SMA Data

Specification

Version 1.25

Definition and Description
of the Data Telegram Format and
Communication Protocol

SMADAT-11:ZE2203

Alteration Review

Document Number SMADAT	**Version and alteration type [1])**		**Comments**	**Author**
-11:ZE2203	1.1		Translation of the German version 1.25	

[1]) A: Alteration due to faulty documentation or improvement of the documentation

B: Alterations maintaining full or upward compatibility

C: Alterations limiting or excluding compatibility

	Name	**Date**	**Signature**
Reviewed	Bröring	02.06.2003	

Explanation of symbols used in this document

To enable optimal usage of this manual and safe operation of the device during installation, operation and maintenance routines, please note the following description of symbols:

This indicates a feature that is important either for optimal and comfortable usage or optimal operation of the system.

Example: „You will finished C routines on the support disc.“

This indicates a fact or feature which is very important for the safety of the user and / or which can cause a serious defect if not applied appropriately.

Example: „Disconnect the mains plug before opening the device!“

This indicates an example.

Contents

1 Introduction

The data telegrams of SMA Data Communication consist of two parts, the telegram frame and the actual contents of the telegram (SMA Data).

2 Telegram Frame

Two different telegram frames are used:

Sunny Net can only transfer SMA Data telegrams as contents and it is not used any more in case of new developments.

SMA Net can transfer besides SMA Data telegrams many other kinds up to TCP/IP due to its multi-protocol capability. Thus, it is the preferred telegram frame in case of new developments.

As not every function is used and/or supported by each device, not every command specified in this document is inevitably implemented.

2.1 Sunny Net Telegram Frame

The Sunny Net telegram frame is intended only for the use together with the SMA Data Protocol.

- The **Telegram Frame** consists of a start character, twice the user data length, a 16 bit check sum and a stop character.
- The **Telegram Content** is composed of 7 bytes of protocol header (addressing, command) and a maximum of 255 bytes for the actual user data.

Optionally, a synchronization sequence of two consecutive AAH bytes is sent before every telegram. This sequence does not belong to the telegram content. It just serves as „training“ of the FSK (Frequency Shift Keying Demodulator).

		Sunny Net Telegram							
Powerline Synchroni-zation *(optional)*		**Frame**				Content SMA Data Telegram		**Frame**	
Sync	Sync	**Start**	**Length**	**Length**	**Start**	Protocol header	User data	**16 Bit CS**	**Stop**
1 Byte	1 Byte	**1 Byte**	**1 Byte**	**1 Byte**	**1 Byte**	7 Bytes	0..255 Bytes	**2 Bytes**	**1 Bytes**
AA	AA	**68**			**68**				**16**

Abbr.	**Name**	**Description**	**Value**
Start	Start byte	Telegram start character	68H
Length(2x)	Length byte	Number of user data bytes	0-255
Start	Start byte	Telegram start character	68H
Content	*Telegram content*	SMA Data Telegram consisiting of: 7 bytes of protocol header 0 up to a max. of 255 bytes of user data	
CS	Check sum	16 bit checksum The checksum is generated by adding byte by byte of the complete telegram content (protocol header and user data)	
Stop	Stop byte	Telegram stop character	16H

Start byte

In order to make a synchronization to the telegram start possible, all telegrams begin with the start character (68 hex). This character is always the first and the fourth character in a new telegram sequence.

Length byte

The length byte informs about the length of the user data included in the telegram. This information appears double as second and third character in a telegram.

Telegram content

Only SMA Data telegrams, consisting of a protocol header and user data, are acceptable as the content of a telegram.

Protocol header

The protocol header contains the addressing and the command.

User data

The content of user data of a telegram depends on the respective command. A maximum of 255 bytes can be transmitted per telegram.

Check sum

The check sum is a 16-bit value. It is generated by adding byte by byte of the telegram content (protocol header and user data) without signs.

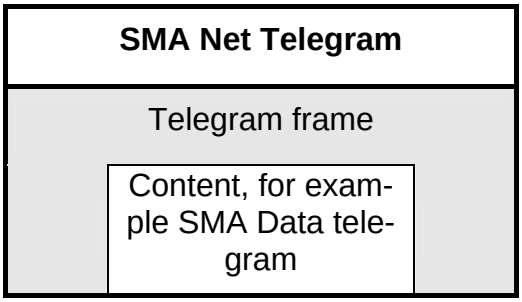
Stop character

In order to recognize a distinct end of the telegram, all telegrams will be completed with the stop character (16 hex).

2.2 SMA Net Telegram Frame

This chapter describes a frame format, an access procedure and protocol conventions suitable for the use on serial connections, including multipoint connections, such as RS485. The frame format complies with the Point-to-Point Protocol (PPP) of the TCP/IP protocol family. PPP itself is depicted in RFC (Request for Comments) 1661 [1], an appropriate frame format in RFC1662 [2]. On the one hand, this frame format complies with the international HDLC Standard (ISO 3309, [3]), that is used in many packet oriented networks (X.25, Datex-P).

In addition, this chapter describes the embedding of SMA Data telegrams into the SMA Net frame format.



SMA Data Telegram	
Protocol header	User data

Due to its multi-protocol capability, SMA Net can transfer many other telegrams up to TCP/IP besides SMA Data telegrams.

The embedding of SMA Data telegrams into the SMA Net frame format results in a number of advantages compared with the classical Sunny Net frame format.

- Compatibility with the Internet and HDLC Standard
- Option concerning the use of TCP/IP protocols without changing the frame format
- Integrated multi-protocol capability
- Safe data transfer due to 16-bit CRC
- Simple packet detection due to unique start flag
- A minimum overhead of only 8 bytes with user data of up to 1500 bytes
- Transparent transfer allows for XON/XOFF software handshake

2.2.1 PPP Frame Format

The frame format for PPP (cp. RFC 1662, [2]) has the following structure:

PPP Telegram						
Frame			Content		**Frame**	
Start	**Address**	**Control**	Protocol header	User data	**FCS**	**Stop**
1 byte	**1 byte**	**1 byte**	2 bytes	n bytes	**2 bytes**	**1 byte**
7E	**FF**	**03**				**7E**

Abbr.	Name	Description	Value
Start	Start byte	Start flag	0x7E
Address	Address byte		0xFF
Control	Control field		0x03
Content	*Telegram content*	Telegram consisting of: 2 bytes of protocol header n bytes of user data	s.u.
FCS	Frame Check Sequence	16 bit check sum The Frame Check Sequence is a 16-bit-CRC according to the polynomial $X^{16} + X^{12} + X^5 + 1$	
Stop	Stop byte	Stop flag	0x7E

This frame format orientates to the HDLC format in the following way:

- The start flag is identical under HDLC
- HDLC uses the values 3, 5, 7 and 15 as address
- The control field is a bit coded HDLC control field: 0x03 stands for unnumbered data blocks.
- The FCS field (Frame Check Sequence) and its calculation are identical under HDLC.
- The stop flag is identical under HDLC.

Protocol

From RFC1661 [1]: The structure of this field complies with the ISO 3309 Standard „Extension Mechanism for Address Fields“:

- All protocols must be oddly numbered. The least significant bit of the least significant octet must be „1“. In addition, all protocols must be defined in the way that the least significant bit of the most significant octet is „0“.
- Frames not complying with the rules above must be treated as unknown protocols.
- Protocol field values in the range of “0xxx“ up to “3xxx“ belong to the Network Layer Protocol of certain packets. Values in the range of “8xxx“ up to “3xxx“ belong to the appropriate Network Control Protocol (NCP).
- Protocol field values in the range of "4xxx" up to "7xxx" are used for protocols with small amounts of data without associated NCP.
- Protocol field values in the range of "Cxxx" up to "Fxxx" are used in order to apply Link Layer Control Protocols.

Protocol Entries

First the MSB (Most Significatn Byte) and then the LSB (Least Significant Byte) of the 2 byte protocol field is transferred.

Current values for the protocol field are respectively contained in the latest RFC version ([4]) entitled "Assigned Numbers".

2.2.2 The SMA Net Frame Format

SMA Net Telegram						
Frame			Content		**Frame**	
Start	**Address**	**Control**	Protocol header	SMA Data	**FCS**	**Stop**
1 byte	**1 byte**	**1 byte**	2 bytes	255 bytes	**2 bytes**	**1 byte**
7E	**FF**	**03**				**7E**

Abbr.	Name	Description	Value
Start	Start byte	Start flag / Start of a new telegram	0x7E
Address	Address byte	Broadcast – Address of the SMA Net frame	0xFF
Control	Control field		0x03
Content	*Telegram content*	Telegram consisting of: 2 bytes of protocol header 255 bytes of user data (SMA Data) ($7 \leq n \leq 262$)	see below
FCS	Frame Check Sequence	16 bit checksum The Frame Check Sequence is a 16-bit CRC according to the polynomial $X^{16} + X^{12} + X^5 + 1$	
Stop	Stop byte	Stop flag	0x7E

Start and Stop Flag

Every data frame starts and ends with the character 0x7E. All participants within the SMA Net continuously monitor the data stream in respect of this character and synchronize explicitly to the start of a new telegram.

In the case of consecutive frames only one character 0x7E is required. If two characters 0x7E occur anyway, this is regarded als (acceptable) empty packet.

Address

In the SMA Net the address 0xFF is the broadcast address and must be received by all participants in the SMA Net.

Control

The control field is a byte of the value 0x03. According to the HDLC Standard, it is an „Unnumbered Information (UI) command“, where the Poll/Final (P/F) Flag is set to 0.

Protocol

First the MSB (Most Significatn Byte) and then the LSB (Least Significant Byte) of the 2 byte protocol field is transferred.

General instructions for the protocol field according to RFC1661 are specified in the chapter concerning PPP Protocols. The following protocol identifications are defined within the framework of SMA Net.

- ID 1 4041 SMA Data Telegrams
- ID 2 4051 TCP / IP Supplementary Module
- ID 3 4043 Software Update System, SUSy

Frame Check Sequence (FCS)

The Frame Check Sequence is a 16-bit CRC according to the polynomial

- $X^{16} + x^{12} + x^5 + 1$

First the LSB (Least Significant Byte) of the FCS value of 2 bytes is transferred.

The FCS value is calculated by means of the fields address, control, protocol and data. The start and stop flag as well as the escape character inserted in the context of the transparancy, are not considered in the calculation. This also applies to the escape characters within the FCS value itself.

If characters identified as control characters in the ACCS (see below) are received, they are removed from the data stream before the FCS calculation.

Escape Character and ACCM

In order to achieve a transparent transfer where no start / stop flag appears in the data stream, the escape character 0x7D is used. For the use of the escape character the following rules apply:

- 0x7E is transferred as 0x7D / 0x5E
- 0x7D is transferred as 0x7D / 0x5D
- All characters with a bit set in the ACCM are transferred as 0x7D / 0xNN, where 0xNN is the exclusive OR from the original value and 0x20.

The Async Control Character Map (ACCM)

The ACCM is a 32 bit figure. The bits correspond to the ASCII control characters 0x00 (bit 0) up to 0x1F (bit 31).

- The default value of the ACCM in the SMA Net is 0x000E0000

In this default value the bits 17 (0x11, XON), 18 (0x12, DC2) and 19 (0x13, XOFF) are set. The control characters 0x11 and 0x13 used in the XON/XOFF procedure are transferred in this way as 0x7D / 0x31 and 0x7D / 0x33.

The default value is defined in a way that the fields address and control in the SMA Net are always transferred without escape character, to avoid an unnecessary extension of the frames.

SMA Net Sender

Firstly, the sender completely compiles the telegram to be transferred (including FCS) and then replaces all characters 0xNN between the start and stop flag by the couple 0x7D / (0xNN ^ 0x20) corresponding either to the flag character or the escape character or an ASCII control character with a ACCM bit set.

SMA Net Receiver

The receiver replaces all incoming characters having a preceding escape character (0x7D) with their 0x20 xor-tes XXX equivalent. In the special case of 0x7D / 0x7E the receiver cancels the reception and interprets 0x7E as the start of a new telegram.

In addition, the receiver checks the respective bit in the ACCM for all characters smaller than 0x20. If the bit is set, the character is removed from the data stream allowing that telegrams for example can be filled with appended zeros to a minimum length by the sender.

The FCS value for the data stream modified in this way is then calculated and evaluated.

Fast Frame Check Sequence (FCS) Implementation

The Frame Check Sequence was originally considered as a hardware implementation. While the serial bit stream goes beyond the line, the FCS is calculated and attached to the serial stream as complement. Finally, the stop flag follows.

The receiver has no chance to determine that the FCS of the received data stream is completed before it detects the stop flag. Thus, the FCS is defined in a way that a special bit pattern develops, when the FCS is calculated via the received complementary FCS. A valid frame is displayed by a „ good“ FCS value.

FCS Table Generator

The following source text generates a table for the calculation of the FCS:

```
/*
 * Generate a FCS-16 table.
 *
 * Drew D. Perkins at Carnegie Mellon University.
 *
 * Code liberally borrowed from Mohsen Banan and D. Hugh Redelmeier.
 */

/*
 * The FCS-16 generator polynomial: x**0 + x**5 + x**12 + x**16.
 */
#define P       0x8408


main()
{
    register unsigned int b, v;
    register int i;

    printf("typedef unsigned short u16;\n");
    printf("static u16 fcstab[256] = {");
    for (b = 0; ; )
    {
        if (b % 8 == 0)
            printf("\n");

        v = b;
        for (i = 8; i--; )
           v = v & 1 ? (v >> 1) ^ P : v >> 1;

        printf("\t0x%04x", v & 0xFFFF);
        if (++b == 256)
            break;
        printf(",");
    }
    printf("\n};\n");
}
```

FCS Calculation Method

The following source text provides a faster algorithm for the calculation of the FCS by means of the look-up table:

```
/*
    * u16 represents an unsigned 16-bit number. Adjust the typedef for
    * your hardware.
    */
   typedef unsigned short u16;

   /*
    * FCS lookup table as calculated by the table generator.
    */
   static u16 fcstab[256] = {
      0x0000, 0x1189, 0x2312, 0x329b, 0x4624, 0x57ad, 0x6536, 0x74bf,
      0x8c48, 0x9dc1, 0xaf5a, 0xbed3, 0xca6c, 0xdbe5, 0xe97e, 0xf8f7,
      0x1081, 0x0108, 0x3393, 0x221a, 0x56a5, 0x472c, 0x75b7, 0x643e,
      0x9cc9, 0x8d40, 0xbfdb, 0xae52, 0xdaed, 0xcb64, 0xf9ff, 0xe876,
      0x2102, 0x308b, 0x0210, 0x1399, 0x6726, 0x76af, 0x4434, 0x55bd,
      0xad4a, 0xbcc3, 0x8e58, 0x9fd1, 0xeb6e, 0xfae7, 0xc87c, 0xd9f5,
      0x3183, 0x200a, 0x1291, 0x0318, 0x77a7, 0x662e, 0x54b5, 0x453c,
      0xbdcb, 0xac42, 0x9ed9, 0x8f50, 0xfbef, 0xea66, 0xd8fd, 0xc974,
      0x4204, 0x538d, 0x6116, 0x709f, 0x0420, 0x15a9, 0x2732, 0x36bb,
      0xce4c, 0xdfc5, 0xed5e, 0xfcd7, 0x8868, 0x99e1, 0xab7a, 0xbaf3,
      0x5285, 0x430c, 0x7197, 0x601e, 0x14a1, 0x0528, 0x37b3, 0x263a,
      0xdecd, 0xcf44, 0xfddf, 0xec56, 0x98e9, 0x8960, 0xbbfb, 0xaa72,
      0x6306, 0x728f, 0x4014, 0x519d, 0x2522, 0x34ab, 0x0630, 0x17b9,
      0xef4e, 0xfec7, 0xcc5c, 0xddd5, 0xa96a, 0xb8e3, 0x8a78, 0x9bf1,
      0x7387, 0x620e, 0x5095, 0x411c, 0x35a3, 0x242a, 0x16b1, 0x0738,
      0xffcf, 0xee46, 0xdcdd, 0xcd54, 0xb9eb, 0xa862, 0x9af9, 0x8b70,
      0x8408, 0x9581, 0xa71a, 0xb693, 0xc22c, 0xd3a5, 0xe13e, 0xf0b7,
      0x0840, 0x19c9, 0x2b52, 0x3adb, 0x4e64, 0x5fed, 0x6d76, 0x7cff,
      0x9489, 0x8500, 0xb79b, 0xa612, 0xd2ad, 0xc324, 0xf1bf, 0xe036,
      0x18c1, 0x0948, 0x3bd3, 0x2a5a, 0x5ee5, 0x4f6c, 0x7df7, 0x6c7e,
      0xa50a, 0xb483, 0x8618, 0x9791, 0xe32e, 0xf2a7, 0xc03c, 0xd1b5,
      0x2942, 0x38cb, 0x0a50, 0x1bd9, 0x6f66, 0x7eef, 0x4c74, 0x5dfd,
      0xb58b, 0xa402, 0x9699, 0x8710, 0xf3af, 0xe226, 0xd0bd, 0xc134,
      0x39c3, 0x284a, 0x1ad1, 0x0b58, 0x7fe7, 0x6e6e, 0x5cf5, 0x4d7c,
      0xc60c, 0xd785, 0xe51e, 0xf497, 0x8028, 0x91a1, 0xa33a, 0xb2b3,
      0x4a44, 0x5bcd, 0x6956, 0x78df, 0x0c60, 0x1de9, 0x2f72, 0x3efb,
      0xd68d, 0xc704, 0xf59f, 0xe416, 0x90a9, 0x8120, 0xb3bb, 0xa232,
      0x5ac5, 0x4b4c, 0x79d7, 0x685e, 0x1ce1, 0x0d68, 0x3ff3, 0x2e7a,
      0xe70e, 0xf687, 0xc41c, 0xd595, 0xa12a, 0xb0a3, 0x8238, 0x93b1,
      0x6b46, 0x7acf, 0x4854, 0x59dd, 0x2d62, 0x3ceb, 0x0e70, 0x1ff9,
      0xf78f, 0xe606, 0xd49d, 0xc514, 0xb1ab, 0xa022, 0x92b9, 0x8330,
      0x7bc7, 0x6a4e, 0x58d5, 0x495c, 0x3de3, 0x2c6a, 0x1ef1, 0x0f78
   };

   #define PPPINITFCS16    0xffff  /* Initial FCS value    */
   #define PPPGOODFCS16    0xf0b8  /* Good final FCS value */

   /*
    * Calculate a new fcs given the current fcs and the new data.
    */
   u16 pppfcs16(fcs, cp, len)
       register u16 fcs;
       register unsigned char *cp;
       register int len;
   {
       ASSERT(sizeof (u16) == 2);
```

```
    ASSERT(((u16) -1) > 0);
    while (len--)
        fcs = (fcs >> 8) ^ fcstab[(fcs ^ *cp++) & 0xff];

    return (fcs);
}



/*
 * How to use the fcs
 */
tryfcs16(cp, len)
    register unsigned char *cp;
    register int len;
{
    u16 trialfcs;

    /* add on output */
    trialfcs = pppfcs16( PPPINITFCS16, cp, len );
    trialfcs ^= 0xffff;                /* complement */
    cp[len] = (trialfcs & 0x00ff);    /* least significant byte first */
    cp[len+1] = ((trialfcs >> 8) & 0x00ff);
    /* check on input */
    trialfcs = pppfcs16( PPPINITFCS16, cp, len + 2 );
    if ( trialfcs == PPPGOODFCS16 )
        printf("Good FCS\n");
}
```

3 Transfer Media and Time Response

3.1 RS485 Transfer

The data transfer in the SMA Net only requires the transmitter and receiver lines. A handshake via additional signal lines, such as „Request To Send“ (RTS), „Clear To Send“ (CTS), „Data Carrier Detect“ (DCD) or „Data Terminal Ready“ (DTR), is not provided.

The SMA Net frame format is designed for serial multipoint connections according to RS485, where all participants are considered to be equal. Thus, additional rules for the access to the media (Media Access Control) are required. These rules comply with the CSMA/CD procedure of the Ethernet.

3.1.1 Carrier Sense (CS)

Before any attempt to transmit an inquiry or notification telegram (bit 6 in Ctrl is 0) a participant within the SMA Net must have seen the RxD line for a time of 30 ms without any activity (i. e. with high level). At least 50 ms must elapse between receiving a response and transmitting the next inquiry.

A response telegram (bit 6 in Ctrl is 1) shall be transmitted after 25 ms at the earliest (the inquirer must disable its transmitter) and after 30 ms at the latest. Thus, the response to an inquiry always has priority over further inquiries (as the necessary rest period of 30 ms is not reached). A collision-free accomplishment of query/response cycles is possible as soon as the inquirer has attained token holding.

The procedure is selected deliberately that each polling participant always checks the RxD line in a fixed time pattern of 5 ms, this means, reads out MSR and checks for activity after 5 ms. If several participants want to transmit, they will normally have time-shifted pollings. In all probability, one

of the participants will find the line free early enough that the other participants already realize its transmitting activity and remain in the polling status. The winner of this round has then the disadvantage over the waiting participants, that it has to wait 40 ms.

The CS Polling is only effective, if there is no gap of more than 30 ms between two characters during the transmission of a telegram. During transmission the participants have to pay attention to this.

3.1.2 Multiple Acces (MA)

All stations having assured, according to the CS polling described above, that the line is free are allowed to start transmitting.

3.1.3 Collision Detection (CD)

Each participant must receive the characters sent by him and stop transmitting immediately in case of a failure (deviation between TX and RX data). The procedure shall be organized that on the one hand collisions can be detected as early as possible (immediately after every TX character) and on the other hand that the operation of serial modules with FIFO is supported.

In case of a collision, a CS polling occurs optionally with a rest time of 0 ms or 5 ms after n collisions (n = 1, 2, ..) in series.

- If bit n-1 of the own participant number is 0, choose 0 ms for the CS polling, otherwise 5 ms.
- After 16 collisions in series the transmission is regarded as definitely failed.

Only the collision partner compete for the transmission, because all rest periods on the line are under 30 ms.

periods on the line are under 30 ms.

3.2 Powerline Transmission

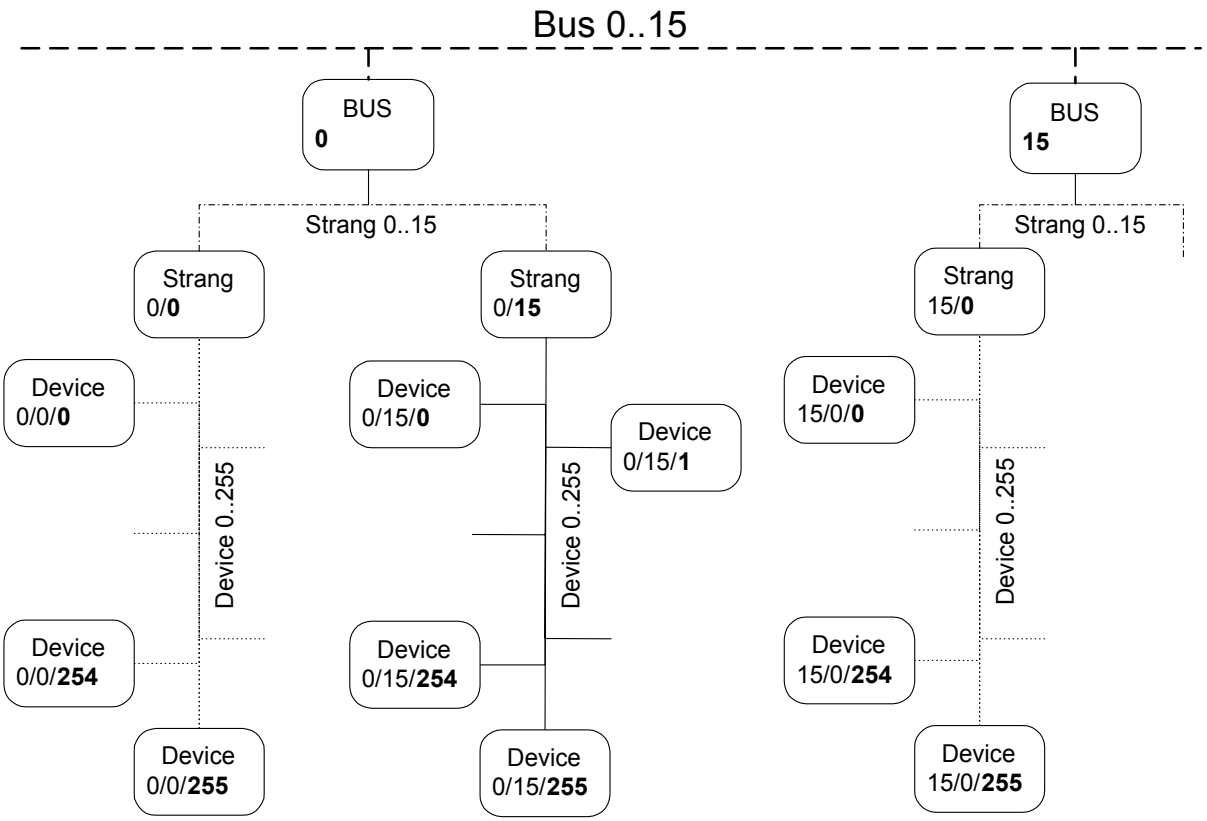
Within EN 50065-1 „Signalling on low voltage electrical installations in the frequency range 3 kHz to 148.5 kHz“ the powerline transmission is defined with the following fundamental conditions:

- It shall not be transmitted continuously for a time of more than 1 second.
- After every transmission a mandatory pause of 125 ms has to be made.
- A transmission shall not contain breaks of more than 80 ms without signal transmission.
- Each participant must identify occupancies of the transmission band. The frequency band is occupied, when a signal with a length of at least 4 ms exists.
- Each partcipant is allowed to transmit, if the band is detected to be unoccupied for a time period of 85 ms to 125 ms with at least seven possible times of inquiry.

4 SMA Data Transmission Protocol

The following transmission protocol is designed to provide for greatest possible flexibility and transmission reliability.

The protocol is based on the following network topology:



Each network participant has a unique network address composed of a bus address, a network address, a string address and a device address. 256 devices per logical string can be addressed by this address mode resulting in an address range of up to 4096 devices in case of a maximum of 16 strings. Contiguous plants (a maximum of 15) are separated by the bus address.

Another address mode is the group addressing. Up to 2^{16} = 65536 groups can be defined. Each network participant can be assigned to one or more group/s irrespective of string and bus. In order to allow for network-wide broadcast messages all bus participants are automatically assigned to the groupe 0.

Data is exchanged between the individual network participants via individual data telegrams containing telegram source address as well as telegram destination address.
This way of addressing allows that in principal each network participant can communicate with every other network participant. This means, that there is no strict master / slave communication.

The data is transmitted according to „Little Endian, low-Byte first“ (Intel format).

4.1 Telegram Format

The telegram content of a SMA Data telegram consists of two parts, the protocol header and the actual user data.

- The **protocol header** contains the telegram source and destination addresses, a control byte for packet handling, a packet counter and the respective command.
- The **user data** consist of the appropriate data of the respective command.

SMA Data Telegram Content					
Protocol header					**User data**
Source	**Destination**	**Ctrl**	**PktCnt**	**Cmd**	**Data**
2 bytes	**2 bytes**	**1 byte**	**1 byte**	**1 byte**	**max. of 255 bytes**

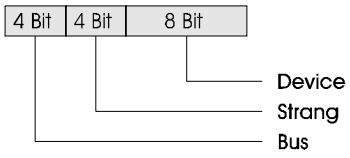
Abbr.	Name	Description	Value
Source	Source address	Telegram sender *Network address*	
Destination	Destination Address	Telegram receiver *Network address* or *group address*	
Ctrl	Control Byte	Bit 7 Address mode (0=Network address / 1=Group address) Bit 6 Acknowledge (0=Inquiry / 1=Response) Bit 4 Gateway blocking (0=not set / 1=set) Bit 0-3,5 Reserve (0)	
PktCnt	Packet counter	Number of telegram sequential packets	0-255H
Cmd	Command	The command of the telegram sender to the Destination address	0-255H
Data	User data	The content of a telegram's user data can be between 0 and 255 bytes depending on the command.	

Telegram Addresses (Source / Destination)

The source address of a telegram is always the network address of the telegram sender. The destination address of a telegram can either be the network address of a single receiver or the group address of several receivers. The destination address mode is stipulated via the bit 7 of the Ctrl bytes of a telegram.

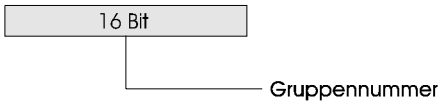
Network address (as source or destination address)

The network address is composed of the bus (4 bit), string (4 bit) and device addresses (8 bit):



Group address (only as destination address)

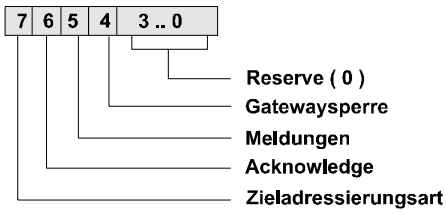
A group address consists of a 16 bit number.



Each device is automatically assigned to group 0.

Control Byte

The control byte is composed as follows:



- Bit 7: This bit determines the destination address. If it is set, the destination address is a group address, otherwise the destination address is a network address.

  0 = Network address
  1 = Group address

- Bit 6: 0 = Request telegram
  1 = Response telegram

- Bit 4: This bit determines if a data logger blocks the string or functions transparently.
  0 = Gateway blocking not set (telegram transmission)
  1 = Gateway blocking set (telegram transmission is blocked)

- Bit 0..3: Reserve (=0)

Packet Counter

This byte contains a value between 0 and 255. In case the amount of the user data is more than 255 bytes, several packets are necessary for the transmission. The counting of the packets starts with a value of n > 0. The packet counter is decremented per packet. The last packet contains the value 0. It can be transmitted with or without user data. If more than 255 packets are necessary, it will be decremented to one and then continued with 255.

Command Byte

The command byte contains the actual command to one or more network participants.

User Data

The content of a telegram's user data depends on the respective command. A maximum of 255 characters per telegram can be transmitted. But according to the command, the content of the user data can have the length of 0.

4.2 Communication Procedure

The following rules apply:

- Only one network participant transmits at a time, all the other participants behave passively and listen only.
- Basically, each telegram transmitted per network address with valid content of user data is acknowledged by the receiver irrespective of the telegram content with set acknowledge bit in the control byte of the telegram.

4.2.1 Data Request from a Destination Device (Single Telegram)

Request of source device:	acknowledge bit	= 0
	user data	= according to command
Response of destination device:	acknowledge bit	= 1
	packet counter	= 0
	user data	= according to command

4.2.2 Data Request from a Destination Device (Multiple Telegram)

Example 1:

600 bytes of user data shall be transmitted from the destination device to the source device. As a maximum of 255 characters can be transmitted per telegram, the transmission is carried out in three packets:

Packet no. 1:

Request of source device	acknowledge bit	= 0
	packet counter	= 0
Response of destination device	acknowledge bit	= 1
	packet counter	= 255
	user data	= 255 bytes

Packet no. 2:

Request of source device	acknowledge bit	= 0
	packet counter	= 255
Response of source device	acknowledge bit	= 1
	packet counter	= 254
	user data	= 255 bytes

Packet no. 3:

Request of source device	acknowledge bit	= 0
	packet counter	= 254
Response of source device	acknowledge bit	= 1
	packet counter	= 0
	user data	= 90 bytes

Example 2:

This mechanism also allows the source device to request faulty packets again:

Packet no. 1:

Request of source device	acknowledge bit	= 0
Response of destination device	acknowledge bit	= 1
	packet counter	= 255
	user data	= 255 bytes

Packet no. 2:

Request of source device	acknowledge bit	= 0
	packet counter	= 255

Faulty packet is recognized or Timeout !!!

Request of source device	acknowledge bit	= 0
	packet counter	= 255
Response of destination device	acknowledge bit	= 1
	packet counter	= 254
	user data	= 255 bytes

Packet no. 3:

Request of source device	acknowledge bit	= 0
	packet counter	= 254
Response of destination device	acknowledge bit	= 1
	packet counter	= 0
	user data	= 90 bytes

Example 3:

Data request from several destination devices per group address.
By group addresses several (multicast) or all (broadcast) of the participants can be reached.

In order to avoid bus collisions, the response to broadcast requests is always carried out after a transmission pause of 85 + random value (0-4765) ms.

Request of source device	acknowledge bit	= 0
	user data	= acc. to command
Transmission pause		
Response of group member 1	acknowledge bit	= 1
	user data	= acc. to command
Transmission pause		
Response of group member 2	acknowledge bit	= 1
	user data	= acc. to command
Transmission pause		
Response of group member 3	acknowledge bit	= 1
	user data	= acc. to command
Transmission pause		

4.3 Commands

The SMA Data commands serve for the registration / configuration of the system, the data acquisition, the binary communication and the transmission of variables.

A participant of SMA Data does not always support all commands, but only the appropriate commands for the according device type, respectively.

Overview of the SMA Data commands:

Command:	**Cmd:**	**Description:**	**Chapter**
Commands for the system configuration:			
CMD_GET_NET_START	06	Start of configuration requirements of netzwork	0
CMD_GET_NET	01	Request for network configuration	0
CMD_SEARCH_DEV	02	Find devices	0
CMD_CFG_NETADR	03	Assign network address	4.3.1.4
CMD_GET_CINFO	09	Request for channel information	4.3.1.5
CMD_SET_GRPADR	04	Assign group address (reserved)	
CMD_DEL_GRPADR	05	Delete group address (reserved)	
CMD_SET_MPARA	15	Parameterize data acquisition (reserved)	
CMD_TNR_VERIFY	50	Verify participant number (reserved)	
Commands for the data acquisition:			
CMD_SYN_ONLINE	10	Synchronize online data	4.3.2.1
CMD_GET_DATA	11	Data request	4.3.2.2
CMD_SET_DATA	12	Transmit data	4.3.2.3
CMD_GET_SINFO	13	Query data memory information	4.3.2.4
Commands for the configuration of the data storage:			
CMD_GET_MTIME	20	Read storage intervals	4.3.3.1
CMD_SET_MTIME	21	Set storage intervals	4.3.3.2
Commands for the binary transmission:			
CMD_GET_BINFO	30	Request binary range information	4.3.4.1
CMD_GET_BIN	31	Binary data request	4.3.4.2
CMD_SET_BIN	32	Send binary data	4.3.4.3

Command:	Cmd:	Description:	Chapter
Commands for variables:			
CMD_VAR_VALUE	51	Request system variables	4.3.5.1
CMD_VAR_FIND	52	Owner of a variable (reserved)	
CMD_VAR_STATUS_OUT	53	Allocation variable – channel (reserved)	
CMD_VAR_DEFINE_OUT	54	Allocate variable – channel (reserved)	
CMD_VAR_STATUS_IN	55	Allocation input parameter – status variable (reserved)	
CMD_VAR_DEFINE_IN	56	Allocate input parameter – variable (reserved)	
Other commands:			
CMD_PDELIMIT	40	Limitation of device power	4.3.6.1
CMD_TEAM_FUNCTION	60	Team function for PV inverters	4.3.6.2

Reserved commands should not be used for historical reasons and are not defined in this document.

4.3.1 Commands for System Configuration

These commands serve for the registration and configuration of a system. Each passive device within the network must be able to respond to these commands and react accordingly. Active devices, such as a central PC for the administration of the system, must support these commands in order to be able to detect a system properly.

4.3.1.1 Request for Start of Network Configuration (CMD_GET_NET_START)

This command serves for the starting of SMA Data network configuration. All devices receiving this command, reset their internal flag for „already notified“ and respond. The data content of the response packets corresponds to those of the command CMD_GET_NET. After receipt of the (new) network address by CMD_CFG_NETADR the internal flag of the devices for „already notified“ is set.

The command CMD_GET_NET_START is to be used only once at the beginning of the registration. For all following registration cycles the command CMD_GET_NET must be used, as otherwise devices already registered answer again.

Request:

Cmd:	06
Ctrl:	10000000 b
Data content:	-
Data length:	0 byte
Source address:	SMA Data network address
Targe address:	Group 0

Protocol header					User Data
Source	Desti-nation	Ctrl	PktCnt	Cmd	-
01 00	00 00	80	00	06	-

Response:

Cmd:	06
Ctrl:	01000000 b
Data Content:	Serial number of the devices (dword) Device type, eg. „WR700-07“ (8 char) if the type name contains less than 8 character, the difference has to be filled up with ASCII zeros
Data length:	12 bytes
Source address:	Network address of the device
Destination address:	SMA Data network address

Protocol header					User Data
Source	**Desti-nation**	**Ctrl**	**PktCnt**	**Cmd**	**Serial number, device type SN=9380933, Typ= “WR700-07“**
02 00	**01 00**	**40**	**00**	**06**	**45 24 8F 00, 57 52 37 30 30 2D 30 37**

4.3.1.2 Request for Network Configuration (CMD_GET_NET)

This command serves for the registration of the SMA Data network configuration. All participants of the SMA Data whose flag for ‚already notified' is not set, respond with their current network address, serial number and device type designation. In order to avoid bus collisions the response is carried out after a transmission pause of 85 + random value (0-4765) ms, respectively. After receipt of the (new) network address by CMD_CFG_NETADR the internal flag of the devices for „already notified" is set.

Request:

Cmd:	01
Ctrl:	10000000 b
Data content:	-
Data length:	0 byte
Source address:	SMA Data network address
Destination address:	Group 0

Protocol header					User Data
Source	Desti-nation	Ctrl	PktCnt	Cmd	-
01 00	00 00	80	00	01	-

Response:

Cmd:	01
Ctrl:	01000000 b
Data content:	Serial number of the device (dword) Device type, e.g. „WR700-07“ (8 char) if the type name contains less than 8 characters, the difference has to be filled up with ASCII zeros
Data length:	12 bytes
Source address:	Network address of the device
Destination address:	SMA Data network address

Protocol header					User Data
Quelle	Ziel	Ctrl	PktCnt	Cmd	Serial number, device type SN=9380933, type= “WR700-07“
02 00	01 00	40	00	01	45 24 8F 00, 57 52 37 30 30 2D 30 37

Process of registration cycle of a system without data logger:

Step:	Command:	Explanation:
1	CMD_GET_NET_START	Start of the SMA Data network configuration, devices reset the internal flag for „already notified“ and respond.
2	CMD_CFG_NET_ADR	All detected devices are assigned a network address. They set the internal flag for „already notified“.
3	CMD_GET_NET	Continuation of the SMA Data network configuration. Only devices with the internal flag for „already notified“ not set respond.
4	CMD_CFG_NET_ADR	All detected devices are assigned a network address. They set the internal flag for „already notified“.
5		Repeat steps 3 and 4 until all devices are registered.

Process of registration of a system with data loggers:

Step:	Command:	Explanation:
1	CMD_GET_NET_START with ‘Ctrl Bit’ ‘Gateway blocking set’	The following devices respond: a) Devices that can be reached directly b) All data loggers
2	CMD_CFG_NET_ADR	All detected devices are assigned a network address. They set the internal flag for „already notified“. Data loggers are assigned a string with the network address.
3	CMD_GET_NET with ‘Ctrl Bit‘ ‘Gateway blocking set’ and network address of the data logger	The data logger responds for each administered device in the following way: „Info administered device’ . „Info data logger” e. g.: S/N-SWR, „WR-850-01”, NetAdr-SBC, S/N-SBC, „SunBC-01”
4	CMD_CFG_NET_ADR	All devices administered by the data logger are assigned a network address (bus and string address according to data logger). They set the internal flag for „already notified“.
5	GET_CINFO with ‘Ctrl Bit‘ ‘Gateway blocking set’ and network address of the device administered by the data logger	The PC requests the channel list of the unknown devices. The data logger responds with the channel list for the device

4.3.1.3 Search Devices (CMD_SEARCH_DEV)

This command serves for the contact with a certain device via its serial number. The device responds with its current network address, serial number and type designation.

Request:

Cmd:	02
Ctrl:	10000000 b
Data content:	Serial number of the device (dword)
Data length:	4 bytes
Source address:	SMA Data network address
Destination address:	Group 0

Response:

Cmd:	02
Ctrl:	01000000 b
Data content:	Serial number of the device (dword) Device type, e.g. ‘WR700-7’ (8 char) if the type name contains less than 8 characters, the difference has to be filled up with ASCII zeros
Data length:	12 bytes
Source address:	Network address of the device
Destination address:	SMA Data network address

4.3.1.4 Assign Network Address (CMD_CFG_NETADR)

This command serves for the configuration of a certain device via its serial number. The device responds with its new network address and serial number. After receipt of this command, a response to CMD_GET_NET is suppressed by setting the internal flag for “already notified“. The flag for “already notified” is reset by another broadcast command, such as CMD_SYN_ONLINE or CMD_GET_NET_START reenabling the response to CMD_GET_NET. This command is necessary for the integration of new or exchanged devices with an unvalid or without an address into the SMA Data network.

Request:

Cmd:	03
Ctrl:	10000000 b
Data content:	Serial number of the device (dword)
	New network address for the device (word)
Data length:	6 bytes
Source address:	SMA Data network address
Destination address:	Group 0

Protocol header					User data
Source	Desti-nation	Ctrl	PktCnt	Cmd	Serial number, Network address SN=9380933, NetAdr=3
01 00	02 00	80	00	03	45 24 8F 00, 03 00

Response:

Cmd:	03
Ctrl:	01000000 b
Data content:	Serial number of the device
Data length:	4 bytes
Source Address:	New network address of the device
Destination Address:	SMA Data network address

Protocol header					User data
Source	Desti-nation	Ctrl	PktCnt	Cmd	Serial number SN=9380933
03 00	01 00	40	00	03	45 24 8F 00

4.3.1.5 Request of Channel Information (CMD_GET_CINFO)

The memory and parameter configuration of an SMA Data participant can be requested in order to reach the highest possible flexibility in view of memory channel extensions and changes.

The participant responses with a listing of his/her (memory) channel information, i. e. the description of all Inputs / Outputs and parameters is transmitted. Such a description divides into a type independent header and a type specific extension.

Request:

Cmd:	09
Ctrl:	00000000 b
Data content:	-
Data length:	0 byte
Source address:	SMA Data network address
Destination address:	Network address of the device

Protocol header					User data
Source	Destination	Ctrl	PktCnt	Cmd	-
01 00	02 00	00	00	09	-

Response:

Cmd:	09
Ctrl:	01000000 b
PktCnt	nnnn = (n-1) packets
Data content:	see table(s)
Data length:	acc. to the length of the channel list of the device

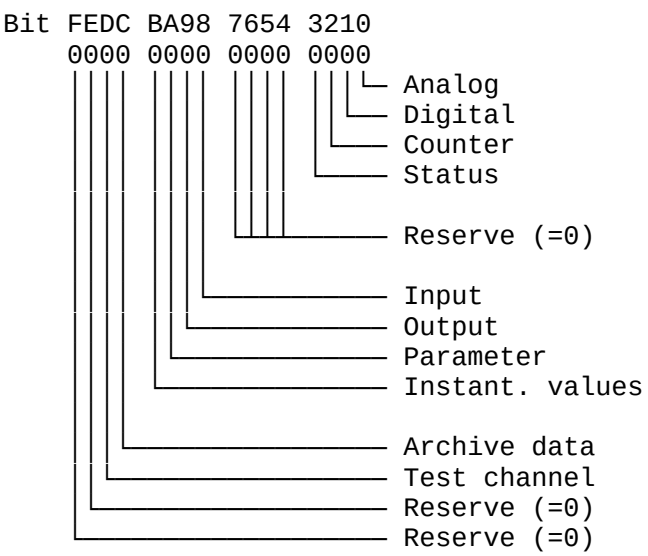
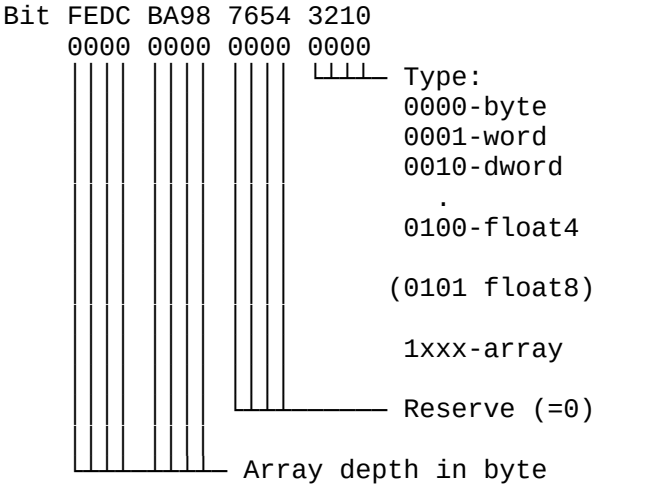
Source address: Network address of the device

Destination address: SMA Data network address

Protocol header					User data
Source	**Desti-ation**	**Ctrl**	**PktCnt**	**Cmd**	**Acc. to channel: Index, channel type, data format, level, name, unit, LoVal, HiVal, .. z.B.: 1, 01 04, 02 01, 02 00, 'SMA-SN ',' ', 0.0, 1.0E6**
02 00	**01 00**	**40**	**3B**	**09**	**01, 01 04, 02 01, 02 00, 53 4D 41 2D 53 4E 20 20 20 20 20 20 20 20 00, 20 20 20 20 20 20 20 00, 00 00 00 00, 00 24 74 49**

Structure of a channel description:

Type independent part:

Abb.	Name	Description	Type	Value
No.	Index	current channel number of this channel type	byte	1..255
cType	Channel type		word	see description
nType	Data format		word	
nFill			word	
Name	Channel name	Channel name (clear text)	char 16	
		Sum	**23**	**Byte**

Analog Sizes:

Abb.	Name	Description	Type	Value
Unit	Unit	Unit of the channels	char 8	
Gain	Gain	Gain of the channel (Parameter LoVal)	float 4	
Ofs	Offset	Offset of the channel (Parameter HiVal)	float 4	
		SUM	**16**	**Byte**

Digital Sizes:

Abb.	Name	Description	Type	Value
TxtLo	Lo-Text	Text Signal = 0	char 16	
TxtHi	Hi-Text	Text Signal <> 0	char 16	
		SUM	**32**	**Byte**

Counter Sizes:

Abb.	Name	Description	Type	Value
Unit	Unit	Unit of the channel	char 8	
Gain	Gain	Gain of the channel	float 4	
		SUM	**12**	**Byte**

Status Sizes:

Abb.	Name	Description	Type	Value
SizeT	Size of text field	Length of the following string lists	word	
StatT	Status texts	List of strings according to the maximum number of states. (the length of an individual entry is limited to a maximum of 16 characters)	dynamic	
		SUM	**2+SizeT**	**Byte**

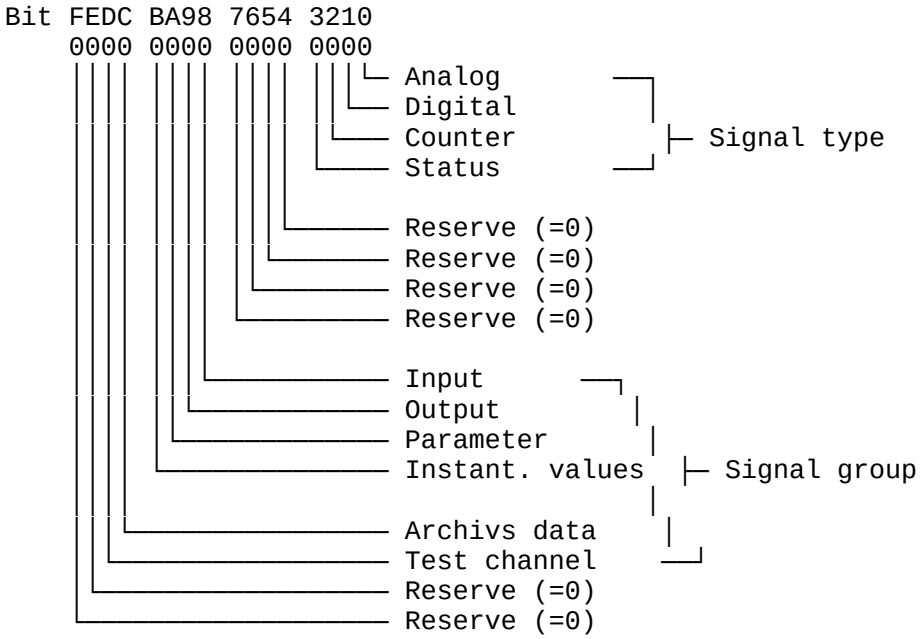
4.3.2 Commands for Data Acquisition

With the commands for Data Acquisition a device can request and/or set data. Data of an individual channel as well as data of a group of channels can be transmitted. The transfer mask (3 bytes) communicates the selection which channels shall be transmitted.

The transfer mask is to be considered as a filter. All channels complying with this filter are affected by the command to be effected.

Structure of the transfer mask

1. Mask for channel type (word)



2. Channel index (byte)

= 0 ⇒ all channels according to ‘Mask for channel type’

> 0 ⇒ only one channel (according to channel No.)

In principle, any bit combination in the mask for the channel type is acceptable, although this may be useless. Some reasonable combinations are the following:

Bit	FEDC	BA98	7654	3210		
	0000	1001	0000	1111	⇒	Request of instant. values Input
=	0	9	0	F	(Hex)	signals of all signal types
	0001	0001	0000	1111	⇒	Request of archivs data Input
=	1	1	0	F	(Hex)	singals of all signal types
	0000	0001	0000	0100	⇒	Request of current counter
=	0	1	0	4	(Hex)	counts
	0000	0100	0000	1111	⇒	Request of the parameters
=	0	4	0	F	(Hex)	of all signal types

If just one individual channel shall be addressed, the transfer mask is composed of the channel type and the channel index of the channel to be addressed.

Please note that during the transfer of the channel type first the low order part and then the high order part is transmitted.

4.3.2.1 Synchronization of Online Data (CMD_SYN_ONLINE)

Before each request cycle, a synchronization command with the updating time must be sent to all bus participants as broadcast message in order to guarantee a simultaneous acquisition of online data. All devices having received this command “freeze“ a copy of the current (instantaneous) memory values together with the transferred time. There is no feedback. After the transmission of the synchronization command, the online data request is effected at each device.

Request:

Cmd:	10
Ctrl:	10000000 b
Data content:	Local time in seconds since 1.1.1970-00:00:00 as Longint (without considering the summer / winter time and/or time zones).
Data length:	4 bytes
Source address:	SMA Data network address
Destination address:	Group x

Protocol header					User data
Source	Desti-nation	Ctrl	PktCnt	Cmd	Unix time format Zeit=843504044s = 0 x 32 46 09 AC
01 00	00 00	80	00	0A	AC D9 46 32

Response: none

4.3.2.2 Data Request (CMD_GET_DATA)

This command allows the data request for any request mask. This mask is identical with the channel type definition of the respective (memory) channels and is always composed of the channel type (2 bytes) and the channel index (1 byte).

The response is according to the channels that are selected in the mask. The structure of the data records to be transmitted arises according to the request mask:

Request:

Cmd:	11
Ctrl:	00000000 b
Data content:	Transfer mask (3 bytes)
Data length:	3 bytes
Source address:	SMA Data network address = 1
Dest. address:	Network address of the device = 2

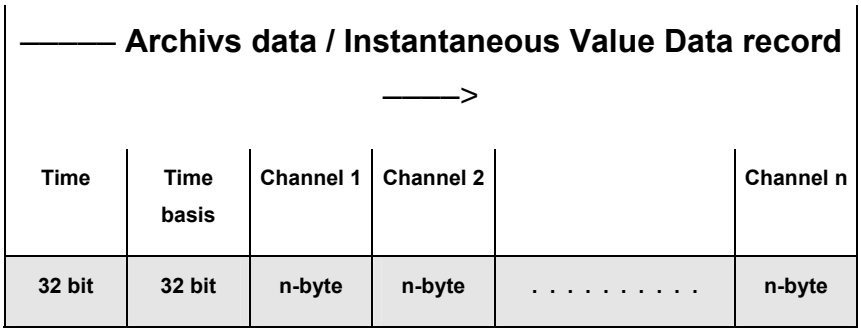
Protocol header					User data		
Source	Desti-nation	Ctrl	PktCnt	Cmd	Transfer mask Channel type=09 0F, Channel Idx=00	opt. Unix time 1 843517290	opt. Unix time 2 843603690
01 00	02 00	00	00	0B	0F 09, 00	6A 0D 47 32	EA 5E 48 32

Unix time 1 and Unix time 2 are only contained in case of archive data (see example below).

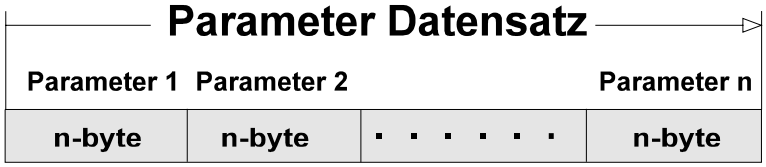
Request:

Cmd:	11
Ctrl:	01000000 b
Data content:	Transfer mask (3 bytes)
	Number of data records (word) (to be calculated)
	Data records (nn bytes) (to be calculated)
Data length:	3 bytes + nn bytes data records
Source address:	Network address of the device
Dest. address:	SMA Data network address

Data record format of request of archive data / instantaneous values



Data record format of request of parameters



Response packet of instantaneous value request

Protocol header					User data ...
					Byte number, decimal: 00 01 02
Quelle	**Ziel**	**Ctrl**	**PktCnt**	**Cmd**	**Transfer mask, ... Channel type=09 0F, channel Idx=00;**
02 00	**01 00**	**40**	**00**	**0B**	**0F 09, 00;**

...User data...		
03 04	05 06 07 08	09 10 11 12
Number of data records	**Sec since 1.1.70**	**Time basis**
1	**843517290**	**1s**
01 00	**6A 0D 47 32**	**01 00 00 00**

...User data...					
13 14	15 16	17 18	19 20	21 22	23 24
Analog data K1 Upv actual	**Analog data K2 Upv nominal**	**Analog data K3 Iac actual**	**Analog data K4 Iac nominal**	**Analog data K5 Uac**	**Analog data K6 Fac**
117	**196**	**3748**	**3**	**223**	**4983**
75 00	**C4 00**	**A4 0E**	**03 00**	**DF 00**	**77 13**

...User data...					
25 26	27 28	29 30	31 32	33 34	35 36
Analog data K7 Pac	**Analog data K8 Zac**	**Analog data K9 d-Zac**	**Analog data K10 R-Iso**	**Analog data K11 Uac-Srr**	**Analog data K12 Fac-Srr**
835	**37**	**2954**	**2954**	**221**	**4983**
43 03	**25 00**	**7C 13**	**8A 0B**	**DD 00**	**77 13**

...User data...					
37 38	39 40	41 42	43 44 45 46	47 48 49 50	51 52 53 54
Analog data K13 Zac-Srr	**Analog data K14 IZac**	**Analog data K15 TKK**	**Counter data K16 E-Total**	**Counter data K17 h-Total**	**Counter data K18 Net-In**
37	**4765**	**605**	**4361490**	**296068**	**75**
25 00	**9D 12**	**5D 02**	**12 8D 42 00**	**84 84 04 00**	**4B 00 00 00**

...User data			
55 56 57 58	59 60 61 62	63	64
Counter data K19 errorCnt	**Counter data K20 Snr**	**Status data K21 Status**	**Status data K22 error**
86	**9380933**	**= MPP**	**= ------**
56 00 00 00	**45 24 8F 00**	**07**	**00**

Archive data request (with CMD_GET_DATA)

GET_DATA allows for the request of data recordings with the bit ”Archive data“ set.

A time period can be added to the transfer mask of 3 bytes (with the bit „Archive data“ set) specifying the period in form of „from time“, „to time“. „Time“ is a Unix time and contains date and time. If no time period is specified (0 to 0), the maximum time period is assumed (all available archive data).

The data request is to be effected for each archive data channel separately.

The structure of the data records to be transmitted results according to the transfer mask:

Request of archive data:

Cmd:	11	
Ctrl:	00000000 b	
Data content:	Transfer mask	(3 bytes)
	Time period:	
	from time	(unsigned long)
	to time	(unsigned long)
Data length:	11 bytes	
Source address:	SMA Data network address	
Destination address:	Network address of the device	

Example for bit „Archive data“ set for channel type:

Protocol header					User data
Source	Desti-nation	Ctrl	PktCnt	Cmd	Transfer mask; Time period Channel type=01 19, Channel Idx=04; from time= 819936000, to time=843517290
01 00	02 00	00	00	0B	19 01, 04; 00 3B DF 30, 6A 0D 47 32

Response of archive data:

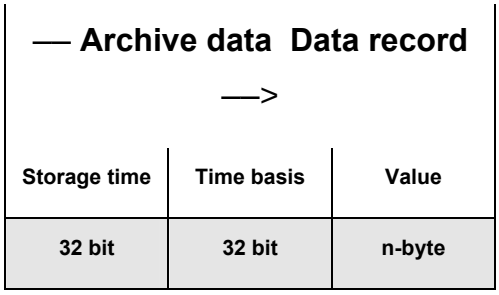
Cmd:	11
Ctrl:	01000000 b
Date content:	Transfer mask (3 bytes)
	Number of data records (word)
	Data records (nn bytes) (to be calculated)
Data length:	3 bytes
	+ 2 bytes number of data records
	+ (8+n) bytes per data record
Source address:	Network address of the device
Destination address:	SMA Data network address

Protocol header					User data ...
Source	Desti-nation	Ctrl	PktCnt	Cmd	Transfer mask, ... Channel type=01 19, Channel Idx=04;
02 00	01 00	40	00	0B	19 01, 04;

...User data...				
Number of data records	Storage time 1	Time basis 1	Value 1	Storage time 2
3	843517290	900s (15min)	114	843518190
02 00	6A 0D 47 32	84 03 00 00	72	EE 10 47 32

...User data				
Time basis 2	Value 2	Storage time 3	Time basis 3	Value 3
900s (15min)	121	843519090	900s (15min)	110
84 03 00 00	C4 00	27 14 47 32	84 03 00 00	6E

Structure of the response data records:



Storage time: Point of time of storage, in the Unix time format

Time basis: Storage interval in seconds (e. g. daily values = 24*60*60)

Value: Value of the channel depending on the channel type

4.3.2.3 Send data (CMD_SET_DATA)

The command allows for the data transmission to any SMA Data network participant. Parameters, counter contents, digital outputs and operating modes (status) can be set. The data content is composed according to the transfer mask. The receiver responses with an acknowledgement in case of correct transmission.

Request:

Cmd:	12
Ctrl:	00000000 b
Data content:	Transfer mask (3 bytes)
	Number of data records (word)
	Data record (to be calculated) (nn bytes)
Data length:	5 bytes + nn bytes
Source address:	SMA Data network address
Destination address:	Network address of the device

Set analog parameter channel 2 to 160 V

Protocol header					User data
Source	Destination	Ctrl	PktCnt	Cmd	Transfer mask Channel type=04 01, Channel Idx=02; Number of data records=01, Value=160
01 00	02 00	00	00	0C	01 04, 02; 01 00, A0 00

Response:

Cmd:	12
Ctrl:	01000000 b
Data content:	Transfer mask (3 bytes)
	Number of data records (word)
Data length:	5 bytes
Source address:	Network address of the device
Destination address:	SMA Data network address

Protocol header					User data
Source	**Desti-nation**	**Ctrl**	**PktCnt**	**Cmd**	**Transfer mask Channel type=04 01, Channel Idx=02; Number of data records=01**
02 00	**01 00**	**40**	**00**	**0C**	**01 04, 02; 01 00**

4.3.2.4 Request of Data Memory Information (CMD_GET_SINFO)

This command allows for the request of the memory information of possible archive data recordings. In addition to the recording interval, the response contains the recording range (from, to) for which memory data are available.

If memory data are available, they can be read out with the command CMD_GET_DATA and the archive data bit set.

Request:

Cmd:	13
Ctrl:	00000000 b
Data content:	Transfer mask (3 bytes)
Data length:	3 bytes
Source address:	SMA Data network address
Destination address:	Network address of the device

Response:

Cmd:	13
Ctrl:	01000000 b
Data content:	Transfer mask (3 bytes)
	Data records (12 bytes each)
Data length:	3 bytes transfer mask
	+ 12 bytes per data record
Source address:	Network address of the device
Destination address:	SMA Data network address

Structure of a response data record:

Name	Storage interval	fromDat - Unix	toDat - Unix
Byte	4	4	4
Example	60	01.01.2002-00:00	04.01.2002-00:00

The user data of the response consist of the transfer mask and the appropriate number of data records, which contain 12 bytes, each. Their number is calculated according to the transfer mask.

Storage interval

The storage interval is specified in seconds. Thus, the storage interval for hours archive data is 3600, for example.

4.3.3 Commands for the Configuration of the Data Storage

This set of commands allows for the specification of channels to be recorded by a device via the appropriate storage interval. Only channels with the flag „Archive data“ set (see structure of the transfer mask) are considered, other channels cannot be recorded.

4.3.3.1 Reading of Storage Intervals (CMD_GET_MTIME)

This command allows to read out the storage intervals presently set for the archive data channels. An individual channel or a group of channels can be selected via the transfer mask. The response contains a list of the storage intervals (in seconds) according to the transfer mask.

Request:

Cmd:	20	
Ctrl:	10000000 b	
Data content:	Transfer mask	(3 bytes)
Data length:	3 bytes	
Source address:	SMA Data network address	
Destination address:	Network address of the device	

The request contains the transfer mask of the channels, of which the storage intervals shall be read out.

Protocol header					User data
Source	Destination	Ctrl	PktCnt	Cmd	Transfer mask Channel type=19 01, Channel Idx=01;
01 00	02 00	00	00	14	04 21, 02

Response:

Cmd:	20	
Ctrl:	01000000 b	
Data content:	Transfer mask	(3 bytes)
	Data records (storage intervals)	(4 bytes, each)
Data length:	3 bytes transfer mask	
	+ 4 bytes per data record	
Source address:	Network address of the device	
Destination address:	SMA Data network address	

The user data of the response consist of the transfer mask (3 bytes) and the appropriate number of data records. Each data record contains the storage interval for a channel selected by the transfer mask.

Protocol header					User data
Source	Destination	Ctrl	PktCnt	Cmd	Transfer mask Channel type=19 01, Channel Idx=01; Storage interval (in seconds)
02 00	01 00	40	00	14	01 19, 01; 3C 00 00 00

4.3.3.2 Set Storage Intervals (CMD_SET_MTIME)

This command allwos to change the archive data channels presently selected for the data recording. If a channel is assigned, the storage interval 0, it is not recorded any more. The storage intervals are confirmed by the device. Inacceptable storage intervals are changed to acceptable values. If a device can only process a storage interval which is identical for all channels, the first valid storage interval is used. All other intervals (> 0) deviating from this are set to this storage interval and confirmed.

Request:

Cmd:	21	
Ctrl:	10000000 b	
Data content:	Transfer mask	(3 bytes)
	Data records (storage interval)	(4 bytes, each)
Data length:	3 bytes	
	+ 4 bytes per data record	
Source address:	SMA Data network address	
Destination address:	Network address of the device	

Protocol header					User data
Source	**Desti-nation**	**Ctrl**	**PktCnt**	**Cmd**	**Transfer mask Channel type=19 01, Channel Idx=01; Storage interval (in seconds)**
01 00	**02 00**	**00**	**00**	**15**	**01 19, 01; 3C 00 00 00**

Response:

Cmd:	21	
Ctrl:	01000000 b	
Data content:	Transfer mask	(3 bytes)
	Data records (storage interval)	(4 bytes, each)
Data length:	3 bytes transfer mask	
	+ 4 bytes per data record	
Source address:	Network address of the device	
Destination address:	SMA Data network address	

The response confirms the request, where improper storage intervals are corrected.

Protocol header					User data
Source	**Desti-nation**	**Ctrl**	**PktCnt**	**Cmd**	**Transfer mask Channel type =19 01, Channel Idx=01; Storage interval (in seconds)**
02 00	**01 00**	**40**	**00**	**15**	**01 19, 01; 3C 00 00 00**

4.3.4 Commands for Binary Transmission

This set of commands allows for the direct reading and/or setting of storage areas of an SMA Data participant. These commands are optional and are not supported by all of the devices.

4.3.4.1 Request of binary area information (CMD_GET_BINFO)

This command allows to query the storage areas and data files available for binary transmission.

The response packet contains a list of the storage areas and data files for which binary transmission is acceptable. If the commands CMD_GET_BIN or CMD_SET_BIN with inacceptable area information are used, they are refused by the target system.

Request:

Cmd:	30
Ctrl:	00000000 b
Data content:	-
Data length:	0 byte
Source address:	SMA Data network address
Destination address:	Network address of the device

Response:

Cmd: 30

Ctrl: 01000000 b

Data content:

List of the binary areas:

Binary area No.	(byte)
Counter of changes	(byte)
Area name	(16 bytes)
Start address	(unsigned long)
Size	(unsigned long)
Mode	(bytes)

Data length: 27 bytes per area

Source address: Network address of the device

Destination address: SMA Data network address

Area No.	Counter of changes	Area name	Start address	Size	Mode
1	0	Data RTC	0x0000 0000	0x0000 0080	0
2	0	Program-Flash	0x0004 0000	0x0004 0000	2
3	0	IO Addresses	0x0000 0000	0x0000 FFFF	1

Mode: 0 := only read

1 := only write

2 := read / write

If data files are transferred instead of physical storage areas, the start address is 0x0000 0000.

The size contained in the response to CMD_GET_BINFO indicates the maximum size of the file. In case of transfer with CMD_GET_BIN, this size is not necessarily reached.

The counter of the changes shows if the content of the data area has changed since the last file transfer. It is incremented with every change.

4.3.4.2 Binary Data Request (CMD_GET_BIN)

This command allows for the specific data query of individual storage areas and data files.

The request mask contains an index for the storage area to be addressed (Device: SRAM, Flash1,...), the start address of the storage area and the end address.

The data are requested as described; for the start address and the area length the following applies:

The start address contains the value where to start or to continue the transfer, when not all of the data could be sent with one packet, i. e. the start address of the second request results from the sum of the start address of the first request and the area length from the first response telegram. The area length from the previous response is always added to the new start address.

In case of the request, the maximum data record length that can be received is to be specified under area length without the bytes for the binary are no., start address and area length.

The response provides the actual length of the returned data reocord (without binary no., start address and area length). The start address contains the value of the request.

In case of sequential packets, the devices check the successive start address and cancel the transfer, when the successive start address does not correspond to the calculated start address.

In case the request under area length contains the value zero, this mechanism is not applied.

The structure of the data records to be transferred arises according to the area data of the request.

Request:

Cmd: 31
Ctrl: 00000000 b
Data content:
Area data:
Binary area no. (byte)
Start address (unsigned long)
Area length (unsigned short)
Data length: 7 bytes
Source address: SMA Data network address
Destination address: Network address of the device

1. Example:

Protocol header					User data
Source	Desti-nation	Ctrl	PktCnt	Cmd	Binary area no =01,Start address= 0000 0000, Area length=1000 (4096 bytes)
01 00	02 00	00	00	1F	01, 0000 0000, 00 10

Response:

Cmd: 31

Ctrl: 01000000 b

Data content:

Area data:

Binary area no.	(byte)
Start address	(unsigned long)
Area length	(unsigned short)
Data record	(Data content of the memory area)

Data length: 7 bytes + nn bytes data record

Source address: Network address of the device

Destination address: SMA Data network address

2. Example: Transfer of a text file

Format 1.Telegram:

Request:

- Protocol header:
  - Source : 01 00 (2 bytes)
  - Destination: 02 00 (2byte)
  - Ctrl: 00 (1byte)
  - PktCnt: 00 (1byte)
  - Cmd: 1F (1byte)
- Data content:
  Area data:
  - Binary area no: 03 (1byte)
  - Start address : 00 00 00 00 (4 bytes)
  - Area length: 00 04 (2 bytes)

Response:

- Protocol header:
  - Source : 02 00 (2 bytes)
  - Destination: 01 00 (2byte)
  - Ctrl: 40 (1byte)
  - PktCnt: FF (1byte)
  - Cmd: 1F (1byte)
- Data content:
  Area data:

- Binary area no: 03 (1byte)
- Start address : 00 00 00 00 (4 bytes)
- Area length: C8 00 (2 bytes)

Data record (Textfile example.) : 00 46 01 02 20 46 61 68

Format 2.Telegram:

Request:

- Protocol header:
  - Source : 01 00 (2 bytes)
  - Destination: 02 00 (2byte)
  - Ctrl: 00 (1byte)
  - PktCnt: FF (1byte)
  - Cmd: 1F (1byte)
- Data content:
  Area data:
  - Binary are no: 03 (1byte)
  - Start address : C8 00 00 00 (4 bytes)
  - Area length: 00 04 (2 bytes)

Response:

- Protocol header:
  - Source : 02 00 (2 bytes)
  - Destination: 01 00 (2byte)
  - Ctrl: 40 (1byte)
  - PktCnt: FE (1byte)

- Cmd: 1F (1byte)

- Data content:

  Area data:

  - Binary area no: 03 (1byte)
  - Start address : C8 00 00 00 (4 bytes)
  - Area length: C8 00 (2 bytes)

Data record (Textfile example.) : 00 46 01 03 20 46 61 68 ....

4.3.4.3 Send Binary Data (CMD_SET_BIN)

The command allows for the systematic writing of individual memory areas.

The request mask contains an index for the memory area to be addressed, the start address and end address of the area to be written.

The structure of the data records to be transferred arises according to the area date of the request:

Request:

Cmd:	32
Ctrl:	00000000 b
Data content:	
Area data:	
Binary area no.	(byte)
Start address	(unsigned long)
Area length	(unsigned short)
Data record	(Data content of the memory area)
Data length:	7 bytes + nn bytes data record
Source address:	SMA Data network address
Source address:	Network address of the device

Response:

Cmd:	32	
Ctrl:	01000000 b	
Data content:		
Area data:		
Binary area no.		(byte)
Start address		(unsigned long)
Area length		(unsigned short)
Data length:	7 bytes	
Source address:	Network address of the device	
destination address:	SMA Data network address	

4.3.5 Commands for Variables

Essentially, the commands contain variable number, contents of variables and memory channel numbers.

For the identification of the type and index of a channel, a channel number is used. The channel number is a four-digit number. The channel type is derived from the first digit:

1: Analog channel

2: Digital channel

3: Counter channel

4: Status channel

5: All digital channels, bit masked, a max. of 32 bit large

6-9: not defined.

The remaining digits form the index of the channel in the channel list (see tables of memory channels).

- Channel number 1011: analog channel with index no. 11
- Channel 5001 bit 3: digital input with index 4

4.3.5.1 Inquire System Variables (CMD_VAR_VALUE)

This command requests the content of one or several variables.

Several participants can respond to this command. Each component responds to its variables (output variables).

Request:

Cmd:	51
Ctrl:	10000000 b
Data content:	Number of variables: n ,1 up to a max. of 25 (2 bytes)
	Variable number X_1 ... X_n (n*2 bytes)
Source address:	Network address
Destination address:	Broadcast

Protocol header					User data
Source	Ad-dress	Ctrl	PktCnt	Cmd	Number , Variable number. X_1 up to X_n
03 00	00 00	80	00	33	XX XX XX XX ... XX XX

Response:

Cmd: 51

Ctrl: 11000000 b

Data content: Number of variables: m, 1 up to a max. of 25 (2 bytes)

Variable number X_1 (2 bytes) - Content W_1 (4 bytes)

...

Variable number X_m - Content W_m

(altogether: 2 bytes + m * 6 bytes)

Source address: network address

Destination address: Broadcast

Protocol header					User data
Source	Destination	Ctrl	PktCnt	Cmd	Number , variable no. X_1 , content W_1 .... variable no. X_n , content Wn
03 00	00 00	C0	00	33	XX XX XX XX xx xx xx xx ..... XX XX xx xx xx xx

Variables - Request

Variables: **21 01 (fan device 1)**

22 01 (fan device 2)

Request:

Protocol header					User data		
Source	Destination	Ctrl	PktCnt	Cmd	Number	Variable 1	Variable 2
03 00	00 00	80	00	33	02 00	01 21	01 22

Responses:

By device 1 (fan ON):

Protocol header					User data		
Source	Desti-nation	Ctrl	PktCnt	Cmd	Number	Variable	Value
01 00	03 00	C0	00	33	01 00	01 21	01 00 00 00

By device 2 (fan OUT):

Protocol header					User data		
Source	Desti-nation	Ctrl	PktCnt	Cmd	Number	Variable	Value
02 00	03 00	C0	00	33	01 00	01 22	00 00 00 00

4.3.6 Other Commands

4.3.6.1 Limitation of the Device Power (CMD_PDELIMIT)

In order to allow for a fast limitation of the devices a broadcast message can be sent to all bus participants. All the devices having received this command (and support the command) limit their output power.

Request:

Cmd:	40
Ctrl:	10000000 b
Data content:	Type of the limitation (1 bytes) 0: relative limitation (referring to the current value –100 ... +100 %) 1: absolute limitation (to limiting value 0 .. 100 %) Limiting value (in % of nominal value) (1 byte – signed)
Data length:	2 bytes
Source address:	SMA Data network address
Destination address:	Group x

Relative limitation of the current actual value by 5 % of the nominal value downward

Protocol header					User data
Source	**Desti-nation**	**Ctrl**	**PktCnt**	**Cmd**	**Type relative = 0 Limitation –5%**
01 00	**00 00**	**80**	**00**	**28**	**00 FB**

Response: None

4.3.6.2 Team Function for PV Inverters (CMD_TEAM_FUNCTION)

This command is reserved.

Request:

Cmd: 60

5 Literature

[1] Simpson, W., Editor, The Point-to-Point Protocol (PPP)", STD 50, RFC 1661, Daydreamer, July 1994.

[2] Simpson, W., Editor, "PPP in HDLC Framing", STD 51, RFC 1662, Daydreamer, July 1994.

[3] ISO/IEC 3309:1991(E), "Information Technology - Telecommunications and information exchange between systems - High-level data link control (HDLC) procedures - Framestructure", International Organization For Standardization, Fourth edition 1991-06-01.

[4] Postel, J. ASSIGNED NUMBERS, RFC 1700, , Daydreamer, October 1994.

[S] European Standards:
EN 50065-1: 1991 +
A1: 1992 + A2: 1995 +
A3: 1996 D