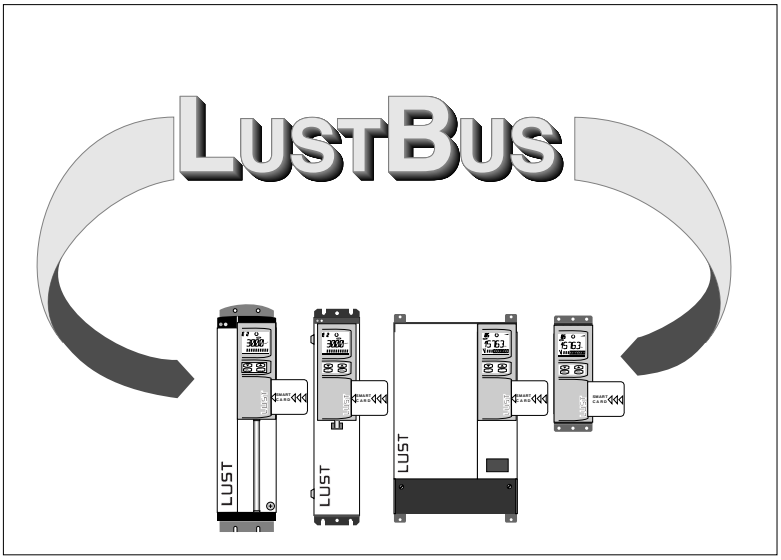
LUST
ANTRIEBSTECHNIK

Networking Inverters and Servocontrollers

on the LustBus

EN



Data Transmission Protocol

LUSTBUS Data Transmission Protocol

for The Inverter series
SMARTDRIVE VF1000 S/M/L

The servocontroller series
MASTERDRIVE MC6000/7000

Effective: November 1996

Id.No.: A040.22B.0-00

Table of Contents:

1 Introduction 6

Supplementary Documentation 6

System Requirements 6

Joint Operation of Inverters and Servocontrollers on the LUSTBUS 6

2 Telegram Structure, Syntax, and Creation 7

2.1 General 7

2.2 Telegram Frames 8

2.3 Description of the Parameter Data 8

2.3.1 SMARTDRIVE VF1000 Parameter Data Types 8

Notes on using single data types in C 8

2.3.2 MASTERCONTROL Servocontroller MC6000/7000 Parameter Data Types 10

Notes on using single data types in C 10

2.4 Displaying the Parameter Number 12

2.4.1 SMARTDRIVE VF1000 Frequency Inverter 13

2.4.2 MASTERCONTROL MC6000/7000 Servocontroller 13

3 Telegram Execution and Testing 14

Access limitations depend on the type of operation 14

4 Default Values 15

4.1 Stand-alone Operation 15

4.2 Network Operation 15

5 Starting the System 16

5.1 Stand-alone Operation 16

5.2 Network Operation 16

Network-monitoring watchdog 16

6 Parameterization 17

6.1 ENQUIRY-Telegram 17

6.1.1 Examples 18

6.1.1.1 SMARTDRIVE VF1000 Inverter 18

6.1.1.2 MasterControl MC6000/7000 Servocontroller 18

6.2 SELECT-Telegram 19

6.2.1 Examples 19

6.2.2 SMARTDRIVE VF1000 Inverter 19

6.2.3 MASTERCONTROL MC6000/7000 Servocontroller 20

7 Control Functions 21

7.1 SMARTDRIVE VF1000 Frequency Inverter 21

7.1.1 Example 21

7.2 MASTERCONTROL MC6000/7000 Servocontroller 21

7.2.1 Example 22

8 When a Device malfunctions 23

8.1 Acknowledging Device Errors 23

8.1.1 SMARTDRIVE VF1000 Inverter 23

8.1.2 MASTERDRIVE MC6000/7000 Servocontroller 23

9 When an Error Occurs 24

9.1 Notes on Starting Operation 24

9.2 SELECT-Telegram 24

10 Timing 26

10.1 Telegram Durations 27

10.1.1 Telegram Time for an ENQUIRY-Telegram 27

10.1.2 Telegram Time for a SELECT-Telegram 28

10.2 Chronological Order of the Telegrams 29

10.2.1 SMARTDRIVE VF1000 Frequency Inverter 29

10.2.2 MASTERCONTROL MC6000/7000 Servocontroller 29

11 Device Special Functions 30

11.1 Requesting the Inverter Parameter Description 30

Parameter name.....30
Unit 30
Parameter kind.....30
Parameter type.....31
Value area.....31

11.2 Requesting the Servocontroller-Parameter Description.....31

11.2.1 Detailed Description of the Individual Elements.....32
The parameter name 32
The parameter unit 33
The parameter kind 33
The parameter type 33
The parameter reference value.....33
The user level.....33
The attribute.....34
Hardware flags for filtering the parameter 34
The structure word for filtering the parameter.....35
Filtering the Subject Area parameter.....36
Activity Areas and the SMARTCARD 36

11.3 Loading the Parameter-Value Substitution Text 37

11.4 Download Handshaking for Parameter-data Records.....37

11.5 Changing the Baudrate Online.....38

11.6 Communicating with the POSMOD1 (MC6000 only).....38

11.6.1 SELECT-Telegram 39
11.6.2 ENQUIRY-Telegram.....39
11.6.3 User Data Structure.....40
11.6.4 Example telegrams.....40
Passing a manual command to POSMOD1 40
Transmitting a program to POSMOD1 41
Status request from POSMOD1.....41
Program request from POSMOD1 42

1 Introduction

This LustBus documentation is valid for all Inverters of the series SMARTDRIVE VF1000 and all servocontrollers of the MASTERCONTROL MC6000/7000 series. The telegram structure of both device series is very similar, and for this reason both are documented here together.

Supplementary Documentation

- parameter description of each drive unit

System Requirements

All systems equipped with a RS485 interface are suitable. There are no particular requirements regarding processor speed because a timeout watchdog has not been installed in the devices. This is the factory setting.

Joint Operation of Inverters and Servocontrollers on the LUSTBUS

Data is exchanged with the Inverters over the standard LUSTBUS data-transmission protocol. Data exchange with the servocontrollers takes place using the enhanced LUSTBUS data-transmission protocol which is downward compatible with the standard LUSTBUS protocol.

Both device series can be operated in a network, since the most important parameters for network control are numerically compatible (address, transmission rate, device type).

When the term **"device"** is used in this documentation, it applies to both Inverters of the **SMARTDRIVE** series as well as the **MASTERCONTROL** servocontroller series.

Note: The LUSTBUS interface is implemented in each device at a relatively high user level. This allows use of parameters that are inaccessible to users with the KEYPAD (KP100). Some of the parameters in these user levels are service parameters and are not documented in the standard user manual. Unintentional writes to such parameters could severely interfere with the function of the device!

2 Telegram Structure, Syntax, and Creation

2.1 General

Data is transmitted in the LUSTBUS data-transmission protocol in the following format:

- 7 data bits
- even-numbered parity
- 1 stop bit

The transmission protocol is based on the ISO standard 1745 (Fast Select). Two types of calls are used:

- Request to send (ENQUIRY-Telegram) – for requesting parameter data in the inverter/servo-controller.
- Request to position (SELECT-Telegram) – for passing parameter data to the inverter/servo-controller.

Bus activities are controlled exclusively via the master (e.g. PC). Exchanging data directly between two slaves (Inverter/servocontroller) is thus not possible. The master can communicate either with only one device via its special address, or with all devices, using address 31 (broadcast message).

Note: This user manual describes the data-transmission protocol exclusively. Information concerning the serial interface, wiring, and accessories can be found in the operating manual of the drive system.

To simplify the syntax explanantion and the examples, the individual telegram components are indicated with identifiers.

The telegram components are divided into:

- *Special / Control characters* for the telegram frames
- *VALUE-String* for displaying parameter data
- *CODE-String* for displaying the parameter numbers

The individual telegram components are described in detail in the following chapters.

2.2 Telegram Frames

The following table contains all components with the corresponding expressions:

Special / Control Characters	Hexadecimal Value	Decimal Value	Explanation
ADR	40 … 5F	64 … 95	Device address (address 0 = ´@´)
STX	02	2	Start of text
ETX	03	3	End of text
EOT	04	4	End of transmission
ENQ	05	5	Enquiry
ACK	06	6	Acknowledge
NAK	15	15	Negative acknowledge
=	3D	61	Equal sign for separating parameter numbers from the parameter values
BCC	0 … 7F	0 … 127	Checksum byte All bytes between STX (exclusive) and ETX (inclusive) are EXOR'ed

2.3 Description of the Parameter Data

The transmitted values (VALUE) are displayed in hexadecimal form. The digits A-F are valid only as capital letters.

The parameter values for the SMARTDRIVE Inverters always have a length of exactly four bytes. Leading zeros are transmitted.

This restriction was dropped for the MASTERDRIVE servocontrollers due to their expanded functionality. The length of the parameter value (VALUE) can be between 1 and 100 characters depending on the data type. Leading zeros do not have to be transmitted.

Depending on the parameter data types, the interpretation of the data passed in the *VALUE-String* can vary. A summary of all device parameters, with their respective data types, is available in a separate document from LUST Antriebstechnik GmbH.

2.3.1 SMARTDRIVE VF1000 Parameter Data Types

The SMARTDRIVE Inverter supports the following parameter data formats:

Data type	Scale	Range of values	Representation KP100	MC6000/7000-compatible
PT_BINARY8	1	0 ... 255	00H ... FFH	range of values PT_USIGN8
PT_FIXPOINT16	0.05	0.00 ... 3276,80	0.00 ... 999.95	yes
PT_USIGN16	1	0 ... 65535	0 ... 65535	yes
PT_USIGN8	1	0 ... 255	0 ... 255	yes
PT_VF_ERROR	1	0 ... 65535	plain text e.g. "E-OV"	no
PT_PASSWORD	1	0 ... 65535	0 ... 65535	yes

Notes on using single data types in C

Data type PT_BINARY8, PT_PASSWORD, PT_USIGN16 and PT_USIGN8:

The data type PT_BINARY8 is shown on the display of the KEYPAD KP100 in hexadecimal form, since it is usually used to represent bit fields. In the following example, it is mapped to an unsigned 16-bit integer because there is no format string with hexadecimal conversion for 8-bit integer values in C.

Passwords are represented as unsigned 16-bit integer values. The following conversion functions can thus be used for these data types: PT_BINARY8, PT_PASSWORD, PT_USIGN16, and PT_USIGN8.

Conversion examples in C:

- from *VALUE-String* to C data types:

```
extern char *VALUE;       /* The already received VALUE-String */
unsigned int Value;       /* The variable to receive the parameter value */
sscanf( VALUE, "%X", &Value);
```

- from C data types to *VALUE-String*:

```
char VALUE[5];            /* Storage for the Send-VALUE-String */
extern unsigned int Value;/* The variable with the new parameter value */
sprintf( VALUE, "%04X", Value); /* Always expand to 4 characters */
```

Data type PT_FIXPOINT16:

By scaling a "normal" unsigned 16 bit integer to 0.05 you obtain this data type. The KP100 has a range of values from 0.00 to 999.95.

Conversion examples in C:

- from *VALUE-String* to C data types:

```
extern char *VALUE;       /* The already received VALUE-String */
float Value;              /* The variable to receive the parameter value */
unsigned int uValue;      /* Temporary help variable for converting */
sscanf( VALUE, "%X", &uValue);
Value = (float)uValue / 20.;
```

- from C data types to *VALUE String*:

```
char VALUE[ 5];           /* Storage for the Send-VALUE-String */
extern float Value;       /* The variable with the new parameter value */
sprintf( VALUE, "%04X", (unsigned int)( Value * 20.));
```

Data type PT_VF_ERROR:

This data type corresponds to a structured 16-bit integer. The high byte indentifies the error number (0-15) and the low byte the time of error (0-15h), i.e., the value of the time-in-operation hour counter when the error occured.

Structure element	Data width	Range of values	Unit
error number	1 Byte	0 – 15	none
time of error	1 Byte	0 – 15	1/10 h

Note: Error number 15 means "no error" because erasing the EEPROM results in the value FFH in all memory locations.

Conversion examples in C:

- from *VALUE-String* to C data types:

```
struct VF1000_ErrorStruc
{
 unsigned char ErrorCode;
 unsigned char ErrorTime;
};

union ErrorIntMix
{
 struct VF1000_ErrorStruc ErrStruc16;
 unsigned int u16;
};
```

```
extern char *VALUE;       /* The already received VALUE-String */
union ErrorIntMix Value;  /* The variable to receive the parameter value */
sscanf( VALUE, "%X", &Value.u16);
/* accessing the individual structure elements */
... = Value.ErrStruc16.ErrorCode;
... = Value.ErrStruc16.ErrorTime;
```

- from C data types to *VALUE-string*:

```
char VALUE[5];                  /* Storage for the Send-VALUE-String */
extern union ErrorIntMix Value;  /* The variable with the new parameter value */
sprintf( VALUE, "%04X", Value.u16);    /* Always expand to 4 characters */
```

2.3.2 MASTERCONTROL Servocontroller MC6000/7000 Parameter Data Types

The servocontrollers MASTERCONTROL support the following parameter data formats:

Data type	Scale/ Increment	Range of values	Representation KP100	VF1000-compatible
PT_USIGN8	1	0 ... 255	0 ... 255	Value range PT_BINARY8
PT_USIGN16	1	0 ... 65535	0 ... 65535	yes
PT_FIXPOINT16	0.05	0.00 ... 3276,80	0.00 ... 999.95	yes
PT_INTEGER16	1	-32768 ... 32767	-9999 ... 32767	no
PT_INT32Q16	1/65536	-32767.99 ... 32766.99	-32.76E3 ... 32.76E3	no
PT_FLOAT_IEEE	see IEEE	see IEEE	-99.99E9 ... 99.99E9	no
PT_MC_ERROR	1	0 ... 65535	plain text, e.g. "E-OV"	no
PT_PASSWORD	1	0 ... 65535	0 ... 65535	yes

Notes on using single data types in C

Data type PT_USIGN8, PT_USIGN16 and PT_PASSWORD:

In the following example, the data type PT_USIGN8 is mapped as an unsigned 16-bit integer because there is no C-format string for converting hexadecimals from 8-bit integer values in C.

Passwords are represented as unsigned 16-bit integer values. The following conversion functions can thus be used for the data types: PT_UISGN8, PT_USIGN16, and PT_PASSWORD.

Conversion examples in C:

- from *VALUE-String* to C data types:

```
extern char *VALUE;       /* The already received VALUE-String */
unsigned int Value; /* The variable to receive the parameter value */
sscanf( VALUE, "%X", &Value);
```

- from C data types to *VALUE-String*:

```
char VALUE[5];            /* Storage for the Send-VALUE-String */
extern unsigned int Value;/* The variable with the new parameter value */
sprintf( VALUE, "%X", Value);
```

Data type PT_FIXPOINT16:

By scaling a "normal" unsigned 16-bit integer to 0.05, you obtain this data type. The KP100 has a range of values from 0.00 to 999.95.

Conversion examples in C:

- from *VALUE-String* to C data types:

```
extern char *VALUE;       /* The already received VALUE-String */
float Value;              /* The variable to receive the parameter value */
unsigned int uValue;      /* Temporary help variable for converting */
sscanf( VALUE, "%X", &uValue);
Value = (float)uValue / 20.;
```

- from C data types to *VALUE-String*:

```
char VALUE[ 5];           /* Storage for the Send- VALUE-String */
extern float Value;       /* The variable with the new parameter value */
sprintf( VALUE, "%X", (unsigned int)( Value * 20));
```

Data type PT_INTEGER16:

This data type corresponds to a "signed int" in C.

Conversion examples in C:

- from *VALUE-String* to C data types:

```
extern char *VALUE;       /* The already received VALUE-String */
signed int Value;         /* The variable to receive the parameter value */
sscanf( VALUE, "%X", &Value);
```

- from C data types to *VALUE-String*:

```
char VALUE[5];            /* Storage for the Send- VALUE-String */
extern signed int Value;  /* The variable with the new parameter value */
sprintf( VALUE, "%X", Value);
```

Data type PT_INT32Q16:

This data type is a "signed long" (32-bit) with the scaling of 1/65536. It is represented on the KP100 the same way as an PT_FLOAT_IEEE.

Conversion examples in C:

- from *VALUE-String* to C data types:

```
extern char *VALUE;       /* The already received VALUE-String */
signed long lValue;       /* Temporary help variable for converting */
double dValue;            /* Variable to receive the parameter value */
sscanf( VALUE, "%lX", &lValue);
dValue = (double)lValue / 65536.;
```

- from C data types to *VALUE-String*:

```
char VALUE[9];            /* Storage for the Send- VALUE-String */
extern double Value;      /* The variable with the new parameter value */
sprintf( VALUE, "%lX", (unsigned long)(Value * 65536.));
```

Data type PT_FLOAT_IEEE:

This data type corresponds to a 32-bit, floating-point number in IEEE format.

Conversion examples in C:

- from *VALUE-String* to C data types:

```
union MC_FLOAT
{
  float f32;
  unsigned long u32;
```

```
}
extern char *VALUE;       /* The already received VALUE-String */
union MC_FLOAT unValue;   /* Temporary help variable */
float Value;              /* Variable to receive the parameter value */
sscanf( VALUE, "%lX", &unValue.u32);
Value = unValue.f32;      /* Changing without forcing a conversion */
```

- from C data types to *VALUE-String*:

```
char VALUE[9];            /* Storage for the Send- VALUE-String */
extern float Value;       /* The variable with the new parameter value */
union MC_FLOAT unValue;   /* Temporary help variable */
unValue.f32 = Value;
sprintf( VALUE, "%lX", unValue.u32);
```

Data type PT_MC_ERROR:

This data type is structured and is 32-bits wide. Its elements are:

Structure	Data width
Error number	1 Byte
Error location	1 Byte
Error time	2 Byte

Conversion examples in C:

- from *VALUE-String* to C data types:

```
struct MC_ErrorStruc       /* The MasterControl error structure */
{
  unsigned char EType;     /* Whas type of error */
  unsigned char ELocation; /* Module which registered the error */
  unsigned int ETime;     /* Error time of the time-in-opertion hour counter */
};

union ErrorLongMix        /* Union for Data converting */
{
  unsigned long u32;
  struct MC_ErrorStruc ErrStruc32;
};

extern char *VALUE;        /* The already received VALUE-String */
struct ErrorLongMix Value; /* The variable to receive the parameter value */
sscanf( VALUE, "%lX", &Value.u32);
/* Accessing the individual structure elements */
..... = Value.ErrStruc32.EType;
..... = Value.ErrStruc32.ELocation;
..... = Value.ErrStruc32.ETime;
```

- from C data types to *VALUE-String*:

```
char VALUE[9];                    /* Storage for the Send- VALUE-String */
extern union ErrorLongMix Value;  /* The vari. with the new parameter value */
sprintf( VALUE, "%lX", Value.u32);
```

2.4 Displaying the Parameter Number

The parameter number (CODE) is represented as a five-digit-wide decimal number. The coding of the five numbers has different meaning for Inverters and servocontrollers. The device-dependent description follows.

2.4.1 SMARTDRIVE VF1000 Frequency Inverter

The CODE field contains the five-digit-wide parameter address with the following meaning:

CODE =	X	X	X	X	X
	1	2	3	4	5

1 Specificator: With a request or return of a value, this position always equals null. If the parameter list of the selected parameter is requested, then Specificator = 1 must be set. Thus, Specificator = 1 means that the parameter description (parameter list) will be given – not the value of the corresponding parameter. The content of the parameter list will be explained later.

2 Data Set No.: For VF1000, always 0

3 Devices No.: For VF1000, always 1

4,5 Parameter No.: parameter number (00 ... 99, 2-digit)

The five digits of the CODE are given as five ASCII characters. The highest-valued digit is first.

Example of generating a parameter number in C:

```
/* Example for parameter 1-MODE */
char CODE[ 6];
sprintf( CODE, "000%02u", 1);
```

2.4.2 MASTERCONTROL MC6000/7000 Servocontroller

The *CODE-String* is a five-digit, ASCII-decimal number with the following coding:

CODE =	X	X	X	X	X
	1	2	3	4	5

1 Specificator: For MASTERCONTROL, the specificator is set so:

- 2 for parameter manipulation and parameter value request
- 3 for receipt of a parameter description
- 6 for special functions of the MASTERCONTROL

2 Data Set No.: For MASTERCONTROL, always 0

3,4,5 Parameter No.: parameter number (0 – 999, 3-digit)

The five digits of the CODE are given as five ASCII characters. The highest-valued digit is first.

Example of generating a parameter number in C:

```
/* Example for parameter 1-MODE */
char CODE[ 6];
sprintf( CODE, "20%03u", 1);
```

3 Telegram Execution and Testing

The serial interface of the devices process the telegram strictly as a state-machine without timeout monitoring. Via the control character EOT, this state-machine is set again to the starting value and it can be given a new telegram. Telegram fragments that have already been given to the device can be reset by sending the EOT. For this reason, the Host computer sends the EOT character before and after each data sequence.

The devices, and the Host computer, test the telegram syntax, the slave-address, and the text portion, giving a receipt for the telegrams accordingly.

Enquiry-telegram:

No Answer:	occurs with improper telegram construction; incorrect received control characters; or incorrect slave-address ADR
ADR NAK:	occurs with improper datacode CODE; or by read protected data
ADR STX text:	occurs when the telegram is valid

Select-telegram:

No answer:	occurs with improper telegram construction; incorrect received control characters; or incorrect slave-address ADR
ADR NAK:	occurs with wrong BCC; improper CODE; access disallowed; or improper Value
ADR ACK:	occurs when the telegram is valid

Generally, ACK is sent first after the new parameter value has been successfully entered in the device. Since new, RAM-parameter values can be taken directly, this parameter group causes no delay of the ACK-answer telegram.

For EEPROM parameters, SMARTDRIVE Inverters delay the answer telegram until the parameter value is stored without error in EEPROM.

MASTERCONTROL servocontrollers have a RAM memory location for each parameter and, for EEPROM parameters, an additional EEPROM memory location. After receiving a SELECT-telegram requesting an EEPROM parameter, the value is first placed in RAM and then ACK is sent immediately to the Host computer – thus the new parameter value is available immediately.

Saving in EEPROM is done in the background and does not delay the main program. To save a byte in EEPROM, the servocontroller needs 15 ms because every byte is saved three times. The EEPROM-write routine can become overloaded, if many SELECT-telegrams requesting EEPROM parameters follow each other in series. The servocontroller flags this condition by setting the corresponding bit in the SERR parameter. In this case, the corresponding SELECT-telegram must be repeated until the new parameter value is accepted by the servocontroller, which it indicates by ACK.

Access limitations depend on the type of operation

If a SELECT-telegram requesting a parameter is answered with a NAK and simultaneously bit 6 (hexadecimal 40H) of the serial-interface error-status word SIOF/SERR is set, then the system has refused write access to this parameter, irrespective of the given value.

This does not mean, however, that the writability of parameters is thus generally denied. Under certain conditions, write access is refused depending only on the actual, current operational state of the device. For example, the EEPROM parameters of SMARTDRIVE Inverters are not generally writable when control is active.

To determine which parameters are writable and when, please consult the device's reference material.

4 Default Values

In principle, the Baudrates for all devices must be set manually (e.g., via the KEYPAD), if the transfer rate in use does not match the factory setting of 9600 Baud. The transfer rates can be set in steps ranging from 2400 Baud to 9600 Baud for frequency Inverters, and from 2400 Baud to 57600 Baud for Servocontrollers. The configuration is set via parameter 81-SIOC/SBAUD.

Attention: SMARTDRIVE Inverters with the RS-485 interface must additionally have the parameter 61-SOUTA = 7 set. This activates the data direction switch of the internal interface software driver.

4.1 Stand-alone Operation

Operating a single device on the LUSTBUS requires no address presettings. Communication with the device can be done via address 0. It will respond, if the Baudrate and the data format (data bit, parity, etc.) are set correctly on the Host computer.

4.2 Network Operation

Since the special device address SIOA/SADDR of all devices is set at the factory to the same value, the addresses of the servocontrollers must be set individually before connecting them to the LUSTBUS. (This is done via the KP100 or the LUSTBUS in stand-alone operation.) It is important to make sure that no address is assigned twice and that address 0 is not used. The total address space is:

address 0 (@)	Every device answers on this address, independent of the value its special address has.
address 1 ... 30 (A ... ^)	Special address of **one** device.
address 31 (_)	Address for a broadcast message to all devices (makes sense only for SELECT-telegrams). Every device reacts to the position command without giving a receipt for its telegram (no ACK is sent to the Host computer).

5 Starting the System

5.1 Stand-alone Operation

In stand-alone operation, a connection to the device can be started via the standard address (address 0). Additionally, the connection will be attempted using all possible transfer rates, if the device does not respond to the standard transfer rate of 9600 bps.

5.2 Network Operation

At the beginning of network operation, a network list with the addresses of the devices active on the bus needs to be made. To do this, an ENQUIRY-telegram is sent to every possible address in the system (1 – 30), waiting on a possible ACK. The parameter used to carry out this test should be a numerically compatible parameter (e.g., TYPE).

The follow list contains the numerically compatible parameters of the SMARTDRIVE Inverter and the MASTERCONTROL servocontroller:

Parameter number	Parameter name Inverter	Parameter name servocon-troller
81	SIOC	SBAUD
82	SIOA	SADDR
83	SIOW	SDMMY
84	SIOT	SWDGT
85	SIOF	SERR
91	TYPE	TYPE

The parameters above can be communicated with via the same *CODE-String*. This CODE has the same structure as that for the SMARTDRIVE Inverter.

CODE = 000xx where xx = parameter number (a two-digit, decimal)

After the network list has been created this way, the individual devices can be communicated with, based on their type (Inverter or servocontroller).

Network-monitoring watchdog

All devices offer the possiblity of network monitoring using a programmable time monitor (watchdog). The communication-timeout time of the watchdog is set via the SIOT/SWDGT parameter. Setting it to 0 deactivates the watchdog. If the watchdog is activated, the device monitors for a valid telegram to arrive within the set time. If one does not arrive in this period, the device changes to the error state, registering the corresponding error. For servocontrollers, the additional possiblity exists to program the error reaction of the device via parameter.

A SELECT-telegram or an ENQUIRY-telegram resets the watchdog. The SIOW/SDMMY parameter, which is used to reset the watchdog, is sent via a dummy-telegram. This parameter has its own RAM-memory area in the device; the Host computer can use it as temporary data memory.

6 Parameterization

After every new start of the device, the bit for flagging "power on" in the SIOF/SERR parameter is set. As long as an error bit in SIOF/SERR is set, no parameter in the device can be changed. The set error bits in SIOF/SERR are erased after reading from SIOF.

If an error occurs while transfering a value (SELECT-telegram), an error bit in SIOF/SERR is likewise set (e.g., bit 7 = improper value); and the value transfer or the position command is rejected by the device with a NAK in an answer telegram.

It is thus generally necessary, after a NAK answer from the device (on a value transfer), to read the SIOF/SERR parameter to:

a) get an error identification,

b) be able to change further values, since reading from SIOF again erases the set error bits.

6.1 ENQUIRY-Telegram

An ENQUIRY-telegram (value inquiry) from the Host computer requests the device to send the data belonging to the parameter number (CODE). With an ENQUIRY-answer telegram, the device transmits the requested data to the Host computer. The latter ends the transaction with EOT.

Transaction sequence:

Host computer:

EOT	ADR	**C O D E**	ENQ

Device:

ADR	STX	**C O D E**	=	**V A L**	ETX	BCC

or (on an invalid parameter number):

ADR	NAK

Host computer:

EOT

Note: If the device answers an ENQUIRY-telegram with NAK, then the corresponding parameter is not available or is read-protected in the current operating state. For ENQUIRY-telegrams, entering a cause of error in the SIOF/SERR parameter is fundimentally **not** done.

6.1.1 Examples

Using example telegrams, the data-exchange sequence transacted between Host computer and device will be demonstated. The numbers in small print, under the telegram, are the decimal values of the corresponding character.

6.1.1.1 SMARTDRIVE VF1000 Inverter

Requesting the rotary-field frequency (parameter 12) via address 0

Host computer: value request

Character	EOT	ADR	0	0	1	1	2	ENQ
Number	4	64	48	48	48	49	50	5

VF1000: return of the requested value

Character	ADR	STX	0	0	1	1	2	=	0	3	E	8	ETX	BCC
Number	0	2	48	48	48	49	50	61	48	51	69	56	3	?

Host computer: end communication

Character	EOT
Number	4

6.1.1.2 MasterControl MC6000/7000 Servocontroller

Requesting the current motor speed (parameter 77-SPEED) via address 0

Host computer: value request

Character	EOT	ADR	2	0	0	7	7	ENQ
Number	4	64	50	48	48	55	55	5

Servocontroller: return of the requested value

Character	ADR	STX	2	0	0	7	7	=	0	4	F	A	4	0	0	0	ETX	BCC
Number	64	2	50	48	48	55	55	61	52	52	70	65	52	48	48	48	2	?

Host computer: end communication

Character	EOT
Number	4

Conversion of the hex value to a floating-point, rotation value:

n = 4FA4000H = 83509248 / 65536 = 1274.25 1/min

6.2 SELECT-Telegram

The SELECT-telegram (position instruction) gives parameter data from the Host computer to the device. With error-free acceptance, the position instruction is answered with an ACK receipt.

Transaction sequence

Host computer:

EOT	ADR	STX	**CODE**	=	**VALUE**	ETX	BCC

Device:

ADR	ACK

or:

ADR	NAK

Host computer:

EOT

Note: If the servo, on a transfering a value, answers with NAK, then the Host computer must read the SERR parameter, using an ENQUIRY-telegram, before the dispatch of a further SELECT-Telegram.

6.2.1 Examples

6.2.2 SMARTDRIVE VF1000 Inverter

Configuring the frequency reference value (05-FSSIO) to 50.00 Hz via address 0

Conversion from PT_FIXPOINT16 to integer: value = 50.00 · 20 = 1000 = 03E8H

Host computer:

Character	EOT	ADR	STX	0	0	1	0	5	=	0	3	E	8	ETX	BCC
Number	4	64	2	48	48	48	48	53	=	48	51	69	56	3	?

Inverter:

Character	ADR	ACK
Number	64	6

Host computer:

Character	EOT
Number	4

6.2.3 MASTERCONTROL MC6000/7000 Servocontroller

Configuring the reference valueof the serial interface to 333.254 1/min to the control type SCON (speed control).

Conversion from PT_INT32Q16 to integer: value = 333.254 · 65536 = 21840134 = 14D4106H

Host computer:

Character	EOT	ADR	STX	0	0	0	0	5	=	1	4	D	4	1	0	6	ETX	BCC
Number	4	64	2	48	48	48	48	53	=	48	51	69	56	49	48	54	3	?

Servocontroller:

Character	ADR	ACK
Number	64	6

Host computer:

Character	EOT
Number	4

7 Control Functions

7.1 SMARTDRIVE VF1000 Frequency Inverter

To control the Inverter (i.e., control comands and reference valuetransfers) via the LUSTBUS, the following steps must be taken:

1. Set the operation mode (1-MODE) on the serial interface (0)

2. Select the control location (2-CSEL) on the serial interface (3)

3. Set the frequency reference value selector (4-FSSEL) to "reference valuefrom FSSIO" (8)

4. Control the Inverter by manipulating the 3-CNTL and 5-FSSIO parameters

7.1.1 Example

Control turned on via SIO, reference valuepreset via FSSIO, start with 23.15 Hz, clockwise

This is done using these parameter settings:

MODE	= 0	(Operating mode: Clamp or SIO)
CSEL	= 3	(Control location: Control via SIO)
FSSEL	= 8	(Frequency reference value selector = FSSIO)
FSSIO	= 23.15 · 20 = 463	(Frequency on 23.15 Hz)
CNTL	= 128	(Control word: Start Clockwise)

Telegram sequence:

Telegram type	CODE	VALUE
SELECT	00001	0000
SELECT	00002	0003
SELECT	00003	0008
SELECT	00004	01CF
SELECT	00005	0080

7.2 MASTERCONTROL MC6000/7000 Servocontroller

To begin with, it is assummed that all system settings corresponding to the application have already been set – e.g., programming the EA´s; configuring the desired control typeand the majority of the reference valuestructure.

To control a MASTERCONTROL servocontroller via the LUSTBUS, the following steps are necessary:

1. Bring the drive to a standstill (depends on the current control location).

2. Configure, i.e., program, the desired reference valuestructure. If the reference valuethen needs to be set via the LUSTBUS, then three of the four available reference valueselectors (RSSL1 – RSSL4) must be set to 0 (i.e., RCON). One of these, on the LUSTBUS (RSSLx = 3, that is, RSIO), must be set as the reference valuesource.

3. Preset the control word of the LUSTBUS interface 416-SCTL1 to correspond to the desired function.

4. Set the control location on the LUSTBUS.

5. Now the drive can be controlled via the control word 416-SCTL1, or the reference valuepreset via the 428-RSIO parameter can be carried out.

7.2.1 Example

Setting a task:

The following configurations are desired:

- control type - speed control -
- control via the serial interface (SIO)
- reference valuedirect via the SIO without offset and without ramp function via the reference valueselector 1
- Start with 132.15 rpm, right run

For this, the following configurations are necessary:

Action	Parameter	Newer value
Bring drive to a standstill	corresponding control location	?
Set control type "speed control"	300-CFCON	2 (SCON)
Preload control word (Standstill, or all functions off)	416-SCTL1	0
Control location on LUSTBUS	402-CLSEL	2 (SIO)
Deactivate unnecessary reference value selectors	418-RSSL2 419-RSSL3 420-RSSL4	0 (RCON) 0 (RCON) 0 (RCON)
Set reference valueselector 1 on LUSTBUS	417-RSSL1	3 (RSIO)
Set desired reference value	428-RSIO	132.15 · 65536
Start drive	416-SCTL1	set bit 0; value = 1

Telegram sequence:

Telegram Type	CODE	VALUE
SELECT	20300	2
SELECT	20416	0
SELECT	20402	2
SELECT	20418	0
SELECT	20419	0
SELECT	20420	0
SELECT	20417	3
SELECT	20428	842666
SELECT	20416	1

8 When a Device malfunctions

Error conditions are indicated by the LEDs on the device, by the red-background lighting of the KEYPAD display, and by the device status word.

During errors, the SMARTDRIVE Inverter fundimentally blocks the end step. The error reaction of the MasterControl servocontroller is programmable, in five steps, for every error.

Errant devices are still accessable via the KEYPAD and all connected bus systems.

8.1 Acknowledging Device Errors

8.1.1 SMARTDRIVE VF1000 Inverter

With a set error bit in the 11-STAT parameter, a reset can be triggered by a SELECT-telegram on this parameter, using the new value 000F Hex (VALUE = "000F"). Since the reset happens immediately after the receipt of error, the Inverter's ACK-answer telegram, under certain circumstances, is not sent in full. Also, the connection to the Inverter is broken for the duration of the boot-up (self test).

After communications with the Inverter are again available, the SIOF parameter (power-down report) must be read with an ENQUIRY-telegram before the next SELECT-telegram.

8.1.2 MASTERDRIVE MC6000/7000 Servocontroller

If the servocontroller is in an errant condition, a reset of the error condition can be triggered by setting bit 7 (value 80H) in parameter 416-SCTL1.

The individual error reactions of the servocontroller are programmable. Resetting an error, which has a 5 ("RESET") programmed for its reaction, makes the servocontroller boot. Since the reset happens immediately after the receipt of error, the servocontroller's ACK-answer telegram, under certain circumstances, is not sent in full. Also, the connection to the Inverter is broken for the duration of the boot-up (initialization and self test).

After communications with the servocontroller are again available, the SERR parameter (powerdown-report) must be read with an ENQUIRY-telegram before the next SELECT-telegram.

9 When an Error occurs

9.1 Notes on Starting Operation

For various reasons, it can happen that a device does not answer a telegram:

- There is no answer, if the telegram frame (Baudrate, data width, start bit, stop bit) are not set correctly on the Host computer.
- There is no answer, if communications with a device is attempted using the wrong bus address.
- There is no answer, if the serial connection between the Host computer and device is not correctly established.
- There is no answer using bus address 31 (5Fh). This address is reserved for Broadcast-Telegrams.
- There is no valid answer, if mulitple devices are connected to bus and a telegram with the address 0 (40h) is dispatched, since, in this case, all devices answer simultaneously.
- There is no valid answer, if mulitple devices with the same bus address are connected to the bus and communications are attempted using this address. Also in this case, these devices answer simultaneously.

9.2 SELECT-Telegram

If an error occurs during the transfer of a value by a SELECT-telegram, the cause of error is documented in the 85-SIOF/SERR parameter. All further SELECT-telegrams are rejected as long as bits within this parameter are set. This parameter can only be erased by doing an ENQUIRY-telegram on this parameter itself. Additionally, after every device power on, the "power-on" bit of this parameter is set. Thus it is possible to determine if the device has been newly started, e.g., after a power failure or reset of an error.

Thus, after a device answers with NAK, during a transfer of a value (SELECT-Telegram), it is generally necessary to read the 85-SIOF/SERR parameter, to:

a) get an error identification,

b) be able to change further values, since reading from SIOF/SERR again erases the set error bits.

The following table lists the conditions that lead to an error being set in SIOF/SERR and notes what can be done about it.

Bit in SIOF/SERR	Hex-Value	Short descriptor	Cause	What to do
0	01H	Power on	Bit is set for each ON	No error, information about device new start
1	02H	Watchdog	Device watchdog has timed out	Fix the malfunction in the Host computer. Or Host computer is too slow
2	04H	EEPROM busy	EEPROM is currently doing a memory burn in	Wait (> 60 ms) and repeat telegram
3	08H	Checksum error	Error in data transfer – checksum is wrong	Repeat telegram
5	20H	No parameter	There is no parameter for the given number	General service error or error in the software of the Host computer.
6	40H	No change	Change of the parameter is not permitted	Check if the parameter cannot be writen to generally, or if it is due to the curent operating mode.
7	80H	Invalid value	The transfer parameter value is not permissable	Choose a different value

10 Timing

The following explains the principle transaction sequences of a telegram cycle, irrespective of telegram type:

1	2	3	4	5
Host computer sends telegram	Device reaction time until the telegram is processed	Device sends answer telegram	Lead-computer reaction time	Host computer sends EOT

Points 1, 3, and 5 are calculatable amounts, which depend only on the amount of data and the transfer rate of the bus system.

Point 2 depends on the type and operating condition of the device. For this, the following estimates can be made:

The timing of the devices on the bus, i.e., the time to react to a telegram depends on the individual device and its load. Fundimentally, the reaction time can be reduced for all devices by operating without a KEYPAD.

The following table shows the relationships:

Device series	Reaction time with KEYPAD	Reaction time without KEYPAD
MASTERCONTROL Servocontroller	1…30 ms	1…25 ms
SMARTDRIVE Inverter	1…60 ms	1…30 ms

Exceptions:

1. If a 1 is entered in the device's 4-PROG parameter (for MC6000/7000) or in the 71-PROG parameter (for VF1000), then the divice overwrites all parameter settings with its reference value. In this case, the ACK answer telegram is then first sent, if the complete parameter list is newly initialized. This action can take up to 10 seconds, depending on the device.

2. While a SMARTCARD is being read in, no communications with the device via the LUSTBUS is possible. This condition can last up to 10 seconds, depending on the device.

3. If the receipt of an error condition is noted via the LUSTBUS, under certain circumstances, this can lead to a new start of the device. More detailed information can be found in the chapter "When a Device Malfunctions".

4. When writing EEPROM, VF1xxx parameters with a value, which differs from the old value, the reaction time increases about 120 ms for two-byte parameters and 60 ms one-byte parameters.

10.1 Telegram Durations

10.1.1 Telegram Time for an ENQUIRY-Telegram

Example 1: Value request for a VF1000 Inverter with KEYPAD with 9600 Baud

Max. reaction time of the Inverter:

Action	Data size	Time
Host computer sends telegram	8 Byte	8Byte · 8 Bit/Byte / 9600 bps = 7 ms
Reaction time of the Inverter	-	Max. 60 ms
Inverter answer	14 Byte	14 · 8 /9600 = 12 ms
Reaction time of the Host computer		e.g., 2 ms
Host computer sends EOT	1	8 /9600 = 1 ms
Total		= 82 ms i.e., => 12 telegrams/sec

Min. reaction time of the Inverter:

Action	Data size	Time
Host computer sends telegram	8 Byte	8Byte · 8 Bit/Byte / 9600 bps = 7 ms
Reaction time of the Inverter	-	Min. 1 ms
Inverter answer	14 Byte	14 · 8 /9600 = 12 ms
Reaction time of the Host computer		e.g., 2 ms
Host computer sends EOT	1	8 /9600 = 1 ms
Total		= 23 ms i.e., => 43 telegrams/sec

Result: The telegram frequency for this case is between 12 and 43 telegrams/sec

Example 2: Value request for a 32-bit parameter of a servocontroller with KEYPAD and 9600 Baud

Max. reaction time of the servocontroller:

Action	Data size	Time
Host computer sends telegram	8 Byte	8Byte · 8 Bit/Byte / 9600 bps = 7 ms
Reaction time of the servocontroller	-	Max. 30 ms
Servocontroller answer	18 Byte	18 · 8 /9600 = 15 ms
Reaction time of the Host computer		e.g., 2 ms
Host computer sends EOT	1	8 /9600 = 1 ms
Total		=55 ms i.e., => 18 telegrams/sec

Min. reaction time of the servocontroller:

Action	Data size	Time
Host computer sends telegram	8 Byte	8Byte · 8 Bit/Byte / 9600 bps = 7 ms
Reaction time of the servocontroller	-	Min. 1 ms
Servocontroller answer	18 Byte	18 · 8 /9600 = 15 ms
Reaction time of the Host computer		e.g., 2 ms
Host computer sends EOT	1	8 /9600 = 1 ms
Total		=26 ms i.e., => 38 telegrams/sec

Result: The telegram frequency for this case is between 18 and 38 telegrams/sec

10.1.2 Telegram Time for a SELECT-Telegram

Example 1: transfer of a value to a SMARTDRIVE INVERTER with KEYPAD with 9600 Baud

Max. reaction time of the Inverter:

Action	Data size	Time
Host computer sends telegram	15 Byte	15 Byte · 8 Bit/Byte / 9600 bps = 13 ms
Reaction time of the Inverter	-	Max. 60 ms
Inverter answer	2 Byte	2 · 8 /9600 = 2 ms
Reaction time of the Host computer		e.g., 2 ms
Host computer sends EOT	1	8 /9600 = 1 ms
Total		= 78 ms i.e., => 13 telegrams/sec

Min. reaction time of the Inverter:

Action	Data size	Time
Host computer sends telegram	15 Byte	8 Byte · 8 Bit/Byte / 9600 bps = 13 ms
Reaction time of the Inverter	-	Min. 1 ms
Inverter answer	2 Byte	2 · 8 /9600 = 2 ms
Reaction time of the Host computer		e.g., 2 ms
Host computer sends EOT	1	8 /9600 = 1 ms
Total		= 19 ms i.e., => 53 telegrams/sec

Result: The telegram frequency for this case is between 13 and 53 telegrams/sec

Example 2: transfer of a value to a 32-bit parameter of a MASTERDRIVE servocontroller with KEYPAD and 9600 Baud

Max. reaction time of the servocontroller:

Action	Data size	Time
Host computer sends telegram	19 Byte	19 Byte · 8 Bit/Byte / 9600 bps = 16 ms
Reaction time of the servocontroller	-	Max. 30 ms
Servocontroller answer	2 Byte	2 · 8 /9600 = 2 ms
Reaction time of the Host computer		e.g., 2 ms
Host computer sends EOT	1	8 /9600 = 1 ms
Total		=51 ms i.e., => 20 telegrams/sec

Min. reaction time of the servocontroller:

Action	Data size	Time
Host computer sends telegram	19 Byte	19 Byte · 8 Bit/Byte / 9600 bps = 16 ms
Reaction time of the servocontroller	-	Min. 1 ms
Servocontroller answer	2 Byte	2 · 8 /9600 = 2 ms
Reaction time of the Host computer		e.g., 2 ms
Host computer sends EOT	1	8 /9600 = 1 ms
Total		=22 ms i.e., => 45 telegrams/sec

Result: The telegram frequency for this case is between 20 and 45 telegrams/sec

10.2 Chronological Order of the Telegrams

10.2.1 SMARTDRIVE VF1000 Frequency Inverter

After the device has sent an ACK-answer telegram, the Host computer can directly send the next telegram; wait time between these telegrams is not necessary.

10.2.2 MASTERCONTROL MC6000/7000 Servocontroller

For ENQUIRY-telegrams and SELECT-telegrams to RAM parameters, no wait time is necessary. For SELECT-telegrams to EEPROM parameters, a wait time is needed, if the duration of a telegram sequence is shorter than the time required to save the parameter in EEPROM. The determination of the time for a telegram sequence has been explained previously.

Time to save an EEPROM parameter is calculated so:

memory time= parameter data width · 3 · 5 ms

The required wait time is the difference between the telegram duration time and time needed to save in EEPROM. If the following resulting term is positive, a wait time, as calulated in the term, must be used:

wait time = memory time – telegram duration

If SELECT-telegrams to EEPROM parameters follow one right after another, and the servocontroller cannot save the the EEPROM-parameter data fast enough, it flags this condition by setting the bits "EEPROM busy" in the LUSTBUS error-status parameter, SERR.

Example:

A servocontroller operates on the LUSTBUS at 9600 Baud and with an attached KEYPAD. Parameter data having a maximum length of 4 bytes in series are to be given to the Inverter:

In a previous example in this chapter, the telegram duration time for this specific case was calulated. Decisive for this example is the minimal telegram duration.

The calculation result was: Minimum telegram duration time = 22 ms

The memory time is: 4 bytes · 3 · 5 ms/byte = 60 ms

To guarantee error-free telegram transfer in this case, the wait time required is:

wait time = memory time – telegram duration time = 60 ms – 22 ms = 38 ms

11 Device Special Functions

11.1 Requesting the Inverter Parameter Description

If, in an ENQUIRY-telegram the CODE's 1st digit (the specificator) = 1 is selected, then the value of the requested parameter is not given, but instead, the parameter description which belongs to this parameter is given. The length of the *VALUE-String* in the answer telegram is, in this case, not 4 characters, but 26.

This *VALUE-String,* that is, the parameter description has following contents:

Description	No. of Characters	Coding
Parameter name	6 ASCII – chars	ASCII – text
Unit	6 ASCII – chars	ASCII – text
Parameter kind	1 ASCII – chars	counter type
Parameter type	1 ASCII – chars	counter type
Minimal value	4 ASCII – chars	value type
Maximal value	4 ASCII – chars	value type
Basis value	4 ASCII – chars	value type
Total	26 ASCII – chars	

Parameter name

The parameter name has a maximum of 6 letters (right-justified, shorter names are padded with blanks). Combinations of digits, capital letters, underscores (_), and hyphens (-) are permissible.

Unit

Abbreviation for the Unit has a maximum of 6 letters (right-justified, shorter Unit abbreviations are padded with blanks).

Parameter kind

For frequency Inverters, there a fundimentally two kinds of parameters to differentiate:

- the basis parameter, and
- the Inverter specific parameter

Basis parameters are those which have an identical number and meaning for all Inverter series. The value range of this parameter can be device dependent. With parameters of this kind, the device number in the telegram CODE (see the chapter, "Telegram Structure, Syntax, and Creation") can contain a 0 (general for basis parameter) as well as the specific device number.

Inverter specific parameters have differing functions in the various devices. These parameters are accessable only via a specific device number in the telegram CODE.

The following kinds of parameters are specified:

Character	Reference	Description
'0'	RAM control size (Basis parameter)	Parameters of this kind are usually editable and have a memory location in the RAM data area. They loose their values on power down.
'1'	RAM Actual value (Basis parameter)	These parameters are not editable and are only for displaying values.
'2'	EEPROM (Basis parameter)	Values of these parameter are stored in the EEPROM data area of the device and are not lost on power down. For SMARTDRIVES, the can be edited only during motor standstill.
'3'	FSEL/ISEL variable (Basis parameter)	FSEL/ISEL variables: if CSEL ≤ 3, they are RAM variables (control via serial interface). If CSEL < 3, they are EEPROM-Variables.
'4'	RAM control size (Param. Inverter specific)	see '0'
'5'	RAM-Is value (Param. Inverter specific)	see '1'
'6'	EEPROM (Param. Inverter specific)	see '2'
'7'	FSEL/ISEL variable (Param. Inverter specific)	see '3'

Parameter type

The parameter type describes the data type and possibly the denormalization requirements of the parameters. How to use the individual types is explained in chapter "Telegram Structure, Syntax, and Creation" under " Description and syntax of the Parameter Data".

The following table shows the coding of the Data types for the SMARTDRIVE:

Character	Data type	Scaling	Value range	Display KP100
'1'	PT_BINARY8	1	0 ... 255	00H ... FFH
'2'	PT_FIXPOINT16	0.05	0.00 ... 3276,80	0.00 ... 999.95
'3'	PT_USIGN16	1	0 ... 65535	0 ... 63535
'4'	PT_ USIGN8	1	0 ... 255	0 ... 255
'5'	PT_VF_ERROR	1	0 ... 65535	plain text e.g., E-OV
'6'	PT_PASSWORD	1	0 ... 65535	0 ... 65535

Value area

The folowing elements describe the range of values and the reference values. The have the same scaling as the parameter value:

Minimal value: Minimum value of the parameters

Maximal value: Maximum value of the parameters

Basis value: Default value (factory setting) of the parameters

11.2 Requesting the Servocontroller-Parameter Description

If, in an ENQUIRY-telegram the CODE's 1st digit (the specificator) = 3 is selected, then the value of the requested parameter is not given, but instead, the parameter description which belongs to this parameter is given. The length of the *VALUE-String* in the answer telegram is, in this case, 43 characters.

This *VALUE-String,* the parameter description, contains the following:

Element of the parameter-description	Element length in bytes	Description of the Elements
Parameter name	6	Short name, ending with ´\0´
Parameter unit	1	Code for the physical unit of the parameter
Parameter kind	1	What kind of parameter (e.g., EEPROM)
Parameter type	1	Type of the parameter (e.g., USIGN8)
Minimum	8	Minimum value
Maximum	8	Maximum value
Reference value	8	Factory preset
Level	1	Read \ write level
Attribute	2	Special characteristic
Hardware filter	2	Filter for automatic selection of the current hardware-relevant parameter
Structure filter	2	Filter for automatic selection of the current controller-relevant parameter
Subject Area	1	parameter specific subject area (e.g., _MOT)
Activity Area	1	Activity Area in connection with the SMARTCARD
Total length	**43**	

For usual applications (adjusting of a parameter value via the serial interface), the parameter information, up to the element "reference value," is sufficient.

The extended parameter description (starting with the element "Level") makes it possible to imitate exactly the dynamic menu of the KEYPAD (KP100) on the Host computer.

11.2.1 Detailed Description of the Individual Elements

For configuration or monitoring several known parameters, the exact evaluation of the parameter description is not absolutely necessary, since the description for several special parameters can be firmly integrated into the Host computer's program.

However, if universal and version-independent software for the Host computer is to be written, then the software must supply the parameter description directly out of the respective servo. This is done, as already described, by an ENQUIRY-telegram with a 3 in the first character position of the *CODE-String*.

In following, the individual elements of the parameter description are described in detail:

The parameter name

The parameter name is made up of five characters ending with the ´\0´ character (i.e., compatable to string termination as in C). Combinations of digits, capital letters, underscores (_), and hyphens (-) are permissible.

The parameter name does not need to be unique. If there are multiple instances of a parameter name, then it will be assumed automatically that a virtual parameter is being treated.

Virtual parameters have ambiguous parameter names, but unique parameter numbers. According to the Inverter configuration, the parameter, which belongs to the current structure, will be automatically selected from the KEYPAD (KP100), so that, for the user, only a virtual parameter of the same name is always visible.

By evaluating the parameter description, it is possible to imitate this functionality in the Host computer's software.

The parameter unit

The element, unit, of the parameter description is an index for the 107-UNIT parameter. The unit indicates the physical unit of the parameter. By reading in the value substitution text of the UNIT parameter, the unit-string of the parameter can be determined via this index.

Reading in the value substitution text is explained in detail in the chapter "Loading Parameter-value Substitution Text".

The parameter kind

The parameter kind is coded in a byte. The code is in the range of the displayable charaters in the ASCII-Table, so that they are not confused with control characters when transfered via the LUSTBUS.

The following table shows the various parameter kinds and their coding:

Parameter kind	ASCII Code	Characteristic
RAM Actual value	´0´	Parameters of this kind are not editable and have no EEPROM memory area. They loose their values on power down of the servocontroller. A new start of the device pre-initializes them to 0 .
RAM control value	´1´	On new start, this parameter is set to its reference value. Like the RAM Is- value, RAM control values also do not have EEPROM memory addresses, i.e., they loose their values on power down. These can be edited. After a write command (SELECT-telegram) on such a parameter, no wait time is needed, since, as already indicated, they are not saved in the EEPROM of the servocontroller.
EEPROM value	´2´	Parameters of this kind each have an EEPROM address and a RAM address. When the device is started, the EEPROM value is copied into the corresponding RAM. In write accessing this parameter, the RAM is always written to first and then the EEPROM. This guarantees that the new parameter value is immediately available, i.e., the EEPROM memory functions here only as a backup. After a SELECT-telegram on such a parameter, a wait time is needed, corresponding to the parameter data width (5 ms/byte).

The parameter type

The parameter type indicates the maximum value range of a parameter and its representation on the display of the KEYPAD (KP100).

Several of the data types are not displayable on the KEYPAD (KP100) across their full range of values. More detailed descriptions of the available data types can be found in the chapter "MASTER-CONTROL Servocontroller MC6000/7000 Parameter Data Types".

The parameter reference value

These values give the parameter's value range and its factory setting.

For each and every size (min., max., default), eight characters are transfered.

The user level

These and the following structure elements serve only in reproducing the service structure of the KP100. If this functionality is not required, then these elements do not need to be evaluated.

In each 4 bits of the "user level" byte, the read and write access levels for the parameter are encoded respectively. This makes 16 read, and 16 write, levels possible.

In the byte, the LO nibble is for the read level, and the HI nibble for the write level, of the parameter.

The attribute

Special characteristics of the parameter are indicated via the attribute. For each characteristic, a bit in the "attribute" word is reserved. A parameter can have none or many attributes. For every attribute is a unique bit reserved.

The following attributes exist at this time:

Attribute	Bit-position	Bit-mask	Description
ShowParaText	2	0004H	The name of the parameters is indicated with this number. Parameters with this attribute serve to select other parameters. For example, BARG is used to call the parameters which should be shown in the bargraph display. Then, e.g., for the parameter value 1, MODE is indicated in the display of the KEYPAD.
ShowNoFract	3	0008H	For the data type FIXPOINT16, which commonly has a 2, trailing positions for decimal fractions, the KEYPAD shows no decimals.
ShowFract1	4	0010H	For the data type FIXPOINT16, which commonly has a 2, trailing positions for decimal fractions, the KEYPAD shows only 1 decimal position.
ShowText	5	0020H	Parameter values are shown as plain text instead of numbers. For this, the value substitution text for this parameter must be read from the the servo. The exact procedure is explained in the chapter **"Loading Parameter-value Substitution Text"**. For this procedure, only type *usign8* parameters are allowed. Every value of the parameter (beginning with 0) is thus assigned a specific text from the field of the parameter-value substitution text.
InitControl	6	0040H	If a parameter is overwritten with the attribute, then the Inverter will register a Inverter initialization because it must calculate its initialization value anew.
NoInit	7	0080H	RAM-control parameters that have this attribute will not be pre-initialized, during start up, with the factory setting from the parameter list.
ShowHex	8	0100H	Parameter value is displayed in hexadecimal.
Password	9	0200H	Parameter is a password

Hardware flags for filtering the parameter

In the KEYPAD of the servocontroller, not all of the parameters are available at the same time – only those, which are important for the current operating condition and are necessary for the current user's understanding, are available. Also the parameters must originate from the currently set Subject Areas. These are four crtieria for selecting a parameter, which logically seen, work like four filters in a row.

The element "level" was already explained above as one of these filters applicable to the user's understanding of the running system.

The "hardware filter" element defines which hardware configuration for this parameter is of interest. For each hardware unit there is one bit respectively.

At this time, the following references exist, which correspond to the desired filter action. They can be bit OR'ed:

Flag	Bit-position	Bit-mask	Description
Resolver	0	0001H	The parameter passes through the filter, if the connected sender is a resolver.
EncoderSin	1	0002H	The parameter passes through the filter, if the connected sender is an incremental encoder having a sinusoidal output signal.
EncoderRec	2	0004H	The parameter passes through the filter, if the connected sender is an incremental encoder having a with square-wave output signal.
EncdSsiSingle	3	0008H	The parameter passes through the filter, if the connected sender is an incremental encoder having an SSI interface, sinusoidal output signal, and absolute information about a single rotation.
EncdSsiMulti	4	0010H	The parameter passes through the filter, if the connected sender is an incremental encoder having an SSI interface, sinusoidal output signal, and absolute information about multiple rotations.
Option1	5	0020H	The parameter passes through the filter, if option slot 1 has a valid option board.
Option2	6	0040H	The parameter passes through the filter, if option slot 2 has a valid option board.
ASM	7	0080H	The parameter passes through the filter, if the connected motor is an asynchronous machine.
SM	8	0100H	The parameter passes through the filter, if the connected motor is an synchronous machine.
Special-drive	9	0200H	The parameter passes through the filter, if the connected motor is a special drive (e.g., a reluctance motor).
EA1	10	0400H	The parameter passes through the filter, if the "EA-extention 1" is available.
EA2	11	0800H	The parameter passes through the filter, if the "EA-extention 2" is available.
CAN	12	1000H	The parameter passes through the filter, if the CAN-module is available.
Extended Memory	13	2000H	The parameter passes through the filter, if memory extension is available.
MPTC	14	8000H	The parameter passes through the filter, if a Motor-PTC is available.
Brake	15	8000H	The parameter passes through the filter, if the brake driver is available.

The structure word for filtering the parameter

As already explained, there are four filters for selecting the important parameters for the KEYPAD. The element "structure filter" is used for filtering the parameter of the current operating condition.

The element is bit encoded and thus has 16 different possible combinations of values:

Flag-reference	Bit position	Bit mask	Description
TorqueCtrl	0	0001H	The parameter passes through this filter, if torque control is active.
SpeedCtrl	1	0002H	The parameter passes through this filter, if speed control is active.
PosCtrl	2	0004H	The parameter passes through this filter, if position control is active.
VF_Conv	3	0008H	The parameter passes through this filter, if the voltage-frequency control is active.
EMaster	4	0010H	The parameter passes through this filter, if the control type "Electric Master" is active.
ESlave	5	0020H	The parameter passes through this filter, if the control type "Electric Slave" is active.

Filtering the Subject Area parameter

In the PARA-menu of the KEYPAD, the parameters are divided into Specific Subject Areas. The structure element "Specific Subject Area" indicates to which menu the corresponding parameter belongs. The Subject Areas are sequentially numbered and thus are not combinable. This means that every parameter is unique and is assigned to only one Subject Area.

The following Specific Subject Areas are defined at this time:

Specfic Operational Area Abbreviation	Sequential Number	Description
CONF	0	System configuration
ENCD	1	Encoder
OPT1	2	Option slot 1
OPT2	3	Option slot 2
MOT	4	Motor type
TCON	5	Torque control
SCON	6	Speed control
PCON	7	Position control
VFCN	8	Voltage-frequency characteristic curve
SIO	9	Serial interface
KPAD	10	KEYPAD
PWM	11	Pulse-width modulation
SCTY	12	Security-and-error-reaction parameter
USER	13	Special user-triggered parameter from special software
SYS	14	Diagnosis and digital scope
REF	15	Reference value structure
IO1	16	EA-extension 1
IO2	17	EA-extension 2
CAN	18	CAN-bus
PMOD	19	Positioner unit
MENU	20	Parameter for menu control on the KEYPAD KP100
VAL	21	Actual value parameter from the VAL menu of the KEYPAD

Activity Areas and the SMARTCARD

The parameter structure element *sc_group* serves to divide the parameter into Activity Areas relating to the SMARTCARD.

This byte assigns or gives the parameter's Activity Area. This also clarifies if the parameter is appropriate for exchanging data with the SMARTCARD (and the SIO loads with the function parameter database). If the value does not equal 0, it gives the number of the Activity Area; if the value is equal to 0, then the parameter is inappropriate for data exchange with the SMARTCARD.

The following Activity Areas are defined at this time:

Reference	Value	Description
SC_Nothing	0	Parameter is for data exchange with the SMARTCARD and is not appropriate for the database function of the serial interface.
SC_OPT1	1	Programming of optional slot 1
SC_OPT2	2	Programming of optional slot 2
SC_APP	3	Equipment specific parameter (e.g., moment of inertia in the equipment)
SC_SYS	4	General system parameter
SC_DRIVE	5	Parameter for configuration of the drive system
SC_REF	6	Parameter for EA´s and the reference value generation
SC_CAN	7	Parameter for CAN-bus
SC_PMOD	8	Parameter for Positioner Unit

11.3 Loading the Parameter-Value Substitution Text

In the KEYPAD of the servo, various parameter values are displayed as text. The text ultimately stands for a number that is saved as parameter value.

So that these KEYPAD functions are also available to the Host computer's software, it is possible to send the parameter's corresponding value substitution text to the Host computer via the serial interface.

The Host computer's software can test if the parameter has a value substitution text, by checking the "attribute" description element.

The following steps must be taken to get the value substitution text of a parameter:

1. Test, using the structure-element attribute of the parameter, if the parameter has a value substitution text.
2. Set the STEXT parameter to the value of the desired value substitution text.
3. Using an ENQUIRY-telegram and the Code 61xxx (xxx = ParameterNumber), get the value substitution text corresponding to the value via the 108-STEXT parameter.

The Servo answers with NAK when:

- the parameter does not exist
- the parameter has no value substitution text
- the value (of the 108-STEXT parameter) for the value substitution text is larger than the maximum for the parameter

11.4 Download Handshaking for Parameter-data Records

A problem:

A coherent and valid dataset should be transfered from the Host computer to the servocontroller. With each transfer of a single parameter, the servocontroller tests if the parameter correctly fits its dataset.

During the test of the new parameter value, existing parameter values may also be taken into account. This creates the possibility that the servocontroller may reject a parameter, even though it came from a valid dataset.

A solution:

The new parameter dataset of the Host computer is transfered without individual testing of the parameter values on the servocontroller. If the upload has finished, the servocontroller tests the new, complete dataset to see if it is plausible. If the data is implausible, the full dataset is rejected and the old dataset is reactivated.

This procedure requires handshaking, which is described in more detail, as follows.

Important: In this action only the parameters that have the "CardWriteable" attribute are changed. This way, an upload of a parameter dataset via the serial interface works analogous to the SMARTCARD. If, during the upload, a Select-telegram to a parameter is sent without the "CardWriteable" attribute, the servocontroller still answers "acknowlege" to this telegram, but does not take new parameter value.

Handshaking for uploading of a complete parameter dataset:

1. Register an upload with parameter 80-SLOAD = -1

Write accessing this parameter is possible only when the system is in a stillstand mode. After the write access, the servocontroller will be secured against being engaged until the download has ended.

2. Complete parameter dataset transfer

Using multiple SELECT-telegrams, the Host computer gives the single parameters to the servocontroller. The latter takes the new parameter values without doing a plausibilty check.

3. End upload with parameter 80-SLOAD = -2

If all parameter data has been transfered, then the Host computer sets SLOAD to the value (-2). This signals the end of the data transmission to the servocontroller. It then starts testing the new and complete dataset for plausibilty. If the dataset is valid, the parameters having the "CardWriteable" attribute are put in EEPROM. The drive is freed and can be started. The parameter 80-SLOAD is set corresponding to the results of the parameter-test.

4. Poll parameter 80-SLOAD with a timeout of 10sec.

If SLOAD goes to 0 within the timeout interval, then the transfer has ended correctly. The parameters having the "CardWriteable" attribute are put in EEPROM. The drive is freed and can be started.
If SLOAD = (-1) within the timeout interval, then the servocontroller is still busy testing and saving.
If SLOAD > 0, then the servocontroller has rejected the dataset; the SLOAD value then corresponds to the number of the first parameter that has an invalid value.

Important: If during the transfer, the connection is broken or the timeout runs out, the transfer must be repeated or the servocontroller must be restarted.

11.5 Changing the Baudrate Online

The Baudrate of the devices is not altered immediately after each change. Instead, the change first becomes active once the device is restarted. This prevents broken connections during the upload of a new dataset under certain circumstances.

The servocontroller has the possibility to activate a new Baudrate also without having to do a restart. For this, first set the new Baudrate. Then use the value 111 in the 4-PROG parameter to order the servocontroller to activate the Baudrate.

11.6 Communicating with the POSMOD1 (MC6000 only)

Using the parameter number CODE = 49999, telegrams can be sent to the POSMOD1 option of the servocontroller MC6000.

There are two kinds of telegrams: the SELECT-telegram and the ENQUIRY-telegram. A SELECT-telegram, which is always the first sent, can transfer data to POSMOD1. And an ENQUIRY-telegram can request data from the POSMOD1. For exchanging data with the POSMOD1, only a specific CODE-String (49999) is designated.

If the MC6000 signals an error condition with NACK, the Host computer ascertains the error using the read command %RDF.

11.6.1 SELECT-Telegram

Host computer:

EOT	@	STX	4	9	9	9	9	=	0 ... 80 Byte ASCII-Data (use-data)	ETX	BCC

With the help of the SELECT-Telegram, data is sent to the POSMOD1 or the data to be requested is specified.

11.6.2 ENQUIRY-Telegram

Host computer:

EOT	@	4	9	9	9	9	ENQ

The ENQUIRY-telegram requests the data which has been specified by a SELECT-telegram. The MC6000 then sends a receipt telegram having following structure:

Servocontroller:

@	STX	4	9	9	9	9	=	0 ... 80 Byte ASCII-Daten (use-data)	ETX	BCC

11.6.3 User Data Structure

The user data is a status byte followed by the actual telegram data. The status byte has the following structure:

Bit 0: 1 = Request telegram
This involves a SELECT-telegram, which is specified by the kind of requested data (e.g., a status request from the POSMOD1).
0 = Data telegram
This involves a telegram, which contains no data request, requiring only an ACK receipt from the receiver (e.g., a traversing command in manual mode).

Bit 1: is always 0

Bit 2: 1 = Multiblock telegram
This involves a data telegram, which – after the receiver having sent a receipt using an ENQUIRY-telegram – has more telegrams following (e.g., program transfer).
0 = Single telegram
This involves a single telegram, or the last telegram of a multiblock transfer that has no more following telegrams.

Bit 4, 5 always = 1

Bit 6, 7 always = 0

Please find the user data in the command description of the user handbook.

11.6.4 Example telegrams

Passing a manual command to POSMOD1

Host computer:

EOT	ADR	STX	4	9	9	9	9	=	38H	GOA0V1000	ETX	BCC

Servocontroller:

ADR	ACK

or

ADR	NAK

Host computer:

EOT

Transmitting a program to POSMOD1

Host computer:

EOT	ADR	STX	4	9	9	9	9	=	3CH	%P10 (RPF)	ETX	BCC

Servocontroller:

ADR	ACK

or

ADR	NAK

Host computer:

EOT

EOT	ADR	STX	4	9	9	9	9	=	3CH	N10 GO 0	ETX	BCC

Servocontroller:

ADR	ACK

or

ADR	NAK

Host computer:

EOT

EOT	ADR	STX	4	9	9	9	9	=	38H	END	ETX	BCC

Servocontroller:

ADR	ACK

or

ADR	NAK

Host computer:

EOT

Status request from POSMOD1

Host computer:

EOT	ADR	STX	4	9	9	9	9	=	31H	?	ETX	BCC

Servocontroller:

ADR	ACK

or

ADR	NAK

Host computer:

EOT

EOT	ADR	4	9	9	9	9	ENQ

Servocontroller:

ADR	STX	4	9	9	9	9	=	30H	G10H10 ...	ETX	BCC

Host computer:

EOT

Program request from POSMOD1

Host computer:

EOT	ADR	STX	4	9	9	9	9	=	31H	%RDP10	ETX	BCC

Servocontroller:

ADR	ACK

or

ADR	NAK

Host computer:

EOT

Host computer:

EOT	ADR	4	9	9	9	9	ENQ

Servocontroller:

ADR	STX	4	9	9	9	9	=	34H	%P10 (RPF)	ETX	BCC

Host computer:

EOT

Host computer:

EOT	ADR	4	9	9	9	9	ENQ

Servocontroller:

ADR	STX	4	9	9	9	9	=	34H	N10 GO 0	ETX	BCC

Host computer:

EOT

Host computer:

EOT	ADR	4	9	9	9	9	ENQ

Servocontroller:

ADR	STX	4	9	9	9	9	=	30H	END	ETX	BCC

Host computer:

EOT

Lust Antriebstechnik GmbH • Gewerbestr. 5-9 • 35633 Lahnau • GermanyTelephone + 49 64 41/9 66-0 • Telefax + 49 64 41/9 66-1 37