PC-S5-LINK

Data coupling

PC - SIMATIC S5

AG 90 - AG 155

Via PG socket (AS511)

Version 1.00

Requirements:

Operating system: MS-Windows 95, 98, 2000 or NT 4.0

Hardware: V.24 - TTY converter to RS232 of the PC to the PLC

PLC: Simatic S5, all PLCs beginning with AG 90

Delivery contents:

Files on floppy disk / CD:

Main directory	
PCS5EASY.HTM	file documentation
VERSION.HTM	file for error recovery
Directory 'C'	**files for C / C ++**
PCS5EASY.H	header file for C / C ++
PCS5EASY.DLL	driver DLL
PCS5EASY.LIB	lib file for linking with C ++
ES5DEMO.DSP	Project file for Visual C ++ V 6.00
ES5DEMO.C	example program in 'C ' of a console application
ES5DEMO.EXE	EXE file of the C-Demo

Directory 'Delphi'	**Files for Delphi**
PCS5EASY.PAS	Delphi header TPU in source code
PCS5EASY.DLL	driver DLL
ES5DEMO.exe	EXE file of Delphi demo
ES5DEMO.cfg ES5DEMO.dof ES5DEMO.dpr ES5DEMO.res PCS5EASY.dcu MAIN.DCU MAIN.DFM MAIN.PAS OEM.BMP	Delphi project files
Directory Visual Basic	**files for Visual Basic**
PCS5EASY.DLL	Driver DLL, **Attention: For Visual Basic and Excel, please copy this file into the Windows directory!**
PCS5EASY.BAS	header / module file for

	Visual Basic
ES5DEMO.XLS	Excel file with macro for demo

Directory Excel	**files for Excel**
PCS5EASY.DLL	Driver DLL, **Attention: For Visual Basic and Excel, please copy this file into the Windows directory!**
PCS5EASY.BAS	header / module file for Visual Basic
ES5DEMO.XLS	Excel file with macro for demo

Functionality:

PC-S5-LINK is a DLL for MS-Windows (95/98/2000 or NT 4.0), which enables the connection of a PC to the PG interface of the SIMATIC S5. The PC is connected directly to the PG interface of the PLC via the RS 232 by a V.24 / TTY converter. With simple functions, the user can quickly access the PLC data using C, C ++, Delphi, Visual Basic or Excel. No additional communication processor is required in the PLC for the coupling. Flags, inputs, outputs the PLC can be read and written now.

Two communication modes are supported. The point-to-point communication and the bus communication (PG-BUS)

The two operating modes (setting see function S5Init):

1. The PG mode protocol:

The PG mode protocol is the replication of the AS511 protocol with respect to the S5's memory manipulations by the connected programming device. In this mode, only one PLC can be connected per interface. The advantage is that no programming or parametrisation in the PLC is required in this operating mode.

2. The PG-BUS protocol

In this version, up to 30 slaves (S5 PLCs from no. 1 to 30) can be connected to the PC using an RS232 interface with the L1 bus topology. The PLCs are inserted into the bus

system at the programming device interface with the Bus Terminal BT-777. There must be no master in the bus system (communication processor or master PLC). Ideally take the **UNICOM adapter** in this case. This adapter is in a stand-alone housing with integrated V.24 to the PC and Bus Terminal to the L1-BUS. In addition, this adapter has its own voltage supply with 220V Euro connection. The installation guidelines that SIEMENS requires for their L1-BUS. Each connected slave CPU must be configured as described in the device manuals of the CPUs, PG-Bus devices. For this purpose, the SIMATIC must be PG bus ready. The parametrisation of the S5 must be carried out in the start-up OBs OB21 and OB22 or in DB1. The PG number is high byte of the SD57 in the PLC.

Example: The slave is a PG bus node with the number 4

```
L  BS  57
      L  KH  =00ff
      UW                ;  L1 number must be retained, PGNr gets deleted
      L KH  =0400
      OW                ; set PGNr
      T  BS  57
```

These two statements must be in OB21 and OB22. The PLC programmer does not have to worry about further data traffic. The PC has free access to all data (such as the programming device.)

Functional description in detail:

Please note: The functions are executed with the standard RS232 interface, which means that the function returns to the caller only after the task has been completed. For asynchronous operation, simply call these functions from a separate thread, which is responsible for communicating the system

The following functions are available:

Initialization functions:

Function	Description / purpose
ES5Open	Initializes the serial interface. At this point, no check is made as to whether the PLC is available.

	Caution: The connection setup is started automatically the first time the read or write functions are called.

Calling parameter:

No.	Memory width	Designation	Function
1	32-bit value unsigned	Com	Serial port number: 0 = COM1 1 = COM2 .. etc.
2	32-bit value unsigned	Station no.	Is used with PG-BUS and corresponds to the number of the PLC to be addressed. 0 means point-to-point coupling. **ATTENTION ! This parameter is not considered in the DEMO version and always set to 0 internally. Please request separate DEMO version if required.**

Function	Description / purpose
ES5Close	Closes the connection identified by Ref. If necessary, the serial interface is also closed here

Calling parameter:

No.	Memory width	Designation	Function
1	32-bit value unsigned	Ref	The reference of the connection that was generated with ES5Open. Used to identify the connection internally.

Functions for reading and writing

Function	Description / Purpose
ES5RdW	reading from the PLC (E, A, M, DB) word by word
ES5RdB	reading from PLC (E, A, M) byte orientated
ES5WrW	write into the PLC (E, A, M, DB) word by word
ES5WrB	write into the PLC (E, A, M) byte orientated

Calling parameters:

The read and write functions have the same input parameters:

No.	Memory width	Designation	Function
1	32-bit value unsigned	Ref	The reference of the connection that was generated with ES5Open. Used to identify the connection internally.
2	32-bit value unsigned	Typ	The selection of the memory area in the PLC (DB, input, output, flags), which is to be processed: 'D' = 68 dec. means data block 'E' = 69 dec. means inputs 'A' = 65 dec. means outputs 'M'= 77 dec. means flags
3	32-bit value unsigned	DBNo	Data block number, this is only used for type'D '. Otherwise, the value "0"
4	32-bit value unsigned	Cnt	Number of units (bytes or words) to read or write.
5	32-bit address	Buffer	The address on the source or target

			memory in the PC. For the word functions, this is a pointer to a field of 16-bit-wide words, in the case of the byte functions this is an address on a field with 8-bit-wide bytes.

Return values:

value	Error Description	Meaning
> = 0	everything OK	For ES5Open, this is the reference number for this connection and must be used as input parameter Ref for all other functions. For the other functions, this value is 0 if the action was successful. Otherwise the following errors should be checked.
2	For read and write accesses, this means that the desired data area does not exist. Data block is not available or too small.	Check the module length
-1	time-out	Timeout occurred, partner does not answer or no more.
-2	no more resources free.	There are no resources left, ES5Open can be called up to 32 times.
-3	the specified reference was not open	No ES5Open was executed with the specified reference.
-4	interface not available	The specified RS232 interface either does not exist in the system or is already being used by another application.
-5	general error	Unspecified error occurred
-6	wrong character received	Transmission error occurred
-10	desired data type not allowed or not	Check that the code for the data type is

	supported.	correct.
-99	the reference number is invalid	Have you called ES5Open?
4660	demo time has expired	get the full version

!!! Consider with word operations !!!

Example for flags. This also applies to inputs and outputs but **not for data blocks.**

The word addressing in the PLC occupies the following bytes, respectively.

Word address	***assigned bytes***
FW0	FB 0 and FB 1
FW1	FB 1 and FB 2
FW2	FB 2 and FB 3

You can see that the use of odd word addresses can result in a double assignment. Therefore, the word functions (ES5RdW and ES5WrW) only support access to even word addresses. This means that the start word number in the driver is always multiplied by 2. This method allows a simple image of the PLC memory in the PC. So a word step in the PC are 16 bits in the PC and 16 bits in the PLC

Example:

WORD Buf [64];

The call **ES5RdW (Ref, 'M', 0, 0, 5, Buf)** has the following effect:

PC	**PLC**
Buf [0]	DW0
Buf [1]	DW 2
Buf [2]	DW 4

So you have to halve the starting word number in order to access the PC correctly. This does not apply to data blocks !! -> Odd word addresses in the I, O and M area of the PLC can not be read or written by word.

Program examples:

a) Call from C or C ++:

```
unsigned char ByteBuffer [512];
unsigned short int WordBuffer [512];
// call the word function e.g. read DB 10, from DW0, 10 words
ES5RdW (Ref,'D', 10, 0, 10, WordBuffer);
// call the byte function e.g. read DB0 0, 10 bytes
ES5RdB (Ref, 'M', 0, 0, 10, ByteBuffer);
```

After successfully call:

PC	=	**PLC**
Word Buffer [0]	=	DB10.DBW0
Word Buffer [1]	=	DB10.DBW1
Word Buffer [2]	=	DW10.DBW2
Byte Buffer [0]	=	MB 0
Byte Buffer [1]	=	MB 1

b) Call from Delphi:

ByteBuffer array [0..511] of Byte;

WordBuffer array [0..511] of Word;

// call the word function e.g. read DB 10, from DW0, 10 words

ES5RdW (Ref, LongWord ('D '), 10, 0, 10, @WordBuffer [0]);

// call the byte function e.g. read MB 0, 10 bytes

ES5RdB (Ref, 'M', 0, 0, 10, @ByteBuffer [0]);

c) Call from Visual Basic:

Dim ByteBuffer (0 to 511) as Byte;

Dim WordBuffer (0..511) as Word;

// calling the word function e.g. read DB 10, from DW0, 10 words

ES5RdW (Ref, 68, 10, 0, 10, WordBuffer (0));

// calling the byte function e.g. read MB 0, 10 bytes

ES5RdB (Ref, 77, 0, 10, ByteBuffer (0));

After successful call:

PC	=	PLC
Word Buffer [0]	=	DB10.DBW0
Word Buffer [1]	=	DB10.DBW1
Word Buffer [2]	=	DW10.DBW2
Byte Buffer [0]	=	MB 0
Byte Buffer [1]	=	MB 1

C file header:

/* PCS5EASY.H

*/

```
long WINAPI
ES5Open (DWORD Com, DWORD PGNr);
long WINAPI
ES5Close (long Ref);
long WINAPI
ES5RdW (long Ref, DWORD Typ,
        DWORD DBNr, DWORD Ab, DWORD Anz, LPWORD Buffer);
long WINAPI
ES5RdB (long Ref, DWORD Typ,
        DWORD DBNr, DWORD Ab, DWORD Anz, LPBYTE Buffer);
long WINAPI
ES5WrW (long Ref, DWORD Typ,
        DWORD DBNr, DWORD Ab, DWORD Anz, LPWORD Buffer);
long WINAPI
ES5WrB (long Ref, DWORD Typ,
        DWORD DBNr, DWORD Ab, DWORD Anz, LPBYTE Buffer);
```

Visualbasic file header:

```
'

'PCS5EASY.BAS
'


Declare Function ES5Open& Lib "PCS5EASY.DLL" (ByVal Com&, ByVal PGNr&)
Declare Function ES5Close& Lib "PCS5EASY.DLL" (ByVal Ref&)
Declare Function ES5RdW& Lib "PCS5EASY.DLL" (ByVal Ref&, _
                ByVal Typ&, _
                ByVal DBNr&, _
                ByVal AbWort&, _
                ByVal WortAnz&, _
```

```
                Wert As Integer)
Declare Function ES5RdB& Lib "PCS5EASY.DLL" (ByVal Ref&, _
                ByVal Typ&, _
                ByVal DBNr&, _
                ByVal AbWort&, _
                ByVal WortAnz&, _
                Wert As Byte)
Declare Function ES5WrW& Lib "PCS5EASY.DLL" (ByVal Ref&, _
                ByVal Typ&, _
                ByVal DBNr&, _
                ByVal AbWort&, _
                ByVal WortAnz&, _
                Wert As Integer)
Declare Function ES5WrB& Lib "PCS5EASY.DLL" (ByVal Ref&, _
                ByVal Typ&, _
                ByVal DBNr&, _
                ByVal AbWort&, _
                ByVal WortAnz&, _
                Wert As Byte)
```

Delphi file header:

```
(*
  Modul : PCS5EASY.PAS
*)
unit PCS5EASY;
interface
TYPE PWORD =  ^WORD;
TYPE PBYTE =  ^BYTE;
```

```
FUNCTION
ES5Open (Com : LongWord; PGNr : LongWord) : LongInt; stdcall; external
'PCS5EASY.DLL';


FUNCTION
ES5Close (Ref : LongInt) : LongInt; stdcall; external 'PCS5EASY.DLL';
FUNCTION
ES5RdW (Ref : LongInt;
        Typ : Longword;
        DBNr : Longword;
        AbWort : Longword;
        WortAnz : Longword;
        Buffer : PWORD)  : LongInt; stdcall; external 'PCS5EASY.DLL';
FUNCTION
ES5RdrB (Ref : LongInt;
        Typ : Longword;
        DBNr : Longword;
        Ab : Longword;
        Anz : Longword;
        Buffer: PBYTE) : LongInt; stdcall; external 'PCS5EASY.DLL';
FUNCTION
ES5WrW (Ref : LongInt;
        Typ : Longword;
        DBNr : Longword;
        AbWort : Longword;
        WortAnz : Longword;
        Buffer : PWORD)  : LongInt; stdcall; external 'PCS5EASY.DLL';
FUNCTION
ES5WrB (Ref : LongInt;
        Typ : Longword;
        DBNr : Longword;
        Ab  : Longword;
        Anz : Longword;
```

```
        Buffer: PBYTE) : LongInt; stdcall; external 'PCS5EASY.DLL';
implementation
begin
end.
```