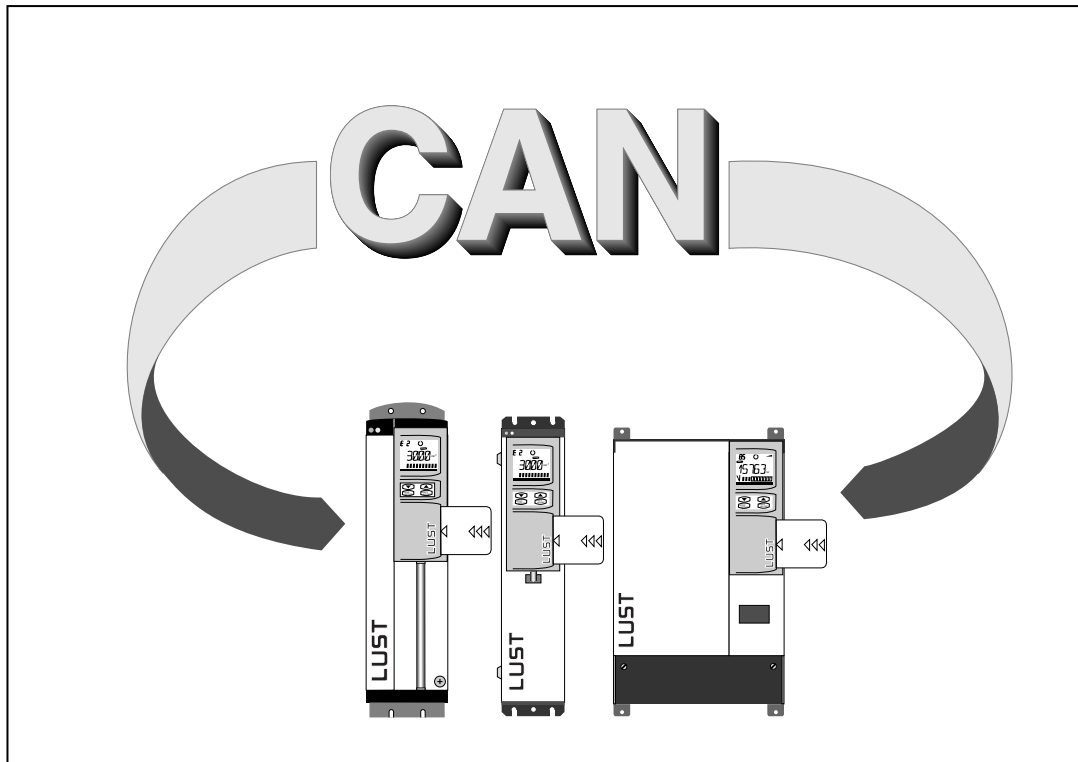


Interconnection of inverters and servocontrollers

EN

on the CAN BUS



Data transfer protocol

CAN BUS data transfer protocol

applicable to Inverters of series
SMARTDRIVE VF1000 S/M/L

Servocontrollers of series
MASTERDRIVE MC6000/7000

Date: March 1999

ID no.: A047.22B.1-00

We reserve the right to make technical changes.

Table of contents

1	General introduction.....	7
1.1	System requirements.....	7
1.2	User level in operation over CAN bus	7
1.3	Further documentation	7
1.4	General information on the structure of a CAN network.....	8
1.4.1	Multimaster capability	8
1.4.2	Access rights	8
1.4.3	Size of identifiers	9
1.4.4	Telegram structure	9
1.4.5	Time response	9
1.4.6	Transmission speeds.....	10
1.5	CAN protocol for LUST drives.....	10
1.5.1	System states	11
1.5.2	Device states	11
1.5.3	Device control.....	11
1.5.4	Control functions.....	12
1.5.5	Parameter channel	13
1.5.6	Error messages	14
1.5.7	Watchdog for network monitoring.....	14
2	Installation.....	15
2.1	Electrical connection	15
2.2	Controller enable (ENPO)	15
2.3	Assignment of device address by connector coding.....	15
3	Commissioning and configuration	16
3.1	Presets for control over CAN bus.....	16
3.2	Commissioning sequence	16
3.3	Commissioning instructions	17
3.4	Data handling	17
3.4.1	Saving settings	17
3.4.2	Restoring factory defaults	17
4	Setting the device parameters	18
4.1	VF1000 parameters for bus operation	18
4.1.1	82-SIOA - Device address for interface operation.....	18
4.1.2	40-TCAN - Sampling time monitoring.....	18
4.1.3	37-BCAN - CAN baud rate	18
4.1.4	90-FCAN - CAN function selector.....	19
4.1.5	60-ACAN - Analog output.....	20

4.2	MC6000/MC7000 parameters for bus operation	21
4.2.1	492-CACNF - CAN configuration	21
4.2.2	491-CACTR - Control word	21
4.2.3	490-CASTA - Status word	21
4.2.4	409-BUTWD - Bus watchdog time in ms	21
4.2.5	486-BUTCD - Maximum permissible cycle deviation relative to master	21
4.2.6	411-BUTCY - Bus sampling time	22
4.2.7	487-BUTCS - Sampling time of status message relative to "BUTCY"	22
4.2.8	488-BUSYE - Activating/deactivating synchronization	22
4.2.9	493-CAADR - CAN bus device address	22
4.2.10	489-CABDR - CAN bus baud rate	22
4.2.11	Automatic intervention in parameter setting of MC6000	23
4.3	Representation of parameter data	24
4.4	Representation of parameter number	28
4.5	Telegram execution and verification	29
4.6	Parameter channel	30
4.7	Field parameters (MC7000 only)	32
4.7.1	Access to individual field parameters	32
4.7.2	Access to field range	33
4.7.3	Reading field parameters	34
4.7.4	Reading string parameters	34
4.7.5	Writing string parameters	35
4.8	Handshake for downloading parameter data sets on the MC6000/MC7000 servocontroller	37
5	Control and reference input	39
5.1	System states	39
5.2	Device states	39
5.3	Device control	40
5.3.1	Terminal emulation	40
5.3.2	DRIVECOM state machine	40
5.3.3	Control word	41
5.3.3.1	Device control commands	41
5.4	Device status	42
5.4.1	Terminal emulation	42
5.4.2	Status word and device states	42
5.4.2.1	Device states	42
5.5	Identifiers	43
5.5.1	Selective transmissions	43
5.5.2	Broadcast transmissions	43
5.5.3	Station logon	43
5.5.4	System start/stop	44
5.5.5	Control functions	44
5.5.5.1	Control in Speed and Torque Control modes	46
5.5.5.2	Control in MC7000 modes:	47
5.5.5.3	Control word to control MC7000 PosMOD, electronic gearing and stepper motor interface	48
5.5.6	Control mechanisms, MC7000 PosMOD	49
5.5.6.1	Control enable	49
5.5.6.2	Automatic enable and start of sequence program	49
5.5.6.3	Feed hold and update	49

5.5.6.4	Jog+ and jog-.....	49
5.5.6.5	Program number.....	50
5.5.6.6	Reference run number.....	50
5.5.6.7	Table index	50
5.5.6.8	Flags and variables.....	51
5.5.7	Status messages	52
5.5.7.1	Inverter status messages.....	52
5.5.7.2	Servodrive status messages in modes: Speed and Torque Control.....	53
5.5.8	Status word for control of PosMOD over CAN	54
6	Response to device fault.....	56
6.1	Error messages	56
6.2	Acknowledgment of error messages.....	58
7	Examples	59
7.1	Activation of a VF1000 frequency inverter	59
7.2	Activation of an MC7000 in Speed Control mode	61
7.2.1	Terminal emulation control mode	61
7.2.2	Control mode: DRIVECOM state machine	62
7.3	Example: MC7000 PosMOD activation.....	65
7.4	Loading and deleting the positioning program of the positioning and sequence control	67
7.5	Example: Activation in "Electronic Gearing" mode	68

Appendix A: Glossary of terms

1 General introduction

This CAN bus documentation is applicable to inverter of the VF1000 series and all servocontrollers of the MASTERCONTROL series. The telegram structures of each of the two series are very similar, and so are documented together here. Both series can be operated together in a network.

Where the term "**device**" is used in general terms in the following, it refers both to frequency inverters of the SMARTDRIVE VF1000 series and to all servocontrollers of the MASTERCONTROL MC6000/MC7000 series.

The term "**master**" as used in the following designates a higher-order controller which organizes the bus system.

1.1 System requirements

Any system with a CAN interface is suitable. No requirements are set out in terms of processor speed, as the timeout monitoring functions on the devices can be adapted to the respective processor performance.

1.2 User level in operation over CAN bus

The CAN BUS interface always operates on the device at a relatively high user level. This means that parameters are accessible which cannot be accessed on the KEYPAD . Some of the parameters at those user levels are service parameters, and are not documented in the Operation Manuals of the individual devices.

Note: Unintentional write operations to such parameters may severely impair the functioning of the device!

1.3 Further documentation

- Parameter description of the relevant drive unit
- ISO 11898, Road Vehicles, Interchange of digital information - Controller Area Network (CAN) for high-speed communication
- CiA/DS20x : CAN Application Layer for Industrial Applications
- CiA/DS 102-1 : CAN Physical Layer for Industrial Applications - Part 1: Two Wire Differential Transmission

Exceptions:

1. If a 1 is entered in device parameter 04-PROG (for servos) or 71-PROG (for the VF1000), the device overwrites all the parameter settings with its default value. In this case the reply telegram is only sent when the complete parameter list has been reinitialized. This operation may take up to 10 seconds depending on the device.
2. During reading of the SMARTCARD no communication with the device is possible over the CAN bus. This condition may last up to 10 seconds depending on the device.
3. If an error state is acknowledged over the CAN bus, a device restart may result. For more detailed information on this subject refer to the section headed "Response to device fault".

1.4.6 Transmission speeds

The CAN bus can be operated at the following transmission speeds:

	Transmission speed	Maximum line length over the entire network
0	1 MBaud	40 m
1	800 KBaud	50 m
2	500 KBaud	100 m
3	250 KBaud	200 m
4	125 KBaud	450 m
5	75 KBaud	770 m
6	50 KBaud	1000 m
7	25 KBaud	1000 m

However, when selecting the transfer rate it must be ensured that the **line length** does not exceed the permissible line length for that transfer rate (cable length in device of approx. 30 cm must be taken into account).

The following factors influence calculation of the permissible line length:

- Propagation time of the signal on the line
- Signal propagation time of the optocouplers
- Signal propagation time of the gates

The line length values specified in the Operation Manuals for the VF1000, MC6000 and MC7000 devices already allow for the signal propagation times. For the control a signal propagation time from the bus connector to the CAN controller of max. 80 ns is assumed. If these control values are exceeded, the transfer rate must be reduced by at least one increment!

1.5 CAN protocol for LUST drives

The CAN protocol for LUST drives permits integration of the device in a CAN network. The identifiers are assigned in the devices by setting of the device address.

After power-up the device responds cyclically with its "logon identifier". From that identifier the higher-order controller can identify which devices are connected to the bus and which addresses have been assigned for the devices.

Note: Operation of two devices with the same address on the bus is not permitted.

Once a device has been addressed by the controller with an identifier applicable to that device, the device switches to status message send. As a result the controller detects that the device is connected to the network and is now ready for control and parameter-setting. For control of the devices, a protocol for selective control of each drive is available.

For synchronized starting/stopping of all drives a broadcast telegram is available.

1.5.1 System states

The "system state" describes the status of the overall bus system. The following system states are currently supported:

- **System Stop**

After power-on each device is in the System Stop state. In this state parameters can be set over the bus, or control commands and reference values can be transmitted to the individual devices. The control commands and reference values are only stored however (1 reference value / 1 control command) and are only executed in the system state **System Start**.

- **System Start**

System Start is the normal operating state. The devices can be controlled by way of their selective control commands. If control commands were transmitted to the devices during **System Stop**, they are only executed on transition to **System Start**. This behavior allows the individual devices to be preset before the overall system is up and running. Then, with **System Start** all the devices receive their start command absolutely synchronously.

1.5.2 Device states

In contrast to the system state, which describes the status of the overall bus system, the device states in the various devices of a bus system may differ.

The device state is determined, firstly, by the selective control commands over the bus and, secondly, by means of information from the respective process.

For example, an error in an application results in a change of device state.

The devices run a so-called state machine, which assigns defined responses to events for each state.

1.5.3 Device control

There are two different modes of controlling the devices over the CAN bus.

In the **first control mode** the control terminal function of the drives is emulated. The terminal emulation is available on all devices.

In the **second control mode** the device is controlled by the DRIVECOM state machine. This control mode is supported only by the servodrives.

Possible operation modes in use of CAN bus activation:

	VF1000	MC6000	MC7000
Open-loop speed control	□		
Closed-loop speed control		□	□
Closed-loop torque control		□	□
Closed-loop position control (positioning and sequence control)			□

Terminal emulation

The terminal emulation delivers control bits by way of the CAN control word which emulate the input terminals (e.g. STL/STR - Start enable or S1IND - prog. digital input) of the device. For more detailed information on the setting and availability of these functions, refer to the operation manual accompanying each device.

DRIVECOM control word

To control a device in the second control mode over CAN, the state machine defined in the DRIVECOM profile no. 22 of January 1994 for INTERBUS-S must be followed.

1.5.4 Control functions

Function: Control functions/reference
Data direction: Master -> Device
Type: selective

Control functions can be optimally adapted to the relevant application. Consequently, several control formats are offered. The appropriate formats can be selected by the master during the setup phase over the bus, or by adjusting the relevant device parameters.

A) Control functions for inverters

An inverter can receive and process control commands at the full transfer rate of 1 Mbaud. Since the internal state machine has a sampling time of 8 ms, these values enter the control cycle of the inverter every 8 ms. Data byte 2 contains the terminal emulation of the inverter.

Sampling time: 8 ms

Priority based on CAL	Base ID	Data byte 0	Data byte 1	Data byte 2
3	661	REF_LO	REF_HI	InBits 0 = STR 1 = STL 2 = S1IND 3 = S2IND 4 = S1OUT 5 = S2OUT 6 = S3OUT 7 = ERROR_RESET

B) Control functions for servos

The servo state machine has a sampling time of 1 ms. All control commands and reference values are processed within that sampling time by the servocontroller.

Sampling time: 1 ms

Selection of state control and reference input by way of parameter 492-CACNF.

CACNF	1	2	3	4	5
Reference 1	16 bits	32 bits	32 bits	32 bits	
Reference 2					
Actual 1	16 bits	16 bits	16 bits	32 bits	
Actual 2		16 bits	16 bits		
DRIVECOM	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>
Terminal emulation				<input type="checkbox"/>	

for MC6000/7000 modes:
 - Speed control
 - Torque control

only for MC7000 modes:
 - Positioning and sequence control
 - Electronic gearing

1.5.5 Parameter channel

A) Inverter parameter channel

Function: Parameter enquiry/transfer

The data in this transfer are scaled according to the stipulations in the inverter parameter list (see section 4.1).

For more detailed information on the setting and availability of these functions, refer to the parameter description which is available as a separate document for each device.

Data direction: Master -> FI

At this ID parameters are transferred or enquiries entered. Each transmission of this ID results in a reply with ID 1321.

Priority based on CAL	Base ID	Data byte 0	Data byte 1	Data byte 2	Data byte 3	Data byte 4
5	1101	PARA_LO	PARA_HI	Mode of transfer: "ENQ" "SEL"	DATA_LO	DATA_HI

B) Servo parameter channel

Function: Parameter enquiry/transfer

The data in this transfer are scaled according to the stipulations in the servo parameter list (see section 4.2).

Data direction: Master -> Servo

At this ID parameters are transferred or enquiries entered. Each transmission of this ID results in a reply with ID 1321.

Priority based on CAL	Base ID	Data byte 0	Data byte 1	Data byte 2	Data byte 3+4+5+6	Data byte 7
5	1101	PARA_LO	PARA_HI	Mode of transfer: "SEL" "ENQ"	DATA	COUNTER

1.5.6 Error messages

Messages are only sent in the system state "System Start".

Function: Error message
Data direction: Device > Master
Type: selective

A) For VF1000 inverters:

Priority based on CAL	Base ID	Data byte 0
2	441	Error number The number corresponds to the error number of the inverter (For definition see Operation Manual >> Error messages)

B) For MC6000/7000 servos:

Priority based on CAL	Base ID	Data byte 0	Data byte 1
2	441	Error number The number corresponds to the error number of the servo (For definition see Operation Manual >> Error messages)	Error location This number permits a more precise definition of the causes of error in servodrives.

1.5.7 Watchdog for network monitoring

All devices offer a facility for network monitoring by means of a programmable time monitor (watchdog). The response time (timeout) of the watchdog can be set by way of parameter 40-TCAN (for VF1000) and parameter 409-BUTWD (for MC6000 and MC7000) (for scaling see parameter description).

The watchdog monitors the device to check whether a valid telegram has been received within the preset time. If no such telegram has been received, the device switches to the error state with the relevant error message. Servocontrollers also offer the facility to program the response by parameter (e.g. Subject area _SCTY param. R-CAN).

The watchdog is reset in the same way as any other device error.

2 Installation

2.1 Electrical connection

The electrical connection of the power and control electronics is described in the relevant operation manuals of the devices used.

2.2 Controller enable (ENPO)

A) *On the VF1000 frequency inverter*

On the VF1000 frequency inverter no additional controller enable via control terminal is necessary in the case of control over the CAN bus.

B) *On the MC6000/MC7000 servocontroller*

On the servocontroller an additional hardware enable via control terminal X5/11 ENPO is required for control over the CAN bus. This control signal is high-active. When this control signal is removed the motor runs out freely. Also refer to the description in the MC6000/MC7000 servocontroller operation manual.

2.3 Assignment of device address by connector coding

Only for VF1000 frequency inverter.

The contact assignment for address assignment by means of connector coding is described in the relevant operation manuals of the devices used.

3 Commissioning and configuration

3.1 Presets for control over CAN bus

Control of devices over the CAN bus requires a number of parameter settings. The devices are factory configured for operation via terminals. It is possible to control them over the CAN bus after adjusting the relevant configuration parameters appropriately.

A) *VF1000 frequency inverter*

Presets:

- Parameter 01-MODE = 4 Control location interface
- Parameter 04-FSSEL = 25 (26) Reference source interface
- Parameter 37-BCAN = 2 Baud rate = 500 kB
- Parameter 82-SIOA = x Device address
- Parameter 40-TCAN = xx Watchdog

B) *Servocontroller MC7000*

Presets:

- Parameter 402-CLSEL = CAN Control location CAN
- Parameter 419-RSSL3 = CAN Reference source interface (only in Speed and Torque Control modes)
- Parameter 489-CABDR = 500 Baud rate = 500 kB
- Parameter 493-CAADR = x Device address
- Parameter 409-BUTWD = xx Watchdog

3.2 Commissioning sequence

This section outlines the steps for initial commissioning of a servodrive.

For more detailed information on optimizing the speed or position control circuit, refer to the device Operation Manual.

1. Wire up the device, encoder and motor.
Ensure the motor is connected to the correct phase.
2. Load motor data from the motor database of the DRIVEMANAGER user interface or from a SMARTCARD into the servocontroller.
3. Activate the operation mode you want in the device by way of the DRIVEMANAGER user interface (speed control, position control, ...).
4. Configure the device according to the application requirements.
(inputs/outputs, encoder simulation, position control, ...).
5. Test the control quality and optimize the controller settings (speed controller) as set out in the device Operation Manual.
6. Set the CAN-specific parameters; see below.
7. Test higher-order controller.
8. Save device configuration.

3.3 Commissioning instructions

For various reasons, it may be that a device fails to reply to a telegram:

- There is no reply if the telegram frame (baud rate, data length) on the master computer is not correct.
- There is no reply if a device is addressed by the wrong bus address.
- There is no reply if the serial link between the master computer and the device is not correctly set up.
- There is no valid reply if several devices with the same device address are connected to the bus.

3.4 Data handling

3.4.1 Saving settings

All configuration data, apart from sequence programs of the positioning and sequence control, can be backed up on a SmartCard or as a data file with the Drivemanager. Sequence programs can only be saved as data files on PC.

3.4.2 Restoring factory defaults

There are two way of restoring the factory defaults of the device parameter settings:

1. Set parameter PROG (VF1000: 71-PROG, MC6000/7000: 04-PROG) to the value 1
2. MC6000/7000 servocontrollers only: During power-up hold down the two cursor keys on the KeyPad KP100 control unit

4 Setting the device parameters

4.1 VF1000 parameters for bus operation

4.1.1 82-SIOA - Device address for interface operation

By way of this parameter the device address can be assigned. The parameter setting has priority over a hardware setting by DIP switch or address coding plug. A hardware setting is only adopted if 82-SIOA = 0 is set.

Default: 0
Setting range: 0 - 29

4.1.2 40-TCAN - Sampling time monitoring

The inverter checks the specified sampling times. If the sampling times are exceeded, the inverter switches to fault mode. With parameter TCAN the sampling time can be changed.

Scaling: 1 bit = 8.2 ms
Default: 3
Setting range: 0 - 255 (0 not permitted)

4.1.3 37-BCAN - CAN baud rate

Setting of the CAN baud rate; changes only take effect after mains power reset.

37-BCAN	Baud rate	Comments
0	1 MBaud	
1	800 KBaud	
2	500 KBaud	Factory setting
3	250 KBaud	
4	125 KBaud	
5	75 KBaud	
6	50 KBaud	
7	25 KBaud	

Default: 2
Setting range: 0 - 7

4.1.4 90-FCAN - CAN function selector

For selection of special functions:

90-FCAN	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
Function	-	-	-	-	-	Expanded status message	Brake ramp with /S1IND (emergency breakdown function)	Braking ramp with E_CAN

Default: 0
 Setting range: 0 - 255

Braking ramp with error message E_CAN

If bit 0 in 90-FCAN is set, when the error E_CAN (error no. 12) is detected a stop ramp is activated. The steepness of the braking ramp can be adjusted by way of parameter 36 - RSTOP. The drive runs down to 0 Hz.

E_CAN can only be reset when the drive is at a standstill.

Emergency-off function

If bit 1 in 90-FCAN is set, where S1IND = 0 the error F_STP (error no. 13) is generated. When this signal is activated the inverter runs down to a frequency of 0 Hz with a braking ramp, and switches to **System Stop**.

The steepness of the braking ramp can be adjusted by way of parameter 36 - RSTOP. Only when the frequency 0 Hz has been reached and the DC holding time has expired does the inverter again respond to new control commands.

Expanded status message

If bit 2 in 90-FCAN is set, the status word is expanded by 1 byte:

Priority	Base ID	Data byte 0	Data byte 1	Data byte 2	Data byte 3
4	881	IST_LO	IST_HI	OUT_BIT Bit 0 = STR 1 = STL 2 = S1IND 3 = S2IND 4 = S1OUT 5 = S2OUT 6 = S3OUT 7 = ERROR	Bit 0 = 1 = 2 = 3 = 4 = 5 = 6 = Terminal X2/25 7 = Terminal X2/24

The two terminals X2/24 and X2/25 are used in standard devices of the VF1000M/L series for the STR (Start Right) and STL (Start Left) function (VF1000S = terminals X1/3 & X1/4). Where 01-MODE = 4 (CAN operation), these inputs are used solely as inputs with no special function. The Start function in CAN operation is executed by way of the CAN control word.

4.1.5 60-ACAN - Analog output

By way of parameter 60-ACAN (Analog_CAN) a 16-bit value can be set for delivery via output SOUTA (VF1000M and VF1000L only).

Manipulating range 0 - 6000 HEX

0	HEX	==>	0 V
4000	HEX	==>	10 V
6000	HEX	==>	15 V

Note: To activate output of this value via SOUTA, it is necessary **to set parameter 61 - SOUTA to 16.**

4.2 MC6000/MC7000 parameters for bus operation

4.2.1 492-CACNF - CAN configuration

CACNF	Reference	Actual	Comments
0	No reference adopted	No actual transferred	CAN bus is switched off
1	16 bits, torque, speed or position	16 bits, torque, speed or position	
2	32 bits, torque, speed or position	32 bits, torque, speed or position	Factory setting
3	32 bits Speed	16 bits torque (1st word) 16 bits speed (2nd word)	
4	Format VF1400 (Sa) 32 bits frequency + 16 bit in-bits	Format VF1400 (Sa) 32 bits frequency + 16 bit out-bits	
5	No transfer of PosMod variable and flags to positioning and sequence control	Actual position (absolute) Actual position from positioning and sequence control in scaled travel units	

Scaling of 32-bit values:

Torque: $\text{Nm} \cdot 2^{-16}$
Speed: $\text{rpm} \cdot 2^{-16}$
Position: $\text{Revolutions} \cdot 2^{-16}$

Scaling of 16-bit values:

Torque: Nm
Speed: rpm
Position: Revolutions

4.2.2 491-CACTR - Control word

Current control word (CAN control), see section 5.3.3. Display value only!

4.2.3 490-CASTA - Status word

Current status word (CAN status), see section 5.4.2. Display value only!

4.2.4 409-BUTWD - Bus watchdog time in ms

Value for watchdog to monitor the CAN bus. The watchdog time is adjustable in millisecond increments. The value 0 deactivates the watchdog.

Default: 0
Setting range: 0 - 255 ms (0 = off)

4.2.5 486-BUTCD - Maximum permissible cycle deviation relative to master

Maximum permissible deviation of the internal cycle relative to the cycle of drive 1.

Scaling: 1 bit = 0.1 μs .
Default: 5000
Setting range: 1 - 32000

4.2.6 411-BUTCY - Bus sampling time

Sampling time of the CAN bus in microseconds.

This parameter only needs to be set if a time-equidistant input of the reference value is required.

Scaling: 1 bit = 1 μ s.
Default: 1000
Setting range: 100 - 32000

4.2.7 487-BUTCS - Sampling time of status message relative to "BUTCY"

BUTCS Bus_timer_statuscycle

Sampling time of the status message relative to "BUTCY"

Scaling: 1 = every cycle
2 = every second cycle
etc.

Default: 10
Setting range: 1 - 255

4.2.8 488-BUSYE - Activating/deactivating synchronization

With this parameter synchronization of the drive is activated. (0 = no synchronization)
Function not yet implemented!

4.2.9 493-CAADR - CAN bus device address

Address on the CAN bus. This parameter has priority over hardware settings!

Default: 0
Setting range: 0 - 29

MC6000 only: For address input by hardware means (DIP switch or DSUB connector), the parameter must be set to 0. 0 is the default value of the parameter.

Note: Operation of two devices with the same device address on a bus is not permitted.

4.2.10 489-CABDR - CAN bus baud rate

By way of this parameter the baud rate of the CAN controller is set:

CABDR	Transmission speed	Comments
0	1 MBaud	
1	800 KBaud	
2	500 KBaud	Factory setting
3	250 KBaud	
4	125 KBaud	
5	75 KBaud	
6	50 KBaud	
7	25 KBaud	

4.2.11 Automatic intervention in parameter setting of MC6000

When the MC6000 is switched on the CAN option is automatically detected. If the device was not previously equipped with the CAN option, the following parameters are automatically adjusted **once only** in the MC6000:

Parameter:	Value:	Comments
402-CLSEL	OPT1	Control location CAN option
417-RSSL1	RCON	Reference source off
418-RSSL2	RCON	Reference source off
419-RSSL3	OPT1	Reference source CAN option
420-RSSL4	RCON	Reference source off
439-FIS00	OPT1	Input assigned to CAN
440-FIS01	OPT1	Input assigned to CAN
445-FOS00	OPT1	Output assigned to CAN
446-FOS01	OPT1	Output assigned to CAN

When the CAN board is removed from the MC6000 these parameters are reset to their factory defaults on power-up.

4.3 Representation of parameter data

The parameters are set by way of the parameter channel described in sections 1.5.5 and 4.6.

The parameter data are transmitted in binary format within a CAN data block. The parameter data begin with data byte 3. On the VF1000 they have a max. length of 2 bytes and on the servo 4 bytes.

Interpretation of the data transferred in the data block differs depending on parameter data type. A list of all device parameters with their respective data types is obtainable in a separate document from LUST Antriebstechnik GmbH.

A) *Parameter data types of the inverters*

The VF1000 inverters support the following parameter data formats:

Data type	Scaling	Value range	KP100 display	MC6000/7000-compatible
PT_USIGN8	1	0 .. 255	00H .. FFH	Value range PT_USIGN8
PT_FIXPOINT16	0.05	0.00 .. 3276.80	0.00 .. 999.95	yes
PT_INTEGER16	1	-32768 .. 32767	-9999 .. 32767	yes
PT_INTEGER8	1	-128 .. 128	-128 .. 128	no
PT_VF_ERROR	1	0 .. 65535	Plain text e.g. "E-OV"	no
PT_PASSWORD	1	0 .. 65535	0 .. 65535	yes

Notes on individual data types:

Data type PT_USIGN8:

The data type PT_USIGN8 is an 8-bit integer without preceding sign and is represented on the display of the KEYPAD in hexadecimal format, as it is usually used to represent bit fields.

Data type PT_PASSWORD:

Passwords are represented as 16-bit integer values without preceding sign.

Data type PT_VF_ERROR:

This data type corresponds to a structured 16-bit integer. The Low byte designates the error number (0 - 15) and the High byte the error time (0-15h) - that is, the value of the operating hours meter when the error occurred.

Structure element	Error number	Error time
Data length	1 byte	1 byte

Notes on use of the individual data types in C

In the following examples it is assumed that the CAN data block has been imported from the CAN controller into the RAM of the master. The parameter data start from the memory location *Data byte[3]* in the RAM.

If the same physical address is accessed with different data types in C, this is best done by way of "union" structures.

The following example demonstrates data accessing via the "union":

```
/*-----*\
|
|   Example program for parameter data access within
|   the CAN data block
|
\*-----*/

/*-----*\
|   Externals
\*-----*/
extern unsigned char Data byte[];          /* The Can-
data block */

/*-----*\
|   Definitions
\*-----*/

typedef struct                               /* Structure of the error data */
{
    unsigned char Number;
    unsigned char Time;
}VF_Error;

typedef union                                /* Union for parameter data access */
{
    signed char int8;
    signed int int16;
    unsigned char usign8;
    unsigned int usign16;
    VF_Error err16;
} VF_Data;

/*-----*\
|   Function MAIN
\*-----*/
void main( void)
{
    /*-----*\
    |   Test variables for data exchange
    \*-----*/
    signed char TestInt8;
    signed int TestInt16;
    unsigned char TestUsign8;
    unsigned int TestUsign16;
    float TestFix16;
    unsigned char ErrorTime;
    unsigned char ErrorNumber;
    VF_Data *CanParaData;

    /*-----*\
    |   Read access to the parameter data
    \*-----*/
    /* Position data pointer */
    CanParaData = &(amp;Data byte[ 3]);

    /* Data access PT_UISGN8 */
    TestUsign8 = CanParaData->usign8;

    /* Data access PT_PASSWORD */
    TestUsign16 = CanParaData->usign16;

    /* Data access PT_INTEGER8 */
    TestInt8 = CanParaData->int8;

    /* Data access PT_INTEGER16 */
    TestInt16 = CanParaData->int16;
}
```



```

/* Data access PT_FIXPOINT16 */
TestFix16 = (float)( CanParaData->usign16) / 20.;

/* Data access PT_VF_ERROR */
ErrorTime = CanParaData->err16.Time;
ErrorNumber = CanParaData->err16.Number;

/*-----*\
|      Write access to the parameter data
\*-----*/
/* Place the value 3 on a parameter of type PT_USIGN8 */
CanParaData->usign8 = 3;

/* Place the value 3 on a parameter of type PT_BINARY16 */
CanParaData->usign16 = 3;

/* Place the value -3 on a parameter of type PT_INTEGER8 */
CanParaData->int8 = -3;

/* Place the value -3 on a parameter of type PT_INTEGER16 */
CanParaData->int16 = -3;

/* Place the value 326.15 on a parameter of type PT_FIXPOINT16 */
CanParaData->usign16 = (unsigned int)(326.15 * 20.);

/* PT_VF_ERROR are read-only*/
}

```

B) *Parameter data types of the servos*

The servocontrollers of the MASTERCONTROL series support the following parameter data formats:

Data type	Scaling / Increment	Value range	KP100 display	VF1000-compatible
PT_USIGN8	1	0 .. 255	0 .. 255	Value range PT_BINARY8
PT_USIGN16	1	0 .. 65535	0 .. 65535	no
PT_FIXPOINT16	0.05	0.00 .. 3276.80	0.00 .. 999.95	yes
PT_INTEGER16	1	-32768 .. 32767	-9999 .. 32767	yes
PT_INTEGER32	1/65536	-32767.99 .. 32766.99	-9999 .. 32767	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 the data types:

Data type PT_INTEGER32

This data type is a "signed long" (32-bit) type with scaling 1/65536. It is displayed on the KEYPAD in the same way as a PT_FLOAT_IEEE.

Data type PT_FLOAT_IEEE

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

Data type PT_MC_ERROR

This data type is structured and is 32 bits long. Its structure is as follows:

Structure element	Error number	Error location	Error time
Data length	1 byte	1 byte	2 bytes

Notes on use of the individual data types in C

In the following examples it is assumed that the CAN data block has been imported from the CAN controller into the RAM of the master. The parameter data start from the memory location *Data byte[3]* in the RAM.

If the same physical address is accessed with different data types in C, this is best done by way of "union" structures.

The following example demonstrates data accessing via the "union":

```
/*-----*\
|
|   Example program for parameter data access within
|   the CAN data block
|
\*-----*/

/*-----*\
|
|   Externals
|
\*-----*/
extern unsigned char Data byte[];          /* The Can-
data block */

/*-----*\
|
|   Definitions
|
\*-----*/

typedef struct                          /* Structure of the error data */
{
    unsigned char Number;
    unsigned char Location;
    unsigned int Time;
}MC_Error;

typedef union                            /* Union for parameter data access */
{
    unsigned char usign8;
    unsigned int usign16;
    signed int int16;
    signed long int32;
    float float32;
    MC_Error err32;
} MC_Data;

/*-----*\
|
|   Function MAIN
|
\*-----*/
void main( void)
{
    /*-----*\
    |
    |   Test variables for data exchange
    |
    \*-----*/
    signed int   TestInt16;
    unsigned char TestUsign8;
    unsigned int TestUsign16;
    float TestFloat32;
    float TestFix16;
    float TestInt32Q16;
    unsigned int ErrorTime;
    unsigned char ErrorLocation;
    unsigned char ErrorNumber;
    MC_Data *CanParaData;

    /*-----*\
    |
    |   Read access to the parameter data
    |
    \*-----*/
    /* Position data pointer */
    CanParaData = &(amp;Data byte[ 3]);

    /* Data access PT_USIGN8 */
    TestUsign8 = CanParaData->usign8;

    /* Data access PT_PASSWORD or PT_USIGN16 */
    TestUsign16 = CanParaData->usign16;

    /* Data access PT_INTEGER16 */
```

```

TestInt16 = CanParaData->int16;

/* Data access PT_INTEGER32 */
TestInt32Q16 = (float)( CanParaData->int32) / 65536.;

/* Data access PT_FIXPOINT16 */
TestFix16 = (float)( CanParaData->usign16) /20.;

/* Data access PT_MC_ERROR */
ErrorTime = CanParaData->err32.Time;
ErrorNumber = CanParaData->err32.Number;
ErrorLocation = CanParaData->err32.Location;

/* Data access PT_MC_ERROR */
TestFloat32 = CanPara->float32;

/*-----*\
|      Write access to the parameter data
\*-----*/
/* Place the value 3 on a parameter of type PT_USIGN8 */
CanParaData->usign8 = 3;

/* Place the value 3 on PT_USIGN16 or PT_PASSWORD */
CanParaData->usign16 = 3;

/* Place the value -3 on a parameter of type PT_INTEGER16 */
CanParaData->int16 = -3;

/* Place the value 2345.3456 on a parameter of type PT_INTEGER32 */
CanParaData->int32 = (signed long)( 2345.3456 * 65536.);

/* Place the value 23.15 on a parameter of type PT_FIXPOINT16 */
CanParaData->usign16 = ( unsigned int)( 23.15 * 20.);

/* PT_MC_ERROR is read-only */

/* Place the value 2345.1234 on a parameter of type PT_FLOAT_IEEE */
CanPara->float32 = 2345.1234;
}

```

4.4 Representation of parameter number

The parameter number (PARA_HI PARA_LO) is represented as a four-digit hexadecimal number. The coding of these four hexadecimal numbers has different meanings for inverters and servocontrollers. The device-specific descriptions follow.

A) *Parameter numbers for inverters*

Frequency inverters have parameter numbers from 0 - 99. These parameters must be converted into four-digit hexadecimal numbers and inserted in the protocol frame under PARA_HI and PARA_LO, with PARA_LO representing the Low byte and PARA_HI the High byte of the parameter number. Leading zeroes must be entered.

B) *Parameter numbers for servocontrollers*

Servocontrollers have parameter numbers from 0 - 999. These parameters must be converted into four-digit hexadecimal numbers and inserted in the protocol frame under PARA_HI and PARA_LO, with PARA_LO representing the Low byte and PARA_HI the High byte of the parameter number. Leading zeroes must be entered.

4.5 Telegram execution and verification

Data transfers are acknowledged by reply telegram which contain the same data content and have the same parameter number. Only data byte 2 is different in the reply, containing the SIO STATUS instead of the mode of transfer.

The SIO STATUS indicates whether the transfer was successful, or what problems occurred if any.

In general a reply is only sent after successful entry of the new parameter value in the device. Since the new values of **RAM parameters** can be adopted directly, there is no delay in sending of the reply telegram for that parameter group.

In the case of **EEPROM parameter** inverters of the SMARTDRIVE series delay the reply telegram until the parameter value has been stored free of errors in the EEPROM (approx. 40 ms).

Servocontrollers of the MASTERCONTROL series have a RAM memory location for each parameter and an additional EEPROM memory location for each EEPROM parameter. After a SELECT telegram relating to an EEPROM parameter the value is first entered in the RAM and then the ACK is immediately sent to the master computer - that is to say, the new parameter value is available immediately.

The save operation to the EEPROM is executed in the background, and does not delay the main program. To store one byte in the EEPROM takes the servocontroller 15 ms. In the case of several Select telegrams relating to EEPROM parameters in sequence, the EEPROM write routine may be overloaded. The servocontroller signals this state by setting the relevant bit in parameter 85-SERR. In this case the corresponding SELECT telegram must be repeated until the servocontroller indicates that the new parameter value has been adopted with ACK.

Mode-related access restrictions

If in a reply telegram bit 6 (hexadecimal value 40H) of SIO_STATUS is set, write access to that parameter has been refused regardless of the transferred value.

This, on other hand, does not necessarily mean that the parameter is generally write-protected. Write access may have been refused only on the basis of the current operating state of the device. On inverters of the SMARTDRIVE series, for example, EEPROM parameters are generally write-protected when the control is active.

Note: To find out which parameters are accessible when, refer to the operation manual of the relevant device.

4.6 Parameter channel

Function: Parameter setting
Data direction: Master -> Device
 Device -> Master
Type: selective

All device parameters can be addressed by way of these identifiers. These transfers are processed at a low priority level in the device.

A) *Inverter parameter channel*

Parameter enquiry/transfer

The data in this transfer are scaled according to the stipulations in the inverter parameter list. For more detailed information on the setting and availability of these functions, refer to the parameter description which is available as a separate document for each device.

Data direction: Master -> FI

At ID 1101 parameters are transferred or enquiries entered. Each transmission of this ID results in a reply with ID 1321.

Priority based on CAL	Base ID	Data byte 0	Data byte 1	Data byte 2	Data byte 3	Data byte 4
5	1101	PARA_LO	PARA_HI	Mode of transfer: "ENQ" "SEL"	DATA_LO	DATA_HI

Data request: ENQ = 05H
 Data transfer: SEL = 02H

Data direction: FI -> Master

Priority based on CAL	Base ID	Data byte 0	Data byte 1	Data byte 2	Data byte 3	Data byte 4
6	1321	PARA_LO	PARA_HI	SIO STATUS 0 = Transfer OK	DATA_LO	DATA_HI
				Function of bits:		
				0 = Power on 1 = SIO Watchdog 2 = Transfer mode unknown 3 = Read not permitted 4 = NN 5 = Parameter unknown 6 = Change not permitted 7 = Impermissible value		

B) Servo parameter channel

Parameter enquiry/transfer 130

The data in this transfer are scaled according to the stipulations in the inverter parameter list.

Data direction: Master -> Servo

At ID 1101 parameters are transferred or enquiries entered. Each transmission of this ID results in a reply with ID 1321.

Priority based on CAL	Base ID	Data byte 0	Data byte 1	Data byte 2	Data byte 3+4+5+6	Data byte 7
5	1101	PARA_LO	PARA_HI	Mode of transfer: See below	DATA	Counter or Index for field parameters

Transfer mode	Value (Hex)	Description
ENQ	(05)	Request standard parameter
SEL	(02)	Write standard parameter
ENQuery_List	(04)	Request parameter data description
SElect_String	(08)	Write string parameter
SElect_Field	(09)	Write field parameter
ENQuery_Field,	(10)	Read field parameter
ENQuery_String	(11)	Read string parameter
SElect_Single Element	(12)	Write individual field parameter
ENQuery_Single Element	(13)	Read individual field parameter
ENQuery_Para_Text	(14)	Read value substitution text
ENQuery_Trans	(15)	Read transient memory
List_End	(16)	Universal List-End identifier

PARA_LO: Parameter number Low byte

PARA_HI: Parameter number High byte

DATA: 32-bit data
(at "List-End" : Checksum)

COUNTER: Block counter for data lengths > 4 bytes, e.g. string parameter
(incremented on every transmission)

The data byte is used as the subindex when accessing an individual field parameter. Field parameters always have a data length < 4 bytes.

When accessing standard parameters always set the data byte to zero!

Data direction: SERVO -> Master

Priority based on CAL	Base ID	Data byte 0	Data byte 1	Data byte 2	Data byte 3+4+5+6	Data byte 7
6	1321	PARA_LO	PARA_HI	SIO STATUS 0 = Transfer OK	DATA	Counter or Index for field parameters
				Function of bits:		
				0 = Power on 1 = SIO Watchdog 2 = Transfer mode unknown 3 = Read not permitted 4 = Repeat action 5 = Parameter unknown 6 = Change not permitted 7 = Impermissible value		

PARA_LO: Parameter number Low byte

PARA_HI: Parameter number High byte

DATA: 32-bit data (at "List-End" : Checksum)

COUNTER: Block counter for data lengths > 4 bytes, e.g. string parameter (incremented on every transmission).

The data byte is used as the subindex when accessing an individual field parameter. Field parameters always have a data length < 4 bytes.

4.7 Field parameters (MC7000 only)

All variables, flags, table positions and counters of the positioning and sequence control are stored in the device as field parameters. This means, for example, that PosMOD variables are accessible under parameter 528-POVAR in the device. Individual elements of this parameter are addressed by way of a subindex.

Example: Variable H10 = 528-POVAR (Subindex 10)

By way of the parameter channel of the servo, individual field parameters (SElect_Single Element) or a range of parameters (SElect_Field) can be accessed.

4.7.1 Access to individual field parameters

Individual field parameters can be accessed with a single protocol. For this, the parameter number of the field is entered in data bytes 0+1, the mode of transfer in data byte 2 (12 - write, 13 - read), the value in data bytes 3-6 and the number of the element (subindex) in data byte 7.

Example: Set variable H11 = 10 (528_{Dec} = 0210_{Hex})

ID	Data byte 0	Data byte 1	Data byte 2	Data byte 3-6	Data byte 7
1101	10	02	c	0a 00 00 00	b

Note: Waiting time when using this protocol type until the next telegram in the servocontroller approx. 30 ms maximum, typically 10 ms.

4.7.2 Access to field range

If data are to be transferred into a field parameter, the first telegram (data byte 7 = 0) contains the information on the field index and the number of fields to be written.

In the following telegrams one element per data block is transferred. Each data block is confirmed by a reply from the servo or rejected with an error in the status word.

The last data block sent contains a 32-bit checksum formed by the logical XOR link between the data areas of all individual telegrams including the first (index and length). Consequently, the number of individual telegram blocks is calculated as:

$$\text{BlockCount} = \text{Size} + 2$$

where Size = number of elements to be written

Note: Waiting time between the individual telegram blocks in the servocontroller approx. 1 ms.

Telegram sequence:

1. Block:

Master

Data byte (selector)	2	Data bytes (data)	3+4+5+6	Data byte (counter)	7
SEL_Field=		3+4=Index, 5+6=Size		0	

Servo

Data byte (selector)	2	Data bytes (data)	3+4+5+6	Data byte (counter)	7
Status		3+4=Index, 5+6=Size		0	

2nd block:

Master

Data byte (selector)	2	Data bytes (data)	3+4+5+6	Data byte (counter)	7
SEL_Field =		Individual data element		1	

Servo

Data byte (selector)	2	Data bytes (data)	3+4+5+6	Data byte (counter)	7
Status		Individual data element		1	

Last block:

Master

Data byte (selector)	2	Data bytes (data)	3+4+5+6	Data byte (counter)	7
LIST_End		32-bit checksum		Number of blocks	

Servo:

Data byte (selector)	2	Data bytes (data)	3+4+5+6	Data byte (counter)	7
Status		32-bit checksum		Number of blocks	

4.7.3 Reading field parameters

In the first telegram block the master transmits the field data (Index, Size) of the parameter of which the values are to be read. The servo confirms this request and enters in the reply telegram the number of readable elements as from the index, if the number of elements requested by the master is greater than the number of readable elements.

Then the servo sends the requested field elements in series. Only one element per telegram is ever transmitted.

In the last telegram (with the LIST_End identifier) a 32-bit checksum is sent by logical XOR linking of all data blocks including the first block (field description).

The number of telegram blocks is calculated as:

$$\text{BlockCount} = \text{Size} + 2$$

where Size = number of elements to be read

Note: Waiting time between the individual telegram blocks in the servocontroller approx. 1 ms.

Telegram sequence

1st block:

Master		
Data byte 2 (selector)	Data bytes 3+4+5+6 (data)	Data byte 7 (counter)
ENQ_Field=	3+4=Index, 5+6=Size	0
Servo		
Data byte 2 (selector)	Data bytes 3+4+5+6 (data)	Data byte 7 (counter)
Status	3+4=Index, 5+6=Size	0

2nd block:

Master		
Data byte 2 (selector)	Data bytes 3+4+5+6 (data)	Data byte 7 (counter)
ENQ_Field=	xxxx	1
Servo		
Data byte 2 (selector)	Data bytes 3+4+5+6 (data)	Data byte 7 (counter)
Status	Individual data element	1

Last block:

Master		
Data byte 2 (selector)	Data bytes 3+4+5+6 (data)	Data byte 7 (counter)
ENQ_Field=	xxxx	Number of blocks
Servo:		
Data byte 2 (selector)	Data bytes 3+4+5+6 (data)	Data byte 7 (counter)
LIST_End	32-bit checksum	Number of blocks

4.7.4 Reading string parameters

The master registers in data byte 2 (mode of transfer) that it wants to read a string parameter. The contents of data bytes 3-6 are not relevant.

The block counter (data byte 7) contains the value 0. The reply telegram from the servo contains the status byte and gives information on the readability of the parameter. If the parameter is readable, the first reply telegram contains the first four characters.

For synchronization purposes the master requests each substring by means of a request telegram. The servo copies the received telegram to its send area, overwrites the data area of that telegram with the string data and sends it back to the master. The last but one reply telegram from the servo contains the 0-terminator of the string in the data area.

In the last reply telegram the servo enters the LIST_End identifier in the status byte. The data area of the telegram now contains the checksum covering all individual data areas (0 to number of blocks -1) of the servo reply telegrams which have contained part of the data string.

If the string length = 4, the 2nd block is the last block and then already contains the checksum.

Telegram sequence

1st block:

Master

Data byte (selector)	2	Data bytes (data)	3+4+5+6	Data byte 7 (counter)
ENQ_String=		xxxx		0

Servo

Data byte (selector)	2	Data bytes (data)	3+4+5+6	Data byte 7 (counter)
Status		4 characters where stat= 0		0

Where status = 0 and string longer than 3 characters, 2nd block:

Master

Data byte (selector)	2	Data bytes (data)	3+4+5+6	Data byte 7 (counter)
ENQ_String		xxxx		1

Servo

Data byte (selector)	2	Data bytes (data)	3+4+5+6	Data byte 7 (counter)
Status		4 characters		1

Last but one block

Master

Data byte (selector)	2	Data bytes (data)	3+4+5+6	Data byte 7 (counter)
ENQ_String		xxxx		Number of blocks - 1

Servo

Data byte (selector)	2	Data bytes (data)	3+4+5+6	Data byte 7 (counter)
Status		0-3 char., 0-terminator		Number of blocks - 1

Last block:

Master

Data byte (selector)	2	Data bytes (data)	3+4+5+6	Data byte 7 (counter)
ENQ_String		xxxx		Number of blocks

Servo

Data byte (selector)	2	Data bytes (data)	3+4+5+6	Data byte 7 (counter)
LIST_End		32-bit checksum		Number of blocks

Plausibility checks:

After the first block the existence and access rights of the parameter are checked. If the servo inserts an error in the status of the reply telegram of block 1, communication is terminated for that parameter.

The last telegram contains the checksum of the overall string. If it is wrong, or if the counter is incorrect, the transmitted string is not valid.

The telegram sequence within the servo is executed according to a state machine. This state machine is automatically reset if the master transmits a telegram with an incorrect value for the block counter.

4.7.5 Writing string parameters

The master enters in data byte 2 the code for "Write string parameter". Data bytes 3-6 are of no significance in block 0. In the reply telegram from the servo in block 0 a status message is entered in data byte 2. If the status is 0, write access is permitted and the master begins sending the first string. Status != 0 terminates the communication.

The master transmits 4 characters per data block. The last but one data block must contain a 0-terminator. For the purpose of synchronization, the servo returns each data block to the master.

Explanatory note on checksum: Blocks 1 to n - 1 by logical XOR linking of data bytes 3 - 6.

Telegram sequence

1st block

Master

Data byte 2 (selector)	Data bytes 3+4+5+6 (data)	Data byte 7 (counter)
SEL_String=	xxxx	0

Servo

Data byte 2 (selector)	Data bytes 3+4+5+6 (data)	Data byte 7 (counter)
Status	xxxx	0

If status = 0 then 2nd block:

Master

Data byte 2 (selector)	Data bytes 3+4+5+6 (data)	Data byte 7 (counter)
SEL_String	4 characters	1

Servo

Data byte 2 (selector)	Data bytes 3+4+5+6 (data)	Data byte 7 (counter)
Status	4 characters	1

Last but one block:

Master

Data byte 2 (selector)	Data bytes 3+4+5+6 (data)	Data byte 7 (counter)
SEL_String	0-3 char., 0-terminator	Number of blocks -1

Servo

Data byte 2 (selector)	Data bytes 3+4+5+6 (data)	Data byte 7 (counter)
Status	0-3 char., 0-terminator	Number of blocks - 1

Last block:

Master

Data byte 2 (selector)	Data bytes 3+4+5+6 (data)	Data byte 7 (counter)
LIST_End	32-bit checksum	Number of blocks

Servo

Data byte 2 (selector)	Data bytes 3+4+5+6 (data)	Data byte 7 (counter)
Status	32-bit checksum	Number of blocks

Plausibility checks:

After data block 1 the servo checks the access authorization to the parameter. If the status is unequal to 0, communication is terminated for the parameter.

If the master transmits more than 100 characters for a string, the string is not saved and the error is entered at the end of the block sequence in the status (bit 7 = 1).

If the checksum in the last telegram is not identical with the original, a repetition of the telegram is requested by setting of bit 4 in the status.

4.8 Handshake for downloading parameter data sets on the MC6000/MC7000 servocontroller

Problem:

A unified, valid data set - that is, not just individual parameters - needs to be transmitted from the master computer to the servocontroller. On every transmission of an individual parameter the servocontroller checks whether the parameter matches its existing data set.

The check of the new parameter value in part adds existing parameter values. This creates the possibility that the servocontroller may reject a parameter even though it originates from a valid parameter data set. Possible error messages are E-PLS and E-PAR.

E-PLS	Plausibility error	->	Parameter settings not mutually plausible (control parameters)
E-PAR	Parameter setting error	->	Parameter settings in the reference structure are mutually exclusive

Remedy:

The new parameter data set of the master computer is transmitted to the servocontroller without individual checking of the parameter values. When the upload is complete the servocontroller checks the now completed new data set for plausibility. If the data are not logical, the entire data set is rejected and the old data set is reactivated.

This procedure requires a handshake, which is described in more detail in the following.

Important: In this action only parameters having the attribute "CardWriteable" are changed. Consequently, the upload of a parameter data set by way of the serial interface runs in the same way as by way of the SMARTCARD. If, during the upload, a Select telegram is sent to a parameter without the "CardWriteable" attribute, the servocontroller does reply to the telegram with "Acknowledge", but does not accept the new parameter value.

Handshake to upload a complete parameter data set

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

A write operation to this parameter is only possible when the system is at a standstill. After the write operation the servocontroller is secured against being switched on until the download is completed.

2. Transfer complete parameter data set

With several Select telegrams the individual parameters are transmitted from the master computer to the servocontroller. The servocontroller initially accepts the new parameter values without carrying out a plausibility check.

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

When all parameter data have been transmitted, the master computer sets SLOAD to the value (-2). This signals the end of data transfer to the servocontroller. The servocontroller then begins checking its entire data set for plausibility. If the data set is valid, the parameters are accepted with the attribute "CardWriteable" into the EEPROM. The drive is enabled again and can be started. The parameter 80-SLOAD is set according to the result of the parameter check.

4. Poll parameter 80-SLOAD with timeout (10s)

If SLOAD becomes 0 within the timeout the transfer was completed correctly. The parameters are accepted into the EEPROM with the attribute "CardWriteable". The drive is enabled again and can be started.

If SLOAD = (-1) within the timeout, the servocontroller is still busy checking and saving. If SLOAD > 0, the servocontroller has rejected the data set.

The value of SLOAD then corresponds to the number of the first parameter of which the value is invalid.

Important: If the connection is broken during transmission, or if the timeout is reached, the transmission must be repeated or the servocontroller restarted.

5 Control and reference input

5.1 System states

The "system state" describes the status of the overall bus system. The following system states are currently supported:

- **System Stop**

After power-on each device is in the System Stop state.

In this state parameters can be set over the bus, or control commands and reference values can be transmitted to the individual devices. The control commands and reference values are only stored however (1 reference value / 1 control command) and are only executed in the system state **System Start**.

- **System Start**

System Start is the normal operating state. The devices can be controlled by way of their selective control commands. If control commands were transmitted to the devices during **System Stop**, they are only executed on transition to **System Start**. This behavior allows the individual devices to be preset before the overall system is up and running. Then, with **System Start** all the devices receive their start command absolutely synchronously.

5.2 Device states

In contrast to the system state, which describes the status of the overall bus system, the device states in the various devices of a bus system may differ.

The device state is determined, firstly, by the selective control commands over the bus and, secondly, by means of information from the respective process. For example, an error in an application results in a change of device state.

The devices run a so-called state machine, which assigns defined responses to events for each state.

5.3 Device control

There are two different modes of controlling the devices over the CAN bus.

In the **first control mode** the control terminal function of the drives is emulated. The terminal emulation is available on all devices.

In the **second control mode** the device is controlled by the DRIVECOM state machine. This control mode is supported only by the servodrives.

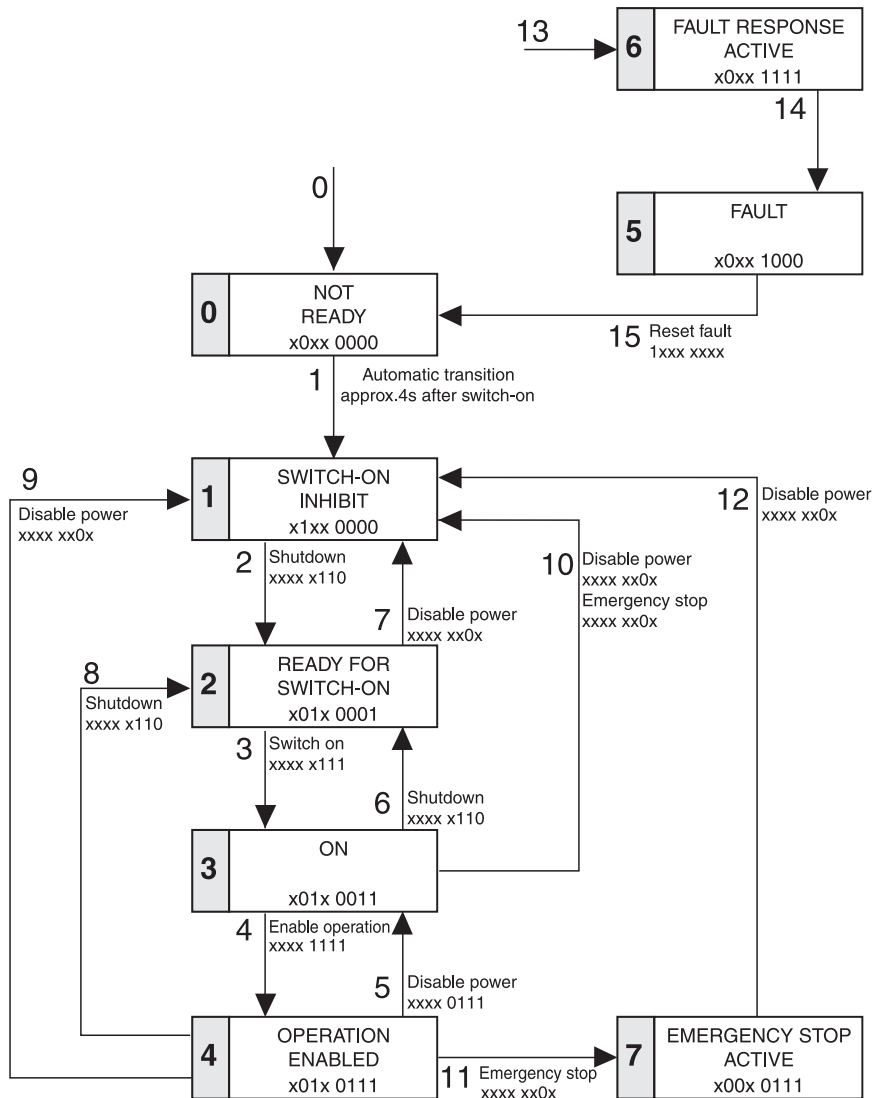
5.3.1 Terminal emulation

The terminal emulation delivers control bits by way of the CAN control word which emulate the input terminals (e.g. STL/STR - Start enable or S1IND - Prog. digital input) of the device.

For more detailed information on the setting and availability of these functions, refer to the operation manual accompanying each device.

5.3.2 DRIVECOM state machine

To control a device in the second control mode over CAN, the state machine defined in the DRIVECOM profile no. 22 of January 1994 for INTERBUS-S must be followed.



5.3.3 Control word

The 16 bits of the control word result from the logical linking of control commands which act on the state machine. The following bits of the DRIVECOM control word are supported:

Bit	Name	Comments
0	Switch-on	
1	Disable power	
2	Emergency stop	
3	Operation enabled	
4	Mode-dependent	Device or mode dependent assignment
5	More detailed definition:	
6	DRIVECOM profile no. 22 from January 1994	
7	Reset fault	
8	reserve	Device or mode dependent assignment
9	reserve	
10	reserve	
11	vacant	
12	vacant	
13	vacant	
14	Reference state output OS00 ¹⁾ , S1OUT ²⁾	
15	Reference state output OS01 ¹⁾ , S2OUT ²⁾	

¹⁾MCxxxx, ²⁾VF1xxx

5.3.3.1 Device control commands

The following bit combinations form the device control commands:

Command:	Control bit					Transitions:
	7	3	2	1	0	
SHUTDOWN	X	X	1	1	0	2, 6, 8
POWER-UP	X	X	1	1	1	3
DISABLE POWER	X	X	X	0	X	7, 9, 10, 12
EMERGENCY STOP	X	X	0	1	X	11
DISABLE OPERATION	X	0	1	1	1	5
ENABLE OPERATION	X	1	1	1	1	4
RESET FAULT	0⇒1	X	X	X	X	15

Transitions 4 and 5 are also influenced by 'System Start Stop'!

5.4 Device status

Depending on the control mode in which the device is operated, the relevant mode for the status message is automatically selected.

5.4.1 Terminal emulation

In terminal emulation you get the information available on the device control terminal as an image in a status word.

For more detailed information on the setting and availability of these functions, refer to the operation manual accompanying each device.

5.4.2 Status word and device states

In the status word the current state of the device and additional messages are displayed. The following bits of the DRIVECOM status word are supported:

Bit	Name	Comments
0	Ready for start	
1	On	
2	Operation enabled	
3	Fault	
4	Power disabled	
5	Emergency stop	
6	Switch-on inhibit	
7	Warning	
8	No function	
9	Remote	
10	Reference reached	Device or mode dependent assignment
11	Limit value	
12	Mode-dependent	
13	More detailed definition: DRIVECOM profile no. 22 from January 1994	
14	Actual state input IS00 ¹⁾ , S1IND ²⁾	
15	Actual state input IS01 ¹⁾ , S2IND ²⁾	

1) MCxxx, ²⁾VF1xxx

5.4.2.1 Device states

The following bit combinations form the device states:

State:	Status bit					
	6	5	3	2	1	0
NOT READY	0	X	0	0	0	0
SWITCH-ON INHIBIT	1	X	0	0	0	0
READY	0	1	0	0	0	1
ON	0	1	0	0	1	1
OPERATION ENABLED	0	1	0	1	1	1
FAULT	0	X	1	0	0	0
FAULT RESPONSE ACTIVE	0	X	1	1	1	1
EMERGENCY STOP ACTIVE	0	0	0	1	1	1

5.5 Identifiers

5.5.1 Selective transmissions

For communication between the various CAN bus stations a "base" CAN identifier is defined for each data transfer.

Each station on the bus is assigned a number (0 - 29) which can be set on the devices in a variety of ways:

On the VF1000: By way of a DIP switch, coding on the D_SUB connector or parameter 82-SIOA

On the MCxxxx: By way of parameter 493-CAADR

Setting by way of parameter has priority. Only if the address set in the parameter is 0 does the VF1000 accept the hardware setting.

Station 0 operates with the "base" CAN identifier. All other stations operate with identifiers calculated according to the following formula:

$$\text{ID} = (\text{"base" CAN identifier}) + 2 * (\text{station number})$$

5.5.2 Broadcast transmissions

Broadcast transmissions are received and evaluated by all devices simultaneously. The 'Remote Transmission Request' flag must not be set for these transfers. No reply is given to such transmissions.

A broadcast transmission may be sent by only **one** bus user.

5.5.3 Station logon

Function: Log on system after power ON

Data direction: Device -> Master

Type: selective

This message is also delivered in event of **System Stop**

Priority based on CAL	Base ID	Data byte
7	1543	No data

After Power On each bus user attempts to log on to the master.

The device transmits this identifier with a sampling time of 100 ms. The master identifies from the identifiers which devices are connected to the bus and which address is assigned to the devices concerned.

The identifier is transmitted until the device has been addressed once by the master over the bus (function: **System Start**).

When the master has addressed the device over the bus assigned to the device by means of an identifier, the device detects that the master has accepted the logon, terminates transmission of the "logon identifier" and immediately starts sending cyclic device status messages onto the bus.

5.5.4 System start/stop

Function: System Start/ Stop
Data direction: Master -> All
Type: broadcast

Priority based on CAL	ID	Data byte 0
1	221	00 = STOP 01 = START

STOP

- The device is at System Stop:
- - The device stops the drive
- - Reference values are then only received and stored

START

- - Enable time monitoring (watchdog functions)
- - FI / SERVO is allowed to transmit messages over the bus
- - Control functions are processed
- - Error messages can be sent over the bus

5.5.5 Control functions

Function: Control functions/reference
Data direction: Master -> Device
Type: selective

Control functions can be optimally adapted to the relevant application. Consequently, several control formats are offered. The appropriate formats can be selected by the master during the setup phase over the bus, or by adjusting the relevant device parameters.

A) Control functions for inverters

An inverter can receive and process control commands at the full transfer rate of 1 Mbaud. Since the internal state machine has a sampling time of 8 ms, these values enter the control cycle of the inverter every 8 ms. Data byte 2 contains the terminal emulation of the inverter.

Sampling time: 8 ms

Priority based on CAL	Base ID	Data byte 0	Data byte 1	Data byte 2
3	661	REF_LO	REF_HI	InBits 0 = STR 1 = STL 2 = S1IND 3 = S2IND 4 = S1OUT 5 = S2OUT 6 = S3OUT 7 = ERROR_RESET

Explanatory note on the in-bits:

STR /STL	=	Start commands: Where 01-MODE = 4 CAN as control location (device default setting in CAN operation). Where 01-MODE < 4 the device can only be activated via terminals.
S1IND/S2IND	=	Freely programmable inputs of the inverter. If these bits are set in the control word , they are linked by an OR function in the inverter to the values of the hardware inputs (with the default setting fixed frequencies are selected by way of the inputs).
S1OUT/S2OUT/S3OUT	=	Freely programmable outputs of the inverter. If these bits are set in the control word, they are linked in the inverter to the values of the inverter outputs by an OR function. To control the outputs over the CAN bus alone, all functions of the inverter of which the parameters can be set to this output must be shut down. For more detailed information on the setting and availability of these functions, refer to the parameter description which is available as a separate document for each device.
ERROR_RESET	=	With this bit errors are reset and the inverter switches to "System_Stop".

04-FSSEL defines which reference source is used to control the inverter. The values 25 and 26 refer to the CAN bus as the reference source. Note that any different setting of these parameters will result in reference sources other than the CAN bus being processed. For more detailed information on the setting and availability of these functions, refer to the parameter description which is available as a separate document for each device.

- **With parameter setting: 04-FSSEL = 26**
Reference scaling: 1 Bit = 0.009934Hz

In this mode the maximum resolution of the inverter is reached. The accuracy of the delivered frequency incorporates no conversions or ramps whatever. The output frequency of the inverter depends only on the quartz oscillator used.

Attention: With 04-FSSEL=26, the drive rigidly follows the reference input (no ramp function !!) Consequently, the reference ramp must be generated by the higher-order controller.

- **With parameter setting: 04-FSSEL = 25**
Reference scaling: 1 Bit = 0.05Hz

In this mode all ramp functions and control functions are applicable. For more detailed information on the setting and availability of these functions, refer to the parameter description which is available as a separate document for each device.

Attention: With 04-FSSEL=25, the drive follows the reference input with an internal ramp function. This means that the higher-order controller does not need to generate a ramp.

B) Control functions for servos

The servo state machine has a sampling time of 1 ms. All control commands and reference values are processed within that sampling time by the servocontroller.

Sampling time: 1 ms

Selection of state control and reference input by way of parameter 492-CACNF.

CACNF	1	2	3	4	5
Reference 1	16 bits	32 bits	32 bits	32 bits	
Reference 2					
Actual 1	16 bits	16 bits	16 bits	32 bits	
Actual 2		16 bits	16 bits		
DRIVECOM	□	□	□		□
Terminal emulation				□	

for MC6000/7000 modes:
- Speed control
- Torque control

only for MC7000 modes:
- Positioning and sequence control
- Electronic gearing

5.5.5.1 Control in Speed and Torque Control modes

Control with DRIVECOM state machine

Where 492-CACNF = 1

Priority based on CAL	Base ID	Data byte 0+1	Data byte 2+3	Data byte 4
3	661	Control word	Reference	Cycle

Control word: See description of control word

Cycle: Number of the internal time cycle of the drive in which the reference value is to become valid. Evaluated only in synchronized operation.

Reference: Reference (dependent on the respective control)
The data format is Int16 -> Value range: -32767 to +32768 without decimal place

Where 492-CACNF = 2,3

Priority based on CAL	Base ID	Data byte 0+1	Data byte 2+3+4+5	Data byte 6
3	661	Control word	Reference	Cycle

Control word: See description of control word

Cycle: Number of the internal time cycle of the drive in which the reference value is to become valid. Evaluated only in synchronized operation.

Reference: Reference (dependent on the respective control). The data format is Int32Q16 -> Value range: -32767.999 to +32768.999
(High word = pre-decimal point, Low word = post-decimal point; see also 492-CACNF =4)

Control via terminal emulation

Where 492-CACNF = 4

Priority based on CAL	Base ID	Data byte 0	Data byte 1	Data byte 2	Data byte 3	Data byte 4	Data byte 5
3	661	Bit: 0 = START 1 = INV 2 = /STOP 3 = E_EXT 4 = MP_UP* 5 = MP_DOWN* 6 = not assigned 7 = ERROR_RESET	Bit: 0 = OS00 1 = OS01 2 = n.a. 3 = n.a. 4 = n.a. 5 = n.a. 6 = n.a. 7 = n.a.	SOLLW_LWLB	SOLLW_LWHB	SOLLW_HWLB	SOLLW_HWHB

* Note: Only active if MOP function is set via reference selector.

SOLLW_LWLB: Reference value Low Word Low Byte
SOLLW_LWHB: Reference value Low Word High Byte
SOLLW_HWLB: Reference value High Word Low Byte
SOLLW_HWHB: Reference value High Word High Byte

The data format of the reference value is Int32Q16 -> Value range: -32767.999 to +32768.999
 (High Word = pre-decimal point, Low Word = post-decimal point)

5.5.5.2 Control in MC7000 modes:

- Positioning and sequence control (PosMod)
- Electronic gearing and
- Stepper motor interface

Where 492-CACNF = 5 (MC7000 only)

For information on the functions of flags (529-POMER), the table index (527-POTAB) and variables (528-POVAR) refer to the MC7000 PosMod programming manual.
 In Electronic Gearing and Stepper Motor Interface modes data bytes 2 - 7 have no function.

492-CACNF = 5 ; Control of POSMOD over CAN

Base ID	Data byte 0-1	Data byte 2	Data byte 3	Data bytes 4-7
661	Control word to control the PosMod (see above)	559-POQTI Table index	529-POMER Flag Index 90-97 Flag byte	528-POVAR Variable Index 98

5.5.5.3 Control word to control MC7000 PosMOD, electronic gearing and stepper motor interface

The first word of the control identifier 661 is always interpreted as the control word. The 16 bits of the control word result from the logical linking of control commands which act on the state machine. Bits 4 to 6 are mode-specific, bits 11 to 13 manufacturer-specific. Here the POSMOD-specific control is coded.

The POSMOD-specific bits are only used by the POSMOD in CAN system state 4, "Operation enabled"; otherwise they are deleted.

Control word in "Positioning and sequence control" mode:

Bit	Name	Comments
0	Switch-on	DRIVECOM state machine for controller enable
1	Disable power	
2	Emergency stop	
3	Enable operation	
4	Auto	Switch between Manual/Automatic mode (0/1)
5	Start / Referencing	Start a sequence program with a high edge; trigger a reference run in manual mode
6	Feed hold	Enable for axle movement (high-active)
7	Reset fault	
8	<i>Reserve</i>	
9	<i>Reserve</i>	
10	<i>Reserve</i>	
11	Update	Enable sequence prog. processing (high-active)
12	Jog+	Jog, in manual mode only (bit 4)
13	Jog-	Jog, in manual mode only (bit 4)
14	<i>Engage/disengage</i>	"Electronic gearing", "Stepper motor interface" mode only
15	<i>vacant</i>	

The reserve bits 8 to 10 are reserved for profile expansions and must always be set to 0. Only the bits printed in bold in the control word are used by the device.

Control word in "Electronic gearing" or "Stepper motor interface" mode:

Bit	Name	Comments
0	Switch-on	DRIVECOM state machine for controller enable
1	Disable power	
2	Emergency stop	
3	Enable operation	
4	<i>Auto</i>	"Positioning and sequence control" mode only
5	Start / Referencing	Trigger a reference run
6	<i>Feed hold</i>	"Positioning and sequence control" mode only
7	Reset fault	
8	<i>Reserve</i>	
9	<i>Reserve</i>	
10	<i>Reserve</i>	
11	<i>Update</i>	"Positioning and sequence control" mode only
12	<i>Jog+</i>	"Positioning and sequence control" mode only
13	<i>Jog-</i>	"Positioning and sequence control" mode only
14	Engage/disengage	1= Engage, axle follows guide reference
15	<i>vacant</i>	not used

The reserve bits 8 to 10 are reserved for profile expansions and must always be set to 0. Only the bits printed in bold in the control word are used by the device.

If the CAN control location is selected, the reference run may be triggered over CAN or by way of an appropriately set input. The logic link between the two settings may be seen as an OR function. Triggering of referencing is not accepted if the electronic gearing is already engaged.

5.5.6 Control mechanisms, MC7000 PosMod

The following functions can be reached by way of the bus system

- | | |
|---|---|
| • Start control (currently with ENPO) | CAN control word according to DRIVECOM |
| • Start sequence program | CAN control word (mode-specific) |
| • Automatic enable | CAN control word (mode-specific) |
| • Feed hold | CAN control word (mode-specific) |
| • Update | CAN control word (mode-specific) |
| • Jog + - | CAN control word (mode-specific) |
| • Transfer of table index,
program no. and reference run no. | Parameter channel, |
| • Transfer of flags and variables | Parameter channel |
| • Loading of positioning program | Parameter channel, parameter <i>551-POCMD</i> |
| • CAN outputs | CAN control telegram |

5.5.6.1 Control enable

When positioning and sequence control (546-POENA = ON), electronic gearing or stepper motor interface mode is active control is enabled by way of the CAN control word (DRIVECOM state machine), and not solely via the control contact ENPO.

5.5.6.2 Automatic enable and start of sequence program

"Positioning and sequence control" mode only!

The *Auto* and *Start* control functions are permanently linked to the control location. If the control location (402-CLSEL = PMOD) is positioned on terminals, the digital inputs IS00 and IS01 are active. If the control location is located on CAN (402-CLSEL = CAN), these functions are determined by bits from the incoming CAN control word.

Note: Between the ***Auto*** and ***Start*** control functions a delay of 10 ms must be maintained, to ensure that the positioning and sequence control has switched safely to automatic mode.

5.5.6.3 Feed hold and update

"Positioning and sequence control" mode only!

The Feed Hold and Update functions are described simultaneously by the incoming CAN control word and the configured digital inputs of the MC7000. Both functions are high-active. The setting under which the relevant bit or input - as appropriate - is deleted has priority.

5.5.6.4 Jog+ and jog-

"Positioning and sequence control" mode only!

The jog+ and jog- functions are described with equally by the incoming CAN control word and the configured digital inputs of the MC. The effect can be seen as an OR function of the two settings.

5.5.6.5 Program number

"Positioning and sequence control" mode only!

The program number can be selected via terminals or by way of parameter *535-POPKD - Coding of program number*. Parameter *534-POQPN-Source of program number* determines, with the *fix* setting, that the program number is set by way of parameter *535-POPKD*. A setting unequal to *fix* means that the program number is read-in with the appropriate coding via the digital inputs (IE00 - IE07) (see MC7000 PosMOD programming manual).

To enable the program number to be changed over CAN, parameter *534-POQPN* must be set to *fix*. Parameter *535-POPKD-Coding of program number* can be set by way of the parameter channel. The setting only takes effect when Auto mode is reselected.

5.5.6.6 Reference run number

"Positioning and sequence control" mode only!

The reference run number can only be set by way of parameter *522-PORTY Reference run type [0-8]*. This parameter can be set via the parameter channel. (see MC7000 PosMOD programming manual).

5.5.6.7 Table index

"Positioning and sequence control" mode only!

Similarly to the program number, the table index can be set via terminal or CAN control word. The configuration is specified by the two parameters described in the following (see MC7000 PosMOD programming manual).

558-POTKD-Coding of table index

Default setting BIN => Table index via defined control inputs.

Setting FIX => Index from control word

Data type: usign8, EEPROM

Value range: fix, bin

Subject area: Positioning and sequence control

SC subject area: PMOD

559-POQTI-Source of table index

The value is determined by the data content in the control identifier (data byte 2).

Default setting 0

Data type: usign8, RAM control value

Value range: 0-15

Subject area: Positioning and sequence control

SC subject area: NON

5.5.6.8 Flags and variables

"Positioning and sequence control" mode only!

The field parameters *529-POMER-Flag* and *528-POVAR-Variable* can be read and written to in the sequence program. By way of parameter *529-POMER* the PosMOD can be informed of events to which the sequence program responds. Likewise, the sequence program can indicate the occurrence of certain events in parameter *529-POMER*. By way of parameter *528-POVAR* data can be exchanged with the PosMOD sequence program.

They can be accessed via the parameter channel.

If the MC7000 PosMOD is controlled over CAN, some indices of the field parameters are accessible via the control/status identifier.

In the control identifier:	Flags M90 - 97	Data byte 3
	Variable H98	Data bytes 4 - 7

In the status identifier:	Flags M80 - 87	Data byte 3
---------------------------	----------------	-------------

(see also MC7000 PosMOD programming manual).

5.5.7 Status messages

Function: Status/actual value
Data direction: Device -> Master
Type: selective

5.5.7.1 Inverter status messages

The inverter transmits its status message every 80 ms.

Cycle: 80 ms

Priority	Base ID	Data byte 0	Data byte 1	Data byte 2
4	881	IST_LO	IST_HI	OUT_BIT 0 = STR 1 = STL 2 = S1IND 3 = S2IND 4 = S1OUT 5 = S2OUT 6 = S3OUT 7 = ERROR

Where 04-FSSEL = 26, actual value scaling: 1 bit = 0.009934Hz

Where 04-FSSEL = 25, actual value scaling: 1 bit = 0.05Hz

Explanatory note on the out-bits:

STR /STL	=	Status of the start commands
S1IND/S2IND	=	Freely programmable inputs of the inverter. If these bits are set in the control word, they are linked by an OR function in the inverter to the values of the inverter inputs, and signaled in the inverted as the status.
S1OUT/S2OUT/S3OUT	=	Freely programmable outputs of the inverter. If these bits are set in the control word, they are linked in the inverter to the values of the inverter outputs by an OR function. To control the outputs over the CAN bus alone, all functions of the inverter of which the parameters can be set to this output must be shut down. For more detailed information on the setting and availability of these functions, refer to the parameter description which is available as a separate document for each device.
ERROR	=	Error message

5.5.7.2 Servodrive status messages in modes: Speed and Torque Control

With regard to selection of the required status message refer to "Selection of control commands".

Status with DRIVECOM state machine

Where 492-CACNF = 1

Priority based on CAL	Base ID	Data byte 0+1	Data byte 2+3	Data byte 4
3	881	Status word	Actual	Cycle

Where 492-CACNF = 2,3

Priority based on CAL	Base ID	Data byte 0+1	Data byte 2+3	Data byte 4+5	Data byte 6
3	881	Status word	Actual 1	Actual 2	Cycle

Status word: See description of status word

Cycle: Number of the internal cycle in which the status word was determined - evaluated only in synchronized operation.

Actual 1: See definition of 492-CACNF, actual value of active operation mode, data format Int16

Actual 2: See definition of 492-CACNF, actual torque, data format Int16

Status with terminal emulation

Where 492-CACNF = 4

Priority based on CAL	Base ID	Data byte 0	Data byte 1	Data byte 2	Data byte 3	Data byte 4	Data byte 5
4	881	Bit: 0 = ERROR 1 = WARN 2 = REF 3 = LIMIT 4 = ACTIV 5 = ROT_0 6 = ROT_R 7 = ROT_L	Bit: 0 = ENPO 1 = IS00 2 = IS01 3 = OS00 4 = OS01 5 = A0 6 = A1 7 = EXT_SYN1	IST _LWLB	IST _LWHB	IST _HWLB	IST _HWHB

IST_LWLB: Actual value Low Word Low Byte

IST_LWHB: Actual value Low Word High Byte

IST_HWLB: Actual value High Word Low Byte

IST_HWHB: Actual value High Word High Byte

The data format is Int32Q16 -> Value range: -32767.999 to +32768.999
(High Word = pre-decimal point, Low Word = post-decimal point)

Status message of MC7000 in modes:

- Positioning and sequence control (POSMOD)
- Electronic gearing and
- Stepper motor interface

Where $492-CACNF = 5$

For information on the functions of flags (POMER) and the table index (POQTI) refer to the positioning and sequence control programming manual.

In Electronic Gearing and Stepper Motor Interface modes data bytes 2 - 7 have no function.

Base ID	Data bytes 0 -1	Data byte 2	Data byte 3	Data bytes 4-7
881	Status word to control PosMOD (see above)	559-POQTI Table index	529-POMER Flag Index 80-87 Flag byte	545-POAIP Current actual position in travel units

5.5.8 Status word for control of PosMOD over CAN

The status word is always written to the first word of status identifier 881. In the status word the current state of the device and additional messages are displayed. Bits 12 to 15 offer room for mode-dependent or manufacturer-specific displays.

The following bits of the status word are supported:

Bit	Name	Comments
0	Ready for start	DRIVECOM state machine
1	On	
2	Operation enabled	
3	Fault	
4	Power disabled	
5	Emergency stop	
6	Switch-on inhibit	
7	Warning	
8	vacant	
9	Remote, always 1	
10	Mode-dependent, Axle in position	POSMOD ONLY
11	Limit value	
12	Mode-dependent, <i>engaged</i>	Electronic gearing and stepper motor interface only
13	Mode-dependent, Ref.pt. defined	
14	Manufacturer-specific, Prog. end	POSMOD ONLY
15	Manufacturer-specific, error	POSMOD ONLY

Only the bits printed in bold are used by the device.

Status word in "Electronic gearing" or "Stepper motor interface" mode:

Bit	Name	Comments
0	Ready for start	DRIVECOM state machine
1	On	
2	Operation enabled	
3	Fault	
4	Power disabled	
5	Emergency stop	
6	Switch-on inhibit	
7	Warning	
8	vacant	
9	Remote, always 1	
10	Mode-dependent, Axle in position	PosMOD ONLY
11	Limit value	
12	Mode-dependent, Engaged	Electronic gearing and stepper motor interface only
13	Mode-dependent, Ref.pt. defined	
14	<i>Manufacturer-specific, Prog. end</i>	PosMOD ONLY
15	<i>Manufacturer-specific, Tracking error</i>	PosMOD ONLY

Only the bits printed in bold are used by the device.

6 Response to device fault

In case of error this state is indicated by the existing LEDs on the device, by the red back-lighting of the KEYPAD and by the device status word.

The SMARTDRIVE always disables the power stage in case of error. The error response of the MASTERCONTROL is programmable for each error in five stages.

In the error state the devices remain operable via the KEYPAD and all connected bus systems.

6.1 Error messages

Function: Error messages

Data direction: Device > Master

Type: selective

Messages may only be sent when Start = 1.

A) Error messages relating to inverters

Priority based on CAL	Base ID	Data byte 0
2	441	Error number The number corresponds to the error number of the inverter (for definition see Operation Manual >> Error messages)

B) Error messages relating to servos

Priority based on CAL	Base ID	Data byte 0	Data byte 1
2	441	Error number The number corresponds to the error number of the servo (for definition see Operation Manual >> Error messages)	Error location This number permits a more precise definition of the causes of error in servodrives.

Error codes, MC6000/7000

Value	Error text	Description
1	E-CPU	Hardware or software error
2	OFF	Power failure
3	E-OC	Current overload shut-off
4	E-OV	Voltage overload shut-off
5	E-OLI	Ixt shut-off
6	E-OTM	Motor overheating
7	E-OTI	Servo overheating
8	E-EEP	Faulty EEPROM
9	E-OLM	I?xt shut-off
10	E-PLS	Plausibility error in parameter or program sequence
11	E-PAR	Faulty parameter setting
12	E-FLT	Floating-point error
13	E-PWR	Power pack not recognized
14	E-EXT	External error message (input)
15	E-ENC	Encoder evaluation defective

16	E-OP1	Error in module in option slot 1
17	E-OP2	Error in module in option slot 2
18	E-TIM	Runtime error
19	E-FLW	Tracking error
20	E-WDG	SIO watchdog
21	E-CAN	Error in CAN hardware or software
22	E-IO1	Input submodule not recognized
23	E-IO2	Output submodule not recognized
24	E-VEC	Error initializing VeCon processor
25	E-BRK	Error at brake output OS03
26	E-POS	Error message from PosMod1
27	E-FLH	Error in FLASH memory

MC7000 PosMod error messages

All errors signaled by the POSMOD are displayed with the error text 'E-POS'. Various error locations are used to differentiate between the errors.

Error text	Error location	Description of error
E-CPU	210	Positive hardware limit switch approached
	211	Negative hardware limit switch approached
	212	Positive software limit switch approached
	213	Negative software limit switch approached
	214	Reference point not defined
	215	Addressed hardware not available
	216	Selected program not available
	217	Jump to non-existent record number
	218	Called subroutine not available
	219	Destination position outside positioning range
	220	Division by zero
	221	Max. nesting depth exceeded
	222	Timeout in manual mode
	223	Destination position not reached
	224	No feed hold
	225	Selection (Auto/Referencing/Jog) not permitted
	226	Index overflow (indexed addressing)
	227	not used
	228	not used
	229	not used
	230	Max. velocity of servo exceeded
	231	not used
	232	No controller enable
	233	not used
	234	not used
	235	Impermissible command during axle movement

6.2 Acknowledgment of error messages

A) *Inverters*

When the error flag is set in the status message the error message is additionally transmitted over the CAN bus alternately with the status message.

Errors can be reset in different ways:

- By changing the system status from System Start to System Stop and then back to System Start - This operation allows an error to be reset on any number of devices in the network simultaneously
- By setting the ERROR_RESET flag in the control word or in the DRIVECOM control word by change of state to Error Reset (control code 0080 Hex)
- On the KEYPAD (see KEYPAD instructions)
- By way of control terminals (only with appropriate parameter setting)

When the error bit is set in parameter 11-STAT a reset can be triggered relating to that parameter by a SELECT telegram with the new value 000F Hex (VALUE = "000F").

B) *Servos*

The basic response of a servo to errors is the same as the inverter as described above.

7 Examples

7.1 Activation of a VF1000 frequency inverter

Presets:

- Parameter 01-MODE = 4 Control location interface
- Parameter 04-FSSEL = 25 Reference selector
- Parameter 82-SIOA = 0 Device address

Action	Who is transmitting	ID on bus	Data bytes	Comments
Log on system	FI 0	1543	None	The frequency inverter transmits this identifier in a 100 ms cycle until the master has addressed an identifier of the inverter.
Send control identifier	Master	For FI 0: 661 For FI 1: 663 etc.	Data byte 0 = 0 Data byte 1 = 0 Data byte 2 = 0	The master sends the control identifier to the FI to terminate the system logon. The transmitted data are only relevant when "System Start" is set.
Start system	Master	For all FIs: 221	Data byte 0 = 01	The master sends "System Start". With this command the control commands stored in the control word of the FI are activated. From this point on the master must transmit a control identifier for the corresponding inverter at least every 10 ms. If this time is exceeded an error is generated.
Send control identifier	Master	For FI 0: 661 For FI 1: 663 etc.	For example: Data byte 0 = 01 Data byte 1 = 00 Data byte 2 = 01	Example: FI 0 is to rotate clockwise at 0.009934 Hz.
Status message	FI	For FI 0: 881 For FI 1: 883 etc.	For example: Data byte 0 = 01 Data byte 1 = 00 Data byte 2 = 01	Example: FI rotates at 0.009934 Hz clockwise The FI must signal its status at least every 80 ms. If the status is not signaled within that time, the master must signal an error

Parameter channel for VF1000 frequency inverter:

Action	Who is transmitting	ID on bus	Data bytes	Comments
Enquire for parameter	Master to FI 0	1101	Data byte 0 = 0E Data byte 1 = 00 Data byte 2 = 05 Data byte 3 = XX Data byte 4 = XX	Enquire for parameter for phase current (parameter 14-IS)
Reply from FI	FI 0	1321	Data byte 0 = 0E Data byte 1 = 00 Data byte 2 = 05 Data byte 3 = 28 Data byte 4 = 00	Phase current signal 2 A (scaling 0.05 A = 1 bit)
Send parameter	Master to FI 0	1101	Data byte 0 = 4E Data byte 1 = 00 Data byte 2 = 02 Data byte 3 = 03 Data byte 4 = 00	Set parameter 78-OPT4) to 3
Reply from FI	FI 0	1321	Data byte 0 = 4E Data byte 1 = 00 Data byte 2 = 05 Data byte 3 = 28 Data byte 4 = 00	

7.2 Activation of an MC7000 in Speed Control mode

7.2.1 Terminal emulation control mode

Preset:

- Load motor data via DRIVEMANAGER user interface
- Activate Speed Control mode via DRIVEMANAGER user interface
- Optimize controller
- Parameter 402-CLSEL = CAN (OPT1 on MC6000)
- Parameter 419-RSSL3 = CAN
- Parameter 489-CABDR = 500 Set baud rate
- Parameter 493-CAADR = 1 Device address
- Parameter 492-CACNF = 4 (speed-controlled) Control mode: terminal emulation

- Mains reset to reinitialize
- Wire control contact hardware enable ENPO

Action	Who is transmitting	ID on bus	Data bytes	Comments
Log on system	MC 1	1545	None	The MC sends this identifier in a 100 ms cycle until the master has addressed an identifier of the MC.
Send control identifier	Master	For MC 0: 661 For MC 1: 663 etc.	Data byte 0 = 0 Data byte 1 = 0 Data byte 2 = 0 Data byte 3 = 0 Data byte 4 = 0 Data byte 5 = 0	The master sends the control identifier to the MC to complete the system logon. The transmitted data are only relevant when "System Start" is set.
Start system	Master	For all MCs: 221	Data byte 0 = 01	The master sends "System Start". With this command the control commands stored in the control word of the MC are activated. From this point on the preset timeout 409-BUTWD is monitored. If this time is exceeded an error is generated.
Send control identifier	Master	For MC 0: 661 For MC 1: 663 etc.	For example: Data byte 0 = 01 Data byte 1 = 00 Data byte 2 = 00 Data byte 3 = 00 Data byte 4 = 0A Data byte 5 = 00	Example: MC 1 is to rotate clockwise at 10 rpm
Status message	MC	For MC 0: 881 For MC 1: 883 etc.	For example: Data byte 0 = 01 Data byte 1 = 00 Data byte 2 = 00 Data byte 3 = 00 Data byte 4 = 0A Data byte 5 = 00	Example: MC 1 rotates clockwise at 1 rpm

7.2.2 Control mode: DRIVECOM state machine

Preset:

- Load motor data via DRIVEMANAGER user interface
- Activate Speed Control mode via DRIVEMANAGER user interface
- Optimize controller
- Parameter 402-CLSEL = CAN (OPT1 on MC6000)
- Parameter 419-RSSL3 = CAN
- Parameter 489-CABDR = 500 Set baud rate
- Parameter 493-CAADR = 1 Device address
- Parameter 492-CACNF = 2 (speed-controlled) Control mode: DRIVECOM state machine

- Mains reset to reinitialize
- Wire control contact hardware enable ENPO

Action	Who is transmitting	ID on bus	Data bytes	Comments
Log on system	MC 1	1545	None	The MC sends this identifier in a 100 ms cycle until the master has addressed an identifier of the MC.
Send control identifier	Master	For MC 0: 661 For MC 1: 663 etc.	Data byte 0 = 0 Data byte 1 = 0 Data byte 2 = 0 Data byte 3 = 0 Data byte 4 = 0 Data byte 5 = 0	The master sends the control identifier to the MC to complete the system logon. The transmitted data are only relevant when "System Start" is set.
Start system	Master	For all MCs: 221	Data byte 0 = 01	The master sends "System Start". With this command the control commands stored in the control word of the MC are activated. From this point on the preset timeout 409-BUTWD is monitored. If this time is exceeded an error is generated.
Send control identifier	Master	For MC 0: 661 For MC 1: 663 etc.	For example: Data byte 0 = 00 Data byte 1 = 00 Data byte 2 = 00 Data byte 3 = 00 Data byte 4 = 0A Data byte 5 = 00	Example: MC 1 is to dwell in the "Ready for start" state. Reference value 10 rpm clockwise applied.
Status message	MC	For MC 0: 881 For MC 1: 883 etc.	For example: Data byte 0 = 40 Data byte 1 = 02 Data byte 2 = 00 Data byte 3 = 00 Data byte 4 = 00 Data byte 5 = 00	Example: MC 1 signals "Ready"

Send control identifier	Master	For MC 0: 661 For MC 1: 663 etc.	For example: Data byte 0 = 06 Data byte 1 = 00 Data byte 2 = 00 Data byte 3 = 00 Data byte 4 = 0A Data byte 5 = 00	Example: MC 1 is to switch from "Ready" to "On". Reference value 10 rpm clockwise applied.
Status message	MC	For MC 0: 881 For MC 1: 883 etc.	For example: Data byte 0 = 31 Data byte 1 = 02 Data byte 2 = 00 Data byte 3 = 00 Data byte 4 = 00 Data byte 5 = 00	Example: MC 1 signals "On"
Send control identifier	Master	For MC 0: 661 For MC 1: 663 etc.	For example: Data byte 0 = 0F Data byte 1 = 00 Data byte 2 = 00 Data byte 3 = 00 Data byte 4 = 0A Data byte 5 = 00	Example: MC 1 is to switch from "On" to "Operation enabled". Reference value 10 rpm clockwise applied.
Status message	MC	For MC 0: 881 For MC 1: 883 etc.	For example: Data byte 0 = 37 Data byte 1 = 02 Data byte 2 = 00 Data byte 3 = 00 Data byte 4 = 0A Data byte 5 = 00	Example: MC 1 rotates clockwise at 10 rpm and signals "Operation enabled"

Parameter setting

Action	Who is transmitting	ID on bus	Data bytes	Comments
Enquire for parameter	Master to MC 1	1103	Data byte 0 = 02 Data byte 1 = 00 Data byte 2 = 05 Data byte 3 = XX Data byte 4 = XX Data byte 5 = XX Data byte 6 = XX Data byte 7 = 00	Enquire for parameter to be displayed as continuous actual value (parameter 02-DISP)
Reply from FI	MC 1	1323	Data byte 0 = 02 Data byte 1 = 00 Data byte 2 = 00 Data byte 3 = 05 Data byte 4 = 00 Data byte 5 = 00 Data byte 6 = 00 Data byte 7 = 00	Message: Parameter DISP = 5 (5-CTLFA)
Send parameter	Master to MC 3	1103	Data byte 0 = 02 Data byte 1 = 00 Data byte 2 = 02 Data byte 3 = 03 Data byte 4 = 00 Data byte 5 = 00 Data byte 6 = 00 Data byte 7 = 00	Set parameter 02-DISP to 3

Reply from MC	MC 1	1323	Data byte 0 = 02 Data byte 1 = 00 Data byte 2 = 00 Data byte 3 = 03 Data byte 4 = 00 Data byte 5 = 00 Data byte 6 = 00 Data byte 7 = 00	Checkback from MC after successful data transfer
---------------	------	------	--	--

7.3 Example: MC7000 PosMod activation

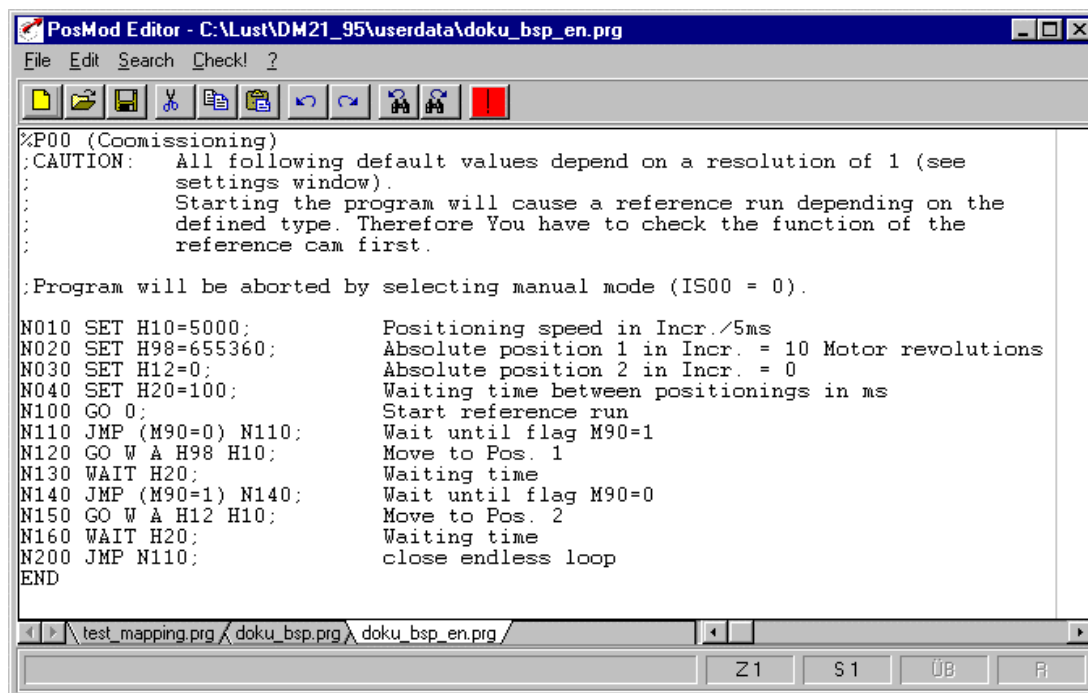
Task:

Load a sequence program into the servo axle and activate it over CAN.
In this process, the positioning is to be controlled by the status of a flag between the absolute position 0 and a freely adjustable position.

Presets:

- Load motor data set via DRIVEMANAGER user interface
- Activate "Positioning and sequence control" mode via DRIVEMANAGER user interface
- In Parameter Editor set following parameters:

492-CACNF = 5	Select control mode
402-CLSEL = CAN (OPT1 on MC6000)	
489-CABDR = 500	Set baud rate
493-CAADR = 1	Device address
- Load sequence program into servocontroller
- Mains reset to activate changed settings
- Wire control contact hardware enable ENPO



The screenshot shows the PosMod Editor window with the following content:

```
PosMod Editor - C:\Lust\ADM21_95\userdata\doku_bsp_en.prg
File Edit Search Check! ?
[Icons: New, Open, Save, Copy, Paste, Undo, Redo, Home, End, Stop]

;%P00 (Cocommissioning)
;CAUTION: All following default values depend on a resolution of 1 (see
; settings window).
; Starting the program will cause a reference run depending on the
; defined type. Therefore You have to check the function of the
; reference cam first.

;Program will be aborted by selecting manual mode (IS00 = 0).

N010 SET H10=5000;      Positioning speed in Incr./5ms
N020 SET H98=655360;   Absolute position 1 in Incr. = 10 Motor revolutions
N030 SET H12=0;        Absolute position 2 in Incr. = 0
N040 SET H20=100;      Waiting time between positionings in ms
N100 GO 0;             Start reference run
N110 JMP (M90=0) N110; Wait until flag M90=1
N120 GO W A H98 H10;   Move to Pos. 1
N130 WAIT H20;         Waiting time
N140 JMP (M90=1) N140; Wait until flag M90=0
N150 GO W A H12 H10;   Move to Pos. 2
N160 WAIT H20;         Waiting time
N200 JMP N110;         close endless loop
END

[Navigation: test_mapping.prg, doku_bsp.prg, doku_bsp_en.prg]
[Z1 S1 UB R]
```

Note:

Flag M90 triggers the positioning operations with an edge change.
Variable H98 contains the freely selectable reference position. Unit = increments

The drive can now be started with input ENPO set, with the following control sequence:

ID	Data bytes	Comments
DD	01	;Systemstart
297	06 00 00 00 00 00 00 00	;SHUT DOWN DRIVECOM (0->6)
297	4F 08 00 00 00 00 00 00	;SWITCH ON DRIVECOM (6->F)
		;Feed hold (4) always set
		;Update (8) always set
297	5F 08 00 00 00 00 00 00	;Automatic enable (5)
		;10ms delay until start of sequence program
297	7F 08 00 00 00 00 00 00	;Start enable, sequence program started (7)
297	7F 08 00 00 00 00 0A 00	;POMER[90] = 0, ;POVAR[98] = 655360 incr., destination position = 10 motor revolutions (absolute)
297	7F 08 00 01 00 00 0A 00	;POMER[90] = 1, Start positioning ;POVAR[98] = 655360 incr.,
297	7F 08 00 00 64 00 00 00	;POMER[90] = 0, Trigger positioning to pos. 0 ;POVAR[98] = 100 incr.,
297	7F 08 00 01 64 00 00 00	;POMER[90] = 1, Trigger positioning ;POVAR[98] = 100,
.		
.		
.		
297	4F 08 00 00 00 00 00 00	;SWITCH ON DRIVECOM (6->F)
		;Feed hold (4) set
		;Update (8) set
		;Axle in manual mode, axle can be moved by "Jog" function.
		;Parameter settings and download of se- quence prog. possible
297	06 00 00 00 00 00 00 00	;SHUT DOWN DRIVECOM (0->6), power stage off
DD	00	;System stop, <u>also error reset</u>

7.4 Loading and deleting the positioning program of the positioning and sequence control

A positioning program can be downloaded line-by-line to the POSMOD software by writing to the string parameter *551-POCMD-PosMOD Direct command input in manual mode*.

Example program:

%P00 (Commissioning)	
N010 SET H10=5000;	Positioning speed in inc/5ms
N020 SET H11=655360;	Absolute position 1 in inc. = 10 motor revs
N030 SET H12=0;	Absolute position 2 in inc.
N040 SET H20=100;	Waiting time between positioning operations in ms
N100 GO 0;	Trigger referencing
N110 WAIT (IE01=1);	Wait until input IE01=1
N120 GO W A H11 H10;	Approach pos. 1
N130 WAIT H20;	Waiting time
N140 WAIT (IE01=0);	Wait until input IE01=0
N150 GO W A H12 H10;	Approach pos. 2
N160 WAIT H20;	Waiting time
N200 JMP N110;	Close endless loop
END	

The above example program is transferred line-by-line to the servocontroller as a string via the parameter channel. The comments separated by semicolons are eliminated in the process. That is to say, the following strings are transmitted as data:

1st string	"%P00 (Commissioning)"	
2nd string	"N010 SET H10=5000"	
3rd string	"N020 SET H11=655360"	
.		
.		
14th string	"END"	Servo detects end of program transmission
15th string	"%SAV"	Back-up program code in Flash.
		The execution time depends on the program length (approx. 100ms).

Note: The program sets being transferred must have no comments or semicolons at the end, otherwise the transfer will be rejected by the device.

If a sequence program is to be overwritten, the original must first be deleted from the device memory. To this end the following string is transmitted:

String	"%CLPxx"
	↗ xx = program number 00 - 99

If the string "xx" is actually inserted for the program number, all sequence programs in the servocontroller are deleted!

Sequence programs can only be transmitted with the sequence control in manual mode!

7.5 Example: Activation in "Electronic Gearing" mode

Presets:

- Load motor data set via DriveManager user interface
- Activate Electronic Gearing mode via DriveManager user interface
- In Parameter Editor set following parameters:

492-CACNF = 5	Select control mode
402-CLSEL = CAN (OPT1 on MC6000)	
489-CABDR = 500	Set baud rate
493-CAADR = 1	Device address

- Mains reset to activate changed settings

Note: With parameters 387-VRNOM and 388-VRDOM the transmission ratio of the electronic gearing is specified as the numerator/denominator ratio

The drive can now be started with input ENPO set, with the following control sequence:

ID	Data bytes	;Description
DD	01	;Systemstart
297	06 00 00 00 00 00 00 00	;;SHUT DOWN DRIVECOM
297	0F 00 00 00 00 00 00 00	;;SWITCH ON DRIVECOM
297	2F 00 00 00 00 00 00 00	;Request reference run (not engaged)
297	0F 40 00 00 00 00 00 00	;Engage EGear
44F	83 01 02 0A 00 00 00 00	;Parameter channel VRNOM-183h = 10
44F	84 01 02 01 00 00 00 00	;Parameter channel VRDOM-184h = 1
		;Transmission ratio = 10-1
44F	83 01 02 01 00 00 00 00	;Parameter channel VRNOM-183h = 1
44F	84 01 02 0A 00 00 00 00	;Parameter channel VRDOM-184h = 10
		;Transmission ratio = 1-10
297	0F 00 00 00 00 00 00 00	;Disengage EGear
297	06 00 00 00 00 00 00 00	;SHUT DOWN DRIVECOM
DD	00	;Systemstop

Appendix A: Glossary of terms

- CiA:** ("CAN in Automation") CAN bus user group, generally defines a protocol for automation
- CAL:** (CAN Application Layer) CiA protocol, primarily describes the way in which variables are transmitted without defining their function or content
- Subsets:
CMC: (CAN based Message Specification) Specifies the above-described definition; accepted by most CAN suppliers; LUST conforms to this specification.
NMT: (Network Management) Required for the master in the CAN system; not implemented by Lust, as drive controllers always have a slave function but no "control function".
LMT: (Layer Management) See NMT
DBT: (Identifier Distributor) See NMT
- CANopen:** Based on CAL definition
- Corresponds to CiA Draft Standard 301
- Expands the CAL definition to include function and unit assignment of the predefined variables
- This definition is being drafted by CiA and various user groups (MOTION for drive technology and I/O for inputs/outputs) (e.g. variable for torque in Nm).
- Definition scheduled for completion by the end of 1996.
- ⇒ First devices scheduled for 2nd quarter 97
- MOTION:** User group under CiA tasked to draft a profile of the CANopen protocol for drive technology
- I/O:** User group under CiA tasked to draft a profile of the CANopen protocol for sensors and actuators
- Note:** Operation with drive controllers in a network is possible, as long as there are no overlaps of the identifiers used.

General points on the various protocol definitions

CAL	Mainly in use in Europe LUST has currently implemented a protocol which can be activated by a CAL master. Initialization has been simplified relative to CAL (CMC): e.g. addressing via jumper, but without impacting on operation.
Device Net	Mainly in the USA (corresponds to CAL definition)
SDS	Has not established itself.

We reserve the right to make technical changes.

ID no.: A047.22B.1-00

EN 03/99

Lust Antriebstechnik GmbH * Gewebestr. 5-9 * D-35633 Lahnau * Phone +49 64 41 / 966 -0 * Fax +49 64 41 / 966 -137

Internet: <http://www.lust-tec.de> * e-mail: lust@lust-tec.de