ECKELMANN

E•xNC55

Interface HMI <-> NCR

Author:

Copyright protection:

File name: cnc55_mmi-nc.doc

Version: 2.1 First release: 11/1998

Release:

Modification protocol

Chapter	Date	Person in charge	Modification	Release Date / Initials
All all	07/11/05 02/06	T. Ebert WN	General editorial revision of CNC55 corrections	

Table of contents

Definition of terms and abbreviations

HMI	Human-Machine-Interface
NCR	NC kernel
PLC	Integrated sequential control (PLC)
CNC	NC kernel with PLC
DPR	Dual-Port RAM
SW	Software
Download	Transfer of firmware, PLC and other files from the PC to the CNC controller
Booting	Putting of the entire system in an operating condition by means of self-test, download and loading of the application software

Used data types:

CHAR	8-bit integer with sign, value range –128 ... +127
UCHAR	8-bit integer without sign, value range 0 ... 255
SHORT	16-bit integer with sign, value range –32768 ... +32767
USHORT	16-bit integer without sign, value range 0 ... 65535
LONG	32-bit integer with sign, value range -2^{31} ... $+2^{31}-1$
ULONG	32-bit integer without sign, value range 0 ... $2^{32}-1$
FLOAT	Single precision real in IEEE format, range of representation $\pm10^{-38}$ ... $\pm10^{38}$
DOUBLE	Double precision real in IEEE format, range of representation $\pm10^{-308}$ ... $\pm10^{308}$
RECORD	Data structure
BITn	Single bit of a word with $0 \le n \le 15$, n=0 is the least significant bit
var[m]	Identifies a variable *var* as array with m elements
var[m][n]	Identifies a variable *var* as two-dimensional array with m*n elements

1 Introduction

This document describes the interface between the NC kernel (NCR), the CNC controllers CNC55/PNC55/ENC55 and a human-machine-interface (HMI). The description is made only with regard to a technical data point of view. Both the physical realization and the technical data are included in the hardware description.

In the description it is assumed that the HMI is connected to the NC controller via a dual-port RAM (DPR). In case of the PNC55, the DPR is part of the controller. In case of the CNC55 and the ENC55, the connection is made by a virtual DPR that is updated via the Ethernet interface. If the term CNC55 is mentioned in the following text, the information always refers to all three controller variants.

Moreover, the NCR includes a SW package for the HMI that carries out the downloading of all operating programs and the configuration of the DPR, i.e. the HMI gateway MMIGTWAY.EXE and the communication DLL MMICTRL.DLL. A separate description is available.

This description only includes the standard data in the DPR. Application-specific data and areas of the DPR as well as their meaning are defined in separate documents. The present specification, however, is always the basis.

The use of the interface is made in two phases:

- Booting with RAM test and configuration
- Normal operation, i.e. application-specific data exchange between the HMI and the NCR

Booting is made via the above-mentioned SW package that is included in the software of the HMI. Exchange of the NC data and messages is made in normal operation, and sequence and evaluation is exclusively determined by the operator prompting.

In the following text, a cross-definition of the interface is described first. A short overview of the sequence of booting follows as well as its configuration options with indications to the integration into the HMI program. The last section includes a description of the data exchange between HMI and NCR. All data and admitted messages are defined.

2 Definition of the interface

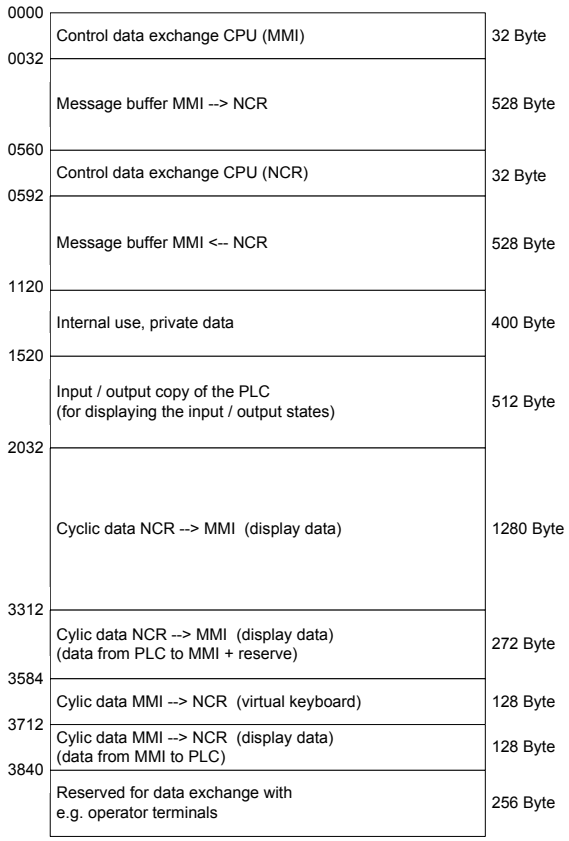
2.1 Structure of the DPR

Data exchange between the NC computer and an operating computer (human-machine-interface, HMI) is made logically via a dual-port RAM of 4 KByte size. As far as the data are concerned, this RAM is subdivided into the following areas:

- Data from NCR to HMI
- Data from HMI to NCR
- Input/output copy of PLC
- Private data from NCR

The detailed structuring of the DPR is shown in the following figure.

Detailed reference is made in a separate section to the data contents of the single blocks that are transferred between the HMI and the NCR. Structure of the DPR:



3 Booting

3.1 Sequence

Upon energizing of the controller, firmware and PLC programs are to be transferred first to the controller, unless they are not firmly installed in the controller. This procedure is called "booting". The procedure is to be initialized by the HMI. A program is available to the HMI for this purpose that carries out the communication and the data exchange via the DPR and in the HMI.

The program must have been started before a data exchange is possible with the NC computer. An access to the data of the DPR in normal operation is only possible after a successful booting.

Booting sequence:

1) Test of the entire DPR from the NC and PC side.
2) Booting of the communication processor by the DPR, if the controller is connected to the PC via a special coupling card (e.g. ET-PC 01).
3) Booting of the NC computer via DPR
4) Loading of the PLC into the NC computer via the DPR
5) Configuration phase in which the position of the communication ranges are established within the DPR.

At this point, the detailed sequence of the single phases is irrelevant. The entire booting is made via MMICTRL.DLL and MMIGTWAY.EXE. The interface to the application is included in the document MMICTRL.PDF.

3.2 Configuration

Booting is controlled by a configuration file. The name of this file can be freely selected and is transferred to function ncrLoadFirmware after calling. This file is structured on the basis of a fixed scheme.

The <computer type> is at the beginning of each single configuration input and is completed by a semicolon. The name of the file follows that is to be loaded into the computer, also completed by a semicolon. No firmware is loaded into the respective computer if the file list is empty ("<typ>;;").

All elements of a configuration input must be separated by at least one blank. Comments can be entered in the configuration file between '/*' and '*/'.

The order of inputs is optional, only the input <ncr> must always be the last input in the configuration file. If more than one card of the same type is configured, these cards are called in their physical order.

Syntax:

<computer type> ; [<file name>] ;

Computer type:

Key word for the computer to which the file is to be transferred.

Possible computer types:

Key word	Computer to be booted
tmmi	Communication processor on the coupling card (ET-PC-01)
tnc	Processor of the NC computer (firmware)
ncr	Processor of the NC computer (PLC)

File name:

Name (incl. indication of path) of the file that is to be loaded into the computer.

Example:

```
tnc;    bin\cnc55.rsc;        /* Code for the CNC55-CPU */
ncr;    bin\machine.prg;      /* PLC code for the CNC55-CPU */
```

4 Data transfer

4.1 General

Data transfer is structured into the following three areas:

a) Cyclic transfer of state information; this is made automatically without external initiation. Data exchange is made in two directions.

b) Calling of data, messages, commands, acknowledgments. When requesting data, one of the two subscribers takes the initiative and expects a response in the form of data transfer. Messages are made without being called and do not require a response of the other subscriber. Commands trigger determined actions with the other subscriber and require an acknowledgment. The transferring subscriber waits for the acknowledgment prior to be able to continue its operation. Acknowledgments are the confirmation of a received command.

c) Data transfer

Data transfer is made in blocks. The single blocks are identified (1st block, subsequent block, last block). The next block can be transferred as soon as the previous one has been taken, i.e. as soon as the message buffer is empty and an acknowledgment has been made. The length of transferred user data is indicated for each block. A transfer always comprises at least two blocks: the 1st block and the last block. The last block has the length "0" if all user data have already been transferred in the first block.

Each transferred block is to be acknowledged by the receiving side with a special acknowledgment message. Only subsequently, the next block can be transferred. For further details please check section 5.2.

Data transfer as per b) and c) is made via the message buffer. The message buffer has a length of 512 user data plus the necessary control characters (identification, length, check flag).

4.2 Cyclic data

Cyclic data are continuously transferred without request. The respective data can be read at any time. It is to be observed, however, that the data are written asynchronously and that they can, therefore, vary during reading. For this reason, check flags are provided for single data fields on the basis of which it is possible to recognize whether the data just read are consistent. The flags are at the beginning and at the end of a data group. With each modification of the data, they are incremented by the transmitter at the beginning and at the end of the writing procedure and include the same contents.

In case of data that are needed not only for mere displaying but also for synchronization, the transmitter is to store the first check flag prior to the reading and the second check flag after the reading. Subsequently, the stored check flags are to be compared and reading is to be repeated in case of inequality. It is even better to synchronously request the important data from the parameter field of the controller by means of message communication.

Cyclic data are, among others, the display and state data transferred by the NCR to the HMI as well as the virtual keyboard transferred from the HMI to the NCR. The ranges can be seen in the RAM structure. The single saved blocks can be seen in section "Data contents".

4.3 Messages

Messages are synchronized data that require the acknowledgment by the receiver before a new block can be transferred. Messages are always transferred in a standardized form:

SB0	SB2	SB1	IDX	MOD	HDL	LEN	User data (max. 512 bytes)
0	1	2	3	4	5	6	8

Synchronization of data transfer is made via check flag. A detailed description of the message transfer is included in the description of the message buffer in section 5.1.2.

The length of the user data must be max. 512 bytes. Data fields exceeding 512 bytes are to be subdivided accordingly. The actual length of the transferred user data is stored in LEN.

The control data are integral part of the message buffer. They occupy the first 8 bytes. The user data range has a fixed length of 512 bytes.

„SB0", „SB1“ and „SB2" are used to identify the message. They inform the receiver about the further processing. The identification "MOD" includes a transmitter identification. In case of messages from the HMI to the NC, this transmitter identification is always to be set to 15. In the HMI, the transmitter identification is irrelevant.

In case of all responses, „HDL“ and „IDX“ are passed by the controller and are primarily used for the assignment of a response message to a determined request in the HMI or the MMIGTWAY.

The determination of the control blocks and the definition of the messages are made in chapter "Data contents".

5 Data contents

In this chapter, the contents of the single data blocks in the dual-port RAM are described in detail. The DPR is subdivided into structures and substructures. The structuring of the following chapter corresponds to the hierarchy of the DPR.

Basic structure

Offset	Name	Type	Meaning
0	*mmi_r*	RECORD	Communication range from HMI to NCR
560	*nc_r*	RECORD	Communication range from NCR to HMI
1120	*private_r*	RECORD	Private communication range od the NCR
1520	*sps_r*	RECORD	Input and output signals of the PLC
2032	*mc2mmi_r*	RECORD	Cyclic data from NCR to HMI
3312	*plc2mmi_auc[256]*	UCHAR	Cyclic data from PLC to HMI
3584	*tast_r*	RECORD	Virtual keyboard from HMI to NCR
3712	*mmi2plc_auc[128]*	UCHAR	Cyclic data from HMI to PLC
3840	*mmi2kbd_auc[128]*	UCHAR	Reserved for the remote-control unit
3968	*kbd2mmi_auc[128]*	UCHAR	Reserved for the remote-control unit

Detailed description of the elements

Name	Meaning
mmi_r	See section 5.1
nc_r	See section 5.2
private_r	Reserved
sps_r	See section 5.3
nc2mmi_r	See section 5.4
plc2mmi_auc	The contents of data word 0 to 127 are copied one to one from the data block 2 (DB2) of the PLC into this data field. The contents can be used to transfer messages and other optional information from the PLC to the HMI.
tast_r	See section 5.5
mmi2plc_auc	The contents of this data field are copied one to one in data word 128 to 192 into the data block 2 (DB2) of the PLC. The contents can be used to transfer messages and other optional information from the HMI to the PLC.

5.1 Communication range of HMI->NCR (*mmi_r*)

This communication range is used for the transfer of commands to the communication program that carries out the data exchange between the PC and the controller as well as for the synchronization and for the data exchange with the NC computer.

Offset	Name	Type	Meaning
0	*mmi2t_order_us*	USHORT	USHORT command to the communication processor
2	*t2mmi_quitt_us*	USHORT	Acknowledgment by the communication processor
4	*t2mmi_status_us*	USHORT	State of the communication processor
6	*mmi2t_quitt_us*	USHORT	Acknowledgment to the communication processor
8	*mmi2t_count_us*	USHORT	Acknowledgment counter to the communication processor
10	*t2mmi_count_us*	USHORT	Message counter from the communication processor
12	*mmi2t_dat_aus[5]*	USHORT	Optional data to the communication processor, depending on the command
22	*t2mmi_err_mc_uc*	UCHAR	Message counter for error messages of the communication processor
23	*mmi2t_err_qc_uc*	UCHAR	Acknowledgment counter for error messages
24	*t2mmi_err_r*	RECORD	Error messages from the communication processor
32	*msg2nc_r*	RECORD	Message buffer to NC computer

Name	Meaning
mmi2t_order_us, t2mmi_quitt_us, t2mmi_status_us, mmi2t_quitt_us, mmi2t_dat_aus	These inputs are used for command transfer and synchronization in the download phase and must not be modified afterwards.
t2mmi_count_us, mmi2t_count_us	These two counters are no longer used with the current controller.
t2mmi_err_mc_uc, mmi2t_err_qc_uc	With t2mmi_err_mc_uc, the communication program signals a communication error. A valid error message is always available in t2mmi_err_r if the two counters are not equal. The PC is to acknowledge the error message after the reading of the error, by overwriting of mmi2t_err_qc_uc with the value of t2mmi_err_mc_uc. This task is assumed by MMIGTWAY.EXE.
t2mmi_err_r	See section 5.1.1
msg2nc_r	See section 5.1.2

5.1.1 Error message in the communication range (*t2mmi_err_r*)

Offset	Name	Type	Meaning
24	*status_uc*	UCHAR	Error number
25	*abs_prim_id_uc*	UCHAR	Primary ID of the subscriber that signaled the error
26	*abs_sec_id_uc*	UCHAR	Number of subscriber that signaled the error
27	*msg_sb0_uc*	UCHAR	Control block of message in which the error occurred
28	*msg_prim_id_uc*	UCHAR	Primary ID of the subscriber to which the message was directed
29	*msg_sec_id_uc*	UCHAR	Number of subscriber to which the message was directed
30	*msg_dat_us*	USHORT	First user data network of message in which the error occurred

Name	Meaning	Wert
status_uc	This function includes one of the following error codes: Timeout during transferring Timeout during receiving Timeout during waiting for acknowledgment Receiving of invalid message No application is loaded Defective rooting information Memory is not sufficient for download Application is already loaded (only in case of download) Error during RAM test Startup error in the firmware	 1 2 3 10 11 12 13 14 20 21
abs_prim_id_u	This function identifies the communication link of the subscriber that signaled the error. Message comes from the HMI communication program Message comes from the controller Message comes from the keyboard communication link (no longer implemented) Message comes from the IO communication link (no longer implemented) Message comes from the axis computer communication link (no longer implemented)	 1 2 3 4 5
abs_sec_id_uc	Number of the subscriber in the communication link that signaled the error. Number 0 is the first subscriber, 255 corresponds to the communication program.	
msg_sb0_uc, msg_prim_id_uc, msg_sec_id_uc, msg_dat_us	These variables describe the message during which the error occurred. It is intended for information only.	

5.1.2 Message buffer HMI->NCR (*msg2nc_r*)

The message buffer is used for the transfer of non-cyclic data to the NC computer. The buffer is available only after the download and is mainly required for the message transfer and the block transfer to the NC.

Offset	Name	Type	Meaning
32	*qc_uc*	USHORT	Message acknowledgment counter
34	*mc1_us*	USHORT	Message start message counter
36	*zp_r*	RECORD	Message line buffer HMI->NCR
556	*mc2_us*	USHORT	Message end message counter
556	*free_us*	USHORT	Reserve

Name	Meaning
mc1_us, mc2_us, qc_us	These variables are used for the synchronization of the data exchange via the message line buffer. The input of a new message into the line buffer is only possible if the buffer is empty, i.e. if the last message has been acknowledged. This is always the case if *msg2nc_r.mc1_us* and *msg2mmi_r.qc_us* are equal. In general, meet the following sequence during the input of a message into the line buffer: 1. Wait until *msg2nc_r.mc1_us* and *msg2mmi_r.qc_us* are equal, probably with timeout monitoring. 2. Increment *msg2nc_r.mc1_us* by 1, in order to make the line buffer invalid. 3. Input a message into the line buffer. 4. Overwrite msg2nc_r.mc2_us with the value of *msg2nc_r.mc1_us*, in order to make the line buffer valid.
zp_r	See section 5.1.2.1

5.1.2.1 Message line buffer HMI->NCR (*zp_r*)

The message line buffer includes the message to the NC computer. Each message is composed of a header that classifies the following user data, and of the user data load.

Offset	Name	Type	Meaning
36	*sb0_uc*	UCHAR	Messages main group
37	*sb2_uc*	UCHAR	Messages modifier
38	*sb1_uc*	UCHAR	Messages sub-group
39	*index_uc*	UCHAR	Index for assignment response to request
40	*modul_uc*	UCHAR	Transmitter task recognition
41	*handle_uc*	UCHAR	Handle for assignment of requesting application
42	*len_us*	USHORT	Number of the following data bytes
44	*dby_ac[512]*	CHAR	Data of the message max. 512 bytes

Name	Meaning
sb0_uc, sb1_uc	The variables define main group and sub-group of message. Display messages, commands, block transfers and debug messages are differentiated. Evaluation of the sub-group is made as a function of the main group.
sb2_uc	This function is used as additional switch for the message type, if necessary
handle_uc, Index_uc	During each request, the handle and the index are passed by the controller up to the response. In this way, the requiring unit can clearly assign the response.
modul_uc, len_us	These variables include transmitter of message and number of included user data bytes. The HMI always includes 15 for the transmitter. The length of the user data can be between 0 and 512 bytes.
len_us	This variable includes the number of the user data bytes. The length can be between 0 and 512 bytes.
dby_ac	This variable includes max. 512 bytes of user data that are evaluated in the NC as a function of main group and sub-group.

5.2 Communication range of NCR->HMI (nc_r)

This communication range is used for the synchronization between PC and NCR in the download phase as well as for data exchange with the NC computer.

Offset	Name	Type	Meaning
560	*nc2t_order_us*	USHORT	Command to the communication processor
562	*t2nc_quitt_us*	USHORT	Acknowledgment by the communication processor
564	*t2nc_status_us*	USHORT	State of the communication processor
566	*nc2t_quitt_us*	USHORT	Acknowledgment to the communication processor
568	*nc2t_count_us*	USHORT	Counter for cyclic messages to the communication processor
570	*t2nc_count_us*	USHORT	Acknowledgment counter to the communic. processor
572	*nc2t_dat_aus[5]*	USHORT	Optional data to the communication processor, depending on the command
582	*t2nc_err_mc_uc*	UCHAR	Message counter for error messages of the communication processor
583	*nc2t_err_qc_uc*	UCHAR	Acknowledgment counter for error messages
584	*t2nc_err_r*	RECORD	Error messages from the communication processor
592	*msg2mmi_r*	RECORD	Message buffer to HMI

Name	Meaning
nc2t_order_us, t2nc_quitt_us, t2nc_status_us, nc2t_quitt_us, nc2t_dat_aus	These variables are a copy of the communication range between the NC and its communication processor and are used for synchronization in the download phase.
nc2t_count_us, t2nc_count_us, nc2t_dat_aus, t2nc_err_mc_uc, nc2t_err_qc_uc, t2nc_err_r	These variables are a copy of the communication range between the NC and its communication processor and are irrelevant for the PC.
msg2mmi_r	See section 5.2.1.

5.2.1 Message puffer NCR->HMI (msg2mm_r)

The message buffer is used for the transfer of non-cyclic data from the NC computer to the PC. It is available only after the download and is required mainly for order acknowledgment and block transfer.

Offset	Name	Type	Meaning
592	*Qc_us*	USHORT	Message acknowledgment counter
594	*mc1_us*	USHORT	Message start message counter
596	*zp_r*	RECORD	Message line buffer NCR->HMI
1116	*mc2_us*	USHORT	Message end message counter
1118	*free_us*	USHORT	Reserve

Name	Bedeutung
mc1_us, *mc2_us*, *qc_us*	These variables are used for synchronization of the data exchange via the message line puffer. A new message is always available if msg2mmi_r.mc1_us is not equal to msg2nc_r.qc_us and msg2mmi_r.mc1_us is equal to msg2mmi_r.mc2_us. Upon evaluation of the new message, it is to be acknowledged by the HMI by overwriting of msg2nc_r.qc_us with the value of msg2mmi_r.mc1_us.
zp_r	Message line buffer NCR->HMI, see section 5.2.1.1.

5.2.1.1 Message line buffer NCR->HMI (*zp_r*)

Offset	Name	Type	Meaning
596	*sb0_uc*	UCHAR	Message main group
597	*sb2_uc*	UCHAR	Message modifier
598	*sb1_uc*	UCHAR	Message sub-group
599	*index_uc*	UCHAR	Index for assignment response to request
600	*modul_uc*	UCHAR	Transmitter task recognition
601	*handle_uc*	UCHAR	Handle for assignment of requesting application
602	*len_us*	USHORT	Number of the following data bytes
604	*dby_ac[512]*	CHAR	Data of the messages max. 512 bytes

Name	Meaning
sb0_uc, sb1_uc	These variables define main group and sub-group of the message. Display messages, commands, block transfers and debug messages are differentiated. Evaluation of the sub-group is made as a function of the main group.
sb2_uc	This variable is used as additional switch for the message type, if necessary
handle_uc, Index_uc	The handle and the index receive a copy of the respective request message by the HMI. In this way, the requiring unit can clearly assign the response.
modul_uc, len_us	These variables include transmitter of message and number of included user data bytes.. The length of the user data can be between 0 and 512 bytes.
dby_ac	This variable includes max. 512 bytes of user data that are to be evaluated in the PC as a function of main group and sub-group.

5.3 Input and output signals of the PLC (*sps_r*)

This area includes a copy of the input/output copy of the PLC and is provided for the display of the respective signals on the HMI.

Offset	Name	Type	Meaning
1520	*tasten_aus[2]*	USHORT	Reserved for key signals to the PLC
1524	*dout_aus[32]*	USHORT	Digital outputs of the PLC
1588	*aout_aus[16]*	USHORT	Analog outputs of the PLC
1620	*din_aus[32]*	USHORT	Digital inputs of the PLC
1684	*ain_aus[16]*	USHORT	Analog inputs of the PLC
1716	*reserve_aus[30]*	USHORT	-
1776	*canin_auc[128]*	UCHAR	CAN bus inputs of the PLC (only the first 128 bytes)
1904	*canout_auc[128]*	UCHAR	CAN bus outputs of the PLC (only the first 128 bytes)

Name	Meaning
tasten_aus	This variable is reserved for 32 key signals to the PLC, currently no function.
dout_aus, *aout_aus*	Digital and analog outputs of the PLC. The outputs are assigned to the local IO components in the CNC module. Assignment depends on the used IO components.
din_aus, *ain_aus*	Digital and analog inputs of the PLC. The inputs are assigned to the local IO components in the CNC module. Assignment depends on the used IO components.
canin_auc, *canout_auc*	CAN bus inputs and outputs of the PLC. The assignment depends on the controller configuration and the used CAN IO components. Of the process range, always the first 128 bytes of the CAN bus I/Os are displayed only, i.e. %IW64...%IW127, or %QW64...%QW127.

5.4 Cyclic data from NC to HMI (*nc2mmi_r*)

In this area, all information of the NC computer is transferred where time assignment to the actual sequence in the NC computer is not absolutely necessary. In general, these are all display data that are renewed cyclically on the screen and that do not have a direct influence on the further sequence. The transfer of these data is time-controlled, not event-controlled (e.g. block change)! The data amount is limited to such information that a needed continuously or frequently in the HMI. All other information is to be called upon request via messages or via the cyclic parameter field display.

Offset	Name	Type	Meaning
2032	*start_uc*	UCHAR	Start counter
2033	*pad_auc[3]*	UCHAR	Alignment
2036	*sollpos_ad[12]*	DOUBLE	NC target positions of axes
2132	*offset_ad[12]*	DOUBLE	Zero point offset of axes
2228	*istpos_ad[12]*	DOUBLE	NC actual position of axes
2324	*schlepp_as[12]*	SHORT	Position lag of axes
2348	*spindeldrehzahl_al[6]*	LONG	Spindle speed
2372	*messen_al[6]*	LONG	Reserved for data acquisition
2396	*eventcount_ul*	ULONG	Event counter, operated by AXE
2400	*eventtime_ul*	ULONG	Time between two events
2404	*zust_kanal_un_r*	RECORD	Channel-independent states
2436	*data_kanal_un_r*	RECORD	Channel-independent data
2628	*kanalanzeige_ar[2]*	RECORD	Channel-dependent displays
2756	*modpos_ad[12]*	DOUBLE	Modal offset of axes
2852	*sps_anzeige_r*	RECORD	Copy of signals from PLC to NC
2916	*diagnose_r*	RECORD	Diagnosis display
2980	*pfeldanz_r*	RECORD	Freely selectable "permanent" parameter field displays
3300	*reserve_auc[11]*	UCHAR	-
3311	*ende_uc*	UCHAR	Final counter

Name	Meaning
start_uc, ende_uc	These variables are check flags that are incremented by the controller prior to each display update (start_uc) and after each display update (ende_uc). By means of a comparison of the two counters, the HMI is able to check whether the display data was valid at the moment of reading. In general, however, this is not necessary since they are mere display data.
sollpos_ad	This variable includes the interpolation target position of all axes in input units. In case of modulo 360° rotation axes, the position is converted to 0°–360°. Via tast_r.gen_abspos_us in the virtual keyboard, one of the following axes modes can

	be selected for each axis: If bit N is set, the position of axis N is determined as the sum of the NC target position (P32-P43) and the modal target position (P144-P155), otherwise only as NC target position.
offset_ad	This variable includes the overall offset of the current coordinate system for all axes in input units. This is a copy of P192-P203.
istpos_ad	This variable includes the interpolation actual position of all axes in input units. In case of modulo 360° rotation axes, the position is converted to 0°–360°. Via tast_r.gen_abspos_us in the virtual keyboard, one of the following axes modes can be selected for each axis: If bit N is set, the position of axis N is determined as the sum of the NC actual position (P0-P11) and the modal actual position (P160-P171), otherwise only as NC target position. Note: The displayed position must not be mixed up with the actual position, but results internally from the interpolation of the axes. The target position is determined of this position after all necessary transformations. The actual position results from the position control.
schlepp_as	This variable includes the current position lag of all axes in increments (P224-P235).
spindeldrehzahl_al	This variable includes the current command speed of all NC spindles in 1/min.
eventcount_ul, eventtime_ul	The event counter is incremented with each positive edge at the reserve input of the first analog axis. The pulse length and the pause of the input pulse must have a length of at least one fine interpolation cycle, in order to be safely counted. In addition, the time between the last two event pulses is acquired accurately to one fine interpolation cycle and is stored in eventtime_ul.
zust_kanal_un_r	See section 5.4.1.
data_kanal_un_r	See section 5.4.2.
kanalanzeige_ar	See section 5.4.3.
modpos_ad	This variable includes the modal actual position (P160-P171) of all axes in input units. In case of modulo 360° rotation axes, the position is converted to 0°–360°. The modal actual position can be interpolated parallelly to the NC program in progress by PLC or HMI. The position referred to the current coordinate system results by the summing of the NC actual position (P0-P11) and the modal actual position (see istpos_ad).
sps_anzeige_r	See section 5.4.4
diagnose_r	See section 5.4.5
pfeldanz_r	See section 5.4.6

5.4.1 Channel-independent states (zust_kanal_un_r)

Offset	Name	Type	Meaning
2404	*unterbrechen_aktiv_uc*	UCHAR	Program in progress is interrupted
2405	*progaktiv_uc*	UCHAR	Program is processed
2406	*satzvorlauf_uc*	UCHAR	Block search is active
2407	*masssystem_uc*	UCHAR	Reserved for inch/metric conversion
2408	*satzausblenden_uc*	UCHAR	Block skip is active
2409	*einzelsatz_uc*	UCHAR	Single block (not following block) is active
2410	*wahlweisehalt_uc*	UCHAR	Optionally stop at M1
2411	*programmhalt_uc*	UCHAR	Programmed stop (M0/M1) is active
2412	*referenzpunkt_us*	USHORT	Reference points are approached
2414	*pos_endschalter_us*	USHORT	States of pos. range of travel limit switch
2416	*neg_endschalter_us*	USHORT	States of neg. range of travel limit switch
2418	*referenz_nocken_us*	USHORT	States of reserve switches
2420	*reserve_eingang_us*	USHORT	States of reserve inputs
2422	*slave_pos_endschalter_us*	USHORT	Pos. limit switch states of slave axes
2424	*slave_neg_endschalter_us*	USHORT	Neg. limit switch states of slave axes
2426	*slave_referenz_nocken_us*	USHORT	Reference switch states of slave axes
2428	*slave_reserve_eingang_us*	USHORT	Reserve input states of slave axes
2430	*spindel_aktiv_auc[6]*	UCHAR	Assignment of axes to spindles

Name	Meaning
unterbrechen_aktiv_uc	This variable indicates whether the program in progress was interrupted due to the actuating of the interruption key (*tast_r.masch_r.unterbrechen_uc*) of the virtual keyboard. This state can be undone by renewed actuating of the interruption key and the interrupted program can be continued by actuating the start key (*tast_r.start_auc[0]*) or interrupted by actuating the stop key (*tast_r.masch_r.stop_uc*). In interruption state, programs or single functions can be carried out without having an influence on the interrupted program.
progaktiv_uc	This variable indicates whether a program or a single function is currently processed or whether axes movements are carried out. Together with *unterbrechen_aktiv_uc*, the following state displays are possible:
satzvorlauf_uc	This vartiable signals that the current program is read in block search. In block search, the program is read up to a defined block search limit,

	without implementing the read blocks and is then processed normally.
satzausblenden_uc	This variable indicates that the function Block skip is active. All DIN blocks are ignored that have a slash (/) put in front. This function is actuated via the virtual keyboard (*tast_r.mash_r.satzausblenden_uc*).
einzelsatz_uc	This variable indicates that the current program is process in single block mode. The start key is to be actuated after each block in order to be able to implement the next block. The single block mode is actuated via the virtual keyboard (*tast_r.mash_r.einzel_folgesatz_uc*) and remains active after the resetting of the signal until the start key is actuated again (*tast_r.start_auc[0]*).
wahlweisehalt_uc	This variable signals that the function "Optional stop " is active. In this state, the program is stopped at M1. This function is to be realized via the PLC by disabling of the read enabling at M1 and re-enabling of the read enable in case of a positive edge of the start key.
programmhalt_uc	This variable signals a programmed stop after M0 or M1. This signal is generated by the PLC, it is a copy of the data block 1 of the PLC. The function "Programmed stop" is to be realized via the PLC by disabling of the read enabling at M0 or M1 and re-enabling of the read enable in case of a positive edge of the start key.
referenzpunkt_us	This variable selects the axes, the reference point of which have already been approached, i.e. bit-coded for all axes. Bit0 corresponds to axis 0, bit11 corresponds to axis 11.
pos_endschalter_us, neg_endschalter_us, referenz_nocken_us, reserve_eingang_us	These variables show the states of the digital inputs of all axes in a bit-coded way. Bit0-11 corresponds to the axes 0-11. State '1' corresponds to the active state, e.g. limit switch active, this corresponds to a signal state of 0 or a cable break.
slave_pos_endschalter_us, slave_neg_endschalter_us, slave_referenz_nocken_us, slave_reserve_eingang_us	These variables show the states of the digital inputs of the slave axes of synchronous axes in a bit-coded way. Bit0-11 corresponds to the axes 0-11. Definition as above.
spindel_aktiv_auc	This variable includes the assigned axis number or 255 for each configured spindle, if the spindle is not active. Spindles are to be activated by means of G96 or G97.

The meaning of the states of the combination of the two variables *progaktiv_uc* and *unterbrechen_aktiv_uc* is as follows.

unter-brechen-aktiv_uc	***prog-aktiv_uc***	**Meaning**
0	0	No task is processed
0	1	Task is processed without axis movement
0	2	No task is processed, but min. one axis is moved modally
0	3	Task is processed with axis movement
1	0	Program is interrupted, no task in interruption state active
1	1	Task is processed without axis movement in interruption state
1	2	No task is processed in interruption state, but min. one axis is moved modally
1	3	Task is processed with axis movement in interruption state

5.4.2 Channel-independent data (*data_kanal_un_r*)

Offset	**Name**	**Type**	**Meaning**
2436	*grunddrehwinkel_d*	DOUBLE	Rotary angle of C-axis
2444	*einh_schlepp_ad[12]*	DOUBLE	Position lag in the input unit
2540	*zeitzaehler_aul[4]*	ULONG	Reserved for time acquisition
2556	*achsen_override_s*	SHORT	Path and axis override (override 0)
2558	*spindel_override_s*	SHORT	Override for spindles (override 1)
2560	*drehwinkel_ad[3]*	DOUBLE	Overall rotary angle in ABC
2584	*achskenn_ac[16]*	CHAR	Identifying letter of the configured axes
2600	*reserve_auc[28]*	UCHAR	-

Name	Meaning
grunddrehwinkel_d	This variable includes the rotary angle of the C-axis in degrees and is a copy of *drehwinkel_ad[2]*.
einh_schlepp_ad	This variable includes the current position lag of all axes in input units. The position lag is the difference between target position and actual position.
achsen_override_s	Current override value for simultaneous path mode, non-simultaneous path mode and manual mode in per thousand. The source of this value is selected by the PLC and can be either override 0 of the virtual keyboard (tast_r.mash_r.override_as[0]) or override 0 of the DB1 of the PLC.
spindel_override_s	Current override value of spindles in per thousand. The source of this value is selected by the PLC and can be either override 1 of the virtual keyboard (*tast_r.mash_r.override_as[1]*) or override 1 of the DB1 of the PLC. Note: The effect of the overrides on the spindles depends on the selected spindle type, see G96/G97 in the programming instructions.
drehwinkel_ad	This variable includes the sum of basic rotary angle and contour rotary angle for the axes A, B and C in degrees, see G88/G89 in the programming instructions.
achskenn_ac	This variable includes the ASCII code of the letters of all configured axes or 0, if the axis is not configured or if no letter is assigned to it.

5.4.3 Channel-depended displays (kanalanzeige_ar)

First channel:

Offset	Name	Type	Meaning
2628	*kanalnr_uc*	UCHAR	Number of channel for the following data
2629	*prog_aktiv_uc*	UCHAR	Program or single function in operation
2630	*w_zeug_uc*	UCHAR	Active tool coordinate system
2631	*w_stueck_uc*	UCHAR	Active tool coordinate system
2632	*achse_ul*	ULONG	Axis assignment to this channel
2636	*vmax_d*	DOUBLE	Current path speed
2644	*prognr_us*	USHORT	Actual program number
2646	*satznr_us*	USHORT	Last programmed block number
2648	*logsatznr_us*	USHORT	Actual logic block number
2650	*m_fkt_us*	USHORT	Last M-function
2652	*verweilzeit_d*	DOUBLE	Current retention time
2660	*kont_drehwinkel_us*	USHORT	Reserved for angle of contour turning
2662	*level_uc*	UCHAR	Reserved for program nesting depth
2663	*wdh_zaehler_uc*	UCHAR	Reserved for repetition counter
2664	*ebene_uc*	UCHAR	Reserved for normal level/interrupt level
2665	*bahn_strecke_uc*	UCHAR	Simult. path / non-simult. path
2666	*kart_polar_uc*	UCHAR	Reserved for polar/cartesian changeover
2667	*abs_kett_uc*	UCHAR	Reserved for absolute/incremental dimension
2668	*interpol_art_uc*	UCHAR	Reserved for interpolation type
2669	*zustand_uc*	UCHAR	Internal channel state
2670	*wdh_zaehler_soll_ uc*	UCHAR	Reserved for repetition counter
2671	*rotvbahn_uc*	UCHAR	Display of path speed in 1/min
2672	*vorlauf_anzeige_us*	USHORT	Number of blocks in the preprocessing memory
2674	*h_fkt_us*	USHORT	Last H-function
2676	*zeilennr_ul*	ULONG	Current line number in the DIN program
2680	*reserve_auc[12]*	UCHAR	-

Second channel:

Offset	Name	Type	Meaning
2692	*kanalnr_uc*	UCHAR	Number of the channel for the following data
2693	*prog_aktiv_uc*	UCHAR	Program or single function in operation
2694	*w_zeug_uc*	UCHAR	Active tool coordinate system
2695	*w_stueck_uc*	UCHAR	Active tool coordinate system
2696	*achse_ul*	ULONG	Axis assignment to this channel
2700	*vmax_d*	DOUBLE	Current path speed
2708	*prognr_us*	USHORT	Actual program number
2710	*satznr_us*	USHORT	Last programmed block number
2712	*logsatznr_us*	USHORT	Actual logic block number
2714	*m_fkt_us*	USHORT	Last M-function
2716	*verweilzeit_d*	DOUBLE	Current retention time
2724	*kont_drehwinkel_us*	USHORT	Reserved for angle of contour turning
2726	*level_uc*	UCHAR	Reserved for program nesting depth
2727	*wdh_zaehler_uc*	UCHAR	Reserved for repetition counter
2728	*ebene_uc*	UCHAR	Reserved for normal level/interrupt level
2729	*bahn_strecke_uc*	UCHAR	Simult. path / non-simult. path
2730	*kart_polar_uc*	UCHAR	Reserved for polar/cartesian changeover
2731	*abs_kett_uc*	UCHAR	Reserved for absolute/incremental dimension
2732	*interpol_art_uc*	UCHAR	Reserved for interpolation type
2733	*zustand_uc*	UCHAR	Internal channel state
2734	*wdh_zaehler_soll_ uc*	UCHAR	Reserved for repetition counter
2735	*rotvbahn_uc*	UCHAR	Display of path speed in 1/min
2736	*vorlauf_anzeige_us*	USHORT	Number of blocks in the preprocessing memory
2738	*h_fkt_us*	USHORT	Last H-function
2740	*zeilennr_ul*	ULONG	Current line number in the DIN program
2744	*reserve_auc[12]*	UCHAR	-

Name	Bedeutung
kanalnr_uc	This variable includes the number of the NC-channel, the data of which are displayed in the channel display. In normal state, the channel number of the first display channel is always 0. The channel number for the second display channel can be selected via the parameter field (P576). In interruption state, the interruption channel is displayed in the first display channel and the NC channel 0 is displayed in the second display channel.
prog_aktiv_uc	This variable is 1, if a task is processed in the current channel or if an error occurs, and it is 0 if the channel is inactive.
w_zeug_uc	This variable includes the number of the current tool coordinate system of this channel. The number corresponds the value of the last programmed T in this channel.
w_stueck_uc	This variable includes the number of the current tool coordinate system of this channel. The number corresponds to the value of the last programmed S in this channel.
achse_ul	This variable indicates which axes are moved by this channel at the current point in time. The display is bit-coded, Bit0-11 corresponds to axis 0-11.
vmax_d	This variable includes the current path speed in input units per minute or in revolutions per minute as a function of the value in rotvbahn_uc.
prognr_us, satznr_us, logsatznr_us, zeilennr_ul	These variables include program number, block number, logic block number and line number of the block last started in this channel. The program number corresponds to the number of the last opened program, the block number corresponds to that of the last programmed N and the logic block number corresponds to the number of blocks since the last programmed block number (N). A block in which a block number is programmed, always gets the logic block number 1. The line number includes the current program line, calculated from 1.
m_fkt_us, h_fkt_us	These variables include the numbers of the last M and H function output to the PLC in this channel. M-functions are output only at the end of the block, the M-function number can, therefore, be shown much later in the display than the respective block number, depending on the block structure.
verweilzeit_d	This variable includes the retention time to be expected in seconds, until the next block is processed. Retention times can be programmed by means of G4 and G60, see programming instructions.
bahn_strecke_uc	This variable indicates whether the current interpolation is made in simultaneous path mode or in non-simultaneous path mode. In simultaneous path mode 1 is displayed, in non-simultaneous path mode 0 is displayed. In simultaneous path mode, all programmed axes arrive at their target at the same time, in non-simultaneous path mode, all axes move independently from each other with the programmed
zustand_uc	This variable corresponds to the internal channel state of the displayed channel. This state can be displayed for diagnosis purposes.

	0 = POS_KANAL_ZUST_IDLE_E 1 = POS_KANAL_ZUST_RUN_E 2 = POS_KANAL_ZUST_BREMSEN_ABBRUCH_E 3 = POS_KANAL_ZUST_ABBRUCH_WAIT_QUITT_E 4 = POS_KANAL_ZUST_ABBRUCH_E 5 = POS_KANAL_ZUST_BREMSEN_UB_E 6 = POS_KANAL_ZUST_UB_E 7 = POS_KANAL_ZUST_BREMSEN_SATZJMP_E 8 = POS_KANAL_ZUST_SATZJMP_E 9 = POS_KANAL_ZUST_BREMSEN_PROGCALL_E 10 = POS_KANAL_ZUST_PROGCALL_E 11 = POS_KANAL_ZUST_BREMSEN_FEHLER_E 12 = POS_KANAL_ZUST_FEHLER_E 13 = POS_KANAL_ZUST_AUSGLEICH_E
rotvbahn_uc	This variable indicates whether the current path velocity in vmax_d is to be interpreted as rotary (1) or translational (0) velocity. The changeover to rotary velocity is made implicitly if a rotation axis is used for the path as master axis and not as slave axis.
vorlauf_anzeige_us	This variable includes the number of interpreted blocks in the preprocessing memory. The interpreted program blocks are passed to the interpolator for implementation via a ring buffer. This display includes the number of blocks interpreted but not yet implemented.

5.4.4 Copy of signals from the PLC to the NC (*sps_anzeige_r*)

This display structure includes a copy of the first 32 data words from data block 1 of the PLC. The enabling displayed here and other signals are used for diagnosis of machine and PLC.

Offset	*Name*	Type	Meaning
2852	*notaus_bit*	BIT0	Emergency stop signal to the NC
2852	*vorschubfreigabe_bit*	BIT1	State of the overall feed enabling
2852	*soforthalt_bit*	BIT2	State of the immediate stop signal
2852	*einzelfkt_sperre_bit*	BIT3	Disabling of single functions from the HMI
2854	*vorschubfreigabe_us*	USHORT	State of axis feed enabling
2856	*position_halt_us*	USHORT	State of the position stop signals
2858	*verfahrtastenfg_plus_us*	USHORT	State of the pos. traversing key enabling
2860	*verfahrtastenfg_minus_us*	USHORT	State of the neg. traversingf key enabling
2862	*reglerfreigabe_us*	USHORT	State of controller enabling
2864	*slavefreigabe_us*	USHORT	Controller enabling treatment of slave axes
2866	*einlesefreigabe_uc*	UCHAR	State of read enabling
2867	*reserve2_auc[3]*	UCHAR	-
2870	*programmstart_uc*	UCHAR	Value of the start key
2871	*use_sps_override_uc*	UCHAR	Enabling of PLC override
2872	*spindel_ein_aus_us*	USHORT	Enabling for spindles
2874	*spindel_richtung_us*	USHORT	Reversal of direction of rotation of spindles
2876	*programmstop_uc*	UCHAR	Value of the stop key
2877	*unterbrechen_uc*	UCHAR	Value of the interruption key
2878	*einzel_folgesatz_uc*	UCHAR	Single (1) or subsequent block mode (0)
2879	*satzausblenden_uc*	UCHAR	Block skip active
2880	*rueckzug_uc*	UCHAR	Retract on the path is active
2881	*reserve3_auc[3]*	UCHAR	-
2884	*tastensignale_aus[2]*	USHORT	Copy of the key signals from PLC to HMI
2888	*qbit_signale_aus[4]*	USHORT	Copy of the Q-bit-signals from PLC to NC
2896	*mfkt_leds_aus[8]*	USHORT	Reserved for 128 LEDs at the HMI
2904	*reserve4_aus[5]*	USHORT	-
2914	*freigaben_mmi_us*	USHORT	Enabling for display update on BDT20

The meaning of the single signals is described in detail in the interface description between PLC and NC.

5.4.5 Diagnosis displays (*diagnose_r*)

Offset	Name	Type	Meaning
2916	*git_startzeit_us*	USHORT	Latency at the beginning of the coarse interpolator cycle
2918	*git_restzeit_us*	USHORT	Remaining time to the next coarse interpolator cycle
2920	*plc_timer_ul*	ULONG	Reserved for PLC time measurement
2924	*git_count_ul*	ULONG	Number of coarse interpolator cycles since energizing
2928	*zst_zustand_uc*	UCHAR	Global state of the central controller
2929	*pos_zustand_uc*	UCHAR	Global state of the coarse interpolator
2930	*trace_entries_us*	USHORT	Number of inputs in the trace buffer
2932	*fit_zyklustime_us*	USHORT	Current cycle time of the fine interpolator
2934	*fit_zyklusmin_us*	USHORT	Minimum cycle time of the fine interpolator
2936	*fit_zyklusmax_us*	USHORT	USHORT Maximum cycle time of the fine interpolator
2938	*fit_totaltime_us*	USHORT	Current operating time of the fine interpolator
2940	*fit_totalmin_us*	USHORT	Minimum operating time of the fine interpolator
2942	*fit_totalmax_us*	USHORT	Maximum operating time of the fine interpolator
2944	*git_zyklustime_us*	USHORT	Current cycle time of the coarse interpolator
2946	*git_zyklusmin_us*	USHORT	Minimum cycle time of the coarse interpolator
2948	*git_zyklusmax_us*	USHORT	Maximum cycle time of the coarse interpolator
2950	*git_totaltime_us*	USHORT	Current operating time of the coarse interpolator
2952	*git_totalmin_us*	USHORT	Minimum operating time of the coarse interpolator
2954	*git_totalmax_us*	USHORT	Maximum operating time of the coarse interpolator
2956	*plc_zyklustime_us*	USHORT	Current cycle time of the PLC
2958	*plc_zyklusmin_us*	USHORT	Minimum cycle time of the PLC
2960	*plc_zyklusmax_us*	USHORT	Maximum cycle time of the PLC
2962	*plc_totaltime_us*	USHORT	Current operating time of the PLC
2964	*plc_totalmin_us*	USHORT	Minimum operating time of the PLC
2966	*plc_totalmax_us*	USHORT	Maximum operating time of the PLC
2968	*reserve_auc[12]*	UCHAR	-

Name	Bedeutung
git_startzeit_us, git_restzeit_us	These variables indicate the time between the regular and the current start of the coarse interpolator as well as the remaining time between the end of the current coarse interpolator cycle and the start of the next coarse interpolator cycle. The unit of time is 1.19 microseconds. These times are of the CNC200 and are not very significant for the CNC55.
git_count_ul	This counter is incremented by 1 with each coarse interpolator cycle and is used as time base for dwell times and time measurements.
zst_zustand_uc	Interconnected state of the central controller. This state should be displayed for diagnosis purposes. 0 = Z_ZST_NOT_INIT_E 1 = Z_ZST_INIT_E 2 = Z_ZST_WAIT_Z_POS_KALTSTART_E 3 = Z_ZST_WAIT_Z_POS_WARMSTART_E 4 = Z_ZST_MK_UPDATE_E 5 = Z_ZST_WAIT_Z_POS_IDLE_E 6 = Z_ZST_IDLE_E 7 = Z_ZST_IDLE_UB_SOLL_AUS_E 8 = Z_ZST_WAIT_Z_POS_RUN_E 9 = Z_ZST_RUN_E 10 = Z_ZST_RUN_UB_SOLL_EIN_E 11 = Z_ZST_WAIT_Z_POS_FEHLER_HALT_E 12 = Z_ZST_FEHLER_HALT_E 13 = Z_ZST_WAIT_Z_POS_FEHLER_FREI_E 14 = Z_ZST_FEHLER_FREI_E 15 = Z_ZST_WAIT_FEHLER_FREI_UB_SOLL_EIN_E 16 = Z_ZST_WAIT_Z_POS_RESTART_K0_E 17 = Z_ZST_WAIT_Z_POS_IDLE_OR_RUN_E 18 = Z_ZST_WAIT_Z_POS_FEHLER_ABBRUCH_E 19 = Z_ZST_FEHLER_ABBRUCH_E 20 = Z_ZST_FEHLER_FATALE_E
pos_zustand_uc	Interconnected state of the coarse interpolator. This state should be displayed for diagnosis purposes. 0 = Z_POS_NOT_INIT_E 1 = Z_POS_INIT_E 2 = Z_POS_KALTSTART_E 3 = Z_POS_WARMSTART_E 4 = Z_POS_IDLE_E 5 = Z_POS_RUN_E 6 = Z_POS_IDLE_EINGELEITET_E 7 = Z_POS_RUN_UB_IDLE_EINGELEITET_E 8 = Z_POS_RESTART_K0_EINGELEITET_E 9 = Z_POS_FEHLER_HALT_EINGELEITET_E 10 = Z_POS_FEHLER_HALT_E 11 = Z_POS_FEHLER_FREI_E 12 = Z_POS_FEHLER_ABBRUCH_EINGELEITET_E 13 = Z_POS_FEHLER_ABBRUCH_E

	14 = Z_POS_FEHLER_INTERN_E 15 = Z_POS_FEHLER_FATALE_E
trace_entries_us	This variable includes the number of trace entries accumulated in the trace buffer of the controller. Via the trace, changes of state, DIN blocks and even axis positions can be recorded in the controller and can be transferred asynchronously via the message buffer.
fit_zyklustime_us, *fit_zyklusmin_us,* *fit_zyklusmax_us*	These variables include the currently determined cycle time of the fine interpolator in microseconds as well as the minimum and maximum cycle time in the last two seconds.
fit_totaltime_us, *fit_totalmin_us,* *fit_totalmax_us*	These variables include the mere currently measured operating time of the fine interpolator in microseconds as well as the minimum and maximum operating time in the last two seconds.
git_zyklustime_us, *git_zyklusmin_us,* *git_zyklusmax_us*	These variables include the currently determined cycle time of the coarse interpolator in microseconds as well as the minimum and maximum cycle time in the last two seconds.
git_totaltime_us, *git_totalmin_us,* *git_totalmax_us*	These variables include the mere currently measured operating time of the coarse interpolator in microseconds as well as the minimum and maximum operating time in the last two seconds.
plc_zyklustime_us, *plc_zyklusmin_us,* *plc_zyklusmax_us*	These variables include the currently determined cycle time of the PLC in microseconds as well as the minimum and maximum cycle time in the last two seconds.
plc_totaltime_us, *plc_totalmin_us,* *plc_totalmax_us*	These variables include the mere currently measured operating time of the PLC in microseconds as well as the minimum and maximum operating time in the last two seconds.

5.4.6 Freely selectable "permanent" parameter field display (*pfeldanz_r*)

Up to 32 freely selectable parameters can be updated cyclically in the parameter field display. The 32 parameter field indices are selected by means of the message.

Offset	Name	Type	Meaning
2980	*idx_aus[32]*	USHORT	P-field indices for the following 32 P-field values
3044	*val_ad[32]*	DOUBLE	P-field values for the respective indices

Note:

Like all other cyclic displays, the parameter field display is intended for visualization and is suited only to a limited extent for synchronization of HMI and NC.

The following example is to explain this:

P1555 is to be used for the synchronization between HMI and DIN program.

- Initial value in the parameter field display P1555=0
- By means of a message, the HMI writes in P1555=1 and waits until the DIN program in progress has recognized and reset the 1. Up to the next display update, the previous value of P1555 is still the parameter field display.
- The HMI reads P1555=0 and makes a wrong decision.

Such synchronization tasks should be made possibly by means of messages because this way is much less susceptible to errors since a synchronization of messages is imperative.

5.5 Virtual keyboard from HMI to NC (*tast_r*)

The virtual keyboard is used as simplified interface between operator and NC controller. The overwriting of a signal in the virtual keyboard is sufficient for most of the tasks such as referencing, manual mode, zero point setting, start of the NC program and others.

Offset	Name	Type	Meaning
3584	*mfkt_auc[16]*	UCHAR	M-function keys to PLC -> %MW1.212ff
3600	*masch_r*	RECORD	Machine keyboard
3632	*start_auc[8]*	UCHAR	Start key -> %MB1.137.0
3640	*verfahr_ac[16]*	CHAR	Traversing keys for max. 16 axes -> %MW1.204ff
3656	*override_as[4]*	SHORT	Override values -> %MW1.200ff
3664	*vmax_vorwahl_d*	DOUBLE	Speed presetting for traversing keys
3672	*schrittweite_vorwahl_d*	DOUBLE	Increments presetting for incremental traversing
3680	*betriebsart_uc*	UCHAR	Operating mode to PLC -> %MW1.128

Offset	Name	Type	Meaning
3681	*tdbreak_uc*	UCHAR	Reserved for debug purposes
3682	*gen_abspos_us*	USHORT	Combination of Modpos and Istpos
3684	*gen_abspos2_us*	USHORT	Display position including all transformations
3686	*reserve_auc[26]*	UCHAR	-

Name	Meaning
mfkt_auc	This variable includes 128 single signals to the PLC that are copied by the NC controller into data block 1 of the PLC (%MW1.212...219). They can be used optionally in the PLC.
masch_r	See section 5.5.4.
start_auc[0]	This variable includes the start key for the start of the automatic reference point approach or of a NC program. The other 7 start keys (start_auc[1..7]) are without function. The function can accept values between 0 and 255. A change of the start key from 0 to unequal 0 signals valid start conditions. The contents of the start key are copied into data block 1 of the PLC (%MB1.137.0). The start information is structured in two nibbles. The most significant nibble (bit 4-7) includes the information about the type of program that is to be started, the least significant bit (bit 0-3) includes the mode in which the start is to be implemented.
verfahr_ac	This function includes the traversing key signals of all axes. In this way, each axis can carry out modal traversing, incremental traversing, reference traversing etc., independently of the other axes. For each axis, one byte is available as traversing key. The contents of verfahr_ac are copied one to one into DB1 of the PLC (DB1_NC2SPS_VERFAHR_AB) starting with %MW1.204.
override_as	This function includes four velocity superposition values in per thousand. The respective overrides are only effective if the PLC did not disable them via the interface PLC ->NCR (DB1_SPS2NC_USE_SPSOVERRIDE_B=%MB1.9.1). In addition, the overrides are copied into DB1 of the PLC (%MW1.200ff). 0 = Override of simultaneous path mode and non-simultaneous path mode as well as traversing key in mode -104...+104. 1 = Override of some spindle types, see G97 in the programming instructions. 2 = Override of the function "modal oscillation", see G36 in the programming instructions. 3 = Override of traversing key mode +/-110 and +/-111.
vmax_vorwahl_d	This function includes the velocity presetting of all linear axes in input units/min. The presetting is written with each modification in parameter P672 in the parameter field. This preset velocity acts only on traversing key mode -100...+100 and is limited in an axis-specific way in the NC computer to the max. modal traversing velocity of MK_MODVMAX.

schrittweite_vorwahl_d	This function includes the increments presetting of all axes for jogging and incremental traversing (traversing mode +/-101 and +/-104) in input units. With each modification, its value is written in parameter P673 in the parameter field.
betriebsart_uc	The operating mode is written one to one into the respective byte of the DB1 of the PLC. The mode has no implicit influence on functions of the NC controller. The following operating modes are predefined: 0 = Basic state without axis movement 1 = Manual mode with setup functions and manual mode of axes 2 = Manual single functions 3 = Automatics, automatic processing of programs (single block, following block) 0x80 = Diagnosis Additional operating modes can be defined at request.
gen_abspos_us, gen_abspos2_us	These variables define in a bit-coded way for each axis how the display of command and actual positions is to be made.

5.5.1 Assignment of the start key

Bit 4-7	Meaning
0	Start of the NC program entered in parameter 512, 513
1	Start of the NC program entered in parameter 514, 515
2	Start of the NC program entered in parameter 516, 517
3	Start of the NC program entered in parameter 518, 519
4	Start of the NC program entered in parameter 520, 521
5	Start of the NC program entered in parameter 522, 523
6	Start of the NC program entered in parameter 524, 525
7	Start of the NC program entered in parameter 526, 527
15	Start of the automatic reference point approach

Bit 0-3	Meaning
0	No reaction
1	Normal start
2	Start in block search (P528-P532 configuration of the input point)
11	Restart of a program interrupted due to an error at a defined point (G10)
12	Quitting of the interruption state. Subsequently, the interrupted program can be restarted with a normal start. The function is identical with the actuating of the interruption key in interruption state.
13	Restart of a program interrupted due to an error in the interrupted point. The same function can also be triggered by means of a normal start in error state.
15	Start of the next block in single block mode. The same function can also be triggered by means of a normal start in single block mode.

Examples:

Start of reference point approach:

- Writing of 0xF1 in the start key

Normal start of a NC program:

- Writing of the program number of the NC program to be started in P512 – writing of the number of the block where the program is to be started in P513 or 0 for the first block.
- Writing of 0x01 in the start key

Start of the NC program of the P520, P521 in block search:

- Writing of 0x42 in the start key

5.5.2 Assignment of the traversing key *verfahr_ac*

Effects of the traversing key:

Value	Meaning
100 to +100	Traversing of the respective axis with % of the max. modal traversing velocity or the preselected traversing velocity if the latter is smaller than the maximum velocity. The axis moves as long as the value is available in the traversing key. The sign indicates the traversing direction.
+/-101	Jogging by the increment width of the respective axis with the traversing movement programmed in non-simultaneous path mode with missing feed enable. In simultaneous path mode the same function can be carried out on the path with the traversing key [0]. The direction depends on the sign, at +101, jogging is made in the direction of the path target while at -101 jogging is made in the direction of the path start.
+/-102	Reference point approach of the respective axis. The sign is not considered.
+/-103	Zeroing of the respective axis in the current coordinate system. The sign is not considered.
+/-104	Incremental traversing of the respective axis by the set increment. The sign indicates the direction.
+/-105	Activating of manual wheel as per sign of the traversing key with evaluation factor +1 or -1 for the respective axis.
+/-106	Activating of manual wheel with evaluation factor +3 or -3.
+/-107	Activating of manual wheel with evaluation factor +10 or -10.
+/-108	Activating of manual wheel with evaluation factor +30 or -30.
+/-109	Activating of manual wheel with evaluation factor +100 or -100.
+/-110	Target point approach of the respective axis in compliance with the inputs in the parameter field. The sign is not considered.
+/-111	Modal traversing of the respective axis with the traversing velocity entered in P208, as long as 111 is included in the traversing key. The sign indicates the direction.

Please observe that an action (traversing movement or zeroing) can only be triggered by a change of the traversing key, e.g. from 0 to 100.

The functions modal traversing and target point approach (-100 to +100, 110, 111) are also possible in the program in progress, if necessary, see G122 of the programming instructions. These traversing movements are always interpolated by means of the "modal actual position". The modal actual position is managed by the controller parallelly to the NC actual position, in this way, a superposition of modal traversing movements and path interpolation is possible. The modal actual position can be zeroed if necessary and offset with the NC actual position, see G121 in the programming instructions.

Target point approach

Target point approach of an axis is started with the value 110 in its traversing key. The axis moves as long as 110 is included in the traversing key and as long as the target point is not reached. Target position and acceleration ramps are accepted when entering the value with the edge as per 110 of the parameter field. Unlike the velocity, they cannot be changed during approach.

Parameter	Meaning
P176-P187	Target position of axis 0-11 in input units
P208-P219	Traversing velocity of axis 0-11 in input units /min or revolutions/min. The velocity is initiated during loading of the machine constants with the max. traversing velocity as per MK_VMAX
P240-P251	Acceleration and deceleration ramp of 0-11 in m/min or 1/min. The ramp is initiated during loading of the machine constants with the max. acceleration as per MK_BESCHL
P144-P155	Modal target position of axis 0-11 accepted at the start edge
P160-P171	Current modal actual position of axis 0-11

Two cases are distinguished when accepting the modal target position:

1. G122 X0 (Default configuration)

The modal target position results from the difference between the target position presetting in P176ff and the current NC actual position. The presetting refers to the current coordinate system point of origin since the axis target position corresponds to the sum of NC actual position and modal actual position.

2. G122 X1

The modal target position directly corresponds to the target position presetting in P176ff. The presetting refers to the current NC actual position. The target position referred to the current coordinate system point of origin depends, therefore, on the current NC actual position.

5.5.3 Assignment of the variables *gen_abspos_us, gen_abspos2_us*

Bit N in *gen_abs-pos_us*	**Bit N in *gen_abs-pos2_us***	**Meaning**
0	0	Target position display of axis N is NC target position (P32-P43). Actual position display is NC actual position (P0-P11)
0	1	Not defined
1	0	Target position display of axis N is the sum of NC target position (P32-P43) and modal target position (P144-P155). Actual position display is the sum of NC actual position (P0-P11) and modal actual position (P160-P171)
1	1	Actual position display of axis N is the absolute target position (P944-P959) including all transformations and coordinate system offsets. The difference of the actual position variants 1/1 and 1/0 are added to the target position of variant 1/0

5.5.4 Machine keyboard (masch_r)

Offset	Name	Type	Meaning
3600	*vorschubhalt_uc*	UCHAR	Feed stop at PLC (%MB1.144.0)
3601	*unterbrechen_uc*	UCHAR	Interruption of an NC program in progress
3602	*flucht_uc*	UCHAR	Return to the programmed path
3603	*stop_uc*	UCHAR	Stop key of the PLC (%MB1.142.0)
3604	*wahlweisehalt_uc*	UCHAR	Optional stop at M1 to PLC (%MB1.133.0)
3605	*einzel_folgesatz_uc*	UCHAR	Changeover between single and following block mode
3606	*satzausblenden_uc*	UCHAR	Hiding of selected DIN blocks
3607	*inch_mm_uc*	UCHAR	Reserved for measurement system changeover
3608	*bahn_strecke_uc*	UCHAR	Reserved for simult. and non-simult.path mode changeover
3609	*polar_kart_uc*	UCHAR	Reserved for polar/cartesian changeover
3610	*absolut_relativ_uc*	UCHAR	Reserved for absolute/incremental dimension changeover
3611	*reserve_auc[21]*	UCHAR	Reserve

Name	Meaning
vorschubhalt_uc	The feed stop key is designed as signal to the PLC and is copied one to one into the DB1 of the PLC.
unterbrechen_uc	A program in progress or a program stopped due to an error of class 3 can be interrupted by actuating the interruption key. In this state, the axes can be moved manually, and single functions or other programs can be carried out. The interruption state can be terminated by a repeated actuation of the interruption key if no program and no single function is active. Subsequently, the interrupted program can be restarted with normal start or aborted by actuating the stop key. In case of a restart, a compensation travel of the axes is carried out first with 1/10 of the max. traversing velocity on the shortest way to the interruption point. *Note: n interruption state, no displacement of coordinates or zero point corrections can be carried out.*
flucht_uc	This variable activates the return function of the controller, i.e. the path is interpolated by means of the programmed return velocity in the direction of the starting point. The number of blocks that can be returned in this way is limited (see MK_RUECKLAUFGRENZE in the configuration instructions).
stop_uc	The stop key is copied one to one into the DB1 (DB1_NC2SPS_PROGRAMMSTOP_B) of the PLC. This state is then normally passed by the PLC to the NC controller as OR operation with external stop keys. A '1' in the stop key interrupts each program processing or single function and disables a restart as long as the value is available in the stop key.

wahlweisehalt_uc	The optional stop key is copied one to one into the DB1 (DB1_NC2SPS_WAHLWEISE_HALT_B) of the PLC and is to be evaluated by the PLC, if necessary. In case of optional stop, after a M1 function, the PLC should disable the read or feed enabling and re-enable it only after the actuation of the start key. 0 = Optional stop inactive, PLC is to let M1 pass without stop 1 = Optional stop active, PLC is to stop the program in case of M1
einzel_folgesatz_uc	This function switches between single and following block mode during program processing. In single block mode (einzel_folgesatz_uc=1) only one block is processed after each actuating of the start key.
satzausblenden_uc	This function activates the function hiding of block, all DIN blocks that start with a slash (/) are ignored and are not processed.

6 Messages

In the following chapter, all admissible messages as well as their handling are described in detail. The messages can be transferred partly in own initiative partly as response to a request.

The messages are characterized by an "identification" (SB0, SB1 and SB2), by the "length" of the message (in byte) as well as by the "data" (user data). The identification "Module" is formally necessary but presently irrelevant during data exchange between HMI and NCR. In case of messages from the HMI to the NCR, the identification should be set generally to 15.

„Handle“ and „Index“ are passed by the controller with all requests and reappear accordingly in the response.

For each message, its meaning is specified and, moreover, its effect on the respective receiver. With messages requiring a response (i.e. requests), also the identification of the necessary response message is specified under "effect".

The identifications are specified as „SB0" and „SB1", they are always valid for an entire group of messages and are specified only once at the beginning of a group. „SB1" defines the message. „SB2“ is specified only where this parameter is presently relevant. In case of all other message types, „SB2“ should be set to 0 for capacity reasons.

The identifications are specified in symbolic form as "defines", an assignment is made in the last section.

Some messages can be transferred both from the HMI to the NCR and from the NCR to the HMI, in case of other messages, only one transfer direction is possible. The possible transfer directions are specified under "Direction".

Under "Data", the declaration of message is specified in compliance with the programming.

If “variable” is specified under length of the message, please observe that the max. length of a message of 512 must not be exceeded!

Data fields exceeding 512 byte are to be transferred in several messages ("transfer block by block"). The single blocks are identified in the SB1 (first block, following block, last block). Please observe that a block transfer must always be composed of at least the "first block" and the "last block".

The messages are subdivided into the following groups:

- Data request and data transfer
- Commands
- Display
- Error messages
- Debug
- Message polling
- CAN open
- Codesys runtime system
- DLL service

6.1 Data request and data transfer

Data transfers can be made in both directions by means of block transfer. From HMI to NCR, these transfers are normally made without being requested while data transfers from NCR to HMI are only made after previous request.

In case of block transfer, on principle the single blocks are to be acknowledged by the receiving side with a special acknowledgment message (block acknowledgment). The acknowledgment decides on a continuation of the transfer or its abort. After the abort, data communication can be continued with a new message.

With the SB0 of the message it is distinguished whether data are to be requested, or a data packet is to be transferred, acknowledged or the data transfer is to be aborted. The SB1 of the message determines which type of data is to be transferred.

Identification:

Name	Meaning
SB0_DATENANFORDERUNG_KUC	Request of a block transfer
SB0_DUE_FIRST_BLOCK_KUC SB0_DUE_NEXT_BLOCK_KUC SB0_DUE_LAST_BLOCK_KUC	First, next and last block of a block transfer
SB0_DUE_FIRST_QUITT_KUC SB0_DUE_NEXT_QUITT_KUC SB0_DUE_LAST_QUITT_KUC	Acknowledgment of the reception of the first, next and last block
SB0_DUE_BREAK_BLOCK_KUC	Premature abort of a block transfer by the transmitter

Messages:

Name	Meaning
SB1_NC_PROG_KUC SB1_ONLINE_PROG_KUC	Transfer of NC programs
SB1_WERKZEUG_KORR_KUC SB1_WERKSTUECK_KORR_KUC	Transfer of tool and workpiece corrections
SB1_ACHS_KORR_KUC	Transfer of axis corrections
SB1_MASCHINENKONSTANTEN_KUC	Transfer of machine constants
SB1_PFELD_BLOCK_KUC	Transfer of parameter field blocks
SB1_IOMODULOFFSET_KUC	Transfer of input/output configuration data
SB1_3D_KORREKTUR_KUC	Transfer of 2D/3D axis correction data
SB1_CAN1_OBJECT_KUC SB1_CAN2_OBJECT_KUC	Transfer of CANopen objects by means of SDO transfer

General sequence:

Direction	SB0	Meaning
A -> B	SB0_DATENANFORDERUNG_KUC	optional, see above
A <- B	SB0_DUE_FIRST_BLOCK_KUC	-
A -> B	SB0_DUE_FIRST_QUITT_KUC	-
A <- B	SB0_DUE_NEXT_BLOCK_KUC	only if more than two blocks are to be sent
A -> B	SB0_DUE_NEXT_QUITT_KUC	only if more than two blocks are to be sent
A <- B	SB0_DUE_LAST_BLOCK_KUC	may have the length of 0
A -> B	SB0_DUE_LAST_QUITT_KUC	-

6.1.1 Request of block transfer (except P-field and CAN)

Block transfers from the NCR to the HMI are normally made only upon previous request. The following message is used.

	Remark
Direction	HMI <- NCR only for machine constants after the download of the NCR HMI -> NCR
Identification	SB0_DATENANFORDERUNG_KUC SB1_WERKZEUG_KORR_KUC SB1_WERKSTUECK_KORR_KUC SB1_MASCHINENKONSTANTEN_KUC SB1_IOMODULOFFSET_KUC
Length	0
Data	-
Meaning	The request of a block transfer is mainly used by the HMI to obtain current data from the NCR. These can be machine constants, parameter field values (section 5.2.1.2), input/output configuration and tool or workpiece correction data. After the download of the controller data are required only once by the NC computer from the HMI, i.e. only the machine constants.
Effect	The receiver starts a block transfer with the required data (SB1_..._KUC). If the required data are not available, a respective error message is generated by the receiver and the request is rejected.
Note	-

6.1.2 Request of parameter field blocks

A P-field block can be transferred from the NCR to the HMI by means of block transfer. The following message is used for the request of block transfer:

	Remark
Direction	HMI -> NCR
Identification	SB0_DATENANFORDERUNG_KUC SB1_PFELD_BLOCK_KUC
Length	4
Data	*startindex_us* [USHORT] *anzahl_us* [USHORT]
Meaning	By means of this message, the HMI is able to read a block of parameters, starting with *startindex_us*. The number of parameters to be read is defined by *anzahl_us*.
Effect	The NC computer transfers the contents of the required parameter field area to the HMI by means of block transfer with SB1_PFELD_BLOCK_KUC. If one of the required parameters cannot be read, a respective error message is signaled and the block transfer is terminated prematurely with SB0_DUE_LAST_BLOCK_KUC.
Note	Parameter field contents can also be transferred in both directions as a list of single parameters, see section 6.2.5.

6.1.3 Request for the reading of CAN objects

With the following message, the HMI can read up to 64 different inputs of the object dictionary of CANopen devices.

	Remark
Direction	HMI -> NCR
Identification	SB0_DATENANFORDERUNG_KUC SB1_CAN1_OBJECT_KUC SB1_CAN2_OBJECT_KUC SB2_CAN_OPERATIONAL_KUC SB2_CAN_PREOPERATIONAL_KUC
Length	8n
Data	*sdoauftrag_ar[n]* (RECORD) *id_uc* (UCHAR) *objektidx_us* (USHORT) *subidx_uc* (UCHAR) *len_l* (LONG)
Meaning	By means of this message, the HMI has reading access to the object directory of CANopen devices. The request message includes n complete *sdoauftrag_ar* structures. For each object to be read, the following data are required: *id_uc* Node number of the CANopen device *objektidx_us* Index of the object to be read *subidx_uc* Index of the sub-object to be read *len_l* Max length of the data to be read in bytes
Effect	If SB2_CAN_PREOPERATIONAL_KUC is transferred together with the request message, all CANopen slaves are first set to state 'PreOperational' (only CAN2). The NCR processes the read requests included in the message one after the other and transfers the read data to the HMI by means of block transfer.
Note	-

6.1.4 Block acknowledgment

Acknowledgment of the transferred block on the application level after evaluation of the block. The format of the block acknowledgment is identical for all transfers.

	Remark
Direction	HMI <- NCR in case of all block transfers from HMI to NCR HMI -> NCR in case of all block transfers from NCR to HMI
Identification	SB0_DUE_FIRST_QUITT_KUC SB0_DUE_NEXT_QUITT_KUC SB0_DUE_LAST_QUITT_KUC SB1_NC_PROG_KUC SB1_ONLINE_PROG_KUC SB1_WERKZEUG_KORR_KUC SB1_WERKSTUECK_KORR_KUC SB1_ACHS_KORR_KUC SB1_MASCHINENKONSTANTEN_KUC SB1_PFELD_BLOCK_KUC SB1_IOMODULOFFSET_KUC SB1_3D_KORREKTUR_KUC
Length	2
Data	*quittung_s* [SHORT]
Meaning	The SB1 of the message determines which transfer is to be acknowledged. Only if it was possible to regularly process the block, the transfer of the next block is possible. The value of *quittung_s* determines whether the transfer is to be continued or aborted. In the NC computer, the reaction time up to the acknowledgment of a block is normally not more than 5 seconds. On the NC side, in no case a timeout monitoring is made of the blocks and acknowledgements expected by the HMI.
Effect	Continuation or abort of the transfer of the current data field or restart of the timeout time for block transfer.
Note	-

quittung_s	**Meaning**
0	Block OK, next block can be transferred.
-50	Provisional block acknowledgment, is repeated every 2 seconds during online program transfer up to final block acknowledgment (online wait). Next block cannot be transferred yet.
All others	Block not OK, interrupt block transfer.

6.1.5 Transfer of NC programs

Prior to the implementation, the NC programs are to be transferred to the main memory of the NC controller. The following messages are used:

	Remark
Direction	HMI -> NCR
Identification	SB0_DUE_FIRST_BLOCK_KUC SB0_DUE_NEXT_BLOCK_KUC SB0_DUE_LAST_BLOCK_KUC SB1_NC_PROG_KUC SB1_ONLINE_PROG_KUC
Length	Variable
Data	Contents of the NC program
Meaning	In case of NC programs, a distinction is made between online and offline transfer. In case of offline transfer (SB1_NC_PROG_KUC), the NC program is to fit completely into the memory of the NC controller and it remains there to the deleting of the program, the deleting of the entire program memory or the de-energizing of the controller. The NC program can be started at an optional frequency without new transfer. On the contrary, a program transferred online (SB1_ONLINE_PROG_KUC) can have an optional size, without new transfer it can, however, be started only once. It is valid for both features that the program can be started already during the transfer.
Effect	After the syntactic checking of the block, the NC computer transfers the respective acknowledgment message, see section 6.1.4 Block acknowledgment. In case of an online transfer, a provisional acknowledgment is transferred (-50) as soon as the memory limit is reached. This acknowledgment is repeated by the NCR without being called until the NC program is processed to such an extent that the last transferred block can be stored. Only subsequently, a block acknowledgment is made.
Note	Online programs can be processed only sequentially. Therefore, branchings and loops are not allowed. Subroutine calls are, however, possible.

6.1.6 Transfer of tool and workpiece corrections

Tool and workpiece corrections can be transferred with the following messages. The structure of these files is shown hereafter.

	Remark
Direction	HMI <- NCR HMI -> NCR
Identification	SB0_DUE_FIRST_BLOCK_KUC SB0_DUE_NEXT_BLOCK_KUC SB0_DUE_LAST_BLOCK_KUC SB1_WERKZEUG_KORR_KUC SB1_WERKSTUECK_KORR_KUC
Length	Variable
Data	Contents of the correction file
Meaning	A tool correction file is transferred with SB1_WERKZEUG_KORR_KUC. The file includes axis-specific tool lengths and axis-independent tool data. For each tool, the tool lengths define the distance of the tool tip from a common point, e.g. the tool carrier. The tool data are freely usable parameters for the individual description of each tool. The tools are numbered from T0 to Tn where T0 as reference tool should have tool lengths of 0. A workpiece correction file is transferred with SB1_WERKSTUECK_KORR_KUC. The file includes axis-specific workpiece offsets. These offsets define the distance of the workpiece points from the reference coordinate system S0. The workpiece zero points are numbered from S1 to Sn. As reference coordinate system, S0 has a special state, irrespective of that, it is, however, always transferred together with the workpiece file. For further information about the tool and workpiece corrections please check the programming instructions of the S and T functions.
Effect	In the NCR, the transferred corrections are accepted if currently no task is being processed. Otherwise, a respective error is signaled and the transfer is aborted with a negative acknowledgment. A transfer of correction data is, therefore, not possible in the program in progress! In the HMI, the corrections can be stored or processed and can be returned to the NCR. Upon accepting of a block, the receiver transfers the respective acknowledgment message, see section 6.1.4 Block acknowledgment.
Note	-

Structure of the correction files:

In general, tool and workpiece correction files are structured on the basis of the same scheme. After a 128 byte long header, the correction values follow in binary format.

Offset	Name	Type	Meaning
0	*kenn_ac[127]*	CHAR	Zero-terminated list with k identifying letters for the identification of the following data
127	*system_uc*	UCHAR	Number of the S or T system depending whether there are tool or workpiece corrections
128	*korr_aad[n][k]*	DOUBLE	Correction values in input units. n=number of coordinate systems, k=number of identifying letters in kenn_ac

The list of identifying letters (*kenn_ac*) includes a letter each for each correction value of a coordinate system (Tn or Sn). In case of a workpiece correction file, these are the axis letters (X,Y,Z,...) of the axes, the coordinate system offset of which is included in the file.

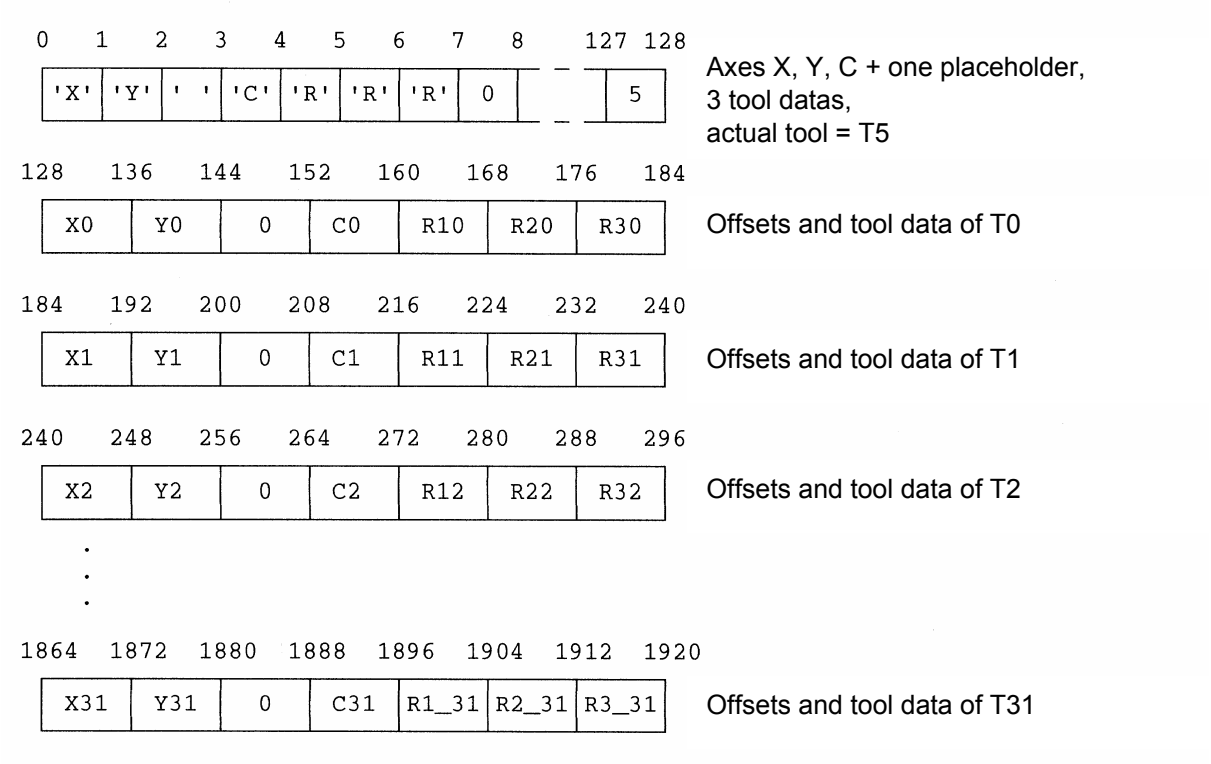
In case of a tool correction file, this list includes in addition a 'R' for each input of tool data of a tool (Tn). Tools are to be equated with coordinate systems. The coordinate system offsets correspond to the tool lengths of the single tools.

In the list of the identifying letters, blanks (ASCII 32) are allowed as wildcards for non-configured axes or tool data. The correction value assigned to a single wildcard is to be entered into the file, it is, however, ignored by the receiver.

The number of the current or to be switched S or T coordinate system (*system_uc*) can be transferred optionally. A value < 255 signals a valid S or T number.

The correction data (*korr_aad*) always include the coordinate systems S0-Sn-1 or T0-Tn-1 in ascending order where n corresponds to the number of available systems. The HMI may also transfer less systems, it must, however, always start with S0 or T0. Within each system, the correction data are organized in the order of the identifying letters.

Example:



6.1.7 Transfer of axis corrections

Axis corrections are used for the compensation of leadscrew errors. The following message is used for the transfer of an axis correction. The structure of the file is specified afterwards.

	Remark
Direction	HMI -> NCR
Identification	SB0_DUE_FIRST_BLOCK_KUC SB0_DUE_NEXT_BLOCK_KUC SB0_DUE_LAST_BLOCK_KUC SB1_ACHS_KORR_KUC
Length	Variable
Data	Contents of the correction file
Meaning	An axis correction file is transferred with SB1_ACHS_KORR_KUC. The file includes a table with axis-specific correction values. The correction values correct the target position versus the target position presetting of the NC controller. The correction table can be used for the compensation of leadscrew errors. By means of the definition range in the axis correction file it is also possible to compensate an error returning periodically in the traversing range. For this compensation, the range size of a period is input in the modulo value instead of the difference between maximum and minimum value. The correction values are always distributed on the path entered in the modulo value, starting with the minimum value of the definition range. Moreover, the correction values are repeated up to the maximum value of the definition range.
Effect	In the NCR, the transferred axis corrections are accepted if currently no task is being processed. Otherwise, a respective error is signaled and the transfer is aborted with a negative acknowledgment. A transfer of axis correction data is, therefore, not possible in the program in progress! Upon accepting of a block, the NC computer transfers the respective acknowledgment message, see section 6.1.4 Block acknowledgment.
Note	-

Structure of the correction file:

The controller distinguishes 2 different formats of correction data. The old format includes 4096 correction values of 8 bit per axis each with a correction range of –128 to +127 increments. The new format includes 1024 correction values of 32 bit per axis each with a possible correction range of +/- 2^{20} increments.

Old format:

The axis correction file is structured as binary file in compliance with the following scheme. After a header of 128 byte, fields are following that describe the definition range of the axis correction, and then 4096 correction values per axis.

Offset	Name	Type	Meaning
0	*kenn_ac[128]*	CHAR	Zero-terminated list with k identifying letters for the identification of the following data and subsequently, as an option, a comment in ASCII format
128	*def1_aal[m][2]*	LONG	Min. and max. value of the correction range in increments per axis
128+8m	*def2_aaul[m][2]*	ULONG	Modulo value for the relative actual position range in increments per axis
128+16m	*korr_aaac[n][m][8]*	CHAR	Correction values in increments. n=512=number of 8-byte packets per axis, m=number of identifying letters in kenn_ac

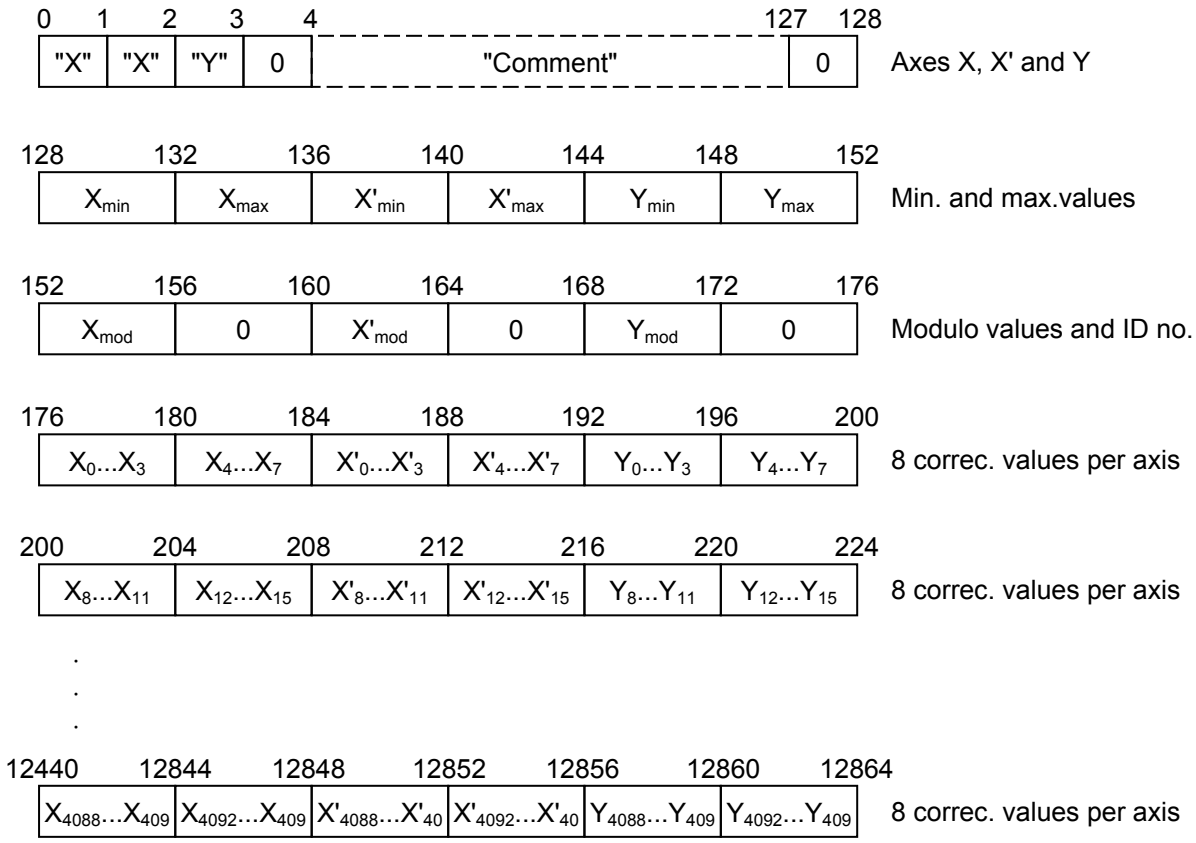
The list of identifying letters (*kenn_ac*) includes a letter each for each axis in the correction file. These are the axis letters (X,Y,Z,...) of the respective axes. If an axis letter occurs twice, the first one represents the master axis and the second one the slave axis. If in case of synchronous axis, the letter occurs only once, master and slave axis are corrected to the same extent.

The first definition block (*def1_aal*) includes per axis the minimum and maximum value of the actual position range in which the correction table is effective. The order is the same as the order of the identifying letters in kenn_ac. For each axis, first the minimum value and then the maximum value is specified in increments referred to the reference point (without basic offset).

The second definition block (*def2_aaul*) includes per axis a modulo value each in increments and the identifying number 0 that defines that the data in the following block are available in the old format. The modulo value defines the relative actual position range that is covered by the correction table. This range can be repeated several times between the minimum and the maximum value. The order corresponds to the order of the identifying letters in kenn_ac.

The correction data (*korr_aaac*) always include 4096 correction values with signs per axis. The arrangement of the correction values is made in 512 packets in groups of 8 values per axis. The order of the group corresponds to the order of the identifying letters in kenn_ac. Each correction value has a value range of -128 to +127 increments. In case of a slave axis, the correction value refers to the already corrected position of the master axis.

Example:



New format:

The axis correction file is structured as binary file in compliance with the following scheme. After a header of 128 byte, fields are following that describe the definition range of the axis correction, and then 1024 correction values per axis.

Offset	Name	Type	Meaning
0	*kenn_ac[128]*	CHAR	Zero-terminated list with k identifying letters for the identification of the following data and subsequently, as an option, a comment in ASCII format
128	*def1_aal[m][2]*	LONG	Min. and max. value of the correction range in increments per axis
128+8m	*def2_aaul[m][2]*	ULONG	Modulo value for the relative actual position range in increments and identifying number for the file type per axis
128+16m	*korr_aaal[n][m][2]*	LONG	Correction values in increments. n=512=number of 8-byte packets per axis, m=number of identifying letters in kenn_ac

The list of identifying letters (*kenn_ac*) includes a letter each for each axis of the correction file. These

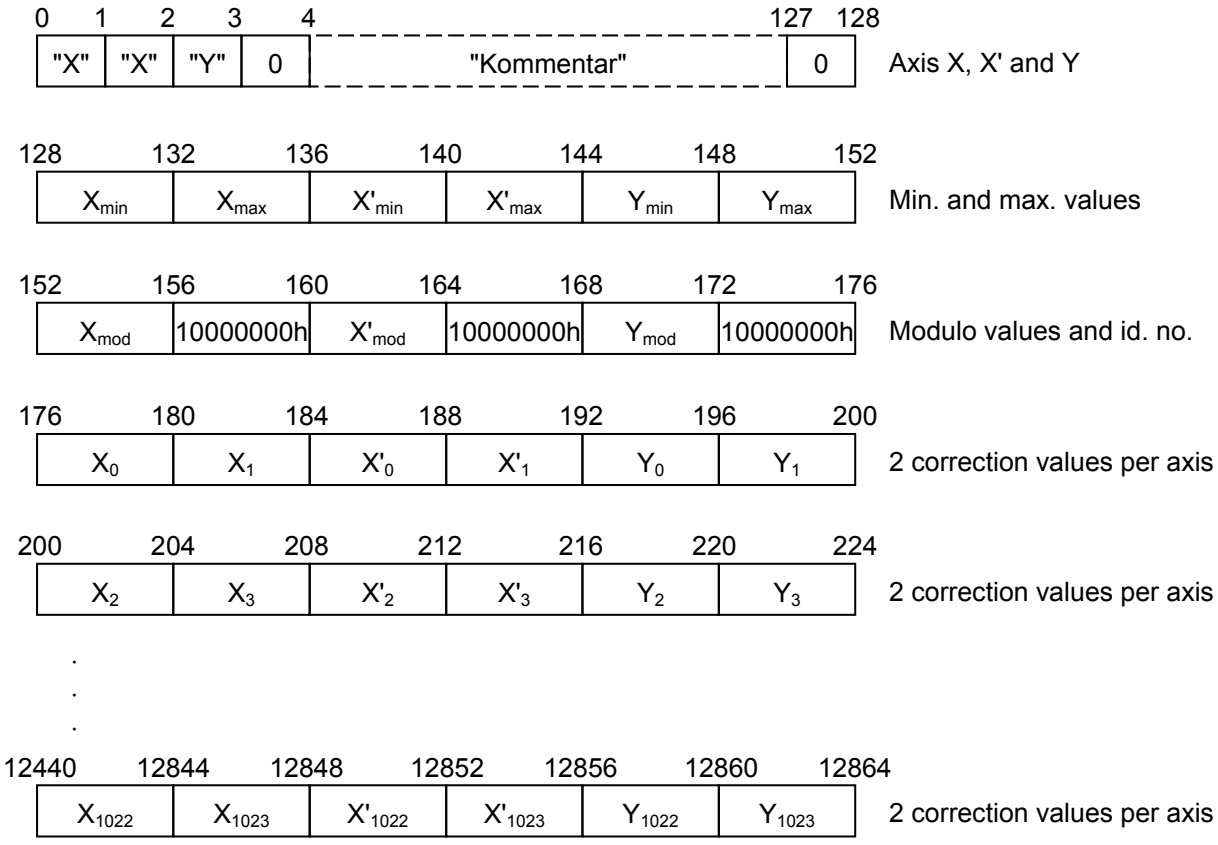
are the axis letters (X,Y,Z,...) of the respective axes If an axis letter occurs twice, the first one represents the master axis and the second one the slave axis. If in case of a synchronous axis, the letter occurs only once, master and slave axis are corrected to the same extent.

The first definition block (*def1_aal*) includes per axis the minimum and maximum value of the actual position range in which the correction table is effective. The order is the same as the order of the identifying letters in kenn_ac. For each axis, first the minimum value and then the maximum value is specified in increments referred to the reference point (without basic offset).

The second definition block (*def2_aaul*) includes per axis a modulo value each in increments and the identifying number 0x10000000 that defines that the data in the following block are available in the new format. The modulo value defines the relative actual position range that is covered by the correction table. This range can be repeated several times between the minimum and the maximum value. The order corresponds to the order of the identifying letters in kenn_ac.

The correction data (*korr_aaac*) always include 1024 correction values with signs per axis. The arrangement of the correction values is made in 512 packet in groups of 2 values per axis. The order of the group corresponds to the order of the identifying letters in kenn_ac. Each correction value has a value range of +/-2^20 increments. In case of a slave axis, the correction value refers to the already corrected position of the master axis.

Example:



6.1.8 Transfer of machine constants

The machine constants include a machine-specific configuration of the NC controller. They are transferred with the following message:

	Remark
Direction	HMI <- NCR HMI -> NCR
Identification	SB0_DUE_FIRST_BLOCK_KUC SB0_DUE_NEXT_BLOCK_KUC SB0_DUE_LAST_BLOCK_KUC SB1_MASCHINENKONSTANTEN_KUC
Length	Variable
Data	Contents of the machine constant file
Meaning	The machine constants file is an ASCII file with machine constant identifiers, with assigned constants and optional comments. The possible machine constant identifiers are included in the configuration instructions of the NC controller.
Effect	The NC computer accepts the transferred machine constants and saves them in the flash PROM. Some of them become effective immediately other only if the task currently in process has been terminated. Machine constants that influence the memory distribution within the controller become effective only after the next reset. In the HMI, the MK can be stored or processed and returned to the NCR. Upon accepting of a block, the receiver transfers the respective acknowledgment message, see section 6.1.4 Block acknowledgment.
Note	-

6.1.9 Transfer of parameter field blocks

With the following message it is possible to read a block from the parameter field or to write the block into the parameter field.

	Remark
Direction	HMI <- NCR Reading from the parameter field HMI -> NCR Writing into the parameter field
Identification	SB0_DUE_FIRST_BLOCK_KUC SB0_DUE_NEXT_BLOCK_KUC SB0_DUE_LAST_BLOCK_KUC SB1_PFELD_BLOCK_KUC
Length	4 + 8n
Data	*startindex_us* [USHORT] *anzahl_us* [USHORT] *pfeld_ad[n]* [DOUBLE]
Meaning	Each transferred block includes at least the index (*startindex_us*) and the number (*anzahl_us*=n) of parameter field values (*pfeld_ad*) in this block. The max. 63 parameter field values in a block are entered in ascending order, starting with *startindex_us*.
Effect	Upon reception of the data in the NCR, the transferred values are input into the respective parameters. If one of the parameters cannot be written, a respective error message is signaled and the block transfer is aborted with a negative acknowledgment. In the HMI, the transferred values can be displayed or processed, if necessary. Upon accepting of a block, the receiver transfers the respective acknowledgment message, see section 6.1.4 Block acknowledgment.
Note	Parameter field contents can be transferred in both directions, also as list of single parameters, see section 6.2.5.

6.1.10 Transfer of input and output configuration data

For a module-specific display of the current states of the analog and digital inputs and outputs of the PLC, the HMI requires the information which of the input/output modules are connected to the controller and how many inputs and outputs are connected. The following message is used for the transfer of this information:

	Remark
Direction	HMI <- NCR
Identification	SB0_DUE_FIRST_BLOCK_KUC SB0_DUE_NEXT_BLOCK_KUC SB0_DUE_LAST_BLOCK_KUC SB1_IOMODULOFFSET_KUC
Length	n*12
Data	*iomodulinfo_ar[n]* (RECORD) *modultyp_uc* (UCHAR) *dianzahl_uc* (UCHAR) *doanzahl_uc* (UCHAR) *aianzahl_uc* (UCHAR) *aoanzahl_uc* (UCHAR) *dioffset_uc* (UCHAR) *dooffset_uc* (UCHAR) *aioffset_uc* (UCHAR) *aooffset_uc* (UCHAR) *nodeid_uc* (UCHAR) *reserve_auc[2]* (UCHAR)
Meaning	Each block includes n (0-42) IO-module information structures of 12 byte length each. For each input/output module connected to the controller, the following information is transferred: *modultyp_uc* Type of connected input/output module 1 = local digital IO-module (EC-IO) 2 = local analog IO-module (EC-ADA, EC-ADC) 3 = CAN-IO-module (SLIO, CANopen) *dianzahl_uc* Number of digital inputs of the module *doanzahl_uc* Number of digital outputs of the module *aianzahl_uc* Number of analog inputs of the module *aoanzahl_uc* Number of analog outputs of the module *dioffset_uc* Start index of the digital inputs in the input field in case of local modules referred to in *dpr_PR->sps_r.din_aus,* in case of CAN-IO modules referred to in *dpr_PR->sps_r.canin_aus* *dooffset_uc* Start index of the digital outputs in the output field in case of local modules referred to in *dpr_PR->sps_r.dout_aus,* in case of CAN-IO modules referred to in *dpr_PR->sps_r.canout_aus*

	aioffset_uc	Start index of the analog inputs in the input field in case of local modules referred to in *dpr_PR->sps_r.ain_aus,* in case of CAN-IO modules referred to in *dpr_PR->sps_r.canin_aus*
	aooffset_uc	Start index of the analog outputs in the output field in case of local modules referred to in *dpr_PR->sps_r.aout_aus,* in case of CAN-IO modules referred to in *dpr_PR->sps_r.canout_aus*
	nodeid_uc	In case of *modultyp_uc* = 3 CAN address of the module, otherwise 0
	The assigning of the input/output copy is described in section 5.3 „Input and output signals of the PLC (sps_r)“.	
Effect	The HMI stores the transferred information about the module-specific display of the input/output states. Upon accepting of a block, the HMI transfers an acknowledgment message with SB1_IOMODULOFFSET_KUC, see section 6.1.4 Block acknowledgment.	
Note	-	

6.1.11 Transfer of 2D/3D axis correction data

The 2D/3D axis correction is used for height compensation of an axis as a function of one or two other axes. For the transfer of a 2D/3D axis correction file, the following message is used. The structure of the file is shown in the following.

	Remark
Direction	HMI -> NCR
Identification	SB0_DUE_FIRST_BLOCK_KUC SB0_DUE_NEXT_BLOCK_KUC SB0_DUE_LAST_BLOCK_KUC SB1_3D_KORREKTUR_KUC
Length	Variable
Data	Contents of the correction file
Meaning	A 2D/3D axis correction file is transferred with SB1_3D_KORREKTUR_KUC. It includes a table with equidistant reference points for a target position correction of the correction axis as a function of the target position of basic axes. A linear interpolation is made between the reference points. The definition range of the table is determined by the basic axes, the range refers to the reference coordinate system including the basic offset. Outside the definition range of the table, the correction value of the adjacent reference point is used. For further information about the activation/deactivation of the 2D/3D axis correction, please check the programming instructions of the G233.
Effect	The transferred 2D/3D corrections are accepted in the NCR if currently no task is being processed and if the 2D/3D correction is not activated. Otherwise, a respective error is signaled and the transfer is aborted with a negative acknowledgment. After the transfer, the correction is not activated automatically but is to be activated with the G233. A transfer of correction data is not possible in the program in progress. A probably already activated 2D/3D correction is to be deactivated prior to make the transfer. Upon accepting of a block, the NC computer transfers the respective acknowledgment message with SB1_3D_KORREKTUR_KUC, see section 6.1.4 Block acknowledgment.
Note	-

Structure of the correction file:

The 2D/3D correction file is structured as binary file in compliance with the following scheme. After the 128 byte long header, fields follow that describe the definition range and the number of reference points of the table for each basic axis. Subsequently, the reference point table follows.

Offset	Name	Type	Meaning
0	*kenn_ac[128]*	CHAR	Zero-terminated list with k+1 identifying letters for the identification of the following data
128	*def_ar[k]* *min_f* *max_f* *anzahl_ul*	RECORD FLOAT FLOAT ULONG	Definition range of the basic axes Start position in input units End position in input units Number (m|n) of reference points between min_f and max_f
128+12k	*korr_aaf[m][n]*	FLOAT	n*m Reference points in input units m = Number of reference points of the 1st basic axis n = Number of reference points of the 1st basic axis or 1, if not available

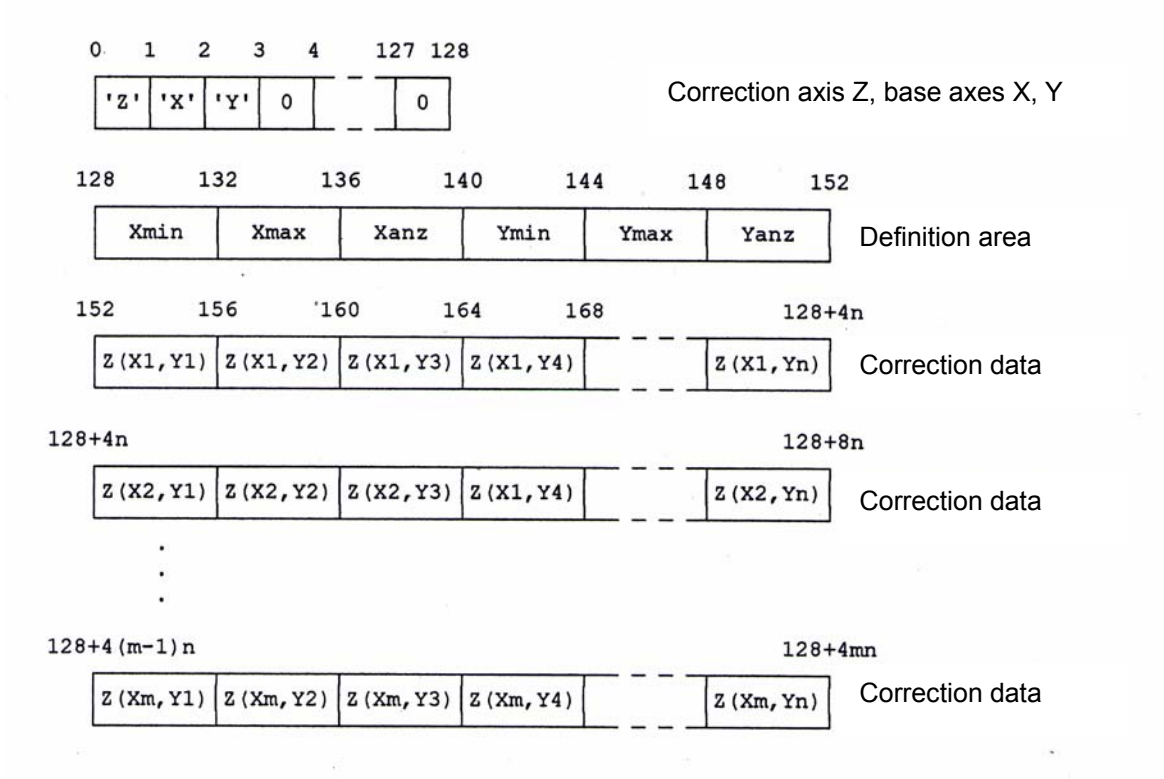
The list of identifying letters (*kenn_ac*) includes the axis letters of the respective axis. The first letter is the correction axis, the following letters are the basic axes. One or two basic axes can be specified.

The definition range (*def_ar*) is described for each basic axis by means of minimum position (*min_f*) and maximum position (*max_f*) of the axis. In case of two basic axes, a square range is generated in this way for which the table is defined. The number of reference points per edge length is established by *anzahl_ul*. The axis sequence in *def_ar* corresponds to the letter sequence of the basic axes in *kenn_ac*.

The table of the correction data (*korr_aaf*) is structured in lines and columns. The number of lines corresponds to the number of reference points of the 1st basic axis and the number of columns corresponds to the number of reference points of the 2nd basic axis. The number of columns is 1, if only one basic axis is available. In the table, first all columns of the first line (*korr_aaf[0][0..n-1*]) are specified, then all columns of the second line (*korr_aaf[1][0..n-1]*) and finally all columns of the last line (*korr_aaf[m-1][0..n-1]*).

Example:

Correction of the Z-axis as a function of X,Y



6.1.12 Reading and writing of CAN objects

Asynchronous data that are to be written or read via the CAN bus with SDO can be exchanged between HMI and NCR by means of block transfer.

	Remark
Direction	HMI -> NCR (write) HMI <- NCR (read)
Identification	SB0_DUE_FIRST_BLOCK_KUC SB0_DUE_NEXT_BLOCK_KUC SB0_DUE_LAST_BLOCK_KUC SB1_CAN1_OBJECT_KUC SB1_CAN2_OBJECT_KUC SB2_CAN_OPERATIONAL_KUC SB2_CAN_PREOPERATIONAL_KUC
Length	Variable
Data	*sdoauftrag_ar[n]* (RECORD) *nodeid_uc* (UCHAR) *objektidx_us* (USHORT) *subidx_uc* (UCHAR) *len_l* (LONG) *data_auc[]* (UCHAR)
Meaning	For parameterizing of CANopen devices, the HMI has writing and reading access to the object directory of the devices by means of these messages. The objects to be written or the objects read are transferred by means of block transfer from the HMI to the NCR or from the NCR to the HMI. The size of each single block is limited to 512 byte. The number of blocks of a block transfer is not limited. Within block transfer, several objects can be written or read. The size of the single objects is variable and is max. 2 GByte. Each object is defined by the following information: *nodeid_uc* Module number/ ID of the CANopen device *objektidx_us* Index of the object to be written *subidx_uc* Index of the sub-objects to be written *len_l* Length of data in byte or < 0 in case of error during reading *data_uc[]* Data to be written or data read The header information (*nodeid_uc* bis *len_l*) is always completely included in a block.
Effect	See following description
Note	-

Effect during writing:

If SB2_CAN_PREOPERATIONAL_KUC is transferred with the first block, all CANopen slaves are first set to state 'PreOperational' (only CAN2). The NC controller transfers the data transferred in blocks by the HMI to the respective nodes by means of SDO writing. The NCR acknowledges each received block with a positive acknowledgment if all available SDO transfers have been made successfully. If errors occurred during the transfer of the SDOs, a negative block acknowledgment is transferred.

Acknowledgment	Status	Meaning
0	SDO_READY	Writing successful
-70	SDO_ERROR	Transfer aborted
-71	SDO_TIMEOUT	Timeout (module does not respond)
-72	SDO_ABBRUCH	Transfer aborted by the module

Effect during reading:

The NC controller reads the objects requested by the HMI by means of SDO reading from the respective nodes and transfers them by means of block transfer to the HMI. The HMI acknowledges each received block with the respective acknowledgment message. If an error occurs during reading of an object, it is identified by a negative value in the *len_l*.

len_l	Status	Meaning
>= 0	SDO_READY	Reading successful
-70	SDO_ERROR	Transfer aborted
-71	SDO_TIMEOUT	Timeout (module does not respond)
-72	SDO_ABBRUCH	Transfer aborted by the module

6.2 Single commands

Single commands are in general initiated by the HMI. Upon implementation of the command, some of them are responded by a message from the NCR. The response is also re-transferred to the HMI as a single command.

Under "Effect" it is described in addition whether and how the command is responded by the NC-controller.

Identification:

Name	Meaning
SB0_AUFTRAG_KUC	

Messages:

Name	Meaning
SB1_ASCII_EINZELFUNKTION_KUC SB1_EINZELFKT_MIT_QUITTUNG_KUC	Implementation of a single NC block
SB1_CLEAR_MEM_KUC SB1_CLEAR_USERMEM_KUC	Clearing of all NC programs from the main memory
SB1_CLEAR_PROG_KUC	Clearing of a NC programs from the main memory
SB1_ANF_PFELD_LISTE_KUC	Reading of parameters via a list of P-field numbers
SB1_PFELD_LISTE_KUC	Transfer of parameters via a P-field list
SB1_PFELD_ANZEIGE_KUC	Selection of the parameters for the cyclic P-field display
SB1_EINGABE_ENDE_KUC	Message of the end of the input function to the NC controller
SB1_VORLAUF_ENDE_KUC SB1_VORLAUF_AM_ZIEL_KUC	Status messages in case of block search
SB1_QUITTUNG_AUF_EINZELFKT_KUC	Acknowledgment upon implementation of a single function
SB1_GET_ACHSINFO_KUC	Request of information about an axis
SB1_ACHSINFO_KUC	Transfer of information about an axis
SB1_CHANGE_REGPARAM_KUC	Online change of the control parameters of an axis
SB1_PROGRAMM_START_KUC	Start of a NC program with acknowledgment message
SB1_PROGRAMM_ENDE_KUC	Acknowledgment message at the program end
SB1_PROGRAMM_ABBRUCH_KUC	Abort of a NC program
SB1_GET_PROGNAME_KUC	Request of a program name in case of symbolic programming
SB1_PROGNAME_KUC	Transfer of a program name, response to SB1_GET_PROGNAME_KUC

6.2.1 Implementation of a single NC block

This command is used for the implementation of a single NC block in the controller. The block to be implemented is included in the user data load of the command.

	Remark
Direction	HMI -> NCR
Identification	SB0_AUFTRAG_KUC SB1_ASCII_EINZELFUNKTION_KUC SB1_EINZELFKT_MIT_QUITTUNG_KUC
Length	Variable
Data	NC block as zero-terminated ASCII string
Meaning	With SB1_ASCII_EINZELFUNKTION_KUC, a NC block is transferred to the NCR for implementation that is implemented by the controller without acknowledgment message. With SB1_EINZELFKT_MIT_QUITTUNG_KUC, a NC block is transferred to the NCR for implementation that is implemented by the controller with acknowledgment message. The NC block is to be input as zero-terminated string into the message. The length of the message corresponds to the length of the string including the terminating zero. Please observe that the G-functions G22 and G252 cannot be implemented as single function. During the implementation, presently they are ignored as single function.
Effect	The NC block is checked syntactically in the controller and implemented if currently no task is being processed. A respective error message is signaled if the block cannot be implemented or if a syntactic error is available. If the single function is triggered with SB1_EINZELFKT_MIT_QUITTUNG_KUC, the NCR transfers an acknowledgment message with SB1_QUITTUNG_AUF_EINZELFKT_KUC. Upon correct implementtation of the block, a positive acknowledgment is signaled, otherwise a negative one.
Note	If several single functions are transferred one after the other with SB1_EINZELFKT_MIT_QUITTUNG_KUC without expecting the end of the first transfer only an acknowledgment message is transferred at the end of the first function. Irrespective of that, an error message is signaled with the following blocks that does not cause the abort of the already active NC block!

6.2.2 Clearing of all NC programs from the main memory

All NC programs with different program numbers loaded into the NCR are stored in the main memory of the controller. With the following two messages, either all NC programs can be cleared from the main memory or all NC programs except the cycle programs.

	Remark
Direction	HMI -> NCR
Identification	SB0_AUFTRAG_KUC SB1_CLEAR_MEM_KUC SB1_CLEAR_USERMEM_KUC
Length	0
Data	-
Meaning	In case of NC programs, a distinction is made between cycle programs and user programs. For the NCR, the following definitions are valid: all NC programs with program numbers greater than or equal to 8000 are cycle programs. In general, they are generated technology-specific by the machine manufacturer. All programs with program numbers less than 8000 are user programs. With SB1_CLEAR_MEM_KUC, the entire NC program memory including all cycle and user programs is cleared and the respective memory is made available again to the NC program management. With SB1_CLEAR_USERMEM_KUC, only the user programs are cleared and the respective memory is made available again to the NC program management. The messages do not include a user data load.
Effect	The NCR clears the NC programs and initializes the respective management structures. No response message is transferred to the HMI.
Note	-

6.2.3 Clearing of a single NC program from the main memory

With the following message, a NC program is cleared from the main memory of the NCR and the required memory space is enabled.

	Remark
Direction	HMI -> NCR
Identification	SB0_AUFTRAG_KUC SB1_CLEAR_PROG_KUC
Length	2
Data	*prognr_us* (USHORT)
Meaning	In *prognr_us,* the program number of the NC program to be cleared is specified. Valid program numbers are between 1 and 65534.
Effect	The NC program with the specified program number is cleared from the main memory. No response message is transferred to the HMI.
Note	-

6.2.4 Reading of parameters via a list of P-field numbers

With the following message, up to 51 different parameters at a time can be read from the parameter field of the controller.

	Remark
Direction	HMI -> NCR
Identification	SB0_AUFTRAG_KUC SB1_ANF_PFELD_LISTE_KUC
Length	2*n*
Data	*pidx_aus[n]* (USHORT)
Meaning	In *pidx_aus*, the P-field numbers of the parameters to be read are stored. The value n corresponds to the number of parameters to be read. The last P-field number of the list (*pidx_aus[n-1]*) can be set at the list end to 0xffff. In this case, the number is also transferred in the response message. Please observe that the number *n* of the parameters to be requested at a time is limited to 51 due to the message buffer size.
Effect	The NCR reads the requested parameters from the parameter field and returns them with SB1_PFELD_LISTE_KUC to the HMI.
Note	-

6.2.5 Transfer of parameters via a P-field list

With the following message, up to 51 different parameter field values at a time can be transferred in both directions.

	Remark
Direction	HMI <- NCR nach Anforderung mit SB1_ANF_PFELD_LISTE_KUC HMI -> NCR
Identification	SB0_AUFTRAG_KUC SB1_PFELD_LISTE_KUC
Length	10*n*
Data	*pfeld_ar[n]* (RECORD) *pidx_us* (USHORT) *pval_d* (DOUBLE)
Meaning	This message is transferred from the HMI to the NCR in order to write parameters into the P-field. The message is transferred from the NCR to the HMI as response to the SB1_ANF_PFELD_LISTE_KUC in order to read parameters of the P-field from the HMI. The parameters to be transferred are transferred by P-field number and P-field value in the order number, value, number, value etc.. In *pidx_us* the P-field number is transferred and in *pval_d* the respective value. The value n corresponds to the number of parameters to be transferred. The P-field number of the last element of list (*pfeld_ar[n-1].pidx_us*) may be set at the end of the list to 0xffff in order to simplify the evaluation in the HMI. The NCR transfers the list end flag only if the request message also ended with a list end flag. Please observe that the number *n* of the parameters to be transferred with the message is limited to 51 due to the message buffer size.
Effect	The NCR writes the values transferred to it into the respective parameters in the parameter field. An error message is signaled if one of the parameters cannot be overwritten. The HMI may display the read parameters or process them.
Note	-

6.2.6 Selection of the parameters for the cyclic P-field display

Within the cyclic data from NCR to HMI (dpr_PR->nc2mmi_r.pfeldanz_r) in the DPR, 32 freely definable parameters of the parameter field can be displayed. The selection of the parameters to be displayed is made with the following message:

	Remark
Direction	HMI -> NCR
Identification	SB0_AUFTRAG_KUC SB1_PFELD_ANZEIGE_KUC
Length	4n
Data	*panz_ar[n]* (RECORD) *aidx_us* (USHORT) *pidx_us* (USHORT)
Meaning	Each value in the cyclic display is determined by a display index (*aidx_us*) and the assigned parameter field number (*pidx_us*). Optional P-field numbers can be assigned to the available display indices 0 to 31. In the user data load, the display index and the P-field number are always specified in pairs. For an identification of unused display fields, a value of 0xffff can be assigned to the assigned P-field numbers.
Effect	In the display indices to be changed, first P-field numbers of 0xffff are entered in order to identify the assigned P-field value as being invalid. Subsequently, the correct P-field values are updated in the display and are entered finally in the respective P-field number. In this way, in case of a display changeover, the HMI can be sure to get the correct P-field values by comparing the P-field numbers. A display index that is to large (>31) causes a respective error message in the NCR. For P-field numbers outside of the defined parameter field, a P-field value of 0 is displayed in the P-field display.
Note	-

6.2.7 Message of the end of the input function to the NC controller

By means of the G-function G252, a NC program can request a value input on the control panel. The following message is used as information about the end (Enter) or the abort (Escape) of the input function.

	Remark
Direction	HMI -> NCR
Identification	SB0_AUFTRAG_KUC SB1_EINGABE_ENDE_KUC
Length	1
Data	*abbruch_uc* (UCHAR)
Meaning	Upon termination of the input by the operator, the HMI is to inform the NCR about the end of the input in order to make both a termination of the G252 by the controller possible and the implementation of the next NC-block. A value of 0 for *abbruch_uc* signals a terminated input (Enter). The value 1 signals the abort of the input. In case that a new value has been input and the input was not aborted, the HMI is to store the input value in the target parameter in the P-field, i.e. in front of the message of the input end to the NCR. The input function is started with SB0_DISPLAY_KUC and SB1_GFKT_EINGABEXY_KUC. For further information please check section 6.3.3.
Effect	According to the programming of G252, the value *abbruch_uc* is to decide whether the logic next block is to be implemented or whether a jump is to be made to the block numbers programmed under G252. The jump is made if *abbruch_uc* is unequal 0 and if a block number is programmed under the address 'E' of G252.
Note	-

6.2.8 State messages in case of block search

The function block search makes it possible to enter a NC program at a determined point without passing over the previous blocks (the blocks are interpreted but not output). With the following message, the NCR informs the HMI about the block search status.

	Remark
Direction	HMI <- NCR
Identification	SB0_AUFTRAG_KUC SB1_VORLAUF_ENDE_KUC SB1_VORLAUF_AM_ZIEL_KUC
Length	0
Data	-
Meaning	With SB1_VORLAUF_ENDE_KUC, the reaching of the search block limit is signaled to the HMI after a "Start in block search ". Depending on the program size, this may last a few seconds or several minutes. The search block limit is the point in which the trigger condition defined in P528-P533 is met and the NC-program is to be continued regularly. With SB1_VORLAUF_AM_ZIEL_KUC, the reaching of the start position after a "Start after block search" is signaled to the HMI. A "Start after block search " is a normal start after the reaching of the search block limit. With this start, the start position of the axes determined in the block search is approached, in order to be able to continue the NC-program with the next start from the search block limit. Normal start and "Start in block search " can be triggered by the HMI with the start button (section 5.5) or by means of a start message (section 6.2.13).
Effect	Upon reception of SB1_VORLAUF_ENDE_KUC, the HMI should inform the operator about the search limit reached and should prompt him to actuate the start button in order to approach the start position of the axes. Upon reception of SB1_VORLAUF_AM_ZIEL_KUC, the HMI should inform the operator about the reaching of the start position and prompt him to actuate the start button, in order to start the NC program in the approached point. As an alternative, an additional "Start in block search " can be made to search the next trigger position, e.g. in a loop.
Note	-

6.2.9 Acknowledgment upon implementation of a single function

The following message is transferred from the NCR after the implementation of a single function with SB1_EINZELFKT_MIT_QUITTUNG_KUC:

	Remark
Direction	HMI <- NCR
Identification	SB0_AUFTRAG_KUC SB1_QUITTUNG_AUF_EINZELFKT_KUC
Length	1
Data	*quittung_c* (CHAR)
Meaning	In *quittung_c* the NCR signals whether the single function could be implemented or not. A value of 0 means that the function was implemented completely. Each other value indicates that the signal function could either not be started or was aborted prematurely.
Effect	The HMI is enabled to start the next single function or the next program.
Note	-

6.2.10 Request of information about an axis

With the following message, the HMI is able to call static information from the NCR about an axis.

	Remark
Direction	HMI -> NCR
Identification	SB0_AUFTRAG_KUC SB1_GET_ACHSINFO_KUC
Length	2
Data	*kenn_c* (CHAR) *achsnr_uc* (UCHAR)
Meaning	In the message, the identifying letter (*kenn_c*) of the axis is specified to which the required information is to be transferred. Please observe that the axis identifying letter is to be specified in a case-sensitive way. As an alternative to the identifying letter, also the axis number (*achsnr_uc*) can be specified. In this case, *kenn_c* is to be set to 0.
Effect	The controller returns the requested information with SB1_ACHSINFO_KUC. If the specified axis is not configured, the NCR sets the response message *kenn_c* to -1.
Note	-

6.2.11 Transfer of information about an axis

With the following message, the NCR transfers the axis information to HMI requested with SB1_GET_ACHSINFO_KUC.

	Remark
Direction	HMI <- NCR
Identification	SB0_AUFTRAG_KUC SB1_ACHSINFO_KUC
Length	138
Data	*achsinfo_r* (RECORD)
Meaning	The data structure *achsinfo_r* includes many values of the axis-specific machine constants and some that result from other machine constants. A detailed description follows.
Effect	The HMI may display the transferred information or may use it internally.
Note	-

Structure of *achsinfo_r*:

Offset	**Name**	**Type**	**Meaning**
0	*kenn_c*	CHAR	Identifying letter of the requested axis or–1, if not configured
1	*applachsidx_uc*	UCHAR	Number [i] of the axis of MK_APPLACHSIDX. The number is the index for all axis-specific fields.
2	*achsflags_us*	USHORT	Bit-coded axis flags 0x0001 Rotation axis 0x0002 Axis without limit switch 0x0004 Axis is a spindle 0x0008 Axis is a measuring axis 0x0010 Modulo 360° axis 0x0020 with Mod360°-axis, the shortest travel is made 0x0040 Axis is a gantry axis 0x0080 Axis is a handwheel 0x8000 Axis is a synchronous axis
4	*faktor_v_d*	DOUBLE	Conversion factor for velocities [units/min] -> [units/GIT]
12	*faktor_b_d*	DOUBLE	Conversion factor for acceleration [units/s^2] -> [units/GIT^2]
20	*deltat_f*	FLOAT	Coarse interpolation cycle (GIT) in [ms] MK_DELTAT
24	*fit_pro_git_ul*	ULONG	Number of fine interpolation cycles per GIT MK_FIT_PRO_GIT
28	*impulse_f*	FLOAT	MK_IMPULSE[i]
32	*weg_f*	FLOAT	MK_IMPULSE[i]
36	*massstab_f*	FLOAT	MK_MASSSTAB[i]
40	*grundoffset_f*	FLOAT	MK_GRUNDOFFSET[i]
44	*spindelumkehr_f*	FLOAT	MK_SPINDELUMKEHRSPIEL[i]
48	*synchronabweichung_f*	FLOAT	MK_SYNCHRONABWEICHUNG[i]

Offset	Name	Type	Meaning
52	*sw_ends_minus_f*	FLOAT	MK_SW_ENDS_MINUS[i]
56	*sw_ends_plus_f*	FLOAT	MK_SW_ENDS_PLUS[i]
60	*schleppgenauhalt_f*	FLOAT	MK_SCHLEPPGENAUHALT[i]
64	*schleppabstand_us*	USHORT	MK_SCHLEPPABSTAND[i]
66	*schleppcnt_uc*	UCHAR	MK_SCHLEPPZAEHLER[i]
67	*regler_mode_uc*	UCHAR	MK_REGLER_MODE[i]
68	*kp_f*	FLOAT	MK_KP[i]
72	*kf_f*	FLOAT	MK_KF[i]
76	*tv_f*	FLOAT	MK_KB[i]
80	*tn_f*	FLOAT	MK_TV[i]
84	*t2_f*	FLOAT	MK_TN[i]
88	*achseingaenge_us*	USHORT	MK_ACHSEINGAENGE[i]
90	*reftyp_uc*	UCHAR	MK_REF_TYP[i]
91	*richtungundfolge_c*	CHAR	MK_REF_RICHTUNG_UND_FOLGE[i]
92	*refvmax1_f*	FLOAT	MK_REF_VMAX1[i]
96	*refbmax1_f*	FLOAT	MK_REF_BMAX1[i]
100	*refvmax2_f*	FLOAT	MK_REF_VMAX2[i]
104	*refbmax2_f*	FLOAT	MK_REF_BMAX2[i]
108	*modvmax_f*	FLOAT	MK_MODVMAX[i]
112	*vmax_f*	FLOAT	MK_VMAX[i]
116	*beschl_f*	FLOAT	MK_BESCHL[i]
120	*brems_f*	FLOAT	MK_BREMS[i]
124	*t_beschl_f*	FLOAT	MK_T_BESCHL[i]
128	*einheit_f*	FLOAT	Resolution of an input unit [mm] or [degree] MK_METRISCH, MK_CONST_REL_MM, MK_CONST_REL_INCH
132	*kb_f*	FLOAT	MK_KB[i]
136	*phyachsidx_uc*	UCHAR	Physical axis number of the master axis
137	*physlaveidx_uc*	UCHAR	Physical axis number of the slave axis

6.2.12 Online change of the control parameters of an axis

The following message is provided for axis setting programs and diagnosis programs. The message makes a change of the control parameters of an axis with analog interface possible in the program in progress.

	Remark
Direction	HMI -> NCR
Identification	SB0_AUFTRAG_KUC SB1_CHANGE_REGPARAM_KUC
Length	28
Data	*checksum_uc* (UCHAR) *achsnr_uc* (UCHAR) *enable_us* (USHORT) *kp_f* (FLOAT) *kf_f* (FLOAT) *tv_f* (FLOAT) *tn_f* (FLOAT) *reglermode_ul* (ULONG) *kb_f* (FLOAT)
Meaning	For safety reasons, also an additional checksum is transferred, in order to avoid an inadvertent change, e.g. in case of a system crash. The value of *checksum_uc* is to be calculated in such a way that the sum of the values of the user data load in the message is 0. The number of the application axis, the control setting of which is to be changed is to be input in *achsnr_uc*. An enable bit exists in *enable_us* for each changeable control parameter. If several parameters are to be changed at a time, the respective enabling is to be connected by OR. *kp_f* receives the new value of the proportional gain (MK_KP), if bit 0 is set in *enable_us*. *kf_f* receives the new value of the velocity pre-control (MK_KF), if bit 1 is set in *enable_us*. *tv_f* includes the new derivative time (MK_TV), if bit 2 is set in *enable_us*. *tn_f* includes the new integrator time (MK_TN), if bit 3 is set in *enable_us*. *reglermode_ul* includes the new controller mode (MK_REGLER_MODE), if bit 4 is set in *enable_us*. *kb_f* includes the new value for the acceleration pre-control (MK_KB), if bit 5 is set in *enable_us*.
Effect	The NC controller accepts the changed control parameters in the machine constants in the main memory. The parameters, however, become only effective if at least for one coarse interpolation cycle, no order is to be transferred to the axis computer, e.g. the retention time between two traversing blocks. The changed data are not stored

	automatically in the flash PROM. A subsequent machine constant request transfers the changed control parameters.
Note	-

6.2.13 Start of an NC program with acknowledgment message

The following message is used for the direct start of a NC program without making use of the virtual keyboard and the PLC. At the end of the program, the NCR transfers an acknowledgment message to the HMI.

	Remark
Direction	HMI -> NCR
Identification	SB0_AUFTRAG_KUC SB1_PROGRAMM_START_KUC
Length	8
Data	*prognr_us* (USHORT) *satznr_us* (USHORT) *kanal_uc* (UCHAR) *errlevel_c* (CHAR) *startmode_uc* (UCHAR)
Meaning	The meaning of *startmode_uc* corresponds to the start button of the virtual keyboard (*dpr_PR->tast_r.start_auc[0]*). Unlike in the case of the start button, in addition the value 8 is allowed as type of the program to be started in the most significant nibble - (bit 4-7). This corresponds to the start of the NC program entered in P512, P513 with previous inputs of program number (*prognr_us*) and block number (*satz_us*) in P512, P513. Program number (*prognr_us*) and block number (*satz_us*) are only relevant if the type of program to be started is 8. The channel number (*kanal_uc*) is presently not used and is to be initialized to 0. With *errlevel_c,* selective error messages can be suppressed during the start that are normally signaled if the start is not possible. The error is only signaled if the assigned level is higher than the value of *errlevel_c*. In case of 0, all errors are signaled, in case of 5 none of them.
Effect	The NC controller starts the required NC program if all conditions for the start of a program are met. Since the start is not made via the PLC, the PLC is able to avoid a program start only by setting of the Stop button. Upon program implementation, the NCR transfers an acknowledgment message with SB1_PROGRAMM_ENDE_KUC. A positive acknowledgment message is transferred if the program was completed with M30 and a negative message if the start of the program was not possible or if it was aborted prematurely.
Note	If several messages are transferred one after the other with SB1_PROGRAMM_START_KUC without expecting the acknowledgment of the first

	one, only one acknowledgment message is transferred at the end of the first program. The following start attempts are ignored until the end of the first program.

Level	Error message
1	Stop signal is not reset, program start is not possible
2	Controller is not ready, program start is not possible
3	No program is active, start of next block is not possible
4	Emerg. stop signal is not reset, program start is not possible
5	Invalid start mode, program start is not possible

6.2.14 Acknowledgment message at the program end

The following message is transferred by the NCR after the implementation of an NC program started with SB1_PROGRAMM_START_KUC:

	Remark
Direction	HMI <- NCR
Identification	SB0_AUFTRAG_KUC SB1_PROGRAMM_ENDE_KUC
Length	2
Data	*quittung_s* (SHORT)
Meaning	In *quittung_s*, the NCR signals if the program was completed, if it could not be started or if it was aborted prematurely.
Effect	The HMI is enabled to start the next program or the next single function.
Note	-

quittung_s	**Meaning**
-1	No program started, start mode was 0
0	Program was completed with M30
1	Stop signal is not reset, program start is not possible
2	Controller is not ready, program start is not possible
3	No program is active, start of next block is not possible
4	Emerg. stop signal is not reset, program start is not possible
5	Invalid start mode, program start is not possible
-128	Program was started and aborted prematurely

6.2.15 Abort of a NC program

With this message, the HMI can abort the program in progress or the single function in progress:

	Remark
Direction	HMI -> NCR
Identification	SB0_AUFTRAG_KUC SB1_PROGRAMM_ABBRUCH_KUC
Length	0
Data	-
Meaning	The message has no user data load.
Effect	The program or the single function currently in progress is aborted and the respective acknowledgment message is generated, if necessary.
Note	-

6.2.16 Request of a program name

With the following message, the HMI can prompt the symbolic name of a DIN program by means of the program number.

	Remark
Direction	HMI -> NCR
Identification	SB0_AUFTRAG_KUC SB1_GET_PROGNAME_KUC
Length	4
Data	*prognr_ul* (ULONG)
Meaning	In the message, the program number (*prog_ul*) is specified to which the symbolic name is to be transferred.
Effect	The NCR transfers the requested information with SB1_PROGNAME_KUC. If no symbol exists for the specified program number, the program number is returned as string. This message is only relevant if symbolic program names are used during NC programming.
Note	-

6.2.17 Transfer of a program name

With the following message, the NCR transfers to the HMI the program name required with the SB1_GET_PROGNAME_KUC.

	Remark
Direction	HMI <- NCR
Identification	SB0_AUFTRAG_KUC SB1_PROGNAME_KUC
Length	4 + n
Data	*prognr_ul* (ULONG) *progname_ac[n]* (CHAR)
Meaning	The message includes the symbolic (*progname_ac*) name for the specified program number (*prognr_ul*).
Effect	The HMI can display the transferred information or use it internally. This message is only relevant if symbolic program names are used during the NC programming.
Note	-

6.3 Display commands

Display commands are transferred by the NCR to the HMI for visualization. They are triggered by the G-functions G252 and G253 in the NC-program.

Identification:

Name	Meaning
SB0_DISPLAY_KUC	

Messages:

Name	Meaning
SB1_GFKT_MELDUNG_KUC	Display of a message by means of G-function
SB1_GFKT_MELDUNGXY_KUC	Display of a message by means of G-function in a free position
SB1_GFKT_EINGABEXY_KUC	Input of a value by means of G-function

6.3.1 Display of a message by means of G-function

In the NC program, the function G253 can be used to output a message on the control panel. The NCR transfers the following message if G253 is programmed without the addresses X, Y, Z and E and if no text number is specified with F.

	Remark
Direction	HMI <- NCR
Identification	SB0_DISPLAY_KUC SB1_GFKT_MELDUNG_KUC
Length	n
Data	*string_ac[n]* (CHAR)
Meaning	This message is used to make a formatted output of a state message without indication of output column (X), output line (Y) and attributes (Z). The optional arguments of G253 (A, B, C) are formatted by the controller already in the output string. The zero-terminated output string of the G253 is included in *string_ac*. The message length n corresponds to the string length including the terminating zero.
Effect	The transferring string can be displayed by the HMI in a predefined position, e.g. in the state line.
Note	The function G253 Exxx does not generate a display message but an error message with SB0_EXCEPTION.

6.3.2 Display of a message by means of G-function in a free position

In the NC program, the function G253 can be used to output a message on the control panel. The NCR transfers the following message if the G253 is programmed with the addresses X, Y or Z or if a text number is indicated under F and no E is programmed.

	Remark
Direction	HMI <- NCR
Identification	SB0_DISPLAY_KUC SB1_GFKT_MELDUNGXY_KUC
Length	n+36
Data	*curx_s* (SHORT) *cury_s* (SHORT) *attrib_s* (SHORT) *pad_s* (SHORT) *textidx_ul* (ULONG) *werta_d* (DOUBLE) *wertb_d* (DOUBLE) *wertc_d* (DOUBLE) *format_ac[n]* (CHAR)
Meaning	This message is used to make a formatted output of a state message with indication of an output column (X), output line (Y), attributes (Z) and test number (F). The formatting of the output string is made in the HMI. *curx_s*, *cury_s*, *attrib_s* include output columns (X), output lines (Y) and display attributes (Z), if programmed in G253 or –1 if not programmed. The effect of these values is not predefined and can be determined by the HMI. *pad_s* is a filler and is, therefore, not defined. *textidx_ul* includes the text number programmed in G253 under F if programmed or 0 if the format string is directly programmed. *werta_d*, *wertb_d* and *wertc_d* include the optional parameters of the G253. The evaluation is to be made in compliance with the indications in the format string. *format_ac* includes the zero-terminated format string. The string is only valid if no text number is programmed *(textidx_ul*=0). The length is specified including the terminating zero. The format string is not checked by the NCR. Only the formats with DOUBLE arguments are permitted. This is to be checked by the HMI during evaluation of the format string in order to avoid error messages.
Effect	The transferred data are to be formatted by the HMI in an output string and are to be displayed as per curx_s, cury_s and attrib_s. The meaning of line, column, attribute and text number can be determined by the HMI in an application-specific way.
Note	The function G253 Exxx does not generate a display message but an error message with SB0_EXCEPTION.

6.3.3 Input of a value by means of G-function

In the NC program, the function G252 can be used to call an input dialog and to prompt the operator to input a value. The NCR transfers the following message if it meets the G252 during the interpretation of the NC program:

	Remark
Direction	HMI <- NCR
Identification	SB0_DISPLAY_KUC SB1_GFKT_EINGABEXY_KUC
Length	n+36
Data	*curx_s* (SHORT) *cury_s* (SHORT) *attrib_s* (SHORT) *idx_s* (SHORT) *textidx_ul* (ULONG) *min_d* (DOUBLE) *max_d* (DOUBLE) *default_d* (DOUBLE) *format_ac[n]* (CHAR)
Meaning	This message is used for a formatted input of a value via the dialog box. The formatting of the input string is made by the HMI. *curx_s*, *cury_s*, *attrib_s* include column (X), line (Y) and text attributes (Z) of the dialog box or –1 if not programmed. The effect of these values is not predefined and can be determined by the HMI. *idx_s* includes the index (C) of the target parameter in the parameter field, the value of which is to be input by the operator. *textidx_ul* includes the text number programmed in G252 under F if programmed, or 0 if the format string is directly programmed. *min_d*, *max_d* include the permitted range limits for the input function. These limits correspond to the addresses A and B in the G252. *default_d* includes the initial value of the input value. This value n corresponds to the value of the target parameter during interpretation of the G252. *format_ac* includes the zero-terminated format string. The string is only valid if no text number is programmed *(textidx_ul*=0). The length is specified including the terminating zero. The format string is not checked by the NCR. Only the formats with DOUBLE arguments are permitted. This is to be checked by the HMI during evaluation of the format string in order to avoid error messages
Effect	The transferred data are to be converted by the HMI into a dialog box and are to be displayed as per *curx_s*, *cury_s* and *attrib_s*. The meaning of line, column, attribute and text number can be determined by the HMI in an application-specific way.

	Upon successful completion of the input, the HMI is to write the input value into the target parameter (SB0_AUFTRAG_KUC, SB1_PFELD_LIST_KUC) and is to inform the NC computer about the end of input (SB0_AUFTRAG_KUC, SB1_EINGABE_ENDE_KUC), see section 6.2.7.
Note	-

6.4 Error messages

In this group, all messages for error messages and error acknowledgment are combined. Error messages are always transferred from the NCR to the HMI and error acknowledgments are always transferred from HMI to NCR.

Identification:

Name	Meaning
SB0_EXCEPTION_KUC	

Messages:

Name	Bedeutung
SB1_FEHLERNUMMER_KUC	Signaling of errors and warnings
SB1_FEHLERQUITTUNG_KUC	Acknowledgment of an error
SB1_FEHLERQUITTUNG_ALLE_KUC	Acknowledging of all errors not reset

6.4.1 Signaling of errors and warnings

The NCR signals errors and warnings by means of the following message.

	Remark
Direction	HMI <- NCR
Identification	SB0_EXCEPTION_KUC SB1_FEHLERNUMMER_KUC
Length	116
Data	*task_uc* (UCHAR) *klasse_uc* (UCHAR) *nummer_s* (SHORT) *format_ac[32]* (CHAR) *data_ac[80]* (CHAR)
Meaning	Error signals are clearly identified by error-signaling tasks (*task_uc*) and error numbers (*nummer_s*). By means of these two data, the HMI is able to assign the correct error text. The scope of supply of the NCR firmware includes a file with all error texts. A description of the error text file is included in the annex.
Effect	The HMI assigns the respective error text, if available, formats the additional information and displays the error. Upon acknowledgment of the error by the operator, a respective acknowledgment message is to be transferred to the NCR with SB1_FEHLERQUITTUNG_KUC or SB1_FEHLERQUITTUNG_ALLE_KUC.
Note	-

Meaning:

The controller distinguishes five error classes (*klasse_uc*). Errors of class 2-4 are to be acknowledged.

Class	Meaning	Effect
1	Locally minor	Only indication, no effect Example: unknown machine constant identifier
2	Locally major	Abort of the local command. Effect is limited to the error-signaling task and does, therefore, not generate the automatic abort of the traversing command. Example: memory capacity is out in case of NC program transfer
3	Globally minor	If MK_FEHLERRESTART=0, effects as in class 4, otherwise interruption of all commands, the NCR remains ready to continue the commands after the acknowledgment. A definite abort is only made by "Stop". Continuation with the respective Start. Example: position lag or PLC signals emergency stop
4	Globally major	Immediate abort of all commands, continuation impossible. New start is possible after the acknowledgment. Example: unknown G-function
5	Fatal	Immediate abort of all commands. New start is possible only after Reset of the controller. Presently, no respective errors are defined.

Most of the error messages include additional information that describe the circumstances of the error in detail, for instance: program number, block number etc. The additional information is described by a printf-compatible format string (*format_ac*) and the respective data (*data_ac*). The HMI is to format this information to a string for a display of the error.

6.4.2 Acknowledgment of an error

Errors of classes 2-4 are to be acknowledged by the HMI. The following message is used for the acknowledgment of a single error:

	Remark
Direction	HMI -> NCR
Identification	SB0_EXCEPTION_KUC SB1_FEHLERQUITTUNG_KUC
Length	2
Data	*task_uc* (UCHAR) *klasse_uc* (UCHAR)
Meaning	In case of single acknowledgment, the errors are to be acknowledged by the HMI in the order in which they arrived at the HMI. The acknowledgment message includes a copy of task_uc and klasse_uc of the error message, in order to be able to transfer the message within the NCR to the error-signaling task.
Effect	Upon acknowledgment of all errors to be reset, the NC controller exits the error state and is ready for the continuation or a new start of a task.
Note	In case of error acknowledgments still expected for some time, the controller is enabled to automatically exit the error state to avoid a remaining in the error state. A self-acknowledgment within the controller is presently made only in case of standalone controllers (firmware is installed in the controller and is not loaded during booting).

6.4.3 Acknowledgment of errors not reset

Errors of classes 2-4 are to be acknowledged by the HMI. The following message is used for the acknowledgment at a time of all errors not reset.

	Remark
Direction	HMI -> NCR
Identification	SB0_EXCEPTION_KUC SB1_FEHLERQUITTUNG_ALLE_KUC
Length	0
Data	-
Meaning	The message is a simplified form of the error acknowledgment. Since the acknowledgment is made asynchronously, there is the risk that error messages still not reset and not displayed are also acknowledged.
Effect	The NC controller exits the error state and is ready for continuation or new start of the task.
Note	In case of error acknowledgments still expected for some time, the controller is enabled to automatically exit the error state to avoid a remaining in the error state. A self-acknowledgment within the controller is presently made only in case of standalone controllers (firmware is installed in the controller and is not loaded during booting).

6.5 Debug messages

The NCR has feature-rich trace options. Internally, a 32KByte large range is available for recording and buffering of traces with 128 bytes each per trance input. The transfer of the recorded trace messages is made asynchronously to the recording.

Debug messages are used for the activation and transfer of trace recordings in the controller.

Identification:

Name	Bedeutung
SB0_DEBUG_KUC	

Messages:

Name	Bedeutung
SB1_TRACE_FLAGS_KUC	General activating/deactivating of trace recording and transfer
SB1_TRACE_SELECTIONS_KUC	Transfer of trace enablings to the NCR
SB1_TRACE_MELDUNG_KUC	Transfer of trace messages to the HMI
SB1_TRACE_ACHSDATEN_KUC	Transfer of axis data traces to the HMI
SB1_ACHSAUFZEICHUNG_KUC	Activating/deactivating of an axis recording
SB1_SPRUNGANTWORT_KUC	Recording of step response of an axis

6.5.1 General activating/deactivating of trace recording and transfer

With the following message, the general enabling for the recording and transfer of traces in the NCR is activated or deactivated:

	Remark
Direction	HMI -> NCR
Identification	SB0_DEBUG_KUC SB1_TRACE_FLAGS_KUC
Length	6
Data	*change_freigabe_uc* (UCHAR) *freigabe_uc* (UCHAR) *change_overwrite_uc* (UCHAR) *overwrite_uc* (UCHAR) *change_transfer_uc* (UCHAR) *transfer_uc* (UCHAR)
Meaning	*freigabe_uc* gets the general enabling for trace recording. Traces are only recorded if the enabling is not equal to 0. The value of *freigabe_uc* is only accepted if *change_freigabe_uc* is not equal to 0. *transfer_uc* gets the general enabling for the transfer of traces to the HMI. Recorded traces are only transferred if the transfer enabling is not equal to 0. The value of *transfer_uc* is only accepted if *change_transfer_uc* is not equal to 0. With *overwrite_uc* it is defined whether trace recording continues in case of an overflow of the trace buffer (overwrite_uc=1) or whether it is to be deactivated. The value of *overwrite_uc* is only accepted if *change_overwrite_uc* is not equal to 0.
Effect	The message influences the general trace enabling flags of the controllers, as described above.
Note	-

6.5.2 Transfer of trace enablings to the NCR

In the NCR, each trace can be activated separately. 256 trace enablings are available to that end. The following message is used to describe the enabling:

	Remark
Direction	HMI -> NCR
Identification	SB0_DEBUG_KUC SB1_TRACE_SELECTIONS_KUC
Length	256
Data	*freigaben_auc[256]* (UCHAR)
Meaning	Trace numbers between 0 and 255 are assigned to the single traces in the NCR. Each number is assigned to a byte in *freigaben_auc*. A value of 1 in the byte activates the recording and a value of 0 deactivates a recording. Both the NCR and the PLC are able to generate trace messages. The trace numbers 0 to 100 and 200 to 255 are reserved for the NCR, all other numbers are available to the PLC. A list of trace numbers defined in the NCR is included in the annex.
Effect	The message influences all trace enabling flags of the controller, as described above.
Note	-

6.5.3 Transfer of trace messages to the HMI

The following message is transferred by the NCR in order to transfer trace messages to the HMI:

	Remark
Direction	HMI <- NCR
Identification	SB0_DEBUG_KUC SB1_TRACE_MELDUNG_KUC
Length	128n
Data	*trace_ar[n]* (RECORD) *format_ac[48]* (CHAR) *daten_ac[80]* (CHAR)
Meaning	A trace message includes up to 4 trace messages (n=1-4). Each trace message is composed of a zero-terminated format string (*format_ac*) and of 80 byte for optional data (*daten_ac*). Format string and data a designed in a printf-compatible way. As an alternative, a trace message can also include a zero-terminated string without optional data, of up to 127 characters. Therefore, zero-termination at the end of the format string must not be forced by the HMI.
Effect	The HMI is able to format the trace messages with printf or similar, record them in a file and display them immediately.
Note	-

6.5.4 Transfer of axis data traces to the HMI

As an alternative to the trace messages, the NC controller can also record axis positions or axis velocities and transfer them asynchronously to the HMI. The transfer of the recorded axis data is made with the following message:

	Remark
Direction	HMI <- NCR
Identification	SB0_DEBUG_KUC SB1_TRACE_ACHSDATEN_KUC
Length	128n
Data	*achsdaten_ar[n]* (RECORD)
Meaning	The axis data recording is activated or deactivated with the G-functions G250 and G251. No trace messages are possible during axis data recording since for buffering, the trace buffer is used. As in the case of trace transfer, packets of 128 byte are transferred also with the axis data. A message includes max. 4 of such packets (n=1-4). Each packet includes a number of axis data inputs. The input format is variable and is determined by the functions G250 and G251. See also sections 6.5.5 and 6.5.6.
Effect	The transferred axis data can be recorded by the HMI or can be further processed.
Note	-

6.5.5 Activating/deactivating of an axis recording

As an alternative to the input of a G250, the axis data recording can also be started and stopped by the following message:

	Remark
Direction	HMI -> NCR
Identification	SB0_DEBUG_KUC SB1_ACHSAUFZEICHNUNG_KUC
Length	36
Data	*achsnr_auc[12]* (UCHAR) *mode_auc[12]* (UCHAR) *typ_us* (USHORT) *reserved_us* (USHORT) *param_d* (DOUBLE)
Meaning	The data in the message correspond to the parameters in the G250. *achsnr_auc* includes the axis numbers of the axes to be recorded. Max. 12 axes can be recorded at the same time. In case of less than 12 axes, the table end is identified by 0xff. *reserved_us* is presently not used and should be set to 0. *param_d* includes a recorder type-dependent parameter.
Effect	If axes were specified (*achsnr_auc[0]* != 0), the axis recording is activated with the specified parameters. Otherwise, the axis recording is deactivated. Both is only possible if the controller is ready for operation and if no program and no single function is active. Recording type 0 records velocity, position lag and control variable each as 16-bit integer data type in increments. The recording is made in the following order: command velocity, actual velocity, position lag and control variable. The command velocity is determined on the basis of the difference of the last two target positions, and the actual velocity on the basis of the last two actual positions. The position lag is the difference between target position and actual position. Control variable is the speed presetting at the drive amplifiers, and a value of 32767 corresponds to a voltage of 10V at the command value output. Recording type 1 records the target position and the actual position of the axes as 32-bit integer in increments. The order is target position then actual position. The recorded values are generated by the same fine interpolation cycle. Recording type 2 records only the actual position of the axes as 32-bit integer data type in increments. Recording type 3 always records the actual position of the first 6 physical axes as 32-bit integer and the first 4 analog measured values as 16-bit integer. The axes are also recorded if they are not configured, i.e. the data block length is always 32 bytes. Also in this case, the positions of the axes are recorded in increments. The recording of the analog inputs is not yet implemented, therefore, 0 is always supplied as measured

	value. With this recording type, the distance in [mm] between two measured values on the path is to be specified under *param_d*. The axes that are to be considered for the determination of the path length, are to be specified under *achsnr_auc*. The measured values and the position are linearly interpolated between the fine interpolation cycles in order to meet the requested distance. In case of *param_d* = 0 a data block is recorded per fine interpolation cycle. Recording type 4 records the actual position of the first 2 physical axes as 32-bit integer and the first 4 analog measured values as 16-bit integer. Data block length is always 16 byte. The position values are recorded in increments and 32767 corresponds to 10V at the analog input. The recording of analog measured values is possible only under determined conditions (EC-CPU05, EC-ADC01, MK_KUNDE, ...). 0 is supplied as measured value if these conditions are not met. Recording type 5 records the actual position of the first three physical axes as 32-bit integer and the first two analog measured values as 16-bit integer. For the recording of the analog inputs, a EC-CPU05 and a special variant of the EC-ADC01 is required. The data block length is always 16 bytes. The position values are recorded in increments and 32767 corresponds to 10V at the analog input. Recording type 6 records velocity, position lag and control variable as 32-bit integer data type each in increments. The recording is made in the following order: command velocity, actual velocity, position lag and control variable. The command velocity is determined from the difference of the last two target positions and the actual velocity of the last two actual positions. The position lag is the difference between target position and actual position. Control variable is the speed presetting at the drive amplifiers, and a value of 32767 corresponds to a voltage of 10V at the command value output. Recording type 7 records command and actual position in increments as 32-bit integer, as well as control word, status word, traversing state and a monitoring signal as 16-bit integer. This recording type was implemented especially for the recording of axes at the CAN bus interface. Presently, the monitor signal is available only with Lenze and can be configured there by means of SDO command. Recording type 8 also records velocities, position lag and control variable each as 32-bit integer data type in increments. In case of a synchronous axis, however, two data blocks are recorded, one for the master axis and one for the slave axis. In case of param_d, the number of fine interpolation cycles can be specified with all recording types except type 3, for instance to make the recoding of measured values only every 10ms instead of all 2ms.
Note	The recording of axis data is always made in the physical order of the axes, even if the axis numbers were specified in a different order. In case of doubt, the data are to be put in the required order prior to evaluation. For the determination of the physical axis order, SB0_AUFTRAG_KUC/SB1_GET_ACHSINFO_KUC can be used.

mode_auc selects the recording mode of synchronous axes:

mode_auc	Meaning
0	Recording of master axis only
1	Recording of slave axis only
2	Recording of master and slave axis

typ_us selects one of the following recording types:

typ_us	Meaning
0	V_{com}, V_{act}, position lag and control variable as SHORT
1	Target position and actual position as LONG
2	Actual position as LONG
3	Actual position of the first 6 physical axes as LONG and of the first 4 analog inputs as SHORT
4	Actual position of the first 2 physical axes as LONG and of the first 4 analog inputs as SHORT
5	Actual position of the first 3 physical axes as LONG and of the first 2 analog inputs as SHORT
6	V_{com}, V_{act}, position lag and control variable as LONG
7	Target position and actual position as LONG, control word and status word as USHORT, traversing state and monitor value as SHORT
8	As 6, but with separate data block for slave axes

6.5.6 Recording of a step response

As an alternative for the input of a G251, the recording of the step response of an axis is also possible by means of the following message:

	Remark
Direction	HMI -> NCR
Identification	SB0_DEBUG_KUC SB1_SPRUNGANTWORT_KUC
Length	28
Data	*achsnr_uc* (UCHAR) *mode_uc* (UCHAR) *count_us* (USHORT) *vsoll_d* (DOUBLE) *zeit_d* (DOUBLE) *bzeit_d* (DOUBLE)
Meaning	The data in the message correspond to the parameters of G251. *achsnr_uc* includes the axis number of the axis, the step response of which is to be recorded. *count_us* is the number of steps to be recorded during this command. *vsoll_d* includes the command velocity of the command step in the velocity units normally used for the selected axis, e.g. [mm/min] or [1/min]. *zeit_d* includes the duration of the step in seconds. *bzeit_d* includes the duration of acceleration in seconds.
Effect	The function oscillates between the specified axis *count_us* with the specified velocity, the duration of acceleration *bzeit_d* and the duration of the step *zeit_d*. At the end of the movement, the axis is again at its starting point. The traversing travel corresponds to *zeit_d* * *vsoll_d*. The interval between the steps is *zeit_d* - *bzeit_d*. During the output of the step function, command velocity, actual velocity, position lag and control variable are recorded in increments and are transferred asynchronously to the PC. As a function of *mode_uc*, the data are recorded either as 16 or as 32 bit integer. In case of a synchronous axis, the actual velocity of master and slave axis, the synchronous distance between the axes and the command velocity are recorded with mode 2. The function can only be implemented if the controller is ready for operation and presently no program and no single function is active.
Note	-

mode_uc selects the recording mode:

mode_uc	Meaning
0	In case of synchronous axis, recording of master axis only
1	In case of synchronous axis, recording of slave axis only
2	In case of synchronous axis, recording of master and slave axis
+128	Recording with 32 bit integer data type

6.6 Application specific commands

This group includes messages that are provided for a direct message communication between HMI and PLC.

Identification:

Name	Meaning
SB0_SPSAUFTRAG_KUC	

Messages:

Name	Meaning
-	The available control blocks and their meaning are application-specific and are determined by the PLC or HMI programmer.

6.6.1 Task from HMI to PLC or vice versa

The following message includes application-specific data.

	Remark
Direction	HMI -> NCR HMI <- NCR
Identification	SB0_EXCEPTION_KUC variabel
Length	variable
Data	variable
Meaning	The meaning of the transferred data is application-specific and is determined by the HMI or PLC programmer.
Effect	The effect is also application-specific and is determined by the HMI or PLC programmer.
Note	-

6.7 CANopen messages

These messages are used for communication with devices at the CAN busses of the controller. Presently, only functions are available for the reading and writing of objects by means of SDO communication to the CAN bus.

Identification:

Name	Bedeutung
SB0_CANOPEN_KUC	

Messages:

Name	Bedeutung
SB1_CAN1_READ_SDO_KUC	Reading of an object from a module at the CAN1 by means of SDO
SB1_CAN2_READ_SDO_KUC	Reading of an object from a module at the CAN2 by means of SDO
SB1_CAN1_WRITE_SDO_KUC	Writing of an object from a module at the CAN1 by means of SDO
SB1_CAN2_WRITE_SDO_KUC	Writing of an object from a module at the CAN1 by means of SDO

6.7.1 Reading of an object from a CAN module

The following message makes the reading of a single CANopen object possible from a module at one of the two CAN busses:

	Remark
Direction	HMI -> NCR request for the reading of an object HMI <- NCR message of the read object
Identification	SB0_CANOPEN_KUC SB1_CAN1_READ_SDO_KUC SB1_CAN2_READ_SDO_KUC
Length	8 in case of direction HMI -> NCR 8+n in case of direction HMI <- NCR
Data	*nodeid_uc* (UCHAR) *objectidx_us* (USHORT) *subidx_uc* (UCHAR) *len_l* (LONG) *data_auc[n]* (UCHAR) only in case of direction HMI <- NCR
Meaning	By means of this message, the HMI has reading access to the object directory of the devices at the CAN bus. The read object is transferred with the same message from the controller to the HMI. With each message, only one object can be read. The object to be read is defined by the following information: *nodeid_uc* Node number of the CANopen device *objektidx_us* Index of the object to be read *subidx_uc* Index of the sub-object to be read *len_l* Max. length of the data to be read in byte The controller response includes the following information: *nodeid_uc* Node number of the CANopen device *objektidx_us* Index of the read object *subidx_uc* Index of the read sub-object *len_l* Max. length of the read data in byte (max. 504) or < 0 in case of error *data_auc[]* Read data
Effect	The requested object is read by means of SDO command at the CAN bus and is returned with the same message to the HMI. In each case, the length is limited to 504 bytes. In case of error, *len_l* gets a negative value and *data_auc* is empty.
Note	Reading and writing of several CAN objects of unlimited length is possible by means of block transfer with SB1_CAN1_OBJECT_KUC or SB1_CAN2_OBJECT_KUC. For the transfer of several objects, block transfer can be clearly quicker than single transfer.

Effect:

len_l	Status	Meaning
>= 0	SDO_READY	Reading successful
-70	SDO_ERROR	Transfer aborted
-71	SDO_TIMEOUT	Timeout (module does not respond)
-72	SDO_ABBRUCH	Transfer aborted by the module

6.7.2 Writing of an object in a CAN module

The following message makes the writing of a single CANopen object possible in a module at one of the two CAN busses:

	Remark
Direction	HMI -> NCR request for the writing of an object HMI <- NCR acknowledgment after writing of the object
Identification	SB0_CANOPEN_KUC SB1_CAN1_WRITE_SDO_KUC SB1_CAN2_WRITE_SDO_KUC
Length	8+n HMI -> NCR 8 HMI <- NCR
Data	*nodeid_uc* (UCHAR) *objectidx_us* (USHORT) *subidx_uc* (UCHAR) *len_l* (LONG) *data_auc[n]* (UCHAR) only in case of direction HMI -> NCR
Meaning	By means of this message, the HMI has writing access to the object directory of the devices at the CAN bus. With each message, only one object can be written. The following parameters are transferred in the message: *nodeid_uc* Node number of the CANopen device *objektidx_us* Index of the object to be written *subidx_uc* Index of the sub-object to be written *len_l* Length of the data to be written in byte (max. 504) *data_auc[]* Data to be written The controller response includes the following information: *nodeid_uc* Node number of the CANopen device *objektidx_us* Index of the written object *subidx_uc* Index of the written sub-object *len_l* Acknowledgment: 0 in case of success, less than 0 in case of an error
Effect	The transferred object is written by means of SDO command at the CAN bus and acknowledgment is returned with the same message to the HMI. In case of error, *len_l*

	gets a negative value.
Note	Reading and writing of several CAN objects of unlimited length is possible by means of block transfer with SB1_CAN1_OBJECT_KUC or SB1_CAN2_OBJECT_KUC. For the transfer of several objects, block transfer can be clearly quicker than single transfer

Effect:

len_l	Status	Meaning
= 0	SDO_READY	Writing successful
-70	SDO_ERROR	Transfer aborted
-71	SDO_TIMEOUT	Timeout (module does not respond)
-72	SDO_ABBRUCH	Transfer aborted by the module

6.8 Codesys messages

These messages are used for communication of 3S-tools like Codesys, visualization and OPC servers with the PLC runtime systems in the controller.

Identification:

Name	Meaning
SB0_CODESYS_KUC	

Messages:

Name	Meaning
SB1_LZS_REQUEST_KUC	Service from the PC to the runtime system and message from the runtime system to the PC
SB1_LZS_ACKN_KUC	Acknowledgment from and to the runtime system

6.8.1 Service to and message from the runtime system

The following message is used for the transfer of a service to the runtime system and for messages from the runtime system.

	Remark
Direction	HMI -> NCR service to the runtime system HMI <- NCR message from the runtime system
Identification	SB0_CODESYS_KUC SB1_LZS_REQUEST_KUC
Length	Variable
Data	variable
Meaning	The meaning of the data to be transferred is defined by 3S. For further information please check the 3S-documentation regarding the runtime system.
Effect	Upon reception in the controller, messages of this type are directly transferred to the runtime system and are processed there. Normally, an acknowledgment is first made by the runtime system with SB1_LZS_ACKN_KUC and subsequently, a response is given with SB1_LZS_REQUEST_KUC. The PC makes an acknowledgment with SB1_LZS_ACKN_KUC that is directly passed by the controller to the runtime system.
Note	-

6.8.2 Acknowledgments from and to the runtime system

The following message is used for the transfer of an acknowledgment in the PLC runtime system.

	Remark
Direction	HMI <- NCR acknowledgment from the runtime system HMI -> NCR acknowledgment to the runtime system
Identification	SB0_CODESYS_KUC SB1_LZS_REQUEST_KUC
Length	0
Data	None
Meaning	The message does not include user data.
Effect	See section 6.8.1.
Note	-

6.9 DLL messages

These messages are used for the implementation of simple, mostly application-independent host services.

Identification:

Name	Meaning
SB0_DLL_SERVICE_KUC	

Nachrichten:

Name	Meaning
SB1_OPEN_FILE_KUC	Opening of file for reading and writing
SB1_CLOSE_FILE_KUC	Closing of file
SB1_READ_FILE_KUC	Reading of block from file
SB1_WRITE_FILE_KUC	Writing of block into file
SB1_DELETE_FILE_KUC	Deleting of file
SB1_FIRST_FILE_KUC SB1_FIRST_FILE_EX_KUC	Searching the first n files with a suited name pattern
SB1_NEXT_FILE_KUC SB1_NEXT_FILE_EX_KUC	Searching the next n files with a suited name pattern
SB1_DISKINFO_KUC	Getting occupation information about the PC disc

6.9.1 Opening of file

With the following message, the controller opens a file on the host PC for reading or writing in a directory enabled for that purpose.

	Remark
Direction	HMI <- NCR I/O service to the host HMI -> NCR I/O acknowledgment to the controller
Identification	SB0_DLL_SERVICE_KUC SB1_OPEN_FILE_KUC
Header	*handle_uc* *index_uc* *sb2_uc*
Length	4+n HMI <- NCR 4 HMI -> NCR
Data	HMI <- NCR: *mode_ul* (ULONG) *filename_ac[n]* (CHAR) HMI -> NCR: *filesize_ul* (ULONG)
Meaning	*handle_uc* is part of the message header and includes the file handle of the controller for the file to be opened. At each access to the file, both the respective handle and also the responses from the host are entered in the header. *index_uc* is part of the header and includes a consecutive number that is also to be returned with the response of the host. *filename_ac* includes the path and file name as zero-terminated string. *filesize_ul* in the response of the host includes the length of the opened file.
Effect	The host tries to open the file with the specified name in the enabled directory and in the required mode. If this operation is completed successfully, the file handle on the host side is stored, assigned in a table to the controller-side file handle. A file opened before under the same controller handle is closed before. The response message of the host includes an error code in *sb2_uc*, the controller-side file handle in *handle_uc*, the consecutive number in *index_uc* and the file length of the opened file in *filesize_ul*.
Note	-

Meaning:

sb2_uc is part of the header and includes one of the following error codes:

sb2_uc	Status	Meaning
0	DLL_IOERR_ZERO_KUC	Operation successful, no error
1	DLL_IOERR_NOFILE_KUC	File not found
2	DLL_IOERR_NOPATH_KUC	Path not found
3	DLL_IOERR_MFILE_KUC	Too much files opened
4	DLL_IOERR_ACCES_KUC	Access disabled
5	DLL_IOERR_NOPEN_KUC	File not opened
6	DLL_IOERR_OHTER_KUC	Not clearly specified error occurred
7	DLL_IOERR_NOMEM_KUC	Not enough memory capacity available
8	DLL_IOERR_HANDLE_KUC	Invalid file handle
9	DLL_IOERR_INVNAME_KUC	Invalid path or file name

mode_ul indicates whether the file is to be opened for reading or writing:

mode_ul	Status	Meaning
0	DLL_IOMODE_RDONLY_KUL	Open file for reading
1	DLL_IOMODE_WRONLY_KUL	Open file for writing

6.9.2 Closing of file

With the following message, one of the files is closed again that was opened by the controller on the host side.

	Remark
Direction	HMI <- NCR I/O service to the host HMI -> NCR I/O acknowledgment to the controller
Identification	SB0_DLL_SERVICE_KUC SB1_CLOSE_FILE_KUC
Header	*handle_uc* *index_uc* *sb2_uc*
Length	0
Data	none
Meaning	*handle_uc* is part of the message header and includes the file handle of the controller for the file to be closed. At each access to the file, both the respective handle and also the responses from the host are entered in the header. *index_uc* is part of the header and includes a consecutive number that is also to be returned with the response of the host. *sb2_uc* is part of the header and includes one of the following error codes, see section 6.9.1.
Effect	The host closes the file opened under the specified handle by means of the stored file handle on the host side. The response message of the host includes an error code in *sb2_uc*, the controller-side file handle in *handle_uc* and the consecutive number in *index_uc*.
Note	-

6.9.3 Reading of block from file

With the following message a block is read by the controller from a file opened on the host side.

	Remark
Direction	HMI <- NCR I/O service to the host HMI -> NCR I/O acknowledgment to the controller
Identification	SB0_DLL_SERVICE_KUC SB1_READ_FILE_KUC
Header	*handle_uc* *index_uc* *sb2_uc*
Length	4 HMI <- NCR 4+n HMI -> NCR
Data	iooffset_ul (ULONG) iodata_auc[n] (UCHAR) only in case of HMI -> NCR
Meaning	*handle_uc* is part of the message header and includes the file handle of the controller. *index_uc* is part of the header and includes the consecutive number. *sb2_uc* is part of the header and includes an error code, see section 6.9.1. *iooffset_ul* includes the offset in the file, the block is to be read starting with this offset. *Iodata_auc* includes the read block with a max. length of 508 bytes.
Effect	The host positions the file pointer of the file opened under the specified handle to the value specified in *iooffset_ul* and subsequently reads 508 bytes from the file. Should the rest up to the file end be less than 508 bytes, only this rest is read. The length specified in the header of the response message includes the number of the read bytes + 4. The response message of the host includes an error host in *sb2_uc*, the file handle in *handle_uc* and the consecutive number in *index_uc*.
Note	-

6.9.4 Writing of block into file

With the following message, a block is written by the controller into a file opened on the host side.

	Remark
Direction	HMI <- NCR I/O service to the host HMI -> NCR I/O acknowledgment to the controller
Identification	SB0_DLL_SERVICE_KUC SB1_WRITE_FILE_KUC
Header	*handle_uc* *index_uc* *sb2_uc*
Length	4 HMI <- NCR 4+n HMI -> NCR
Data	iooffset_ul (ULONG) iodata_auc[n] (UCHAR) only in case of HMI -> NCR
Meaning	*handle_uc* is part of the message header and includes the file handle of the controller. *index_uc* is part of the header and includes the consecutive number. *sb2_uc* is part of the header and includes an error code, see section 6.9.1. *iooffset_ul* includes the offset in the file, the block is to be written starting with this offset. *Iodata_auc* includes the block to be written with a max. length of 508 bytes.
Effect	The host positions the file pointer of the file opened under the specified handle to the value specified in *iooffset_ul* and subsequently writes the block from the message into the file with up to 508 bytes. The length specified in the header includes the number of the bytes to be written + 4. The response message of the host includes an error host in *sb2_uc*, the file handle in *handle_uc*, the consecutive number in *index_uc* and the file offset from which on the block was written in *iooffset_ul*.
Note	-

6.9.5 Deleting of file

With the following message, a file in the enabled directory is deleted on the host by the controller.

	Remark
Direction	HMI <- NCR I/O service to the host HMI -> NCR I/O acknowledgment to the controller
Identification	SB0_DLL_SERVICE_KUC SB1_DELETE_FILE_KUC
Header	*handle_uc* *index_uc* *sb2_uc*
Length	n HMI <- NCR 0 HMI -> NCR
Data	*filename_ac[n]* (CHAR) only in case of HMI <- NCR
Meaning	*handle_uc* is part of the message header and includes the file handle of the controller. *index_uc* is part of the header and includes the consecutive number. *sb2_uc* is part of the header and includes an error code, see section 6.9.1. *filename_ac* includes the path and the file names of the file to be deleted as zero-terminated string.
Effect	The host tries to delete the specified file and transfers a response to the controller. The response message includes an error code in *sb2_uc*, the file handle in *handle_uc* and the current number in *index_uc*.
Note	-

6.9.6 Reading of table of contents

With the following messages, the table of contents of the directory is read by the controller that is enabled on the host.

	Remark
Direction	HMI <- NCR I/O service to the host HMI -> NCR I/O acknowledgment to the controller
Identification	SB0_DLL_SERVICE_KUC SB1_FIRST_FILE_KUC SB1_NEXT_FILE_KUC
Header	*handle_uc* *index_uc* *sb2_uc*
Length	m HMI <- NCR 16n HMI -> NCR
Data	*pattern_ac[m]* (CHAR) only in case of HMI <- NCR *direntry_ar[n]* (RECORD) only in case of HMI -> NCR *filename_ac[16]* (CHAR) *filesize_ul* (ULONG)
Meaning	*handle_uc* is part of the message header and includes the file handle of the controller. *index_uc* is part of the header and includes the consecutive number. *sb2_uc* is part of the header and selects in case of direction HMI <- NCR whether 8+3-file names (*sb2_uc*=0) or long file names (*sb2_uc*=1) are to be read. In case of direction HMI -> NCR, *sb2_uc* includes an error code, see section 6.9.1. *pattern_ac* is transferred by the controller with SB1_FIRST_FILE_KUC and includes a DOS-compatible wildcard string. *direntry_ar* is transferred by the host and includes up to 32 directory inputs with file name and file size. *filename_ac* includes a file name in the 8+3-format. The file name is zero-terminated if it has less than 12 characters. *filesize_ul* includes the file size in bytes.
Effect	With SB1_FIRST_FILE_KUC, the controller calls the first n directory inputs from the host that suit the preset pattern. With SB1_NEXT_FILE_KUC, the controller subsequently calls the next n suited directory inputs with the same directory handle. The response of the host depends on *sb2_uc*: In case of *sb2_uc* = 0, max. 32 directory inputs are read with each call message in the format 8+3 and are re-transferred to the controller with SB1_FIRST_FILE_KUC or SB1_NEXT_FILE_KUC. The response message is accordingly shorter if less than 32 directory inputs are left. In case of *sb2_uc* = 1, the response is made with SB1_FIRST_FILE_EX_KUC or SB1_NEXT_FILE_EX_KUC. The host always reads as much directory inputs as

	capacity is available in the response message. The message is accordingly shorter if less inputs are left. The response format of the host is described in section 6.9.7. The response of the host always includes an error code in *sb2_uc*, the directory handle in *handle_uc* and the consecutive number in *index_uc*. If no more suited file is available, the error code DLL_IOERR_NOFILE_KUC is included in *sb2_uc*.
Note	-

6.9.7 Table of contents with long file names

With the following messages, the host transfers directory inputs to the controller that have long file names.

	Remark
Direction	HMI -> NCR acknowledgment to the controller
Identification	SB0_DLL_SERVICE_KUC SB1_FIRST_FILE_KUC SB1_NEXT_FILE_KUC
Header	*handle_uc* *index_uc* *sb2_uc*
Length	variable
Data	*direntry_ar[n]* (RECORD) *filename_ac[]* (CHAR) variable length! *filesize_ul* (ULONG) *timestamp_ul* (ULONG)
Meaning	*handle_uc* is part of the message header and includes the file handle of the controller. *index_uc* is part of the header and includes the consecutive number. *sb2_uc* is part of the header and includes an error code, see section 6.9.1. *direntry_ar* includes a variable number of directory inputs with file name, file size and time stamp. The number is limited by the max. user data length of 512 bytes. *filename_ac* includes a zero-terminated file name of variable length. The start of the following data is postponed accordingly. *filesize_ul* includes the file size in bytes. *timestamp_ul* includes the time stamp of the file in seconds since 1/1/1970, 0:00:00.
Effect	With these messages, the host transfers the directory inputs to the controller requested with SB1_FIRST_FILE_KUC and SB1_NEXT_FLIE_KUC, if *sb2_uc*=1 was included in the request message. See section 6.9.6. The host transfers as many directory inputs as capacity is available in the message. The message is accordingly shorter if less inputs are left. Each directory input has a variable length and includes the file name, the file size and the date of the last change.

	The response message of the host always includes an error code in *sb2_uc*, the directory handle in *handle_uc* and the consecutive number in *index_uc*. If no more suited file is available, the error code DLL_IOERR_NOFILE_KUC is included in *sb2_uc*.
Note	-

6.9.8 Reading of occupation information

With the following message, the controller reads the empty/occupied information of the enabled directory on the host.

	Remark
Direction	HMI <- NCR I/O service to host HMI -> NCR I/O acknowledgment to controller
Identification	SB0_DLL_SERVICE_KUC SB1_DISKINFO_KUC
Header	*handle_uc* *index_uc* *sb2_uc*
Length	0 HMI <- NCR 16 HMI -> NCR
Data	*blocksize_ul* (ULONG) only in case of HMI -> NCR *blockanz_ul* (ULONG) only in case of HMI -> NCR *blockused_ul* (ULONG) only in case of HMI -> NCR *blockfree_ul* (ULONG) only in case of HMI -> NCR
Meaning	*handle_uc* is part of the message header and is always 0 with this message. *index_uc* is part of the header and includes a consecutive number. *sb2_uc* is part of the header and includes an error code, see section 6.9.1. *blocksize_ul* includes the size of a logic block (management units) in bytes. *blockanz_ul* includes the entire number of blocks of the data carrier on which the enabled directory is stored. *blockused_ul* includes the number of occupied blocks of the data carrier on which the enabled directory is stored. *blockfree_ul* includes the number of free blocks of the data carrier on which the enabled directory is stored.
Effect	The host determines the desired data and transfers them with the response message to the controller. The message of the host includes an error code in *sb2_uc*, the *handle_uc* transferred by the controller and the consecutive number in *index_uc*.
Note	-

6.10 Message polling

The following message is used for the checking of the communication path if no transfer is available for a longer period. If the message is not accepted by the NCR, the timeout of the communication path is able to recognize it and to abort all activities.

	Remark
Direction	HMI -> NCR
Identification	SB0_POLL_KUC SB1_POLL_KUC
Length	0
Data	-
Meaning	The message does not include a user data load
Effect	The NC computer sets the acknowledgment count (dpr_PR->nc_r.qc_us) upon reception of the message as upon reception of each message. The message is not evaluated and there is no additional reaction.
Note	-

Annex A: Structure of the error text file

With the NC firmware, an error text file is transferred and on the basis of this file, the HMI is able to determine the respective error text out of the error-signaling task and the error number. The file is structured in such a way that it can be imported easily into the data base.

Each line includes three strings in inverted commas that are separated by a comma:

"2.1.4","Neg. position lag: mechanics do not meet the presetting ",""

The first string includes computer identification (2), error-signaling task (1) and error number (4). The three elements are separated from each other by a point.

The second string includes the assigned error text.

The third string is reserved for a description of the additional information and is presently always empty.

Computer identification: is to be set always to 2 in case of error messages from the NCR.

Error-signaling task: Identifies the transmitter of the error message.

Task	Module	Meaning
1	AXE	Axis computer
3	INT	Interpreter
4	KOP	Coupling
5	PLC	Runtime system of the sequence control
6	POS	Coarse interpolator
7	SPV	Memory management
9	ZST	Central controller
10	SPS	Sequence control
11	SYS	Operating system
12	CAN	CAN bus communication
13	UTI	Auxiliary functions
17	TPR	Communication path HMI <-> NCR (previously Transputer)
18	DRV	Error messages of CAN drives as per DS402
19	LNZ	Error messages of Lenze drives
20	NOV	Error messages of Novotron drives
21	SDL	Error messages of Seidel drives
22	IDR	Error messages of Indramat drives
23	ENC	Error messages of CAN Encodern as per DS406

Error no. : Provides for an unambiguous assignment of an error together with the other information.

Annex B: List of all trace numbers of the NCR

Trace-Nr.	Trace text deutsch	Trace text english
000		
001		
002		
003		
004		
005		
006		
007		
008		
009	Änderung von S- und T-Koordinatensystemen (POS)	Change of S- and T-coordinate systems (POS)
010	Positionsvorgabe und Positionsrückmeldung vom Achsrechner	Position input and acknowledgement from axis controller
011	Parameterübergabe vom Achsrechner an SERCOS-Antrieb (AXE)	Parameter transfer from axes control to SERCOS-drive (AXE)
012	Zustandswechsel bei Referenzpunktfahrt (AXE)	state change at referencing (AXE)
013	Zustandswechsel von Fahrstatus (AXE)	state change at drive status (AXE)
014	Versionsstring der identifizierten CAN-Antriebe (AXE)	Version string of the identified CAN drives (AXE)
015	Ausgabe von Sollwert und Korrekturwert bei Online-Messerkorrektur	Output of reference and correction value at online knife correction
016	Zustandsänderungen von Endschalter und Referenznocken (AXE)	Status change of end switch and reference cam (AXE)
017	Steuerwort und Sollwerte an SERCOS-Antrieb	Control word and reference values of SERCOS drive
018	Statuswort und Istwerte von SERCOS-Antrieb	Status word and actual values of SERCOS drive
019		
020	Tastenmeldung an integriertes MMI	Key signalling to integrated MMI
021	Laden und Speichern von Werkzeug-, Werkstück- und Achskorrekturen	load and save of tool-, workpiece and axes corrections
022	Kundenspezifischer Spindelhandler (POS)	Customer-specific spindle handler (POS)
023	Kundenspezifischer Spindelhandler (POS)	Customer-specific spindle handler (POS)
024	Kundenspezifischer Spindelhandler (POS)	Customer-specific spindle handler (POS)

025	Kundenspezifischer Spindelhandler (POS)	Customer-specific spindle handler (POS)
026	Kundenspezifischer Spindelhandler (POS)	Customer-specific spindle handler (POS)
027	Kundenspezifischer Spindelhandler (POS)	Customer-specific spindle handler (POS)
028	Neue Offsets im Interpreter-Koordinatensystem (INT)	New offsets in interpreter coordinate system (INT)
029	Neue Offsets im Interpolator-Koordinatensystem (INT)	New offsets in interpolator coordinate system (INT)
030	Look-Ahead für Spindel (INT)	Look-Ahead for spindle (INT)
031	Auftrag an Grobinterpolator (INT)	Job to coarse interpolator (INT)
032	Quittung von Grobinterpolator bei Zeitsynchronisation (INT)	Acknowledgement from coarse interpolator in the case of time synchronisation (INT)
033	Nachricht vom Grobinterpolator (INT)	Message from coarse interpolator (INT)
034	Nachricht aus Mailbox (INT)	Message from mailbox (INT)
035	Einzelsatz (INT)	Single block (INT)
036	Look-Ahead für Bahn (INT)	Look-Ahead for contour (INT)
037	Zielposition aus Verfahrauftrag an Grobinterpolator (INT)	Target position from travel job to coarse interpolator (INT)
038	Istpositionsmeldung vom Grobinterpolator (INT)	Actual position message from coarse interpolator (INT)
039	G10-Verwaltung (INT)	G10 management (INT)
040	P-Feld-Lesen (INT)	Read P field (INT)
041	P-Feld-Schreiben (INT)	Write P field (INT)
042	Anzeige von Bahngeschwindigkeitsbegrenzungen am Satzübergang (INT)	Display of contour speed limits at the block transition (INT)
043	Anzeige von Bahngeschwindigkeitsbegrenzungen im Satz (INT)	Display of contour speed limits in the block (INT)
044	Spline (INT)	Spline (INT)
045	Laden von Werkzeug- und Werkstückdaten (INT)	Load of tool and tool piece data
046	Anzeige von empfangen Nachrichten (KOP)	Display of received messages (KOP)
047	Anzeige von empfangen Quittungen (KOP)	Display of received handshake (KOP)
048	Anzeige von Kommunikationsfehlern bei der Linkübertragung (KOP)	Display of communication errors during link transfer (KOP)
049		
050	Nachricht von PLCOS über V24 (PLC)	Message from PLCOS via V24 (PLC)
051	Nachricht an PLCOS über V24 (PLC)	Message to PLCOS via V24 (PLC)
052	STX, ETX, ACK, NAK bei Nachrichtenübertragung (PLC)	STX, ETX, ACK, NAK in the case of message transfer (PLC)

053	Blockübertragung von PLCOS an PLC (PLC)	Block transfer from PLCOS to PLC (PLC)
054	P-Feld-Lesen (SPS)	Read P field (PLC)
055	P-Feld-Schreiben (SPS)	Write P field (PLC)
056	Bahngeschwindigkeit < 0.000001 (POS)	Contour speed < 0.000001 (POS)
057	Kanal-Zustandswechsel (POS)	Channel status change (POS)
058	Zustandswechsel im ne_modhander (pos)	Status change in ne_modhander (pos)
059	M-Fkt Timing (POS)	M fct timing (POS)
060	Prognr, Satznr, Logsatznr bei Open Satz (POS)	Prog No., block No., log block No. in the case of open block (POS)
061	Bahngeschwindigkeit, NC-Istposition und Schleppabstand von X,Y,Z (POS)	Actual contour speed, NC actual position and following error of X, Y, Z (POS)
062	Fehlermeldung von Achsrechner (POS)	Error message from axis computer (POS)
063	Aktuelle Istpositionsvorgabe an Achsrechner (POS)	Actual actual position to axis computer (POS)
064	Aktueller Auftrag an Achsrechner (POS)	Actual job to axis computer (POS)
065	Aktuelle Geschwindigkeit des Handrades (POS)	Actual contour speed of the handwheel (POS)
066	Startpositionen und S- und T-Koordinatensystem Open Satz (POS)	
067	Vsoll, Vist und Vziel bei Open Satz (POS)	Vnominal, Vactual and Vtarget in the case of open block (POS)
068	Aktuelles Vsoll, Vist und Vziel aus Bahnrechner (POS)	Actual Vnominal, Vactual and Vtarget from contour computer (POS)
069	Aktuelle Radius und Winkelposition im Polarkoordinatensystem (POS)	Actual radius and angle position in the polar coordinate system (POS)
070	Satz von Speicherverwaltung lesen (SPV)	Read block from memory management (SPV)
071	Satz an Satzkonverter (SPV)	Block to block converter (SPV)
072	Auftragsquittung (SPV)	Job acknowledgement (SPV)
073	Start der Programmanlage (SPV)	Start of program creation (SPV)
074	Ende der Programmanlage (SPV)	End of program creation (SPV)
075	Programmblock vor Bearbeitung durch SPV	Program block before editing by SPV
076	Neue Nachricht emfangen (SPV)	New message received (SPV)
077	Konvertierter Einzelsatz an ZST (SPV)	Converted single block to ZST (SPV)
078	Einmalige Anzeige der Symboltabelle (SPV)	Single display of the symbol table (SPV)
079	Neuer Satz aus Speicherverwaltung (Original)	New block from memory management (original)
080	Neuer Satz aus Speicherverwaltung (nach Eintrag der indirekten Programmierung)	New block from memory management (after input of indirect programming)

081	WLK_out Dinsatz aus Korrekturpuffer	WLK_out Din block from compensation buffer
082	WLK_out Dinsatz direkt aus intd_PR	WLK_out Din block directly from intd_PR
083	Neuer Satz nach Korrekturmodulen	New block after compensation modules
084	Satz nach polar_haupt()	Block after polar_haupt()
085	Satz nach Radiusübergang()	Block after radius transition()
086	Kreissatz vor Mittelpunktsbestimmung	Circular block before centre point determination
087		
088	CoDeSys TCP Kommunikation (NET)	CoDeSys TCP communication (NET)
089	Zusatzinformation bei Dinsatz-Trace einschalten	Activate additional information in the case of Din block trace
090	Download-Nachricht über DPR (ZST)	Download message via DPR (ZST)
091	Einzelsatz von SPS (ZST)	Single block from PLC (ZST)
092	Einzelsatz an SPV (ZST)	Single block to SPV (ZST)
093	Zustandswechsel (ZST)	Status change (ZST)
094	Nachricht aus Mailbox (ZST)	Message from mailbox (ZST)
095	P-Feld-Schreiben (ZST)	Write P field (ZST)
096	Nachricht vom MMI über DPR (ZST)	Message from MMI via DPR (ZST)
097	Nachricht ans MMI über DPR (ZST)	Message to MMI via DPR (ZST)
098	Änderung der Starttaste (ZST)	Change of start key (ZST)
099	Einzelsatz von MMI (ZST)	Single block from MMI (ZST)
100	Nutzdaten bei Blockübertragungen vom MMI (ZST)	User data in the case of block transfers from MMI (ZST)
101		
102		
103		
104		
105		
106		
107		
108		
109		
110		
111		
112		

113		
114		
115		
116		
117		
118		
119		
120	reserviert für SPS	Reserved for PLC
121	reserviert für SPS	Reserved for PLC
122	reserviert für SPS	Reserved for PLC
123	reserviert für SPS	Reserved for PLC
124	reserviert für SPS	Reserved for PLC
125	reserviert für SPS	Reserved for PLC
126	reserviert für SPS	Reserved for PLC
127	reserviert für SPS	Reserved for PLC
128	reserviert für SPS	Reserved for PLC
129	reserviert für SPS	Reserved for PLC
130	reserviert für SPS	Reserved for PLC
131	reserviert für SPS	Reserved for PLC
132	reserviert für SPS	Reserved for PLC
133	reserviert für SPS	Reserved for PLC
134	reserviert für SPS	Reserved for PLC
135	reserviert für SPS	Reserved for PLC
136	reserviert für SPS	Reserved for PLC
137	reserviert für SPS	Reserved for PLC
138	reserviert für SPS	Reserved for PLC
139	reserviert für SPS	Reserved for PLC
140		
141		
142		
143		
144		

145		
146		
147		
148		
149		
150	CAN-Bus: Empangene Nachrichten (SLIO)	CAN-Bus: received messages (SLIO)
151	CAN-Bus: Gesendete Nachrichten (SLIO)	CAN-Bus: transmitted messages (SLIO)
152	CAN-Bus: Empfangsfehler	CAN-Bus: receive error
153	CAN-Bus: Sendepufferüberlauf	CAN-Bus: transmitbuffer overflow
154		
155	CAN-Open: SearchNode	CAN-Open: SearchNode
156	CAN-Open: RestartNode	CAN-Open: RestartNode
157		
158		
159		
160	CAN1: NMT-Object gesendet/empfangen	CAN1: NMT-object transmitted/received
161	CAN1: SYNC- oder EMERGENCY-Object gesendet/empfangen	CAN1: SYNC- or EMERGENCY-object transmitted/received
162	CAN1: TIME-STAMP-Object gesendet/empfangen	CAN1: TIME-STAMP-object transmitted/received
163	CAN1: PDO1(tx) gesendet/empfangen	CAN1: PDO1(tx) transmitted/received
164	CAN1: PDO1(rx) gesendet/empfangen	CAN1: PDO1(rx) transmitted/received
165	CAN1: PDO2(tx) gesendet/empfangen	CAN1: PDO2(tx) transmitted/received
166	CAN1: PDO2(rx) gesendet/empfangen	CAN1: PDO2(rx) transmitted/received
167	CAN1: PDO3(tx) gesendet/empfangen	CAN1: PDO3(tx) transmitted/received
168	CAN1: PDO3(rx) gesendet/empfangen	CAN1: PDO3(rx) transmitted/received
169	CAN1: PDO4(tx) gesendet/empfangen	CAN1: PDO4(tx) transmitted/received
170	CAN1: PDO4(rx) gesendet/empfangen	CAN1: PDO4(rx) transmitted/received
171	CAN1: SDO(tx) gesendet/empfangen	CAN1: SDO(tx) transmitted/received
172	CAN1: SDO(rx) gesendet/empfangen	CAN1: SDO(rx) transmitted/received
173	CAN1: Function Code 13 gesendet/empfangen	CAN1: Function Code 13 transmitted/received
174	CAN1: Nodeguard-Object gesendet/empfangen	CAN1: Nodeguard-Object transmitted/received
175	CAN1: Function Code 15 gesendet/empfangen	CAN1: Function Code 15 transmitted/received
176		

177		
178		
179		
180	CAN2: NMT-Object gesendet/empfangen	CAN2: NMT-Object transmitted/received
181	CAN2: SYNC- oder EMERGENCY-Object gesendet/empfangen	CAN2: SYNC- oder EMERGENCY-object transmitted/received
182	CAN2: TIME-STAMP-Object gesendet/empfangen	CAN2: TIME-STAMP-Object transmitted/received
183	CAN2: PDO1(tx) gesendet/empfangen	CAN2: PDO1(tx) transmitted/received
184	CAN2: PDO1(rx) gesendet/empfangen	CAN2: PDO1(rx) transmitted/received
185	CAN2: PDO2(tx) gesendet/empfangen	CAN2: PDO2(tx) transmitted/received
186	CAN2: PDO2(rx) gesendet/empfangen	CAN2: PDO2(rx) transmitted/received
187	CAN2: PDO3(tx) gesendet/empfangen	CAN2: PDO3(tx) transmitted/received
188	CAN2: PDO3(rx) gesendet/empfangen	CAN2: PDO3(rx) transmitted/received
189	CAN2: PDO4(tx) gesendet/empfangen	CAN2: PDO4(tx) transmitted/received
190	CAN2: PDO4(rx) gesendet/empfangen	CAN2: PDO4(rx) transmitted/received
191	CAN2: SDO(tx) gesendet/empfangen	CAN2: SDO(tx) transmitted/received
192	CAN2: SDO(rx) gesendet/empfangen	CAN2: SDO(rx) transmitted/received
193	CAN2: Function Code 13 gesendet/empfangen	CAN2: Function Code 13 transmitted/received
194	CAN2: Nodeguard-Object gesendet/empfangen	CAN2: Nodeguard-Object transmitted/received
195	CAN2: Function Code 15 gesendet/empfangen	CAN2: Function Code 15 transmitted/received
196		
197		
198		
199		
200		
201	Start- und Zielpositionen bei Ausgleichsfahrt (POS)	Start- and targetpositions at compensation drive (POS)
202	Beteiligte Achsen und Modpos bei Ausgleichsfahrt (POS)	Axis involved and Modpos at compensation drive (POS)
203	Beteiligte Achsen, Zielpos und Modpos bei Ausgleichsfahrt (POS)	Axis involved, targetpos and Modpos at compensation drive (POS)
204	Beteiligte Achsen, Zielpos und NC-Istpos bei Ausgleichsfahrt (POS)	Axis involved, targetpos and NC-actualpos at compensation drive (POS)
205	Unterbrochener Verfahrsatz bei Ausgleichsfahrt (POS)	Interrupted drive block at compensation drive (POS)

206	Korrekturwert bei 3D-Achskorrektur (POS)	Correction value at 3D-axis correction (POS)
207		
208		
209		
210	CAN2: Senden einer PDO mit NodeId 0	CAN2: Transmission of PDO with NodeId 0
211	CAN2: Senden einer PDO mit NodeId 1	CAN2: Transmission of PDO with NodeId 1
212	CAN2: Senden einer PDO mit NodeId 2	CAN2: Transmission of PDO with NodeId 2
213	CAN2: Senden einer PDO mit NodeId 3	CAN2: Transmission of PDO with NodeId 3
214	CAN2: Senden einer PDO mit NodeId 4	CAN2: Transmission of PDO with NodeId 4
215	CAN2: Senden einer PDO mit NodeId 5	CAN2: Transmission of PDO with NodeId 5
216	CAN2: Senden einer PDO mit NodeId 6	CAN2: Transmission of PDO with NodeId 6
217	CAN2: Senden einer PDO mit NodeId 7	CAN2: Transmission of PDO with NodeId 7
218	CAN2: Senden einer PDO mit NodeId 8	CAN2: Transmission of PDO with NodeId 8
219	CAN2: Senden einer PDO mit NodeId 9	CAN2: Transmission of PDO with NodeId 9
220	CAN2: Senden einer PDO mit NodeId 10	CAN2: Transmission of PDO with NodeId 10
221	CAN2: Senden einer PDO mit NodeId 11	CAN2: Transmission of PDO with NodeId 11
222	CAN2: Senden einer PDO mit NodeId 12	CAN2: Transmission of PDO with NodeId 12
223		
224		
225	CAN2: Empfangen einer PDO mit NodeId 0	CAN2: Receiving of PDO with NodeId 0
226	CAN2: Empfangen einer PDO mit NodeId 1	CAN2: Receiving of PDO with NodeId 1
227	CAN2: Empfangen einer PDO mit NodeId 2	CAN2: Receiving of PDO with NodeId 2
228	CAN2: Empfangen einer PDO mit NodeId 3	CAN2: Receiving of PDO with NodeId 3
229	CAN2: Empfangen einer PDO mit NodeId 4	CAN2: Receiving of PDO with NodeId 4
230	CAN2: Empfangen einer PDO mit NodeId 5	CAN2: Receiving of PDO with NodeId 5
231	CAN2: Empfangen einer PDO mit NodeId 6	CAN2: Receiving of PDO with NodeId 6
232	CAN2: Empfangen einer PDO mit NodeId 7	CAN2: Receiving of PDO with NodeId 7
233	CAN2: Empfangen einer PDO mit NodeId 8	CAN2: Receiving of PDO with NodeId 8
234	CAN2: Empfangen einer PDO mit NodeId 9	CAN2: Receiving of PDO with NodeId 9
235	CAN2: Empfangen einer PDO mit NodeId 10	CAN2: Receiving of PDO with NodeId 10
236	CAN2: Empfangen einer PDO mit NodeId 11	CAN2: Receiving of PDO with NodeId 11
237	CAN2: Empfangen einer PDO mit NodeId 12	CAN2: Receiving of PDO with NodeId 12

238		
239		
240	SMTP Transfer (NET)	SMTP Transfer (NET)
241	SDO Server (CAN)	SDO Server (CAN)
242	HTTP Server (NET)	HTTP Server (NET)
243		
244		
245		
246		
247		
248		
249		
250		
251		
252		
253	Meldung: Aufgabe beendet	Message: Job done
254	Fehlermeldung als Nummer	Error message as number
255	Fehlermeldung im Klartext	Error message in plain language