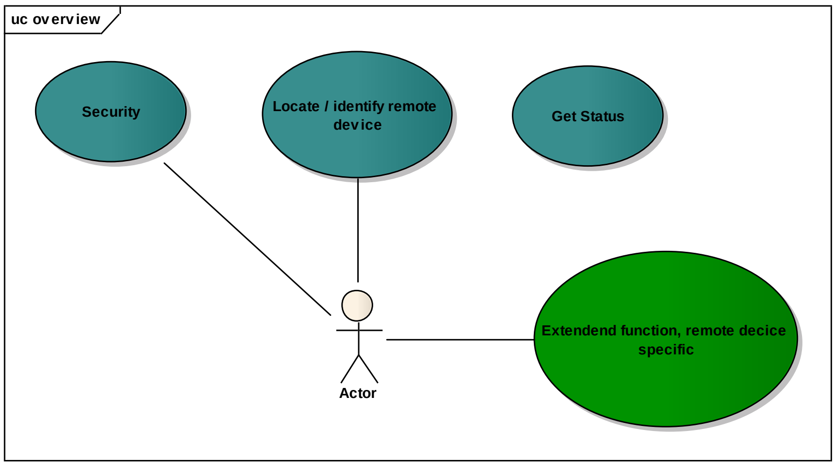

uc overview
Security
Locate / identify remote device
Get Status
Extendend function, remote decice specific
Actor

Figure 1 Use case Overview

The management is often done with a group of Remote Devices. Commands are sent as addressed unicast telegrams, usually. In special cases broadcast transmission is also available. To avoid telegram collisions the Remote Devices respond to broadcast commands with a random delay.

The Security, Locate, and Get Status options provide to the actor basic operability of Remote management. Their purpose is to ensure the proper work of Remote Management when operating with several Remote Devices. These functions behave in the same way on every Remote Device. Every product that supports Remote Management provides these options.

Extended functions provide the real benefit of Remote Management. They vary from Remote Device to Remote Device. They depend on how and where the Remote Device is used. Therefore, not every Remote Device provides every extended function. It depends on the programmer / customer what extended functions he wants to add. There is a list of specified commands, but the manufacturer can also add manufacturer specific extended functions. These functions are identified by the manufacturer ID.

2.1 Security

For security reasons the remote management commands can only be accessed in the unlock period. The period can be entered in two cases:

- Within 30min after device power-up if no CODE is set
- Within 30min after an unlock command with a correct 32bit security code is received

The unlock/lock period can be accessed only with the security code. The security code can be set whenever the Remote Device accepts remote management commands. The state diagram below describes the various states of Remote Management in the view of security.

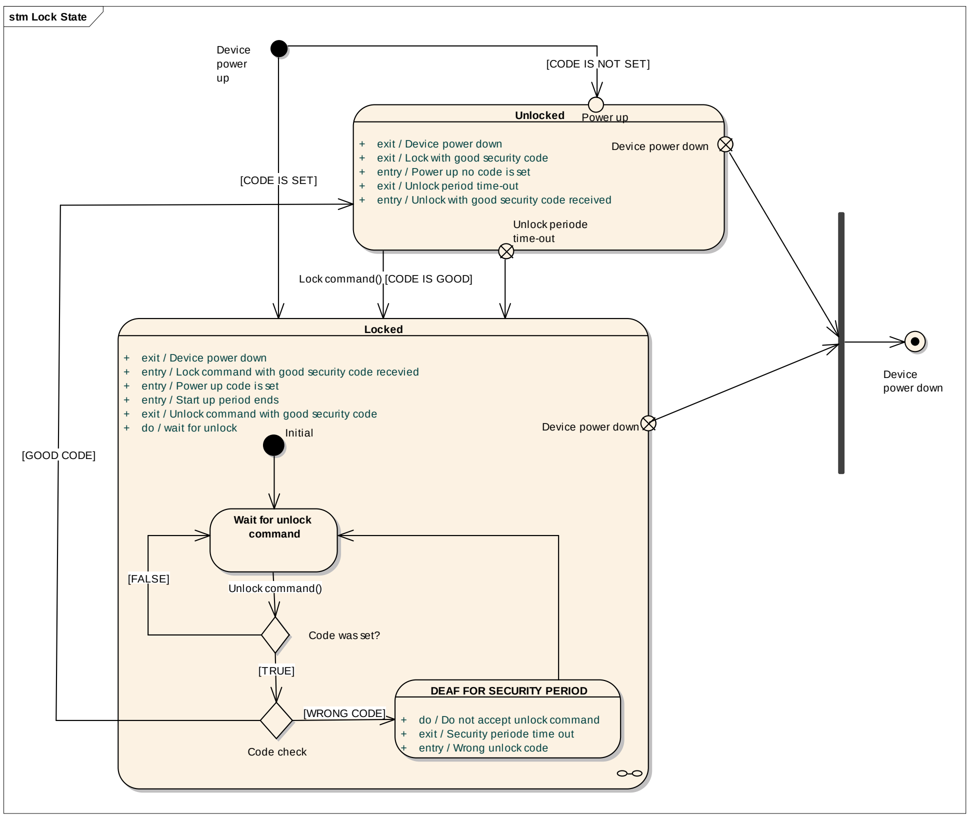

stm Lock State
Device power up
[CODE IS NOT SET]
Unlocked
Power up
+ exit / Device power down
+ exit / Lock with good security code
+ entry / Power up no code is set
+ exit / Unlock period time-out
+ entry / Unlock with good security code received
Device power down
[CODE IS SET]
Unlock periode time-out
Lock command() [CODE IS GOOD]
Locked
+ exit / Device power down
+ entry / Lock command with good security code recevied
+ entry / Power up code is set
+ entry / Start up period ends
+ exit / Unlock command with good security code
+ do / wait for unlock
Initial
Device power down
Device power down
[GOOD CODE]
Wait for unlock command
[FALSE]
Unlock command()
Code was set?
[TRUE]
[WRONG CODE]
Code check
DEAF FOR SECURITY PERIOD
+ do / Do not accept unlock command
+ exit / Security periode time out
+ entry / Wrong unlock code

Figure 2 State diagram

When the Remote Device is locked it does not respond to any command, but unlock and ping. When a wrong security code is received the Remote Device does not process unlock commands for a security period of 30 seconds. This limits the risk of finding the security code through random generator.

Security code=0x000000 is the default value and has to be interpreted as: no CODE has been set. The actor can also set the security code to 0x000000 from a previously set value. If no security code is set, unlock after the unlock period is not processed. Only ping will be processed. Remote Management is not available until next power up. 0xFFFFFFFF is reserved and can not be used as security code.

The following commands belong to the security option:

- Set code command

The command enables the actor to set the security code.

- Lock Command

The command explicitly locks the device. With the *lock* command it is mandatory to transfer the appropriate security code.

- Unlock Command

The command unlocks the device. With the *unlock* command it is mandatory to transfer the appropriate security code. When Remote Device receives an *unlock* command with wrong security code it does not evaluate *unlock* commands for 30 seconds (security period).

An overview is below.

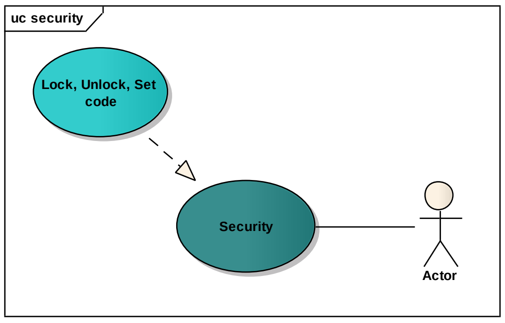


Figure 3 Use case Security

2.2 Locate/identify remote device

This option identifies one desired Remote Device in a group of devices in an unknown environment. To communicate with the Device it is necessary to find out the device ID. For this purpose the following commands are available:

- Query ID Command

  Query ID is sent always as broadcast telegram. All unlocked devices respond to the Query ID with their ID and their EEP.

  The EEP is a 21 bit and it is defined as following: ORG-FUNC-TYPE. For more information about the EEP be sure to read EEP2.1 specification.

  The Query ID command contains an EEP definition and mask bits. When the mask bits are set to 0x01 only Remote Devices with the matching EEP will process the remote command. If the query ID with mask bit 0x00 is transmitted, the EEP bytes in this command will be ignored and every Remote Device will answer to this command. If a Remote Device has no EEP, then it will only respond to the Query ID command where the mask bits are set to 0x00.

  The mask bits in the Query ID answer telegrams are set to 0x00.

- Action Command

  When this command is received then the addressed device performs an action (audio, visual, etc), depending on the functionality of the device. With this function a remote device with known ID can be can be clearly localized. A detailed description is listed in the use case scenarios.

An overview is below.

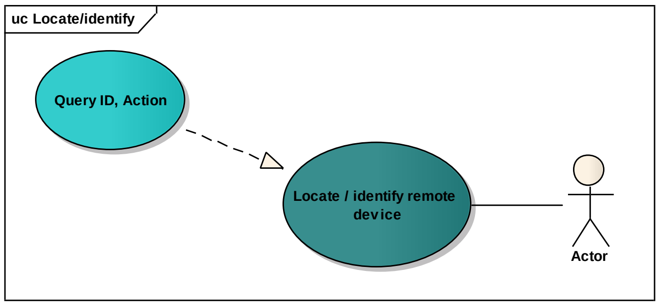


Figure 4 Use Case Locate/identify

2.3 Get status

This option is mainly intended for obtaining debug information from the managed Remote Device. For more information see chapter 4.2.3. The following commands belong to this group:

- Query Status

  With this command the actor directly asks for the status info of the Remote Device. The Remote Device answers with remote management debug data.

  The answer will contain:

  — If a security code is set or not

  — Last remote command function number (RMCC or RPC)

  — Last commands return code (OK, error, etc.)

  — Telegram merge info – last SEQ number and if merge successful or error in receive

- Ping

  The ping command functionality is similar to ping in TCP / IP communication. The actor sends a ping request to see if the Remote Device is alive and communicating. Ping requests are processed also when device is in lock status. Remote device sends the radio signal strength of the received request within the ping response.

An overview is given below.

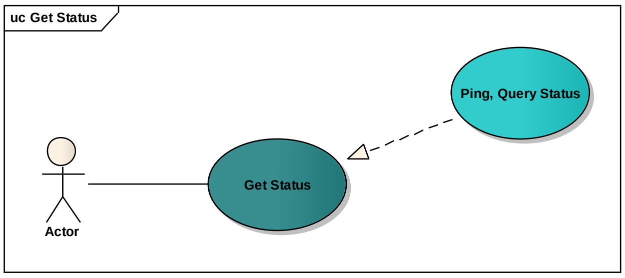


Figure 5 Use Case Get Status

2.4 Extended options

The benefit of extended options in Remote Management is that special and user defined remote device functions can be called remotely. The Remote Management offers ways to call those functions with appropriate commands and parameters. The following actions after the commands are specific for the remote device. The extended functions are specified by their function code and manufacturer ID. Not every remote device supports every extended function.

The following commands belong to this option:

- Call Function Command

  With this command extended functions can be called.
- Query function Command

  With this command the actor requests the supported extended functions list.

It is expected that some functions need to send data back to the actor. The length of the data can vary and exceed the length of the data field in one telegram. For this purpose, the remote manager provides the merge/divide option of data with accurate encapsulation. An overview is given below.

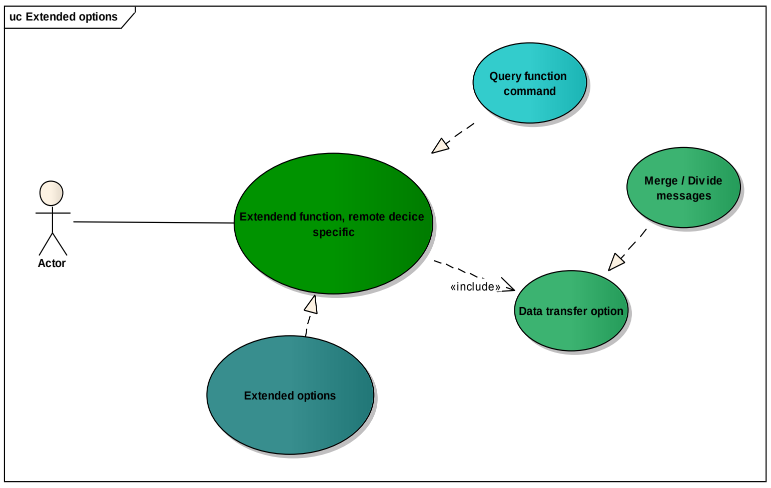


Figure 6 Use Case Extended options

3 Functions and responses

As mentioned the management is done by the commands that directly request an action. For communication reasons they are addressable and support broadcast. The various commands are specific by their function code and remote function they call.

There are two types of remote management commands:

- Remote Management Control Commands - RMCC
- Remote Procedure Calls - RPC

Remote Management Control Commands - RMCCs are available in every product with Remote Management feature. They provide the basic functionality for Remote Management. RMCCs have a common definition. Remote Devices react always in the same way on RMCC.

These commands are:

- Lock
- Unlock
- Set CODE
- Query ID
- Action
- Query status
- Ping
- Query function

RPCs functions strongly depended on the Remote Device. They provide additional functions like remote learn or remote clear of the learned IDs. Not every Remote Device provides the same RPCs. The manufacturer can also determine and implement RPC for his needs. These special RPCs are defined by the function code and Manufacturer Id. The RPC are called with the call function command.

Complete overview of the use case is below.

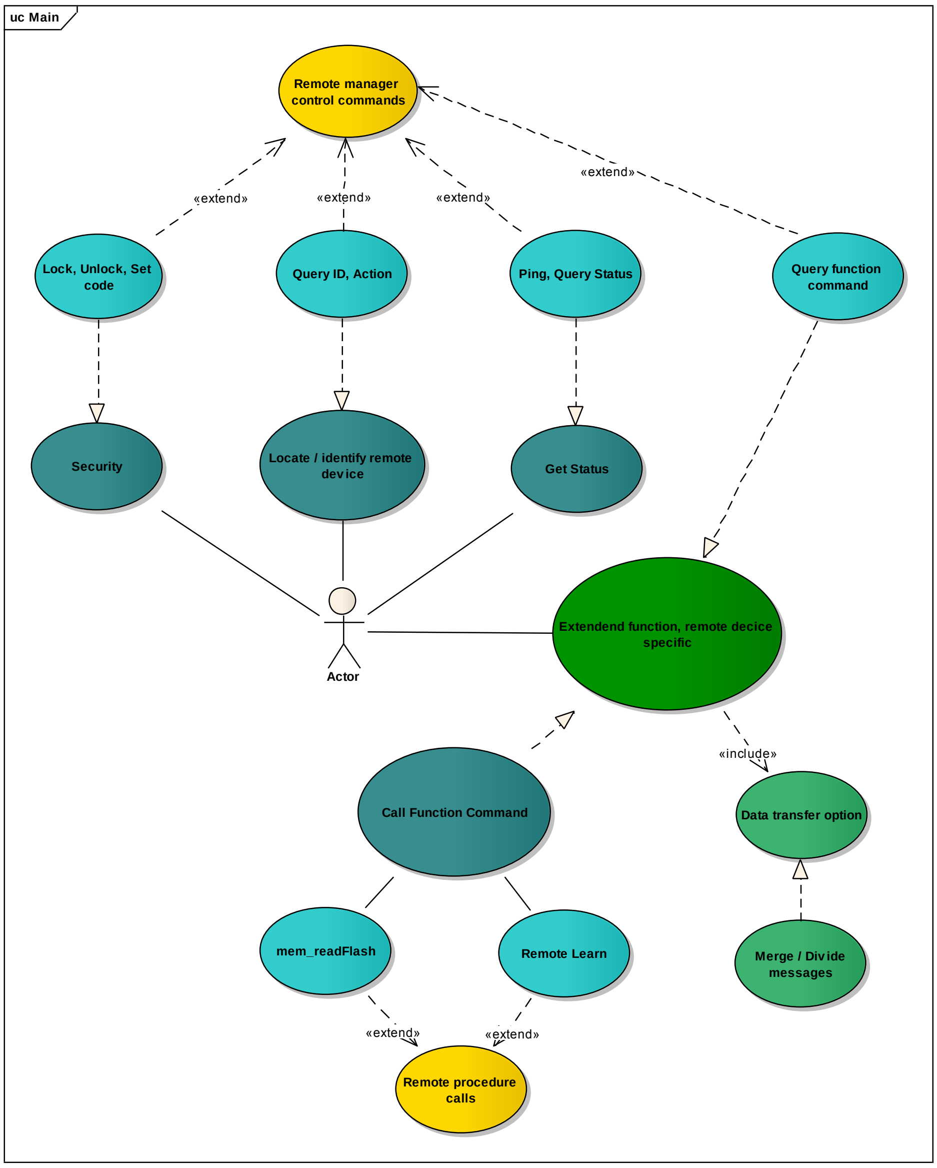
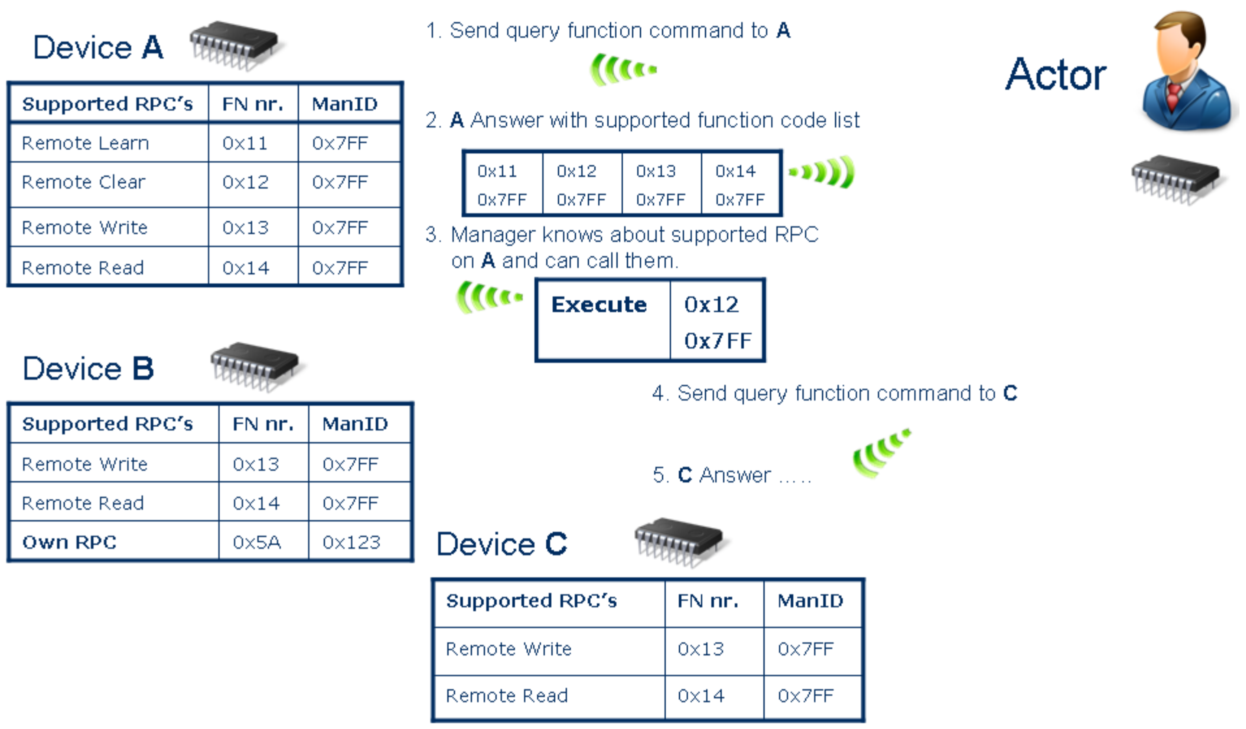


Figure 7 Use Case Main

3.1.1 Query supported function list

The list of supported RPC can be fetched with the query function command. Every Remote Device can support different commands so the Remote Devices have to be queried one by one. In the figure below a scenario of fetching the supported RPC list is shown.


Device A
1. Send query function command to A
Actor
2. A Answer with supported function code list
3. Manager knows about supported RPC on A and can call them.
Device B
4. Send query function command to C
5. C Answer .....
Device C

Device A

Supported RPC's	FN nr.	ManID
Remote Learn	0x11	0x7FF
Remote Clear	0x12	0x7FF
Remote Write	0x13	0x7FF
Remote Read	0x14	0x7FF

0x11	0x12	0x13	0x14
0x7FF	0x7FF	0x7FF	0x7FF

Execute	0x12
	0x7FF

Device B

Supported RPC's	FN nr.	ManID
Remote Write	0x13	0x7FF
Remote Read	0x14	0x7FF
Own RPC	0x5A	0x123

Device C

Supported RPC's	FN nr.	ManID
Remote Write	0x13	0x7FF
Remote Read	0x14	0x7FF

Figure 8 Query supported function list

3.1.2 Remote Device’s response to RMCC

In case that a RMCC directly requires to send something back to the actor (query ID, ping, query status, query function command) then this answer will be send immediately with according answer function code. There is strong need to separate the answer function codes from original commands so Remote Device does not process the answers or even react to them. Otherwise that will cause an error state, deadlock or message - answer cycles. In case there is no answer needed, the debug result of the last RMCC will be saved locally and can be fetched remotely with the query status command.

Table 1 RMCC Answers

RMCC	Information contained in answer
Query ID	Remote device’s ID and EEP.
Ping command	EEP
Query function command	Extended functions list supported by the Remote Device.
Query status	Debug information about the Remote Device.

3.1.3 Remote Device’s response to RPC

When a RPC function wants to send an answer with data, then the answer should be send in form of a remote management response. It means that a different function number should be used. Doing so other Remote Devices do not have to process the answers or even react to answers, what can cause an error state. If the amount of data exceeds one telegram, then the response can be separated into more telegrams.

The debug information about the last call is saved locally and can be fetched with query status. It is necessary to separate the answers function codes from original commands.

3.1.4 Broadcast

If the Remote Device recognizes that the received message was a broadcast message and the command requests to send a response, then the answer is send with random delay to avoid collisions. The random delay is between 0 – 2000 ms.

4 Communication Protocol

4.1 Messages structure

The remote management communication happens on the Transport OSI layer and uses Messages as containers. The messages are chained SYS_EX telegrams.

A SYS_EX message may consist of several SYS_EX telegrams. When a message is received the Remote Device merges the SYS_EX telegrams that the message consists of. Then the information is passed as a unit to the application (when RPC).

The length of transferred data is also a part of the protocol and it is used to count the amount of message parts when receiving.

For sending, the SYS_EX message is split into several parts capsulated into SYS_EX telegrams and sent to the receiver module. There the telegrams are merged to a message again. This mechanism is supported by the Remote Manager and also by the Remote Device, so the communication with SYS_EX messages is bidirectional. A status diagram of merge is shown below.

Endianness is Big-endian like in other protocol stacks. But in contrast to other protocol stacks the data field gets filled from left to right. So when three data bytes are sent the three most left bytes get filled and the one most right stays empty.

4.1.1 Message addressing

Each SYS_EX telegram can be addressed. The addressing mechanism is done using ADT encapsulation. For more information please see EnOcean System Specification Address Destination Telegrams. The Remote Device ignores telegrams with other than Remote Device ID or broadcast ID. The broadcast address is 0xFFFFFFFF.

4.1.2 SYS_EX Telegram Structure

For ERP1:

RORG	msg_id (1 b)		data field	sender id	status	crc8
	SEQ	IDX				
1 byte	2 B	6 B	**8 bytes**	4 bytes	1 byte	1 byte

RORG: 0xC5

length: 16 bytes

For ERP2:

Length	Header	Teltype	sender id	msg_id (1 b)		data field	crc8
				SEQ	IDX		
1 byte	1 byte	1 byte	4 bytes	2 B	6 B	**8 bytes**	1 byte

RORG: Extended telegram type 0x00

length: 18 bytes

msg_id:
The telegram identification is a composite of the sequence number, the telegram index and the sender ID.

Sequence number SEQ – the number is for error handling. Telegrams of the message have the same sequence number SEQ. So when a message consists of several telegrams, every telegram of that message has the same SEQ number. The SEQ is random generated by the Remote Management. It ensures that telegrams of several sys_ex messages do not get mixed; we can clearly identify the telegrams by their SEQ numbers.

Telegram index IDX – indicates the order of the telegram in the sys_ex message. This counter starts from 0.

For detailed information please read 4.1.3.

Data field:

Indicates the data transmitted or received. The data field structure is depended on the message.

The first telegram of sys_ex message, with index 0, contains information about the message (Length, Count, Fn_Number, Target_ID). All other telegrams have the same structure.

IDX = 0 (first telegram)

data_length	manufacturer ID	fn_number	payload
9 bits	11 bits	12 bits	32 bits

data_length – total number of data bytes in the message (excluding data_length, manufacturer ID and Fn_Number, may be distributed over several telegrams)

Manufacturer ID – RMCC and EEP defined RPCs will be sent using the multi user (0x7FF) manufacturer ID. Custom RPCs will be sent with the manufacturer ID of the Remote Device. All responses to commands from Remote Devices will use the manufacturer ID of the device responding. The response cannot be multi user.

fn_number – function number to call

payload – custom data, can be interpreted depending on the function

Note: the number of telegrams can be calculated from the data_length

IDX > 0 (all next telegrams)

payload

8 byte

Note: any further telegram / part of the sys_ex message contains only payload

Restrictions:

The telegrams must use the whole data length, except the last telegram.

The last telegram is the only one that may have the DATA field partially filled.

The used bytes in the last telegram can be calculated from the data length field in the first telegram.

4.1.3 Sequence (SEQ) and Index (IDX)

The SEQ, IDX numbers are in every SYS_EX telegram. Telegrams of the same message have the same SEQ number. Every SYS_EX telegram has a specific IDX number. The SEQ = 0 is not allowed. IDX is used to sort and identify message parts. IDX starts at 0 in every following telegram the IDX is incremented. Error handling:

- If parts of a message are not received and the actor sends a new SYS_EX message, the protocol handler recognizes this by the messages SEQ number.
- The telegrams have arrived in a different order as they were sent. In this case, the protocol handler sorts the data with the IDX number

SEQ handling is demonstrated in Figure below.

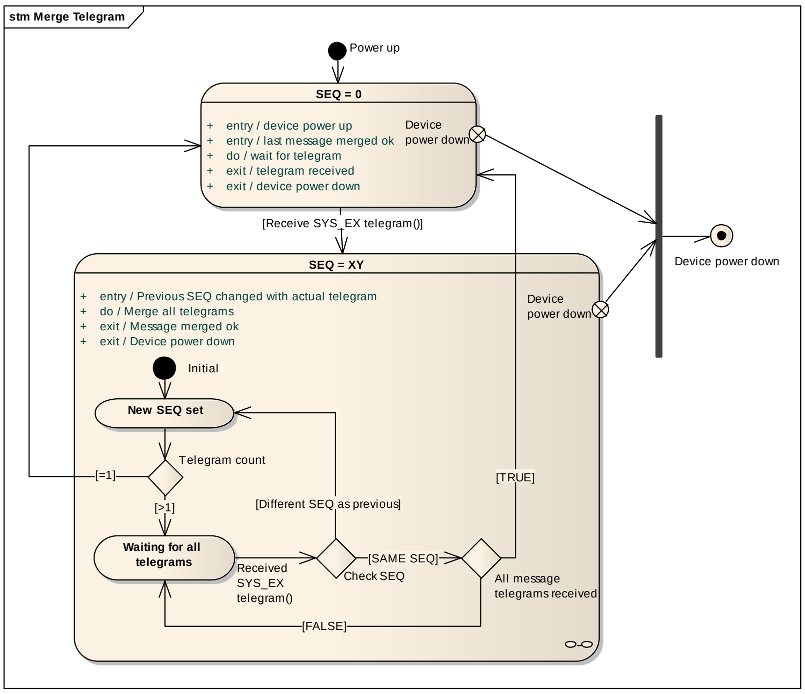

stm Merge Telegram
Power up
SEQ = 0
+ entry / device power up
+ entry / last message merged ok
+ do / wait for telegram
+ exit / telegram received
+ exit / device power down
Device power down
[Receive SYS_EX telegram()]
SEQ = XY
+ entry / Previous SEQ changed with actual telegram
+ do / Merge all telegrams
+ exit / Message merged ok
+ exit / Device power down
Device power down
Device power down
Initial
New SEQ set
Telegram count
[=1]
[>1]
[TRUE]
[Different SEQ as previous]
Waiting for all telegrams
Received SYS_EX telegram()
[SAME SEQ]
Check SEQ
All message telegrams received
[FALSE]

Figure 9 State Diagram Merge Telegram with SEQ

Dividing and merging telegrams into messages enables to transfer bigger amount of data. The dividing and merging process has these characteristics:

- Telegram with IDX 0 has header information about the whole message
- Messages are merged based on same destination Id, source Id, SEQ. IDX orders the telegrams parts.
- In the telegram with IDX = 0 are transmitted 4 bytes of data
- In all next telegrams are transmitted 8 bytes of data

A common message composite of 4 telegrams looks like in the tables bellow. The amount of data is 22 bytes and the function number is 0x210. The telegrams have incremented IDX numbers.

4.2 Error handling in message merge process

Transferring more telegrams chained to a message demands error handling. Individual telegrams of a message can get lost e.g. because of a collision. When working with Remote Management a safer operation must be achieved, even when telegrams get lost or other failures occur. Therefore we declare some error handling and error avoiding mechanisms.

For error handling in Remote Management we declare the following mechanism:

INDEX number:

— Order the telegram within a message.

— When telegram with same IDX received twice discard previous message.

a) SEQ number:
   - Group telegrams with same SEQ to one message.

b) CHAIN period:
   - Telegrams in sequence must be sent within the chain period.
   - If a message is not received completely and the chain period expires the message is discarded
   - telegrams with other Sender Id are discarded within the chain period

c) Message grouping:
   - Messages are differentiated by their Sender ID and SEQ number

Also we declare:

- chained telegrams must be sent in sequence within the chain period
- every telegram can be sent only once (with 3 subtelegrams)

The receive process can work as in Figure 10. On this figure the chain period and the message Sender Id are evaluated. This should happen before the SEQ gets checked.

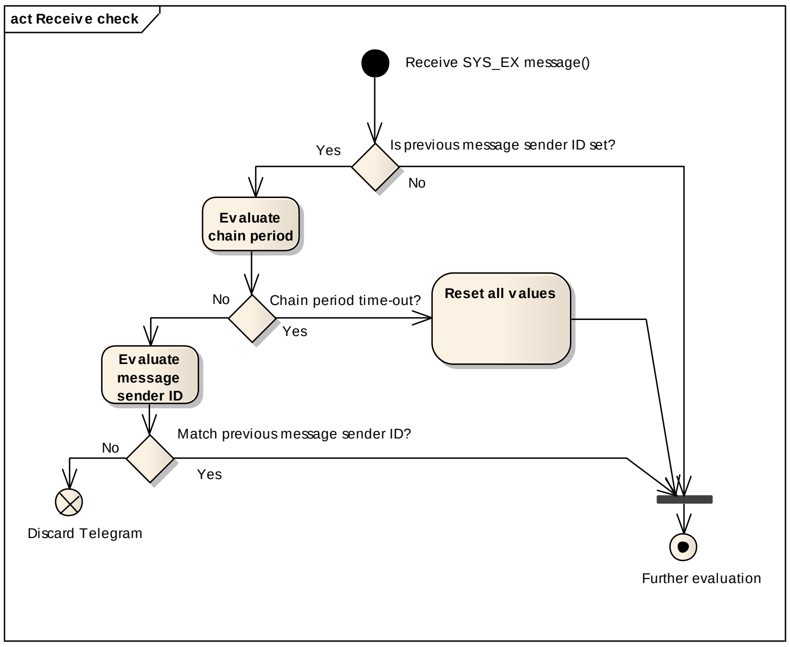


Figure 10 Activity Diagram Check Message ID and Chain period

4.2.1 Timeout

When more managers are operating at the same time telegrams can interfere. Telegrams of one message mix with other telegrams. Telegrams can be sort based on Sender Id, Destination Id and SEQ. To exclude any possible failure we declare a chain period. The chain period determines the maximum time between any two telegrams in a message. When the next telegram is not received within the chain period, the message is declared as corrupted. So if more chained telegram the next must be transferred within the chain period.

Problem Description:

One actor sends a SYS_EX message with 5 telegrams to a desired Remote Device with ID A.

d) 2 of 5 telegrams are received.

e) Between the second and third telegram the second actor sends a broadcast message with the query ID command.

f) The remote device with ID A receives the query ID command.

g) When the remote device has only buffer for one message, the message from the second actor gets processed and the current message from the first actor is corrupt. It cannot be processed, although all telegrams have been received, because the second message has overwritten the first message in buffer.

If we declare, that telegrams with other sender ID do not get processed within the chain period, this failure will not occur. When a remote device can receive more messages at once this will not occur.

Problem Description:

The actor sends a SYS_EX message with 5 telegrams to a desired Remote Device with ID A.

h) All telegrams are sent but only 4 are received (with IDX 0,1,2,4).

i) The Remote Device is still waiting for the fifth telegram (with IDX 3).

j) The actor sends a different SYS_EX message with same SEQ and with 4 telegrams.

k) The last telegram (IDX 3) is received as first on the remote device.

l) The Remote Device thinks that the currently received telegram is the last one from the previous message and processes the previous message – error behaviour will occur.

When we discard the message after the chain period expires, it can be clearly declared when a new message is transferred and this issue will not occur. The chain period is 1 second.

4.2.2 Check already received telegram

Every telegram has its IDX number that identifies the position within the whole message. Only if all telegrams are received the message gets processed. Every telegram is transferred only once, so if a telegram with same IDX is within a message received again it indicates that a possible failure occurred.

Problem Description:

The actor sends a SYS_EX message with 5 telegrams to a desired Remote Device with Id A.

m) All telegrams are send but only 4 are received (with IDX 0,1,2,4).

n) The Remote Device is still waiting for the fifth telegram (with IDX 3).

o) The actor sends a different SYS_EX message within the chain period with same SEQ and with 4 telegrams.

p) When the first telegram is received (IDX 0) the remote device can assume that it is dealing with new message. Data will be overwritten and error state can occur.

When we discard a message when a telegram with already received IDX number is been received again, we can avoid this failure. This will work only when we declare that every telegram is transmitted only once.

4.2.3 Query Status

With the query status command the actor can query debug information about the Remote Management on the Remote Device. He can query this information:

- If security code is set or not
- Last remote command function number (RMCC or RPC)

  The last remote command code is same as the function code of the command.
- Last function return code (OK, error, etc.)

  Last function return code is relevant to function when merge was successful, because only then a function gets executed. When merge failed then last function return code give a clue why the merge failed (0x09, 0x0A, 0x0B, 0x0C).
- Telegram merge info – if merge successful or error in receive

  Telegram merge info is information if the last message was received OK. When the value is 0 then it is OK – the last message was completely received and merged if needed.

  If value is >0 the merge was not successful, the message was not received. In this case the value is the last SEQ that was received by the remote device.

  Only if the last merge was successful we can evaluate the last command function number. If last merge failed we cannot know if the last command function code is up to date.

The last function number, return code and merge info belong to the most recent command and merge process.

The last function return code is listened in table bellow.

Table 2 Function return codes

Status name	Code number
OK	0x00
Wrong target ID	0x01
Wrong unlock code	0x02
Wrong EEP	0x03
Wrong manufacturer ID	0x04
Wrong data size	0x05
No code set	0x06
Not send	0x07

RPC failed	0x08
Message time out	0x09
Too Long Message	0x0A
Message part already received	0x0B
Message part not received	0x0C
Address out of range	0x0D
Code data size exceeded	0x0E
Wrong data	0x0F

4.3 REMAN concept and repeating

When using REMAN telegrams the telegram repeater status should be considered. When the repeater status of the telegrams is 0 the REMAN telegrams will be repeated by repeaters. It can happen that in this can lead to an unexpected behaviour.

Consider the following scenario:

An application sends a QueryID telegram from PC with repeater status set to 0.

Setup:

PC – Computer running application like DolphinView

Device A – radio to serial gateway

Device B – device we want to control per remote management

Step-by-step:

DolhinView send QueryID telegram over serial, repeater status is 0.

Device A receives serial telegram and forwards it to radio.

Device B replays with QueryID Answer telegram.

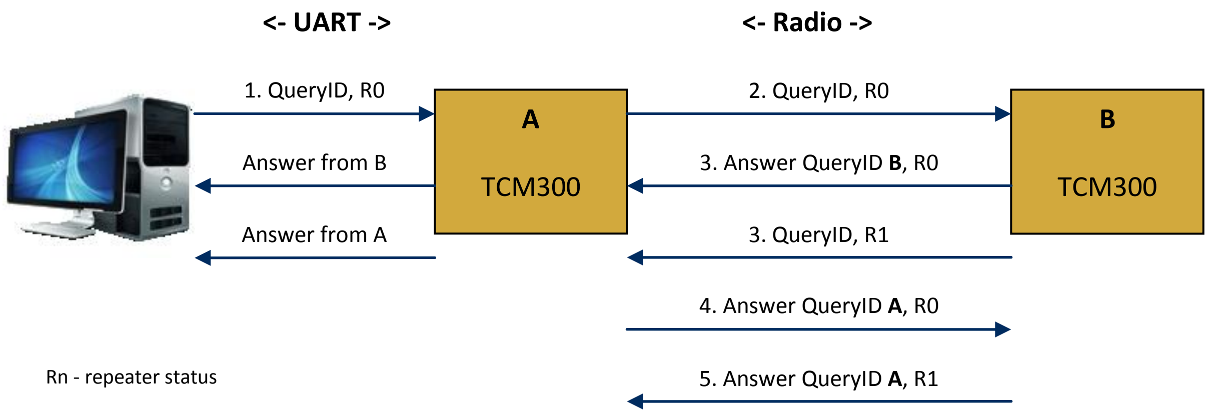
Device B repeats the QueryID, increases repeater status bits to 1 (R1).

Device A receives the QueryID telegram.

Device A replays witch QueryID Answer telegram.

Device A repeats the QueryID Answer, increases repeater status.

PC receives QueryID answer from device B and device A.



Rn - repeater status

R0 - original telegram

The problem with this scenario is that we actually queried also our gateway. There are several solutions for this problem:

- When sending remote management telegrams from an application set the status byte to 0xF – Telegram must not be repeated. This way the repeaters will not repeat the telegrams again. In some scenarios this also reduces the radio traffic.
- Use gateway software without remote management possibilities
- Filter the remote management answers with gateway ID on application side.
- Recommended is always use the first solution and set the status of REMAN telegrams to 0xF.

5 RMC and RPC structure definitions

5.1 Remote Procedure Calls (RPC)

These commands are described in the actual EEP specification.

5.2 Remote Management Commands (RMCC)

Note that all the telegrams in this chapter are represented in ADT encapsulated form. ADT encapsulation is used to makes possible transmit telegrams with UNICAST i.e. with a certain destination ID (see chapter XY). To transmit telegrams as a broadcast either the ADT with the broadcast destination ID 0xFFFFFFFF or the SYS_EX telegrams without ADT encapsulation can be sent.

Function code	RMCC – Remote Management Control Commands
0x000	RESERVED
0x001	UNLOCK
0x002	LOCK
0x003	Set CODE
0x004	Query ID
0x005	Action command
0x006	Ping command
0x007	Query function command
0x008	Query status

Table 3 RMCC function codes

In the following description only the payload is shown independent of the radio or serial protocol.

5.2.1 Unlock

UNLOCK		
Function code		0x001
Manufacturer Id		0x7FF
Data length		4 bytes
Data content	Security code	4 bytes
Unicast		yes
Broadcast		yes
Device responses to command		no
Status return code		
	OK	0x00
	Wrong target ID	0x01
	Wrong unlock code	0x02
	Wrong manufacturer ID	0x04
	Wrong data size	0x05
	No code set	0x06

Table 4 Unlock telegram

	7	6	5	4	3	2	1	0
0	SEQ		0x00					
1	0x04							
2		0x7FF						
3								
4	0x001							
5	SECURITY CODE							
6								
7								
8								

5.2.2 Lock

LOCK		
Function code		0x002
Manufacturer Id		0x7FF
Data length		4 bytes
Data content	Security code	4 bytes
Unicast		yes
Broadcast		yes
Device responses to command		no
Status return code		
OK	0x00	
Wrong target ID	0x01	
Wrong unlock code	0x02	
Wrong manufacturer ID	0x04	
Wrong data size	0x05	
No code set	0x06	

Table 5 Lock telegram

	7	6	5	4	3	2	1	0
0	SEQ		0x00					
1	0x04							
2		0x7FF						
3								
4	0x002							
5	SECURITY CODE							
6								
7								
8								

5.2.2.1 Set code

SET CODE		
Function code		0x003
Manufacturer Id		0x7FF
Data length		4 bytes
Data content	Security code	4 bytes
Unicast		yes
Broadcast		yes
Device responses to command		no
Status return code		
OK	0x00	
Wrong target ID	0x01	
Wrong manufacturer ID	0x04	
Wrong data size	0x05	

Table 6 Set code telegram

	7	6	5	4	3	2	1	0
0	SEQ		0x00					
1	0x04							
2		0x7FF						
3								
4	0x003							
5	SECURITY CODE							
6								
7								
8								

5.2.2.2 Query ID

QUERY ID		
Function code		0x004
Manufacturer Id		0x7FF
Data length		3 bytes
Data content	Desired EEP	21bits
	Mask bits	3bits
Unicast		no
Broadcast		yes
Device responsesto command		yes
Status return code		
	OK	0x00
	Wrong EEP	0x03
	Wrong manufacturer ID	0x04
	Wrong data size	0x05
	Not send	0x07
	Wrong data	0x0F

Table 7 Query Id telegram

	7	6	5	4	3	2	1	0
0	SEQ		0x00					
1	0x03							
2		0x7FF						
3								
4	0x004							
5	EEP							
6								
7						Mask bits		
8	NOT USED							

5.2.2.3 Query Id answer

QUERY ID ANSWER		
Function code		0x604
Manufacturer Id		Device Manufacturer ID
Data length		3 bytes
Data content	Desired EEP	21 bit
	Mask bits	3 bits
Unicast		yes
Broadcast		no

Table 8 Query Id answer telegram

	7	6	5	4	3	2	1	0
0	SEQ		0x00					
1	0x03							
2		Device Manufacturer ID						
3								
4	0x604							
5	EEP							
6								
7						0x00		
8	NOT USED							

5.2.3 Action

ACTION	
Function code	0x005
Manufacturer Id	0x7FF
Data length	0 bytes
Unicast	yes
Broadcast	yes
Device responses to command	no
Status return code	

OK	0x00
Wrong target Id	0x01
Wrong manufacturer Id	0x04

Table 9 Action telegram

	7	6	5	4	3	2	1	0
0	SEQ		0x00					
1	0x00							
2		0x7FF						
3								
4	0x005							
5	NOT USED							
6								
7								
8								

5.2.4 Ping

PING	
Function code	0x006
Manufacturer Id	0x7FF
Data length	0 bytes
Unicast	yes
Broadcast	no
Device responses to command	yes
Status return code OK 0x00 Wrong target Id 0x01 Wrong manufacturer Id 0x04 Not send 0x07	

Table 10 Ping telegram

	7	6	5	4	3	2	1	0
0	SEQ		0x00					
1	0x00							
2		0x7FF						
3								
4	0x006							
5	NOT USED							
6								
7								
8								

5.2.4.1 Ping answer

PING ANSWER		
Function code		0x606
Manufacturer Id		Device Manufacturer ID
Data length		4 bytes
Data content	The EEP of the remote device	21 bits
	Mask Bits	3 bits
	RSSI of received telegram by device	1 byte
Unicast		yes
Broadcast		no

Table 11 Ping answer telegram

	7	6	5	4	3	2	1	0
0	SEQ		0x00					
1	0x04							
2		Device Manufacturer ID						
3								
4	0x606							
5	EEP							
6								
7						0x00		
8	RSSI							

5.2.4.2 Query function

QUERY FUNCTION	
Function code	0x007
Manufacturer Id	0x7FF
Data length	0 bytes
Unicast	yes
Broadcast	no
Device responses to command	yes
Status return code OK 0x00 Wrong target Id 0x01 Wrong manufacturer Id 0x04 Not send 0x07	

Table 12 Query function telegram

	7	6	5	4	3	2	1	0
0	SEQ		0x00					
1	0x00							
2		0x7FF						
3								
4	0x007							
5	NOT USED							
6								
7								
8								

5.2.4.3 Query function answer

QUERY FUNCTION ANSWER	
Function code	0x607
Manufacturer Id	Device Manufacturer ID
Data length	n * 4 bytes
Data content	Supported extended function list with manufacturer ID
One list entry	Function number 2 bytes Manufacturer Id 2 bytes
Unicast	yes
Broadcast	no

n is the entry count of the RPC list

Table 13 Query function answer telegram IDX = 0

	7	6	5	4	3	2	1	0
0	SEQ		0x00					
1	0x04 * n							
2		Device Manufacturer ID						
3								
4	0x607							
5	NOT USED							
6	FUNCTION NUMBER							
7	NOT USED							
8	MANUFACTURER ID							

n is the entry count of the RPC list

5.2.4.4 Query function answer telegram IDX = 1

	7	6	5	4	3	2	1	0
0	SEQ		0x01					
1	NOT USED							
2	FUNCTION NUMBER							
3	NOT USED							
4	MANUFACTURER ID							
5	NOT USED							
6	FUNCTION NUMBER							
7	NOT USED							
8	MANUFACTURER ID							

NOTE: Query function answer length depends on the RPC list. All next list entries are transmitted in following telegrams and merged at target.

5.2.5 Query status

QUERY STATUS		
Function code		0x008
Manufacturer Id		0x7FF
Data length		0 bytes
Unicast		yes
Broadcast		yes
Device responses to command		yes
Status return code		
OK	0x00	
Wrong target Id	0x01	
Wrong manufacturer Id	0x04	
Not send	0x07	

Table 14 Query status telegram

	7	6	5	4	3	2	1	0
0	SEQ		0x00					
1	0x00							
2		0x7FF						
3								
4	0x008							
5	NOT USED							
6								
7								
8								

5.2.6 Query status answer

QUERY FUNCTION ANSWER		
Function code		0x60
Manufacturer Id		Device Manufacturer I
Data length		4 byte
Data content	Code set flag Last SEQ Last function code Last function return code Not used	1 bit 2 bits 12 bits 8 bits 9 bits
Unicast		ye
Broadcast		n

Table 15 Query status answer description

	7	6	5	4	3	2	1	0
0	SEQ		0x00					
1	0x04							
2		Device Manufacturer ID						
3								
4	0x608							
5	*	NOT USED					LAST SEQ	
6	NOT USED							
7	LAST FUNCTION NUMBER							
8	LAST FUNCTION RETURN CODE							

- - Code set flag

6 Important parameters

Table 16 Data length

Maximum number of SYS_EX message parts	64
Maximum length of transferred data	508 bytes

Table 17 Function Numbers

Available function numbers	(0x000 – 0xFFF)	4096
Reserved	(0x000)	1
Commands RMCC	(0x001 – 0x1FF)	511
Commands RPC	(0x200 – 0x5FF)	1024
Answers	(0x600 – 0xFFF)	2560

Table 18 Security code

Available security codes	(0x00000000 – 0xFFFFFFFF)	2^{32}
Reserved	(0x00000000), (0xFFFFFFFF)	2
Free	(0x00000001 – 0xFFFFFFFE)	2^{32} minus 2

Table 19 Periods

Chain period	1 s
Broadcast delay interval	0 s – 2 s
Power up unlock period	30 min
Unlock period	30 min
Security period	30 s