

WinTools

Open Communication with Control Systems with Ethernet Capability Software Manual

Version

101

WinTools

Open Communication with Control Systems with Ethernet Capability

1070 072 274-101 (01.12) GB

© 1999 - 2001

All rights reserved by Robert Bosch GmbH,
including applications for protective rights.
Reproduction or distribution by any means subject to our prior written
permission.

Discretionary charge € 6.00

Table of contents

1	Safety Instructions	1-1
1.1	Proper use	1-1
1.2	Qualified personnel	1-1
1.3	Safety instructions in this manual	1-2
1.4	Safety instructions for the described product	1-2
1.5	Trademarks	1-2
2	General	2-1
2.1	BoschCOM.DLL	2-1
2.2	Bosch PlcServer	2-1
2.3	Bosch OPC	2-2
3	Licensing and base configuration	3-1
3.1	General principles	3-1
3.2	Installation	3-1
3.3	License	3-2
3.4	Licensing	3-3
3.4.1	Test license	3-3
3.4.2	Displays	3-3
3.5	Import	3-3
3.5.1	Export	3-4
3.5.2	Apply	3-4
3.5.3	Transfer from hard disk into Hardlock	3-5
3.5.4	Apply for Hardlock	3-5
4	Overview of functions BoschCOM	4-1
4.1	Error handling	4-1
4.2	Time properties	4-2
4.3	Introductory example	4-3
4.4	Creating components	4-4
4.4.1	Visual Basic 6.0	4-4
4.4.2	Visual Basic for Applications	4-5
4.4.3	Delphi	4-5
4.4.4	C, C++ (direct access)	4-5
4.4.5	Mfc applications	4-6
4.5	Optional parameters	4-7
4.5.1	TargetIPAddr	4-7
4.5.2	TargetPort	4-7
4.5.3	HostPort	4-7
4.5.4	BlockAddr	4-8
4.5.5	TimeToWait	4-8
4.5.6	MaxTime	4-8
4.5.7	RepeatCounter	4-8
4.5.8	RecvDelay	4-8
4.5.9	Examples	4-9
4.6	Connection functions	4-10
4.6.1	Init	4-10
4.6.2	Deinit	4-11
4.6.3	OpenChannel	4-11
4.6.4	InitAndOpenChannel	4-11
4.6.5	CloseChannel	4-11
4.6.6	Login	4-11
4.6.7	Examples	4-13
4.7	DB access functions	4-13
4.7.1	DM_String	4-13
4.7.2	DM_Dword	4-14

4.7.3	DM_Word	4-14
4.7.4	DM_Real	4-14
4.7.5	DM_LReal	4-14
4.7.6	DM	4-14
4.7.7	ReadDM	4-15
4.7.8	PutMaskedDM	4-15
4.8	Marker access functions	4-15
4.8.1	M_Dword	4-15
4.8.2	M_Word	4-16
4.8.3	M_Real	4-16
4.8.4	M_LReal	4-16
4.8.5	M	4-16
4.8.6	ReadM	4-17
4.8.7	PutMaskedM	4-17
4.9	Data field access functions	4-18
4.9.1	DF_Dword	4-18
4.9.2	DF_Word	4-18
4.9.3	DF_Real	4-18
4.9.4	DF_LReal	4-18
4.9.5	DF	4-19
4.9.6	ReadDF	4-19
4.9.7	PutMaskedDF	4-19
4.10	Symbolic access	4-19
4.10.1	SymData	4-19
5	Examples BoschCOM	5-1
5.1	Visual Basic	5-1
5.2	Visual C++	5-4
5.3	MFC	5-5
5.4	Delphi	5-6
5.5	Excel	5-7
5.6	HTML and VB Script	5-9
6	BoschPlcServer	6-1
6.1	General	6-1
6.2	Licensing	6-1
6.3	Functions	6-1
7	BoschOPC	7-1
7.1	General	7-1
7.2	Invocation	7-1
7.2.1	Hardlock/Softlicense	7-1
7.2.2	Destination Controller IP Address and Port	7-1
7.2.3	Start via OPC Client	7-1
7.2.4	User Interface	7-2
7.3	OPC Data	7-2
7.3.1	Limits	7-2
7.3.2	Data	7-2

1 Safety Instructions

Before you start working with the module / software, we recommend that you thoroughly familiarize yourself with the contents of this manual. Keep this manual in a place where it is always accessible to all users.

1.1 Proper use

This instruction manual presents a comprehensive set of instructions and information required for the standard operation of the described products.

The products described hereunder were developed, manufactured, tested and documented in accordance with the relevant safety standards. In standard operation, and provided that the specifications and safety instructions relating to the project phase, installation and correct operation of the product are followed, there should arise no risk of danger to personnel or property.

1.2 Qualified personnel

This instruction manual is designed for specially trained personnel. The relevant requirements are based on the job specifications as outlined by the ZVEI and VDMA professional associations in Germany. Please refer to the following German-Language publication:

Weiterbildung in der Automatisierungstechnik
Publishers: ZVEI and VDMA Maschinenbau Verlag
Postfach 71 08 64
60498 Frankfurt/Germany

Interventions in the hardware and software of our products not described in this instruction manual may only be performed by our skilled personnel.

Unqualified interventions in the hardware or software or non-compliance with the warnings listed in this instruction manual or indicated on the product may result in serious personal injury or damage to property.

Installation and maintenance of the products described hereunder is the exclusive domain of trained electricians as per IEV 826-09-01 (modified) who are familiar with the contents of this manual.

Trained electricians are persons of whom the following is true:

- They are capable, due to their professional training, skills and expertise, and based upon their knowledge of and familiarity with applicable technical standards, of assessing the work to be carried out, and of recognizing possible dangers.
- They possess, subsequent to several years' experience in a comparable field of endeavour, a level of knowledge and skills that may be deemed commensurate with that attainable in the course of a formal professional education.

With regard to the foregoing, please read the information about our comprehensive training program. The professional staff at our training centre will be pleased to provide detailed information. You may contact the centre by telephone at (+49) 6062 78-258.

1.3 Safety instructions in this manual

**DANGER**

This symbol is used wherever insufficient or lacking observance of this instruction can result in **personal injury**.

**CAUTION**

This symbol is used wherever insufficient or lacking observance of instructions can result in **damage to equipment or data files**.

⇒ This symbol is used to alert the user to an item of special interest.

1.4 Safety instructions for the described product

**DANGER**

Danger to persons and equipment!
Test every new program before operating the system!

**DANGER**

Retrofits or modifications may interfere with the safety of the products described hereunder!

The consequences may be severe personal injury or damage to equipment or the environment. Therefore, any system retrofitting or modification utilizing equipment components from other manufacturers will require express approval by Bosch.

1.5 Trademarks

All trademarks referring to software that is installed on Bosch products when shipped from the factory represent the property of their respective owners.

At the time of shipment from the factory, all installed software is protected by copyright. Software may therefore be duplicated only with the prior permission of the respective manufacturer or copyright owner.

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

2 General

The WinTools software package consists of three parts:

- COM server software (BoschCOM.DLL, for detailed description see chapters 4 and 5)
- Event utility software (BoschPlcServer, for detailed description see chapter 6)
- OPC Server for Bosch SoftSPS and CL550 (BoschOPC, for detailed description see chapter 7).

All three parts are installed and licensed via the WinTools user interface. After installation and licensing each part can be invoked independently.

Following this introductory chapter, chapter 3 first describes the installation and licensing of WinTools. The subsequent chapters provide further information on the individual parts as shown above.

2.1 BoschCOM.DLL

Equipping Bosch control systems with Ethernet interfaces enables open communication to the control systems.

This communication should not be bound to a programming language, rather it should be possible to make contact with a Bosch control system from Excel, Access and also based on HTML.

These requirements are met by COM, the 'Component Object Mode' from Microsoft. This is why COM was selected as the basis for our solution.

The BoschCOM.DLL is implemented as a COM server and it provides all the interfaces (ports) for efficient access to Bosch control systems with Ethernet capability by Basic, VBA, VBScript, Jscript but also C++ and Delphi.

The BoschCOM.DLL creates the possibility to link Bosch control systems (CL400/CL500 with COM-E, CL200 with COM-2E and PCL) to PCs with Windows-based software.

The range of functions enables reading and writing of PLC operands in the data module, data field and marker areas, as well as queries of control system status and peripherals.

For the communication, in chapter 5 this manual provides examples in Excel, Basic, Delphi and C++.

2.2 Bosch PlcServer

The event utility BoschPlcServer makes it possible to query data of a control system on the computer. This utility is implemented within the framework of COM/DCOM, i.e. as 'Connection Point'. That means that the message can be picked up or not; there is no handshake. If a number of messages arrive on the BoschCOM.DLL, they are routed in the order of arrival. If a handshake is required, this must take place in the PLC program.

It is possible to both start events from control systems and exchange data between different applications (computers).

2.3 Bosch OPC

Bosch controllers can be equipped with the OPC technology which already became a sort of industry standard.

This kind of communication is not bound to a specific programming language but it allows access to controller data from Basic, VBA (Excel, Access), VBScript, Jscript, and also from C++, Delphi and an HTML site. However, prerequisite is that an OPC client is available.

Normally, each Scada system comes with an OPC client. If no OPC client is available, it is also no difficulty to develop one.

A simple test is possible by using the OPC client supplied by Factory Soft.

3 Licensing and base configuration

3.1 General principles

In order to use the WinTools software packages, the installation procedure must be run (cf. chapter 3.2); only when this has taken place will all the necessary components be installed and the WinTools components will be made known in the registry.

Communication

A network adapter and the TCP/IP protocol are required for external communication; a local host address 127.0.0.1 is enough for access to an integrated SoftPLC.

For communication with a number of control systems, a method is applied to enable switching between control systems or creation of more than one instance of the BoschCOM. However, if BoschOPC is used, only one server instance can be created.

The assemblies COM-E and COM2-E and the SoftPLC communicate via fixed port numbers:

5006 general port number (default setting)
5001 extended port number only for the SoftPLC.

BoschOPC-Server

BoschOPC-Server uses the Microsoft COM technology. Therefore, Windows NT (incl. Service Pack 3 or higher) is required. If Windows 95 or Windows 98 are used, an additional package DCOM98 has to be installed. This EXE file is part of the installation set, a download from the Microsoft homepage is possible as well.

Password Protection

Write accesses can be protected in the control system by a password; this is why there is also a 'Login' command. The default password of the control systems is 'BoschPlcSystems'.

3.2 Installation

Installation is done via the **WinTools** set of floppy disks (Order No. 1070 083 957), which is the software package for licensing

- BoschCOM DLL Client
- BoschCOM DLL Server
- OPC-Server (PCL und CL550).

The installation routine is started by invoking **Setup** (on Disk1 or Disk1_e respectively).

The directory you specify during installation will only contain the WinTOOLS.EXE file, the examples and the documentation files. The parts BoschCOM.DLL, BoschPlcServer and BoschOPC server are located in the Windows system directory.

3.3 License

Licensing is in the same way as with PLC utility programs per Hardlock or per soft licensing by means of Crypkey.

Demo operating mode (BoschCOM Client only)

If no licensing takes place, a demo operating mode becomes active; after one hour, the communication to the control system is interrupted.

Init file

The selection of either Hardlock or Softlicense can be made using the switch '/H' (same as WinPLC, WinDP). For the application BoschCOM, this can be set in the file BoschCOM.INI. For the application BoschOPC, this setting is done in the file BoschOPC.INI. This file looks as follows for Softlicense:

```
[License]
Hardlock=/h
```

and for Hardlock (default setting if no file is present)

```
[License]
Hardlock=/H
```

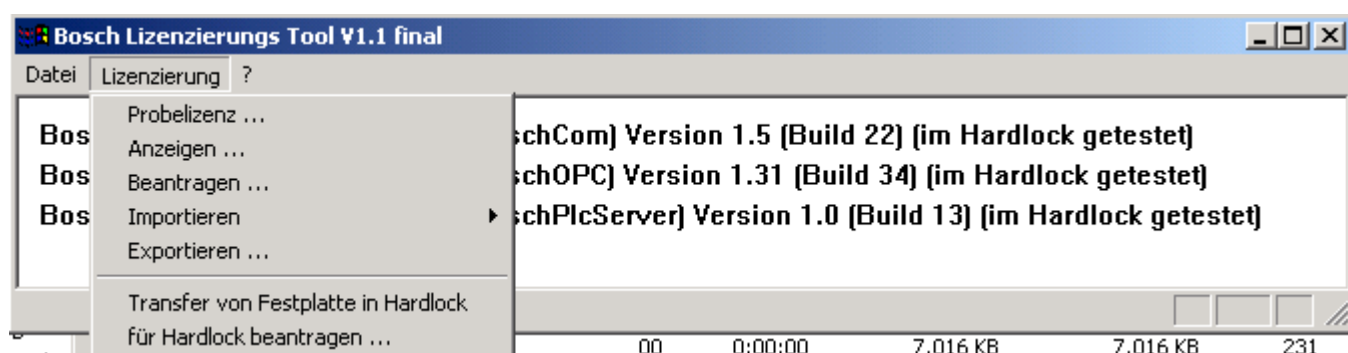
User guidance

WinTools guides the user through the different steps of licensing. It is invoked via **wintools.exe**.

Menu item 'Lizenzierung'

After WinTools has been invoked, a window appears which lists the licensing status of the individual components.

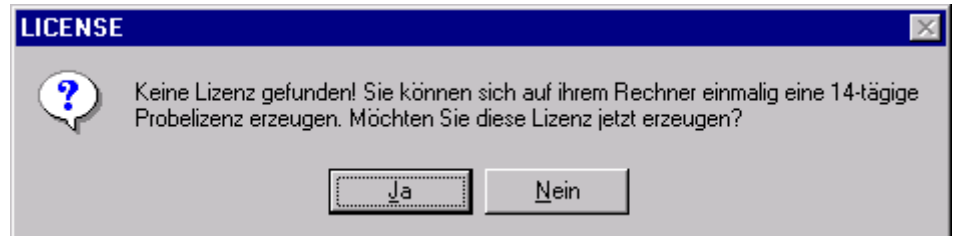
If the menu item **Lizenzierung** (Licensing) is selected, a menu appears with the licensing points.



3.4 Licensing

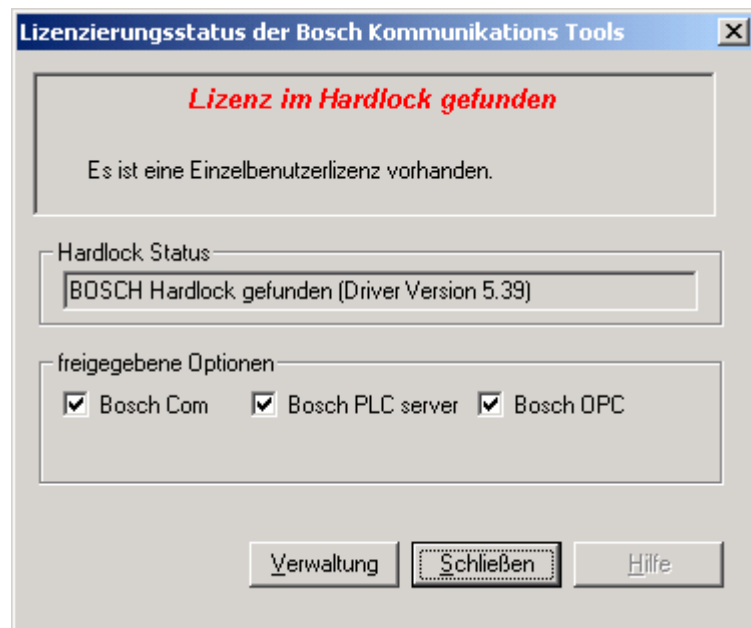
3.4.1 Test license

A test license can only be created for software protection, which means the switch in the BoschCOM.INI or BoschOPC.INI file must be set to 'h'.



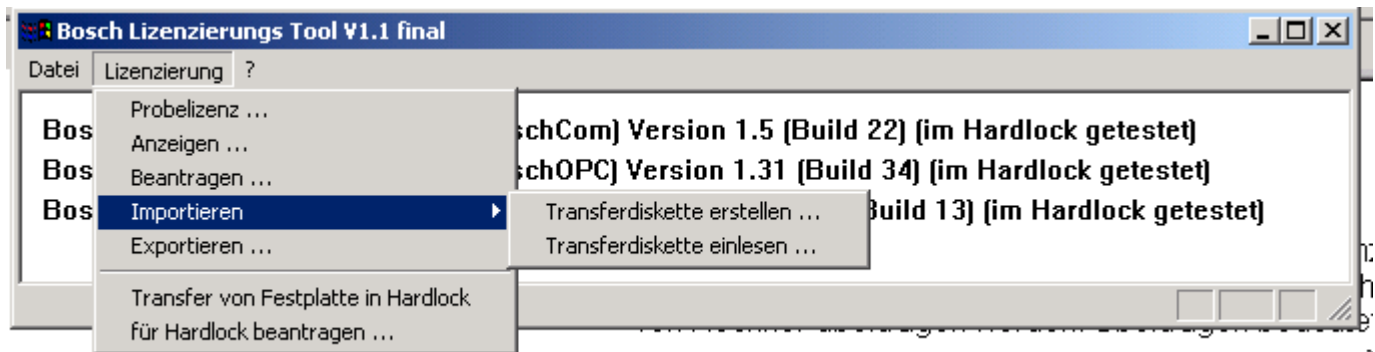
3.4.2 Displays

The current licensing is displayed here in a dialog box.



3.5 Import

Using the functions **Lizenz > Importieren** (License > Import) and **Lizenz > Exportieren** (License > Export), an existing license can be transferred from one computer to another. Transfer means that the license is passed on in such a way that the first computer loses the license. Here, the license is passed on using a transfer floppy disk, i.e. the licensing is on a floppy disk for a short period. This floppy disk can be a commercially available, formatted 3 1/2" or 5 1/4" disk. Following the license transfer, the floppy disk can be reused in the normal manner.



The operation is divided into three steps:

- 1 The target computer (has no license yet) uses the **Importieren > Transferdiskette erstellen** (Import > Create transfer disk) function to create a floppy disk.
- 2 The source computer (has a license) uses the **Exportieren** (Export) function to transfer its license information to the prepared transfer floppy disk. In doing so, it relinquishes its license.
- 3 The target computer uses the **Importieren > Transferdiskette einlesen** (Import > Read transfer disk) function to read the transfer disk. In doing so, it takes over the license information. The transfer disk cannot be used for other licensing on other computers. If you require other computer licenses, please apply for them from Robert Bosch GmbH.

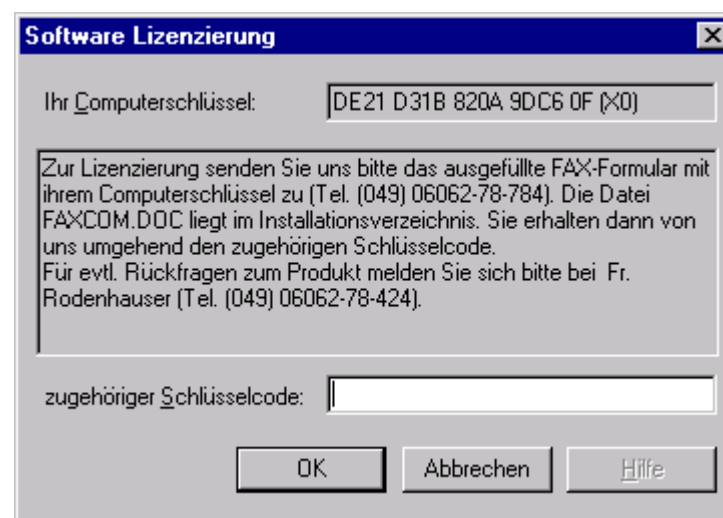
This function is used to perform Step 1. Select the floppy disk drive and insert an empty floppy disk. When you click on the **OK** button, the transfer disk is created.

3.5.1 Export

See Import.

3.5.2 Apply

This can be used to apply for a license for the Crypkey procedure.



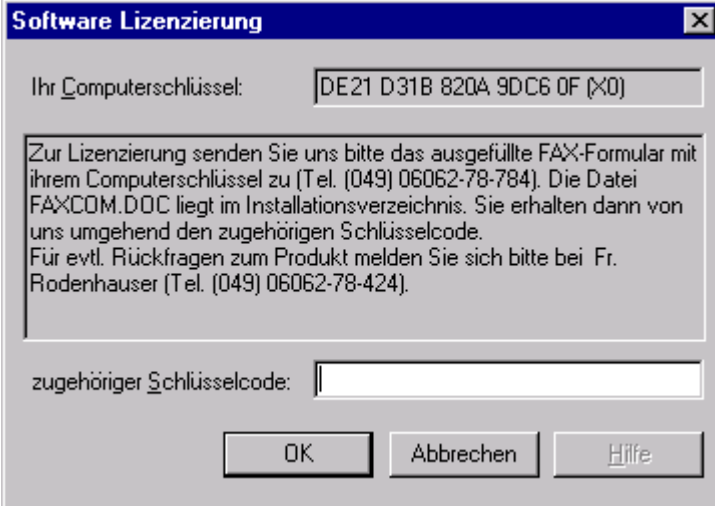
3.5.3 Transfer from hard disk into Hardlock

The Softlicense can be transferred into the Hardlock here. Afterwards, only the Hardlock license is valid.

⇒ **A return transfer is not possible.**

3.5.4 Apply for Hardlock

A licensing code for the Hardlock can be applied for here.



The image shows a Windows-style dialog box titled "Software Lizenzierung". It contains a text field for "Ihr Computerschlüssel:" with the value "DE21 D31B 820A 9DC6 0F (X0)". Below this is a large text area with instructions in German: "Zur Lizenzierung senden Sie uns bitte das ausgefüllte FAX-Formular mit ihrem Computerschlüssel zu (Tel. (049) 06062-78-784). Die Datei FAXCOM.DOC liegt im Installationsverzeichnis. Sie erhalten dann von uns umgehend den zugehörigen Schlüsselcode. Für evtl. Rückfragen zum Produkt melden Sie sich bitte bei Fr. Rodenhauser (Tel. (049) 06062-78-424)." At the bottom, there is a text field for "zugehöriger Schlüsselcode:" and three buttons: "OK", "Abbrechen", and "Hilfe".

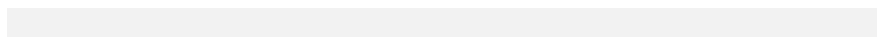
Software Lizenzierung

Ihr Computerschlüssel: DE21 D31B 820A 9DC6 0F (X0)

Zur Lizenzierung senden Sie uns bitte das ausgefüllte FAX-Formular mit ihrem Computerschlüssel zu (Tel. (049) 06062-78-784). Die Datei FAXCOM.DOC liegt im Installationsverzeichnis. Sie erhalten dann von uns umgehend den zugehörigen Schlüsselcode.
Für evtl. Rückfragen zum Produkt melden Sie sich bitte bei Fr. Rodenhauser (Tel. (049) 06062-78-424).

zugehöriger Schlüsselcode:

OK Abbrechen Hilfe



4 Overview of functions BoschCOM

After an overview on error handling, time properties and an introductory example, this chapter describes the range of functions of the BoschCOM.DLL..

At the point IDL syntax, each function description contains an excerpt from the IDL description (Interface Description Language) of the Bosch components. This enables every programmer, independent of programming language, to view the parameters of the function.

In order to illustrate the syntax in each programming environment, an example block is inserted after each subchapter.

An important point is the telegram size and the amount of useful data. The limit here is 1200 bytes.

The data types selected are the COM data types. This makes it possible to use standard marshalling.

4.1 Error handling

All functions are linked to the COM-specific error handling, i.e. an error can be cleared with the corresponding procedure.

At the moment, a return value of 'S_OK' is regarded as 'no error' and all other values are errors. In Visual Basic, an error string generated by the BoschCOM component can also be output. The procedure is

```
Sub Test
  On Error Goto ErrHandler
  Pcl.Init
  Exit Sub
ErrHandler:
  MsgBox Err.Description
End Sub
```

The actual error codes and the confirmations of the control systems are coded separately. There is also a header file '**BoschErr.h**', in which all errors are listed.

These errors are read using the property 'LastError' as numbers and using 'Error' as text.

<i>Symbolic name</i>	<i>Value</i>
BOSCH_PLC_NO_ERROR	0x00
BOSCHPLC_ERROR_MEMORY_ACCESS	0x02
BOSCHPLC_ERROR_MEMORY	0x03
BOSCHPLC_ERROR_INTERNAL	0x05
BOSCHPLC_COME_BLKADR_FLR	0x10
BOSCHPLC_ERROR_COMMAND_UNKNOWN	0x20
BOSCHPLC_ERROR_PROTOCOL_UNKNOWN	0x21
BOSCHPLC_ERROR_COORDINATION_UNKNOWN	0x23
BOSCHPLC_ERROR_PARAMETER	0x25
BOSCHPLC_ERROR_BLOCKLENGTH	0x26
BOSCHPLC_ERROR_TELEGRAM_TYPE	0x28
BOSCHPLC_ERROR_COMMAND_DIRECTION	0x29
BOSCHPLC_ERROR_ALIGNMENT	0x3A
BOSCHPLC_ERROR_ADDRESS	0x3B
BOSCHPLC_ERROR_PARAMETER_INVALID	0x3C
BOSCHPLC_ERROR_OPERAND	0x3D
BOSCHPLC_ERROR_NO_IDENTIFICATION	0x40
BOSCHPLC_ERROR_STRUCTURE	0x50
BOSCHPLC_ERROR_NO_STRUCTURE	0x51
BOSCHPLC_ERROR_STRUCTURE_LENGTH	0x52
BOSCHPLC_ERROR_BUFFER_OVERFLOW	0x63
BOSCHPLC_ERROR_MODULE_SPECIFIC	0x82
BOSCHPLC_ERROR_COORDINATION	0xD2
BOSCHPLC_ERROR_MODULE_NOT_FOUND	(0x01+0x8200)
BOSCHPLC_ERROR_LOGIN	(0x08+0x8200)
BOSCHPLC_ERROR_PASSWORD	(0x09+0x8200)
BOSCHPLC_ERROR_PASSWORD_WRONG	(0x0A+0x8200)
BOSCHPLC_ERROR_LOGIN_TIMEOUT	(0x0B+0x8200)
BOSCHPLC_ERROR_STOP	(0x20+0x8200)
BOSCHPLC_ERROR_RUN	(0x21+0x8200)
BOSCHPLC_ERROR_CHANGE_OPERATINGMODE	(0x22+0x8200)
BOSCHPLC_ERROR_ACCESS_MODE	(0x23+0x8200)
BOSCHPLC_ERROR_PROTECTION	(0x24+0x8200)
BOSCHPLC_ERROR_SET_TIMER	(0x25+0x8200)
BOSCHPLC_ERROR_MODULE_NO	(0x26+0x8200)
BOSCHPLC_ERROR_MODULE_MISSING	(0x27+0x8200)
BOSCHPLC_ERROR_DATAMODULE_TOO_SMALL	(0x28+0x8200)
BOSCHPLC_ERROR_TRANSFER_NOT_ALLOWED	(0x30+0x8200)
BOSCHPLC_ERROR_TCP_SEND_ERROR	(0x77+0x8200)
BOSCHPLC_ERROR_TCP_RECEIVE_ERROR	(0x78+0x8200)

4.2 Time properties

All functions currently work coordinated with PE + STOP, i.e. in the IO state, all jobs are processed. This ensures a consistent status of data.

However, this also means that the time for data interchange depends decisively on the PLC cycle time. In order to take this into account, there is a function that enables the BoschCOM to wait a time 'x' before obtaining a reply.

Another important factor is of course the Ethernet network; waiting periods for example in the normal company network caused by collisions etc. must not be neglected.

The table shows the values for reading a complete DM from each control system. The time test was performed on a PC with Pentium 233 MMX processor with software PLC.

Control system	Cycle	C++ (*)	Basic
COM-E CL400	8 ms	30 ms	70 ms
COM2-E	4 ms	40 ms	70 ms
Ext. SoftPLC	8 ms	10 ms	< 20 ms
Int. SoftPLC	4 ms	< 10 ms	< 20 ms

(*) For access per MFC wrapper class (and thus access per IDISPATCH) the execution time increases by 25-30 %.

The same time was measured for writing a data module.

4.3 Introductory example

How easy it is under Basic to communicate with our control systems is shown by our example, which determines whether the control system is in STOP.

```

Sub Test
  On Error Goto ErrHandler
  Dim Mode As Integer
  Pcl.TargetIpAddr = „127.0.0.1“
  Pcl.Init
  Pcl.OpenChannel
  Mode = Pcl.PLCState
  if Mode > 0 then
    MsgBox „STOP“
  End if
  Exit Sub
ErrHandler:
  MsgBox Err.Description 'Error output via Standard Error Handling
End Sub

```


4.4 Creating components

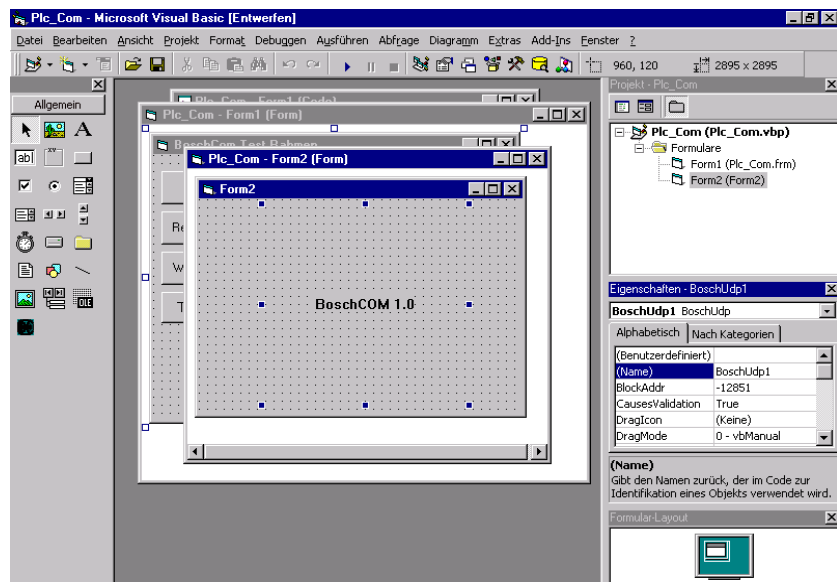
The different programming environments require different mechanisms to make a COM interface known. Only then can the methods and properties be used.

4.4.1 Visual Basic 6.0

Here, the menu item **Projekt > Verweise** (Project > References) must first be used to link the BoschCOM type library. If this has taken place, there are several possibilities to make the desired interface known, in this case 'BoschUDP'.

Visual method

There is an icon for BoschUDP in the toolbox; double-clicking places an object on the form. In the properties (right), the assigned property name can be changed.



Non-visual method

Program code has to be written here:

```
Dim Pcl As BoschUdp
Set Pcl = New BoschUdp
```


4.4.2 Visual Basic for Applications

Here, in the same way as for Visual Basic, the BoschCOM.DLL is first made known per reference. This takes place in the BasicEditor with the menu item **Extras > Verweise** (Tools > References).

Depending on the version of the VBA, the program code to make the interface known is:

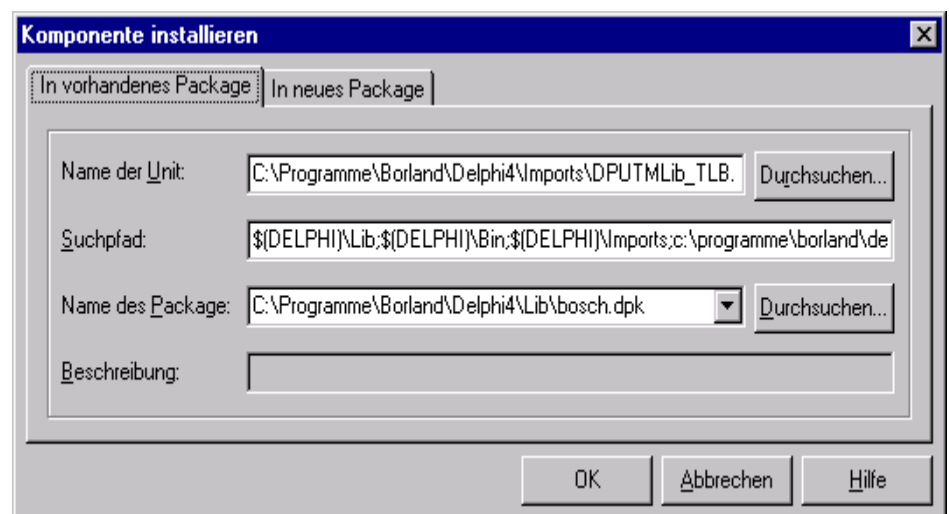
```
Dim Pcl As BoschUdp
Set Pcl = New BoschUdp
```

or

```
Dim Pcl
Set Pcl = CreateObject("BoschUdp.BoschUdp.1")
```

4.4.3 Delphi

Under Delphi, the BoschCOM has to be imported first; this is done using the menu item **Komponente > ActiveX importieren** (Component > Import ActiveX). This component must then be installed using **Komponente > Komponente installieren** (Component > Install component). In the subsequent dialog, it can be specified in which package this is to be installed.



The selected package is now compiled; then the BoschCOM appears in the Delphi toolbar table ActiveX and can be placed on a form.

4.4.4 C, C++ (direct access)

C, C++ is the most direct programming environment; a certain amount of knowledge of COM is necessary to understand the individual COM functions.

This type of access has the best performance.

The include file 'BoschCom.h' describes the interfaces; the file 'BoschCOM_i.c' contains the code for the GUID (globally unique ID), without which COM does not function.

The program code for creating the interface looks like this:

```
#include "BoschCom.h"

HRESULT hr;
IBoschUdp* pPcl;
VARIANT vt, vtRead, vtNoIni;
short Count = 0;

hr = CoInitialize(NULL);
if(FAILED(hr)) return ;

// Pointer to BoschCom - get BoschUdp
hr = CoCreateInstance (CLSID_BoschUdp, NULL,
                      CLSCTX_INPROC_SERVER,
                      IID_IBoschUdp, (void**) &pPcl);
if (FAILED(hr)) return ;
```

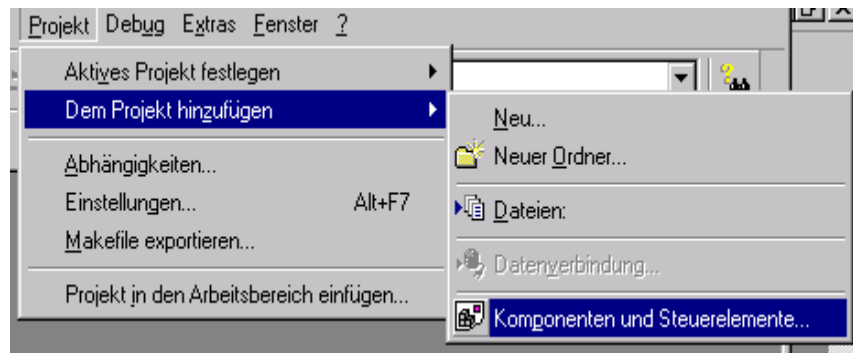
The syntax of the properties is given the prefix '**put_**' or '**get_**', as the case may be. For example, 'HostPort' becomes 'SetHostPort' or 'GetHostPort'.

4.4.5 Mfc applications

The direct C++ interface can of course be used for Mfc applications. This has the advantage of higher speed, but provides little convenience for accessing the interface and data types.

A convenient interface can, however, be created easily by inserting a wrapper class into the project.

This means that all methods and properties of the BoschCOM are encapsulated in a C++ class.



This C++ class gives the syntax of the properties the prefix '**Set**' or '**Get**', as the case may be. For example, 'HostPort' becomes 'SetHostPort' or 'GetHostPort'.

This syntax is not listed explicitly on description of the individual functions.

A small excerpt shows the syntax :


```

CString GetAbout();
short GetRepeatCount();
void SetRepeatCount(short nNewValue);
void SetTargetIPAddr(LPCTSTR lpszNewValue);
CString GetError();
void OpenChannel();
void CloseChannel();
short GetDf(short Offset, short Count, const VARIANT& Vars);
void SetDf(short Offset, short Count, const VARIANT& Vars, short

```

In the following, it is assumed that one of the preceding login sequences has been run through.

Access in Basic or Delphi is therefore via Pcl; access in C++ is via pPcl->, or via the Mfc wrapper class.

4.5 Optional parameters

This includes data that always has default values, but in running operation can/must be adapted to the circumstances.

4.5.1 TargetIPAddr

Specifies the target IP address, or reads the target IP address.

```

Read:   HRESULT TargetIPAddr ([out, retval] BSTR *pVal);
Write:  HRESULT TargetIPAddr ([in] BSTR newVal);

```

ReturnCode:S_OK

The default address is 127.0.0.1.

4.5.2 TargetPort

Specifies the port number in the target device, or reads it.

```

Read:   HRESULT TargetPort([out, retval] short *pVal)
Write:  HRESULT TargetPort([in] short newVal)

```

ReturnCode:S_OK

The default port number is 5006.

This is the only port number for the assemblies COM-E and COM2-E; for PCL, there is the additional port number 5001.

4.5.3 HostPort

Specifies the port number in the host device, or reads it.

```

Read:   HRESULT HostPort([out, retval] short *pVal)
Write:  HRESULT HostPort([in] short newVal)

```

ReturnCode:S_OK

The default port number is 5007.

This parameter should not be changed.

4.5.4 BlockAddr

Specifies the block address in the target computer. This is necessary to inform the COM-E as regards which assembly it is to communicate with. For PCL and CL200, a 0 must be entered here. For the CL400, it must be 240 (16#F0); for CL500, the setting of the block addresses of the individual ZS types is most important; these can be read in the SK table in the WinPLC. The following block addresses are preset: ZS0: 0 , ZS1: 8, ZS2: 16, ZS3: 24.

A detailed description of this can be found in the manual for assembly COM-E.

Read: *HRESULT BlockAddr([out, retval] short *pVal)*
Write: *HRESULT BlockAddr([in] short newVal)*

ReturnCode:S_OK

Default value is 0.

4.5.5 TimeToWait

Specifies the waiting period (in ms) before a repetition of a telegram is started, or reads the set value.

Read: *HRESULT TimeToWait ([out, retval] short *pVal)*
Write: *HRESULT TimeToWait ([in] short newVal)*

ReturnCode:S_OK

250 ms is assumed as default value.

4.5.6 MaxTime

Specifies the maximum waiting period (in ms) for processing a telegram, or reads the set value.

Read: *HRESULT MaxTime([out, retval] short *pVal)*
Write: *HRESULT MaxTime([in] short newVal)*

ReturnCode:S_OK

5000 ms is assumed as default value.

4.5.7 RepeatCounter

Specifies the number of repetitions before an error is reported, or reads the set value.

Read: *HRESULT RepeatCount ([out, retval] short *pVal)*
Write: *HRESULT RepeatCount ([in] short newVal)*

ReturnCode:S_OK

The set default value is 5.

4.5.8 RecvDelay

Specifies a delay time to be waited between sending a job and receiving the reply.

Read: *HRESULT RecvDelay([out, retval] short *pVal)*
 Write: *HRESULT RecvDelay([in] short newVal)*

ReturnCode:S_OK

The default setting is 0, i.e. no delay.

4.5.9 Examples

Basic, VBA

COM is initialized by the programming environment or runtime automatically.

```
Dim Pcl As BoschUdp
Dim HostPort As Integer

Pcl.TargetIPAddr = „127.0.0.1“
Pcl.TargetPort =5006
Pcl.RepeatCounter =3
Pcl.BlockAddr =240
Pcl.TimeToWait=500
HostPort = Pcl.HostPort
```

C++

COM must be explicitly initialized.

```
IBoschUdp* pPcl;
HRESULT hr;           // must always be evaluated explicitly
short HostPort;

hr = CoInitialize (NULL); // Initialize COM
if (FAILED(hr))
    return ;

// Pointer to BoschCom - get BoschUdp
hr = CoCreateInstance (CLSID_BoschUdp, NULL,
                      CLSCTX_INPROC_SERVER,
                      IID_IBoschUdp, (void**)&pPcl);
if (FAILED(hr))
    return ;

hr = pPcl->put_TargetIPAddr (L"127.0.0.1");
hr = pPcl->put_TargetPort (5006)
hr = pPcl->put_RepeatCounter =3
hr = pPcl->put_BlockAddr =240
hr = pPcl->put_TimeToWait=500
hr = pPcl->get_HostPort(&HostPort);
```


MFC application

The initialization of the COM must be performed by AfxOleInit before the BoschCOM.DLL is used.

```
CBoschUdp Udp;
HRESULT hr;

//Create wrapper class
if (Udp.Create (NULL, WS_VISIBLE, CRect (50,50,100,100),
              this, 0) == FALSE)
{
    return;
}
Udp.SetTargetIPAddr („127.0.0.1“);
Udp.SetTargetPort (5006);
Udp.SetRepeatCounter (3);
Udp.SetBlockAddr (240);
Udp.SetTimeToWait (500);
HostPort =Udp.GetHostPort();
```

Delphi

```
Type
    Udp: TBoschUdp;

Var
    HostPort: smallint;

Udp.TargetIpAddr := '142.2.20.11';
Udp.TargetPort := 5006;
Udp.RepeatCounter :=3;
Udp.BlockAddr := 240;
Udp.TimeToWait := 500;
HostPort =Udp.HostPort;
```

4.6 Connection functions

The functions that concern setting up and breaking a connection are described here.

4.6.1 Init

Sets up a connection with the set parameters (TargetIPAddr, TargetPort, etc.). To terminate the connection, a Deinit must take place.

Call instruction: *HRESULT Init ()*

ReturnCode:S_OK

E_FAIL

If a connection is set up, a logical channel is installed that can be opened/closed using OpenChannel and CloseChannel (if no parameters were set, the default setting is used).

In the event of an error, a message is generated which can be output by the client and the return code is not equal to S_OK.

4.6.2 Deinit

Ensures that a connection is terminated and releases the resources used.

Call instruction: *HRESULT Deinit ()*

ReturnCode:S_OK

4.6.3 OpenChannel

Opens a logical channel (which must have been set up using Init).

Call instruction: *HRESULT OpenChannel ()*

ReturnCode:S_OK

If the channel is used exclusively by the calling client, an OpenChannel at the start and a CloseChannel at the end of the session are sufficient.

The WinPLC and WinDP usually occupy the same port in the target computer, which means that in the case of simultaneous use the OpenChannel / CloseChannel parentheses must be present. Here, time aspects can be neglected.

4.6.4 InitAndOpenChannel

Opens a logical channel including specification of an IP address and of port numbers.

Call instruction: *HRESULT InitAndOpenChannel (BSTR IpAdr, short HostPort, short TargetPort)*

ReturnCode:S_OK

HostPort: port number on windows side (e.g. 5079)

TargetPort: currently 5006

This function checks whether a connection can be set up with the specified parameters. This channel remains open until the CloseChannel command is invoked.

Via this channel a communication to different controllers can take place.

4.6.5 CloseChannel

Closes a logical channel (which must have been set up using Init).

Call instruction: *HRESULT CloseChannel ()*

ReturnCode:S_OK

4.6.6 Login

Enables logging in to a control system. This is only necessary if a write job is to be started.

Call instruction: *HRESULT Login (BSTR Pwd)*

ReturnCode:S_OK

E_FAIL

The default password of the control systems is 'BoschPlcSystems'.

If a write job is started and no login has taken place, an error is set (HRESULT is not equal to S_OK) and the Error command (description below) can be used to determine the error cause.

4.6.7 Examples

The default settings of the parameters are as described in chapter 3.2.9.

Basic

```
Pcl.Init
Pcl.OpenChannel
'Access to data control system possible
Pcl.Login ("BoschPlcSystems")
```

```
Pcl.CloseChannel
Pcl.Deinit
```

C++

```
HRESULT hr;

hr = pPcl->Init();
hr = pPcl->OpenChannel();
hr = pPcl->Login (L„BoschPlcSystems“);
hr = pPcl->CloseChannel();
hr = pPcl->Deinit();
```

Mfc

```
Udp.Init ();
Udp.OpenChannel();
Udp.Login(„BoschPlcSystems“);
Udp.CloseChannel ();
Udp.Deinit();
```

Delphi

```
Udp.Init ();
Udp.OpenChannel();
Udp.Login(„BoschPlcSystems“);
Udp.CloseChannel();
Udp.Deinit();
```

4.7 DB access functions

These functions can be used to read and write data. It is possible to read/write individual data in different types, but a memory excerpt can also be read/written.

If a non-existent data module is activated or if the limits are exceeded, this is stored in the internal error memory and an 'E_FAIL' is output as HRESULT.

4.7.1 DM_String

Writes a string in a data module or reads a string from a data module.

Read: *HRESULT DM_String (short No, short Offset, short Count, [out, retval] BSTR *pVal)*

Write: *HRESULT DM_String(short No, short Offset, short Count, [in] BSTR newVal)*

4.7.2 DM_Dword

Writes a double word in a data module or reads a double word from a data module.

Read: *HRESULT DM_Dword (short No, short Offset, short Count, [out, retval] long *pVal)*

Write: *HRESULT DM_Dword (short No, short Offset, short Count, [in] long newVal)*

The start address must be a multiple of 4 so that the grid is geared to double words (DWORD).

4.7.3 DM_Word

Writes a word in a data module or reads a word from a data module.

Read: *HRESULT DM_Word (short No, short Offset, short Count, [out, retval] short *pVal)*

Write: *HRESULT DM_Word (short No, short Offset, short Count, [in] short newVal)*

The start address must be a multiple of 2 so that the grid is geared to words (WORD).

4.7.4 DM_Real

Writes a real value in a data module or reads a real value from a data module.

Read: *HRESULT DM_Real (short No, short Offset, short Count, [out, retval] float *pVal)*

Write: *HRESULT DM_Real (short No, short Offset, short Count, [in] float newVal)*

The start address must be a multiple of 4 so that the grid is geared to double words (DWORD).

4.7.5 DM_LReal

Writes an LReal value (real with double accuracy) in a data module or reads an LReal value from a data module.

Read: *HRESULT DM_LReal (short No, short Offset, short Count, [out, retval] double *pVal)*

Write: *HRESULT DM_LReal (short No, short Offset, short Count, [in] double newVal)*

The start address must be a multiple of 8 so that the grid is geared to QWORD.

4.7.6 DM

Writes a data area in a data module or reads a data area from a data module.

Read: *HRESULT DM (short No, short Offset, short Count, VARIANT Vars, [out, retval] short *pVal)*

Write: *HRESULT DM (short No, short Offset, short Count, VARIANT Vars, [in] short newVal)*

Here, the client must create a corresponding array. On reading, the number of read bytes is returned; on writing, the value to be transferred has no effect. The variant array must be configured by the calling function, otherwise an error message appears. The type can be VT_UI1, VT_I2, VT_I4.

⇒ **This function does not work under DELPHI.**

4.7.7 ReadDM

Reads a data area from a data module.

Call instruction: *HRESULT ReadDM (short No, short Offset, short Count, [out, retval] VARIANT *pVal)*

Generally returns a byte array. This byte array is created and released in the BoschCOM.DLL; the client can access it following the call instruction.

⇒ **This function has been specially conceived for DELPHI users, but it also works under Basic and C++.**

4.7.8 PutMaskedDM

Uses a Setmask and ResetMask to write bits in a byte/word/double word in a data module.

Call instruction: *HRESULT PutMaskedDM(short DmNo, short offset, short len, long SetMask, long ResetMask)*

The size of the Setmask / ResetMask is regulated by the 'Len' information.

Double word only functions with the SoftPLC.

4.8 Marker access functions

These functions can be used to read and write data. It is possible to read/write individual data in different types, but a memory excerpt can also be read/written.

If a non-existent marker is activated, this is stored in the internal error memory and an 'E_FAIL' is output as HRESULT.

4.8.1 M_Dword

Writes a double word in a marker double word or reads a double word from a marker double word.

Read: *HRESULT M_Dword (short Offset, short Count, [out, retval] long *pVal)*

Write: *HRESULT M_Dword (short Offset, short Count, [in] long newVal)*

The start address must be a multiple of 4 so that the grid is geared to double words (DWORD).

4.8.2 M_Word

Writes a word in a marker word or reads a word from a marker word.

Read: *HRESULT M_Word (short Offset, short Count, [out, retval] short *pVal)*

Write: *HRESULT M_Word (short Offset, short Count, [in] short newVal)*

The start address must be a multiple of 2 so that the grid is geared to words (WORD).

4.8.3 M_Real

Writes a real value in a marker area or reads a real value from a marker area.

Write: *HRESULT M_Real (short Offset, short Count, [out, retval] float *pVal)*

Write: *HRESULT M_Real (short Offset, short Count, [in] float newVal)*

The start address must be a multiple of 4 so that the grid is geared to double words (DWORD).

4.8.4 M_LReal

Writes an LReal value (real with double accuracy) in a marker area or reads an LReal value from a marker area.

Read: *HRESULT M_LReal (short Offset, short Count, [out, retval] double *pVal)*

Write: *HRESULT M_LReal (short Offset, short Count, [in] double newVal)*

The start address must be a multiple of 8 so that the grid is geared to QWORD.

4.8.5 M

Writes a data area in a marker area or reads a data area from a marker area.

Read: *HRESULT M (short Offset, short Count, VARIANT Vars, [out, retval] short *pVal)*

Write: *HRESULT M (short Offset, short Count, VARIANT Vars, [in] short newVal)*

Here, the client must create a corresponding array. On reading, the number of read bytes is returned; on writing, the value to be transferred has no effect. The variant array must be configured by the calling function, otherwise an error message appears. The type can be VT_UI1, VT_I2, VT_I4.

⇒ **This function does not work under DELPHI.**

4.8.6 ReadM

Reads a data area from a marker area.

Call instruction: *HRESULT ReadM (short Offset, short Count, [out, retval] VARIANT *pVal)*

Generally returns a byte array. This byte array is created and released in the BoschCOM.DLL; the client can access it following the call instruction.

⇒ **This function has been specially conceived for DELPHI users, but it also works under Basic and C++.**

4.8.7 PutMaskedM

Uses a Setmask and ResetMask to write bits in a byte/word/double word in a marker area.

Call instruction: *HRESULT PutMaskedM (short offset, short len, long SetMask, long ResetMask)*

The size of the Setmask / ResetMask is regulated by the 'Len' information.

Double word only functions with the SoftPLC.

4.9 Data field access functions

These functions can be used to read and write data. It is possible to read/write individual data in different types, but a memory excerpt can also be read/written.

If a non-existent data field area is activated, this is stored in the internal error memory and an 'E_FAIL' is output as HRESULT.

4.9.1 DF_Dword

Writes a double word in a data field double word or reads a double word from a data field double word.

Read: *HRESULT DF_Dword (short Offset, short Count, [out, retval] long *pVal)*

Write: *HRESULT DF_Dword (short Offset, short Count, [in] long newVal)*

The start address must be a multiple of 4 so that the grid is geared to double words (DWORD).

4.9.2 DF_Word

Writes a word in a data field word or reads a word from a data field word.

Read: *HRESULT DF_Word (short Offset, short Count, [out, retval] short *pVal)*

Write: *HRESULT DF_Word (short Offset, short Count, [in] short newVal)*

The start address must be a multiple of 2 so that the grid is geared to words (WORD).

4.9.3 DF_Real

Writes a real value in a data field area or reads a real value from a data field area.

Read: *HRESULT DF_Real (short Offset, short Count, [out, retval] float *pVal)*

Write: *HRESULT DF_Real (short Offset, short Count, [in] float newVal)*

The start address must be a multiple of 4 so that the grid is geared to double words (DWORD).

4.9.4 DF_LReal

Writes an LReal value (real with double accuracy) in a data field area or reads an LReal value from a data field area.

Read: *HRESULT DF_LReal (short Offset, short Count, [out, retval] double *pVal)*

Write: *HRESULT DF_LReal (short Offset, short Count, [in] double newVal)*

The start address must be a multiple of 8 so that the grid is geared to QWORD.

4.9.5 DF

Writes a data area in a data field area or reads a data area from a data field area.

Read: *HRESULT DF (short Offset, short Count, VARIANT Vars, [out, retval] short *pVal)*

Write: *HRESULT DF (short Offset, short Count, VARIANT Vars, [in] short newVal)*

Here, the client must create a corresponding array. On reading, the number of read bytes is returned; on writing, the value to be transferred has no effect. The variant array must be configured by the calling function, otherwise an error message appears. The type can be VT_UI1, VT_I2, VT_I4.

⇒ **This function does not work under DELPHI.**

4.9.6 ReadDF

Reads a data area from a data field area.

Call instruction: *HRESULT ReadDF (short Offset, short Count, [out, retval] VARIANT *pVal)*

Generally returns a byte array. This byte array is created and released in the BoschCOM.DLL; the client can access it following the call instruction.

⇒ **This function has been specially conceived for DELPHI users, but it also works under Basic and C++.**

4.9.7 PutMaskedDF

Uses a Setmask and ResetMask to write bits in a byte/word/double word in a data field area.

Call instruction: *HRESULT PutMaskedDF (short offset, short len, long SetMask, long ResetMask)*

The size of the Setmask / ResetMask is regulated by the 'Len' information.

Double word only functions with the SoftPLC.

4.10 Symbolic access

This access is in preparation.

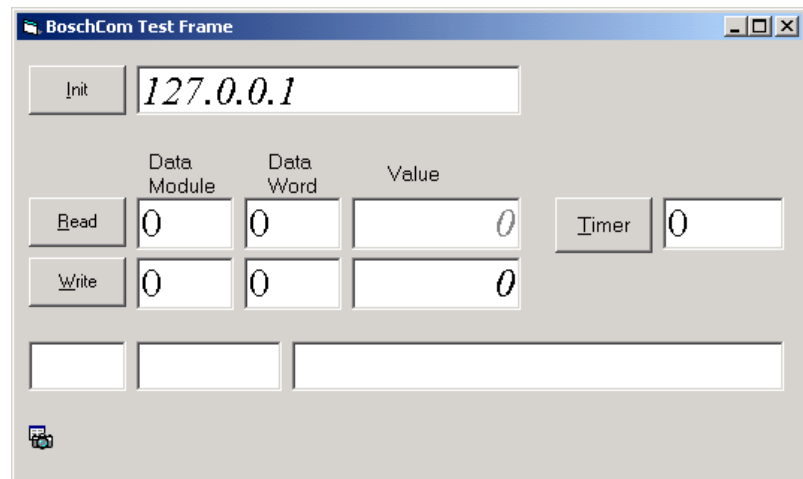
4.10.1 SymData

Currently not supported.

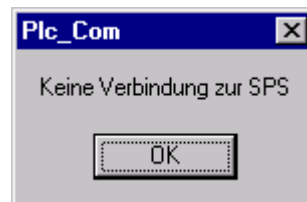
5 Examples BoschCOM

5.1 Visual Basic

This form enables reading / writing of a data word; if the timer is set, this takes place in cycles.



If no connection is set up, the standard error handling causes a message box to appear.



Option Explicit
Option Base 1

Dim Pcl As BoschUdp
Dim IsInit As Boolean
Dim XPos As Single
Dim YPos As Single

'Init App

```
Private Sub Form_Load()
  IsInit = False
  Set Pcl = New BoschUdp
  Pcl.TargetPort = 5006
  Pcl.BlockAddr = 240
  Pcl.TargetIPAddr = IpAddr.Text
  Pcl.Init
End Sub
```



```

' Deinit App

Private Sub Form_Unload(Cancel As Integer)
    Pcl.CloseChannel
    Pcl.Deinit
End Sub

' Init PLC Communication

Private Sub Init_Click()
    XPos = Image1.Left
    YPos = Image1.Top

    If IsInit = True Then
        Timer1.Enabled = False
        Pcl.CloseChannel
        IsInit = False
    End If

    If IsInit = False Then

        If connect = True Then
            DB0_Write.BackColor = &HFFFFFFC0
            IpAddr.BackColor = &HFFFFFFC0
            DB0_Read.BackColor = &HFFFFFFC0
            IsInit = True
        End If
    End If
End Sub

'Connect to PLC

Private Function connect() As Boolean
    On Error GoTo ErrHandler

    Dim RetVal As Integer
    Pcl.InitAndOpenChannel IpAddr.Text, 5007, 5006
    connect = True
    Exit Function

ErrHandler:
    ErrorTxt.Text = Err.Description
    connect = False
End Function

' Read Data Word

Private Sub Read_Click()
    On Error Resume Next

    Dim PlcMode As Integer
    Dim PLCError As Long
    Err.Number = 0

    If IsInit = True Then
        PlcMode = Pcl.PLCState
        If PlcMode > 0 Then
            PlcModeTxt = "Stop"
        Else
            PlcModeTxt = "Run"
        End If
    End If
End Sub

```



```
End If

DB0_Read.Text = Pcl.DM_Word(RDBNo, ROffset)
Error.Text = Pcl.LastError()
ErrorTxt.Text = Pcl.Error()
End If
End Sub

' Start Timer

Private Sub Timer_Click()
If IsInit = True Then
    Timer1.Enabled = True
    Timer1.Interval = Time1.Text
End If
End Sub

' Cyclic Read Data Word and show action

Private Sub Timer1_Timer()
Read_Click
If (XPos < 7000) Then
    XPos = XPos + 120
Else
    XPos = 120
End If
Image1.Move XPos, Ypos
End Sub

' Write Data Word

Private Sub Write_Click()
On Error Resume Next

Err.Number = 0
If IsInit = True Then
    Pcl.DM_Word(WDBNo, WOffset) = DB0_Write.Text
    Error.Text = Pcl.LastError()
    ErrorTxt.Text = Pcl.Error()
End If
End Sub
```


5.2 Visual C++

Access from C++ is slightly more complicated and requires basic knowledge of the COM architecture from Microsoft.

```
#include "iostream.h"
#include "BoschCom.h"

int main(int argc, char* argv[])
{
    unsigned char DM[512];

    IUnknown* pUnknown;
    IBoschUdp* pPcl;

    cout << "Bosch Client : Colnitalize()" << endl;
    HRESULT hr = Colnitalize(NULL);
    if(FAILED(hr))
        cout << "Colnitalize failed" << endl;

    cout << "Client: Calling CoCreateInstance()" << endl;
    hr = CoCreateInstance(CLSID_BoschUdp, NULL,
        CLSCTX_INPROC_SERVER,
        IID_IUnknown, (void**)&pUnknown);
    if(FAILED(hr))
        cout << "CoCreateInstance failed" << endl;

    cout << "Client: Calling QueryInterface() for ISum on " <<
        pUnknown << endl;
    hr = pUnknown->QueryInterface(IID_IBoschUdp, (void**)&pSum);
    if(FAILED(hr))
        cout << "IID_ISum not supported" << endl;

    hr = pUnknown->Release();
    cout << "Client: Calling pUnknown->Release()" << hr << endl;

    BSTR pX;
    hr = pPcl->get_About (&pX);
    char CharPath [MAX_PATH];
    WideCharToMultiByte (CP_ACP, 0, (WCHAR *) (pX), -1,
        CharPath, MAX_PATH, NULL, NULL);
    hr = pPcl->put_TargetIPAddr (L"142.2.20.11");
    hr = pPcl->Init ();
    hr = pPcl->OpenChannel ();
    for (int i = 0; i < 10; i++)
    {
        hr = pPcl->get_DM_Word (0,0,(short *) &DM[i*2]);
    }
    hr = pPcl->CloseChannel ();
    hr = pPcl->Deinit ();
    hr = pPcl->Release();
    cout << "Client: Calling pSum->Release() reference count = " <<
        hr << endl;

    cout << "Client: Calling CoUninitialize()" << endl;
    CoUninitialize();
    return 0;
}
```


5.3 MFC

The wrapper class CBoschUDP is created using the menu item 'Insert component'. The following source code shows a function that uses this class.

```
void CTestComDlg::OnCallBoschcom()
{
    CBoschUdp Udp;
    HRESULT hr;
    VARIANT vt;
    short Count = 0;

    if (Udp.Create(NULL, WS_VISIBLE, CRect(50,50,100,100),
        this, 0) == FALSE)
        return;

    Udp.SetTargetIPAddr("142.2.20.11");
    Udp.SetTargetPort(5006);
    Udp.Init();

    long ix, data;

    // Create SafeArray and fill with data
    SAFEARRAYBOUND rgsabound[1];
    rgsabound[0].lbound = 0;
    rgsabound[0].cElements = 512;
    SAFEARRAY *sfDB = SafeArrayCreate(VT_I2, 1, rgsabound);

    VariantInit(&vt);
    VariantChangeType(&vt, &vt, 0, VT_ARRAY|VT_I2);
    vt.vt = VT_ARRAY|VT_I2;
    vt.parray = sfDB;

    for(int i = 0; i < 512 / 2; ++i)
    {
        ix = (long) i;
        data = (long) i;
        hr = SafeArrayPutElement(sfDB, &ix, &data);
        if(FAILED(hr)) { }
    }
    // write 256 words in DB 0
    //Udp.SetDm(0, 0, 256, vt, Count);

    // read 256 words from DB 0
    Udp.OpenChannel();
    for (i = 0; i < 100; i++)
    {
        Udp.GetDm(0,0,256, vt);
    }
    Udp.CloseChannel();
    Udp.Deinit();
}
```


5.4 Delphi

Under Delphi, the BoschCOM has to be imported first; this is done using 'Import ActiveX'. This component must then be installed using 'Install component'. The BoschCOM appears in the toolbar after compilation in the table ActiveX and can be placed on a form.

```

unit Unit1;

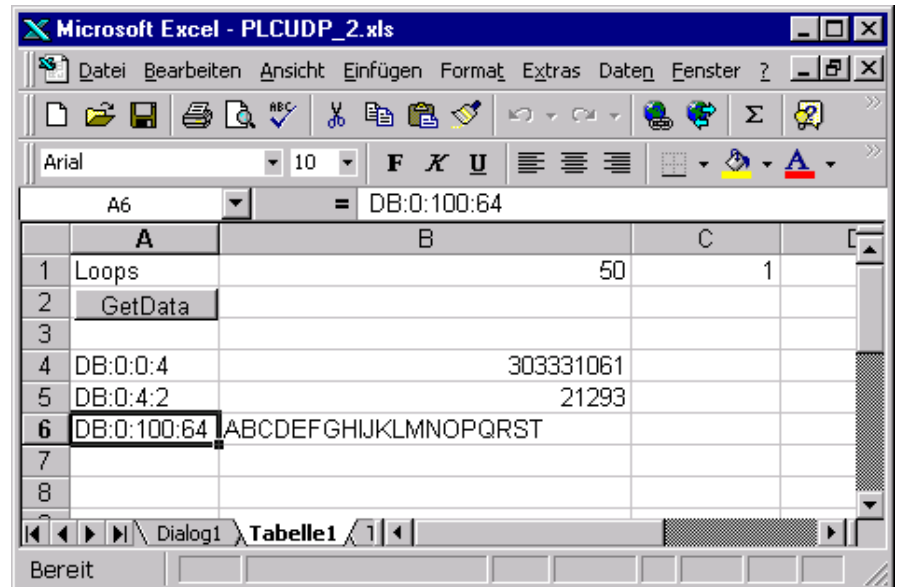
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs,
  StdCtrls, OleCtrls, BOSCHCOMLib_TLB, BOSCHUNILib_TLB;
type
  TForm1 = class(TForm)
    Edit1: TEdit;
    Button1: TButton;
    BoschUdp1: TBoschUdp;
    Edit4: TEdit;
    procedure Button1Click(Sender: TObject);
  private
    { Private Declarations }
  public
    { Public Declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.DFM}
procedure TForm1.Button1Click(Sender: TObject);
var
  s1:smallint;
  i:smallint;
  Vari:OleVariant;
  VarRef:OleVariant;

begin
  BoschUdp1.TargetIpAddr := Edit4.Text; (*142.2.20.11;*)
  BoschUdp1.Init ();
  BoschUdp1.OpenChannel();
  Vari := VarArrayCreate([0, 100], varByte);
  for i := 0 to 1 do
  begin
    s1 := BoschUdp1.DM [0,0,100, Vari];
    VarRef := BoschUdp1.ReadDM [0,0,200];
    BoschUdp1.DM [0,200,200, VarRef] := 200;
    if (s1 > 0) then
      s1 := VarRef[0]
    else
      s1 := VarRef[1];
    Edit1.Text := IntToStr (s1);
  end
end;
end.

```


5.5 Excel

The data is stored directly in the cells B4-B6; column A contains the comment.



Option Explicit

Option Base 1

Dim Pcl As BoschUdp

Dim DB As Variant

Dim No As Integer

Dim Offset As Integer

Dim Anz As Integer

Dim i As Integer

Dim MaxLoop As Integer

Dim itemcount As Long

Dim err As Integer

Dim TextRet As String

Private Sub Connect(Pclx As Object)

Pcl.TargetIPAddr = „127.0.0.1“

Pcl.Init

End Sub


```
Private Sub GetPlcData(ByRef Pcl As Object)

    Dim MyStringErg As String
    Dim MyErg As Long
    Dim MyWordErg As Integer

    Cells(1, 3).Value = i

    Pcl.OpenChannel

    MyStringErg = Pcl.DM_String(No, 100, 20)
    Cells(6, 2).Value = MyStringErg

    MyErg = Pcl.DM_Dword(No, 20)
    Cells(4, 2).Value = MyErg

    MyWordErg = Pcl.DM_Word(No, 2)    'ActiveCell.Value = MyErg
    Cells(5, 2).Value = MyWordErg
    Pcl.CloseChannel
    Pcl.Deinit
End Sub

Sub PlcUdpMain()

    Set Pcl = CreateObject("BoschUdp.BoschUdp.1")
    Connect Pcl

    Worksheets("Table1").Activate

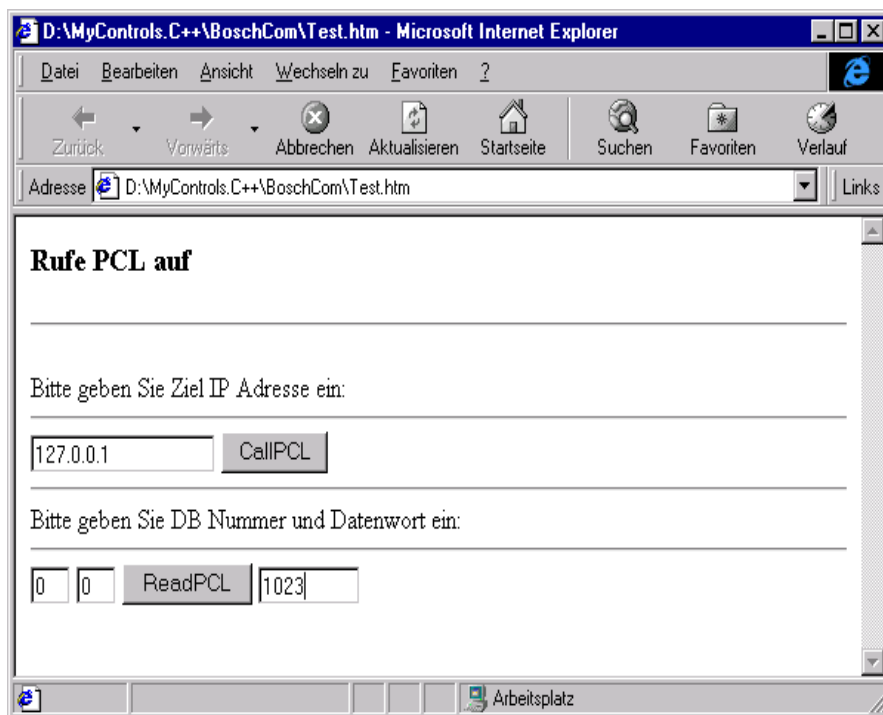
    Cells(1, 1).Value = "Loops"
    MaxLoop = Cells(1, 2).Value

    For i = 1 To MaxLoop
        GetPlcData Pcl
    Next i

End Sub
```


5.6 HTML and VB Script

BoschCOM.Dll can also be used to access the control systems per VB Script and HTML. This means that control system data can be visualized without a development environment, using only an ASCII editor and an Internet browser. Our example shows a simple page:



The above page is created by the following text:

```
</html>
<p>
<script language="VBScript">
<!--
Dim Pcl
Sub CallPCL_OnClick
  On Error Resume Next
  Dim PclForm
  Set PclForm=Document.PCLForm
  Set Pcl = CreateObject("BoschUdp.BoschUdp.1")
  MsgBox Pcl.About
  Pcl.TargetIPAddr = PclForm.IPADDR.Value
  Pcl.Init
  if Err <> 0 then
    MsgBox Err.Description
  end if
End Sub
```



```

Sub ReadPCL_OnClick
On Error Resume Next
if Err = 0 then
    Dim PclForm
    Set PclForm=Document.PCLForm
    Pcl.OpenChannel
    PclForm.DATA.Value=Pcl.DM_Word(PclForm.DBNO.Value,
PclForm.DBOFFSET.Value)
    Pcl.CloseChannel
end if
End Sub
--></script></p>
</HEAD>
<BODY><H3>Call PCL</H3><HR>
<FORM NAME="PCLForm">Please enter the target IP address: <HR>
<INPUT NAME="IPADDR" TYPE="TEXT" SIZE="16">
<INPUT NAME="CallPCL" TYPE="BUTTON" VALUE="CallPCL"><HR>
Please enter the DB number and data word: <HR>
<INPUT NAME="DBNO" TYPE="TEXT" SIZE="2">
<INPUT NAME="DBOFFSET" TYPE="TEXT" SIZE="2">
<INPUT NAME="ReadPCL" TYPE="BUTTON" VALUE="ReadPCL">
<INPUT NAME="DATA" TYPE="TEXT" SIZE="8">
</FORM>
</BODY>
</HTML>

```


6 BoschPlcServer

6.1 General

The event utility (implemented in BoschCOM version 1.4 or later¹) makes it possible to query data of a control system on the computer. This utility is implemented within the framework of COM/DCOM, i.e. as 'Connection Point'. That means that the message can be picked up or not; there is no handshake. If a number of messages arrive on the BoschCOM.DLL, they are routed in the order of arrival. If a handshake is required, this must take place in the PLC program.

It is possible to both start events from control systems and exchange data between different applications (computers).

In the current version all incoming events are entered into a queue and then submitted to the client applications by means of a firing event.

6.2 Licensing

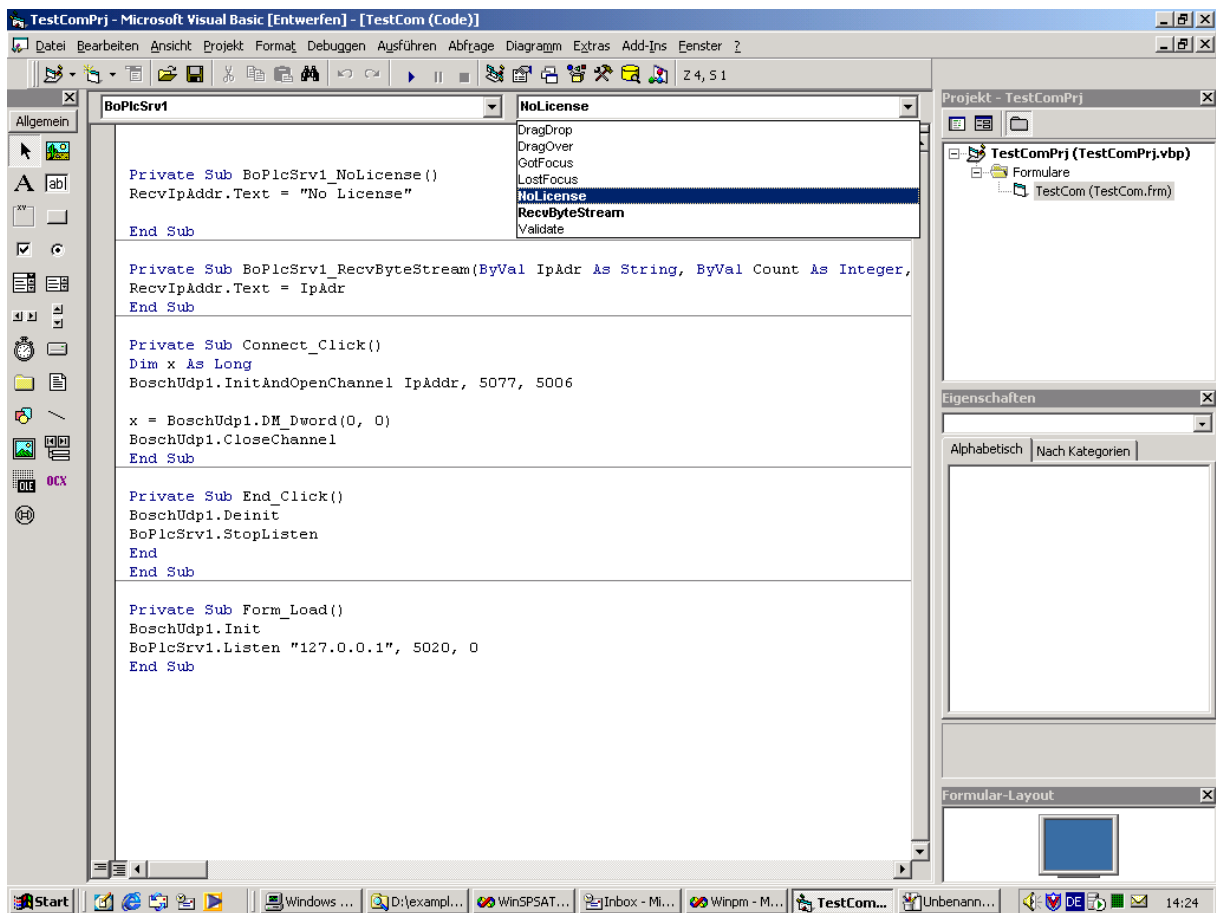
Licensing of this separate component is done as described in chapter 3.3.

6.3 Functions

The BoschPlcServer.OCX file has to be entered into a client application and has to be installed there. The following example shows how this can be achieved by using Visual Basic:

1. In the Basic environment, enter the component 'BoschPlcServer' and enter the component to the own form.
2. The 'BoPlcSrv1' object is created (cf. combo box **Objekt (Object)**)
3. If you select this object, the combo box **Prozedur (Procedure)** shows the events NoLicense and RecvByteStream.
4. If you select these, the subroutines are entered into the code.

¹ BoschCOM version 1.2 already provided a restricted event utility, without queue and with an interface adapted to Visual Basic.



After that, processing can be started via the "Listen" command and it can be stopped via the "StopListen" command.

short Listen (BSTR IpAddr, short Port, short BlockAddr);

This function activates event processing.

IpAddr : currently not used, should be set to 127.0.0.1.
 Port : here the monitored port has to be specified, currently only 5020 is possible.
 BlockAdr: only 0.

short StopListen();

This function deactivates event processing.

BSTR About();

This function can be used to read a string with versioning and licensing information.

After activation of event processing two events can occur:

void RecvByteStream(BSTR IpAdr, short Count, VARIANT Data)

IpAdr: source address of the event
Count: number of data
Data: data as byte stream in a safe array

Normally, this is shown if a license is available.

void NoLicense();

This is shown if no license has been found.

7 BoschOPC

7.1 General

In WinTools, 6 OPC servers are implemented:

<i>File name (.exe)</i>	<i>Name from Client perspective</i>
BOSCHOPC.EXE	CLOPC.CLOPC.1
BOSCHOPC1.EXE	CLOPC1.CLOPC.1
BOSCHOPS2.EXE	CLOPC2.CLOPC.1
BOSCHOPC3.EXE	CLOPC3.CLOPC.1
BOSCHOPC4.EXE	CLOPC4.CLOPC.1
BOSCHOP5.EXE	CLOPC5.CLOPC.1

7.2 Invocation

The application can be parameterized by an invocation switch. These switches can be filed in the BoschOPC.INI file or are attached directly when the application is invoked.

7.2.1 Hardlock/Softlicense

The selection of either Hardlock or Softlicense can be made using the switch '/H':

./H' : (default setting) looking for Hardlock
./h' : looking for Softlicense

7.2.2 Destination Controller IP Address and Port

The BoschOPC server runs on the same computer as the OPC client does. The connection to the destination SPS is done via Ethernet. Therefore, it is necessary to allocate an IP address, default setting is the LocalHost address ,127.0.0.1'. The SoftSPS and the CL550 work with different port addresses. For this reason, the port address has to be specified as well, default setting is 5002.

/I111.222.111.111 (destination IP address)
/P5015 (Port address for CL550)
/P5002 (Port address for SoftSPS)

When using this communication channel, the DCOM security settings can remain unchanged.

7.2.3 Start via OPC Client

If the OPC server is not started explicitly, an OPC client is able to force the start of an OPC server. In this case the invocation switches are needed, therefore a valid 'BoschOPC.ini' file should be available in the directory of the OPC server.

7.2.4 User Interface

The user interface indicates to which controller the OPC server is connected. It also shows the version. By means of the check boxes **Lese Meldungen** (Read Messages) and **Schreib Meldungen** (Write Messages) the command traffic can be monitored.



When the server is stopped, the clients which are still active are notified.

7.3 OPC Data

This chapter describes which data can be read from the PLC, which number of file entries are allowed and what has to be observed concerning the syntax of the data.

7.3.1 Limits

In an OPC server the data is divided in different groups. The assignment of the data to the groups is the task of the OPC client. Because of the controller conditions the following limits result:

The total number of data is restricted to 16k (beginning with version 1.2; prior to that version 2048). Each OPC group must only have 255 data entries.

7.3.2 Data

In principle, there are two possibilities to access data located in the PLC:

- absolute
- symbolic.

'Absolute' means access by absolute operand, in detail these are:

Marker
DataModule
DataField
Inputs
Outputs
SpecialMarker

'Symbolic' means choice from a data list maintained by the PLC. This data list is stored in the controller by the WinSPS software (Version 3.02).

The syntax for the absolute access is simple; ':' serves as a separator.

- for Data Modules:
DM:<Number_of_Data_Module>:<Start_Address>:<Type>
- other: <Operand>:<Start_Address>:<Type>

Start_Address here means an offset in the respective operand area.

The following types are allowed:

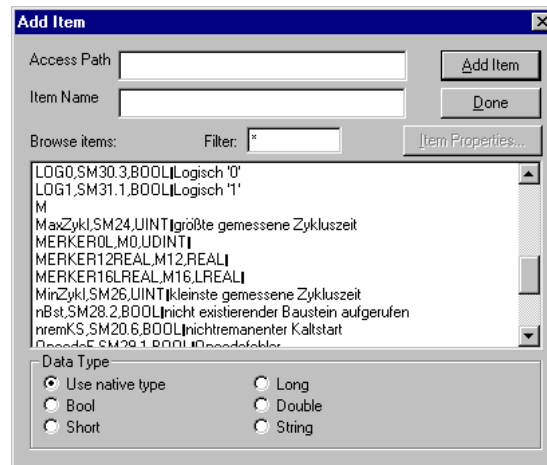
Type identifier	COM Data type	Description
C	VT_I1	signed BYTE (beginning with version 1.2)
B	VT_U1	BYTE
Xn	VT_BOOL	Bit access, whereas ,n' is the bit number (0..7)
Sn	VT_BSTR	String access, whereas ,n' is the number of characters (default setting: 32)
I	VT_I2	signed WORD (beginning with version 1.2)
W	VT_UI2	WORD (2 BYTE) beginning with version 1.2, prior to this version signed WORD
L	VT_I4	Signed Double WORD
F	VT_R4	simple floating point value (4Byte)
D	VT_R8	Double precision floating point value (8Byte)

Examples:

M:32:X3	Bit 3 in MarkerByte 32
DF:1024:L	Double Word in Data Field beginning at Byte 1024
DB:10:32:X4	Bit4 in Data Word 32 of the Daten Module 10
M:0:S24	String beginning at MarkerByte 0

Symbolic Access

Each OPC server has the possibility to 'browse', i.e., on a request by the OPC client the server can report its data. The provided OPC client then indicates this data as follows:



By selection of an entry it is included into the OPC group.

A group in the OPC client of the visualisation software Win Studio / Indusoft Web Studio shows this as follows:

