PC-S5-LINK
Datenkopplung

PC – SIMATIC S5

AG 90 - AG 155

über PG-Buchse (AS511)

Version 1.00

Vorausetzungen:

Betriebssystem: MS-Windows 95 ,98, 2000 oder NT 4.0

Hardware: V.24 – TTY -Konverter an RS232 des PC zur SPS

SPS: Simatic S5, alle Steuerungen ab AG 90

Lieferumfang:

Auf der Lieferdiskette / CD finden Sie folgende Dateien

Hauptverzeichnis	
PCS5EASY.HTM	diese Datei – Dokumentation
VERSION.HTM	Datei über Fehlerbeseitigung
Verzeichnis ´C´	**Dateien für C / C++**
PCS5EASY.H	Header-Datei für C / C++
PCS5EASY.DLL	Treiber DLL
PCS5EASY.LIB	Lib-Datei zum Linken mit C++
ES5DEMO.DSP	Projektdatei für Visual C++ V 6.00
ES5DEMO.C	Beispielprogramm in ´C´ einer Konsolenapplikation
ES5DEMO.EXE	EXE-Datei der C-Demo
Verzeichnis ´Delphi´	**Dateien für Delphi**
PCS5EASY.PAS	Delphi-Header TPU im Quellcode
PCS5EASY.DLL	Treiber DLL
ES5DEMO.exe	EXE-Datei der Delphidemo
ES5DEMO.cfg ES5DEMO.dof ES5DEMO.dpr	Delphi - Projektdateien

ES5DEMO.res PCS5EASY.dcu MAIN.DCU MAIN.DFM MAIN.PAS OEM.BMP	
Verzeichnis VisualBasic	**Dateien für Visual Basic**
PCS5EASY.DLL	Treiber DLL, **Achtung: Für Visual Basic und Excel diese Datei ins Windowsverzeichnis kopieren !**
PCS5EASY.BAS	Header / Moduldatei für Visual Basic
ES5DEMO.XLS	Excel-Datei mit Makro für Demo
Verzeichnis Excel	**Dateien für Excel**
PCS5EASY.DLL	Treiber DLL, **Achtung: Für Visual Basic und Excel diese Datei ins Windowsverzeichnis kopieren !**
PCS5EASY.BAS	Header/Moduldatei für Visualbasic
ES5DEMO.XLS	Excel-Datei mit Makro für Demo

Funktionsweise:

PC-S5-LINK ist eine DLL für MS-Windows (95/98/200 oder NT 4.0), welche die Anbindung eines PC an PG-Schnittstelle der SIMATIC S5 ermöglicht. Der PC wird über die RS 232 mittels eins V.24/TTY-Konverters direkt mit der PG-Schnittstelle der SPS verbunden. Mit einfachen Funktionen kann der Anwender schnell mit C, C++, Delphi, Visual Basic oder auch Excel auf die Daten der SPS zugreifen. Zur Kopplung wird kein zusätzlicher Kommunikationsprozessor in der SPS benötigt. Sofort kann auf Merker. Eingänge, Ausgänge und auch Datenbausteine der SPS lesend oder auch schreiben zugegriffen werden.
Es werden zwei Kommunikationsmodi unterstütz. Die Punkt zu Punktkommunikation und die Bus-Kommunikation (PG-BUS)

Die beiden Betriebsmodi (Einstellung siehe Funktion S5Init):

1. Das PG-Mode-Protokoll:

Das PG-Mode-Protokoll ist die Nachbildung des AS511-Protokolls in Bezug auf Speichermanipulationen der S5 durch das angeschlossene Programmiergerät. In diesem Modus darf pro Schnittstelle nur ein AG (Automatisierungsgerät) angeschlossen werden. Der Vorteil ist, daß in dieser Betriebsart keinerlei Programmierarbeit oder Parametrierung in der SPS nötig ist.

2. Das PG-BUS-Protokoll

Bei dieser Variante können an den PC an einer RS232- Schnittstelle bis zu 30 Slaves (S5-AGs von Nr 1 bis 30) mit der L1-BUS-Topologie angeschlossen werden. Die AG's werden an der Programmiergeräteschnittstelle mit die Busklemme BT-777 in das Bussystem eingefügt. Im Bussystem darf sich kein Master befinden (Kommunikationsprozessor oder Master-AG). Idealerweise nehmen Sie hier den **UNICOM-Adapter**. Dieser Adapter ist in einem eigenständigem Gehäuse mit integrierter V.24 zum PC und Busklemme zum L1-BUS. Zusätzlich verfügt dieser Adapter über eine eigene Spannungsversorgung gespeist mit 220V Euroanschluß. Es sind die Aufbaurichtlinien, wie sie SIEMENS für ihren L1-BUS fordert zu beachten. Jede angeschlossene Slave-CPU muß wie in den Gerätehandbüchern der CPU's beschrieben , PG-Bus-Teilnehmer konfiguriert werden. Dazu muß die anzuschließende SIMATIC PG-Bus-fähig sein. Die Parametrierung der S5 muß in den Anlauf-OBs OB21 und OB22 oder im DB1 geschehen. Die PG.Nummer steht Highbyte des SD57 im AG.

Beispiel: Der Slave ist PG-Busteilnehmer mit der Nummer 4

```
L   BS  57
L  KH  =00ff
UW                ; L1-Nummer muß erhalten bleiben, PGNr wird gelöscht
L KH  =0400
OW                ; PGNr setzten
T  BS  57
```

Das war's. Diese beiden Anweisungen müssen im OB21 und OB 22 stehen. Um den weiteren Datenverkehr hat sich der SPS-Programmierer nicht zu kümmern. Der PC hat freien Zugriff auf alle Daten (wie das Programmiergerät.)

Funktionsbeschreibung im Detail:

Bitte beachten Sie: Die Funktionen werden mit der Standard RS232 -Schnittstelle ausgeführt, was zur Folge hat, dass die Funktion erst nach Erfüllung der Aufgabe zum Aufrufer zurückkehrt. Zum Asynchronen Betrieb rufen Sie diese Funktionen einfach von einem separaten Thread aus auf, welcher für die Kommunikation des System zuständig ist

Folgende Funktionen stehen zur Verfügung:

Funktionen zur Initialisierung:

Funktion	*Beschreibung / Zweck*
ES5Open	Dient zur Initialisierung der seriellen Schnittstelle. Es findet an dieser Stelle keine Prüfung statt, ob die SPS vorhanden ist Achtung: Beim ersten Aufruf der Lese- oder Schreibfunktionen wird der –Verbindungsaufbau automatisch gestartet.

Aufrufparameter:

Nr.	***Speicherbreite***	***Bezeichnung***	***Funktion***
1	32-Bit-Wert ohne Vorzei.	Com	Nummer der seriellen Schnittstelle: 0 = COM1 1 = COM2 .. usw
2	32-Bit-Wert ohne Vorzei.	StationsNr	Wir bei PG-BUS verwendet und entspricht der Nummer der SPS, die angesprochen werden soll. 0 heiß Punkt zu Punkt-Kopplung. **ACHTUNG ! Dieser Parameter wird in der DEMO-Version nicht beachtet und intern immer mit 0 vorbesetzt. Bitte bei Bedarf gesonderte DEMO-Version anfordern.**

Funktion	*Beschreibung / Zweck*
ES5Close	Schließt die durch Ref gekennzeichnete Verbindung, Gegebenenfalls wird hier auch die serielle Schnittstelle geschlossen

Aufrufparameter:

Nr.	***Speicherbreite***	***Bezeichnung***	***Funktion***
1	32-Bit Wert ohne Vorzei.	Ref	Die Referenz der Verbindung, welche mit ES5Open generiert wurde. Dient zur internen Identifikation der Verbindung

Funktionen zum Lesen und Schreiben

Funktion	*Beschreibung / Zweck*
ES5RdW	wortweise lesen aus der SPS (E,A,M, DB)
ES5RdB	byteweise lesen aus der SPS (E,A,M)
ES5WrW	wortweise schreiben in die SPS (E,A,M, DB)
ES5WrB	byteweise schreiben in die SPS ((E,A,M)

Aufrufparameter:

Die Lese- und Schreibfunktionen besitzen die selben Eingangparameter:

Nr.	*Speicherbreite*	*Bezeichnung*	*Funktion*
1	32-Bit Wert ohne Vorzei.	Ref	Die Referenz der Verbindung, welche mit ES5Open generiert wurde. Dient zur internen Identifikation der Verbindung
2	32-Bit Wert ohne Vorzei.	Typ	Die Auswahl des Speicherbereichs in der SPS (DB, Eingang, Ausgang, Merker), welcher bearbeitet werden soll: ´D´ = 68 dez. steht für Datenbaustein ´E´ = 69, dez. steht für Eingänge ´A´ = 65 dez. steht für Ausgänge ´M´ = 77 dez. steht für Merker
3	32-Bit Wert ohne Vorzei.	DBNr	Datenbausteinnummer, diese wird nur beim Typ ´D´ verwendet. Ansonsten steht dort der Wert „0“
4	32-Bit Wert ohne Vorzei.	Anz	Anzahl der Einheiten (Byte oder Worte), die gelesen bzw. geschrieben werden sollen.
5	32-Bit Adresse	Buffer	De Adresse auf den Quell- bzw. Zielspeicher im PC. Bei den Wortfunktionen ist dies ein Zeiger auf ein Feld von 16-Bit breiten Worten, bei den Bytefunktionen ist das eine Adresse auf ein Feld mit 8-Bit breiten Bytes.

Rückgabewerte:

Wert	**Fehlerbeschreibung**	**Bedeutung**
>= 0	alles OK	Bei ES5Open ist dies die Referenznummer für diese Verbindung und muss bei allen anderen Funktionen als Eingangsparameter Ref verwendet werden. Für die übrigen Funktionen ist dieser Wert 0, wenn die Aktion erfolgreich war. Ansonsten sind folgende Fehler zu prüfen.
2	Bei Lese- und Schreibzugriffen, bedeutet heißt dies, der gewünschte Datenbereich existiert nicht. Z.B Datenbaustein nicht vorhanden oder zu klein.	Bausteinlänge prüfen
-1	Timeout	Timeout aufgetreten, Partner Antwortet nicht oder nicht mehr.
-2	Keine Resourcen mehr frei.	Es stehen keine Resourcen mehr zur verfügung, ES5Open kann maximal 32 mal aufgerufen werden.
-3	Die angegebene Referenz war nicht geöffnet	Mit der angebenen Referenz wurde kein ES5Open ausgeführt-
-4	Schnittstelle nicht vorhanden	Die angegebene RS232-Schnittstelle ist im System entweder nicht vorhanden oder wird bereits von einer anderen Anwendung verwendet.

-5	Allgemeiner Fehler	nicht spezifizierter Fehler aufgetreten
-6	falsches Zeichen erhalten	Übertragungsfehler aufgetreten
-10	Gewünschter Datentyp nicht erlaubt oder wird nicht unterstützt.	Prüfen, ob der Code für Datentyp in Ordnung ist.
-99	Die Referenznummer ist ungültig	Haben Sie ES5Open aufgerufen ?
4660	Demozeit ist abgelaufen	Vollversion erwerben

!!! Beachte bei Wortoperationen !!

Beispiel für Merker. Dies gilt auch für Eingänge Ausgänge jedoch **nicht für Datenbausteine**

Die Wortadressierung in der SPS belegt jeweils folgende Bytes.

Worteadresse	***zugeordnete Bytes***
MW0	MB 0 und MB 1
MW1	MB 1 und MB 2
MW2	MB 2 und MB 3

Sie sehen, dass es bei Verwendung von ungeraden Wortadressen zu einer Doppelbelegung kommen kann. Deshalb unterstützen die Wortfunktionen (ES5RdW und ES5WrW) nur den Zugriff auf gerade Wortadressen. Dies bedeutet, dass die Start-Wort-Nr im Treiber immer mit 2 multipliziert wird. Diese Methode erlaubt zu dem ein einfaches Abbild des SPS-Speichers in den PC. Also ein Wortschritt im PC sind 16 Bit im PC und 16 Bit in der SPS

Beispiel:

WORD Buf[64];

Der Aufruf ***ES5RdW (Ref,'M' ,0, 0, 5, Buf)*** hat folgende Wirkung:

PC	***SPS***
Buf[0]	MW0
Buf[1]	MW 2
Buf[2]	MW 4

Sie müssen also die Start-Wortnummer halbieren, um im PC richtig zugreifen zu können. Dies gilt nicht für Datenbausteine !! --> Ungerade Wortadressen im E,A und M-Bereich der SPS können nicht wortweise gelesen oder geschrieben werden.

Programmbeispiele:

a) Aufruf von C oder C++ aus:

unsigned char ByteBuffer[512];

unsigned short int WordBuffer[512];

// Aufruf der Wortfunktion z.B. Lese DB 10, ab DW0, 10 Worte
ES5RdW (Ref, ´D´, 10, 0, 10, WordBuffer);
// Aufruf der Bytefunktion z.B. Lese MB 0 , 10 Bytes
ES5RdB (Ref, ´M´, 0, 0, 10, ByteBuffer);

Nach erfolgreichem Aufruf gilt:

PC	=	SPS
WordBuffer[0]	=	DB10.DBW0
WordBuffer[1]	=	DB10.DBW1
WordBuffer[2]	=	DW10.DBW2
ByteBuffer[0]	=	MB 0
ByteBuffer[1]	=	MB 1

b) Aufruf von Delphi aus:

ByteBuffer array [0..511] of Byte;

WordBuffer array [0..511] of Word;

// Aufruf der Wortfunktion z.B. Lese DB 10, ab DW0, 10 Worte
ES5RdW (Ref, LongWord (´D´), 10, 0, 10, @WordBuffer[0]);
// Aufruf der Bytefunktion z.B. Lese MB 0 , 10 Bytes
ES5RdB (Ref, ´M´, 0, 0, 10, @ByteBuffer[0]);

c) Aufruf von Visual Basic aus:

Dim ByteBuffer (0 to 511) as Byte;

Dim WordBuffer (0..511) as Word;

// Aufruf der Wortfunktion z.B. Lese DB 10, ab DW0, 10 Worte
ES5RdW (Ref, 68, 10, 0, 10, WordBuffer(0))
// Aufruf der Bytefunktion z.B. Lese MB 0 , 10 Bytes
ES5RdB (Ref, 77, 0, 0, 10, ByteBuffer(0));

Nach erfolgreichem Aufruf gilt:

PC	=	SPS
WordBuffer[0]	=	DB10.DBW0
WordBuffer[1]	=	DB10.DBW1
WordBuffer[2]	=	DW10.DBW2
ByteBuffer[0]	=	MB 0
ByteBuffer[1]	=	MB 1

C-Dateiheader:

```
/* 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-Dateiheader:

```
'
'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-Dateiheader:

```
(*
   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.
```