ECKELMANN

E•CONTROL

Interface NCRX

Description of Function

Author: M. Funk

First edition: 10/02/2006

File name: NCRX_Schnittstelle_V1.0_de.doc

Version: 1.0

Release:	Roland Simon Name in block letters	05/24/2007 Date	signed Simon Signature

Modification protocol

Version	Date	Person in charge	Modification: Chapter/contents
1.0	10/02/06	M. Funk	First edition

Table of contents

1 Introduction

ActiveX Control NCRXControl.ocx is the interface between the HMI applications and the ECKELMANN CNC controllers E•CNC55, E•PNC55, E•ENC55 and E•ExC66. The interface is based on the MMICTRL.DLL (see also description of function E•CONTROL/MMICTRL.DLL).

1.1 Prerequisites

Prerequisites are the drivers that are also installed in the StdHMI:

- **Netconf.exe** For the configuration of the IP address of an E•ENC55 or E•CNC55
- **MMIGTWAY.exe** For the communication with an E•xNC55
- **GtwConf.exe** For the configuration of the MMIGateway
- **IPCOM.DLL** For the communication with an E•xNC55
- **MMICTRL.DLL** For the communication with an E•xNC55

Moreover, please install:

- **NCRXControl.ocx** The ActiveX Control

  The license file must be copied only to the development computer.
- **NCRXControl.lic** The developer's license file must not be handed over to third parties.

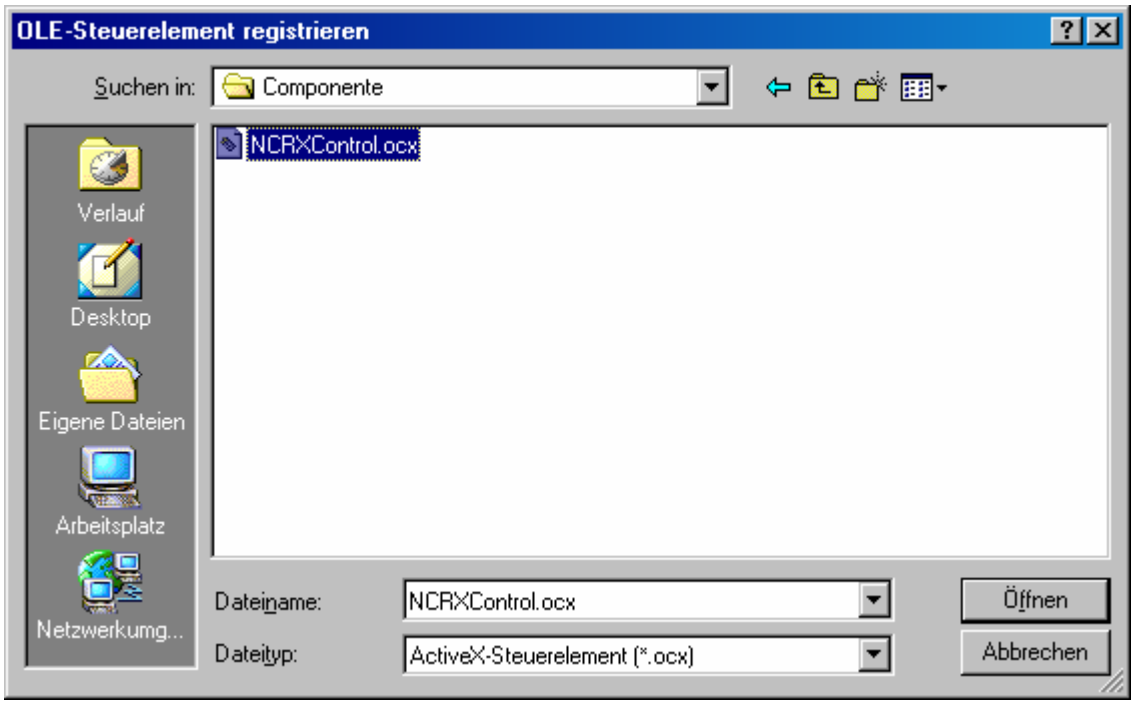
First, ActiveX Control is to be registered in Windows.

Call: **RegSvr32 NCRXControl.ocx**

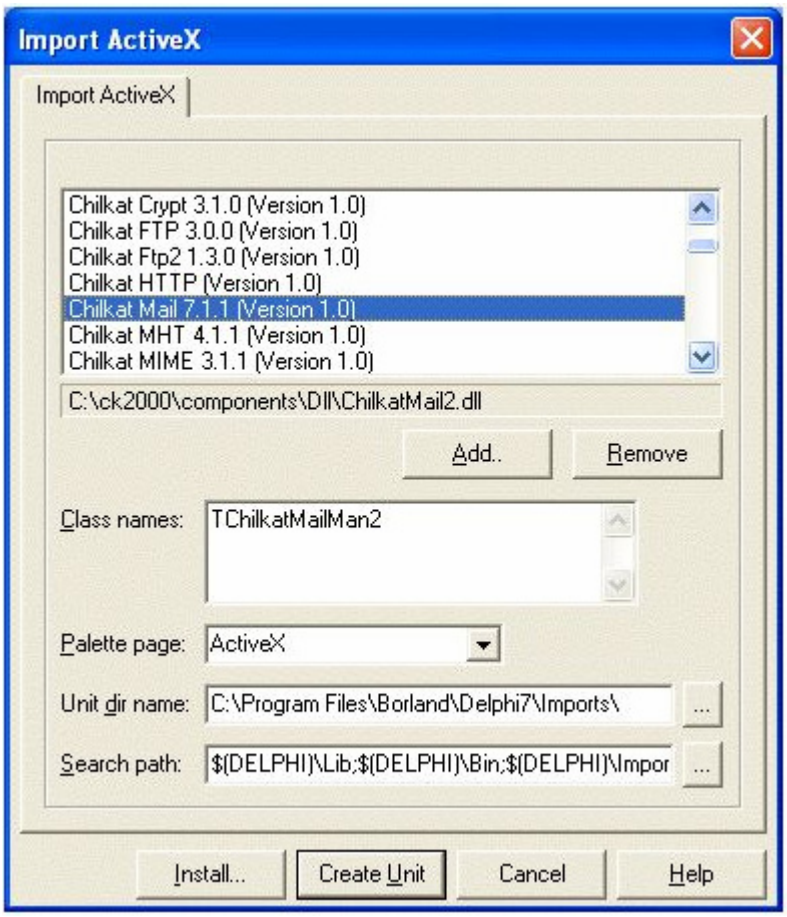
ActiveX Control can be deinstalled by means of **RegSvr32 /u NCRXControl.ocx**.

1.1.1 Delphi7

- Menu *Components:* Add *Active X*
- If Active X NCRCControl is not yet registered, select file NCRXControl1.ocx by clicking on *Add...* .

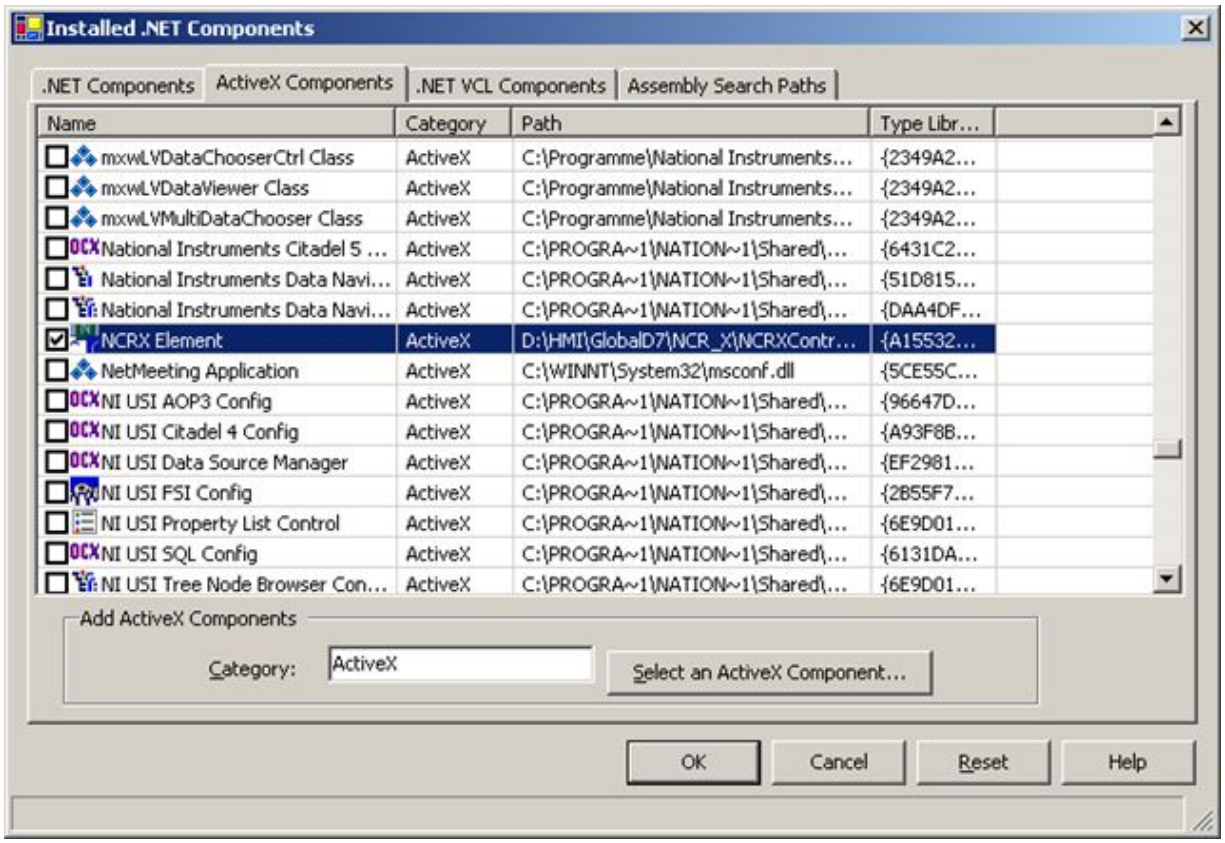


- Subsequently click on *Install...* .
- If no package is open, a new package Project is generated and the new ActiveX component is generated in tab ActiveX.



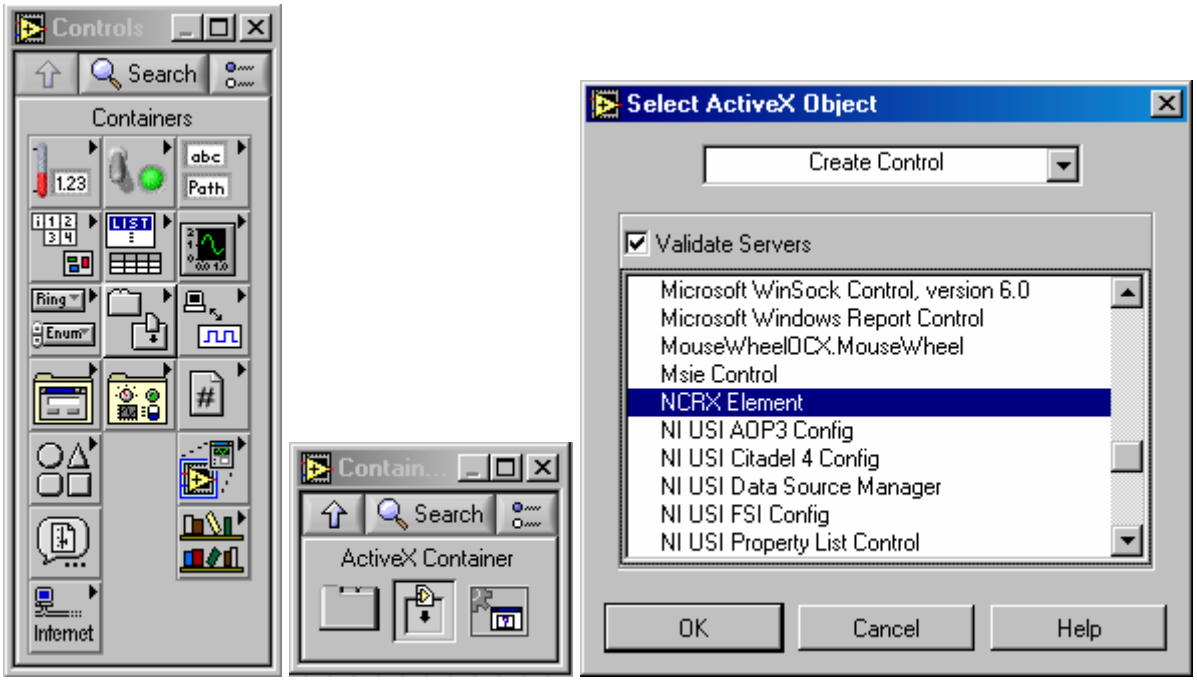
1.1.2 Borland 2006 Studio

- Menu *Components: Installed .NET components*
- Select tab ActiveX Components
- If the Active X NCRX element is not yet registered, select by clicking on button *Select an ActiveX Component...,* add file NCRXControl.ocx
- Select the check box in the list and ActiveX appears in the tool palette

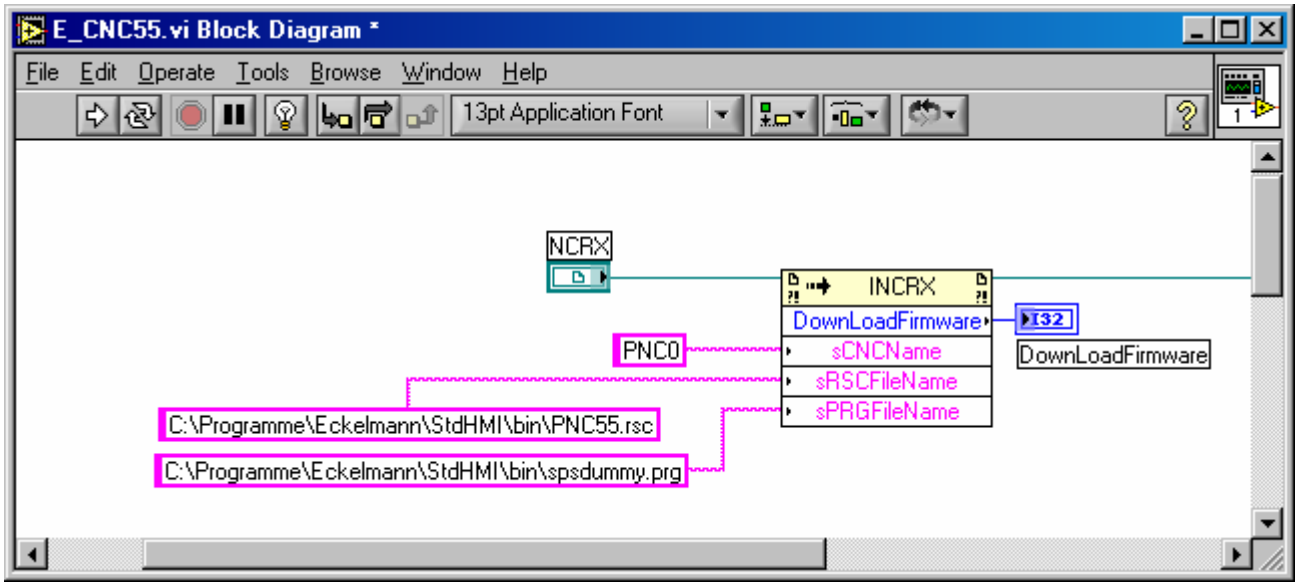


1.1.3 Labview7

- Select  in Controls *Containers,* subsequently select  in ActiveX Container and put it on the front panel.
- Select *Insert ActiveX Object...* in the shortcut menu by means of the right mouse button and select NCRX Element in dialog *Select ActiveX Object*.



- In the block diagram, access can be made to the ActiveX Element by means of Create Method or Create Property.



1.1.4 Visual Basic

- Menu *Project – Components*

  A dialog with Controls is displayed. Please select the NCRX Element. Subsequently, NCRX should be available with the components.

2 Interface

The interface encapsulates features of the dual port RAM (DPR) (see also Manual E•xNC55: Interface HMI <-> NCR) and methods of mmictrl.dll (see also description of function of E•CONTROL: MMICTRL.DLL). Depending on the controller type (E•ENC55, E•ELC55, E•EMC55, E•ENC66), the DPR is subdivided differently. The controller type can be determined via the firmware version.

2.1 Characteristics

2.1.1 ComponentVersion

- **ComponentVersion: WideString;**

  Read Only: Version of the component

2.1.2 NCRVersion

- **NCRVersion: WideString;**

  Read Only: Version of the NC firmware

2.1.3 PLCVersion

- **PLCVersion: WideString;**

  Read Only: Version of the PLC program; **only,** if the PLC supports a version WideString!

2.1.4 OperatingMode

- **OperatingMode Integer;**

  R/W Current operating mode: 1= Setup, 2 = Automatic, 3= Programming, 128 = Diagnosis

2.1.5 SubOperatingMode

- **SubOperatingMode: Integer;**

  R/W Current sub-operating mode (PLC-dependent!)

2.1.6 StepSize

- **StepSize: Double;**

  R/W: Increment for step travel see StepAxisPos or StepAxisNeg

2.1.7 NCConnectionName

- **NCName: WideString;**

  Read Only: Name of controller specified in MMIGateWay.

2.1.8 TraceActive

- **TraceActive: Integer;**

  Traces within the ActiveX component can be activated. The traces are stored in the HMI.LOG. Each bit is assigned to a trace. Several traces can be activated via the combination of the bits. Up to now, the following traces have been defined:

  - Trace 01 Write everything received via Postmessage
  - Trace 02 Messages to the NC
  - Trace 04 Messages from the NC
  - Trace 08 Download of a file: Din, Zyk, Mk, Wtk, Wsk, etc.
  - Trace 16 Upload of a file: Mk, Wtk, Wsk, etc.

2.1.9 ErrorHistoryRange

- **ErrorHistoryRange: Integer;**

  Number of days that are to be stored in the fault file. Previous entries are removed; see also ErrorlogActive.

2.1.10 ErrorlogActive

- **ErrorlogActive: WordBool;**

  Error logbook is activated. All errors are stored in a file named Errorlog.txt with error number and additional information. The error text is not stored, since it is not available; see also ErrorHistoryRange, OnErrorMsg, LogPath and ShowOwnError.

2.1.11 LogPath

- **LogPath: WideString**

  Path in which the component is allowed to write files. This includes the error file errorlog.txt and the trace file HMI.log.

2.2 Methods

2.2.1 Files to the NC

2.2.1.1 DownloadFirmware

- **DownloadFirmware(sCNCName, sRSCFileName, SPRGFileName: WideString): Integer;**

  sCNCNAME: Name of the controller (e. g. CNC1)

  sRSCFileName: Name of the firmware file (e. g. PNC55.RSC)

  sPRGFileName: Name of the PLC program (e. g. SPSdummy.PRG)

Return: 0: Newly loaded

10 : was already loaded. Controller ready

Other errors: -10 error during loading of MMICtrl.DLL

-11 error during loading of the firmware

via Hex: 20 000 000 an error message is transmitted via OnErrorMsg

Generates a connection with the controller and loads the firmware and the PLC program, if necessary. Calling is always required at the start of the application even if the firmware is already running.

2.2.1.2 DownloadMK

- **DownloadMK(sMKFileName:WideString):Integer;**

  sMKFileName: Name of the machine constant file

Return: 0: Transmission successful.

Otherwise error

Loads machine constant file in the controller; see also manual Configuration Instructions.

2.2.1.3 DownloadDIN

- **DownloadDIN (sDinFileName:WideString):Integer;**

  SDINFileName: Name of the DIN program

Return: 0: Transmission successful.

Otherwise error

Loads DIN program in the controller. In case of program number < 8000 , the number is input in P512, in order to enter the program as start program; see also: StartProgram, StopProgram.

2.2.1.4 DownloadDINOnline

- **DownloadDINOnline(sDinFileName:WideString):Integer;**

  sDINFileName: Name of the DIN program

Return: 0: Transmission successful.

Otherwise error

Loads Online program in the controller; see also: StartProgram, StopProgram, BreakAction.

2.2.1.5 DownloadWTK

- **DownloadWTK (sWTKFileName:WideString):Integer;**

  sWTKFileName: Name of the file with tool correction data

Return: 0: Transmission successful.

Otherwise error

Loads tool compensation file in the controller.

2.2.1.6 DownloadWSK

- **DownloadWSK (sWSKFileName:WideString):Integer;**

  sWSKFileName: Name of file with workpiece correction data

Return: 0: Transmission successful.

Otherwise error

Loads file with workpiece correction data in the controller.

2.2.1.7 DownloadKor3D

- **DownloadKor3D(sKor3DFileName:WideString):Integer;**

  sKor3DFileName: Name of the 3D correction file (*.kor)

Return: 0: Transmission successful.

Otherwise error

Loads the 3D correction file for the correction of an axis as a function of another axis. Up to 3 files are possible; see also manual Programming Instructions G233.

2.2.1.8 DownloadKorAxis

- **DownloadKorAxis(sKorAxisFileName:WideString):Integer;**

  sKorAxisFileName: Name of the leadscrew correction file (*.kor)

Return: 0: Transmission successful.

Otherwise error

Loads leadscrew correction file in the controller. Leadscrew is corrected by means of the axis length.

2.2.1.9 BreakAction

- **BreakAction;**

  Current file transmission is interrupted.

2.2.1.10 Error codes download

- -1 Other file transmission is active
- -2 File not found
- -3 Controller not initialized (download firmware not implemented successfully)
- -4 TimeOut during transmission
- -6 Error occurring
- -20 Transmission interrupted by the user

2.2.2 Loading of files from the NC

2.2.2.1 UploadMK

- **UploadMK(sMKFileName :WideString):Integer;**

  sMKFileName: Name of the machine constant file

Return: 0: Transmission successful.

Otherwise error

Loads machine constant file from controller to HMI.

2.2.2.2 UploadWTK

- **UploadWTK (sWTKFileName:WideString):Integer;**

  sWTKFileName: Name of the file with tool correction data (*.wtk)

Return: 0: Transmission successful.

Otherwise error

Loads file with tool correction data from controller to HMI.

2.2.2.3 UploadWSK

- **UploadWSK (sWSKFileName:WideString):Integer;**

  sWSKFileName: Name of file with workpiece correction data (*.wsk)

Return: 0: Transmission successful.

Otherwise error

Loads file with workpiece correction data from the controller to the HMI.

2.2.3 P-Fields

2.2.3.1 WritePField

- **WritePField(iFieldNo:longint;dValue:Double):Integer;**

  iFieldNo: Index of the P-field which is to be written

  dValue: Value to be written

Return: 0 OK

Writing of a value in a P-field

2.2.3.2 WritePFieldArray

- **WritePFieldArray(awFieldNo: count: Integer; OleVariant; adValue: OleVariant);**

  awFieldNo: List of indices of P-fields which are to be written (Word)

  adValue: List of values to be written (Double)

Return: 0 OK

Writing of several values in P-fields

2.2.3.3 ReadPField

- **ReadPField(iFieldNo:longint;var dValue:Double):Integer;**

  iFieldNo: Index of the P-field which is to be read

  out **dValue**: Read value

Return: 0 OK

Reads the value of a P-field. A request is transmitted and the controller response is waited for. Duration may be up to 20 ms. Avoid the cyclic call of this function.

2.2.3.4 ReadPFieldArray

- **ReadPFieldArray(iCount :Integer;**
  AwIndex :OleVariant;
  var adValue :OleVariant: Integer;

  iCount: Number of values which are to be read (Word)

  awIndex: List of indices of P-fields which are to be read (Double)

  out **adValue:** List of read values

Return: 0 OK

Reads a list of P-field values the indices of which are transmitted. A request is transmitted and the controller response is waited for. Duration may be up to 20 ms. Avoid the cyclic call of this function.

2.2.3.5 ReadPFieldArrayIndex

- **ReadPFieldArray(iFieldNo :longint;**
  iCount :Integer;
  var adValue:OleVariant: Integer;

  iFieldNo: Index of the first P-field which is to be read

  iCount: Number of values which are to be read

  out **adValue:** Read values

Return: 0 OK

Reads a sequential list of P-field values starting with the specified index. A request is transmitted and the controller response is waited for. Duration may be up to 20 ms. Avoid the cyclic call of this function.

2.2.3.6 SetFastPField

- **SetFastPField(Index: Integer; iPFieldNo: Integer);**

  In **Index:** Index for fast P-field accesses [0..31]

  In **iPFieldNo:** Number of the P-field

P-fields are copied in the DPR in order to have fast access to them. The value of the P-field can be read by means of ReadFastpField.

2.2.3.7 ReadFastPField

- **ReadFastPField(Index: Integer; out iPFieldNo: Integer; out Value: Double): long**;

  In: **Index:** Index for fast P-field accesses [0..31] in case of E•ENC55

  Out: **iPfieldNo:** Number of the P-field

  Out: **Value:** Value of the P-field

Return: 0: OK, -2: Index outside the permitted range. Fast reading of a P-field initialized with SetFastpField.

2.2.3.8 Error codes P-field

- -1 Other P-field transmission is active
- -3 Controller not initialized (download firmware not implemented successfully)

2.2.4 Messages to the NC

2.2.4.1 SendPLCMsg

- **SendPLCMsg(iID1, iID2, iIndex: Byte; iLen: Smallint; sData: OleVariant):Integer;**

  iID1: First identifier of message (255 reserved for ECKELMANN messages) (corresponds to SB1)

  iID2: Second identifier of message is freely selectable (corresponds to SB2)

  iIndex: The index can be used for a counter in order to be able to assign a received response to a request message.

  iLen: Length of the message buffer

  aData: Useful data, max. 512 byte data buffer

Return: 0: OK –1 error (e. g. more than 512 byte in data buffer). Transmission of a message to the PLC;

see also OnPLCMsg.

2.2.4.2 AcknAllErrors

- **AcknAllErrors:boolean;**

Return: true message was transmitted. All available errors are acknowledged.

2.2.4.3 StartProgram

- **StartProgram;**

Starts the DIN program the number of which is entered in P512.

2.2.4.4 StopProgram

- **StopProgram;**

Stops the running DIN program.

2.2.4.5 SendSingleBlock

- **SendSingleBlock(sGCode: WideString);**

  sGCode: WideString with single set (e. g. G0 X100)

Transmits a single set to the controller without waiting time.

2.2.4.6 DisConnect

The controller connection is disconnected. In some programming languages it may happen that the Active X component is destroyed prior to the termination of communication. An access violation occurs. In this case it is recommended to call DisConnect prior to close the application.

2.2.5 Access Dual Port RAM (DPR)

The subdivision of the DPR depends on the controller. The general methods have direct access to the memory. The special method should possibly be used, e. g. for the reading of the axis position, in order to make a use of the software for following controller generations possible.

2.2.5.1 GetDprDouble

- **GetDprDouble(iAddress: Smallint): Double;**

  iAddress: Offset to the DPR pointer in bytes

Return: Value of the DPR of the address in double format (8 bytes)

Reading of a double value from the DPR. Subdivision of the DPR depends on the controller. Special access routines should possibly be used.

2.2.5.2 GetDprWord

- **GetDprWord(iAddress: Smallint): Smallint;**

  iAddress: Offset to the DPR pointer in bytes

Return: Value of the DPR of the address in word format (2 bytes)

Reading of a word value from the DPR. Subdivision of the DPR depends on the controller. Special access routines should possibly be used.

2.2.5.3 GetDprByte

- **GetDprWord(iAddress: Smallint): Byte;**

  iAddress: Offset to the DPR pointer in bytes

Return: Value of the DPR of the address in byte

Reading of a word value from the DPR. Subdivision of the DPR depends on the controller. Special access routines should possibly be used.

2.2.5.4 SetDprDouble

- **SetDprDouble(iAddress: Smallint; dValue: Double): Integer;**

  iAddress: Offset to the DPR pointer in bytes

  dValue: Value of the DPR of the address in double format (8 bytes)

Return: 0: Value was written. Writing of a double value in the DPR.

Direct writing in the DPR memory is made without checking. The programmer is to safeguard that writing is made only in permitted areas.

2.2.5.5 SetDprWord

- **SetDprWord (iAddress: Smallint; iValue: Smallint): Integer;**

  iAddress: Offset to the DPR pointer in bytes

  iValue: Value in the DPR to address in word format (2 bytes)

Return: 0: Value was written. Writing of a word value in the DPR.

Direct writing in the DPR memory is made without checking. The programmer is to safeguard that writing is made only in permitted areas.

2.2.5.6 SetDprByte

- **SetDprWord (iAddress: Smallint; iValue: Byte): Integer;**

  iAddress: Offset to the DPR pointer in bytes

  iValue: Value of the DPR of the address in byte

Return: 0: Value was written. Writing of a byte value in the DPR.

Direct writing in the DPR memory is made without checking. The programmer is to safeguard that writing is made only in permitted areas.

2.2.5.7 GetAxisPos

- **GetAxisPos (iAxId: Smallint): Double;**

  iAxID: Number of axis

Return: Actual position of the requested axis

2.2.5.8 GetAxisPosReal

- **GetAxisPosReal (iAxId: Smallint): Double;**

  iAxID: Number of axis

Return: Actual position of the requested axis referred to the basic offset

2.2.5.9 GetAxisOffset

- **GetAxisOffset (iAxId: Smallint): Double;**

  iAxID: Number of axis

Return: Offset versus S0 of the requested axis.

2.2.5.10 GetAxisLetter

- **GetAxisLetter(iAxId: Smallint): uChar;**

  iAxID: Number of axis

Return: Letter of requested axis.

2.2.5.11 IsAxisReferenced

- **IsAxisReferenced (iAxId : Smallint): Integer;**

  iAxID: Number of axis

Return: 1 if axis was referenced otherwise 0.

2.2.5.12 GetActiveT

- **GetActiveT (iChanID: Byte): Byte;**

  iChanID: Number of channel

Return: Number of the current T system of coordinates.

2.2.5.13 GetActiveS

- **GetActiveS (iChanID: Byte): Byte;**

  iChanID: Number of channel

Return: Number of the current S system of coordinates.

2.2.5.14 GetCurrentSpeed

- **GetCurrentSpeed(iChanID: Byte): Double;**

  iChanID: Number of channel

Return: Current path speed in input units per minute.

2.2.5.15 GetProgState

- **GetProgState(iChanID: Byte; iProgNo, iBlockNo, iLogSatzNo, iLine: Integer);**

  iChanID: Number of channel. Normally, channel 0 is inquired.

  Out **iProgNo**: Current program number

  Out **iBlockno**: Number of current set: N

  Out **iLogBlockNo**: Logic number of the current set

  Out **iLine**: Current line number

Combines the previous 4 inquiries to a single inquiry.

2.2.5.16 GetLastMFunc

- **GetLastMFunc(iChanID: Byte): Smallint;**

  iChanID: Number of channel

Return: Number of the last M-function.

2.2.5.17 GetLastHFunc

- **GetLastHFunc (iChanID: Byte): Smallint;**

  iChanID: Number of channel

Return: Number of the last H-function.

2.2.5.18 GetChanState

- **GetChanState(iChanID: Byte): Byte;**

  iChanID: Number of channel

Return: 0..13 (0=Idle, 1=RUN, 2=Brake, .. 12=Error, 13= Compensation).

2.2.5.19 StartJogAxisPositive

- **StartJogAxisPositive(iAxId: Byte);**

  iAxID: Number of axis

Axis moves in positive direction until StopAxis(iAxId) is called; see also StopAxis, StartJogAxisNegative.

2.2.5.20 StartJogAxisNegative

- **StartJogAxisNegative(iAxId: Byte);**

  iAxID: Number of axis

Axis moves in negative direction until StopAxis(iAxId) is called; see also StopAxis, StartJogAxisPositive.

2.2.5.21 StopAxis

- **StopAxis(iAxId: Byte);**

  iAxID: Number of axis

Stops the axis; see also StartJogAxisPositive, StartJogAxisNegative.

2.2.5.22 StepAxisPositive

- **StepAxisPos(iAxId: Byte);**

  iAxID: Number of axis

Moves the specified axis in positive direction by the distance specified by StepSize; see also StepSize, StepAxisNegative, StopAxis.

2.2.5.23 StepAxisNegtive

- **StepAxisNeg(iAxID: Byte);**

  iAxID: Number of axis

Moves the specified axis in negative direction by the distance specified by StepSize; see also StepSize, StepAxisPositive, StopAxis.

2.2.6 General methods

2.2.6.1 ShowOwnError

- **ShowOwnError (wError: Integer; ucKlasse: Byte; sText: WideString);**

  wError: Number of error; becomes 1.15.wError

  ucKlasse: Weighing of error; weighing has, however, no influence on the controller

  sText: Additional information regarding the error

Error is transmitted to the error processing routine of the Active X component. The error is displayed in the error logbook and triggers the error event. The call has no influence on the behavior of the controller; see also OnErrorMsg.

2.2.6.2 MmiTraceMessage

- **mmiTraceMessage (lTrace: Integer; sTrace: WideString);**

  lTrace: Number of trace: Only one bit is to be used here which can be set via TraceActive.

  sTrace: WideString to trace

Application-specific traces can be added to the component-specific traces; see also TraceActive.

2.3 Events

2.3.1 OnPLCMsg

- **OnPLCMsg(Sender: TObject; iID1, iID2, Index: Byte; aData: OleVariant)**

  Sender: Calling object

  iID1: Message identifier 1 (corresponds to SB1)

  iID2: Message identifier 2 (corresponds to SB2)

  iIndex: The index can be used for a counter in order to be able to assign a received response to a request message.

  aData: Useful data, max. 512 bytes

Receipt of a message sent by the PLC; see also SendPLCMsg.

2.3.2 OnErrorMsg

- **OnErrorMsg(Sender; iWeight:Integer; sModul, sMsg, sInfo: WideString);**

  Sender: Calling object

  iWeight: Weighing of the error 0...5

  sModul: Module which caused the error

  sMessage: WideString with error number (e. g. 2.3.102)

  sInfo: Additional information; the text is to be determined via the subWideString error number

Errors of NC, PLC or interface (e. g. Gateway) are indicated:

- Texts for 1.X.Y are included in mmi_fehl.DB
- Texts for 2.X.Y are included in ncr_fehl.DB (for X without 10)
- Texts for 2.10.Y are included in SPS_fehl.DB and are prepared by the PLC programmer.

See also AcknAllErrors.

2.3.3 OnTransferStateMsg

- **OnTransferStateMsg(Sender:TObject; sProgress, sMessage:WideString; iProgress: Integer);**

  Sender: Calling object

  sProgress: File name, relative progress in % and absolute progress in byte or KByte

  sMessage: In case of firmware download of additional information

  iProgress: Progress in %

Progress message in case of transfer of files to the controller.

2.3.4 OnStateMsg

- **OnStateMsg(Sender:TObject; sMessage: WideString, iTime : Integer);**

  Sender: Calling object

  sMessage: Text of messages

  iTime: Time after which the display is allowed to disappear

Message initiated by the controller; e. g. via G253 F="ABC".

2.4 Valid Types

Delphi type	IDL type	Variant type	Automation	Description
Smallint	short	VT_I2	Yes	2-byte signed int
Integer	long	VT_i4	Yes	4-byte signed integer
Single	single	VT_R4	Yes	4-byte real
Double	double	VT_R8	Yes	8-byte real
Currency	CURRENCY	VT_CY	Yes	currency
TDateTime	DATE	VT_DATE	Yes	date
WideString	BSTR	VT_BSTR	Yes	binary string
IDispatch	IDispatch	VT_DISPATCH	Yes	pointer to IDispatch
SCODE	SCODE	VT_ERROR	Yes	OLE Error Code
WordBool	VARIANT_BOOL	VT_BOOL	Yes	true = -1, false = 0
OleVariant	VARIANT	VT_VARIANT	Yes	OLE Variant
IUnknown	IUnknown	VT_UNKNOWN	Yes	pointer to IUnknown
Shortint	byte	VT_I1	No	1 byte signed integer
Byte	unsigned char	VT_UI1	Yes	1 byte unsigned integer
Word	unsigned short	VT_UI2	No*	2 byte unsigned integer
UNIT	unsigned long	VT_UI4	No*	4 byte unsigned integer
Int64	_int64	VT_I8	No	8 byte signed real
LargeUInt	uint64	VT_UI8	No	8 byte unsigned real
SysInt	int	VT_INT	No*	system dependant integer
SysUInt	unsigned int	VT_UINT	No*	system dependant unsigned integer
HResult	HRESULT	VT_HRESULT	No	32 bit error code
Pointer	Unsigned int	VT_VOID	No	untyped pointer
SafeArray	SAFEARRAY	VT_SAFEARRAY	No	OLE Safe Array
PChar	LPSTR	VT_LPSTR	No	a pointer to char
PWideChar	LPWSTR	VT_LPWSTR	No	a pointer to widechar

* The types Word, UINT, SYSINT and SYSUINT may allow automation with certain applications.

3 Annex

3.1 System files

These files are included in the STDHMI installation.

File name	Path	Development system	Target system	Description
Netconf.exe	System32	X	X	For the configuration of the IP address of an E•ENC55 or E•CNC55
MMIGTWAY.exe	System32	X	X	For the communication with an E•xNC55
GtwConf.exe		X	X	For the configuration of the MMIGtway
IPCOM.DLL	System32	X	X	For the communication with an E•xNC55
MMICTRL.DLL	System32	X	X	For the communication with an E•xNC55
NCRXControl.ocx	System32	X	X	The ActiveX Control
NCRXControl.lic	System32	X		Developer of license file: The file must not be handed over to third parties!

3.2 Component

After the purchase of the components these files are submitted to you.

File name	Path	Development system	Target system	Description
NCRXControl.ocx	System32	X	X	The ActiveX Control
NCRXControl.lic	System32	X		Developer of license file: The file must not be handed over to third parties!

3.3 Language files

The error messages are included in these files. The error messages are to be evaluated by the application programmer upon receipt of an error message. The files are included in the StdHMI intallation.

File name	Language	Description
MMI_Fehl_DE.DB	German	Error messages 1.X.Y HMI/driver
NCR_Fehl_DE.DB	German	Error messages 2.X.Y NCR
SPS_Fehl_DE.DB	German	Error messages 2.10.Y to be adapted by the PLC programmer
MMI_Fehl_EN.DB	English	Error messages HMI/driver
NCR_Fehl_EN.DB	English	Error messages NCR/driver
SPS_Fehl_EN.DB	English	Error messages 2.10.Y to be adapted by the PLC programmer

3.4 Examples

With the component, example projects are submitted in C# and Delphi7. The most important mechanisms are implemented there on an exemplary basis. These example applications are not released for the operation of a real machine.