

Title Rexroth Rho 4
DDE–Server 4

Type of Documentation Software manual

Document Typecode DOK-RHO*4*-DDE*SERVER*-PR06-EN-P

Purpose of Documentation The present manual informs about:

- the use of the programming of the rho4 with the program DDE–Server 4.

Record of Revisions

Description	Release Date	Notes
DOK-RHO*4*-DDE*SERVER*-PR05-EN-P	10.2003	Valid from VO07
DOK-RHO*4*-DDE*SERVER*-PR06-EN-P	01.2005	Valid from VO08

Copyright © Bosch Rexroth AG, 1998 – 2005
Copying this document, giving it to others and the use or communication of the contents thereof without express authority, are forbidden. Offenders are liable for the payment of damages. All rights are reserved in the event of the grant of a patent or the registration of a utility model or design (DIN 34–1).

Validity The specified data is for product description purposes only and may not be deemed to be guaranteed unless expressly confirmed in the contract. All rights are reserved with respect to the content of this documentation and the availability of the product.

Published by Bosch Rexroth AG
Postfach 11 62
D-64701 Erbach
Berliner Straße 25
D-64711 Erbach
Tel.: +49 (0) 60 62/78-0
Fax: +49 (0) 60 62/78-4 28
Abt.: BRC/ESH (KW)

Overview of all manuals

Overview of all manuals

Manual	Contents
Connection conditions Rho 4.0	2 System overview
	3 Installation
	4 Electrical connection
	5 Interfaces
	6 LED display
	7 Maintenance and replacement
	8 Order numbers
System description Rho 4.0	2 System overview
	3 Structure of the rho4.0
	4 PCLrho4.0
	5 CAN-Bus peripheral unit
	6 SERCOS interface
	7 Software
	8 File management
Connection conditions Rho4.1, Rho 4.1/IPC300	2 System overview
	3 Security functions
	4 Installation
	5 Electrical connection
	6 Interfaces
	7 LED display
	8 Maintenance and replacement
	9 Software
	10 Order numbers
Connection conditions Rho 4.1/BT155, Rho 4.1/BT155T, Rho 4.1/BT205	2 System overview
	3 Security functions
	4 Installation
	5 Electrical Connections
	6 Interfaces
	7 Display and Operating Controls
	8 Maintenance and Replacemant
	9 Software
	10 Order numbers
System description Rho 4.1	2 Structure of the rho4.1
	3 PCL
	4 CAN-Bus peripheral unit
	5 SERCOS interface

Overview of all manuals

Manual	Contents
	6 Software
	7 File management
	8 Scope of the rho4.1 Software

Manual	Contents
Control functions	2 Survey of special functions
	3 Accurate position switching
	4 Setting the machine position
	5 Calling operating system functions
	6 Parameterization of the belt characteristic
	7 Selecting a point-file
	8 Mirroring
	9 Belt type
	10 System date and time
	11 System counter
	12 WC main range
	13 Setting the belt counter
	14 Recording of reference path
	15 Flying measurement (rho4.1 only)
	16 MOVE_FILE
	17 Setting the block preparation
	18 Exception-Handling
	19 Belt counter current value
	20 Automatic velocity adjustment for PTP movements
	21 Belt-synchronous working area belt kind 4
	22 Current belt speed
	23 Changing the belt simulation speed
	24 General functions
	25 Process-oriented functions
	26 BAPS3 keywords
	Machine parameters
3 Application of the machine parameters	
4 General system parameters	
5 Speeds	
6 Positions	
7 Kinematic parameters	

Overview of all manuals

Manual	Contents
	8 Measuring system parameters
	9 Belt parameters
	10 Drive parameters Servodyn-GC
	11 Drive parameter Servodyn-D
	12 Table of parameters
Manual	Contents
BAPS3 Programming manual	2 Program structure
	3 Constants
	4 Variables
	5 Program control
	6 Value assignments and combinations
	7 Functions
	8 Movement statement
	9 Write/read functions
	10 BAPS3 keywords
	BAPS3 Short description
3 Constants and variables	
4 Program structure	
5 Value assignments and combinations	
6 Standard functions	
7 Movements and speeds	
8 Belt synchronous	
9 Workspace limitation	
10 Write/read functions	
11 Special functions	
12 Library functions	
13 Fix files	
14 BAPS3 keywords	
Signal descriptions	
	3 Signal description of PCL inputs
	4 Signal description of PCL outputs
Status messages and warnings	2 rho4 status messages
	3 Warnings
	4 CANopen error codes
ROPS4/Online	2 General information
	3 Activation and functions of Online
	4 The function key box

Overview of all manuals

Manual	Contents
	5 Function key assignment
	6 The marker box
	7 File ROPS4WIN.ini
	8 Selection of a file
	9 TCP/IP settings for ROPS4
Manual	Contents
DLL library	2 Library functions
	3 Calling library functions in BAPS
	4 Block structure of the rho4.1
	5 Library server
	6 Application development
	7 rho4 library functions
	8 Variable access per DLL
	PHG2000
3 PHG2000 system variables	
4 Selection of PHG functions	
5 Info function of the PHG	
6 Controlling the PHG2000 output	
7 Define/Teach	
8 SRCAN functions	
9 File and User Memory Functions	
10 File list	
11 Process info	
12 Restoring the PGH display	
13 Variable assignment of PHG keys	
14 Select point file and point name	
15 BDT editor	
Connection conditions Rho 4.1/IPC 40.2	
	3 Security Functions
	4 Installation
	5 Eelectrical Connections
	6 Interface Ports & Connectors
	7 Display- and Operating Components
	8 Maintenance and Replacement
	9 Software
	10 Ordering Informations

Overview of all manuals

Manual	Contents
DDE-Server	2 Introduction
	3 Hardware and Software
	4 Operation
	5 Items of Server 4
	6 Scope of function

Overview of all manuals

Notes:

Contents

Contents

	Page
1	Safety Instructions 1-1
1.1	Intended use 1-1
1.2	Qualified personnel 1-2
1.3	Safety markings on products 1-3
1.4	Safety instructions in this manual 1-4
1.5	Safety instructions for the described product 1-5
1.6	Documentation, software release and trademarks 1-7
2	Introduction 2-1
2.1	DDE and DDEML 2-2
2.2	Connection set-up 2-2
2.3	Static data exchange 2-3
2.4	Request of dynamic data 2-4
2.5	Terminate connection 2-4
2.6	Presentation in the description 2-5
3	Hardware and Software 3-1
3.1	Scope of supply 3-1
3.2	Requirements 3-1
3.3	Software protection 3-2
4	Operation 4-1
5	Items of Server4 5-1
5.1	Connection between Client and Server4 5-1
5.2	Connection between PC and control 5-3
5.3	Items 5-4
5.3.1	File administration functions 5-4
5.3.2	Cyclic items 5-4
5.3.3	Non-cyclic items 5-4
5.3.4	Items with ASCII log 5-5
5.3.5	Special function GStatus 5-5
5.4	DDESVR.INI 5-6

Contents

6	Scope of functions	6-1
6.1	Status and initializing functions	6-1
6.1.1	Global status	6-1
6.1.2	Control and check possibilities for ASCII items	6-4
6.1.3	Server error	6-7
6.1.4	Report of control status messages/warnings	6-8
6.1.5	List of all control errors/warnings	6-9
6.1.6	List of all control errors/warnings in ASCII	6-11
6.1.7	Initialization of a serial interface	6-12
6.1.8	Initialization of a TCP/IP connection	6-13
6.1.9	Closing of a serial interface	6-14
6.1.10	Closing of a TCP/IP connection	6-14
6.1.11	Automatic initialization	6-15
6.1.12	Monitoring of the connection between control and Server4	6-15
6.2	File transfer functions	6-17
6.2.1	Download	6-17
6.2.2	ASCII download	6-19
6.2.3	Upload	6-22
6.2.4	ASCII upload	6-24
6.2.5	Dir	6-26
6.2.6	Rename	6-28
6.2.7	Delete	6-29
6.3	Online functions	6-30
6.3.1	Kinematics info	6-30
6.3.2	Axis positions	6-31
6.3.3	Axis data in ASCII	6-34
6.3.4	Tool	6-36
6.3.5	WC system	6-37
6.3.6	Process selection	6-38
6.3.7	Process stop	6-41
6.3.8	Process list	6-41
6.3.9	Process status	6-43
6.3.10	Reset via PG	6-46
6.3.11	Set RCO	6-49
6.3.12	Signal display	6-49
6.4	Access to user variables	6-52
6.4.1	General information	6-52
6.4.2	Reading of variables	6-55
6.4.3	Reading of variables with an ASCII protocol	6-61
6.4.4	Writing of variables	6-65
6.4.5	Writing of variables with ASCII protocol	6-69
6.4.6	Example	6-73
A	Appendix	A-1
A.1	Abbreviations	A-1
A.2	Index	A-2

Safety Instructions

1 Safety Instructions

Please read this manual before you startup the rho4.
Store this manual in a place to which all users have access at any time.

1.1 Intended use


This instruction manual presents a comprehensive set of instructions and information required for the standard operation of the described products. The described products are used for the purpose of operating with a robot control rho4.

The products described

- have been developed, manufactured, tested and documented in compliance with the safety standards. These products normally pose no danger to persons or property if they are used in accordance with the handling stipulations and safety notes prescribed for their configuration, mounting, and proper operation.
- comply with the requirements of
 - the EMC Directives (89/336/EEC, 93/68/EEC and 93/44/EEC)
 - the Low-Voltage Directive (73/23/EEC)
 - the harmonized standards EN 50081-2 and EN 50082-2
- are designed for operation in industrial environments, i.e.
 - no direct connection to public low-voltage power supply,
 - connection to the medium- or high-voltage system via a transformer.

The following applies for application within a personal residence, in business areas, on retail premises or in a small-industry setting:

- Installation in a control cabinet or housing with high shield attenuation.
- Cables that exit the screened area must be provided with filtering or screening measures.
- The user will be required to obtain a single operating license issued by the appropriate national authority or approval body. In Germany, this is the Federal Institute for Posts and Telecommunications, and/or its local branch offices.

 **This is a Class A device. In a residential area, this device may cause radio interference. In such case, the user may be required to introduce suitable countermeasures, and to bear the cost of the same.**

The faultless, safe functioning of the product requires proper transport, storage, erection and installation as well as careful operation.

Safety Instructions

1.2 Qualified personnel

The requirements as to qualified personnel depend on the qualification profiles described by ZVEI (central association of the electrical industry) and VDMA (association of German machine and plant builders) in:

Weiterbildung in der Automatisierungstechnik

edited by: ZVEI and VDMA

MaschinenbauVerlag

Postfach 71 08 64

D-60498 Frankfurt.

The present manual is designed for RC technicians. They need special knowledge on handling and programming robots.

Interventions in the hardware and software of our products, unless described otherwise in this manual, are reserved to specialized Rexroth personnel.

Tampering with the hardware or software, ignoring warning signs attached to the components, or non-compliance with the warning notes given in this manual may result in serious bodily injury or damage to property.

Only electrotechnicians as recognized under IEC 60947-1 (modified) who are familiar with the contents of this manual may install and service the products described.

Such personnel are

- those who, being well trained and experienced in their field and familiar with the relevant norms, are able to analyze the jobs being carried out and recognize any hazards which may have arisen.
- those who have acquired the same amount of expert knowledge through years of experience that would normally be acquired through formal technical training.

With regard to the foregoing, please note our comprehensive range of training courses. Please visit our website at

<http://www.boschrexroth.com>

for the latest information concerning training courses, teachware and training systems. Personal information is available from our Didactic Center Erbach,

Telephone: (+49) (0) 60 62 78-600.

Safety Instructions

1.3 Safety markings on products

Warning of dangerous electrical voltage!



Warning of danger caused by batteries!



Electrostatically sensitive components!



Warning of hazardous light emissions
(optical fiber cable emissions)!



Disconnect mains power before opening!



Lug for connecting PE conductor only!



Functional earthing or low-noise earth only!



Connection of shield conductor only

Safety Instructions

1.4 Safety instructions in this manual



DANGEROUS ELECTRICAL VOLTAGE

This symbol is used to warn of a **dangerous electrical voltage**. The failure to observe the instructions in this manual in whole or in part may result in **personal injury**.



DANGER

This symbol is used wherever insufficient or lacking compliance with instructions may result in **personal injury**.



CAUTION

This symbol is used wherever insufficient or lacking compliance with instructions may result in **damage to equipment or data files**.

☞ This symbol is used to draw the user's attention to special circumstances.

★ This symbol is used if user activities are required.

Safety Instructions

1.5 Safety instructions for the described product**DANGER**

Danger of life through inadequate EMERGENCY-STOP devices! EMERGENCY-STOP devices must be active and within reach in all system modes. Releasing an EMERGENCY-STOP device must not result in an uncontrolled restart of the system! First check the EMERGENCY-STOP circuit, then switch the system on!

**DANGER**

**Danger for persons and equipment!
Test every new program before starting up a system!**

**DANGER**

**Retrofits or modifications may adversely affect the safety of the products described!
The consequences may include severe injury, damage to equipment, or environmental hazards. Possible retrofits or modifications to the system using third-party equipment therefore have to be approved by Rexroth.**

**DANGER**

Do not look directly into the LEDs in the optical fiber connection. Due to their high output, this may result in eye injuries. When the inverter is switched on, do not look into the LED or the open end of a short connected lead.

**DANGEROUS ELECTRICAL VOLTAGE**

Unless described otherwise, maintenance works must be performed on inactive systems! The system must be protected against unauthorized or accidental reclosing.

Measuring or test activities on the live system are reserved to qualified electrical personnel!

Safety Instructions



CAUTION

Danger to the module!

Do not insert or remove the module while the controller is switched ON! This may destroy the module. Prior to inserting or removing the module, switch OFF or remove the power supply module of the controller, external power supply and signal voltage!



CAUTION

use only spare parts approved by Rexroth!



CAUTION

Danger to the module!

All ESD protection measures must be observed when using the module! Prevent electrostatic discharges!

The following protective measures must be observed for modules and components sensitive to electrostatic discharge (ESD)!

- Personnel responsible for storage, transport, and handling must have training in ESD protection.
- ESD-sensitive components must be stored and transported in the prescribed protective packaging.
- ESD-sensitive components may only be handled at special ESD-workplaces.
- Personnel, working surfaces, as well as all equipment and tools which may come into contact with ESD-sensitive components must have the same potential (e.g. by grounding).
- Wear an approved grounding bracelet. The grounding bracelet must be connected with the working surface through a cable with an integrated 1 MΩ resistor.
- ESD-sensitive components may by no means come into contact with chargeable objects, including most plastic materials.
- When ESD-sensitive components are installed in or removed from equipment, the equipment must be de-energized.

Safety Instructions

1.6 Documentation, software release and trademarks

Documentation

The present manual provides information on the use of the programming of the rho4 with the program DDE server 4.

Overview of available documentation	Part no.	
	German	English
Rho 4.0 Connectivity Manual	1070 072 364	1070 072 365
Rho 4.0 System description	1070 072 366	1070 072 367
Rho 4.1/IPC 40.2 Connectivity Manual	R911308219	R911308220
Rho 4.1/BT155, Rho 4.1/BT155T, Rho 4.1/BT205 Connectivity manual	1070 072 362	1070 072 363
Rho 4.1, Rho 4.1/IPC300 Connectivity manual	1070 072 360	1070 072 361
Control panels BF2xxT/BF3xxT, connection	1070 073 814	1070 073 824
Rho 4.1 System description	1070 072 434	1070 072 185
ROPS4/Online	1070 072 423	1070 072 180
BAPS plus	1070 072 422	1070 072 187
BAPS3 Short description	1070 072 412	1070 072 177
BAPS3 Programming manual	1070 072 413	1070 072 178
Control functions	1070 072 420	1070 072 179
Signal descriptions	1070 072 415	1070 072 182
Status messages and warnings	1070 072 417	1070 072 181
Machine parameters	1070 072 414	1070 072 175
PHG2000	1070 072 421	1070 072 183
DDE-Server 4	1070 072 433	1070 072 184
DLL-Library	1070 072 418	1070 072 176
Rho 4 available documentation on CD ROM	1070 086 145	1070 086 145

 **In this manual the floppy disk drive always uses drive letter A:, and the hard disk drive always uses drive letter C:.**

Special keys or key combinations are shown enclosed in pointed brackets:

- Named keys: e.g., <Enter>, <PgUp>,
- Key combinations (pressed simultaneously): e.g., <Ctrl> + <PgUp>

Safety Instructions

Release

 **This manual refers to the following versions:**

Hardware version: rho4

Software release: ROPS4

Trademarks

All trademarks of software installed on Rexroth products upon delivery are the property of the respective manufacturer.

Upon delivery, all installed software is copyright-protected. The software may only be reproduced with the approval of Rexroth or in accordance with the license agreement of the respective manufacturer.

MS-DOS® and Windows™ are registered trademarks of Microsoft Corporation.

PROFIBUS® is a registered trademark of the PROFIBUS Nutzerorganisation e.V. (user organization).

MOBY® is a registered trademark of Siemens AG.

AS-I® is a registered trademark of AS-International Association.

SERCOS interface™ is a registered trademark of Interessengemeinschaft SERCOS interface e.V. (Joint VDW/ZVEI Working Committee).

INTERBUS-S® is a registered trade mark of Phoenix Contact.

DeviceNet® is a registered trade mark (TM) of ODVA (Open DeviceNet Vendor Association, Inc.).

Introduction

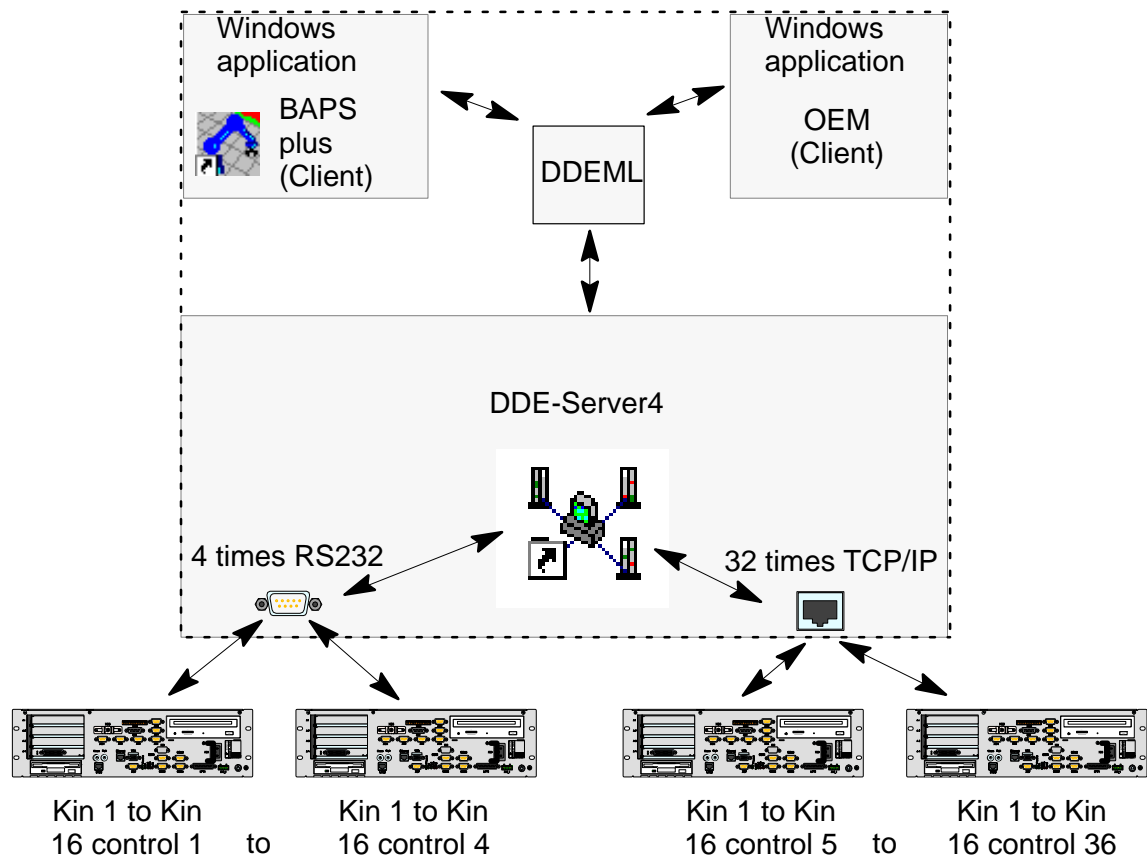
2 Introduction

The software package ROPS4 offers a variety of functions to communicate with the control rho4. It contains file transfer functions, process and status functions, in the following also called Online functions. These items are contained in a program package, equipped with a comfortable operating surface, and they run under Windows.

In order to offer to the user the possibility of integrating the Online functions into his own surface or to operate the rho4.1 under Windows 'remote control', i.e. without operation from the PC, a function library with a standardized interface is necessary.

For MS-Windows, the inter-process-communication DDE is available for this purpose. It is supported by the operating systems Windows NT (>=3.5 build 807) and Windows95.

The description on hand is based on the current software version of the DDE-Server4.



Introduction

2.1 DDE and DDEML

Dynamic Data Exchange (DDE) is a form of inter-process communication of Windows using 'Shared Memory' to exchange data between Windows applications. For the data exchange of two Windows applications one must serve as Client (e.g. the surface) and another as Server (Online-DDE-Server4).

In this context, an application is called Server when it offers items (in the following also called items) to other applications. The application using the items of a Server is called Client.

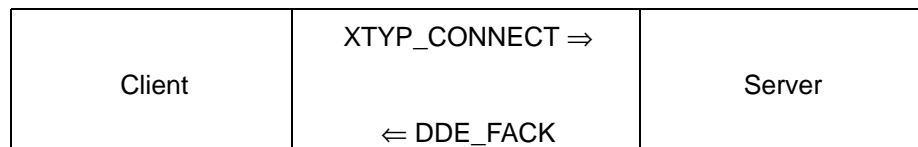
For this communication the Online-DDE-Server4 offers several items the Client can use. These items provide the connection set-up, data exchange, monitoring, execution etc. Concerning the data exchange, it is possible to distinguish between a single transfer (e.g. process start) and dynamic data exchange (e.g. axis display). All functions for the process communication between Client and Server are located in the DDEML Library (Dynamic Data Exchange Management Library). Generally, Client and Server can only communicate via these functions.

A DDE-Server can support several formats for the data exchange. Standard and minimum at the same time is the clipboard format CF_TEXT. The Online-DDE-Server4 is based on the functions of the DDEML and the format CF_TEXT is the only format it supports.

In the following, the actions between Client and Server are explained in principle. All functions for the message exchange and the message types themselves (e.g. XTYP_CONNECT) are defined in the DDEML.

2.2 Connection set-up

Before a Client can request data from a Server, a connection with the latter must be established first.



The Client sends the message XTYP_CONNECT (via DDEML) to the Server. The Server initializes the connection and answers with DDE_FACK, if this has been carried out without error.

Introduction

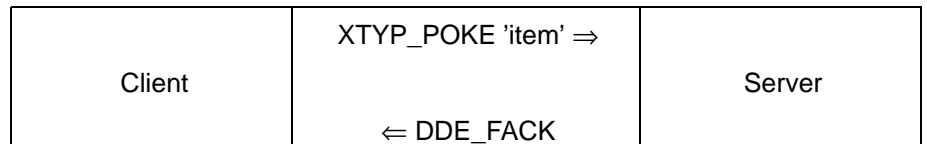
2.3 Static data exchange

The single data exchange between Client and Server is also called 'cold link'.

There are two possibilities for exchanging static data:

First possibility

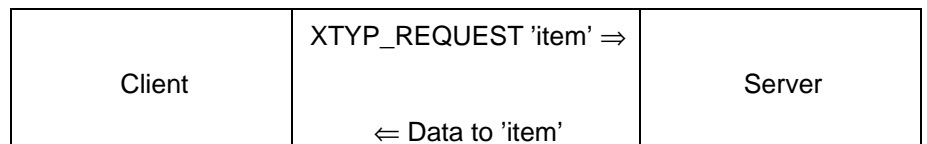
The Client sends data to the Server (e.g. interface parameters)



The Client sends the message XTYP_POKE (via DDEML) to the Server with an identification ('item') and the corresponding data. Through the 'items' the Server recognizes which kind of data are sent. With the message DDE_FACK the Server acknowledges the data receipt.

Second possibility

The Client orders from the Server once only specific data (e.g. kinematics information)

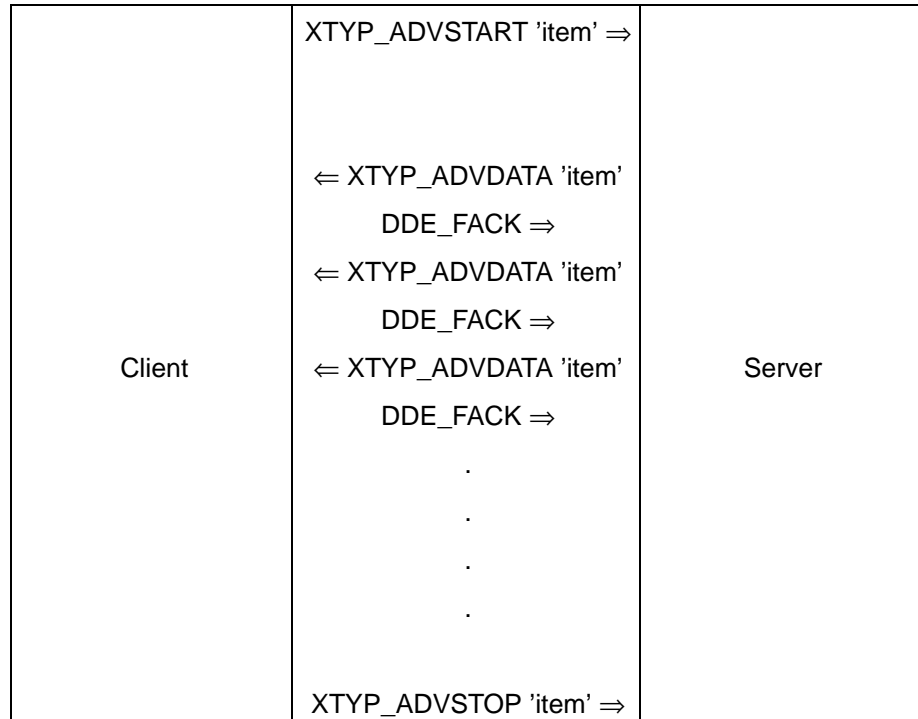


The Client sends the message XTYP_REQUEST (via DDEML) to the Server with an identification ('item'). With the help of the 'item' sent together with the message, the Server recognizes which data are to be sent to the Client.

Introduction

2.4 Request of dynamic data

For data changing continuously, the Client can set up a dynamic connection (hot link). Afterwards the Server is sending its data cyclically in certain time intervals. This takes place until the Client cancels the dynamic connection. In order to prevent that the system is stressed unnecessarily, certain data are only sent when their contents have changed. An example for this kind of transfer is the request of the axes positions in ASCII.



The Client sends the message XTYP_ADVSTART (via DDEML) to the Server with an identification ('item') and so starts the dynamic data exchange. With the help of the 'item' sent together with the message, the Server recognizes which dynamic data are to be sent to the Client. The data are sent from the Server to the Client with the message XTYP_ADVDATA. The Client must acknowledge reception with DDE_FACK.

With the message XTYP_ADVSTOP, the dynamic data exchange is completed.

2.5 Terminate connection

When a Client does not need any more data from the Server, the connection must be terminated. Only then, the interface occupied by a Connect is enabled again.

Introduction

Client	XTYP_DISCONNECT ⇒	Server
--------	-------------------	--------

The Client sends the message XTYP_DISCONNECT (via DDEML) to the Server. The Server terminates the connection and enables the interface.

2.6 Presentation in the description

In the following chapters of the description, the data traffic between Client and Server4 is presented in tables.

Example of a DDE table

Client	Message 'Item'	Data	↔	Server4
Cyclic status inquiry start	XTYP_ADVSTART 'StFehler'	---	⇒	
	TRUE	---	⇐	acknowledge instruction
until stop	XTYP_ADVDATA 'StFehler'	StFehler	⇐	send data cyclically
	DDE_FACK 'StFehler'	---	⇒	
Stop status	XTYP_ADVSTOP 'StFehler'	---	⇒	

Explaining the table

- Column 1 (Client) Short explanation of DDE commands from the Client's point of view.
- Column 2 (message) DDE commands and possible 'items'.
- Column 3 (data) Names of the structures via which the data exchange is taking place. These structures are explained subsequently to the respective table. (The affiliated structs resp. defines are to be found in the file Client.h delivered simultaneously). A --- means that with the respective message no data are exchanged.
- Column 4 (↔) This column indicates the data direction (⇒ means from the Client to the Server4; ⇐ means from the Server4 to the Client).
- Column 5 (Server4) Short explanation of the DDE-command from the Server's point of view.

Introduction

Notes:

Hardware and Software

3 Hardware and Software

3.1 Scope of supply

The list of the files contained in the scope of delivery of the DDE server can be found in the file FILELIST.TXT in the installation directory.

The directory \BOSCH\DDESVR\EXAMPLES contains example programs for clients under EXCEL, ACCES, WINWORD and C dealing with the access to user variables. Notes and special information are to be found in the corresponding directories of the README.WRI file. In the DDESVR.INI-files, also included there, the initializations required for the respective client have already been carried out and can be adopted from there.

The Online-DDE-Server4 is available in a German and in an English version. The desired language is determined by an entry into the file DDESVR.INI (Group 'SERVERINIT' entry 'language').

3.2 Requirements

The following conditions must be fulfilled:

- IBM-AT-compatible PCs (at least 486)
- 4 MByte RAM
- Hard disk
- CD-ROM drive
- at least one serial port (16 byte FIFO recommended) or a network card for TCP/IP connections
- Windows95 or WindowsNT (from version 3.5 on)

The person developing a Client should have profound knowledge about the programming of Windows applications and the DDE-interface. Appropriate tools (InTouch, Visual Basic, Visual C etc.) facilitate the development of a Client. The compiler must be set to ALIGNMENT=4. The Timeout time required by different DDE functions must be set to five seconds.

Helpful literature on the subject of Windows programming and DDE is about others the book Programming under Windows by Charles Petzold from the Microsoft specialist library.

Hardware and Software

In the control, file names with a maximum length of eight characters for the name and three characters for the extension are permitted. The On-line-DDE-Server4 automatically abbreviates file names with more than eight characters before loading into the control as it is usual under Windows95. To exclude problems, it is therefore recommended to restrict the name of files to be handled by Server4 to eight characters.

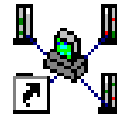
3.3 Software protection

The DDE-Server4 is protected by a software dongle. After installation the DDE-Server4 must be enabled by entering a computer key (a number predefined by Bosch). The process of licence application and allocation is described in the file Liesmich.wri. An application form for the enable key can be found in the Fax.wri.file.

Operation

4 Operation

The Online-DDE-Server4 is designed as an independent Windows application. The Server4 has no active surface, but is lying as icon in the background.



For operating, a menu can be opened (double click on the icon) with which the Server4 can be configured and observed.




Operation

The menu includes the following functions:

rho

Display of the control versions and the set interface parameters. This function serves for testing the communication between Server and control. If errors should occur they are displayed on the monitor. Before the first call, the interface parameters must be set (see Setup).

After selection of the menu point, the following dialog appears:



The screenshot shows a dialog box titled "Teste Verbindung zur rho4". It contains the following fields and buttons:

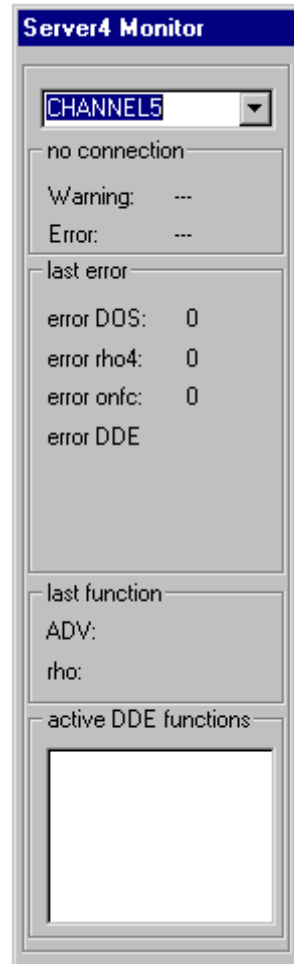
- "Verbindung über:" with a dropdown menu showing "CHANNEL1".
- "Parameter:" with a text input field containing "COM1,9600,8,N,1,1,1,1".
- "Antwort:" with an empty text input field.
- "Test" button.
- "Abbruch" button.

When the connection between control and Server has been correctly set up, the version of the control software in the control is displayed in the response field Antwort.

Operation

Monitor

The monitor serves for visualizing the internal server statuses. It displays simultaneously a number of information texts for four freely selectable logical channels (see explanation). The monitor dialog for a channel looks as follows



Selection field for the log. channel

Connection to the physical interface (no connection at the moment)

Errors and warnings of the control

Last error
(also see GStatus),
DOS error numbers and
error texts

Last function between
Client and Server

Last function between
Server and control

List of the active DDE functions

Setup

Setting of the communication parameter and the refresh rate. The interface data, adjusted here, are relevant for the test of the connection to the control, but they are also entered into the file DDESVR.INI and serve as interface parameter for Autoinit, also see section 5.2. The refresh rate is the pulse for all cyclic services of Server4 (in ms). This value is dependent on the hardware. A short refresh rate leads to a high system stress. The standard value is at 50 ms. The data of the menu point Setup are saved in the ini-file.

Operation

Licensing

After the installation of Server4 licensing is necessary. The licensing determines the user as the registered user who has correctly acquired the Online-DDE-Server4 and he is now authorized to work with it. A detailed description of the installation and licensing process can be found in the file Liesmich.wri.

Info

Display of the Server4 version.

Items of Server4

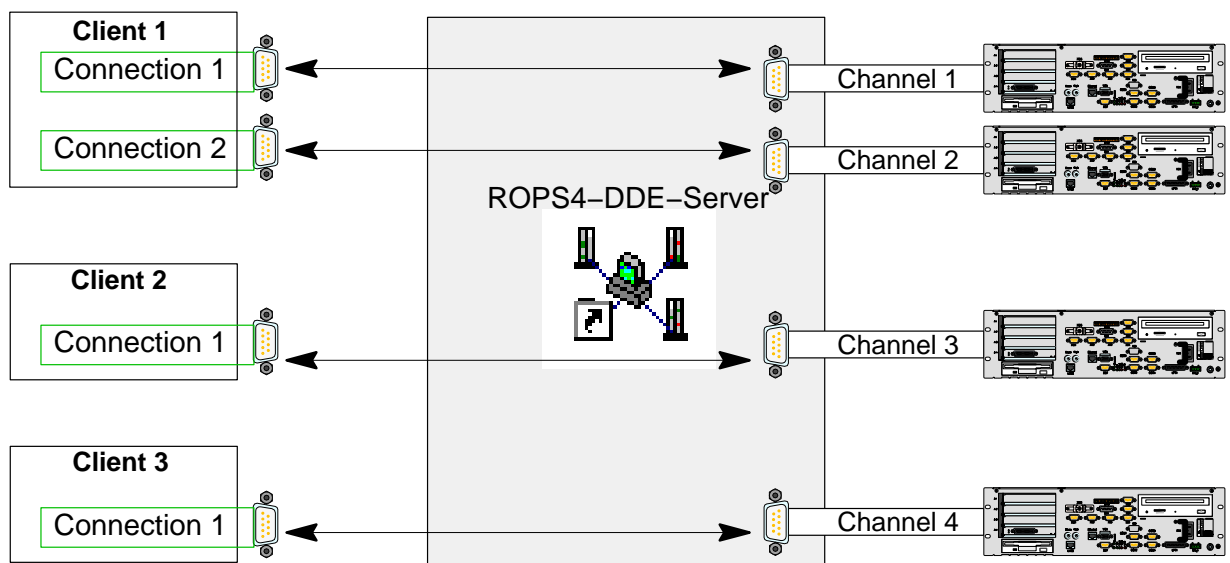
5 Items of Server4

5.1 Connection between Client and Server4

The DDE-Server4 supports 36 interfaces (Com1 to Com4 and 32 TCP/IP connections) and 36 logical channels (Channel 1 to Channel 36) also called Topics. The logical channels Channel 1 to Channel 4 are determined for the connection to serial interfaces while the channels 5 to 36 are designed for TCP/IP connections.

A connection between Client and Server4 is established with a DDE-Connect. Parameter for this Connect are the name of the Server (SERVER4) and the channel or topic name. The connection of the physical interface and the logical channel is established by a DDE-Connect. Only one channel can be assigned per interface and vice-versa.

Server4 can hold simultaneously 36 connections to 36 Clients. A Client that must have connections to several controls, must, however, carry out several DDE-Connects.



In this example, three Clients are connected via four logical channels with four controls. The connection of Server4 and control can be set up via a serial interface or via TCP/IP.

After the DDE-Connect Server4 only supplies five items per channel, see section 5.3.

GStatus	Global status
InitUART	Initialize serial interface
InitTCPIP	Initialize TCP/IP connection
TopicItemList	List of all items available at the moment

Items of Server4

After initialization of the interface, see section 5.2, all items for the selected channel are enabled.

Del	Delete a file in the rho4
Dir	Request directory of rho4
UpLoad	Copy file(s) from rho4 to PC
DownLoad	Copy file(s) from PC to rho4
UpLoad_A	Copy file(s) from rho4 to PC (ASCII log)
DownLoad_A	Copy file(s) from PC to rho4 (ASCII log)
Ren	Rename file in rho4
ADVKinAchsen	Axis info by kinematics cyclic
ADVGlobAchsen	Axis info of all axes cyclic
Werkzeug	Tool cyclic
RK_SYS	WC system cyclic
ProzListe	List of all processes cyclic
ProzStatus	Status of a process cyclic
Signale	Signal display cyclic
FehlerFlag	rho4 error has occurred
Fehler_A	rho4 errors/warnings
Control_Client	Control functions Client/Server4
Control_Server	Control functions Server4/Client
ServerFehler	Error of Servers
A1_POS to A20_POS	Axis positions
A1_ENDPOS to A20_ENDPOS	End positions of axes
A1_INPOS to A20_INPOS	In-position messages of axes
B1_POS to B8_POS	Belt positions
TopicItemList	List of all items available at the moment (this list)
CloseUART	Closing of the serial interface
CloseTCPIP	Closing of the TCP/IP connection
GRDStellung	Starting position RC
KinInfo	Kinematic info rho4
KinAchsen	Axis info by kinematics

Items of Server4

GlobAchsen	Axes info of all axes
Fehler	rho4 error
ProzAnw	Selection of a process
ProzStopp	Stop of a process
SetRCA	Setting the RCO signals 28.0 to 28.7
GStatus	Global status
VarRead1 to VarRead32	Reading of user variables
VarWrite1 to VarWrite32	Writing of user variables
VarRead1_A to VarRead32_A	Reading of user variables (ASCII protocol)
VarWrite1_A to VarWrite32_A	Writing of user variables (ASCII protocol)
Heartbeat	Monitoring of the connection between control and PC


5.2 Connection between PC and control

In order to be able to carry out a data exchange between control and Server4, the interface, by which the PC is connected with the control, must be initialized.

The initialization can be effected in two ways:

- by calling the items of the Server 'InitUART' resp. 'InitTCPIP' and the corresponding parameters, or
- by setting the entry Autoinit=1 in the file DDESVR.INI (also see section 5.4 DDESVR.INI and point 6.1.11. With this kind of initialization, the parameters from the ini-file are used.

The items of the Server are only available after a correct initialization.

 **The complete data traffic with the Server is troubled by accidental interruptions of the connection between control and PC, e.g. by disconnecting the plug connector or by the running up of the RC during data exchange. To facilitate the new communication establishment for the Server, the monitoring service Heartbeat should always be active, see point 6.1.12.**

Items of Server4

5.3 Items

The items of Server4 are divided into four categories.

5.3.1 File administration functions

'Del', 'Dir', 'UpLoad', 'DownLoad', 'UpLoad_A', 'DownLoad_A' and 'Ren'.

Only one of these seven items can be active (per channel). When initializing one of the functions, the six others are deleted from the 'TopicItem-List'. After having terminated the function, all others are added again. Already active cyclic items are stopped for the time of the file transfer function.

5.3.2 Cyclic items

'ADVKinAachsen', 'ADVGlobAachsen', 'Werkzeug', 'RK_SYS', 'ProzListe', 'ProzStatus', 'Signale', 'FehlerFlag', 'Fehler_A', 'Control_Client', 'Control_Server', 'ServerFehler', 'A1_POS' to 'A20_POS', 'A1_ENDPOS' to 'A20_ENDPOS', 'A1_INPOS' to 'A20_INPOS' and 'B1_POS' to 'B8_POS', 'VarRead1' to 'VarRead32', 'VarWrite1' to 'VarWrite32', 'VarRead1_A' to 'VarRead32_A', 'VarWrite1_A' to 'VarWrite32_A' and 'HeartBeat'.

Server4 has an order list for each channel. The cyclic items are entered into this queue during initialization and started by a timer. At each tick of the timer an order from the queue is processed. The active functions are alternating with each other (Round Robin). The timer can be adjusted with the help of the menu point 'Setup', 'Cycle rate'.

5.3.3 Non-cyclic items

'TopicItemList', 'CloseUART', 'CloseTCPIP', 'GRDStellung', 'KinInfo', 'KinAachsen', 'GlobAachsen', 'Fehler', 'ProzAnw', 'ProzStopp', 'SetRCA', 'FehlerFlag', 'Fehler_A', 'Control_Client', 'Control_Server', 'ServerFehler', 'A1_POS' to 'A20_POS', 'A1_ENDPOS' to 'A20_ENDPOS', 'A1_INPOS' to 'A20_INPOS' and 'B1_POS' to 'B8_POS', 'VarRead1' to 'VarRead32', 'VarWrite1' to 'VarWrite32', 'VarRead1_A' to 'VarRead32_A' and 'VarWrite1_A' to 'VarWrite32_A'.

These functions can be called at any time when Server4 is ready for processing a function, i.e. also parallel to cyclic items.

Items of Server4

5.3.4 Items with ASCII log

'UpLoad_A', 'DownLoad_A', 'FehlerFlag', 'Fehler_A', 'Control_Client', 'Control_Server', 'ServerFehler', 'A1_POS' to 'A20_POS', 'A1_END-POS' to 'A20_ENDPOS', 'A1_INPOS' to 'A20_INPOS', 'B1_POS' to 'B8_POS', 'VarRead1_A' to 'VarRead32_A', 'VarWrite1_A' to 'VarWrite32_A' and 'HeartBeat'.

These items communicate with the Client per ASCII string.

5.3.5 Special function GStatus

Each error that occurs and also rho4 error/warning are entered into the GStatus of the corresponding channel. Then the internal error is neutralized (not the rho4 errors/warnings).

When function 'GStatus' is active, Server4 delivers the record, see point 6.1.1, to the Client. This item should always be active to be able to respond to the errors.

Items of Server4

5.4 DDESVR.INI

The Online-DDE-Server4 has an ini-file with the following entries

```
[CHANNEL1]
  INIT=0
  COM=COM1
  BAUD=9600
  DATA=8
  STOP=1
  HANDSHAKE=1
  TIMEOUT=2
  ERRTIMEOUT=300
  PARITY=N
[CHANNEL5]
  TCPALIAS=rho4
  TCPKANAL=6010
  TCPTIMEOUT=10

[DEFEXTENSION]
  //EXT=*.QLL,*.IRD

[SERVERINIT]
  REFRESH=50
  AUTOINIT=0
  KOORDINATEN=1
  LANGUAGE=DEUTSCH
  DISCONNECTMON=1




[ITEMLIMITS]
  ASCIITEMS=1
  BINAERITEMS=1
  READITEMS=32
  WRITEITEMS=32
  ASCIACHSEN=20
```

The values behind the individual parameters are to be regarded as examples. The entry in the group 'CHANNEL1', COM=COM1 determines the connection of the logical channel CHANNEL1 to the COM1 interface. If you want to establish a TCP/IP connection via CHANNEL1, 'COM=TCP/IP' must be entered.

Items of Server4

Section	Description	
[CHANNELx]	Interface data; one section per channel These entries are adjusted via the menu Setup (see in the preceding) Exception: The entries COM.INIT and ERRTIMEOUT must be directly edited in the file DDESVR.INI.	
	Entry	Meaning
	INIT	This entry is only important with AUTOINIT=1. If AUTOINIT is set to 1, the interfaces for which INIT is set to 1 are automatically initialized. With INIT=0 the corresponding channel is ignored during automatic interface initialization.
	COM	Allocation of the physical interface to the logical channel. Possible entries are COM1 to COM4 and TCP/IP. Dependent on these setting the subsequent parameters are read and inserted. Subsequent parameters for a serial interface are e.g. BAUD,DATA,STOP etc and e.g. TCPALIAS for a TCP/IP connection.
	BAUD	Baud rate
	DATA	Number of databits
	STOP	Number of stop bits
	HANDSHAKE	0 = no hardware handshake 1 = hardware handshake A transfer with software handshake is not possible
	TIMEOUT	Timeout time in seconds with intact serial interface
	TCPTIMEOUT	Timeout time in seconds with intact TCP/IP connection
	ERRTIMEOUT	Timeout time in ms with troubled connection. A setting starting from 300 ms is recommended, also see point 6.1.12
	PARITY	Parity monitoring N = No parity monitoring E = Even parity O = Odd parity
TCPALIAS	IP address or alias name of the TCP/IP connection (max. 30 characters)	
TCPKANAL	Channel number (description of rho4-Gateway)	

Items of Server4

Section	Description					
[DEFEXTENSION]	<p>In case of data transfer, deleting and listing of files with wildcards, only those files are selected the extensions of which agree with the filter 'Ext=' of the ini-file. The extension 'bin' (machine parameters) is not considered for transfer and deletion with wildcards.</p> <table border="1"> <thead> <tr> <th>Entry</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>EXT</td> <td> <p>In the example of a file filter shown above, only files with the extension 'qll' and 'ird' are offered for file functions. All other files remain invisible when working with wildcards. If no filter is to be active, i.e. all files with arbitrary extension are permitted, the filter must be set in comment (//EXT=) as shown in the example above.</p> <p> The notation of the filter must be observed (*.ext and semicolon as separator!).</p> </td> </tr> </tbody> </table>		Entry	Meaning	EXT	<p>In the example of a file filter shown above, only files with the extension 'qll' and 'ird' are offered for file functions. All other files remain invisible when working with wildcards. If no filter is to be active, i.e. all files with arbitrary extension are permitted, the filter must be set in comment (//EXT=) as shown in the example above.</p> <p> The notation of the filter must be observed (*.ext and semicolon as separator!).</p>
Entry	Meaning					
EXT	<p>In the example of a file filter shown above, only files with the extension 'qll' and 'ird' are offered for file functions. All other files remain invisible when working with wildcards. If no filter is to be active, i.e. all files with arbitrary extension are permitted, the filter must be set in comment (//EXT=) as shown in the example above.</p> <p> The notation of the filter must be observed (*.ext and semicolon as separator!).</p>					

Section	Description													
[SERVERINIT]	<p>Initialization data of the Servers</p> <table border="1"> <thead> <tr> <th>Entry</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>REFRESH</td> <td> <p>This entry is adjusted via the menu Setup,cycle rate. It indicates, in which cycle cyclic data are transferred (the set value should be more than 50 ms) This value is also decisive for file transfer functions (e.g. Download)</p> </td> </tr> <tr> <td>AUTOINIT</td> <td> <p>= 1: with Connect the interface of this channel is automatically initialized if INIT is set to 1 = 0: automatic initialization switched off</p> </td> </tr> <tr> <td>KOORDINATEN</td> <td> <p>Coordinate selection for axes/belt data that are transferred into ASCII = 0 positions in the coordinate system active at the moment = 1 positions in machine coordinates = 2 positions in room coordinates = 3 positions in original coordinates</p> </td> </tr> <tr> <td>LANGUAGE</td> <td> <p>Selection of language version (GERMAN or ENGLISH)</p> </td> </tr> <tr> <td>DISCONNECTMON</td> <td> <p>Automatic display of the monitor when terminating a connection (Disconnect) = 0 do not indicate monitor at Disconnect = 1 indicate monitor at Disconnect (default)</p> </td> </tr> </tbody> </table>		Entry	Meaning	REFRESH	<p>This entry is adjusted via the menu Setup,cycle rate. It indicates, in which cycle cyclic data are transferred (the set value should be more than 50 ms) This value is also decisive for file transfer functions (e.g. Download)</p>	AUTOINIT	<p>= 1: with Connect the interface of this channel is automatically initialized if INIT is set to 1 = 0: automatic initialization switched off</p>	KOORDINATEN	<p>Coordinate selection for axes/belt data that are transferred into ASCII = 0 positions in the coordinate system active at the moment = 1 positions in machine coordinates = 2 positions in room coordinates = 3 positions in original coordinates</p>	LANGUAGE	<p>Selection of language version (GERMAN or ENGLISH)</p>	DISCONNECTMON	<p>Automatic display of the monitor when terminating a connection (Disconnect) = 0 do not indicate monitor at Disconnect = 1 indicate monitor at Disconnect (default)</p>
Entry	Meaning													
REFRESH	<p>This entry is adjusted via the menu Setup,cycle rate. It indicates, in which cycle cyclic data are transferred (the set value should be more than 50 ms) This value is also decisive for file transfer functions (e.g. Download)</p>													
AUTOINIT	<p>= 1: with Connect the interface of this channel is automatically initialized if INIT is set to 1 = 0: automatic initialization switched off</p>													
KOORDINATEN	<p>Coordinate selection for axes/belt data that are transferred into ASCII = 0 positions in the coordinate system active at the moment = 1 positions in machine coordinates = 2 positions in room coordinates = 3 positions in original coordinates</p>													
LANGUAGE	<p>Selection of language version (GERMAN or ENGLISH)</p>													
DISCONNECTMON	<p>Automatic display of the monitor when terminating a connection (Disconnect) = 0 do not indicate monitor at Disconnect = 1 indicate monitor at Disconnect (default)</p>													

Items of Server4

Section	Description	
[ITEMLIMITS]	Limitation of the Server-items. A limitation to the actually required items can result in shorter reaction times of the Server.	
	Entry	Meaning
	ASCIITEMS	0 = Items with ASCII log locked 1 = Items with ASCII log active
	BINAERITEMS	0 = Items with binary log locked 1 = Items with binary log active
	READITEMS	Number of items for reading user variables
	WRITEITEMS	Number of items for writing user variables
ASCIACHSEN	Number of items for axes positions, in-positions and end positions per ASCII log	

Items of Server4

Notes:

Scope of functions

6 Scope of functions

The Online-DDE-Server4 includes three function groups: Status, File transfer and Online functions.

6.1 Status and initializing functions

With the following functions, Server4 and the connected controls can be monitored and the interfaces can be initialized.

6.1.1 Global status

Function for monitoring Server4 and the connected control. The status can be requested once or cyclically. This status record is also included in each response record.

Request GStatus once

Client	Message 'Item'	Data	↔	Server4
Request status	XTYP_REQUEST 'GStatus'	---	⇒	
		TGSTATUS	⇐	send GStatus

Request GStatus cyclically

Client	Message 'Item'	Data	↔	Server4
Start cyclic status request	XTYP_ADVSTART 'GStatus'	---	⇒	
	TRUE	---	⇐	acknowledge
until stop	XTYP_ADVDATA 'GStatus'	TGSTATUS	⇐	send GStatus
	DDE_FACK 'GStatus'	---	⇒	
Stop status	XTYP_ADVSTOP 'GStatus'	---	⇒	

Scope of functions

Call parameters

none

Return parameters

```

struct TGSTATUS
{
int      nStWarnungen;
int      nStFehler;
int      nFehler;
UINT     nLastDDEError;
/*-----*/
UINT     f3Frei      :3;
UINT     fDOSFehler :1;
UINT     frhoFehler  :1;
UINT     fOnFktFehler :1;
UINT     f9Frei      :9;
UINT     fServerStatus :1;
int      nFc;
int      nState;
char     szItem[50];
WORD     wTransaction;
WORD     wState;
}
    
```

Parameter	Description	
nStWarnungen, nStFehler	Control status; is read from the control with each Online function No update with basic functions	
	Value	Meaning
	-1	Undefined, the control status is unknown
	0	No warnings or errors
	1	Warnings or errors have occurred in the control

Scope of functions

Parameter	Description	
nFehler	Error number see error file ra_err.h	
nLastDDEError	Last DDE error see error file ra_err.h	
	Bit	Meaning
	0 to 2	Not yet occupied
	3	Error DOS see nFehler
	4	Error rho4 (during transfer) see nFehler
	5	Error during last Online function
	5 to 14	Not yet occupied
	15	Server status = ready

Parameter	Description	
nFc	Indicates the last Online function carried out	
	Value	Meaning
	-1	Undefined
	1	Dir
	2	Copy PC ⇒ RC
	3	Copy RC ⇒ PC
	4	Rename
	5	Delete
	1003	Find process
	1005	Find next process
	1007	Process selection
	1010	Position KinX
	1011 or 1044	Kinematics info
	1013	Error
	1016	Version
	1022	Process stop
	1023	Set RCO
	1030	Signals
	1031	Position rho4
	1034	Basic position RC
	1037	List of processes
	1042	Tool
	1045	Write/read BAPS variable

Scope of functions

Parameter	Description	
nState	Indicates the transaction status of item szItem; the value only serves for internal purposes	
	Value	Meaning
	0	Ready
	1	Init
	2	Running
	3	Stop
	4	Waiting for stop
5	Abort	

Parameter	Description
szItem	Name of last item
wTransaction	Last DDE command

The flags f3Frei to wState are only important for diagnosis purposes, they need not be evaluated in normal operation.

Each occurring error, also rho4-errors/warnings, is entered into GStatus of the corresponding Channels. Then the internal error is cleared (not the rho4-errors/warnings). With an activated GStatus function, Server4 now supplies the record TGSTATUS to the Client.

6.1.2 Control and check possibilities for ASCII items

These functions are determined for the control and check of items exchanging their data via ASCII-strings.

Server control

Via this item, Server4 can inform the Client about the status of other items.

Request Control_Server once

Client	Message 'Item'	Data	↔	Server4
Request Control_Server	XTyp_REQUEST 'Control_Server'	---	⇒	
		szServerControl	⇐	send Control_Server

Scope of functions

Request Control_Server cyclically

Client	Message 'Item'	Data	↔	Server4
Start cyclic Server control	XTYP_ADVSTART 'Control_Server'	---	⇒	
	TRUE	---	←	acknowledge
until stop	XTYP_ADVDATA 'Control_Server'	szServerControl	←	send Control_Server
	DDE_FACK 'Control_Server'	---	⇒	
Stop status	XTYP_ADVSTOP 'Control_Server'	---	⇒	

Call parameters

none

Return parameters

char szServerControl[_MAX_STRING];

Parameter	Description
szServerControl	Byte 1: Bit 0: 1 = error/warning of rho Bit 1: 1 = Server error occurred Bit 2: 1 = UpLoad_A terminated Bit 3: 1 = DownLoad_A terminated Bit 4 to 7: reserve Byte 2 to 4: reserve

Server4 supplies control data only in case of a change. Bits 0 and 1 are preoccupied with 0, bits 2 and 3 with 1. A timeout of the interface is also detected if no further item is active.

Client control

Via this item, the Client can indirectly influence active items of Server4.

Client	Message 'Item'	Data	↔	Server4
Start cyclic Client control	XTYP_ADVSTART 'Control_Client'	---	⇒	
	TRUE	---	←	acknowledge
Control Server item	XTYP_POKE 'Control_Client'	szClientControl	⇒	acknowledge
	DDE_FACK 'Control_Client'	---	←	
Stop status	XTYP_ADVSTOP	---	⇒	
	'Control_Client'			

XTYP_ADVSTART or XTYP_ADVSTOP are not necessarily required


Scope of functions

Call parameters

char szClientControl[_MAX_STRING];

Parameter	Description
szClientControl	Byte 1: Bit 0: 1 = UpLoad_A abort Bit 1: 1 = DownLoad_A abort Bit 2: 1 = numb. axis/belt numb. (ASCII) stop 0 = numb. axis/belt numb. (ASCII) start Bit 3: 1 = ServerFehler stop 0 = ServerFehler restart Bit 4: 1 = Fehler_A stop 0 = Fehler_A restart Bit 5: 1 = FehlerFlag stop 0 = FehlerFlag restart Bit 6: 1 = cycl. reading of user variables stop 0 = cycl. reading of user variables restart Bit 7: reserve Byte 2 to 4: reserve

All functions that can be disconnected are initialized as active (bit = 0).

 **During transfer to Server4 the statuses of all bits are interpreted as a matter of principle. The Client must administrate the statuses of disconnected functions itself.**

Return parameters

none

Scope of functions

6.1.3 Server error

Function of control of Server4, of the DOS and Online functions.

Request ServerFehler once

Client	Message 'Item'	Data	↔	Server4
Request ServerFehler	XTYP_REQUEST 'ServerFehler'	---	⇒	
		szServerFehler	⇐	send ServerFehler

Request ServerFehler cyclically

Client	Message 'Item'	Data	↔	Server4
Start cyclic status request	XTYP_ADVSTART 'ServerFehler'	---	⇒	
	TRUE	---	⇐	acknowledge
until	XTYP_ADVDATA 'ServerFehler'	szServerFehler	⇐	send ServerFehler
stop	DDE_FACK 'ServerFehler'	---	⇒	
Stop status	XTYP_ADVSTOP 'ServerFehler'	---	⇒	

Call parameters

none

Return parameters

char szServerFehler[60];

Parameter	Description
szServerFehler	Error number; ASCII string with closing '\0' ¹⁾ The item supplies '0\0' if there is no Server error

Server4 supplies Server errors only in case of a change. The transfer of Server errors can temporarily be stopped by setting a control bit in the 'Control_Client' function.

 **The item 'ServerFehler' does not require data from the control and also has no access to the interface PC/control. A time-out of the interface can therefore not be detected if no item is active that requires this connection²⁾.**

¹⁾ Meaning: see error file ra_err.h

Scope of functions

2) See item 'Control server' in section 6.1.2, control and check possibilities for ASCII items

6.1.4 Report of control status messages/warnings

Function for monitoring the connected control.

Request FehlerFlag once

Client	Message 'Item'	Data	↔	Server4
Request FehlerFlag	XTYP_REQUEST 'FehlerFlag'	---	⇒	
		szFehlerFlag	⇐	send FehlerFlag

Request FehlerFlag cyclically

Client	Message 'Item'	Data	↔	Server4
Start cyclic status request	XTYP_ADVSTART 'FehlerFlag'	---	⇒	
	TRUE	---	⇐	acknowledge
until stop	XTYP_ADVDATA 'FehlerFlag'	szFehlerFlag	⇐	send FehlerFlag
	DDE_FACK 'FehlerFlag'	---	⇒	
stop status	XTYP_ADVSTOP 'FehlerFlag'	---	⇒	

Call parameters

none

Return parameters

char szFehlerFlag[60];

Parameter	Description						
szFehlerFlag	Control status; ASCII string with closing '\0'; Update only in case of status change						
	<table border="1"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No error and no warning</td> </tr> <tr> <td>1</td> <td>Error and/or warning</td> </tr> </tbody> </table>	Value	Meaning	0	No error and no warning	1	Error and/or warning
	Value	Meaning					
0	No error and no warning						
1	Error and/or warning						

Server4 supplies the FehlerFlag only in case of a change. The transfer of the Server status can temporarily be stopped by setting a control bit in the 'Control_Client' function.

Scope of functions

6.1.5 List of all control errors/warnings

The function supplies errors and warnings of the rho4.

Supplied are

- Number of current warnings
- Number of current errors
- Error code
- Error text in ASCII with kinematics or axis reference

The Client can decide whether it takes the error texts from the control or from an ASCII file. In this file, an ASCII text is allocated to each error code. The file can be edited by the user. It is thus possible to enter own error texts and supplementary notes. The file is named 'Fehler.txt' or 'Error.txt'. The standard file includes the texts from the manual 'Status messages and warnings'. A maximum of 20 (_MAX_FEHLER) error and warning texts and error codes are supplied by the control. The real number of warnings and errors in the control can be higher.

Syntax of the error file

The file is structured as follows

Syntax	Description
No = Text	Text is copied to szFehMsg (TDDEFEHLER)
PHG display: 'text'	Text is not copied
Cause: Cause	Text is copied to szUrsache (TDDEFEHLER)
Note: Note	Text is copied to szHinweis (TDDEFEHLER)

Example of an entry in the file Fehler.txt

1 = in Automatic Programmed kinematics is in manual mode
 PHG display 'inadmiss. in manual op.'
 Cause The corresponding kinematics is in MANUAL
 Note Set to AUTOMATIC

Client	Message 'Item'	Data	↔	Server4
Initialize error request	XTYP_POKE 'Fehler'	nModus	⇒	
	DDE_FACK 'Fehler'	---	←	acknowledge
Request error	XTYP_REQUEST 'Fehler'	---	⇒	
		TDDEFEHLER	←	send error

Scope of functions

Call parameters

int nModus;

Parameter	Description	
nModus	Display mode	
	Value	Meaning
	0	The error texts from the control
	1	The error texts from the 'Fehler.txt' file
	2	The error texts from the file 'Error.txt' file

Return parameters

```

struct TDDEFehler
{
    TGSTATUS GStatus;
    int      nAnzWarnungen;
    int      nAnzLaufzeitFehler;
    int      reserved;
    int      nFehKode[_MAX_FEHLER];
    char     szFehMsg[_MAX_FEHLER][_MAX_FEH_LEN];
    char     szUrsache[_MAX_FEHLER][_MAX_FEH_LEN];
    char     szHinweis[_MAX_FEHLER][_MAX_FEH_LEN];
};

```

Parameter	Description
GStatus	Global status, see point 6.1.1
nAnzWarnungen	Number of warnings that have occurred in the control
nAnzLaufzeitFehler	Number of run time errors that have occurred in the control
nFehKode	Codes of warnings and errors in the signal description
szFehMsg[]	Associated error texts
szUrsache[]	Associated texts from the error file; only with mode 2/3 szHinweis[]

Scope of functions

6.1.6 List of all control errors/warnings in ASCII

This function supplies the codes of all errors and warnings of the rho4 as ASCII string. Max. 20 (`_MAX_FEHLER`) codes are supplied from the control. The real number of warnings and errors in the control can be higher.

Request Fehler_A once

Client	Message 'Item'	Data	↔	Server4
Request Fehler_A	XTYP_REQUEST Fehler_A	---	⇒	
		szFehler	⇐	send Fehler_A

Cyclical request Fehler_A

Client	Message 'Item'	Data	↔	Server4
Start cyclic error request	XTYP_ADVSTART Fehler_A	---	⇒	
	TRUE	---	⇐	acknowledge
until stop	XTYP_ADVDATA Fehler_A	szFehler	⇐	send Fehler_A
	DDE_FACK Fehler_A	---	⇒	
Stop status	XTYP_ADVSTOP Fehler_A	---	⇒	

Call parameters

none

Return parameters

```
char szFehler[_MAX_STRING];    'WarnCode','FehCode,
                                FehCode...\0'
```

Parameter	Description
szFehler	Codes of warnings and errors as described in the signal description. If no errors have occurred, only '\0' is transferred.

Server4 supplies the error codes only in case of a change. The transfer of the error code can temporarily be stopped by setting a control bit in the 'Control_Client' function.

Scope of functions

6.1.7 Initialization of a serial interface

With this function, the serial interface is initialized and all items of this channel are enabled. The UART remains occupied until closing and cannot be used by any other application.

The standard interface parameters are:

'9600', 'N', '8', '1', 'HW-Handshake', 'Timeout=8 sec.'

After initialization, the current status should be determined to detect errors that have possibly occurred.

Client	Message 'Item'	Data	↔	Server4
Initialize interface	XTYP_POKE 'InitUART'	TUART	⇒	
	DDE_FACK 'InitUART'	---	⇐	acknowledge
Request status	XTYP_REQUEST 'GStatus'	---	⇒	
		TGSTATUS	⇐	send GStatus

Call parameters

```
struct TUART
{
    int      nConNo;
    int      nBaud;
    char     cParity;
    int      nDatenBits;
    int      nStopBits;
    int      nHandShake;
    int      nTimeOut;
}
```

Parameter	Description
nComNo	Indicates number of interface (1 to 4)
nBaud	Baud rate (110, 300, 1200, 4800, 9600, 19200)
cParity	Parity (N, E, O)
nDatenBits	Data bits (7, 8)
nStopBits	Stop bit (1, 2)
nHandShake	Handshake (0= no handshake, 1= hardware handshake)
nTimeOut	Timeout in sec. (1 to 99)

Return parameters

none

Scope of functions

6.1.8 Initialization of a TCP/IP connection

With this function, a TCP/IP connection is initialized and all items of this channel are enabled.

The standard parameters are

```
TCPALIAS      = rho4
TCPKANAL      = 6010
TCPTIMEOUT    = 10
```

After initialization, the current status should be determined to detect errors that have possibly occurred.

Client	Message 'Item'	Data	↔	Server4
Initialize interface	XTYP_POKE 'InitTCPIP'	TTCP	⇒	
	DDE_FACK 'InitTCPIP'	---	⇐	acknowledge
Request status	XTYP_REQUEST 'GStatus'	---	⇒	
		TGSTATUS	⇐	send GStatus

Call parameters

```
struct TTCP
{
    long    ChannelId;
    int     nComNo;
    int     nTimeOut;
    char    Rho4AliasName[MAX_ALIAS_NAME];
    u_short Rho4PortNo;
}
```

Parameter	Description
ChannelId and nComNo	Internal parameters; must not be set!
nTimeOut	Timeout in sec.
Rho4AliasName	Alias name or IP number of the TCP/IP connection.
Rho4PortNo	Channel number, see section 'Gateway' in the manual system description.

Return parameters

none

Scope of functions

6.1.9 Closing of a serial interface

With this function, a serial interface is closed and the UART is enabled again. Simultaneously, all cyclic functions of this topic are deleted.

Only five items remain available to this channel after closing:

'GStatus', 'initUART', 'initTCPIP', 'Formats' and 'TopicItemList'.

Client	Message 'Item'	Data	↔	Server4
Reset	XTYP_POKE 'CloseUART'	TCOMNO	⇒	
Server	DDE_FACK 'CloseUART'	---	⇐	acknowledge
Request	XTYP_REQUEST 'GStatus'	---	⇒	
status		TGSTATUS	⇐	send GStatus

Call parameters

```
int    nComNo;
```

Parameter	Description
nComNo	Indicates the number of the interface (1 to 4)

Return parameters

none

6.1.10 Closing of a TCP/IP connection

With this function, a TCP/IP connection is closed and the channel is enabled. Simultaneously, all cyclic functions of this topic are deleted.

Only five items remain available for this channel after closing:

'GStatus', 'initUART', 'initTCPIP', 'Formats' und 'TopicItemList'

Client	Message 'Item'	Data	↔	Server4
Reset	XTYP_POKE 'CloseTCPIP'	TTCP	⇒	
Server	DDE_FACK 'CloseTCPIP'	---	⇐	acknowledge
Request	XTYP_REQUEST 'GStatus'	---	⇒	
status		TGSTATUS	⇐	send GStatus

Scope of functions

Call parameters

```
struct TTCP
{
    long    ChannelId;
    int     nComNo;
    int     nTimeOut;
    char    Rho4AliasName[MAX_ALIAS_NAME];
    u_short Rho4PortNo;
}
```

Parameter	Description
ChannelId and nComNo	Internal parameters; must not be set!
nTimeOut	Of no importance when closing a connection
Rho4AliasName	Alias name or IP number of the TCP/IP connection
Rho4PortNo	Channel number (see description of the rho4-Gateway)

6.1.11 Automatic initialization

The interfaces of Server4 can be automatically initialized by certain remarks in the file DDESVR.INI. To do so, 'AUTOINIT'=1 must be entered in the group [SERVERINIT]. During a CONNECT, the corresponding interface is then initialized automatically together with the values of the associated group ([CHANNEL 1] to [CHANNEL 36]) entry of which 'INIT' is set to 1, also see section 5.4.

6.1.12 Monitoring of the connection between control and Server4

This item monitors the serial connection between control and Server4. If an error in data transfer occurs concerning a cyclic item (besides 'Heartbeat' itself), as e.g. SW timeout, Overrun error or similar, the monitoring supplies a count continually incrementing itself. After trouble-shooting, the item supplies once the count 0.

The item Heartbeat should always be active in order to facilitate the new initialization of its items for Server4 if, due to a trouble (e.g. run-up of the control or connection interruption RC \leftrightarrow PC), the data traffic had been interrupted.

As Heartbeat is only activated in case of a trouble, the item normally does not at all affect the runtime behaviour of Server4.

Scope of functions

6.2 File transfer functions

Seven functions are available for the handling of files.

By initializing one of these functions, all file transfer functions of this channel (Topic) are simultaneously disabled, Server4 answers with 'DDE_Fnotprocessed' when initialization is attempted.

In case of a file transfer with wildcards only those files are selected which correspond to the setting WildcardExt= of the Inifile, also see section 5.4. The file extension '.bin' (machine parameters) is not transferred when loading with wildcards.

6.2.1 Download

With this function, the Client can load files into the control. For initialization, the Client transfers the file name to Server4. The file name can contain wildcards.

With the start of the cyclical request, the data transfer starts. During transfer, Server4 reports always after 200 bytes, the total number of the bytes transferred to the Client. The end of a transfer is reported to the Client with nStatus=2. If the order contains more files, the next transfer starts. The number of the remaining files to be transferred is displayed in dwCounter.

The Client can at any time stop the data transfer by sending XTYP_ADVSTOP DownLoad. When an error occurs in the download phase, it is displayed with nStatus=-1 and the order is stopped.

Client	Message 'Item'	Data	↔	Server4
Initialize transfer	XTYP_POKE 'DownLoad'	TCALLDOWNLOAD	⇒	
	DDE_FACK 'DownLoad'	---	⇐	acknowledge
Start cyclic request	XTYP_ADVSTART 'DownLoad'	---	⇒	
	TRUE	---	⇐	acknowledge
until end of file, error or stop until all files are transferred	XTYP_ADVDATA 'DownLoad'	TUPLOADRET	⇐	transfer file(s)
	DDE_FACK 'DownLoad'	---	⇒	
Stop transfer	XTYP_ADVSTOP 'DownLoad'	---	⇒	

Scope of functions

Call parameters

```

struct TCALLDOWNLOAD
{
char      szSRCName[_MAX_DOSNAME];
char      szDSTName[_MAX_RHONAME];
int       nUeberschreiben;
}

```

Parameter	Description						
szSRCName	<p>Complete file name (drive,path,name,ext) of the file to be transferred. Name and file extension can be replaced by wildcards (*').</p> <p>Control file name after Download. Name and file extension can be replaced by wildcards (*'). The file name need not be the same as in szSRCName; the file extension must be the same.</p> <p>Overwrite rho-file; this parameter can have one of the following values:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>File is not overwritten. If file is available, the process is stopped.</td> </tr> <tr> <td>1</td> <td>File is overwritten.</td> </tr> </tbody> </table>	Value	Meaning	0	File is not overwritten. If file is available, the process is stopped.	1	File is overwritten.
Value		Meaning					
0		File is not overwritten. If file is available, the process is stopped.					
1	File is overwritten.						
szDSTName							
nUeberschreiben							

Return-Parameter

```

struct TUPLOADRET
{
TGSTATUS  GStatus;
char      szActName[_MAX_DOSNAME];
int       nStatus;
DWORD     dwCounter;
int       nAnzDateien;
}

```

Scope of functions

Parameter	Description										
GStatus	Global status, see point 6.1.1.										
szActName	The rho4 name										
nStatus	Transfer status; this parameter can have one of the following values:										
	<table border="1"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>File transfer initialized, counter = file length</td> </tr> <tr> <td>1</td> <td>File transfer is running, counter = number of transferred bytes</td> </tr> <tr> <td>2</td> <td>File transfer completed, counter = file length</td> </tr> <tr> <td>-1</td> <td>Error see point 6.1.3.</td> </tr> </tbody> </table>	Value	Meaning	0	File transfer initialized, counter = file length	1	File transfer is running, counter = number of transferred bytes	2	File transfer completed, counter = file length	-1	Error see point 6.1.3.
Value	Meaning										
0	File transfer initialized, counter = file length										
1	File transfer is running, counter = number of transferred bytes										
2	File transfer completed, counter = file length										
-1	Error see point 6.1.3.										

Parameter	Description
dwCounter	Indicates the number of transferred bytes.
nAnzDateien	Indicates the number of remaining files to be transferred which results from the wildcards. After each transfer this counter is decremented.

6.2.2 ASCII download

The ASCII download behaves in the same way as the download described in the preceding. Only the transfer parameters are ASCII strings.

Download_A with download status message on request

Client	Message 'Item'	Data	↔	Server4
Start download	XTYP_POKE 'Download_A'	szDownload	⇒	
	DDE_FACK 'Download_A'	---	⇐	acknowledge
Request download status	XTYP_REQUEST 'Download_A'	---	⇒	
		szDownloadRet	⇐	send download status

Scope of functions

Parameter	Description										
DestName	Control file name for the Download										
Status	Transfer status; this parameter can have one of the following values:										
	<table border="1"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>File transfer initialized, counter = file length</td> </tr> <tr> <td>1</td> <td>File transfer is running, counter = number of transferred bytes</td> </tr> <tr> <td>2</td> <td>File transfer completed, counter = file length</td> </tr> <tr> <td>-1</td> <td>Error see point 6.1.3.</td> </tr> </tbody> </table>	Value	Meaning	0	File transfer initialized, counter = file length	1	File transfer is running, counter = number of transferred bytes	2	File transfer completed, counter = file length	-1	Error see point 6.1.3.
Value	Meaning										
0	File transfer initialized, counter = file length										
1	File transfer is running, counter = number of transferred bytes										
2	File transfer completed, counter = file length										
-1	Error see point 6.1.3.										

Parameter	Description
Counter	Indicates the number of transferred bytes an, see status
AnzDat	Indicates the number of remaining files to be transferred which results from the wildcards. After each transfer this counter is decremented.

Control or abort of the process can be effected via the function 'Control_Client'. Occurring errors are displayed by 'ServerFehler'.

Scope of functions

6.2.3 Upload

Via this function, the Client can transfer files from the control to the PC.

For initialization, the Client transfers the file name to Server4. The file name can include wildcards. Together with the start of the cyclical request, the file transfer starts.

During transfer, Server4 reports the total number of bytes transferred to the Client, always after 200 bytes. The end of the transfer is reported with nStatus=2 to the Client. If the order includes more files, the next transfer starts. The number of the remaining files to be transferred is displayed in dwCounter.

The Client can at any time stop the data transfer by sending XTYP_ADVSTOP 'Upload'. When an error occurs in the Upload phase, it is displayed with nStatus=-1.

Client	Message 'Item'	Data	↔	Server4
Initialize transfer	XTYP_POKE 'UpLoad'	TCALLUPLOAD	⇒	
	DDE_FACK 'UpLoad'	---	⇐	acknowledge
Start cyclical request	XTYP_ADVSTART 'UpLoad'	---	⇒	
	TRUE	---	⇐	acknowledge
until file end, error or stop until all files are transferred	XTYP_ADVDATA 'UpLoad'	TUPLOADRET	⇐	transfer file(s)
	DDE_FACK 'UpLoad'	---	⇒	
Stop transfer	XTYP_ADVSTOP 'UpLoad'	---	⇒	

Call parameters

```
TCALLUPLOAD
{
char    szSRCName[_MAX_PATH];
char    szDSTName[_MAX_RHONAME];
int     nUeberschreiben;
}
```

Scope of functions

Parameter	Description						
szSRCName	Control file name, name and file extension can be replaced by wildcards (*). The file name must not be the same as in szSRCName, the file extension must be the same.						
szDSTName	Complete file name (drive,path,name,Ext) of the PC file. Name and file extension can be replaced by wildcards (*).						
nUeberschreiben	Overwrite PC file; this parameter can have one of the following values: <table border="1"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>File is not overwritten. If the file is available, the process is stopped.</td> </tr> <tr> <td>1</td> <td>File is overwritten</td> </tr> </tbody> </table>	Value	Meaning	0	File is not overwritten. If the file is available, the process is stopped.	1	File is overwritten
Value	Meaning						
0	File is not overwritten. If the file is available, the process is stopped.						
1	File is overwritten						

Return parameters

```

struct TUPLOADRET
{
    TGSTATUS  GStatus;
    char      szActName[_MAX_DOSNAME];
    int       nStatus;
    DWORD     dwCounter;
    int       nAnzDateien;
}

```

Parameter	Description										
GStatus	Global status, see point 6.1.1.										
szActName	The rho4-Name										
nStatus	Transfer status; this parameter can have one of the following values: <table border="1"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>File transfer initialized, counter = file length</td> </tr> <tr> <td>1</td> <td>File transfer is running, counter = number of transferred bytes</td> </tr> <tr> <td>2</td> <td>File transfer completed, counter = file length</td> </tr> <tr> <td>-1</td> <td>Error see point 6.1.3.</td> </tr> </tbody> </table>	Value	Meaning	0	File transfer initialized, counter = file length	1	File transfer is running, counter = number of transferred bytes	2	File transfer completed, counter = file length	-1	Error see point 6.1.3.
Value	Meaning										
0	File transfer initialized, counter = file length										
1	File transfer is running, counter = number of transferred bytes										
2	File transfer completed, counter = file length										
-1	Error see point 6.1.3.										

Parameter	Description
dwCounter	Indicates the number of transferred bytes, see nStatus
nAnzDateien	Indicates the number of remaining files to be transferred which results from the wildcards. After each transfer this counter is decremented.

Scope of functions

6.2.4 ASCII upload

The ASCII upload behaves in the same way as the upload described above. Only the transfer parameters are ASCII strings.

UpLoad_A with upload status message on request

Client	Message 'Item'	Data	↔	Server4
Start download	XTYP_POKE 'UpLoad_A'	szUpload	⇒	
	DDE_FACK 'UpLoad_A'	---	⇐	acknowledge
Request upload status	XTYP_REQUEST 'UpLoad_A'	---	⇒	
		szUploadRet	⇐	send upload-status

UpLoad_A with cyclical upload status message

Client	Message 'Item'	Data	↔	Server4
Transfer initialized	XTYP_ADVSTART 'UpLoad_A'	---	⇒	
	TRUE	---	⇐	acknowledge
start cyclic request	XTYP_POKE 'UpLoad_A'	szUpload	⇒	
	DDE_FACK 'UpLoad_A'	---	⇐	acknowledge
until file end, error or stop until all files are transferred or abort	XTYP_ADVDATA 'UpLoad_A'	szUploadRet	⇐	send file(s)
	DDE_FACK 'UpLoad_A'	---	⇒	
Stop transfer	XTYP_ADVSTOP 'UpLoad_A'	---	⇒	

Call parameters

```
char szUpload[_MAX_STRING]; 'SourceName, DestName, ü\0'
```


Scope of functions

Parameters	Description						
SourceName	Control file name for the upload. Name and file extension can be replaced by wildcards (*').						
DestName	Complete file name (drive,path,name,ext) of the file to be transferred. Name and file extension can be replaced by wildcards (*').						
ü	Overwrite PC file; this parameter can have the following values						
	<table border="1"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>File is not overwritten. If the file is available, the process is stopped.</td> </tr> <tr> <td>1</td> <td>File is overwritten</td> </tr> </tbody> </table>	Value	Meaning	0	File is not overwritten. If the file is available, the process is stopped.	1	File is overwritten
Value	Meaning						
0	File is not overwritten. If the file is available, the process is stopped.						
1	File is overwritten						

All three components are separated by a comma.

Return parameters

```
char szUpLoadRet[_MAX_STRING];
```

```
'DestName,Status,Counter,AnzDat\0'
```

Parameters	Description										
DestName	PC file name for the upload										
Status	Transfer status; this parameter can have one of the following values:										
	<table border="1"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>File transfer initialized, counter = file length</td> </tr> <tr> <td>1</td> <td>File transfer is running, counter = number of transferred bytes</td> </tr> <tr> <td>2</td> <td>File transfer completed, counter = file length</td> </tr> <tr> <td>-1</td> <td>Error see point 6.1.3.</td> </tr> </tbody> </table>	Value	Meaning	0	File transfer initialized, counter = file length	1	File transfer is running, counter = number of transferred bytes	2	File transfer completed, counter = file length	-1	Error see point 6.1.3.
Value	Meaning										
0	File transfer initialized, counter = file length										
1	File transfer is running, counter = number of transferred bytes										
2	File transfer completed, counter = file length										
-1	Error see point 6.1.3.										

Parameters	Description
Counter	Indicates the number of transferred bytes, see status.
AnzDat	Indicates the number of the remaining files to be transferred that depends on the wildcards. After each transfer this counter is decremented.

A control or abort of the process can be effected via the function 'Control_Client'. Occurring error are displayed by 'ServerFehler'.

Scope of functions

6.2.5 Dir

This function is listing the files of the control.

For initialization, the file name is transferred whereby wildcards are supported. Then the client starts the Directory transfer.

Server4 at firsts sends the version of the control software and then the file names with indication of length and date of the last change. Then the number of files follows after the list and the memory occupied by them. At the end, the free and the occupied memory store is supplied. The Client can interrupt the function at any time.

Client	Message 'Item'	Data	↔	Server4
Initialize Dir request	XTYP_POKE 'Dir'	szDirName	⇒	
	DDE_FACK 'Dir'	---	⇐	acknowledge
Start cyclical request	XTYP_ADVSTART 'Dir'	---	⇒	
	TRUE	---	⇐	acknowledge
until Dir is transferred or stop	XTYP_ADVDATA 'Dir'	TRHO3DIR	⇐	send Dir
	DDE_FACK 'Dir'	---	⇒	
Stop Dir	XTYP_ADVSTOP 'Dir'	---	⇒	

Call parameters

```
char    szDirName[_MAX_RHONAME];
```

Parameter	Description
szDirName	Control file name; name and file extension can be replaced by wildcards (*).

Return parameters

```
struct TRHO3DIR
{
    TGSTATUS  GStatus;
    int       nStatus;
    char      szData[_MAX_RHO3_DIR];
}
```

Scope of functions

Parameter	Description	
GStatus	Global status, see point 6.1.1.	
nStatus	Dir status; this parameter can have one of the following values:	
	Value	Meaning
	1	szData includes the software version and date of the control
	2	szData includes a file name
	3	Reserved
	4	szData includes the number of files
	5	szData includes the memory allocation; end of transfer
	-1	Error see point 6.1.3.

Parameter	Description	
szData	Zero terminated ASCII string. This parameter can have one of the following contents	
	Contents	Format with example
		'12345678901234567890123456789012345678901234567'
	SW version	'rho4 : VO01L 03.02.1998'
	File	'WERKZ .IRD 1012 29.03.95 08:44'
	Numb. files	'1 file occupied 1012 byte.'
	Memory	'122880 bytes out of 124160 free.'

Scope of functions

6.2.6 Rename

With the rename function a file is renamed in the rho4. This function does not support wildcards.

After the rename, the current status should be determined to detect errors that have possibly occurred.

Client	Message 'Item'	Data	↔	Server4
Initialize	XTYP_POKE 'Ren'	TREN	⇒	
Ren request	DDE_FACK 'Ren'	---	⇐	acknowledge
Request	XTYP_REQUEST 'Ren'	---	⇒	
status		TGSTATUS	⇐	send G-Status

Call parameters

```
struct TREN
{
char    szOldName[_MAX_RHONAME];
char    szNewName[_MAX_RHONAME];
int     nUeberschreiben;
}
```

Parameter	Description						
szOldName	Old control file name						
szNewName	New control file name						
nUeberschreiben	Overwrite rho4 file; this parameter can have one of the following values:						
	<table border="1"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>File is not overwritten. If the file is available, the process is aborted.</td> </tr> <tr> <td>1</td> <td>File is overwritten.</td> </tr> </tbody> </table>	Value	Meaning	0	File is not overwritten. If the file is available, the process is aborted.	1	File is overwritten.
Value	Meaning						
0	File is not overwritten. If the file is available, the process is aborted.						
1	File is overwritten.						

Return parameters

none

Scope of functions

6.2.7 Delete

Function for deleting a control file. Wildcards are supported. After initialization and the start of the cyclical request, Server4 reports all deleted files to the Client. The order can be aborted at any time.

Client	Message 'Item'	Data	↔	Server4
Initialize	XTYP_POKE 'Del'	szDelName	⇒	
Del order	DDE_FACK 'Del'	---	⇐	acknowledge
start	XTYP_ADVSTART 'Del'	---	⇒	
cyclical request	TRUE	---	⇐	acknowledge
until	XTYP_ADVDATA 'Del'	TDEL	⇐	send delete response
all files are deleted	DDE_FACK 'Del'	---	⇒	
Stop delete	XTYP_ADVSTOP 'Del'	---	⇒	

Call parameters

```
char    szDelName[_MAX_RHONAME];
```

Parameter	Description
szDelName	Control file name; name and file extension can be replaced by wildcards (*).

Return parameters

```
struct TDEL
{
    TGSTATUS GStatus;
    int      nAnzDateien;
    char     szActName[_MAX_RHONAME];
}
```

Parameter	Description
GStatus	Global status, see point 6.1.1.
nAnzDateien	Number of the remaining files to be deleted
szActName	Name of the last control file deleted

Scope of functions

6.3 Online functions

Functions for the visualisation of the control statuses and for the remote control.

6.3.1 Kinematics info

This function supplies information on all kinematics applied in the control.

Client	Message 'Item'	Data	↔	Server4
Request kinematics info	XTyp_REQUEST'KinInfo'	---	⇒	send KinInfo
		TDDEKININFO	⇐	

Call parameters

none

Return parameters

```

struct TDDEKINDATA
{
char      szKinName[_MAX_KINNAME];
int       nReferenz;
int       nAchsAnzahl;
int       nBandAnzahl;
};

struct TDDEKININFO
{
TGSTATUS  GStatus;
int       nKinAnzahl;
TDDEKINDATA KinArray[_MAX_KIN];
};

```

Scope of functions

Parameter	Description
GStatus	Global status, see point 6.1.1.
nKinAnzahl	Number of applied kinematics
TDDEKINDATA:	
szKinName	Name of kinematics
nReferenz	Indicates whether this kinematics has referenced (TRUE/FALSE)
nAchsAnzahl	Number of axes of this kinematics
nBandAnzahl	Number of belts of this kinematics

6.3.2 Axis positions

With this function, the axis data and belt data of the control can be requested. This data can be requested by kinematics (item KinAachsen) or kinematics-exceeding (item GlobAachsen). Up to max. 24 axes and 16 belts can be displayed simultaneously. There are two procedures available to the Client.

The Client requires the data once and instructs the Server4 to make the data available. For the initialization, it transfers the TGETACHSINFO record. The record describes which axes and belts are to be sent in which form. Then, the axis data can be requested.

Axis information by kinematics

Client	Message 'Item'	Data	↔	Server4
Request axis information by kinematics	XTYP_POKE 'KinAachsen'	TGETACHSINFO	⇒	
	DDE_FACK 'KinAachsen'	---	⇐	acknowledge
	XTYP_REQUEST 'KinAachsen'	---	⇒	send KinAachsen
		TACHSDATEN	⇐	

Axis information exceeding kinematics

Client	Message 'Item'	Data	↔	Server4
Request axis information exceeding kinematics	XTYP_POKE 'GlobAachsen'	TGETACHSINFO	⇒	
	DDE_FACK 'GlobAachsen'	---	⇐	acknowledge
	XTYP_REQUEST 'GlobAachsen'	---	⇒	send GlobAachsen
		TACHSDATEN	⇐	

Scope of functions

Polling of axis data

At first, the Client initializes the cycle by sending the TGETACHSINFO record. Then the cycle is started. Server4 then keeps supplying the axis data until the client completes the order (stop). The Client has the possibility to stop the polling cycle to start e. g. a data transfer.

Client	Message 'Item'	Data	↔	Server4
Initialization	XTYP_POKE 'ADVKinAachsen'	TGETACHSINFO	⇒	
	DDE_FACK 'ADVKinAachsen'	---	⇐	acknowledge
Cycle start	XTYP_ADVSTART 'ADVKinAachsen'	---	⇒	
	TRUE	---	⇐	acknowledge
until stop	XTYP_ADVDATA 'ADVKinAachsen'	TACHSDATEN	⇐	send ADVKinAachsen
	DDE_FACK 'ADVKinAachsen'	---	⇒	
Stop	XTYP_ADVSTOP 'ADVKinAachsen'	---	⇒	

The treatment of the kinematics-exceeding function is made accordingly.

Call parameters

```

struct TGETACHSINFO
{
    int      nFc;
    int      nKinNr;
    int      nKoord;
    int      nAchsAnfang;
    int      nAchsAnz;
    int      nBandAnfang;
    int      nBandAnz;
};
    
```

Parameter	Description										
nFc	<p>Determines the subfunction. This parameter can have one of the following values:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>OM_BAND</td> <td>Supplies the belt position</td> </tr> <tr> <td>OM_NAME</td> <td>Supplies WC or JC names</td> </tr> <tr> <td>OM_STAPOS</td> <td>Supplies axis positions, lag, final point, inpos flag, WC, referenced, auto</td> </tr> <tr> <td>OM_STA-POSBND</td> <td>Supplies OM_STAPOS + belt position</td> </tr> </tbody> </table>	Value	Meaning	OM_BAND	Supplies the belt position	OM_NAME	Supplies WC or JC names	OM_STAPOS	Supplies axis positions, lag, final point, inpos flag, WC, referenced, auto	OM_STA-POSBND	Supplies OM_STAPOS + belt position
Value	Meaning										
OM_BAND	Supplies the belt position										
OM_NAME	Supplies WC or JC names										
OM_STAPOS	Supplies axis positions, lag, final point, inpos flag, WC, referenced, auto										
OM_STA-POSBND	Supplies OM_STAPOS + belt position										

Scope of functions

Parameter	Description										
nKinNr	Number of kinematics at item='KinAchsen', otherwise undefined										
nKoord	Describes the requested coordinate system. This parameter can have one of the following values:										
	<table border="1"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>AUTO_SYS</td> <td>Supplies axis positions in the currently active coordinate system</td> </tr> <tr> <td>JC_SYS</td> <td>Supplies axis positions in the machine coordinates</td> </tr> <tr> <td>WC_SYS</td> <td>Supplies axis positions in world coordinates</td> </tr> <tr> <td>UK_SYS</td> <td>Supplies original coordinates</td> </tr> </tbody> </table>	Value	Meaning	AUTO_SYS	Supplies axis positions in the currently active coordinate system	JC_SYS	Supplies axis positions in the machine coordinates	WC_SYS	Supplies axis positions in world coordinates	UK_SYS	Supplies original coordinates
Value	Meaning										
AUTO_SYS	Supplies axis positions in the currently active coordinate system										
JC_SYS	Supplies axis positions in the machine coordinates										
WC_SYS	Supplies axis positions in world coordinates										
UK_SYS	Supplies original coordinates										

Parameter	Description
nAchsAnf	Determines the first axis
nAchsAnz	Determines the number of desired axes
nBandAnf	Determines the first belt
nBandAnz	Determines the number of desired belts

Return parameters

```

struct TACHSDATEN
{
    TGSTATUS  GStatus;
    int       nAchsAnz;
    char      aszName [_MAX_ACHS][_MAX_ACHS_NAME];
    int       nKoord [_MAX_ACHS];
    int       nInPos [_MAX_ACHS];
    int       nReferiert [_MAX_ACHS];
    int       nAutoHand [_MAX_ACHS];
    float     AchsPos [_MAX_ACHS];
    float     EndPos [_MAX_ACHS];
    float     NachPos [_MAX_ACHS];
    int       nBandAnz;
    char      aszBandName [_MAX_BAND][_MAX_BND_NAME];
    float     BandPos [_MAX_BAND];
};

```

Scope of functions

Parameter	Description
GStatus	Global status, see point 6.1.1.
nAchsAnz	Number of axes
aszName	Coordinate names or axis name
nKoord	Coordinate system of the axis positions (WC, JC, OC)
nInPos	InPos flag, indicates whether the axis is IN POSITION
nReferiert	Indicates whether the axis has referenced
nAutoHand	Indicates whether the kinematics belonging to this axis is in the automatic or manual operation
AchsPos	Indicates the current axis position in WC or JC. This value is not valid for unreferenced WC axes.
EndPos	Indicates the programmed final position. This value is not valid in the manual operation.
NachPos	Indicates the required lag
nBandAnz	Number of belts
aszBandName	Belt name as set in MPP
BandPos	Current belt position

6.3.3 Axis data in ASCII

With this function, the axis and belt data of the control can be requested. Of each axis (max. 24), axis position, end point and InPos flag can be determined. For belts (max. 16), the belt position is available.

Request axis/belt data once

Client	Message 'Item'	Data	↔	Server4
Request axis/belt data	XTYP_REQUEST 'item'	---	⇒	
		szAchsDaten	⇐	send axis/belt data

Request axis/belt data cyclically

Client	Message 'Item'	Data	↔	Server4
Cycle start	XTYP_ADVSTART 'item'	---	⇒	
	TRUE	---	⇐	acknowledge
until stop	XTYP_ADVDATA 'item'	szAchsDaten	⇐	send axis data
	DDE_FACK 'item'	---	⇒	
Stop	XTYP_ADVSTOP 'item'	---	⇒	

Scope of functions

Items

A1_POS to A20_POS	Request of axis positions
A1_ENDPOS to A20_END- POS	Request of axis end positions (only reasonable in automatic operation)
A1_INPOS to A20_INPOS	InPos flag; indicates whether the axis is in position.
B1_POS to B8_POS	Request of belt positions

The number of the items can be limited in the DDESVR.INI.

Call parameters


none

Return parameters

char szAchsDaten[60] '+123456.78\0'

Server4 supplies axis data only when they have changed. The cyclical display of the axis data is interrupted by the file transfer functions (upload, download, etc.). The axis data are supplied in the 6.2 format used in the control.

The transmission of axis/belt data can temporarily be stopped by setting a control bit in the Control_Client function.

 **If the Server4 detects an error (e. g. invalid number of axes) all items containing axis and/or belt information of the channel will be closed. The coordinate system of the axis data can be selected in the DDESVR.INI file ([SERVERINIT] COORDINATES).**

Scope of functions

6.3.4 Tool

This function supplies tool name and tool coordinates cyclically.

Client	Message 'Item'	Data	↔	Server4
Initialization of tool order	XTYP_POKE 'Werkzeug'	nKinNr	⇒	
	DDE_FACK 'Werkzeug'	---	⇐	acknowledge
Start cyclical request	XTYP_ADVSTART 'Werkzeug'	---	⇒	
	TRUE	---	⇐	acknowledge
until stop	XTYP_ADVDATA 'Werkzeug'	TWERKZEUG	⇐	send tool
	DDE_FACK 'Werkzeug'	---	⇒	
Tool stop	XTYP_ADVSTOP 'Werkzeug'	---	⇒	

Call parameters

```
int    nKinNr;
```

Parameter	Description
nKinNr	Number of kinematics the tool of which is to be determined

Return parameters

```
struct TWERKZEUG
{
    TGSTATUS  GStatus;
    char      szWerkName[_MAX_WERKNAME];
    float     Value[_MAX_VALUE];
};
```

Parameter	Description
GStatus	Global status, see point 6.1.1.
szWerkName	Name of the currently selected tool of this kinematics
Value[]	Gripper X, gripper Y, gripper Z, gripper orientation1, gripper orientation2, gripper orientation3

Scope of functions

6.3.5 WC system

This function supplies the WC system cyclically.

Client	Message 'Item'	Data	↔	Server4
Initialization of WC system	XTYP_POKE 'RK_Sys'	nKinNr	⇒	
	DDE_FACK 'RK_Sys'	---	⇐	acknowledge
Start cyclical request	XTYP_ADVSTART 'RK_Sys'	---	⇒	
	TRUE	---	⇐	acknowledge
until stop	XTYP_ADVDATA 'RK_Sys'	TRK_SYSTEM	⇐	send RK_Sys
	DDE_FACK 'RK_Sys'	---	⇒	
WC system stop	XTYP_ADVSTOP 'RK_Sys'	---	⇒	

Call parameters

int nKinNr;

Parameter	Description
nKinNr	Number of kinematics the WC system of which is to be determined

Return parameters

```

struct TRK_SYSTEM
{
  TGSTATUS GStatus;
  float Value[_MAX_VALUE];
}

```

Parameter	Description
GStatus	Global status, see point 6.1.1.
Value[]	Shifting of WC in X direction, shifting of WC in Y direction, Shifting of WC in Z direction, Twisting a by X, twisting b by Y, twisting c by Z

Scope of functions

6.3.6 Process selection

This function selects a process within the control.

Client	Message 'Item'	Data	↔	Server4
Initialization of process selection	XTYP_POKE 'ProzAnw'	TPROZANW	⇒	
	DDE_FACK 'ProzAnw'	---	⇐	acknowledge
Select process	XTYP_REQUEST 'ProzAnw'	---	⇒	
		TPROZSTATUS	⇐	send ProzAnw

Call parameters

```
struct TPROZANWAHL
{
char      szRhoName[_MAX_RHONAME];
int       nPrio;
};
```

Parameter	Description
szRhoName	Control file name
nPrio	Priority of the process

Return parameters

```
typedef char TRHONAME[_MAX_RHONAME];
```

Scope of functions

```

struct TPROZSTATUS
{
  TGSTATUS  GStatus;
  int       nProzFound;
  TRHONAME  szProzName;
  int       nProzArt;
  int       nAnzSubProz;
  int       nProzPrio;
  int       nProzZustand;
  long      ProzFehler;
  char      szFehlerText[_MAX_FEH];
  int       nProzZeile;
  int       nProzSubZeile;
  int       nProzKin;
  int       nProzEbene;
  TRHONAME  szHPName;
};

```

Parameter	Description								
GStatus	Global status, see point 6.1.1.								
nProzFound	Indicates whether the required process is available (TRUE/FALSE)								
szProzName	Name of the process								
nProzArt	Indicates the type of process. This parameter can have one of the following values:								
	<table border="1"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Normal process</td> </tr> <tr> <td>1</td> <td>Permanent process</td> </tr> <tr> <td>2</td> <td>Subprocess</td> </tr> </tbody> </table>	Value	Meaning	0	Normal process	1	Permanent process	2	Subprocess
Value	Meaning								
0	Normal process								
1	Permanent process								
2	Subprocess								

Scope of functions

Parameter	Description																
nAnzSubProz	Indicates the number of subprocesses of this main process																
nProzPrio	Indicates the process priorities																
nProzZustand	Indicates the process status. This parameter can have one of the following values:																
	<table border="1"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Process is waiting</td> </tr> <tr> <td>1</td> <td>Process is ready</td> </tr> <tr> <td>2</td> <td>Process is stopped</td> </tr> <tr> <td>3</td> <td>Process is running</td> </tr> <tr> <td>6</td> <td>Process is jogging</td> </tr> <tr> <td>7</td> <td>Process error</td> </tr> <tr> <td>8</td> <td>Process is interrupted</td> </tr> </tbody> </table>	Value	Meaning	0	Process is waiting	1	Process is ready	2	Process is stopped	3	Process is running	6	Process is jogging	7	Process error	8	Process is interrupted
Value	Meaning																
0	Process is waiting																
1	Process is ready																
2	Process is stopped																
3	Process is running																
6	Process is jogging																
7	Process error																
8	Process is interrupted																

Parameter	Description
ProzFehler	Process error, see list of errors
szFehlerText	Error text in ASCII (only valid when ProzFehler is unequal zero)
nProzZeile	Indicates the currently active qll line
nProzSubZeile	Indicates the qll line of the file to be inserted
nProzKin	Active kinematics of this process
nProzEbene	Main program level
szHPName	External main program

Scope of functions

6.3.7 Process stop

With this function, a process in the control can be stopped.

Following to this function, the current status should be determined to recognize errors that have possibly occurred.

Client	Message 'Item'	Data	↔	Server4
Process stop	XTYP_POKE 'ProzStopp'	szProzName	⇒	
	DDE_FACK 'ProzStopp'	---	⇐	acknowledge
Request status	XTYP_REQUEST 'GStatus'	---	⇒	
		TGSTATUS	⇐	send ProzStopp

Call parameters

```
char szProzName[_MAX_RHONAME];
```

Parameter	Description
szProzName	Name of a main process

Return parameters

none

6.3.8 Process list

This function supplies the list of all processes dynamically.

Client	Message 'Item'	Data	↔	Server4
Start cyclical request	XTYP_ADVSTART 'ProzListe'	---	⇒	
	TRUE	---	⇐	acknowledge
until stop	XTYP_ADVDATA 'ProzListe'	TDDEPROZLISTE	⇐	send ProzListe
	DDE_FACK 'ProzListe'	---	⇒	
Stop ProzListe	XTYP_ADVSTOP 'ProzListe'	---	⇒	

Call parameters

none

Scope of functions

Return parameters

```
typedef char TRHONAME[_MAX_RHONAME];
```

```
struct TPARRAY
```

```
}
```

```
TRHONAME          szProzName;
```

```
unsigned          ProzZustand;
```

```
char
```

```
int              nqllZeile;
```

```
};
```

```
struct TDDEPROZLISTE
```

```
{
```

```
TGSTATUS          GStatus;
```

```
int               nAnzPerm;
```

```
int               nAnzNorm;
```

```
int               nAnzSub;
```

```
int               nAnzErr;
```

```
TPARRAY          ProzArray[_MAX_PROZ];
```

```
};
```

Scope of functions

Parameter	Description																
GStatus	Global status, see point 6.1.1.																
nAnzPerm	Number of permanent processes																
nAnzNorm	Number of normal processes																
nAnzSub	Number of subprocesses																
nAnzErr	Number of faulty processes																
szProzName	Process name; main processes have the file extension '.ird', the corresponding subprocesses have the same names and the file extension '.Sxx', and xx is the number of the subprocess.																
ProzZustand	Indicates the process status. This parameter can have one of the following values: <table border="1" data-bbox="646 705 1516 1081"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Process is waiting</td> </tr> <tr> <td>1</td> <td>Process is ready</td> </tr> <tr> <td>2</td> <td>Process is stopped</td> </tr> <tr> <td>3</td> <td>Process is running</td> </tr> <tr> <td>6</td> <td>Process is jogging</td> </tr> <tr> <td>7</td> <td>Process error</td> </tr> <tr> <td>8</td> <td>Process is interrupted</td> </tr> </tbody> </table>	Value	Meaning	0	Process is waiting	1	Process is ready	2	Process is stopped	3	Process is running	6	Process is jogging	7	Process error	8	Process is interrupted
Value	Meaning																
0	Process is waiting																
1	Process is ready																
2	Process is stopped																
3	Process is running																
6	Process is jogging																
7	Process error																
8	Process is interrupted																

Parameter	Description
nqllZeile	Indicates the currently active qll line

6.3.9 Process status

This function supplies the status of a process cyclically.

Client	Message 'Item'	Data	↔	Server4
Initialize order	XTYP_POKE 'ProzStatus'	szProzName	⇒	
	DDE_FACK 'ProzStatus'	---	⇐	acknowledge
Start cyclical request	XTYP_ADVSTART 'ProzStatus'	---	⇒	
	TRUE	---	⇐	acknowledge
until stop	XTYP_ADVDATA 'ProzStatus'	TPROZSTATUS	⇐	send ProzStatus
	DDE_FACK 'ProzStatus'	---	⇒	
Stop status	XTYP_ADVSTOP 'ProzStatus'	---	⇒	

Call parameters

```
char    szProzName[_MAX_RHONAME];
```

Scope of functions

Parameter	Description
szProzName	Process name; main processes have the file extension .ird, the corresponding subprocesses have the same name and the file extension .Sxx, and xx is the number of the subprocess.

Return parameters

```
typedef char TRHONAME[_MAX_RHONAME];
```

```
struct TPROZSTATUS
{
    TGSTATUS    GStatus;
    int         nProzFound;
    TRHONAME    szProzName;
    int         nProzArt;
    int         nAnzSubProz;
    int         nProzPrio;
    int         nProzZustand;
    long        ProzFehler;
    char        szFehlerText[_MAX_FEH];
    int         nProzZeile;
    int         nProzSubZeile;
    int         nProzKin;
    int         nProzEbene;
    TRHONAME    szHPName;
};
```

Parameter	Description								
GStatus	Global status, see point 6.1.1.								
nProzFound	Indicates whether the required process is available (TRUE/FALSE).								
szProzName	Name of the process								
nProzArt	Indicates the type of process. This parameter can have one of the following values:								
	<table border="1"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Normal process</td> </tr> <tr> <td>1</td> <td>Permanent process</td> </tr> <tr> <td>2</td> <td>Subprocess</td> </tr> </tbody> </table>	Value	Meaning	0	Normal process	1	Permanent process	2	Subprocess
Value	Meaning								
0	Normal process								
1	Permanent process								
2	Subprocess								

Scope of functions

Parameter	Description	
nAnzSubProz	Indicates the number of subprocesses in this main process.	
nProzPrio	Indicates the process priorities.	
ProzZustand	Indicates the process status. This parameter can have one of the following values:	
	Value	Meaning
	0	Process is waiting
	1	Process is ready
	2	Process is stopped
	3	Process is running
	6	Process is jogging
	7	Process error
	8	Process is interrupted

Parameter	Description
ProzFehler	Error of the process, see list of errors
szFehlerText	Error text in ASCII (only valid when ProzFehler is unequal zero)
nProzZeile	Indicates the currently active qll line
nProzSubZeile	Indicates the qll line of the file to be inserted
nProzKin	Active kinematics of this process
nProzEbene	Main program level
szHPName	External main program

Scope of functions

6.3.10 Reset via PG

By means of this function, the command 'Reset via PG' can be implemented, see RC output 27.0 in the manual 'Signal descriptions'.

Client	Message 'Item'	Data	↔	Server4
Implement starting position	XTYP_REQUEST 'GRDStellung'	---	⇒	
		TGSTATUS	⇐	send GStatus

Call parameters

none

Return parameters

```

struct TGSTATUS
{
int      nStWarnungen;
int      nStFehler;
int      nFehler;
UINT     nLastDDEError;
/*-----*/
UINT     f3Frei           :3;
UINT     fDOSFehler      :1;
UINT     frhoFehler      :1;
UINT     fOnFktFehler    :1;
UINT     f9Frei          :9;
UINT     fServerStatus   :1;
int      nFc;
int      nState;
char     szItem[50];
WORD     wTransaction;
WORD     wState;
}

```

Scope of functions

Parameter	Description	
nStWarnungen, nStFehler	Control status; is read from the control with each Online function No update with basis functions	
	Value	Meaning
	-1	Undefined, the control status is unknown
	0	No warnings or errors
	1	In this control, warnings and/or errors have occurred

Parameter	Description	
nFehler	Error number, see error file ra_err.h	
nLastDDEError	Last DDE error, see error file ra_err.h	
	Bit	Meaning
	0 to 2	Not yet allocated
	3	DOS error, see nFehler
	4	rho4 error (during transmission), see nFehler
	5	Error during last Online function
	5 to 14	Not yet allocated
	15	Server status = ready

Scope of functions

Parameter	Description	
nFc	Indicates the Online function implemented last	
	Value	Meaning
	-1	Undefined
	1	Dir
	2	Copy PC ⇒ RC
	3	Copy RC ⇒ PC
	4	Rename
	5	Delete
	1003	Find process
	1005	Find next process
	1007	Select process
	1010	KinX position
	1011 or 1044	Kinematics info
	1013	Error
	1016	Version
	1022	Process stop
	1023	Set RCA
1030	Signals	
1031	rho4 position	
1034	RC starting position	
1037	Process list	
1042	Tool	
1045	Write/read BAPS variable	

Parameter	Description	
nState	Indicates the transaction status of the item; the value only serves internal purposes	
	Value	Meaning
	0	Ready
	1	Init
	2	Running
	3	Stop
	4	Waiting for stop
5	Exit	

Scope of functions

Parameter	Description
szItem	Name of the last item
wTransaction	Last DDE command

The flags f3Frei until wState are only of importance for diagnosis purposes, during the normal operation there is no need for evaluating them.

6.3.11 Set RCO

With this function, the RCO signals 28.0 to 28.7 can be set. Following to this function, the current status should be determined to recognize errors that have possibly occurred.

Client	Message 'Item'	Data	↔	Server4
Set RCA signals	XTYP_POKE 'SetRCA'	SigArray	⇒	
	DDE_FACK 'SetRCA'	---	⇐	acknowledge
Request status	XTYP_REQUEST 'GStatus'	---	⇒	
		TGSTATUS	⇐	send GStatus

Call parameters

int SigArray[8];

Parameter	Description	
SigArray	Describes the setting state of the signals. This parameter can have one of the following values:	
	Value	Meaning
	0	Low
	1	High
	127	Don't care

Return parameters

none

6.3.12 Signal display

Function for the cyclical request of signal statuses. The signals can only be requested in bytes.

Scope of functions

Client	Message 'Item'	Data	↔	Server4
Initialization of order	XTYP_POKE 'Signale'	TMIXEDARRAY	⇒	
	DDE_FACK 'Signale'	---	⇐	acknowledge
Start cyclical request	XTYP_ADVSTART 'Signale'	---	⇒	
	TRUE	---	⇐	acknowledge
until stop	XTYP_ADVDATA 'Signale'	TDDESIGNALE	⇐	send signals
	DDE_FACK 'Signale'	---	⇒	
Stop signals	XTYP_ADVSTOP 'Signale'	---	⇒	

Call parameters

```
struct TMIXED
```

```
{
int      nSigTyp;
int      nSigAdr;
};
```

```
struct TMIXEDARRAY
```

```
{
int      nAnzSignale;
TMIXED  Mixed[_MAX_STATUS_SIGNALE];
}
```

Parameter	Description										
nAnzSignale	Number of signal bytes;										
nSigAdr	Byte number acc. to signal description (standard interface address)										
nSigTyp	Describes the signal type. This parameter can have one of the following values:										
	<table border="1"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>RC outputs</td> </tr> <tr> <td>2</td> <td>RC inputs</td> </tr> <tr> <td>5</td> <td>Digital inputs</td> </tr> <tr> <td>4</td> <td>Digital outputs</td> </tr> </tbody> </table>	Value	Meaning	0	RC outputs	2	RC inputs	5	Digital inputs	4	Digital outputs
Value	Meaning										
0	RC outputs										
2	RC inputs										
5	Digital inputs										
4	Digital outputs										

Scope of functions

Return parameters

```
struct TDDESIGNALE
{
  TGSTATUS   GStatus;
  int        nAnzSignale;
  unsigned   SigArray[_MAX_STATUS_SIGNALE];
  char
};
```

Parameter	Description
GStatus	Global status, see point 6.1.1.
nAnzSignale	Number of signals
SigArray[]	Status of the required signal bytes

Scope of functions

6.4 Access to user variables

Online-DDE-Server4 has the possibility to monitor and edit contents of user variables of any BAPS program. For this it is not important whether the file with the variable contents is on the PC or within the control, whether a process is active or already terminated.

6.4.1 General information

Conditions

To permit the symbolic access to variables, Server4 needs information from the sym file. This file must be available on the PC and Server4 must be told where it is to be found (indication of path).

The ird file, in which the contents of most of the variables is contained, can be filed in both the control and the PC. Which file Server4 is to access is indicated in the corresponding DDE message. If an ird file on the PC is to be accessed, the file must be in the same path as the sym file.

When accessing point variables filed in the point file, the pkt file is required additionally. In this case the user also decides by his DDE message where the file to be accessed is stored (PC or RC). Server4 has simultaneous access to user variables in up to 20 different user files.

 **The Online-DDE-Server4 supports in its items only file names with a length of up to eight characters.**

Admissible variables

Server4 can in principle make use of all user variables the contents of which are filed in the ird or pkt file, i.e. variables which are defined in the main program.

User variables which are not filed by the control in the ird or pkt file, but are only for the running time on the internal ird stack, are not accessible for Server4 and can thus neither be read nor described. This type of variables includes e. g. transfer parameters to subprograms or variables which are directly defined in the subprograms.

Server4 has no access to so-called system variables. These are variables that always exist in each process and which need not be especially declared by the programmer.

The system variables include

Scope of functions

POS, @POS, @MPOS, LIMIT_MIN, LIMIT_MAX, V, VFIX, T, TFIX, A, AFIX, V_PTP, VFIX_PTP, VFACTOR, AFACTOR, WC_SYSTEM, DFACTOR, R_PTP and R.

The current values of these variables are not filed in the ird file, but processed separately by the operating system of the control.

Entry of variable names

Server4 must be transferred the variable name exactly as defined in the BAPS program. Upper and lower cases are treated in the same way, as by the translator.

Name additions, such as kinematics names or components of point variables are separated from the actual variable name by a dot. The entry of wildcards is not permitted.

Example

Access to a point component ('name.Komponente')

p1.a_1 This entry supplies the component 'a_1' of point 'p1'

Access to a point with indication of kinematics ('kinematik.name')

SR6.p1 This entry supplies the value of point 'p1', which belongs to the kinematics 'SR6'

Access to a point component with indication of kinematics ('kinematik.name.Komponente')

SR6.p1.a_1 This entry supplies the value of component 'a_1' of point 'p1', which belongs to the kinematics 'SR6'

When requesting fields, the indices of the individual field dimensions are put in angular brackets. In case of multidimensional fields it is not necessary to indicate all dimensions. Sections of a dimension are separated by a dash. The indication of a section, however, is only allowed to be made once per request, and that only for the dimension indicated last.

Scope of functions

Example

Definition of a two-dimensional field in BAPS

```
ARRAY [1..30] ARRAY [1..10] INTEGER : INT_ARRAY
```

Access to a field variable

```
int_array[1][1]    Supplies a INTEGER value of the field 'int_array'
```

Access to a complete field dimension

```
int_array[1]      Supplies 10 INTEGER values of the field 'int_ar-  
ray' ('int_array'[1][1] until 'int_array'[1][10])
```

Access to the section of a field dimension

```
int_array[1][2-5] Supplies 4 INTEGER values of the field 'int_ar-  
ray' ('int_array'[1][2],; 'int_array'[1][3], 'int_ar-  
ray'[1][4] and 'int_array'[1][5])
```

or

```
int_array[1-2]    Supplies 20 INTEGER values of the field 'int_ar-  
ray'
```

Not admissible are the entries

```
int_array[1-5][2-5] or 'int_array[1-5][2]
```

The indications for the description of variables are made accordingly.

Safety inquiry (CommonID)

Server4 can only guarantee the correctness of the supplied variable contents if the complete information required originates from files created in the same translation process. To ensure this, a so-called COMMON-ID monitoring is made ('COMMON-ID' is an identification number, which is recorded during the translation of each file and indicates at what time this file has been created or edited last).

If the COMMON-ID of the ird, pkt and sym files are not identical, a corresponding error message will be displayed in the GStatus items or at the ServerControl and the item will be stopped.

The COMMON-ID monitoring can also upon request by the Client be switched off. The user, however, should be aware of the consequences. In the worst case, a switching-off can even lead to a destruction of a pkt or ird file.

Scope of functions


6.4.2 Reading of variables

This Server4 item supplies the content of any user variable. Up to max. 32 user variables with altogether 200 bytes of information can be monitored simultaneously.

Possible errors will be communicated by the GStatus item or ServerFehler.

Request variable contents once

Client	Message 'Item'	Data	↔	Server4
Initialize order	XTYP_POKE 'item'	TINITREADWRITE 'INIT_POKE'	⇒	
	DDE_FACK 'item'	---	⇐	acknowledge
Request contents (as often as desired)	XTYP_REQUEST 'item'	---	⇒	
		TREADVARDATA	⇐	send variable content
Complete order	XTYP_POKE 'item'	TEXITREADWRITE 'EXIT_POKE'	⇒	
	DDE_FACK 'item'	---	⇐	acknowledge

 **With 'INIT_POKE', the corresponding files required by Server4 to access the variable are opened. The user has to make sure that an initialized order is correctly terminated with 'EXIT_POKE' since only then Server4 will close all these files and enable its internally required memory location! Max. 200 bytes can be read per item.**

Request variable cyclically

Client	Message 'Item'	Data	↔	Server4
Initialize order	XTYP_POKE 'item'	TINITREADWRITE 'INIT_POKE'	⇒	
	DDE_FACK 'item'	---	⇐	acknowledge
Start cyclical request	XTYP_ADVSTART 'item'	---	⇒	
	TRUE	---	⇐	acknowledge
until stop	XTYP_ADVDATA 'item'	TREADVARDATA	⇐	send variable content
	DDE_FACK 'item'	---	⇒	
Read stop	XTYP_ADVSTOP 'item'	---	⇒	

Scope of functions

- ☞ **When starting the cyclical request, the corresponding files required by Server4 to access the variable, are opened. The user has to make sure that a cyclically initialized order is correctly terminated with XTYP_ADVSTOP since only then Server4 will close all these files, enable its internally required memory location and register the completion of the cyclical item with the DDE management.**
- ☞ **When reading cyclically, all active items are summarized and their contents isochronously requested by the RC (or by the PC). Thus, an isochronous image of the desired variable contents is achieved. For this reason, it is possible to read cyclically altogether max. 200 bytes per channel.**

Items

'VarRead1' until 'VarRead32'

The number of items can be limited in the DDESVR.INI file.

Call parameters for initialization

```
typedef enum { INIT_POKE, EXIT_POKE, DATA_POKE } TPOKESTATUS;
```

```
struct TINITREADWRITE  
{  
    TPOKESTATUS PokeStatus;  
    char        szPath [MAX_DIR];  
    char        szVarName [_MAX_STRING];  
    BOOL        bCommonID;  
    BOOL        bPCRC;  
};
```


Scope of functions

Parameter	Description	
PokeStatus	This date has three statuses (INIT_POKE, EXIT_POKE and DATA_POKE) and helps Server4 to distinguish which type of message a POKE is. During initialization, this date has to be set to 'INIT_POKE'.	
szPath	Complete indication of path and file name (without file extension) of the sym file	
szVarName	Variable name (including possible field indices)	
bCommonID	'COMMON-ID' monitoring	
	Value	Meaning
	0	Monitoring is switched off
	1	Monitoring is active

Parameter	Description	
bPCRC	Read variable from file in the RC or on the PC	
	Value	Meaning
	0	Read variable from file in the RC
	1	Read variable from file on the PC

Call parameters for ending

```
typedef enum {INIT_POKE, EXIT_POKE, DATA_POKE} TPOKESTATUS;
```

```
struct TEXTITREADWRITE
{
    TPOKESTATUS PokeStatus;
}
```

Parameter	Description
PokeStatus	This date has three statuses (INIT_POKE, EXIT_POKE and DATA_POKE) and helps the Server4 to distinguish which type of message a POKE is. When ending, this date has to be set to 'EXIT_POKE'.

Scope of functions

Return parameters

```
struct TBINEA
{
    long      IBinEA;
    long      IKanal;
}
```

```
struct TDEZEA
{
    float     fdezEA;
    long      IKanal;
}
```


```
struct TGANZEA
{
    long      IGanzEA;
    long      IKanal;
}
```

Scope of functions

```
struct TREADVARDATA
{
  TGSTATUS   GStatus;
  int        nGroesse;
  union{
    float    fDez           [50];
    long     IGanz          [50];
    long     IBinaer        [50];
    char     cZeichen       [200];
    char     szText         [200];
    float    fPunkt         [50];
    float    fMKPunkt       [50];
    float    fRKRahmen      [50];
    TBINEA   IBinEingang    [25];
    TBINEA   IBinAusgang    [25];
    TDEZEA   fDezEingang    [25];
    TDEZEA   fDezAusgang    [25];
    TGANZEA  IGanzEingang   [25];
    TGANZEA  IGanzAusgang   [25];
  }Var;
}
```

Scope of functions

Parameter	Description
bBinEA	Status of the binary channel
fdezEA	Status of the REAL channel
IGanzEA	Status of the INTEGER channel
IKanal	Channel number for inputs/outputs
GStatus	Global status, see status functions
nGroesse	Number of transferred bytes
fDez	Content of a type REAL variable
IGanz	Content of a type INTEGER variable
IBinaer	Content of a type BINARY variable
cZeichen	Content of a type CHAR variable
szText	Content of a type TEXT variable
fPunkt	Content of a type POINT variable
fMKPunkt	Content of a type JC_POINT variable
fRKRahmen	Content of a type WC_FRAME variable
IBinEingang	Content and channel number of a binary input
IBinAusgang	Content and channel number of a binary output
IDezEingang	Content and channel number of a REAL input
IDezAusgang	Content and channel number of a REAL output
IGanzEingang	Content and channel number of a INTEGER input
IGanzAusgang	Content and channel number of a INTEGER output

 **In case of undefined points, the Server4 supplies the content 'fffffff'. In case of 'POS' and '@POS', the channel number (long) will also be transferred as the last data.**

In case of a cyclical request, the Server4 supplies the content of the variable only if it has changed. The transfer of the variable content can temporarily be stopped by setting a control bit in the Control_Client function.

Scope of functions


6.4.3 Reading of variables with an ASCII protocol

This Server4 item supplies the content of any user variable. The communication between Client and Server4 is made in ASCII characters. Max. 32 variables with altogether 200 bytes of information can be monitored simultaneously.

Possible errors are indicated by the item GStatus or by ServerFehler.

Request variable once

Client	Message 'Item'	Data	↔	Server4
Initialize order	XTYP_POKE 'item'	szReadVar 'INIT'	⇒	
	DDE_FACK 'item'	---	⇐	acknowledge
Apply for content (as often as desired)	XTYP_REQUEST 'item'	---	⇒	
		szReadVarData	⇐	send variable content
Complete order	XTYP_POKE 'item'	szReadVar 'EXIT'	⇒	
	DDE_FACK 'item'	---	⇐	acknowledge

 **For the initialization, the files required by the Server4 for an access to the variable are opened. The user has to make sure that an initialized order is correctly terminated with EXIT since only then will the Server4 close all these files and enable its internally required memory location! Max. 200 bytes can be read per item.**

Request variable cyclically

Client	Message 'Item'	Data	↔	Server4
Initialize order	XTYP_POKE 'item'	szReadVar 'INIT'	⇒	
	DDE_FACK 'item'	---	⇐	acknowledge
Start cyclical request	XTYP_ADVSTART 'item'	---	⇒	
	TRUE	---	⇐	acknowledge
until	XTYP_ADVDATA 'item'	szReadVarData	⇐	send variable content
stop	DDE_FACK 'item'	---	⇒	
Complete order	XTYP_ADVSTOP 'item'	---	⇒	

Scope of functions

- ☞ **When starting the cyclical request, the corresponding files required by the Server4 to access the variable, are opened. The user has to make sure that a cyclically initialized order is correctly terminated with XTYP_ADVSTOP since only then will the Server4 close all these files, enable its internally required memory location and register the completion of the cyclical item with the DDE management.**

- ☞ **When reading cyclically, all active items are summarized and their contents isochronously requested by the RC (or by the PC). Thus, an isochronous image of the desired variable contents is achieved. For this reason, it is possible to read cyclically altogether max. 200 bytes per channel.**

Item

'VarRead1_A' until 'VarRead32_A'

The number of items can be limited in the DDESVR.INI file.

Call parameters for initialization

char szReadVar[_MAX_STRING]; 'INIT, szPath, szVarName
[,cCommonId,cPCRC]\0'

The indications cCommonId and cPCRC can also be omitted. In this case the tiled values apply.

Parameter	Description	
INIT	Keyword to initialize an order	
szPath	Indication of complete path and file name (without file extension) of the sym file	
szVarName	Variable name (including possible field indices)	
cCommonID	COMMON-ID monitoring (optional)	
	Value	Meaning
	0	Monitoring is switched off
	1	Monitoring is active (tiled)

Parameter	Description	
cPCRC	Read variable from file in the RC or on the PC (optional)	
	Value	Meaning
	0	Read variable from file in the RC (tiled)
	1	Read variable from file on the PC

Scope of functions

Call parameters for ending

```
char szReadVar[_MAX_STRING]; 'EXIT\0'
```

Parameter	Description
EXIT	Keyword to end an order

Return parameters

```
char szReadVarData[_MAX_ASCII_ANS- 'szValue1[,szValue2, WER]; szValue3..]\0'
```

Parameter	Description
szValue1,szValue2...	Content(s) of the variable(s) in ASCII. In case of more than one value (e.g. for points), the individual values are separated by a comma.

Scope of functions

Examples for the structure of an ASCII string

REAL '1.0,-32.66,0,177\0'

INTEGER '10,20,-33,1235\0'

BINARY '1,1,1,0,0,0\0'

CHAR 'x\0'

Particularity of character fields:

Here, the individual characters are not separated by a comma!

TEXT 'ABCDEFghIjKl\0\0'

Particularity of texts and text fields:

A text in BAPS can have max. 80 characters. If a text has less than 80 characters, the remaining characters of the text (up to the maximum size) are filled up by 0's. If a text has a length of 80 characters, the 0 at the end of the text will be omitted.

Server4 always transfers 80 characters per text respectively per field element of a text field.

POINT,JCPOINT, WCFRAME '333.444,-777.44,0.98\0'

Particularity of points:

In case of undefined points, the Server4 will supply the content '--.--\0'

BINARY INPUT, BINARY OUTPUT '1,1,0,2\0'

REAL INPUT, REAL OUTPUT '11.22,201,-44.55,202\0'

INTEGER INPUT, INTEGER OUTPUT '11,401,44,402\0'

Particularity of channels:

In case of channels, always two values are transferred, the first value supplying the channel status resp. KanalValue, while the second value corresponds to the channel number.

In case of a cyclical request, the Server4 supplies the content of the variable only if it has changed. The transfer of the variable content can temporarily be stopped by setting a control bit in the Control_Client function.

Scope of functions

6.4.4 Writing of variables


These items enable the user to change variable contents. After the writing, the current status should be determined (GSTATUS or ServerFehler) to detect any errors that have possibly occurred.

The Online-DDE-Server4 supports with its items only file names up to a length of eight characters. Since a simultaneous access to one and the same variable is possible by the BAPS process of the control and by the Server4, the user programmer has to exclude any possible conflicts. The user programmer is responsible for any unintended responses of the control when writing variables by means of this server function.

Server4 does not check the new values sent by the Client with regard to their validity or value range, it only writes these values directly into the indicated file.

Write variable once


Client	Message 'Item'	Data	↔	Server4
Initialize order	XTYP_POKE 'item'	TINITREADWRITE 'INIT_POKE'	⇒	
	DDE_FACK 'item'	---	⇐	acknowledge
Write content (as often as desired)	XTYP_POKE 'item'	TWRITEVAR 'DATA_POKE'	⇒	send variable content
	DDE_FACK 'item'	---	⇐	acknowledge
Complete order	XTYP_POKE 'item'	TEXITREADWRITE 'EXIT_POKE'	⇒	
	DDE_FACK 'item'	---	⇐	acknowledge

 **With 'INIT_POKE', the corresponding files required by the Server4 to access the variable are opened. The user has to make sure that an initialized order is correctly terminated with 'EXIT_POKE' since only then will the Server4 close all these files and enable its internally required memory location! Max. 200 bytes can be read per item.**

Scope of functions

Write variable cyclically

Client	Message 'Item'	Daten	↔	Server4
Initialize order	XTYP_POKE 'item'	TINITREADWRITE 'INIT_POKE'	⇒	
	DDE_FACK 'item'	---	⇐	acknowledge
Start cyclical description	XTYP_ADVSTART 'item'	---	⇒	
	TRUE	---	⇐	acknowledge
Write content (as often as desired)	XTYP_POKE 'item'	TWRITEVAR 'DATA_POKE'	⇒	send variable content
	DDE_FACK 'item'		⇐	acknowledge
Stop reading	XTYP_ADVSTOP 'item'	---	⇒	

 **When starting the cyclical order, the corresponding files required by the Server4 to access the variable, are opened. The user has to make sure that a cyclically initialized order is correctly terminated with ADV_STOP since only then will the Server4 close all these files, enable its internally required memory location and register the completion of the cyclical item with the DDE management.**

Items

'VarWrite1' until 'VarWrite32'

The number of items can be limited in the DDESVR.INI file.

Call parameters for initialization

```
typedef enum {INIT_POKE, EXIT_POKE, DATA_POKE} TPOKESTATUS;
```

```
struct TINITREADWRITE
{
    TPOKESTATUS PokeStatus;
    char        szPath        [MAX_DIR];
    char        szVarName    [_MAX_STRING];
    BOOL        bCommonID;
    BOOL        bPCRC;
};
```

Scope of functions

Parameter	Description						
PokeStatus	This date has three statuses (INIT_POKE, EXIT_POKE and DATA_POKE) and helps the Server4 to distinguish which type of message a POKE is. During initialization, this date has to be set to 'INIT_POKE'.						
szPath	Complete indication of path and file name (without file extension) of the sym file						
szVarName	Variable name (including possible field indices)						
bCommonID	'COMMON-ID' monitoring						
	<table border="1"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Monitoring is switched off</td> </tr> <tr> <td>1</td> <td>Monitoring is active</td> </tr> </tbody> </table>	Value	Meaning	0	Monitoring is switched off	1	Monitoring is active
Value	Meaning						
0	Monitoring is switched off						
1	Monitoring is active						

Parameter	Description						
bPCRC	Write variable into file in the RC or on the PC						
	<table border="1"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Write variable into file in the RC</td> </tr> <tr> <td>1</td> <td>Write variable into file on the PC</td> </tr> </tbody> </table>	Value	Meaning	0	Write variable into file in the RC	1	Write variable into file on the PC
Value	Meaning						
0	Write variable into file in the RC						
1	Write variable into file on the PC						

Call parameters for ending

```
typedef enum {INIT_POKE, EXIT_POKE, DATA_POKE} TPOKESTATUS;
```

```
struct TEXTITREADWRITE
{
    TPOKESTATUS PokeStatus;
}
```

Parameter	Description
PokeStatus	This date has three statuses (INIT_POKE, EXIT_POKE and DATA_POKE) and helps the Server4 to distinguish which type of message a POKE is. During initialization, this date has to be set to 'INIT_POKE'.

Call parameters to write

```
typedef enum {INIT_POKE, EXIT_POKE, DATA_POKE} TPOKESTATUS;
```

Scope of functions

```
struct TWRITEVAR
{
  TPOKESTATUS PokeStatus;
  int          nGrosse;
  union{
    float      fDez          [50];
    long       IGanz         [50];
    long       IBinaer       [50];
    char       cZeichen      [200];
    char       szText        [200];
    float      fPunkt        [50];
    float      fMKPunkt      [50];
    float      fRKRahmen     [50];
    long       IBinEingang   [50];
    long       IBinAusgang   [50];
    float      fDezEingang   [50];
    float      fDezAusgang   [50];
    long       IGanzEingang  [50];
    long       IGanzAusgang  [50];
  }Var;
}
```

Scope of functions

Parameter	Description
PokeStatus	This date has three statuses (INIT_POKE, EXIT_POKE and DATA_POKE) and helps the Server4 to distinguish which type of message a POKE is. During initialization, this date has to be set to 'INIT_POKE'.
nGroesse	Number of bytes to be written (Example: if a REAL variable is written, nGroesse must be set to 1; if a field of 5 REAL variables is set, nGroesse must be set to 5)
fDez	New content of a type REAL variable
IGanz	New content of a type INTEGER variable
IBinaer	New content of a type BINARY variable
cZeichen	New content of a type CHAR variable
szText	New content of a type TEXT variable
fPunkt	New content of a type POINT variable
fMKPunkt	New content of a type JC_POINT variable
fRKRahmen	New content of a type WC_FRAME variable
IBinEingang	New content of a type BINARY INPUT variable
IBinAusgang	New content of a type BINARY OUTPUT variable
IDezEingang	New content of a type REAL INPUT variable
IDezAusgang	New content of a type REAL OUTPUT variable
IGanzEingang	New content of a type INTEGER INPUT variable
IGanzAusgang	New content of a type INTEGER OUTPUT variable

Return parameters

none

6.4.5 Writing of variables with ASCII protocol

The user can also change variable contents by means of these items. The communication between Client and Server4 is in this case done in ASCII characters. After the writing, the current status should be determined (GSTATUS or ServerFehler) to detect any errors that have possibly occurred.


Since a simultaneous access to one and the same variable is possible by the BAPS process of the control and by the Server4, the user programmer has to exclude any possible conflicts. The user programmer is responsible for any unintended responses of the control when writing variables by means of this server function.

Scope of functions

Server4 does not check the new values sent by the Client with regard to their validity or value range, it only writes these values after their conversion (ASCII ⇒ corresponding format) directly into the indicated file.


Write variable once

Client	Message 'Item'	Data	↔	Server4
Initialize order	XTYP_POKE 'item'	szWriteVar 'INIT'	⇒	
	DDE_FACK 'item'	---	⇐	acknowledge
Write content (as often as desired)	XTYP_POKE 'item'	szWriteVar 'DATA'	⇒	send variable content
	DDE_FACK 'item'	---	⇐	acknowledge
Complete order	XTYP_POKE 'item'	szWriteVar 'EXIT'	⇒	
	DDE_FACK 'item'	---	⇐	acknowledge

 **For the initialization, the files required by the Server4 for an access to the variable are opened. The user has to make sure that an initialized order is correctly terminated with EXIT since only then will the Server4 close all these files and enable its internally required memory location! Max. 200 bytes can be read per item.**

Write variable cyclically

Client	Message 'Item'	Data	↔	Server4
Initialize order	XTYP_POKE 'item'	szWriteVar 'INIT'	⇒	
	DDE_FACK 'item'	---	⇐	acknowledge
Start cyclical request	XTYP_ADVSTART 'item'	---	⇒	
	TRUE	---	⇐	acknowledge
Write content (as often as desired)	XTYP_POKE 'item'	szWriteVar 'DATA'	⇒	send variable content
	DDE_FACK 'item'	---	⇐	acknowledge
Complete order	XTYP_ADVSTOP 'item'	---	⇒	

 **When starting the cyclical request, the corresponding files required by the Server4 to access the variable, are opened. The user has to make sure that a cyclically initialized order is correctly terminated with XTYP_ADVSTOP since only then will the Server4 close all these files, enable its internally required memory location and register the completion of the cyclical item with the DDE management.**

Item

'VarWrite1_A' until 'VarWrite32_A'

The number of items can be limited in the DDESVR.INI file.

Scope of functions

Call parameters for initialization

```
char szWriteVar[_MAX_STRING]; 'INIT, szPath, szVarName
                               [, cCommonId, cPCRC]0'
```

The indications cCommonId and cPCRC can also be omitted. In this case the tiled values apply.

Parameter	Description	
INIT	Keyword to initialize an order	
szPath	Indication of complete path and file name (without file extension) of the sym file	
szVarName	Variable name (including possible field indices)	
cCommonID	COMMON-ID monitoring (optional)	
	Value	Meaning
	0	Monitoring is switched off
	1	Monitoring is active (tiled)

Parameter	Description	
cPCRC	Write variable into file in the RC or on the PC (optional)	
	Value	Meaning
	0	Write variable into file in the RC (tiled)
	1	Write variable into file on the PC

Call parameters for ending

```
char szWriteVar[_MAX_STRING]; 'EXIT0'
```

Parameter	Description
EXIT	Key word for ending an order

Call parameters to write

```
char szWriteVarData[_MAX_ASCII_ANS- 'DATA, szValue1
WER];                               [, szValue2,
                                     szValue3..]0'
```

Parameter	Description
DATA	Key word to send new values
szValue1, szValue2...	New content(s) of the variable(s) in ASCII. In case of more than one value (e.g. in case of points), the individual values are separated by a comma.

Scope of functions

Examples for the structure of the ASCII string to send new values

REAL	'DATA,1.0,-32.66,0,177\0'
------	---------------------------

INTEGER	'DATA,10,20,-33,1235\0'
---------	-------------------------

BINARY	'DATA,1,1,1,0,0,0\0'
--------	----------------------

CHAR	'DATA,x\0'
------	------------

Particularity of character fields:

Here, the individual characters are not separated by a comma!

TEXT	'DATA,ABCDEFGHIJKO\0'
------	-----------------------

Particularity of texts and text fields:

A text in BAPS can have max. 80 characters. If a text has less than 80 characters, the remaining characters of the text (up to the maximum size) are filled up by 0's. If a text has a length of 80 characters, the 0 at the end of the text will be omitted.

Server4 always transfers 80 characters per text respectively per field element of a text field.

POINT,JCPOINT, WCFRAME	'DATA,333.444,-777.44,0.98\0'
---------------------------	-------------------------------

Particularity of points:

The description of a point with '---.--\0' (undefined) is not possible.

BINARY INPUT, BI- NARY OUTPUT	'DATA,1,0\0'
----------------------------------	--------------

REAL INPUT, REAL OUTPUT	'DATA,11.22,-44.55\0'
----------------------------	-----------------------

INTEGER INPUT, INTEGER OUT- PUT	'DATA,11,44\0'
---------------------------------------	----------------


Return parameters

none	
------	--

Scope of functions

6.4.6 Example

A plant with a control can make one product in four different designs. The number of the desired product and its design are entered in the PC (any Client surface) and sent via DDE to the Online-Server4. The sequence program in the control receives the data from the Server4 and initiates the production of the desired parts.

 **Additional user examples for a Client programming under ACCESS, EXCEL and WORD are to be found in the directory C:\Bosch\ddesvr4\Example ...**

The principal sequence program

```
;;CONTROL = rho4

;;KINEMATICS: (1=SR6)

PROGRAM prod

;*****

;Variables described by the Client

;*****

INTEGER:   auftrag           ; product design
INTEGER:   anzahl           ; desired number of products
INPUT BINARY: 1 = startsig   ; start signal -> assemble desired product(s)
;*****

;Variables read by the Client

;*****

BINARY: fprodukt           ; unknown design
BINARY: fanzahl           ; wrong number
OUTPUT BINARY: 1 = endesig  ; end signal -> product(s) assembled
INTEGER:   summe1          ; total product design 1
INTEGER:   summe2          ; total product design 2
INTEGER:   summe3          ; total product design 3
INTEGER:   summe4          ; total product design 4
```

Scope of functions

```
BEGIN

loop:

;*****

;Wait until start signal from Client is received

;*****

    WAIT UNTIL startsig = 1

;*****

;Initialization and monitoring of the values from the Client

;*****

fprodukt = 0

endesig = 0

IF anzahl < 0 THEN BEGIN      ; check number

    fanzahl = 1

    JUMP loop

END

ELSE fanzahl = 0

;*****

;Branch according to order

;*****

CASE auftrag

    EQUAL 1:                  ; finished design 1

        BEGIN

            REPEAT anzahl TIMES

                prod1          ; subroutine assembles product 1

                summe1 = summe1 + 1

            REPEAT_END

        END
```


Scope of functions

```
;*****  
;Subroutines for product assembly  
;*****
```

```
;production flow design 1
```

```
SUBROUTINE prod1
```

```
  BEGIN
```

```
    ; .
```

```
    ; .
```

```
    ; .
```

```
  SUB_END
```

```
;production flow design 2
```

```
SUBROUTINE prod2
```

```
  BEGIN
```

```
    ; .
```

```
    ; .
```

```
    ; .
```

```
  SUB_END
```

```
;production flow design 3
```

```
SUBROUTINE prod3
```

```
  BEGIN
```

```
    ; .
```

```
    ; .
```

```
    ; .
```

```
  SUB_END
```

```
;production flow design 4
```

```
SUBROUTINE prod4
```

```
  BEGIN
```

```
    ; .
```

```
    ; .
```

```
    ; .
```

```
  SUB_END
```

Scope of functions

Flow of Client/Server operation

Starting position

- Server4 has started and is running
- The sym file Prod.sym is filed in the path c:\projekt of the PC
- The Client has already contacted the Server4
- In the RC the PROD process has been selected and started

Initialization (start of cyclical items) to read variables

Reading of error inquiry for wrong design number, Server item: Var-Read1

Variable in BAPS program: FPRODUKT

Data struct for the transfer to the Server4: TINITREADWRITE

Content of the struct elements

PokeStatus	INIT_POKE
szPath	'c:\projekt\prod\0'
szVarName	'FPRODUKT\0'
bCommonID	1
bPCRC	0

Reading of error inquiry for wrong number of products

Server item: VarRead2

Variable in BAPS program: FANZAHL

Data struct for the transfer to the Server4: TINITREADWRITE

Content of the struct elements

PokeStatus	INIT_POKE
szPath	'c:\projekt\prod\0'
szVarName	'FANZAHL\0'
bCommonID	1
bPCRC	0

Reading of the total of designs already produced

Server item: VarRead3

Variable in BAPS program: z. B. SUMME1

Data struct for the transfer to the Server4: TINITREADWRITE

Scope of functions

Content of the struct elements

PokeStatus	INIT_POKE
szPath	'c:\projekt\prod\0'
szVarName	'SUMME1\0'
bCommonID	1
bPCRC	0

Reading of the output signal communicating that the order is completed

Server item: VarRead4
 Variable in BAPS program: z. B. ENDESIG
 Data struct for the transfer to the Server4: TReadVar

Content of the struct elements

PokeStatus	INIT_POKE
szPath	'c:\projekt\prod\0'
szVarName	'ENDSIG/0'
bCommonID	1
bPCRC	0

All cyclical items must be started by XTYP_ADVSTART.

Reply to a cyclical item

The Client receives a message from the Server4 when the content of one of the before mentioned variables has changed. In the course of production, the variable SUMME1 will change. As response to this change, the Server4 submits the following data package to the Client:

Data received by the Client:

Data struct for the transfer to the Client: TREADVARDATA

Content of the struct elements

GStatus	Check data record (is not further explained here)
nGroesse	Number of bytes transferred (in this case 4)
Var.lGanz	Content of 'SUMME1'

Initialization (setting once) of variables to be written

The Client communicates to the Server4 the design and number of the parts to be produced. In the example, five parts of design 1 are to be made.

Writing of the desired design

Server item: VarWrite1

Scope of functions

Variable in BAPS program: ANZAHL
Data struct for the initialization: TINITREADWRITE

Content of the struct elements

PokeStatus	INIT_POKE
szPath	'c:\projekt\prod\0'
szVarName	'ANZAHL\0'
bCommonID	1
bPCRC	0

Via the item VarWrite1, the variable ANZAHL can then be set. In the RC, the Prod.ird file is opened for writing.

Data struct for the transfer to the Server4: TWRITEVAR

Content of the struct elements

PokeStatus	DATA_POKE
nGroesse	4 (number of bytes)
Var.lGanz	5

The variable ANZAHL is set to the value 5.

Data struct for ending : TEXTWRITEPOKE

Content of the struct elements

PokeStatus	EXIT_POKE
------------	-----------

The Prod.ird file is closed again in the control. The VarWrite1 item can now be allocated again to other variables.

The BAPS process is now completely initialized and waits for the start signal. The production is started by setting the binary variable STARTSIG. Since the STARTSIG signal has to be set more frequently, a cyclical item is selected for this task.

Writing of the start signal

Server item: VarWrite1
Variable in BAPS program: STARTSIG
Data struct for the Initialization: TINITREADWRITE

Content of the struct elements

PokeStatus	INIT_POKE
szPath	'c:\projekt\prod\0'
szVarName	'STARTSIG\0'
bCommonID	1
bPCRC	0

Scope of functions

Above the VarWrite1 item, the STARTSIG variable can then be set. In the RC, the Prod.ird file is open for writing.

The cyclical setting of the signal is possible by XTYP_ADVSTARTt.

Data struct for the transfer to the Server4: TWRITEVAR

Content of the struct elements

PokeStatus	DATA_POKE
nGroesse	4 (number of bytes)
Var.lGanz	1

The STARTSIG variable is set to the value 1.

Data struct for the transfer to the Server4: TWRITEVAR

Content of the struct elements

PokeStatus	DATA_POKE
nGroesse	4 (number of bytes)
Var.lGanz	0

The STARTSIG variable is reset to the value 0 and prepared for the next start. The signal can then be set as often as desired.

The cyclical setting of the signal is ended again by ADV_STOP. The Prod.ird file is closed again in the control. The VarWrite1 item can then be allocated again to other variables.

With the cyclical items mentioned before, the production flow can be monitored via the Client. Wrong inputs and the end of the production procedure are displayed by the Server4 immediately. At the end of production, the BAPS process waits for a new start. If the type of design and the number are not overwritten, the values set before will be maintained. The Client is informed about the end of a production procedure by a message through the cyclical item VarRead4, which monitors the ENDESIG output.

Appendix

A Appendix

A.1 Abbreviations

Abbreviation	Meaning
ASCII	American Standard Code for Information Interchange
BAPS3	Programming language; Bewegungs- und Ablaufprogrammiersprache, Version 3
C:	Hard disk drive C
DDE	Dynamic Data Exchange
DDEML	Dynamic Data Exchange Management Library
EGB	Elektrostatic sensitive components
ESD	Electrostatic discharge
JC	Machine coordinates
MPP	Machine parameter program
PC	Personal Computer
PE	Protective Earth
POS	Actual position
PG	Programming unit
RC	Robot Control
RCO	Robot Control Output
ROPS4	Robot programming system for rho4
TCP/IP	Transmission Control Protocol/Internet Protocol
OC	Original coordinates
WC	World coordinates

Appendix

A.2 Index**A**

Axis positions, 6–31

C

Client, 2–2

Client control, 6–5

clipboard format, 2–2

cold link, 2–3

communication parameter, 4–3

computer key, 3–2

Connect, 2–4

control file name, 6–26

Control status, 6–2

cyclic items, 5–4

D

DDE, 2–1

DDE–Connect, 5–1

DDE–Server4, 2–2, 4–1

DDEML Library, 2–2

Delete, 6–29

Directory transfer, 6–26

Documentation, 1–7

Download, 6–17

dynamic data, 2–4

Dynamic Data Exchange, 2–2

E

EMC Directive, 1–1

EMERGENCY–STOP devices, 1–5

Error number, 6–3

ESD

Electrostatic discharge, 1–6

grounding, 1–6

workplace, 1–6

ESD–sensitive components, 1–6

F

file administration functions, 5–4

file transfer functions, 6–17

Floppy disk drive, 1–7

G

global status, 6–1

Grounding bracelet, 1–6

GStatus, 5–5

H

Handshake, 6–12

Hard disk drive, 1–7

Heartbeat, 5–3

hot link, 2–4

I

initialization, 5–3, 6–70

ird file, 6–52

item, 5–1

K

kinematics, 6–30

Kinematics info, 6–30

L

licence application, 3–2

Licensing, 4–4

Low–Voltage Directive, 1–1

M

main processes, 6–44

Modules sensitive to electrostatic discharge. See
ESD–sensitive components

monitor, 4–3

monitor dialog, 4–3

monitoring, 6–15

O

Online functions, 6–30

operation, 4–1

P

point file, 6–52

polling cycle, 6–32

Process list, 6–41

Process selection, 6–38

Process status, 6–43

Process stop, 6–41

Q

Qualified personnel, 1–2

R

RCA, 6–49

refresh rate, 4–3

Release, 1–8

rename, 6–28

ROPS4 , 2–1

S

Safety instructions, 1–4

Safety markings, 1–3

Server, 2–2

Server control, 6–4

server error, 6–7

Signal display, 6–49

single transfer, 2–2

Appendix

- Spare parts, 1–6
- Standard operation, 1–1
- Starting position, 6–46
- static data, 2–3
- static data exchange, 2–3
- status and initializing functions, 6–1
- subprocesses, 6–44
- system stress, 4–3

T

- TCP/IP connections, 5–1, 6–13
- Test activities, 1–5
- timer, 5–4
- Tool, 6–36
- Topics, 5–1
- Trademarks, 1–8

U

- undefined points, 6–60
- Upload, 6–22
- user variables, 6–52

V

- variable name, 6–53

W

- WC system, 6–37
- wildcards, 6–17

Appendix

Notes:

Bosch Rexroth AG
Electric Drives and Controls
P.O. Box 13 57
97803 Lohr, Germany
Bgm.-Dr.-Nebel-Str. 2
97816 Lohr, Germany
Phone +49 (0)93 52-40-50 60
Fax +49 (0)93 52-40-49 41
service.svc@boschrexroth.de
www.boschrexroth.com

