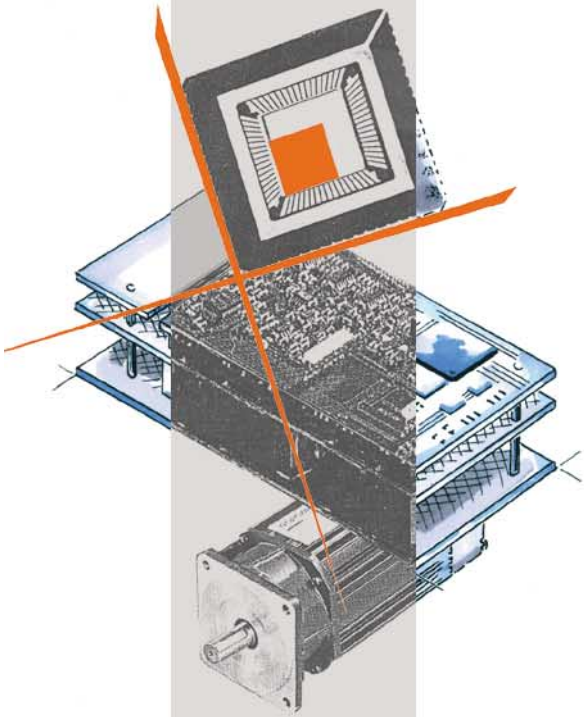
Manual Bus Functions ND31 and ND32

Version 4/2004



NOVOTRON

für Dynamik und Bewegung

Please read the following information about the symbols used in the manual:

Danger! Voltages may cause serious or fatal injury!

Noncompliance with instructions can endanger the life and sanity of persons!

Caution ! ☛ **Caution! Make sure to handle the device correctly!**

Noncompliance with instructions can lead to the destruction or can cause malfunction of the device or the entire equipment!

☞ **Link or recommendation**
Link to other sections of the text or recommendation for practical usage

1 2 **Sequencing of an instruction**

Menu *Limit values* **Designation of a menu or submenu**

Command *Channel1* **Designation of a command or function**

[ ], [enter] **Designation of a key or key combination**

Graphical representation of registers

SwVersion	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0xFF3D	7	6	5	4	3	2	1	0

SwVersion	Designation
0xFF3D	Address
R/W	Read/Write
R	ReadOnly

NOVOTRON

1 General Information

1.1 About this manual

The entire documentation of NOVODRIVE comprises 7 parts:

1 Manual **Basic Device ND31 and ND32**
Standard

2 Manual **Bus Functions ND31 and ND32**
On demand *)

3 Manual **Basic Functions ND31 and ND32**
On demand

4 Manual **Additional Functions ND31 and ND32**
On demand

5 Reserved

6 Manual **Start-up ND31 and ND32**
Standard

7 **Instructions for installation/exchange of ND31 and ND32**
Standard (leaflet)

The symbols used in the manuals are listed and explained on the inside front cover.

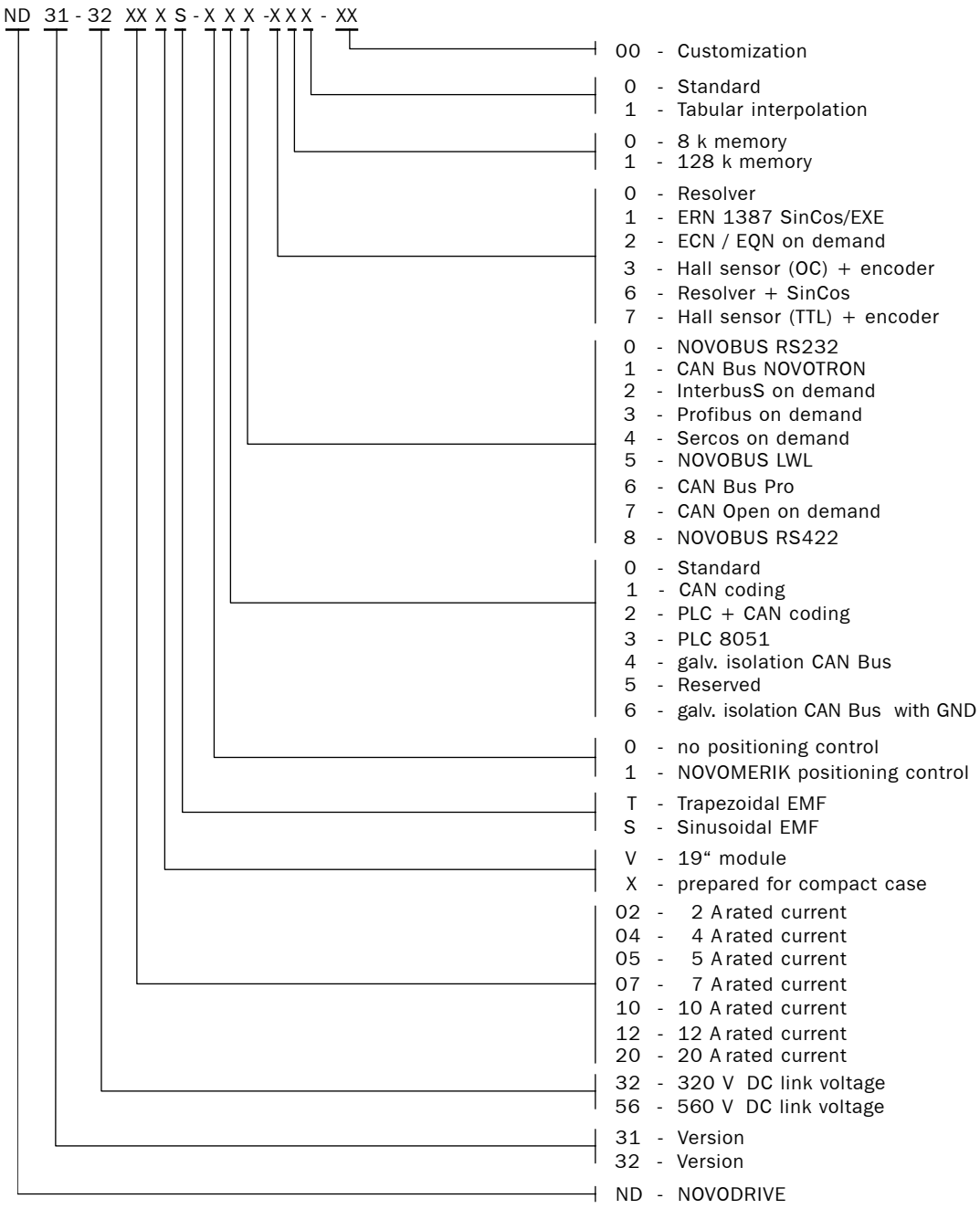
*) This manual

1.2 After-sales service

NOVOTRON GmbH
Mauserstrasse 31
71640 Ludwigsburg
Germany

Phone: +49 - (0)71 41 - 29 69 - 0
Fax: +49 - (0)71 41 - 29 69 - 22

1.3 Designations



1.4 Industrial property rights

IBM is a registered trademark of the IBM Corporation.

NOVOTRON

2 Safety Instructions

NOVODRIVE contains voltages that can be fatal!

Wiring Before switching on NOVODRIVE, carefully check the wiring. Make sure all plugs are properly connected and the device is properly grounded.

Protection Make sure no voltage-carrying parts may be accidentally touched and NOVODRIVE safety components are in place and properly connected.

Emergency power-off Provide an emergency power-off by which the motor can be stopped at any time.

Discharge time and contact voltage After being switched off the electrolytic capacitors require at least five (5) minutes to discharge. That means: After being switched off the device still contains dangerous voltage for up to five minutes. During this time, do not touch the device or disconnect any plug.

In case the motor is still turning after the supply voltage has been switched off, hazardous contact voltage may be present in the device until its standstill. Discharge of the capacitors then begins after the standstill.

Inrush current limitation Frequent switching of the supply voltage should be avoided, since thereby the inrush current limiter of NOVODRIVE may be overcharged, which may lead to the destruction of the inrush current limiting resistor. Wait one minute between switching on and switching off again.

Switching on/off sequence When switching on, first apply the 24 VDC supply voltage for the NOVODRIVE control section before connecting with the power supply. When switching off, proceed vice versa.

☞ Please read:
„Basic Device ND31 and ND32“ manual,
Chapter 2, „Safety Instructions“

3 NOVOBUS

This chapter informs you about the use of NOVOBUS for controlling drives.

3.1 General Information

NOVOBUS protocol and NOVOBUS driver for IBM-compatible PCs is an industrial property of the manufacturer.

NOVOBUS is a cost-efficient solution for interconnecting digital drives. NOVOBUS allows fast communication between a central computer (e.g. PC or PLC) and drives. Main functions are:

- process data transfer (e.g. setpoints and actual values for speed),
- transmission of new position setpoints for positioning axes,
- parameterization of drive (e.g. adjustment and online modification of controller structures, controller parameters and maximal values),
- transmission of control commands (start, stop, controller disable etc.),
- reading out important information (e.g. heat-sink and motor temperature, limit switch signal, ready-to-operate signal, in-position signal, additional external signals as process information, integrated operating-hours counter, drive states, error messages),
- control of the programmable analog and digital outputs of the drives (e.g. for activating and deactivating contactors or brakes, outputting warning signals etc.).

RS232/RS422 NOVOBUS is using RS232 or RS422, the standard serial interface of all PCs and state-of-the-art controls. Communication via NOVOBUS requires no additional hardware (e.g. bus controller, communications card, protocol chip, intelligent bus clamps).

Driver All digital drives have the serial interface as well as the software driver for the NOVOBUS protocol as standard features. For PC programs a NOVOBUS driver is available free of charge as a software library (NB31.LIB).

Ring structure Communication via NOVOBUS requires a ring-shaped wiring of the computer and the drives either by means of a RS232/RS422 cable or by means of fiber optic.

In one ring, up to 250 axes can be controlled. The drives are automatically addressed by their respective position in the ring. For achieving higher transmission speed or for combining autonomous machine parts, the drives can be distributed to several rings.

3.2 Characteristics

Transmission of one byte takes 286,46 μs at 38400 bit/s transmission speed.

Speed In case of speed-controlled drives, a complete process data transfer takes:

Drives per ring	Duration
1	0,86 ms
2	2,0 ms
3	2,9 ms
4	3,7 ms
5	4,6 ms
6	5,4 ms

The transmission of new position setpoints for positioning axes including the start of the positioning control calculation takes:

Drives per ring	Duration
1	4,01 ms
10	40,39 ms
100	401,33 ms
250	1002,79 ms

3.3 Physical Data Transmission Device

RS232 or RS422.

Transmission speed: Standard 38400 bits/s

Transmission format: 8 data bits, 1 parity bit (odd parity) and 1 stop bit.

3.4 Bus Structure

NOVOBUS is characterized by a ring-shaped structure. The drives can be distributed to one or several rings.

Master/Slave

Per ring: 1 bus master (master computer)
max. 250 slaves (drives)

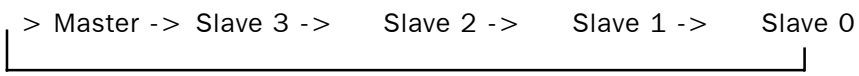
During the normal operating mode, the slaves may respond to the master telegrams only. If the timeout error occurs, the slaves may send error telegrams on their own.

3.5 Addressing The Participants

250 axes per ring

In one ring, up to 250 axes can be controlled. The drives are automatically addressed by their respective position in the ring. The numbering of the participants starts with the last participant in the ring, which is assigned address '0'. The address of the first participant in the ring is N-1, with N indicating the number of slaves.

For example, in case of four axes:



4 NOVOBUS Protocol

The master (central computer) continuously sends telegrams. Most of the telegrams contain an address (exception: 'SYNCO' and 'PAUSE'). The participants not addressed pass on the telegrams. The participant addressed responds to the telegram. The length of the response is identical with the length of the master telegram.

4.1 Telegram Structure

Synchronous byte (obligatory)

Address byte (optional)

Process data channel (2 bytes, optional)

Parameter channel (1-7 bytes, optional)

All bytes are sent with odd parity. The maximum net length of a telegram (process data channel + parameter channel) is 7 bytes. In case of a telegram with process data channel, the parameter channel can have a maximum length of 5 bytes.

4.2 Synchronous Byte

The synchronous byte is always the first byte of a telegram. It contains the coded telegram length.

Bit	7	6	5	4	3	2	1	0
Meaning	1	N	S	0	T2	T1	T0	D

N = 1 next address (otherwise = 0)

S = 1 short address

T2-T0 net length of telegram
(without synchronous byte and address byte, 0...7 bytes)

D = 1 telegram contains process data information
(2 bytes of process data channel)

Short addressing

If S=1 and N=0, the same drive is addressed as in the previous telegram.

If S=1 and N=1, the next drive is addressed (Address = Address0+1).

If S=0 and N=0, an address byte follows.

If no process data channel or parameter channel is active, the synchronous byte 'SYNC0' is sent in order to maintain a continuous data stream. The recipient leaves 'SYNC0' unchanged and passes it on.

'SYNC0': 0x80

To keep up the data stream, a synchronous byte 'PAUSE' can be sent. 'PAUSE' as synchronous byte is ignored by the recipient (no response).

'PAUSE': 0x81

4.3 Address Byte

Bit	7	6	5	4	3	2	1	0
Meaning	A7	A6	A5	A4	A3	A2	A1	A0

In the master telegram, A7-A0 contains:

k-N

k = participant's address in the ring

N = number of passive participants (slaves).

Each participant increases the address by one and passes on the telegram with the new address. A drive is addressd, if the address equals zero after the incrementation. Except for the change of the address byte, the drives not addressed leave the telegram unchanged and pass it on without checking the information.

Example N=5 , the master computer wants to address participant Axis2: (k-N = 2-5 = -3 = 0xfd).

Bit	7	6	5	4	3	2	1	0	
Meaning	A7	A6	A5	A4	A3	A2	A1	A0	
Master:	1	1	1	1	1	1	0	1	(0x FD=k-N)
Axis4:	1	1	1	1	1	1	1	0	(0x FE)
Axis3:	1	1	1	1	1	1	1	1	(0x FF)
Axis2:	0	0	0	0	0	0	0	0	(0x 00) <- !!!
Axis1:	0	0	0	0	0	0	0	1	(0x 01)
Axis0:	0	0	0	0	0	0	1	0	(0x 02 = k)

The incremented address equals zero at axis 2. The master receives the response to its telegram with address ‘k’ (in this example: 0x02).

4.4 Process Data Channel

By means of this channel, the process data transfer can be made faster. To do so, the lowest bit in the synchronous byte must be set.

☞ see Chapter 4.2

The data always have a length of 2 bytes. In the telegram, the most significant byte (MSB) is sent first and the least significant byte (LSB) is sent second.

4.5 Parameter Channel

By means of the parameter channel it is possible to assign the drives with parameters and to request information from the drives. A command in the parameter channel consists of one command byte, data bytes (0...4) and one check sum byte.

By means of one NOVOBUS telegram several commands can be sent through the parameter channel. On the other hand, a command in the parameter channel can be distributed to several telegrams.

The length of a response to a command is identical with the length of the command itself (2...6 bytes).

☞ The command byte and the check sum byte may never have the value 0x00.

4.6 NOVODRIVE Commands In Parameter Channel

Abbreviations used:

A0	low byte of address
A1	high byte of address
D0	LSB of data (8-bit, 16-bit and 32-bit values)
D1	...
D2	...
D3	MSB of data (32-bit values)
CS	the check sum computed by byte-wise adding of all bytes of the command telegram
NCS	two's complement of the check sum
-	ignored

☞ In case of 16-bit and 32-bit data, the address always refers to the MSB.

4.6.1 Access To Integrated Memory

Command **Read Byte**

Function Reading out an 8-bit value from the integrated memory.

Byte	0	1	2	3	4	5	6	7
Command telegram	0xC0	A0	A1	CS	-	-	-	-
Response telegram	0xC0	A0	D0	NCS	-	-	-	-
Address range	0x0000 - 0xFFFF							

Command **Read Word**

Function Reading out a 16-bit value from the integrated memory

Byte	0	1	2	3	4	5	6	7
Command telegram	0xC1	A0	A1	0x3F	CS	-	-	-
Response telegram	0xC1	A0	D0	D1	NCS	-	-	-
Address range	0x0000 - 0xFFFF							

Command **Read Long**

Function Reading out a 32-bit value from the integrated memory

Byte	0	1	2	3	4	5	6	7
Command telegram	0xC7	A0	A1	0x31	0x32	0x3F	CS	-
Response telegram	0xC7	A0	D0	D1	D2	D3	NCS	-
Address range	0x0000 - 0xFFFF							

Command **Write Byte**

Function Writing an 8-bit value to the integrated memory

Byte	0	1	2	3	4	5	6	7
Command telegram	0x82	D0	A0	A1	CS	-	-	-
Response telegram	0x82	D0	A0	A1	NCS	-	-	-
Address range	0xFE00 - 0xFE7F, 0xFEA0 - 0xFF7F							

Command **Write Word**

Function Writing a 16-bit value to the integrated memory

Byte	0	1	2	3	4	5	6	7
Command telegram	0x63	D0	D1	A0	A1	CS	-	-
Response telegram	0x63	D0	D1	A0	A1	NCS	-	-
Address range	0xFE00 - 0xFE7F, 0xFEA0 - 0xFF7F							

Command **Write Long**

Function Writing a 32-bit value to the integrated memory

Byte	0	1	2	3	4	5	6	7
Command telegram	0xC8	D0	D1	D2	D3	A0	A1	CS
Response telegram	0xC8	D0	D1	D2	D3	A0	A1	NCS
Address range	0xFE00 - 0xFE7F, 0xFEA0 - 0xFF7F							

☞ As Write Long telegrams have a length of 8 bytes, they must be split into two NOVOBUS telegrams.

4.6.2 Bit Manipulation

Command **AND**

Function Bit-wise logical AND with a register in the integrated memory. The upper 8 bits of the address are always 0xFF.

Byte	0	1	2	3	4	5	6	7
Command telegram	0xA4	D0	A0	CS	-	-	-	-
Response telegram	0xA4	D0	A0	NCS	-	-	-	-
Address range	0xFF00 - 0xFFFF							

Command **OR**

Function Bit-wise logical OR with a register in the integrated memory. The upper 8 bits of the address are always 0xFF.

Byte	0	1	2	3	4	5	6	7
Command telegram	0xA5	D0	A0	CS	-	-	-	-
Response telegram	0xA5	D0	A0	NCS	-	-	-	-
Address range	0xFF00 - 0xFFFF							

4.6.3 Access To External Memory

Command **Read Word X**

Function Reading out a 16-bit value from the external memory

Byte	0	1	2	3	4	5	6	7
Command telegram	0xC9	A0	A1	0x3F	CS	-	-	-
Response telegram	0xC9	A0	D0	D1	NCS	-	-	-
Address range	0x0000 - 0xFFFF							

Comand **Write Word X**

Function Writing a 16-bit value to the external memory

Byte	0	1	2	3	4	5	6	7
Command telegram	0x6A	D0	D1	A0	A1	CS	-	-
Response telegram	0x6A	D0	D1	A0	A1	NCS	-	-
Address range	0x4000 - 0xFFFF							

4.6.4 Control Commands

Command **Reset**

Function Reset of NOVODRIVE

Byte	0	1	2	3	4	5	6	7
Command telegram	0xDD	0x21	CS	-	-	-	-	-
Response telegram	-	-	-	-	-	-	-	-

4.7 Check Sum In Parameter Channel

The master (central computer) adds a check sum (CS) to all commands it sends in the parameter channel. This check sum is checked by the drive addressed. The command is executed only if the check sum is correct. If the check sum is not correct, the drive sends an error message.

☞ see Chapter 4.8 Error Treatment

Check sum

The check sum is the sum of the bytes of one command. If the sum equals 0, the check sum must be set to 1 (0x00 is reserved for error messages). The drive addressed creates a new check sum from its response and passes on the two's complement (negative check sum, NCS) of this check sum.

The check sum refers to the parameter channel only and is constituted from the bytes of the parameter channel only. Synchronous byte, address byte and process data are not taken into consideration for the check sum.

4.8 Error Treatment

If the drive recognizes an error during communication (parity error, framing error, invalid synchronous byte, invalid command in parameter channel, invalid command parameter or invalid check sum), it indicates an error status.

If an error occurs, the drive recognizing the error first responds with 0x00 to all bytes it receives. The next axis is able to recognize the error quickly when addressed: invalid synchronous, command or control byte (may never be 0x00), probably invalid command parameters (if 0x00 cannot be accepted here).

A drive that has not been addressed does not check the telegram's content. Such a drive is able to recognize the error only after the next syncronous byte fails to appear. The maximum length of a telegram is 9 bytes. If the error occurs in the address byte, the other axes, which have since received only 0x00, recognize this error only with the 9. byte after the last synchronous byte.

A valid telegram may contain 0x00 only in the address byte (1 byte), in the process data channel (2 bytes) and in the data range of the parameter channel (max. 4 bytes). 0x00 may occur consecutively seven times at the most. Therefore a drive indicating an error status counts how many zero bytes the previous drive is sending. If it receives 0x00 eight times without interruption, this means that also the previous drive is in the error status.

In that case the drive checks the bytes received as to whether a check sequence has been sent. If the drive recognizes a check sequence, it switches to the operating status again.

☞ see Chapter 4.9. Check Sequence

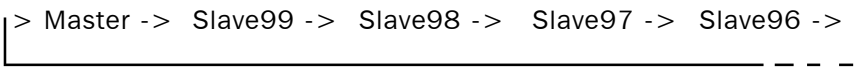
In order to make sure that also the next drive will recognize the error, the drive in the middle must send 0x00 at least nine times. For this subsequent drive to recognize that the previous one is also in the error status, it must receive 8 more zero bytes.

Therefore, every drive that is in the error status sends 0x00 seventeen times. If a drive has recognized that the previous one is in the error status, it increments all bytes received and passes them on.

In case of an error, the master first receives 0x00 twenty-five times at the most, then it receives the address of the drive that has recognized the error. Thereby easy error localization is guaranteed.

Example: 100 axes

The master wants to read 1 Byte of each RAM of the drives No.95 to No.99. Between the drives No.98 and No.97 a bit is transmitted faultily. Drive No.97 identifies a parity error.



The 1. telegram

1. Byte	0x88 =	synchronous byte (followed by address byte and 4 bytes in the parameter channel, no process data)
2. Byte	0xFB =	-5 = 95-100 (slave 95 is addressed, 100 slaves in the ring)
3. Byte	0xC0 =	Read Byte command for ND31/ND32
4. Byte	0x13 =	AddressL, LSB of address 0xFE13
5. Byte	0xFE =	AddressM, MSB of address 0xFE13
6. Byte	0xD1 =	Control Byte (C0+13+FE = 1D1)

NOVOTRON

The following telegrams differ only in the 2. byte (96-100 = -4 = 0xFC, 97-100 = -3 = 0xFD, ...):

Master	Slave 99	Slave 98	Fehler	Slave 97	Slave 96	Slave 95	...	Slave 00
88	88	88		88	88	88	...	88
FB	FC	FD		FE	FF	00		5F
C0	C0	C0		C0	C0	C0		C0
13	13	13		13	13	13		13
FE	FE	FE		FE	FE	>88		88
D1	D1	D1		D1	D1	+A5		A5
88	88	88		88	88	88		88
FC	FD	FE		FF	00	01		60
C0	C0	C0		C0	C0	C0		C0
13	13	13	12	!00	00	00		00
FE	FE	FE		00	!00	00		00
D1	D1	D1		00	00	00		00
88	88	88		00	00	!00		!00
FD	FE	FF		00	00	00		00
C0	C0	C0		00	00	00		00
13	13	13		00	00	00		00
FE	FE	FE		00	00	00		00
D1	D1	D1		00	*00	00		00
88	88	88		00	00	00		00
FE	FF	00		00	00	*00		*00
C0	C0	C0		00	00	00		00
13	13	13		00	00	00		00
FE	FE	88		00	00	00		00
D1	D1	A5		00	00	00		00
88	88	88		00	00	00		00
FF	00	01		#00	00	00		00
C0	C0	C0		00	#00	00		00
13	13	13		00	01	00		00
FE	88	88		00	01	#00		#00
D1	A5	A5		00	01	02	...	61
88	88	88		00	01	02		61
9C	9D	9e		00	01	02		61
.	.	.	.	.	.	.	.	

>: register value read out from response telegram

+: new check sum

!: drive recognizes an error

*: drive recognizes that previous drive is in the error status

#: the 17. zero byte sent after error recognition

☞ The messages 0x00 show the master the transmission error, bytes 0x61= 97 means that the error was recognized by slave 97.

4.9 Check Sequence

When the master has recognized an error during communication, it sends a check sequence to reset the drives. The first seventeen bytes before the actual check sequence are 0x00, so that also the drives that have not yet recognized the error will indicate an error status. After the check sequence has been sent, the master can repeat the sending of the failed telegrams.

The check sequence is:

(17 times 0x00) 0xFF 0x44 0x72 0x4C 0x41

Example:

100 axes in total, axes No.97 - No.0 are indicating an error status, axes No.99 - No.98 have not recognized an error during communication. The master is just about to send a telegram when it recognizes the error because of an unexpected zero byte. After it has recognized the error, it sends 0x00 seventeen times. Then it resets the bus by means of a check sequence.

> Master -> Slave99 -> Slave98 -> Slave97 -> Slave96 -> _ _ _

	Master	Slave99	Slave98	Slave97	Slave96	Slave95	...	Slave00
Telegram in process	88	88	88	00	01	02	...	61
17 zero bytes	00	01	02	00	01	02	...	61
	00	00	00	00	01	02		61
	00	00	00	00	01	02		61
	00	00	00	00	01	02		61
	00	!00	!00	00	01	02		61
	00	00	00	00	01	02		61
	00	00	00	00	01	02		61
	00	00	00	00	01	02		61
	00	00	00	*01	02	03		62
	00	00	00	01	02	03		62
	00	00	00	01	02	03		62
	00	*00	*00	01	02	03		62
	00	00	00	01	02	03		62
	00	00	00	01	02	03		62
	00	00	00	01	02	03		62
	00	00	00	01	02	03		62
	00	00	00	01	02	03		62

Check code	FF	@FF	@FF	@FF	@FF	@FF	...	@FF
	44	44	44	44	44	44		44
	72	72	72	72	72	72		72
	4C	4C	4C	4C	4C	4C		4C
	41	41	41	41	41	41		41
Repeat of telegram	88	88	88	88	88	88	...	88
	FC	FD	FE	FF	00	01		60
	C0	C0	C0	C0	C0	C0		C0
	13	13	13	13	13	13		13
	FE	FE	FE	FE	>88	88		88
	D1	D1	D1	D1	+A5	A5		A5

Notes:

!: drive recognizes an error

*: drive recognizes that previous drive is in the error status

@: drive recognizes first byte of check sequence

>: register value read out from response tlegram

+: new check sum

☞ After the check sequence procedure has ended, the bus is ready to operate again.

NOVOTRON

5 CAN Bus

5.1 General Information

As an option, NOVODRIVE offers a CAN interface CAN 2.0 A and B according to ISO/DIS 11898. The software supports only 11-bit identifiers.

As another option, the CAN bus can be delivered with galvanic isolation.

5.2 Baud Rate

This section refers to special applications only. As a standard, a baud rate of 1 MBit/s is set.

You may control the settings in the following registers:

Register	Address	Value
CANInitBTR0	0xFEE8	0x40
CANInitBTR1	0xFEE9	0x25

Configuration

NOVODRIVE is equipped with an Intel CAN Controller 82527. The bit rate is determined by two parameters in the parameter set, by which two registers in the CAN Controller of NOVODRIVE are initialized. The bit rate can be modified by means of these two parameters (reset to activate).

CANinitBTR0	7	6	5	4	3	2	1	0
0xFEE8	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Function	SJW		BRP					
Standard value for 1 MBit	0	1	0	0	0	0	0	0

Explanation **BRP** Baud Rate Prescaler

CANinitBTR1	7	6	5	4	3	2	1	0
0xFEE9	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Function	SPL	TSEG2			TSEG1			
Standard value for 1 MBit	0	0	1	0	0	1	0	1

Explanation

TSEG1 Time Segment 1

TSEG2 Time Segment 2

Example 1

Baud rate = 10 MHz / ((BRP + 1) * (3 + TSEG1 + TSEG2))

For 1 MBit applies:

Baud rate = 10 MHz / ((0 + 1) * (3 + 5 + 2)) = 1 MHz

A single bit on the CAN Bus is constituted by three segments:

$t_{Sync}+t_{TSEG1}+t_{TSEG2}$ length of one bit

$t_{Sync}+t_{TSEG1}$ time before sampling point

t_{TSEG2} time after sampling point

The length of the three segments is programmable. The three segments are constituted by a number of so-called time quanta t_q. The length of a t_q is determined with the parameter **CANInitBTR0**. For BRP a value range from 0 to 63 is permitted. For one t_q applies:

t_q = (BRP+1)*100 ns

t_{Sync} always equals 1 t_q

Bits 7 and 6 in **CANInitBTR0** are 0 and 1 and should not be modified.

By means of **CANInitBTR1** the length of t_{TSEG1} and t_{TSEG2} is determined. The SPL bit determines the sampling mode:

1 = each bit gets sampled three times

0 = each bit gets sampled one time

In case of bit rates exceeding 100 kBit, sampling may be done only once.

The length of t_{TSEG2} is:
t_{TSEG2} = (TSEG2+1)*t_q valid values for TSEG2 are 1 – 7

The length of t_{TSEG1} is:
t_{TSEG1} = (TSEG1+1)*t_q valid values for TSEG1 are 2 – 15

Example 2 **1 MBit/s:**

CANInitBTR0 = 0x40
CANInitBTR1 = 0x16

BPR = 0 -> t_q = (0+1)*100 ns = 100 ns
t_{Sync} = 1 tq = 100 ns

TSEG2 = 1
t_{TSEG2} = (1+1)*100 ns = 200 ns

TSEG1 = 6
t_{TSEG1} =(6+1)*100 ns = 700 ns
$t_{Sync}+t_{TSEG1}+t_{TSEG2}$ = 100 ns+200 ns+700 ns =1 μs

The bit gets sampled after 700 ns.

Example 3 **500 kBit/s**

CANInitBTR0 = 0x40
CANInitBTR1 = 0x2F

BPR = 0 -> t_q = (0+1)*100 ns = 100 ns
t_{Sync} = 1 tq = 100 ns

TSEG2 = 2
t_{TSEG2} = (2+1)*100 ns = 300 ns

TSEG1 = 6
t_{TSEG1} =(15+1)*100 ns = 1600 ns
$t_{Sync}+t_{TSEG1}+t_{TSEG2}$ = 100 ns+300 ns+1600 ns =2 μs

The bit gets sampled after 1700 ns.

Notes

- Choose t_q as small as possible.
- Place the sampling point in the bit as far back as possible (allows larger line lengths).

NOVOTRON

6 CAN Protocol NOVOTRON

6.1 Service Channel

For operations that are not time-critical a service channel with various commands is available.

6.1.1 NOVODRIVE Message Objects

Message object **CAN Service Receive (receiving commands)**

Identifier Bit	10	9	8	7	6	5	4	3	2	1	0
Register	7	6	5	4	3	2	1	0	4	3	2
	CANService								**CANIDLSB**		
Address	0xFF38								0xFED2		

Message object **CAN Service Transmit (sending response)**

Identifier Bit	10	9	8	7	6	5	4	3	2	1	0
	(Identifier CAN Service Receive) + 1										

☞ If **CANService** is set to 0, the CAN Bus gets deactivated after reset.

6.1.2 Determination Of CANService Identifier Over Digital Inputs

For generating the identifiers for the service channel also digital inputs can be used. Selection is made by means of the **CANServiceSel** parameter.

Register	**CANServiceSel**
Address	0xFEEC
Size	16 Bit

Function

A ‘0’ bit means: Value of digital input is ignored.
A ‘1’ bit means: Digital input is used for generating **CANService**.

Meaning of the bits in **CANServiceSel**:

Bit0	DipSwitch 1	(D1)	On = 0
Bit1	Dipswitch 2	(D2)	On = 0
Bit2	Dipswitch 3	(D3)	On = 0
Bit3	Dipswitch 4	(D4)	On = 0
Bit4	Dipswitch 5	(D5)	On = 0
Bit5	Dipswitch 6	(D6)	On = 0
Bit6	Dipswitch 7	(D7)	On = 0
Bit7	Dipswitch 8	(D8)	On = 0
Bit8	Tast8	(T8)	24 V or open = 0
Bit9	Tast7	(T7)	24 V or open = 0
Bit10	Tast6	(T6)	24 V or open = 0
Bit11	Tast5	(T5)	24 V or open = 0
Bit12	GPIn2	(G2)	0 V or open = 1
Bit13	GPIn6	(G6)	0 V or open = 1
Bit14	GPIn7	(G7)	0 V or open = 1
Bit15	GPIn10	(G10)	0 V or open = 1

Of bits 0 - 10 of the CAN Service identifier, as many bits are modified, (starting with Bit 0) as digital inputs have been selected. The other bits remain unchanged.

If more than 11 bits are selected, only the first eleven bits are taken into consideration.

The function is switched off by **CANServiceSel** = 0.

The Dipswitch 1 - 8 inputs are available as an option only in case of 19“ devices. All other inputs are standard features.

☞ If Dipswitch 7 = On, the encoder emulator and the RS442 interface are switched off!

Example 1:

CANServiceSelect:	B’0000 0000 0000 1111
CANServiceIDReceive:	B’uuuu uuuu uuu u D4 D3 D2 D1
CANServiceIDSend:	B’uuuu uuuu uuu u D4 D3 D2 D1 + 1

Example 2:

CANServiceSelect:	B’0100 1111 0000 0001
CANServiceIDReceive:	B’uuuu uuuu uu G7 T5 T6 T7 T8 D1
CANServiceIDSend:	B’uuuu uuuu uu G7 T5 T6 T7 T8 D1+1

NOVOTRON

6.2 Commands for ND31/ND32 on the service channel

Abbreviations used:

A0	Low Byte of address
A1	High Byte of address
D0	LSB of data (8-bit , 16-bit and 32-bit values)
D1	...
D2	...
D3	MSB of data (32-bit values)
CS	the check sum computed by byte-wise adding of all bytes of the command telegram
NCS	two's complement of the check sum
-	ignored

☞ In case of 16-bit and 32-bit data, the address refers to the MSB.

6.2.1. Access To Integrated Memory

Command **Read Byte**

Function Reading out an 8-bit value from the integrated memory

Byte	0	1	2	3	4	5	6	7
Command telegram	0xC0	A0	A1	CS	-	-	-	-
Response telegram	0xC0	A0	D0	NCS	-	-	-	-
Address range	0x0000 - 0xFFFF							

Command **Read Word**

Function Reading out a 16-bit value from the integrated memory

Byte	0	1	2	3	4	5	6	7
Command telegram	0xC1	A0	A1	0x3F	CS	-	-	-
Response telegram	0xC1	A0	D0	D1	NCS	-	-	-
Address range	0x0000 - 0xFFFF							

Command **Read Long**

Function Reading out a 32-bit value from the integrated memory

Byte	0	1	2	3	4	5	6	7
Command telegram	0xC7	A0	A1	0x3F	0x3F	0x3F	CS	-
Response telegram	0xC7	A0	D0	D1	D2	D3	NCS	-
Address range	0x0000 - 0xFFFF							

Write Byte

Command Writing an 8-bit value to the integrated memory

Function

Byte	0	1	2	3	4	5	6	7
Command telegram	0x82	D0	A0	A1	CS	-	-	-
Response telegram	0x82	D0	A0	A1	NCS	-	-	-
Address range	0xFE00 - 0xFE7F, 0xFEA0 - 0xFF7F							

Write Word

Command Writing a 16-bit value to the integrated memory

Function

Byte	0	1	2	3	4	5	6	7
Command telegram	0x63	D0	D1	A0	A1	CS	-	-
Response telegram	0x63	D0	D1	A0	A1	NCS	-	-
Address range	0xFE00 - 0xFE7F, 0xFEA0 - 0xFF7F							

Command **Write Long**

Function Writing a 32-bit value to the integrated memory

Byte	0	1	2	3	4	5	6	7
Command telegram	0xC8	D0	D1	D2	D3	A0	A1	CS
Response telegram	0xC8	D0	D1	D2	D3	A0	A1	NCS
Address range	0xFE00 - 0xFE7F, 0xFEA0 - 0xFF7F							

6.2.2. Bit Manipulation

Command **AND**

Function Bit-wise logical AND with a register in the integrated memory. The upper 8 bits of the address are always 0xFF.

Byte	0	1	2	3	4	5	6	7
Command telegram	0xA4	D0	A0	CS	-	-	-	-
Response telegram	0xA4	D0	A0	NCS	-	-	-	-
Address range	0xFF00 - 0xFFFF							

Command **OR**

Function Bit-wise logical OR with a register in the integrated memory. The upper 8 bits of the address are always 0xFF.

Byte	0	1	2	3	4	5	6	7
Command telegram	0xA5	D0	A0	CS	-	-	-	-
Response telegram	0xA5	D0	A0	NCS	-	-	-	-
Address range	0xFF00 - 0xFFFF							

6.2.3. Access To External Memory

Command **Read Word X**

Function Reading out a 16-bit value from the external memory

Byte	0	1	2	3	4	5	6	7
Command telegram	0xC9	A0	A1	0x3F	CS	-	-	-
Response telegram	0xC9	A0	D0	D1	NCS	-	-	-
Address range	0x0000 - 0xFFFF							

Command **Read Long X**

Function Reading out a 32-bit value from the external memory

Byte	0	1	2	3	4	5	6	7
Command telegram	0xCB	A0	A1	0x3F	0x3F	0x3F	CS	-
Response telegram	0xCB	A0	D0	D1	D2	D3	NCS	-
Address range	0x0000 - 0xFFFF							

Command **Write Word X**

Function Writing a 16-bit value to the external memory

Byte	0	1	2	3	4	5	6	7
Command telegram	0x6A	D0	D1	A0	A1	CS	-	-
Response telegram	0x6A	D0	D1	A0	A1	NCS	-	-
Address range	0x4000 - 0xFFFF							

Command **Write Long X**

Function Writing a 32-bit value to the external memory

Byte	0	1	2	3	4	5	6	7
Command telegram	0x6C	D0	D1	D2	D3	A0	A1	CS
Response telegram	0x6C	D0	D1	D2	D3	A0	A1	NCS
Address range	0x4000 - 0xFFFF							

6.2.4. Control Commands

Command **Reset**

Function Reset of NOVODRIVE

Byte	0	1	2	3	4	5	6	7
Command telegram	0xDD	0x21	CS	-	-	-	-	-
Response telegram	-	-	-	-	-	-	-	-

6.3 Process Data Channel

The process data channel provides a cyclic data exchange between central computer and NOVODRIVE.

The process goes as follows:

1. The central computer sends NOVODRIVE a process-data write-telegram of up to 8 bytes.

2. The central computer sends NOVODRIVE a sync telegram.

3. After NOVODRIVE has received the sync telegram, the data of the process-data write-telegram are copied to the registers **CANinput1** - **CANinput4**. Then the process-data read-telegram is compiled and sent.

The software modules of NOVODRIVE have access to the data of the process-data write-telegrams over pointers. The process-data read-telegrams can be configurated arbitrarily over address fields.

6.3.1 NOVODRIVE Message Objects

Message object **CAN Time (receiving sync telegrams)**

Identifier Bit	10	9	8	7	6	5	4	3	2	1	0
Register	7	6	5	4	3	2	1	0	15	14	13
	CANTime								**CANIDLSB**		
Address	0xFF3B								0xFED2		

Message object **CAN Master (receiving process-data write-telegrams)**

Identifier Bit	10	9	8	7	6	5	4	3	2	1	0
Register	7	6	5	4	3	2	1	0	12	11	10
	CANMaster								**CANIDLSB**		
Address	0xFF39								0xFED2		

Message object **CAN Slave (sending process-data read-telegrams)**

Identifier Bit	10	9	8	7	6	5	4	3	2	1	0
Register	7	6	5	4	3	2	1	0	7	6	5
	CANSlave								**CANIDLSB**		
Address	0xFF3A								0xFED2		

6.3.2 Configuration

CANCFG	7	6	5	4	3	2	1	0	Bit
0xFF11	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Bit 0 reserved = 0
Bit 1 1 = send process-data read-telegrams
Bit 2 - 5 length of process-data read-telegram
0 - 8 Byte
Bit 6 sequencing of bytes of process-data write-telegrams
0 = Little Endian Mode
1 = Big Endian Mode
Bit 7 1 = Special Mode (see below)

6.3.3 Assignment of process-data read-telegrams

The content of process-data read-telegrams is selected over address fields.
You may select all registers that lie within the address range 0xFE00 - 0xFFFF of the integrated memory.

Register	Address	Content
CANSlaveBit8	0xFF08	Selection bits for address range. If Bit i is set, the address range 0xFF00 - 0xFFFF is selected for the telegram byte i. If Bit i is zero, the address range 0xFE00 - 0xFEFF applies.
CANSlaveB0	0xFF09	the lower 8 bits of the address for telegram byte 0
CANSlaveB1	0xFF0A	the lower 8 bits of the address for telegram byte 1
CANSlaveB2	0xFF0B	the lower 8 bits of the address for telegram byte 2
CANSlaveB3	0xFF0C	the lower 8 bits of the address for telegram byte 3
CANSlaveB4	0xFF0D	the lower 8 bits of the address for telegram byte 4
CANSlaveB5	0xFF0E	the lower 8 bits of the address for telegram byte 5
CANSlaveB6	0xFF0F	the lower 8 bits of the address for telegram byte 6
CANSlaveB7	0xFF10	the lower 8 bits of the address for telegram byte 7

Example	Register	Content	Telegram structure
	CANSlaveBit8	1000 0001	
	CANSlaveB0	0xF2	0xFFF2 **Status**
	CANSlaveB1	0x68	0xFE68 **nIst** (MSB)
	CANSlaveB2	0x69	0xFE69 **nIst** (LSB)
	CANSlaveB3	0x7C	0xFE7C **LageIst** (MSB)
	CANSlaveB4	0x7D	0xFE7D **Lageist** (LSB)
	CANSlaveB5	0x00	0xFE00 **errorcode** (MSB)
	CANSlaveB6	0x01	0xFE01 **errorcode** (LSB)
	CANSlaveB7	0xFF	0xFFFF

6.3.4 Process-Data Write-Telegram Byte Order

The byte order (Big Endian Mode or Little Endian Mode) can be determined by register **CANCfg**.

Process-data write telegram (8 bytes)							
b0	b1	b2	b3	b4	b5	b6	b7

CANCfg Bit 6		1		0	
		Big Endian Mode		Little Endian Mode	
Register	Address	MSB	LSB	MSB	LSB
CANinput1	0xFE3C	b0	b1	b7	b6
CANinput2	0xFE3E	b2	b3	b5	b4
CANinput3	0xFE40	b4	b5	b3	b2
CANinput4	0xFE42	b6	b7	b1	b0

The telegram data in **CANinput1** ... **CANinput4** can be accessed over pointers.

Pointer name	Description	Address	Access	Bit	Standard value
?LageSollExt	in position control mode: position setpoint	0xFEA6	R/W	16	0
?Rampe+	acceleration ramp	0xFEA8	R/W	16	**Rampe+**
?Rampe-	braking ramp	0xFEAA	R/W	16	**Rampe-**
?SchRampe	quick-stop ramp	0xFEC4	R/W	16	**SchRampe**
?msoll	in torque control mode: torque setpoint	0xFEBC	R/W	16	**msoll**
?FIUmdr	input value of fine interpolator or PS0	0xFECC	R/W	16	-
?FILage	input value of fine interpolator or PS0	0xFEB0	R/W	16	-
?CANControl	see **CAN_Control**	0xFEB2	R/W	16	**CANControl**
?mmax	limitation of peak torque	0xFF36	R/W	16	**mmax16**
?Sollwert	speed setpoint	0xFEBE	R/W	16	**Sollwert**
?GPO	output values for digital outputs	0xFEC2	R/W	16	**GPO**
?AnOut1	output value of Analog Output 1	0xFF24	R/W	16	-
?AnOut2	output value of Analog Output 2	0xFF12	R/W	16	-
	or PS0 positioning speed				
?FForward	speed pre-control	0xFEB6	R/W	16	0

Example **Little Endian Mode**

?Sollwert = address of **CANinput2**
?CANControl = address of **CANinput1**

The speed setpoints for the controller must now be given in bytes b5 and b4 of the process-data write-telegram.

The controller can be enabled and started/stopped over byte b7.

6.3.5 Special Mode

The configuration of the process-data write-telegram is as follows:

b0	b1	b2	b3	b4	b5	b6	b7
CANControl	unused	unused	unused	unused	unused	unused	unused
CANinput1		**CANInput2**		**CANInput3**		**CANInput4**	

The configuration of the process-data read-telegram is as follows:

b0	b1	b2	b3	b4	b5	b6	b7
Status	**nist** MSB	**nist** LSB	**LageIst** MSB	**LageIst** LSB	**?CANOut** MSB	**?CANOut** LSB	-

6.3.6 Timeout

If the sync telegrams fails to appear for a period longer than indicated in **CANTimeout**, the drive gets blocked and Error 518 is generated. The monitoring starts with the first sync telegram.

Register	**CANTimeout**
Address	0xFF16
Size	8 Bit
Access	R/W
Function	timeout value for process data channel
Scaling	10 ms/bit
Value range	0x00 no monitoring 0xFF no monitoring 0x01 ... 0xFE monitoring on

6.3.7 Bus Reactivation

After 255 transmission errors have occurred, the CAN controller switches into the BOFF (= bus-off) status. After a BOFF NOVODRIVE can provide for an automatic reactivation of the CAN Bus.

SWVersion	7	6	5	4	3	2	1	0
0xFF3D	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

0 —— no reactivation of CAN Bus after BOFF

1 —— reactivation of CAN Bus after BOFF

6.4 Fine Interpolator

The fine interpolator is available starting with H8 Version of 06/05/1997.

6.4.1 Functionality

The central computer cyclically sends position setpoints to NOVODRIVE over the CAN Bus. NOVODRIVE is given the cycle time by the register **Tcycle**. The fine interpolator computes from these data a speed setpoint for NOVODRIVE, so that the position setpoint is achieved with the next sync telegram.

The position setting can be done with 32 bits (absolute mode) or with 16 bits (relative mode). It can have a 16-bit or a 12-bit resolution per revolution. By means of the **FIVerrundung** parameter, the intersection points of the linear interpolation can be smoothened. The smoothening is switched off with the value 0. The higher the value, the stronger the smoothening. To the extent the smoothening gets stronger, the tracking error increases.

A relative setpoint or the difference between the actual setpoint and the previous absolute setpoint may never be more than ±28670 increments (16-bit resolution), otherwise there will be an overflow.

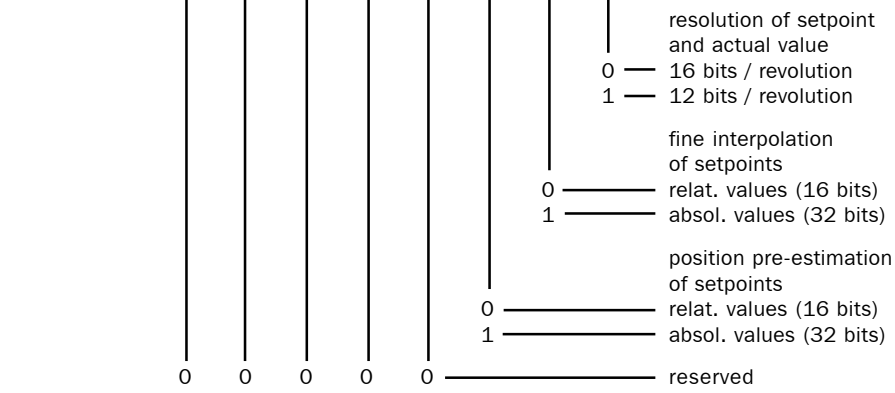
By means of the position pre-estimation the actual values are computed for the time of the next sync telegram. Thereby it is possible to send back more exact actual values to the central computer. The actual values can have a size of 16 bits (relative mode) and 32 bits (absolute mode), respectively. The resolution can be 16 bits or 12 bits per revolution.

Before the controller is enabled, the position setpoints and the actual position of the drive must be adjusted to prevent the motor run in an uncontrolled way!

The CAN timeout monitoring should be active in order to prevent that the drive continues to move along at constant speed when the connection is interrupted.

6.4.2 Register

FICSR	R	R	R	R	R	R/W	R/W	R/W
0xFF06	7	6	5	4	3	2	1	0



Register	**?512usA**
Address	0xFEC8
Size	16-bit unsigned
Access	R/W
Function	activation of fine interpolator
Value range	Name Address Notes **@Dummy** 0x01EA fine interpolator not active **@Feininterpolator** 0x01E8 fine interpolator active

Register	**FIVerundung**
Address	0xFEEE
Size	8-bit unsigned
Access	R/W
Function	smoothening of linear interpolation
Value range	0 = no smoothening 1 ... 10 smoothening

Register	Tcycle
Address	0xFEEA
Size	16-bit unsigned
Access	R/W
Function	cycle time of 1 ms ... 6,5 ms
Scaling	200 ns /bit

Register	?FILage		
Address	0xFEB0		
Size	16-bit unsigned		
Access	R/W		
Function	pointer at lower 16 bits of position setpoint for fine interpolation		
Value range	Name **CANInput1** **CANInput2** **CANInput3** **CANInput4**	Address 0xFE3C 0xFE3E 0xFE40 0xFE42	Notes see Chapter 6.3.4

Register	?FIUmdr		
Address	0xFECC		
Size	16-bit unsigned		
Access	R/W		
Function	pointer at upper 16 bits of position setpoint for fine interpolation (necessary only for absolute position setting)		
Value range	Name **CANInput1** **CANInput2** **CANInput3** **CANInput4**	Address 0xFE3C 0xFE3E 0xFE40 0xFE42	Notes see Chapter 6.3.4

Register	FISoll
Address	0xFE1A
Size	16-bit signed
Access	R
Function	the speed setpoint computed by the fine interpolator; setpoint setting is switched by setting **?Sollwert** to fine interpolator
Scaling	see manual Basic Functions, Chapter 3.6.5

Register	Pa0
Address	0xFE6A
Size	16-bit signed
Access	R
Function	estimated actual position value at the time of next sync telegram; in absolute mode the register contains the upper 16 bits of the position, in relative mode the register is not defined
Scaling	see manual Basic Functions, Chapter 3.6.3.5

Register	Pa1
Address	0xFE6C
Size	16-bit signed
Access	R
Function	estimated actual position value at the time of next sync telegram; in absolute mode the register contains the lower 16 bits of the position, in relative mode the register contains the position change
Scaling	see manual Basic Functions, Chapter 3.6.3.5

6.4.3 Initialization

1. Stop NOVODRIVE.
2. Switch on position controller by setting register **?nsoll** to the address of **nsoll2**.
3. Setpoint setting is done over register **FISoll** (**?Sollwert** = **FISoll**).
4. Set pointers **?FIUmdr** and **?FILage** to the process-data write-telegram in **CANinput1** ... **CANinput4**.
5. Set cycle time in register **Tcycle**.
6. Switch on timeout monitoring (see Chapter 6.3.6).
7. Do the configuration of the fine interpolator over register **FICSR**.

☞ It is recommended that the acceleration ramp and the braking ramp be switched off (see manual Basic Functions, Chapter 3.6.3.4) in order to prevent that the target position gets falsified when the setpoints exceed the ramp values.

8. Activate the function by setting register **?512usA** to **@Feininterpolator**.
9. Adjust the position value of the control to the position value of the drive. To do so, you may read out the actual position value cyclically over the process-data read-telegrams. Alternatively, you may adjust the position value of NOVODRIVE to the position value of the control by means of actual-value setting (see manual Additional Functions, Chapter 6).

10 Before starting the fine interpolation, the cyclic data exchange must be done over the process data channel.
11 It is not until this point that NOVODRIVE may be set into the start status.

6.4.4 Examples

32-bit setting (absolute) with 16-bit resolution:
- initialize **FICSR** with 0xxx xx10

32-bit setting (absolute) with 12-bit resolution:
- initialize **FICSR** with 0xxx xx11

16-bit setting (relative) with 16-bit resolution:
- initialize **FICSR** with 0xxx xx00

16-bit setting (relative) with 12-bit resolution:
- initialize **FICSR** with 0xxx xx01

32-bit setting (absolute), Little Endian Mode bytes b0 – b3 of CAN process-data write telegram (Bit 6 in **CANCFG** = 0):
- enter 0xFE42 in **?FILage**
- enter 0xFE40 in **?FIUmdr**

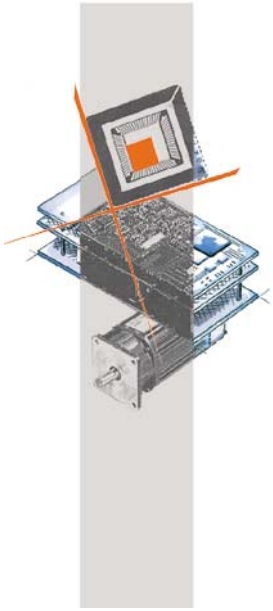
32-bit setting (absolute), Little Endian Mode bytes b4 – b7 of CAN process-data write telegram (Bit 6 in **CANCFG** = 0):
- enter 0xFE3E in **?FILage**
- enter 0xFE3C in **?FIUmdr**

32-bit setting (absolute), Big Endian Mode bytes b0 – b3 of CAN process-data write telegram (Bit 6 in **CANCFG** = 1):
- enter 0xFE3C in **?FILage**
- enter 0xFE3E in **?FIUmdr**

32-bit setting (absolute), Big Endian Mode bytes b4 – b7 of CAN process-data write telegram (Bit 6 in **CANCFG** = 1):
- enter 0xFE40 in **?FILage**
- enter 0xFE42 in **?FIUmdr**

16-bit setting (relative), Big Endian Mode bytes b0 – b1 of CAN process-data write telegram (Bit 6 in **CANCFG** = 1):
- enter 0xFE3C in **?FILage**

32-bit setting (relative), Little Endian Mode bytes b0 – b1 of CAN process-data write telegram (Bit 6 in **CANCFG** = 0):
- enter 0xFE42 in **?FILage**



NOVOTRON

für Dynamik und Bewegung

N O V O T R O N

Industrie - Automation GmbH
Mauserstrasse 31
D - 71640 Ludwigsburg
Telefon 07141/2969 - 0
Telefax 07141/2969 - 22
e-mail: info@novotron-online.com
http: //www.novotron-online.com