Technical Specification

EnOcean Serial Protocol Version 3 (ESP3)

V1.58

17-July 2025

Content

1 EnOcean Serial Protocol 3 (ESP3)

REVISION HISTORY

The following major modifications and improvements have recently been made to this document:

No.	Major Changes	Date	Who	Reviewer
1.50	Added the CO_WR_RSSITESTMODE and CO_RD_RSSITESTMODE Added CO_TX_DONE, CO_LRN_MODE_DISABLED events and CO_EVENT_SECURE_DEVICES subevent for device added and RLC sync Added the COMMAND_ACCEPTED and logic for long operations	2020-06-03	tm,dv	
1.51	Corrected the Teach-Info field on the CO_WR_SECUREDEVICEV2_ADD command.	2020-08-12	dv	
1.52	Corrected RADIO MESSAGE type optional length information.	2021-11-15	mf	
1.53	Added CO_SET_LEGACY_CRC and CO_GET_LEGACY_CRC	2023-08-18	dl	
1.54	Changed ERP2 optional data size to the correct value of 3 under 2.9.1	2024-01-08	mp	
1.55	Fixed optional length fields.	2024-06-13	sa	
1.56	Fixed response for CO_RD_SECUREDEVICE_MAINTENANCEKEY	2025-01-07	rs	
1.57	Initial review and typos fixes	2025-04-23	sa	mka, rs
1.58	Complete review and improvement	2025-07-17	mka	rs, sa

Published by EnOcean GmbH
Kolpingring 18a, 82041 Oberhaching
Germany

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

Important!

The information in this document provides a high-level functional description. It shall not be considered as assured characteristics for specific EnOcean devices. No responsibility is assumed for possible omissions or inaccuracies.

1.1 Terms and Abbreviations

Term / Abbr.	Description
API	Application Programming Interface
BIST	Built-in self-test
CRC-8	Cyclic redundancy check (CRC) or polynomial code checksum (CRC8 uses 9-bit polynomial length)
CRC8-D	CRC-8 for DATA and OPTIONAL DATA
CRC8-H	CRC-8 for HEADER
EEP	EnOcean Equipment Profile
ERP1	EnOcean Radio Protocol version 1 (Used mainly in Europe)
ERP2	EnOcean Radio Protocol version 2 (Used mainly outside of Europe)
ESP3	EnOcean Serial Protocol version 3
Group	Part of ESP3 packet (HEADER, DATA, OPTIONAL DATA)
Host	External device (e.g. a processor) communicating with the radio module
LSB	Least significant bit
Mailbox	Message filing of the Postmaster for each Smart Ack Sensor/Client
MSB	Most significant bit
Packet	ESP3 data unit
Packet Type	Type of ESP3 Packet (Command, Event, Radio, ...)
Postmaster	Includes multiple mailboxes for each Smart Ack Sensor/Client
R-ORG	Radio telegram type
RS-232	Serial communication standard for binary data and control signals
RSSI	Received signal strength indication (dBm)
SMART ACK	EnOcean standard for energy-optimized bidirectional transmission
SYNC Byte	Identifier for the start of an ESP3 packet
UART	Universal Asynchronous Receiver and Transmitter

1.2 Introduction

This document provides the specification for EnOcean Serial Protocol 3.0 (ESP3) as defined by EnOcean GmbH.

ESP3 defines a serial communication protocol between host systems such as microcontrollers, gateways or PCs and EnOcean radio transceiver modules such as TCM 310, TCM 410J, TCM 515 and TCM 615.

ESP3 communication is based on a 3-wire UART connection (using RX, TX and GND signals) with software handshake enabling full-duplex communication similar to an RS-232 serial interface as shown in Figure 1 below.

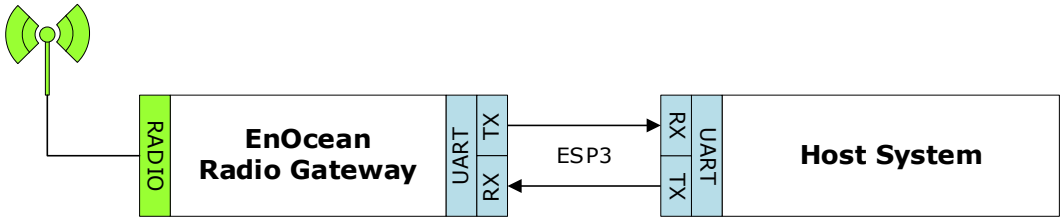

Figure 1

ESP3 enhances the previously used ESP2 protocol version by adding new structures and extending the data content. New functionality of ESP3 includes:

- Higher throughput due to a significantly higher baud rate
- Improved data security and consistency by CRC-8 Data verification
- More reliable ESP3 packet detection within the serial byte stream
- Option for backwards-compatible future extensions using the "Optional Data" feature
- Reporting of the radio signal strength and the number of the received subtelegrams

Table 1 below summarizes key differences between ESP2 and ESP3.

	ESP2	ESP3
Subtelegram count	--	•
Receive signal strength (RSSI)	--	•
Upward compatible with 'Optional Data'	--	•
Data verification	Checksum	CRC-8
UART Synchronization (packet detection)	2 bytes	6 bytes
Max. number of ESP packet types	8	256
Types of data	Radio, Command	Any type of data
Max. size of transferred data	28 bytes	65535 bytes
Communication speed	9600 baud	**57600 baud** 115200 baud 230400 baud 460800 baud

Table 1

1.3 Packet Structure

ESP3 is a point-to-point protocol using a packet data structure consisting of three groups:

- HEADER
  The HEADER group provides all required information to parse the ESP3 packet. This group contains the following items:
  - Data Length (number of bytes of the group Data)
  - Optional Data Length (number of bytes of the group Optional Data)
  - Packet Type (RADIO, RESPONSE, EVENT, COMMAND ...)

- DATA
  The DATA group contains the mandatory data of an ESP3 packet.
  The format of the DATA group will not change for a given ESP3 packet type (e.g. for a specific ESP3 command) to ensure backwards compatibility.

- OPTIONAL DATA
  The OPTIONAL DATA group may be used to extend an existing ESP3 packet in a backwards-compatible way.

In addition, ESP3 packets contain the following items to enable proper packet handling:
- Synchronization Byte (for detecting the start of the ESP3 packet)
- CRC-8 checksum for the content of the HEADER group
- CRC-8 checksum for the content of the DATA and OPTIONAL DATA groups

This generic ESP3 packet structure is shown in Figure 2 below.

<table>
<tr><td colspan="2">SYNC BYTE (1 byte)</td></tr>
<tr><td rowspan="3">HEADER</td><td>Length of DATA</td></tr>
<tr><td>Length of OPTIONAL DATA</td></tr>
<tr><td>PACKET TYPE</td></tr>
<tr><td colspan="2">CRC8 for HEADER (1 byte)</td></tr>
<tr><td>DATA</td><td>VARIABLE CONTENT</td></tr>
<tr><td>OPTIONAL DATA</td><td>VARIABLE CONTENT</td></tr>
<tr><td colspan="2">CRC8 for DATA and OPTIONAL DATA (1 byte)</td></tr>
</table>

Figure 2

1.4 Compatibility

ESP3 packet types and packet content (e.g. commands and their parameters) may only be extended in a backwards-compatible way as the protocol evolves over time.

This means that for a specific packet type (e.g. a specific ESP3 command), the content of the DATA group is immutable (i.e. it must not be changed in future implementations) to ensure backwards-compatibility.

Required modifications may be implemented by extending the content of the OPTIONAL DATA group. Products that do not know or support such extension shall ignore any unsupported or unknown content of the OPTIONAL DATA group.

If extension via the OPTIONAL DATA group is not sufficient to meet the implementation needs, then a new packet (e.g. a new ESP3 command) must be defined.

ESP3 devices shall react to new or modified ESP3 packets follows:

- Devices shall respond to unknown ESP3 packet types by using the RESPONSE 'Not Supported'. Such ESP3 packets shall not be processed further by the device.

- Devices shall ignore unknown or unsupported fields in the OPTIONAL DATA section of an existing (and supported) ESP3 packet type and process such packets as if these fields were not present.

- Devices may choose not to transmit all or some fields of the OPTIONAL DATA group.
  If some fields of the OPTIONAL DATA group are omitted, then those fields must be located contiguously at the end of the OPTIONAL DATA group. The receiving device shall assume that the default behavior applies if such fields are omitted by the transmitting device.

1.5 UART Parameters

ESP3 UART communication uses the following parameters:

- 8 data bits
- No parity bit
- One start bit (logical 0)
- One stop bit (logical 1)
- Line idle (≙neutral) is logical 1 (standard).

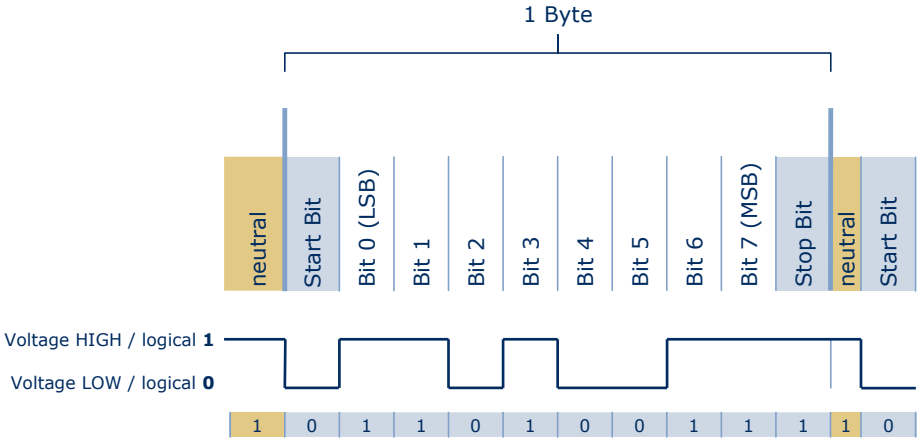

Figure 3

1.6 UART Synchronization (Start of Packet Detection)

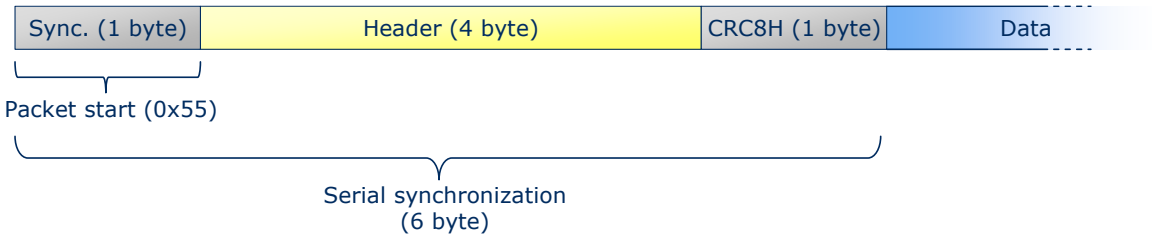

Figure 4

The start of an ESP3 packet is indicated by a SYNC BYTE which always has the value 0x55.

Once such SYNC BYTE is identified, the following 4 byte are assumed to form the HEADER group. The CRC-8 checksum (CRC8H) generated for these four bytes is then compared with the next received byte (which would be the corresponding CRC-8 for the HEADER group) to verify that this indeed is the start of an ESP3 packet.

If the received value does not match the CRC-8 value calculated for the 4 byte of HEADER, then the value 0x55 does not correspond to a SYNC BYTE indicating the start of an ESP3 packet and the decoding logic will wait for the next occurrence of 0x55 within the data stream.

Chapter 3.4 provides an example for such implementation.

1.7 Generic Packet Format

Table 2 below describes the generic format of ESP3 packets.

Group	Offset	Size	Field	Value hex	Description
-	0	1	SYNC BYTE	0x55	Serial synchronization byte This always has the value 0x55
Header	1	2	DATA Length	0xnnnn	Specifies the length (in bytes) of DATA
	3	1	OPTIONAL DATA Length	0xnn	Specifies the length (in bytes) of OPTIONAL_DATA
	4	1	PACKET Type	0xnn	Specifies the PACKET type
-	5	1	CRC-8 for HEADER	0xnn	CRC-8 for HEADER (CRC8H)
Data	6	x	... ...	... ...	Contains the data payload, for instance: - Raw Data (e.g. radio telegram content) - Function codes + optional parameters - RESPONSE Codes + optional parameters - Event codes
Optional Data	6+x	y	... ...	... ...	Contains additional data that extends the content / functionality of the DATA field
-	6+x+y	1	CRC-8 for DATA and OPTIONAL DATA	0xnn	CRC-8 for DATA and OPTIONAL_DATA (CRC8D)

Table 2

1.8 Packet Types

The ESP3 protocol defines a range of different packet types as listed in Table 3 below.

Packet ID	Packet Name	Packet Description
0x00	---	Reserved
0x01	RADIO_ERP1	ERP1 radio telegram
0x02	RESPONSE	Response (to the host)
0x03	RADIO_SUB_TEL	Radio subtelegram
0x04	EVENT	Event (to the host)
0x05	COMMON_COMMAND	Command (from the host)
0x06	SMART_ACK_COMMAND	Smart Acknowledge command
0x07	REMOTE_MAN_COMMAND	Remote Management command
0x08	---	Reserved
0x09	RADIO_MESSAGE	Radio message
0x0A	RADIO_ERP2	ERP2 radio telegram
0x0B	---	Reserved
0x0C	COMMAND_ACCEPTED	Command acceptance (to the host)
0x0D ... 0x0F	---	Reserved
0x10	RADIO_802.15.4	IEEE 802.15.4 radio telegram
0x11	2.4_GHZ_CONFIG	2.4 GHz radio configuration
0x12 ... 0xFF	---	Reserved

Table 3

1.9 Packet Flow

ESP3 packet flow might be uni-directional from the radio module to the host, uni-directional from the host to the radio module or bi-directional depending on the packet type.
Figure 5 below illustrates different examples which are subsequently described.

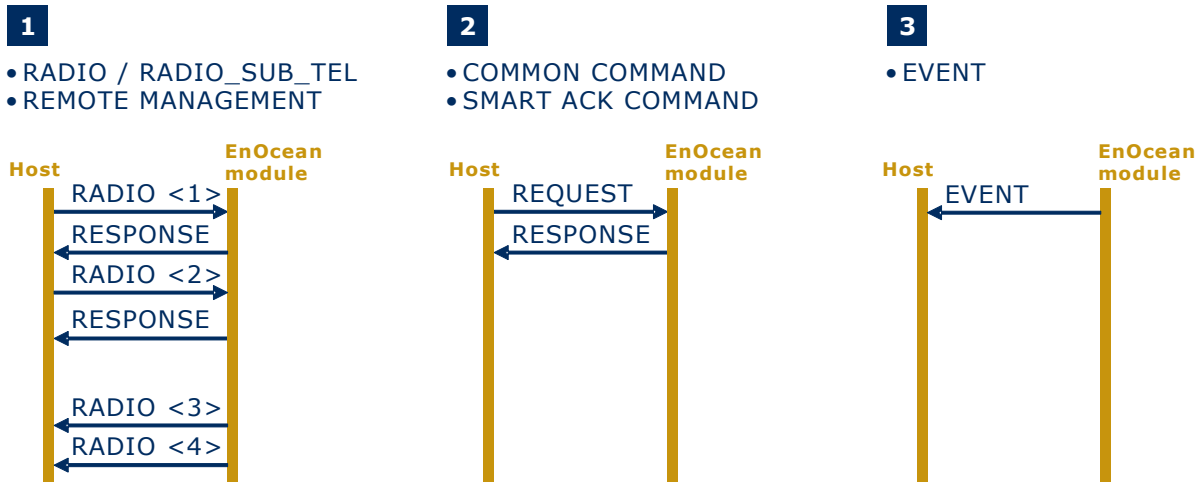

Figure 5

1 ESP3 radio packets such as RADIO_ERP1, RADIO_ERP2, RADIO_MESSAGE or REMOTE_MAN_COMMAND are bi-directional if sent by the host to the radio module (e.g. to transmit a message). The radio module will acknowledge processing of such messages to the host using a RESPONSE message.
For the case of reception (where these commands are sent by the radio module to the host), no RESPONSE message is required from the host to the radio module.

2 ESP3 request packets (such as COMMON_COMMAND and SMART_ACK) sent from the host to the radio module will be acknowledged by the radio module with a RESPONSE message (OK, ERROR, etc.). The reverse direction module-to-host is not possible.

3 ESP3 EVENT packets may be sent by the EnOcean radio gateway to the host to report specific conditions or events (for instance, security processing issues or duty cycle limits). No RESPONSE packet is required by the host for such EVENT packets.

1.10 Packet Time-out

An ESP3 time-out occurs if a required RESPONSE / COMMAND ACCEPTED from the radio module is not received within 500 milliseconds of the corresponding requesting packet from the host.

If an operation requires more than 500 milliseconds for execution, then the radio module may send a COMMAND_ACCEPTED packet instead of a RESPONSE packet to inform the host that it will perform the requested operation. Once the requested operation has been executed, the radio module shall then send a RESPONSE packet to the host.

2 ESP3 Command Description

2.1 Packet Type 0x01: RADIO_ERP1

The RADIO_ERP1 packet is used to transport the content of an ERP1 radio telegram from the host to the radio gateway (for telegram transmission) or from the radio gateway to the host (for telegram reception).

2.1.1 Structure

The structure of the RADIO_ERP1 packet is shown in Figure 6 below.

SYNC BYTE (1 byte)	
HEADER	Length of DATA
	Length of OPTIONAL DATA
	PACKET TYPE 0x01: RADIO_ERP1
CRC8 for HEADER (1 byte)	
DATA	Telegram R-ORG
	Telegram Data
	Telegram Sender (EURID)
	Telegram Status (STATUS)
OPTIONAL DATA	Number of Subtelegrams
	Telegram Detination (EURID)
	Signal Strength (dBm)
	Security Level
CRC8 for DATA and OPTIONAL DATA (1 byte)	

Figure 6

2.1.2 Description

The following packet fields are used with all radio telegram types that are transported using the RADIO_ERP1 packet. Examples for the DATA payload of different radio telegram types are provided in chapter 2.1.3.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0xnnnn	Variable length of radio telegram
	3	1	Optional Length	0x07	7 byte of Optional Data
	4	1	Packet Type	0x01	0x01: RADIO_ERP1
-	5	1	CRC8H	0xnn	
Data	6	x	... ...	... ...	ERP1 radio telegram content: Telegram R-ORG (1 byte) Telegram payload (1 … 14 byte) Telegram Sender ID (4 byte) Telegram Status (1 byte)
Optional Data	6+x	1	SubTelNum	0xnn	Number of sub-telegrams Send: Use 0x03 Receive: Number of received sub-telegrams
	7+x	4	Destination ID	0xnnnnnnnn	Broadcast: FFFFFFFF Addressed (ADT): Destination ID
	11+x	1	Signal Strength (dBm)	0xnn	Send: Use 0xFF Receive: Best RSSI value of all received sub-telegrams (value decimal without minus)
	12+x	1	Security Level	0xnn	Send: Will be ignored Receive: 0x00: Telegram was not processed 0x01: Obsolete (old security concept) 0x02: Telegram was decrypted 0x03: Telegram was authenticated 0x04: Telegram was decrypted + authenticated
-	13+x	1	CRC8D	0xnn	

Table 4

When receiving a telegram, no RESPONSE has to be sent.
When sending a telegram, one of the following RESPONSE messages is expected:

- 00: RET_OK
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM
- 05: RET_LOCK_SET

Note:
When sending a radio telegram (either as broadcast or as addressed transmission telegram) and the destination address has been setup in the outbound security link table, then Security Level will be determined by the corresponding entry in the security link table. The Security Level field provided in the RADIO_ERP1 packet will be ignored in this case.

When sending a radio telegram (either as broadcast or as addressed transmission telegram) and the destination address has not been setup in the outbound security link table, then no encryption / authentication will be used.

2.1.3 ERP1 Telegram Examples

The tables below show the content of the DATA group in RADIO_ERP1 packets for some common telegram types.

2.1.3.1 1BS Telegrams

Group	Offset	Size	Field	Value hex	Description
Data	6	1	R-ORG	0xD5	0xD5: Telegram type 1BS
	7	1	Telegram Data	0xnn...0xnn	1 byte data payload
	8	4	Sender ID	0xnnnnnnnn	Device-unique sender ID
	12	1	Status	0xnn	Telegram Status field

Table 5

2.1.3.2 4BS Telegrams

Group	Offset	Size	Field	Value hex	Description
Data	6	1	R-ORG	0xA5	0xA5: Telegram type 4BS
	7	4	Telegram Data	0xnn...0xnn	4 byte data payload
	11	4	Sender ID	0xnnnnnnnn	Sender ID
	15	1	Status	0xnn	Telegram Status field

Table 6

2.1.3.3 4BS Telegram using EEP A5-07-02

Group	Offset	Size	Field	Value hex	Description
Data	6	1	R-ORG	0xA5	0xA5: Telegram type 4BS
	7	1	Telegram Data Byte 3	0x00	Unused in this EEP profile
	8	1	Telegram Data Byte 2	0x00	Unused in this EEP profile
	9	1	Telegram Data Byte 1	0xnn	Temperature value 255 ... 0
	10	1	Telegram Data Byte 0	0b0000**n**000	DB_0.BIT 3 = Learn Bit Normal mode = 1 / Teach In = 0
	11	4	Sender ID	0xnnnnnnnn	Sender ID
	15	1	Status	0xnn	Telegram Status field

Table 7

2.1.3.4 VLD Telegrams

Group	Offset	Size	Field	Value hex	Description
Data	6	1	R-ORG	0xD2	0xD2: Telegram type VLD
	7	x	Telegram Data	0xnn...0xnn	1 ... 14 byte data payload
	7+x	4	Sender ID	0xnnnnnnnn	Sender ID
	11+x	1	Status	0xnn	Telegram Status field

Table 8

2.1.3.5 MSC Telegrams

Group	Offset	Size	Field	Value hex	Description
Data	6	1	R-ORG	0xD1	0xD1: Telegram type MSC
	7	x	Telegram Data	0xnn...0xnn	1 ... 14 byte data payload
	7+x	4	Sender ID	0xnnnnnnnn	Sender ID
	11+x	1	Status	0xnn	Telegram Status field

Table 9

2.1.3.6 ADT (Addressed Destination) VLD Telegram

Addressed Data Telegrams provide the intended Destination Address in the Optional Data field as shown below.

Group	Offset	Size	Field	Value hex	Description
Data	6	1	ADT	0xA6	0xA6: Telegram type ADT
	7	1	R-ORG	0xD2	0xD2: Telegram R-ORG VLD
	8	x	Telegram Data	0xnn...0xnn	1 ... 9 byte data payload
	8+x	4	Sender ID	0xnnnnnnnn	Device-unique sender ID
	12+x	1	Status	0xnn	Telegram Status field
Optional Data	13+x	1	SubTelNum	0xnn	Number of sub-telegrams Send: 0x03 Receive: 0x01 ... 0xFF
	14+x	4	Destination ID	0xnnnnnnnn	Destination ID (Address of the intended receiver)
	18+x	1	Signal Strength (dBm)	0xnn	Send: Omit or set to 0xFF Receive: Highest RSSI value of all received subtelegrams (Expressed as positive value)
	12+x	1	Security Level	0x0n	Send: Omit (Will be ignored if present) Receive: 0x00: Telegram was not processed 0x01: Obsolete (old security concept) 0x02: Telegram was decrypted 0x03: Telegram was authenticated 0x04: Telegram was decrypted + authenticated

Table 10

2.2 Packet Type 0x02: RESPONSE

2.2.1 Structure

The ESP3 packet type RESPONSE informs the host about the status of a previous request.

SYNC BYTE	
HEADER	Length of DATA
	Length of OPTIONAL DATA
	PACKET TYPE 0x02: RESPONSE
CRC8 for HEADER	
DATA	RESPONSE Code
	RESPONSE Data
OPTIONAL DATA	RESPONSE Optional Data
CRC8 for DATA and OPTIONAL DATA	

Figure 7

The type of the preceding request and the type of the provided RESPONSE code determines whether RESPONSE Data / Optional Data is included (as described in the corresponding chapters of this document) or if only the RESPONSE code (without any RESPONSE Data / Optional Data) is provided.

2.2.2 List of RESPONSE Codes

Code	Name	Description
0x00	RET_OK	The request was executed
0x01	RET_ERROR	The request resulted in an error
0x02	RET_NOT_SUPPORTED	The requested functionality is not supported by this module
0x03	RET_WRONG_PARAM	An incorrect parameter value was provided in the request
0x04	RET_OPERATION_DENIED	The requested operation is not permitted
0x05	RET_LOCK_SET	Duty cycle lock is active (This is temporarily preventing telegram transmission)
0x06	RET_BUFFER_TO_SMALL	The provided telegram is too long for transmission
0x07	RET_NO_FREE_BUFFER	The provided telegram cannot currently be transmitted due to other telegrams still being queued up for transmission
0x90	BASEID_OUT_OF_RANGE	The selected BaseID range is not valid
0x91	BASEID_MAX_REACHED	The maximum number of BaseID changes has been reached
OTHER	---	Reserved

Table 11

2.2.3 Structure of Generic RESPONSE Types

The following chapters illustrate the generic structure of commonly used RESPONSE types. These generic structures may be extended with additional data depending for specific requests as described in this document.

2.2.3.1 Generic Structure of RESPONSE RET_OK

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0001	1 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x02	0x02: RESPONSE
-	5	1	CRC8H	0xnn	
Data	6	1	RESPONSE Code	0x00	0x00: RET_OK
-	7	1	CRC8D	0xnn	

Table 12

2.2.3.2 Generic Structure of RESPONSE RET_ERROR

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0001	1 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x02	0x02: RESPONSE
-	5	1	CRC8H	0xnn	
Data	6	1	RESPONSE Code	0x01	0x01: RET_ERROR
-	7	1	CRC8D	0xnn	

Table 13

2.2.3.3 Generic Structure of RESPONSE RET_NOT_SUPPORTED

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0001	1 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x02	0x02: RESPONSE
-	5	1	CRC8H	0xnn	
Data	6	1	RESPONSE Code	0x02	0x02: RET_NOT_SUPPORTED
-	7	1	CRC8D	0xnn	

Table 14

2.2.3.4 Generic Structure of RESPONSE RET_WRONG_PARAM

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0001	1 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x02	0x02: RESPONSE
-	5	1	CRC8H	0xnn	
Data	6	1	RESPONSE Code	0x03	0x03: RET_WRONG_PARAM
-	7	1	CRC8D	0xnn	

Table 15

2.3 Packet Type 0x03: RADIO_SUB_TEL

The ESP3 packet type RADIO_SUB_TEL may be used for detailed analysis of radio conditions.

2.3.1 Structure

The RADIO_SUB_TEL packet extends the RADIO_ERP1 packet with additional telegram information provided in the OPTIONAL_DATA field as shown in Figure 8 below.

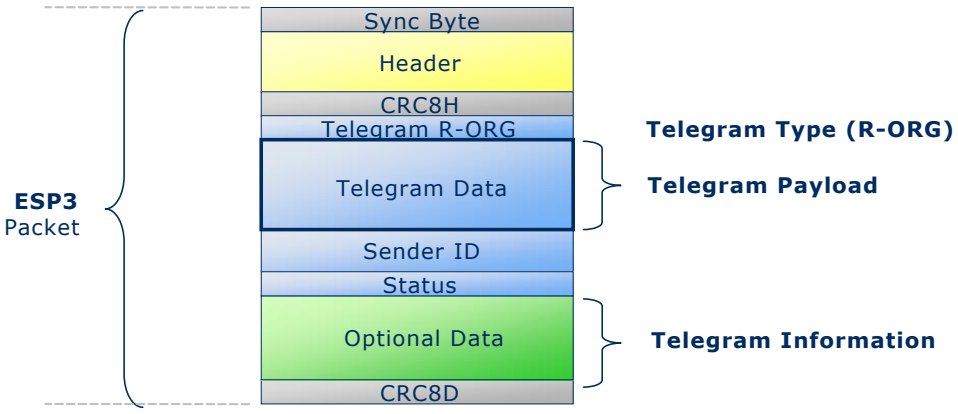

Figure 8

2.3.2 Description

The following fields are used in the RADIO_SUBTEL_PACKET.

Group	Offset	Size	Field	Value hex	Description	
-	0	1	Sync. Byte	0x55		
Header	1	2	Data Length	0xnnnn	Variable length (x byte of telegram data content)	
	3	1	Optional Length	0xnn	Variable length (9 + 3 * *s*) byte where *s* = number of received subtelegrams	
	4	1	Packet Type	0x03	0x03: RADIO_SUB_TEL	
-	5	1	CRC8H	0xnn		
Data	6	x	... ...	... ...	Telegram data content (length = *x*) of the radio telegram without Checksum/CRC	
Optional Data	6+x	1	SubTelNum	0xnn	Total number *s* of received sub-telegrams (including repeated sub-telegrams)	
	7+x	4	Destination ID	0xnnnnnnnn	Broadcast: 0xFFFFFFFF Addressed (ADT): Destination ID	
	11+x	1	dBm	0xnn	Send case: 0xFF Receive case: best RSSI value of all received subtelegrams (value decimal without minus)	
	12+x	1	Security Level	0xnn	Send Case: Will be ignored (Security is selected by link table entries) Receive case: 0x00: Telegram not processed 0x01: Obsolete (old security concept) 0x02: Telegram decrypted 0x03: Telegram authenticated 0x04: Telegram decrypted + authenticated	
	13+x	2	TimeStamp	0xnnnn	Reference timestamp of 1st sub-telegram (using system time with 1ms increments)	
	15+x+3*i	1	SubTelOffset	0xnn	Relative time offset (in ms) between this sub-telegram and the 1st sub-telegram	**s**
	16+x+3*i	1	SubTelRssi	0xnn	RSSI value of each subtelegram	
	17+x+3*i	1	SubTelStatus	0xnn	Telegram control bits of each subtelegram – used in case of repeating, switch telegram encapsulation, checksum type identification	
-	15+x+3*s	1	CRC8D	0xnn		

Table 16

The field *SubTelNum* provides the total number s of received subtelegrams.
For each of these sub-telegrams, the fields *SubTelOffset*, *SubTelRssi* and *SubTelStatus* provide information on time offset, received signal strength and telegram status (which can be used to determine the repeater level of that telegram). Index *i* is used to identify individual subtelegrams with *i = {0; s-1}.*

2.4 Packet Type 0x04: EVENT

An EVENT is primarily used as a notification about the result of processes and procedures. Some EVENT types require a RESPONSE to determine additional processing steps.

2.4.1 Structure

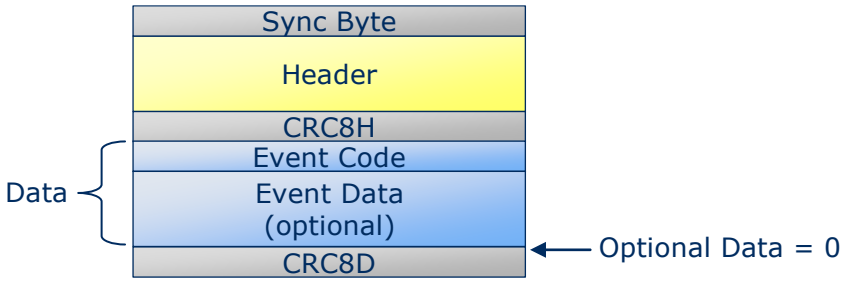

Figure 9

2.4.2 List of EVENT Codes

Code	Name	Description
0x01	SA_RECLAIM_NOT_SUCCESSFUL	Informs the host about an unsuccessful reclaim attempt by a Smart Ack client
0x02	SA_CONFIRM_LEARN	Request to the host for the handling of a received learn-in / learn-out request from a Smart Ack client
0x03	SA_LEARN_ACK	Response to the Smart Ack client about the result of its Smart Acknowledge learn request
0x04	CO_READY	Inform the host about readiness for operation
0x05	CO_EVENT_SECUREDEVICES	Informs the host about an event in relation to security processing
0x06	CO_DUTYCYCLE_LIMIT	Informs the host that the duty cycle limit has been reached
0x07	CO_TRANSMIT_FAILED	Informs the host about not being able to send a telegram
0x08	CO _TX_DONE	Informs the host about completion of transmission
0x09	CO_LRN_MODE_DISABLED	Informs the host that learn mode time-out has been reached

Table 17

2.4.3 Code 0x01: SA_RECLAIM_NOT_SUCCESSFUL

Function: Informs about an unsuccessful Smart Acknowledge client reclaim.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0001	1 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x04	0x04: EVENT
-	5	1	CRC8H	0xnn	
Data	6	1	Event Code	0x01	0x01: SA_RECLAIM_NOT_SUCCESSFUL
-	7	1	CRC8D	0xnn	

Table 18

2.4.4 Code 0x02: SA_CONFIRM_LEARN

Function: Request the external host to determine how to handle a received learn request.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0011	17 bytes of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x04	0x04: EVENT
-	5	1	CRC8H	0xnn	
Data	6	1	Event Code	0x02	0x02: SA_CONFIRM_LEARN
	7	1	Priority of the postmaster candidate	0xnn	Already post master 0b xxxx 1xxx Place for mailbox 0b xxxx x1xx Good RSSI 0b xxxx xx1x Local 0b xxxx xxx1
	8	1	2^2 ... 2^0: Manufacturer ID 2^7 ... 2^3: Res.	0b00000nnn	nnn = Most significant 3 bits of the Manufacturer ID 00000 = reserved
	9	1	Manufacturer ID	0xnn	Least significant bits of the Manufact. ID
	10	3	EEP	0xnnnnnn	Code of used EEP profile
	13	1	RSSI	0xnn	Signal strength; Send case: FF Receive case: actual RSSI
	14	4	Postmaster Candidate ID	0xnnnnnnnn	Device ID of the Postmaster candidate
	18	4	Smart Ack ClientID	0xnnnnnnnn	This sensor would be Learn IN
	22	1	Hop Count	0xnn	Number of repeater hops
-	23	1	CRC8D	0xnn	

Table 19

The host shall respond to such request with one of the following RESPONSE messages:

- 00: RET_OK (using the structure shown below)
- 01: RET_ERROR
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0004	4 bytes of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x02	0x02: RESPONSE
-	5	1	CRC8H	0xnn	
Data	6	1	RESPONSE Code	0x00	0x00: RET_OK
	7	2	Response time	0xnnnn	Response time for Smart Ack Client in ms in which the external host can prepare the data and send it to the postmaster. Only relevant if Confirm code is Learn IN
	9	1	Confirm code	0xnn	0x00: Learn IN 0x11: Discard Learn IN EEP not accepted 0x12: Discard Learn IN No mailbox space available 0x13: Discard Learn IN No space available at Controller 0x14: Discard Learn IN RSSI not sufficient 0x20: Learn OUT 0xFF: Function not supported
-	10	1	CRC8D	0xnn	

Table 20

2.4.5 Code 0x03: SA_LEARN_ACK

Function: Informs Smart Acknowledge client about the result of a previous sent learn request.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0004	4 bytes of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x04	0x04: EVENT
-	5	1	CRC8H	0xnn	
Data	6	1	Event Code	0x03	0x03: SA_LEARN_ACK
	7	2	Response time	0xnnnn	Response time for Smart Ack Client in ms in which the controller can prepare the data and send it to the postmaster. Only relevant if Confirm code is Learn IN
	9	1	Confirm code	0xnn	0x00: Learn IN 0x11: Discard Learn IN EEP not accepted 0x12: Discard Learn IN No mailbox space available 0x13: Discard Learn IN No space available at Controller 0x14: Discard Learn IN RSSI not sufficient 0x20: Learn OUT
-	10	1	CRC8D	0xnn	

Table 21

This EVENT does not require a RESPONSE message.

2.4.6 Code 0x04: CO_READY

Function: Informs host about the readiness for operation.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0002	2 bytes of Data
	3	1	Optional Length	0x01	1 byte of Optional Data
	4	1	Packet Type	0x04	0x04: EVENT
-	5	1	CRC8H	0xnn	
Data	6	1	Event Code	0x04	0x04: CO_READY
	7	1	Wakeup Cause	0xnn	0x00: Power-on Reset 0x01: HW Reset (via nRST pin) 0x02: Watchdog timer event 0x03: Periodic timer counter event 0x04 / 0x05: HW error 0x06: Memory access error 0x07: Wake-up via input pin 0 0x08: Wake-up via input pin 1 0x09: Unknown reset source 0x0A: UART Wake up 0x0B: SW reset
Optional Data	8	1	Mode	0xnn	0x00: Standard Security 0x01: Extended Security
-	8	1	CRC8D	0xnn	

Table 22

This EVENT does not require a RESPONSE message.

2.4.7 Code 0x05: CO_EVENT_SECUREDEVICES

Function: Informs external host about events regarding security processing

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0006	6 bytes of data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x04	0x04: EVENT
-	5	1	CRC8H	0xnn	
Data	6	1	Event Code	0x05	0x05: CO_EVENT_SECUREDEVICES
	7	1	Event Cause	0xnn	0x00: Teach-in failed No space available in secure link table. 0x01: Reserved 0x02: RLC resynchronization attempt with wrong security key 0x03: Configured number of telegrams with wrong CMAC received 0x04: Teach-In failed Incorrect security teach-in telegram 0x05: PSK Teach-In failed No PSK is set for the device 0x06: Teach-In failed. Use of PSK is required for this device 0x07: Authentication error CMAC or RLC not correct 0x08: Security level error Standard telegram from secure device 0x09: Teach-In successful Device added to secure link table 0x0A: RLC resync successful Updated RLC value 0x0B..0xFF: Reserved
	8	4	Device ID	0xnnnnnnnn	Device ID
-	12	1	CRC8D	0xnn	

Table 23

This EVENT does not require a RESPONSE message.

2.4.8 Code 0x06: CO_DUTYCYCLE_LIMIT

Function: Informs external host about duty cycle limit

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0002	2 bytes of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x04	0x04: EVENT
-	5	1	CRC8H	0xnn	
Data	6	1	Event Code	0x06	0x06: CO_DUTYCYCLE_LIMIT
	7	1	Event Cause	0xnn	0x00: Duty cycle limit not yet reached It is currently possible to send telegrams 0x01: Duty cycle limit reached It is currently not possible to send telegrams 0x02…0xFF: Reserved
-	8	1	CRC8D	0xnn	

Table 24

This EVENT does not require a RESPONSE message.

2.4.9 Code 0x07: CO_TRANSMIT_FAILED

Function: Informs the external host that the device was not able to send a telegram or that it did not receive an acknowledge for a transmitted telegram

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0002	2 bytes of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x04	0x04: EVENT
-	5	1	CRC8H	0xnn	
Data	6	1	Event Code	0x07	0x07: CO_TRANSMIT_FAILED
	7	1	Event Cause	0xnn	0x00: Channel Assesment failed Radio channel is occupied 0x01: No Acknowledge received 0x02…0xFF = reserved
-	8	1	CRC8D	0xnn	

Table 25

This EVENT does not require a RESPONSE message.

2.4.10 Code 0x08: CO_TX_DONE

Function: Informs the external host that the device has finished transmission.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0001	1 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x04	0x04: EVENT
-	5	1	CRC8H	0xnn	
Data	6	1	Event Code	0x08	0x08: CO_TX_DONE
-	7	1	CRC8D	0xnn	

Table 26

This EVENT does not require any RESPONSE message.

2.4.11 Code 0x09: CO_LRN_MODE_DISABLED

Function: Informs the external host that the learn mode has been disabled due to time-out.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0001	1 bytes of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x04	0x04: EVENT
-	5	1	CRC8H	0xnn	
Data	6	1	Event Code	0x09	0x09: CO_LRN_MODE_DISABLED
-	7	1	CRC8D	0xnn	

Table 27

This EVENT does not require any RESPONSE message.

2.5 Packet Type 0x05: COMMON_COMMAND

2.5.1 Structure

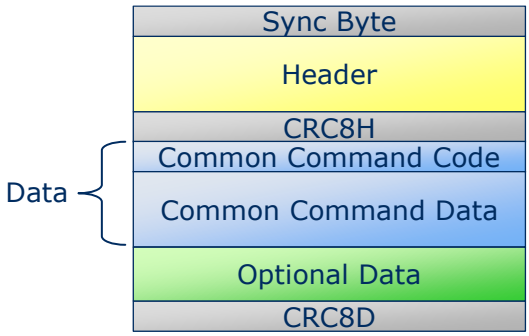

Figure 10

2.5.2 List of COMMAND Codes

Code	Function Name	Description
0x01	CO_WR_SLEEP	Enter energy saving mode
0x02	CO_WR_RESET	Reset the device
0x03	CO_RD_VERSION	Read the device version information
0x04	CO_RD_SYS_LOG	DEPRECATED (Read system log)
0x05	CO_RESET_SYS_LOG	DEPRECATED (Reset system log)
0x06	CO_WR_BIST	DEPRECATED (Perform self test)
0x07	CO_WR_IDBASE	Set ID range base address
0x08	CO_RD_IDBASE	Read ID range base address
0x09	CO_WR_REPEATER	Set Repeater Level
0x0A	CO_RD_REPEATER	Read Repeater Level
0x0B	CO_WR_FILTER_ADD	Add filter to filter list
0x0C	CO_WR_FILTER_DEL	Delete a specific filter from filter list
0x0D	CO_WR_FILTER_DEL_ALL	Delete all filters from filter list
0x0E	CO_WR_FILTER_ENABLE	Enable / disable filter list
0x0F	CO_RD_FILTER	Read filters from filter list
0x10	CO_WR_WAIT_MATURITY	Wait until the end of telegram maturity time before received radio telegrams will be forwarded to the external host
0x11	CO_WR_SUBTEL	Enable / Disable transmission of additional subtelegram info to the external host
0x12	CO_WR_MEM	Write data to device memory
0x13	CO_RD_MEM	Read data from device memory
0x14	CO_RD_MEM_ADDRESS	Read address and length of the configuration area and the Smart Ack Table
0x15	CO_RD_SECURITY	DEPRECATED (Read security information (level, key) of own device)
0x16	CO_WR_SECURITY	DEPRECATED (Write security information (level, key) for own device)
0x17	CO_WR_LEARNMODE	Enable / disable learn mode
0x18	CO_RD_LEARNMODE	Read learn mode status
0x19	CO_WR_SECUREDEVICE_ADD	DEPRECATED (Add a secure device)
0x1A	CO_WR_SECUREDEVICE_DEL	Delete a secure device from the link table
0x1B	CO_RD_SECUREDEVICE_BY_INDEX	DEPRECATED (Read secure device by index)
0x1C	CO_WR_MODE	Set the packet format used for forwarding received ERP2 telegrams to the host

0x1D	CO_RD_NUMSECUREDEVICES	Read the number of devices that are set up in the secure link table
0x1E	CO_RD_SECUREDEVICE_BY_ID	Read information about a specific device from the secure link table using the device ID
0x1F	CO_WR_SECUREDEVICE_ADD_PSK	Add pre-shared key (PSK) for a remote secure device
0x20	CO_WR_SECUREDEVICE_SENDTEACHIN	Send Security Teach-In message
0x21	CO_WR_TEMPORARY_RLC_WINDOW	Set a temporary rolling-code window for every taught-in device
0x22	CO_RD_SECUREDEVICE_PSK	Read the PSK of a device
0x23	CO_RD_DUTYCYCLE_LIMIT	Read the status of the duty cycle limit monitor
0x24	CO_SET_BAUDRATE	Set the baud rate used to communicate with the external host
0x25	CO_GET_FREQUENCY_INFO	Read the radio frequency and protocol supported by the device
0x26	Reserved	Reserved
0x27	CO_GET_STEPCODE	Read Hardware Step code and Revision of the Device
0x28	Reserved	Reserved
…	Reserved	Reserved
0x2D	Reserved	Reserved
0x2E	CO_WR_REMAN_CODE	Set the security code used to unlock Remote Management functionality via radio
0x2F	CO_WR_STARTUP_DELAY	Set the startup delay (Time from power up until start of operation)
0x30	CO_WR_REMAN_REPEATING	Select if REMAN telegrams originating from this module can be repeated
0x31	CO_RD_REMAN_REPEATING	Check if REMAN telegrams originating from this module can be repeated
0x32	CO_SET_NOISETHRESHOLD	DEPRECATED (Set the RSSI noise threshold level for telegram reception)
0x33	CO_GET_NOISETHRESHOLD	DEPRECATED (Read the RSSI noise threshold level for telegram reception)
0x34	CO_SET_CRC_MODE	Select CRC mode (7-bit or 8-bit mode)
0x35	CO_GET_CRC_MODE	Read the CRC mode (7-bit or 8-bit mode)
0x36	CO_WR_RLC_SAVE_PERIOD	Set the period in which outgoing RLCs are saved to the EEPROM
0x37	CO_WR_RLC_LEGACY_MODE	Activate the legacy RLC security mode allowing roll-over and using the RLC acceptance window for 24-bit explicit RLC
0x38	CO_WR_SECUREDEVICEV2_ADD	Add secure device to secure link table
0x39	CO_RD_SECUREDEVICEV2_BY_INDEX	Read secure device from secure link table using the table index
0x3A	CO_WR_RSSITEST_MODE	Control the state of the RSSI-Test mode
0x3B	CO_RD_RSSITEST_MODE	Read the state of the RSSI-Test Mode
0x3C	CO_WR_SECUREDEVICE_REMANKEY	Add the Reman (maintenance) key information into the secure link table
0x3D	CO_RD_SECUREDEVICE_REMANKEY	Read by index of the Reman (maintenance) key information from the secure link table
0x3E	CO_WR_TRANSPARENT_MODE	Control the state of the transparent mode
0x3F	CO_RD_TRANSPARENT_MODE	Read the state of the transparent mode
0x40	CO_WR_TX_ONLY_MODE	Control the state of the TX only mode
0x41	CO_RD_TX_ONLY_MODE	Read the state of the TX only mode

Table 28

2.5.3 Code 0x01: CO_WR_SLEEP

Function: Request to enter energy saving mode

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0005	5 bytes of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x01	0x01: CO_WR_SLEEP
	7	4	Sleep Period	0x00nnnnnn	0x00000000: Wake by UART Not supported on all devices 0x00000001 … 0x00FFFFFF: Duration of sleep in 10 ms units (maximum value ~ 46h)
-	11	1	CRC8D	0xnn	

Table 29

One of the following RESPONSE messages is expected in response to this request:
- 00: RET_OK
- 02: RET_NOT_SUPPORTED

2.5.4 Code 0x02: CO_WR_RESET

Function: Request to reset the device with the option to also reset persistently stored configuration parameters to their default values.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0001	1 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x02	0x02: CO_WR_RESET
Optional Data	7	1	Reset Type	0xnn	0x00: Restart application only 0x01: Restart application and reset parameters
-	7	1	CRC8D	0xnn	

Table 30

One of the following RESPONSE messages is expected in response to this request:
- 00: RET_OK
- 01: RET_ERROR
- 02: RET_NOT_SUPPORTED

2.5.5 Code 0x03: CO_RD_VERSION

Function: Read device SW version / HW version, chip-ID, etc.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0001	1 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x03	0x03: CO_RD_VERSION
-	7	1	CRC8D	0xnn	

Table 31

One of the following RESPONSE messages is expected in response to this request:
- 00: RET_OK (using the structure shown below)
- 02: RET_NOT_SUPPORTED

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0021	33 bytes of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x02	0x02: RESPONSE
-	5	1	CRC8H	0xnn	
Data	6	1	RESPONSE Code	0x00	0x00: RET_OK
	7	4	App Version	0xnnnnnnnn	Application Version Byte 1: Main version Byte 2: Beta version Byte 3: Alpha version Byte 4: Build
	11	4	SW Version	0xnnnnnnnn	SW (API) Version Byte 1: Main version Byte 2: Beta version Byte 3: Alpha version Byte 4: Build
	15	4	EURID	0xnnnnnnnn	Unique EnOcean Radio ID
	19	4	Device Version	0xnnnnnnnn	Device Version
	23	16	App Description	ASCII char	Application Description 16 ASCII characters (8-bit each) with null-termination
-	39	1	CRC8D	0xnn	

Table 32

2.5.6 Code 0x04: CO_RD_SYS_LOG

Function: Read System Log

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0001	1 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x04	0x04: CO_RD_SYS_LOG
-	7	1	CRC8D	0xnn	

Table 33

One of the following RESPONSE messages is expected in response to this request:
- 00: RET_OK (using the structure shown below)
- 02: RET_NOT_SUPPORTED

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0xnnnn	1+x bytes of Data
	3	1	Optional Length	0xnn	y bytes of Optional Data
	4	1	Packet Type	0x02	0x02: RESPONSE
-	5	1	CRC8H	0xnn	
Data	6	1	RESPONSE Code	0x00	0x00: RET_OK
	7	x	API Log Entry 000 API Log Entry 001 API Log Entry 002	0xnn 0xnn 0xnn	Log entry 000 - xxx in DATA: Log counter of API
Optional Data	7+x	y	APP Log Entry 000 APP Log Entry 001 APP Log Entry 002	0xnn 0xnn 0xnn	Log entry 000 - xxx in OPTIONAL_DATA: Log counter of APP
-	7+x+y	1	CRC8D	0xnn	

Table 34

After a reset, the counter starts with 0xFF and decrement with each new EVENT down to 0x00 and will then be stopped. After a reset command, the counter starts again with 0xFF.

2.5.7 Code 0x05: CO_RESET_SYS_LOG

Function: Reset System Log

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0001	1 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x05	0x05: CO_RESET_SYS_LOG
-	7	1	CRC8D	0xnn	

Table 35

One of the following RESPONSE messages is expected in response to this request:
- 00: RET_OK
- 02: RET_NOT_SUPPORTED

2.5.8 Code 0x06: CO_WR_BIST

Function: Perform Built-in-self-test

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0001	1 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x06	0x06: CO_WR_BIST
-	7	1	CRC8D	0xnn	

Table 36

One of the following RESPONSE messages is expected in response to this request:
- 00: RET_OK (using the structure shown below)
- 02: RET_NOT_SUPPORTED

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0002	2 bytes of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x02	0x02: RESPONSE
-	5	1	CRC8H	0xnn	
Data	6	1	RESPONSE Code	0x00	0x00: RET_OK
	7	1	BIST Result	0xnn	0x00: BIST OK Other: BIST Failed
-	8	1	CRC8D	0xnn	

Table 37

2.5.9 Code 0x07: CO_WR_IDBASE

Function: Set the start address of the Base ID range.

IMPORTANT: On TCM 3xx / TCM 4xx devices, the function to change the Base ID can only be used 10 times. There is no possibility to reset this constraint!

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0005	5 bytes of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x07	0x07: CO_WR_IDBASE
	7	4	Base ID	0xFFnnnnnn	Address between 0xFF800000 … 0xFFFFFF80
-	11	1	CRC8D	0xnn	

Table 38

One of the following RESPONSE messages is expected in response to this request:

- 0x00: RET_OK
- 0x02: RET_NOT_SUPPORTED
- 0x90: BASEID_OUT_OF_RANGE
- 0x91: BASEID_MAX_REACHED

The structure of these RESPONSE messages is shown below.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0001	1 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x02	0x02: RESPONSE
-	5	1	CRC8H	0xnn	
Data	6	1	RESPONSE Code	0xnn	0x00: RET_OK 0x02: RET_NOT_SUPPORTED 0x90: BASEID_OUT_OF_RANGE Incorrect address provided 0x91: BASEID_MAX_REACHED BaseID has been changed 10 times No more changes allowed
-	7	1	CRC8D	0xnn	

Table 39

2.5.10 Code 0x08: CO_RD_IDBASE

Function: Read start address of Base ID range

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0001	1 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x08	0x08: CO_RD_IDBASE
-	7	1	CRC8D	0xnn	

Table 40

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK (using the structure shown below)
- 02: RET_NOT_SUPPORTED

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0005	5 bytes of Data
	3	1	Optional Length	0x01	1 byte of Optional Data
	4	1	Packet Type	0x02	0x02: RESPONSE
-	5	1	CRC8H	0xnn	
Data	6	1	RESPONSE Code	0x00	0x00: RET_OK
	7	4	Base ID	0xFFnnnnnn	Start address of Base ID Range 0xFF800000 … 0xFFFFFF80
Optional Data	8	1	Remaining Base ID Write Cycles	0xnn	0x00 … 0xFE: Remaining Base ID write operations 0xFF: No limit for Base ID write operations
-	9	1	CRC8D	0xnn	

Table 41

2.5.11 Code 0x09: CO_WR_REPEATER

Function: Set Repeater Mode and Repeater Level

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0003	3 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x09	0x09: CO_WR_REPEATER
	7	1	Repeater Mode	0x00…0x02	Repeater Mode 0x00: OFF (Do not repeat telegrams) 0x01: ON (Repeat all telegrams) 0x02: SELECTIVE (Use filter list)
	8	1	Repeater Level	0x00…0x02	Repeater Level 0x00: OFF (use if Repeater Mode = 0x00) 0x01: 1-level repeating (when enabled) 0x02: 2-level repeating (when enabled)
-	9	1	CRC8D	0xnn	

Table 42

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM

2.5.12 Code 0x0A: CO_RD_REPEATER

Function: Read Repeater Mode

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0001	1 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x0A	0x0A: CO_RD_REPEATER
-	7	1	CRC8D	0xnn	

Table 43

One of the following RESPONSE messages is expected in response to this request:
- 00: RET_OK (using the structure shown below)
- 02: RET_NOT_SUPPORTED

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0003	3 bytes of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x02	0x02: RESPONSE
-	5	1	CRC8H	0xnn	
Data	6	1	RESPONSE Code	0x00	0x00: RET_OK
	7	1	Repeater Mode	0x00…0x02	Repeater Mode 0x00: OFF (Do not repeat telegrams) 0x01: ON (Repeat all telegrams) 0x02: SELECTIVE (Use filter list)
	8	1	Repeater Level	0x00…0x02	Repeater Level 0x00: OFF (used if REP_ENABLE = 0x00) 0x01: 1-level repeating (when enabled) 0x02: 2-level repeating (when enabled)
-	9	1	CRC8D	0xnn	

Table 44

2.5.13 Code 0x0B: CO_WR_FILTER_ADD

Function: Add filter to receiver filter list

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0007	7 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x0B	0x0B: CO_WR_FILTER_ADD
	7	1	Filter Criterion	0x00…0x03	Filter criterion to be used 0x00: Sender ID 0x01: Telegram type (R-ORG) 0x02: Minimum signal strength (dBm) 0x03: Destination ID
	8	4	Filter Value	0xnnnnnnnn	Value to compare filter criterium against - Sender ID - R-ORG - Signal strength (interpreted as negative dBm, e.g. 85 means -85 dBm) - Destination ID
	12	1	Filter Action	0x00 0x80 0x40 0xC0	0x00: Do not forward received telegram 0x80: Forward received telegram 0x40: Do not repeat received telegram 0xC0: Repeat received telegram
-	13	1	CRC8D	0xnn	

Table 45

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK
- 01: RET_ERROR (memory space full)
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM

2.5.13.1 **Additional information on filters**

Filters allow selecting only specific received telegrams for forwarding to the external host or for repeating (in case the repeater mode is set to SELECTIVE).

Both positive logic (forward / repeat specific telegrams) and negative logic (do not forward / repeat specific telegrams) are supported. For instance, it is possible to forward only telegrams originating from a specific device. Alternatively, it is possible not to repeat telegrams above a certain received signal strength (i.e. do not repeat telegrams originating from nearby senders).

More than one condition (filter) can be specified. If several filters are specified, then they can be combined either as logical OR (at least one filter condition must be true) or as logical AND (all filter conditions must be true).

2.5.13.2 Filter Examples

//Drop telegrams from a specified Sender ID
Filter_criterion = 0x00 (Filter by Sender ID)
Filter_value = 0x12345678 (Sender ID)
Filter_action = 0x00 (Do not forward received telegram)

//Drop all telegrams except that from a specified Sender ID
Filter_ criterion = 0x00 (Filter by Sender ID)
Filter_value = 0x12345678 (Sender ID)
Filter_action = 0x80 (Forward received telegram)

//Drop telegrams of a specific type (R-ORG)
Filter_ criterion = 0x01 (Filter by R-ORG)
Filter_value = 0xA5 (4BS Telegram Type)
Filter_action = 0x00 (Do not forward received telegram)

//Forward only telegrams of a specific type (R-ORG)
Filter_ criterion = 0x01 (Filter by R-ORG)
Filter_value = 0xA5 (4BS)
Filter_action = 0x80 (Forward received telegram)

//Drop telegrams below a minimum signal strength of -70dBm
Filter_ criterion = 0x02 (Filter by Minimum signal strength)
Filter_value = 0x00000046 (decimal 70 which means -70 dBm)
Filter_action = 0x00 (Do not forward received telegram)

//Repeat only telegrams from the specified sender ID (in SELECTIVE repeater mode)
Filter_ criterion = 0x00 (Filter by Sender ID)
Filter_value = 0x12345678 (device source ID)
Filter_action = 0xC0 (Repeat received Telegram)

//Do not repeat telegrams of a certain type (in SELECTIVE repeater mode)
Filter_ criterion = 0x01 (Filter by R-ORG)
Filter_value = 0xA5 (4BS)
Filter_action = 0x40 (Do not repeat received telegram)

// Do not repeat signals stronger than -70dBm (in SELECTIVE repeater mode)
Filter_ criterion = 0x02 (Filter by Minimum signal strength)
Filter_value = 0x00000046 (decimal 70 which means -70 dBm)
Filter_action = 0xC0 (Repeat received Telegram)

2.5.14 Code 0x0C: CO_WR_FILTER_DEL

Function: Delete a specific filter from filter list.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0007	7 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x0C	0x0C: CO_WR_FILTER_DEL
	7	1	Filter Criterion	0x00…0x03	Filter criterion to be used 0x00: Sender ID 0x01: Telegram type (R-ORG) 0x02: Minimum signal strength (dBm) 0x03: Destination ID
	8	4	Filter Value	0xnnnnnnnn	Value to compare filter criterium against - Sender or Destination ID - Telegram R-ORG - Signal strength (interpreted as negative dBm, e.g. 85 -> -85 dBm)
	9	1	Filter Action	0x00 0x80 0x40 0xC0	0x00: Do not forward received telegram 0x80: Forward received telegram 0x40: Do not repeat received telegram 0xC0: Repeat received telegram
-	12	1	CRC8D	0xnn	

Table 46

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK
- 01: RET_ERROR (memory space full or filter not found)
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM

2.5.15 Code 0x0D: CO_WR_FILTER_DEL_ALL

Function: Delete all filters from filter list

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0001	1 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x0D	0x0D: CO_WR_FILTER_DEL
-	7	1	CRC8D	0xnn	

Table 47

One of the following RESPONSE messages is expected in response to this request:
- 00: RET_OK
- 02: RET_NOT_SUPPORTED

2.5.16 Code 0x0E: CO_WR_FILTER_ENABLE

Function: Enable / disable the configure filters

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0003	3 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x0E	0x0E: CO_WR_FILTER_ENABLE
	7	1	Filter Enable	0x00 0x01	0x00: Filter Disabled (OFF) 0x01: Filter Enabled (ON)
	8	1	Filter Operator	0x00 0x01 0x08 0x09	0x00: OR combination of all filters 0x01: AND combination of all filters 0x08: OR combination for receive filters AND combination for repeat filters 0x09: AND combination for receive filters OR combination for repeat filters
-	9	1	CRC8D	0xnn	

Table 48

One of the following RESPONSE messages is expected in response to this request:
- 00: RET_OK
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM

2.5.17 Code 0x0F: CO_RD_FILTER

Function: Read all configured filters

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0001	1 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x0F	0x0F: CO_RD_FILTER
-	7	1	CRC8D	0xnn	

Table 49

One of the following RESPONSE messages is expected in response to this request:
- 00: RET_OK (using the syntax shown below)
- 02: RET_NOT_SUPPORTED

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0xnnnn	1 + 5*f bytes (f = number of filters)
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x02	0x02: RESPONSE
-	5	1	CRC8H	0xnn	
Data	6	1	RESPONSE Code	0x00	0x00: RET_OK
	7+5*f	1	Filter Criterion	0xnn	0x00 : Sender ID 0x01 : R-ORG 0x02 : RSSI (dBm) 0x03 : Destination ID
	8+5*f	4	Filter Value	0xnnnnnnnn	Value of filter function 'compare': - Sender or Destination ID - R-ORG - RSSI of radio telegram in dBm
-	12+5*f	1	CRC8D	0xnn	

f

Table 50

For each defined filter, one group **f** will be returned containing Filter Type and Filter Value.

2.5.18 Code 0x10: CO_WR_WAIT_MATURITY

Function: Select the desired sub-telegram processing mode.

The processing of sub-Telegrams belonging to the same radio telegram depends on the setting of WAIT_MATURITY as follows:

- If WAIT_MATURITY = 0x00 (Default)
  - The first received sub-telegram will be forwarded immediately to the host
  - Additional received sub-telegrams (belonging to the same telegram) will be discarded
  - This is the common setting during normal operation as it enables fastest possible reaction time by the host to a received telegram.

- If WAIT_MATURITY = 0x01 (Wait for maturity time)
  - After reception of the first sub-telegram, the receiver will wait for possible additional sub-telegrams (belonging to the same telegram) for the duration of the maturity time (100 milliseconds)
  - After maturity time has elapsed, the receiver will provide the received AND information about the number of received sub-telegrams, the repeater level of the first received sub-telegram and the highest RSSI of all received sub-telegrams
  - This mode is typically used during installation to determine the reliability of the radio connection

- If WAIT_MATURITY = 0x02 (Forward all subtelegrams)
  - All received sub-telegrams are forwarded immediately to the for processing. No merge or discard functionality is applied by the module
  - This mode is typically used in conjunction with specialized tools, for instance to determine specific properties of all received sub-telegrams
  - This mode is currently only supported for TCM 615 devices

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0002	2 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x10	0x10: CO_WR_WAIT_MATURITY
	7	1	Wait End Maturity	0xnn	0x00: Received telegrams are forwarded to the external host immediately. 0x01: Received telegrams are forwarded to the external host after the maturity time elapsed. 0x02: Every subtelegram is forwarded to the external host immediately. No subtelegram merging is performed.
-	8	1	CRC8D	0xnn	

Table 51

One of the following RESPONSE messages is expected in response to this request:
- 00: RET_OK
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM

2.5.19 Code 0x11: CO_WR_SUBTEL

Function: Enable / disable additional subtelegram info to be provided to the external host.
If subtelegram info is enabled in Compatible Mode (ERP1), telegrams will be forwarded with RADIO_SUB_TEL Type 3.
Note: This functionality is not supported by standard radio modules. It requires a transceiver with dedicated (Dolphin Sniffer) firmware.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0002	2 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x11	0x11: CO_WR_SUBTEL
	7	1	Enable Subtelegram Info	0xnn	0x00: Disable 0x01: Enable
-	8	1	CRC8D	0xnn	

Table 52

One of the following RESPONSE messages is expected in response to this request:
- 00: RET_OK
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM

2.5.20 Code 0x12: CO_WR_MEM

Function: Write data to memory (only supported on TCM 3xx / TCM 4xx platforms)

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0xnnnn	6 + x byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x12	0x12: CO_WR_MEM
	7	1	Memory Type	0xnn	0x00: Flash 0x01: RAM 0 0x02: Data RAM 0x03: Idata RAM 0x04: Xdata RAM 0x05: EEPROM
	8	4	Memory Address	0xnnnnnnnn	Start address to write
	12	x	Memory Content	0xnn ... 0xnn	Data content to write ... ...
-	12+x	1	CRC8D	0xnn	

Table 53

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM
- 04: RET_OPERATION_DENIED (memory access denied / code-protected)

2.5.21 Code 0x13: CO_RD_MEM

Function: Read data from memory (only supported on TCM 3xx / TCM 4xx platforms)

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0008	8 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x13	0x13: CO_RD_MEM
	7	1	Memory Type	0xnn	0x00: Flash 0x01: RAM 0 0x02: Data RAM 0x03: Idata RAM 0x04: Xdata RAM 0x05: EEPROM
	8	4	Memory Address	0xnnnnnnnn	Start address to read
	12	2	Data Length	0xnnnn	Data length to read
-	14	1	CRC8D	0xnn	

Table 54

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK (using the syntax shown below)
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM
- 04: RET_OPERATION_DENIED (memory access denied / code-protected)

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0xnnnn	1 + x bytes of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x02	0x02: RESPONSE
-	5	1	CRC8H	0xnn	
Data	6	1	RESPONSE Code	0x00	0x00: RET_OK
	7	x	Memory Content	0xnn ... 0xnn	Memory content ... ...
-	7+x	1	CRC8D	0xnn	

Table 55

2.5.22 Code 0x14: CO_RD_MEM_ADDRESS

Function: Read start address and length of the configuration area and the Smart Ack table (only supported on TCM 3xx / TCM 4xx platforms).

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0002	2 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x14	0x14: CO_RD_MEM_ADDRESS
	7	1	Memory Area	0xnn	0x00: Config area 0x01: Smart Ack table 0x02: System error log
-	8	1	CRC8D	0xnn	

Table 56

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK (using the syntax shown below)
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM
- 04: RET_OPERATION_DENIED (memory access denied / code-protected)

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x000A	10 bytes of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x02	0x02: RESPONSE
-	5	1	CRC8H	0xnn	
Data	6	1	RESPONSE Code	0x00	0x00: RET_OK
	7	1	Memory Type	0xnn	0x00: Flash 0x01: RAM 0 0x02: Data RAM 0x03: Idata RAM 0x04: Xdata RAM 0x05: EEPROM
	8	4	Memory Address	0xnnnnnnnn	Start address of config area / Smart Ack table / system error log
	12	4	Memory Length	0xnnnnnnnn	Data length of config area / Smart Ack table / system error log
-	16	1	CRC8D	0xnn	

Table 57

2.5.23 Code 0x15: CO_RD_SECURITY

Function: Read security information (mode, key). This function does not support the latest security concept and should not be used any more.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0001	1 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x15	0x15: CO_RD_SECURITY
-	7	1	CRC8D	0xnn	

Table 58

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK (using the syntax shown below)
- 02: RET_NOT_SUPPORTED

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x000A	10 bytes of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x02	0x02: RESPONSE
-	5	1	CRC8H	0xnn	
Data	6	1	RESPONSE Code	0x00	0x00: RET_OK
	7	1	Security Mode	0xnn	Security Mode
	8	4	Security Key	0xnnnnnnnn	Security Key
	12	4	Rolling Code	0x00000000	Reserved
-	16	1	CRC8D	0xnn	

Table 59

2.5.24 Code 0x16: CO_WR_SECURITY

Function: Write security information (mode, key). This function does not support the latest security concept and should not be used any more.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x000A	10 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x16	0x16: CO_WR_SECURITY
	7	1	Security Mode	0xnn	Security Mode
	8	4	Security Key	0xnnnnnnnn	Security Key
	12	4	Rolling Code	0x00000000	Reserved
-	16	1	CRC8D	0xnn	

Table 60

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK
- 01: RET_ERROR
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM

2.5.25 Code 0x17: CO_WR_LEARNMODE

Function: Enables or disable learn mode

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0006	6 byte of Data
	3	1	Optional Length	0x01	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x17	0x17: CO_WR_LEARNMODE
	7	1	Enable Learn Mode	0xnn	0x00: Stop Learn mode 0x01: Start Learn mode
	8	4	Timeout	0xnnnnnnnn	Time-out for the learn mode (if enabled) in milliseconds. If Timeout = 0, then the default period of 60.000 ms (60 seconds) is used
Optional Data	12	1	Channel	0xnn	0x00 … 0xFD: Channel No. absolute 0xFE: Previous channel relative 0xFF: Next channel relative
-	-	1	CRC8D	0xnn	

Table 61

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM

2.5.26 Code 0x18: CO_RD_LEARNMODE

Function: Read the learn-mode status

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0001	1 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x18	0x18: CO_RD_LEARNMODE
-	7	1	CRC8D	0xnn	

Table 62

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK (using the syntax below)
- 01: RET_ERROR
- 02: RET_NOT_SUPPORTED

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0002	2 byte of Data
	3	1	Optional Length	0x01	1 byte of Optional Data
	4	1	Packet Type	0x02	0x02: RESPONSE
-	5	1	CRC8H	0xnn	
Data	6	1	RESPONSE Code	0x00	0x00: RET_OK
	7	1	Learn Mode Status	0xnn	0x00: Learn mode not active 0x01: Learn mode active
Optional Data	8	1	Channel	0xnn	0x00 … 0xFD: Channel No. absolute 0xFE, 0xFF: Not used
-	-	1	CRC8D	0xnn	

Table 63

2.5.27 Code 0x19: CO_WR_SECUREDEVICE_ADD

Function: Add entry for a secure device to the secure link table. It is possible to add only one or more rockers with this function.

For newer devices (e.g. TCM 515), users should use CO_WR_SECUREDEVICEV2_ADD as this command supports 32-bit RLC and 1-byte for the Teach-in Info.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0019	25 byte of Data
	3	1	Optional Length	0x03	3 byte of Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x19	0x19: CO_WR_SECUREDEVICE_ADD
	7	1	SLF	0xnn	Security Level Format (SLF)
	8	4	Device ID	0xnnnnnnnn	Device ID
	12	16	Security Key	0xnnnnnnnn 0xnnnnnnnn 0xnnnnnnnn 0xnnnnnnnn	Security Key used by the device (16 byte)
	28	3	Rolling Code	0xnnnnnn	If a 16 bit rolling code is defined in SLF, the MSB is undefined
Optional Data	31	1	Direction	0xnn	Add device security information to: 0x00: Inbound table (default), ID = Source ID 0x01: Outbound table, ID = Destination ID 0x01: Outbound table, ID = own ID or 0x00000000 Secure broadcast 0x02: Outbound broadcast table, ID = Source ID (can be Device ID or Base ID) 0x03 … 0xFF: Not used
	32	1	PTM Module	0xnn	0x00: Device is a PTM module 0x01: Device is not a PTM module 0x02 … 0xFF: Not Used
	33	1	Teach-Info	0x0n	Secure device Teach-In info
-	-	1	CRC8D	0xnn	

Table 64

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK
- 01: RET_ERROR
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM

2.5.28 Code 0x1A: CO_WR_SECUREDEVICE_DEL

Function: Delete secure device from controller. It is only possible to delete ALL rockers of a secure device. If there was a Pre-Shared Key entry specified for that device, then it would be removed as well.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0005	5 byte of Data
	3	1	Optional Length	0x01	1 byte of Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x1A	0x1A: CO_WR_SECUREDEVICE_DEL
	7	4	Device ID	0xnnnnnnnn	Device ID to be deleted Using 0xFFFFFFFF will delete all devices (supported only by some devices, refer to user manual for further details).
Optional Data	8	1	Direction	0xnn	Remove this device from: 0x00: Inbound table (default) 0x01: Outbound table 0x02: Outbound broadcast table 0x03: Both inbound and outbound table 0x04 ... 0xFF: Not used
-	-	1	CRC8D	0xnn	

Table 65

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK
- 01: RET_ERROR
- 02: RET_NOT_SUPPORTED

2.5.29 Code 0x1B: CO_RD_SECUREDEVICE_BY_INDEX

Function: Read secure device by index. This function has been deprecated and is replaced by Code 0x39: CO_RD_SECUREDEVICEV2_BY_INDEX

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0002	2 byte of Data
	3	1	Optional Length	0x01	1 byte of Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x1B	0x1B: CO_RD_SECUREDEVICE
	7	1	Index	0x00…0xFE	Index of secure device to read (0…254) Maximum number dependent on device
Optional Data	8	1	Direction	0xnn	Read device security information from: 0x00: Inbound table (default) 0x01: Outbound table, 0x02: Outbound broadcast table 0x03 … 0xFF: Not used
-	9	1	CRC8D	0xnn	

Table 66

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK (using the syntax below)
- 01: RET_ERROR (entry not in use or out of bound)
- 02: RET_NOT_SUPPORTED

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0006	6 bytes of Data
	3	1	Optional Length	0x24	36 byte of Optional Data
	4	1	Packet Type	0x02	0x02: RESPONSE
-	5	1	CRC8H	0xnn	
Data	6	1	RESPONSE Code	0x00	0x00: RET_OK
	7	1	SLF	0xnn	Security Level Format (SLF)
	8	4	ID	0xnnnnnnnn	Device ID
Optional Data	12	16	Private Key	0xnnnnnnnn 0xnnnnnnnn 0xnnnnnnnn 0xnnnnnnnn	Security Key used by the device (16 byte)
	28	3	Rolling Code	0xnnnnnn	Rolling Code used by the device If a 16 bit rolling code is defined in SLF, then the MSB is undefined
	31	16	PSK	0xnnnnnnnn 0xnnnnnnnn 0xnnnnnnnn 0xnnnnnnnn	Pre-shared key used by the device (16 bytes, will be to all 0x00 if not present)
	47	1	Teach-In info	0xnn	Teach-In info of the device
-	47	1	CRC8D	0xnn	

Table 67

2.5.30 Code 0x1C: CO_WR_MODE

Function: Sets the packet format used by the radio module to forward ERP2 telegrams to the host. Note that this option is only available on radio modules for 902 MHz (US / Canada).

Two packet formats are available:

- RADIO_ERP1 (Packet Type 0x01)
  Radio module uses RADIO_ERP1 (Packet Type 0x01) to forward received radio telegrams to the host
- RADIO_ERP2 (Packet Type 0x0A)
  Radio module uses RADIO_ERP2 (Packet Type 0x0A) to forward received radio telegrams to the host

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0002	2 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x1C	0x1C: CO_WR_MODE
	6	1	Packet Type	0xnn	0x00: RADIO_ERP1 (default) 0x01: RADIO_ERP2
-	7	1	CRC8D	0xnn	

Table 68

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK
- 01: RET_ERROR
- 02: RET_NOT_SUPPORTED

2.5.31 Code 0x1D: CO_RD_NUMSECUREDEVICES

Function: Read number of learned-in secure devices

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0001	1 byte of Data
	3	1	Optional Length	0x00...0x01	0 or 1 byte of Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x1D	0x1D: CO_RD_NUMSECUREDEVICES
Optional Data	7	1	Direction	0xnn	0x00: Inbound table (default) 0x01: Outbound table 0x02: Outbound broadcast table 0x03: Total number of link table entries 0x04…0x0FF: Not used
-	8	1	CRC8D	0xnn	

Table 69

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK (using the syntax below)
- 02: RET_NOT_SUPPORTED

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0002	2 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x02	0x02: RESPONSE
-	5	1	CRC8H	0xnn	
Data	6	1	RESPONSE Code	0x00	0x00: RET_OK
	7	1	Number	0xnn	Number of entries in the table(s)
-	8	1	CRC8D	0xnn	

Table 70

2.5.32 Code 0x1E: CO_RD_SECUREDEVICE_BY_ID

Function: Read index of secure device entry by device ID

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0005	5 byte of Data
	3	1	Optional Length	0x00...0x01	0 or 1 byte of Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x1E	0x1E: CO_RD_SECUREDEVICE_BY_ID
	7	4	ID	0xnnnnnnnn	Device ID
Optional Data	11	1	Direction	0xnn	0x00: Inbound table (default) 0x01: Outbound table 0x02: Outbound broadcast table 0x03: Reman table 0x04 ... 0xFF: Not used
-	12	1	CRC8D	0xnn	

Table 71

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK (using the syntax below)
- 01: RET_ERROR (Device ID not found)
- 02: RET_NOT_SUPPORTED

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0002	2 bytes
	3	1	Optional Length	0x01	1 byte
	4	1	Packet Type	0x02	0x02: RESPONSE
-	5	1	CRC8H	0xnn	
Data	6	1	RESPONSE Code	0x00	0x00: RET_OK
	7	1	SLF	0xnn	Security Level Format
Optional Data	8	1	Index	0x00…0xFE	Index of this device in the security link table
-	9	1	CRC8D	0xnn	

Table 72

2.5.33 Code 0x1F: CO_WR_SECUREDEVICE_ADD_PSK

Function: Add pre-shared key for inbound secure device.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0015	21 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x1F	0x1F: CO_WR_SECUREDEVICE_ADD_PSK
	8	4	ID	0xnnnnnnnn	Device ID
	12	16	Pre-Shared Key	0xnnnnnnnn 0xnnnnnnnn 0xnnnnnnnn 0xnnnnnnnn	Pre-shared key (PSK) used by the device (16 bytes)
-	-	1	CRC8D	0xnn	

Table 73

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK (using the syntax below)
- 01: RET_ERROR (no entry in link table available)
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM (added device known, but private key wrong)

2.5.34 Code 0x20: CO_WR_SECUREDEVICE_SENDTEACHIN

Function: Send a secure teach-in message.

It is required that the outbound secure link table contains an entry for the device. Use CO_WR_SECUREDEVICE_ADD / CO_WR_SECUREDEVICEV2_ADD to add the device to the outbound link table.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0005	5 byte of Data
	3	1	Optional Length	0x00…0x01	0 or 1 byte of Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x20	0x20: CO_WR_SECUREDEVICE_SENDTEACHIN
	8	4	ID	0xnnnnnnnn	Device ID
Optional Data	8	1	TeachInInfo	0xnn	Teach-In Info If not present: Teach-in info from the link table is used
-	-	1	CRC8D	0xnn	

Table 74

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK
- 01: RET_ERROR (error sending teach-in telegram)
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM (device not found in link table)

2.5.35 Code 0x21: CO_WR_TEMPORARY_RLC_WINDOW

Function: Set a temporary rolling-code window for every taught-in device. If a telegram from a taught-in secure telegram is received, then the RLC window for this device will be reset to the standard value (128).

This function is intended for the case where the receiver did not receive secure telegrams for an extended period of time e.g. due to power loss. Using this command enables RLC re-synchronization over a wider window under such conditions:

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0006	6 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x21	0x21: CO_WR_TEMPORARY_RLC_WINDOW
	7	1	Enable	0xnn	0x00: Disable temporary RLC window RLC window = 128 will be used 0x01: Enable temporary RLC window Use RLC Window size
	8	4	RLC Window	0xnnnnnnnn	Temporary RLC window size
-	-	1	CRC8D	0xnn	

Table 75

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK
- 01: RET_ERROR (device not in list)
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM (if using zero as RLC window)

2.5.36 Code 0x22: CO_RD_SECUREDEVICE_PSK

Function: Read pre-shared key for inbound secure device or for the module itself.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0005	5 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x22	0x22: CO_RD_SECUREDEVICE_PSK
	8	4	ID	0xnnnnnnnn	Device ID 0x00000000: Return the PSK used by this device Other ID: Return the PSK used by other devices
-	-	1	CRC8D	0xnn	

Table 76

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK (using the syntax below)
- 01: RET_ERROR (no PSK assigned to the ID)
- 02: RET_NOT_SUPPORTED

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0011	17 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x02	0x02: RESPONSE
-	5	1	CRC8H	0xnn	
Data	6	1	RESPONSE Code	0x00	0x00: RET_OK
	7	16	PSK	0xnnnnnnnn 0xnnnnnnnn 0xnnnnnnnn 0xnnnnnnnn	Pre-Shared Key used by the device (16 byte)
-	12	1	CRC8D	0xnn	

Table 77

2.5.37 Code 0x23: CO_RD_DUTYCYCLE_LIMIT

Function: Read status of duty cycle supervisor (for 868 MHz ASK products)

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0001	1 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x23	0x23: CO_RD_DUTYCYCLE_LIMIT
-	-	1	CRC8D	0xnn	

Table 78

One of the following RESPONSE messages is expected in response to this request:
- 00: RET_OK (using the syntax below)
- 02: RET_NOT_SUPPORTED

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0008	8 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x02	0x02: RESPONSE
-	5	1	CRC8H	0xnn	
Data	6	1	RESPONSE Code	0x00	0x00: RET_OK
	7	1	Available duty cycle	0x00 ... 0x64	Number of available 1% duty cycle slots (between 0 … 100%)
	8	1	Slots	0xnn	Total number of duty cycle slots
	9	2	Slot period	0xnnnn	Period of one slot (in seconds)
	11	2	Time left in current slot	0xnnnn	Time left in current slot (in seconds)
	13	1	Available duty cycle after current slot	0x00 … 0x64	Duty cycle available after current slot (between 0 … 100%)
-	14	1	CRC8D	0xnn	

Table 79

2.5.38 Code 0x24: CO_SET_BAUDRATE

Function: Modifies the baud rate of the ESP3 interface (the default baud rate is 57600 bit per second). This function is currently only supported by TCM 515 and TCM 615 devices.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0002	2 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x24	0x24: CO_SET_BAUDRATE
	7	1	BAUDRATE	0xnn	0x00: 57600 bps 0x01: 115200 bps 0x02: 230400 bps 0x03: 460800 bps
-	-	1	CRC8D	0xnn	

Table 80

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK
- 02: RET_NOT_SUPPORTED

Note:
The response is sent with the current baud rate settings. If the baud rate can be modified as requested, then the response is subsequently sent again with the new baud rate.

2.5.39 Code 0x25: CO_GET_FREQUENCY_INFO

Function: Read frequency and protocol used by the transceiver

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0001	1 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x25	0x25: CO_GET_FREQUENCY_INFO
-	7	1	CRC8D	0xnn	

Table 81

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK (using the syntax below)
- 02: RET_NOT_SUPPORTED

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0003	3 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x02	0x02: RESPONSE
-	5	1	CRC8H	0xnn	
Data	6	1	RESPONSE Code	0x00	0x00: RET_OK
	7	1	Frequency	0xnn	0x00: 315.000 MHz 0x01: 868.300 MHz 0x02: 902.875 MHz 0x03: 921.400 MHz 0x04: 928.350 MHz 0x20: 2.4 GHz
	8	1	Protocol	0xnn	0x00: ERP1 0x01: ERP2 0x10: IEEE 802.15.4 0x30: Long Range
-	9	1	CRC8D	0xnn	

Table 82

2.5.40 Code 0x27: CO_GET_STEPCODE

Function: Read Hardware Step code and Revision of the Device.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0001	1 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x27	0x27: CO_GET_STEPCODE
-	7	1	CRC8D	0xnn	

Table 83

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK (using the syntax below)
- 02: RET_NOT_SUPPORTED

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0003	3 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x02	0x02: RESPONSE
-	5	1	CRC8H	0xnn	
Data	6	1	RESPONSE Code	0x00	0x00: RET_OK
	7	1	Step Code	0xnn	e.g. 0xCA, 0xDB, …
	8	1	Revision	0xnn	e.g. 0x01, 0x02, …
-	9	1	CRC8D	0xnn	

Table 84

2.5.41 Code 0x2E: CO_WR_REMAN_CODE

Function: Set the security code required to unlock Remote Management functionality by radio.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0005	5 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x2E	0x2E: CO_WR_REMAN_CODE
	7	4	Secure Code	0xnnnnnnnn	Secure code for unlocking device
-	11	1	CRC8D	0xnn	

Table 85

One of the following RESPONSE messages is expected in response to this request:
- 00: RET_OK
- 01: RET_ERROR (Hardware Error)
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM

2.5.42 Code 0x2F: CO_WR_STARTUP_DELAY

Function: Set the startup delay (time between power up and enabling the receiver)

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0002	2 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x2F	0x2F: CO_WR_STARTUP_DELAY
Data	7	1	Startup Delay	0xnn	Startup delay (as multiple of 10 ms)
-	8	1	CRC8D	0xnn	

Table 86

Start-up Delay = 1 -> Start-up delay = 10ms
Start-up Delay = 90 -> Start-up delay = 900ms

If the minimum time required by the device for start-up is less than the start-up delay specified by this command, then the minimum time required for start-up will be used.

One of the following RESPONSE messages is expected in response to this request:
- 00: RET_OK
- 01: RET_ERROR (Hardware Error)
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM

2.5.43 Code 0x30: CO_WR_REMAN_REPEATING

Function: Select if REMAN telegrams originating from the radio module can be repeated by another device. Note that this feature is not supported by most products.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0002	2 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x30	0x30: CO_WR_REMAN_REPEATING
Data	7	1	Reman Repeating	0xnn	0x00: Reman telegrams will not be repeated (STATUS will be set to 0x8F) 0x01: Reman answers will be repeated (STATUS will be set to 0x80)
-	8	1	CRC8D	0xnn	

Table 87

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM

2.5.44 Code 0x31: CO_RD_REMAN_REPEATING

Function: Check if the REMAN telegrams originating from this module can be repeated.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0001	1 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x31	0x31: CO_RD_REMAN_REPEATING
-	7	1	CRC8D	0xnn	

Table 88

One of the following RESPONSE messages is expected in response to this request:
- 00: RET_OK (using the syntax below)
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0002	2 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x02	0x02: RESPONSE
-	5	1	CRC8H	0xnn	
Data	6	1	RESPONSE Code	0x00	0x00: RET_OK
	7	1	Reman Repeating	0xnn	0x00: Reman telegrams will not be repeated (STATUS will be set to 0x8F) 0x01: Reman answers will be repeated (STATUS will be set to 0x80)
-	8	1	CRC8D	0xnn	

Table 89

2.5.45 Code 0x32: CO_SET_NOISETHRESHOLD

Function: Sets the RSSI noise rejection threshold level for telegram reception. Received Signals below this threshold will be filtered out.
Note that this command is currently only supported for TCM 515 devices with Revision DB and DC.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0002	2 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x32	0x32: CO_SET_NOISETHRESHOLD
	7	1	RSSI Level	0xnn	Minimum RSSI Level for data telegrams (positive offset from the theoretical noise minimum of -146 dBm)
-	8	1	CRC8D	0xnn	

Table 8390

One of the following RESPONSE messages is expected in response to this request:
- 00: RET_OK
- 01: RET_ERROR (Hardware Error)
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM

2.5.46 Code 0x33: CO_GET_NOISETHRESHOLD

Function: Read the RSSI noise threshold level for telegram detection.
Note that this command is only supported for TCM 515 Revision DB and DC.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0001	1 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x33	0x33: CO_GET_NOISETHRESHOLD
-	7	1	CRC8D	0xnn	

Table 91

One of the following RESPONSE messages is expected in response to this request:
- 00: RET_OK (using the syntax below)
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0002	2 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x02	0x02: RESPONSE
-	5	1	CRC8H	0xnn	
Data	6	1	RESPONSE Code	0x00	0x00: RET_OK
	7	1	RSSI Level	0xnn	Minimum RSSI Level for data telegrams (positive offset from the theoretical noise minimum of -146 dBm)
-	8	1	CRC8D	0xnn	

Table 92

2.5.47 Code 0x34: CO_SET_ CRC_MODE

Function: Selects if the CRC mode in ERP2 checks 7-bit or 8-bit. Use of 7-bit CRC mode is recommended for interoperability between TCM 515J and older devices. This command is only supported for TCM 515J.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0002	2 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x34	0x34: CO_SET_ CRC_MODE
	7	1	CRC Mode	0xnn	0x00: Use 8-bit CRC mode 0x01: Use 7-bit CRC mode
-	8	1	CRC8D	0xnn	

Table 93

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM

2.5.48 Code 0x35: CO_GET_ CRC_MODE

Function: Gets the Status of checked MSB in ERP2 the CRC.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0001	1 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x35	0x35: CO_GET_CRC_MODE
-	7	1	CRC8D	0xnn	

Table 94

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK (using the syntax below)
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0002	2 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x02	0x02: RESPONSE
-	5	1	CRC8H	0xnn	
Data	6	1	RESPONSE Code	0x00	0x00: RET_OK
	7	1	CRC Mode	0xnn	0x00: 8-bit CRC mode 0x01: 7-bit CRC mode (legacy)
-	8	1	CRC8D	0xnn	

Table 95

2.5.49 Code 0x36: CO_WR_RLC_SAVE_PERIOD

Function: Select how frequently the rolling codes in the outbound secure link table will be backed up to the persistent memory (EEPROM or Flash).

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0002	2 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x36	0x36: CO_WR_RLC_SAVE_PERIOD
Data	7	1	Save Period	0xnn	0x00: RLC will be saved immediately 0x01 … 0xFF: RLC will be saved every *n* updates
-	8	1	CRC8D	0xnn	

Table 96

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM

2.5.50 Code 0x37: CO_WR_RLC_LEGACY_MODE

Function: Select between standard and legacy RLC processing mode for 24-bit RLC:

- Standard mode
  RLC roll-over is not allowed and no RLC window is used
- Legacy mode
  RLC roll-over is allowed, but all RLCs must be within an RLC window of 128

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0002	2 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x37	0x37: CO_WR_RLC_LEGACY_MODE
Data	7	1	RLC Mode	0xnn	0x00: Standard Mode Roll-over is not allowed and RLC window is not used 0x01: Legacy Mode 24-bit RLC explicit mode Roll-over is allowed and RLC window is used
-	8	1	CRC8D	0xnn	

Table 97

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM

2.5.51 Code 0x38: CO_WR_SECUREDEVICEV2_ADD

Function: Add device to a secure link table.
This is a new version of this command to allow for 32 bit RLC.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x001B	27 byte of Data
	3	1	Optional Length	0x01	1 byte Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x38	0x38: CO_WR_SECUREDEVICE2_ADD
	7	1	SLF	0xnn	Security Level Format
	8	4	ID	0xnnnnnnnn	Device ID
	12	16	Private Key	0xnnnnnnnn 0xnnnnnnnn 0xnnnnnnnn 0xnnnnnnnn	Private key used by the device (16 byte)
	28	4	Rolling Code	0xnnnnnnnn	Most recently used Rolling Code If 24 or 16 bit rolling code is used, then the most significant bytes are undefined
	32	1	Teach-In-Info	0x0n	TEACH_IN_INFO
Optional Data	33	1	Direction	0xnn	0x00: Inbound table (default) Device ID is ID of Remote Device 0x01: Outbound table (addressed), Device ID is ID of Remote Device 0x02: Outbound table (broadcast) Device ID is own ID or Base ID 0x03 … 0xFF: Not used
-	34	1	CRC8D	0xnn	

Table 98

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK
- 01: RET_ERROR (No space available)
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM

2.5.52 Code 0x39: CO_RD_SECUREDEVICEV2_BY_INDEX

Function: Read device from secure link table by index. New version allows for 32 bit RLC.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0002	2 byte of Data
	3	1	Optional Length	0x01	1 byte Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x39	0x39: CO_RD_SECUREDEVICE
	7	1	Index	0x00…0xFE	Index of secure device to read (between 0 … 254, depending on device)
Optional Data	8	1	Direction	0xnn	Read device security information from: 0x00: Inbound table (default) 0x01: Outbound table (addressed) 0x02: Outbound table (broadcast) 0x03 … 0xFF: Not used
-	9	1	CRC8D	0xnn	

Table 99

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK (using the syntax below)
- 01: RET_ERROR (Device not found in link table)
- 02: RET_NOT_SUPPORTED

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x001B	27 bytes
	3	1	Optional Length	0x10	16 bytes
	4	1	Packet Type	0x02	0x02: RESPONSE
-	5	1	CRC8H	0xnn	
Data	6	1	RESPONSE Code	0x00	0x00: RET_OK
	7	1	SLF	0xnn	Security Level Format
	8	4	ID	0xnnnnnnnn	Device ID
	12	16	Private Key	0xnnnnnnnn 0xnnnnnnnn 0xnnnnnnnn 0xnnnnnnnn	Private key used by the device (16 bytes)
	28	4	Rolling Code	0xnnnnnnnn	If a 24/16 bit rolling code is defined in SLF, the MSBs are undefined
	32	1	Teach-Info	0xnn	TEACH_IN_INFO
Optional Data	31	16	PSK	0xnnnnnnnn 0xnnnnnnnn 0xnnnnnnnn 0xnnnnnnnn	Pre-shared key used by the device (16 byte, will be set to all 0x00 if not present)
-	47	1	CRC8D	0xnn	

Table 100

Note: If PSK was not set, it will not be included in the packet. If in the future response will be extended, all bytes of non-existing PSK will be set to 0x00.

2.5.53 Code 0x3A: CO_WR_RSSITEST_MODE

Function: This command enables / disables the RSSI Test Mode. If test mode is enabled, then the device will send a SIGNAL telegram of type RX Channel Quality (MID: 0x0A) for every non-filtered telegram.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0004	4 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x3A	0x3A: CO_WR_RSSITEST_MODE
	7	1	RSSI Test Mode	0xnn	0x00: Disable the RSSI Test Mode 0x01: Enable the RSSI Test Mode
	8	2	Timeout	0xnnnnn	0x0000: No Timeout. RSSI test mode will be active until disabled or Reset 0x0001 … 0xFFFF: Timeout in seconds
-	12	1	CRC8D	0xnn	

Table 101

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK
- 01: RET_ERROR (Hardware Error)
- 02: RET_NOT_SUPPORTED

2.5.54 Code 0x3B: CO_RD_RSSITEST_MODE

Function: This command reads if the RSSI Test Mode. The device will send a Channel Quality Signal Telegram (MID: 0x0A) for every non-filtered telegram.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0001	1 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x3B	0x3B: CO_RD_RSSITEST_MODE
-	7	1	CRC8D	0xnn	

Table 102

One of the following RESPONSE messages is expected in response to this request:
- 00: RET_OK (using the syntax below)
- 01: RET_ERROR (Device not found in link table)
- 02: RET_NOT_SUPPORTED

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0002	2 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x02	0x02: RESPONSE
-	5	1	CRC8H	0xnn	
Data	6	1	RESPONSE Code	0x00	0x00: RET_OK
	7	1	RSSI Test Mode	0xnn	0x00: RSSI Test Mode is disabled 0x01: RSSI Test Mode is enabled
-	8	1	CRC8D	0xnn	

Table 103

2.5.55 Code 0x3C: CO_WR_SECUREDEVICE_REMANKEY

Function: Add a Reman (maintenance) key for one device to the secure link table.

This command will automatically create an outbound and inbound RLC entry. If the device receives a Remote Command Package via ESP3, it will automatically encrypt it using the Reman (maintenance) key pair. It will also decrypt the responses using the same key pair.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0016	22 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x3C	0x3C: CO_WR_SECUREDEVICE_REMANKEY
	7	4	ID	0xnnnnnnnn	Device ID
	11	16	Reman Key	0xnnnnnnnn 0xnnnnnnnn 0xnnnnnnnn 0xnnnnnnnn	Reman key used by the device (16 byte)
	27	1	Reman Key Number	0xnn	0x01 … 0x0F: Identifies the key that is used by the device for Reman messages. Only one key pair is supported by a device.
-	28	1	CRC8D	0xnn	

Table 104

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK
- 01: RET_ERROR (No more space available in link table)
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM (wrong private key)

2.5.56 Code 0x3D: CO_RD_SECUREDEVICE_REMANKEY

Function: Read Reman (maintenance) key from secure link table by index.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0002	2 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x3D	0x3D: CO_RD_SECUREDEVICE_REMANKEY
	7	1	Index	0x00…0xFE	0x01 … 0x0F: Identifies the key that shall be read. Only one key pair is supported by a device.
-	9	1	CRC8D	0xnn	

Table 105

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK (using the syntax below
- 01: RET_ERROR (Device not found in link table)
- 02: RET_NOT_SUPPORTED

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x001F	31 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x02	0x02: RESPONSE
-	5	1	CRC8H	0xnn	
Data	6	1	RESPONSE Code	0x00	0x00: RET_OK
	7	1	Index	0x00…0xFE	Index of the key that is provided
	8	4	ID	0xnnnnnnnn	Device ID
	12	16	Private Key	0xnnnnnnnn 0xnnnnnnnn 0xnnnnnnnn 0xnnnnnnnn	Private key of the device (16 byte)
	28	1	Key Number	0xnn	0x01 … 0x0F: Reman Key Number
	29	4	Inbound Rolling Code	0xnnnnnnnn	Most recently received RLC
	33	4	Outbound Rolling Code	0xnnnnnnnn	Most recently transmitted RLC
-	37	1	CRC8D	0xnn	

Table 106

2.5.57 Code 0x3E: CO_WR_TRANSPARENT_MODE

Function: This command enables/disables transparent mode.

If transparent mode is enabled, then upper layer processing functions such as telegram de-chaining, decryption, authentication and remote management filtering is disabled.

The implementation of this command is device-specific.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0002	2 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x3E	0x3E: CO_WR_TRANSPARENT_MODE
	7	1	Transparent Mode	0xnn	0x00: Disable Transparent Mode 0x01: Enable Transparent Mode
-	8	1	CRC8D	0xnn	

Table 107

One of the following RESPONSE messages is expected in response to this request:
- 00: RET_OK
- 02: RET_NOT_SUPPORTED

2.5.58 Code 0x3F: CO_RD_TRANSPARENT_MODE

Function: This command reads the transparent mode state.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0001	1 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x3F	0x3F: CO_RD_TRANSPARENT_MODE
-	7	1	CRC8D	0xnn	

Table 108

One of the following RESPONSE messages is expected in response to this request:
- 00: RET_OK (using the syntax below)
- 02: RET_NOT_SUPPORTED

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0002	2 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x02	0x02: RESPONSE
-	5	1	CRC8H	0xnn	
Data	6	1	RESPONSE Code	0x00	0x00: RET_OK
	7	1	Transparent Mode	0xnn	0x00: Transparent mode is disabled 0x01: Transparent mode is enabled
-	8	1	CRC8D	0xnn	

Table 109

2.5.59 Code 0x40: CO_WR_TX_ONLY_MODE

Function: This command enables or disables the TX-only mode.

If TX-only mode is enabled, then all RX functionalities will be disabled and the device will enter a low power state after transmission has completed.

While in TX-only mode, the device will send an event (CO_TX_DONE) after the transmission of a telegram is complete (i.e. when all subtelegrams have been transmitted).

If auto-sleep is ON, then the device will automatically enter sleep-mode after completing a transmission and needs to be waken up again via UART or through other methods (see device user manual for further details).

If auto-sleep is OFF, then the device will wait for requests via the ESP3 interface; the receiver functionality will remain disabled.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0002	2 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x40	0x40: CO_WR_TX_ONLY_MODE
	7	1	TX Only Mode	0xnn	0x00: TX-only mode OFF 0x01: TX-only mode ON without auto-sleep 0x02: TX only mode ON with auto-sleep
-	8	1	CRC8D	0xnn	

Table 110

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK
- 02: RET_NOT_SUPPORTED

2.5.60 Code 0x41: CO_RD_TX_ONLY_MODE

Function: This command reads the current state of TX-only mode.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0001	1 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x05	0x05: COMMON_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x41	0x41: CO_RD_TX_ONLY_MODE
-	7	1	CRC8D	0xnn	

Table 111

One of the following RESPONSE messages is expected in response to this request:
- 00: RET_OK (using the syntax below)
- 02: RET_NOT_SUPPORTED

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0002	2 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x02	0x02: RESPONSE
-	5	1	CRC8H	0xnn	
Data	6	1	RESPONSE Code	0x00	0x00: RET_OK
	7	1	TX-only Mode	0xnn	0x00: TX-only mode OFF 0x01: TX-only mode ON without auto-sleep 0x02: TX only mode ON with auto-sleep
-	8	1	CRC8D	0xnn	

Table 112

2.6 Packet Type 0x06: SMART_ACK_COMMAND

SmartAck functionality enables energy-constrained devices (such as energy-harvesting switches) to retrieve messages from permanently powered devices (such as line-powered actuators or gateways) without the need to enable receiver functionality for long intervals.

2.6.1 Structure

The structure used by SMART_ACK_COMMAND is shown in Figure 11 below.

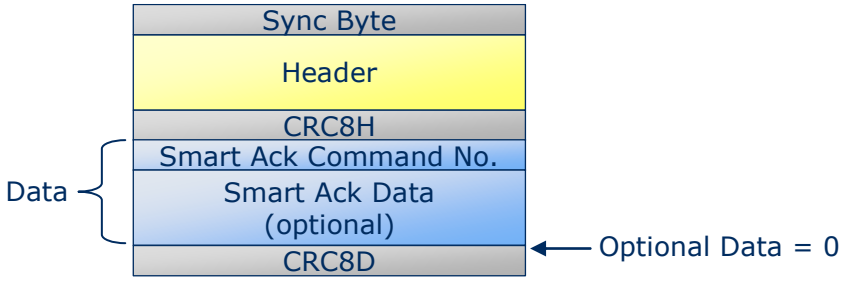

Figure 11

Currently defined packets of type SMART_ACK_COMMAND contain no Optional Data.

2.6.2 List of SMART ACK Codes

Code	Function Name	Description
0x01	SA_WR_LEARNMODE	Set/Reset Smart Ack learn mode
0x02	SA_RD_LEARNMODE	Get Smart Ack learn mode state
0x03	SA_WR_LEARNCONFIRM	Used for Smart Ack to add or delete a mailbox of a client
0x04	SA_WR_CLIENTLEARNRQ	Send Smart Ack Learn request (Client)
0x05	SA_WR_RESET	Send reset command to a Smart Ack client
0x06	SA_RD_LEARNEDCLIENTS	Get Smart Ack learned sensors / mailboxes
0x07	SA_WR_RECLAIMS	Set number of reclaim attempts
0x08	SA_WR_POSTMASTER	Activate/Deactivate Postmaster functionality

Table 113

2.6.3 Code 0x01: SA_WR_LEARNMODE

Function: Enable or disable learn mode of the Smart Acknowledge controller.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0007	7 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x06	0x06: SMART_ACK_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	SMART_ACK Code	0x01	0x01: SA_WR_LEARNMODE
	7	1	Enable	0x0n	0x00: End Learn mode 0x01: Start Learn mode
	8	1	Extended	0x0n	0x00: Simple Learn mode 0x01: Advanced Learn mode 0x02: Advanced Learn mode, select repeater
	9	4	Timeout	0xnnnnnnnn	Time-out for the learn mode in ms. When time is 0 then default period of 60’000 ms is used
-	13	1	CRC8D	0xnn	

Table 114

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM

2.6.4 Code 0x02: SA_RD_LEARNMODE

Function: Read the learn mode status of the Smart Acknowledge controller.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0001	1 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x06	0x06: SMART_ACK_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	SMART_ACK Code	0x02	0x02: SA_RD_LEARNMODE
-	7	1	CRC8D	0xnn	

Table 115

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK (using the syntax below)
- 02: RET_NOT_SUPPORTED

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0003	3 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x02	0x02: RESPONSE
-	5	1	CRC8H	0xnn	
Data	6	1	RESPONSE Code	0x00	0x00: RET_OK
	7	1	Enable	0xnn	0x00: Learn mode not active 0x01: Learn mode active
	8	1	Extended	0xnn	0x00: Simple Learn mode 0x01: Advanced Learn mode 0x02: Advanced Learn mode, select repeater
-	9	1	CRC8D	0xnn	

Table 116

2.6.5 Code 0x03: SA_WR_LEARNCONFIRM

Function: Send Smart Acknowledge learn answer to modify mailbox at Post Master.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x000C	12 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x06	0x06: SMART_ACK_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	SMART_ACK Code	0x03	0x03: SA_WR_LEARNCONFIRM
	7	2	Response time	0xnnnn	Response time for sensor in ms in which the controller can prepare the data and send it to the postmaster. Only relevant, if learn RESPONSE Code is Learn IN.
	9	1	Confirm code	0xnn	0x00: Learn IN 0x20: Learn OUT
	10	4	Postmaster Candidate ID	0xnnnnnnnn	Device ID of the used Post Master
	14	4	Smart Ack Client ID	0xnnnnnnnn	Device ID of the learned IN/OUT Smart Ack client.
-	18	1	CRC8D	0xnn	

Table 117

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM

2.6.6 Code 0x04: SA_WR_CLIENTLEARNRQ

Function: Sends Smart Acknowledge Learn Request telegram to the Smart Acknowledge controller. This function can only be used in a Smart Acknowledge client.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0006	6 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x06	0x06: SMART_ACK_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	SMART_ACK Code	0x04	0x04: SA_WR_ CLIENTLEARNRQ
	7	1	Manufacturer ID	0b11111nnn	nnn:Most significant 3 bits of the Manufacturer ID
	8	1	Manufacturer ID	0xnn	Least significant 8 bits of the Manufacturer ID
	9	3	EEP	0xnnnnnn	EEP of the Smart Ack client, who wants to Teach IN.
-	12	1	CRC8D	0xnn	

Table 118

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM

2.6.7 Code 0x05: SA_WR_RESET

Function: Send reset request to Smart Acknowledge client.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0005	5 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x06	0x06: SMART_ACK_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	SMART_ACK Code	0x05	0x05: SA_WR_ RESET
	7	4	Smart Ack Client ID	0xnnnnnnnn	Device ID of the Smart Ack client
-	11	1	CRC8D	0xnn	

Table 119

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM

2.6.8 Code 0x06: SA_RD_LEARNEDCLIENTS

Read mailbox information at the Smart Acknowledge controller to determine status of learned-in Smart Acknowledge clients.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0001	1 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x06	0x06: SMART_ACK_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	SMART_ACK Code	0x06	0x06: SA_RD_ LEARNEDCLIENTS
-	7	1	CRC8D	0xnn	

Table 120

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK (using the syntax below)
- 02: RET_NOT_SUPPORTED

Group	Offset	Size	Field	Value hex	Description	
-	0	1	Sync. Byte	0x55		
Header	1	2	Data Length	0xnnnn	1 + 9*c bytes of Data (c = number of clients)	
	3	1	Optional Length	0x00	No Optional Data	
	4	1	Packet Type	0x02	0x02: RESPONSE	
-	5	1	CRC8H	0xnn		
Data	6	1	RESPONSE Code	0x00	0x00: RET_OK	
	7+9*i	4	Client ID	0xnnnnnnnn	Device ID of the Smart Ack client	i
	11+9*i	4	Controller ID	0xnnnnnnnn	Controller ID dedicated to this Smart Ack client	
	15+9*i	1	Mailbox Index	0xnn	Internal counter of the Post Master (0x00 ... 0x0E)	
-	16+9*c	1	CRC8D	0xnn		

Table 121

For each of the *c* learned Smart Acknowledge clients, one group with index *i* will be returned where *i = {0; c-1}*.

2.6.9 Code 0x07: SA_WR_RECLAIMS

Function: Set the amount of reclaim tries in the Smart Acknowledge client.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0002	2 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x06	0x06: SMART_ACK_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	SMART_ACK Code	0x07	0x07: SA_WR_RECLAIMS
	7	1	Reclaim count	0xnn	Number of required reclaim tries
-	8	1	CRC8D	0xnn	

Table 122

One of the following RESPONSE messages is expected in response to this request:
- 00: RET_OK
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM

2.6.10 Code 0x08: SA_WR_POSTMASTER

Function: Enable / disable postmaster functionality at the Smart Acknowledge controller

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0002	2 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x06	0x06: SMART_ACK_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	SMART_ACK Code	0x08	0x08: SA_WR_POSTMASTER
	7	1	Mailbox count	0xnn	0x00: Disable postmaster functionality 0x01 … 0xFF: Number of mailboxes available (Device-dependent)
-	8	1	CRC8D	0xnn	

Table 123

One of the following RESPONSE messages is expected in response to this request:
- 00: RET_OK
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM (Mailbox count exceeds number of available mailboxes)

2.6.11 Code 0x09: SA_RD_MAILBOX STATUS

Read mailbox status for a specific Smart Acknowledge client.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0009	9 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x06	0x06: SMART_ACK_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	SMART_ACK Code	0x09	0x09: SA_RD_ MAILBOXSTATUS
	7	4	Client ID	0xnnnnnnnn	Device ID of the Smart Ack Client
	11	4	Controller ID	0xnnnnnnnn	Controller ID dedicated to this Smart Ack Client
-	15	1	CRC8D	0xnn	

Table 124

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK (using the syntax below)
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x02	2 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x02	0x02: RESPONSE
-	5	1	CRC8H	0xnn	
Data	6	1	RESPONSE Code	0x00	0x00: RET_OK
	7	1	MailBox Status	0xnn	0x00: Mailbox is empty 0x01: Mailbox is full 0x02: Mailbox does not exist
-	8	1	CRC8D	0xnn	

Table 125

2.6.12 Code 0x0A: SA_DEL_MAILBOX

Delete mailbox for a certain Smart Acknowledge client.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0009	1 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x06	0x06: SMART_ACK_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	1	SMART_ACK Code	0x0A	0x0A: SA_DEL_MAILBOX
	7	4	Client ID	0xnnnnnnnn	Device ID of the Smart Ack Client
	11	4	Controller ID	0xnnnnnnnn	Controller ID dedicated to this Smart Ack Client
-	15	1	CRC8D	0xnn	

Table 126

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK
- 01: RET_ERROR (Mailbox does not exist)
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM

2.7 Packet Type 0x07: REMOTE_MAN_COMMAND

REMOTE_MAN_COMMAND is used to transport Remote Management messages between host and radio module.

2.7.1 Structure

REMOTE_MAN_COMMAND uses the structure shown in Figure 12 below.

This structure is used both for sending (transmission) and for receiving (reception) of remote management messages.

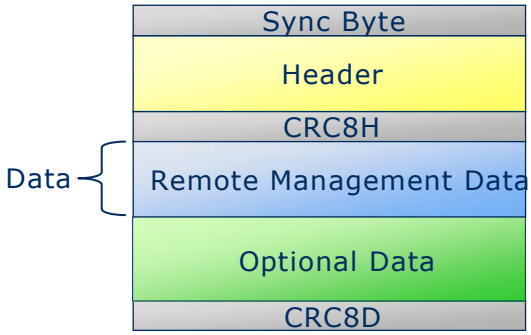

Figure 12

2.7.2 Description

Function: Send or receive a Remote Management message.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0xnnnn	4 + x bytes
	3	1	Optional Length	0x0A	10 bytes
	4	1	Packet Type	0x07	0x07: REMOTE_MAN_COMMAND
-	5	1	CRC8H	0xnn	
Data	6	2	Function No.	0x0nnn	0x0000 ... 0x0FFF
	8	2	Manufacturer ID	0x0nnn	0x0000 ... 0x07FF
	10	x	Message data	...	N bytes
Optional Data	10+x	4	Destination ID	0xnnnnnnnn	Destination ID Broadcast ID: FFFFFFFF
	14+x	4	Source ID	0xnnnnnnnn	Receive case: Source ID of the sender Send case: Set to 0x00000000
	18+x	1	dBm	0xnn	Send case: Set to 0xFF Receive case: Received signal strength (Value expressed as decimal number without the minus sign)
	19+x	1	Random Transmission Delay	0xnn	0x00: No random delay (default) 0x01: Message must be sent with random delay (must be used when replying to a broadcast message)
-	20+x	1	CRC8D	0xnn	

Table 127

For reception, there is no RESPONSE.
For transmission, one of the following RESPONSE messages is expected:

- 00: RET_OK
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM
- 05: RET_LOCK_SET

2.8 Packet Type 0x09: RADIO_MESSAGE

2.8.1 Structure

The RADIO_MESSAGE packet allows sending of messages irrespective or message size providing a unified interface to the host for the transmission of all message types and sizes.

Messages that are longer than the maximum supported payload size will automatically be chained (segmented) on transmission and de-chained (reassembled) on reception. Note that support for the RADIO_MESSAGE packet for transmission and / or reception is not available on all devices.

The DATA field of the RADIO_MESSAGE packet contains the payload of the radio message (payload data without any control fields) according to the structure shown below.

The OPTIONAL DATA field contains the control fields required for transmission or reported for reception of radio messages as defined in Table 128.

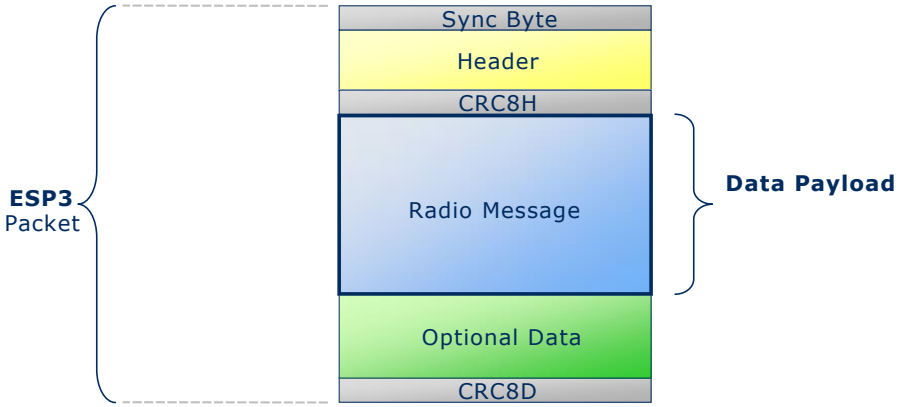

Figure 13

2.8.2 Description

The following structure is applicable to all types of radio messages:

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0xnnnn	Variable length of message
	3	1	Optional Length	0x0A	10 byte of Optional Data
	4	1	Packet Type	0x09	0x09: RADIO_MESSAGE
-	5	1	CRC8H	0xnn	
Data	6	x	Message Data	0xnn	Message Data Content
Optional Data	6+x	4	Destination ID	0xnnnnnnnn	Destination ID Broadcast ID: FFFFFFFF
	10+x	4	Source ID	0xnnnnnnnn	Transmit: Set to 0x00000000 Receive: Sender ID
	14+x	1	dBm	0xnn	Transmit: Set to 0xFF Receive: RSSI value of received sub-telegram (value decimal without minus)
	15+x	1	Security Level	0xnn	Send Case: Will be ignored (Security is selected by link table entries) Receive case: 0x00: Telegram unencrypted 0x01: Obsolete (old security concept) 0x02: Telegram encrypted 0x03: Telegram authenticated 0x04: Telegram encrypted + authenticated
-	16+x	1	CRC8D	0xnn	

Table 128

For reception, there is no RESPONSE.
For transmission, one of the following RESPONSE messages is expected:

- 00: RET_OK
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM
- 05: RET_LOCK_SET

2.9 Packet Type 0x0A: RADIO_ERP2

The RADIO_ERP2 packet contains the content of a radio telegram using ERP2 format.

2.9.1 Structure

The RADIO_ERP2 packet contains the ERP2 payload (excluding the LENGTH field).

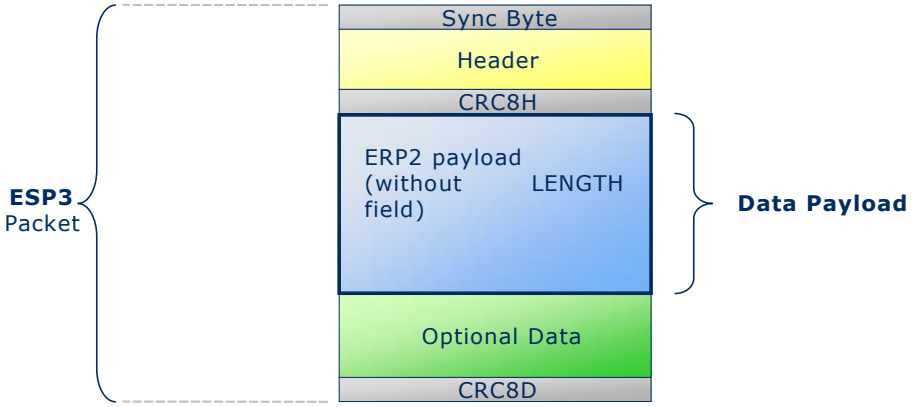

Figure 14

2.9.2 Description

The following structure is applicable to all types of radio telegrams:

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0xnnnn	Variable length of radio telegram
	3	1	Optional Length	0x03	3 byte of Optional Data
	4	1	Packet Type	0x0A	0x0A: RADIO_ERP2
-	5	1	CRC8H	0xnn	
Data	6	x	Message Data	... ...	Message data content excluding the first (Length) byte. For transmission, the CRC8 byte of the ERP2 protocol can be set to any value. x = Data Length
Optional Data	6+x	1	SubTelNum	0xnn	Number of sub telegrams Send: 3 Receive: 1 ... 255
	7+x	1	dBm	0xnn	Send: 0xFF Receive: Highest RSSI value of received sub-telegrams (value decimal without minus)
	8+x	1	Security Level	0x0n	Send: Will be ignored (Security is defined by link table entries) Receive: 0x00: No security processing 0x01: Obsolete (old security concept) 0x02: Telegram decrypted 0x03: Telegram authenticated 0x04: Telegram decrypted + authenticated
-	8+x	1	CRC8D	0xnn	

Table 129

For reception, there is no RESPONSE.
For transmission, one of the following RESPONSE messages is expected:

- 00: RET_OK
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM

2.10 Packet Type 0x0C: COMMAND ACCEPTED

2.10.1 Structure

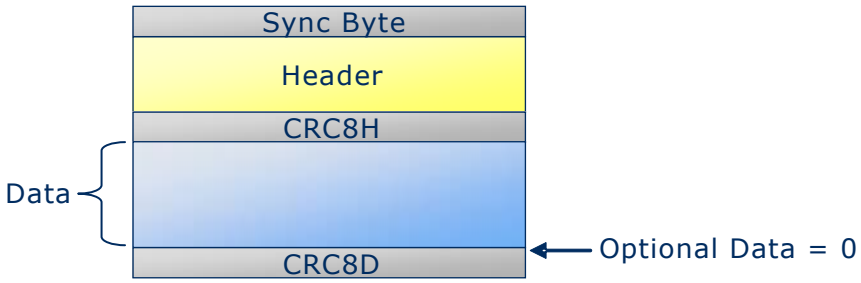

Figure 15

The COMMAND ACCEPTED packet may be transmitted if an ESP3 request has been received for which a RESPONSE packet must be sent after completing execution if the request is expected to complete only after exceeding the ESP3 time-out.

The required RESPONSE packet may then be sent after the ESP3 request has been executed.

2.10.2 Description

The table below shows the content of the COMMAND ACCEPTED structure.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0001	1 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x0C	0x0C: COMMAND_ACCEPTED
-	5	1	CRC8H	0xnn	
Data	6	1	Blocking Operation	0xnn	0x00: No blocking operation Additional commands can be accepted while the current command is executed 0x01: Blocking operation Additional commands will be rejected without RESPONSE while the current command is executed
	3	2	Estimated Completion Time	0xnnnn	0x0000: Unknown 0x0001 ... 0xFFFF: Estimated completion time (1 ms to 65535 ms)
-	7	1	CRC8D	0xnn	

Table 130

2.11 Packet Type 0x10: RADIO_802.15.4

This packet is sent from a module (supporting IEEE 802.15.4 reception) to the host upon receiving a valid IEEE 802.15.4 radio telegram.

If a module supporting IEEE 802.15.4 transmission receives such packet from the host, then the corresponding IEEE 802.15.4 radio telegram will be sent by the module.

2.11.1 Structure

The ESP3 RADIO_802.15.4 packet encapsulates an entire 802.15.4 MAC Frame except the 2-byte FCS field.

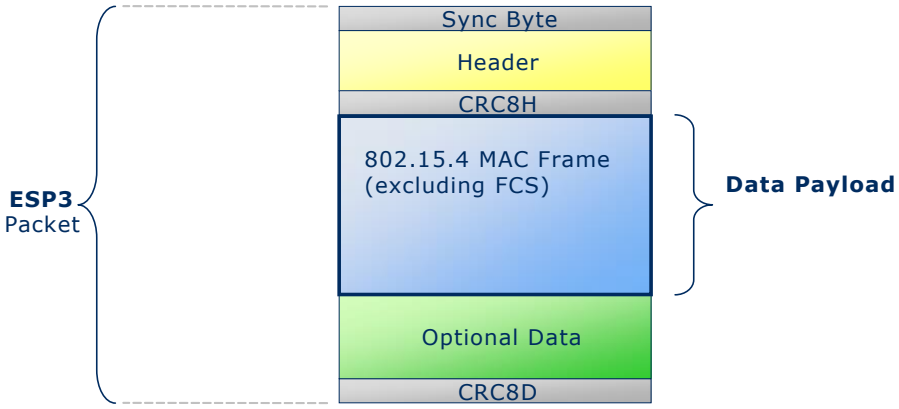

Figure 16

The structure of the 802.15.4 MAC frame is shown in Figure 17 below.

Octets: 2	1	0/2	0/2/8	0/2	0/2/8	0/5/6/10/14	variable	2
Frame Control	Sequence Number	Destination PAN Identifier	Destination Address	Source PAN Identifier	Source Address	Auxiliary Security Header	Frame Payload	FCS
		Addressing fields						
MHR							MAC Payload	MFR

Figure 17

2.11.2 Description

The following structure is applicable:

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0xnnnn	Variable length of radio telegram
	3	1	Optional Length	0x01	1 byte of Optional Data
	4	1	Packet Type	0x10	0x10: RADIO_802_15_4
-	5	1	CRC8H	0xnn	
Data	6	x	MAC Payload	... ...	802.15.4 MAC payload excluding FCS
Optional Data	6+x	1	RSSI	0xnn	Send: Set to 0xFF Receive: RSSI of received telegram
-	8+x	1	CRC8D	0xnn	

Table 131

For reception, there is no RESPONSE.
For transmission, one of the following RESPONSE messages is expected:

- 00: RET_OK
- 02: RET_NOT_SUPPORTED
- 03: RET_WRONG_PARAM
- 07: RET_NO_FREE_BUFFER

2.12 Packet Type 0x11: 2.4_GHZ_CONFIG

2.12.1 List of EnOcean 2.4_GHZ_CONFIG commands

Code	Command Name	Description
0x01	SET_802.15.4_CHANNEL	Set the IEEE 802.15.4 radio channel
0x02	GET_802.15.4_CHANNEL	Read the IEEE 802.15.4 radio channel

Table 132

2.12.2 Code 0x01: SET_802.15.4_CHANNEL

Function: Set the radio channel used by the device for the transmission of IEEE 802.15.4 radio telegrams. Channel 11 (0x0B) … Channel 26 (0x1A) are usually supported by IEEE 802.15.4 implementations in the 2.4 GHz band.

The command structure is shown below.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0002	2 bytes of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x11	0x11: 2.4_GHZ_CONFIG
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x01	0x01: SET_802.15.4_CHANNEL
	7	1	Channel	0x0B … 0x1A	IEEE 802.15.4 Radio Channel (0x0B: CH11 … 0x1A: CH26)
-	8	1	CRC8D	0xnn	

Table 133

One of the following RESPONSE messages is expected in response to this request:

- 00: RET_OK
- 02: RET_NOT_SUPPORTED

2.12.3 Code 0x02: GET_802.15.4_CHANNEL

Function: Determine the radio channel used by the device for the transmission of IEEE 802.15.4 radio telegrams.

The command structure is shown below.

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0001	1 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x11	0x11: 2.4_GHZ_CONFIG
-	5	1	CRC8H	0xnn	
Data	6	1	COMMAND Code	0x02	0x02: GET_802.15.4_CHANNEL
-	7	1	CRC8D	0xnn	

Table 134

One of the following RESPONSE messages is expected in response to this request:
- 00: RET_OK (using the syntax below)
- 02: RET_NOT_SUPPORTED

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	
Header	1	2	Data Length	0x0002	2 byte of Data
	3	1	Optional Length	0x00	No Optional Data
	4	1	Packet Type	0x02	0x02: RESPONSE
-	5	1	CRC8H	0xnn	
Data	6	1	RESPONSE Code	0x00	0x00: RET_OK
	7	1	Channel	0x0B … 0x1A	IEEE 802.15.4 Radio Channel (0x0B: CH11 … 0x1A: CH26)
-	8	1	CRC8D	0xnn	

Table 135

3 Appendix

3.1 ESP3 Data Flow Sequences

The following examples illustrate the ESP3 traffic. In particular, the flow of the Smart Ack commands is more complex.

3.1.1 Client Data Request

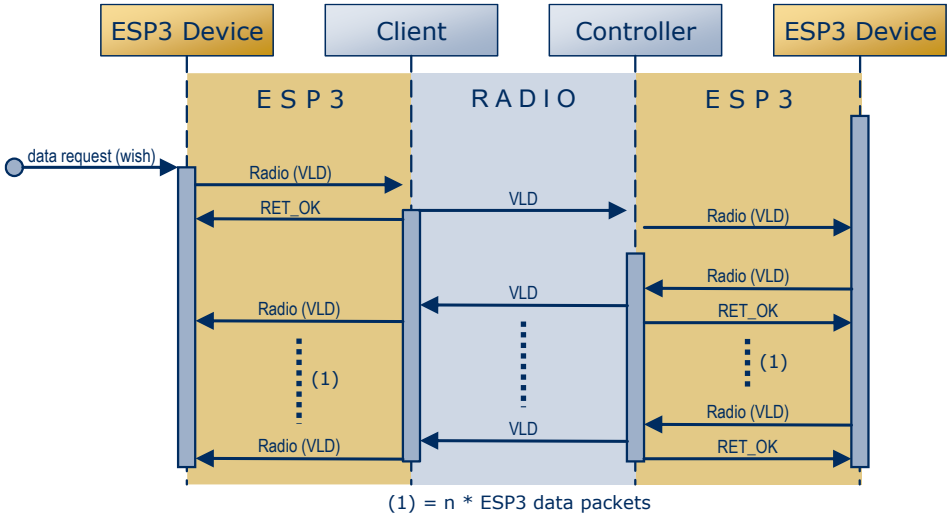

Figure 18

3.1.2 Teach IN via UTE

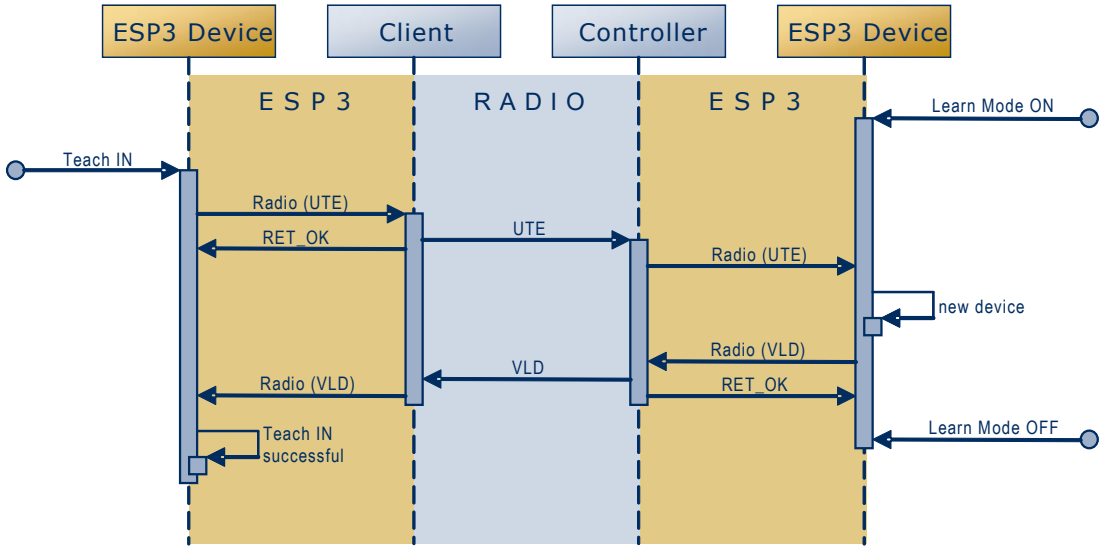

Figure 19

3.1.3 Teach IN via Smart Ack

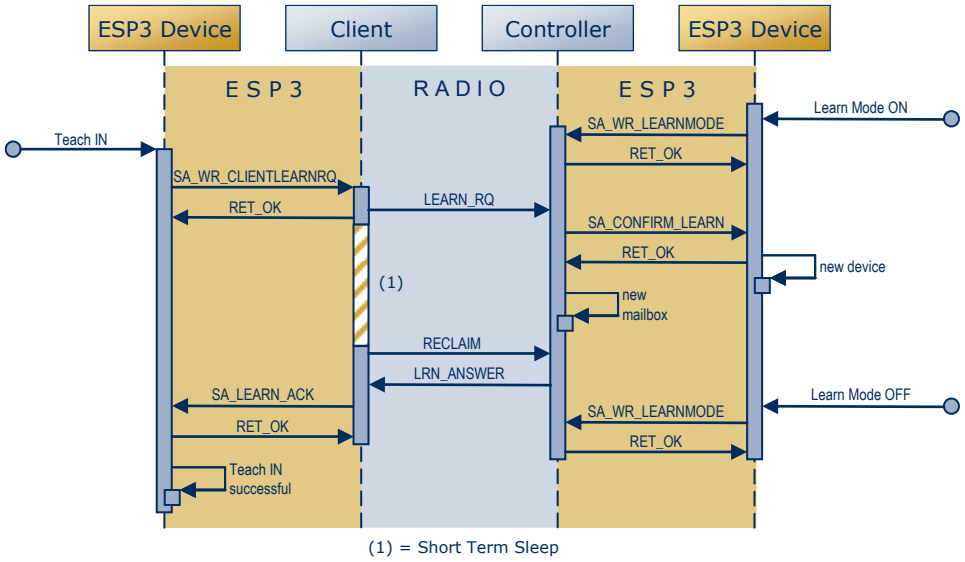

Figure 20

3.1.4 Teach IN via Smart Ack incl. repeater

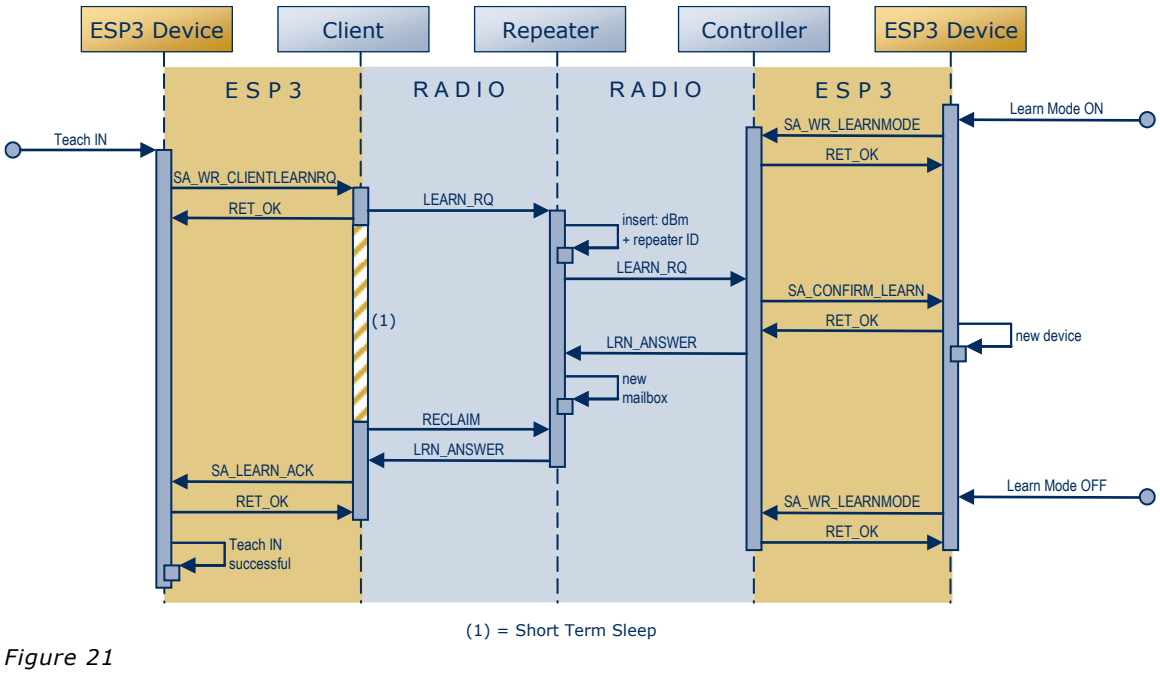

Figure 21

3.2 ESP3 Packet Examples

3.2.1 RADIO_ERP1 (VLD Telegram)

Sy	Header	CRC8	Data	Optional Data	CRC8
55	00 0F 07 01	2B	D2 DD DD DD DD DD DD DD DD DD 00 80 35 C4 00	03 FF FF FF FF 4D 00	36

3.2.2 CO_WR_SLEEP (10 ms)

Sy	Header	CRC8	Data	CRC8
55	00 05 00 05	DB	01 00 00 00 0A	54

Period = 10 (0x0A)

3.2.3 CO_WR_RESET

Sy	Header	CRC8	Data	CRC8
55	00 01 00 05	70	02	0E

3.2.4 CO_RD_IDBASE

Sy	Header	CRC8	Data	CRC8
55	00 01 00 05	70	08	38

Response RET_OK:

Sy	Header	CRC8	Data	CRC8
55	00 05 00 02	CE	00 FF 80 00 00	DA

3.2.5 REMOTE_MAN_COMMAND

Example dummy command:

```
Function       = 0x0876
Manufacture    = 0x07FF
Message data   = 0x000102030405060708090a0b0c0d0e0f
DestinationID = Broadcast = 0xFFFFFFFF
SendWithDelay = 0
```

Sy	Header	CRC8	Data	
				Message data
55	00 14 0A 07	9E	08 76 07 FF	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

Optional Data	CRC8
FF FF FF FF 00 00 00 00 FF 00	86

Example QueryID:

Sy	Header	CRC8	Data	CRC8
55	00 04 00 07	BE	00 04 07 FF	33

3.3 CRC8 calculation

The polynomial $\mathbf{G(x) = x^8 + x^2 + x^1 + x^0}$ is used to generate the CRC8 table which is needed for the CRC8 calculation. The following C code illustrates how the CRC8 value is calculated:

```
uint8 u8CRC8Table[256] = {
    0x00, 0x07, 0x0e, 0x09, 0x1c, 0x1b, 0x12, 0x15,
    0x38, 0x3f, 0x36, 0x31, 0x24, 0x23, 0x2a, 0x2d,
    0x70, 0x77, 0x7e, 0x79, 0x6c, 0x6b, 0x62, 0x65,
    0x48, 0x4f, 0x46, 0x41, 0x54, 0x53, 0x5a, 0x5d,
    0xe0, 0xe7, 0xee, 0xe9, 0xfc, 0xfb, 0xf2, 0xf5,
    0xd8, 0xdf, 0xd6, 0xd1, 0xc4, 0xc3, 0xca, 0xcd,
    0x90, 0x97, 0x9e, 0x99, 0x8c, 0x8b, 0x82, 0x85,
    0xa8, 0xaf, 0xa6, 0xa1, 0xb4, 0xb3, 0xba, 0xbd,
    0xc7, 0xc0, 0xc9, 0xce, 0xdb, 0xdc, 0xd5, 0xd2,
    0xff, 0xf8, 0xf1, 0xf6, 0xe3, 0xe4, 0xed, 0xea,
    0xb7, 0xb0, 0xb9, 0xbe, 0xab, 0xac, 0xa5, 0xa2,
    0x8f, 0x88, 0x81, 0x86, 0x93, 0x94, 0x9d, 0x9a,
    0x27, 0x20, 0x29, 0x2e, 0x3b, 0x3c, 0x35, 0x32,
    0x1f, 0x18, 0x11, 0x16, 0x03, 0x04, 0x0d, 0x0a,
    0x57, 0x50, 0x59, 0x5e, 0x4b, 0x4c, 0x45, 0x42,
    0x6f, 0x68, 0x61, 0x66, 0x73, 0x74, 0x7d, 0x7a,
    0x89, 0x8e, 0x87, 0x80, 0x95, 0x92, 0x9b, 0x9c,
    0xb1, 0xb6, 0xbf, 0xb8, 0xad, 0xaa, 0xa3, 0xa4,
    0xf9, 0xfe, 0xf7, 0xf0, 0xe5, 0xe2, 0xeb, 0xec,
    0xc1, 0xc6, 0xcf, 0xc8, 0xdd, 0xda, 0xd3, 0xd4,
    0x69, 0x6e, 0x67, 0x60, 0x75, 0x72, 0x7b, 0x7c,
    0x51, 0x56, 0x5f, 0x58, 0x4d, 0x4a, 0x43, 0x44,
    0x19, 0x1e, 0x17, 0x10, 0x05, 0x02, 0x0b, 0x0c,
    0x21, 0x26, 0x2f, 0x28, 0x3d, 0x3a, 0x33, 0x34,
    0x4e, 0x49, 0x40, 0x47, 0x52, 0x55, 0x5c, 0x5b,
    0x76, 0x71, 0x78, 0x7f, 0x6A, 0x6d, 0x64, 0x63,
    0x3e, 0x39, 0x30, 0x37, 0x22, 0x25, 0x2c, 0x2b,
    0x06, 0x01, 0x08, 0x0f, 0x1a, 0x1d, 0x14, 0x13,
    0xae, 0xa9, 0xa0, 0xa7, 0xb2, 0xb5, 0xbc, 0xbb,
    0x96, 0x91, 0x98, 0x9f, 0x8a, 0x8D, 0x84, 0x83,
    0xde, 0xd9, 0xd0, 0xd7, 0xc2, 0xc5, 0xcc, 0xcb,
    0xe6, 0xe1, 0xe8, 0xef, 0xfa, 0xfd, 0xf4, 0xf3
    };

#define proccrc8(u8CRC, u8Data) (u8CRC8Table[u8CRC ^ u8Data])

Example:
u8CRC = 0;
for (i = 0 ; i < u16DataSize ; i++)
   u8CRC = proccrc8(u8CRC, u8Data[i]);
printf("CRC8 = %02X\n", u8CRC);
```

3.4 UART Synchronization (example c-code)

This chapter provides an example for UART synchronization. Please note that this code below assumes big endian byte order.

3.4.1 ESP3 Packet Structure

```
  //! Packet structure (ESP3)
    typedef struct
    {
        // Amount of raw data bytes to be received. The most significant byte is sent/received
first
            uint16   u16DataLength;
        // Amount of optional data bytes to be received
            uint8     u8OptionLength;
        // Packe type code
            uint8     u8Type;
        // Data buffer: raw data + optional bytes
            uint8     *u8DataBuffer;
    } PACKET_SERIAL_TYPE;
```

3.4.2 Get ESP3 Packet

```
//! \file uart_getPacket.c

#include "EO3000I_API.h"
#include "proc.h"
#include "uart.h"
#include "time.h"

/*
ESP3 packet structure through the serial port.

Protocol bytes are generated and sent by the application

Sync = 0x55
CRC8H
CRC8D


    1              2                1           1           1      u16DataLen + u8OptionLen
1
+------+------------------+---------------+-----------+-----------+-------------/------------+----
-------+
| 0x55 |     u16DataLen   |   u8OptionLen |  u8Type   |  CRC8H    |             DATAS          |
CRC8D   |
+------+------------------+---------------+-----------+-----------+-------------/------------+----
-------+


DATAS structure:

                    u16DataLen                      u8OptionLen
+-----------------------------------------------+----------------------+
|                    Data                       |     Optional         |
+-----------------------------------------------+----------------------+
*/


RETURN_TYPE uart_getPacket(PACKET_SERIAL_TYPE *pPacket, uint16 u16BufferLength)
{
    //! uart_getPacket state machine states.
    typedef enum
    {
         //! Waiting for the synchronisation byte 0x55
         GET_SYNC_STATE=0,
         //! Copying the 4 after sync byte: raw data length (2 bytes), optional data length (1),
type (1).
         GET_HEADER_STATE,
         //! Checking the header CRC8 checksum. Resynchronisation test is also done here
         CHECK_CRC8H_STATE,
         //! Copying the data and optional data bytes to the paquet buffer
```

```
    GET_DATA_STATE,
    //! Checking the info CRC8 checksum.
    CHECK_CRC8D_STATE,

} STATES_GET_PACKET;

 //! UART received byte code
uint8  u8RxByte;
 //! Checksum calculation
static uint8 u8CRC = 0;
 //! Nr. of bytes received
static uint16 u16Count = 0;
 //! State machine counter
static STATES_GET_PACKET u8State = GET_SYNC_STATE;
//! Timeout measurement
static uint8 u8TickCount = 0;
 // Byte buffer pointing at the paquet address
uint8 *u8Raw = (uint8*)pPacket;
 // Temporal variable
uint8    i;

// Check for timeout between two bytes
if (((uint8)ug32SystemTimer) - u8TickCount > SER_INTERBYTE_TIME_OUT)
{
    // Reset state machine to init state
    u8State = GET_SYNC_STATE;
}

// State machine goes on when a new byte is received
while (uart_getByte(&u8RxByte) == OK)
{
    // Tick count of last received byte
    u8TickCount = (uint8)ug32SystemTimer;

    // State machine to load incoming packet bytes
    switch(u8State)
    {
        // Waiting for packet sync byte 0x55
        case GET_SYNC_STATE:

            if (u8RxByte == SER_SYNCH_CODE)
            {
                u8State = GET_HEADER_STATE;
                u16Count = 0;
                u8CRC   = 0;
            }

            break;


        // Read the header bytes
        case GET_HEADER_STATE:

            // Copy received data to buffer
            u8Raw[u16Count++] = u8RxByte;
            u8CRC = proc_crc8(u8CRC, u8RxByte);

            // All header bytes received?
            if(u16Count == SER_HEADER_NR_BYTES)
            {
                u8State = CHECK_CRC8H_STATE;
            }

            break;


        // Check header checksum & try to resynchonise if error happened
        case CHECK_CRC8H_STATE:

            // Header CRC correct?
            if (u8CRC != u8RxByte)
            {
                // No. Check if there is a sync byte (0x55) in the header
                int a = -1;
                for (i = 0 ; i < SER_HEADER_NR_BYTES ; i++)
                    if (u8Raw[i] == SER_SYNCH_CODE)
                    {
                        // indicates the next position to the sync byte found
                        a=i+1;
```

```
                    break;
                };

            if ((a == -1) && (u8RxByte != SER_SYNCH_CODE))
            {
                // Header and CRC8H does not contain the sync code
                u8State = GET_SYNC_STATE;
                break;
            }
            else if((a == -1) && (u8RxByte == SER_SYNCH_CODE))
            {
                // Header does not have sync code but CRC8H does.
                // The sync code could be the beginning of a packet
                u8State = GET_HEADER_STATE;
                u16Count = 0;
                u8CRC    = 0;
                break;
            }

            // Header has a sync byte. It could be a new telegram.
            // Shift all bytes from the 0x55 code in the buffer.
            // Recalculate CRC8 for those bytes
            u8CRC = 0;
            for (i = 0 ; i < (SER_HEADER_NR_BYTES - a) ; i++)
            {
                u8Raw[i] = u8Raw[a+i];
                u8CRC = proc_crc8(u8CRC, u8Raw[i]);
            }
            u16Count = SER_HEADER_NR_BYTES - a;
            // u16Count = i; // Seems also valid and more intuitive than u16Count -= a;

            // Copy the just received byte to buffer
            u8Raw[u16Count++] = u8RxByte;
            u8CRC = proc_crc8(u8CRC, u8RxByte);

            if(u16Count < SER_HEADER_NR_BYTES)
            {
                u8State = GET_HEADER_STATE;
                break;
            }

            break;
        }

        // CRC8H correct. Length fields values valid?
        if((pPacket->u16DataLength + pPacket->u8OptionLength) == 0)
        {
            //No. Sync byte received?
            if((u8RxByte == SER_SYNCH_CODE))
            {
                //yes
                u8State = GET_HEADER_STATE;
                u16Count = 0;
                u8CRC    = 0;
                break;
            }

            // Packet with correct CRC8H but wrong length fields.
            u8State = GET_SYNC_STATE;
            return OUT_OF_RANGE;
        }

        // Correct header CRC8. Go to the reception of data.
        u8State = GET_DATA_STATE;
        u16Count = 0;
        u8CRC    = 0;

        break;


    // Copy the information bytes
    case GET_DATA_STATE:

        // Copy byte in the packet buffer only if the received bytes have enough room
        if(u16Count < u16BufferLength)
        {
            pPacket->u8DataBuffer[u16Count] = u8RxByte;
            u8CRC = proc_crc8(u8CRC, u8RxByte);
        }
```

```
                // When all expected bytes received, go to calculate data checksum
                if( ++u16Count == (pPacket->u16DataLength + pPacket->u8OptionLength) )
                {
                    u8State = CHECK_CRC8D_STATE;
                }

                break;


            // Check the data CRC8
            case CHECK_CRC8D_STATE:

                // In all cases the state returns to the first state: waiting for next sync byte
                u8State = GET_SYNC_STATE;

                // Received packet bigger than space to allocate bytes?
                if (u16Count > u16BufferLength)  return OUT_OF_RANGE;

                // Enough space to allocate packet. Equals last byte the calculated CRC8?
                if (u8CRC == u8RxByte)        return OK;               // Correct packet
received

                // False CRC8.
                // If the received byte equals sync code, then it could be sync byte for next
            paquet.
                if((u8RxByte == SER_SYNCH_CODE))
                {
                    u8State = GET_HEADER_STATE;
                    u16Count = 0;
                    u8CRC   = 0;
                }

                return NOT_VALID_CHKSUM;


            default:

                // Yes. Go to the reception of info.
                u8State = GET_SYNC_STATE;
                break;
        }
    }

    return (u8State == GET_SYNC_STATE) ? NO_RX_TEL : NEW_RX_BYTE;
}
```

3.4.3 Send ESP3 Packet

```
//! \file uart_sendPacket.c


#include "EO3000I_API.h"
#include "proc.h"
#include "uart.h"

/*
ESP3 packet structure through the serial port.

Protocol bytes are generated and sent by the application

Sync = 0x55
CRC8H
CRC8D


    1              2                1           1           1      u16DataLen + u8OptionLen
1
+------+------------------+---------------+-----------+-----------+-------------/------------+----
-------+
| 0x55 |     u16DataLen   |   u8OptionLen |  u8Type   |    CRC8H  |             DATAS          |
CRC8D   |
+------+------------------+---------------+-----------+-----------+-------------/------------+----
-------+


DATAS structure:

                   u16DataLen                    u8OptionLen
+-----------------------------------------------+----------------------+
|                    Data                       |     Optional         |
+-----------------------------------------------+----------------------+

*/

RETURN_TYPE uart_sendPacket(PACKET_SERIAL_TYPE *pPacket)
{
    uint16 i;
    uint8  u8CRC;

    // When both length fields are 0, then this telegram is not allowed.
    if((pPacket->u16DataLength || pPacket->u8OptionLength) == 0)
    {
        return OUT_OF_RANGE;
    }
    // Sync
    while(uart_sendByte(0x55) != OK);

    // Header
    while(uart_sendBuffer((uint8*)pPacket, 4) != OK);

    // Header CRC
    u8CRC = 0;
    u8CRC = proc_crc8(u8CRC, ((uint8*)pPacket)[0]);
    u8CRC = proc_crc8(u8CRC, ((uint8*)pPacket)[1]);
    u8CRC = proc_crc8(u8CRC, ((uint8*)pPacket)[2]);
    u8CRC = proc_crc8(u8CRC, ((uint8*)pPacket)[3]);
    while(uart_sendByte(u8CRC) != OK);

    // Data
    u8CRC = 0;
    for (i = 0 ; i < (pPacket->u16DataLength + pPacket->u8OptionLength) ; i++)
    {
        u8CRC = proc_crc8(u8CRC, pPacket->u8DataBuffer[i]);
        while(uart_sendByte(pPacket->u8DataBuffer[i]) != OK);
    }

    // Data CRC
    while(uart_sendByte(u8CRC)!=OK);

    return OK;
}
```