SIEMENS

Motion Control Information System

SINUMERIK 840D/840Di/810D RPC SINUMERIK Computer Link

Function Manual

Valid for

Control	*Software version*
Motion Control Information System	2

10.05 Edition

SINUMERIK® Documentation

Printing history

Brief details of this edition and previous editions are listed below.

The status of each version is indicated by the code in the "Remarks" columns.

Status code in the "Remarks" column:

A .... New documentation.
B .... Unrevised reprint with new order No.
C .... Revised edition with new status.
If factual changes have been made on the page since the last edition, this is indicated by the new version code in the header on that page.

Edition	Order No.	Comments
01.04	6FC5297-6AD61-0BP0	**A**
10.05	6FC5297-6AD61-0BP1	**C**

Additional information is available on the Internet at:
http://www.ad.siemens.com/sinumerik

This publication has been produced using WinWord V 8.0, Designer V 7.0 and the DocuTool AutWinDoc .

Other functions not described in this documentation may be executable in the control. This does not, however, represent an obligation to supply such functions with a new control or when servicing.

We have checked the contents of this manual for agreement with the hardware and software described. Nevertheless, differences might exist and we cannot, therefore, guarantee that they are completely identical. The information given in this publication is reviewed at regular intervals and any corrections that might be necessary are made in the subsequent printings. Suggestions for improvement are welcomed.

Subject to change without prior notice

Order No. 6FC5297-6AD61-0BP1
Printed in the Federal Republic of Germany

Siemens AG

Preface

SINUMERIK Documentation

The SINUMERIK documentation is organized in 3 parts:

- General documentation
- User documentation
- Manufacturer/Service documentation

Please contact your local Siemens office for more detailed information about other SINUMERIK 840D/840Di/810D publications and publications that apply to all SINUMERIK controls (e.g. universal interface, measuring cycles, etc.).

An overview of publications, which is updated on a monthly and also provides information about the language versions available, can be found on the Internet at:
http://www.siemens.com/motioncontrol
Follow the menu items "Support" → "Technical Documentation" → "Overview of Publications".

The Internet version of DOConCD (DOConWEB) is available at:
http://www.automation.siemens.com/doconweb

Target audience

This Manual is intended for machine-tool users. This publication describes in detail all the facts the user needs to know in order to program the SINUMERIK 840D sl/840D/840Di/810D control.

Standard version

This Programming Guide describes the functionality afforded by standard functions. Extensions or changes made by the machine tool manufacturer are documented by the machine tool manufacturer.
Other functions not described in this documentation might be executable in the control. This does not, however, represent an obligation to supply such functions with a new control or when servicing.

Hotline

If you have any questions, please contact our hotline:

A&D Technical Support
Phone.: +49 (0) 180 / 5050 - 222
Fax: +49 (0) 180 / 5050 - 223
E-mail: mailto:adsupport@siemens.com
Internet: http://www.siemens.com/automation/support-request

If you have any queries (suggestions, corrections) in relation to this documentation, please fax or e-mail us:

Fax: +49 (0) 9131 / 98 - 63315
E-mail: mailto:motioncontrol.docu@siemens.com

Fax form: See the reply form at the end of the brochure.

SINUMERIK Internet address

http://www.siemens.com/motioncontrol

Subject matter of this manual

The manual describes the computer link software on the PCU component (item supplied) and the functions which a host computer software program must provide on the communication side. The necessary interfaces are included in the description.

The publication consists of the following two parts:
Part 1: Interface for host computer
Part 2: PLC/NCK interface

Safety Information

This Manual contains information which you should carefully observe to ensure your own personal safety and the prevention of material damage. Notes relating to your personal safety are highlighted in the manual by means of a warning triangle, no warning triangle appears in conjunction with notes that relate to property damage. The warnings appear in decreasing order of risk as given below.

Danger

Indicates that death or severe personal injury **will** result if proper precautions are not taken.

Warning

Indicates that death or severe personal injury **may** result if proper precautions are not taken.

Caution

With a warning triangle this indicates that minor personal injury may result if proper precautions are not taken.

Caution

Without a warning triangle this indicates that property damage may occur if proper precautions are not taken.

Notice

indicates that an unintended result or situation can occur if the corresponding information is not taken into account.

If several hazards of different degrees are present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified personnel

The device/system may only be set up and used in conjunction with this documentation. The device/system may only be commissioned and operated by **qualified personnel**. Within the context of the safety notes in this documentation qualified persons are defined as persons who are authorized to commission, ground and label devices, systems and circuits in accordance with established safety practices and standards.

Correct usage

Note the following:

Warning

The equipment may only be used for single purpose applications explicitly described in the catalog and in the technical description and it may only be used along with third-party devices and components recommended by Siemens. To ensure trouble-free and safe operation of the product, it must be transported, stored and installed properly and maintained and operated with care.

Additional notes

Important

This notice indicates important facts that must be taken into consideration.

Note

This symbol always appears in the document where further information is provided.

Machine manufacturer

This pictorial symbol always appears in this document to indicate that the machine manufacturer can affect or modify the function described. Never ignore information provided by the machine manufacturer!

■

Motion Control Information System

SINUMERIK 840D/840Di/810D
RPC SINUMERIK Computer Link

Interface to Host Computer (NFL) (FBR/NFL)

1

1 Introduction

1.1 General information about RPC SINUMERIK

Description of the interface between a host computer and a SINUMERIK 840D or 810D.

- Communication between the systems is based on Ethernet and TCP/IP protocol.
- Data transfer can be initiated by either the host computer or SINUMERIK.
- The interface to RPC SINUMERIK is based on RPC (Remote Procedure Call) and the transfer of files according to the DCE standard.

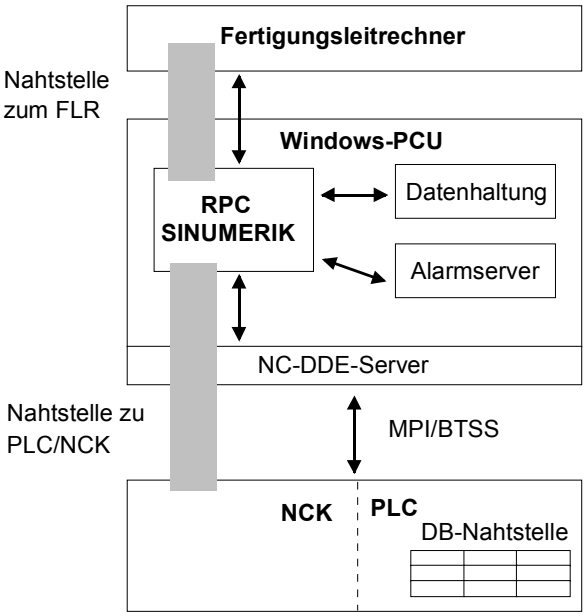


Fig. 1-1 Interfaces

Data transfer

If there is only a small amount of data to be transferred, this is done in the RPC function call. NC programs and other data already available as files are also transferred as files. More extensive data, e.g. tool data, are written to a file and then also transferred as a file.
The filename and other parameters are transferred to the peer with an RPC function call.
Depending on the transfer options between the host computer and SINUMERIK, the files are transferred either by FTP or, in a network file system, by copying.

RPC functions

The RPC functions in this document are described from the perspective of the host computer and therefore of the user. RPC calls arriving at the host computer are referred to in the rest of this document as "called functions". An RPC call arriving at the host computer causes the corresponding function to be started in the host computer.
RPC calls made from the host computer are referred to in the rest of this document as "function calls". A function call from the host computer causes the corresponding function to be called in RPC SINUMERIK.

COM port

The RPC SINUMERIK-OCX development package is an add-on product for the RPC SINUMERIK computer link. The MCIS_RPC.OCX encapsulates the RPC interface from RPC SINUMERIK in COM calls (Component Object Model). Using the MCIS_RPC.OCX interface, the RPC SINUMERIK interface is accessible to a whole range of common Windows development systems without the need to program in C/C++.

The description of the RPC SINUMERIK-OCX can be found in **Chapter 10** of the first part of this publication (NFL).

1.2 Languages in RPC SINUMERIK

RPC SINUMERIK generally supports all languages which are covered by the ASCII character set or for which a language package can be installed on the SINUMERIK. Languages which use an extended character set (e.g. Chinese, Russian, etc.) are processed by SINUMERIK in DBCS format. The following must not be specified in any language which uses an extended character set:

- NC programs
- Tool data

■

2

2 System Installation

2.1 System requirements

2.1.1 Software

SINUMERIK 840D/810D

The following software is required on the SINUMERIK 840D/810D:

- PCU50/70 V1/V2, HMI ADVANCED 6.x
- The drivers for the Ethernet interface, and TCP/IP must be installed.
- MCIS-TDI (tool handling module) (only if the functions x_DATA_x SFct 21-23 are to be used).
- RPC SINUMERIK computer link software package

Host computer

Drivers for the installed Ethernet interface module.
TCP/IP with the FTP or NFS services and RPC according to the DCE standard.

2.1.2 Hardware

SINUMERIK 840D/810D

- PCU50

- For the network interface kits used, please contact the SINUMERIK Hotline or see SINUMERIK FAQs: (http://www.siemens.com/support)

Host computer

For the network interface kits used, please contact the SINUMERIK Hotline or see SINUMERIK FAQs: (http://www.siemens.com/support)

SINUMERIK/host computer connection

A network cable is used for the Ethernet interface.

2.2 Start-up

The RPC SINUMERIK software is loaded onto the SINUMERIK and the Ini file of the program is configured.
DB12 for the DB interface must be set up on the PLC. This will normally be performed when the PLC of the machine manufacturer is installed.
The data block structure is described in "RPC SINUMERIK Computer Link, PLC/NCK Interface (NPL)" in **Chapter 1** (see /2/ Chapter 1 RPC interface and machine PLC).

■

For notes

3

3 Operating Modes and States

3.1 Operating modes

The SINUMERIK 840D operates in the Automatic, MDA, JOG and TEACH IN modes. The computer link requires its own operating modes in addition to these. The functional behavior of RPC SINUMERIK, even together with the PLC, depends on the host computer communication mode. A separate dialog is offered on the SINUMERIK to operate and display the host computer communication mode.
The host computer communication modes are:

- Host computer mode unmanned
- Host computer mode manned
- Manual mode
- Special mode
- Offline

3.1.1 Host computer mode unmanned/manned

In the two host computer communication modes (manned and unmanned), it is possible to specify an NC program on the host computer and start it from the PLC in connection with the Automatic mode. The distinction between manned and unmanned modes serves to initiate different responses in the event of a fault.
If a fault occurs during manned production, the workpiece carrier remains in the machine. The operator must remedy the fault before the machining process can be continued.
In the event of a fault during unmanned production, it is possible to terminate machining of the workpiece carrier in order to continue with another workpiece carrier.

3.1.2 Manual mode

The NC programs are specified by the host computer. The NC program is selected over the computer link; the selected NC program appears in the header line of the user interface.
The operator must start the program.

Note

In the host computer communication modes and manual mode, material flow on the machine is automatic, i.e. the transport system automatically delivers workpiece carriers and fetches them again after machining.

3.1.3 Special mode

In special mode, there is no automatic transport of workpiece carriers, the host computer does not transfer program assignments, and there is no automatic starting of NC programs. The machine is controlled by the operator; in other respects full communication takes place with the host computer. The special mode is generally used to test NC programs or to machine unplanned workpieces manually.

3.1.4 Offline

Offline indicates an interruption in the connection between the host computer and the SINUMERIK. The SINUMERIK switches to Offline when it detects an interruption in the connection to the host computer. The host computer also flags a machine as Offline in its status data and in the plant display as soon as it detects an interruption in the connection.
After Offline, the machine must be synchronized on the host computer.
If an operating mode (either a SINUMERIK mode or a computer communication mode) is switched on the machine, this must be reported to the host computer with the RPC call R_MACHINE_H().

3.2 Machine states

The machine can have any of the following states:

- Cold restart: After a restart on the SINUMERIK
- Inactive: No machining
- Active: Machining in progress
- Fault: Machining interrupted
- Units deactivated

3.3 Docking positions/storage location states

The host computer must recognize the states of the docking positions, in order to initiate transport jobs on the transport system.
The following states are possible:

- Enabled
- Disabled for transport control system
- faulty

3.4 Workpiece carrier states

The host computer also requires the workpiece carrier states in order to determine the transport jobs. The workpiece carrier can have any of the following states:

- Not machined – no program assignment:
  An NC program has not yet been assigned for machining, i.e. machining is not yet possible.
- Not machined – program assigned:
- In progress
- Finished
- Finished with error
- Only for buffering:
  The workpiece carrier is only buffered on the machine; no machining will take place.

■

4

4 Tool data

Tool data

Since not all data of a tool are always required, 3 versions are available.

1. The first version includes the full scope of tool data; the other two versions contain only a subset of the data.
2. The configurability of the data scope for each tool only extends to complete areas, i.e. a list, which is provided (in the registry) for each version, specifies the areas which are transferred in this version.

The file format for the tool data corresponds to the data backup format of the NC840D (punched tape/ASCII format acc. to /BA/ ; such as in _N_TOx_TOA or _N_TOx_INI). The complete description of data content and layout is provided in /NPL/ starting at **Chapter 4**.

Note

/NPL/ stands for the second part (NPL) of this document.

Tool data areas

The tool data are stored in different areas on the NCK.
The areas are designated as follows:

Table 4-1 Tool data areas

Area name	Tool identifier
General tool data	$TC_TPx[y]
User-related tool data	$TC_TPCx[y]
Cutting edge data	$TC_DPx[y,z]
User-related cutting edge data	$TC_DPCx[y,z]
Tool monitoring data	$TC_MOPx[y,z]
User-related tool monitoring data	$TC_MOPCx[y,z]

x : A running number for each area used to produce a unique name for the system variable.
Y : T number
z : Cutting edge number

Table 4-2 Magazine location data

Area name	Tool identifier
Magazine and location number	$TC_MPP6[y,z]=x

x : T number
Y : Magazine number
z : Location number.

Structure of the file

Table 4-3 Description of the tool and magazine location data

	Description
$TC_TP1[3]= 2	Duplo number
$TC_TP2[3]= "4711"	ID number
$TC_TP3[3]=1	Size to left in half locations
$TC_TP4[3]=1	Size to right in half locations
$TC_TP5[3]=1	Size upwards in half locations
$TC_TP6[3]=1	Size downwards in half locations
$TC_TP7[3]=2	Magazine location type
$TC_TP8[3]=131	Status
$TC_TP9[3]=0	Tool monitoring method
$TC_TP10[3]=2	Tool replacement strategy
$TC_TP11[3]=0	Tool information
$TC_DP1[3,1] = 0	Edge parameter 1
$TC_DP2[3,1] = 0	Edge parameter 2
$TC_DP3[3,1] = 0	Edge parameter 3
$TC_DP ...	...
$TC_DP24[3,1] =0	Edge parameter 24
$TC_DP25[3,1] =0	Edge parameter 25
*$TC_MPP6[1,5]=3	T number

*

$TC_MPP6 indicates which tool (T number) is in the stated magazine and location. In this case tool number 3 is in magazine 1 in location 5.

■

5

5 Communication between Host Computer and SINUMERIK

5.1 Introduction Remote Procedure Calls

Communication between the host computer and the SINUMERIK takes place using remote procedure calls (RPC) for the transfer of small volumes of data. During communication via RPC, the communication partner is requested to perform a function identified with procedure name with the parameters (data) included in the call.

Note

For the Interface Definition Language IDL (technical program definition) of the calls used in the following, please refer to the appendix

The names are structured according to the following system for the procedures described in this section:

5.1.1 Structure of procedure names

The procedure names contain three components:
1. Command identifier (first character)
2. Data/function identifier
3. Receiver identifier (last character)

Command identifier

The command identifier appears in the first character of the name:
C Command call (Command)
R Receive data request (Receive)
D Transmit data request (Transmit)

Example: **T_**MACHINE_M ().

Data/ function identifier

The identifier indicates the type of data requested or transmitted or the function to which the data are passed.
Examples: T**_MACHINE_**M (), R**_NC4WPC_**M().

Receiver identifier

The receiver identifier indicates the address of the unit which is to execute the function:
H The host computer is the receiver (Host)
M (GND) The machine is the receiver (Machine)

5.1.2 Parameters for general use

Host

Name of the host computer with up to 16 characters. Where several machines are networked to several host computers, Host is a unique identifier for the host computer with which data are to be exchanged.

Machine

Name of a machine (max. 16 characters). Unique identifiers must be available for all machines which exist on the network.

OrderNum

Job number: The number is optional and can be used if RPC requests and their answers have to be assigned uniquely to each other.

Note

With parameters of type String, the string must be delimited by '\0'.
In Visual C++ "\0" and "" denote an empty string; in Visual Basic "" produces an empty string.
The maximum string lengths are specified with the individual parameters.

5.2 Communication processes

Requirements

For error-free communication to take place between the host computer and one or more machines, it is necessary for the host computer to know the communication partners for which it is to process RPCs. The machine names of these machines (clients) must be stored on the host computer.

Parameter

Parameters are data transferred within the RPC. The table block transferred with every RPC describes the parameters to the host computer and machine.

Acknowledgment of

As with local procedures, the return value of the RPC indicates a positive or negative acknowledgment. Where requests are processed asynchronously, this acknowledgment can only acknowledge receipt of the request. After processing, or if an error occurs during processing, the communication partner must be informed by means of an appropriate RPC message. Error messages are displayed on the SINUMERIK with the aid of the alarm server. For correctly performed calls, the return value = 0. A list of the return values in the event of an error is included in the appendix.

Note

The host computer software must return the called functions as quickly as possible, because the component of RPC SINUMERIK that initiates the RPC waits for it to be processed, with the result that no further processing can take place during this period. Within the function called up on the host computer, the data contained in the RPC should be copied into a buffer and the function returned immediately. Actual processing on the host computer should only take place afterwards.

5.3 Machine status data

5.3.1 Transmit machine status data

Called function

R_MACHINE_H (Host,
Machine,
OrderNum,
MachineMode,
MachineStatus,
NCProgram,
ClampCubeSide,
DockPos,
DockPosStatus,
WPC,
WPCStatus,
Resint1,
Resint2,
Resbyte**)**

Direction of transfer: **SINUMERIK → host computer**

Meaning

Transmit machine data to the host computer.

Data

Table 5-1 Parameters for machine status data transfer

Parameter	Description	Format
Host	Name of the host computer	16 bytes (string)
Machine	Machine name	16 bytes (string)
OrderNum	Order number	4 bytes (long int)
MachineMode	Mode • SINUMERIK mode: 1: Automatic 2: MDA 4: JOG 8: TEACH IN • Computer link mode: 100: FLR mode unattended 200: FLR mode attended 300: Manual mode 400: Special mode 500: If bit 6=1 in RPC SINUMERIK mode 600: If bit 7=1 in RPC SINUMERIK mode	4 bytes (long int)
MachineStatus	Machine status 0: Cold restart 1: Inactive 2: Active 3: faulty 4: Components deactivated	4 bytes (long int)
NCProgram	Current NC program*	128 bytes (string)
ClampCubeSide	Side on clamp cube**	4 bytes (long int)
DockPos[3]	Docking position no. The docking position no. corresponds to the index in the docking position list of the interface DB beginning at the 1st Docking position no. = 0 means: 'not assigned' (see /NPL/ Section 1.1 Description)	3 x 4 bytes (long int)
DockPosStatus[3]	Docking position status 0: Enabled 1: Disabled for transport control system 2: faulty	3 x 4 bytes (long int)
WPC[3]	Workpiece carrier name	3 x 6 bytes (string)
WPCStatus[3]	Workpiece carrier status 1: Not machined, no Program assignment 2: Unmachined, program assigned 4: Prepare program selection 8: Program selection done 16: In progress 32: Finished	3 x 4 bytes (long int)

Parameter	Description	Format
	64: Finished with error 128: Only for buffering	
Resint1***	Reserve 1	4 bytes (long int)
Resint2***	Reserve 2	4 bytes (long int)
Resbyte	Reserve 3	8 bytes (string)

* NC program name of the NC program currently running
** Side (on clamp cube) which is currently being machined
*** Resint1 and 2 appear at the DB interface of the PLC; if values are entered there by the PLC they are transferred to the host computer. These values have no impact on the computer link; they are merely transferred to the host computer.

Notes on use

- The SINUMERIK must initiate this RPC on every status change that takes place on the machine. The RPC SINUMERIK computer communication software determines the current data and initiates the call.
- The procedure is triggered by the PLC by setting a specific bit at the DB interface.
- The host computer can also initiate the procedure using the command **T_MACHINE_M** (request machine status data, see below).

Note

- If more than 3 docking positions are to be described, a separate machine assignment file must be transferred. See **Section 5.12**
- In order to report the operating modes (SIN and host computer) of both nodes in a variable (MachineMode), the quantity can be transferred as a value (e.g. 201: host computer mode manned =200 and SIN Automatic =1).
- The RPC SINUMERIK computer communication software does not verify the operating modes.

5.3.2 Request machine status data

Function call

T_MACHINE_M (Host,
Machine,
OrderNum)

Direction of transfer: **Host computer → SINUMERIK**

Meaning

Request machine status data

Table 5-2 Parameters for machine status data request

Parameter	Description	Format
Host	Name of the host computer	16 bytes (string)
Machine	Machine name	16 bytes (string)
OrderNum	Order number	4 bytes (long int)

Notes on use

The host computer can use this call to request machine status data, e.g. during synchronization. The SINUMERIK then transmits the data to the host computer with the command **R_MACHINE_H()**.

Example

T_MACHINE_M("FLR1","BAZ3",0);

5.4 Production dialog

Description

After a workpiece carrier arrives at the machine, the SINUMERIK transmits the machine status data to the host computer on initiation by the PLC.
The host computer can recognize from the data which workpiece carrier has arrived at the machine; it then transfers the program assigned to this workpiece carrier.
Where workpiece carriers have clamp cubes, one NC program is assigned to each side of the cube. RPC SINUMERIK stores these program assignments. Each program assignment consists of workpiece carrier, side and NC program. In each case, the next NC program is transferred and selected. The NC program can then be started from the PLC (in host computer modes manned and unmanned).
The machine reports the NC start and later the NC end in the machine status data.

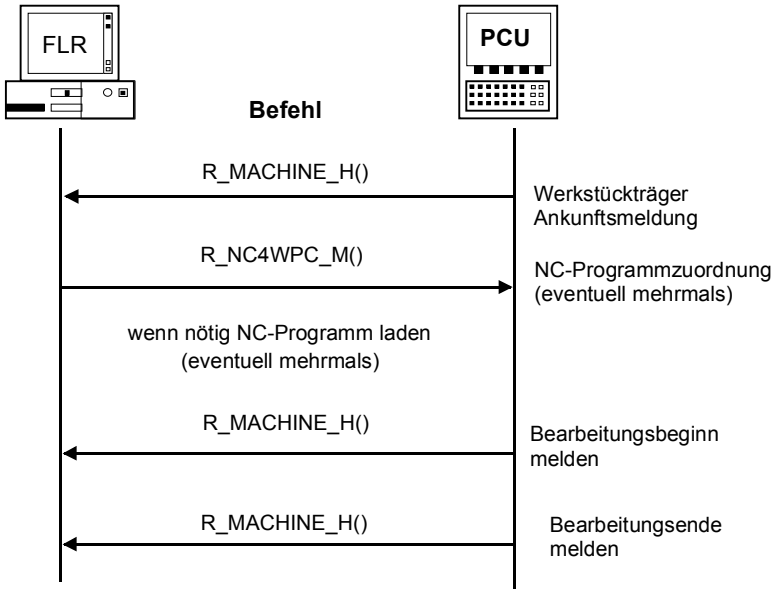


Fig. 5-1 Production dialog, normal execution, no errors

If errors occur during this procedure, appropriate messages are output (see 5.5.1 Messages from the SINUMERIK to the host computer).

5.4.1 Program assignment

Function call

R_NC4WPC_M (Host,
Machine,
OrderNum,
WPC,
NCProg,
Date,
NCPLength,
ClampCubeSide,
TpFlag,
NCExtern,
Resint1,
Resint2,
Resbyte)

Direction of transfer: **Host computer → SINUMERIK**

Meaning

Instruct machine which program is to be activated

Data

Table 5-3 Parameters for program assignment

Parameter	Description			Format
Host	Name of the host computer			16 bytes (string)
Machine	Machine name			16 bytes (string)
OrderNum	Order number			4 bytes (long int)
WPC	Workpiece carrier name			6 bytes (string)
NCProg	NC program NC program in the form: "\mpf.dir\Zylinderkopf.mpf"			128 bytes (string)
Date	Date of the last change of the NC program (Unix time)			4 bytes (long int)
NCPLength	NC program size in bytes			4 bytes (long int)
ClampCubeSide	Side on clamp cube			4 bytes (long int)
TpFlag	Transport flag = 0: No follow-up operation = 1: Follow-up operation = 9: Only for buffering			4 bytes (long int)
NCExtern	Execute from external			4 bytes (long int)
	Bit 1	**Bit 0**	**Function**	
	0	0	Execute NC program on NCK	
	0	1	Execute NC program from external (bit 0=1)	
	1	0	Execute workpiece on NCK (bit 1=1)	
	1	1	Execute workpiece from external (bits 0 and 1=1)	

Parameter	Description	Format
Resint1	Reserve 1	4 bytes (long int)
Resint2	Reserve 2	4 bytes (long int)
Resbyte	Reserve 3	8 bytes (string)

Notes on use

- This RPC can occur several times for a workpiece carrier if each side of a clamp cube is to be machined with a separate NC program.
  The sides are machined in the order in which the program assignments are reported by the host computer to RPC SINUMERIK .
- In all program assignments of a workpiece carrier except for the last one, the transport flag "1 = follow-up operation" must be set. If the follow-up operation flag is set for a side, the workpiece carrier remains at the machining station at the end of the NC program. The fact that the flag is not set for the last side indicates that no further machining is required and the workpiece carrier can be moved away from the machining station.
- If a workpiece carrier is only stored on a machine for buffering purposes, this can be reported by setting transport flag "9 = only for buffering". In this case, no NC program is specified.

Example

R_NC4WPC_M ("FLR1", "BAZ3", 0, "WPC05", "\mpf.dir\Kw15.mpf", 862826400, 3210, 1, 0, 0, 0, 0, "\0");

5.5 Messages

5.5.1 Messages from SINUMERIK to the host computer

Called function

R_REPORT_H (Host,
Machine,
OrderNum,
Type,
Number,
Time,
Flag,
Resint1,
Resint2,
Resbyte)

Direction of transfer: **SINUMERIK → host computer**

Meaning

Send message to host computer

Data

Table 5-4 Parameters for the message from SINUMERIK to host computer

Parameter	Description	Format
Host	Name of the host computer	16 bytes (string)
Machine	Machine name	16 bytes (string)
OrderNum	Order number	4 bytes (long int)
Type	Message type 1: Alarms 2: Operator interruption 3: Operational messages 4: Error message from RPC SINUMERIK computer link software 5: Positive acknowledgment with R_DATA_M (), R_DDEDATA_M () and R_VAR_M () 6: Positive acknowledgment with C_ORDER_M ()	4 bytes (long int)
Number[10]	Message number array • Array for up to 10 alarms. • Array elements which are not required should be initialized with 0 • In other message types, only Number[0] is allocated	10 x 4 bytes (long int.)
Time[10]	Time stamp array • Array for up to 10 entries. • Array elements which are not required should be initialized with 0	10 x 4 bytes (long int.)

Parameter	Description	Format
Flag[10]	Incoming/outgoing flag, array up to 10 entries C: Incoming message, no machine standstill S: Incoming message, machine standstill G: Message outgoing L: All messages outgoing The following distinction must be made in incoming messages: Results in machine standstill yes/no	10 x 1 bytes
Resint1	Reserve 1	4 bytes (long int)
Resint2	Reserve 2	4 bytes (long int)
Resbyte	Reserve 3	8 bytes (string)

With R_DATA_M (), R_DDEDATA_M (), R_VAR_H () and C_ORDER_M (), synchronous processing cannot take place during the call, therefore RPC SINUMERIK must send a positive acknowledgment to the host computer after processing, in order to inform the host computer that the process has terminated. The subfunction number of R_DATA_M is used as an "error number" with a positive acknowledgment to R_DATA_M; it is used to assign the acknowledgment. In the case of a positive acknowledgment to C_ORDER_M (), the SubFnc is also entered as the error number; the type, however, is 6. "Error number" 1000 is returned with R_DDEDATA_M and "error number" 0 is returned with R_VAR_M.

Note

Numbers, times and flags with the same index belong together

Notes on use

RPC R_REPORT_H() is used to send individual messages or groups of up to 10 alarm messages to the host computer.
Special case: If the last alarm is outgoing on the machine, this status is reported by R_REPORT_H() and the parameters:

- Type = 1
- Number[0] = 0
- Flag[0] = L

sent to host computer.

Message filters

A message filter has been implemented in RPC SINUMERIK to suppress the transmission of certain messages or alarms to the host computer. It is possible to suppress alarm types from the alarm server completely and it is also possible to suppress ranges of certain numbers.
This feature is configured using the file ScAlarmEx.ini in the ADD_ON directory. The section [AlarmTypeExclusion] serves to exclude alarm types, and the section [AlarmRangeExclusion] serves to exclude ranges of numbers.
The vocabulary words (Type1 to TypeN or Range1 to RangeN) must be assigned without gaps; the reading function reads with ascending number and stops reading with the first number not existing.

The structure of ScAlarmEx.ini:

```
[AlarmTypeExclusion]
Type1=6
Type2=7
Type3=8
Type4=9
Type5=10
Type6=11
[AlarmRangeExclusion]
Range1=100,199
Range2=250,250
Range3=2001,3999
Range4=5000,5050
Range5=6799,6799
```

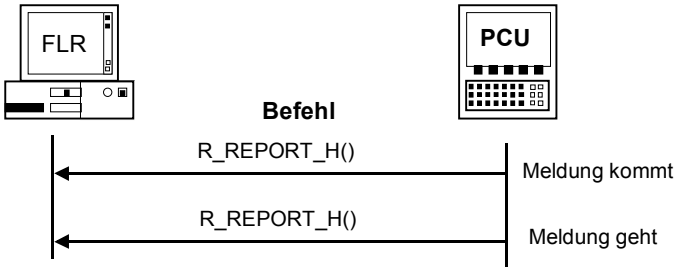


Fig. 5-2 Procedure for alarms/operator interruptions/operational messages

If machining continues on the machine after the end of an alarm or an operator interruption with machine standstill, this must be reported to the host computer with the RPC **R_MACHINE_H ()** together with the machine status (for the transport system, it is reported with **R_TPS_H ()**).
A list of the error messages for message type 4 is included in the appendix.

5.5.2 Messages from the host computer to SINUMERIK

Function call

R_REPORT_M (Host,
Machine,
OrderNum,
Type,
Number,
Resint1,
Resint2,
Resbyte

Direction of transfer: **Host computer → SINUMERIK**

Meaning

Error messages from the host computer are sent to RPC SINUMERIK on the SINUMERIK.

Data

Table 5-5 Parameters of the message from host computer to SINUMERIK

Parameter	Description	Format
Host	Name of the host computer	16 bytes (string)
Machine	Machine name	16 bytes (string)
OrderNum	Order number	4 bytes (long int)
Type	Message type 4: Error messages of the host computer to SINUMERIK	4 bytes (long int)
Number	Error code	4 bytes (long int)
Resint1	Reserve 1	4 bytes (long int)
Resint2	Reserve 2	4 bytes (long int)
Resbyte	Reserve 3	8 bytes (string)

Notes on use

The error messages are entered on the SINUMERIK in a LOG file. In addition, certain error messages trigger an error handling in RPC SINUMERIK.

Example

R_REPORT_M ("FLR1", "BAZ3", 0, 4, -13, 0, 0, "\0");

5.5.3 Request pending alarms

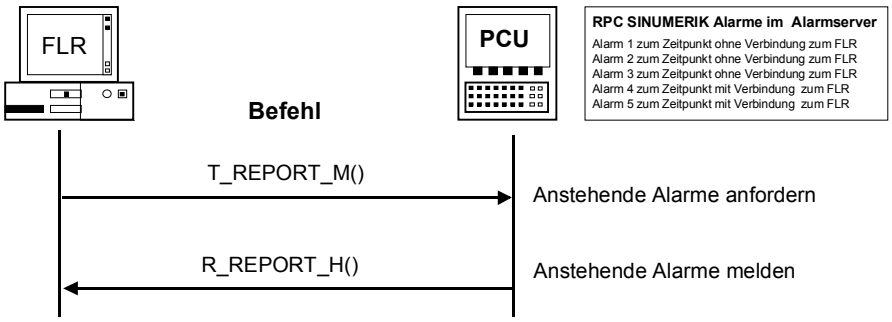


Fig. 5-3 Requesting pending alarms

T_REPORT_M (Host,
Machine,
OrderNum,
Direction of transfer: **Host computer → SINUMERIK**

Meaning

T_REPORT_M can be used to request alarms and messages pending on the alarm server, taking into account the message filter. **All** pending alarms and messages are requested, irrespective of whether they occurred with or without connection to the host computer at any time.
Thus, the function provides a synchronization between host computer and SINUMERIK with regard to the alarms and messages.
Otherwise, only such alarms are transferred to the host computer which occurred **with** connection to the host computer at any time.
The alarms and messages requested with **T_REPORT_M** are reported with **R_REPORT_H**.

Data

Table 5-6 Parameters of the alarm request

Parameter	Description	Format
Host	Name of the host computer	16 bytes (string)
Machine	Machine name	16 bytes (string)
OrderNum	Order number	4 bytes (long int)

5.6 Exchange operator messages

The two RPC functions **R_MESSAGE_M** and **R_MESSAGE_H** serve to exchange plain text messages between the host computer and SINUMERIK.

The display of the messages provided to SINUMERIK from the host computer (**R_MESSAGE_M**) is carried out via the alarm server (in the header). The priority of the messages can be set in the host computer using the **ResInt1** parameter from R_MESSAGE_M. The **ResInt1** parameter can have the following meaning in combination with R_MESSAGE_M:

ResInt1	=	0 :	Alarm priority with the default value 100
	<>	0 :	Alarm priority

Note

The alarm priority influences the position in the alarm list (see description of the alarm server).

5.6.1 Message to the SINUMERIK

Function call

R_MESSAGE_M (Host,
Machine,
OrderNum,
Message,
Resint1,
Resint2,
Resbyte)

Direction of transfer: **Host computer → SINUMERIK**

Meaning

Display message on the operator panel of the SINUMERIK

Data

Table 5-7 Parameters of the message to SINUMERIK

Parameter	Description	Format
Host	Name of the host computer	16 bytes (string)
Machine	Machine name	16 bytes (string)
OrderNum	Order number	4 bytes (long int)
Message	Message text	128 bytes (string)
Resint1	Reserve 1	4 bytes (long int)
Resint2	Reserve 2	4 bytes (long int)
Resbyte	Reserve 3	8 bytes (string)

Notes on use

The host computer sends messages, e.g. about tools or materials to be prepared, to the machine operator to prepare for processing of the next part program.

Note

The display is via the alarm line in the header

Example

R_MESSAGE_M("FLR1", "BAZ3", 0, "Hello Machine", 0, 0, "\0");

Languages

RPC SINUMERIK generally supports all languages which are covered by the ASCII character set or for which a language package has been installed on the SINUMERIK.
Languages which use an extended character set (e.g. Chinese, Russian, etc.) are processed by SINUMERIK in DBCS format.
In this case, messages are displayed correctly on the PCU if the application on the host computer transmits the messages in DBCS format.

5.6.2 Message to the host computer

Called function

R_MESSAGE_H(Host,
Machine,
OrderNum,
Message,
Resint1,
Resint2,
Resbyte)

Direction of transfer: **SINUMERIK → Host computer**

Meaning

Display message on the host computer.

Data

Table 5-8 Parameters of the message to the host computer

Parameter	Description	Format
Host	Name of the host computer	16 bytes (string)
Machine	Machine name	16 bytes (string)
OrderNum	Order number	4 bytes (long int)
Message	Message text	128 bytes (string)
Resint1	Reserve 1	4 bytes (long int)
Resint2	Reserve 2	4 bytes (long int)
Resbyte	Reserve 3	8 bytes (string)

Notes on use

The machine operator informs the host computer of the preparations carried out for program execution.

Note

The text input is carried out via a dialog in the interactive program of RPC SINUMERIK.

Languages

RPC SINUMERIK generally supports all languages which are covered by the ASCII character set or for which a language package has been installed on the SINUMERIK.
Languages which use an extended character set (e.g. Chinese, Russian, etc.) are processed by SINUMERIK in DBCS format.
In this case, messages are displayed correctly on the host computer if the application on the host computer displays the messages in the same font as installed on the SINUMERIK.

5.7 Data dialogs

Description

Larger volumes of data, such as NC programs or tool data, are transferred in file format. Since the SINUMERIK can only act as an FTP client with the FTP protocol (File Transfer Protocol), the file transfer must always be carried out by the RPC SINUMERIK computer communication software on the SINUMERIK. The host computer uses the RPC call **R_DATA_M** to inform the RPC SINUMERIK computer communication software on the SINUMERIK that a file is ready for processing. The computer link software then fetches and processes the file. In the opposite direction, RPC SINUMERIK on the SINUMERIK transfers the file and uses the RPC call **R_DATA_H** to inform the host computer that a file is ready for processing on the host computer. Data dialogs refer to the PUT_/GET directories configured with SCCONFIG. Path names must always be specified from the view of RPC SINUMERIK . Both enabled drive names (shares) and the UNC notation can be used. If the files are located in the configured directories, with the parameter Name2, the file name can be specified without path. With FTP, the path names depend on the possibilities of the FTP server.

Program transfer

NC programs are transferred in file format. NC programs must be stored in the NC data management system after transfer.

Identification of file contents

The data in the R_DATA_M() and R_DATA_H() calls includes a subfunction number Subfnc identifying the type of data and thus the conditions required for correct transfer.
A file with tool data must be read tool-by-tool and processed according to the subfunction number specified. The files must be deleted by the processing function after processing.
When the computer link is started up, all the old data received must be deleted.
The subfunction numbers are the same for:

- Request,
- Transfer, and
- Delete.

Table 5-9 Subfunction numbers: SFct data dialogs

Sub-function no.	Function	Comments
1	NC program	Name1 = NC program Name2 = File name with path on the host computer
10	List of existing NC programs	Name1 = File path Name2 = Name of list file
20	Tool status data of **all** tools **Complete** set of tool data	Name1 = Empty Name2 = File name with path on the host computer
21	Tool status data of one tool Variant 1: **Complete** set of tool data	Name1 = ID number, Duplo number Name2 = File name with path on the host computer
22	Tool status data of one tool Variant 2: **Reduced** set of tool data	Name1 = ID number, Duplo number Name2 = File name with path on the host computer
23	Tool status data of one tool Variant 3: **Reduced** set of tool data	Name1 = ID number, Duplo number Name2 = File name with path on the host computer
24	Tool data of a tool with adapter number **Complete** set of tool data	Name1 = Adapter number Name2 = File name with path on the host computer
26	Optional/manual loading of a tool **Complete** set of tool data	Name1 = ID number, Duplo number Name2 = File name with path on the host computer
27	Optional/manual unloading of a tool	Name1 = ID number, Duplo number Name2 = File name with path on the host computer
28	Load tool from tool pallet	Name1 = Tool pallet number Name2 = Name of file with tool status data
29	Unload tool to tool pallet	Name1 = Tool pallet number Name2 = Name of file with tool status data
50	Machine assignment data	Name1 = Empty Name2 = File name with path on host computer
90	Transfer any file No further machining - optional	Name1 = Source file name with path Name2 = Target file with path

5.7.1 Request data as file from the SINUMERIK

Function call

T_DATA_M (Host,
Machine,
OrderNum,
SFct,
Name1,
Name2)

Direction of transfer: **Host computer → SINUMERIK**

Meaning

Request data as file from the SINUMERIK.

Data

Table 5-10 Parameters for data request from the SINUMERIK

Parameter	Description	Format
Host	Name of the host computer	16 bytes (string)
Machine	Machine name	16 bytes (string)
OrderNum	Order number	4 bytes (long int)
SFct	Subfunction number	4 bytes (long int)
Name1	For name see Table 5-9 Subfunction numbers:SFct	128 bytes (string)
Name2	For additional name see Table 5-9: Subfunction numbers:SFct	128 bytes (string)

Example

```
T_DATA_M ("FLR1", "BAZ3", 0, 1, "\mpf.dir\ Kw15.mpf", "\0");
T_DATA_M ("FLR1", "BAZ3", 0, 10, "\mpf.dir", "\0");
T_DATA_M ("FLR1", "BAZ3", 0, 20, "\0", "\0");
T_DATA_M ("FLR1", "BAZ3", 0, 21, "Drill10mm,0002", "\0");
T_DATA_M ("FLR1", "BAZ3", 0, 22, "Drill10mm,0002", "\0");
T_DATA_M ("FLR1", "BAZ3", 0, 23, "Drill10mm,0002", "\0");
T_DATA_M ("FLR1", "BAZ3", 0, 50, "\0", "\0");
T_DATA_M ("FLR1", "BAZ3", 0, 90, "f:\add_on\mcis_rpc.log", "\0");
```

5.7.2 Request data as file from the host computer

Called function

T_DATA_H (Host,
Machine,
OrderNum,
SFct,
Name1,
Name2)

Direction of transfer: **SINUMERIK → host computer**

Meaning

Request data as file from host computer.

Data

Table 5-11 Parameters for data request from the host computer

Parameter	Description	Format
Host	Name of the host computer	16 bytes (string)
Machine	Machine name	16 bytes (string)
OrderNum	Order number	4 bytes (long int)
SFct	Subfunction number	4 bytes (long int)
Name1	For name see Table 5-9 Subfunction numbers:SFct	128 bytes (string)
Name2	For additional name see Table 5-9 Subfunction numbers:SFct	128 bytes (string)

5.8 Accept transferred files

5.8.1 Request to machine: Accept data

Function call

R_DATA_M (Host,
Machine,
OrderNum,
SFct,
Name1,
Name2,
Date,
LastFile)

Direction of transfer: **Host computer → SINUMERIK**

Meaning

The host computer requests RPC SINUMERIK to fetch the specified file from the host computer and to process it (e.g. to include it in the data management system).

Data

Table 5-12 Parameters for accept request

Parameter	Description			Format
Host	Name of the host computer			16 bytes (string)
Machine	Machine name			16 bytes (string)
OrderNum	Order number			4 bytes (long int)
SFct	Subfunction number			4 bytes (long int)
Name1	For name see Table 5-9 Subfunction numbers:SFct			128 bytes (string)
Name2	For additional name see Table 5-9 Subfunction numbers:SFct			128 bytes (string)
Date	Date of last change (Unix time, only for NC program file)			4 bytes (long int)
LastFile	Last file of a workpiece			4 bytes (long int)
	Bit 7	**Bit 0**	**Function**	
	0	0	More files to follow	
	0	1	Last file of a workpiece or single file	
	1	0	Reset request identifier	

Notes on use

Bit 7 is only relevant if an NC program is requested from the host computer via the RK840D user interface, but the host computer intentionally transmits an NC program with a different name. By default, the RPC SINUMERIK internal request identifier is only reset if SFct and Name1 are identical; the host computer can initiate a reset of the request identifier via bit 7.

Example

R_DATA_M ("FLR1", "BAZ3", 0, 1, "\mpf.dir\ Kw15.mpf", "f:\ncpro\NCKW0815.txt", 862826400, 1);

R_DATA_M ("FLR1", "BAZ3", 0, 1, "WKS.DIR\Zylinderkopf.wpd\ Kw15.mpf", "f:\ncpro\NCKW0815.txt", 862826400, 1);
R_DATA_M ("FLR1", "BAZ3", 0, 10, "Main programs", "f:\tmp\NCListe.txt", 0, 1);
R_DATA_M ("FLR1", "BAZ3", 0, 26, "Drill10mm,0002", "f:\tmp\wzfile.txt");
R_DATA_M ("FLR1", "BAZ3", 0, 27, "TP003", " f:\tmp\tp003.txt ");
R_DATA_M ("FLR1", "BAZ3", 0, 28, "TP003", " f:\tmp\tp003.txt ");
R_DATA_M ("FLR1", "BAZ3", 0, 1001, "c:\mmc2\oemdata.txt", "c:\tmp\oemdata.txt");

R_DATA_M ("FLR1", "BAZ3", 0, 1, "WKS.DIR\Zylinderkopf.wpd\ Kw15.mpf", "NCKW0815.txt", 862826400, 1);

The file is fetched synchronously from the host computer during R_DATA_M, however processing cannot take place synchronously in the RPC. The return value of the RPC can therefore only indicate whether or not the file transfer was successful. After processing of the file, RPC SINUMERIK sends R_REPORT_H with type = 5 and error number = subfunction number (SFct) from R_DATA_M as a positive acknowledgment.

5.8.2 Request to host computer: Accept data

Called function

R_DATA_H (Host,
Machine,
OrderNum,
SFct,
Name1,
Name2,
Date,
LastFile)

Direction of transfer: **SINUMERIK → host computer**

Meaning

The file already transferred should be saved in the data management system on the host computer.

Data

Table 5-13 Parameters for accept request

Parameter	Description	Format
Host	Name of the host computer	16 bytes (string)
Machine	Machine name	16 bytes (string)
OrderNum	Order number	4 bytes (long int)
SFct	Subfunction number	4 bytes (long int)
Name1	For name see: Table 5-9 Subfunction numbers:SFct	128 bytes (string)
Name2	For additional name see: Table 5-9 Subfunction numbers:SFct	128 bytes (string)
Date	Date of last change (Unix time, only for NC program file)	4 bytes (long int)
LastFile	Last file of a workpiece 0: More files to follow 1: Last file of a workpiece or single file	4 bytes (long int)

5.9 Delete data

5.9.1 Delete data on the SINUMERIK

Function call

C_DELETE_M (Host,
Machine,
OrderNum,
SFct,
Name1,
Name2)

Direction of transfer: **Host computer → SINUMERIK**

Meaning

Delete data of previous file transfer.

Data

Table 5-14 Parameters for delete request

Parameter	Description	Format
Host	Name of the host computer	16 bytes (string)
Machine	Machine name	16 bytes (string)
OrderNum	Order number	4 bytes (long int)
SFct	Subfunction number	4 bytes (long int)
Name1	Name of file to be deleted	128 bytes (string)
Name2	Additional name	128 bytes (string)

Note

At present, only files in the data management can be accessed,
SFct = 1
e.g.: Name1 ="\mpf.dir\cylinderhead.mpf"

Example

C_DELETE_M ("FLR1", "BAZ3", 0, 1, "\mpf.dir\ Kw15.mpf", "\0");

5.10 NC programs

The following description of program handling is a special application of the functions described in **Section 5.7**. Knowledge of these functions is assumed.

5.10.1 Request NC program, initiated by host computer

1. Function call

T_DATA_M ()
SFct = 1
Name1 = Program name in the data management, e.g.: \mpf.dir\carrier4711.mpf
Direction of transfer **Host computer → SINUMERIK**

Meaning

The host computer requests a specific NC program from the SINUMERIK.

2. File transfer

The file with the requested NC program is transferred.

3. Called function

R_DATA_H()
SFct = 1
Name1 = Program name in the data management system
Name2 = Name including file path on the host computer
Date = Date of last change
Direction of transfer **SINUMERIK → Host computer**

Meaning

The SINUMERIK requests a specific program from the host computer.

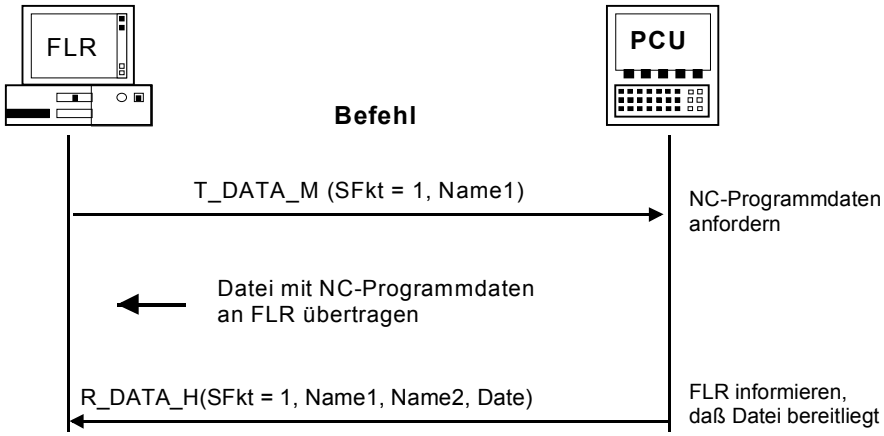


Fig. 5-4 Request NC program, initiated by host computer

5.10.2 Request NC program, initiated by SINUMERIK

1. Called function

T_DATA_H ()
SFct = 1
Name 1 = Program name in the data management system
Direction of transfer: **Host computer -> SINUMERIK**

Meaning

The SINUMERIK requests a specific program from the host computer.

2. Function call

R_DATA_M ()
SFct = 1
Name1 = Program name in the data management system
Name2 = Name including file path on the host computer
Date = Date of last change
Direction of transfer: **Host computer -> SINUMERIK**

Meaning

The host computer prepares a specific NC program for the SINUMERIK.

3. File transfer

RPC SINUMERIK transfers the file of the requested NC program to the SINUMERIK and to data management

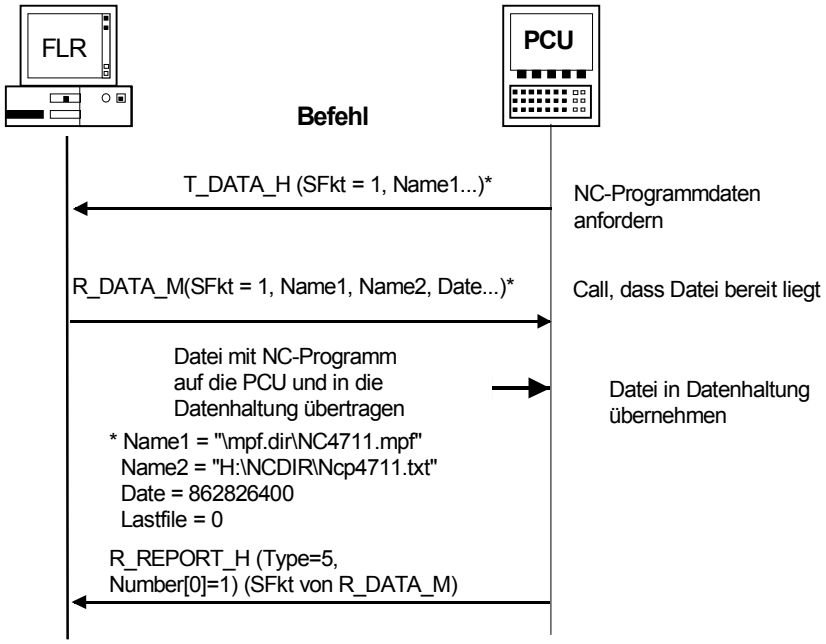


Fig. 5-5 Request NC program, initiated by the SINUMERIK

Note

NC programs can only be requested and transferred individually.

The workpiece carrier designation must be completed with '\0' and must have a maximum length of 6 bytes including '\0'.
The NC program must always include the data management path.
e.g.: NCProg = "\mpf.dir\cylinderhead.mpf"

Since data management in the SINUMERIK does not know any versions, the last modification date and the file size can be indicated. If the NC program is included in the data management of the SINUMERIK, but with another size or another modification date, RPC SINUMERIK must request the file from the host computer before the program may be activated for machining.

5.10.3 Transfer NC program

Function call

R_DATA_M ()
SFct = 1
Name1 = Program name in the data management system
Name2 = Name including file path on the host computer
Date = Date of last change
Direction of transfer: **Host computer → SINUMERIK**

Meaning

The host computer prepares a specific NC program for the SINUMERIK.

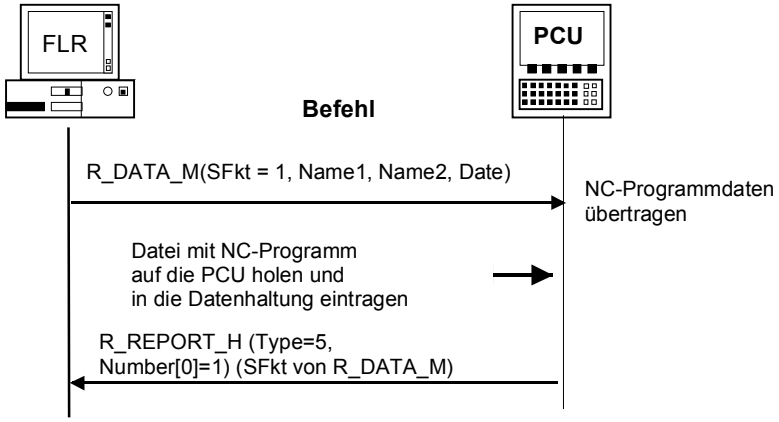


Fig. 5-6 Transfer NC program, initiated by host computer

Called function

R_DATA_H()
SFct = 1
Name1 = Program name in the data management system
Name2 = Name including file path on the host computer
Date = Date of last change

Direction of transfer: **SINUMERIK → Host computer**

Meaning

The SINUMERIK prepares a specific program for the host computer.

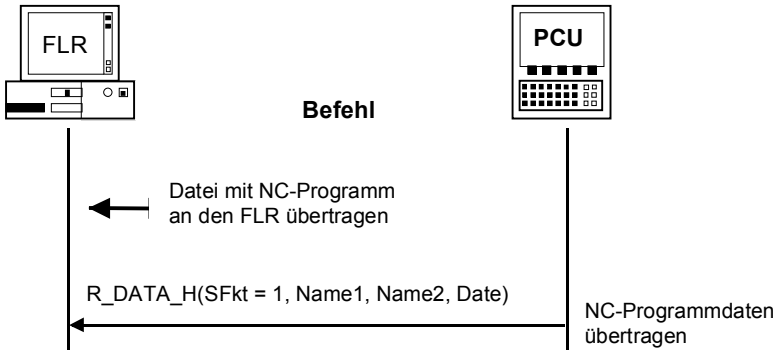


Fig. 5-7 Transfer NC program, initiated by the SINUMERIK

5.10.4 Delete programs on machine

Function call

C_DELETE_M (SFkt = 1, Name1)
Direction of transfer: **Host computer → SINUMERIK**

Meaning

The host computer instructs the SINUMERIK to delete the program specified by Name1.

Data

The data to be passed are described above under C_DELETE_M(). The following parameters must be assigned:
SFct = 1, Name1 = "\mpf.dir\zylinderkopf.mpf" or
"\spf.dir\4711.spf"

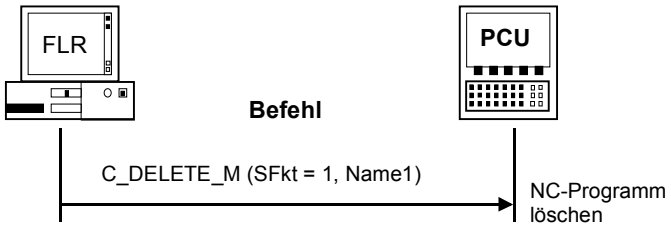


Fig. 5-8 Delete NC program, initiated by host computer

5.10.5 Request list of existing NC programs, initiated by host computer

1. Function call

T_DATA_M() mit
SFct = 10
Name 1 = Path in data management (e.g. "\mpf.dir")
Direction of transfer: **Host computer → SINUMERIK**

Meaning

Request from the host computer to the machine to transfer the list of existing programs.

2. File transfer

The file with NC program list is transferred to the host computer

3. Called function

R_DATA_H() with
SFct = 10
Name1 = Path in the data storage
Name2 = File name with NC program list
Direction of transfer: **SINUMERIK → host computer**

Meaning

Transfer from machine to the host computer: List of existing NC programs.

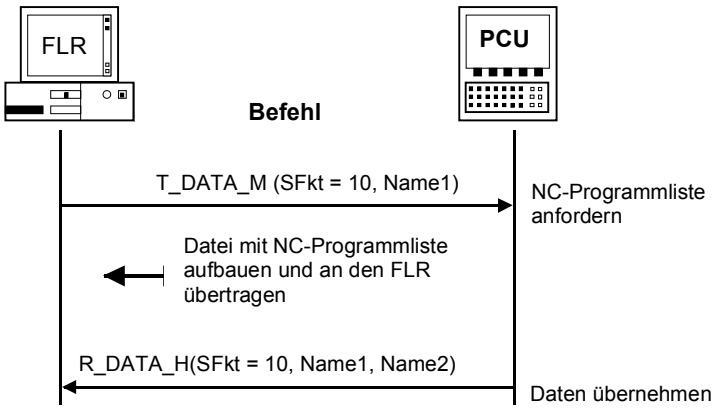


Fig. 5-9 Request list of existing NC programs, initiated by host computer

5.10.6 Request list of existing NC programs, initiated by SINUMERIK

1. Called function

T_DATA_H ()
SFct = 10
Name1 = Path in data management (e.g. "\mpf.dir")
Direction of transfer: **Host computer → SINUMERIK**

Meaning

Transfer from machine to the host computer:
Transfer list of existing NC programs.

2. Function call

R_DATA_M ()
SFct = 10
Name1 = Path in data management
Name2 = File name with NC program list

Direction of transfer: **Host computer → SINUMERIK**

Meaning

Request to the machine to accept the provided NC program list.

3. File transfer

RPC SINUMERIK accepts the file with the NC program list on the SINUMERIK.

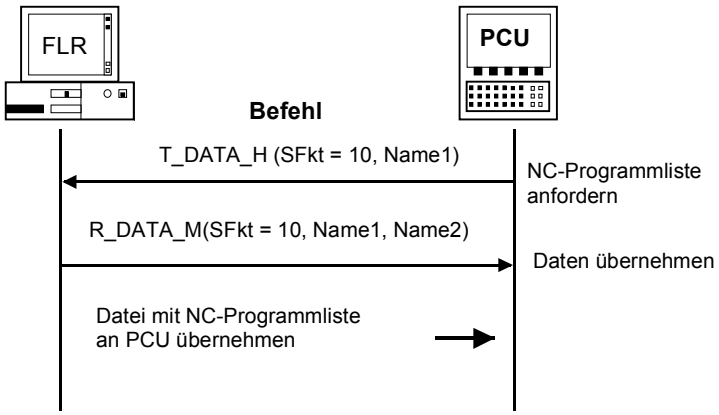


Fig. 5-10 Request list of existing NC programs, initiated by the SINUMERIK

5.10.7 Transfer NC program list

Function call

R_DATA_M() with
SFct = 10
Name1 = Path in the data management system for
Name2 = File name with NC program list
Direction of transfer: **Host computer → SINUMERIK**

Meaning

Transfer from host computer to machine: List of existing programs

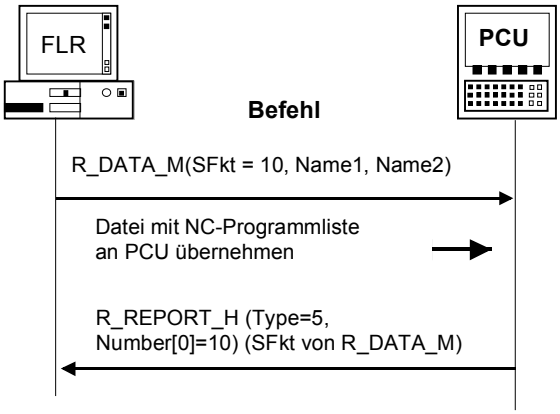


Fig. 5-11 Transfer NC program list

Called function

R_DATA_H() with
SFct = 10
Name1 = Path in the data storage
Name2 = File name with NC program list
Direction of transfer: **SINUMERIK → host computer**

Meaning

Transfer from machine to the host computer: List of existing programs

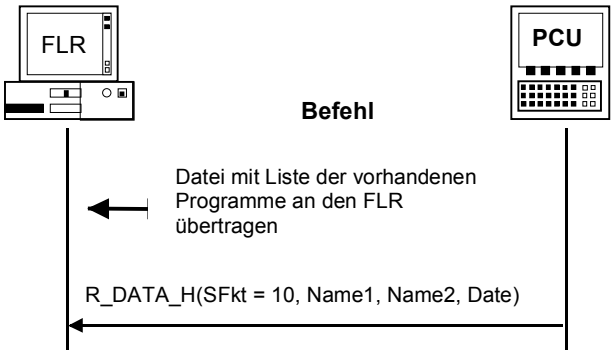


Fig. 5-12 Transfer NC program list to the host computer

The file with the NC program list includes a list of the files as well as subdirectories included in this data management path. The contents of these subdirectories are not listed and must be determined with a separate request, if necessary.

Structure of the file returned with R_DATA_H

\mpf.dir
Cylinderhead.MPF,FM,5320,876403708
Crankshaft.MPF,FN,8300,862826400

Rows

The first line of the file indicates the directory which has been stated for Name1 and the contents of which are listed. Each further line indicates a file or subdirectory name with the additional information, separated by comma.

Columns

The first column indicates the NC program name or the name of the subdirectory.
The second column includes two letters.

- The first indicates whether it is a file (F) or a subdirectory (D).
- The second character indicates whether the file is in the PCU or NCK.

Examples:
"Fx" - File
"Dx" - Directory
"xM" - on PCU
"xN" - in NCK or in NCK and PCU

The third column indicates the file size in bytes.

The fourth column indicates as a decimal number the **date** of the file as UNIX time in seconds since January 1, 1970.

The time for cylinder head 876403715 means October 9, 1997 15:28:35,
for crankshaft 862826400 it means May 5, 1997 12:00:00.

5.11 Tool dialogs

Tool data are always transferred in file format.
The file structure is described in **Section 4.1**.

5.11.1 Scan all tools in magazine

Direction of transfer	Command		Meaning
Host computer → SINUMERIK	T_DATA_M	(SFct = 20)	Request all tool data
Host computer ← SINUMERIK			Transfer file with all tool status data to the host computer
Host computer ← SINUMERIK	R_DATA_H	(SFct = 20, Name1=Empty, Name2=File with all tool status data)	Report arrival of file to the host computer

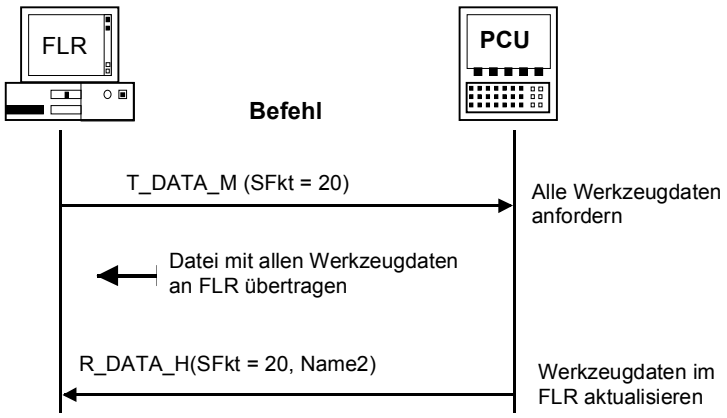


Fig. 5-13 Scan all tools, initiated by host computer

5.11.2 Tool data with tool adapter number (option)

Direction of transfer	Command		Meaning
Host computer ← SINUMERIK	T_DATA_H	(SFct = 24, Name1=Adapter number)	Request tool data with adapter number
Host computer → SINUMERIK	R_DATA_M	(SFct = 24, Name1=Adapter number, Name2=Filename with tool status data)	Transfer tool data with adapter number

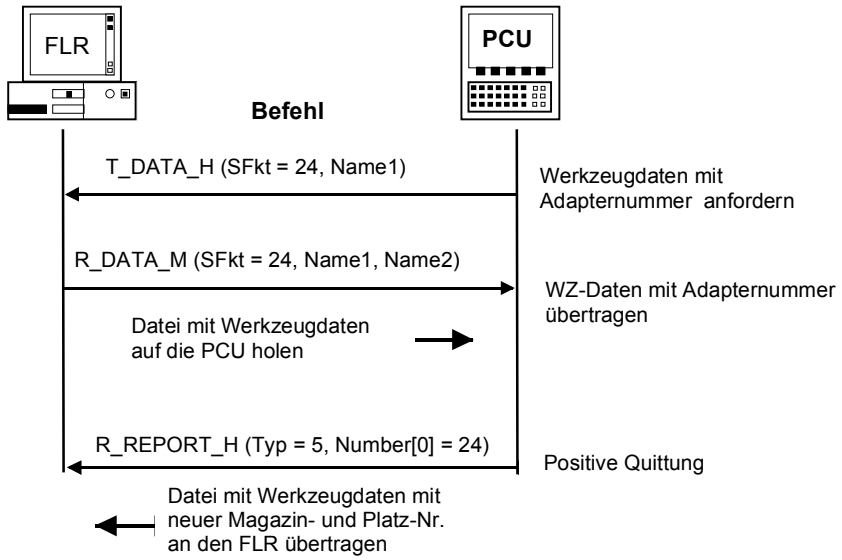


Figure 5-14 Tool data with tool adapter number (option)

5.11.3 Optional/manual loading

Direction of transfer	Command		Meaning
Host computer ← SINUMERIK	T_DATA_H	(SFct = 26, Name1= ID no., Duplo no.)	Request tool data
Host computer → SINUMERIK	R_DATA_M	(SFct = 26, Name1=ID no., Duplo no., Name2=Filename with tool status data)	Requests MMC to fetch file. Fetch file with tool status data on SINUMERIK. Load tool
Host computer ← SINUMERIK	R_DATA_H	(SFct = 21, Name1= ID no., Duplo no. Name2=Filename with tool status data)	File with tool status data with new magazine and location no. to the host computer (this is the load acknowledgment)

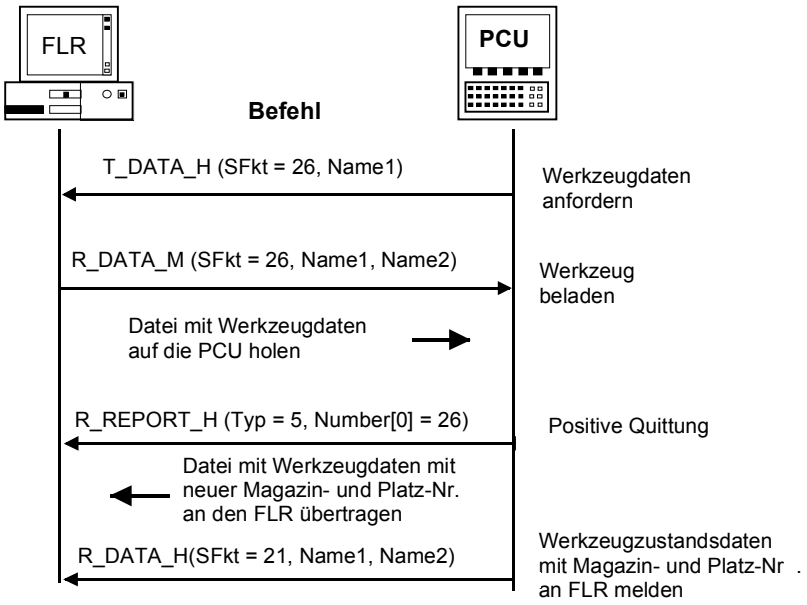


Fig. 5-15 Load tool, initiated by the SINUMERIK

5.11.4 Optional/manual unloading

Direction of transfer	Command		Meaning
Host computer ← SINUMERIK	R_DATA_H	(SFct = 27, Name1=ID no., Duplo no., Name2=Filename with tool status data)	File with tool status data with new magazine and location no. to the host computer (Unload acknowledgment)

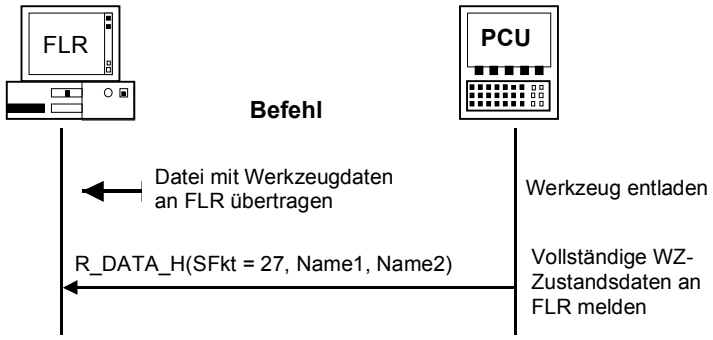


Fig. 5-16 Unload tool, initiated by the SINUMERIK

5.11.5 Report tool

Direction of transfer	Command		Meaning
Host computer ← SINUMERIK	R_DATA_H	(SFct = 21, Name1=ID no., Duplo no., Name2=Filename with tool status data)	File with tool status data to host computer (report tool)

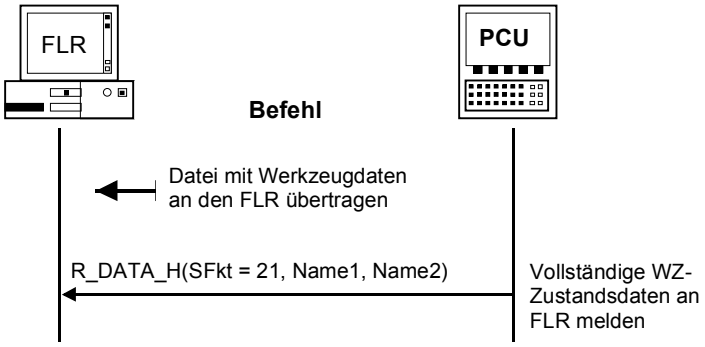


Fig. 5-17 Report tool, initiated by the SINUMERIK

5.11.6 Load tool pallet/cassette (option)

Direction of transfer	Command		Meaning
Host computer → SINUMERIK	R_DATA_M	(SFct = 27, Name1=Tool pallet number, Name2=File with tool status data)	Fetch file with tool status data on the SINUMERIK
Host computer ← SINUMERIK	R_DATA_H	(SFct = 20, Name1= ID no., Duplo no., Name2=Filename with tool status data)	File with tool status data with new magazine and location no. to the host computer

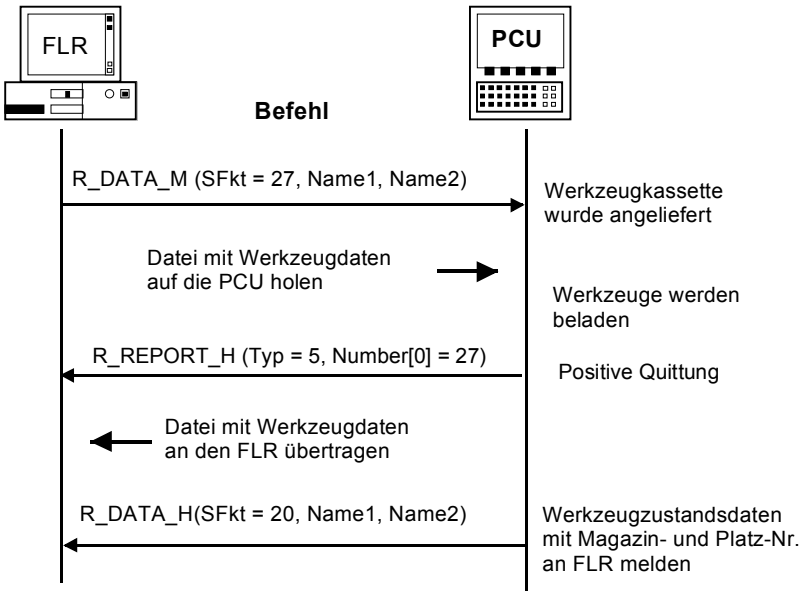


Fig. 5-18 Load tool pallet/cassette

5.11.7 Unload tool pallet/cassette (option)

Direction of transfer	Command		Meaning
Host computer → SINUMERIK	R_DATA_M	(SFct = 28, Name1=Tool pallet number, Name2=File with tool status data)	Fetch file with tool status data on the SINUMERIK Tools are unloaded
Host computer ← SINUMERIK	R_DATA_H	(SFct = 20, Name1= ID no., Duplo no., Name2=Filename with tool status data)	File with tool status data with new magazine and location no. to the host computer

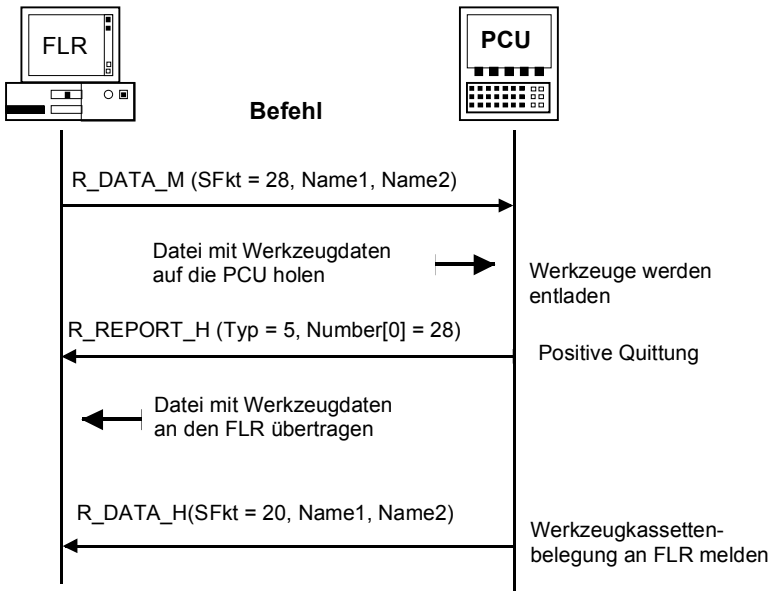


Fig. 5-19 Unload tool pallet/cassette

5.12 Machine assignment data

On machines with more than 3 docking positions/storage locations, e.g. where transport systems with storage locations are used, the data of all docking positions/storage locations are transferred in an ASCII file. After the file has been transferred to the host computer, the RPC call **R_DATA_H (SFct = 50)** is initiated (see **Subsection 5.8.2**).
For each docking position/storage location,

- the docking position number
- the workpiece carrier number and
- the workpiece carrier status

are transferred.
The data fields are stored in the file with **comma** separated as follows:

Structure of ASCII file

DockPos 1, DockPosStatus, WPC, WPCStatus **<Line feed>**
DockPos 2, DockPosStatus, WPC, WPCStatus **<Line feed>**
...
DockPos n, DockPosStatus, WPC, WPCStatus **<EOF>**

Table 5-15 Description of file parameters

Parameter	Description	Format
Dockpos	Docking position number	ASCII
DockPosStatus	Docking position status 0: Enabled 1: Disabled for transport control system 2: faulty	ASCII
WPC	Workpiece carrier name	ASCII
WPCStatus	Workpiece carrier status 1: Not machined, no program assigned 2: Not machined, program assigned 16: In progress 32: Finished 64: Finished with errors 128: Only for buffering	ASCII

The host computer can request these data with T_DATA_M(SFct = 50).

5.13 General order function

C_ORDER_M (Host,
Machine,
OrderNum,
SFct,
Name1,
Name2,
Name3,
Name4,
Parameter1,
Parameter2,
Parameter3,
Parameter4)

Direction of transfer: **Host computer → SINUMERIK**

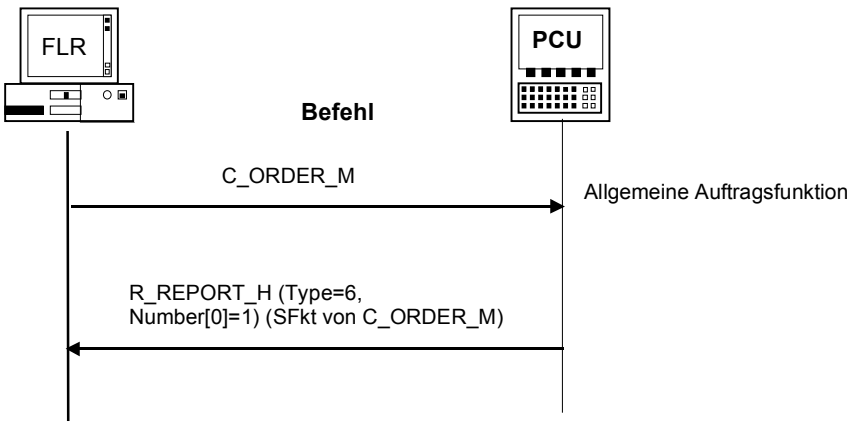


Fig.5-20 General order function

Meaning

After execution, RPC SINUMERIK transmits either R_REPORT_H with type= 6 and error number=SFct as a positive acknowledgment of C_ORDER_M or R_REPORT_H with type= 4 and an appropriate error number as a negative acknowledgment.

Table 5-16 Subfunction number SFct for the general order function

Subfunction no. (SFct)	Function	Comments	
2	Loading of NC programs	Name1 = Name2 = Parameter1=	NC program name NCK name (optional) Function (0,2,3)
3	Loading and selection of NC programs	Name1 = Name2 = Parameter1= Parameter2=	NC program name NCK name (optional) Function (0-3) Channel number
4	Unloading of NC programs	Name1 = Name2 =	NC program NCK name
5	Selection of NC programs	Name1 = Name2 = Parameter1= Parameter2=	NC program name NCK name (optional) Function (0,1) Channel number
6	Selection of NC programs	Name2 = Parameter2=	NCK name (optional) Channel number
100	Setting date/time on the SINUMERIK/PLC	Name2 = Parameter1 Parameter2=	NCK name (optional) UNIX time Date/time SINUMERIK / PLC
200	Set protection level	Name2 = Parameter1	NCK name (optional) Protection level
201	Reset protection level	Name2 =	NCK name (optional)

Note

The NCK name can only be used after the required extensions have been made to the files NETNAMES.INI and MMC.INI.

Data

Table 5-17 Parameter mode switching

Parameter	Description	Format
Host	Name of the host computer	16 bytes (string)
Machine	Machine name	16 bytes (string)
OrderNum	Order number	4 bytes (long int)
SFct	Subfunction number	4 bytes (long int)
Name1	*	128 bytes (string)
Name2	*	128 bytes (string)
Name3	*	128 bytes (string)
Name4	*	128 bytes (string)
Parameter1	*	4 bytes (long int)
Parameter2	*	4 bytes (long int)
Parameter3	*	4 bytes (long int)
Parameter4	*	4 bytes (long int)

* For additional name see Table 5-23 Subfunction numbers SFct for the general order function

5.13.1 Load NC programs

Function call

C_ORDER_M () with
SFct=2
Name1 = NC program name
Name2 = NCK name (optional)

Parameter1=	0:	Load single NC program into NCK
	2:	Load complete workpiece into NCK
	3:	Load workpiece into NCK, with exception of the NC program specified in Name1

Direction of transfer: **Host computer → SINUMERIK**

Example

```
C_ORDER_M ("FLR1", "BAZ3", 0, 2, "\mpf.dir\kw15.mpf", "\0", "\0", "\0",0 ,0 ,0 ,0);
C_ORDER_M ("FLR1", "BAZ3", 0, 2, "\mpf.dir\kw15.mpf", "NCU_2", "\0",
"\0",0 ,0 ,0 ,0);
```

Meaning

Loading of individual NC programs or workpieces from the SINUMERIK data management into the NCK. The NCK name (Name2) must only be specified if the NC program need to be loaded into the standard NCK.

Error code

-262: NC program cannot be loaded

5.13.2 Load and select NC programs

Function call

C_ORDER_M () with
SFct=3
Name1 = workpiece / NC program name
Name2 = NCK name (optional)

Parameter1=	0:	Load NC program into NCK and select
	1:	Execute individual program from external
	2:	Load complete workpiece into NCK and select NC program from Name1
	3:	Load complete workpiece into NCK and select the NC program specified in Name1 for execution from external.
Parameter2=		Channel number

Direction of transfer: **Host computer → SINUMERIK**

Example

```
C_ORDER_M ("FLR1", "BAZ3", 0, 3, "\mpf.dir\kw15.mpf", "\0", "\0", "\0",0 ,2 ,0 ,0);
C_ORDER_M ("FLR1", "BAZ3", 0, 3, "\WKS.DIR\Zylinderkopf.wpd\Kw15.mpf",
"NCU_2", "\0", "\0",0 ,0 ,0 ,0);
```

Meaning

The function corresponds to SFct2. In addition, the specified program is selected. The NCK name (Name2) must only be specified if the NC program need to be loaded into the standard NCK.

Note

If the NC program is then to be started automatically or if the NC program is to be displayed for the operator, this has to be implemented via the RPC SINUMERIK service R_VAR_M.

Error code

-263: NC program cannot be loaded and selected

5.13.3 Unload NC programs

Function call

C_ORDER_M () with
SFct=4
Name1 = NC program name or workpiece name
Name2 = NCK name (optional)

Direction of transfer: **Host computer → SINUMERIK**

Example

```
C_ORDER_M ("FLR1", "BAZ3", 0, 4, "\mpf.dir\kw15.mpf", "\0", "\0", "\0",0 ,0 ,0 ,0);
C_ORDER_M ("FLR1", "BAZ3", 0, 4, "\mpf.dir\kw15.mpf", "NCU_2", "\0",
"\0",0 ,0 ,0 ,0);
C_ORDER_M ("FLR1", "BAZ3", 0, 4, "\WKS.DIR\Zylinderkopf.wpd\Kw15.mpf", "\0",
"\0", "\0",0 ,0 ,0 ,0);
```

Meaning

Unloading of NC programs or workpieces from the NCK. The NC programs are then in the PCU data management.
The NCK name (Name2) must only be specified if it is not to be loaded into the standard NCK.

Error code

-264: NC program cannot be unloaded

5.13.4 Select NC programs

Function call

C_ORDER_M () with
SFct=5
Name1 = NC program name
Name2 = NCK name (optional)
Parameter1= 0: Execution on NCK
1: Execution from external
Parameter2= Channel number

Direction of transfer: **Host computer → SINUMERIK**

Example

C_ORDER_M ("FLR1", "BAZ3", 0, 5, "\mpf.dir\kw15.mpf", "\0", "\0", "\0",1 ,5 ,0 ,0);
C_ORDER_M ("FLR1", "BAZ3", 0, 5, "\mpf.dir\kw15.mpf", "NCU_2", "\0",
"\0",0 ,1 ,0 ,0);

Meaning

Selection of NC programs in a certain channel. The function corresponds to SFct 3 **without** prior loading of the NC program.
The NCK name (Name2) must only be specified if the NC program is not to be loaded into the standard NCK.

Error code

-265: NC program cannot be selected

5.13.5 Deselect NC programs

Function call

C_ORDER_M () with
SFct=6
Name2 = NCK name (optional)
Parameter2 =Channel number

Direction of transfer: **Host computer → SINUMERIK**

Example

```
C_ORDER_M ("FLR1", "BAZ3", 0, 6, "\0", "\0", "\0", "\0",0 ,5 ,0 ,0);
C_ORDER_M ("FLR1", "BAZ3", 0, 6, "\0", "NCU_2", "\0", "\0",0 ,1 ,0 ,0);
```

Meaning

Deselection of the current NC program.
The NCK name (Name2) must only be specified if it is not the standard NCK.

Error code

-266: NC program cannot be deselected.

5.13.6 Set protection level

Function call

C_ORDER_M () with
SFct=200
Name2 = NCK name (optional)
Parameter1= Protection level:
0=system password
1=manufacturer password
2=service password
3=user password

Direction of transfer: **Host computer → SINUMERIK**

Example

C_ORDER_M ("FLR1", "BAZ3", 0, 200, "\0", "\0", "\0", "\0",2 ,0 ,0 ,0);

Meaning

Set protection level (in the example: protection level 2)
The NCK name (Name2) must only be specified if it is not the standard NCK.

Certain tasks can be performed only when a specific protection level is set. This call allows the host computer to set a protection level temporarily, e.g. before the data management system is accessed.

Note

This protection level is valid for the whole SINUMERIK system! The original protection level must be set again as soon as possible to protect the system against misuse!

Error code

-271: "Set protection level" has not worked.

5.13.7 Reset protection level

Function call

C_ORDER_M () with
SFct=201
Name2 = NCK name (optional)

Direction of transfer: **Host computer → SINUMERIK**

Example

C_ORDER_M ("FLR1", "BAZ3", 0, 201, "\0", "\0", "\0", "\0",0 ,0 ,0 ,0);

Meaning

Reset protection level.

Error code

-272: "Reset protection level" has not worked.

5.14 Set time/date on SINUMERIK/PLC

Function call

C_ORDER_M () with
SFct=100
Name2 = NCK name (optional)

Parameter1=		UNIX time
Parameter2=	0:	Date/time on SINUMERIK
	1:	Date/time on SINUMERIK and PLC
	2:	Date/time on PLC

Direction of transfer: **Host computer → SINUMERIK**

Example

```
C_ORDER_M ("FLR1", "BAZ3", 0, 100, "\0", "\0", "\0", "\0",862826400 ,0 ,0 ,0);
C_ORDER_M ("FLR1", "BAZ3", 0, 100, "\0", "NCU_2", "\0",
"\0",862826400 ,1 ,0 ,0);
```

Meaning

Setting of date and time on the SINUMERIK / PLC.
The NCK name (Name2) must only be specified if the date or the time need not to be changed on the standard NCK. Parameter2 specifies the target device of the date/time change.

Error code

-270: Time/date cannot be updated.

5.15 MODE switchover

Function call

C_MODE_M (Host,
Machine,
OrderNum,
Mode)

Direction of transfer: **Host computer → SINUMERIK**

Meaning

Request to the machine to change to a certain mode.

Data

Table 5-18 Parameter mode switching

Parameter	Description	Format
Host	Name of the host computer	16 bytes (string)
Machine	Machine name	16 bytes (string)
OrderNum	Order number	4 bytes (long int)
Mode	Mode 1: Special mode ON 2: Special mode OFF 3: Deactivate components 4: Activate components 5: On request from RPC SINUMERIK set bit 4 = 1 6: On request from RPC SINUMERIK set bit 4 = 0 7: On request from RPC SINUMERIK set bit 5 = 1 8: On request from RPC SINUMERIK set bit 5 = 0	4 bytes (long int)

Modes 5 to 8 can be used for project-specific activation and deactivation of special functions.

Example

C_MODE_M ("FLR1", "BAZ3", 0, 3);

5.15.1 Special mode

Description

In special mode, the automatic material flow means that no workpiece carriers are delivered to the machine; there is no program assignment by the host computer as a result of messages indicating the arrival of a workpiece carrier. Manual transport actions can be used to deliver workpiece carriers; the user must select and start the NC program manually.
Workpiece carriers which were delivered by means of automatic material flow continue to be transported automatically in special mode. Workpiece carriers which were delivered by means of a manual transport action must be transported away again manually.
The special mode for a machine can be activated at any time on the host computer; a machining operation which is in progress is finished in the normal way.
As soon as the special mode is activated, the workpiece carrier to be machined can be delivered manually; the machine does **not** have to cruise to a stop.
Mode = 1 Activate special mode
Mode = 2 Deactivate special mode

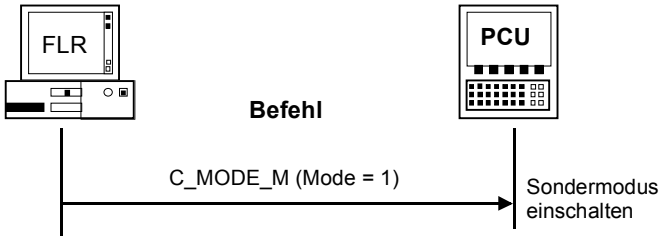


Fig. 5-21 Activate special mode, initiated by the host computer

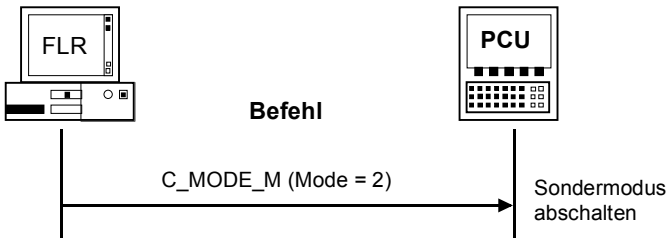


Fig. 5-22 Deactivate special mode, initiated by the host computer

5.15.2 Activate/deactivate components

A request is required in order to deactivate drives or other plant components from the host computer at the end of work. An activation request is also required at the start of work.
It is irrelevant for the computer link whether deactivation is requested at a specific time or after the last workpiece carrier has been machined. The computer link only relays the request to the PLC via the DB interface. The PLC must verify that machining is not in progress before it deactivates the components.
After deactivation of the components, this condition must be reported to the host computer with **R_MACHINE_H ()**; the same applies after activation.

Mode = 3 Deactivate components
Mode = 4 Activate components

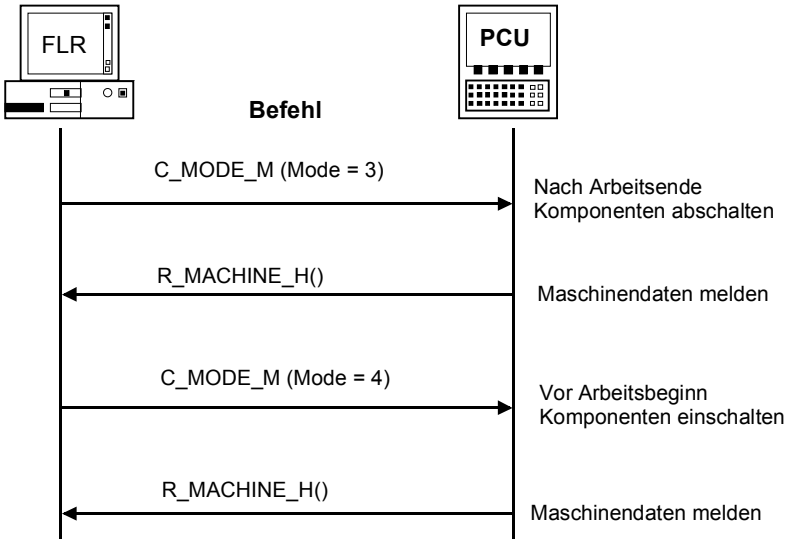


Fig. 5-23 End of work, initiated by the host computer

5.16 Synchronization

Synchronization means transmitting current data to the host computer so that the system image on the computer matches the real situation. This is necessary if the computer or the machine is restarted, or after an interruption in the connection. The machine must not start a new machining operation during synchronization; the operation currently in progress is not affected by the synchronization procedure. The host computer uses **T_MACHINE_M ()** to request the machine status data from the machine, and uses **T_DATA_M (SFct = 50)** to request the machine assignment data; it then transfers the program assignment **R_NC4WPC_M ()** for all sides of those workpiece carriers which have not yet been finished.

If the host computer is responsible for tool preparation/accounting, all the tools should be scanned after a (lengthy) interruption in the connection; this should be performed after synchronization and ensures that the host computer has access to the latest tool data. The tool scan does not take place automatically; it must be requested by the host computer with **T_DATA_M (SFct =20)**.

5.16.1 Synchronization start/end

Function call

C_SYNCH_M (Host,
Machine,
OrderNum,
SynchFlag)

Direction of transfer: **Host computer → SINUMERIK**

Meaning

The host computer initiates or terminates a synchronization process.

Data

Table 5-19 Description of synchronization parameters

Parameter	Description	Format
Host	Name of the host computer	16 bytes (string)
Machine	Machine name	16 bytes (string)
OrderNum	Order number	4 bytes (long int)
SynchFlag	Start/end flag 1: Start 0: End	4 bytes (long int)

Notes on use

The following section explains the interaction that takes place during the synchronization process.

Example

C_SYNCH_M ("FLR1", "BAZ3", 0, 1);

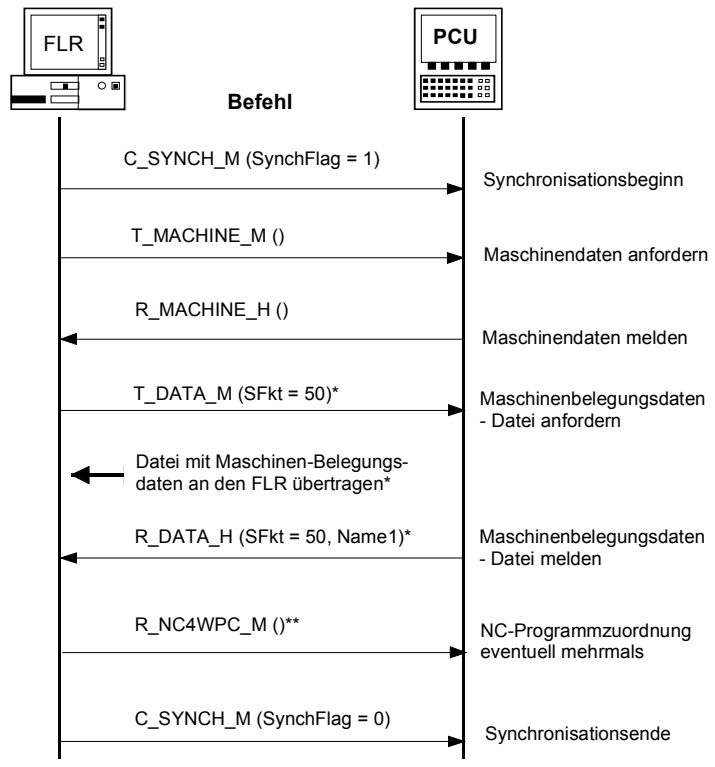
5.16.2 Synchronization sequence

Host computer → SINUMERIK C_SYNCH_M () SynchFlag = 1 //Start
Host computer → SINUMERIK T_MACHINE_M ()
Host computer ← SINUMERIK R_MACHINE_H ()

On machines with more than 3 storage locations or a transport system, this is followed by:
Host computer → SINUMERIK T_DATA_M () SFct = 50
Host computer ← SINUMERIK Transfer machine assignment file
Host computer ← SINUMERIK R_DATA_H () SFct = 50, Name1 = file name

For all workpiece carriers which have not yet been finished and all their sides, the sequence is as follows
Host computer → SINUMERIK Program assignment R_NC4WPC_M ()

Host computer → SINUMERIK C_SYNCH_M () SynchFlag = 0 //End



Schritte * : Bei Maschinen mit mehr als 3 Liegeplätzen (z.B. Transportsystem)
Schritte** : Für alle noch nicht fertig bearbeitete Werkstückträger und alle deren Seiten je einmal

Fig. 5-24 Synchronization, initiated by host computer

■

6 Data Communication for OEM Applications

In order to send data from any OEM application on the SINUMERIK across the computer link to the host computer and back, RPC SINUMERIK on the SINUMERIK must be able to communicate with other applications.
Since communication between programs on the MMC takes place via dynamic data exchange (DDE), DDE is also used for communication between the computer link and OEM applications.
Functions are interfaces are defined for the transfer of data in both directions.

6.1 Data transfer to an OEM application

Function call

R_DDEDATA_M (Host,
Machine,
OrderNum,
Application,
Topic,
Item,
Data**)**

Direction of transfer: **Host computer → SINUMERIK**

Meaning

This RPC call initiates the transfer of user data to the specified OEM application via DDE. All the necessary parameters are passed with R_DDEDATA_M.
Since it is not possible to establish a DDE connection during the RPC, the data transfer to the OEM application can only take place after the return of R_DDEDATA_M.
One DDE connection is set up for each R_DDEDATA_M (). Before copying the user data, RPC SINUMERIK inserts the host name followed by the 'pipe' symbol ('|' ASCII code 124) as sender information for the OEM application. This complete string is transferred to the OEM application with DDE-POKE after which the DDE connection is closed.
When the data have been transferred, RPC SINUMERIK sends R_REPORT_H with type = 5 and error number = 1000 as a positive acknowledgment for the sender of R_DDEDATA_M.

Note

It is permissible to transfer only ASCII data as user data, not binary data.
The string ends with binary 0.

Data

Table 6-1 Parameter description for data transfer to host computer

Parameter	Description	Format
Host	Name of the host computer	16 bytes (string)
Machine	Machine name	16 bytes (string)
OrderNum	Order number	4 bytes (long int)
Application	Application name	64 bytes (string)
Topic	DDE Topic	64 bytes (string)
Item	DDE Item	64 bytes (string)
Data	DDE User data	32 KB (string)

Languages

RPC SINUMERIK generally supports all languages which are covered by the ASCII character set or for which a language package has been installed on the SINUMERIK.
Languages which use an extended character set (e.g. Chinese, Russian, etc.) are processed by SINUMERIK in DBCS format.
In this case, data are displayed correctly on the PCU if the application on the host computer transmits the data in DBCS format.

6.2 Data from an OEM application to the host computer

Called function

R_DDEDATA_H(Host,
Machine,
OrderNum,
Data)

Direction of transfer: **Host computer ← SINUMERIK**

Meaning

This RPC is sent to the host computer if the computer link has received a specific DDE call. The associated scope of functions and parameters are described below.

Data

Table 6-2 Parameter description for data transfer to host computer

Parameter	Description	Format
Host	Name of the host computer	16 bytes (string)
Machine	Machine name	16 bytes (string)
OrderNum	Order number	4 bytes (long int)
Data	DDE User data	32 KB (string)

Languages

RPC SINUMERIK generally supports all languages which are covered by the ASCII character set or for which a language package has been installed on the SINUMERIK.
Languages which use an extended character set (e.g. Chinese, Russian, etc.) are processed by SINUMERIK in DBCS format.
In this case, messages are displayed correctly on the PCU if the application on the host computer transmits the messages in DBCS format.

6.2.1 DDE between an OEM application and RPC SINUMERIK

To enable an OEM application to transfer data to RPC SINUMERIK on the SINUMERIK via DDE, RPC SINUMERIK acts as a DDE server.
The OEM application must set up a DDE connection with the following DDE parameters and pass the data with POKE:

Application = Sincom*,
Topic = OEM,
Item = SendData.

Note

For compatibility reasons the application name is still Sincom.

The data string transferred with the DDE call must contain the host names to which the data are to be passed in front of the actual user data. The 'pipe' symbol ('|' ASCII code 124) should be used as the separator (host name|user data).
When the data have been received with the above R_DDEDATA_H () call, the user data string is passed on to the specified host. The host name must be known to the computer link (see Chapter 5).

6.2.2 File transfer from OEM applications to the host computer

A DDE call is used to transfer files from an OEM application across the computer link to the host computer; this call uses the RPC function R_DATA_H, which is already available. The application name and Topic are the same as in the above DDE call; File should be specified as the Item:
Application = Sincom,
Topic = OEM,
Item = PutFile.
The data string must contain the parameters required for the R_DATA_H call; the 'pipe' symbol ('|' character code 124) must be used as the separator.
Structure of the data string: Host|SFct|Name1|Name2|Date|LastFile.
SFct must be a number greater than 1000; it is used for the file assignment on the host computer.
Name1 is the file name on the SINUMERIK (complete with drive and path)
Name2 is the file name on the host computer (also complete)
Date is optional; it contains the date/time of the file in the Unix time format (in seconds).
LastFile is optional, see Chapter 5.

6.2.3 File transfer from the host computer to the OEM application

R_DATA_M (SFct = 90) is used to transfer the file from the host computer to the SINUMERIK. If SFct 90, the file is merely fetched onto the SINUMERIK; no further processing takes place. A message informing the OEM application about the file received can be transmitted from the host computer with R_DDEDATA_M ().
File transfer for OEM applications only has to be handled by the computer link software if the transfer is to take place using FTP.
If the SINUMERIK can access the drives of the host computer directly across the network, the OEM applications can fetch files from the host computer and copy files to the host computer themselves.

■

7 Configurable Data Transfer/Variable Service

7.1 Description

With the variable services RPC SINUMERIK can read and write any project-specific data from the PLC and NCK. This function is implemented based on the variable service of the OEM package PCU using the NC-DDE server.
The host computer can request data with **T_VAR_M ()** that are then obtained by RPC SINUMERIK and passed to the host computer with **R_VAR_H ()**. The host computer can also pass on data with **R_VAR_M ()** that are then written by RPC SINUMERIK.
In the Ini file SCVARSET.INI in the ADD_ON directory, you can define variable sets that can be used for read and write processes. If the mode is configured correctly, changes in these variables are automatically signaled to the host computer by RPC SINUMERIK with **R_VAR_H ()**.
For the SCVARSET.INI, the following definitions apply:
No more than 10 sets can be defined
No more than 10 variables can be defined in a set
No more than 50 variables can be defined within all sets
No more than 10 hotlinks can be defined within all sets
As long as these limits are observed, combinations can be made as required.

RPC SINUMERIK contains the optional extension allowing the required NCU to be parameterized in the variable set if more than one NCU is connected to a SINUMERIK. However, that only applies to variable sets that are defined in the SCVARSET.INI, not if the data description is stated directly in parameter VarDescr for **T_VAR_M ()** or **R_VAR_M ()**!

7.1.1 Structure of the SCVASRSET.INI file

Each variable set is a section in the Ini file and consists of the section name in square brackets, the access mode, an optional host name, an optional NCU link, and the variable definitions.
If no host name is configured, **R_VAR_H ()** will be transmitted to all hosts configured for this RPC when data is changed.
If no NCU link is configured, the default link will be used.
The name of the variable set and the name of the host must not be longer than 16 characters, NCU links must match the configurations in the file NETNAMES.INI.

Access modes

0	No hotlink is set up, the variables of the set are only read at the request of the host computer (T_VAR_M).
1*	A hotlink is set up for each variable in the set,
2*	A hotlink is set up for the first variable in the set only
3*	With handshake. As with 2 a hotlink is set up for the first variable in the set. After a hotlink, once all variables have been read, the first variable is reset to 0 (handshake). Variable 1 must not define a field, only a single variable. Once the variable has been reset to 0, a time delay is required before it can be set again. If the variable is set again too soon, the change in state might not be detected.

* In the case of a hotlink, all variables of the corresponding variable set are read and transmitted to the host computer with R_VAR_H.

SCVASRSET.INI for RPC SINUMERIK variable service

```
; Name of the variable set
[MeasData]
;Access mode
Mode=2
;Hostname (optional)
Host=H1
;NCU link (optional)
Connection=NCU_1
VAR01=/Plc/DataBlock/Word[c50,0]
VAR02=/Plc/Datablock/Byte[c50,1,20]("!l%d,")
VAR03=/Channel/Parameter/R[5]

[Set02]
Mode=3
Host=FLR2
Var01=/Plc/Datablock/Byte[c50,1]
Var02=/Plc/Datablock/Byte[c50,2]
Var03=/Plc/Datablock/Byte[c50,3,20]("!l%d,")
Var04=/Plc/Datablock/Byte[c50,4]
Var05=/Plc/Datablock/Byte[c50,5,20]("!l%d,")
Var06=/Plc/Datablock/Byte[c50,6]
Var07=/Plc/Datablock/Byte[c50,7,20]("!l%d,")
Var08=/Plc/Datablock/Byte[c50,8]
Var09=/Plc/Datablock/Byte[c50,9,20]("!l%d,")
Var10=/Plc/Datablock/Byte[c50,10]

[Set03]
Mode=0
Var01=/Plc/Datablock/Byte[c51,0,10]("!l%d,")
Var02=/Plc/Datablock/Byte[c51,30,50]("!l%d,")
```

Note

The SINUMERIK software contains a test tool with the name DDETEST. EXE. With it, you can test variable access with the running NCDDE server. You must first test all the variables that you want to use in the SCVARSET.INI by this method.
The appendix includes a chapter of the OEM description that contains a description of the interface between PCU50 and NCK/PLC.
The SCTEST setup contains the help file Btss_gr.hlp (and Btss_uk.hlp for English). These help files contain all information for variable access.
Access to data via $ variables, e.g. $TC_TP1 (T number) is not implemented with this service.

7.2 Transfer data

7.2.1 Variable data transfer to the machine

Function call

R_VAR_M (Host,
Machine,
OrderNum,
VarMode
VarSet
VarDescr
VarData)

Direction of transfer: **Host computer → SINUMERIK**

Meaning

Transfer data from the host computer to RPC SINUMERIK; the data are then written to the PLC or NCK.
M call R_VAR_M can specify either the name of a variable set from SCVARSET.INI with VarSet or the variable descriptions with VarDescr.
When the data have been transferred, RPC SINUMERIK sends R_REPORT_H with type=5 and error number = 0 as a positive acknowledgment for the sender of R_VAR_M.

Data

Table 7-1 Variable data transfer to the machine

Parameter	Description	Format
Host	Name of the host computer	16 bytes (string)
Machine	Machine name	16 bytes (string)
OrderNum	Order number	4 bytes (long int)
VarMode	Variable mode	4 bytes (long int)
VarSet	Name of the variable set	16 bytes (string)
VarDescr *	Data description	1024 bytes (string)
VarData *	User data	10 KB (string)

The variables in both the data description and the user data must be separated by the 'pipe' symbol ('|' ASCII code 124).

Example 1

SET03 is defined in SCVARSET.INI:
[Set03]
Mode=1
Host=FLR2
Var01=/Plc/Datablock/Byte[c51,0,10]("!l%d,")
Var01=/Plc/Datablock/Byte[c51,30,50]("!l%d,")
R_VAR_M ("FLR1", "BAZ3", 0, 0, "Set03", "\0", "33|50");

Example 2

R_VAR_M ("FLR1", "BAZ3", 0, 0, "\0",
"/Plc/DataBlock/Word[c50,20]|/Plc/DataBlock/Word[c60,30]|", "33|50");

7.2.2 Variable data transfer to the host computer

Called function

R_VAR_H (Host,
Machine,
OrderNum,
VarMode
VarSet
VarDescr
VarData**)**

Direction of transfer: **Host computer ← SINUMERIK**

Meaning

RPC SINUMERIK transfers data read from the PLC or NCK to the host computer.

Data

Table 7-2 Variable data transfer to the host computer

Parameter	Description	Format
Host	Name of the host computer	16 bytes (string)
Machine	Machine name	16 bytes (string)
OrderNum	Order number	4 bytes (long int)
VarMode	Variable mode	4 bytes (long int)
VarSet	Name of the variable set	16 bytes (string)
VarDescr *	Data description	1024 bytes (string)
VarData *	User data	10 KB (string)

*The variables in both the data description and the user data must be separated by the 'pipe' symbol ('|' ASCII code 124).
Variable sets which are defined in SCVARSET.INI are reported.

Example:

R_VAR_H ("FLR1", "BAZ3", 0, 0, "Set02", "\0", "33|50");

7.3 Request data

7.3.1 Request variable data from the machine

Function call

T_VAR_M (Host,
Machine,
OrderNum,
VarMode
VarSet
VarDescr)

Direction of transfer: Host **computer → SINUMERIK**

Meaning

The host computer requests RPC SINUMERIK to read data from the PLC or NCK. The data are then returned with R_VAR_H (). VarSet can only specify a variable set defined in SCVARSET.INI. VarDescr is not evaluated.

Data

Table 7-3 Request variable data from the machine

Parameter	Description	Format
Host	Name of the host computer	16 bytes (string)
Machine	Machine name	16 bytes (string)
OrderNum	Order number	4 bytes (long int)
VarMode	Variable mode	4 bytes (long int)
VarSet	Name of the variable set	16 bytes (string)
VarDescr *	Data description	1024 bytes (string)

*The variables in both the data description and the user data must be separated by the 'pipe' symbol ('|' ASCII code 124).

Example

T_VAR_M ("FLR1", "BAZ3", 0, 0, "Set02", "\0", "\0");

7.3.2 Requesting variable data from the host computer

Called function

T_VAR_H (Host,
Machine,
OrderNum,
VarMode
VarSet
VarDescr
VarData)

Direction of transfer: **Host computer ← SINUMERIK**

Meaning

RPC SINUMERIK requests data from the host computer. The data are then returned to RPC SINUMERIK with R_VAR_M ().

Note

This RPC is not currently used by RPC SINUMERIK .

■

For notes

8

8 Communication between the Host Computer and the Transport System (TPS)

8.1 TPS/machine interface

The transport system (TPS) receives transport jobs from the host computer and must check that these are executable (logically and physically). To do this, it needs to verify whether the destination is available and not disabled.

Docking positions

Points of transfer between the TPS and machine tools are known as docking positions.
There are three types of docking position:
Input docking positions, where it is only possible to deliver
Output docking positions, where it is only possible to retrieve
Input/output docking positions, where delivery and retrieval are both possible

8.2 TPS status data

Called function

R_TPS_H (Host,
Machine,
OrderNum,
Mode,
MachineStatus,
TpoStatus,
DockPos,
DockPosStatus,
WPC,
Resint1,
Resint2,
Resbyte)

Direction of transfer: **Host computer ← SINUMERIK**

Meaning

Transmit TPS status data to the host computer.

Data

Table 8-1 Parameters of transport system status data

Parameter	Description	Format
Host	Name of the host computer	16 bytes (string)
Machine	TPS name	16 bytes (string)
OrderNum	Order number	4 bytes (long int)
Mode	Mode SINUMERIK mode: 1: Automatic 2: MDA 4: JOG Computer communication mode: 100: Host computer mode 300: Manual mode 1000: If bit 6=1 in RPC SINUMERIK mode 2000: If bit 7=1 in RPC SINUMERIK mode	4 bytes (long int)
MachineStatus	Machine status 0: Cold restart 1: Inactive 2: Active 3: faulty 4: End of work (drives deactivated)	4 bytes (long int)
TpOStatus	Transport job status 0: No transport job data 1: New transport job (not started) 2: In progress 4: WPC is on the vehicle 8: Job complete 16: Error, job not executable 32: Error, alternative destination approached (DockPos)	4 bytes (long int)
DockPos[2]	Docking position no. The docking position no. corresponds to the index in the docking position list of the interface DB beginning at the 1st docking position no. = 0 means: "not assigned".	4 bytes (long int)
DockPosStatus[2]	Docking position status 0: Enabled 1: Disabled for transport control system 2: faulty	4 bytes (long int)
WPC[2]	Workpiece carrier name	6 bytes (string)*
Resint1 **	Reserve 1	4 bytes (long int)
Resint2 **	Reserve 2	4 bytes (long int)
Resbyte	Reserve 3	8 bytes (string)

* WPC is defined as a 2-dimensional character array (char [2][6]), however each of the 2 workpiece carrier names must be specified as a string terminated by '\0'.

** Resint1 and Resint2 appear at the DB interface of the PLC. If values are entered there by the PLC they are transferred to the host computer. These values have no impact on the computer link. They are merely passed on to the host computer.

Notes on use

The TPS sets this RPC on every status change, mode selection, and at the beginning and end of transport operations.
When RPC SINUMERIK is started on the SINUMERIK, this RPC is sent as a startup message to each configured host computer.

8.3 Request TPS status data

Function call

T_TPS_M (Host,
Machine,
OrderNum)

Direction of transfer: **Host computer → SINUMERIK**

Meaning

Request transport system status data from SINUMERIK.

Data

Table 8-2 Parameters of the TPS status data request

Parameter	Description	Format
Host	Name of the host computer	16 bytes (string)
Machine	TPS name	16 bytes (string)
OrderNum	Order number	4 bytes (long int)

Notes on use

The host computer can use this call to request TPS status data, e.g. during synchronization. The TPS then returns the data with **R_TPS_H()**.

Example

T_TPS_M ("FLR1", "BAZ3", 0);

8.4 Transport job

Function call

C_TPORDER_M (Host,
Machine,
OrderNum,
SDockPos,
DDockPos,
WPC,
WPCType,
BufferFlag,
Priority,
ChainNum,
Vehicle,
Resint1,
Resint2,
Resbyte)

Direction of transfer: **Host computer → SINUMERIK**

Meaning

Send transport job to the machine.

Data

Table 8-3 Parameters of the transport job

Parameter	Description	Format
Host	Name of the host computer	16 bytes (string)
Machine	Machine name	16 bytes (string)
OrderNum	Order number	4 bytes (long int)
SDockPos	Source docking position number The docking position numbers correspond to the index in the docking position list of the interface block beginning with 1.	4 bytes (long int)
DDockPos	Destination docking position number The docking position numbers correspond to the index in the docking position list of the interface block beginning with 1.	4 bytes (long int)
WPC	Workpiece carrier name	6 bytes (string)
WPCTyp	Workpiece carrier type	4 bytes (long int)
BufferFlag	Destination only used for buffering	4 bytes (long int)
Priority	Transport priority	4 bytes (long int)
ChainNum	Job chain number	4 bytes (long int)
Vehicle	Transport vehicle number	4 bytes (long int)
Resint1	Reserve 1	4 bytes (long int)
Resint2	Reserve 2	4 bytes (long int)
Resbyte	Reserve 3	8 bytes (string)

Example

C_TPORDER_M ("FLR1", "BAZ3", 0, 3, 4, "WPC05", 0, 0, 0, 1, 0, 0, "\0");

Source and destination docking position

The source and destination docking positions include the index for the corresponding docking position data record in the docking position list of the interface DB; the index begins with 1.

Workpiece carrier

The name of the workpiece carrier can be used by the transport system for plausibility checks.
The workpiece carrier type is an additional information for the transport system which can contain the type or the size of a workpiece carrier.
With the parameter BufferFlag = 1, the host computer can inform the transport system if a workpiece carrier shall be transported to a machine only for buffering but shall not be machined there (auxiliary buffer location). If necessary, the transport control system (TPS) must pass this information to the machine.
For all other transports, BufferFlag = 0 must be set.

Transport priority

The transport priority is an additional information, if several jobs may be transferred to the transport system. The priority can be used to control the order in which they are processed.

Job chaining

Job chaining is an additional information. In transport vehicles with two storage locations and machines with only one docking position, two transport jobs can be associated logically by means of a chain number.

For example, the first transport job includes the delivery of an unworked workpiece carrier to a machine. A finished workpiece carrier is still placed on the machine, the second transport job with the same job chaining number as the first one includes the collection of the worked workpiece carrier from the machine to a buffer or clamping location. In this case, the new pallet is fetched and moved to the machine where the worked workpiece carrier is stored on the free location of the carriage. Only then is it possible to complete the first job by restoring the unworked workpiece carrier on the machine. Afterwards, it is possible to move to the destination of transport two and to complete also this job.

Transport carriage

The transport carriage is a supplementary unit of information. In transport systems with several transport vehicles, this parameter can be set by the host computer to define which vehicle is to be used for transport.

8.4.1 Sequence of a transport job

Direction of transfer	Command	Comments
Host computer → SINUMERIK	C_TPORDER_M ()	
Host computer ← SINUMERIK	R_TPS_H () (optional)	Reports TPS active, empty storage location, and WPC on vehicle
Host computer ← SINUMERIK	R_TPS_H ()	Reports TPS inactive, new storage location on WPC and empty vehicle

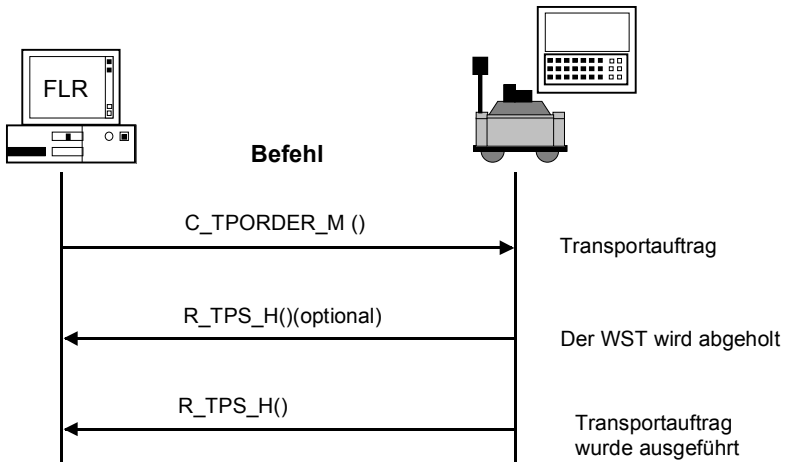


Fig. 8-1 Transport dialog, normal situation, no errors

8.4.2 Errors in transport jobs

In the event of an error, the TPS outputs an error message to the host computer; possible errors which can occur during transport jobs are:
Error numbers used in RPC SINUMERIK:

Table 8-4 Error numbers in transport system

No.	Meaning
-200	The transport job cannot be written in the interface DB
-700	A transport job error is reported by the transport system.

8.5 Synchronization of the transport system (TPS)

The procedure for synchronization of the TPS is the same as for the machine; the host computer requests status and assignment data from the TPS.
During synchronization, the transport system must not report any status changes.

Direction of transfer	Command
Host computer → SINUMERIK	C_SYNCH_M () SynchFlag = Start
Host computer → SINUMERIK	T_TPS_M ()

If a transport job is operational during synchronization, the end of this job may be reported only after synchronization.

Direction of transfer	Command	Comments
Host computer ← SINUMERIK	R_TPS_H ()	
Host computer → SINUMERIK	T_DATA_M () SFct = 50	
Host computer ← SINUMERIK		Transfer machine assignment file to the host computer
Host computer ← SINUMERIK	R_DATA_H () SFct = 50, Name 1 = file name	
Host computer → SINUMERIK	C_SYNCH_M () SynchFlag = End	

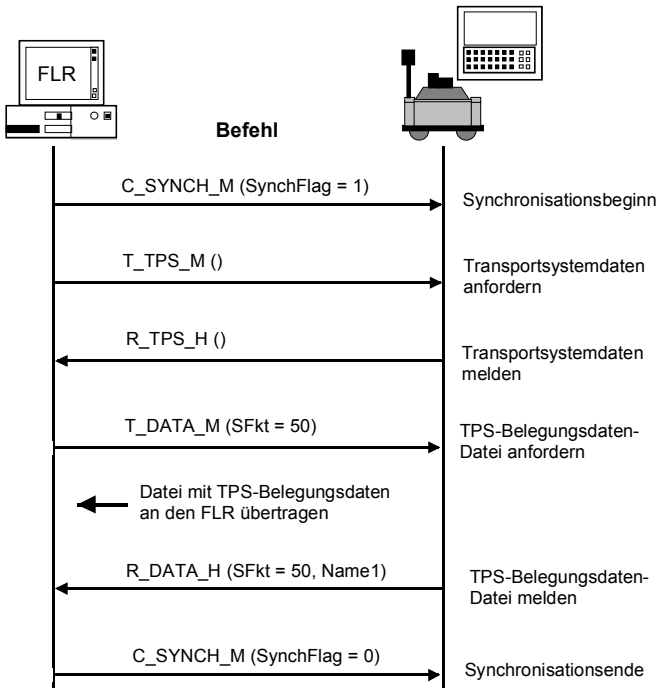


Fig. 8-2 TPS synchronization, initiated by host computer

■

For notes

9

9 Summary of RPC Calls

9.1 Function calls from the host computer to SINUMERIK

Table 9-1 Function calls from host computer to SINUMERIK

Start	Description
T_MACHINE_M ()	Transmit initiation from "server" to client
T_TPS_M ()	Transmit initiation from "server" to client
T_REPORT_M ()	Requesting pending alarms
R_NC4WPC_M ()	Enter in list, initiation to client
R_REPORT_M ()	Transfer message to the machine
C_DELETE_M ()	Delete data in the data management system
C_MODE_M ()	Set bits in the interface
C_SYNCH_M ()	Set/reset bits in the interface
C_TPORDER_M ()	Enter in list, initiation to client
C_ORDER_M ()	General order function
T_DATA_M ()	
R_DATA_M ()	
T_VAR_M ()	
R_VAR_M ()	
R_DDEDATA_M ()	
R_MESSAGE_M ()	

9.2 Function calls from SINUMERIK to the host computer

Table 9-2 Function calls from SINUMERIK to host computer

Start	Description
R_MACHINE_H ()	Offer machine status data to the host computer.
R_TPS_H ()	Offer transport system status data to the host computer
R_REPORT_H ()	Send messages to host computer
T_DATA_H ()	
R_DATA_H ()	
T_VAR_H ()	Option
R_VAR_H ()	
R_DDEDATA_H ()	
R_MESSAGE_H ()	

Note

Not all subfunction numbers (SFct) are possible in both directions.

Table 9-3 Subfunction number SFct for data dialogs

Subfnc. no.	Function	Command	Comments
1	NC program	T_DATA_H () T_DATA_M () R_DATA_H () R_DATA_M ()	Name1 = NC program Name2 = File name with path on host computer
10	List of existing NC programs	T_DATA_M () R_DATA_H ()	
20	Tool status data of all tools Variant 1, complete set of tool data	T_DATA_M () R_DATA_H ()	Name1 = Empty Name2 = File name with path
21	Tool status data of one tool Variant 2 reduced set of tool data	T_DATA_M () R_DATA_H ()	Name1 = ID number, Duplo number Name2 = File name with path
22	Tool status data of one tool Variant 3 reduced set of tool data	T_DATA_M () R_DATA_H ()	Name1 = ID number, Duplo number Name2 = File name with path
23	Tool status data of all tools Variant 3 reduced set of tool data	T_DATA_M () R_DATA_H ()	Name1 = ID number, Duplo number Name2 = File name with path
24	Tool data of a tool with adapter number Complete set of tool data	T_DATA_H R_DATA_M	
26	Optional/manual loading of a tool Complete set of tool data, tool data 1	T_DATA_H () R_DATA_M ()	
27	Optional/manual unloading of a tool	R_DATA_H	
28	Load tool from tool pallet	T_DATA_H () R_DATA_M ()	
29	Unload tool to tool pallet	T_DATA_H () R_DATA_M ()	
50	Machine assignment data	T_DATA_M () R_DATA_H ()	
90	Any file	T_DATA_M () R_DATA_H ()	

Table 9-4 Subfunction number SFct for the general order function

Subfnc. no. (SFct)	Function	Comments	
2	Loading of NC programs	Name1 = Name2 = Parameter1=	NC program name NCK name (optional) Function (0,2,3)
3	Loading and selection of NC programs	Name1 = Name2 = Parameter1= Parameter2=	NC program name NCK name (optional) Function (0-3) Channel number
4	Unloading of NC programs	Name1 = Name2 =	NC program NCK name
5	Selection of NC programs	Name1 = Name2 = Parameter1= Parameter2=	NC program name NCK name (optional) Function (0,1) Channel number
6	Selection of NC programs	Name2 = Parameter2=	NCK name (optional) Channel number
100	Setting date/time on the SINUMERIK/PLC	Name2 = Parameter1 Parameter2=	NCK name (optional) UNIX time Date/time SINUMERIK / PLC

■

10

10 RPC SINUMERIK-OCX

10.1 Introduction

What is RPC SINUMERIK-OCX?

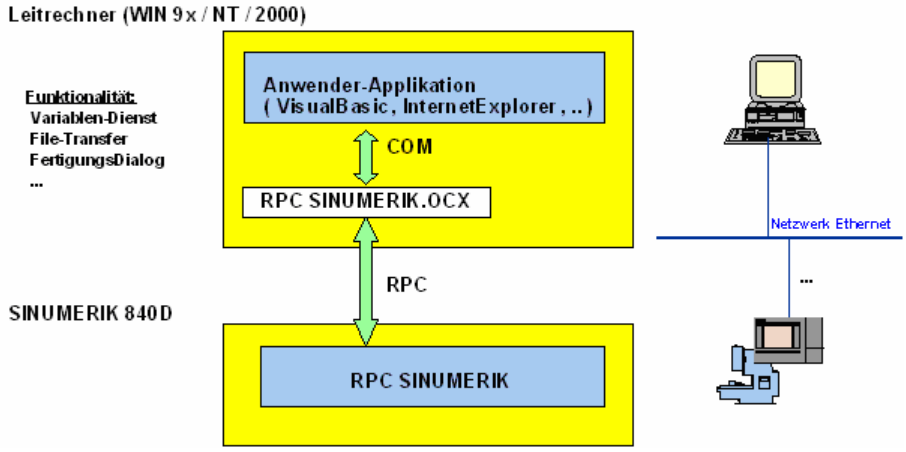
The RPC SINUMERIK-OCX development package is an add-on product for the RPC SINUMERIK computer link.
The RPC SINUMERIK computer link provides an interface for communication between a SINUMERIK 840D machine control and a higher-level host computer. Communication between the host computer and the machine control is performed in this arrangement via RPCs (Remote Procedure Call). Because the RPCs are a platform-independent standard, the RPC SINUMERIK interface can be used both by MS Windows and by UNIX, LINUX and other systems.

Application range

However, use of the RPCs is usually only possible from the C/C++ programming language.
Using the RPC SINUMERIK-OCX, the RPC SINUMERIK interface is accessible to a whole range of common Windows development systems without the need to program in C/C++. All development systems are supported that are capable of linking 32-bit ActiveX components. This includes: MS Visual Basic V4.0 or later (32Bit), MS Visual J++ 6.0, Internet Explorer V4.0 or later, WinDev and many other development systems.

Method of operation

The RPC SINUMERIK-OCX encapsulates the RPC interface from RPC SINUMERIK in COM calls (Component Object Model). The mode of operation of the individual RPCs is not affected by the use of RPC SINUMERIK-OCX.
The mode of operation of the RPCs is described in the RPC SINUMERIK documentation.



Operating systems

The RPC SINUMERIK-OCX can be used on WIN 9x/NT/2000/XP/2003 computers with a TCP/IP network installed. One or more SINUMERIK 840D controls with RPC SINUMERIK are required as communication partners.

Languages

RPC SINUMERIK generally supports all languages which are covered by the ASCII character set or for which a language package has been installed on the SINUMERIK. Languages which use an extended character set (e.g. Chinese, Russian, etc.) are processed by SINUMERIK in DBCS format. In this case, data are displayed correctly on the PCU if the application on the host computer transmits the data in DBCS format.

10.2 Installation of the RPC SINUMERIK-OCX development package

Installation of the RPC SINUMERIK-OCX development package is performed by Setup.exe on the first installation disk.

During installation, you are asked to enter the target directory and the name of the program folder in the start menu.

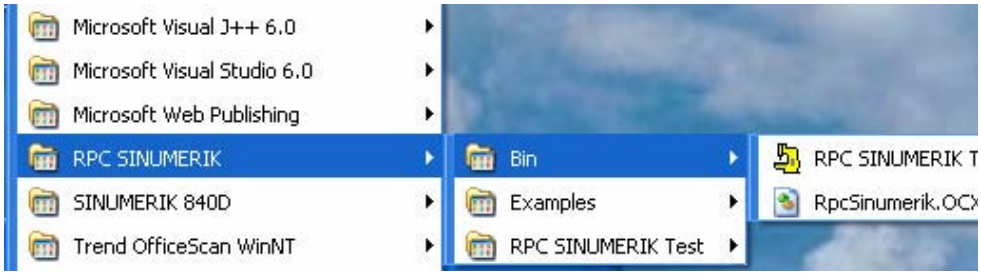
The following subdirectories are created in the target directory:

Table 10-1 Subdirectories in the target directory

Doc	Directory with this documentation
Bin	Directory with the RpcSinumerik.OCX and the application RpcSinumerikTest.EXE
RPC SINUMERIK Test	Directory with the source code for the RPC SINUMERIK Test application
Examples	Directory with examples of the use of the RPC SINUMERIK-OCX

In addition, the Microsoft Visual Basic 6.0 (SP3) runtime system is installed in the <Windows>\System directory, if it is not already present.

This is how to reach the installed files via the start menu:



Test and examples

After installation, you can test the RPC SINUMERIK interface using RPC SINUMERIK-OCX with the RPC SINUMERIK Test application. For the requirements that must be fulfilled before using the RPC SINUMERIK Test application, see section: "10.4 RPC SINUMERIK Test application". After successfully testing the link with RPC SINUMERIK, you can use the examples described in Section: "10.5 Examples of the use of RPC SINUMERIK-OCX".

10.3 Description of the RPC SINUMERIK-OCX component

The RPC SINUMERIK-OCX is implemented as a 32-bit ActiveX/COM component. It can be used in all 32-bit Windows development systems that permit integration of such components.

A separate instance of the RPC SINUMERIK-OCX component must be instantiated in the user application for each link to the machine control.

10.3.1 Installation

With installation of the RPC SINUMERIK-OCX development package, file RpcSinumerik.OCX is stored in the <Windows>\System directory and registered in the Windows© registry.

Further computers

If you wish to use the RPC SINUMERIK-OCX component on other computers too, it must be installed on them as well. This can be done with the installation programs (such as: InstallShield) or by the following manual steps.

1. Copy file RpcSinumerik.OCX from directory C:\Program Files\Siemens\MCIS\RPC SINUMERIK\bin into the <Windows>\System directory of the computer.
2. Register the RPC SINUMERIK-OCX component in the Windows registry with the following command:
   Regsvr32 <Windows>\System\RpcSinumerik.OCX

A TCP/IP link with the RPC SINUMERIK control must also exist.

10.3.2 Attributes of the RPC SINUMERIK-OCX component

The RPC SINUMERIK-OCX component has the following attributes that are used to configure the link.

Table 10-2 Attributes of the RPC SINUMERIK-OCX component

Attributes	Meaning	Example
MachineID	Name of the machine in the RPC SINUMERIK configuration. Freely selectable name for the machine. This information is passed to the communication partner in each RPC for identification purposes.	M1
MachineIP	IP network address of the machine control. This information can be obtained from the Windows network installation. Static IP assignment is assumed. At this point, the network name of the computer can be used in the RPC SINUMERIK-OCX instead.	195.2.208.233
MachinePort	Additional information for addressing an application within a computer in TCP/IP communication. You can freely select the port number in the range 1000 to 64000. We recommend using port number 3011 for the machine control. In the RPC SINUMERIK configuration this information is termed the machine endpoint.	3011
MachineTimeout	This value is used to influence the time response when sending RPCs to RPC SINUMERIK. If an RPC to RPC SINUMERIK cannot be delivered, for example, because the control is not activated, the time before the RPC call is aborted with an error is defined by the timeout value. The attribute should take a value between 0 and 9. These values are not time values but relative values, which are defined in the Microsoft RPC system. 0 - Min TimeOut 5 - Default TimeOut 9 - Max. TimeOut We recommend using the default timeout value of (5).	5
HostID	Name of the host computer in the RPC SINUMERIK configuration. Freely selectable name of the host computer. This information is passed to the communication partner in each RPC for identification purposes.	H1
HostPort	Additional information for addressing an application within a computer in TCP/IP communication. You can freely select the port number in the range 1000 to 64000. We recommend using port number 3010 for the host computer. In the RPC SINUMERIK configuration this information is termed the host endpoint.	3010

Attributes	Meaning	Example
HostEnabled	This attribute indicates readiness of the component to receive RPCs.	True

The attributes HostID, HostPort, and HostEnabled apply to all instances of the RPC SINUMERIK-OCX within an application (EXE). Any change made in one instance is also made in all instances.

10.3.3 Methods of sending RPCs to RPC SINUMERIK

The RPC SINUMERIK-OCX encapsulates the RPC interface from RPC SINUMERIK in COM calls. An RPC to RPC SINUMERIK is triggered by calling a method with the same name as the RPC SINUMERIK-OCX component.

For example the RPC

Ret = T_MACHINE_M (Host, Machine, OrderNum)

is formed by the method T_MACHINE_M on one of the instances of the OCX.

Ret = Machine1.T_MACHINE_M (OrderNum)

In all these methods, the first two parameters (host and machine) are not required. These parameters are derived from the attributes: HostID and MachineID of the instance in question.

Return values that are supplied by methods are described in section: "10.3.6 Error handling".

Note

Empty strings must be represented in different ways for different programming languages.

- Basic: ""
- C: "\0"

10.3.4 Activate readiness to receive

The RPC SINUMERIK-OCX is ready to receive RPCs from the RPC SINUMERIK if at least one RPC has been successfully sent to the machine or the attribute **HostEnabled** is set to **True**.

10.3.5 Receive RPCs from RPC SINUMERIK

The RPCs from the RPC SINUMERIK are passed to the application as events of the instance of the RPC SINUMERIK-OCX component.

For example, the RPC from RPC SINUMERIK

T_DATA_H (Host, Machine, OrderNum, SFct, Name1, Name2)

is passed to the application as event

TxDATAxH (OrderNum, SFct, Name1, Name2)

Because some development systems (e.g.: Visual Basic) do not permit the "_" character in the names of events, the "_" characters in the name of the RPC have been replaced by "**x**".

10.3.6 Error handling

The error numbers that are supplied by the methods of the RPC SINUMERIK-OCX can be subdivided into two categories:

- Error messages from the Microsoft RPC system
- Error messages from RPC SINUMERIK

Error messages from the Microsoft RPC system

These errors occur if an RPC cannot be sent to RPC SINUMERIK or the RPC server cannot be activated by the RPC SINUMERIK-OCX. The error numbers in this category are in the range 1700 to 1938 (RPC status codes).

You will find a description of the errors from this group occurring in practice in the table below: "Typical error situations". The complete description is to be found in the Microsoft Documentation.

http://msdn.microsoft.com/library/

under

Platform SDK -> Networking and Directory services -> Remote Procedure Calls (RPC)

Error messages from RPC SINUMERIK

These errors occur if an RPC has been correctly transferred to RPC SINUMERIK but correct processing was not possible because of the content of the RPC or the current status of the RPC SINUMERIK. The errors of this category are in the range of negative numbers. ". For a complete description, see the RPC SINUMERIK documentation "Appendix A - Error numbers".
Here, the following error constellations are possible:

Typical error situations

At this point, a few frequent error situations are described:

Table 10-3 Typical error situations

Error situation	Return value for an RPC method
Wrong value for the attribute: MachineIP	1722 (RPC_S_SERVER_UNAVAILABLE)
Wrong value for the attribute: MachinePort	1722 (RPC_S_SERVER_UNAVAILABLE)
Wrong value for the attribute: MachineID	-100 (ERR_WRONG_MACHINE)
Wrong value for the attribute: HostID	-110 (ERR_WRONG_HOST)
Wrong value for the attribute: HostPort	0, but no response from RPC SINUMERIK At the same time, an entry is generated in the MCIS_RPCERR.LOG in the control
Readiness of the RPC SINUMERIK-OCX component to receive cannot be activated.	1720 (RPC_S_CANT_CREATE_ENDPOINT) This error usually occurs if another application on the same computer is already using the port defined with attribute HostPort. e.g.: an attempt is made to start an application that is using the RPC SINUMERIK-OCX and, at the same time, the RPC SINUMERIK Test application is being executed – with the same HostPort value.
Interruption of the TCP/IP link	1722 or 1726 (after approx. 20 sec.) After you have restored the link, the RPC calls return 0 again.

10.3.7 Restrictions in connection with the test

Within the development environment of VisualBasic, events are not triggered by the OCX if VisualBasic is interrupted by a break point.

10.4 Test application RPC SINUMERIK Test

The RPC SINUMERIK Test application allows the individual RPCs in the RPC SINUMERIK interface to be sent or received in an interactive form. It is possible to establish a link with several controllers.

The application was written in Visual Basic 6.0. The source code for the application is provided in the directory Siemens\MCIS\RPC SINUMERIK\ RPC SINUMERIK Test (see also Section: 10.4.4 Source code for the RPC SINUMERIK Test application)

Before you can use RPC SINUMERIK Test, the following conditions must be fulfilled:

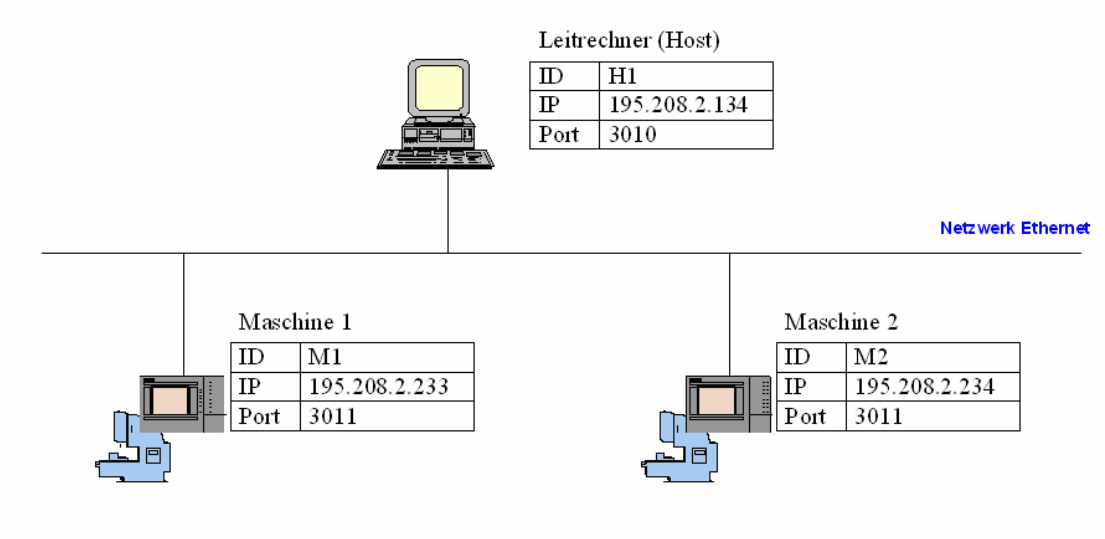
1. At least one SINUMERIK 840D controller must be available with a RPC SINUMERIK package installed.
2. The TCP/IP protocol must be installed on the host computer (Windows PC).
3. There must be a network link between the controllers and the host computer.

Languages

RPC SINUMERIK generally supports all languages which are covered by the ASCII character set or for which a language package has been installed on the SINUMERIK. Languages which use an extended character set (e.g. Chinese, Russian, etc.) are processed by SINUMERIK in DBCS format. In this case, data are displayed correctly on the PCU if the application on the host computer transmits the data in DBCS format.

10.4.1 Configuration

Here is an example of the configuration in the following network architecture

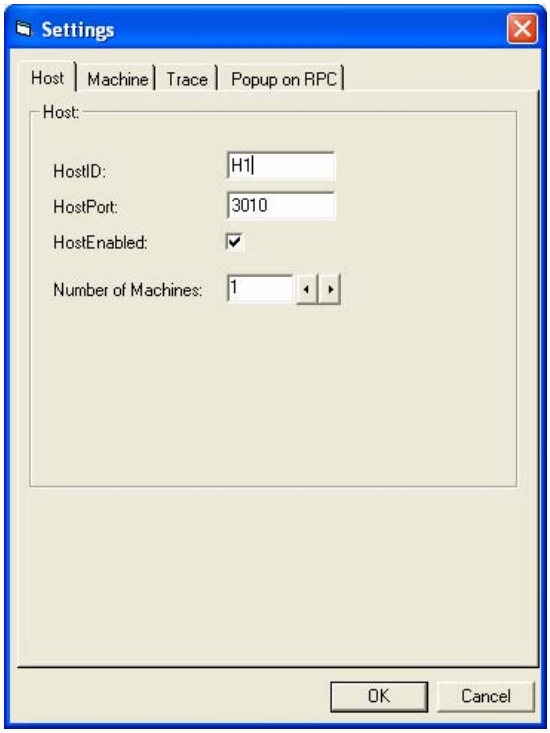


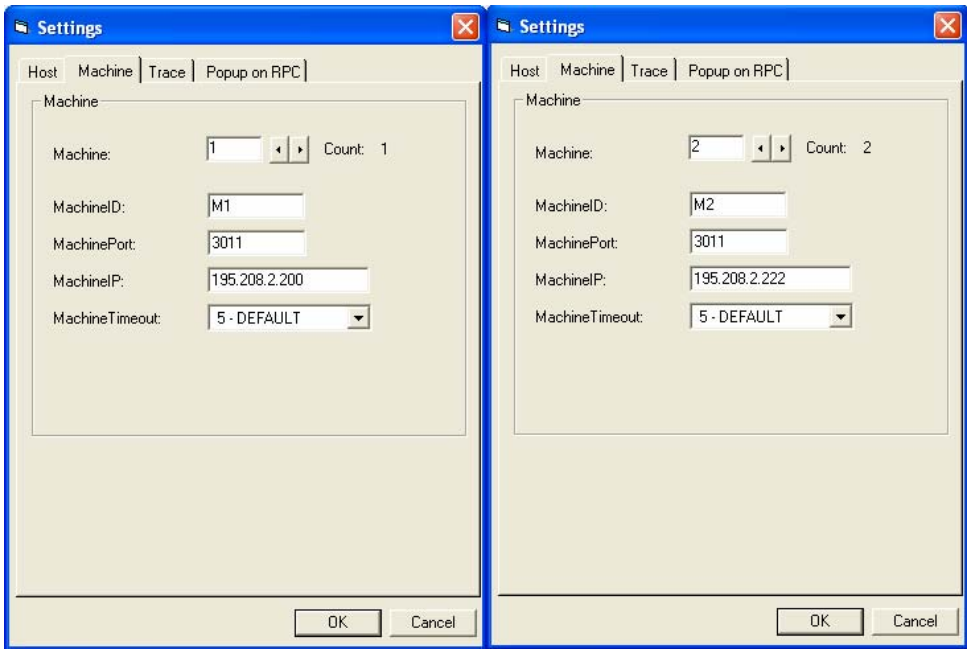
For each computer involved, the following information is required:

Table 10-4 Information required to configure the computers

ID	Freely selectable name for the communication partner. This information is passed to the communication partner in each RPC for identification purposes.
IP	IP network address of the host computer or controller. This information can be obtained from the Windows network installation. Static IP assignment is assumed. At this point, the network name of the computer can be used in the RPC SINUMERIK-OCX instead.
Port	Additional information for addressing an application within a computer in TCP/IP communication. You can freely select the port number in the range 1000 to 64000. We recommend using port number 3010 for the host computer and 3011 for the machine controls. In the RPC SINUMERIK configuration this information is termed the endpoint.

The following entries result from the network architecture shown in the configuration screen forms for RPC SINUMERIK Test. The configuration screen forms can be opened from the **Settings** menu.

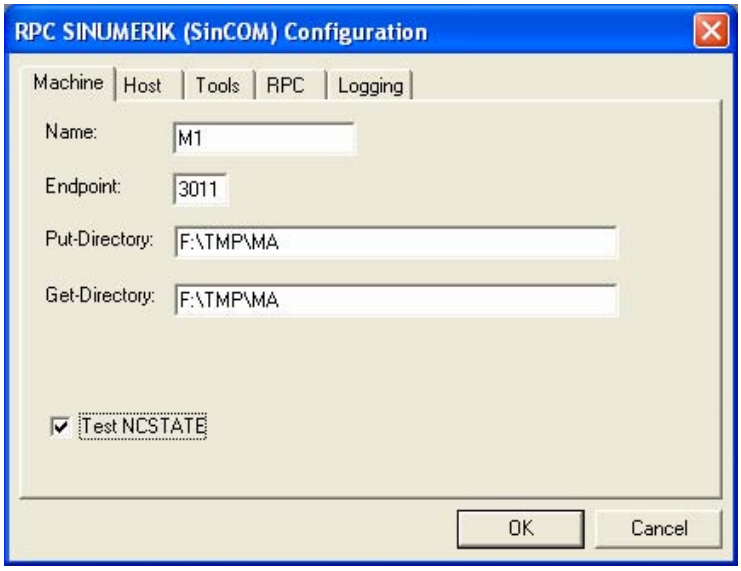


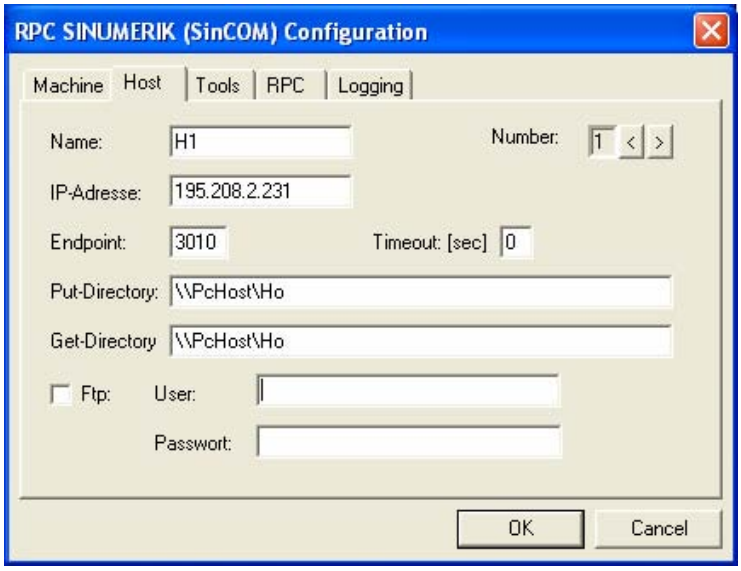


The meaning of the input fields HostEnabled and MachineTimeout corresponds to the attributes of the RPC SINUMERIK-OCX with the same names.

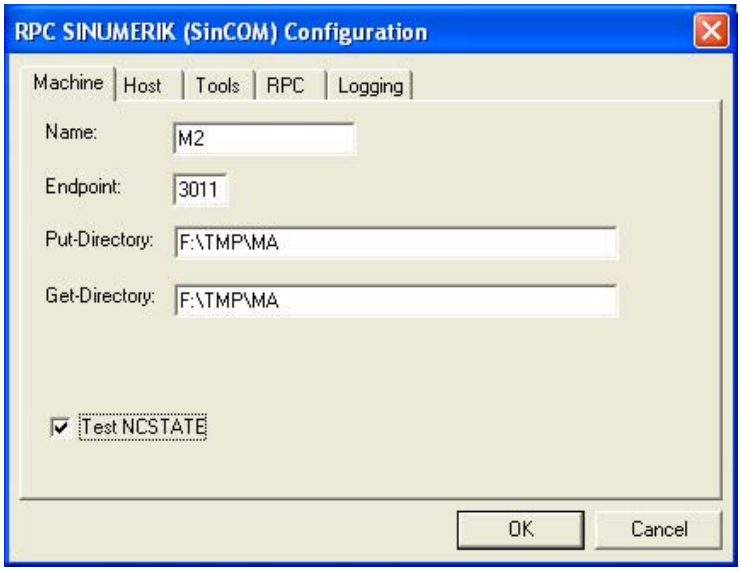
The link with the controls is established with the application **scconfig**.
For the network architecture shown, the following entries must be made.
For a more detailed description of scconfig, please refer to the sections above.

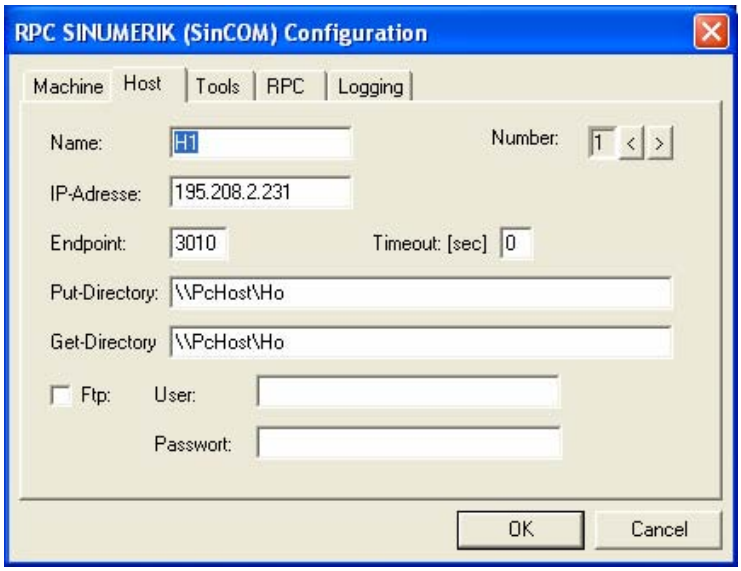
Machine 1



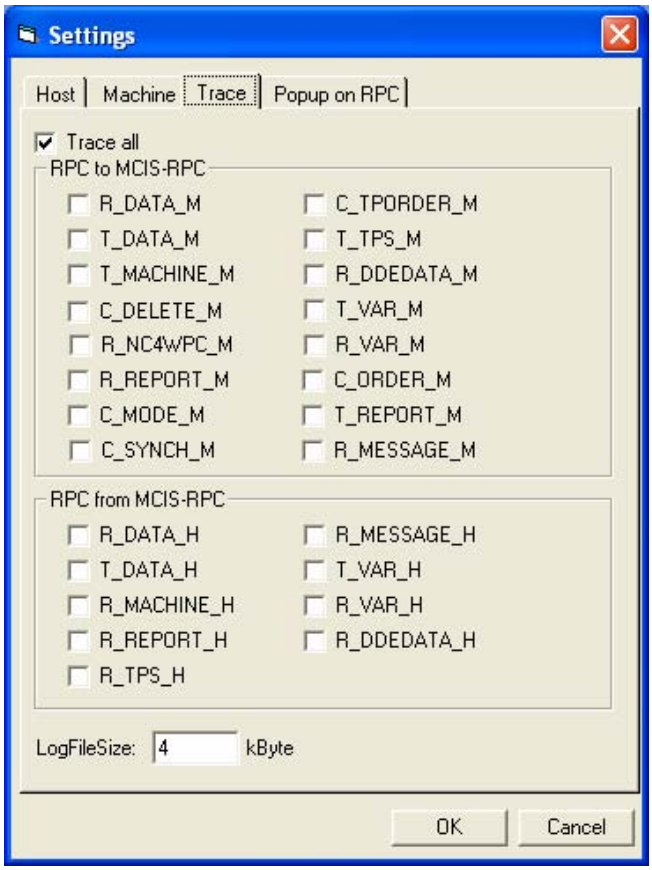


Machine 2

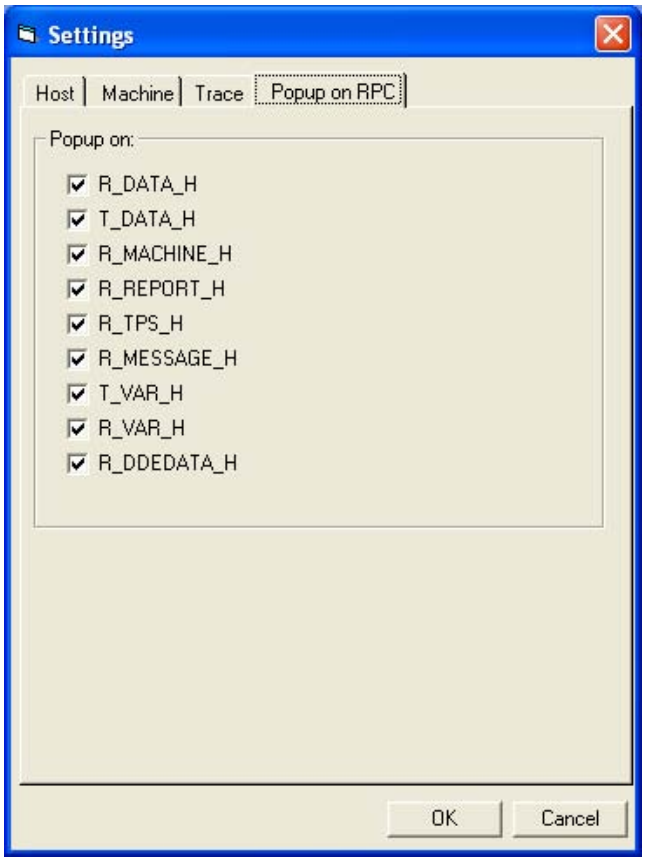




On tab card "**Trace**“ you can select which RPCs are to be traced. The traces are output on the screen and logged in logfile RpcSinumerikTest.LOG, whose maximum size you can define here.

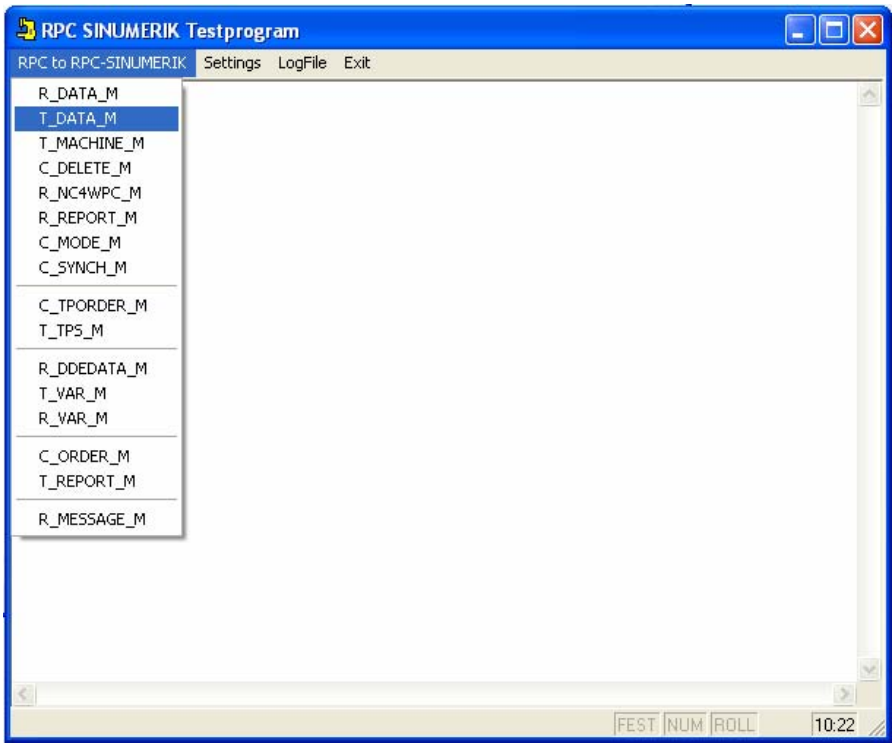


It is possible to define the response to received RPCs on tab card **"Popup on RPC"**. The default response to a received RPC is to display an interactive form with all the data of the RPC.

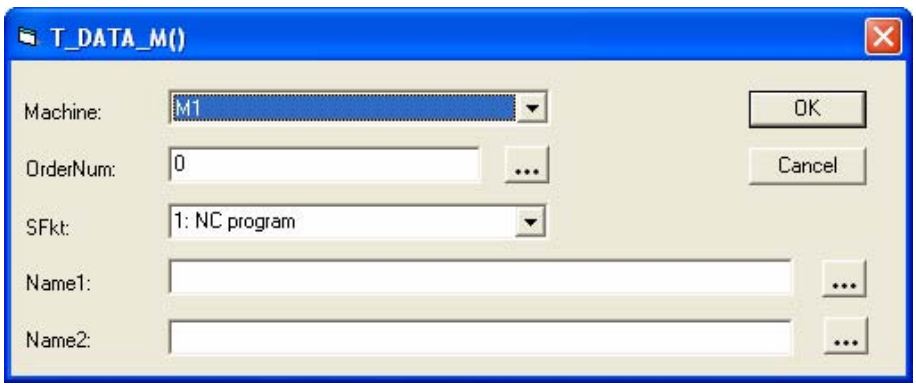


10.4.2 Send RPCs to RPC SINUMERIK

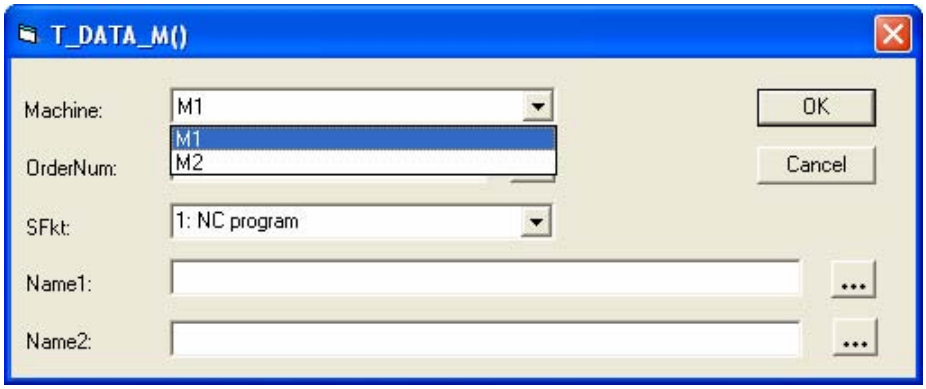
The RPC SINUMERIK Test application allows all defined RPCs to be sent to RPC SINUMERIK via an interactive form. You can call up the interactive forms with menu: **"RPC to RPC SINUMERIK"**.



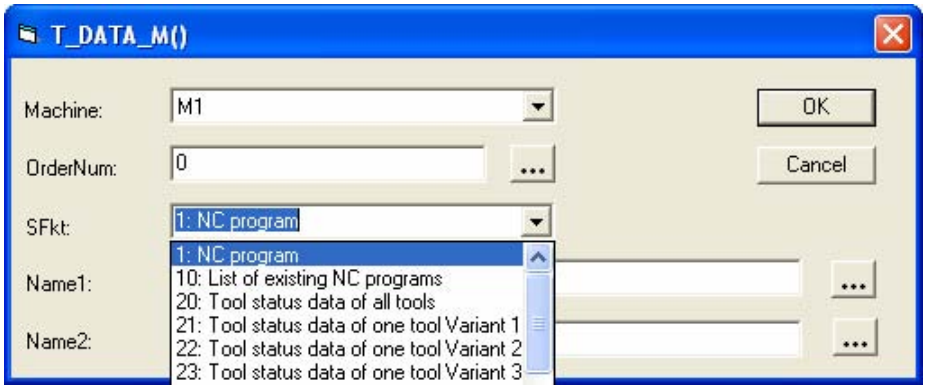
Here is an example of the form for sending the RPC: **T_DATA_M**.



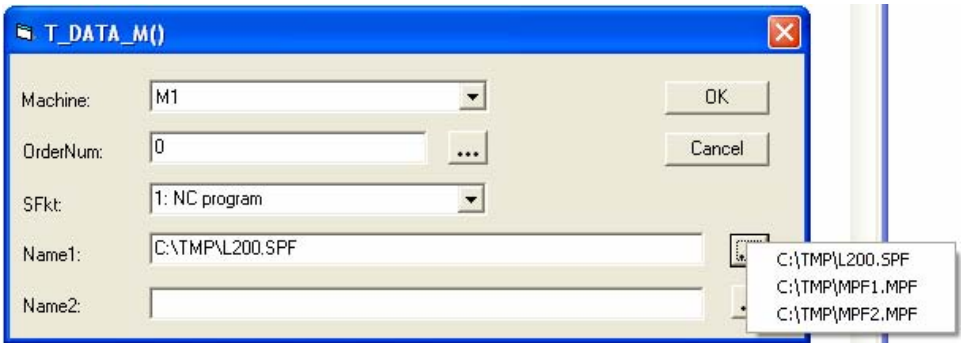
The input fields have the same names as the corresponding RPC parameters in the RPC SINUMERIK documentation. The meanings of these parameters are given in this documentation. With parameter *Machine*, you can define to which control the RPC is to be sent. This input field is implemented as a selection field. It suggests the communication partners defined in the configuration.



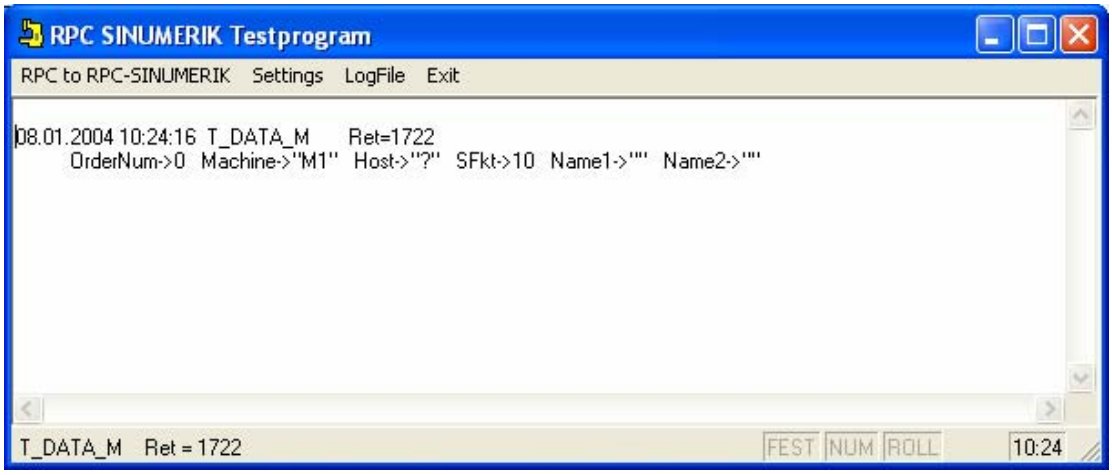
All parameters that can have predefined values in the RPC SINUMERIK documentation (e.g. SFct) are implemented as radio buttons.



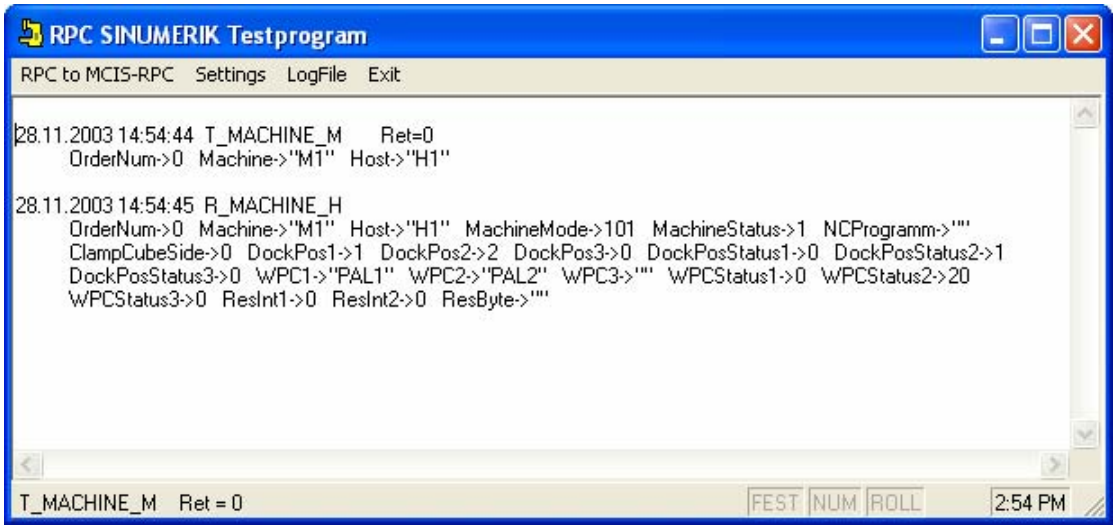
For all other input fields you can select a previously used value with the button "...".



With the "OK" button you can send the RPC. All RPCs sent are traced. The traces are output to the screen in the main window of the RPC SINUMERIK Test application and to the logfile: RpcSinumerikTest.LOG. For every RPC sent, the return value is also logged and displayed in the status line (Ret=0). The meaning of the return value is described in Section: "10.3.6 Error handling".



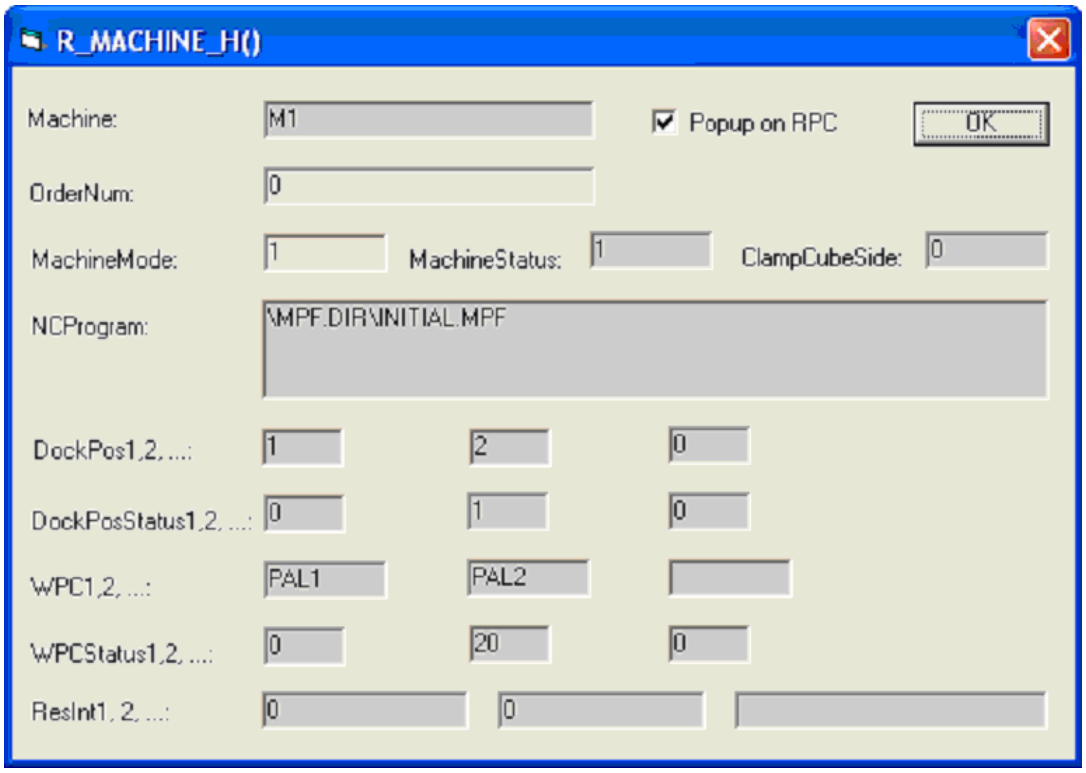
To test the basic link with RPC SINUMERIK, you can send the RPC T_MACHINE_M. RPC SINUMERIK responds with the RPC R_MACHINE_H.



10.4.3 Receive RPCs from RPC SINUMERIK

After the communication partner has been configured by activating the option **HostEnabled**, the RPC SINUMERIK Test application is ready to receive RPCs from RPC SINUMERIK. If the option HostEnabled is not activated in the configuration, RPCs are only received from RPC SINUMERIK when at least one RPC has been successfully transferred to RPC SINUMERIK.

All RPCs received are traced. The traces are output to the screen in the main window of the RPC SINUMERIK Test application and to the logfile: RpcSinumerikTest.LOG. An interactive form containing all parameters of the RPC is also displayed for each RPC.



The output fields have the same names as the corresponding RPC parameters in the RPC SINUMERIK documentation. The meanings of these parameters are given in this documentation.

It is possible to activate or deactivate display of the interactive forms for the RPCs received with the radio button: **"Popup on RPC"** in the relevant form or in the configuration (tab card: Popup on RPC).

10.4.4 Source code for the RPC SINUMERIK Test application

The source code for the application is stored by Setup.exe in the directory Siemens\MCIS\RPC SINUMERIK\RPC SINUMERIK Test.
The application was written in the Microsoft VisualBasic 6.0 development system.

The application consists of the following modules:

Table 10-5 Modules of the RPC SINUMERIK Test application

RpcSinumerikTest.vbp	VisualBasic project file
RpcSinumerikTest.frm **RpcSinumerikTest.frx**	Main window of the application
RpcSinumerikConfig.frm **RpcSinumerikConfig.frx**	Configuration form
History.frm	Selection of input values
Logen.bas	Trace functions
Util.bas	Auxiliary functions
R_DATA_H.frm **...**	Input and display forms for the corresponding RPCs

10.5 Examples of using the RPC SINUMERIK-OCX

In all the examples given in this section, the network configuration used is the one described in Subssection: "10.4.1 Configuration". To adapt the examples to your network configuration, you must alter the IP addresses in the source code of the examples accordingly. It is also assumed that RPC SINUMERIK is installed on the machine controllers and that a network link can be established.

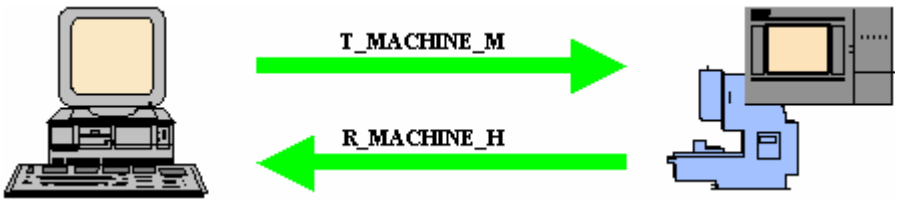
Note

The string delimiter '\0' in the examples of the interface to the host computer (chapter 5) is only necessary in applications created with C++.

10.5.1 Example 1 - querying the machine state (Visual Basic)

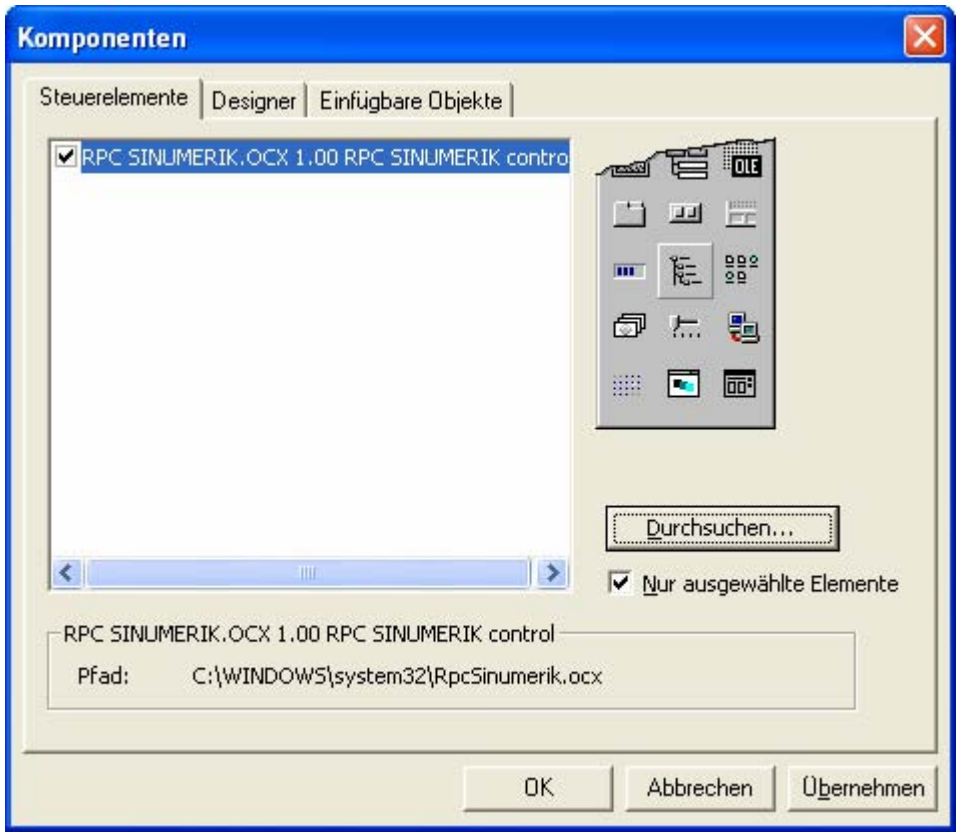
Here is an example of a Visual Basic application that sends the RPC: T_MACHINE_M() to the machine and receives the response to it in the form of RPC: R_MACHINE_H() from the RPC SINUMERIK. The use of the RPCs is described in the RPC SINUMERIK documentation in sections 5.3.1 and 5.3.2.

All the steps are described that are necessary inside the Visual Basic development environment to create the application.



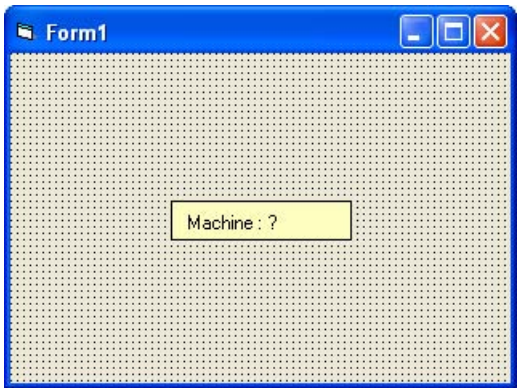
Integrating the RPC SINUMERIK-OCX component in Visual Basic 6.0

Before you can use the RPC SINUMERIK-OCX component in Visual Basic, it must be recognized there. This is done with the menu: Project -> Components.

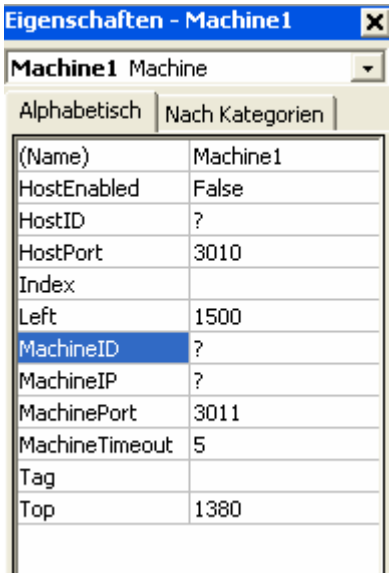


After that, the icon of the RPC SINUMERIK-OCX appears in the toolbox window (yellow machine)

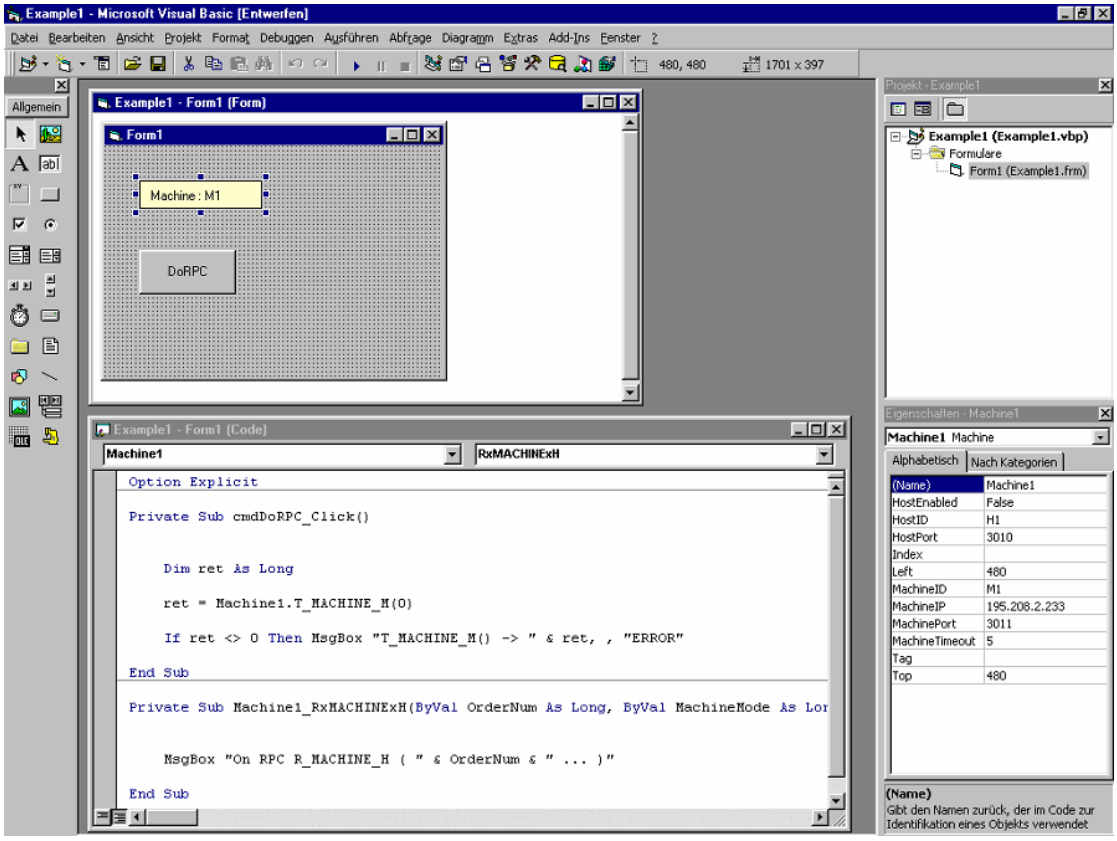
You can now drag and drop the RPC SINUMERIK-OCX into a form.
The component is visible in the development environment. But it is no longer visible during the runtime of a complete application.



In the properties window, you can now store attributes. The IP address of the machine controller must be used in accordance with your network configuration.

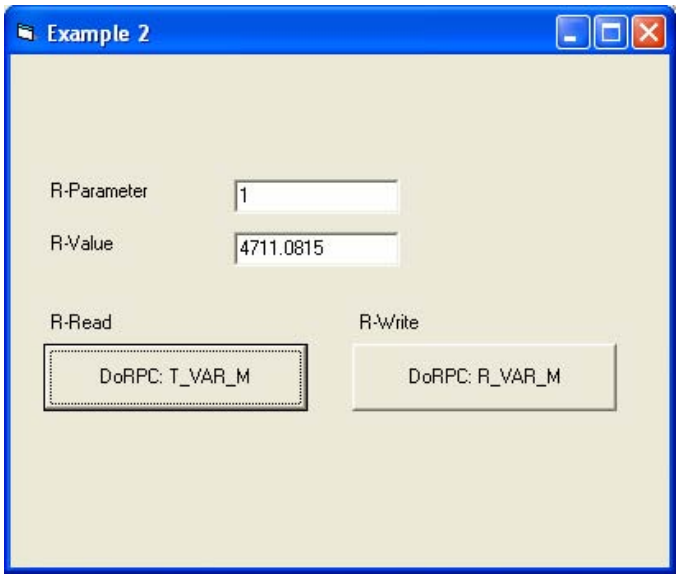


When you change the attribute: **MachineID**, the display of the OCX in the form also changes. Sending of the RPC **T_MACHINE_M** is triggered by clicking on the "DoRPC" button. The RPC SINUMERIK application responds with the RPC **R_MACHINE_H**. The response to this in the example application is a message box.

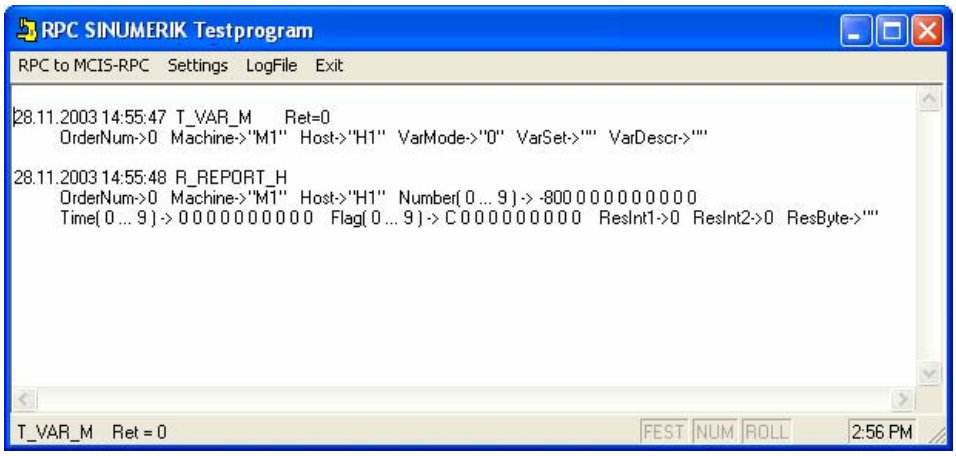


10.5.2 Example 2 - reading and writing R parameters (Visual Basic)

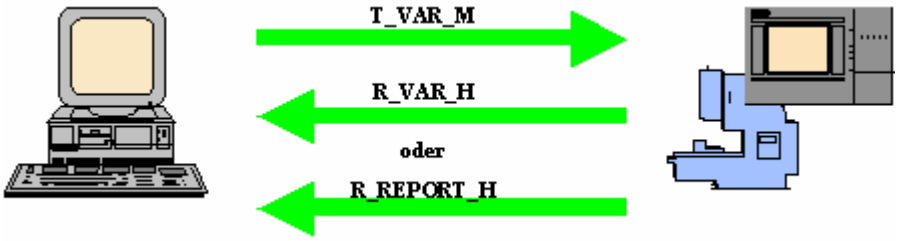
This example explains the use of the RPC SINUMERIK variable service by reference to R parameters. A complete description of the variable service is given in the RPC SINUMERIK documentation in Chapter: "7 Configurable data transmission/variable service".



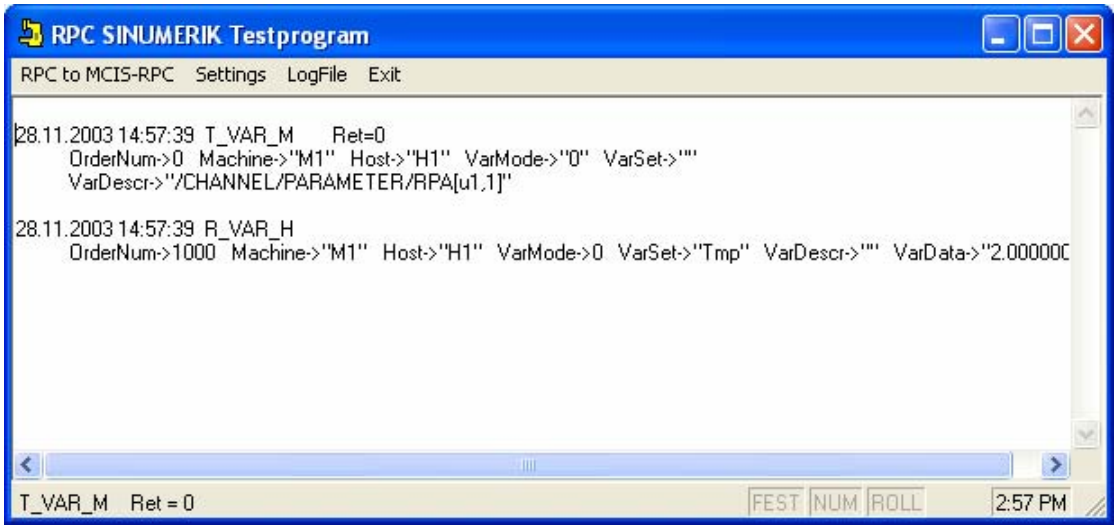
To activate the variable service of RPC SINUMERIK , at least one variable set must be defined in the file c:\add_on\scvarset.ini on the controller. Changes in this file only become effective after the next cold restart of the controller. If this is not the case, RPC SINUMERIK responds with the RPC: R_REPRT_H() and the error: -800.



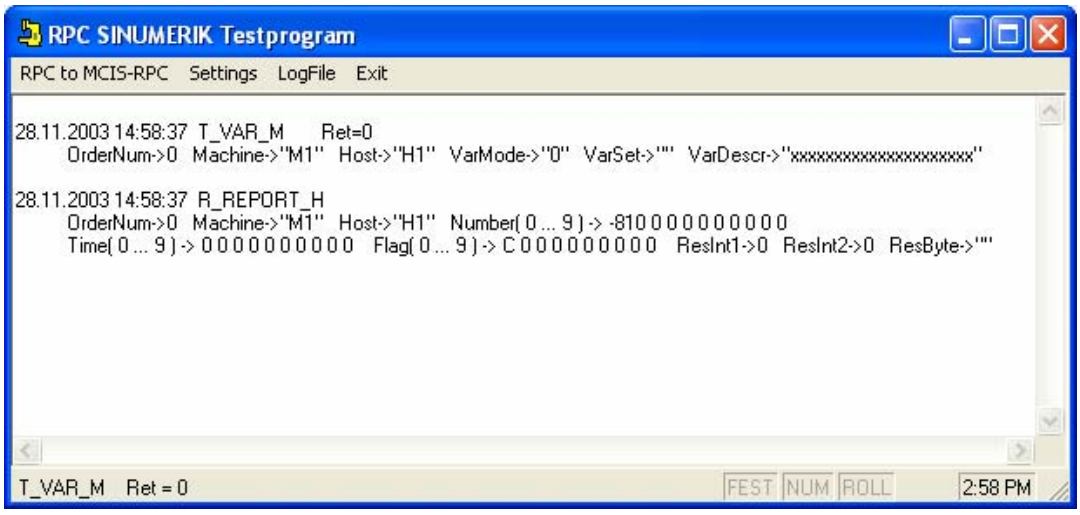
Flowchart for reading R parameters



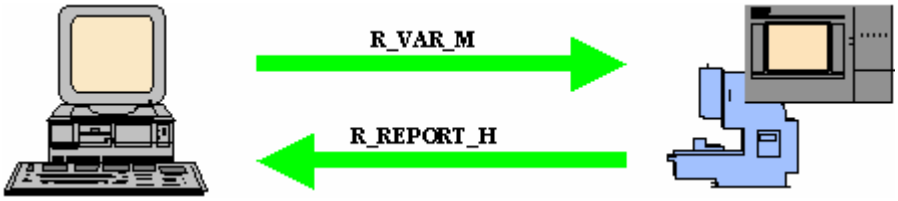
The VisualBasic application requests the value of an R parameter with the RPC: **T_VAR_M()**.
RPC SINUMERIK supplies the current value of the R parameter with the RPC: **R_VAR_H().**



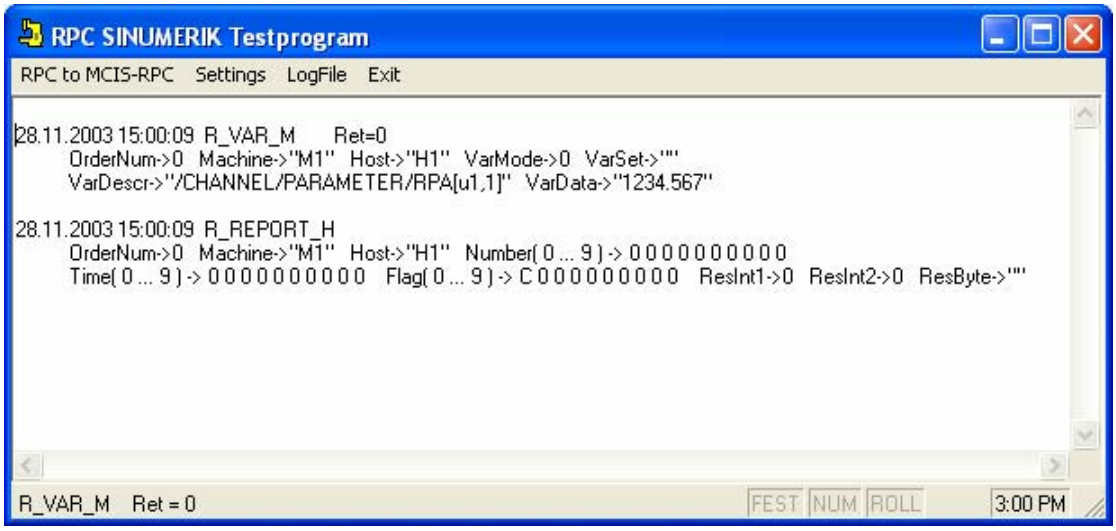
In the event of an error, RPC SINUMERIK responds with the RPC: **R_REPORT_M()**.



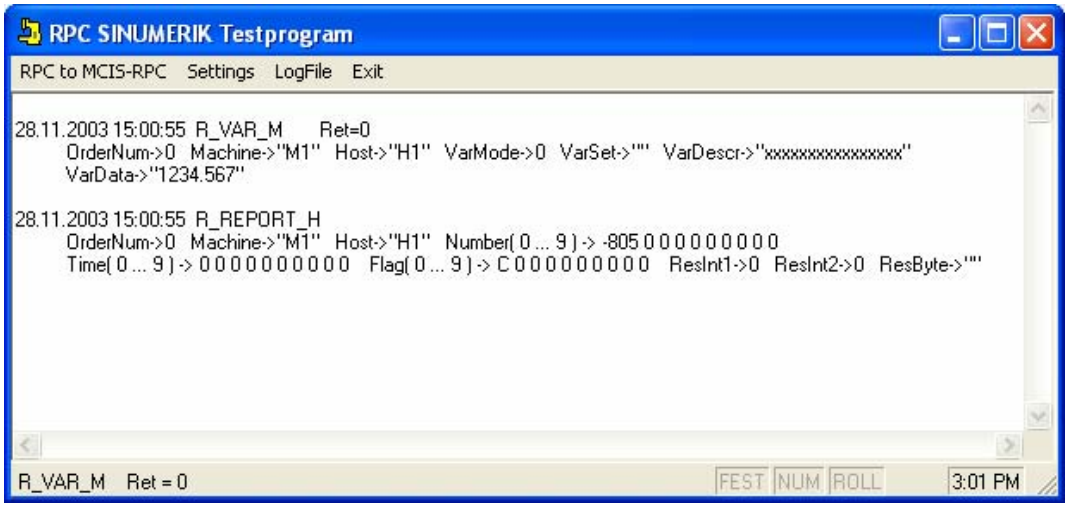
Flowchart for writing R parameters



The VisualBasic application passes the data for an R parameter with the RPC: **R_VAR_M()** to RPC SINUMERIK. RPC SINUMERIK acknowledges the write operation with the RPC: **R_REPORT_H()**.



In the event of an error the RPC: **R_REPORT_H()** is also sent by RPC SINUMERIK. However, the parameter **Number(0)** contains the relevant error code.

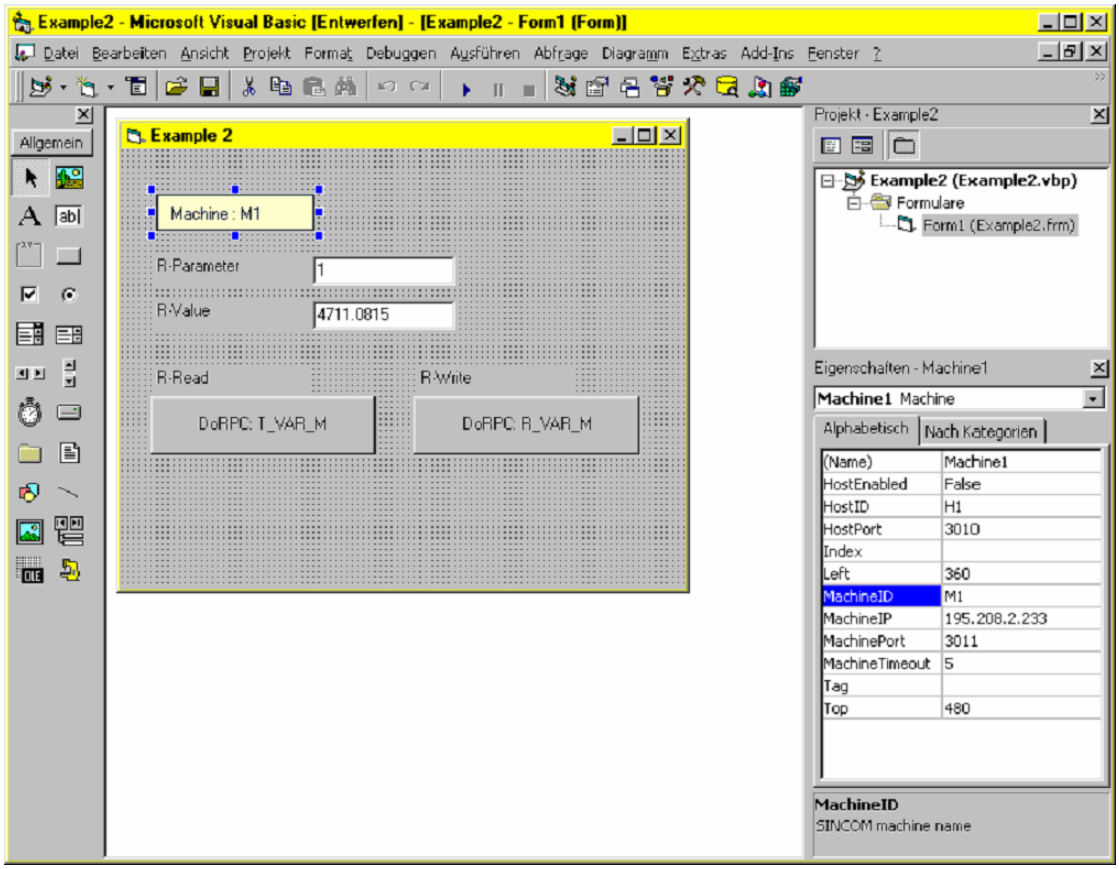


Visual Basic source code

Integration of the RPC SINUMERIK-OCX is performed as already described in section "10.5.1 Example 1 - Querying the machine state (Visual Basic) Integrating the RPC SINUMERIK-OCX component in Visual Basic 6.0".

In input field: **R parameter** the number of the R parameter to be read and written is entered. In input field: **R value** the current value of the R parameter is displayed on reading or the new value entered for writing.

You send the RPCs T_VAR_M() and R_VAR_M() by clicking on the appropriate buttons.



Date: \Examples\Example2\Example2.frm

```
Option Explicit

Option Explicit
Private Sub cmdR_VAR_M_Click()

   'write R parameter

   Dim ret                 As Long
   Dim RParam    As Long        ' R parameter number
   Dim RItem               As String      ' item for access
   Dim RValue              As String


   RParam = Val(txtRParam.Text)
   RItem = "/Channel/Parameter/R[" & RParam & "]"
   RValue = txtRValue.Text

   ret = Machine1.R_VAR_M(0, 0, "", RItem, RValue)
   If ret <> 0 Then MsgBox "R_VAR_M() -> " & ret, , "ERROR"

End Sub

Private Sub Machine1_RxVARxH(ByVal OrderNum As Long, ByVal VarMode As Long, _
                  ByVal VarSet As String, ByVal VarDescr As String, ByVal VarData As String)

   'show R parameter in the form

   txtRValue.Text = VarData

End Sub

Private Sub Machine1_RxREPORTxH(ByVal OrderNum As Long, ByVal Typ As Long, _
          ByVal Number As Variant, ByVal Time As Variant, ByVal Flag As Variant, _                    ByVal
ResInt1 As Long, ByVal ResInt2 As Long, ByVal ResByte As String)

   If Number(0) <> 0 Then
      MsgBox "On RPC R_REPEOR_H ( ... Number(0)->" & Number(0) & " )"
   End If

End Sub
```

10.5.3 Example 3 - active reading of R parameters (Internet Explorer)

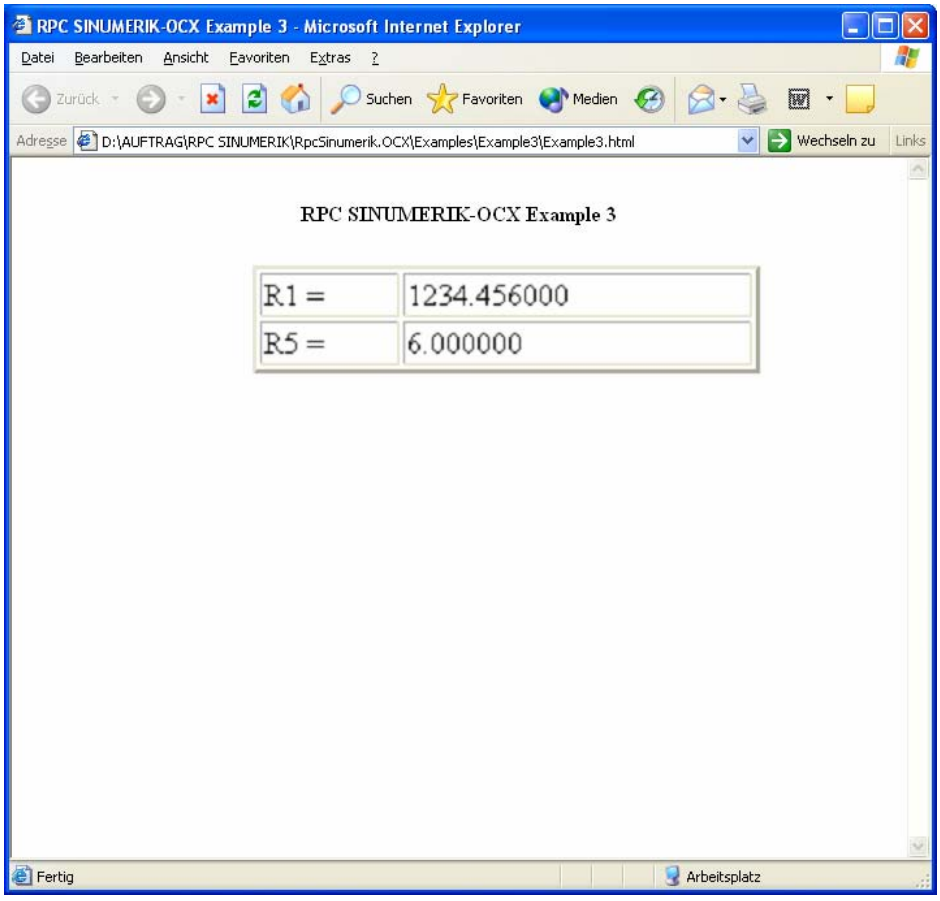
In this example, active reading of R parameters using MS Internet Explorer is demonstrated. The active reading function (also called hotlink) enables the RPC SINUMERIK-OCX to be informed by the RPC SINUMERIK immediately about every change to data from a variable set.

In this example, a variable set with the name "Set01" consisting of R parameters R1 and R5 is used.
The variable set is defined in file: c:\add_on\scvarset.ini on the control.

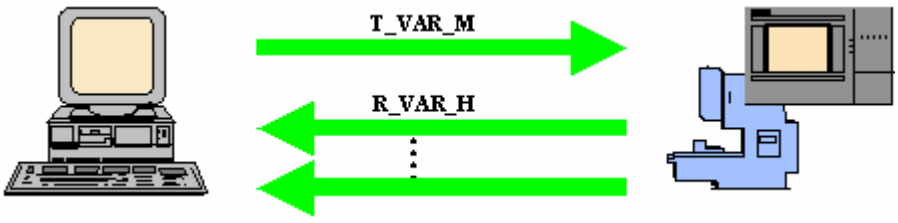
```
[Set01]
Mode=0
Host=FLR1
Var01=/Channel/Parameter/R[1]
Var02=/Channel/Parameter/R[5]
```

Use of MS Internet Explorer requires the RPC SINUMERIK-OCX to have been installed in advance.

File: \Examples\Example3\Example3.html displayed with MS Internet Explorer

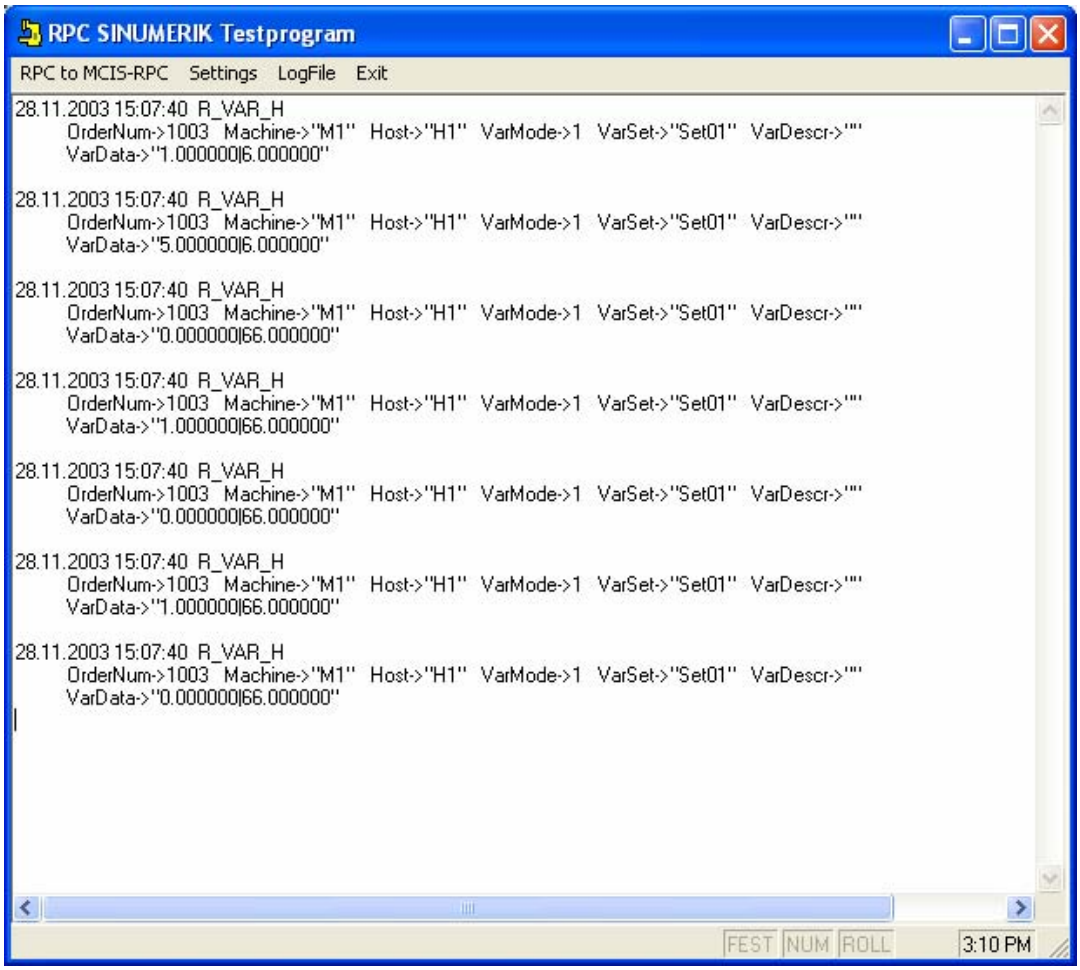


Flowchart for active reading of R parameters



When the HTML page is loaded, the RPC **T_VAR_M()** is sent to RPC SINUMERIK via a VBScript function. With this RPC, the current values of the variables in the set: **"Set01"** are requested. This data is immediately supplied by RPC SINUMERIK with the RPC: **R_VAR_H()**.

RPC SINUMERIK signals any change to the variables from the set (R1 or R5) with the RPC: **R_VAR_H().**



Source code for the HTML page

The RPC SINUMERIK-OCX is integrated in the HTML code by means of the <OBJECT> tag.
The attributes of the RPC SINUMERIK-OCX are stored within the <OBJECT> tag.

File: Examples\Example3\Example3.html

```
<HTML>
  <HEAD>
        <TITLE>MCIS_RPC.OCX Example 3</TITLE>
  </HEAD>

  <BODY>

    <OBJECT classid=CLSID:EDF199C1-4F2E-11D3-9DC3-00A0249B4877 id=Machine1>
        <PARAM NAME="MachineID"                 VALUE="M1">
        <PARAM NAME="MachineIP"                 VALUE="195.208.2.233">
        <PARAM NAME="MachinePort"               VALUE="3011">
        <PARAM NAME="MachineTimeout"      VALUE="5">
        <PARAM NAME="HostID"              VALUE="H1">
        <PARAM NAME="HostPort"                  VALUE="3010">
    </OBJECT>

        <P align=center><STRONG>MCIS_RPC.OCX Example 3</STRONG>  </P>


        <TABLE border=2 align=center width=60% id=TABLE1>

                <TR>
                        <TD> R1 = </TD> <TD><LABEL id=R1Param></LABEL> </TD>
                </TR>

                <TR>
                        <TD> R5 = </TD> <TD><LABEL id=R5Param></LABEL> </TD>
                </TR>

        </TABLE>

  </BODY>
```

The HTML page contains the following three VBScript functions.

Window_OnLoad	Called when the HTML page is loaded.
Machine1_RxVARxH	Called when the RPC R_VAR_H is received.
Machine1_RxREPORTxH	Called when the RPC R_REPORT_H is received.

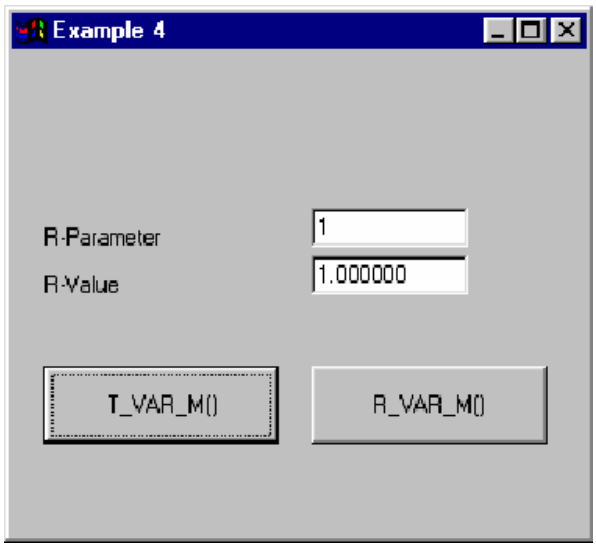
Continuation of the file: Examples\Example3\Example3.html

```
<SCRIPT LANGUAGE="VBScript">

    Option Explicit

    Sub Window_OnLoad

        dim ret

        ret = Machine1.T_VAR_M(0, 0, "Set01", "")
        if ret <> 0 then MsgBox "T_VAR_M()->" & ret


    End Sub

    Sub Machine1_RxVARxH( OrderNum, VarMode, VarSet, VarDescr, VarData )

      dim pos

      pos = InStr( VarData, "|" )

      if pos = 0 then
            R1Param.innerText = VarData
        else
            R1Param.innerText = Left(VarData, pos-1)
            R5Param.innerText = Mid (VarData, pos+1)
        end if

    End Sub

    Sub Machine1_RxREPORTxH( OrderNum, Typ, Number, Time, Flag, ResInt1,
                                                    ResInt2, ResByte )

     If Number(0) <> 0 Then
       MsgBox "On RPC R_REPEOR_H ( ... Number(0)->" & Number(0) & " )"
     End If

    End Sub


</SCRIPT>

</HTML>
```

10.5.4 Example 4 - reading and writing R parameters (Visual J++)

In this example, the same functionality is implemented as in example 2 but this time using MS Visual J++ 6.0 SP3.

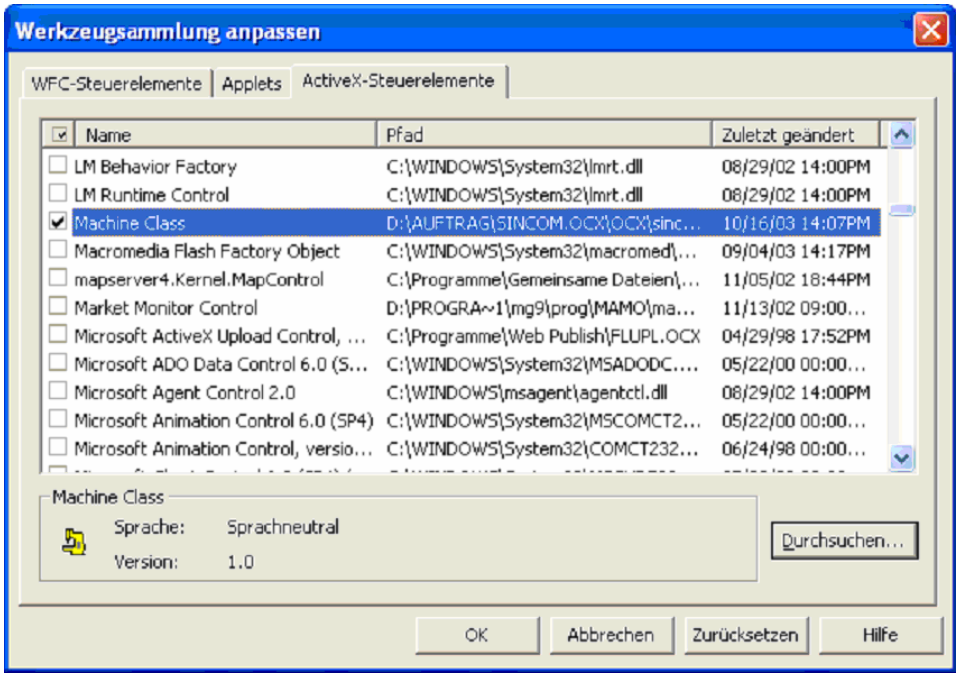
The flowchart is the same as for Example 2.



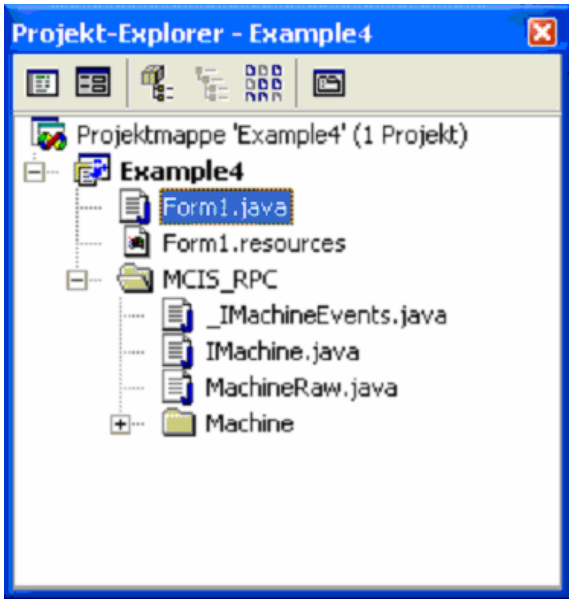
Integrating the RPC SINUMERIK-OCX in MS Visual J++

The MS Visual J++ development environment allows use of ActiveX components. Integration is performed via the menu: **"Tools -> Customize ToolBox -> ActiveX Controls"**

In this form, the entry **"Machine Class"** must be selected.



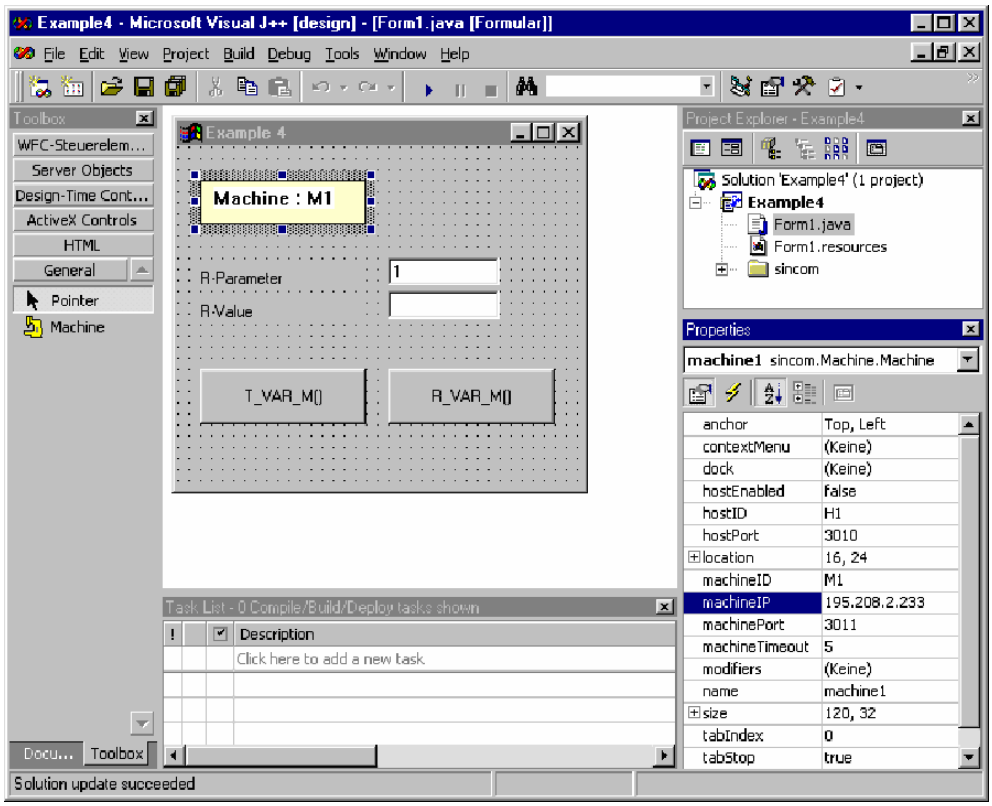
With integration of the RPC SINUMERIK-OCX, additional classes are generated by VJ++ in the RPC SINUMERIK directory.



Source code of the Visual J++ application

In input field **R parameter** the number of the R parameter to be read and written is entered. In input field **R value** the current value of the R parameter is displayed on reading or the new value entered for writing.

You send the RPCs T_VAR_M() and R_VAR_M() by clicking on the appropriate buttons.



File: Examples\Example4\Form1.java

```
private void cmdT_VAR_M_click(Object source, Event e)
{
        // read R parameter

        long    ret;
        String[] VarDescr       = new String[1];        // item for access
        String[] VarSet         = new String[1];

        VarSet[0]       = "";
        VarDescr[0]     = "/Channel/Parameter/R[" + txtRParam.getText() + "]";


        ret = machine1.T_VAR_M(0,0,VarSet,VarDescr );
        if ( ret != 0 ) MessageBox.show("T_VAR_M() -> " + ret);
}

private void cmdR_VAR_M_click(Object source, Event e)
{
        // write R parameter

        long    ret;
        String[] VarDescr       = new String[1];        // item for access
        String[] VarSet         = new String[1];
        String[] VarData = new String[1];

        VarSet[0]       = "";
        VarDescr[0]     = "/Channel/Parameter/R[" + txtRParam.getText() + "]";
        VarData [0]     = txtRValue.getText();

        ret = machine1.R_VAR_M(0, 0, VarSet, VarDescr, VarData);
        if ( ret != 0 ) MessageBox.show("R_VAR_M() -> " + ret);
}

private void machine1_RxVARxH(Object source, MCIS_RPC.Machine.RxVARxHEvent e)
{
        // show R parameter in the form

        txtRValue.setText( e.VarData );
}

private void machine1_RxREPORTxH(Object source,
                                 MCIS_RPC.Machine.RxREPORTxHEvent e)
{
        int    ErrorNr = e.Number.getVariantArray()[0].getInt();

        if ( ErrorNr != 0 )
        {
           MessageBox.show( "On RPC R_REPORT_H ( ... Number(0)->" + ErrorNr + " )" );
        }
}
```

■

I Index

I.1 Keyword index

R

S

T

V

W

■

Motion Control Information System

SINUMERIK 840D/840Di/810D
RPC SINUMERIK Computer link

PLC/NCK Interface (FBR/NPL)

1

1 PC and Machine PLC Interface

1.1 Description

Description of the interface between the computer communication software (RPC SINUMERIK) and the machine PLC.

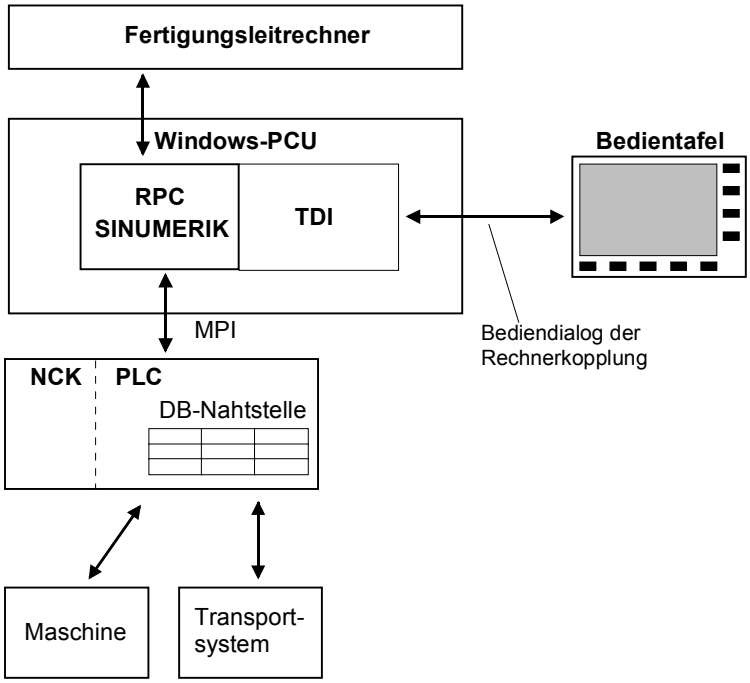


Fig. 1-1 Overview

Note

References to Part 1 "Host Computer Interface" in the following sections are specified using the following notation: (See /NFL/ <section no.> section title), e.g. (See /NFL/Section 5.8: Accepting Transferred Data)

An interface DB is required for communication between the RPC SINUMERIK and the PLC. The Siemens Standard data block 12 is reserved for this. The DB is set up by the user. The data elements of the DB interface are organized in blocks, each relating to an aspect of the interface (e.g. global data, docking position data, NC program assignment). The blocks are presented in tabular format. All blocks are stored contiguously in the interface DB.
Binary data elements of the type "int(WORD)" and "long(DWORD)" are stored in the DB in S7 format (little-endian). When the PLC accesses these data elements, it converts the data to Intel format (big-endian). Data elements which represent names are implemented as byte fields with ASCII characters.
The interface is described in tabular format. The "Access" column indicates who describes the field. The following abbreviations are used in this column:

- RPC SINUMERIK Computer link software on PCU (indirectly from the host computer)
- Operator Interactive program of the computer link
- PLC PLC user program

In order to minimize the communication load on internal resources, each change in the RPC SINUMERIK interface is relayed by means of a "request from the PLC" (part of the interface). RPC SINUMERIK detects such requests immediately (Hotlink).

Machine

Global data	
1. docking position	The number of docking positions is defined in the global data
2. docking position	
...	
nth docking position	

Transport system

Global data	
1. docking position	The number of docking positions is defined in the global data
2. docking position	
...	
nth docking position	
1. transport job	The number of transport jobs is defined in the global data
2. transport job	
...	
nth transport job	

1.2 Global data

Table 1-1 Global data list

Data element	Abbreviated name	Data type	Access from	Offset
Request from PLC	PLCReq	byte	PLC/RPC SINUMERIK	0
Change trigger	Trigger	byte	PLC	1
Request from RPC SINUMERIK	SCReq	byte	RPC SINUMERIK/PLC	2
Machine mode	MODE_PLC	byte	PLC	3
RPC mode	MODE_RPC	byte	RPC SINUMERIK / Operator	4
Flag for tool data	Data type	byte	PLC	5
Magazine number	MagNum	word	PLC	6-7
Location number	PlaceNum	word	PLC	8-9
T number	TNum	word	PLC	10-11
Number of docking positions	DockPosCount	byte	PLC	12
Number of program assignments	NC4WpcCount	byte	PLC	13
Machine status	MachineStatus	byte	PLC	14
NC mode	MachineMode	byte	PLC	15
Reserve 1	Reserve1	word	PLC	16-17
Reserve 2	Reserve2	word	PLC	18-19

Note

The individual entries of the data block are explained in the following table sequence.

Request from PLC

Table 1-2 Request from PLC

Bit No.	Function	Access from
0	Workpiece carrier status changed	PLC - 1/RPC SINUMERIK - 0
1	Report tool	PLC - 1/RPC SINUMERIK - 0
2	Status change	PLC - 1/RPC SINUMERIK - 0

The PLC indicates changes in the interface with this byte. After it has enabled the request byte, the PLC must enable the next bit in the trigger byte (see below).
The PLC can only write to the request byte again when it has been set to 0 by RPC SINUMERIK after processing.

Workpiece carrier status changed

Workpiece carrier status changed is set by the PLC whenever the status of one of the workpiece carriers is changed by the PLC (see Section 1.3: Docking Position Data of the Machine) or a pallet movement takes place within the machine.

Report tool

Report tool is set by the PLC whenever a tool is to be reported to the host computer (e.g. tool breakage). The data elements identify the tool which is to be reported:
"Magazine number" and
"Location number or T number".

Status change

The PLC sets **Status change** for each status change (machine mode, RPC mode, machine status, NC mode) that must be reported to the host computer.
RPC SINUMERIK must then send **R_MACHINE_H()** to the host computer (e.g. workpiece carrier arrival, NC start, NC end, mode changes, etc.).

Change trigger

The computer link sets up a DDE hotlink to this byte.
The PLC sets a bit in this byte when changes take place on the PLC. For each new trigger, the PLC must set the next bit and reset the last; after bit 7 it begins again with bit 0.

Request from RPC SINUMERIK

Table 1-3 Request from RPC SINUMERIK

Bit No.	Function	Access from
0	Synchronization flag	RPC SINUMERIK
1	Deactivate components	RPC SINUMERIK
2	Activate components	RPC SINUMERIK
3	Request write access to docking position data	RPC SINUMERIK
4	Project-specific special function 1	RPC SINUMERIK
5	Project-specific special function 2	RPC SINUMERIK

The individual bits are set and reset by RPC SINUMERIK.

Synchronization flag

The **synchronization flag** is set and reset by the host computer.
(See /NFL/Subsection 5.16.1: Synchronization Start/End **C_SYNCH_M ()**).
The machine status must remain unchanged for the duration of the synchronization process. The PLC must not start a new machining operation or execute any pallet movements.

Deactivate components

Deactivate components is set by the host computer (See /NFL/Section 5.15: Mode Switching **C_MODE_M ()**). This requests the PLC to deactivate components (drives) (see also machine modes, components deactivated).

Activate components

Activate components is set by the host computer (See /NFL/Section: 5.15 Mode Switching **C_MODE_M ()**). This requests the PLC to activate components (drives) (see also machine modes, components activated).

Request write access to docking position data

Request write access to docking position data is set by RPC SINUMERIK whenever it wants to change the docking position data (workpiece carrier status, follow-up machining, machining side). When the PLC has set the "write access to docking position data allowed" flag in the "machine mode" data element, only RPC SINUMERIK can make changes. RPC SINUMERIK must reset the request after carrying out the change, and the PLC must then also reset the "write access to docking position data allowed" flag. This coordination prevents RPC SINUMERIK from writing data to wrong docking positions due to a pallet movement.

Project-specific special function1

Project-specific special function1 is set by the host computer. (See /NFL/ Section 5.15: Mode Switching **C_MODE_M 5**)
Project-specific special function 1 is reset by the host computer. (See /NFL/ section 5.15 Mode switching **C_MODE_M 6**)

Project-specific special function 2

Project-specific special function 2 is set by the host computer. (See /NFL/ section 5.15 Mode switching **C_MODE_M 7**)
Project-specific special function 2 is reset by the host computer. (See /NFL/ section 5.15 Mode switching **C_MODE_M 8**)

Machine mode

Table 1-4 Machine mode

Bit No.	Function	Access from
1	Units deactivated (end of the working day)	PLC
2	Write access to docking position data allowed	PLC

Components deactivated

Components deactivated is set by the PLC when this status is detected. The request is set in bit 1 of the request flags.

Write access to docking position data allowed

Write access to docking position data allowed is set by the PLC as a response to the request "Request write access to docking position data". This coordination prevents RPC SINUMERIK from writing data to wrong docking positions due to a pallet movement.

RPC SINUMERIK mode

Table 1-5 RPC SINUMERIK mode

Bit No.	Mode	Access from
0	Host computer mode unmanned	Operator
1	Host computer mode manned	Operator
2	Manual mode	Operator
3	Special mode	RPC SINUMERIK, operator
4	Host computer 1 offline	RPC SINUMERIK
5	Host computer 2 offline	RPC SINUMERIK
6	Is registered as Machine Mode 500 in R_MACHINE_H	PLC
7	Is registered as Machine Mode 600 in R_MACHINE_H	PLC

RPC SINUMERIK mode can be set interactively by the operator in an RPC SINUMERIK dialog. The special mode can also be selected and deselected by the host computer with **C_MODE_M ()**. If RPC SINUMERIK detects an interruption in the connection, it sets the bit for "offline". When the offline bit is active, data are no longer transmitted from RPC SINUMERIK to the host computer.

Host computer mode manned/unmanned

In the host computer modes **manned** and **unmanned** the NC start is triggered by the PLC (on the initiative of the host computer); the difference between **host computer mode unmanned** and **host computer mode manned** relates to the possible use of different strategies in response to a fault during unmanned and manned production.

Manual mode

In **manual mode** there is no automatic NC start, but material flow on the machine is still automatic.

Special mode

In **special mode** there is neither an automatic NC start nor automatic material flow.

Offline

Offline means that there is no connection to the host computer, and no data are sent to the host computer. **Offline** is canceled when RPC SINUMERIK detects the reestablishment of the connection as a result of the arrival of an RPC from the host computer.

Flag for tool data

The **tool data flag** can be used to select one of three sets of tool data for transfer to the host computer. The data areas contained in these sets of data are defined using the configuration program. Flags 21, 22 and 23 are allowed. The flag is transferred together with the tool data to the host computer (see /NFL/ section 4.1 Tool data).

Magazine number, location number, T number

The tool to be reported is specified by the magazine number and location number, or alternatively by the T number. If the **T number** is specified, the **magazine number** and **location number** must be set to 0, and vice versa.
With the request from the PLC: **Report tool**, RPC SINUMERIK is requested to read data elements:

- Magazine number,
- Location number,
- T number, and
- Flag for tool data

and to transfer the tool data to the host computer. RPC SINUMERIK must subsequently clear these data elements (by filling them with zeros).

Note

Tool messages which occur during tool loading and tool unloading are not initiated by the PLC user program (for more information, please see /NFL/).

Number of docking positions

The number of docking positions on the machine is stored statically during commissioning of the machine. The number matches the number of docking position data blocks on the interface (see Section 1.3: Docking Position Data on the Machine).

Number of program assignments

The number of program assignments on the machine is stored statically during commissioning of the machine. The number matches the number of NC program assignment blocks in the interface (see Section 1.4: NC Program Assignment).

Machine status

Table 1-6 Machine status

Bit No.	Function	Access from
0	Machine is activated	PLC
1	Machine fault detected	PLC
2	Restart machine	PLC

NC mode

Table 1-7 NC mode

Bit No.	Function	Access from
0	Automatic	PLC
1	MDA	PLC
2	JOG	PLC
3	TEACH IN	PLC

The machine status and the NC operating mode are signaled to the host computer with **R_MACHINE_H**. They are not evaluated in the computer link server, however.

Reserve 1 and 2

These variables can be used as required by the machine manufacturer using the PLC. The values are reported to the host computer by **R_MACHINE_H ()**, although these values are not usually processed by the host computer.

1.3 Machine manufacturer data

The docking position data each describe a machine station (machining station, in/out station). The number of machine stations is stored in the **number of docking positions** data element in the global data.

Table 1-8 Docking position data

Data element	Abbreviated name	Data type	Access from
Docking position status	DockPos Status	byte	PLC
Workpiece carrier status	WPCStatus	byte	PLC/RPC SINUMERIK
Workpiece carrier	WPC	Byte[6]	PLC
Machining side	ClampCube Side	word	PLC/RPC SINUMERIK
Follow-up machining	FB	byte	RPC SINUMERIK
Reserve1	Reserve1	byte	PLC

Docking position status

- Bit 0 = fault
- Bit 1 = disabled

The bit array describes the actual status of the docking position and is set by the PLC. If no bit is enabled, the docking position is available. The **fault** bit is set and canceled in response to I/O signals. The cause of the fault is reported to the host computer using the report function (see /NFL/ Section 5.5. **R_REPORT** Messages). The PLC does not execute any pallet transports between stations which have the **fault** status. If the docking position is **disabled**, it cannot be approached by the transport system.

Workpiece carrier status

Table 1-9 Workpiece carrier status

Bit No.	Function	Access from
0	New arrival (no program assignment)	PLC
1	Machining required (with program assignment)	RPC SINUMERIK
2	Prepare program selection	PLC
3	Program selection done	RPC SINUMERIK
4	Machining in progress	PLC
5	Machining finished	PLC
6	Machining aborted	PLC
7	Machining not required (only for buffering)	PLC

New arrival

The PLC allocates the status **new arrival** to a newly arrived workpiece carrier. (Exception: Machining not required). This status causes the host computer to perform the program assignment. When the program assignment is complete, the "machining required" status is set by the computer link server.

Machining required

The PLC sets the **prepare program selection** status for workpiece carriers with the status **machining required** as soon as the currently active machining operation is complete.

Program selection

When the program selection is complete, i.e. when the program assigned to the pallet by the host computer has been loaded in the NCK and selected for execution, the computer link server enables the **program selection complete** status for the associated workpiece carrier. The PLC can now trigger the NC Start. It is however, the responsibility of the machine manufacturer to ensure that an NC Start is not triggered by the PLC until all safety criteria have been met (e.g. safety door shut, etc.).

Machining in progress

The PLC sets the **machining in progress** status when machining has started. When the machining operation is complete, the PLC sets the **machining finished** status for the corresponding workpiece carrier. Workpiece carriers with the status **machining finished** are transported independently by the PLC to an unloading station.

Machining finished

If the **follow-up machining** flag is set, the workpiece carrier remains at the machining station. The computer link server sets the **machining required** status again in response to the **machining finished** status. In response to this action, the PLC uses the **prepare program selection** status to request the computer link server to select a program for follow-up machining. The subsequent procedure is the same as for the initial machining operation.

Machining aborted

The **machining aborted** status is set if a workpiece carrier is no longer processed after a fault. This flag occurs chiefly during unmanned production. A workpiece carrier with this flag may not be used for a further operation on another machine, and may only be transported to a storage location.

Machining not required

A workpiece carrier supplied solely for the purpose of transport control buffering is assigned the **machining not required** status instead of the **new arrival** status. The transport control system transfers this information to the PLC. The host computer does not assign programs to workpiece carriers with this status.

Workpiece carrier

Name of the workpiece carrier currently located at the docking position (e.g. "WST01"). This information is entered by the PLC. This assumes that the information can be transferred from the transport system or directly from the workpiece carrier. If no workpiece carrier is at the docking position, the array must be filled with binary 0s.
The identifier must be terminated with '\0' after the last character because RPC SINUMERIK expects a string. This means only identifiers up to 5 bytes long are permissible.

Follow-up machining

This flag is enabled by the computer link server at the same time as the machining status **program selection complete** is set. It informs the PLC whether a follow-up machining operation is pending after the current operation. The PLC uses this information in order to control transport of the workpiece carrier within the machine.

Machining side

This information is enabled by the computer link server at the same time as the machining status **program selection complete** is set. The PLC uses this value to set the side of a clamp cube for machining, or passes the value on to the NCK. With cubes, the computer link defines the order of machining for the sides with reference to the program assignment. If it is necessary to modify the machining order from the PLC, the program assignment data can be mirrored in a separate PLC data block by means of an entry in the configuration file of the computer link server. This gives the PLC read access to the data. The machining side selected by the PLC is reported to the computer link server in the **machining side** field. This occurs at the same time as the machining status **program selection complete** is set. The computer link server executes the program selection for the side specified by the PLC. The remaining procedure is the same.

1.4 NC program assignment (option)

This interface is optional and is used for workpiece carriers with clamp cubes, when the PLC rather than the host computer is to specify the machining order for the individual sides. These data are managed by RPC SINUMERIK and mirrored here for reading by the PLC.
The number of NC program assignment blocks is stored in the data element **number of program assignments** in the global data (Section 1.3: Docking Position Data for the Machine).

Table 1-10 NC program assignment

Data element	Abbreviated name	Data type	Access from
Workpiece carrier	WPC	Byte[6]	RPC SINUMERIK
Machining side	ClampCubeSide	word	RPC SINUMERIK
NC program flag	NCProgramMark	byte	RPC SINUMERIK
Machining status	Status	byte	RPC SINUMERIK
NC program	NC program	Byte[128]	RPC SINUMERIK

Workpiece carrier, machining side

The NC program assignment allocates an NC program to a "workpiece carrier" and a "clamp cube side" (machining side).

NC program flag

The NC program flag is used to indicate whether the same NC program is used on several sides of a clamp cube. On entry by RPC SINUMERIK, "NCProgramMark" = 1 is set for the first side of a workpiece carrier. If the next side uses a different NC program, "NCProgramMark" = 2 is set, etc. If the third side now has the same NC program as side 1, the same "NCProgramMark" as side 1 is assigned. This makes it possible to control the order of execution such that sides using the same NC program are machined consecutively.
Clamp cube sides which are machined using the same NC programs are assigned the same NC program flag.

Machining status

- Bit 1 = Machining is planned
- Bit 4 = Machining in progress
- Bit 5 = Machining finished

List size

The size of the list for NC program assignment is configurable and determines the length required in the DB. It is calculated from the number of docking positions on the machine times the maximum number of usable sides of a clamp cube.

1.5 Languages in RPC SINUMERIK

RPC SINUMERIK generally supports all languages which are covered by the ASCII character set or for which a language package can be installed on the SINUMERIK.
Languages that use an extended character set (e.g. Chinese, Russian, etc.) are processed by SINUMERIK in DBCS format. The following names must not be specified in any language that uses an extended character set:

- NC programs
- Tool data

■

2

2 Processes on the DB Interface

Description

Communication takes place using the interface DBs described above.
The following section describes what components read and write data to and from the interface fields and when this occurs.

2.1 Arrival of workpiece carrier

When a workpiece carrier arrives at a docking position on the machine, the PLC must read the name of the workpiece carrier or the transport system must pass the name to the machine. In addition, the transport system must inform the machine (if this information is not reported by the host computer to the machine [NC4WPC_M]) whether the workpiece carrier is supplied for machining or only for buffering.
The procedures used to transfer this information between the transport control system and the machine are not described in this manual.
The PLC must write the docking position status, the workpiece number and the workpiece status into the corresponding interface DB, and then set the request flag to indicate a change in the status signal and increase the change trigger.
RPC SINUMERIK then sends **R_MACHINE_H ()** to the host computer.

2.2 Production dialog between PLC/NCK and RPC SINUMERIK

Having detected from R_MACHINE_H () that a workpiece carrier requiring machining has arrived at the machine, the host computer sends one or more NC program assignments NC4WPC_M (). RPC SINUMERIK enters this information in an internal list and optionally in the interface for NC program assignment.

Determining the data for R_MACHINE

The interfaces for states and docking position data must be read along with the SINUMERIK mode (Automatic, MDA, JOG) and the program status. The mode which is reported to the host computer is a combination of the computer link mode and the SINUMERIK mode.

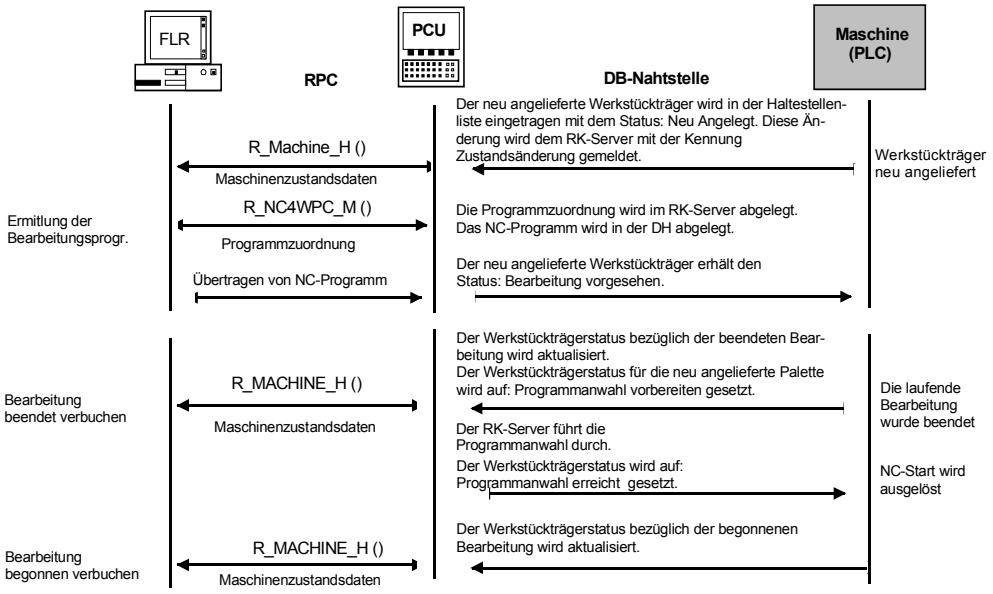


Fig. 2-1 Flow diagram

■

For notes

3

3 Interactive Program for RPC SINUMERIK

3.1 Interactive program in RPC SINUMERIK

General information

The interactive program in RPC SINUMERIK is available as a standalone area application in the HMI Advanced user tree.

The interactive program performs the following functions:

- Selects the status of RPC SINUMERIK
- Requests NC programs from the host computer
- Transfers NC programs to the host computer
- Switches to MCIS TDI (loading/unloading of tools to/from the host computer)

Languages

RPC SINUMERIK generally supports all languages which are covered by the ASCII character set or for which a language package has been installed on the SINUMERIK. Languages which use an extended character set (e.g. Chinese, Russian, etc.) are processed by SINUMERIK in DBCS format. In this case, data are displayed correctly on the PCU if the application on the host computer transmits the data in DBCS format.

3.2 Status of RPC SINUMERIK

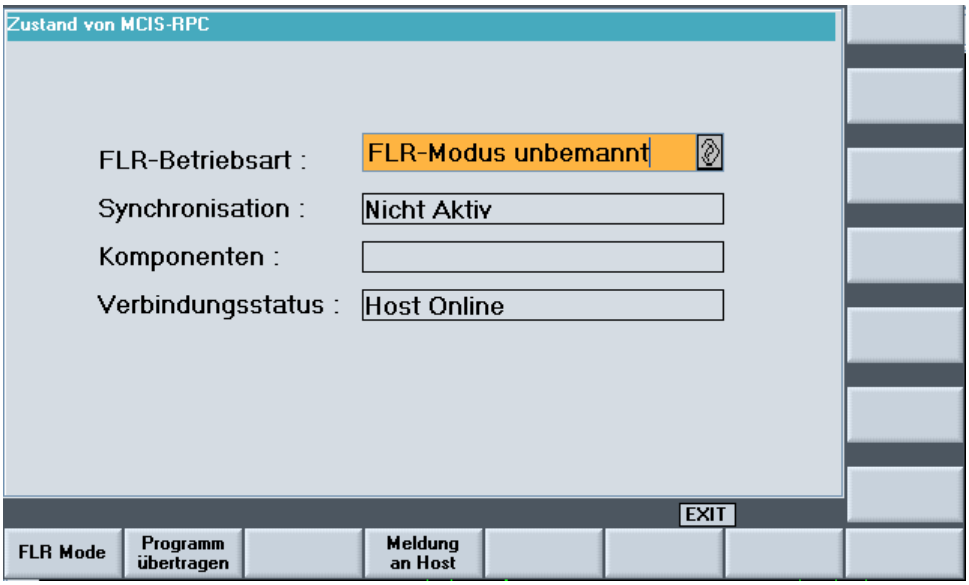


Fig. 3-1 The screen allows you to change the host computer mode

RPC SINUMERIK modes

- Host computer mode unmanned
- Host computer mode manned
- Manual mode
- Special mode

Relevant states are also displayed for RPC SINUMERIK.

3.3 Transfer program

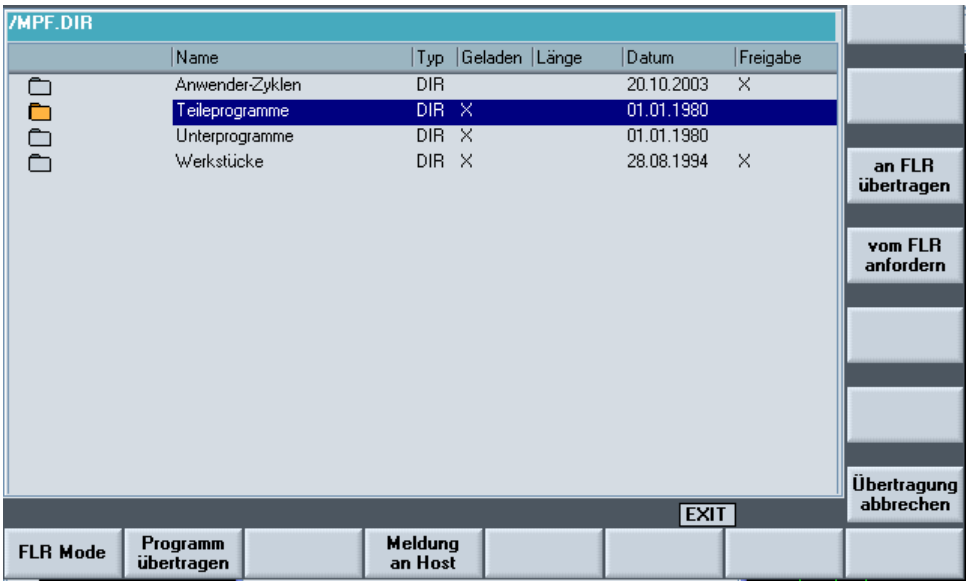


Fig. 3-2 Transferring NC programs/files

The operator can transfer files from the data management tree for the HMI Advanced to the host computer with the "Transfer to host computer" softkey or to the HMI Advanced with the "Request from host computer" softkey.

3.3.1 Send a program to the host computer

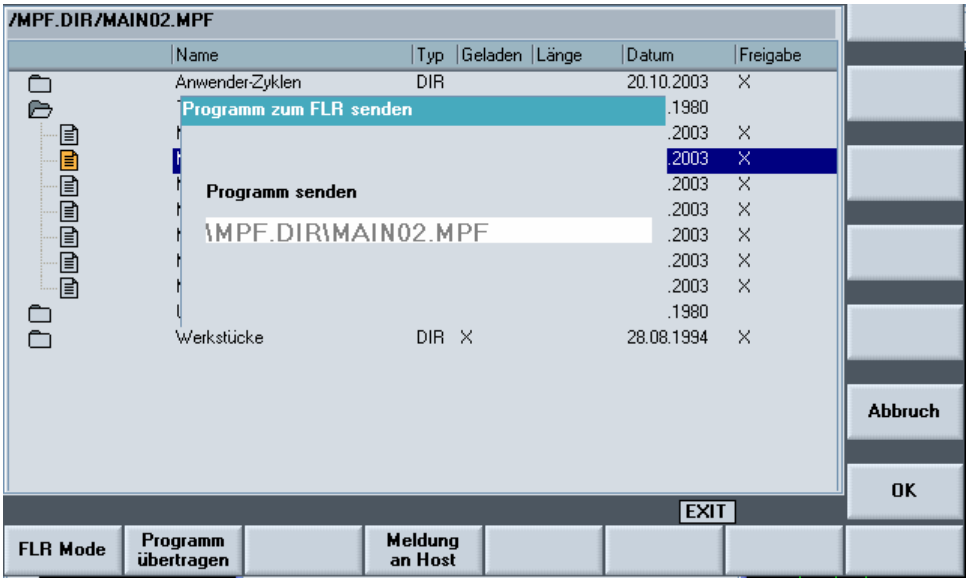


Fig. 3-3 Sending a program to the host computer

3.3.2 Request a program from the host computer

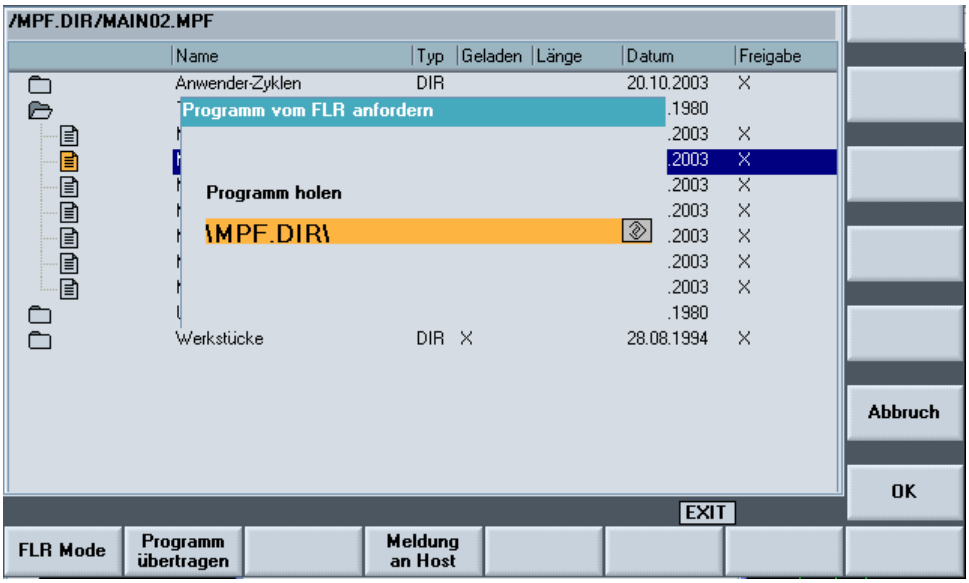


Fig. 3-4 Requesting a program from the host computer

The screen allows the operator to enter the name of the NC program to be requested from the host computer.

3.3.3 Send a message to the host

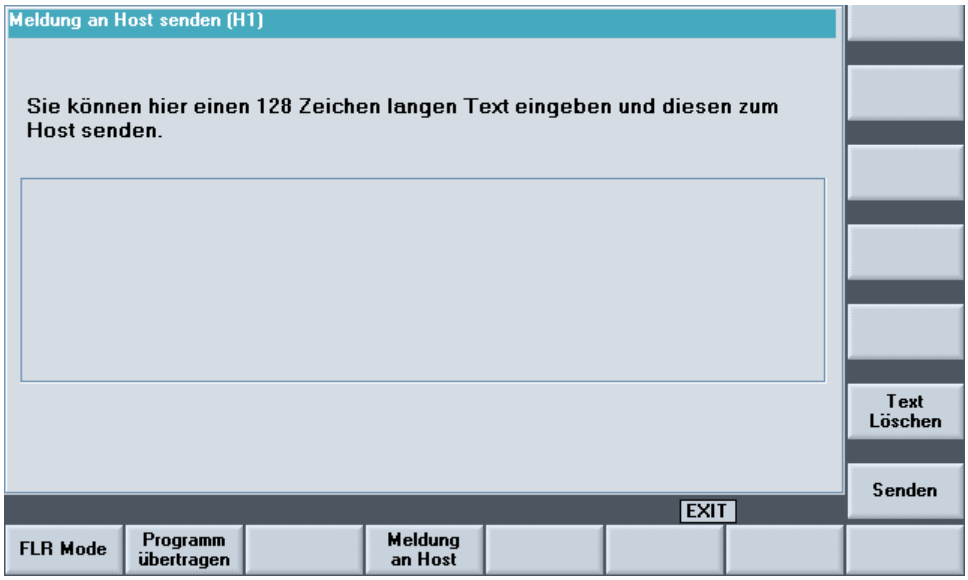


Fig. 3-5 Sending a message to the host

A text consisting of a maximum of 128 characters can be entered in the text box. The following vertical softkeys are provided or are displayed when required.

- Acknowledge host message
- Select host
- Deleting text
- Send

Acknowledge host message

If a message has been sent by the host to the MMC103/PCU50, it is entered with the transferred message text as alarm 65535. The pending alarm can be deleted with the "Acknowledge host message" softkey.

Select host

This button is only unhidden if more than one host has been configured. A list box will appear from which the host to which the message is to be sent can be selected.

Note

Default host is the first host configured.

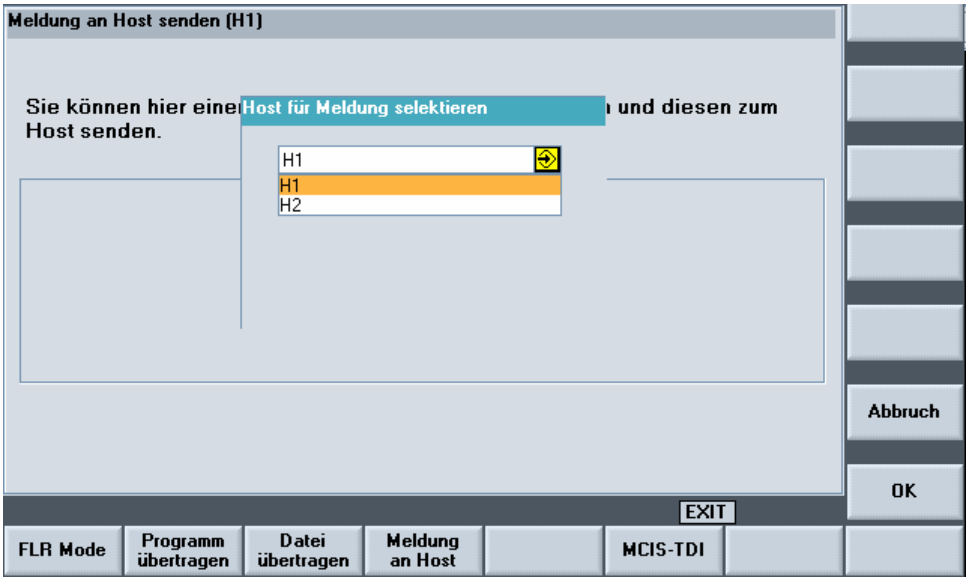


Fig. 3-6 Selecting the host

Deleting text

This softkey is used to delete the text you have entered.

Send

Pressing this softkey sends the entered text to the host.

3.3.4 Load/unload tools to/from the host computer

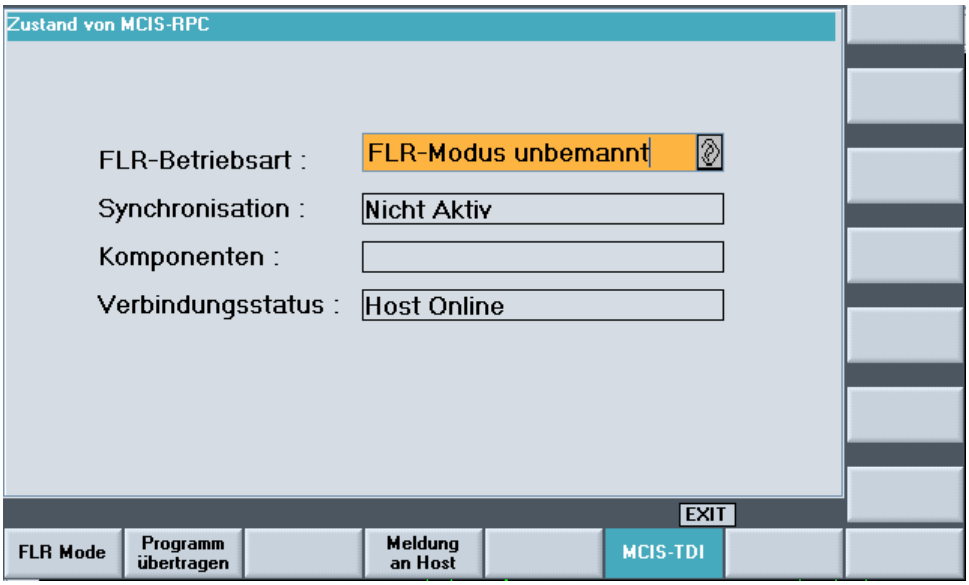


Figure 3-7 Form with MCIS TDI button

MCIS TDI

This button is only visible if MCIS TDI tool handling (tool data information) is installed on the HMI Advanced.

3.3.5 MCIS-TDI (Tool Data Information) tool handling

General information

The MCIS TDI tool handling module allows simulation of all tool handling operations within a plant, carrying out of all loading/unloading and conversion processes on the CNC machine or a tool store, and importing of tool data from a host computer, for example. Any of the containers within a plant can be involved in these processes. The TDI tool handling module can run in a network with TDI Cell/Plant (optional) or on a single-user station (e.g. for importing tool data from a host computer).

Note

MCIS TDI is not supplied as standard with RPC SINUMERIK and must be ordered separately.

Tool handling/selection

The tools to be moved can be selected via the entry screen for tool handling. The tool handling/selection screen form is divided into two parts. The top half displays an Explorer view of all the tool magazines positioned above the local component (operating / installation location) within the plant hierarchy.
In the bottom half the local plant component is displayed.

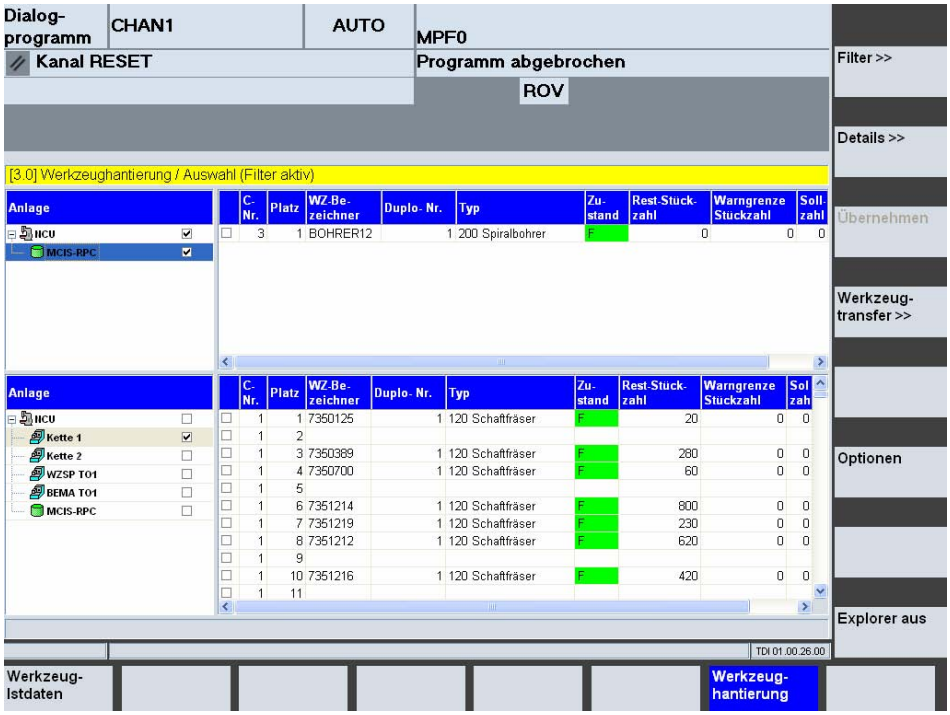


Fig. 3-8 Tool handling / selection

Note

Lower half of screen:
In the CNC version the view always shows the local component (unit).
In the PC version the view shows the entire plant, as set up in the configuration.

Depending on the movement operation (loading or unloading), both the top and the bottom display can be the source or target of the movement.

Specific tools can be selected in the table (by placing check marks in the selection box) for this movement.
Tools selected in this way are added to the movement list with the "Add" softkey.

One or more tools (e.g. cassette, tool carriage, loading magazine) can be selected for the movement list.
If only one tool is selected (source), a specific location can be specified in the target by placing a check mark in the table. This only works if you have selected a single tool.

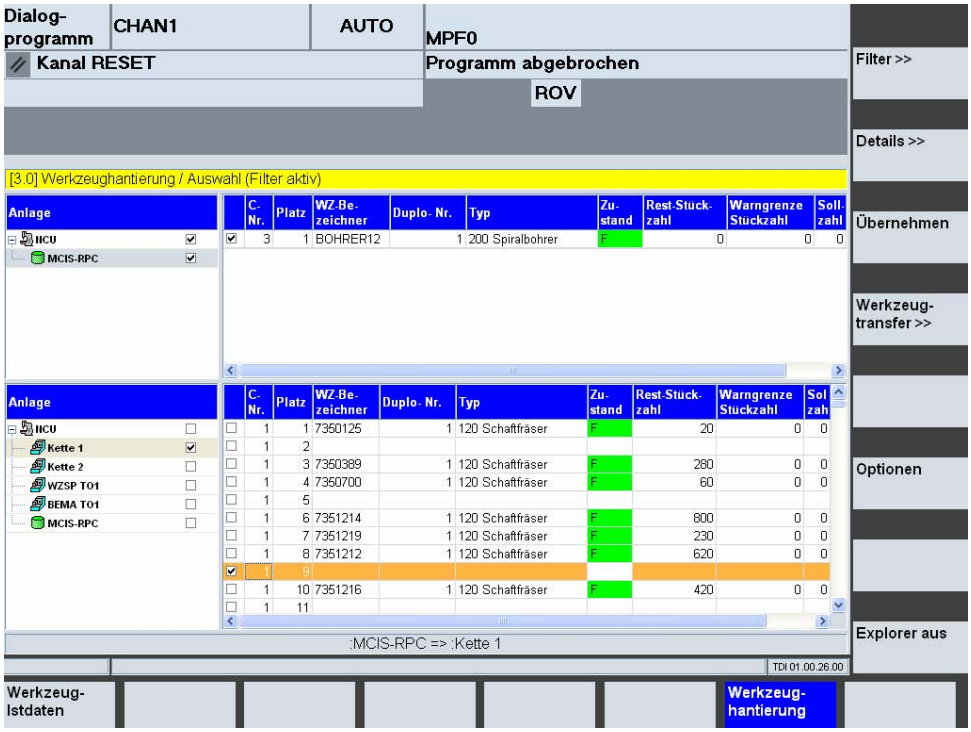


Fig. 3-9 Tool handling with info line

The source and target of the current movement summary are explicitly shown in the status line. A movement is defined by the following information in the info line:

- Location
- Container

Note

- You may only specify one machine as the target for tool movements.
- If tools have been added to the movement list by means of the "Add" operation, they can no longer be selected and the appearance of the check mark changes (grayed). Changes can only be made in the "Organize movements" screen form.
- If you change the selection of containers to be displayed in the Explorer view, you invalidate the tool selection of the deselected container. This applies until the selected tools are added to the movement list.
- The ticks for selected tools which you have transferred to the movement list remain set until you complete the operation.
- The movement list can be extended at any time with "Add tool".
- One possible handling target is the dismantling container, which in the plant configuration (Explorer) is represented by the trashcan symbol. Tools which are transferred to the disposal container remain in the container list for only a limited period. This list can be made available to a higher-level tool management system for a dismantling function.

Description of softkeys

The following functions can be executed using the vertical softkeys:

Filter
This key accesses the screen for defining selection filters and for selecting the columns that appear in the list. Filter settings made here only affect the part of the screen in which the cursor is positioned.

Details
Click this key to access details of the selected tool in order to view and change tool data. A distinction is made between NC data and PLC data.

Reuse
The selected tool movements are transferred to the tool movement list.

Tool movement
This key opens the list of all current tool movements.

Options
Opens the pop-up screen in which you can specify different sequences for tool movements.

Explorer on/off
Shows/hides the plant component tree.

For more information see Description of Functions for MCIS-TDI
Edition: 11/04
Order No.: 6FC5297-6AE01-0AP2

3.3.6 Load/unload tools to/from the host computer with TDI

General

The tools to be moved can be selected via the entry screen for tool handling.
The tool handling/selection screen form is divided into two parts. In the top half all the tool magazines assigned to the local component (point of use/installation) are displayed in Explorer within the plant hierarchy (default setting). The local plant component is displayed in the lower half.

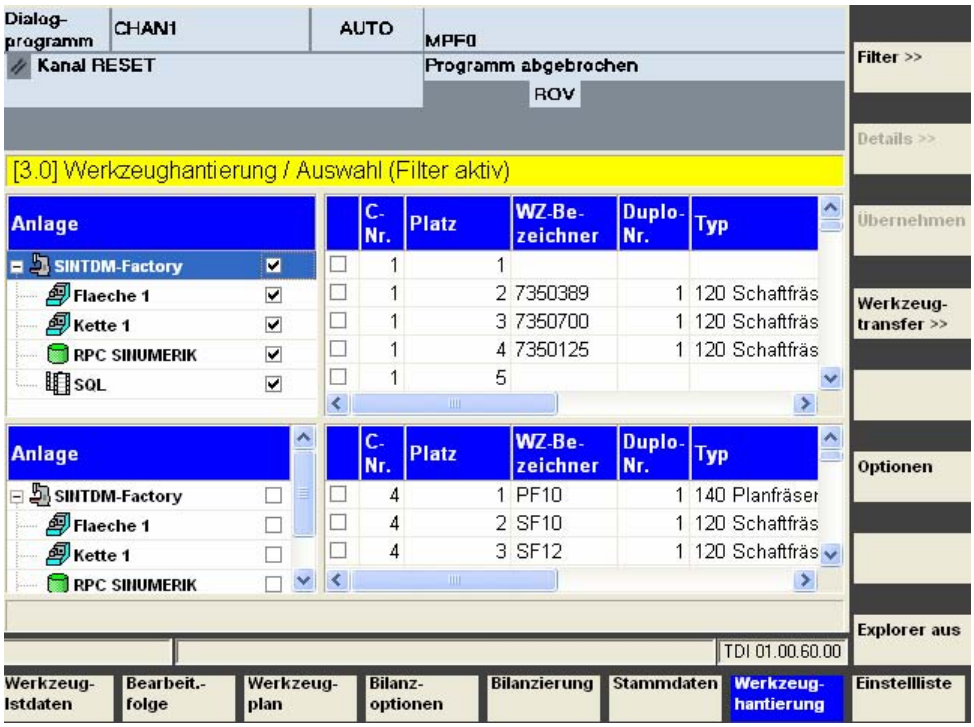


Fig. 3-10 Entry screen for tool handling/selection

Requesting tool data from the host computer

In order to request tool data from the host computer, the RPC container must be selected in the top Explorer view.
The filter settings can be used to restrict the view to the RPC container in the filter screen with "RPC plant view".

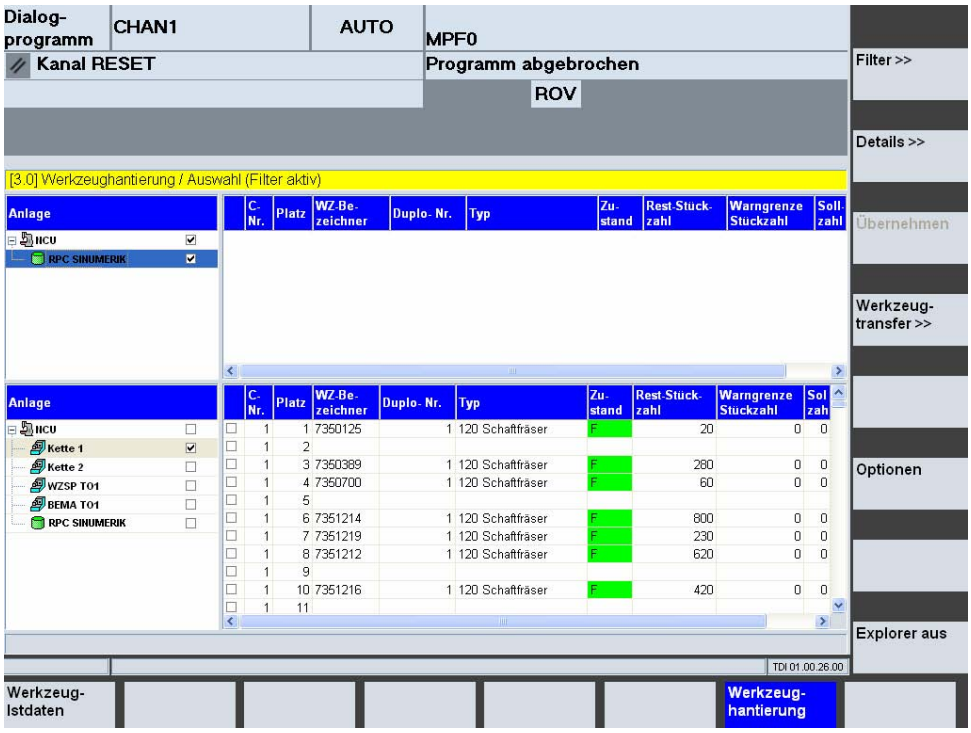


Fig. 3-11 Tool handling/filters

Tool data are then requested from the host computer in the filter screen.

Pressing the "Filter>>" softkey opens the screen for defining selection filters.

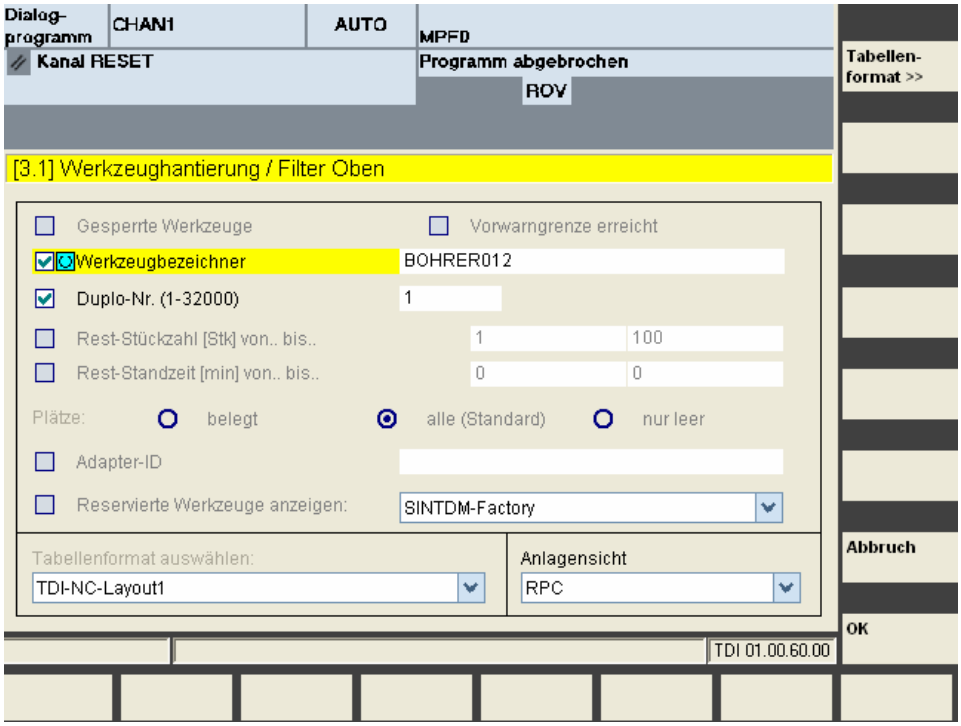


Figure 3-12 Defining selection filters

The tool identifier and duplo number are selected here and the data entered for the tool to be requested.
Press "OK" to request the data from the host computer and return to the tool handling screen.

Once the requested tool data has arrived from the host computer, it is displayed in the RPC container.
The tool can now be loaded (see Description of Functions for MCIS-TDI, "TDI tool handling" section).

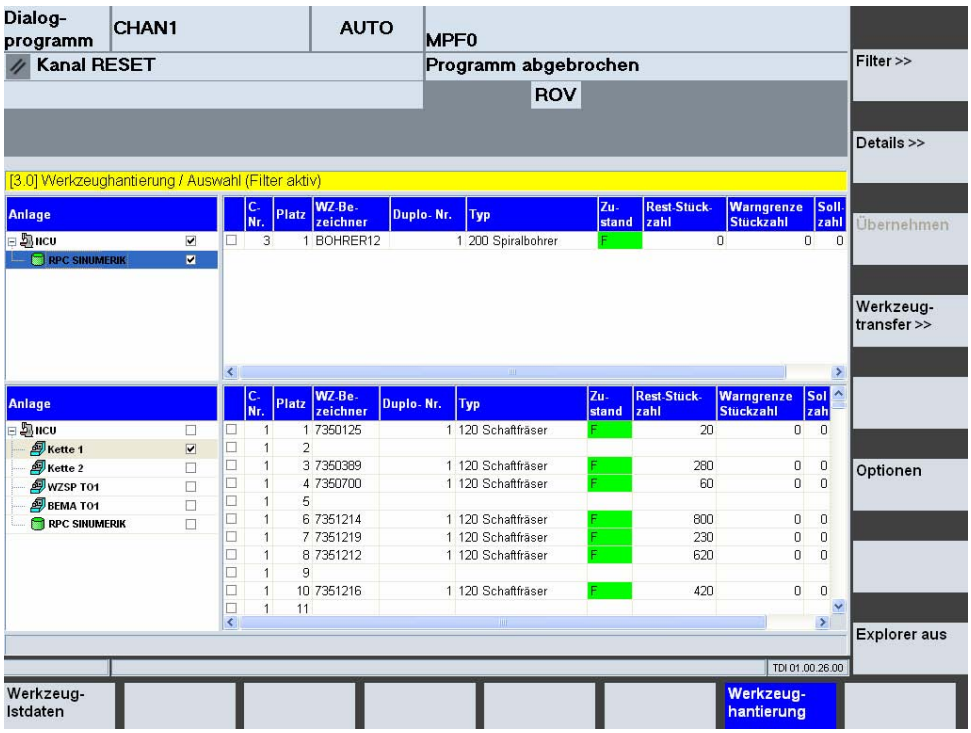


Fig. 3-13 Tool handling/filters: Request completed

Once the loading operation from the RPC container has been successfully completed, the tool data with the magazine and location numbers are reported to the host computer.

Note

The file format for the tool data corresponds to the data backup format of the NC840D (punched tape/ASCII format acc. to /BA/ ; such as in _N_TOx_TOA or _N_TOx_INI). The complete description of data content and layout is provided in /NFL/ chapter 4, Tool data.

Tool data to host computer (unloading tools)

When unloading tools via the RPC container, the tool data are reported to the host computer (tool handling, see Description of Functions for MCIS TDI, "TDI tool handling").

Note

The file format for the tool data corresponds to the data backup format of the NC840D (punched tape/ASCII format acc. to /BA/ ; such as in _N_TOx_TOA or _N_TOx_INI). The complete description of data content and layout is provided in /NFL/ chapter 4, Tool data.

If more than one tool has been unloaded via the RPC container, the tool data are reported separately to the host computer for each tool.

Note

Once the tool data have been successfully reported to the host computer, the tool is deleted from the RPC container.

■

4

4 Interface between RPC SINUMERIK and TPS-PLC

4.1 Description

An interface DB is required for communication between RPC SINUMERIK and TPS PLC. A Siemens-standard DB (DB12) is reserved for this purpose. The DB is set up by the user. The data elements of the DB interface are organized in blocks, each relating to an aspect of the interface (e.g. global data, transport job, docking position data). The blocks are presented in tabular format. All blocks are stored contiguously in the interface DB.
Binary data elements of the type "int(WORD)" and "Long(DWORD)" are stored in the DB in S7 format (little-endian). When the PLC accesses these data elements, it converts the data to Intel format (big-endian). Data elements which represent names are implemented as byte fields with ASCII characters.
The interface is described in tabular format. The "Access" column indicates who describes the field. The following abbreviations are used in this column:

- RPC SINUMERIK Computer link software (indirect from the host computer)
- PLC PLC user program

In order to minimize the communication load on internal resources, each change in the RPC SINUMERIK interface is relayed by means of a "request from the PLC" (part of the interface). RPC SINUMERIK sets up a hot link to this data element.

4.2 Global data

Table 4-1 Global data list

Data element	Abbreviated name	Data type	Access from	Offset
Request from PLC	ANF_PLC	byte	PLC/RPC SINUMERIK	0
Change trigger	Trigger	byte	PLC	1
Request from RPC SINUMERIK	ANF_RPC	byte	RPC SINUMERIK	2
Number of transport jobs	ANZ_TO	byte	PLC	3
Number of docking positions	ANZ_HALT	byte	PLC	4
Machine status	Machine Status	byte	PLC	5
NC mode	Machine Mode	byte	PLC	6
Machine mode	MODE	byte	PLC	7
Reserve 1	Reserve 1	Word	PLC	8-9
Reserve 2	Reserve 2	Word	PLC	10-11

Request from PLC

Table 4-2 Status list of "request from PLC"

Bit No.	Function	Access from
0	Transport job data changed	PLC - 1/RPC SINUMERIK - 0
1	Docking position data changed	PLC - 1/RPC SINUMERIK - 0
2	Status change	PLC - 1/RPC SINUMERIK - 0
3	Signaling a single position	PLC - 1/RPC SINUMERIK - 0

The PLC indicates changes in the interface with this byte. After it has enabled the request byte, the PLC must enable the next bit in the trigger byte (see below).
The PLC can only write to the request byte again when it has been set to 0 by RPC SINUMERIK after processing.

Transport job data changed

Transport job data changed is set by the PLC whenever the status of the PLC is changed during one of the transport jobs.

Docking position data changed

Docking position data changed is set by the PLC whenever data are changed by the PLC at one of the docking positions.

Status change

The PLC sets **status change** on every status change (machine mode, machine status, NC mode) that has to be reported to the host computer. RPC SINUMERIK must then send **R_TPS_H ()** to the host computer.

Signaling a single position

The docking position must be entered by the PLC in "Reserve1". The docking position is signaled to the FLR with R_TPS_H.

Change trigger

RPC SINUMERIK reacts immediately to changes in this byte.
The PLC sets a bit in this byte when changes take place on the PLC. For each new trigger, the PLC must set the next bit and reset the last; after bit 7 it begins again with bit 0.

Requests from RPC SINUMERIK

Table 4-3 Status list of "Request from RPC SINUMERIK"

Bit No.	Function	Access from
0	Synchronization flag	RPC SINUMERIK
1	Deactivate units	RPC SINUMERIK
2	Activate units	RPC SINUMERIK
3	Request write access to docking position data	RPC SINUMERIK
4	Project-specific special function 1	RPC SINUMERIK
5	Project-specific special function 2	RPC SINUMERIK

Synchronization flag

"Synchronization flag" is set and reset by the host computer. (See /NFL/ section 5.14 C_SYNCH_M ()) The status of the transport system must remain the same for the duration of the synchronization process. The PLC must not execute any pallet movements.

Deactivate components

Deactivate components is set by the host computer (See /NFL/section 5.15 Mode switching C_MODE_M ()). This requests the PLC to deactivate units (drives). See also machine mode: **Components deactivated**.

Activate components

Activate components is set by the host computer (See /NFL/section 5.15 Mode switching C_MODE_M ()). This requests the PLC to activate units (drives) again.

Write access to docking position data allowed

Write access to docking position data allowed is set by the PLC as a response to the request **Request write access to docking position data**. This coordination prevents RPC SINUMERIK from writing data to wrong docking positions due to a pallet movement.

Number of transport jobs

The number of transport jobs is stored statically during commissioning. This value specifies the maximum number of transport jobs which are transferred from the host computer to the PLC. The number matches the number of transport job data blocks in the interface.

Number of docking positions

The number of docking positions on the transport system is stored statically during commissioning. The number matches the number of docking position data blocks in the interface.

Machine status

Table 4-4 Status list of "Machine status"

Bit No.	Function	Access from
0	Machine is active	PLC
1	Machine fault detected	PLC
2	Restart machine	PLC

NC mode

Table 4-5 Status list of "NC mode"

Bit No.	Function	Access from
0	Automatic	PLC
1	MDI	PLC
2	JOG	PLC
3	TEACH IN	PLC

The machine status and the NC operating mode are signaled to the host computer with R_TPS_H. They are not evaluated in the computer link server, however.

Machine mode

Table 4-6 Status list of "Machine mode"

Bit No.	Function	Access from
0	Host computer mode	PLC
1	Components deactivated	PLC
4	Offline Host Computer	RPC SINUMERIK
6	Is registered as Machine Mode 1000 in R_TPS_H	PLC
7	Is registered as Machine Mode 2000 in R_TPS_H	PLC

Host computer mode

The host computer mode is set and reset by the PLC via a user dialog.

Components deactivated

Components deactivated is set by the PLC when this status is detected.
The request is set in bit 1 of the request flags (see below).

Reserve1, Reserve2

These variables can be used as required by the machine manufacturer by means of the PLC. The values are reported to the host computer by R_MACHINE_H (), although these values are not processed by the host computer as standard.

4.3 Transport job

Table 4-7 Data of the transport job

Data element	Abbreviated name	Data type	Access from
Source docking position	SDockIdx	Word	RPC SINUMERIK
Destination docking position	DDockIdx	Word	RPC SINUMERIK/PLC
Workpiece carrier	WPC	Byte[6]	RPC SINUMERIK
Workpiece carrier type	WPCTyp	byte	RPC SINUMERIK
For buffering	BufferFlag	byte	RPC SINUMERIK
Priority	Priority	byte	RPC SINUMERIK
Chain number	ChainNum	byte	RPC SINUMERIK
Vehicle	Vehicle	byte	RPC SINUMERIK/PLC
Transport status	TPOStatus	byte	RPC SINUMERIK/PLC

Source and destination docking position

The source and destination docking position is not the same reference as used on the host computer, but is an index. This index locates the position of the docking position within the docking position data on the transport system. The PLC must use the same index to manage the coordinates of this docking position. RPC SINUMERIK implements this feature by means of an assignment list in the Ini file.

Workpiece carrier

The name of the workpiece carrier can be used by the PLC for plausibility checks.

Workpiece carrier type

The workpiece carrier type is an additional item of data which can contain the type or the size of a workpiece carrier.

For buffering

The **buffer** flag is set in cases where a workpiece carrier is transported to a machine but where no machining is to take place (auxiliary buffer station).
The TPS must pass this information to the machine.

Priority

The **priority** is a supplementary unit of information. If several jobs are transferred, the priority can be used to control the order in which they are processed.

Chain number

The **chain number** is a supplementary unit of information. In transport vehicles with two storage locations and machines with only one docking position, two transport jobs can be associated logically by means of a chain number.

Vehicle

The **vehicle** is a supplementary unit of information. In transport systems with several transport vehicles, this parameter can be used to define which vehicle is to be used for transport.

Transport status

Table 4-8 TPOStatus

Bit No.	Function	Access from
0	New transport job	RPC SINUMERIK
1	Transport job started	PLC
2	WPC on vehicle	PLC
3	Job complete	PLC
4	Error, job not executable	PLC
5	Error, alternative destination approached (new destination is entered in DDockPos by the PLC)	PLC

When a new transport job is activated, RPC SINUMERIK enables bit 0 in TPOStatus and clears bits 1 to 7.
Bits 1 to 5 are set individually by the PLC according to the state of processing.

4.4 Docking position data of the transport system

Table 4-9 Docking position data

Data element	Abbreviated name	Data type	Access from
1. docking position status	DockPosStatus	1 bytes	PLC
1. workpiece carrier status	WPCStatus	1 bytes	PLC
1. Workpiece carrier	WPC	6 bytes	PLC
...			

The number of docking positions is configurable and determines the length required in the DB.

1st docking position status

Table 4-10 Docking position status

Bit No.	Function	Access from
0	Fault	PLC
1	Inhibited	PLC
2	Occupied	PLC

The bit array describes the actual status of the docking position. It is set by the PLC. If no bit is enabled, the docking position is available. The **fault** bit is set and canceled in response to I/O signals. The cause of the fault is reported to the host computer using the report function (see /NFL/ section 5.5 R_REPORT_H). The PLC does not execute any pallet transports between stations which have the **fault** status. If the docking position is **disabled**, it cannot be approached by the transport system.
Assigned can be set by the PLC, primarily for internal purposes. The host computer (and also RPC SINUMERIK) can detect from the workpiece carrier array whether or not the docking position is assigned.

1st workpiece carrier status

Not used.

Workpiece carrier

Name of the workpiece carrier currently located at the docking position (e.g. "WST01"). This information is entered by the PLC. If no workpiece carrier is at the docking position, the array must be filled with binary 0s.

4.5 Transport job to TPS (functional sequence)

Interface entry

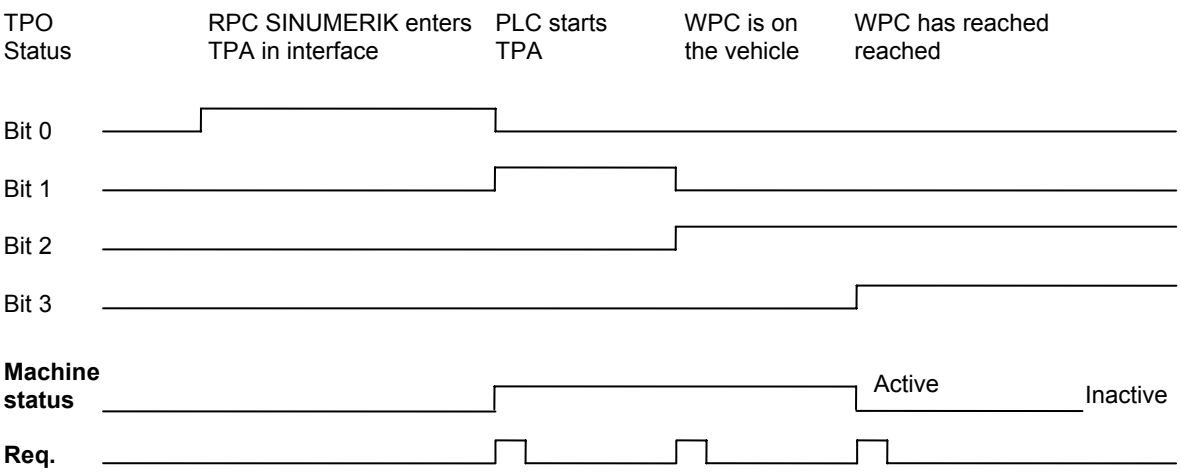
RPC SINUMERIK enters data received from the host computer in the first free data record in the interface for transport jobs. The source and destination docking positions must be referenced using an index, since the docking position name used on the host computer is not known to the PLC.

PLC actions

When the transport PLC starts the job, it must set the status TPOStatus = **job started**. Bit 0 must be enabled in the "requests from PLC" byte so that RPC SINUMERIK can read this status and inform the host computer. RPC SINUMERIK resets the request byte, reads all the report data from the interface, and sends an R_TPS_H call to the host computer.
When the workpiece carrier has been transferred to the transport vehicle, the PLC can enable TPOStatus = **WPC on vehicle** and enable bit 0 in the "requests from PLC" byte to send another R_TPS_H call to the host computer. It is not absolutely necessary to report this intermediate status. It is also possible to enable the request bit at the end of the transport operation. If it is set, another R_TPS_H call is sent to the host computer. SDockPos is transferred in DockPos[0]. The number of the transport vehicle is entered in DockPos[1].
When the workpiece carrier has been delivered to its destination docking position, the PLC must enable TPOStatus = **Job complete** and enable bit 0 again in the **requests from PLC** byte.

If the transport PLC is unable to execute a job, e.g. because the source docking position is empty or disabled, it must set TPOStatus = **error, job not executable**.
If only the specified destination cannot be reached, and the workpiece carrier is therefore diverted to another docking position, the transport PLC must enter the new destination in DDockPos and enable the status TPO-Status = **error, alternative destination approached**.
At the end of the transport job, another R_TPS_H call is initiated with MachineStatus = **inactive**, DockPos[0] = DDockPos etc.

Each time TPOStatus is changed and this modification is to be sent to the host computer with R_TPS_H (), bit 0 must be enabled in the **requests from PLC** byte.



4.6 Manual transport operations by the user at PLC level

RPC SINUMERIK can report status changes or workpiece carrier movements to the host computer in any operating mode when there is a connection to the host computer. This enables workpiece carrier movements performed manually by the user at PLC level to be reported to the host computer. For that, the PLC program must always set the source or target docking position number in the transport job interface and set TPOStatus to **Transport job started** or **Job completed**. If after that bit 0 is set in the byte request from PLC, the data are read by RPC SINUMERIK and sent to the host computer. The movement of the workpiece carrier can thus be displayed in the plant image.

■

For notes

5

5 Configuration Data

5.1 Description

The configuration data required for the computer link are entered in the Registry. These data can be created and changed with the SCCONFIG.EXE program.

Note

For compatibility reasons the registry entries have not been changed to the new name RPC SINUMERIK but can still be found under HKEY_LOCAL_MACHINE\Software\SIEMENS\SinCOM\...

The configuration data are subdivided into machine-related data and computer-related data. If several computers are connected with the computer link, there is one data record with computer-related data (name, IP address ...) for each computer. For each computer, which RPC is received by this computer and which RPC is sent to the computer has to be determined.

5.2 Configuration program SCCONFIG

Machine-related data

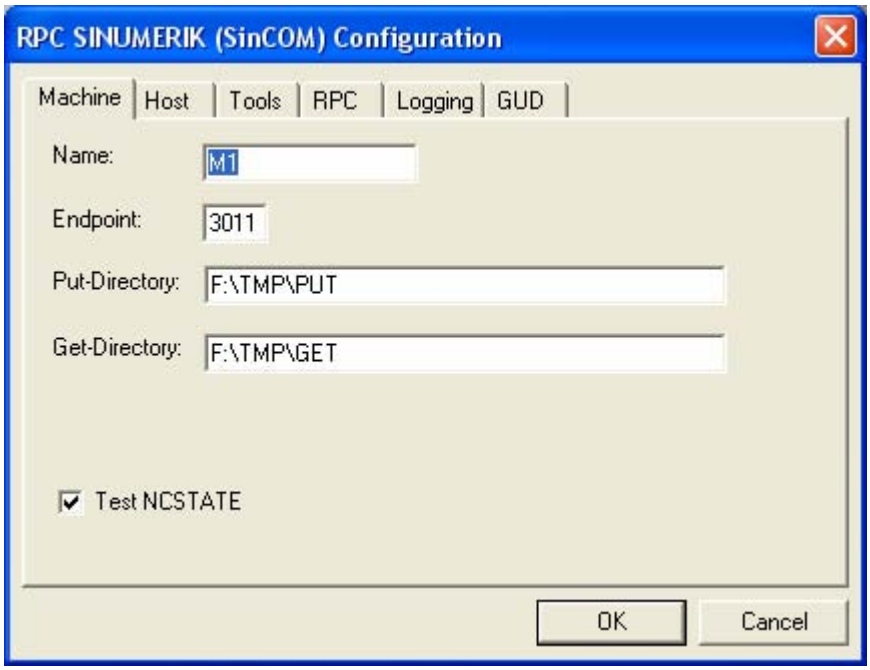


Fig. 5-1 Configuration program: Machine tab

Name

Name is the machine name which is entered as the second parameter for each RPC.

Endpoint

The **Endpoint** number is needed for RPC in addition to the IP address.

Put directory

The **Put Directory** serves as a temporary directory in which RPC SINUMERIK stores files before transferring them to the host computer.)*

Get directory

The **Get Directory** serves as a temporary directory if files are transferred from the host computer. If the files are not processed any further by RPC SINUMERIK, they remain in this directory.)*

)* Both drive designations and the UNC notation can be used.

Computer-related data

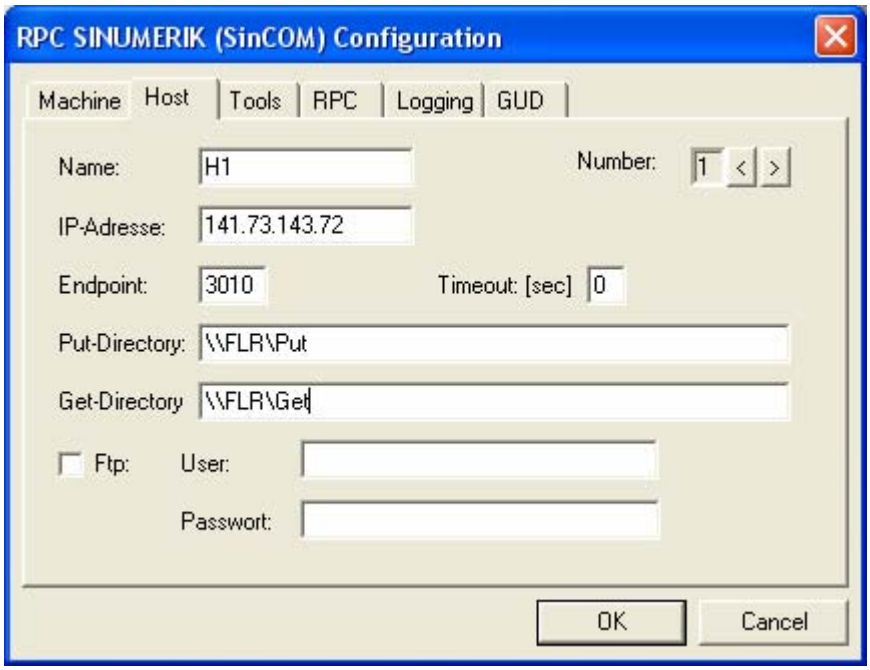


Fig. 5-2 Configuration program: Host tab

Overview

RPC SINUMERIK offers the possibility of communicating with several host computers. This screen form has to be completed for each host computer. Switchover between the host computer takes place by means of the arrows in the top right area.

Host

Name is the **host** name which is entered as first parameter for each RPC call.

IP address

IP Address is the IP address of the host computer.

Put directory

The **Put Directory** is the directory into which RPC SINUMERIK transfers files on the host computer.

Get directory

The **Get Directory** is the directory from which RPC SINUMERIK fetches files on the host computer.

Ftp

If the host computer is a Windows computer (NT or Win95), **Ftp** does not have to be used and the **User** and **Password** fields are then not relevant, because data transfer can take place via SMB (Microsoft notation).
With other host computers, e.g. UNIX or Linux computers, data transfer has to take place via FTP unless Samba, for example, is installed on the host computers, which in turn allows Microsoft notation. The **Ftp** box must be selected and valid entries for login on the host computer must be entered in **User** and **Password**.
The specified user must have write access to the directories specified with **Put Directory** and read access to the directories specified with **Get Directory**.

Directories

The specified directories on the host computer and also on the PCU 50/70 **must exist** or be created before RPC SINUMERIK is started for the first time. They are **not** created by RPC SINUMERIK!

Read tool data

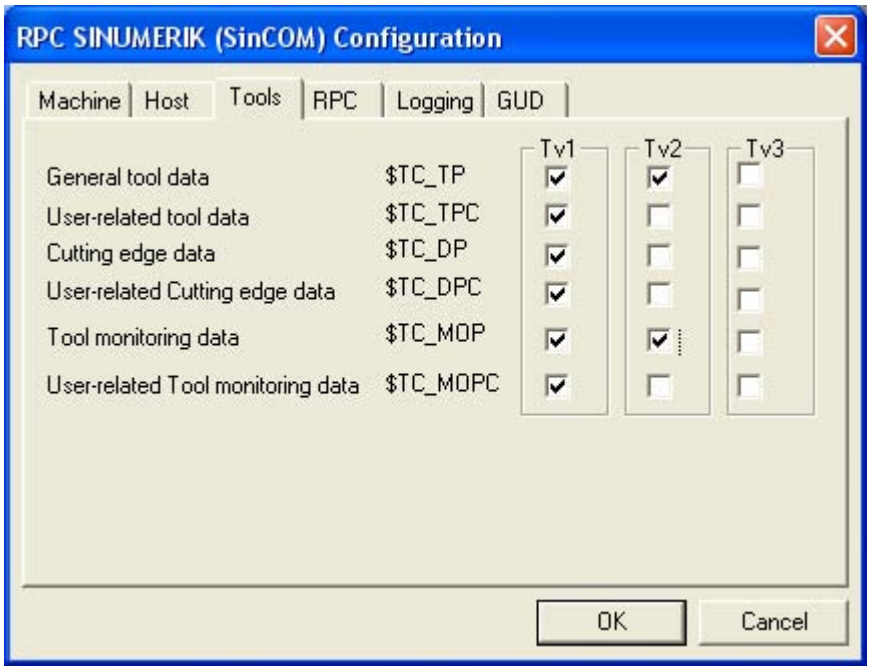


Fig. 5-3 Configuration program: Tools tab

Reading of tool data of an NC tool requires one read call per data storage area. If tool data are transferred to the host computer, it is not necessary to transfer all tool data to the host computer in each case. For that reason, three tool structures can be defined (Tv1-Tv3). Normally, the first structure includes all **existing** data storage areas (if there are no user-related data storage areas, they should be deselected). The other two structures should be defined as needed with a reduced scope.

RPC functions

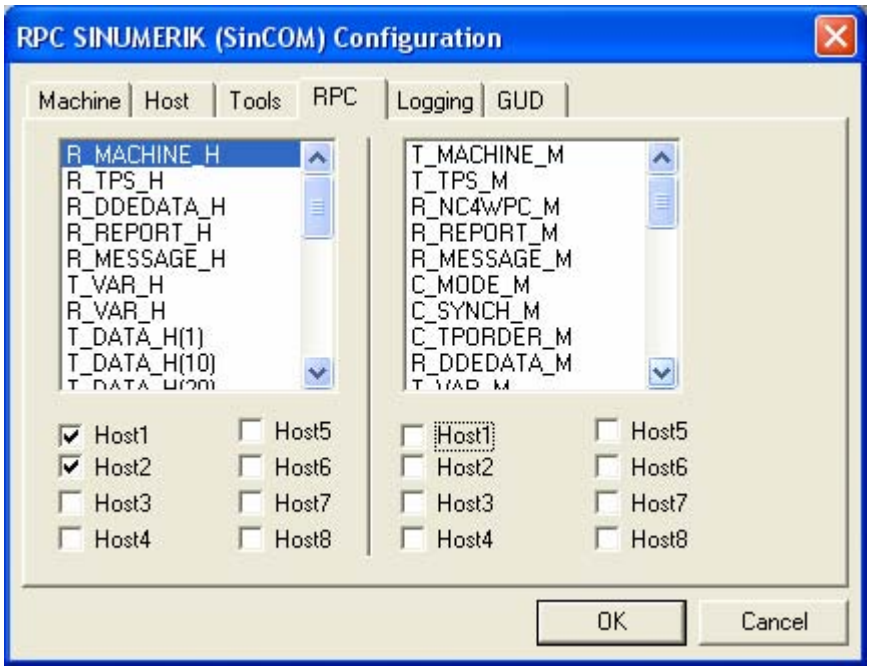


Fig. 5-4 Configuration program: RPC tab

For each RPC function the host computer to which it is sent (RPC screen, left side) and which RPC function is received from which host computer (RPC screen, right side) must be determined.

Logging

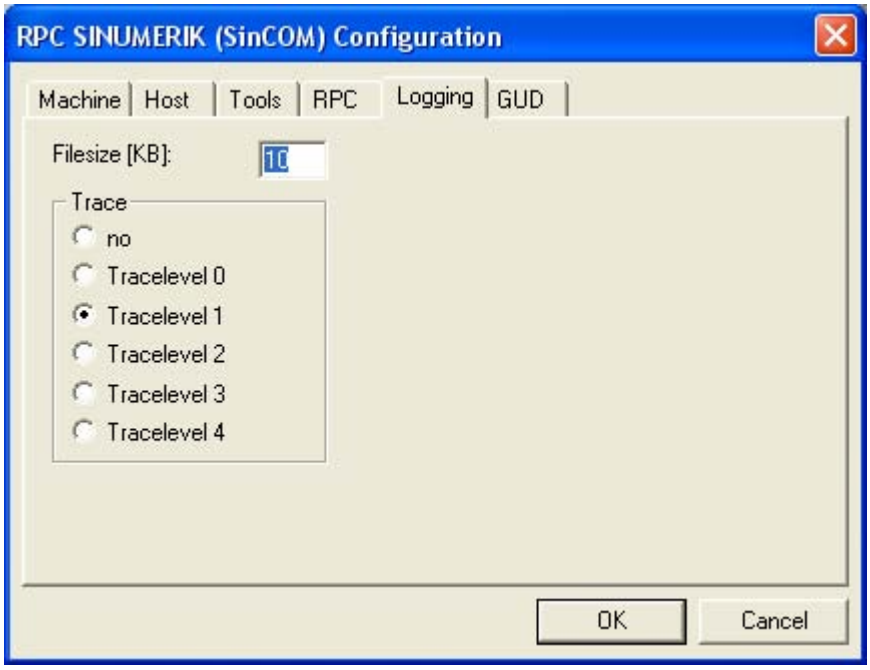


Fig. 5-5 Configuration program: Logging tab

The logging dialog box can be used to set the size of the log file and to choose a trace level.

Global user data (GUD)

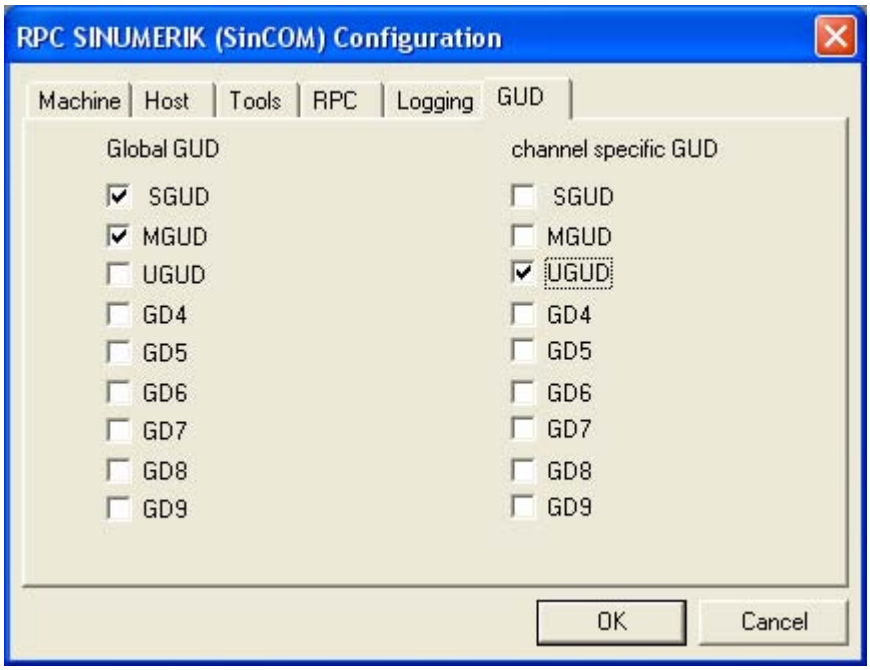


Fig. 5-6 Configuration program: GUD register

Global user data (**G**lobal **U**ser **D**ata) can be created for NCK-specific tasks on the SINUMERIK.
For more detailed information, please refer to **Appendix / 8.13 / Creating definition files**.

5.3 Example of configuration data

The ASCII file with configuration data could look like the following:

```
[HKEY_LOCAL_MACHINE\Software\SIEMENS\SinCOM\Host1]
"Name"="FLR1"
"IpAdr"="195.212.26.110"
"Endpoint"="2010"
"FTPUser"=""
"FTPPassword"=""
"HostDirGet"="h:\\"
"HostDirPut"="h:\\"
"Ftp"=dword:00000000

[HKEY_LOCAL_MACHINE \Software\SIEMENS\SinCOM\RPC_H]
"R_MACHINE_H"=dword:0000ffff
"R_TPS_H"=dword:0000ffff
"R_DDEDATA_H"=dword:0000ffff
"R_REPORT_H"=dword:0000ffff
"T_VAR_H"=dword:00000000
"R_VAR_H"=dword:00000000
"T_DATA_H(1)"=dword:00000001
"T_DATA_H(10)"=dword:00000001
"T_DATA_H(20)"=dword:00000001
"T_DATA_H(21)"=dword:00000001
"T_DATA_H(22)"=dword:00000001
"T_DATA_H(23)"=dword:00000001
"T_DATA_H(26)"=dword:00000001
"T_DATA_H(27)"=dword:00000001
"T_DATA_H(28)"=dword:00000001
"T_DATA_H(90)"=dword:00000001
"R_DATA_H(1)"=dword:000000ff
"R_DATA_H(10)"=dword:000000ff
"R_DATA_H(20)"=dword:000000ff
"R_DATA_H(21)"=dword:00000001
"R_DATA_H(22)"=dword:00000001
"R_DATA_H(23)"=dword:00000001
"R_DATA_H(50)"=dword:00000001
"R_DATA_H(90)"=dword:00000001

[HKEY_LOCAL_MACHINE \Software\SIEMENS\SinCOM\RPC_M]
"T_MACHINE_M"=dword:0000ffff
"T_TPS_M"=dword:0000ffff
"T_DATA_M"=dword:00000001
"T_REPORT_M"=dword:00000001
"R_DATA_M"=dword:00000001
"R_NC4WPC_M"=dword:00000000
"R_DDEDATA_M"=dword:0000ffff
"R_REPORT_M"=dword:0000ffff
"C_DELETE_M"=dword:00000001
"C_TPORDER_M"=dword:00000001
"R_DATA_M(1)"=dword:00000001
"R_DATA_M(10)"=dword:00000001
"R_DATA_M(26)"=dword:00000001
```

```
"R_DATA_M(27)"=dword:00000001
"R_DATA_M(28)"=dword:00000001
"R_DATA_M(90)"=dword:00000001
"C_DELETE_M(1)"=dword:00000001
"C_ORDER_M(1)"=dword:00000001

[HKEY_LOCAL_MACHINE\Software\SIEMENS\SinCOM\Settings]
"Machine"="FMS-TPS________"
"EndpointMach"="3011"
"MMCDirGet"="C:\\TMP\\MMC"
"MMCDirPut"="C:\\TMP\\MMC"
"TraceFilesize"=dword:00000005
"TraceLevel"=dword:00000002
"ToolData1"=dword:0000003f
"ToolData2"=dword:00000015
"ToolData3"=dword:00000001
```

■

FilDocEnd

For Notes

I Index

I.1 Keyword index

■

For notes

A

A Appendix

A.1 Interface Definition Language (IDL)

Notes for use

The following files are contained on the installation CD of the RPCTEST installation

- SCHOST.IDL
- SCMACH.IDL
- SCHOST.ACF
- SCMACH.ACF.

The IDL files describe the function calls and their parameters. With the ACF files for the IDL compiler, it is determined whether an internal or external binding is required. If the host computer is to communicate with a number of machines, an external binding is used.
The ACF files supplied contain the appropriate instructions for both types, only one of which however is commented on. Make sure that the required form is active and the form not required is "comment".
From these files the IDL compiler generates the client and server stubs as well as the header.
If Microsoft WINDOWS (>= WINDOWS 95) is used on the host, the VC++ development system is sufficient. It contains an IDL compiler as well as files, otherwise necessary for the RPC.

Calling the Microsoft IDL compiler:
MIDL SCHOST.IDL /osf for the RPCs on the host computer
(for example, R_MACHINE_H)
MIDL SCHMACH.IDL /osf for the RPCs on SINCOM
(for example, T_MACHINE_M)
The /osf parameter must be stated in order to be DCE compatible.

For other operating systems, a DCE RPC development system is necessary.

Important

It must be a DCE RPC development system and not a SUN RPC!

The RPCTEST installation contains the sources for a VC++ 6.0 prototype program as a basis for an RPC SINUMERIK host computer application as well as a short description of the most important program parts.

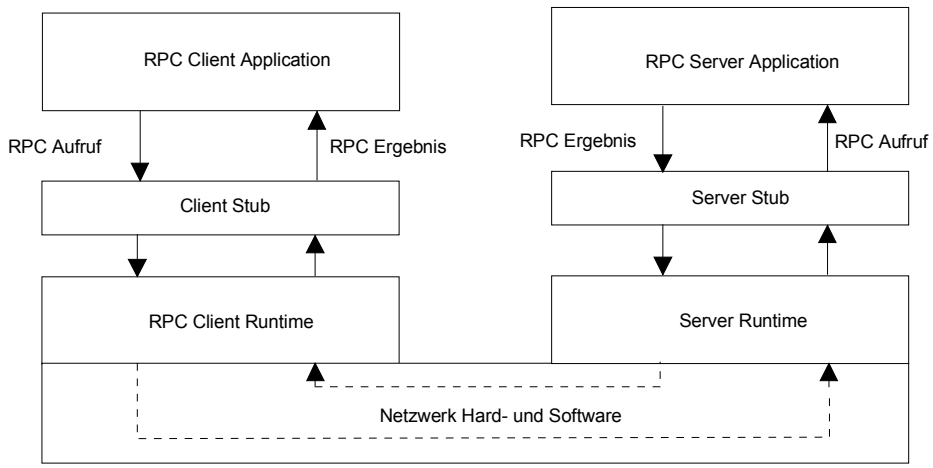


Fig. A-1 Sequence of an RPC call

A.1.1 Functions for the host computer - SCHOST.IDL

```
 [ uuid(d3d7d860-c15a-11d0-a0cb-00a0244ce687),
  version(1.0),
  pointer_default(unique)
]
interface SINCOMHOST
{

const long maxWPCLen = 6;
const long maxMPos = 3;

long R_MACHINE_H([in, string] unsigned char* pszHost,
    [in, string] unsigned char* pszMachine,
    [in] long OrderNum,
    [in] long MachineMode,
    [in] long MachineStatus,
    [in, string] unsigned char* pszNCProgram,
    [in] long ClampCubeSide,
    [in] long DockPos[maxMPos],
    [in] long DockPosStatus[maxMPos],
    [in] unsigned char pszWPC[maxMPos][maxWPCLen],
    [in] long WPCStatus[maxMPos],
    [in] long ResInt1,
    [in] long ResInt2,
    [in, string] unsigned char* pszResByte );

const long maxTPos = 2;

long R_TPS_H([in, string] unsigned char* pszHost,
    [in, string] unsigned char* pszMachine,
    [in] long OrderNum,
    [in] long MachineMode,
    [in] long MachineStatus,
    [in] long TpOStatus,
    [in] long DockPos[maxTPos],
    [in] long DockPosStatus[maxTPos],
    [in] unsigned char pszWPC[maxTPos][maxWPCLen],
    [in] long ResInt1,
    [in] long ResInt2,
    [in, string] unsigned char* pszResByte );

const long maxAlarms = 10;

long R_REPORT_H([in, string] unsigned char* pszHost,
    [in, string] unsigned char* pszMachine,
    [in] long OrderNum,
    [in] long Type,
    [in] long Number[maxAlarms],
    [in] long Time[maxAlarms],
    [in] char Flag[maxAlarms],
    [in] long ResInt1,
    [in] long ResInt2,
    [in, string] unsigned char* pszResByte );
```

```
long R_MESSAGE_H([in, string] unsigned char* pszHost,
    [in, string] unsigned char* pszMachine,
    [in] long OrderNum,
    [in, string] unsigned char* pszMessage,
    [in] long ResInt1,
    [in] long ResInt2,
    [in, string] unsigned char* pszResByte );

long T_DATA_H([in, string] unsigned char* pszHost,
    [in, string] unsigned char* pszMachine,
    [in] long OrderNum,
    [in] long SFc,
    [in, string] unsigned char* pszName1,
    [in, string] unsigned char* pszName2 );

long R_DATA_H([in, string] unsigned char* pszHost,
    [in, string] unsigned char* pszMachine,
    [in] long OrderNum,
    [in] long SFc,
    [in, string] unsigned char* pszName1,
    [in, string] unsigned char* pszName2,
    [in] long Date,
    [in] long LastFile );

long T_VAR_H([in, string] unsigned char* pszHost,
    [in, string] unsigned char* pszMachine,
    [in] long OrderNum,
    [in] long VarMode,
    [in, string] unsigned char* pszVarSet,
    [in, string] unsigned char* pszVarDescr );

long R_VAR_H([in, string] unsigned char* pszHost,
    [in, string] unsigned char* pszMachine,
    [in] long OrderNum,
    [in] long VarMode,
    [in, string] unsigned char* pszVarSet,
    [in, string] unsigned char* pszVarDescr,
    [in, string] unsigned char* pszVarData );

long R_DDEDATA_H([in, string] unsigned char* pszHost,
    [in, string] unsigned char* pszMachine,
    [in] long OrderNum,
    [in, string] unsigned char* pszData );

void Shutdown_H(void);

}
```

A.1.2 Functions for SINUMERIK - SCMACH.IDL

```
[ uuid(d6542300-c15a-11d0-a0cb-00a0244ce687),
  version(1.0),
  pointer_default(unique)
]
interface SINCOMMACHINE
{

long T_MACHINE_M([in, string] unsigned char* pszHost,
    [in, string] unsigned char* pszMachine,
    [in] long OrderNum);

long T_TPS_M([in, string] unsigned char* pszHost,
    [in, string] unsigned char* pszMachine,
    [in] long OrderNum);

long T_DATA_M([in, string] unsigned char* pszHost,
    [in, string] unsigned char* pszMachine,
    [in] long OrderNum,
    [in] long SFc,
    [in, string] unsigned char* pszName1,
    [in, string] unsigned char* pszName2 );

long T_VAR_M([in, string] unsigned char* pszHost,
    [in, string] unsigned char* pszMachine,
    [in] long OrderNum,
    [in] long VarMode,
    [in, string] unsigned char* pszVarSet,
    [in, string] unsigned char* pszVarDescr );

long R_NC4WPC_M([in, string] unsigned char* pszHost,
    [in, string] unsigned char* pszMachine,
    [in] long OrderNum,
    [in, string] unsigned char* pszWPC,
    [in, string] unsigned char* pszNCProg,
    [in] long Date,
    [in] long NCPLength,
    [in] long ClampCubeSide,
    [in] long TpFlag,
    [in] long NCExtern,
    [in] long ResInt1,
    [in] long ResInt2,
    [in, string] unsigned char* pszResByte );
```

```
long R_REPORT_M([in, string] unsigned char* pszHost,
    [in, string] unsigned char* pszMachine,
    [in] long OrderNum,
    [in] long Type,
    [in] long Number,
    [in] long ResInt1,
    [in] long ResInt2,
    [in, string] unsigned char* pszResByte );

long R_MESSAGE_M([in, string] unsigned char* pszHost,
    [in, string] unsigned char* pszMachine,
    [in] long OrderNum,
    [in, string] unsigned char* pszMessage,
    [in] long ResInt1,
    [in] long ResInt2,
    [in, string] unsigned char* pszResByte );

long R_DATA_M([in, string] unsigned char* pszHost,
    [in, string] unsigned char* pszMachine,
    [in] long OrderNum,
    [in] long SFc,
    [in, string] unsigned char* pszName1,
    [in, string] unsigned char* pszName2,
    [in] long Date,
    [in] long LastFile );

long R_VAR_M([in, string] unsigned char* pszHost,
    [in, string] unsigned char* pszMachine,
    [in] long OrderNum,
    [in] long VarMode,
    [in, string] unsigned char* pszVarSet,
    [in, string] unsigned char* pszVarDescr,
    [in, string] unsigned char* pszVarData );

long R_DDEDATA_M([in, string] unsigned char* pszHost,
    [in, string] unsigned char* pszMachine,
    [in] long OrderNum,
    [in, string] unsigned char* pszApplication,
    [in, string] unsigned char* pszTopic,
    [in, string] unsigned char* pszItem,
    [in, string] unsigned char* pszData );

long C_DELETE_M([in, string] unsigned char* pszHost,
    [in, string] unsigned char* pszMachine,
    [in] long OrderNum,
    [in] long SFc,
    [in, string] unsigned char* pszName1,
    [in, string] unsigned char* pszName2      );

long C_MODE_M([in, string] unsigned char* pszHost,
    [in, string] unsigned char* pszMachine,
    [in] long OrderNum,
    [in] long Mode    );
```

```
long C_SYNCH_M([in, string] unsigned char* pszHost,
    [in, string] unsigned char* pszMachine,
    [in] long OrderNum,
    [in] long SynchFlag);

long C_TPORDER_M([in, string] unsigned char* pszHost,
    [in, string] unsigned char* pszMachine,
    [in] long OrderNum,
    [in] long SDockPos,
    [in] long DDockPos,
    [in, string] unsigned char* pszWPC,
    [in] long WPCType,
    [in] long BufferFlag,
    [in] long Priority,
    [in] long ChainNum,
    [in] long Vehicle,
    [in] long ResInt1,
    [in] long ResInt2,
    [in, string] unsigned char* pszResByte );

void Shutdown_M(void);

}
```

A.2 HMI <=> NCK/PLC (OEM) interface

Note

The following subchapter has been taken unaltered from the HMI <=> NCK/PLC (OEM) interface documentation.

8

HMI <=> NCK/PLC (OEM) interface

Overview

There are three different communication services available to the developer:

Variable service	Accessing NC, PLC and drive data	
	Via	**OPC-DataAccess**
	Or	**DCTL-Control**
Domain service	Copying files between HMI and NCK:	
	File access via data management	
	With	**IMC-File**
	And	**IADS**
	Or	**FileViewer-Control**
PI service:	Starting program-invocation services of the NC	
	With	**IMC-Command**

8.1 General

Communication between applications and NC/PLC takes place via the OPC or the Sinumerik COM interfaces. For compatibility reasons, direct access via the NCDDE server is also possible for the time-being. With new developments, however, the new COM interfaces must be front panel.

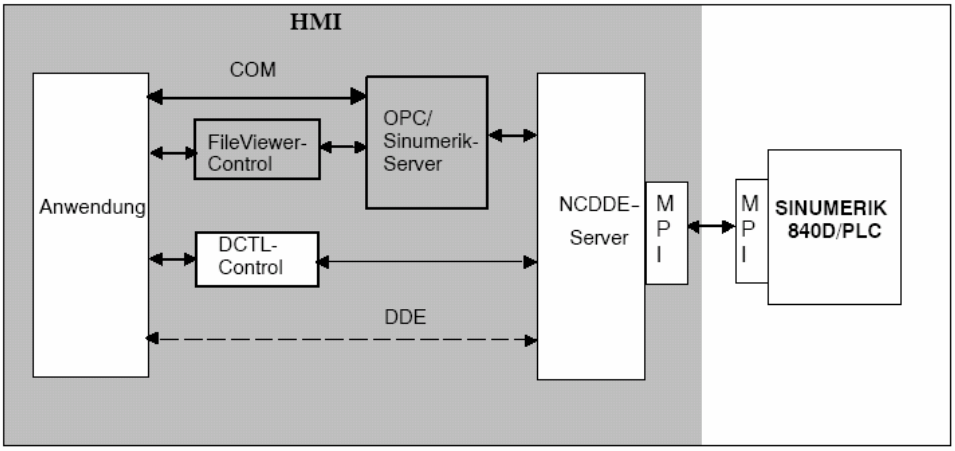


Fig. 8-1 Overview

The NCDDE server can be configured by means of initialization files. This allows the user to adapt the NCDDE to his development environment.
Is there a control available for testing purposes? Are there one or more NCUs from which data is to be accessed?

Note

The use of the Windows environment means that there is no time guarantee for SINUMERIK 840D. Real-time tasks can therefore not be implemented in HMI. They must be implemented in the NCU area with the OEM package NCK.

8.2 Basics of DDE

Overview

The WINDOWS operating system provides application developers with the DDE (Dynamic Data Exchange) mechanism as a communication tool between WINDOWS processes.

DDE features

DDE characterizes the dynamic data exchange under WINDOWS with the following Properties:

- DDE is communication between WINDOWS applications
- DDE is executed between two processes according to the client-server model
- One process acts as client: it requests data from the server
- One process acts as server: it provides the data for the client
- The connection is established by the client
- A program may act as server as well as client
- The communication is specified according to the internal WINDOWS protocol

Establishing a DDE connection

In order to establish a connection to a DDE server, the client developer must know the following information:

- Link Server Name of the DDE server
- Link Topic Your Area of Interest
- Link Item The data item to be accessed
- Link Mode Phone line type

DDE Link Mode

- Request The client requests the data only once
- Warmlink The server informs the client if the data has changed. The client can then retrieve the data
- Hotlink If the data changes, the server automatically sends the current data value to the client
- Poke The client instructs the server to write a data item
- Execute The client instructs the server to execute a command

8.3 Configuring the NCDDE server

8.3.1 The MMC.INI initialization file

Description

The NCDDE server is initialized via the [GLOBAL] section of the "MMC.INI" file. This file is located in the directory \MMC2 of the OEM system.
Here the Link Server and the Link Topic to which the local NCDDE server must connect are defined. The terms Link Server and Link Topic are explained in section 8.2.
Depending on the scope of the installation, the NCDDE server can have one of four basic configurations:

- Connection to one NC
- Connection to one or more NCs
  (for the M:N feature see section 8.3.3)
- Local operation mode on a PC
  Allows the developer to test his application locally on his PC without a connection to an NC. In this case the NCDDE server supplies substitute values which can be defined with the command "NEW" (section 8.8) and modified with the command "ANIMATE" (section 8.8) to simulate an active NC.
- Local operation on a PC with NC simulator.
  Allows the developer to test his application locally on a PC without a connection to an NC. The NC simulator can be used to reproduce a behavior resembling that of an NC.

NcddeServiceName

DDE Link Service name of the NCDDE server. The default name is always "ncdde".

Note

All examples in chapter 8 are based on "NcddeServiceName=ncdde". If the name is different, these examples must be modified accordingly, otherwise they will not work.

Ncdde-MachineName

The NCU name for the standard applications is entered here. If "MachineSwitch" is entered here, you can switch between individual NCUs (for the M:N feature see section 8.3.3).

NcddeDefault-MachineName

This initializes the M:N feature, i.e. a connection is established to this NCU when the HMI starts up.

Ncdde-MachineNames

Here the names of the NCUs that can be connected are entered. For each NCU name entered here a section of the same name **must** exist in the file MMC.INI.

NcddeStartupFile

The NSK file (section 8.3.2) to be loaded when starting up the NCDDE server.

NcddeMachineNamesAdd1

Identifier for an installed NC simulation. If no simulation has been installed, this entry has no relevance.

In the following example the parameterization of the MMC.INI file is illustrated by installation on a PC with no NC and no simulation.

Example 8-1 Extract from the MMC.INI file

```
[GLOBAL]
; for using M:N function set NcddeMachineName=MachineSwitch
; for working without NC set NcddeMachineName=local
; for working with SIMNC set NcddeMachineName=SIM1
; for connecting to a NC set NcddeMachineName=NCU840D
NcddeMachineName=local
; for using M:N function set NcddeDefaultMachineName=net:NCU_1
; for working without NC set NcddeDefaultMachineName=local
; for working with SIMNC set NcddeDefaultMachineName=SIM1
; for connecting to a NC set NcddeDefaultMachineName=NCU840D
NcddeDefaultMachineName=local
; for using M:N function set NcddeMachineNames=net,NCU840D
; for working without NC set NcddeMachineNames=
; for working with SIMNC set NcddeMachineNames=SIM1
; for connecting to a NC set NcddeMachineNames=NCU840D
NcddeMachineNames=
; for using M:N function set NcddeStartupFile=ncdde5.nsk
; for working without NC set NcddeStartupFile=ncdde202.nsk
; for working with SIMNC set NcddeStartupFile=sim1dde5.nsk
; for connecting to a NC set NcddeStartupFile=ncdde5.nsk
NcddeStartupFile=ncdde202.nsk
```

Name spaces

In LOCAL mode the NCDDE server does not differentiate between variables in 'name spaces'. Name space refers to a differentiation by TOPIC.
For example, if a variable is applied to TOPIC LOCAL and the same variable to TOPIC Sim0, NCDDE does not differentiate between these variables.
This can mean that the current record display stops working in the simulation if you switch to MACHINE in a screen with the current record display, since local variables are applied there from the program and they supersede the 'simulation variables'.

8.3.2 Command files on the NCDDE server

NSK files

The command files (with the extension .NSK) contain for example the Link items to which the NCDDE connections refer. These files can contain commands, which are described in section 8.8.
In these files the Data Link items that can be accessed are programmed.
Other .nsk files can also be included by means of CALL front panel. This allows structuring. Example 8-2 shows a Link item (LastError) and the structuring of the global variables for HMI with CALL instructions.

Note

You may load your own NSK files with CALL instructions. The NSK files can be created with a MAP function (section 8.6.3).

Example 8-2 File NCDDE311.NSK in directory \mmc2

```
REM NSK ROOT FOR 840D
============================================================
REM
REM WRITE-ACCESS FOR NC-BUSADDRESS
LINK("/Nck/Nck/busAddress",200,"7 31 0 0 E0# /NC 1 0 11",10)
LINK(”/Nck/Nck/busState”,300,””,0);
REM
REM ACCESS TO CONNECTION ERROR STATE
LINK(”LastError”,1,””,0);
REM
REM IMPORT 840D BASIC NC VARIABLES
CALL(nc311.nsk)
REM
REM IMPORT 840D BASIC PLC VARIABLES
CALL(plc311.nsk)
REM
REM IMPORT ADDITIONAL LINK VARIABLES
CALL(add311.nsk)
REM
REM IMPORT COMIC STARTS
CALL(comic.nsk)
REM
```

8.3.3 Connecting several NCs

M:N feature

This feature allows more than one HMI to be connected to more than one NCU.
For example, data from two NCUs can be accessed from one HMI.
The file "NETNAMES.INI" is interpreted in this basic configuration.

Connection part

The section "[conn MMC_1]" specifies the partners to which the HMI can connect.

Network parameters

In the section "[param network]" the transfer rate is set:
OPI 1.5 MBit
MPI 187.5 Kbit

Bus devices

In the sections "[param NCU_n]" the bus addresses of the NC and the PLC as well as the NCU name are set. The HMI must use these names to address the NCU.
A description must be given for each NCU.

Example 8-3 The NETNAMES.INI file

```
; owner TECHNICAL reference to the bus description
; computer-specific
[own]
owner = MMC_1
; Description of possible connections
[conn MMC_1]
conn_1=     NCU_1
conn_2=     NCU_2
; Description of significant net parameters
; opi =1.5 Mbit
; mpi =187.5 Kbit
[param network]
bus=        opi
; Bus addresses for all bus nodes
[param MMC_1]
mmc_address = 1
[param NCU_1]
nck_address = 10
plc_address = 10
name=Standard_Machine
[param NCU_2]
nck_address = 11
plc_address = 11
name=Test_Machine
```

Application

In order for OEM applications to function correctly in an M:N configuration, the following points must be observed:

- For any communication with the NC, only "machineswitch" should be used as the Link Topic. This ensures that the application always communicates with the NC that is selected if the M:N or NC is changed.
  If on the other hand the name of a specific NCU is given when establishing a communication connection in Link Topic, e.g. NCU1, it is disregarded if the M:N is changed, i.e. if the operator panel is switched from NCU1 to another NC, the connection is maintained. Such "fixed" or static connections to a specific NC should only be set up by an application if the operator panel on which the application is running is configured as the M:N server.
- For the OEM application an NC switch appears as an NC reset or communication failure. The application should therefore also respond in the same way as after an NC reset or communication failure. An NC reset can be detected by the establishing of a hotlink to the OPI variable /bag/state/opmode. In the event of an NCK reset the hotlink supplies the value "#".
- To prevent communication operations, e.g. file transfers, that are running in the background from being interrupted or aborted by an M:N or NC switch, NC switching should be disabled before the start of such operations and enabled again on completion. The functions LockCurrentNCU/LockChanMenu and UnlockCurrentNCU/UnlockChanMenu (see section 6.4) can be used for this. LockCurrentNCU only disables switching to another NC, whilst allowing channel switching on the current NC, whereas LockChanMenu disables channel switching too.
- If certain services or variables are only available on a specific NCU, before the application accesses these services or variables you must check that a connection exists to the NCU in question. The NCU to which a connection currently exists can be determined by reading the "machineswitch" variable (LinkItem).

8.4 Establishing a DDE connection

Overview

This subchapter shows how to establish a DDE connection to the NCDDE server with Visual Basic and with Visual C++.

Note

In the examples below only the standard Visual Basic control "LABEL" is used for DDE communication. An OEM application should use the Siemens control DCTL (see section 8.9.3) for DDE communication, however.

The following requirements must be fulfilled for the examples below to work:

Development environment

- We recommend MS Visual Basic 4.0_16
- In order to test the examples on the SINUMERIK 840D directly from a PC, an MPI connection is required and the NCDDE server must be configured for NC operation.
  If the NCDDE server is operated without SINUMERIK 840D, not all of the data can be accessed.
- The NCDDE server (C:\MMC2\NCDDE.EXE) must have been started (e.g. via Explorer or the Start menu).

8.4.1 Establishing a DDE connection with Visual Basic

For the standard Visual Basic controls that are DDE enabled, e.g.

- Label
- Textbox
- Picture

Link Service and Link Topic are combined in the property (attribute) "LinkTopic". They are separated by the pipe symbol "|" (e.g. LinkTopic "ncdde|local").

Reading variables only once

The following example reads the actual position of the first axis in the first channel of the workpiece coordinate system **once**. In order for the following example to work, the NCDDE server must be configured for local operation and **"NcddeServiceName = ncdde"**. In other words the **NCK** is not accessed here. For one-off reading, **LinkMode** must be set to **2**.

Note

For one-off reading, "LinkMode" must be set to **2**. The value is then also requested by the first channel with the "LinkRequest" method.

Example 8-4 One-off reading of variables

```
Sub Form_Load ()
        Label1.LinkTopic = "ncdde|local"
        Label1.LinkItem=”/Channel/GeometricAxis/actToolBase-
        Pos[u1,1]"
        Label1.LinkMode = 2
        Label1.LinkRequest
End Sub
```

Note

If the channel identifier "u1" is not specified, the first channel is accessed automatically.

Updating when changed

The following example automatically updates (hotlink) the actual position of the third axis in the second channel of the machine coordinate system in the "label1", i.e. the current actual position of this axis is displayed.

Note

For hotlinks "LinkMode" must be set to 1.

Example 8–5 Updating when changed

```
Sub Form_Load ()
        Label1.LinkTopic = "ncdde|ncu840d"
        Label1.LinkItem="/Channel/MachineAxis/actToolBasePos[u2,3]"
        Label1.LinkMode = 1 ’Hotlink
End Sub
```

Notifying when changed

In this example the NCDDE server notifies the application/client if the first PLC input byte changes (Warmlink). The "Sub LinkNotify" of Label1 then runs automatically. The user must then also call a "LinkRequest" in order to get the data. This means that you can check or convert the data, for example, before they are displayed.

Note

For notifying when changed (Warmlink) "LinkMode" must be set to 3.

Example 8–6 Notifying when changed

```
Sub Form_Load ()
        Label1.LinkTopic = "ncdde|ncu840D"
        Label1.LinkItem = "/PLC/Input/Byte[1]" Label1.LinkMode = 3
End Sub
Sub Label1_LinkNotify ()
        Label1.LinkRequest
End Sub
```

Writing NC data

In this example the client **writes** the value **"4"** to the first R parameter R[1] of the first channel.

Note

When writing data (Poke), "LinkMode" must be set to 2. The value is written with LinkPoke.

Example 8–7 Writing NC data

```
Sub Form_Load ()
        Label1.LinkTopic = "ncdde|ncu840d"
        Label1.LinkItem = "/Channel/Parameter/R[1]"
        Label1.LinkMode = 2 'Manual
        Label1.Caption = "4"
        Label1.LinkPoke
End Sub
```

Writing PLC data

In this example the client writes the value "250" to flag byte 5 of the PLC.

Example 8–8 Writing PLC data

```
Sub Form_Load ()
        Label1.LinkTopic = "ncdde|ncu840d"
        Label1.LinkItem = "/PLC/Memory/Byte[5]"
        Label1.LinkMode = 2 'Manual
        Label1.Caption = "250"
        Label1.LinkPoke
End Sub
```

Executing a command

For executing commands described in sections 8.6.1, 8.6.3, 8.7 and 8.8.
The following example starts a file transfer of the "test.mpf" file from the HMI to the NCK.

Note

When executing commands (Execute), "LinkMode" must be set to 2. The command is executed with LinkExecute.

Example 8–9 Executing a command

```
Sub Form_Load ()
        Label1.LinkTopic = "ncdde|ncu840d"
        Label1.LinkMode = 2
        Label1.LinkExecute "COPY_TO_NC(""C:\NC\test.mpf"",
        /NC/_N_MPF_DIR/N_TEST_MPF, trans)"
End Sub
```

8.4.2 Establishing a DDE connection with Visual C/C++

Overview

C/C++ can make use of all the features of the DDE interface. In particular, it allows asynchronous calls to be addressed to the DDE interface. This is also possible in Visual Basic using OEM Visual Basic controls such as DCTL.

Note

DDE with C/C++ is recommended only for OEM users who are familiar with C programming under WINDOWS and who require little or no integration in the sequence control of the OEM package.

DDE access with C/C++

This example shows how to establish a Hotlink (Advise) connection (with acknowledgment) between

- C/C++ program
- Variable "/Channel/GeometricAxis/toolBaseDistToGo[1]"
- NcddeServiceName = ncdde
- NcddeMachineName = local

The changes of a variable's value are received in a XTYP_ADVDATA transaction by a call back routine which has been reported to the DDEML.

Example 8–10 Hotlink at C level

```
DWORD       idInst; // created with DdeInitialize
HSZ         hszService, hszTopic, hszItem; // String Handles
HCONV       hConv; // Conversation Handle

HszService  = DdeCreateStringHandle ( idInst , "ncdde" , ZERO );
hszTopic    = DdeCreateStringHandle ( idInst , "local" , ZERO );
hszItem     = DdeCreateStringHandle ( idInst ,
              Ä"/Channel/GeometricAxis/toolBaseDistToGo[1]" , ZERO );
hConv       = DdeConnect(idInst,hszService,hszTopic,ZERO);
                    // Connection to server
                    // Hotlink follows
if ( DdeClientTransaction ( (LPBYTE)ZERO , 0 , hConv , hszItem ,
              ÄCF_TEXT ,XTYP_ADVSTART|XTYP_ACKREQ , 1000 , NULL )
              Ä==TRUE) { } // Hotlink connection successful
```

8.4.3 Establishing a DDE connection from MS Excel

Overview

In Excel Advise (Hotlink) connections can be established to the NCDDE server's interface for variables using cell formulas.

Syntax in an EXCEL cell

Within a cell
= NcddeServiceName|NcddeMachineName!Variables

Displaying a PLC bit in Excel

The following example shows an Advise (Hotlink) connection from a cell in Excel (German version) to the 3rd bit in byte 9 of data block 100.
The variable name is: "/Plc/DataBlock/Bit[c100,9.3]"
for an "ncu840D" machine connected via the NCDDE server.

Example 8–11 Displaying a PLC bit in MS Excel

	A			**A**
1	=ncdde|ncu840D!'/Plc/DataBlock/Bit[c100,9.3]'		1	1

On the left you see the cell formula, on the right the resulting, continuously updated display.

8.5 Variable service

Overview

NC variables are accessed via **OPC data** or via the **DCTL control**.
For compatibility reasons the variables can also be accessed via DDE.
The variable services provided by the NCDDE server allow two types of data access:

- Single variable access
- Array variable access

In the Link item variables can be specified by means of an additional data format and also by array range if necessary. This allows the user to request data from the NCDDE server in such a way that in most cases no further conversion is necessary.

Note

A complete description of the variables that can be accessed can be found in **Chapter 11** Reference or in the online help for variables.

Formats of NCDDE variables

Formatting instructions for NCDDE variables are included at the end of the LinkItem. Internal data preparation allows formatting of integers, floating numbers and text.
An extended ‘printf’ format in the C programming language is used to specify the format. The syntax of an NCDDE format instruction is:

Format:	""	<Params>	<‘printf’ format>
Params:	'!' 'b'	<Params>	// conversion to a bit string
	'!' 'd'	<Params>	// d for double as 64 bit floating
	'!' 'l'	<Params>	// l for long as 32 bit integer //value
	'!' 't'	<Params>	// t for text as string
	'!' '#'	<Params>	// # , index for variable //access as 32 bit integer

The data type for the corresponding DDE variables can be found in Chapter 11 or in the "NCDDE variables help".

Note

If the types of the data selection and the variable actually read do not match, there is no automatic adjustment of the data format, i.e. the wrong data are displayed.

Formatting a number value

Here the actual position of the **second** axis is read from the NC and is displayed with maximum 11 digits before and three after the decimal point. With no formatting instructions only three digits are obtained after the decimal point.

Example 8–12 Formatting with max. 11 digits before and a fixed 3 digits after the decimal point

```
Sub Form_Load ()
        Label1.LinkTopic = "ncdde|ncu840d"
        Label1.LinkItem = "/Channel/MachineAxis/actToolBasePos[2]
                        Ä(""!d%11.3lf"")"
        Label1.LinkMode = 2 'Manual
        Label1.LinkRequest
End Sub
```

Converting to hexadecimal format

In this example flag byte **5** is read and then displayed as a two-digit hexadecimal number with leading **zeros**.

Example 8–13 Converting to hexadecimal format

```
Sub Form_Load ()
        Label1.LinkTopic = "ncdde|ncu840d"
        Label1.LinkItem = "/PLC/Memory/Byte[5] (""!I%02lx"")"
        Label1.LinkMode = 2 'Manual
        Label1.LinkRequest
End Sub
```

Converting to bit string

In this example flag byte **5** is read and then output as a 32-bit string.

Example 8–14 Converting to bit string

```
Sub Form_Load ()
        Label1.LinkTopic = "ncdde|ncu840d"
        Label1.LinkItem = "/PLC/Memory/Word[5] (""!b%16.16s"")"
        Label1.LinkMode = 2 'Manual
        Label1.LinkRequest
End Sub
```

Result: 10101010101010101

Reading a string from the PLC

In this example **10** bytes are read from the data module **81** starting from byte **20** and then are displayed as a string with terminating zeros

Example 8–15 Reading a string from the PLC

```
Sub Form_Load ()
        Label1.LinkTopic = "ncdde|ncu840d"
        Label1.LinkItem = "/PLC/DataBlock/Byte[c81,20,#10] (""!l%lc"")"
        Label1.LinkMode = 2 'Manual
        Label1.LinkRequest
End Sub
```

Result: e.g. hello

8.5.1 Single variable access

If you are working within the sequence control (see chapter 7), it is advantageous to use the global variable **"g_chNCDDEServiceName"** as LinkTopic. It always contains the **NCDDEServiceName** and the **NcddeMachineName** as they have been entered in the file MMC.INI. They are separated by the pipe symbol ("|").

Single access to three variables

Reading the names of the first three geometry axes

Example 8–16 Single access to three variables

```
Sub Form_Load
        achsname(0).LinkTopic = g_chNCDDEServiceName
        achsname(0).LinkItem = "/Channel/MachineAxis/name[1]"
        achsname(0).LinkMode = 2
        achsname(0).LinkRequest
        achsname(1).LinkTopic = g_chNCDDEServiceName
        achsname(1).LinkItem = "/Channel/MachineAxis/name[2]"
        achsname(1).LinkMode = 2
        achsname(1).LinkRequest
        achsname(2).LinkTopic = g_chNCDDEServiceName
        achsname(2).LinkItem = "/Channel/MachineAxis/name[3]"
        achsname(2).LinkMode = 2
        achsname(2).LinkRequest
End Sub
```

PLC bit access

With the following Link Item bit 4 of the input byte 2 can be accessed.
/Plc/Input/Bit[2.4]

PLC byte access

With the following Link Item the output byte 4 can be accessed.
/Plc/Output/Byte[4]

PLC word access

With the following Link Item the register word 4 can be accessed.
/Plc/Memory/Word[8]
Access to other variables is described in section 12.1.5.

8.5.2 Array variable access

Application

An array access is advantageous when several data are to be read from one area. This reduces the computing load on the NCDDE server and improves the performance of the user's own applications. Example 8-16 is a bad example of this.

Note

Array accesses speed up the data access and as well the speed of the complete system, since the time needed for communication is considerably reduced.

Syntax

As an introduction here is a brief glimpse of the syntax for array areas:
Variable name[c, u, StartIndex, [EndIndex]]

Parameter

Table 8-1 Data access parameters

Name	Description
Variable name	Name of the NCK/PLC variable (see Chapter 11)
c	When accessing **NCK** variables (see Chapter 11): Column index c stands for column and applies to multi-dimensional arrays only. When accessing **PLC** data blocks c characterizes the **data block** to be accessed.
e	Unit index for NCK variables **only** (e.g. channel): u stands for unit
StartIndex	Index for the variables to be read. When accessing arrays this is the first of the values to be read.
EndIndex (optional)	For accessing **arrays** only: Specifies the number of values to be read.

Accessing an axis name array

The following example reads the first three axis names from the NCK. The result is a string which contains these axis names e.g. X1,Y1,Z1 in the format "X1Y1Z1". Using the Visual Basic functions "Trim$" and "Mid$" the axis names can be extracted from this result string.

Example 8–17 Accessing an axis name array

```
m_a_namen.LinkTopic = g_chNCDDEServiceName
m_a_namen.LinkItem = "/Channel/MachineAxis/name[u1,1,3]"
m_a_namen.LinkMode = 2
m_a_namen.LinkRequest

'Extracting single names from the text array

achsname1.Caption = Trim$(Mid$(m_a_namen.Caption,1,2))
achsname2.Caption = Trim$(Mid$(m_a_namen.Caption,4,2))
achsname3.Caption = Trim$(Mid$(m_a_namen.Caption,7,2))
```

Accessing an axis name array

In the following example the axis names of two axes in the second channel are read, starting with axis 3. The names of axes three and four are read. Apart from the following line it is the same as example 8-16

Example 8–18 Accessing an axis name array

```
...
LinkItem = "/channel/machineaxis/name[u2,3,4]"
...
```

Array access to PLC data

This example reads the three bytes byte **2** to byte **4** of DB **8** from the PLC: These are bytes 2,3,4 as two-digit hexadecimal values. The bytes are then extracted using the Visual Basic functions "Trim$" and "Mid$".

Example 8–19 Array access to PLC data

```
Label1.LinkTopic = "ncdde|ncu840d"
Label1.LinkItem = "/PLC/Datablock/Byte[c8,2,4]( ""!I%02lx"")"
Label1.LinkMode = 1 'hotlink

'Extracting single bytes in hexadecimal Strings

byte_1 = Trim$(Mid$(Label1.Caption,1,2))
byte_2 = Trim$(Mid$(Label1.Caption,3,2))
byte_3 = Trim$(Mid$(Label1.Caption,5,2))
```

PLC access with specification of number

The following example reads **5** words of DB **8** starting from word **2** as a four-digit hexadecimal number from the PLC. The words are separated with "_".

Example 8–20 Array access to PLC data

```
Label1.LinkItem = "/PLC/Datablock/Word[c8,2,#5](""!I%04lx_"")"
```

Array access to R parameters

This example writes the following values to the three R parameters R3, R4 and R5:
R3 = 2.2
R4 = 3.5
R5 = 4.9.

Example 8–21 Array access to R parameters

```
Label1.LinkTopic = "ncdde|ncu840d"
Label1.LinkItem = "/CHANNEL/PARAMETER/R[U1,3,5]"
Label1.LinkMode = 2 ’Manual
Label1.Caption = ":2.2:3.5:4.9"
Label1.LinkPoke
```

8.6 File transfer services (domain services)

Overview

File transfer services can be used to transfer files between the areas (domains) HMI and NCK/PLC.
IMC File, **IADSI** and **IMC Domain** can be used for transfers between HMI and NCK/PLC. For compatibility reasons, data transfer can also be performed via DDE.
A total of five commands are available, which are summarized in Table 8-2.
The file transfer is executed in the background.
From software release 3.3 onwards, extended copy functions between the individual areas can be used. These are especially suitable for editing programs in the NC. The new functions are described in more detail in section 9.6.2.

Table 8-2 Data access parameters

Name	Description
COPY_FROM_NC	Transfer from NCK to HMI
COPY_FROM_NC_BINA RY	Transfer from NCK (PLC) to HMI
COPY_TO_NC	Transfer from HMI to NCK
COPY_TO_NC_BINARY	Transfer from HMI to NCK (PLC)
MAP_ACC_NC	Load ACC files from the NCK and prepare for use in the DDE interface

The status of a data transfer can be monitored by means of a status variable.

8.6.1 File transfer between HMI and NC/PLC

Description

Using these functions you can transfer data/files between the HMI and the NCK/PLC.

Application

These functions are suitable for transferring part programs and tool data to the NCK or for transferring S7 and C programs to the PLC.
Functions **without** the extension "BINARY" can transfer files, e.g. part programs, to the NC. The NCDDE server adds a block header to the data to be transferred. This header contains the size and date of the block and the path in the NCK file system.

Note

- Used for data transfer to the NCK.
- Cannot be used for transferring files to the PLC because an NC block header is always added to the data stream.

BINARY functions

Functions **with** the extension "BINARY" can transfer files, e.g. part programs, to the NCK. The NCDDE server transfers the data without an NC block header.

Note

- Can be used for transferring files to the PLC and to the NCK.
- PLC modules are always copied to the passive file system of the PLC. At that time they are not yet active. The passive blocks still have to be activated.

Syntax

The copy functions must be written as strings following the syntax:
COPY_FROM_NC (WinFile,NcFile,TransferState)
COPY_TO_NC (WinFile,NcFile,TransferState)
COPY_FROM_NC_BINARY (WinFile,NcFile,TransferState)
COPY_TO_NC_BINARY (WinFile,NcFile,TransferState)

Parameter

Table 8-3 Data access parameters

Name	Description
WinFile	Source or destination of the information in the HMI area
NcFile	File name for the NCK/PLC environment
TransferState	Variable characterizing the transfer state

WinFile parameter

Describes the source or the destination of the information on the HMI side.
The first character specifies the type.
This parameter is the default file name in a WINDOWS environment. It should include the drive specification, the path and the file name. e.g.
"C:\NC\test.MPF".

Piping with the parameter WinFile

If the first character of WinFile is the character @, the parameter is interpreted as a pipe name. In connection with the function COPY_TO_NC the "Copy via pipes" service can be executed.

Note

Suitable for reading and writing blocks up to a size of 500 bytes. Larger blocks are refused by the NCDDE server.

During transfers to the NCK/PLC (download) DDE pokes fill the pipeline and thus provide a direct transfer to the NCK/PLC. An empty poke indicates the end of the transfer.
During transfers from the NCK/PLC (upload) DDE requests empty the pipeline that is being filled for the running transfer. If a request returns empty data, this indicates the end of the transfer.

Shared memory access with the WinFile parameter

If the first character of WinFile is the character #, followed by a number in hexadecimal format, this is interpreted as WINDOWS shared memory, which is assigned to the global heap. The memory allocated with the WINDOWS function Global/Alloc must be initialized with the following structure. Following this header the utilizable data are appended. Example 8-22 shows the use of this parameter in Visual Basic.

Example 8–22 Shared memory access with WinFile

```
struct NCDDE_DOMAINMAP_HEADER {
    unsigned short handle;          //  buffer handle (HGLOBAL) is preassigned
                                    //  by client)
    unsigned short header_size;     //  length of header (preassigned by client)
    unsigned long shared_size;      //  useable length of data area (preassigned
                                    //  by client)
    unsigned long fill_count;       //  number of valid bytes in data area (preassigned
                                    //  by client when downloading and by server when
                                    //  uploading)
    unsigned long state;            //  corresponds to the transfer data variable in
                                    //  the transfer command
                                    //  < 100: Transfer is running; "state" shows what percentage
                                    //  of the transfer has been completed
                                    //   ==100: Transfer successfully completed
                                    //  > 100: Transfer was stopped with error,
                                    //  "state" shows the ncdde error code
                                    //  (set by the server)
    unsigned long file_mod_time;//  file modification time value:
                                    //  0 denotes current time (preassigned by client when
                                    //  downloading and by server when uploading)
    unsigned long server_private;//  server-specific data (set by the server)
    unsigned long client_private;  //  client-specific data (set by the client)
    unsigned long magic;            //  signature for an additional type check, value always
                                    //  NCDDE_MAGIC = 0xF6F7F8F9 (preassigned
                                    //  by client)
};
```

NcFile parameter

The parameter "NcFile" is the path name in the NCK/PLC environment. This name is built up of a configurable part that is required for addressing the NCK in question and additionally the domain path in the NCK environment.
Domains in NCK are addressed via the NCDDE server using an NC file name.

/NC	Which area: PLC or NCK
/_N_MPF_DIR	Path specification for the NC
/_N_WS03_MPF	File name

TransferState parameter

The parameter "TransferState" is the name of a local NCDDE variable (variable type: fixed) to be used for returning the state of the transfer that is carried out in the background. This variable can also be created by the NCDDE server.
The TransferState variable characterizes the state of the file transfer:

Table 8-4 Transfer state

Transfer state	Value	Meaning
Transfer is started	0	The opening protocol to the CNC is being handled (opens file)
Transfer is running	1 ... 98	Transfer is running. The number approximately reports the percentage of the file, that has already been transferred (see note).
Transfer is terminated	99	The closing protocol to the CNC is being handled (closes file)
Transfer successful	100	The job has been executed without error
Transfer stopped with error code	>100	Transfer is stopped. TransferState contains the reported error code (Chapter 11.7). This corresponds to the value of the LastError variable (section 11.7).

The range of values has been chosen, so that values <=100 indicate normal conditions, whereas all other values indicate error conditions.

Note

As long as the variable has a value between 1 and 99 it cannot be used for carrying out further file transfers.

Stopping the file transfer

In order to stop a running file transfer, the transfer variable must be overwritten with a valid error code, i.e. each byte of the transfer variable which is defined as "LONG" (4 bytes) must have a value **not equal** to 0.
Example of a valid error code: **16909060**

Visualization

For visualizing the transfer state, the variable TransferState may be used e.g. in a bar display via an Advise/Hotlink connection.

Note

During a transfer in BINARY mode and during a transfer using pipes, no information on the block size is available. Therefore no information regarding the current percentage of transferred data can be provided to the variable TransferState and it stays constant at 50%.

With very short files the display can jump immediately from 1 to 99. This is essentially a problem with hotlinks, however, and is due to the fact that the client/application does not retrieve the data quickly enough from the NCDDE server.

Uploading a part program

The following example copies the part program "BSP.MPF" to the file "test.mpf" in the directory "C:\NC". The file "test.mpf" is newly created. The part program "BSP.MPF" must exist in the NCK.

Example 8–23 Uploading a part program

```
Sub Form_Load ()
        Label1.LinkTopic = "NCDDE|ncu840d"
        Label1.LinkMode = 2
        Label1.LinkExecute  "COPY_FROM_NC (C:\nc\test.mpf,
                                  Ä/NC/_N_MPF_DIR/_N_BSP_MPF,trans)"
End Sub
```

Downloading a part program

The following example copies a file called "test.mpf" from the directory "C:\NC" to the NCK directory "_N_MPF_DIR". In the NC the part program's name is "**BSP.MPF**".

Example 8–26 Downloading a part program

```
Sub Form_Load ()
        Label1.LinkTopic = "NCDDE|ncu840d"
        Label1.LinkMode = 2
        Label1.LinkExecute "COPY_TO_NC(C:\NC\test.MPF,
                                  Ä/NC/_N_MPF_DIR/_N_BSP_MPF,trans)"
End Sub
```

Downloading a part program with piping

The following example shows an application of the **Pipe** mechanism. The file PIPE1.MPF is created in the NCK and the NC block "G01 F11111 X5555" is written to it.

Example 8–27 Downloading a part program with piping

```
Sub Form_Load ()
        'start pipe
        Label1.LinkTopic = "NCDDE|ncu840d"
        Label1.LinkMode = 2
        Label1.LinkExecute "COPY_TO_NC(@pipe,
                                Ä/NC/_N_MPF_DIR/_N_PIPE1_MPF,trans)"
        'describe pipe
        Label2.LinkTopic = "NCDDE|NCU840D"
        Label2.LinkMode = 2
        Label2.LinkItem = "@pipe"
        Label2.Caption = "G01 F11111 X5555"
        Label2.LinkPoke
        'end pipe
        Label2.Caption = ""
        Label2.LinkPoke
End Sub
```

Downloading an S7 block to the PLC

Transferring the block "**OB1.PLC**" to the PLC's passive file system.

Note

PLC blocks are always copied to the passive file system of the PLC.
At that time they are not yet active. The passive blocks then have to be activated (see example 8-36).

Example 8–28 Downloading an S7 block to the PLC

```
Label1.LinkItem = "ncdde|ncu840d"
Label1.LinkMode = 2
Label1.LinkExecute "COPY_TO_NC_BINARY(C:\TMP\OB1.PLC,
                        Ä/PLC/_0800001P, trans)"
```

8.6.2 Extended data transfer between HMI and NC/PLC

Description

These functions allow the user to transfer data between the NCK/PLC and the HMI.

Application

These functions are especially suitable for transferring single blocks, sections of part program or for editing part programs stored on the NC.

Note

The difference between the "normal" and "BINARY" variants of the functions is described in section 8.6.1.

Syntax

The extended copy functions must be written as strings following the syntax below:

COPY_FROM_NC	(WinFile,NcFile,seekPos,seekLen,compare String,skipCount)
COPY_FROM_NC(_BINARY)	(WinFile,NcFile,seekPos,seekLen,compare String,skipCount)
COPY_TO_NC	(WinFile,NcFile,seekPos,seekLen,compare String,skipCount)
COPY_TO_NC(_BINARY)	(WinFile,NcFile,seekPos,seekLen,compare String,skipCount)

Parameter

The parameters are described in the table.

Table 8-5 Parameters for COPY_TO/FROM_NC commands

Name	Description
WinFile	Source or destination of the information in the HMI area
NcFile	File name for the NCK/PLC environment
seekPos	Seek pointer: Start point for the copy process, identifier B for block or C for character
seekLen	Window size: Area to be transferred, identifier B for block or C for character
compareString	Search string, maximum length is 32 characters
skipCount	Number of found search-strings to be skipped

The commands return after having completely processed all of its sub-commands. Errors which occur during the execution of the command, can be analyzed using the variable LastError.

The following example shows a typical application of the extended commands.

File transfer of a program part

File transfer of the first **1024** bytes of the part program **"TP1.MPF"** to the file "test.dat" in the directory "C:\NC"

Example 8–29 File transfer of a program part

```
Sub Form_Load ()
        Label1.LinkTopic = "NCDDE|NCU840D"
        Label1.LinkMode = 2
        Label1.LinkExecute"COPY_FROM_NC(C:\NC\test.dat,
                                Ä/NC/_N_MPF_DIR/_N_TP1_MPF,1 ,1024 , , 0
)"
End Sub
```

Transferring single blocks

Pipe transfer of blocks 2 to 4 to the part program X.MPF. Existing blocks are overwritten.

Example 8–30 Transferring single blocks

```
Sub Form_Load ()
          Label1.LinkTopic = ”NCDDE|NCU840D”
          Label1.LinkMode = 2
          Label1.LinkExecute " COPY_TO_NC_BINARY ( @xpipe ,
                    Ä/NC/_N_MPF_DIR/_N_X_MPF , B2 , 3 , , 0 )"
End Sub
```

Transferring a block

Text transfer (max. text length: 200 bytes) to the 2nd block of the part program TEST.MPF. The second block is overwritten.

Example 8–31 Transferring a block

```
Sub Form_Load ()
          Label1.LinkTopic = "NCDDE|NCU840D"
          Label1.LinkMode = 2
          Label1.LinkExecute "COPY_TO_NC ( ""!This is the content of the
          2nd block""  , /NC/_N_MPF_DIR/_N_TEST_MPF, B2 ,1 , , 0 )"
End Sub
```

8.6.3 MAP functions between domains

MAP_ACC_NC

Description
This function allows the user to make global user data (GUD) and NCK machine data available to the NCDDE server. These data are stored in files with the extension ACC. Files with the extension ACC are located in the NCK and contain access descriptions for the variables.

Application
The command MAP_ACC_NC allows ACC files from the NCK to be read and to be prepared for use in the NCDDE interface i.e. the corresponding connections to these data are created or identified in the NCDDE server.

Note

Allows the user to make new NCK data available to the NCDDE server.
Otherwise these variables/data could not be accessed.

The command works in the same way as the command COPY_FROM_NC except that the information from the ACC file is also decoded and prepared in such a way that it can be viewed on the DDE interface.

Syntax
The call follows the syntax below:
MAP_ACC_NC
Ä(WinFile, NcFile, TransferState, Area, DataBlock, Timeout, Prefix)

Parameter
Table 8-6 contains a more detailed description of the parameters. The first three parameters correspond to those of the other domain services (see 9.6.1). For the sake of completeness they are also included in the following table.

Table 8-6 Parameters for the command MAP_ACC_NC

Name	Description
WinFile	Source or destination of the information in the HMI area
NcFile	File name for the NCK/PLC environment
TransferState	Variable characterizing the transfer state
Area	Area address of the ACC data, described in section 11.1.1, table 11-1. Here is a complete list: **Section** **Area address** NCK 0 Mode group 1 Channel 2 Axis 3 Tool magazine 4 Feed drive 5 Main spindle drive 6 Reserved 7
DataBlock	Module type for Variable Service: Numerical hexadecimal value ranging from 00 to FF, described in section 11.3.1, for example (extract): **Block name** **(DataBlock)** System state data (Y) 10 Global user data (GUD) 17 OEM tool data (TU) 24 Magazine directory (TMV) 2B
Timeout	Timing of an NCK-HMI transaction in seconds
Prefix	Any string inserted in front of the ACC variable.

Note

If the parameter WinFile is a file with the extension .NSK, the Domain Service in addition to an ACC file creates an NSK file, which contains the corresponding LINK commands.

ACC files

```
/NC/_N_NCK_GD2_ACC      ;    global NCK user variable MGUD
/NC/_N_CH02_GUD_ACC     ;    global user variable in the 2nd channel
/NC/_N_AX_SEA_ACC       ;    axis-specific setting data
/NC/_N_CH_TEA_ACC       ;    channel-specific NC machine data
```

Creating connections for drive machine data

MAP_ACC_NC	Command header
L:\MMC2\NCMDACC.NSK	Filename in the WINDOWS environment
/NC/_N_VS_DIR/_N_VS_TEA_ACC	NC domain
trans	TransferState variable
5	Area, here number 5 stands for the area Feed drive
7F	DataBlock, here address 7F stands for the drive service values block
10	Time monitoring, here 10 seconds
/ACC/driveVSA/MD/	Prefix, here the string which will be used for accessing the data later

Example 8–32 Creating connections for drive machine data

```
Sub Form_Load ()
        Label1.LinkTopic = "NCDDE|NCU840D"
        Label1.LinkMode = 2
        Label1.LinkExecute "MAP_ACC_NC (L:\MMC2\NCMDACC.NSK,
        /NC/_N_VS_DIR/_N_VS_TEA_ACC,trans,5,7F,10,/ACC/driveVSA/MD/)"
End Sub
```

Accessing already established connections

Accessing a link already established in the previous example with the following components:

/ACC/driveVSA/MD/	Prefix resulting from the previous call of the MAP command
$MD_TORQUE_THRESHOLD_X[1]	Name of the machine data, starting with $.

Example 8–33 Accessing already established connections

```
Sub Form_Load ()
        Label1.LinkTopic = "NCDDE|NCU840D"
        Label1.LinkMode = 2
        Label1.LinkItem=
"/ACC/driveVSA/MD/$MD_TORQUE_THRESHOLD_X[1]"
        Label1.LinkRequest
End Sub
```

Accessing global user variables

Accessing global user variables is described in section 9.13

Examples of MAP_ACC_NC commands

In these examples make sure that the WinFile and NcFile parameters are separated by a comma followed by a space.

Example 8–34 Examples of MAP_ACC_NC commands

```
All machine data:
MAP_ACC_NC(c:\tmp\c.nsk, /NC/_N_COMPLETE_TEA_ACC,trans,0,1A,10,/MD/)

All NCK machine data:
MAP_ACC_NC(c:\tmp\nc.nsk, /NC/_N_NC_TEA_ACC, trans,0,1A,10,/NC/)

Channel machine data for channel 1:
MAP_ACC_NC(c:\tmp\ch1.nsk, /NC/_N_CH1_TEA_ACC, trans,2,1A,10,/CH1/)

All axis-specific machine data:
MAP_ACC_NC(c:\tmp\ax.nsk, /NC/_N_AX_TEA_ACC, trans,3,1A,10,/AX/)

All global NC setting data:
MAP_ACC_NC(c:\tmp\sea.nsk, /NC/_N_NC_SEA_ACC, trans,0,16,10,/SEA/)

All axis-specific setting data:
MAP_ACC_NC(c:\tmp\axs.nsk, /NC/_N_AX_SEA_ACC, trans,3,16,10,/AXSEA/)

All global NC user data:
MAP_ACC_NC(c:\tmp\gud.nsk, /NC/_N_NC_GUD_ACC, trans,0,17,10,/GUD/)

All channel-specific user data:
MAP_ACC_NC(c:\tmp\gud.nsk, /NC/_N_CH_GUD_ACC, trans,2,17,10,/GUD/)

All global NC user data 1 (=SGUD):
MAP_ACC_NC(c:\tmp\gd1.nsk, /NC/_N_NC_GD1_ACC, trans,0,17,10,/GUD1/)

All channel-specific user data 1 (=SGUD):
MAP_ACC_NC(c:\tmp\gd1.nsk, /NC/_N_CH_GD1_ACC, trans,2,17,10,/GUD1/)

All global NC user data 2 (=MGUD):
MAP_ACC_NC(c:\tmp\gd2.nsk, /NC/_N_NC_GD2_ACC, trans,0,2D,10,/GUD2/)

All channel-specific user data 2 (=MGUD):
MAP_ACC_NC(c:\tmp\gd2.nsk, /NC/_N_CH_GD2_ACC, trans,2,2D,10,/MGUD/)

All global NC user data 3 (=UGUD):
MAP_ACC_NC(c:\tmp\gd3.nsk, /NC/_N_NC_GD3_ACC, trans,0,2E,10,/GUD3/)

All global NC user data 4 (=GUD4):
MAP_ACC_NC(c:\tmp\gd4.nsk, /NC/_N_NC_GD4_ACC, trans,0,2F,10,/GUD4/)

All global NC user data 5 (=GUD5):
MAP_ACC_NC(c:\tmp\gd5.nsk, /NC/_N_NC_GD5_ACC, trans,0,30,10,/GUD5/)

All global NC user data 6 (=GUD6):
MAP_ACC_NC(c:\tmp\gd6.nsk, /NC/_N_NC_GD6_ACC, trans,0,31,10,/GUD6/)

All global NC user data 7 (=GUD7):
MAP_ACC_NC(c:\tmp\gd7.nsk, /NC/_N_NC_GD7_ACC, trans,0,32,10,/GUD7/)

All global NC user data 8 (=GUD8):
MAP_ACC_NC(c:\tmp\gd8.nsk, /NC/_N_NC_GD8_ACC, trans,0,33,10,/GUD8/)

All global NC user data 9 (=GUD9):
MAP_ACC_NC(c:\tmp\gd9.nsk, /NC/_N_NC_GD9_ACC, trans,0,34,10,/GUD9/)
```

8.7 PI services

Overview

IMC Command is used to execute program instance (PI) services on the NC/PLC. For compatibility reasons jobs can also be determined via DDE on the NCK and the PLC. A summary of the PI services can be found in **PI.hlp**.
The PI services on the NCDDE server comprise:

PI_START	Instructs the NCK to execute a command
PI_START_BINARY	Instructs the PLC to execute a command
PI_STOP	Instructs the NCK to stop the execution of a command
PI_STOP_BINARY	Instructs the PLC to stop the execution of a command
PI_RESUME	Instructs the NCK to resume a stopped execution
PI_RESUME_BINARY	Instructs the PLC to resume a stopped execution

PI_START(_BINARY)

Description

This function allows you to send an instruction from the HMI to the NCK.

Application

These functions are suitable for starting jobs in the NCK.
The non-binary transfer is suitable for transfers to the NCK.
The binary transfer is suitable for transferring data to the PLC, the NC and the drives.

Syntax

The command line for calling PI services follow the syntax:
PI_START(server name, parameter 1, parameter 2 ... parameter n, PI name)
PI_START_BINARY (server name, parameter , PI name)
The PI name for NCK starts with _N_ , followed by 6 characters. Slightly different conventions apply to the PLC.

Parameter

The parameters are described in detail in the online help, since their functions depend on the PI service in which they are used.

Select a part program

This example shows how the PI service **"SELECT"** (selecting a program for execution in a channel) selects the part program **"BSP.MPF"**.
Note the fact that you must enter the area path in this command, not the NC file path.

Example 8–35 Select a part program

```
Sub Form_Load ()
        Label1.LinkTopic = "NCDDE|NCU840D"
        Label1.LinkMode = 2
        Label1.LinkExecute "PI_START(/NC,201,/_N_MPF_DIR/_N_BSP_MPF,
        _N_SELECT)"
End Sub
```

Activating OB 1

Activate an OB1 already stored in the passive file system:

Example 8–36 Activating OB 1

```
Sub Form_Load ()
        Label1.LinkTopic = "NCDDE|NCU840D"
        Label1.LinkMode = 2
        Label1.LinkExecute "PI_START_BINARY( /PLC,
        ""@1d1@1d0@@0800001P"",_INSE)"
End Sub
```

Stop selecting a part program

This example shows how the PI service "SELECT" (select program for execution in a channel) for the part program "BSP.MPF" is **stopped**.

Example 8–37 **Stop** selecting a part program

```
Sub Form_Load ()
      Label1.LinkTopic = "NCDDE|NCU840D"
      Label1.LinkMode = 2
      Label1.LinkExecute "PI_STOP(/NC,201,/_N_MPF_DIR/
                          Ä_N_BSP_MPF, _N_SELECT)"
End Sub
```

Stop activating OB 1

Example 8–38 **Stop** activating OB 1

```
Sub Form_Load ()
        Label1.LinkTopic = "NCDDE|NCU840D"
        Label1.LinkMode = 2
        Label1.LinkExecute "PI_STOP_BINARY( /PLC,
        Ä""@1d1@1d0@@0800001P"", _INSE)"
End Sub
```

8.8 Other commands on the NCDDE server

Overview

Other NCDDE server commands are summarized in Table 8-7:

Table 8-7

Command	Meaning
NEW	Create local variables
FREE	Delete variables
ANIMATE	Continuously change a local variable
CALL	Execute NCDDE commands in files
PLC_MEMORYRESET	Reset the PLC memory

NEW

Description

Creates a local/internal variable in the NCDDE server which can then be accessed.

Application

With the command NEW a new local/internal variable on the NCDDE server is created. When accessing this variable, no communication with the NCK takes place. If a variable by the name VarName already exists, it is deleted before the new one is created (similar to the FREE command, section 8.8).

Syntax

NEW (VarName , value)

Parameter

Table 8-8 Parameters for NEW

Parameter	Syntax	Meaning
VarName	<String>	Name of the variable to be created
Value	<Parameter>	Initialization value for the variable

Create an internal variable

Creates the variable "test" in the NCDDE server and initializes it with the value **10.0**.

Example 8–39 Creating an internal variable

```
Sub Form_Load ()
Label1.LinkTopic = "NCDDE|NCU840D"
Label1.LinkMode = 2
Label1.LinkExecute " NEW ( test , 10.0 )"
End Sub
```

FREE

Description
Deletes a variable in the NCDDE server

Application
The command "FREE" deletes variables created by the commands "NEW" and "LINK". If the variable is currently being used by a file transfer service (section 9.6.1) as a state variable, the command "FREE" is rejected. If there exist Advise Links (Hotlinks) to the variables, these links are removed. Other transactions with the NCK and PLC are aborted.

Syntax
FREE (VarName)

Parameter

Table 8-9 Parameters for FREE

Parameter	Syntax	Meaning
VarName	<String>	Name of the variable to be deleted

Delete an internal variable
Deletes the variable "**test**" in the NCDDE server

Example 8–40 Deleting an internal variable

```
Sub Form_Load ()
        Label1.LinkTopic = "NCDDE|NCU840D"
        Label1.LinkMode = 2
        Label1.LinkExecute " FREE( test )"
End Sub
```

ANIMATE

Description
Causes the NCDDE server to continuously change the value of a local variable that has been created with "NEW". Values are incremented in cycles of approximately 1 second.

Application
Used for testing your application.

Syntax
Animate (VarName)

Parameter

Table 8-10 Parameters for Animate

Parameter	Syntax	Meaning
VarName	<String>	Name of the variable to be changed

Change an internal variable
Continuously changes the variable **"test"** in the NCDDE server.

Example 8–41 Changing an internal variable

```
Sub Form_Load ()
        Label1.LinkTopic = "NCDDE|NCU840D"
        Label1.LinkMode = 2
        Label1.LinkExecute " ANIMATE( test )"
End Sub
```

CALL

Description
Interprets command files

Application
With the command CALL NCDDE commands recorded in files are executed. Each line of the file is passed as a command to the NCDDE server. The file may contain comments and space lines. The extension .NSK should be used consistently for all NCDDE command files.

Note

Allows you to customize the NCDDE server for your applications.

Syntax
CALL (FileName)

Parameter

Table 8-11 Parameters for CALL

Parameter	Syntax	Meaning
VarName	<String>	Name of the NCDDE command file

For example see file "\MMC2\ NCDDE311.NSK"

PLC_MEMORYRESET

Description
The command PLC_MEMORYRESET on the NCDDE server resets the PLC memory. Specify /PLC as area address.

Application
Resets the PLC memory
Syntax
PLC_MEMORYRESET(AreaAddr)

Parameter

Table 8-12 Parameters for PLC_MEMORYRESET

Parameter	Syntax	Meaning
AreaAddr	<String>	Area address

Reset the PLC
Resets the PLC; the PLC must be stopped before this command can be used

Example 8–42 Reset the PLC memory

```
Sub Form_Load ()
        Label1.LinkTopic = "NCDDE|NCU840D"
        Label1.LinkMode = 2
        Label1.LinkExecute "PLC_MEMORYRESET(/PLC)"
End Sub
```

8.9 OEM VisualBasic controls (OCX files)

Overview

These controls overcome some of the inadequacies of DDE communication with standard controls in Visual Basic.

Note

When developing an OEM application, we recommend that these OEM Visual Basic controls be used for accessing the NCDDE server

Standard controls such as Label and TextField can be used for DDE communication. This communication however has some insufficiencies:

- Events get lost.
  There is no certainty that the change procedure in the VB program will be initiated if the value of a DDE variable changes with LinkMode = 1. (The only solution is to pole the value via "Timercontrol").
- The DDE functionality cannot be nested.
  In a DDE change procedure, other DDE functions in the control cannot be activated (only solution is via timers or the like).
- Only synchronous transactions are realized. The response time intervals for installed Hotlinks and for requests are quite long. This especially applies, when these actions involve more than one CPU (NCK, PLC).
- Demand for resources is high
  For each control instance that uses DDE, a DDE connection is established. Each connection uses 2 WINDOW handles, blocking scarce user resources.
- LastError is not handled conveniently in NCDDE
  When communication with the NCK is being carried out via NCDDE, the NCDDE server provides the DDE variable "LastError" for detailed analysis of errors. This variable is specific for each DDE conversation and is valid only if the DDE return is DDE_FNOTPROCESSED.

8.9.1 File DDECTL.VBX

The conversion of VB applications from 16-bit to 32-bit means that DDECTL.VBX is no longer used (see also "Converting VB applications from 16-bit to 32-bit").

8.9.2 File DCTL.VBX

The conversion of VB applications from 16-bit to 32-bit means that DCTL.VBX is no longer used (see the section below and "Converting VB applications from 16-bit to 32-bit).

8.9.3 File DCTL.OCX

Overview

The Visual Basic control DCTL.OCX is a graphical control with extended DDE capabilities. It is similar to the standard control Label but offers several additional advantages:

- Minimal consumption of Windows resources:
  "DDE Requests", "DDE Pokes" and "DDE Executes" use resources only temporarily. With the DCTL.OCX control all DDE hotlinks in a Windows process use only one Windows handle in total.
- Close cooperation with the NCDDE server:
  For example, it supplies the LastError value in the event of failed DDE transactions
- Higher speed:
  Applications are speeded up because it allows multiple/parallel transactions with a server.
- Faster output:
  Optimized screen output and index filtering allows faster screen display. Additionally it makes the BASIC programming easier.
- Avoiding side effects:
  Typical side effects of Visual Basic controls like aborting programmed connections by pressing the ESCAPE key can be avoided.

This section describes the features of this new control, followed by the additional events. Finally, some examples illustrate the potential applications of DCTL.OCX.

Characteristics

Most properties of the DCTL.OCX control correspond to those of standard Visual Basic controls. These accesses include:

- Style properties
- Color properties
- Base properties
- Drag properties
- Font properties.

Some properties are different from standard Visual Basic controls:

- DDE properties
- HorAlignment property
- VertAlignment property
- WordBreak property
- TabSize property
- LastError property
- Data property
- DataToCaption property
- LinkCmd property
- LinkNext property
- LinkFilter property.

DDE properties

The DDE properties include
LinkItem
LinkTopic (the default NCDDE)
LinkTimeout (for synchronizing LinkCmd).

HorAlignment property

This property controls the horizontal text justification of the caption display:

Table 8-13 Horizontal alignment

Value	Property
LeftJustify	Left-justified (default)
RightJustify	Right-justified
HorCenter	centered

VertAlignment/ Multiline property

This property controls the vertical text justification of the caption display;
alternatively to a vertical text justification you can select multiline display.
In this case the word wrapping is determined by the **WordBreak** property:

Table 8-14 Vertical alignment

Value	Property
VertCenter	Vertically centered (default)
TopJustify	top
BottomJustify	bottom
MultiLine	Multiline

WordBreak property

If the property **VertAlignment/Multiline** has been set **Multiline** (value = 3), the property **WordBreak** determines the word wrapping:

Table 8-15 Type of word wrap

Value	Property
False	Word wrap by CR/LF (carriage-return line-feed sequence)
True	Automatic word wrap if the word does not fit in the line. A carriagereturn line-feed sequence will also wrap the line.

TabSize property

This property defines the tab increment. The default value is 8 characters.
The value range can go up to 255.

LastError property

This property allows error messages to be sent.
The error variable **LastError** is reset (LastError = 0) when a DDE connection is established with a server. When an error occurs during a transaction and DCTL control detects this error, it then will ask for a detailed error code, that can be accessed using the property **LastError**.

Note

The DCTL control does not decode errors which are transferred as data:
These include # characters and spaces from the NCDDE server.

The LastError variable for the DCTL control is described in more detail in section 11.7.1.

Data property

The **Data** property is used as an argument for the following DDE transactions:

Table 8-16 Arguments for DDE transactions

DDE transaction	Argument
Request	Requested variable value, if the "DataToCaption" property is set to FALSE
Advise Link	Updated values, if the "DataToCaption" property is set to FALSE
Poke	Value to be transferred
Execute	Command to be executed

DataToCaption property

The DataToCaption property determines the destination of data from a DDE transaction.

Table 8-17 Destination for data

Value	Meaning
True	Data destination is the Caption property
False	Data destination is the Data property

LinkCmd property

Changes to the **LinkCmd** property initiate DDE activities in the DCTL control. If there is no activity, LinkCmd = 0.

Table 8-18 Features of LinkCmd property

No.	Change to	DDE activity	Terminated by
1	Advise Link	Establishes an AdviseLink. Returns after the Advise Link has been established. The AdviseLink can be deleted by the Stop command.	Stop
2	Advise Link_ NotifyData	As in 1, plus action (1) when DDE data arrive	Stop
3	Advise Link_ NotifyDataWhenVisible	As in 1, plus action (2) when DDE data arrive	Stop
4	Advise LinkAsync	Initiates an AdviseLink. Returns before the Advise Link has been established. The AdviseLink can be deleted by the Stop command.	Stop
5	Advise LinkAsync_ NotifyData	As in 4, plus action (1) when DDE data arrive	Stop
6	Advise LinkAsync_ NotifyDataWhenVisible	As in 4, plus action (2) when DDE data arrive	Stop
7	Stop	Deletes an AdviseLink. Returns after the AdviseLink has been deleted.	Itself
8	StopAsync	Deletes an AdviseLink. Returns before the AdviseLink has been broken.	sync
9	StopAsync_ Notify	As in 8, plus action (1) when the AdviseLink has been broken.	sync
10	StopAsync_ NotifyWhenVisible	As in 8, plus action (2) when the AdviseLink has been broken.	sync
11	Request	Reads a DDE variable. Returns after the reading has been completed.	Itself
12	RequestAsync	Starts reading a DDE variable. Returns before the reading has been completed.	sync
13	RequestAsync_ Notify	As in 12, plus action (1) when reading has been completed.	sync
14	RequestAsync_ NotifyWhenVisible	As in 12, plus action (2) when reading has been completed.	sync
15	Execute	Sends a command to the server. Returns after the execution of the command is completed.	Itself
16	ExecuteAsync	Sending a command to the server. Returns before the execution of the command is completed.	sync
17	ExecuteAsync_ Notify	As in 16, plus action (1) when the execution of the command has been completed.	sync
18	ExecuteAsync_ NotifyWhenVisible	As in 16, plus action (2) when the execution of the command has been completed.	sync
19	Poke	Writes a DDE variable. Returns after the writing has been completed. The value is written from Data (not from Caption).	Itself

No.	Change to	DDE activity	Terminated by
20	PokeAsync	Initiates writing of a DDE variable. Returns before the execution of the command is completed.	sync
21	PokeAsync_Notify	As in 20, plus action (1) when writing has been completed.	sync
22	PokeAsync_ NotifyWhenVisible	As in 20, plus action (2) when writing has been completed.	sync
23	sync	Terminates asynchronous commands like synchronous commands. No operation if there is no asynchronous command working.	Itself

Actions

The actions used in the above table are:

Action (1)
An attempt is made to call the event procedure **DdeNotify**. If Visual Basic does not call an event procedure at this point or if the parameter of the event procedure has not been changed, the DCTL control tries ten times per second to send this event as long as the parameter for the DdeNotify event procedure remains unchanged.

Action (2)
The DCTL control calls the **DdeNotify** event procedure when it receives a paint message from WINDOWS. To ensure that these paint messages are created, the pixel in the top left corner of the control is invalid as long as the parameter of DdeNotify remains unchanged. In effect, this mechanism suppresses the display if the control is not visible.

Note

New DDE activities should be started only after preceding DDE activities have been terminated. This can be achieved using the parameter in the rightmost column (terminated by) of the table.

The Hotlinks of all DCTL controls located in the same window share one DDE connection if they use the same "LinkTopic" property. The DDE connections of the other activities (excluding hotlink) are dynamically created and deleted. Because of that and since a DCTL control does not have a window, the demand on WINDOWS resources is reduced dramatically.

Note

Changing the LinkCmd property causes the LinkTopic, LinkTimeout and LinkItem properties to be evaluated. This means that any errors in these values will emerge when the LinkCmd property is changed. Therefore they must be evaluated at this point.

LinkNext property

The optional property **LinkNext** holds the name and optionally the index of another DCTL control.
If the LinkNext property is not empty, the DCTL control scans a string transmitted via AdviseLink for the NCDDE index specifications (5 digits followed by 'a colon'). It separates the string into indexed substrings and forwards these along the chained list of DCTL controls built up by the LinkNext properties. The control whose **LinkFilter** property matches the index receives the corresponding substring. Substrings not taken in this way get lost.

LinkFilter property

The value for **LinkFilter** can be between 0 and 65535. Its use is illustrated in the LinkNext paragraph above.

Events for DCTL.OCX

Most DCTL control events are identical to other standard Visual Basic controls, such as

- Click
- DblClick
- MouseDown
- MouseMove
- MouseUp
- DragDrop
- DragOver
- KeyDown
- KeyPress
- KeyUp.

Event DdeNotify

The **DdeNotify** event was created especially for DDE communication: It indicates the arrival of new AdviseLink data or the conclusion of an asynchronous DDE transaction. Details of application are described in the paragraph LinkCmd (actions (1) and (2)).

Syntax

Sub *ctlname*_DdeNotify (*Flag* as integer)
with the argument Flag, which indicates to the DCTL control that the event has actually arrived at the basic level.
The value of Flag is expected to change whenever the event procedure is called. This is because the DCTL control continues to initiate the DdeNotify event until the Flag argument changes. If this change does not take place, a cooperative permanent activity results, that unnecessarily burdens the system.

8.9.4 Applications of DCTL.OCX

Reading and displaying a variable

A DDE variable is to be read immediately and displayed on screen. Then a DCTL control (e.g. named DCTL1) is to be placed in a suitable position on the screen. The associated code would then look like this:

Example 8–43 Reading and displaying a variable

```
Sub Form_Load ()
        Dctl1.LinkItem = "/Channel/Parameter/R[1]" ’ the variable name
        Dctl1.DataToCaption = TRUE ’that’s default, can be omitted
        Dctl1.LinkCmd = 11 ’ commands the reading
        ' here Dctl1.Caption holds the value of the DDE variable
End Sub
```

Reading a variable to the Data property

A DDE variable is to be read immediately and processed without being displayed on screen. Then a LABEL type DCTL control (e.g. named DCTL2) is to be positioned in a form. The associated code would then look like this:

Example 8–44 Reading to the Data property

```
Sub Form_Load ()
        Dctl1.LinkItem = "/Channel/Parameter/R[1]" ’ the variable name
        Dctl1.DataToCaption = FALSE ’ routing data to the Data property
        Dctl1.LinkCmd = 11 ’ commands the reading
        ' here Dctl1.Data holds the value of the DDE variable
End Sub
```

Writing a DDE variable

A DDE variable is to be written. Then a LABEL type DCTL control (e.g. named DCTL3) is to be positioned in a form. The associated code would then look like this:

Example 8–45 Writing a variable

```
Sub Form_Load ()
        Dctl1.LinkItem = "/Channel/Parameter/R[1]" ’ the variable name
        Dctl1.Data = 12 ’ the value
        Dctl1.LinkCmd = 19 ’ commands the writing
        ' here the NC variable is already successfully set to 12
End Sub
```

Executing a command

A DDE command is to be transferred to a server. Then a LABEL type DCTL control (e.g. named DCTL4) is to be positioned in a form. The associated code would then look like this:

Example 8–46 Executing a command

```
Sub Form_Load ()
        Dctl1.Data = "Pi_start(/NC,001,_N_SET_OF)" ’ the command
        Dctl1.LinkCmd = 15 ’ sends the command
        ' here the command is already successfully executed
End Sub
```

Displaying a DDE hotlink

The value of a DDE variable is to be displayed on screen. Then a DCTL control (e.g. named DCTL4) is to be placed in a suitable position on the screen. A hotlink is to be set up to run as a background activity of the DCTL control. The associated code would then appear as shown in the example below.
The coded Property setting can also be executed as soon as it is created, however.

Example 8–47 Hotlink with DCTL

```
Sub Form_Load ()
Dctl1.LinkItem = "/Channel/Parameter/R[1]" ’ the variable name
Dctl1.DataToCaption = TRUE ’that’s default, can be omitted
Dctl1.LinkCmd = 4 ’ initiates the creation of a hotlink
End Sub
```

Accelerating operations with parallel running

When a form is loaded, a number of separate DDE activities have to be executed. It is desirable for the form to load quickly. The best way to achieve this – with the DCTL control – is with DDE activities running in parallel. A suitable code is shown in the example below.

Example 8–48 Accelerating operations with parallel running

```
Sub Form_Load ()
’ start reading variable 1
Dctl1.LinkItem = "/Channel/Parameter/R[1]" ’ the variable name
Dctl1.LinkCmd = 12
' initiates the reading
’ start reading variable 2
Dctl2.LinkItem = "/Channel/Parameter/R[2]" ’ the variable name
Dctl2.LinkCmd = 12
' initiates the reading
’ start reading variable 3
Dctl3.LinkItem = "/Channel/Parameter/R[3]" ’ the variable name
Dctl3.LinkCmd = 12
' initiates the reading
’ start a hotlink into display
Dctl4.LinkItem = "/Channel/Parameter/R[4]" ’ the variable name
Dctl4.DataToCaption = TRUE
'that's default, can be omitted
Dctl4.LinkCmd = 4
' creates the hotlink
’ start a command execution
Dctl5.Data = "Pi_start(/NC,001,_N_SET_OF)" ’ the command
Dctl5.LinkCmd = 16
' commands execution
’ here the tree variable accesses, the hotlink creation and the ’ ’command are
working in parallel. You can not be sure that any of ' ' them has completed.
Dctl1.LinkCmd = 23
’ wait until variable 1 read
Dctl2.LinkCmd = 23
’ wait until variable 2 read
Dctl3.LinkCmd = 23
’ wait until variable 3 read
Dctl5.LinkCmd = 23
' wait until command executed
’ here the variable accesses and the command have completed, the hotlink will
’ show it’s value on screen as soon as possible.
End Sub
```

Accelerating operations with text positioning

Large amounts of data are to be read and displayed very frequently. BASIC should not be used for displaying the data. In addition, the size of the data being transferred should be minimized. On the NCDDE side, array access to data and a combination of array access with data preparation ensures that these requirements are met. The DCTL control also offers multiline display and index filter functions.

Example 8–49 Accelerating operations with text positioning

```
'NCDDE array access with "Field" data preparation - Dctl index filtering:
' high frequency display of 5 values in 5 different controls
Dctl1.LinkItem = "/Channel/Parameter/R[1,5](!""!d%12.5g"")" ' variable
Dctl1.LinkFilter = 1          '   index of accepted data
Dctl1.LinkNext = "Dctl2"      '   linkage to the next control
Dctl2.LinkFilter = 2          '   index of accepted data
Dctl2.LinkNext = "Dctl3"      '   linkage to the next control
Dctl3.LinkFilter = 3          '   index of accepted data
Dctl3.LinkNext = "Dctl4"      '   linkage to the next control
Dctl4.LinkFilter = 4          '   index of accepted data
Dctl4.LinkNext = "Dctl5"      '   linkage to the next control
Dctl5.LinkFilter = 5          '   index of accepted data
Dctl1.LinkCmd = 4             '   initiates the creation of a hotlink
NCDDE array access - Dctl multiline display:
' high frequency display of 5 values in a column
Dctl1.LinkItem = "/Channel/Parameter/R[1,5](""!d%12.5g"
Dctl1.LinkItem = Dctl1.LinkItem + Chr$(13) + Chr$(10)+ """)"
Dctl1.DataToCaption = TRUE ' that's default, can be omitted
Dctl1.VertAlignment = 3 ' multiline selection
Dctl1.LinkCmd = 4 ' initiates the creation of a hotlink
```

Notifying when changed

When the layout of a screen display depends on a variable that is accessed via DDE, this variable is retrieved via hotlink in the DCTL control. If a variable change is notified, the screen content can then be rearranged. Since this process is very slow, it should only be executed when the form is visible.

Example 8–50 Notifying when changed

```
Sub Form_Load ()
‘basic code that creates a hotlink with notification “when visible”
Dctl1.LinkItem = "/Channel/Parameter/R[1]" ‘the variable name
Dctl1.LinkCmd = 6 ‘initiates the creation of a hotlink
‘handler for the notification event
End Sub
Sub Dct1_DdeNotify ( Index As Integer, Flag As Integer )
Flag = Flag + 1 ‘Flag MUST change
... ‘rearrangement to be done
End Sub
```

Error Handling

The typical error handling for reading, writing and executing is illustrated here.

Example 8–51 Error Handling

```
On Error Goto TypicalErrorHandling
Dctl1.LinkCmd = 11 ’ a DDE activity
...
TypicalErrorHandling:
Select Case Dctl1.Lasterror \ 16777216’ selection by error source
        Case 2 ’ MPI level error
        ... ’ e. g. no connection to NC
        Case 3 , 5 ’ NC/PLC level error
        ... ’ e. g. non existing variable
        Case 7 ’ Dctl level error
        Select Case Dctl1.Lasterror MOD 256 ’ selection by error code
        Case 7 ’ Dctl level timeout occurred
        ...
        Case Else ’ other Dctl level errors
        ...
            End Select
        Case Else ’ other error sources
        ...
End Select
...
```

8.10 Diagnostics – possibilities for NCDDE access

8.10.1 Test functions on the NCDDE server

Overview

Among other things, the testing functions of the NCDDE server give information on the local and external variables that were declared in the NCDDE server at the time the file was created. They are called as follows:

1. Start the NCDDE server in the SINUMERIK 840D MMC-OEM program group
2. Press ALT+TAB to switch to the NCDDE program, i.e. the NC communication DDE server; an icon is created
3. Click the icon: The following window appears.

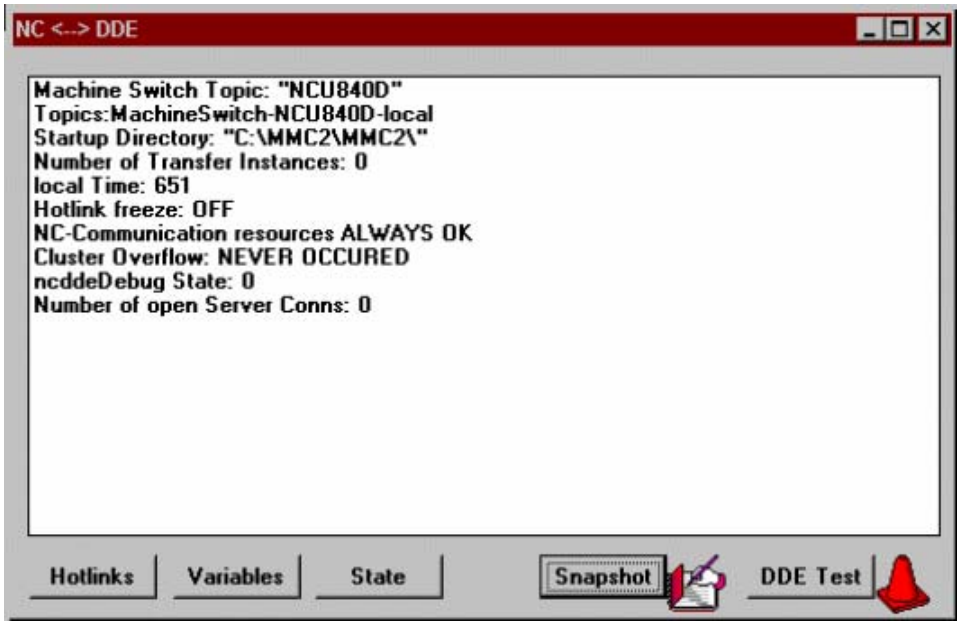


Fig. 8–2 Standard NCDDE server screen

These functions are mainly intended for debugging in the environment of the NCDDE server.

Hotlinks

Here a list is created which contains all existing Advise Links (Hotlinks and Warmlinks). It is organized as a 5-column table with the following meanings:

Table 8-19 Hotlinks

Column	About	Comments
1	PDU reference	Internal value: Possible PDU reference for communicating with the NCK and the PLC
2	Advise Link	LOCAL Link to a local variable REMOTE Link to an external variable PILED Double external Advise Link, connected to another job.
3	Update time	Time when the PDU was last refreshed, in an internal time scale for the NCDDE server
4	LastError variable	LastError value as described in section 11.7. It is not necessarily identical to the value reported at the server's DDE interface, since the last error from several transactions on one connection can be queried there.
5	Variable name	Variable name according to Chapter 11.

Variables

Shows the variables to which the NCDDE server is linked and where they are located: "LOCAL" or "PLC/NC".

Snapshot

Pressing this button creates a file named "NCDDE_X.TXT" that contains the status, hotlinks and variables for the NCDDE server.

DDE test

Pressing this button starts a testing program "DDETEST.EXE" with the following features

Table 8-20 DDE test commands

Command	Action	Meaning
Passive	none	Reset state, no function active
Hotlink	Start	Establish Advise Link
Request	DoIt	Read variable
Poke	DoIt	Write variable
Execute	DoIt	Execute a service

Specify the installed NC under **Service|Topic** e.g.: **NCDDE|NCU840D**.
"DEFAULT_NC" reads the settings from the file "MMC.INI".
The **Command** function is switched by clicking one of the 5 possibilities
LastError error messages are described in section 11.7.

8.10.2 Connection status

Variable NcState

The server makes known the state of its connection to the CNC via its local variable NcState. This variable exists immediately after the server has been started. It differs from the other local variables of the server only by the fact that it cannot be modified via the DDE interface.
The variable can have one of the following states:

Table 8-21 Values for the variable NcState

Value	Meaning
0	Normal operation
1	Some failed connections to CNC
2	All connections to CNC failed
3	Interpreting the startup file
4	Server initialization

8.10.3 Troubleshooting

Error acknowledgments by NCK

Error conditions such as lack of resources, access protection violations, incorrect operating mode, etc., are reported by the NCK via acknowledgments. If the NCDDE server cannot handle these error conditions, the corresponding transactions Request, Peek and Execute for the DDE interface are terminated with an error state, i.e. no result is obtained.

Variable LastError

A detailed diagnosis is provided by the variable LastError which has information about the last transaction on a connection. It can be read via the Link Item LastError. Once it has been read, the variable is set to zero. It always shows the last error registered in the NCDDE server.
The LastError variable consists of 4 bytes, with each byte containing the following error groups in descending order (high byte to low byte):

- Higher-level error class, error source
- Error area
- Error class
- Error code

The meaning of the individual error codes can be found in section 11.7 under NCDDE error messages.

Termination of connection to NCK

When the connection fails, the NCDDE server gives a negative acknowledgment to active Request, Poke and Execute transactions. As long as the connection has not been reestablished, the execution of further transactions is refused. At the same time the server tries to resume connection with the NCK. The state of the connection is indicated in the server's local variable NcState.

Handling Advise Links

If an Advise Link connection to the NCK has failed, the value returned by the NCDDE server is the character '#'. Advise Links are restored on the NCK after the connection has been reestablished.

Insufficient resources in the NCDDE server

If a lack of resources occurs in the NCDDE server, the corresponding DDE interface transactions are terminated with an error code.

8.11 NCDDE server configurations for network access

...
for development purposes only

8.12 Additional functions on the NCDDE server

8.12.1 Multivariable service

Overview

The multivariable service allows DDE access to multiple variables with a single NCDDE job. This accelerates access to a number of single variables but can only be used for read and write operations (not hotlinks).
As Item you should enter the items for the corresponding single variables/arrays, separated by '|'. The data supplied in read access is tightly packed. As with array access operations in the past, separators must be configured via format definitions or via a new access modification (see section 8.12.3). In read access operations the first character of the data supplied is interpreted as a separator for the individual data blocks.

Limits

- A **maximum of 8** tightly packed PDUs are sent per order. This means that normally more than 100 individual access operations can be executed in a single job. (The exact number must be determined by trial and error)
- The PDUs are all sent to a **single destination address**. This means that PLC and NC access operations cannot be combined in a single job. In addition, channel-specific access operations for different **channels cannot be mixed**. (NC requirement). The same applies to access operations to drive-specific variables.
- **Only true variables** (OPI interface/PLC BUB) can be addressed with the multivariable service. This means that date and time, system status list, directory information, etc., cannot be accessed.
- Note that the DDE **item size is limited to 255 characters**. If the item string exceeds this size, the item must be specified indirectly (see section 8.12.2).

Example for writing and reading with the multivariable service
Item:
/channel/parameter/r[1,2](|"!I%Id")|/channel/parameter/r[10](|)
Data e.g.: |1|2|10.000000

8.12.2 Indirect item output

Indirect item definition allows items with more than 255 characters (up to 4 KB). The content of a local variable can be used as an item for DDE access operations. In this case the name of the local variable must be specified as the item, prefixed by '>'.

Example for access to R10:

Exec: NEW(x,“/channel/parameter/r[10]“)
Item: >x
Data e.g. 10.000000

Note

When writing variables and executing commands in the NCDDE server the data length is restricted to 4KB. If this value is exceeded, error message 0X01050414 is returned.

8.12.3 New access modifications

By adding the control characters '|' and '^' in round brackets to the item string, access can be modified as follows:

'|' In CF_TEXT read access operations, inserts a '|' character in front of each data item. This is not evaluated in write access operations (see the example for the multivariable service in section 8.12.1).
'^' For a variable labeled in this way, hotlink disconnection (DEBA/DEBR) is disabled.

8.13 Accessing global user variables GUD, SGUD, MGUD, UGUD, GD3 to GD9

Overview

Global user variables are available for both the NCK and for one channel each. The NCK-specific global user variables exist in one instance per control. They are suitable for channel independent settings as well as for program coordination between channels.

Channel-specific global user variables exist once for each channel. They are suitable for channel-specific settings and for data transfer between different programs running in one channel.
The same applies to local user data. The same comments apply accordingly.
First of all you have to define and activate user variables before the NCDDE server can access them. For clustering the variables you then have to create and integrate the corresponding NSK files. This is done in five steps:

1. Create a definition file
2. Copy this definition file to the /_N_DEF_DIR of the NCK directory
3. Activate the user data as a *.ACC file by loading an INITIAL.INI file
4. Create the *.NSK file with the MAP command
5. Copy the new *.NSK file to the NSK file on the NCDDE server.

Definition file

Global user variables must be defined in definition files (modules) with fixed names:

- _N_GUD_DEF for GUD
- _N_SGUD_DEF for GD1 = SGUD Siemens global data
- _N_MGUD_DEF for GD2 = MGUD machine manufacturer global data
- _N_UGUD_DEF for GD3 = UGUD user global data
- _N_GUD4_DEF to _N_GUD9_DEF for GD4 to GD9

These files must be stored in the /_N_DEF_DIR directory of the NCK.
The total number of files that define global data depends on the value of the general machine data 18118 (MM_NUM_GUD_MODULES) (for more details see start-up manual). The default value of this machine data is 4.

Defining global data

Global data are defined by:

- Definition header DEF
- Section NCK or CHAN
- Type e.g. REAL or INT
- Variable name e.g. LIFTOFF_DIST
- Parameter In square brackets
- Comment Text starting with a semicolon

You can find more details in the 840D Programming Guide.

Creating definition files

Definition files can be created in the NCK or in the MMC.

In the NCK:

A definition file for global variables, as created at the part program level in the NCK, must be located in the /_N_DEF_DIR directory and comprises:

- Program code in the first line
- Comment line with path specification (will be evaluated)
- Definitions
- Terminating instruction M02, M17 or M30.

Example 8–55 Definition of a global variable in NCK

```
%_N_MGUD_DEF
;$PATH=/_N_DEF_DIR
DEF NCK REAL RUECKZUG ; Defining a global variable for NCK
DEF CHAN INT TABLE[100]  ; Defining a channel-specific variable
DEF CHAN REAL BLF_OFFS_X
M17 ; Terminate this line with RETURN
```

In the HMI:

A definition file for global variables with the file name MGUD.DEF, as can be created in the HMI, is located in the directory C:\TMP, for example, and consists of:

- Definitions
- Terminating instruction M02, M17 or M30.

Example 8–56 Definition of a global variable in HMI

```
DEF NCK REAL RUECKZUG ; Defining a global variable for NCK
DEF CHAN INT TABLE[100]  ; Defining a channel-specific variable
DEF CHAN REAL BLF_OFFS_X
M17 ; Terminate this line with RETURN
```

Note

This file must also be transferred from the HMI to the directory /_N_DEF_DIR on the NCK. This is done with the domain service COPY_TO_NC:COPY_TO_NC(C:\TMP\MGUD.DEF,/NC/_N_DEF_DIR/Ä_N_MGUD_DEF,trans)

Activating user data

User data are activated by copying a file named INITIAL.INI to the NCK. This file might be very short: Entering M17 followed by RETURN suffices. The following applies to a file INITIAL.INI located in the directory C:\TMP:

```
COPY_TO_NC(C:\TMP\INITIAL.INI, /NC/_N_INITIAL_INI, Ätrans)
```

This generates two ACC files named:
_N_NC_GD2_ACC For the global user variables
_N_CH_GD2_ACC For the channel-specific user variables
(applies for the example above with MGUD = GD2).

Note

Backup all programs, frames and machine data before you load the file INITIAL.INI, since this reformats the static memory.

Creating an NSK file for NCK

By calling the MAP command, ACC files can be used to create NSK files with the same names for global user variables on the NCK.
The example shows the call under Visual Basic.

Call the "MAP_ACC_NC" command

C:\MMC2\MGUD_NCK.NSK	: Filename in the WINDOWS environment
/NC/_N_NC_GD2_ACC	: NC domain
trans	: TransferState variable
0	: NCK area
2D	: MGUD block type
10	: Time monitoring of transaction with 10s
/ACC/NCK/MGUD/	: Any character string, chosen by the user, to be inserted in front of the user variable.

Example 8–57 Creating an NSK file for NCK

```
Sub Form_Load ()
        Label1.LinkTopic = "NCDDE|MMC2HW0"
        Label1.LinkMode = 2
        Label1.LinkExecute "MAP_ACC_NC(C:\MMC2\MGUD_NCK.NSK,
                Ä/NC/_N_NC_GD2_ACC, trans, 0, 2D , 10,
                Ä/ACC/NCK/MGUD/)"
End Sub
```

Creating an NSK file for a channel

By calling the MAP command, ACC files can be used to create NSK files with the same names for global user variables in the channel. The example shows the call under Visual Basic.

Call the "MAP command"

C:\MMC2\MGUD_CH.NSK	: Filename in the WINDOWS environment
/NC/_N_CH_GD2_ACC	: NC domain
trans	: TransferState variable
2	: Channel area
2D	: MGUD block type
10	: Time monitoring of transaction with 10s
/ACC/CH/MGUD/	: Any character string, chosen by the user, to be inserted in front of the user variable.

Example 8–58 Creating an NSK file for a channel

```
Sub Form_Load ()
        Label1.LinkTopic = "NCDDE|MMC2HW0"
        Label1.LinkMode = 2
        Label1.LinkExecute
"MAP_ACC_NC(C:\MMC2\MGUD_CH.NSK,/NC/_N_CH_GD2_ACC
        ,trans,2,2D,10,/ACC/CH/MGUD/)"
End Sub
```

Note

The NSK file is generated in both binary format (*.MAP) and ASCII format (*.NSK).

Copying to the NSK file on the NCDDE server

The files MGUD_NCK.NSK and MGUD_CH.NSK generated in this example can be copied to the NSK file on the NCDDE server NCDDE311.NSK as follows:

```
REM IMPORT ADDITIONAL USER VARIABLES
CALL(MGUD_NCK.NSK)
CALL(MGUD_CH.NSK)
REM
```

Accessing NCK user variables

Using the LIFTOFF_DIST variable, the example below shows how an NCK user variable is read from the NCK.

Example 8–59 Accessing the NCK user variable LIFTOFF_DIST

```
Sub Form_Load ()
        CtlName1.LinkTopic = g_chNCDDEServiceName
        CtlName1.LinkItem = "/acc/nck/mgud/RUECKZUG"
        CtlName1.LinkMode = 2
        CtlName1.LinkRequest
        CtlName1.LinkMode = 0
End Sub
```

Accessing channel-specific user variables

Using the variable BLF_OFFS_X, the example below shows how a channel-specific user variable is read.

Example 8-60 Accessing the channel-specific user variable BLF_OFFS_X

```
Sub Form_Load ()
        CtlName.LinkTopic = g_chNCDDEServiceName
        CtlName.LinkItem = "/acc/ch/mgud/BLF_OFFS_X[u2]" 'for 2nd channel
        CtlName.LinkMode = 2
        CtlName.LinkRequest
        CtlName.LinkMode = 0
End Sub
```

Note

You can find more details on how to create and apply user data in the Installation and Start-up Guide /IAD/ and in the Programming Guide /PA/.

8.14 Online help for variables

Overview

Online help for variables provides support for OEM programmers when selecting and defining data from the NCK area. It is structured like all other help files under WINDOWS and offers the same features. The online help for variables is independent of the HMI OEM package and is stored as a help file named BTSS_GR.HLP (with German texts) in the HLP directory.

Target systems

The area of application of the online help for variables is not restricted to OEM programming of the HMI user interface. It is also suitable for configuring MMC 100 and the NC Var selector in the PLC programming environment.

Scope of functions

The online help for variables offers information on all NCK variables as listed in Chapter 11 and described in more detail in the list book /LIS/.
You can get to the information on a special variable using several description levels.
Starting from the data area with:
Data area Block Variable Example
or in alphabetical order via the block:
Block Variable Example
or using the function **SEARCH (FIND)** to search for key words.
Keywords are:

Short description of the variable	e.g. **Spindle type**
Variable name	e.g. **spindleType variable**
Abbreviation for block	e.g. **SSP** (for spindle state data).

Copying data

You may copy parts from the displayed help topic and insert them in other files. This is especially useful for copying examples from the online help for variables directly into OEM programs. To do so, proceed as follows:
Go to the **Edit** menu
Choose the **Copy** menu option
Select the text you require with the mouse
Click the **Copy** button
Switch to the other application
Paste the text.

Additional functions

Using the online help for variables you can also

- Print topics
- Insert your own comments on each topic
- Define bookmarks for quickly finding the information most frequently needed.

Note

Comments on the online help for variables are stored in the file BTSS_VAR.ANN (ANN is short for annex), the bookmark is located in the file WINHELP.BMK (BMK is short for bookmark) in the WINDOWS directory.

8.15 Troubleshooting

8.15.1 Connection breakdown with NCK/PLC

- Connecting cable
- MPI drivers need to be installed
- Check MMC.INI
- WINSTART.BAT
- S7DPMPI.INI

8.15.2 ...didn’t respond to DDE–Initiate

- Check Link Topic
- Check Link Item
- Has the variable been declared? For PLC access operations in particular, has the data module been declared?

8.15.3 Form Load takes a long time due to creation of Hotlinks

- Use DCTL control
- Create asynchronous Hotlinks

8.15.4 Execute command does not work the first time

Cause

For some commands the NCDDE server expects a connection to the NC to exist already.

Solution

First create a hotlink to an NC variable.

8.16 Determining active bus devices

Open the folder "PG/PC interfaces" in the system controller. In the dialog window that opens, select the active communication interface (e.g. CP5611 (MPI) <Active>) in the "Interface parameter settings used" list and click the "Diagnostics" button. Click the "Test" button in the next dialog box to determine the performance capability of the selected communication interface. Click "Read" to identify the active bus devices.
The active bus devices can also be determined by reading the variable /Nck/Nck/BusState. A 32-bit word is returned in which the bits that are set correspond to an active bus device (Bit0 set = active device at bus address 0, bit1 set = active bus device at bus address 1, and so on). This function is available with MPI and Profibus buses. The 32 "lowest" addresses in the local bus segments are included.

A.3 Abbreviations

Table A-1 Abbreviations

Abbreviation	Meaning
DockPos	Docking position
Host computer	Host computer
FTP	File Transfer Protocol
MMC	Man Machine Communication
NCU	Numerical Control Unit
PLC	Programmable Logic Control
RKS	Computer link software in MMC102, MMC103
RPC	Remote Procedure Call
TCP/IP	Transmission Control Protocol / Internet Protocol
TPS	Transport system
WPC	Workpiece carrier
WZ	Tool

A.4 Error numbers

Table A-2 RPC SINUMERIK error numbers

Error number	Meaning
-70	Appears in log file after expiry of a waiting time after an unsuccessful attempt to send
-97	RPC return value, if an RPC arrives during restart
-98	RPC return value if the internal RPC SINUMERIK job list is full, → wait and then repeat the call
-99	RPC return value, if RPC is not supported. E.g. T_TPS_M is sent to a machine
-100	RPC return value, if the machine name is incorrect.
-110	RPC return value, if the host name is incorrect.
-200	RPC return value if an identical function is already running with R_REPORT_H, if C_TPORDER_M arrives before the previous TPA is completed.
-203	R_REPORT_H with an error after R_NC4WPC_M
-250	R_REPORT_H if the file in the DH server cannot be deleted
-262	R_REPORT_H if the file in the DH server cannot be loaded
-263	R_REPORT_H if the file in the DH server cannot be loaded and selected
-264	R_REPORT_H if the file in the DH server cannot be unloaded
-265	R_REPORT_H if the file in the DH server cannot be selected
-266	R_REPORT_H if the file in the DH server cannot be deselected
-270	R_REPORT_H if the time/date cannot be updated
-271	R_REPORT_H if "set protection level" has not worked.
-272	R_REPORT_H if "reset protection level" has not worked.
-300	RPC return value, if the file cannot be obtained from the host computer.
-301	R_REPORT_H when a long file name is shortened to 8.3
-302	R_REPORT_H when the file data is set with R_DATA_M (SFct=1)
-310	R_REPORT_H with an error in dh_create
-320	R_REPORT_H with an error, when the program is not in data management
-400	RPC return value if the file cannot be transferred to the host computer
-500	R_REPORT_H with DDE connect error for R_DDEDATA_M ()
-510	R_REPORT_H with DDE poke error for R_DDEDATA_M ()
-600	R_REPORT_H with tool request with wrong data structure number
-610	R_REPORT_H with tool request, read error in tool data
-700	R_REPORT_H, if TPS acknowledges a TPA with error.
-800	R_REPORT_H on error of T_VAR_M
-805	R_REPORT_H on error of R_VAR_M
-810	R_REPORT_H with unknown variable set (SCVARSET.INI)
-820	R_REPORT_H on error in variable set
-6003	1. R_REPORT_H after T_DATA_M (SFct=10), name 1 probably wrong
-6020	R_REPORT_M after T_DATA_H (SFct =21 ... 23), if the requested tool could not be found.

■

For notes

I Index

I.1 Keyword index

R

S

T

V

W

■

To
Siemens AG

A&D MC MS
P.O. Box 3180

D-91050 Erlangen

Tel. +49 (0) 180 / 5050 – 222 [Hotline]

Fax +49 (0) 9131 / 98 – 2176 [documentation]
E-mail: motioncontrol.docu@siemens.com

Suggestions

Corrections

For Publication/Manual:

Motion Control Information System

SINUMERIK 840D/840Di/810D
RPC SINUMERIK Computer Link

Manufacturer Documentation

From

Name:

Company/Dept.

Address: _____

Zip code: _____ City: _____

Telephone: _____ / _____

Fax: _____ / _____

Function Manual

Order No.: 6FC5297-6AD61-0BP1
10.05 Edition

Should you come across any printing errors when reading this publication, please notify us on this sheet. Suggestions for improvements are also welcome.

Suggestions and/or corrections

Documentation overview SINUMERIK 840D/840Di/810D (08.2005)

General Documentation

- *SINUMERIK* 840D/810D — Brochure
- *SINUMERIK* 840D/840Di/810D — Catalog Ordering info. NC 60 ***)**
- *Safety Integrated* Application Manual — **Safety Integrated** Application Manual

User Documentation

- *SINUMERIK* 840D/810D/FM-NC — **AutoTurn**
  - Short Guide
  - Programming/ Setup
- *SINUMERIK* 840D/840Di/810D — Operator's Guide **– HT 6**
- *SINUMERIK* 840D/840Di/810D — **Diagnostics Guide *)**
- *SINUMERIK* 840D/840Di/810D — **Operators Guide *)**
  - Short Guide
  - HMI Embedded
  - HMI Advanced

User Documentation

- *SINUMERIK* 840D/840Di/810D — **Program. Guide**
  - Short Guide
  - Fundamentals ***)**
  - Advanced ***)**
  - Cycles
  - Measuring Cycles
  - ISO Turning/Milling
  - Lists **System Variable**
- *SINUMERIK* 840D/810D — Operator's Guide
  - **ManualTurn**
  - Short Guide ManualTurn
  - **ShopMill**
  - Short Guide ShopMill
  - **ShopTurn**
  - Short Guide ShopTurn
- *SINUMERIK* 840Di — **Systemoverview**

Manufacturer/Service Documentation

- *SINUMERIK* — **Configuring** (HW) ***)**
  - 810D
  - 840D
- *SINUMERIK* 840D/840Di/810D — **Operator Components** (HW) ***)**
- *SINUMERIK* 840D/840Di/810D — Description of Functions
  - **ManualTurn**
  - **ShopMill**
  - **ShopTurn**

Manufacturer/Service Documentation

- *SINUMERIK SIMODRIVE* 611D 840D/810D — Description of Functions **Synchronized Actions**
- *SINUMERIK* 840D/840Di/810D — Desription of Functions **Drive Functions*)**
- *SINUMERIK* 840D/840Di/810D — **Desciption of Functions**
  - Basic Machine ***)**
  - Extended Functions
  - Special Functions
- *SINUMERIK* 840D/810D — Desription of Functions **Tool Management**
- *SINUMERIK* 840D/810D — **Configuring Kit** HMI Embedded
- *SINUMERIK* 840D/810D — Description of Functions Operator Interface **OP 030**
- *SINUMERIK* 840D/810D — **MCIS**
  - Computer Link
  - Tool Data Information System
  - NC Data Management
  - NC Data Transfer
  - Tool Data Communication

Manufacturer/Service Documentation

- *SINUMERIK SIMODRIVE* — Description of Functions SINUMERIK **Safety Integrated**
- *SINUMERIK* 840D — Description of Functions **Digitizing**
- *SINUMERIK SIMODRIVE* — **Installation & Start-up Guide. *)**
  - 810D
  - 840D/611D
  - HMI
- *SINUMERIK SIMODRIVE* 840D/840Di 810D 611D — **Lists*)**
- *SINUMERIK SIMODRIVE* 840D 611D — Description of Functions **Linear motor**
- *SINUMERIK SIMODRIVE* 840D 611D — Description of Functions
  - **– Hydraulics Modules**
  - **– Analog Modules**
- *MOTION CONTROL SYSTEMS* — **EMV** Guidelines

Electronic Documentation

- *SINUMERIK SIMODRIVE* 840D/840Di/810D 611, Motors — **DOC ON CD *)** The SINUMERIK System

Manufacturer/Service Documentation

- *SINUMERIK* 840D/840Di/810D — Description of Functions **ISO** Dialects for SINUMERIK
- *SINUMERIK* 840Di — **Manual** (HW + Installation and Start-up)
- *SINUMERIK* 840D/810D — Desription of Functions **Remote Diagnosis**
- *SINUMERIK* 840D/840Di/810D — **Manual @ Event**

***)** These documents are a minimum requirement