Wireless M-Bus Documentation

Serial Reference Manual

Steinbeis Transfer Center
Embedded Design and Networking

March 13, 2015

Document History

1.7	2012-03-09	Appearance review	dj
1.8	2012-03-14	'Ping' and 'Power down on idle' command updated	dj
1.9	2012-03-30	AT Parser chapter updated	mr
2.0	2012-04-03	C Mode added and DLL chapter updated	mr & ar
2.1	2012-04-17	Review APL and DLL	mr
2.2	2012-05-25	DLL updated	mr
2.3	2012-06-11	Stack history updated	mr
2.4	2012-06-14	Renamed chapter AT parser to serial, serial documentation and stack history updated	mr
2.5	2012-06-22	Stack history removed.	dj
2.6	2012-06-26	Serial command 0x54 updated.	mr
2.7	2012-07-17	DLL and DLLPLUS updated.	mr
2.8	2012-10-16	SERIELL and APL updated.	mr
2.9	2013-07-12	Added SPI serial section.	md
3.0	2013-08-22	Chapter bidirectional communication updated Memory management updated Wireless M-Bus Modes table updated DLLPLUS Appendix Control field values updated DLL, DLLPLUS, APL and SERIAL chapters updated	mr
3.1	2013-10-22	Review APL	mr
3.2	2013-10-23	Review Serial	mr
3.3	2013-10-25	Review DllPlus	mr
3.4	2013-11-21	Added some new functions.	ar
3.5	2014-01-22	Removed not used function stzedn_apl_evt_reqresp().	ar
3.6	2014-01-30	Removed not used enum E_APL_RESP_STATUS_SUCCESSFUL. Adapted interface of function stzedn_apl_evt_newMeter()	ar
3.7	2014-02-03	Updated SPI documentation	md
3.8	2014-03-03	Updated documentation of the sleep modes. Updated documentation of the function stzedn_apl_meterAdd()	ar
3.9	2014-03-11	Added appendix for license terms.	md
3.10	2014-03-31	Adapted documentation of the flag parameter of the struct s_apl_tlgAttr_t. Added description for two new functions and one new enum: stzedn_apl_setContentOfMessage(), stzedn_apl_getContentOfMessage(), E_APL_COM_t	ar
3.11	2014-05-09	Added serial commands 0x5D,0x44 and 0x45. Removed deprecated chapter "Send status request"	ar
3.12	2014-05-09	Added serial command 0x5F	na
3.13	2014-08-12	Removed deprecated serial confirmation codes	md
3.14	2014-09-12	Added serial command 0x90	dj
3.15	2014-09-24	Changed serial command 0x90 to serial command 0x3B	dj
3.16	2015-02-19	Added section for modulated carrier transmission	md

Table 1: Document History

1 Summary

In many regions of the world M-Bus is recognized as a basis for new advanced metering infrastructure (AMI) installations. Their wireless implementation brings a competitive advantage compared to existing wired solutions. The products are easy to install and to maintain. The Wireless M-Bus standard (EN 13757-4:2005) defines the wireless communication between meters for water, gas, heat and electricity, and the data concentrators. The stzedn stack implements the protocols for Wireless M-Bus. It has demonstrated its interoperability with modules of well-known manufacturers on the different protocol layers and is compliant with the current Wireless M-Bus standard [1]. It is an optimised solution between small footprint and excellent modularity and scalability.
This document describes a demonstration software based on the stzedn protocol stack for Wireless M-Bus.

Contents

List of Tables

List of Figures

2 Wireless M-Bus Basics

2.1 Introduction

The Metering Bus (or in short "M-Bus ") is a field bus specialized for transmission of metering data from gas, electricity, heat, water or other meters to a data collector. It is described by European Norm (EN 13757), which includes the specification of wired and Wireless M-Bus. The specification is divided into five parts:

- EN 13757-1 ([2]):
  Communication systems for meters and remote reading of meters - Part 1: Data exchange
  The first part describes the basic communication between the meters and a central data collector. It provides an overview of the communication system.
- EN 13757-2 ([3]):
  Communication systems for meters and remote reading of meters - Part 2: Physical and link layer
  The second part includes the specification of the physical data transmission using wired connections. It also includes the description of the protocol to transmit the data.
- EN 13757-3 ([4]):
  Communication systems for meters and remote reading of meters - Part 3: Dedicated application layer
  The third part describes a standardized application protocol to enable multivendor capability. So devices of different manufacturers may be combined in one system.
- EN 13757-4 ([1]):
  Communication systems for meters and remote reading of meters - Part 4: Wireless meter readout (Radio meter reading for operation in the 868 MHz to 870 MHz SRD band)
  This part specifies the wireless communication of M-Bus and is the main source document for this implementation. It includes the Physical and the Data Link Layer for wireless devices. It corresponds to specification EN 13757-2 for wired communication.
- EN 13757-5 ([5]):
  Communication systems for meters and remote reading of meters - Part 5: Relaying
  This last part includes different proposals for relaying data frames to overcome the range problem between remote meters and data collectors.

The sixth part of this norm relates only to wired busses and is disregarded in this document.
All parts of EN 13757 are compliant to the European Norm EN 870-5 [6].

2.2 Protocol Stack

2.2.1 Introduction

M-Bus is compatible to the international ISO/OSI-model, but only the layers 1, 2 and 7 are implemented.

Layer 7	Application Layer (EN 13757-3)
Layer 2	Data Link Layer (EN 13757-2 or EN 13757-4)
Layer 1	Physical Layer (EN 13757-2 or EN 13757-4)

Table 2: Operating Modes of Wireless M-Bus

Up to now, the application layer implements all other protocol layers required for a specific appliance. Especially if routing is required according to [5], it is implemented in the application layer. The reduced modularity leads to compact implementations running on very small devices with minimum computing resources. But the lack of modularity certainly is one of the reasons why standardized routing protocols are currently not available for Wireless M-Bus. The M-Bus favours asymmetric network topologies with low-cost or low-power metering devices on the one side and data collectors or gateways with higher performance on the other side. Currently, only point-to-point or star network topologies can be supported. Mesh or multi-hop topologies are not possible. This chapter describes the settings for implementations of the data collector and the metering devices to setup a network in star topology.

2.2.2 Communication Modes

Depending on the application there are various combinations of communication modes for data collectors and metering devices. These settings define the communication flow and the configuration of the radio channel. Table 3 lists available communication modes.

Mode	Communication	Description
S1	Unidirectional	In the stationary mode, the metering devices send their data several times a day. In this mode, the data collector may save power as the metering devices send a wakeup signal before transmitting their data.
S2	Bidirectional	Bidirectional version of S1.
T1	Unidirectional	In the Frequent Transmit mode, the metering devices periodically send their data to collectors in range. The interval is configurable in terms of several seconds or minutes.
T2	Bidirectional	Bidirectional version of T1. The data collector may request dedicated data from the metering devices.
C1	Unidirectional	Compact mode. This mode is similar to mode T but it allows for transmission of more data within the same energy budget and with the same duty cycle. It is suitable for walk-by and/or drive-by readout. The common reception of mode T and mode C frames with a single receiver is possible.
C2	Bidirectional	Bidirectional version of C1. The data collector may request dedicated data from the metering devices.

Table 3: Operating Modes of Wireless M-Bus

Table 4 lists the combinations of the communication modes of meters and data collectors.

In T2 mode, the communication settings for the two data directions are different. Due to different requirements, both communication devices have to support fast switching of communication speed, coding scheme, and frequency.

	Meter Device's channels			
Data Collector channels	**S1, S2**	**T1, T2**	**C1, C2**	**Collector Device's Tx radio settings (Freq., data rate, modulation)**
S1, S2	X			868,30 MHz 32.768 kcps Manchester enc.
T1, T2		X		868,30 MHz 32.768 kcps Manchester enc.
C1, C2		X		868,30 MHz 32.768 kcps Manchester enc.
C1, C2			X	869,525 MHz 50 kcps NRZ enc.
Metering Device's Tx radio settings (Freq., data rate, modulation)	868,30 MHz 32.768 kcps Manchester encoding	868,95 MHz 100 kcps 3-out-of-6 encoding	868,95 MHz 100 kcps NRZ encoding	

Table 4: Compatibility matrix for Wireless M-Bus operating modes

Table 4 shows that although the T- and S-modes cannot be combined, a T2 data collector may not well coexist with a parallel S-mode communication in its vicinity. Table 4 also shows that the C-mode supports the common reception of mode T and mode C frames with a single receiver.

2.2.3 Unidirectional vs. Bidirectional Communication

If unidirectional communication is used, data will be sent from the metering device to the data collector only. This enables simple transmitters as metering devices while the data collector only needs to receive. Because listen-before-talk (LBT) and dynamic network configuration are not possible, it is recommended to use the unidirectional modes for small and simple constellations with little network load.

In case of bidirectional communication, the collector device can request data from the meter device. This is for example the case in S2, T2, C2 and N2 mode. In this modes, a bidirectional communication will only be established if further data or commands need to be exchanged (cf. Fig. 1). Figure 1 shows a typical communication flow.

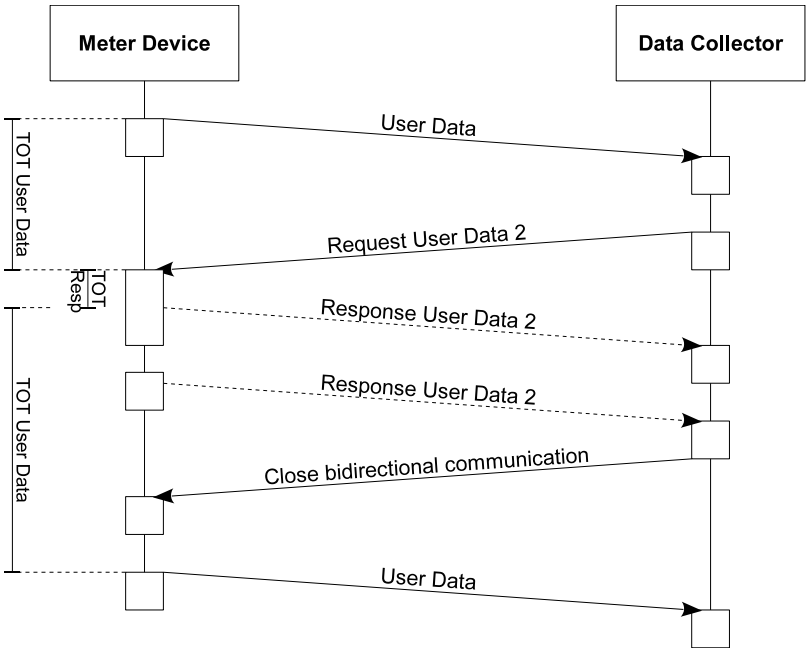

Figure 1: Bidirectional Communication with a request after the reception of a periodical meter packet. Example for mode S2. (Supported by the following modes: T2, S2, C2 and N2)

3 The stzedn protocol stack

3.1 Wireless M-Bus Protocol Stack

The Wireless M-Bus Stack from STZEDN is a fully Wireless M-Bus compliant protocol stack, including OMS v.3.0.1 [7]. It implements the functionality described in EN-13757 ([2], [3], [4], [1], [5]). Refer to figure 2 for a graphical overview.

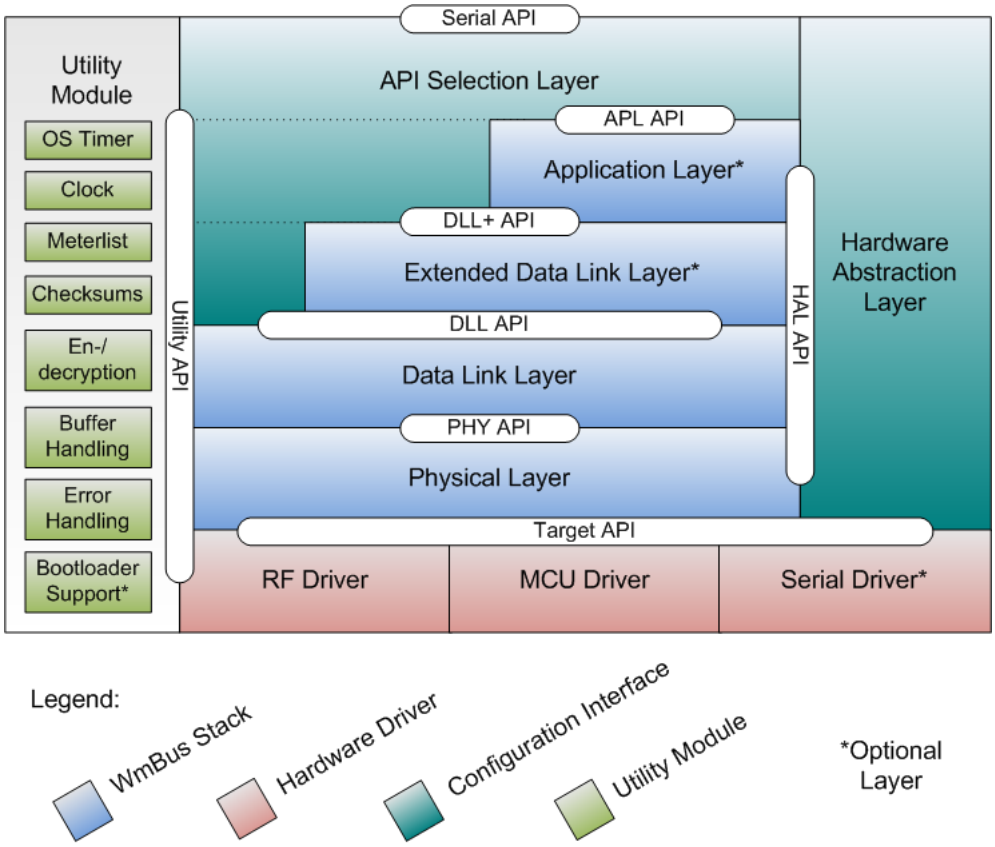

Figure 2: Wireless M-Bus Protocol Stack

The Wireless M-Bus protocol stack from STZEDN basically consists of four different parts:

- Wireless M-Bus Stack:
  The Wireless M-Bus Stack part of the protocol stack contains all Wireless M-Bus functionality.
- Hardware drivers:
  This part of the protocol stack contains all low level drivers to control the microcontroller unit, such as communication interfaces, memory resources and pheripheral units.
- Utility Module:
  This part of the protocol stack contains mechanisms and functions which the stack uses to forward information through the protocol layers, to report runtime information, to handle time intervals and clocks, to manage meterlists and to execute software updates.

- Configuration interfaces:

  The configuration interfaces are used to setup basic parameters of the target, such as location of the communication settings or input interrupt sources.

STZEDN offers different packages of the Wireless M-Bus protocol stack. All packages support full Wireless M-Bus functionality. Refer to table 5 for an overview. To find the most suitable Wireless M-Bus version for the desired application, please contact STZEDN (refer to chapter 5).

Layer	Basic	Extended	Extended serial	Full
Application Layer		x	x	x
Extended Data Link Layer	x	x	x	x
Data Link Layer	x	x	x	x
Physical Layer	x	x	x	x
RF Driver	x	x	x	x
MCU Driver	x	x	x	x
Serial Driver			x	x
Bootloader support				x

Table 5: Available Wireless M-Bus Protocol Stack Packages

Table 6 lists the different memory models of the stack. The memory model specifies where the Wireless M-Bus Stack saves internal data and meterlists.

Package	Volatile memory	Non-volatile memory
Basic	x	x
Extended	x	x
Extended serial	x	x
Full	x	x

Table 6: Available Wireless M-Bus memory models

Please contact STZEDN for information about currently supported microcontrollers and transceivers. Adaptions of the Wireless M-Bus protocol stack are possible upon request as well as driver developments for microcontrollers and transceivers. Please refer to chapter 5 for contact information.

3.2 Wireless M-Bus Protocol Stack Deliveries

3.2.1 Demonstration Applications

Demonstration applications for development boards were delivered as binary files. Please note, that the demonstration application contains the full Wireless M-Bus Stack from STZEDN. The Wireless M-Bus Suite can be used to study, test and evaluate the wmbus demonstration application and Wireless M-Bus, e.g. bidirectional communication between a Wireless M-Bus collector and a Wireless M-Bus meter device. Furthermore, Wireless M-Bus Suite can download the binary files to development boards to switch between Wireless M-Bus device types and Wireless M-Bus modes. Please contact STZEDN for information about supported development boards. The Wireless M-Bus Suite also

supports Wireless M-Bus sniffer applications for all Wireless M-Bus channels, including decryption and encryption of data.

3.2.2 Wireless M-Bus Stack as Library

STZEDN delivers the Wireless M-Bus Stack for the Wireless M-Bus devices types and Wireless M-Bus modes in libraries. Refer to table 7 for an overview. Please contact STZEDN for information about supported integrated development environments. Refer to chapter 5 for contact information. The customer is able to develop application using the Wireless M-Bus Stack libraries. For a fast and easy development, the Wireless M-Bus Stack comes with application code example for a collector and a meter device.

Mode	Meter Device	Collector Device
S1	x	
S2	x	x
T1	x	
T2	x	x
C1	x	
C2	x	x
N1	x	
N2	x	x

Table 7: Available Types and Modes

Please note, that Wireless M-Bus sniffer configrations are only delivered as binaries.

3.3 Configuration of the Wireless M-Bus Stack

STZEDN delivers the Wireless M-Bus protocol stack in fully configured libraries for different device types and modes (refer to table 7). Depending on the target hardware, the Wireless M-Bus Stack supports several communication interfaces, to enable customized hardware designs.

Target configuration file: *stzedn_target_config.h*

The target configuration file contains definitions about pin and port definitions of the communication interfaces. This includes e.g. the SPI and UART communication, as well as GPIO input pin definitions. Please note, that the number of configuration parameters for the hardware drivers depends on the target hardware. Refer to the hardware specific configuration guide.

3.4 Memory Models

3.4.1 Internal data

Table 8 lists the internal data of the Wireless M-Bus Stack, which is effected by the choice of the memory model.

Description	Meter	Collector
Device status	x	x
RF channel	x	x
Frequency offset	x	x
Serial baud rate	x	x
Meter Wireless M-Bus address	x	x
Collector Wireless M-Bus address	x	
Periodical transmission interval	x	
Device specific encryption key	x	x
General encryption key	x	x
Wireless M-Bus standard specific data	x	x
Meterlist		x

Table 8: Internal data of a Wireless M-Bus meter device.

3.4.2 Volatile memory model

The Wireless M-Bus Stack saves all internal variables as well as the meterlist in volatile memory. More precisely, in the random access memory (RAM). Thus, all data which changes at runtime is lost after a reset of the device. Refer to table 8 for a list of parameters which are effected.

3.4.3 Non-volatile memory model

3.4.3.1 Data storage

The non-volatile memory model stores the data listed in table 8 in the non-volatile memory. The Wireless M-Bus Stack does not specify which kind of non-volatile memory must be used. E.g. FLASH, EEPROM or FRAM.

In general, the Wireless M-Bus Stack buffers most parameters in volatile memory. It synchronizes the volatile memory with the non-volatile memory at specific points in time. A complete synchronization of the data in table 8 takes place at the following actions/events:

- During initialization of the Wireless M-Bus Stack.
- A forced synchronization, initiated by user application.
- When the Wireless M-Bus Stack goes to sleep mode, initiated by user application.

However, some data cannot be buffered in volatile memory. Thus, some API function calls cause a write access to non-volatile memory immediately - API functions which cause changes of meterlist entries. Thus, this effects only Wireless M-Bus collector devices. Furthermore, these functions do not initiate a complete synchronization, but apply only the changes to the non-volatile memory which is described by the API function. Refer to table 9 for a corresponding list of API functions.

API
stzedn_apl_meterAdd
stzedn_apl_meterRemove
stzedn_apl_meterSetRfAdapter
stzedn_apl_meterSetKey

Table 9: API functions, which cause a non-volatile write access immediately.

Another direct non-volatile memory write access takes place, when a Wireless M-Bus collector device calls the event `stzedn_apl_evt_newMeter`, if the return value is set to `TRUE`. In this case, a new meter which is currently not in the meterlist will be added to the meterlist.

3.4.3.2 Non-volatile memory management

Figure 3 shows the basic workflow of the Wireless M-Bus Stack regarding the memory management. After the start-up of the device and the initialization of the Wireless M-Bus Stack, the function `stzedn_apl_startMeter` or `stzedn_apl_startCollector` function must be called with appropriate parameters. The Wireless M-Bus Stack then synchronizes with the non-volatile memory.

Synchronization at `stzedn_apl_startMeter` or `stzedn_apl_startCollector` : In this function, the Wireless M-Bus Stack first checks, if the non-volatile memory contains valid information. If the non-volatile memory contains no valid information, the Wireless M-Bus Stack uses the parameters from the `stzedn_apl_startMeter` or `stzedn_apl_startCollector` function to initialize the device and stores the parameters afterwards into the non-volatile memory. If the information is valid, the parameters in the start function `stzedn_apl_startMeter` or `stzedn_apl_startCollector` were omitted and all parameters were loaded from the non-volatile memory. In this case, the Wireless M-Bus Stack also omits the optional static meterlist and loads the one in the non-volatile memory, even if the list is empty (only for collector). After calling this function, the Wireless M-Bus Stack is ready for operation.

Synchronization at `stzedn_apl_sleep` : When the Wireless M-Bus Stack is set into sleep mode, all internal parameters from table 8 were stored into non-volatile memory. If the function returns success, the user application is allowed to set the microcontroller into a hardware sleep mode.

Synchronization at `stzedn_apl_wake` : This function must be called to run the Wireless M-Bus Stack after a sleep mode. The stack synchronizes the buffered parameters in the volatile memory with the parameters in the non-volatile memory.

Synchronization at meterlist write/change operations: When the user changes the meter entries (add, change, remove), the Wireless M-Bus Stack writes the changes immediately into the non-volatile memory. The reason for this is that the meterlist entries were not buffered in the volatile memory.

Synchronization at meter installation mode: When a collector detects a new meter, it is possible to add the meter into the meterlist of the collector. The Wireless M-Bus Stack fires the event `stzedn_apl_evt_newMeter` and

the user application can decide, if the collector should add the meter into the own meterlist. In case the collector adds the meter into the meterlist, the stack writes the meter parameters into the non-volatile memory immediately.

In all other cases, there is no non-volatile memory write access.

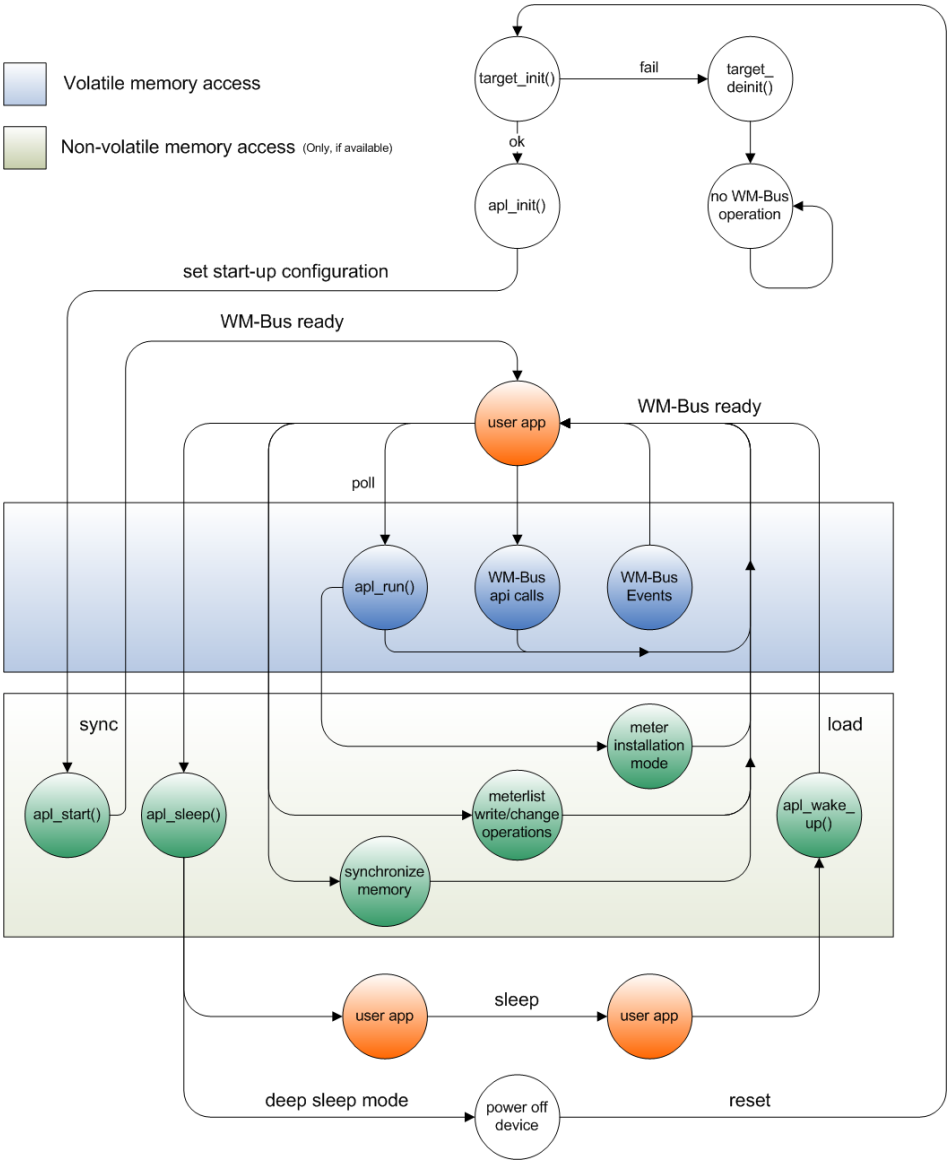

Figure 3: Non-volatile memory work flow.

3.5 Encryption Modes

Refer to table 10 for the supported encryption modes of the Wireless M-Bus protocol stack. The only mandatory encryption mode defined in OMS v.3.0.1 is mode 5.

Mode	Description
0	No encryption.
1	AES Counter Mode (AES-CTR).
5	AES Cipher Block Chaining Mode (AES-CBC) with dynamic initialization vector.

Table 10: Encryption Modes

3.6 Sleep modes

The stack provides several sleep modes, which can be used to save energy, while the stack is not busy.

- **Sleep**

  It is assumed that the mcu will enter a lpm state in which the RAM content will be kept. The radio will be set to power down mode.

- **Deep sleep**

  It is assumed that the mcu will enter a lpm state in which the RAM content will not be kept. The radio will be set to power down mode.

3.7 Wireless M-Bus Channels

The modes of Wireless M-Bus are using different channels. Table 11 lists all available channels. Depending on the device's mode, only a subset of this table can be accessed. Using channels that are not supported result in an error message at compilation of the firmware.

Mode	Frequency
N1a, N2a (Meter Device and Data Collector)	169.406 MHz
N1b, N2b (Meter Device and Data Collector)	169.418 MHz
N1c, N2c (Meter Device and Data Collector)	169.431 MHz
N1d, N2d (Meter Device and Data Collector)	169.443 MHz
N1e, N2e (Meter Device and Data Collector)	169.456 MHz
N1f, N2f (Meter Device and Data Collector)	169.468 MHz
N2g (Meter Device and Data Collector)	169.437 MHz
T2 (Data Collector TX, Meter Device RX)	868.30 MHz
T1, T2 (Meter Device TX, Data Collector RX)	868.95 MHz
S1, S2 (Meter Device and Data Collector)	868.30 MHz
C2 (Data Collector TX, Meter Device RX)	869.525 MHz
C1, C2 (Meter Device TX, Data Collector RX)	868.95 MHz

Table 11: Available channels

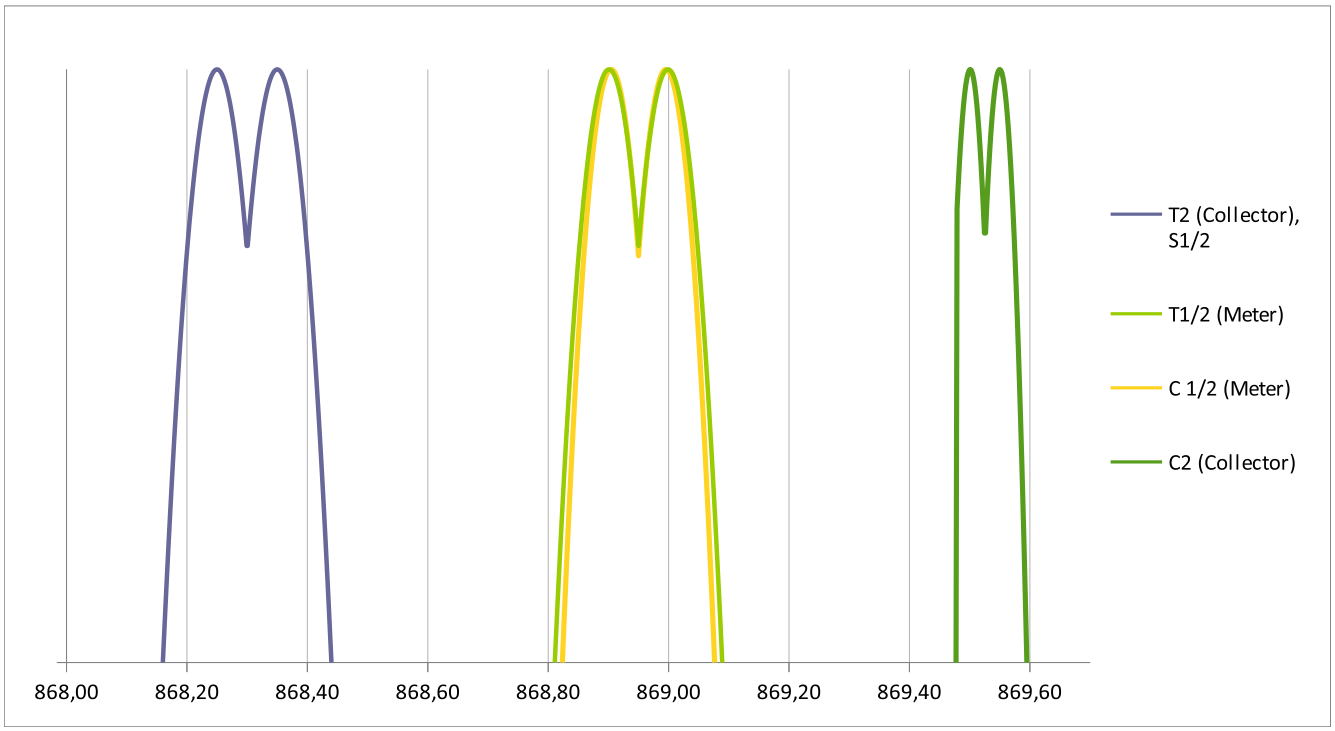

Figure 4: List of frequencies used by Wireless M-Bus mode S, T and C

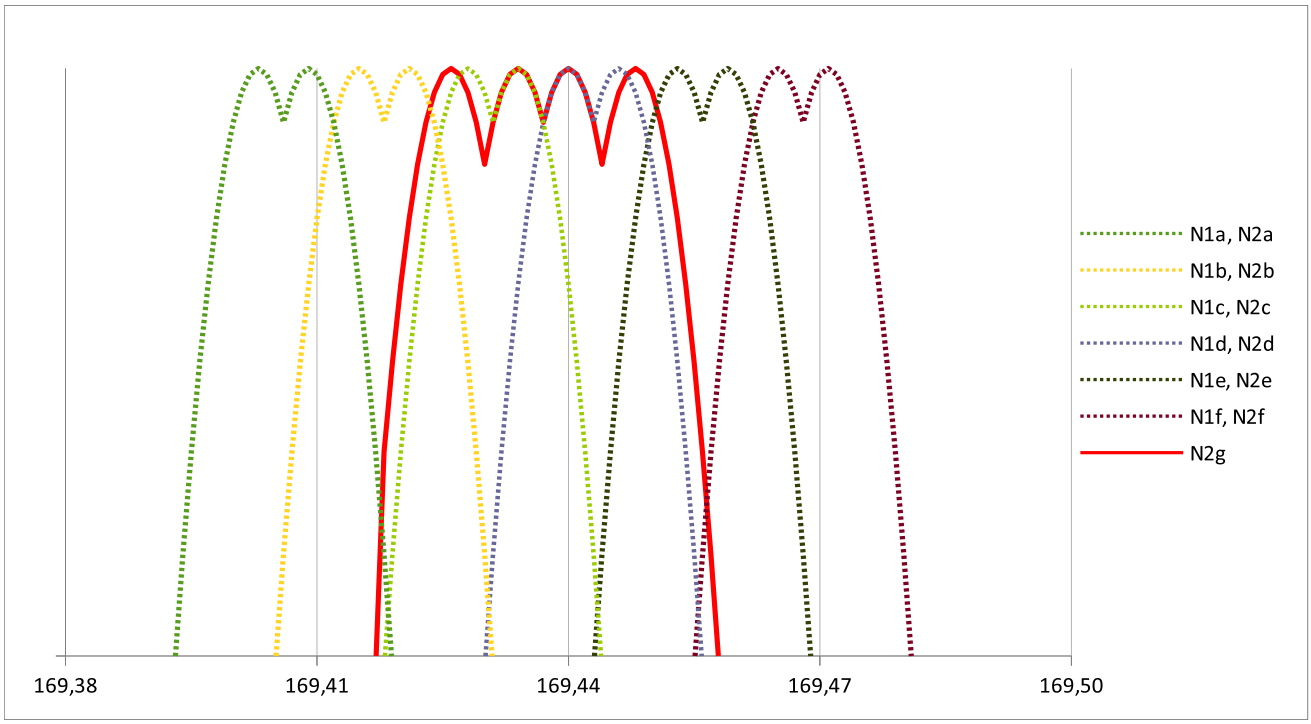

Figure 5: List of frequencies used by Wireless M-Bus mode N

4 Serial Command Set

4.1 Introduction

The serial command set is intended to provide access to the functions of the data link layer and the application layer of Wireless M-Bus via a serial interface. The purpose of this interface is to connect a PC or a host controller, and to use the module like a modem, without local access of the module's firmware. Multiple commands of the serial interface may be combined for an easy start communicating over Wireless M-Bus. The serial interface provides access to every single function of data link layer or application layer, and thus optimizes the firmware access. This version is intended to connect a host controller, and it eases the migration from using a module for communicating to the implementation of the protocol stack in the former host controller.

4.2 Frame format and command flow

4.2.1 General Command Flow

Generally, a serial communication is always started by the host controller (e.g. sending a request). However events can be called by the radio module. Furthermore all commands transmitted by the host controller should be replied by the radio module, otherwise this could indicate a problem.

4.2.2 Binary Command Format

A binary frame is splitted into a header, a data part and a footer.
The header starts with a start frame delimiter (SFD) which has always a value of A5 followed by 2 bytes length (MSB first) which sums all bytes following the length field. The type field describes the data following to the header.
The checksum is created according to the Wireless M-Bus block checksums described in [1]. For checksum calculation all bytes without the sync byte and the two length bytes are used.

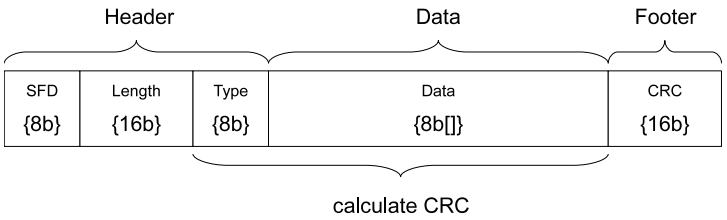

Figure 6: General frame format

After sending a serial telegram a response has to be received always within an appropriate time period. This time period depends on different external influences (e.g. mcu in use, flash write access).
If no response is received the command has to be resent by the application.

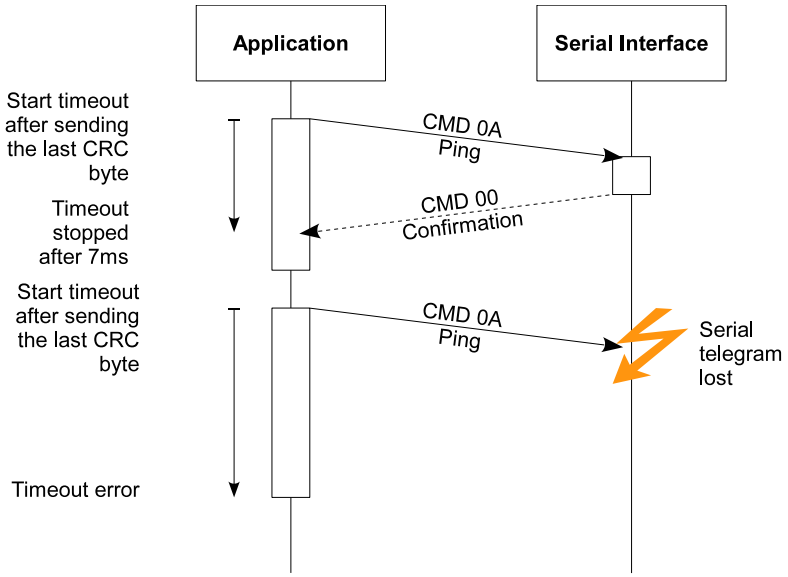

Figure 7: Timeout at sending a serial telegram

4.2.3 Protocol Layer Access

The protocol stack is available in data link layer (DLL) software packages and in application layer software packages. The packages based on the data link layer are intended for hosts running a custom application layer, e.g. from a wired M-Bus implementation. In the application layer packages, the DLL events and API functions are controlled and maintained by the application layer. Unintentional mixing of the two layers' access would cause data inconsistency, and is therefore not supported.

4.2.4 Static Settings

Some configuration options are statically defined and can only be changed using the commissioning tool. This enables the device to load a default configuration after device reset or power cycling. Some of these settings can be overwritten using the serial interface, but others are write-protected to protect the module from failure or damage on host failure. Two important settings are listed here:

- The interface is predefined in the hardware settings. Later changes are not supported.
- The communication settings are controlled using the configuration tool. Changing the settings using the serial interface itself would reduce the reliability of the communication. Therefore the following parameters are protected:
  - Baud rate of the serial interface (default value: 115.2 kbps)
  - Flow control of the serial interface (default value: none)
  - Data format of the serial interface (default value: 8N1: 8 data bits, no parity, 1 stop bit)

4.3 General commands

4.3.1 List of commands

Table 12 shows an overview over all available general commands. Please note, that in some configuration of the Wireless M-Bus Stack , not all general commands are available.

Command	MD	DC	Description	Chapters
00	X	X	Confirmation	4.3.2
01	X	X	RF_DATA	4.3.3
02	X	X	LOCAL_DATA	4.3.4
05	X	X	Set configuration	4.3.5
06	X	X	Get configuration	4.3.6
0A	X	X	Ping	4.3.8
0B	X	X	Power down on idle	4.3.9
20	X	X	Status	4.3.10
21	X	X	Memory synchronization	4.3.11
30...5F	X	X	Wireless M-Bus application layer commands	4.4
60...AF	X	X	Reserved for future commands	
FA	X	X	Clean up lost telegrams	4.3.12
FB	X	X	Manufacturer specific protocol	4.3.13
FC	X	X	Internal module event	4.3.14
FF	X	X	Reset	4.3.15

Table 12: General commands

4.3.2 Confirmation

Command:

```
<LENGTH>00<CODE><REQ_TYPE>[<DATA>]
```

Description:

Each telegram has to be responded by a confirmation or LOCAL_DATA (see chapter 4.3.4). A list of confirmation codes <CODE> (1 byte) is shown in table 13. The field <REQ_TYPE> (1 byte) includes the type of the request telegram.

It is possible to add several data bytes <DATA>. The following chapters describe the data blocks of the confirmation telegrams depending on the request telegrams.

Code	Mode
00	OK
01	Application is busy
02	Action failed
03	No data
13	Unknown command
14	Too few parameter bytes
15	Serial input buffer overflow
16	Command checksum invalid
20	Configuration type not supported
21	Mode not supported
22	Channel not supported
23	Device not supported
24	Encryption not supported
25	Non-volatile memory not ready
26	Indicates that the meter already exitsts

Table 13: Confirmation codes

4.3.3 RF_DATA

Command:

```
<LENGTH>01<RSSI><TTI>[<DATA>]
```

Description:

This telegram is used to transmit raw data.

- Received Signal Strength Indication <RSSI>: The RSSI value indicates the quality of link. It handles values from -137.5dBm (FE) to -10.5dBm (00). If the RSSI value is set to FF, no signal strength is available.
- Telegram Type Indication <TTI>: This value depends on the mode and the frame type of the received telegram.
  Example: TTI Byte

Bit 7 - Bit 6	Bit 5 - Bit 3	Bit 2 - Bit 0
reserved	Mode	Frame type

Table 14: Telegram Type Indication Byte

A list of frame types and modes are shown in table 15 and table 16.

- Raw data <DATA>: The raw data includes a complete received telegram.

For further information read the manual of your transceiver.

Example:

Telegram with raw data 00 01 02 03 04 05 06 07 08 09

SFD	Length	Type	RSSI	TTI	Raw data	CRC
A5	00 0D	01	A0	00	00 01 02 03 04 05 06 07 08 09	EC 62

Code	Frame type
0x00	A
0x01	B
0x07	Unknown

Table 15: Frame type Bits 0-2 (TTI Byte)

Code	Mode
0x00	S
0x01	T
0x02	C
0x03	N
0x07	Unknown

Table 16: Mode Bits (TTI Byte)

4.3.4 LOCAL_DATA

Command:

```
<LENGTH>02<REQ_TYPE>[<DATA>]
```

Description:

This telegram type is used to transmit response data. After the telegram type (02) the type of the request telegram follows <REQ_TYPE> (1 byte). After this data bytes <DATA> may be added to the telegram.

4.3.5 Set configuration

Command:

```
<LENGTH>05<CONFIG>[<PARAM>]
```

Description:

Sets the configuration of the device. After the type field (05) the configuration code follows. Table 24 shows the list of configuration codes <CONFIG> (1 byte). After the configuration code the parameters <PARAM> have to be sent. Please make sure that on every configuration command a constant power supply is ensured.

Response:

- Confirmation: OK (0x0, 4.3.2)

- Confirmation: Configuration code not supported (0x20, 4.3.2)
- Confirmation: Too few bytes if parameters are missing (0x14, 4.3.2)
- Confirmation: Mode not supported (0x21, 4.3.2)
- Confirmation: Channel not supported (0x22, 4.3.2)
- Confirmation: Device not supported (0x23, 4.3.2)

Example:

Set the primary device address to 4E9A 00000001 01 02.

SFD	Length	Type	Config	Data (Address)	CRC
A5	00 0A	05	15	4E 9A 00 00 00 01 01 02	20 D9

SFD	Length	Type	Confirm	Tx Type	Config	CRC
A5	00 04	00	00	05	15	C6 E6

4.3.6 Get configuration

Command:

<LENGTH>06<CONFIG>

Description:

Reads the configuration of the device. After the type field (06) the configuration code <CONFIG> (1 byte) follows. Table 24 shows the list of configuration codes. The response of the device is received as LOCAL_DATA (cf. chapter 4.3.4).

Response:

- Local data with requested data content (4.3.4)
- Confirmation: Configuration code not supported (0x20, 4.3.2)

Example:

Reading the primary device address (Result: 4E9A 00000001 01 02)

SFD	Length	Type	Config	CRC
A5	00 02	06	15	E6 EB

SFD	Length	Type	Tx Type	Config	Data (Address)	CRC
A5	00 0B	02	06	15	4E 9A 00 00 00 01 01 02	B4 9A

4.3.7 Get version

A special case of of the command 06 is reading the firmware version of a RF module.

To read the firmware version the configuration command FE has to be used.

The module responses a LOCAL_DATA telegram within at least two version bytes. These bytes have following format:

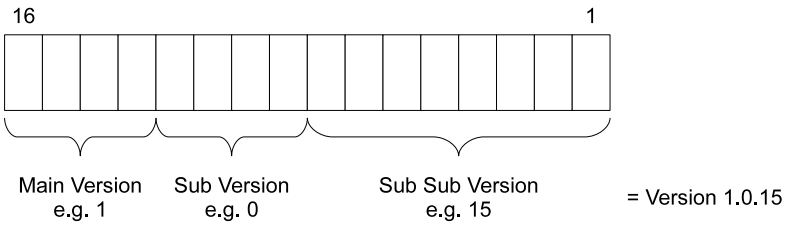

Figure 8: Format of version string

4.3.8 PING

Command:

<LENGTH>0A

Description:

This telegram type is used to check if a device is available. As response always a confirmation telegram is sent.

Response:

- Confirmation: OK (0x0, 4.3.2)

Example:

Sending a ping command.

SFD	Length	Type	CRC
A5	00 01	0A	53 78

SFD	Length	Type	Confirm	Tx Type	CRC
A5	00 03	00	00	0A	53 78

4.3.9 Power down on idle

Command:

<LENGTH>0B<ENABLE>

Description:

If <ENABLE> is set 1 the stack goes into power down mode if it is idle. To disable this mode set <ENABLE> to 0.

This function is only provided for data collectors and S1 or S2 meter devices.

Be careful: If the device is configured as data collector and power down on idle is enabled no further data telegrams are received.

Response:

- Confirmation: OK (0x0, 4.3.2)
- Confirmation: Failed if the mode could not be set (0x2, 4.3.2)
- Confirmation: Too few bytes if parameter <ENABLE> is missing (0x14, 4.3.2)

Example:

Sets the power down on idle mode.

SFD	Length	Type	Enable	CRC
A5	00 02	0B	00	CB 09

SFD	Length	Type	Confirm	Tx Type	CRC
A5	00 03	00	00	0B	6E 1D

4.3.10 Get status

Command:

`<LENGTH>20`

Description:

This function is provided for data collectors amd meter devices. This function returns the current status of the device.

The status byte includes following flags:

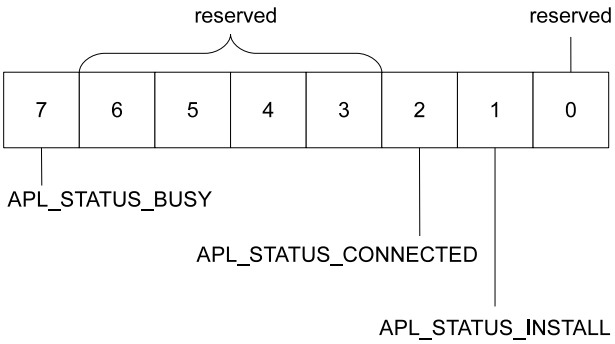

Figure 9: Status byte

- `APL_STATUS_BUSY`: The application layer is busy.
- `APL_STATUS_INACTIVE` (Meter device only): The device is inactive and does not send periodical meter data. Start installation procedure to connect the device to a data collector.
- `APL_STATUS_INSTALL` (Meter device only): This flag is set if the meter device is in installation mode.
- `APL_STATUS_CONNECTED` (Meter device only): This flag is set if the meter device is connected to a data collector.

The other bits are reserved for further versions.

Response:

- Status Byte

Example:

Request the current status.

SFD	Length	Type	CRC
A5	00 01	20	e1 64

SFD	Length	Type	Status Byte	CRC
A5	00 02	20	04	59 87

4.3.11 Memory synchronization

Command:

`<LENGTH>21`

Description:

This command initiates the Wireless M-Bus device to write the temporary parameters into the non-volatile memory (If non-volatile memory is available).

Please note, that the action can fail, if the Wireless M-Bus Stack is in a status where it is not possible to write to the non-volatile memory. In this case the confirmation returns the code 0x25.

Response:

- Confirmation: OK (0x0, 4.3.2)
- Confirmation: Failed if an error occured (0x2, 4.3.2)
- Confirmation: Non-volatile memory not ready (0x25, 4.3.2)

Example:

Memory synchronization.

SFD	Length	Type	CRC
A5	00 01	21	DC 01

SFD	Length	Type	Confirm	Tx Type	CRC
A5	00 03	00	00	21	DC 01

4.3.12 Clean up lost telegrams

Command:

```
<LENGTH>FA
```

Description:

If this command is called all received telegrams are searched and deleted.
This command shall be used if the stack always returns a buffer overflow.

Example:

Clean up lost telegrams.

SFD	Length	Type	CRC
A5	00 01	FA	9B 46

SFD	Length	Type	Confirm	Tx Type	CRC
A5	00 03	00	00	FA	9B 46

4.3.13 Manufacturer specific protocol

Command:

```
<LENGTH>FB[<DATA>]
```

Description:

The serial library allows to use manufacturer specific commands. The handling of the commands has to be implemented in the main application of the software which uses the serial library.

4.3.14 Internal events

Command:

```
<LENGTH>FC<CODE>
```

Description:

The serial library calls this command to signalize an internal event. Following event codes are provided:

- 0x1: Timeout - during receiving a serial telegram an error occured so the telegram could not be received completely. Possibly some telegram bytes are lost.
- 0x2: Overflow - during receiving a serial telegram an serial buffer overflow occured so the telegram could not be received completely.

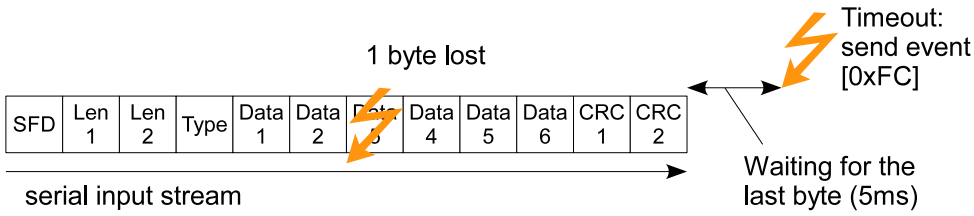

Figure 10: Timeout handling

4.3.15 Reset

Command:

```
<LENGTH>FF
```

Description:

This command is used to reset the device to perform a reset.

Be careful: The device performs a reset immediately regardless of the status of the stack. Please note, that by executing this command static parameters of the stack may be lost.

Response:

- none

Example:

Sending a reset command.

SFD	Length	Type	CRC
A5	00 01	FF	53 B7

4.4 Application Layer commands

4.4.1 List of commands

Table 17 shows an overview over all available application layer commands.

Command	MD	DC	Description	Chapters
30	X		Set error	4.4.2
31	X		Set alarm	4.4.3
33	X	X	Event: Telegram available	4.4.4
34		X	Event: New meter	4.4.5
35		X	Event: Telegram ACD bit set	4.4.6
36	X		Event: User data requested	4.4.7
37	X	X	Set user data (byte)	4.4.8
38	X	X	Event: Packet sent	4.4.9
39	X		Event: Alarm telegram transmitted	4.4.10
3A		X	Event: ACC_DMD telegram received	4.4.11
3B	X		Set FAC Mode	4.4.40
40	X	X	Destroy telegram	4.4.12
41	X		Send installation request	4.4.13
42			Reserved for future use.	
43			Reserved for future use.	
44	X		Set accessibility of the meter device	4.4.14
45	X		Get accessibility of the meter device	4.4.15
46		X	Send command clock sync	4.4.16
47		X	Send error request	4.4.17
48		X	Send user data request	4.4.18
4A		X	Read application error	4.4.19
4C	X		Set error flag(s)	4.4.20
4D	X		Clear error flag(s)	4.4.21
4E		X	Open bidirectional communication	4.4.22
4F		X	Close bidirectional communication	4.4.23
50		X	Add meter device	4.4.24
51		X	Remove meter device	4.4.25
52		X	Get number of connected devices	4.4.26
53		X	Get address of meter device	4.4.27
54	X	X	Create a spontaneous telegram	4.4.28
55	X	X	Send a spontaneous telegram	4.4.29
56	X	X	Read data	4.4.30
57		X	Get rf adapter address	4.4.31
58		X	Set rf adapter address	4.4.32
59		X	Get meter specific encryption key	4.4.33
5A		X	Set meter specific encryption key	4.4.34
5B		X	Get list index of meter device	4.4.35
5C	X		Set long header apl address	4.4.36
5D	X	X	Get long header apl address	4.4.37
5E	X	X	Receive a whole telegram	4.4.38
5F		X	Create DSMR key exchange command	4.4.39

Table 17: Commands on application layer

4.4.2 Set error (available for Meter Device)

Command:

```
<LENGTH>30<ERROR>
```

Description:

Writes an error <ERROR> (1 byte) to the error register. The next time the data collector requests the error register, the error code previously set by this command is transmitted. Afterwards the error register is reset automatically. The command is responded by a confirmation.

Be careful: In this function the error protocol of [4] is handled. For how to set the application error flags read chapter 4.4.20. Table 18 shows the list of application errors.

Response:

- Confirmation: OK (0x0, 4.3.2)
- Confirmation: Too few bytes if parameter <ERROR> is missing (0x14, 4.3.2)

Example:

Sets the application busy error (08)

SFD	Length	Type	Error	CRC
A5	00 02	30	08	53 A8

SFD	Length	Type	Confirm	Tx Type	CRC
A5	00 03	00	00	30	70 9B

Code	Description
00	Unspecified error
01	Unimplemented CI-Field
02	Buffer too long, truncated
03	Too many records
04	Premature end of record
05	More than 10 DIFE's
06	More than 10 VIFE's
08	Application too busy for handling readout request
09	Too many readouts (for slaves with limited readouts per time)

Table 18: Error codes

4.4.3 Set alarm (available for Meter Device)

Command:

```
<LENGTH>31<ALARM>
```

Description:

Writes an alarm byte <ALARM> (1 byte) to the alarm register. The next time the data collector requests the alarm register, the alarm code previously set by this command is transmitted. Afterwards the alarm register is cleared automatically. The command is responded by a confirmation.

Response:

- Confirmation: OK (0x0, 4.3.2)
- Confirmation: Too few bytes if parameter <ALARM> is missing (0x14, 4.3.2)

Example:

Sets the temperature too high alarm (0x40) (0b01000000)

SFD	Length	Type	Alarm	CRC
A5	00 02	31	40	4C 0B

SFD	Length	Type	Confirm	Tx Type	CRC
A5	00 03	00	00	31	4D FE

Bit	Meaning with bit set	Meaning with bit not set
00	Unspecified alarm	No Unspecified alarm
01	Low Battery	No Low Battery
02	Device Malfunctioning	No Device Malfunctioning
03	Transmission Power (RSSI) too high	Transmission Power (RSSI) not too high
04	Transmission Power (RSSI) too low	Transmission Power (RSSI) not too low
05	Meter configuration changed	Meter configuration not changed
06	Temperature too high	Temperature not too high
07	Temperature too low	Temperature not too low

Table 19: Alarm bits (Defined by STZEDN)

4.4.4 Event: Telegram available

Command:

<LENGTH>33<STATUS><REQ_ID>[<TLG_ID><METER_ID><CONTROLINFO><DATALENGTH>
<ACCESSNO><FLAGS><METER_STATUS><QUALITY>]

Description:

This event is generated if a telegram is received and validated. After this, the content of the telegram may be read.

Parameters:

- Status <STATUS> (1 byte): Table 20 shows the list of status bytes.
- Request telegram <REQ_TLG> (1 byte): If the telegram is the response to a previous request, this field includes the request id. If the telegram is no response the field is set to FF.
- Telegram id <TLG_ID> (1 byte): This id field identifies the telegram buffer kept in the memory of the communication stack.

- Meter index <METER_ID> (2 bytes)
- Control information field <CONTROLINFO> (1 byte): Includes the CI field.
- Data length <DATALENGTH> (1 byte): Defines the number of available data bytes. The field includes the number of received data bytes.
- Access number <ACCESSNO> (1 byte): Access number of the telegram.
- Flags <FLAGS> (2 byts):
  - Flow Control (BIT0): Set if further telegrams may be sent to the meter device.
  - Access Demand (BIT1): Set if errors may be requested.
  - Signature (BIT2+BIT3):
    - Signature unknown: The signature algorithm is unknown.
    - Signature error: The encryption key is invalid.
    - No encryption: The telegram data is not encrypted.
  - Bidirectional (BIT5): The meter device uses a bidirectional mode.
  - Always on (BIT6): The data collector is always on.
  - Hop counter (BIT8+BIT9): Maximal 3 hops
- <METER_STATUS> (1 byte): Status byte of the meter telegram.
- Quality <QUALITY> (1 byte): Stores the link quality of the received telegram from 0 dBm (00) to -254 dBm (FE). FF indicates that the RF module does not provide reading the link quality.

Example:

Status=OK (01), no response (FF), telegram id=1, meter id=1 4E9A 01 02, user data no reply (C=44), further telegrams enabled (flow control=1), no user data class 1 (access=00), short header (CI=7A), -54 dBm (quality=36).

SFD	Length	Type	Stat	Req	Tlg	Meter	CI	DLen	Acc	Flags	MStat	Quality	CRC
A5	00 0D	33	01	FF	01	00 01	7A	00	1C	00 02	10	36	F4 7D

Code	Description
00	Unknown error
01	Telegram successfully received
02	Timeout after sending a request
03	Buffer overflow, the telegram could not be received completely
04	Signature unknown, the telegram could not be decrypted
05	Signature error, maybe the decryption key is invalid
06	Sending of a telegram failed

Table 20: Status codes of a received telegram

4.4.5 Event: New meter (available for Data Collector)

Command:

```
<LENGTH>34<ADDRESS><QUALITY>
```

Description:

A new meter device is available and can be added to the meter list. <ADDRESS> (8 byte) includes the address of the new meter device. If the device has to be added to the list, the meter device can be added by sending the appropriate command (cf. 4.4.24). The field <QUALITY> shows the link quality of the installation request from 0 dBm to -254 dBm. If this value is set to 255 no link quality is available.

Example:

New meter device with address 4E9A 12345678 01 02 available (link quality: -54 dBm).

SFD	Length	Type	Address	Quality	CRC
A5	00 09	34	4E 9A 12 34 56 78 01 02	36	EC FD

4.4.6 Event: ACC_DMD bit set (available for Collector Device)

Command:

```
<LENGTH>35<METER_ID>
```

Description:

This event is called if the collector receives a telegram with activated ACD bit in the C-Field. According to the EN-13757 and OMS, the collector should perform REQ-UD and REQ-UD2.

Example:

SFD	Length	Type	Meter ID	CRC
A5	00 03	35	00 00	cf 87

4.4.7 Event: User data requested (available for Meter Device)

Command:

```
<LENGTH>36
```

Description:

This event is called if a data collector wants to get user data from the meter.

Example:

SFD	Length	Type	CRC
A5	00 01	36	FF C5

4.4.8 Set user data (bytes)

Command:

```
<LENGTH>37<TLG_ID><OVERWRITE>[<DATA>]
```

Description:

Sets the user data of the meter device as byte stream. The user data has to be saved in buffers to accomplish the response time of requests.

- Telegram Id <TLG_ID>: Id of the telegram to add the data bytes to. If FE is chosen, the data is added to the next user data class 2 telegram. If FF is chosen, the data is added to the next periodical user data telegram.
- Overwrite all data bytes <OVERWRITE>: If this byte is set to 01, all data bytes of the telegram are removed prior to adding the new data to the telegram. If the byte is set to 00, the new bytes are attached to the telegram. This flag is only used for the telegram ids FE and FF. In all other telegrams this flag is ignored and the data is appended.
- Data bytes <DATA> (optional): Data bytes to add to the telegram.

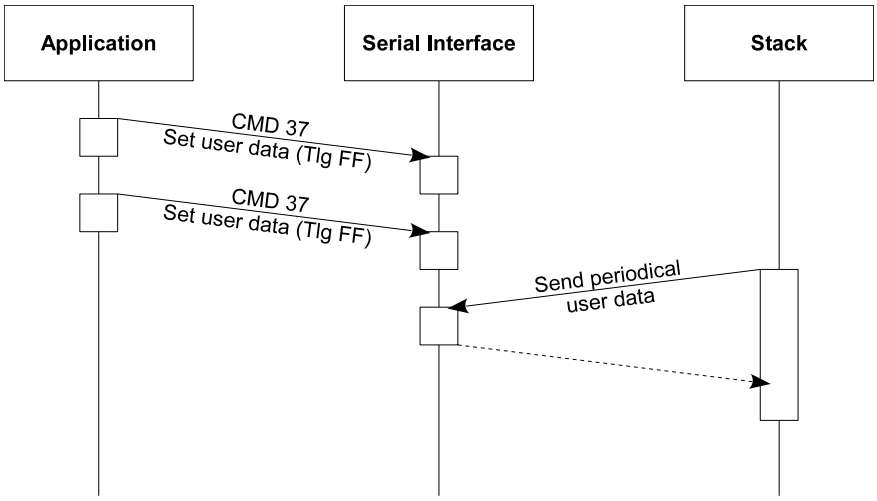

Figure 11: Sending periodical user data

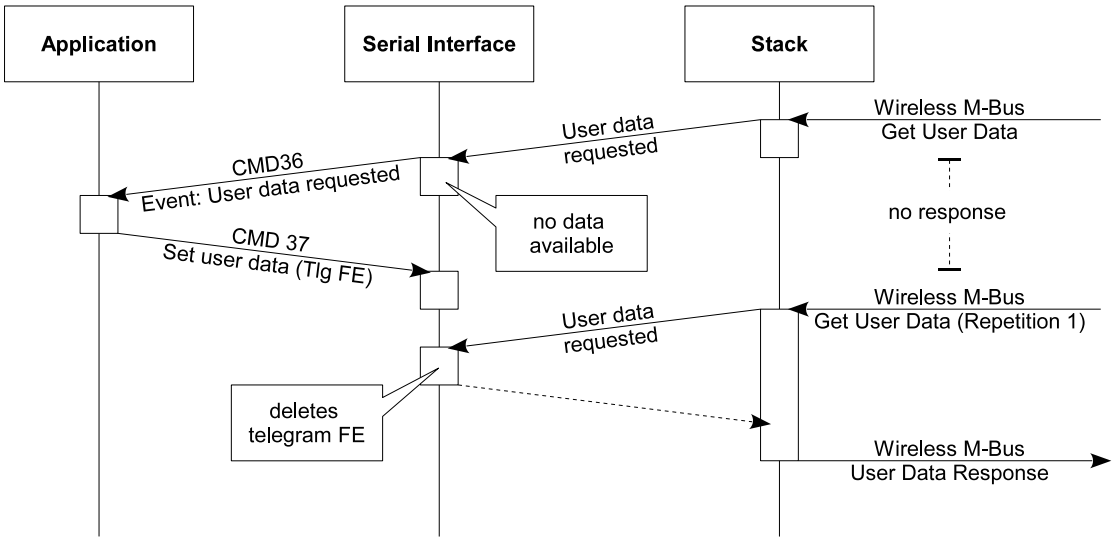

Figure 12: Sending an user data response

Response:

- Confirmation: OK (0x0, 4.3.2)
- Confirmation: Failed if the data could not be added to the selected buffer (0x2, 4.3.2)
- Confirmation: Too few bytes if one or more parameters are missing (0x14, 4.3.2)

Wait for event telegram available 4.4.4 to get the response of the request.

Example:

Adding data bytes to the end of the periodical user data telegram (Tlg Id=FF).

SFD	Length	Type	Tlg	Overw	Data	CRC
A5	00 0B	37	FF	00	01 02 03 04 05 06 07 08	8F 26

SFD	Length	Type	Confirm	Tx Type	CRC
A5	00 03	00	00	37	C2 A0

4.4.9 Event on packet sent

Event:

<LENGTH>38<METER_ID>

Description:

As soon as a packet has left the radio, the event 38 is sent to the host controller. This event occurs for data packets as well as for acknowledgements. The host can use the message for lighting an LED.

Example:

Frame format

SFD	Length	Type	Telegram ID	CRC
A5	00 02	38	00	91 1e

4.4.10 Event: Alarm telegram transmitted (available for Meter Device)

Command:

<LENGTH>39

Description:

This event is called if the meter has transmitted a alarm telegram.

Example:

SFD	Length	Type	CRC
A5	00 01	39	92 48

4.4.11 Event: ACC-DMD received (available for Data Collector)

Command:

<LENGTH>3A<Meter_ID>

Description:

The event is triggered when the collector receives an ACC-DMD telegram from a connected meter.

According to the EN-13757 and OMS, the collector should perform REQ_UD1 or REQ_UD2.

According to DSMR v.2.2, the collector should respond with ACK to this telegram.

DSMR v.4.0 and above does not accept ACC_DMD telegram.

Example:

SFD	Length	Type	Meter ID	CRC
A5	00 03	3A	00 00	d5 e7

4.4.12 Destroy telegram

Command:

```
<LENGTH>40<TLG_ID>
```

Description:

Destroys the telegram with id <TLG_ID> (1 byte). This command has to be called after reading the content of a received telegram.

Response:

- Confirmation: OK (0x0, 4.3.2)
- Confirmation: Failed if the <TLG_ID> is out of buffer memory (0x2, 4.3.2)
- Confirmation: Too few bytes if parameter <TLG_ID> is missing (0x14,

Wait for event telegram available 4.4.4 to get the response of the request.

Example:

Destroys the telegram with id 2

SFD	Length	Type	Tlg Id	CRC
A5	00 02	40	02	22 ED

SFD	Length	Type	Confirm	Tx Type	CRC
A5	00 03	00	00	40	C2 C9

4.4.13 Send installation request (available for Meter Device)

Command:

```
<LENGTH>41
```

Description:

Sends an installation request. After sending this requests the device goes to installation mode and repeats the request until an installation response is received.

Response:

- Local data: Id of the installation request telegram (4.3.4)
- Confirmation: Failed if the telegram could not be sent (0x2, 4.3.2)

Example:

Start installation mode.

SFD	Length	Type	CRC
A5	00 01	41	FF AC

SFD	Length	Type	Tx Type	Tlg Id	CRC
A5	00 03	02	41	03	1D 61

4.4.14 Set accessibility (Meter only)

Command:

<LENGTH>44<ACCESSIBILITY>

Description:

Set the accessibility parameter <ACCESSIBILITY> (1 bytes) of the meter device. The available acessibillity configurations are shown below:

```
/*! Enumeration of accessibility types displayed in the configuration field.
    This enum is only used for bidirectional meter devices */
typedef enum
{
  /* The last two bits of the configuration word will show that the
     telegram was send from an unidirectional device. (BIT15 = 0 Bit 14 = 0) */
  E_WMBUS_ACCESSIBILITY_UNIDIRECTIONAL = 0,
  /* The last two bits of the configuration word will show that the
     telegram was send from a bidirectional device. The meter supports
     bidirectional access in general, but there is no access window after this
     transmission  (BIT15 = 0 Bit 14 = 1) */
  E_WMBUS_ACCESSIBILITY_BIDIRECTIONAL_NO_ACCESS = 1,
  /* The last two bits of the configuration word will show that the
     telegram was send from a bidirectional device. The meter provides a short
     access window only immediately after this transmission   (BIT15 = 1 Bit 14 = 0) */
  E_WMBUS_ACCESSIBILITY_BIDIRECTIONAL_LIMITED_ACCESS = 2,
  /* The last two bits of the configuration word will show that the
     telegram was send from a bidirectional device. The meter provides unlimited
     access at least until next transmission  (BIT15 = 1 Bit 14 = 1) */
  E_WMBUS_ACCESSIBILITY_BIDIRECTIONAL_UNLIMITED_ACCESS = 3,
} E_WMBUS_ACCESSIBILITY_t;
```

Listing 1: Different accessibility configs of a bidirectional meter

Response:

- Confirmation: OK (0x0, 4.3.2)
- Confirmation: Failed if accessibility could not be set (0x2, 4.3.2)
- Confirmation: Too few bytes if parameter <ACCESSIBILITY> is missing. (0x14, 4.3.2)

Use the command 4.4.15 to get the current accessiblity configuration of the meter device.

Example:

Set the accessibility to `E_WMBUS_ACCESSIBILITY_BIDIRECTIONAL_NO_ACCESS`.

SFD	Length	Type	Accessibility	CRC
A5	00 02	44	01	F1 8D

SFD	Length	Type	Confirm	Tx Type	CRC
A5	00 03	00	00	44	37 5D

4.4.15 Get accessibility (Meter only)

Command:

`<LENGTH>45<ACCESSIBILITY>`

Description:

Get the accessibility parameter <ACCESSIBILITY> (1 bytes) of the meter device. The available acessibillity configurations are shown in listing 1.

Response:

- Local data: Current accessiblity configuration (4.3.4)
- Confirmation: Failed if accessibility could not be read (0x2, 4.3.2)

Use the command 4.4.14 to set the accessiblity of the meter device.

Example:

Get the accessibility.

SFD	Length	Type	CRC
A5	00 01	45	0A 38

SFD	Length	Type	Tx Type	Accessiblity	CRC
A5	00 03	02	45	01	F3 64

The current accessibility type is `E_WMBUS_ACCESSIBILITY_BIDIRECTIONAL_NO_ACCESS` according to listing 1.

4.4.16 Send command clock synchronization (available for Data Collector)

Command:

<LENGTH>46<METER_ID>

Description:

Transmits the current system time of the data collector to the meter device with index <METER_ID> (2 bytes).

Response:

- Local data: Id of the command telegram (4.3.4)
- Confirmation: Failed if the command telegram could not be sent (0x2, 4.3.2)
- Confirmation: Too few bytes if the parameter <METER_ID> is missing. (0x14, 4.3.2)

Wait for event telegram available 4.4.4 to get the response of the request.

Example:

Sends the current system time to the meter device with the index 3

SFD	Length	Type	Meter Id	CRC
A5	00 03	46	00 03	91 AA

SFD	Length	Type	Tx Type	Tlg Id	CRC
A5	00 03	02	46	03	9A A3

4.4.17 Send error request (available for Data Collector)

Command:

<LENGTH>47<METER_ID>

Description:

Requests the error register of the meter device with index <METER_ID> (2 bytes).

Response:

- Local data: Id of the request telegram (4.3.4)
- Confirmation: Failed if the request telegram could not be sent (0x2, 4.3.2)
- Confirmation: Too few bytes if the parameter <METER_ID> is missing. (0x14, 4.3.2)

Wait for event telegram available 4.4.4 to get the response of the request.

Example:

Sends the request to the meter device with the index 3

SFD	Length	Type	Meter Id	CRC
A5	00 03	47	00 03	74 00

SFD	Length	Type	Tx Type	Tlg Id	CRC
A5	00 03	02	47	03	5D 7B

4.4.18 Send user data request (available for Data Collector)

Command:

<LENGTH>48<METER_ID>

Description:

Requests the user data of the meter device with index <METER_ID> (2 bytes).

Response:

- Local data: Id of the request telegram (4.3.4)
- Confirmation: Failed if the request telegram could not be sent (0x2, 4.3.2)
- Confirmation: Too few bytes if the parameter <METER_ID> is missing. (0x14, 4.3.2)

Wait for event telegram available 4.4.4 to get the response of the request.

Example:

Sends the request to the meter device with the index 3

SFD	Length	Type	Meter Id	CRC
A5	00 03	48	00 03	29 F7

SFD	Length	Type	Tx Type	Tlg Id	CRC
A5	00 03	02	48	03	FD 42

4.4.19 Read application error (available for Data Collector)

Command:

<LENGTH>4A<TLG_ID>

Description:

Reads the application error of a received telegram <TLG_ID> (1 byte). This command has to be called after receiving a telegram (cf. 4.4.4 with CI=70. The response is received as local data.

Response:

- Local data: 1 byte error code (4.3.4)
- Confirmation: Failed if the error code is invalid (0x2, 4.3.2)
- Confirmation: Too few bytes if parameter <TLG_ID> is missing. (0x14, 4.3.2)

Example:

Reads the application error from telegram 2. The response includes the error code "Unimplemented CI-Field" (01)

SFD	Length	Type	Tlg Id	CRC
A5	00 02	4A	02	E2 C3

SFD	Length	Type	Tx Type	Error	CRC
A5	00 03	02	4A	01	53 5D

4.4.20 Set error flag(s) (available for Meter Device)

Command:

<LENGTH>4C<ERROR>

Description:

Sets the flag that an application error <ERROR> (1 byte) is available. A list of application errors is shown in table 21. Figure 13 shows how to use the command and in which Wireless M-Bus fields the information is transmitted.

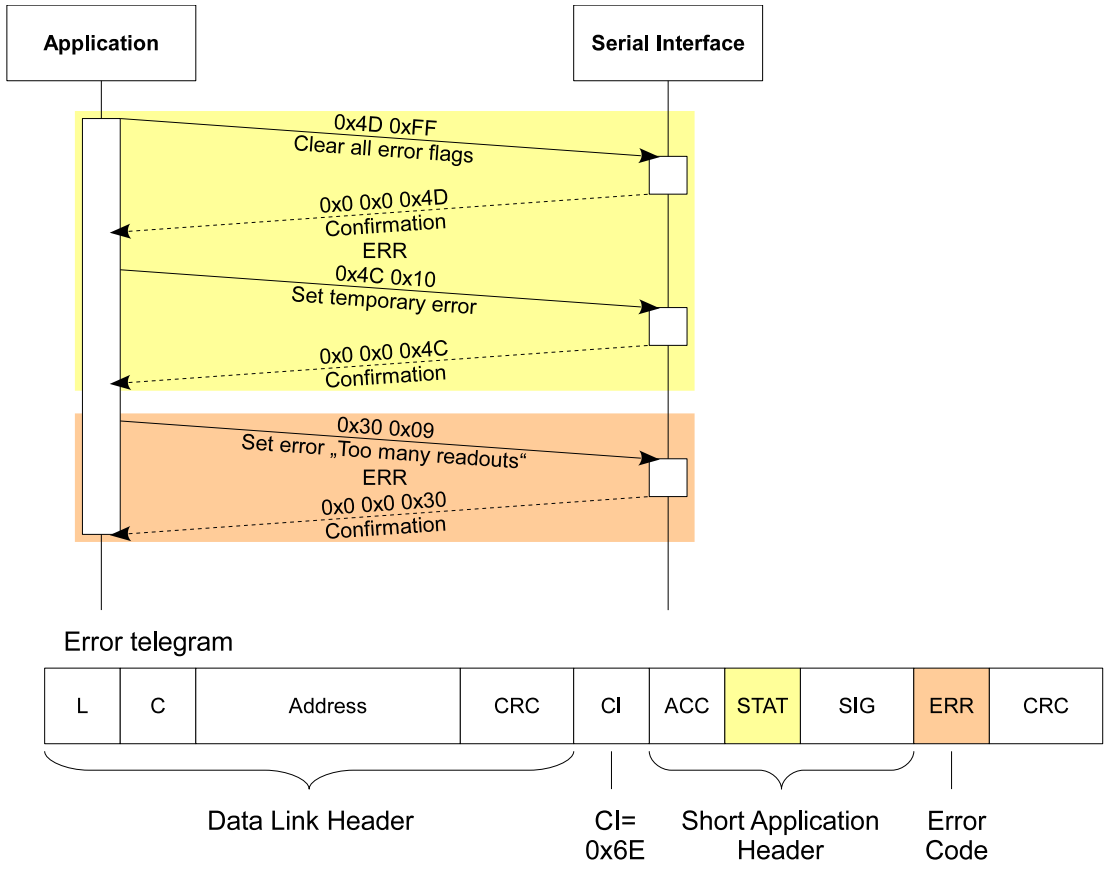

Figure 13: Using the serial error commands

Response:

- Confirmation: OK (0x0, 4.3.2)
- Confirmation: Too few bytes if parameter <ERROR> is missing. (0x14, 4.3.2)

Example:

Sets a permanent error (08)

SFD	Length	Type	Error	CRC
A5	00 02	4C	08	0E 5E

SFD	Length	Type	Confirm	Tx Type	CRC
A5	00 03	00	00	4C	E1 10

Code	Description
00	No error
01	Application busy
02	Unknown application error
04	Power low
08	Permanent error
10	Temporary error

Table 21: Application error flags

4.4.21 Clear error flag(s) (available for Meter Device)

Command:

```
<LENGTH>4D<ERROR>
```

Description:

Clears the flag of an application error <ERROR> (1 byte). A list of application errors is shown in table 21

Response:

- Confirmation: OK (0x0, 4.3.2)
- Confirmation: Too few bytes if parameter <ERROR> is missing. (0x14, 4.3.2)

Example:

Clears the temporary error (10)

SFD	Length	Type	Error	CRC
A5	00 02	4D	10	49 EC

SFD	Length	Type	Confirm	Tx Type	CRC
A5	00 03	00	00	4D	DC 75

4.4.22 Open bidirectional communication (available for Data Collector

Command:

```
<LENGTH>4E<Telegram_ID>
```

Description:

Start a bidirectional communication with a meter device. Befor a bidirectional communication can be opened a telegram has to be prepared. The data collector has to wait until a telegram from the meter device is received. Then, the bidirectional communication is opened and requests may be sent to the meter device.

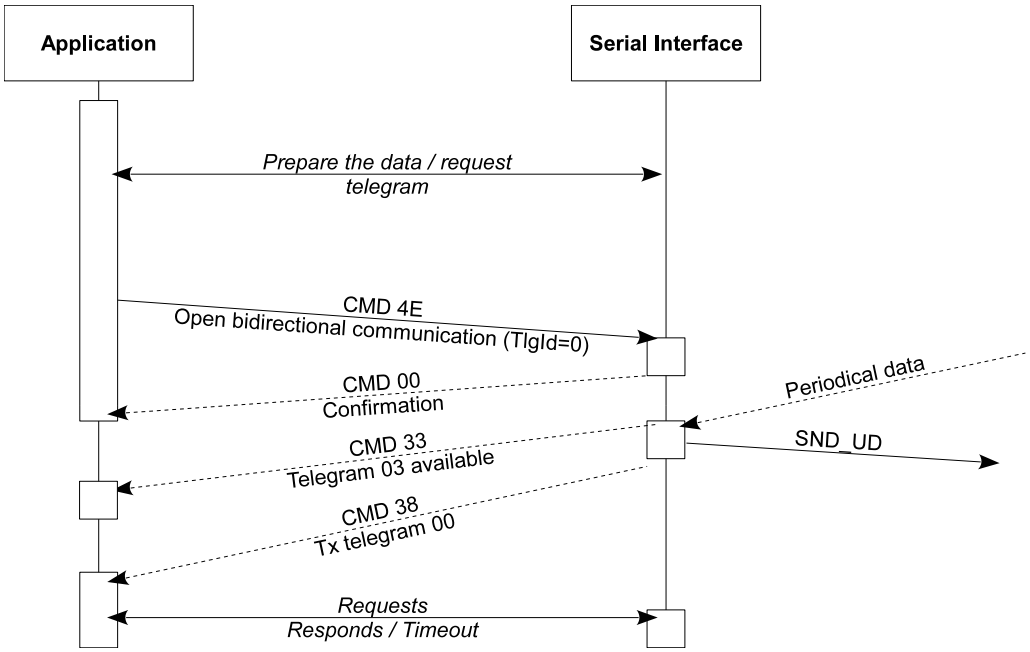

Figure 14: Sequence to start a bidirectional communication

Response:

- Confirmation: OK (0x0, 4.3.2)
- Confirmation: Too few bytes if parameter <Telegram_ID> is missing. (0x14, 4.3.2)

Example:

Opens the communication with telegram id 0

SFD	Length	Type	Telegram Id	CRC
A5	00 02	4E	00	0C C6

SFD	Length	Type	Confirm	Tx Type	CRC
A5	00 03	00	00	4E	9B DA

4.4.23 Close bidirectional communication (available for Data Collector)

Command:

<LENGTH>4F<METER_ID>

Description:

Closes the bidirectional communication with the meter device with index <METER_ID> (2 bytes).

Response:

- Local data: Telegram Id (4.3.4)

- Confirmation: Failed if the bidirectional communication could not be closed. (0x2, 4.3.2)
- Confirmation: Too few bytes if parameter <METER_ID> is missing. (0x14, 4.3.2)

Example:

Closes the communication with meter device 3

SFD	Length	Type	Meter Id	CRC
A5	00 03	4F	00 03	EB 6B

SFD	Length	Type	Tx Type	Telegram Id	CRC
A5	00 03	02	4F	00	0E 2F

4.4.24 Add meter device (available for Data Collector)

Command:

```
<LENGTH>50<METER_ADDR>[<KEY>][<RF_ADAPTER>][WMBUS_MODE]
```

Description:

Adds the meter device with meter address <METER_ADDR> (8 byte) to the meter list. Optional an encryption key <KEY> (16 byte) and a RF adapter address <RF_ADAPTER> (8 byte) can be added.
The response includes the index of the new meter. In case a negative confirmation is received, because the meter list has no space left, it is possible to delete another device (cf. 4.4.25) in order to add the new device anyway.

For a specific feature of C Mode collector, it is possible to specify the mode of the meter either it is a C mode meter or T mode meter via <WMBUS_MODE> parameter (refers to 22) (default is C mode)

Response:

- Local data: Id of the meter device (4.3.4)
- Confirmation: Failed if meter could not be added to the list. (0x2, 4.3.2)
- Confirmation: Too few bytes if the parameter <PRIM_ADDRESS> is missing. (0x14, 4.3.2)
- Confirmation: Indicates that the meter already exitsts. (0x26, 4.3.2)

Example:

Adds a meter device to the meter list with address 4E9A 12345678 01 02 (without secondary address). The response includes the meter index 5.

SFD	Length	Type	Meter Address	CRC
A5	00 09	50	4E 9A 12 34 56 78 01 02	4B 46

SFD	Length	Type	Tx Type	Meter Id	CRC
A5	00 04	02	50	00 05	4C 11

Mode	Code	Description
E_WMBUS_MODE_S	0x00	Mode S @ 868MHz
E_WMBUS_MODE_T	0x01	Mode T @ 868MHz
E_WMBUS_MODE_C	0x02	Mode C @ 868MHz
E_WMBUS_MODE_N	0x03	Mode N @ 169MHz
E_WMBUS_MODE_S_SYNC	0x04	Mode S @ 868MHz. For synchronized messages in meters
E_WMBUS_MODE_UNKNOWN	0x05	Unknown mode

Table 22: Enumeration for Wireless M-Bus Modes

4.4.25 Remove meter device (available for Data Collector)

Command:

<LENGTH>51<METER_ID>

Description:

Removes the meter device with index <METER_ID> (2 bytes) from the meter list. Set the meter id to 0xFFFF to clear the complete meter list.

Response:

- Confirmation: OK. (0x0, 4.3.2)
- Confirmation: Too few bytes if the parameter <METER_ID> is missing. (0x14, 4.3.2)

Example:

Removes the meter device with index 5

SFD	Length	Type	Meter Id	CRC
A5	00 03	51	00 05	DF DB

SFD	Length	Type	Confirm	Tx Type	CRC
A5	00 03	00	00	51	6E 53

4.4.26 Get number of connected devices (available for Data Collector)

Command:

<LENGTH>52

Description:

Reads the number of connected meter devices.

Response:

- Local data: Number of connected meter devices (4.3.4)

Example:

Currently 3 meter devices are connected.

SFD	Length	Type	CRC
A5	00 01	52	29 FC

SFD	Length	Type	Tx Type	Number	CRC
A5	00 04	02	52	00 03	14 9A

4.4.27 Get address of meter device (available for Data Collector)

Command:

```
<LENGTH>53<METER_ID>
```

Description:

Reads the address of the connected meter device with index <METER_ID> (2 bytes).

Response:

- Local data: Address of the meter device (4.3.4)
- Confirmation: Failed if there is no meter with id <METER_ID>. (0x2, 4.3.2)
- Confirmation: Too few bytes if the parameter <METER_ID> is missing. (0x14, 4.3.2)

Example:

Reads the address of meter device 3 (Result: 4E9A 12345678 01 02)

SFD	Length	Type	Meter Id	CRC
A5	00 03	53	00 03	A6 B4

SFD	Length	Type	Tx Type	Meter Address	CRC
A5	00 0A	02	53	4E 9A 12 34 56 78 01 02	46 0B

4.4.28 Creates a spontaneous telegram

Command:

```
<LENGTH>54<CONTROLINFO>[<METER_ID>][<ENCRYPTION>]
```

Description:

Creates a spontaneous telegram. The field <CONTROLINFO> (1 byte) includes the CI field. If the device is collector <METER_ID> (2 bytes) includes the meter index. If the device is a meter this field is not needed. The field [<ENCRYPTION>] (1 byte) is for the encryption of the telegram. If the ENCRYPTION value is not given, the telegram is encrypted by default.

- 0x0: Encryption disabled
- 0x1: Encryption enabled

Before sending the telegram (see chapter 4.4.29) it may be filled with data (see chapter 4.4.8). The respond includes the telegram id. If the telegram can not be created FF is returned.
Figure 15 shows how to create and send a spontaneous telegram.

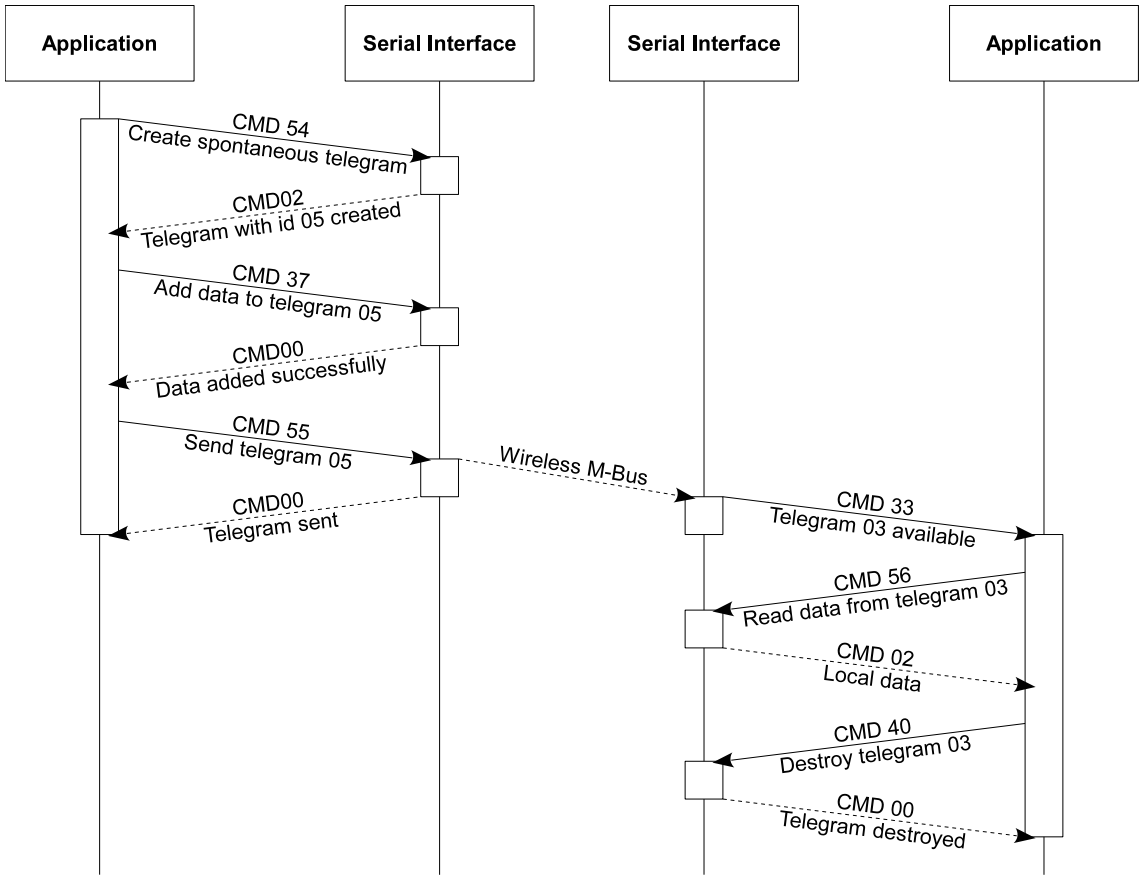

Figure 15: Creating and sending a spontaneous telegram

Response:

- Local data: Id of the telegram (4.3.4)
- Confirmation: Failed if no telegram could be created. (0x2, 4.3.2)
- Confirmation: Too few bytes if the parameter <CONTROLINFO> is missing. (0x14, 4.3.2)

Example:

Creates a spontaneous telegram which gets the index 6.

SFD	Length	Type	CI	Meter Id	CRC
A5	00 04	54	5B	00 00	F3 0C

SFD	Length	Type	Tx Type	Tlg Id	CRC
A5	00 03	02	54	06	9C 71

4.4.29 Sends a spontaneous telegram

Command:

```
<LENGTH>55<TLG_ID>
```

Description:

After creating a spontanoues telegram (see chapter 4.4.28) and filling with data the telegram may be sent by this command.

Response:

- Confirmation: OK (4.3.2)
- Confirmation: Failed if the telegram could not be sent. (0x2, 4.3.2)
- Confirmation: Too few bytes if the parameter <TLG_ID> is missing. (0x14, 4.3.2)

Example:

Send the spontaneous telegram with id 06.

SFD	Length	Type	Tlg Id	CRC
A5	00 02	55	06	9E 98

SFD	Length	Type	Confirm	Tx Type	CRC
A5	00 02	00	00	55	9B C7

4.4.30 Read data (byte)

Command:

```
<LENGTH>56<TLG_ID><LEN><OFFSET>
```

Description:

Reads data bytes of a received telegram <TLG_ID> (1 byte). The length of the data to read may be selected by <LEN> (1 byte). The beginning of data to read is selected by <OFFSET> (1 byte). The response is received as local data. Be careful at reading data. If you read out the data by the command read data (byte) you have

to manage it by yourself.

The data is transmitted without further formating in reverse order. So you have to input the data data type by data type to keep the order at receiving. Figure 16 shows some cases of reading out data.

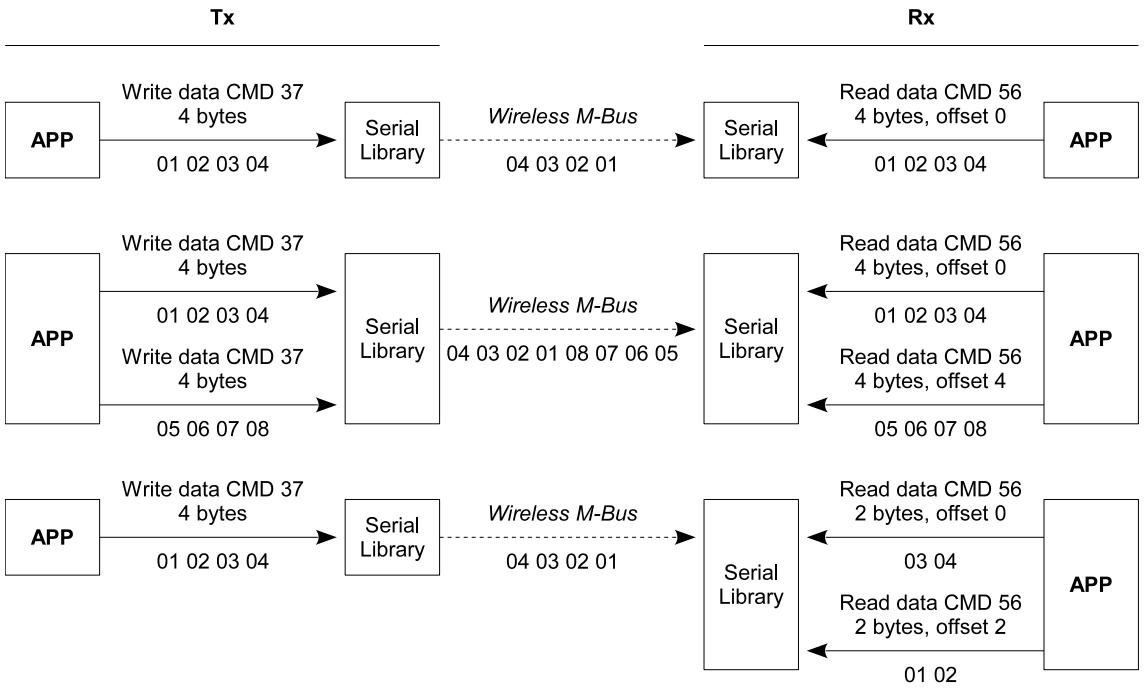

Figure 16: Writing and reading data to a telegram by using binary reading and writing commands

Response:

- Local data with read data bytes. (4.3.4)
- Confirmation: Failed if no data bytes could be read. (0x2, 4.3.2)
- Confirmation: Too few bytes if one or more parameters are missing. (0x14, 4.3.2)

Example:

Reads data from byte 5 to 11.

SFD	Length	Type	Tlg Id	Length	Offset	CRC
A5	00 04	56	01	07	05	33 65

SFD	Length	Type	Tx Type	Data	CRC
A5	00 08	02	56	05 06 07 08 09 0A 0B	3D BD

4.4.31 Get rf adapter address of a meter (available for Data Collector)

Command:

```
<LENGTH>57<METER_ID>
```

Description:

Reads the rf adapter address of the meter width id <METER_ID>.

Response:

- Local data with rf adapter address. (4.3.4)
- Confirmation: Failed if the meter does not exist or no rf adapter address is available.
- Confirmation: Too few bytes if one or more parameters are missing. (0x14, 4.3.2)

Example:

Reads the rf adapter address of the meter on position 1

SFD	Length	Type	Meter Id	CRC
A5	00 03	57	00 01	0D 79

SFD	Length	Type	Tx Type	Address	CRC
A5	00 0A	02	57	4E 9A 00 00 00 01 01 00	3B 20

4.4.32 Set rf adapter address of a meter (available for Data Collector)

Command:

```
<LENGTH>58<METER_ID>[<ADDRESS>]
```

Description:

Sets the rf adapter address of the meter width id <METER_ID>. If the optional parameter <ADDRESS> is not set an existing rf adapter address is deleted.

Response:

- Confirmation: OK (0x0, 4.3.2)
- Confirmation: Failed if the meter does not exist. (0x2, 4.3.2)
- Confirmation: Too few bytes if one or more parameters are missing. (0x14, 4.3.2)

Example:

Sets the rf adapter address of the meter on position 1

SFD	Length	Type	Meter Id	Address	CRC
A5	00 0B	58	00 01	4E 9A 0 0 0 01 01 00	FF 53

SFD	Length	Type	Confirm	Tx Type	CRC
A5	00 03	00	00	58	85 7B

4.4.33 Get meter specific encryption key (available for Data Collector)

Command:

<LENGTH>59<METER_ID>

Description:

Reads the meter specific encryption key of the meter width id <METER_ID>.

Response:

- Local data with encryption key. (4.3.4)
- Confirmation: Failed if the meter does not exist of no rf adapter address is available.
- Confirmation: Too few bytes if one or more parameters are missing. (0x14, 4.3.2)

Example:

Reads the encryption key of the meter on position 1

SFD	Length	Type	Meter Id	CRC
A5	00 03	59	00 01	B5 24

SFD	Length	Type	Tx Type	Key	CRC
A5	00 12	02	59	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF	D2 43

4.4.34 Set meter specific encryption key (available for Data Collector)

Command:

<LENGTH>5A<METER_ID><KEY>

Description:

Sets the encryption key address of the meter width id <METER_ID>. Set <METER_ID> to 0xFFFF to set the same encryption key for all meter devices.

Response:

- Confirmation: OK (0x0, 4.3.2)
- Confirmation: Failed if the meter does not exist. (0x2, 4.3.2)

- Confirmation: Too few bytes if one or more parameters are missing. (0x14, 4.3.2)

Example:

Sets the encryption key of the meter on position 1

SFD	Length	Type	Meter Id	Key	CRC
A5	00 13	5A	00 01	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF	1D 64

SFD	Length	Type	Confirm	Tx Type	CRC
A5	00 03	00	00	5A	FF B1

4.4.35 Get list index of meter device (available for Data Collector)

Command:

<LENGTH>5B<ADDRESS>

Description:

Reads the lsit index of the connected meter device with address <ADDRESS> (8 bytes).

Response:

- Local data: List index of the meter device (4.3.4)
- Confirmation: Failed if there is no meter with address <ADDRESS>. (0x2, 4.3.2)
- Confirmation: Too few bytes if the parameter <ADDRESS> is missing. (0x14, 4.3.2)

Example:

Reads the index of the meter device with the address 4E9A 12345678 01 02 (Result: 3)

SFD	Length	Type	Meter Address	CRC
A5	00 09	5B	4E 9A 12 34 56 78 1 2	42 39

SFD	Length	Type	Tx Type	Meter Index	CRC
A5	00 04	02	5B	00 03	4F BF

4.4.36 Set long header address (available for Meter)

Command:

<LENGTH>5C<TLG_ID><METER_ADDR>

Description:

Sets the long header address for the application layer of an outgoing telegram with long header. By default, this is identical to the own address, but this function call permits to enter the address of e.g. a connected meter Before using this function, create the telegram, and pass the ID here.

- Telegram Id <TLG_ID>: Id of the telegram to add the address.
- Meter address <METER_ADDR> (8byte) address for the apl header.

Response:

- Confirmation: OK (0x0, 4.3.2)
- Confirmation: Failed if the data could not be added to the selected telegram (0x2, 4.3.2)
- Confirmation: Too few bytes if one or more parameters are missing (0x14, 4.3.2)

Example:

Adding apl address to a specific telegram (Tlg Id=01).

SFD	Length	Type	Tlg	Meter Address	CRC
A5	00 0B	5C	01	4E 9A 12 34 56 78 01 02	58 32

SFD	Length	Type	Confirm	Tx Type	CRC
A5	00 03	00	00	5C	70 EF

4.4.37 Get apl address of a received long header telegram (available for Data Collector)

Command:

<LENGTH>5D<TLG_ID>

Description:

Reads the apl address from a received telegram with a long header CI-Field.

Response:

- Local data: Apl address of the meter device (4.3.4)
- Confirmation: Failed if there is no telegram with id <TLG_ID> or if the header type is wrong. (0x2, 4.3.2)
- Confirmation: Too few bytes if the parameter <TLG_ID> is missing. (0x14, 4.3.2)

Example:

Reads the apl address of telegram with id 0 (Result: 4E9A 12345678 01 02)

SFD	Length	Type	Telegram ID	CRC
A5	00 02	5D	00	05 3D

SFD	Length	Type	Tx Type	Meter Address	CRC
A5	00 0A	02	5D	4E 9A 12 34 56 78 01 02	10 50

4.4.38 Read the whole telegram and destroy it afterwards

Command:

```
<LENGTH>5E<TLG_ID>
```

Description:

This command will read out the whole telegram and destroy it afterwards.

Response:

- Local data: payload of the telegram (4.3.4)
- Confirmation: No data if there is no telegram with id <TLG_ID>. (0x03, 4.3.2)
- Confirmation: Too few bytes if the parameter <TLG_ID> is missing. (0x14, 4.3.2)

Example:

Reads the telegram with id 0

SFD	Length	Type	Telegram ID	CRC
A5	00 02	5E	00	25 30

SFD	Length	Type	Tx Type	Payload of the telegram	CRC
A5	00 0A	02	5E	40 01 01 00 80 2A 6D 06	86 DF

4.4.39 Create DSMR key exchange command (available for Data Collector)

Command:

```
<LENGTH>5F<METER_ID><TRANSFER_KEY>
```

Description:

This command will create the encryption key <TRANSFER_KEY> exchange command for DSMR to the meter device with index <METER_ID>. Note that the <TRANSFER_KEY> is the result of the Link Key (the actual encryption key) encrypted by the Master Key.

Response:

- Local data: Id of the command telegram (4.3.4)
- Confirmation: Failed if the key exchange command telegram could not be created. (0x2, 4.3.2)

- Confirmation: Too few bytes if one or more parameters are missing. (0x14, 4.3.2)

Example:

Create the DSMR key exchange for the meter device with index 3.

SFD	Length	Type	Meter Id	Key	CRC
A5	00 13	5F	00 03	00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF	B4 B4

SFD	Length	Type	Tx Type	Tlg Id	CRC
A5	00 03	02	5F	03	60 76

4.4.40 Set FAC mode (available for Meter Device)

Command:

<LENGTH>3B<MODE>

Description:

This command enables or disables the FAC mode for a meter device. Available modes are:

- 0x00: Disable FAC mode
- 0x01: Enable FAC mode

Response:

- Confirmation: OK. (0x0, 4.3.2)
- Confirmation: Failed if the mode is invalid. (0x2, 4.3.2)
- Confirmation: Too few bytes if the parameter <MODE> is missing. (0x14, 4.3.2)

Example:

Enable FAC mode.

SFD	Length	Type	Mode	CRC
A5	00 02	3B	01	8C 76

SFD	Length	Type	Confirm	Tx Type	CRC
A5	00 03	00	00	3B	E1 79

5 Contact information

In case of questions, requests for quotations or ideas for further improvements, please contact:

Steinbeis Transfer Center for Embedded Design and Networking (STZEDN)

Poststrasse 35

D-79423 Heitersheim

Germany

Tel: +49 7634-6949-340

Fax: +49 7634-5049-886

Url: http://www.stzedn.de/smart-metering.html

E-mail: metering@stzedn.de

References

[1] “Communication systems for meters and remote reading of meters.” Part 4: Wireless meter readout (Radio meter reading for operation in the 868 MHz to 870 MHz SRD band); German version EN 13757-4, 2011.

[2] “Communication systems for meters and remote reading of meters.” Part 1: Data exchange; English version EN 13757-1, 2002.

[3] “Communication systems for meters and remote reading of meters.” Part 2: Physical and link layer; English version EN 137572, 2004.

[4] “Communication systems for meters and remote reading of meters.” Part 3: Dedicated application layer; English version EN 137573, 2011.

[5] “Communication systems for meters and remote reading of meters.” Part 5: Wireless Relaying; English version prEN 13757-5, 2009.

[6] “Telecontrol equipment and systems.” Part 5: Transmission protocols; EN 870-5, 2002.

[7] “Open metering system specification - volume 2: Primary communication.” http://www.oms-group.org/download/OMS-Spec_Vol2_Primary_v301.pdf, 2011.

A Application Errors

Define	Description	Value
APL_ERROR_UNSPECIFIED	Unspecified error: also if data field is missing	0x00
APL_ERROR_CI_UNIMPLEMENTED	Unimplemented CI-Field	0x01
APL_ERROR_BUFFER_TOO_LONG	Buffer too long, truncated	0x02
APL_ERROR_TOO_MANY_RECORDS	Too many records	0x03
APL_ERROR_PREMATURE_END_OF_REC	Premature end of record	0x04
APL_ERROR_TOO_MANY_DIFES	More than 10 DIFE's	0x05
APL_ERROR_TOO_MANY_VIFES	More than 10 VIFE's	0x06
APL_ERROR_APPLICATION_BUSY	Application too busy for handling read-out request	0x08
APL_ERROR_TOO_MANY_READOUTS	Too many readouts (for slaves with limited readouts per time)	0x09
...	Reserved	0x0A ... 0xFF

Table 23: Application errors.

B Configuration codes

Command	Description	Parameters
01	M-Bus mode	1 byte mode, the list of available modes is shown in table 25
02	M-Bus Device	1 byte device, the list of available modes is shown in table 26
03	RF Power	1 byte power in dBm. It handles values from -130dBm (00) to 125dBm (FE). Writable only.
05-0F	Reserved	
10-14	Reserved	
15	Primary address	8 bytes: • 2 bytes manufacturer id • 4 bytes ident number • 1 byte version • 1 byte type
16	Periodical interval (Meter only)	4 byte MSB first
17	Encryption key	16 bytes AES128 encryption key
18	System time	7 bytes: • 2 bytes year (2000...2099, MSB first) • 1 byte month (1...12) • 1 byte day (1...31) • 1 byte hours (0...23) • 1 byte minutes (0...59) • 1 byte seconds (0...59)
19	Reserved	
1B	Sets the flag that the meter is connected (used for manual installation)	

Table 24: List of configuration types

1C	Sets the address of the connected data collector (needed for OMS)	
1D	Sets the channel (current only for Mode N)	2 bytes mode, the list of avilable cahnnels is shown in table 27
1E	Configure the transceiver carrier frequency offset	2 Bytet signed Parameter: 0x80 0x00 -> -1MHz 0x7F 0xFF -> +1MHz. Afterwards reset the target with the command FF.
1F–2F	Reserved for further Wireless M-Bus configuration	
30	Enter the hardware test mode	
31	Set/Get the current test mode	1 byte: • 0 = hardware test mode idle • 1 = generate carrier • 2 = generate modulated carrier • 3 = hardware test mode rx
32	Sets the timer counter register.	2 bytes: value
33	Adds a positive or negative offset to the timer counter	2 bytes: The value must be in two's complement notation! The value is the deviation per second in percent, multiplied by 100 . Example: The clock runs every second 6ms to slow. The deviation is: -0,6 percent The value is: -60 The value in two's complement notation is: 0XFFC4
34-FB	Reserved for further applications	
FC	Protocol stack version	2 bytes: • 4 bit main version • 4 bit sub version • 8 bit sub sub version
FD	Flash erase	
FE	Product version	2 bytes: • 4 bit main version • 4 bit sub version • 8 bit sub sub version

Table 24: List of configuration types

Code	Mode
00	Off
01	S1
03	S2
04	T1
05	T2
07	C1
08	C2
09	N1
0A	N2
FF	Unknown

Table 25: Wireless M-Bus modes

Code	Device
01	Meter
02	Data Collector

Table 26: Wireless M-Bus devices

Code	Channel
0000	ignored
0001	S
0002	T
0004	C
0008	Na
0010	Nb
0020	Nc
0040	Nd
0080	Ne
0100	Nf
0200	Ng

Table 27: Wireless M-Bus channels

C STZEDN License Terms

C.1 Product Licensing

The term "Product" pertains to any of the following products: Wireless M-Bus Stack, Wireless M-Bus Suite and any and all files included with the product, or received as an update to the product. The term "User" pertains to the purchaser of the product, be it an individual or a company.

C.2 No warranty

The Product is provided "as is" and use of the Product is at User's risk. Steinbeis GmbH & Co. KG fuer Technologietransfer, represented by Prof. Dr. lng. Axel Sikora, director of the Steinbeis Transfer Center Embedded Design and Networking (STZEDN), with an address of Poststrasse 35, 79423 Heitersheim, Germany, does not warrant nor make any condition or representations that the functions contained in the Product will meet the requirements of the User, that the operation of the Product will be uninterrupted nor error free, that defects in the Product will be corrected. Further, STZEDN does not warrant or make any conditions or representations regarding the use or the results of the use of the Product in terms of its correctness, accuracy, reliability or otherwise. No oral or written information or advice given by STZEDN or others shall create a warranty, condition or representation or in any way extend the scope of the "as is" provision the Product.

C.3 Disclaimer of liability

The entire risk as to the results and performance of the Product is assumed by the User. Neither STZEDN nor its affiliates, distributors, resellers, suppliers, agents, officers and directors shall have any liability to the User or to any other person or entity for any damages howsoever caused including, but not limited to, direct, indirect, incidental, special, general, consequential, punitive or exemplary damages whatsoever including, but not limited to, loss of revenue or profit, damages to property or persons, lost or damaged data, or other commercial or economic loss, even if STZEDN has been advised of the possibility of such damages or they are foreseeable, or for claims by third party. This section shall apply whether or not the breach, default, non-performance of failure is a breach of fundamental condition or term, or a fundamental breach. In no event shall STZEDN be liable to the other party for any loss or injuries to earnings, profits or good will, or for any incidental, special, punitive or consequential damages of any person or entity whether arising in contract, tort or otherwise, even if either party has been advised of the possibility of such damages.

D Third Party License Terms

D.1 AES library

The following license terms apply to the AES module in use by the STZEDN stack.

Issue 09/09/2006

This is an AES implementation that uses only 8-bit byte operations on the cipher state

****