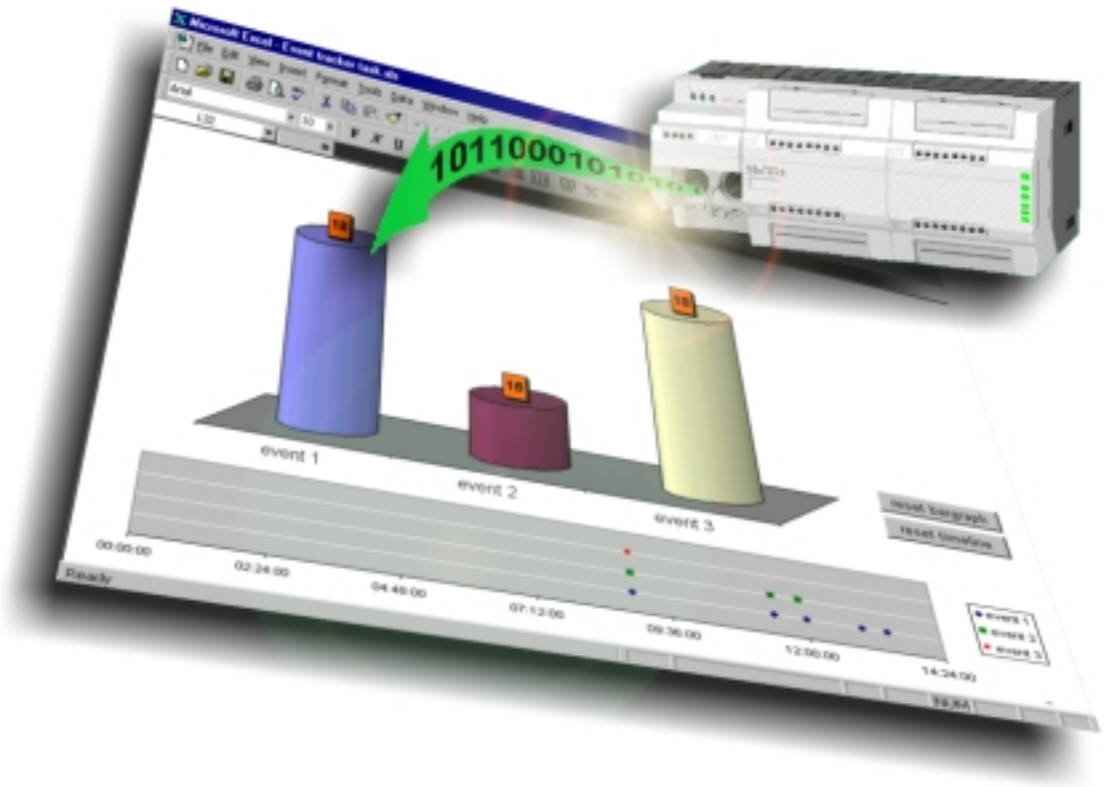
PS40_LINK V3.20
Direct communication between PC and PLC

32 bit DLL for *PS40_LINK* communication between Moeller PS4/PS416 PLCs and PCs running Windows 95/98/NT



Release date: 16 Jan 2001

Table of contents

Introduction

Most industrial applications that are controlled by PLCs require some user interaction in order to collect data, change parameters or to simply operate a machine. In many cases this task can be solved by connecting HMI devices, like the MV4 series from Moeller, to the PLC. Sometimes this is too expensive or not flexible enough.

On the other hand, user interaction can be realized comfortably with PCs which are connected to the PLC. A lot of dedicated visualization software packages are available on the market. The drawback is that these packages are mostly very expensive and sometimes still not flexible enough.

The new *PS40_LINK* DLL opens up a new and inexpensive alternative. By using *PS40_LINK*, standard PC software like Visual Basic, Visual C, Delphi or even MS-EXCEL can be used to create user interfaces for the PLC user interaction or for the visualization of PLC data.

PS40_LINK is a bundle of functions which can be called from e.g. Visual Basic. Each function has a set of call-parameters and return-parameters. These parameters and the functions are explained in this documentation.

For installation of *PS40_LINK* copy the DLL file “sucoma32.dll” in the folder containing your application or in the Windows system folder. Use the appropriate function declarations in your application or Visual Basic program just by using “cut-and-paste” from the file “PS40_LINK_API.txt” which contains all function declarations, error codes and other useful constants.

The new features for Version 3.0 are:
- Up to 490% faster when compared to V2.x (see chapter “Technical data” for more details)
- Selectable baudrate parameter for PS4-341-MM1 and PS416 PLC (See reference of *openComDevice* on page 14.
- Platform for further companion products (like modem connections)
- Brief discussion of performance parameters

The new features for Version 3.10 are:
- Platform for remote TCP/IP connections. You need the additional product “PS40_TCPIP_Link” to use this feature.

The new features for Version 3.20 are:
- Automatic baudrate setting for PS4-341-MM1 and PS416 PLC over remote TCP/IP connections.
- Minor documentation improvements

Function overview

Basic functions:

Function name	Description
GetSucomVersion	Returns the version of the PS40_LINK product
GetVersion	Returns the version of the PS40_LINK Extension products (e.g. PS40_Modem_Link.dll)
WriteDataToPLC	Writes one single data element to the PLC marker range
ReadDataFromPLC	Reads one single data element from the PLC marker range

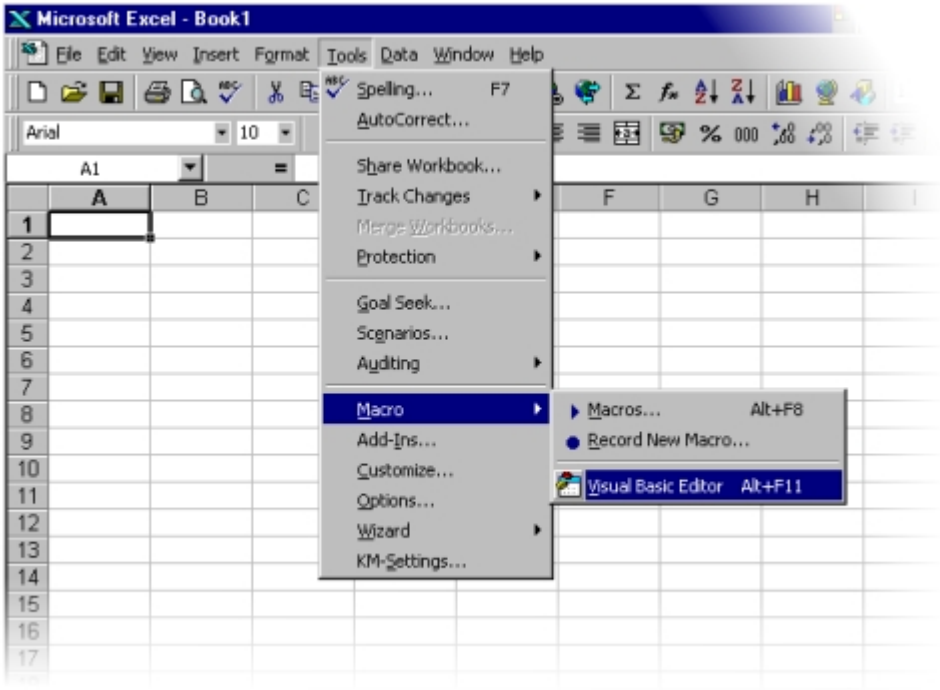
Advanced functions:

Function name	Description
WriteDataToPLCex	Writes a variable amount of data elements to the PLC marker range
ReadDataToPLCex	Reads a variable amount of data elements from the PLC marker range
SetPLCType	This function sets the PLC type for all subsequent function calls
OpenComDevice	Accelerator for read/write functions. This function opens and initializes a COM port interface of the PC.
CloseComDevice	Close the previously opened PC COM port.
SetSucomOptions	Set optional parameters for the operation of the PS40_LINK product

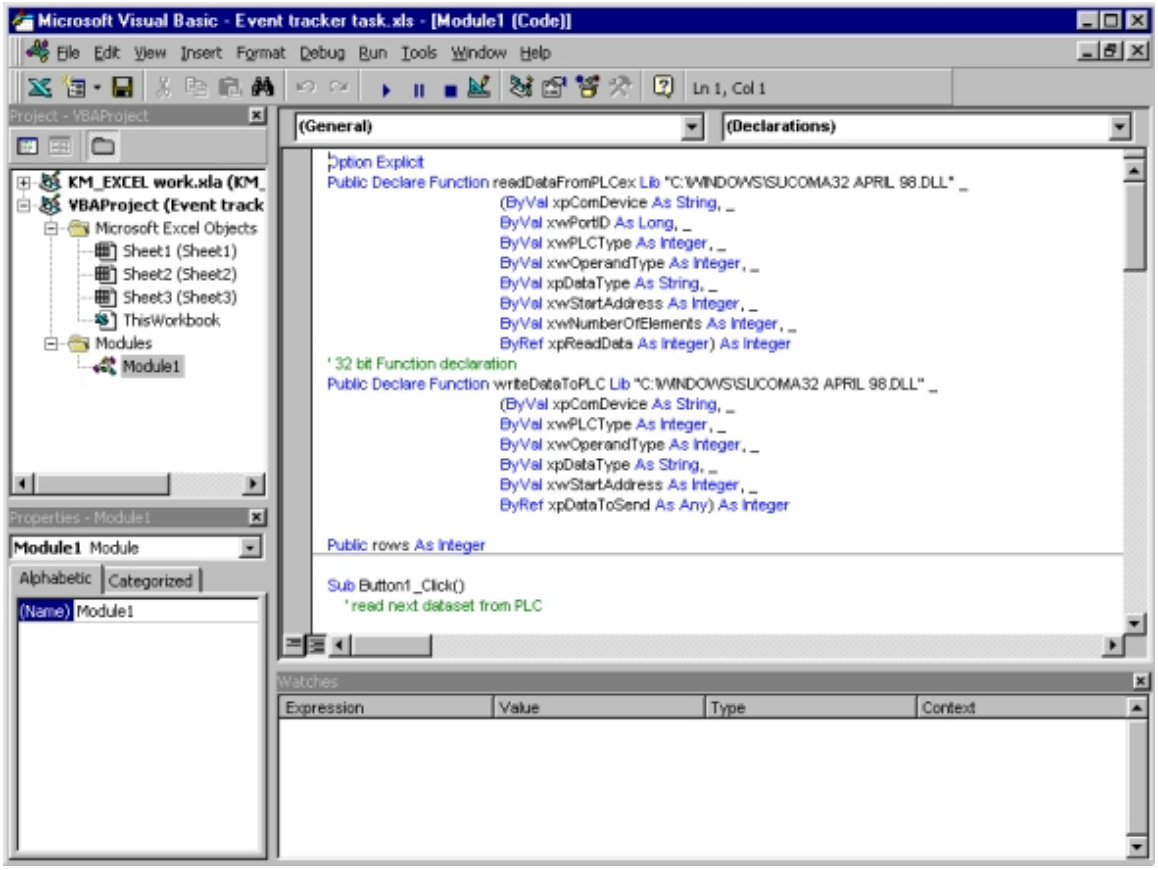
Quick start

The easiest way to learn to use the *PS40_LINK* is to follow this little example. It explains how to use the functions from within VBA (Visual Basic for Applications).

VBA is a core part of Microsoft EXCEL 95/97, WORD 95/97 and ACCESS 95/97. It can be used to write macros, programs or new EXCEL spreadsheet functions. The VBA editor can be opened from the EXCEL TOOLS menu:



The VBA screen looks like this:



The PLC detects external events and adds a time and date stamp to store these events in a stack (The entire example can be found on the accompanying disk.). By using the function ***readDataFromPLCex*** these events are transmitted into the PLC and displayed in an EXCEL spreadsheet.

The function ***writeDataToPLC*** is used to organize the handshake communication between the PLC user program and the VBA program[1].

```
Private Declare Function getSucomVersion Lib "sucoma32.dll" _
                         (ByVal xpVersionBuffer As String) As Integer


Private Declare Function readDataFromPLCex Lib "sucoma32.DLL" _
                         (ByVal xpComDevice As String, _
                          ByVal xwPortID As Long, _
                          ByVal xpDataType As String, _
                          ByVal xwStartAddress As Integer, _
                          ByVal xwNumberOfElements As Integer, _
                          ByRef xpReadData As Any) As Integer
```

[1] In Visual Basic one instruction can be spread over multiple lines by using the characters *SPACE + UNDERSCORE* to connect the lines.

```
Private Declare Function writeDataToPLC Lib "sucoma32.DLL" _
                        (ByVal xpComDevice As String, _
                         ByVal xpDataType As String, _
                         ByVal xwStartAddress As Integer, _
                         ByRef xpDataToSend As Any) As Integer
```

The declarations above are necessary to register the DLL functions with VBA. It also shows which parameters are necessary and what datatype they have.

All results of PS40_LINK functions are 16 bit values. The declaration for the result variable in VBA is accordingly:

```
Dim wResult As Integer
```

In the PLC the data set to be read consists of 9 elements. Each element is a 16 bit integer number. The data set is organized as follows:

Element 1: actual event number
Element 2: timestamp – seconds
Element 3: timestamp – minutes
Element 4: timestamp – hour
Element 5: datestamp – day
Element 6: datestamp – month
Element 7: datestamp – year
Element 8: reserved
Element 9: total number of events still stored in PLC

The easiest way to allocate the target memory in the PC is to declare an array of the appropriate size:

```
Dim MarkerValue(9) As Integer
```

Since VBA is an event oriented programming language, the user has to create an event in order to initiate the execution of the VBA program. One possible event could be to click a button on a spreadsheet. Creating a button on a spreadsheet is very easy:

After right clicking on the EXCEL toolbar area a menu appears that allow to add new toolbars. Please select the FORMS toolbar:

Now it is possible to drag & drop a button from the toolbar on a spreadsheet. A VBA function (***Button1_Click***) is created automatically and can be edited in the VBA editor. Each time this new button on the spreadsheet is being clicked, this function will be executed.

The user program has first to check if the DLL version (at least version V2.00) is correct. The function ***getSucomVersion*** requires a pointer to a string as a call parameter[2].

```
Dim aVersion As String*32

Sub Button1_Click ()

aVersion = "                    "
wResult = getSucomVersion( aVersion )
If  wResult < 200 Then
        ' wrong version
        Msgbox “Data access not possible. Please upgrade to a newer DLL version.”
        Exit Sub
End If
```

[2] In VBA and VB, strings are always pointers (ByRef) even if they are declared *ByVal* in the prototype declaration.

Now the program reads the first dataset (9 elements), located in the PLC at markerword `MW0`. The elements in the PLC have the datatype `WORD` and will be stored in the previously created array *MarkerValue*.

```
wResult = readDataFromPLCex (
                        xpComDevice := "COM1",  _       'communication via COM1
                        xwPortID := 0,  _               'default
                        xpDataType := "WORD", _         ' interpret incoming data as WORD
                        xwStartAddress := 0, _          ' Startaddress in PLC = MW0
                        xwNoOfElements := 9, _          '  read 9 elements
                        xpReadData := MarkerValue(0))   ' target memory location
```

The functions call was successful only if the return parameter *wResult* has a value of zero.

```
If wResult <> 0 Then
        ' Error
        MsgBox “Communication Error. See errorcode: “ & wResult
        Exit Sub
End If
```

The PLC program requires a rising edge in markerword `MW18`. This causes the PLC program to replace the actual (which is the oldest event) event with the next younger event in the stack. The function ***writeDataToPLC*** is used to first write the value 0 and then the value 1 to `MW18`, thus generating the rising edge. (It can be assumed that the PLC program is fast enough so that this sequence will be interpreted as rising edge.)

```
wResult = writeDataToPLC (
                xpComDevice := "COM1", _
                xpDataType := "WORD", _
                xwStartAddress := 18, _
                xpDataToSend := 1 )

wResult = writeDataToPLC (
                xpComDevice := "COM1",  _
                xpDataType := "WORD", _
                xwStartAddress := 18, _
                xpDataToSend := 0 )
```

The easiest way to show these results in EXCEL is to write them into spreadsheet cells. It is then easy to create charts and graphs once the information is organized in a table. In order to organize the dataset in a table on Sheet 1, the following VBA commands could be used:

```
Dim I As Integer
For I = 0 To 8
        ThisWorkbook.Worksheets(“Sheet1”).Range(“A1”).Offset(0,I).value = MarkerValue(I)
Next I
End Sub
```

The entire example can be found on the accompanying disk.

Reference of the PS40_LINK functions

The gray boxes in this chapter contain the function declarations for Visual Basic or VBA. The other declarations are described in C/C++ syntax.

getSucomVersion

```
Private Declare Function getSucomVersion Lib " sucoma32.dll"_
                                (ByVal xpVersionBuffer As String) As Integer
```

Description:	Reads the actual version of PS40_LINK.
Call parameters:	
*char** xpVersionBuffer	If the pointer is not equal null, the DLL version text is being copied to this pointer location. The minimum length of the buffer is 32 characters! The version string use the following format: [*Name of the DLL/Extension*] [*Version*] [*Release date*] The next line is an example for the basic PS40_Link DLL: *PS40_Link V3.20 Jan 3 2001*
Return value:	
	The version of the DLL as numerical value. E.g. version 3.20 is 320

getVersion

```
Private Declare Function getVersion Lib " sucoma32.dll"_
                                (ByVal xpExtensionName As String,_
                                 ByVal xpVersionBuffer As String) As Integer
```

Description:	Reads the actual version of PS40_LINK and all Extension DLLs.
Call parameters:	
*Char** xpExtensionName	Description of the PS40_LINK Extension name. The following Extension names are supported: "SUCOMA32": The basic PS40_LINK DLL "MODEM_LINK": The Modem Extension DLL "TCPIP_LINK: The TCP/IP Extension DLL "TRANSFER_LINK":The Download Extension DLL If the pointer is Null, the basic PS40_LINK version number is returned.
*Char** xpVersionBuffer	If the pointer is not equal null, the Extension version text is being copied to this pointer location. The minimum length of the buffer is 32 characters! The version string use the following format: [*Name of the DLL/Extension*] [*Version*] [*Release date*] The next line is an example for the basic PS40_Link DLL: *PS40_Link V3.20 Jan 3 2001*
Return value:	The version of the PS40_LINK Extension as numerical value. E.g. version 3.20 is 320 or Error code – see reference.

writeDataToPLC

```
Private Declare Function writeDataToPLC Lib "sucoma32.dll" _
                                ( ByVal xpComDevice As String,_
                                  ByVal xpDataType As String,_
                                  ByVal xwStartAddress As Integer,_
                                  ByRef xpDataToSend As Any) As Integer
```

Description:	Writes one data element to the PLC marker range via serial interface.
Call parameters:	
*Char** xpComDevice	Description of the name of the serial PC interface. All serial interfaces supported by Windows can be used. Usually this is "COM1", "COM2", "COM3" and "COM4" (Note: No ending colon!) Additionally you are able to select a baudrate value, other than the default 9600 baud as well as other connection parameters. See chapter "Specify connection parameter" for more information.
*char** xpDataType	This parameter allows specifying the datatype which is required in the PLC user program. Although each data element is stored within Visual Basic as 32 bit variable, it might be appropriate for the PLC to receive only part of the information – e.g. the first bit – in order to save PLC memory and to simplify the PLC program. In addition to this, Visual Basic interprets the bit-pattern for negative numbers differently than the PLC. The following parameters convert 32 bit VBA data elements (VBA datatype *long*) into correct PLC datatypes: VBA ***Long***[3] => PLC ***BYTE*****:** **"VBBYTE"** VBA ***Long*** => PLC ***WORD*****:** **"VBWORD"** VBA ***Long*** => PLC ***SINT***: **"VBSINT"** VBA ***Long*** => PLC ***USINT***: **"VBUSINT"** VBA ***Long*** => PLC ***INT*** : **"VBINT"** VBA ***Long*** => PLC ***UINT*** : **"VBUINT"** VBA ***Long*** => PLC ***DINT*** : **"VBDINT"** VBA ***Long*** => PLC ***UDINT*****:** **"VBUDINT"** The following datatypes can be used if the source data is not of type *long*. VBA ***Byte***[5] => PLC ***BYTE***: "**BYTE**" VBA ***Integer***[6] => PLC ***WORD***: "**WORD**" VBA ***Long*** => PLC ***DINT***: "**LONG**"
Unsigned short xwStartAddress	This parameter contains the start address (e.g. markerword) for the target memory in the PLC. Please consider the following specialties: PLC type 3: The maximum PLC memory available is 2172 markerwords. Do not overwrite PLC memory beyond this limit. PS316: The PS316 has a logical limitation at the markerword 124 (MW124.0). It is not allowed to overwrite this limit. Eventually two write tasks have to be performed. PLC type 5: Addresses outside the range of 8 to 66 are not allowed. PLC type 7: The available PLC marker range has to be defined within the S40 (program generation). Addresses outside this range are not allowed.
*Void** xpDataToSend	Points to the source data in the calling program.
Return value:	
	Error code – see reference (0 if data transfer ok!)

[3] VBA *long* = C/C++ *int, unsigned int*
[4] VBA *long* = C/C++ *int, unsigned int*
[5] VBA *Byte* = C/C++ *char, unsigned char*
[6] VBA *Integer* = C/C++ *short, unsigned short*

readDataFromPLC

```
Private Declare Function readDataFromPLC Lib "sucoma32.dll"_
                                ( ByVal xpComDevice As String,_
                                  ByVal xpDataType As String,_
                                  ByVal xwStartAddress As Integer,_
                                  ByRef xpReadData As Any) As Integer
```

Description:	Reads one data element from the PLC marker range via serial interface.
Call parameters:	
*Char** xpComDevice	Description of the name of the serial PC interface. All interfaces supported by Windows can be used. Usually this is "COM1", "COM2", "COM3" and "COM4" (Note: No ending colon!) Additionally you are able to select a baudrate value other than the default 9600 baud as well as other connection parameters. See chapter "Specify connection parameter" for more information.
*char** xpDataType	This parameter allows specifying the datatype which is used in the PLC user program. Although each data element is stored within Visual Basic as 32 bit variable, the PLC uses different datatypes. In addition to this, Visual Basic interprets the bit-pattern for negative numbers differently than the PLC. The following parameters converts PLC datatypes into correct 32 bit VBA data elements (VBA datatype *long*): PLC ***BOOL*** => VBA ***Long***[7]: **"VBBOOL"**[8] PLC ***BYTE*** => VBA ***Long***: **"VBBYTE"** PLC ***WORD*** => VBA ***Long***: **"VBWORD"** PLC ***SINT*** => VBA ***Long***: **"VBSINT"** PLC ***USINT*** => VBA ***Long***: **"VBUSINT"** PLC ***INT*** => VBA ***Long***: **"VBINT"** PLC ***UINT*** => VBA ***Long***: **"VBUINT"** PLC ***DINT*** => VBA ***Long***: **"VBDINT"** PLC ***UDINT*** => VBA ***Long***: **"VBUDINT"** The following datatypes can be used if the destination data in the PC is not of type *long*. PLC **BOOL** => VBA ***Boolean***[9]: "**BOOL**"[8] PLC **BYTE** => VBA ***Byte***[10]: "**BYTE**" PLC **WORD** => VBA ***Integer***[11]: "**WORD**" PLC **DINT** => VBA ***Long***: "**LONG**"
Unsigned short xwStartAddress	This parameter contains the start address (e.g. markerword) for the source memory in the PLC. Please consider the following specialties: PLC type 3: The maximum PLC memory available is 2172 markerwords. Do not try to read PLC memory beyond this limit. PS316: The PS316 has a logical limitation at the markerword 124 (MW124.0). It is not allowed to read over this limit with the same function call. Eventually two read calls have to be performed. PLC type 5: Addresses outside the range of 8 to 66 are not allowed. PLC type 7: The available PLC marker range has to be defined within the S40 (program generation). Addresses outside this range are not allowed.
*void** xpReadData	Points to the target memory area in Visual Basic. Usually this is a variable or an array of the appropriate size. **CAUTION:** If not enough memory is available in the PC program, the PC very likely will crash!
Return value:	
	Error code – see reference (0 if data transfer ok!)

[7] VBA *long* = C/C++ *int, unsigned int*
[8] See Handling issues at the end of the chapter *readDataFromPLCex*
[9] VBA *Boolean* = C/C++ *short*
[10] VBA *Byte* = C/C++ *char, unsigned char*
[11] VBA *Integer* = C/C++ *short, unsigned short*

writeDataToPLCex

```
Private Declare Function writeDataToPLCex Lib "sucoma32.dll"_
                                  ( ByVal xpComDevice As String,_
                                    ByVal xwPortID As Long,_
                                    ByVal xpDataType As String,_
                                    ByVal xwStartAddress As Integer,_
                                    ByVal xwNoOfElements As Integer,_
                                    ByRef xpDataToSend As Any) As Integer
```

Description:	Writes multiple data elements to the PLC marker range via serial interface.
Call parameters:	
*char** xpComDevice	Description of the name of the serial PC interface. All serial interfaces supported by Windows can be used. Usually this is "COM1", "COM2", "COM3" and "COM4" (Note: No ending colon!) If the COM port has been opened via xwPortID parameter, then this parameter is the NULL pointer. Additionally you are able to select a baudrate value other than the default 9600 baud as well as other connection parameters. See chapter "Specify connection parameter" for more information.
int xwPortID	If this value is zero, then the serial COM interface will be opened and closed for each function call automatically. If the COM port is required for a longer time period, then this parameter should contain the handle to a COM interface that has been opened previously with the function *openComDevice*. The port has to be closed later manually with the function *closeComDevice*.
*char** xpDataType	This parameter allows specifying the datatype which is required in the PLC user program. Although each data element is stored within Visual Basic as 32 bit variable, it might be appropriate for the PLC to receive only part of the information – e.g. the first bit – in order to save PLC memory and to simplify the PLC program. In addition to this, Visual Basic interprets the bit-pattern for negative numbers differently than the PLC. The following parameters convert 32 bit VBA data elements (VBA datatype *long*) into correct PLC datatypes: VBA ***Long*** => PLC ***BYTE*:** **"VBBYTE"** VBA ***Long*** => PLC ***WORD*:** **"VBWORD"** VBA ***Long***[12] =>PLC ***SINT***: **"VBSINT"** VBA ***Long*** => PLC ***USINT***: **"VBUSINT"** VBA ***Long*** => PLC ***INT*** : **"VBINT"** VBA ***Long*** => PLC ***UINT*** : **"VBUINT"** VBA ***Long*** => PLC ***DINT*** : **"VBDINT"** VBA ***Long*** => PLC ***UDINT*:** **"VBUDINT"** The following datatypes can be used if the source data is not of type *long*. VBA ***Byte***[13] => PLC ***BYTE***: "**BYTE**" VBA ***Integer***[14] => PLC ***WORD***: "**WORD**" VBA ***Long*** => PLC ***DINT***: "**LONG**"
unsigned short xwStartAddress	This parameter contains the start address (e.g. markerword) for the target memory in the PLC. Please consider the following specialties: PLC type 3: The maximum PLC memory available is 2172 markerwords. Do not overwrite PLC memory beyond this limit. PS316: The PS316 has a logical limitation at the markerword 124 (MW124.0). It is not allowed to overwrite this limit. Eventually two write tasks have to be performed. PLC type 5: Addresses outside the range of 8 to 66 are not allowed. PLC type 7: The available PLC marker range has to be defined within the S40 (program generation). Addresses outside this range are not allowed.
unsigned short xwNoOfElements	Specifies the number of data elements that are transferred into the PLC. The maximum number of elements that can be transfered depends only on the PLC type and the marker declarations.
*Void** xpDataToSend	Points to the source data in the calling program. The size of the defined memory location must be as least as big as specified with the parameter *NoOfElements*.
Return value:	Error code – see reference (0 if data transfer ok!)

[12] VBA *long* = C/C++ *int, unsigned int*
[13] VBA *Byte* = C/C++ *char, unsigned char*
[14] VBA *Integer* = C/C++ *short, unsigned short*

readDataFromPLCex

```
Private Declare Function readDataFromPLCex Lib "sucoma32.dll"_
                                  ( ByVal xpComDevice As String,_
                                    ByVal xwPortID As Long,_
                                    ByVal xpDataType As String,_
                                    ByVal xwStartAddress As Integer,_
                                    ByVal xwNoOfElements As Integer,_
                                    ByRef xpReadData As Any) As Integer
```

Description:	Reads multiple data element from the PLC marker range via serial interface.
Call parameters:	
*Char** xpComDevice	Description of the name of the serial PC interface. All serial interfaces supported by Windows can be used. Usually this is "COM1", "COM2", "COM3" and "COM4" (Note: No ending colon!) If the COM port has been opened via xwPortID parameter, then this parameter is the NULL pointer. Additionally you are able to select a baudrate value other than the default 9600 baud as well as other connection parameters. See chapter "Specify connection parameter" for more information.
int xwPortID	If this value is zero, then the serial COM interface will be opened and closed for each function call automatically. If the COM port is required for a longer time period, then this parameter should contain the handle to a COM interface that has been opened previously with the function *openComDevice*. The port has to be closed later manually with the function *closeComDevice*.
*char** xpDataType	This parameter allows specifying the datatype which is used in the PLC user program. Although each data element is stored within Visual Basic as 32 bit variable, the PLC uses different datatypes. In addition to this, Visual Basic interprets the bit-pattern for negative numbers differently than the PLC. The following parameters converts PLC datatypes into correct 32 bit VBA data elements (VBA datatype *long*): PLC ***BOOL*** => VBA ***Long***[15]: **"VBBOOL"**[16] PLC ***BYTE*** => VBA ***Long***: **"VBBYTE"** PLC ***WORD*** => VBA ***Long***: **"VBWORD"** PLC ***SINT*** => VBA ***Long***: **"VBSINT"** PLC ***USINT*** => VBA ***Long***: **"VBUSINT"** PLC ***INT*** => VBA ***Long***: **"VBINT"** PLC ***UINT*** => VBA ***Long***: **"VBUINT"** PLC ***DINT*** => VBA ***Long***: **"VBDINT"** PLC ***UDINT*** => VBA ***Long***: **"VBUDINT"** The following datatypes can be used if the destination data in the PC is not of type *long*. PLC **BOOL** => VBA ***Boolean***[17]: "**BOOL**"[16] PLC **BYTE** => VBA ***Byte***[18]: "**BYTE**" PLC **WORD** => VBA ***Integer***[19]: "**WORD**" PLC **DINT** => VBA ***Long***: "**LONG**"

[15] VBA *long* = C/C++ *int, unsigned int*

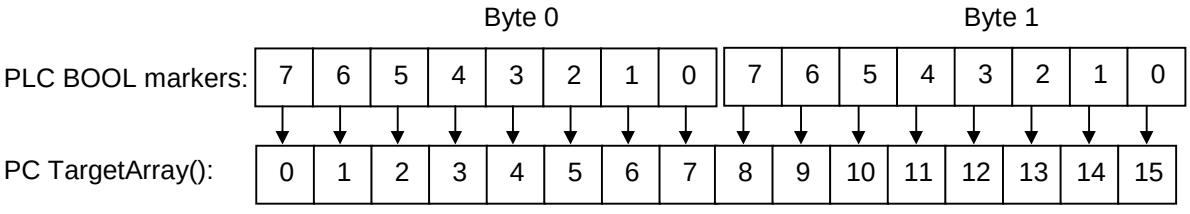
[16] See Handling issues at the end of this chapter

[17] VBA *Boolean* = C/C++ *short*
[18] VBA *Byte* = C/C++ *char, unsigned char*
[19] VBA *Integer* = C/C++ *short, unsigned short*

unsigned short xwStartAddress	This parameter contains the start address (e.g. markerword) for the source memory in the PLC. Please consider the following specialties: PLC type 3: The maximum PLC memory available is 2172 markerwords. Do not try to read PLC memory beyond this limit. PS316: The PS316 has a logical limitation at the markerword 124 (MW124.0). It is not allowed to read over this limit with the same function call. Eventually two read calls have to be performed. PLC type 5: Addresses outside the range of 8 to 66 are not allowed. PLC type 7: The available PLC marker range has to be defined within the S40 (program generation). Addresses outside this range are not allowed.
unsigned short xwNoOfElements	Specifies the number of data elements that are transferred from the PLC. The maximum number of elements that can be transferred depends only on the PLC type and the marker declarations.
*void** xpReadData	Points to the target memory area in the calling program. Usually this is a variable or an array of the appropriate size. **CAUTION**: If not enough memory is available in the PC program, the PC very likely will crash!
Return value:	
	Error code – see reference (0 if data transfer ok!)

Handling issues for BOOL and VBBOOL datatypes:

- It is not possible to read single BOOL markers from the PLC
- The functions *readDataToPLC* and *readDataToPLCex* always read and stores quantities of eights BOOL markers because the smallest entity which can be transferred from the PLC is a single byte.
- The amount of bytes actually transferred can be calculated with the following formula:
  NoOfTransferredBytes = (NoOfBoolMarkers - 1)/8 + 1
  If you want to read for example nine BOOL markers, then two bytes are read from the PLC.
- The amount of transferred and stored BOOL marker depends solely on the amount of transferred bytes (see previous note). Please allocated eight data elements in your program per transferred byte. For example:
  You want to read one BOOL marker with the VBBOOL datatype. Than the functions reads one byte and stores eight VBA long elements in your target memory. Because of that, you have to declare at least the following array: *Dim TargetArray(8) As Long*
- BOOL markers are stored in the following manner (two byte/16 bool markers are read and stored in the target array in your application program):



setPLCType

```
Private Declare Function setPLCType Lib "sucoma32.dll"_
                                ( ByVal xwPLCType As Integer ) As Integer
```

Description:	Set the PLC type for all subsequent function calls, if you use a PS3xx or a PS4-101/111
Call parameters:	
unsigned short xwPLCType	The target PLC type has to be specified by this parameter. The following constants are applicable: PS306, PS316, (PS416) : **3** PS3, PS4-101, PS4-111 : **5** PS4-141, PS4-151, PS4-201, PS4-271, PS4-341, PS416-CPU-200/300/400 : **7** After calling this function all other PS40_LINK function assumes the specified PLC type. The default type is ´7´ which is applicable to all PS4 PLC´s except the PS4-101/111. So, you need to call this function only if you use an PS3xx or PS4-101/111 PLC (type 3 or 5). The use of constants for the PLC types in your program helps getting an easier reading of your program code.
Return value:	
	Error code – see reference (0 if data transfer ok!)

openComDevice

```
Private Declare Function openComDevice Lib "sucoma32.dll"_
                                ( ByVal xpComDevice As String,_
                                  ByRef xwPortID As Long) As Integer
```

Description:	Opens and initializes a COM port to accelerate the functions *readDataFromPLCex* and *writeDataToPLCex* for extensive usage. Multiple COM ports can be opened and therefor multiple PLC connections can be established at the same time.
Call parameters:	
*char** xpComDevice	Description of the name of the serial PC interface. All serial interfaces supported by Windows can be used. Usually this is "COM1", "COM2", "COM3" and "COM4" (Note: No ending colon!) Additionally you are able to select a baudrate value other than the default 9600 baud as well as other connection parameters. See chapter "Specify connection parameter" for more information.
Int xwPortID*	After a successful function call this parameter contains a handle to the specified COM port.
Return value:	
	Error code – see reference (0 if data transfer ok!)

closeComDevice

```
Private Declare Function closeComDevice Lib "sucoma32.dll"_
                                (ByVal xwPortID As Long) As Integer
```

Description:	Closes a COM port that has been opened previously with the function *openComDevice*.
Call parameters:	
int xwPortID	The port handle which has been obtained with the function call *openComDevice*. **CAUTION:** If Visual Basic is being terminated without closing all COM ports, then these ports are not available for other Windows applications.
Return value:	
	Error code – see reference (0 if data transfer ok!)

setSucomOptions

```
Private Declare Function setSucomOptions Lib "sucoma32.dll"_
                         ( ByVal  xpOptionName As String,_
                           ByVal  wOptionValue As Long) As Integer
```

Description:	Set specific PS40_LINK options through individual option names and values.
Call parameter:	
*char** **xpOptionName**	The desired option has to be named by this parameter. The following option-names are applicable for this PS40_LINK version:
	MODEM_CALLBACK With this option you define an application specific callback function, which is called everytime in the lifetime of a remote modem session, when a state change occurs, e.g. from „proceeding“ to „connect“. The parameter *wOptionValue* has to point to (or address) a callback function. Once set, this parameter remains valid during the whole usage of PS40_LINK and can be disabled through a call to this function with option-value NULL. To receive all state-changes you have to activate this option before you open a remote connection. The callback function must have the following prototype, whereas the name of the function is to your liking: **Public Function ShowModemProgress(ByVal xwProgress As Long) As Long** The callback function can be called with the following constants for the parameter *xwProgress*: MODEM_DIAL_TONE 1 => A dial tone is detected MODEM_DIALING 2 => The phone number will be dialed MODEM_PROCEEDING 3 => The call is under way MODEM_CONNECT 4 => A successful connection is established MODEM_DISCONNECT 5 => The line is disconnected (remote or local) MODEM_NO_CONNECTION 6 => No connection could be established (The appropriate function (e.g. openComDevice) will return with an error code. This function could be called, even when no active PS40_LINK call is under way. The function should always return the value 0. **Note:** This option applies only to remote modem connection.

	TCPIP_CALLBACK With this option you define an application specific callback function, which is called if a remote TCP/IP session should be established and the automatic baudrate setting sequence is to be activated. The parameter *wOptionValue* has to point to (or address) a callback function. Once set, this parameter remains valid during the whole usage of the PS40_LINK and can be disabled through a call to this function with option-value NULL. The callback function must have the following prototype, whereas the name of the function is to your liking: **Public Function ShowTCPIPConnection(ByVal xwProgress As Long) As Long** The callback function can be called with the following constants for the parameter *xwProgress*: *TCPIP_OPEN_SEQ* 1 => A Ethernet/Serial gateway is detected and a try to open the PLC connection with the given baudrate (specified in the connection parameter string) will be made when the callback function returns with zero. *TCPIP_START_SEQ* 2 => A connection to the PLC with the given baudrate could not be established. If the callback function returns with zero the automatic baudrate setting sequence will be activated. *TCPIP_END_SEQ* 3 => The baudrate setting sequence succeeds and a connection to the PLC with the given baudrate (specified in the connection parameter string) is established. *TCPIP_SEQ_FAILED* 4 => The baudrate setting sequence fails and no connection to the PLC could be established. Possible reasons are: - PLC not connected to the gateway, - wrong or faulty cable, - faulty serial gateway interface, - given baudrate not supported by the PLC, - PLC is set to a baudrate not supported by the gateway *2400, 4800, 9600, 19200, 38400, 57600* The baudrate setting sequence is active with the given baudrate value *xwProgress* This functions could be called, even when no active PS40_LINK call is under way. The function should always return the value 0. If this function return a non-zero value, the baudrate sequence will be canceled and the PS40_LINK operation (e.g. (openComDevice) will return with the error code SUCOM_BREAK_BY_USER (15). **Note:** This option applies only to remote TCP/IP connections.
int **wOptionValue**	This parameter set the current value of the selected option. The meaning is option specific. See description of supported options above.
Return value:	Error code – see reference SUCOM_OK (0): Option setting is ok! SUCOM_PARAMETER_ERROR (5): Option or option value is invalid.

Visual Basic example to setup the TCP/IP progress function:

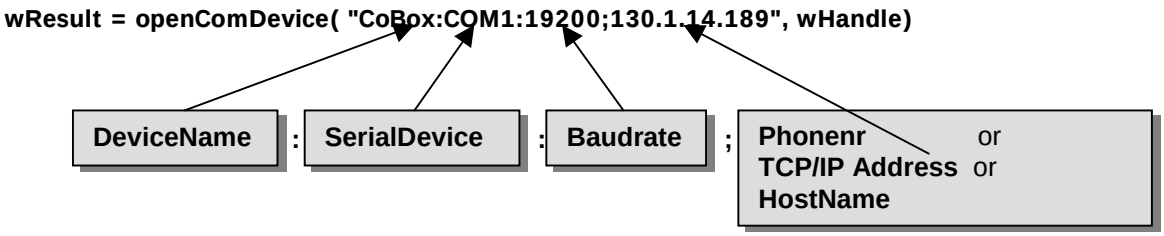
```
´Declare the function „setSucomOptions“ as follows
Private Declare Function setSucomOptions Lib "sucoma32.dll" ( ByVal xpOptionName As
String, ByVal xwOptionValue As Long) As Integer


´Set the adress of your TCP/IP progress function, with the name „ShowTCPIPConnection“
´Set the TCPIP_CALLBACK option before you open the device !!!
 wResult = setSucomOptions( "TCPIP_CALLBACK", AddressOf ShowTCPIPConnection )


´This is the declaration of your modem progress function
´Use the parameter „xwProgress“ to select the actual state of the TCP/IP connection
Public Function ShowTCPIPProgress( ByVal xwProgress As Long ) As Long
        Dim wResult As Integer


        ShowTCPIPProgress = 0

        Select Case xwProgress
        Case TCPIP_OPEN_SEQ
                Form1.Caption = "Baudrate Wizard: Try to connect with Cobox...."
                Form1.MousePointer = vbHourglass
        Case TCPIP_START_SEQ
                wResult = MsgBox("No connection could be established !" + vbCrLf + "Do
                you want to start the Baudrate setting sequence ?",
                vbYesNo+vbQuestion+bMsgBoxSetForeground, "Baudrate Wizard")
                If ( wResult = vbNo ) Then
                        Form1.Caption = ""
                        Form1.MousePointer = 0
                        ShowTCPIPProgress = 1
                Else
                        Form1.MousePointer = vbHourglass
                End If
        Case TCPIP_END_SEQ
                Form1.Caption = ""
                Form1.MousePointer = 0
        Case Else
                Form1.Caption =
                        "Baudrate Wizard: Try to connect @ " + Str(xwProgress) + " Baud"
        End Select
        DoEvents
End Function
```

Specify connection parameter

The following PS40_LINK functions use the „xpComDevice“ string parameter (which is always the first function parameter) to determine the requested connection type:

- writeDataToPLC, readDataFromPLC,
- writeDataToPLCex, readDataFromPLCex,
- openComDevice

The parameter string itself is divided into four fields, which are separated by colons “:” or a semicolon “;”. Please take a look on the following example code snippet, to see how the parameter string matches the different fields:



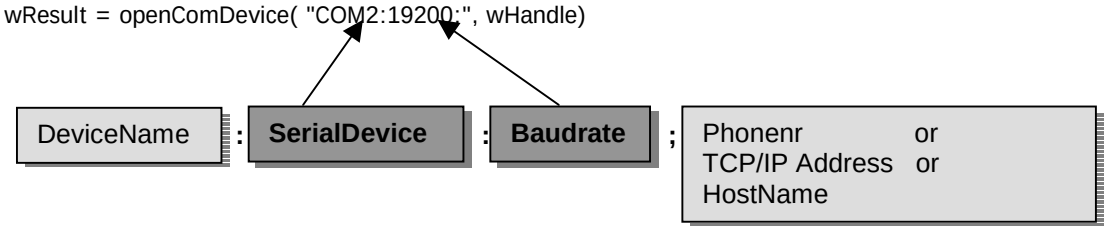
The next table describes in detail the meaning and possible values for the parameter fields.

Field name	***Description***	**Example**
DeviceName	This field names the Ethernet/Serial gateway device. Actual the PS40_Link support the „CoBox“ device. **Default:** „CoBox“	CoBox
SerialDevice	This field selects a specific serial device either locally in your PC or remotely if you specify a remote Ethernet/Serial gateway. **Default:** COM1	COM1
Baudrate	This parameter adjust the baudrate between the PC or a remote Ethernet/Serial gateway and the PLC. **If you use a modem connection make sure the PLC is set to the appropriate baudrate (the baudrate could not be set automatically in this case)!** The table below shows the supported baudrates: 2400 4800 9600 19200 38400 57600 PS3/PS306/PS316 ① --- --- ✓ --- --- --- PS4-111/141/151/201/271 ① --- --- ✓ --- --- --- PS4-341-MM1 PRG port ② ✓ ✓ ✓ ✓ ✓ ✓ PS4-341-MM1 with SBI ③ ✓ ✓ ✓ ✓ --- --- In Transparent mode PS416-CPU-x00 PRG port ② ✓ ✓ ✓ ✓ ✓ ✓ PS416-CPU-x00 with SBI ③ ✓ ✓ ✓ ✓ --- --- In Transparent mode PS416-COM-200 ③ ✓ ✓ ✓ ✓ --- --- Please note the following remarks for the different PS40_LINK connections to a PLC: ① For this PLCs the baudrate to fixed to 9600 and therefore needs not to be configured anywhere in the PLC. ② Your selected baudrate is automatically set in the PLC. No further action at the PLC is necessary from your side. ③ Your chosen baudrate for the selected device (e.g. the SBI port) must be additionally configured with the S40 Topology Configurator and downloaded with the PLC program. Make sure to call the function block “SUCOM_A” in your program to establish a successful connection. **Default:** 9600	19200
Phonenr.	This parameter specify two details: 1. It defines that you want to use a modem connection over the selected „SerialDevice“. 2. If defines the phone number of the remote modem which in turn is connected to the PLC.	049 228 602 4711

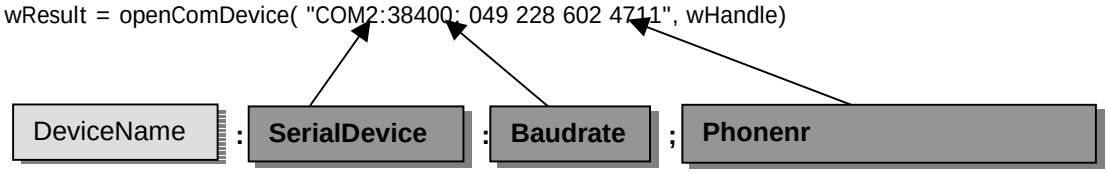
	Note: In this case the field „DeviceName“ has no meaning. **Default:** N/A.	
TCP/IP address	This parameter specify two details: 1. It defines that you want to use a TCP/IP connection. 2. If defines the TCP/IP address of the remote Ethernet/Serial gateway which in turn is connected to the PLC. *Note*: In this case the field „SerialDevice“ specify the serial device in the gateway and not in your PC. **Default:** N/A.	192.168.1.75
Hostname	This parameter is a symbolic representation for an TCP/IP address, which make it easier (for humans) to remember addresses of devices. The mapping (host) name to TCP/IP address is done in a file called „HOSTS“ which is located in the Windows directory. The file can be change with a standard editor like Wordpad. The parameter specify two things: 1. It defines that you want to use a TCP/IP connection. 2. If defines the TCP/IP address of the remote Ethernet/Serial gateway which in turn is connected to the PLC. *Note*: In this case the field „SerialDevice“ specify the serial device in the gateway and not in your PC. **Default:** N/A.	PistonValve

As you can see, it is possible to select a direct or a remote connection between the PC and the PLC. If you choose a remote connection, you have the choice between a Modem or a TCP/IP connection type. You can have up to 255 connections simultaneously. Please keep to the following rules to select and specify your needed connection type (the light grey fields are not necessary for the corresponding connection type)[20].

Direct connection

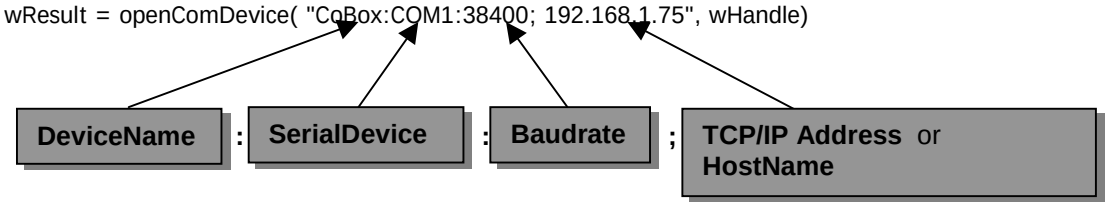


Remote Modem Connection



NOTE: The PLC serial port must be set manually to the 38400 baudrate, used in this example.

Remote TCP/IP Connection



NOTE: Both, the PLC port and the CoBox gateway will be set automatically to the given baudrate (38400 in this example). If no baudrate is specified, 9600 baud is assumed for the CoBox and the PLC port and the automatic baudrate setting sequence will be not activated. Make sure that the PLC port is initially set to a baudrate which is supported by the gateway, otherwise the automatic setting sequence doesn't work. For example, the CoBox support up to 38400 Baud, but if the PLC is set to 57600 no connection could be established. In this case the PLC serial port baudrate must be set manually.

For more details concerning the data exchange functions (*writeDataToPLC, readDataFromPLC, writeDataToPLCex, readDataFromPLCex, openComDevice*), please refer to the appropriate chapters in this documentation. After the functions returns, the connection with the PLC (direct or remote) is established or in the case of an error you get an error code. The next table show the different error code ranges depending on the selected connection type.

Error Codes	***Description***
0	Connection established and data exchange function returns successfully.
1 ... 99	General PS40_LINK error, e.g. a parameter fault, wrong COM device ...
300 ... 399	Error codes corresponding to Modem errors, e.g. the remote modem is busy.
400 ... 499	Error codes related to TCP/IP errors, e.g. the TCP/IP network is down.

[20] Please note, that the colon „:“ and the semicolon „;“ are necessary to separate the different fields. Do not use additional tabs or space characters in this context!

Technical data

The following tables describe some performance parameters divided into data reading, data writing with PS40_LINK version 2.x and 3.x at 9600 and 57600 Baud.[21] Please note as a general rule: For fastest update rates use small frame sizes as possible (at least smaller than 64 bytes) and for best data throughput use large frame sizes as possible (1024 Byte frames are a good compromise between a handy size and a good throughput).

Reading data

This table shows how many calls per second with ***readDataFromPLCex*** with a given frame size you can expect. Frame size means the value for ***xwNoOfElements*** with data type set to "Byte". For example you can see, it is possible to update the PC eight times per second with new PLC data if you transfer 64 Bytes within each call to ***readDataFromPLCex***, which in turn results in a total of approximately 512 bytes per second (with V3.0 and 9600 Baud).

PS40_LINK version *Frame size in bytes*	**PS40_LINK V2.x**	**PS40_LINK V3.0 9600 Baud**	**PS40_LINK V3.0 57600 Baud**
1	16	16	39
64	8	8	25
128	4	5	15
256	2	3	8

The next table shows you the performance ratio in Bytes/sec for bulk data transfers. For example you can see, it is possible to transfer up to 673 bytes per second if you set the frame size to 1024 bytes. The values in parenthesis give you the time needed to transfer 64KB bulk data.

PS40_LINK version *Frame size in bytes*	**PS40_LINK V2.x**	**PS40_LINK V3.0 9600 Baud**	**PS40_LINK V3.0 57600 Baud**
64	479 (137 sec)	479 (137 sec)	1604 (41 sec)
1024	479 (137 sec)	673 (97 sec)	2134 (31 sec)
16384	479 (137 sec)	691 (95 sec)	2358 (28 sec)

Please note that the actual PS40_LINK version 3.0 give you a great performance boost for reading data if you use a frame size greater than 64 bytes at 9600 Baud: The actual version is more than 40 % faster than its predecessor in this cases and about 490% faster if you use 57600 Baud.

Writing data

This table shows how many calls per second with ***writeDataToPLCex*** with a given frame size you can expect. Frame size means the value for ***xwNoOfElements*** with data type set to "Byte". For example you can see, it is possible to update the PLC seven times per second with new data if you transfer 64 Bytes within each call to ***writeDataToPLCex***.

PS40_LINK version *Frame size in bytes*	**PS40_LINK V2.x**	**PS40_LINK V3.0 9600 Baud**	**PS40_LINK V3.0 57600 Baud**
1	14	14	22
64	7	7	20
128	3	4	11
256	2	2	6

[21] The values are determined with a 400MHz Pentium II Windows 98 computer connected to a PS4-151-MM1 Moeller PLC with a 10ms cycle time (PS4-341-MM1 for 57600 Baud). Please note: The values may vary with the PLC cycle time.

Data sheet

Max. connection count	254
Max. direct serial connections	9 (COM1 ... COM9)
Baudrate supported	2400, 4800, 9600, 19200, 38400, 57600 (depends on PLC and connection type)
Automatic baudrate setting	PS4-341, PS416: for direct and TCP/IP connections
Max. bytes per functions call	16KB (depends on size of marker field)
Max. throughput (approximate)	2 KB/sec (depends on PC, PLC and connection type)
Max. calls per second	39 calls/sec (depends on PC, PLC and connection type)
Multi-threaded application support	Yes (Simultaneous open connections)
Moeller PLCs supported	PS4-141/151/201/271, PS4-341, PS416-200/300/400, PS4-101/111, PS3, PS306, PS316
Windows version supported	MS-Windows 95, 98, NT 4.0

Error codes

These are the result and error codes for the PS40_LINK DLL functions:

Code #	Description
0	**SUCOM_OK** The function call was complete and successful
1	**SUCOM_COMM_ERROR** During the reception of a PLC data package errors like parity, overrun, etc. have occurred. Please check the settings of the PLC interface (S40) and the cable.
2	**SUCOM_COM_PORT_NOT_FOUND** The interface specified in the *xpComDevice* parameter does not exist or is faulty.
3	**SUCOM_ERROR_OPEN_COM_PORT** Internal PC interface error. The specified interface has been found, but cannot be initialized with the necessary parameters.
4	**SUCOM_COM_PORT_ALREADY_OPEN** The interface specified with the parameter *xpComDevice* is being used by another Windows application.
5	**SUCOM_PARAMETER_ERROR** At least one of the specified parameters is not valid for the specified PLC type.
6	**SUCOM_ERROR_ON_WRITING** A general write error has occurred. Possible reasons are - The PLC is turned off - The PLC is not connected - The cable is faulty - The interface is faulty - Handle (xwPortID) is faulty
7	**SUCOM_ERROR_ON_READING** A general PLC read error has occurred. Possible reasons are - the PLC is turned off - the PLC is not connected - the cable is faulty - the interface is faulty
8	**SUCOM_WRONG_ANSWER** Error reading PLC. Possibly the connected device is not a Moeller PLC.
9	**SUCOM_READING_TIMEOUT** Timeout reading the PLC. Possible reasons are: - bad or no PLC connection - wrong PLC interface parameters (PS416 CPU switch setting) - wrong or faulty cable - PC interface faulty (device conflicts)
10	**SUCOM_WRITING_TIMEOUT** A timeout during a data send operation has occurred. Possible reasons are: - faulty PC COM interface - COM port device conflicts
11	**SUCOM_ERROR_INIT_TIMEOUT_VALUES** Internal error during initialization process for the PC COM port. The specified COM port has been found but cannot be initialized with the necessary parameters.
12	**SUCOM_TOO_MANY_MARKER** The PC is requesting marker data outside the PLC marker range
13	**SUCOM_NO_EXT_DLL_FOUND** The necessary PS40_LINK extension couldn't be found.
14	**SUCOM_INVALID_DLL** A extension DLL is invalid (doesn't contain Function entrypoint). Wrong DLL with the same extension name.
15	**SUCOM_BREAK_BY_USER** An operation like the automatic baudrate setting over TCP/IP connections was canceled by the user through a callback function.