IP-S7-LINK

Per TCP/IP an die SIMATIC S7 über HMI/PG Protokoll (RFC1006)
Dokumentation zur Version 1.76

Unterstützte Systeme

- S7-1500
- S7-1200
- S7-300/400/200
- !Logo
- WIN AC RTX
- VIPA S7
- Jede S7-kompatible SPS

Betriebssysteme

- MS-Windows Desktop XP/7/8/10 32/64-Bit
- alle MS-Windows Server 32/64-Bit
- Windows CE
- alle Linux 32/64-Bit
- Embedded Linux 32/64Bit x86/ARM

Programmiersprachen

- C/C++
- VB
- Delphi
- Excel
- Access
- PHP und andere
- C#/VB.Net - dafür empfehlen wir unser IPS7LnkNet.Advanced Framework, in purem C# entwickelt

Das Tool ist in purem C/C++ Code entwickelt. Es lässt sich so auf jede beliebige Plattform / Architektur mit geringstem Aufwand portieren. Release Notes

Installation

Windows

- C/C++, Delphi VB etc.:
  - die DLL ins Verzeichnis des Programms oder ins Systemverzeichnis kopieren
- PHP
  - Die Extension ips7lnk_php.dll in das Extensionverzeichnis der PHP-Installation kopieren
  - Über PHP.ini oder im Programm selbst dafür sorgen, dass das Modul geladen wird

php.ini: extension = ips7lnk_php.dll

```
    im Programm: ld ('ips7lnk_php.dll');
```

Linux

- C/C++ und andere
  - Die o.Datei zu Ihrem Programm linken oder ein .so erzeugen
- PHP unter Linux
  - Die Extension ips7lnk_php.so in das Extensionverzeichnis der PHP-Installation kopieren
  - Über PHP.ini oder im Programm selbst dafür sorgen, dass das Modul geladen wird

php.ini: extension = ips7lnk_php.so

```
    im Programm: ld ( ips7lnk_php.so');
```

SPS - Einstellungen

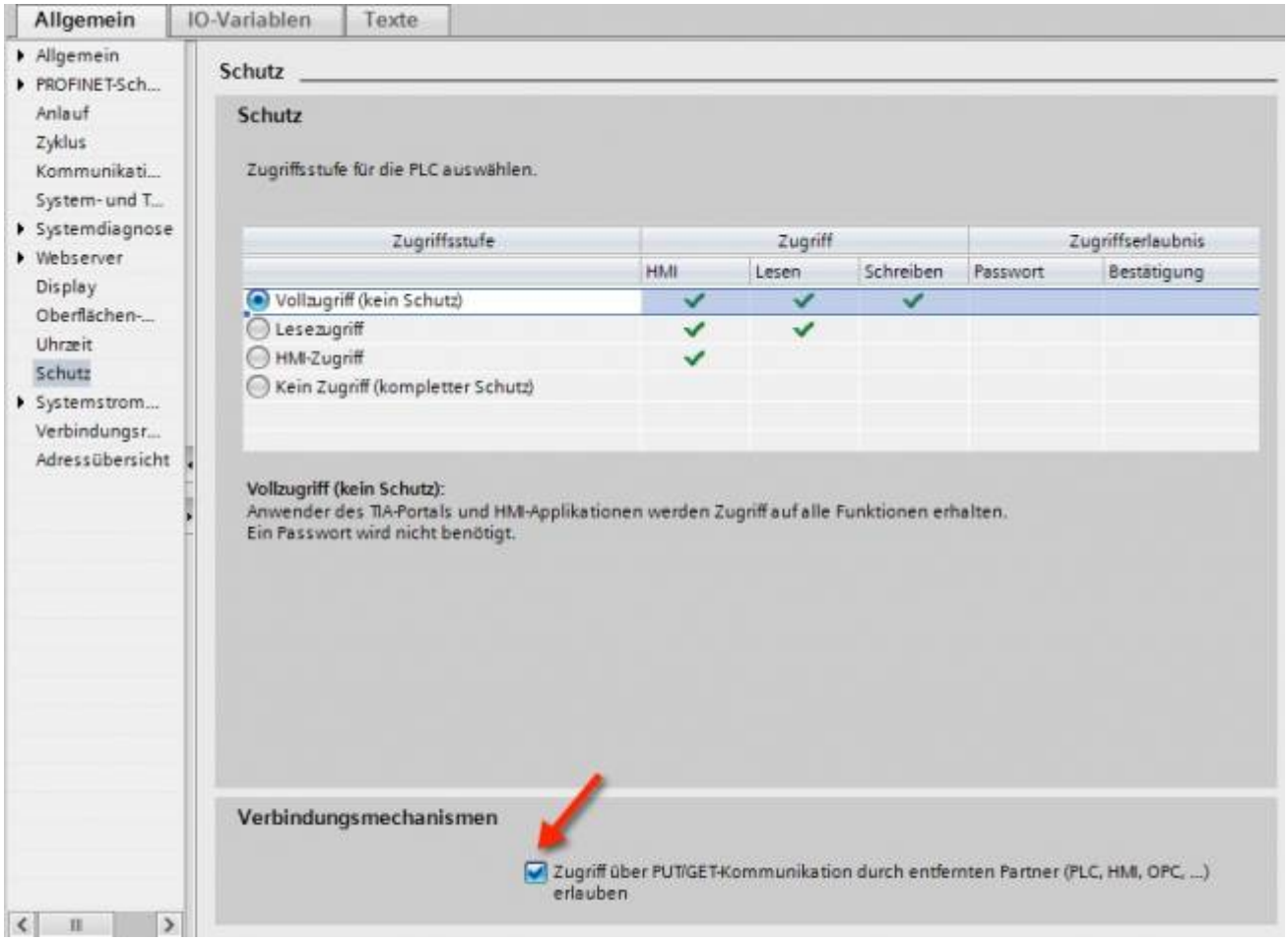
S7-300/400

Für den Zugriff auf die S7-300/400 sind keine besonderen Einstellungen zu tätigen. Es sollte nur nur IP-Adresse, Rack und Slot bekannt sein. In der Regel Rack:0, Slot:2. Slot ist die Nummer des Slots, in dem die CPU steckt.
Bei SPS direkt eingebauter Ethernet-Schnittstelle ist das Slot 1.

S7-1200/1500

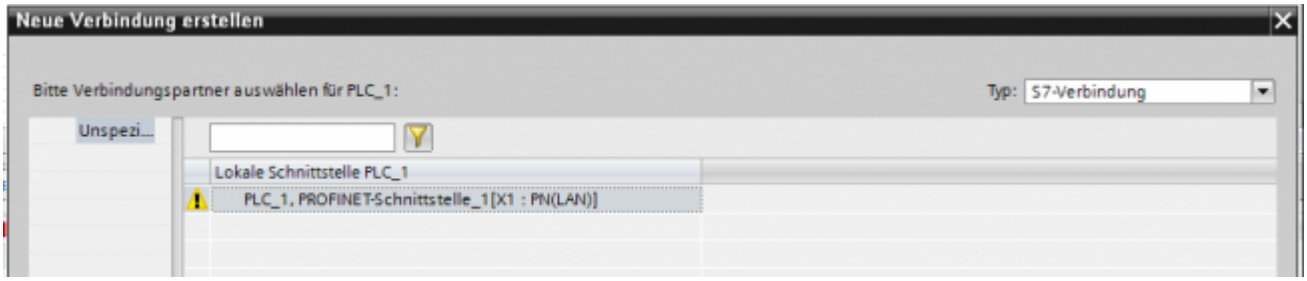
- Diese Einstellungen gelten ab Firmwarestand **4.0** bei der S7-1200
- Im Treiber bzw. in der Software Rack=0 und Slot=1 setzen
- Datenbausteinattribute: den optimierten Baustein Zugriff deaktivieren

- Kommunikationseinstellung: PUT/GET-Zugriff aktivieren. Wie das geht, sehen Sie hier (Snapshot aus TIA-Portal).

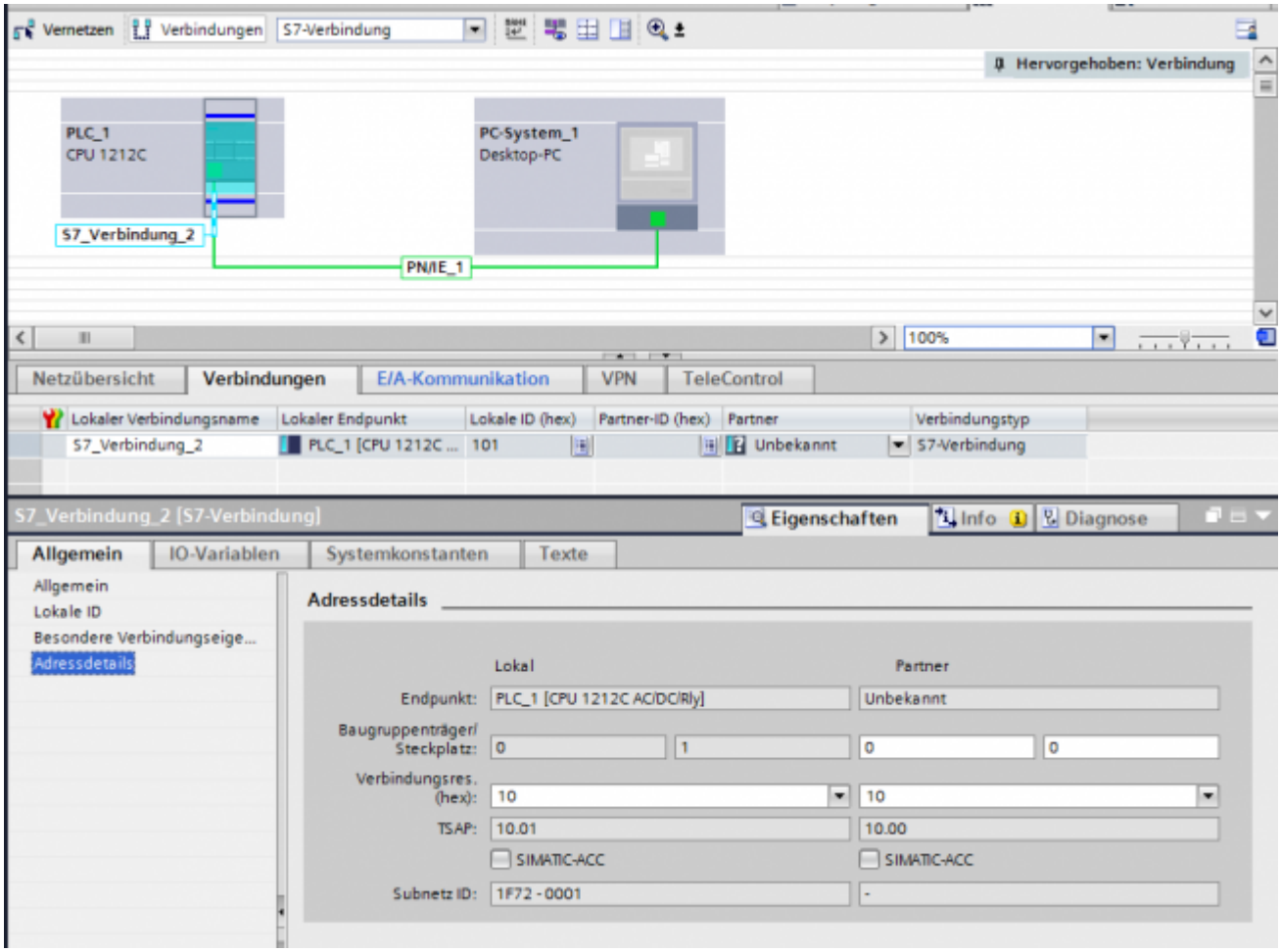


S7-1200 bis Version 4.xx

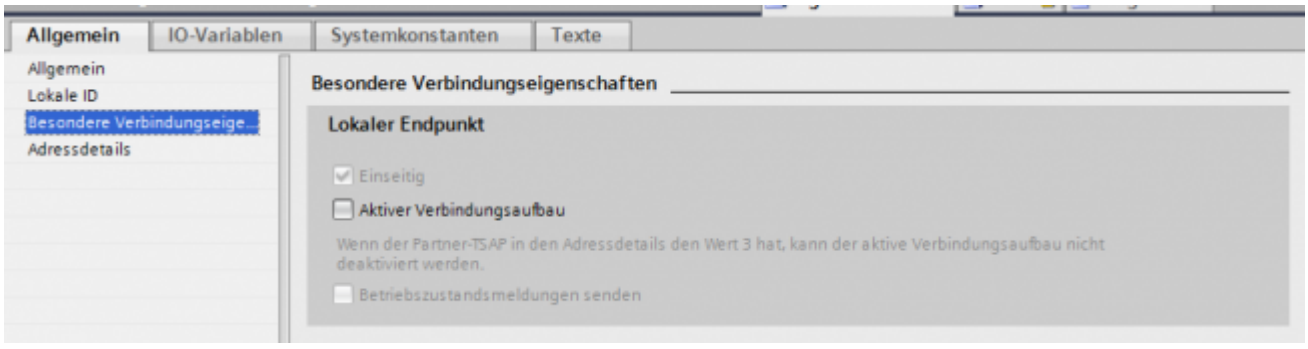
- Wechseln Sie in die `Netzsicht`
- Fügen Sie z.B. ein `PC-System mit einem Port` hinzu
    - Tragen Sie die IP-Adresse Ihres Systems ein
- Mit Rechtsklick auf die S7-1200 legen Sie eine neue S7-Verbindung an



- Nach dem Klick auf `Hinzufügen` und `Schließen` sehen Sie eine undefinierte Verbindung
- Stellen Sie unter `Allgemein` die Partner-IP-Adresse ein
- Klicken Sie auf diese Verbindung (`Eigenschaften`)
- Unter `Adressdetails` sehen Sie folgende Eingabemaske



- Wählen Sie die gewünschten TSAPs (`Verbindungsres (hex)`) für den Lokal und Partner aus
  - **Achtung**: Verwenden Sie nicht den Partner-TSAP 03, da hier der passive Modus nicht einstellbar ist
- Unter `Besondere Verbindungseigenschaften` deaktivieren Sie `Aktiver Verbindugsaufbau`

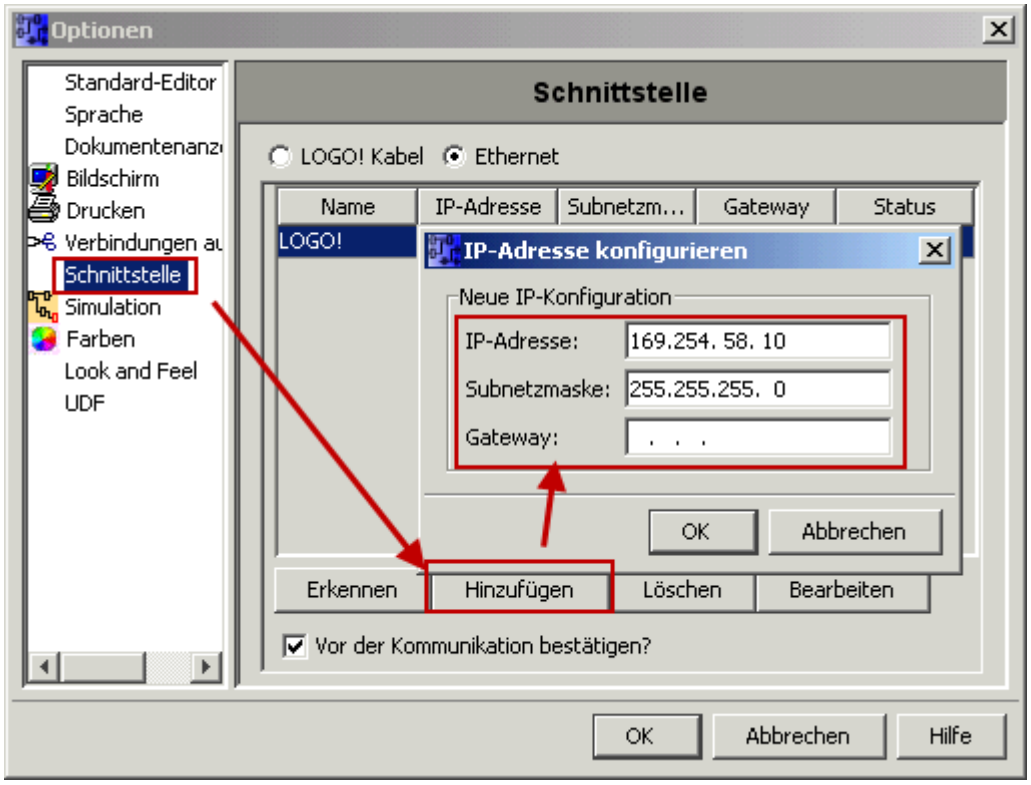


- Das Projekt übersetzen und bei keinem Fehler übertragen

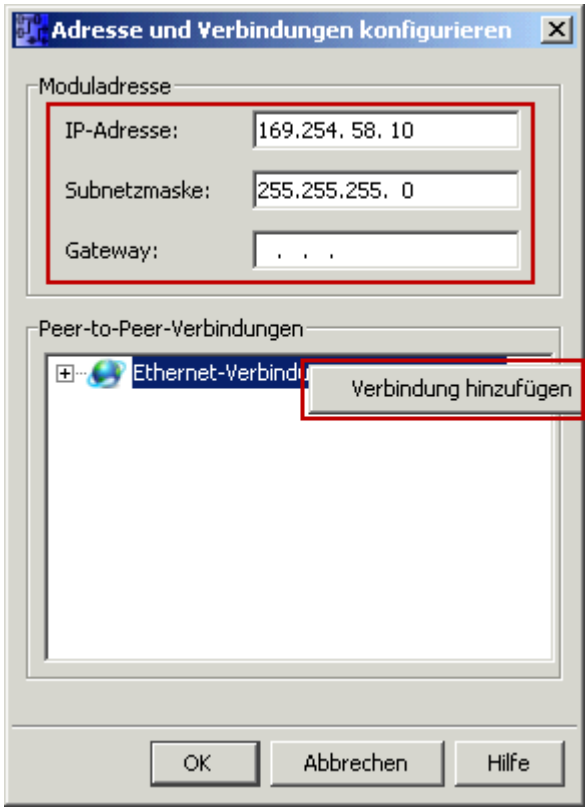
Wenn Sie nun die eingestellten LocalTSAP und RemoteTSAP verwenden, sollten Sie auf die Daten zugreifen können

LOGO!

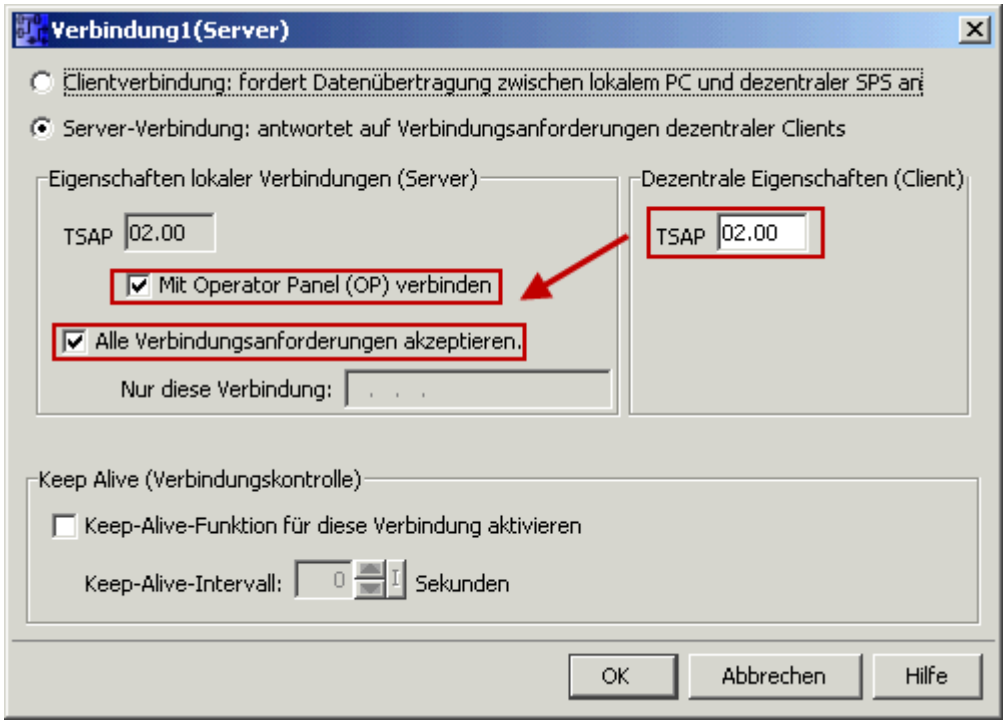
1. Stellen Sie über die Logo!Soft Comfort die IP-Adresse der SPS ein:



2. Konfigurieren Sie die Logo!-SPS so, dass Verbindungen von einem HMI-Gerät angenommen werden. Gehen Sie dazu auch „Extras→Ethernetverbindungen“ und fügen dann eine neue Verbindung hinzu.

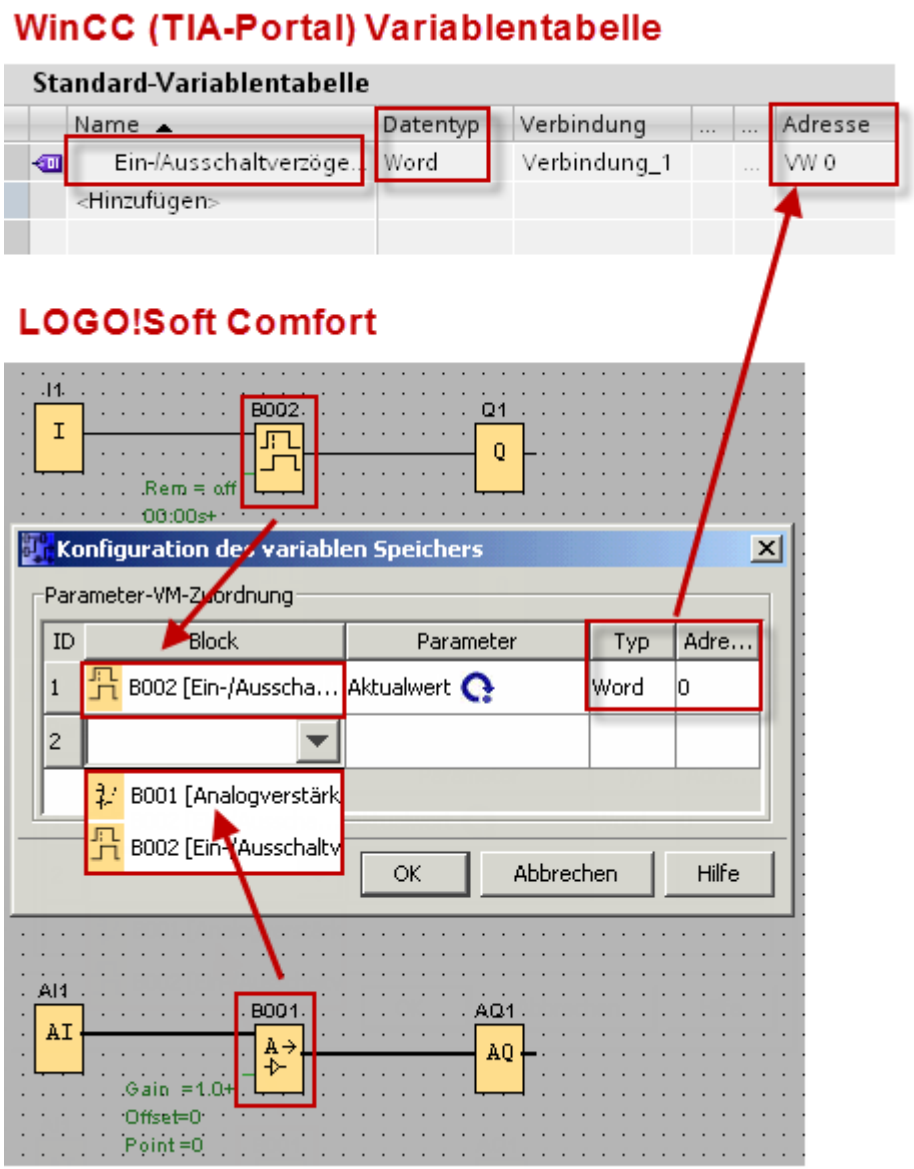


3. Machen Sie einen Doppelklick auf die neu angelegte Verbindung, um in die Eigenschaften zu gelangen.



Wählen Sie: - Server-Verbindung - lokaler TSAP: 02.00 - dezentraler TSAP 02.00 - alle Verbindungen akzeptieren.

Sie können auf DB1, Eingänge, Ausgänge, Merker, Zähler und Timer mit IP-S7-LINK zugreifen. Legen Sie nun über „Extras→Parameter-VM-Zuordnung“ die Variablen fest, die in den DB1 übertragen werden sollen.



Bilder aus Siemens Support Portal

Funktionsweise

IP-S7-LINK realisiert die Anbindung eines PC an Industrial Ethernet der SIMATIC S7.
Die Library ist für die verschiedene Programmiersprachen, Betriebssystem, Architekturen und Plattformen verfügbar.
Das Tool ist in purem C/C++ Code entwickelt. Es lässt sich so auf jede beliebige Plattform / Architektur mit geringstem Aufwand portieren. Die Library stellt die notwendigen Funktionen zur Kommunikation bereit.
Die Verbindung zur SPS wird von IP-S7-LINK eigenständig kontrolliert und im Fehlerfall automatisch wieder hergestellt.
Zur Kopplung wird nur die IP-Adresse der SPS / CP sowie der Steckplatz der CPU im SPS- Rack benötigt.
Sofort können Merker, Eingänge, Ausgänge, Datenbausteine, SPS-Zeit etc. gelesen und geschrieben werden.

Funktionen im Detail

Bitte beachten Sie: Die Funktionen werden mit der Standard Socket-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 Systems zuständig ist. Folgende Funktionen stehen zur Verfügung:

Initialisierung

IPS7Open / IPS7OpenPG / IPS7OpenS7200

Funktionen im Detail

Name	Name (PHP)	Beschreibung / Zweck
IPS7Open	ips7_open	zur Initialisierung der Verbindung, dort wird nur Speicher vorbereitet. Beim ersten Aufruf der Lese- oder Schreibfunktionen wird die TCP/IP-Verbindung automatisch gestartet. Die Verbindung wird über den OP-Kanal hergestellt.
IPS7OpenPG	ips7_openpg	Ab Version 1.17, zur Initialisierung der Verbindung, dort wird nur Speicher vorbereitet. Beim ersten Aufruf der Lese- oder Schreibfunktionen wird die TCP/IP-Verbindung automatisch gestartet. Die Verbindung wird über den PG-Kanal hergestellt.
IPS7OpenS7200	ips7_opens7200	Ab Version 1.21, zur Initialisierung der Verbindung, dort wird nur Speicher vorbereitet. Beim ersten Aufruf der Lese- oder Schreibfunktionen wird die TCP/IP-Verbindung automatisch gestartet. Die Verbindung wird zu einer S7-200 hergestellt.

Aufrufparameter

Nr	Datentyp	PHP-Datentyp	Name	Funktion
1	32-Bit Pointer auf C-String	string	IPAdr	IP-Adresse der SPS im Format: xxx.xxx.xxx.xxx. Beispiel: „192.169.0.100“
2	32-Bit unsigned	long	Rack	Nummer des Rack’s, in dem die SPS-CPU gesteckt ist. Die Zählung beginnt mit „0„. Normalerweise 0. Bei S7-200 egal
3	32-Bit unsigned	long	Slot	Nummer des Steckplatzes der CPU beginnend mit „1“, normalerweise „2„ Bei S7-300-400 oder „1“ bei S7-1200/1500. Bei S7-200 egal
4	32-Bit unsigned	long	RxTimeout	Timeout in Millisekunden für Warten auf TCP/IP-Paket von der SPS, 0 bedeutet Standardeinstellung = 500 ms
5	32-Bit unsigned	long	TxTimeout	Timeout in Millisekunden für Senden eines TCP/IP-Paketes an die SPS, 0 bedeutet Standardeinstellung = 500 ms
6	32-Bit unsigned	long	ConTimeout	Timeout in Millisekunden für Warten auf Verbindungsaufbau mit SPS, 0 bedeutet Standardeinstellung = 5000 ms muss bei Bedarf verlängert werden

C/C++ Funktionsdeklaration

```
extern long WINAPI
IPS7Open (LPCTSTR IPAdr, DWORD Rack, DWORD Slot,
          DWORD RxTimeout, DWORD TxTimeout, DWORD ConnectTimeout);

extern long WINAPI
IPS7OpenPG (LPCTSTR IPAdr, DWORD Rack, DWORD Slot,
            DWORD RxTimeout, DWORD TxTimeout, DWORD ConnectTimeout);

extern long WINAPI
IPS7OpenS7200 (LPCTSTR IPAdr, DWORD Rack, DWORD Slot,
               DWORD RxTimeout, DWORD TxTimeout, DWORD ConnectTimeout);
```

Delphi / Pascal Funktionsdeklaration

```
FUNCTION
IPS7Open (IPAdr : PChar; Rack : LongWord; Slot : LongWord;
   RxTimeout : LongWord; TxTimeout : LongWord ; ConnectTimeout : LongWord) : LongInt;
   stdcall; external 'IPS7LNK.DLL';

FUNCTION
IPS7OpenPG (IPAdr : PChar; Rack : LongWord; Slot : LongWord;
   RxTimeout : LongWord; TxTimeout : LongWord ; ConnectTimeout : LongWord) : LongInt;
   stdcall; external 'IPS7LNK.DLL';

FUNCTION
IPS7OpenS7200 (IPAdr : PChar; Rack : LongWord; Slot : LongWord;
   RxTimeout : LongWord; TxTimeout : LongWord ; ConnectTimeout : LongWord) : LongInt;
   stdcall; external 'IPS7LNK.DLL';
```

Visual Basic Funktionsdeklaration

```
Declare Function IPS7Open& Lib "IPS7LNK.dll" (ByVal IPAdr As String, _
                                              ByVal Rack&, ByVal Slot&, _
                                              ByVal RxTimeout&, _
                                              ByVal TxTimeout&, _
                                              ByVal ConnectTimeout&)
```

IPS7OpenEx

Name	Name (PHP)	Beschreibung / Zweck
IPS7OpenEx	ips7_openex	Ab Version 1.23, zur Initialisierung der Verbindung, dort wird nur Speicher vorbereitet. Beim ersten Aufruf der Lese- oder Schreibfunktionen wird die TCP/IP-Verbindung automatisch gestartet. Es können durch die Wahl der Parameter OP/PG S7200 oder auch Verbindungen über ein Subnetz hergestellt werden.

Aufrufparameter

Nr	Datentyp	Datentyp (PHP)	Name	Funktion
1	32-Bit pointer auf C-String	string	IPAdr	IP-Adresse der SPS im Format: xxx.xxx.xxx.xxx. Beispiel: 192.169.0.100
2	32-Bit unsigned	long	Rack	Nummer des Rack's, in dem die SPS-CPU gesteckt ist. Die Zählung beginnt mit „0“. Normalerweise 0. Bei S7-200 egal

Nr	Datentyp	Datentyp (PHP)	Name	Funktion
3	32-Bit unsigned	long	Slot	Nummer des Steckplatzes der CPU beginnend mit „1„, normalerweise „2“ bei S7-300-400 oder „1“ bei S7-1200/1500. Bei S7-200 egal
4	32-Bit unsigned	long	SubNet-Id	Subnetz-ID, wenn über ein Subnet zugegriffen werden soll. In der Step-S7 Software wird die Adresse z.B. so dargestellt: 0035 – 0001, geben Sie dann 0x00350001 an. Wird nur bei AccessMode 10 oder 11 verwendet
5	32-Bit unsigned	long	DstMPIAdr	Ziel-MPI-Adresse, wenn über ein Subnetz die Verbindung aufgebaut werden soll. Siehe AccessMode 10 und 11!
6	32-Bit unsigned	long	AccessMode	Art des Zugriffs: 0 = OP-Verbindung zur durch Rack und Slot angegebene CPU aufbauen 1 = PG-Verbindung zur durch Rack und Slot angegebene CPU aufbauen 2 = Verbindung zu einer S7-200 über gesteckten TCP/IP-CP 3 = Verbindung einer Siemens-Logo-SPS (ab 1.60.78) 4 = Verbindung über den Kanal „Sonstige“ (ab 1.60.79) 10 = OP-Verbindung über Subnetz, welches an der durch Rack und Slot angegebene CPU angeschlossen ist aufbauen, Subnet-Id und DstMPI Adresse sind anzugeben 11 = PG-Verbindung über Subnetz, welches an der durch Rack und Slot angegebene CPU angeschlossen ist aufbauen, SubnetId und DstMPI Adresse sind anzugeben 20 = Verbindung zu S5-LAN++. Die Konvertierung der Realwerte erfolgt dann auch bei MultiRead-Zugriffen
7	32-Bit unsigned	long	RxTimeout	Timeout in Millisekunden für Warten auf TCP/IP-Paket von der SPS, 0 bedeutet Standardeinstellung = 500 ms
8	32-Bit unsigned	long	TxTimeout	Timeout in Millisekunden für Senden eines TCP/IP-Paketes an die SPS, 0 bedeutet Standardeinstellung = 500 ms
9	32-Bit unsigned	long	ConTimeout	Timeout in Millisekunden für Warten auf Verbindungsaufbau mit SPS, 0 bedeutet Standardeinstellung = 5000 ms (5sec.) muss bei Bedarf verlängert werden.

C/C++ Funktionsdeklaration

```
extern long WINAPI
IPS7OpenEx (LPCTSTR IPAdr, DWORD Rack, DWORD Slot,
            DWORD SubNetId, DWORD DstMPIAdr, DWORD AccessMode,
            DWORD RxTimeout, DWORD TxTimeout, DWORD ConnectTimeout);
```

Delphi / Pascal Funktionsdeklaration

```
FUNCTION
IPS7OpenEx (IPAdr : PChar; Rack : LongWord; Slot : LongWord;
            SubNetId : LongWord; DstMPIAdr : LongWord;    AccessMode : LongWord;
            RxTimeout : LongWord; TxTimeout : LongWord ; ConnectTimeout : LongWord) :
LongInt;
            stdcall; external 'IPS7LNK.DLL';
```

Visual Basic Funktionsdeklaration

```
Declare Function IPS7OpenEx& Lib "IPS7LNK.dll" (ByVal IPAdr As String, _
                                                 ByVal Rack&, ByVal Slot&, _
                                                 ByVal SubNetId&, ByVal DstMPIAdr&, ByVal
AccessMode&, _
                                                 ByVal RxTimeout&, _
                                                 ByVal TxTimeout&, _
                                                 ByVal ConnectTimeout&)
```

IPS7OpenExWithTSAP

Arbeitet wie IPS7OpenEx. Jedoch bietet diese Funktion die Möglichkeit, den LocalTSAP und RemoteTSAP anzugeben. So ist es möglich eine beliebige Verbindung zur SPS mit freier TSAP-Wahl zu verwenden. Die Länge der TSAPs ist auf 2 Byte festgelegt. Dies kann z.B. für die Verbindung mit einer Logo8 Anwednung finden. Dort kann der TSAP für den Kanal festgelegt werden.
Beispiel: Wenn in der SPS vorgegeben ist.

- eigner TSAP: 00.02 → RemoteTSAP[0] = 0x00; RemoteTSAP[1] = 0x02;
- fremder TSAP: 00.03 → LocalTSAP[0] = 0x00; LocalTSAP[1] = 0x03;

Also beachten: Local / Remote bezieht sich auf die Sichtweise zum Partner.

Name	Name (PHP)	Beschreibung / Zweck
IPS7OpenExWithTSAP	ips7_openexwithtsap	

Aufrufparameter

Nr	Datentyp	Datentyp (PHP)	Name	Funktion
1	32-Bit pointer auf C-String	string	IPAdr	IP-Adresse der SPS im Format: xxx.xxx.xxx.xxx. Beispiel: 192.169.0.100
2	32-Bit unsigned	long	Rack	Nummer des Rack's, in dem die SPS-CPU gesteckt ist. Die Zählung beginnt mit „0„. Normalerweise 0. Bei S7-200 egal
3	32-Bit unsigned	long	Slot	Nummer des Steckplatzes der CPU beginnend mit „1“, normalerweise „2„ bei S7-300-400 oder „1“ bei S7-1200/1500. Bei S7-200 egal
4	32-Bit unsigned	long	SubNet-Id	Subnetz-ID, wenn über ein Subnet zugegriffen werden soll. In der Step-S7 Software wird die Adresse z.B. so dargestellt: 0035 – 0001, geben Sie dann 0x00350001 an. Wird nur bei AccessMode 10 oder 11 verwendet
5	32-Bit unsigned	long	DstMPIAdr	Ziel-MPI-Adresse, wenn über ein Subnetz die Verbindung aufgebaut werden soll. Siehe AccessMode 10 und 11!

Nr	Datentyp	Datentyp (PHP)	Name	Funktion
6	32-Bit unsigned	long	AccessMode	Art des Zugriffs: 0 = OP-Verbindung zur durch Rack und Slot angegebene CPU aufbauen 1 = PG-Verbindung zur durch Rack und Slot angegebene CPU aufbauen 2 = Verbindung zu einer S7-200 über gesteckten TCP/IP-CP 3 = Verbindung einer Siemens-Logo-SPS (ab 1.60.78) 4 = Verbindung über den Kanal „Sonstige“ (ab 1.60.79) 10 = OP-Verbindung über Subnetz, welches an der durch Rack und Slot angegebene CPU angeschlossen ist aufbauen, Subnet-Id und DstMPI Adresse sind anzugeben 11 = PG-Verbindung über Subnetz, welches an der durch Rack und Slot angegebene CPU angeschlossen ist aufbauen, SubnetId und DstMPI Adresse sind anzugeben 20 = Verbindung zu S5-LAN++. Die Konvertierung der Realwerte erfolgt dann auch bei MultiRead-Zugriffen
7	32-Bit unsigned	long	RxTimeout	Timeout in Millisekunden für Warten auf TCP/IP-Paket von der SPS, 0 bedeutet Standardeinstellung = 500 ms
8	32-Bit unsigned	long	TxTimeout	Timeout in Millisekunden für Senden eines TCP/IP-Paketes an die SPS, 0 bedeutet Standardeinstellung = 500 ms
9	32-Bit unsigned	long	ConTimeout	Timeout in Millisekunden für Warten auf Verbindungsaufbau mit SPS, 0 bedeutet Standardeinstellung = 5000 ms (5sec.) muss bei Bedarf verlängert werden.
10	pointer auf BYTE Array der Länge 2	BYTE *	LocalTSAP	
11	pointer auf BYTE Array der Länge 2	BYTE *	RemoteTSAP	

C/C++ Funktionsdeklaration

```
extern long WINAPI
IPS7OpenExWithTSAP (LPCTSTR IPAdr, DWORD Rack, DWORD Slot,
            DWORD SubNetId, DWORD DstMPIAdr, DWORD AccessMode,
            DWORD RxTimeout, DWORD TxTimeout, DWORD ConnectTimeout, BYTE *LocalTSAP, BYTE
*RemoteTSAP;
```

Delphi / Pascal Funktionsdeklaration

```
FUNCTION
IPS7OpenExWithTSAP (IPAdr : PChar; Rack : LongWord; Slot : LongWord;
            SubNetId : LongWord; DstMPIAdr : LongWord;    AccessMode : LongWord;
            RxTimeout : LongWord; TxTimeout : LongWord ; ConnectTimeout : LongWord;
LocalTSAP :PBYTE, RemoteTSAP :PBYTE ) : LongInt;
            stdcall; external 'IPS7LNK.DLL';
```

Visual Basic Funktionsdeklaration

```
Declare Function IPS7OpenExWithTSAP& Lib "IPS7LNK.dll" (ByVal IPAdr As String, _
                                                         ByVal Rack&, ByVal Slot&, _
                                                         ByVal SubNetId&, ByVal DstMPIAdr&, ByVal
AccessMode&, _
                                                         ByVal RxTimeout&, _
                                                         ByVal TxTimeout&, _
                                                         ByVal ConnectTimeout&, _
                                                         LocalTSAP as Byte, _
                                                         RemoteTSAP as Byte, _
                                                         )
```

Return Werte

Return-Werte

Wert	Fehlerbeschreibung	Bedeutung
>= 0	Alles OK	Der Returnwert ist die Referenznummer für diese Verbindung und muss bei allen anderen Funktionen als Eingangsparameter Ref verwendet werden
-2	Keine Ressourcen mehr frei	Maximale Anzahl an verfügbaren Verbindungen erreicht
-10	AccessMode nicht möglich (ab 1.23)	wenn eine unzulässige Nummer für AccessMode angegeben wird. Siehe IPS7OpenEx

Deinitialisierung

IPS7Close

Dient zur Deinitialisierung der Verbindung, Speicher wird freigegeben und die TCP/IP-Verbindung wird getrennt.

Aufrufparameter

Nr	Datentyp	Datentyp (PHP)	Name	Funktion
1	32-Bit Wert unsigned	long	Ref	Die Referenz der Verbindung, welche mit IPS7Open generiert wurde. Dient zur internen Identifikation der Verbindung

C/C++ Funktionsdeklaration

```
extern long WINAPI
IPS7Close (long Ref);
```

Delphi / Pascal Funktionsdeklaration

```
FUNCTION
IPS7Close (Ref : LongInt) : LongInt; stdcall;
external 'IPS7LNK.DLL';
```

Visual Basic Funktionsdeklaration

```
Declare Function IPS7Close& Lib "IPS7LNK.dll" (ByVal Ref&)
```

Return-Wert Die Funktion liefert einen 32-Bit Wert mit Vorzeichen als Return-Wert mit folgender Bedeutung:		
Wert	**Fehlerbeschreibung**	**Bedeutung/Reaktion**
0	alles OK	Speicher wieder freigegeben und Verbindung, wenn vorhanden, geschlossen
-3	Mit der angegebenen Referenznummer wurde kein IPS7Open durchgeführt	Haben Sie IPS7Open aufgerufen?
-99	Die Referenznummer ist ungültig	
-30	nur PHP	Die Anzahl oder der Typ der übergebenen Parameter ist falsch
-31	nur PHP	Die interne Konvertierung der Daten konnte nicht durchgeführt werden, z.B. wurde ein String übergeben, wo long notwendig ist

Verbindung

IPS7Connect

Führt einen explizierten Verbindungsaufbau zur SPS aus. Ab Version 1.35! Damit kann ohne Lese/Schreibauftrag die Verbindung zur SPS hergestellt werden.

Aufrufparameter

Nr	Datentyp	Datentyp (PHP)	Name	Funktion
1	32-Bit unsigned	long	Ref	Die Referenz der Verbindung, welche mit IPS7Open generiert wurde. Dient zur internen Identifikation der Verbindung

C/C++ Funktionsdeklaration

```
extern long WINAPI
IPS7Connect (long Ref);
```

Delphi / Pascal Funktionsdeklaration

```
FUNCTION
IPS7Connect (Ref : LongInt) : LongInt;
  stdcall; external 'IPS7LNK.DLL';
```

Return-Werte

Wert	Fehlerbeschreibung
1	Verbindung wurde hergestellt
⇐ 0	Verbindung konnte nicht hergestellt werden. Die genaue Bedeutung finden Sie bei den Return-Werten für die Lese- / Schreibfunktionen

IPS7GetConnectStatus

Prüft den TCP/IP Verbindungsstatus zur SPS.

Aufrufparameter

Nr	Datentyp	Datentyp (PHP)	Name	Funktion
1	32-bit unsigned	long	Ref	Die Referenz der Verbindung, welche mit IPS7Open generiert wurde. Dient zur internen Identifikation der Verbindung

C/C++ Funktionsdeklaration

```
extern long WINAPI
IPS7GetConnectStatus (long Ref);
```

Delphi / Pascal Funktionsdeklaration

```
FUNCTION
IPS7GetConnectStatus ( Ref : LongInt) : LongInt;
  stdcall; external 'IPS7LNK.DLL';
```

Visual Basic Funktionsdeklaration

```
Declare Function IPS7GetConnectStatus& Lib "IPS7LNK.dll" (ByVal Ref&)
```

Return-Werte

Wert	Fehlerbeschreibung
1	Verbindung wurde hergestellt und besteht
⇐ 0	Verbindung ist unterbrochen. Evtl. IPS7Connect, Lese- oder Schreibfunktion aufrufen

IPS7SetKeepAlive

Ab V 1.35! Setzt individuelle TCP/IP KeepAlive Zeiten für die mit Ref angegebene Verbindung. Muss nur verwendet werden, wenn die Standardwerte nicht gelten sollen.
Sie sollten diese Funktion unmittelbar nach dem „Open“- Aufruf ausführen.

Aufrufparameter

Nr	Datentyp	Datentyp (PHP)	Name	Funktion
1	32-bit unsigned	long	Ref	Die Referenz der Verbindung, welche mit IPS7Open generiert wurde. Dient zur internen Identifikation der Verbindung
2	32-bit unsigned	long	AliveTime	Findet innerhalb der Zeit **AliveTime** (ms) kein Datenverkehr auf der TCP/IP-Verbindung statt, so wird ein KeepAlive-Telegram gesendet, um die Verbindung zu prüfen. Wird bei dieser Prüfung ein Fehler festgestellt, sendet der IP-Stack innerhalb der Zeit **AliveInterval** (ms) ein nächstes KeepAlive-Telegram. Dies wird einige male innerhalb der Zeit **AliveInterval** wiederholt (Win 6mal). War der Vorgang nicht erfolgreich, wird die Verbindung beendet

Nr	Datentyp	Datentyp (PHP)	Name	Funktion
3	32-bit unsigned	long	AliveInterval	Das Intervall in ms, in welchem KeepAlive-Telegramme wiederholt werden. Dieses wird aktiv, wenn ein Fehler beim Senden / Empfangen eines KeepAlive-Telegramms aufgetreten ist

C/C++ Funktionsdeklaration

```
extern long WINAPI
IPS7SetKeepAlive (long Ref, DWORD AliveInterval, DWORD AliveTime);
```

Delphi / Pascal Funktionsdeklaration

```
FUNCTION
IPS7SetKeepAlive (Ref : LongInt; AliveInterval : Longword; AliveTime : Longword) : LongInt;
  stdcall; external 'IPS7LNK.DLL';
```

Visual Basic Funktionsdeklaration

```
Declare Function IPS7SetKeepAlive& Lib "IPS7LNK.dll" (ByVal Ref&, ByVal AliveInterval&,
ByVal AliveTime&)
```

Return-Werte

Wert	Fehlerbeschreibung
0	Setzen der Werte war erfolgreich
< 0	Das Setzen der Keep-Alive-Zeit konnte nicht ausgeführt werden

IPS7SetTCPPort

Standardmässig wird der TCP/IP-Port 102 (RFC 1006) verwendet. Mit IPS7SetTCPPort lässt sich dieser verändern. Sie sollten diese Funktion unmittelbar nach dem „Open“- Aufruf ausführen.

Aufrufparameter

Nr	Datentyp	Datentyp (PHP)	Name	Funktion
1	32-Bit unsigned	long	Ref	Die Referenz der Verbindung, welche mit IPS7Open generiert wurde. Dient zur internen Identifikation der Verbindung
2	32-Bit unsigned	long	Port	Die Nummer des neuen TCP/IP-Ports 1 - 65565

C/C++ Funktionsdeklaration

```
extern long WINAPI
IPS7SetTCPPort (long Ref, unsigned long Port);
```

Delphi / Pascal Funktionsdeklaration

```
FUNCTION
IPS7SetTCPPort (Ref : LongInt, Port : LongWord) : LongInt;
  stdcall; external 'IPS7LNK.DLL';
```

Return-Werte

Wert	Fehlerbeschreibung	Bedeutung/Reaktion
0	alles OK	Speicher wieder freigegeben und Verbindung, wenn vorhanden, geschlossen
-3	Mit der angegebenen Referenznummer wurde kein IPS7Open durchgeführt	Haben Sie IPS7Open aufgerufen?
-99	Die Referenznummer ist ungültig	
-30	nur PHP	Die Anzahl oder der Typ der übergebenen Parameter ist falsch
-31	nur PHP	Die interne Konvertierung der Daten konnte nicht durchgeführt werden, z.B. wurde ein String übergeben, wo long notwendig ist

IPS7GetSockErr

Name	Name (PHP)	Beschreibung / Zweck
IPS7GetSockErr	ips7_getsockerr	Liefert den letzten Socket-Fehler zurück

Aufrufparameter

Nr	Datentyp	Datentyp (PHP)	Name	Funktion
1	32-Bit unsigned	long	Ref	Die Referenz der Verbindung, welche mit IPS7Open generiert wurde. Dient zur internen Identifikation der Verbindung

C/C++ Funktionsdeklaration

```
extern long WINAPI
IPS7GetSockErr (long Ref);
```

Delphi / Pascal Funktionsdeklaration

```
FUNCTION
IPS7GetSockErr (Ref : LongInt) : LongInt;
  stdcall; external 'IPS7LNK.DLL';
```

Visual Basic Funktionsdeklaration

```
Declare Function IPS7GetSockErr& Lib "IPS7LNK.dll" (ByVal Ref&)
```

Return-Werte

Wert	Fehlerbeschreibung	Bedeutung / Reaktion
0	alles OK	Es liegt kein Fehler an
-3	Mit der angegebenen Referenznummer wurde kein IPS7Open durchgeführt	Haben Sie IPS7Open aufgerufen?
-99	Die Referenznummer ist ungültig	
Sonstige	Socketerror	Erklärung siehe Liste unterhalb

Lesen und Schreiben

IP-S7-LINK stellt für jeden Datentyp eine Funktionen zum Lesen und Schreiben bereit.
Die Datenyp-Funktionen unterscheiden sich in den Aufrufparametern. Im Folgenden sind die Methoden mit gleichen Parametern in Gruppen zusammengefasst.

Byte

Name	Name (PHP)	Beschreibung/Zweck
IPS7RdB	ips7_rdb	bytweise lesen (E,A,P,M,DB)
IPS7WrB	ips7_wrb	byteweise schreiben (E,A,P,M,DB,Z)

Word

Name	Name (PHP)	Beschreibung/Zweck
IPS7RdW	ips7_rdw	wortweise lesen (E,A,P,M,DB)
IPS7RdPlcW	ips7_rdplcw	wortweise lesen (E,A,P,M,DB) jedoch Startadresse nach SPS-Adressierung damit ein Zugriff auf ungerade Startadressen möglich ist
IPS7WrW	ips7_wrw	wortweise schreiben (E,A,P,M,DB,Z)
IPS7WrPlcW	ips7_wrplcw	wortweise schreiben (E,A,P,M,DB,Z) jedoch Startadresse nach SPS-Adressierung, (ab 1.17) damit ist ein Zugriff auf ungerade Startadressen möglich ist

DWord

Name	Name (PHP)	Beschreibung/Zweck
IPS7RdDW	ips7_rddw	doppelwortweise lesen (E,A,P,M,DB,T)
IPS7WrDW	ips7_wrdw	doppelwortweise schreiben (E,A,P,M,DB,T)

Real

Name	Name (PHP)	Beschreibung/Zweck
IPS7RdReal	ips7_rdreal	Real (Fließpunktzahl) lesen (E,A,P,M,DB))
IPS7WrReal	ips7_wrreal	Real (Fließpunktzahl) schreiben (E,A,P,M,DB))

LInt

Name	Name (PHP)	Beschreibung/Zweck
IPS7RdLInt	ips7_rdlint	LInt (64 Bit) lesen (E,A,P,M,DB)
IPS7WrLInt	ips7_wrlint	LInt (64 Bit) schreiben (E,A,P,M,DB)

LUInt

Name	Name (PHP)	Beschreibung/Zweck
IPS7RdULInt	ips7_rdulint	ULInt (64 Bit) lesen (E,A,P,M,DB)
IPS7WrULInt	ips7_wrulint	ULInt (64 Bit) schreiben (E,A,P,M,DB)

String

Name	Name (PHP)	Beschreibung/Zweck
IPS7RdString	ips7_rdstring	String lesen (E,A,P,M,DB) Startadresse nach SPS-Adressierung 1.Byte mögliche Länge 2. Byte tatsächliche Länge ab 3. Byte Daten (Bsp. Adresse: DB0.DBB0, ab DB0.DBB2 stehen die tatsächlichen Daten). IP-S7-LINK liest mögliche Länge des Strings aus → kürzt auf tatsächliche Länge → fügt NULL-CHAR ('\0') am Ende hinzu
IPS7WrString	ips7_wrstring	String schreiben (E,A,P,M,DB) Startadresse nach SPS-Adressierung. **Der String muss mit NULL-CHAR ('\0') abgeschlossen sein.** Die Länge des übergebenen String wird automatisch ermittelt und beim Schreiben des Inhaltes in den S7-STRING der SPS geschrieben. **Achtung!** Definieren Sie Ihren Datenbereich groß genug, damit es zu keiner Überschreibung vorhandener Prozessdaten kommen kann

Aufrufparameter

Aufrufparameter

Nr	Datentyp	Datentyp (PHP)	Name	Funktion
1	32-Bit unsigned	long	Ref	Die Referenz der Verbindung, welche mit IPS7Open generiert wurde. Dient zur internen Identifikation der Verbindung
2	32-Bit unsigned	long	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 'P' = 80 dez. steht für Peripherie 'T' = 84 dez. steht für Timer (nur mit Doppelwortfunktionen möglich). Die Timer werden in der SPS mit Zeitbasis und Wert im BCD-Format gespeichert. Um dieses Format sofort im PC verarbeiten zu können, führt der Treiber eine automatische Konvertierung in Millisekunden durch. Das kleinst mögliche Raster ist 10 ms. Beim Schreiben in die SPS wählt der Treiber automatisch eine passende Zeitbasis. Dabei kann es zu Rundungen kommen. Der Zeitbereich geht von 0 bis 9990000 ms 'Z' = 90 dez. steht für Zähler (nur mit Wortfunktionen möglich). Auch die Zähler sind in der SPS BCD-Codiert abgelegt. Die Zählerwerte reichen von 0 – 999
3	32-Bit unsigned	32-Bit Wert unsigned	DBNr	Datenbausteinnummer, diese wird nur beim Typ 'D' verwendet. Ansonsten steht dort der Wert „0“
4	32-Bit unsigned	32-Bit Wert unsigned	Ab	Erstes Wort bzw. Byte ab dem gelesen bzw. geschrieben werden soll. Bei Wortoperationen Startwort Bei Byte, Doppelwort und Realfunktionen Startbyte Bei Timer oder Zähler ist dies die Nummer des ersten Elements, welches gelesen werden soll

Nr	Datentyp	Datentyp (PHP)	Name	Funktion
5	32-Bit unsigned	32-Bit Wert unsigned	Anz	Anzahl der Einheiten (Byte, Worte, Doppelworte Real, oder Einheiten, z.B. Timer), die gelesen bzw. geschrieben werden sollen
6	32-Bit Pointer	mixed	Buffer	Die 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 Bei Doppelwort Zeiger auf Long Bei Real Zeiger auf einen Double
				Hinweis für PHP: Bei PHP geben Sie hier die Referenz einer Variablen an
				Bemerkung zu ips7_rdplcw, ips7_rdw, ips7_rddw, ips7_rdreal }} Aufruf z.B.: &Result = ips7_rdplcw ($Ref, ord („M“), 0, 6,5, &$W);
				Werden mehr als 1 Element gelesen, so wird die Variable in ein Array vom Typ long bzw. double umgewandelt. Wird nur ein Wert gelesen und die Variable ist kein Array, wird der Wert als long gespeichert Ist die Variable bereits ein Array und es wird nur ein Wert gelesen, so wird das Ergebnis im ersten Element des Array abgelegt. Doppelworte (ips7_rddw) werden grundsätzlich mit Vorzeichen verarbeitet (signed)
7	———	long (optional)	bSigned bei ipss7_rdplcw, ips7_rdw, ips7_wrplcw, ips7_wrdw bLong bei ips7_rdb	**Hinweis für PHP:** ips7_rdplcw, ips7_rdw, ips7_wrplcw, ips7_wrdw Optional kann bestimmt werden, ob die Werte als signed oder unsigned 16-Bit-Integer gelesen werden sollen. Wird der Parameter nicht angegeben, wird grundsätzlich mit Vorzeichen gearbeitet (signed). Wir der Parameter übergeben gilt: 0 = ohne Vorzeichen (signed) 1 = mit Vorzeichen \\Außerdem stehen für eine nachträgliche Konvertierung einzelner Werte die Funktionen **ips7_i2w** und **ips7_w2i** zu Verfügung. Näheres finden Sie dort.**ips7_rdb**Ips7_rdb speichert das Ergebnis grundsätzlich als string. Wollen Sie jedoch die Werte einfach als Array ansprechen, so können Sie mit bLong = 1 das Ergebnis als long.Array ablegen lassen.

Bitte beachten beim Wortweisen Zugriff

Beim wortweisen Zugriff prüfen Sie bitte, ob Sie die Anfangsadresse nach SPS-Syntax verwenden möchten oder aber die rechnerisch korrekte.

Je nachdem müssen Sie IPS7RdPlcW oder IPS7RdW bzw. IPS7WrPlcW oder IPS7RdW verwenden.

Der PC und die SPS haben verschiedene Adressierungsarten. In der S7 ist der Speicherbereich byteweise orientiert. So adressieren Sie aus der Sicht des SPS-Programmierers mit MW 0 das MB0 und MB1, mit MW1 aber das MB1 und MB2. Sie sehen, dass sich MW0 und MW1 im MB1 überschneiden.

Vor der Version 1.17 war es nur möglich auf gerade Startadressen mit den Wortfunktionen zuzugreifen. Ab V 1.17 ist mit den Funktionen **S7RdPlcW und S7WrPlcW** ein Zugriff auf ungerade Startadressen möglich. Wollen Sie nun MW 1 lesen, so wie der SPS-Programmierer das sieht rufen Sie auf:

IPS7RdPlcW (Ref, 'M', 0, 1, 1, WortBuffer); *!!! Beachten Sie das bei Wortoperationen mit S7RdW und S7WrW!!!*

Beispiel für Merker. Dies gilt auch für Eingänge, Ausgänge und Datenworte. Die Wortadressierung in der SPS belegt jeweils folgende Bytes:

Wortadresse	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 (IPS7RdW und IPS7WrW) 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 ***IPS7RdW (Ref, Typ, DBNr, 0, 5, Buf)*** hat folgende Wirkung:

PC	SPS
Buf[0]	DW 0
Buf[1]	DW 2
Buf[2]	DW 4

Sie müssen also die Start-Wortnummer halbieren, um im PC richtig zugreifen zu können. Dies gilt auch für Datenbausteine –> Ungerade Wortadressen der SPS können nicht wortweise gelesen oder geschrieben werden.
Wollen Sie trotzdem auf ungerade Startadressen adressieren verwenden Sie die Funktionen S7RdPlcW und S7WrPlcW.

Zusatzfunktionen für PHP

long ips7_w2i(mixed Buffer,long Count);
Konvertiert unsigned 16-Bit - Werte in signed 16-Bit Werte

long ips7_i2w(mixed Buffer, long Count);
Konvertiert signed 16-Bit - Werte in unsigned 16-Bit Werte

Parameter	Beschreibung / Zweck
Buffer	Referenz auf die long Werte (Array) oder den long Wert, der konvertiert werden soll
Count	Anzahl der Werte

C/C++ Funktionsdeklaration

```
extern long WINAPI
IPS7RdW (long Ref, DWORD Typ, DWORD DBNr, DWORD AbWort, DWORD WortAnz, LPWORD Buffer) ;

extern long WINAPI // 1.17
IPS7RdPlcW (long Ref, DWORD Typ, DWORD DBNr, DWORD AbWort, DWORD WortAnz, LPWORD Buffer) ;

extern long WINAPI
IPS7RdB (long Ref, DWORD Typ,  DWORD DBNr, DWORD Ab, DWORD Anz, LPBYTE Buffer);

extern long WINAPI
IPS7WrW (long Ref, DWORD Typ, DWORD DBNr, DWORD Ab, DWORD Anz, LPWORD Buffer);

extern long WINAPI // 1.17
IPS7WrPlcW (long Ref, DWORD Typ, DWORD DBNr, DWORD Ab, DWORD Anz, LPWORD Buffer);

extern long WINAPI
IPS7WrB (long Ref, DWORD Typ, DWORD DBNr, DWORD Ab, DWORD Anz, LPBYTE Buffer);

extern long WINAPI
IPS7RdDW (long Ref, DWORD Typ, DWORD DBNr, DWORD Ab, DWORD Anz, LPDWORD Buffer);

extern long WINAPI
IPS7WrDW (long Ref, DWORD Typ, DWORD DBNr, DWORD Ab, DWORD Anz, LPDWORD Buffer);

extern long WINAPI
IPS7RdReal (long Ref, DWORD Typ, DWORD DBNr, DWORD Ab, DWORD Anz, double *Buffer);

extern long WINAPI
IPS7WrReal (long Ref, DWORD Typ, DWORD DBNr, DWORD Ab, DWORD Anz, double *Buffer);

extern long WINAPI
IPS7RdULInt (long Ref, DWORD Typ, DWORD DBNr, DWORD Start, DWORD Cnt, UINT64 *Buffer) ;

extern long WINAPI
IPS7WrULInt (long Ref, DWORD Typ, DWORD DBNr, DWORD Start, DWORD Cnt, UINT64 *Buffer) ;

extern long WINAPI
IPS7RdLInt (long Ref, DWORD Typ, DWORD DBNr, DWORD Start, DWORD Cnt, INT64 *Buffer);

extern long WINAPI
IPS7WrLInt (long Ref, DWORD Typ, DWORD DBNr, DWORD Start, DWORD Cnt, INT64 *Buffer);
```

Delphi / Pascal Funktionsdeklaration

```
FUNCTION
IPS7RdW (Ref : LongInt; Typ : Longword; DBNr : Longword;
  AbWort : Longword; WortAnz : Longword;  Buffer : PWORD)  : LongInt;
  stdcall; external 'IPS7LNK.DLL';

FUNCTION
IPS7RdPlcW (Ref : LongInt; Typ : Longword; DBNr : Longword;
  AbWort : Longword; WortAnz : Longword;  Buffer : PDWORD)  : LongInt;
  stdcall; external 'IPS7LNK.DLL';

FUNCTION
IPS7RdB (Ref : LongInt; Typ : Longword;  DBNr: Longword;
  Ab : Longword;  Anz : Longword; Buffer: PBYTE) : LongInt;
  stdcall; external 'IPS7LNK.DLL';

FUNCTION
IPS7WrW (Ref : LongInt; Typ : Longword; DBNr : Longword;
  AbWort : Longword; WortAnz : Longword; Buffer : PWORD)  : LongInt;
  stdcall; external 'IPS7LNK.DLL';

FUNCTION
IPS7WrPlcW (Ref : LongInt; Typ : Longword; DBNr : Longword;
  AbWort : Longword; WortAnz : Longword;  Buffer : PDWORD)  : LongInt;
  stdcall; external 'IPS7LNK.DLL';

FUNCTION
IPS7WrB (Ref : LongInt; Typ : Longword;  DBNr: Longword;
  Ab : Longword;  Anz : Longword; Buffer : PBYTE) : LongInt;
  stdcall; external 'IPS7LNK.DLL';

FUNCTION
IPS7RdDW (Ref : LongInt; Typ : Longword; DBNr : Longword;
  AbWort : Longword; WortAnz : Longword;  Buffer : PDWORD)  : LongInt;
  stdcall; external 'IPS7LNK.DLL';

FUNCTION
IPS7WrDW (Ref : LongInt; Typ : Longword; DBNr : Longword;
  AbWort : Longword; WortAnz : Longword;  Buffer : PDWORD)  : LongInt;
  stdcall; external 'IPS7LNK.DLL';

FUNCTION
IPS7RdReal (Ref : LongInt; Typ : Longword; DBNr : Longword;
  AbWort : Longword; WortAnz : Longword;  Buffer : PDOUBLE)  : LongInt;
  stdcall; external 'IPS7LNK.DLL';

FUNCTION
IPS7WrReal (Ref : LongInt; Typ : Longword; DBNr : Longword;
  AbWort : Longword; WortAnz : Longword;  Buffer : PDOUBLE)  : LongInt;
  stdcall; external 'IPS7LNK.DLL';
```

Visual Basic Funktionsdeklaration

```
Declare Function IPS7RdW& Lib "IPS7LNK.dll" (ByVal Ref&, ByVal Typ&, ByVal DBNr&, ByVal
AbWort&, ByVal WortAnz&, Wert As Integer)

Declare Function IPS7RdPlcW& Lib "IPS7LNK.dll" (ByVal Ref&, ByVal Typ&, ByVal DBNr&, ByVal
AbWort&, ByVal WortAnz&, Wert As Integer)

Declare Function IPS7RdB& Lib "IPS7LNK.dll" (ByVal Ref&, ByVal Typ&, ByVal DBNr&, ByVal
AbWort&, ByVal WortAnz&, Wert As Byte)

Declare Function IPS7WrW& Lib "IPS7LNK.dll" (ByVal Ref&, ByVal Typ&, ByVal DBNr&, ByVal
AbWort&, ByVal WortAnz&, Wert As Integer)

Declare Function IPS7WrW& Lib "IPS7LNK.dll" (ByVal Ref&, ByVal Typ&, ByVal DBNr&, ByVal
AbWort&, ByVal WortAnz&, Wert As Integer)

Declare Function IPS7WrB& Lib "IPS7LNK.dll" (ByVal Ref&, ByVal Typ&, ByVal DBNr&, ByVal
AbWort&, ByVal WortAnz&, Wert As Byte)

Declare Function IPS7RdDW& Lib "IPS7LNK.dll" (ByVal Ref&, ByVal Typ&, ByVal DBNr&, ByVal
AbWort&, ByVal WortAnz&, Wert As Long)

Declare Function IPS7WrDW& Lib "IPS7LNK.dll" (ByVal Ref&, ByVal Typ&, ByVal DBNr&, ByVal
AbWort&, ByVal WortAnz&, Wert As Long)

Declare Function IPS7RdReal& Lib "IPS7LNK.dll" (ByVal Ref&, ByVal Typ&, ByVal DBNr&, ByVal
AbWort&, ByVal WortAnz&, Wert As Double)

Declare Function IPS7WrReal& Lib "IPS7LNK.dll" (ByVal Ref&, ByVal Typ&, ByVal DBNr&, ByVal
AbWort&, ByVal WortAnz&, Wert As Double)
```

C/C++ Funktionsdeklaration

```
extern long WINAPI
IPS7RdStr (long Ref, DWORD Typ, DWORD DBNr, DWORD Start, DWORD Cnt, LPSTR Buffer);

extern long WINAPI
IPS7WrStr (long Ref, DWORD Typ, DWORD DBNr, DWORD Ab, LPSTR Buffer);
```

Delphi / Pascal Funktionsdeklaration

```
FUNCTION
IPS7RdStr (Ref : LongInt; Typ : Longword; DBNr : Longword;
  Start : Longword; Cnt : Longword;  Buffer : ANSISTRING)  : LongInt;
  stdcall; external 'IPS7LNK.DLL';

FUNCTION
IPS7WrStr (Ref : LongInt; Typ : Longword; DBNr : Longword;
  Ab : Longword; Buffer : ANSISTRING)  : LongInt;
  stdcall; external 'IPS7LNK.DLL';
```

Visual Basic Funktionsdeklaration

```
Declare Function IPS7RdStr& Lib "IPS7LNK.dll" (ByVal Ref&, ByVal Typ&, ByVal DBNr&, ByVal
Start&, ByVal Cnt&, Wert As String)

Declare Function IPS7WrStr& Lib "IPS7LNK.dll" (ByVal Ref&, ByVal Typ&, ByVal DBNr&, ByVal
Ab&, Wert As String)
```

Aufrufparameter

Nr	Datentyp	Datentyp (PHP)	Name	Funktion
1	32-Bit unsigned	long	Ref	Die Referenz der Verbindung, welche mit IPS7Open generiert wurde. Dient zur internen Identifikation der Verbindung
2	32-Bit unsigned	long	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 'P' = 80 dez. steht für Peripherie
3	32-Bit unsigned	long	DBNr	Datenbausteinnummer, diese wird nur beim Typ 'D' verwendet. Ansonsten steht dort der Wert „0„
4	32-Bit unsigned	long	Ab	Byte Adresse z.B. M 10.0, dann steht hier 10. Beachten Sie hier den Unterschied zwischen IPS7RdW und IPS7RdPlcW bzw. IPS7WrW und IPS7WrPlcW. Siehe weiter unten!
5	32-Bit unsigned	long	Bit	BitNr muss zwischen 0 und 7 liegen, z.B. bei M5.4 steht hier 4
6	32-Bit Adresse	mixed	Buffer	**Dieser Parameter ist nur für IPS7RdBit** Die Adresse auf den Zielspeicher im PC. Zeiger auf ein Byte. Wenn bBit gesetzt ist Inhalt 1 sonst 0. Beispiel: lese M 6.5 BYTE W; IPS7RdBit (Ref, 'M', 0, 6,5 & W); **Hinweis für PHP:** Bei PHP geben Sie als hier die Referenz einer Variablen an: ips7_rdbit (Ref, ord („M“), 0, 6,5, & $W); Diese Variable wird automatisch in einen „long“ konvertiert. Verwenden Sie daher eine Variable, die sonst noch nicht verwendet wurde. Der Zustand des Bit’s (0 oder 1) ist somit als long gespeichert

C/C++ Funktionsdeklaration

```
<code c>
extern long WINAPI
IPS7RdBit (long Ref, DWORD Typ, DWORD DBNr, DWORD Byte, DWORD Bit, LPBYTE Buffer);

extern long WINAPI
IPS7SetBit (long Ref, DWORD Typ, DWORD DBNr, DWORD Byte, DWORD Bit);

extern long WINAPI
IPS7ResetBit (long Ref, DWORD Typ, DWORD DBNr, DWORD Byte, DWORD Bit);
```

Delphi / Pascal Funktionsdeklaration

```
FUNCTION
IPS7RdBit (Ref : LongInt; Typ : Longword;  DBNr: Longword;
           ByteNr : Longword;  BitNr : Longword; Buffer: PBYTE) : LongInt;
           stdcall; external 'IPS7LNK.DLL';

FUNCTION
IPS7SetBit (Ref : LongInt; Typ : Longword;  DBNr: Longword;
            ByteNr : Longword;  BitNr : Longword) : LongInt;
            stdcall; external 'IPS7LNK.DLL';

FUNCTION
IPS7ResetBit (Ref : LongInt; Typ : Longword;  DBNr: Longword;
            ByteNr : Longword;  BitNr : Longword) : LongInt;
            stdcall; external 'IPS7LNK.DLL';
```

Visual Basic Funktionsdeklaration

```
Declare Function IPS7RdBit& Lib "IPS7LNK.dll" (ByVal Ref&, ByVal Typ&, ByVal DBNr&, ByVal
ByteNr&, ByVal BitNr&, Wert As Byte)

Declare Function IPS7SetBit& Lib "IPS7LNK.dll" (ByVal Ref&, ByVal Typ&, ByVal DBNr&, ByVal
ByteNr&, ByVal BitNr&)

Declare Function IPS7ResetBit& Lib "IPS7LNK.dll" (ByVal Ref&, ByVal Typ&, ByVal DBNr&, ByVal
ByteNr&, ByVal BitNr&)
```

Delphi / Pascal Funktionsdeklaration

```
FUNCTION
IPS7RdULInt (Ref : LongInt; Typ : Longword; DBNr : Longword;
  Start : Longword; Cnt : Longword;  Buffer : UInt64)  : LongInt;
  stdcall; external 'IPS7LNK.DLL';

FUNCTION
IPS7WrULInt (Ref : LongInt; Typ : Longword; DBNr : Longword;
  Start : Longword; Cnt : Longword;  Buffer : UInt64)  : LongInt;
  stdcall; external 'IPS7LNK.DLL';
FUNCTION
IPS7RdLInt (Ref : LongInt; Typ : Longword; DBNr : Longword;
  Start : Longword; Cnt : Longword;  Buffer : Int64)  : LongInt;
  stdcall; external 'IPS7LNK.DLL';

FUNCTION
IPS7WrLInt (Ref : LongInt; Typ : Longword; DBNr : Longword;
  Start : Longword; Cnt : Longword;  Buffer : Int64)  : LongInt;
  stdcall; external 'IPS7LNK.DLL';
```

Erläuterungen zu PHP
In PHP sind die Variablen grundsätzlich keinem festen Datentyp zugeordnet. Die Bestimmung des Datentyps übernimmt deshalb das Erweiterungsmodul. Grundsätzlich gilt: Die Ziel- bzw. Quellvariable für den Lese / Schreibpuffer (=Parameter 6), muss als Referenz übergeben werden. Also das „&“ – Zeichen verwenden. Z.B. $Res = ips7_rdplcw ($Ref, ord („M“), 0, 0, 2, &$Werte);
Werte[0] ist MW0
Werte[1] ist MW2
Da es in PHP keine 16-Bit-Werte gibt, werden die 16-Bit-Daten als long gespeichert. Die Wortfunktionen

speichern das Ergebnis in einem long-Array, wird nur eine Einheit gelesen, so wird das Ergebnis als einzelner long gespeichert, wenn die Variable noch kein Array ist. Das Lesen- und Schreiben von Worten (16 Bit) geschieht grundsätzlich mit Vorzeichen. D.h. der Wert wird als 16-Bit integer interpretiert. Sollen die Werte als vorzeichenloser 16-Bit-Wert (unsigned) behandelt werden, beachten Sie dann den optionalen Parameter 7 (bSigned). Die Bytefunktionen speichern das Ergebnis grundsätzlich als string. Wollen Sie jedoch die Werte einfach als long Array ansprechen, so können Sie mit bLong = 1 das Ergebnis als long Array ablegen lassen.

Bit

Name	Name(PHP)	Beschreibung/Zweck
IPS7RdBit	ips7_rdbit	ein Bit lesen SPS (E,A,P,M,DB)
IPS7SetBit	ips7_setbit	ein Bit setzen (E,A,P,M,DB)
IPS7ResetBit	ips7_resetbit	ein Bit zurücksetzen in der SPS (E,A,P,M,DB)

Lesen gemischter Datenbereiche

IPS7RdMulti

Name	Name (PHP)	Beschreibung / Zweck
IPS7RdMulti	ips7_rdmulti	Führt einen gemischten Readauftrag aus. Für die Aufträge wird eine Liste/Array von Records / Structs des Typs „IPS7_RQ_MULTI„ ausgefüllt und der Funktion übergeben. Die Funktion sortiert und bündelt diese Aufträge und führt diese optimiert aus. **ACHTUNG!** Die Lesereihenfolge ist eine andere als die Reihenfolge der übergeben Liste

IPS7RdMulti führt automatisch eine Datenkonvertierung zwischen S7 und PC durch. Die Konvertierung ist mit dem **Datentyp-Casting** in der Programmiersprache zu vergleichen. Die Bitbreite des Datentyps im PC muss gleich oder größer als der in der SPS sein.
Der Aufrufer hat einen entsprechend großen Zielbereich zur Verfügung zu stellen.
Das heißt: Sollen z.B. zwei 16-Bit Werte (gesamt 32 Bit) gelesen und in Doublewerte (64 Bit) im PC konvertiert werden, so müssen im PC auch zwei Doublewerte (insgesamt also 128 Bit) bereit gestellt werden.

Die Konvertierung sieht aus wie im Beispiel:

Wert in der SPS	Wert im PC
6	6.0
1	1.0

Die Auswahl der SPS- und PC-Datentypen erfolgt im Requestauftrag. Codiert über Konstanten wie im Folgendem beschrieben.

IPS7RdMultiCalcPacketCnt

Name	Name (PHP)	Beschreibung / Zweck
IPS7RdMultiCalcPacketCnt	ips7_rdmulticalcpacketcnt	Der Returnwert ist die Anzahl der benötigten Kommunikationspakete , die zum Lesen aller Multiread-Aufträge benötigt werden. Damit kann geprüft werden, ob alle gewünschten Variablen an einem Stück gelesen werden können. Dies ist der Fall, wenn der Returnwert == 1. Bitte den Returnwert beachten: **>= 0** die Anzahl der Pakete **< 0** es ist ein Fehler aufgetreten, Auswertung wie unten

Return Werte

Return-Werte Read/Write

Bei Verwendung der Funktion IPS7RdMulti ist der Returnwert im Strukt-Element „Res“ gespeichert.

Wert	Fehlerbeschreibung	Reaktion
0	Alles OK	Daten auswerten
2	Baustein oder Datenbereich existiert nicht, z.B. Zugriff auf DB, der nicht vorhanden oder zu klein ist	Überprüfen, ob der gewünschte Datenbereich in der SPS vorhanden ist
-1	Zeitüberlauf, gewünschte SPS offensichtlich nicht oder nicht mehr vorhanden	Einfach weitere Schreib- und Leseaufträge absetzen der Treiber baut die Verbindung automatisch auf. Evtl. die Timeoutzeiten insbesondere die Connect-Timeoutzeit verlängern
-2	Die maximale Anzahl (256) der möglichen Verbindungen ist erreicht	Unnötige Verbindungen schließen
-3	Kann bei Close auftreten. Mit der angegebenen Referenz wurde kein Open ausgeführt	Überprüfen Sie Ihren Quellcode, ob die Variable für die Referenz nicht überschrieben wurde. Oder es wurde für diese Referenz bereits ein Close ausgeführt
-5	Allgemeiner Fehler	Prüfen ob Netzwerk richtig im PC installiert ist: TCP/IP aktiviert? Winsocket installiert?
-6	Ziel-CPU nicht gefunden	Rack oder Steckplatznummer falsch. Es ist keine Verbindung zu diesem Steckplatz mehr frei. Im CP Konfiguration prüfen
-7	Socketfehler aufgetreten	IPS7GetSockErr aufrufen und Fehler auswerten
-8	Speicherfehler	angeforderter Speicher im PC ist nicht verfügbar
-9	Bereichsüberschreitung	z.B. Timer > 9990000 ms
-10	Gewünschter Datentyp nicht erlaubt oder wird nicht unterstützt	Prüfen, ob der Code für Datentyp in Ordnung ist
-11	Der angegebene PC-Datentyp ist den angegebenen SPS- Datenbereich nicht möglich	Das kann z.B. vorkommen, wenn auf Zähler zugegriffen werden soll und der PC-Datenbereich als BYTE angegeben wird. Abhilfe: Datenbereich im PC ändern
-20	Der angegebene Speicher im PC ist zu klein (z.B. Array ist zu klein), kann nur bei .net oder PHP auftreten	Datenbereich im PC vergrössern bzw. richtig anpassen
-21	Nur .Net!Für diese Instanz der Klasse wurde schon einmal ein Open ausgeführt	Evtl. Close aufrufen
-31	**Nur bei MultiRead:** PC und S7-Datentyp stehen falscher Relation z.B. PC = BYTE SPS = Wort	PC-Datenbereich anpassen

Wert	Fehlerbeschreibung	Reaktion
-32	**Nur bei MultiRead:** S7 liefert falsche Anzahl Daten für den angegeben Datentyp	
-88	**Nur bei MultiRead:** Der entsprechende Auftrag wurde noch nicht bearbeitet	
-99	Die Referenznummer ist ungültig	Haben Sie IPS7Open aufgerufen?
4660	Demozeit ist abgelaufen	Vollversion erwerben

C/C++ Funktionsdeklaration

```
extern long WINAPI
IPS7RdMulti (long Ref, IPS7_RQ_MULTI *pRqList, DWORD Cnt);

extern long WINAPI
IPS7RdMultiSimplex (long Ref, IPS7_RQ_MULTI *pRqList, DWORD Cnt); // 1.36

extern long WINAPI
IPS7RdMultiBuffered (long Ref, IPS7_RQ_MULTI_BUFFERED *pRqList, DWORD Cnt); // 1.40

extern long WINAPI
IPS7RdMultiGetData (long Ref, IPS7_RQ_MULTI_BUFFERED *pRq, void *pData); // 1.40

extern long WINAPI
IPS7RdMultiCalcPacketCnt (long Ref, IPS7_RQ_MULTI *pRqList, DWORD Cnt);
```

Delphi / Pascal Funktionsdeklaration

```
FUNCTION
IPS7RdMulti (Ref : LongInt; pRqList : PIPS7_RQ_MULTI; Cnt : Longword)  : LongInt;
  stdcall; external 'IPS7LNK.DLL';

FUNCTION
IPS7RdMultiSimplex (Ref : LongInt;  pRqList : PIPS7_RQ_MULTI; Cnt : Longword) : LongInt;
  stdcall; external 'IPS7LNK.DLL';

FUNCTION
IPS7RdMultiBuffered (Ref : LingInt; pRqList : IPS7_RQ_MULTI_BUFFERED; Cnt : LongWord) :
LongInt;
  stdcall; external 'IPS7LNK.DLL';

FUNCTION
IPS7RdMultiGetData (Ref : LongInt; pRq : IPS7_RQ_MULTI_BUFFERED; pData : Pointer) : LongInt;
  stdcall; external 'IPS7LNK.DLL';

FUNCTION
IPS7RdMultiCalcPacketCnt (Ref : LongInt; pRqList : Pointer; Cnt : Longword;) : LongInt;
  stdcall; external 'IPS7LNK.DLL';
```

Der Aufbau des Request Records/Structs für IPS7RdMulti

<table>
<tr><th>Name</th><th>Typ</th><th>Beschreibung / Zweck</th></tr>
<tr><td>DataArea</td><td>32-Bit unsigned</td><td>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
T = 84 dez. steht für Timer
(nur mit Doppelwortfunktionen möglich)
Die Timer werden in der SPS mit Zeitbasis und Wert im BCD-Format gespeichert. Um dieses Format sofort im PC verarbeiten zu können, führt der Treiber eine automatische Konvertierung in Millisekunden durch. Das kleinst mögliche Raster ist 10 ms. Beim schreiben in die SPS wählt der Treiber automatisch eine passende Zeitbasis. Dabei kann es zu Rundungen kommen. Der Zeitbereich geht von 0 bis 9990000 ms
Z = 90 dez. steht für Zähler
Auch die Zähler sind in der SPS BCD-Codiert abgelegt. Die Zählerwerte reichen von 0 – 999</td></tr>
<tr><td>DataType</td><td>32-Bit unsigned</td><td>Datentyp in der SPS
<table>
<tr><th>Name</th><th>Wert</th><th>SPS-Datentyp / Bitbreite</th></tr>
<tr><td>IPS7_BIT</td><td>0</td><td>Ein Bit / Boolean</td></tr>
<tr><td>IPS7_BYTE</td><td>1</td><td>Byte (8Bit)</td></tr>
<tr><td>IPS7_WORD</td><td>2</td><td>WORD 16 Bit unsigned integer (ohne Vorzeichen)</td></tr>
<tr><td>IPS7_INT</td><td>3</td><td>INT 16 Bit signed (mit Vorzeichen)</td></tr>
<tr><td>IPS7_DWORD</td><td>4</td><td>DWORD 32 Bit unsigned integer (ohne Vorzeichen)</td></tr>
<tr><td>IPS7_DINT</td><td>5</td><td>long 32 Bit signed integer (mit Vorzeichen)</td></tr>
<tr><td>IPS7_REAL</td><td>6</td><td>S7Real</td></tr>
<tr><td>IPS7_TIMER</td><td>7</td><td>Timer in der S7</td></tr>
<tr><td>IPS7_COUNTER</td><td>8</td><td>Zähler in der S7</td></tr>
<tr><td>IPS7_LINT</td><td>9</td><td>INT 64 Bit signed integer (mit Vorzeichen)</td></tr>
<tr><td>IPS7_ULINT</td><td>10</td><td>UINT 64 Bit unsigned integer (ohne Vorzeichen)</td></tr>
</table></td></tr>
<tr><td>DBNr</td><td>32-Bit unsigned</td><td>Datenbausteinnummer, diese wird nur beim Typ 'D' verwendet. Ansonsten steht dort der Wert</td></tr>
<tr><td>Cnt</td><td>32-Bit unsigned</td><td>Anzahl der Datenelemente, die gelesen/geschrieben werden sollen</td></tr>
<tr><td>Start</td><td>32-Bit unsigned</td><td>Startbyte in der SPS</td></tr>
<tr><td>StartBit</td><td>32-bit unsigned</td><td>Die Nummer des ersten Bits in der SPS, Werte (0 – 7), wird nur bei Bit-Zugriffen verwendet, ansonsten 0</td></tr>
<tr><td>PCDataType</td><td>32-Bit unsigned</td><td>Datentyp im PC
<table>
<tr><th>Name</th><th>Wert</th><th>Datentyp in der SPS</th></tr>
<tr><td>PC_BYTE</td><td>0</td><td>Byte (8 Bit)</td></tr>
<tr><td>PC_WORD16</td><td>1</td><td>16 Bit unsigned integer</td></tr>
<tr><td>PC_INT16</td><td>2</td><td>16 Bit signed integer</td></tr>
<tr><td>PC_WORD32</td><td>3</td><td>32 Bit unsigned integer</td></tr>
<tr><td>PC_INT32</td><td>4</td><td>32 Bit signed integer</td></tr>
<tr><td>PC_FLOAT</td><td>5</td><td>32 Bit Fließpunktzahl im PC (float)</td></tr>
<tr><td>PC_DOUBLE</td><td>6</td><td>64 Bit Fließpunktzahl im PC (double)</td></tr>
<tr><td>PC_WORD64</td><td>7</td><td>64 Bit unsigned integer</td></tr>
<tr><td>PC_INT64</td><td>8</td><td>64 Bit signed integer</td></tr>
</table></td></tr>
</table>

Result	32-Bit unsigned	Ergebnis für diesen Auftrag. Die einzelnen Werte finden Sie weiter unten bei der Beschreibung: Return-Wert für Read/Writefunktionen
UserData_0	32-Bit unsigned	Dieser Eintrag kann vom Aufrufer verwendet werden, um eigene Informationen aufzurufen. Der Wert wird vom Treiber weder ausgewertet noch verändert. Damit könnten z.B. Zusatzinformationen über die Eigenschaften der Variable im PC abgelegt werden
UserData_1	32-Bit unsigned	
Data	32-Bit Pointer	Der eigentliche Zeiger auf den Speicherbereich im PC für diesen Auftrag
pUserData	32-Bit Pointer	wie UserData_0 und UserData_1, jedoch als Pointer

IPS7RdMulti Beispiel C/C++

```
// kleine Funktion zur Initialiesierung eines einzelnen Requests
void InitRq (IPS7_RQ_MULTI *pRq, long DataArea, long DataType, long PcDataArea,
             long DBNr, long Start, long StartBit, long Cnt, void *pData)
{
  pRq->DataArea = DataArea;
  pRq->DataType = DataType;
  pRq->DBNr = DBNr;
  pRq->Cnt = Cnt;
  pRq->Start = Start;
  pRq->StartBit = StartBit;
  pRq->PcDataType = PcDataArea;
  pRq->Data = pData;
}

void DemoRdMulti (int Ref)
{
   int EBits[64];
   BYTE EBytes[64];

   WORD MWords[32];
   WORD DB10Words [150];

   double DB10WordsAsDouble [150];
   float DB20RealAsFloat [60];

   LONG32 TimerAsInt [10];
   int Cnt = 10;

   int Res;
   IPS7_RQ_MULTI Rq[10] ; // Max. 10 Aufträge;

   memset (Rq, , sizeof (Rq));
   Cnt = ;

   // lese ab E 4.0 32 Bit
   InitRq (&Rq[Cnt++], 'E', IPS7_BIT, PC_BYTE, , 4, , 32, EBits);

   // lese ab EB0 20 Byte und lege diese ab [20] ab
   InitRq (&Rq[Cnt++], 'E', IPS7_BYTE, PC_BYTE, , , , 20, &EBytes[20]);

   //lese ab MB 20 10 Worte
   InitRq (&Rq[Cnt++], 'M', IPS7_WORD, PC_WORD16, , 20, , 1, MWords);

   //lese DB10 ab Datenbyte 0 150 Worte
   InitRq (&Rq[Cnt++], 'D', IPS7_WORD, PC_WORD16, 10, , , 150, DB10Words);
```

```
    //lese DB20 ab Datenbyte 0 150 Worte lege diese aber als double im PC ab
    InitRq (&Rq[Cnt++], 'D', IPS7_WORD, PC_DOUBLE, 10, , , 150, DB10WordsAsDouble);

    //lese DB20 ab Datenbyte 6 60 Realwerte und lege diese als Floatwerte im PC ab
    InitRq (&Rq[Cnt++], 'D', IPS7_REAL, PC_FLOAT, 10, 6, , 60, DB20RealAsFloat);

    //lese ab Timer 5 10 Timer und lege diese als int ab
    InitRq (&Rq[Cnt++], 'T', IPS7_TIMER, PC_WORD32, , 5, , 10, TimerAsInt); // T5 10 Timer
    Res = IPS7RdMulti(Ref, Rq, Cnt);

    for (int i = ; i < Cnt; i++)
    {
       if (Rq[i].Result == ) // alles in Ordnung
       {
           //.. Daten auswerten und verarbeiten
       }
       else
       {
            // Fehler auswerten
       }
    }
    }
}
```

Diagnose Buffer lesen

IPS7ReadDiagBuffer

Name	Name (PHP)	Beschreibung/Zweck
IPS7ReadDiagBuffer		Auslesen des Diagnosebuffers

Aufrufparameter

Nr	Datentyp	Datentyp (PHP)	Name	Funktion
1	32-Bit unsigned	long	Ref	Die Referenz der Verbindung, welche mit IPS7Open generiert wurde. Dient zur internen Identifikation der Verbindung
2	S7_EVENTENTRY Pointer	mixed	pEvent	Pointer auf die Struktur des Events, siehe oben. Daten werden größenmäßig aneinander gereit geschrieben (16-Bit Word, 8-Bit Priority ...)
3	32-Bit		MaxEvents	Anzahl der maximal auszulesenden Events
4	32-Bit		pRxEventCnt	Pointer auf die tatsächliche Anzahl ausgelesener Events

C/C++ Funktionsdeklaration

typedef struct {

```
BYTE Year;
BYTE Month;
BYTE Day;
BYTE Hour;
BYTE Minute;
BYTE Second;
BYTE Centisecond;
BYTE Millisecond;
```

} S7_TIMESTAMP;

typedef struct {

```
WORD EventId;
BYTE Priority;
BYTE OBNr;
BYTE Reserved[2];
BYTE Info1[2];
BYTE Info2[4];
S7_TIMESTAMP Time;
const char *Text;
```

} S7_EVENTENTRY;

extern long WINAPI IPS7ReadDiagBuffer(long Ref, S7_EVENTENTRY *pEvent, int MaxEvents, int *pRxEventCnt); </code>

Delphi / Pascal Funktionsdeklaration

```
FUNCTION
IPS7ReadDiagBuffer (Ref : LongInt; pEvent : Pointer; MaxEvents : Int;
  pRxEventCnt : Pointer;)  : LongInt;
  stdcall; external 'IPS7LNK.DLL';
```

Systeminformationen Lesen

IPS7GetPLCTime

Name	Name (PHP)	Beschreibung / Zweck
IPS7GetPLCTime	ips7_getplctime	Liest die Systemzeit der SPS

Aufrufparameter

Nr	Datentyp	Datentyp (PHP)	Name	Funktion
1	32-Bit unsigned	long	Ref	Die Referenz der Verbindung, welche mit IPS7Open generiert wurde. Dient zur internen Identifikation der Verbindung
2	32-Bit Pointer auf 32 Bit Wert	long	Year	Pointer auf den 32 Bit Wert für das Jahr (nach lesen steht dort z.B. 2012)
2	32-Bit Pointer auf 32 Bit Wert	long	Month	Pointer auf den 32 Bit Wert für das Monat (1-12)
2	32-Bit Pointer auf 32 Bit Wert	long	Day	Pointer auf den 32 Bit Wert für den Tag (1 -31)

Nr	Datentyp	Datentyp (PHP)	Name	Funktion
2	32-Bit Pointer auf 32 Bit Wert	long	Hour	Pointer auf den 32 Bit Wert für die Stunde (0 – 23)
2	32-Bit Pointer auf 32 Bit Wert	long	Min	Pointer auf den 32 Bit Wert für die Minuten (0 – 59)
2	32-Bit Pointer auf 32 Bit Wert	long	Sec	Pointer auf den 32 Bit Wert für die Sekunden (0-59)
2	32-Bit Pointer auf 32 Bit Wert	long	Ms	Pointer auf den 32 Bit Wert für die Millisekunden (0-999)

C/C++ Funktionsdeklaration

```
extern long WINAPI
IPS7GetPLCTime (long Ref, long *pYear, long *pMonth, long *pDay,
                long *pHour, long *pMin, long *pSec, long *pMs); // 1.54
```

Delphi / Pascal Funktionsdeklaration

```
{ -- Version 1.54 --}
FUNCTION
IPS7GetPLCTime (Ref : LongInt; pYear : PLongInt; pMonth : PLongInt; pDay : PLongInt;
                pHour : PLongInt; pMin : PLongInt; pSec : PLongInt; pMs : PLongInt) :
LongInt;
stdcall; external 'IPS7LNK.DLL';
```

Es gelten dieselben Return-Werte wie für die Lese/Schreibfunktionen!

IPS7SetPLCTime

Name	Name (PHP)	Beschreibung / Zweck
IPS7SetPLCTime	ips7_setplctime	Setzte die Systemzeit der SPS

Aufrufparameter

Nr	Datentyp	Datentyp (PHP)	Name	Funktion
1	32-Bit Wert	long	Ref	Die Referenz der Verbindung, welche mit IPS7Open generiert wurde. Dient zur internen Identifikation der Verbindung
2	32-Bit Wert	long	Year	32 Bit Wert für das Jahr (1999 – 21xx)
2	32-Bit Wert	long	Month	32 Bit Wert für das Monat (1-12)
2	32-Bit Wert	long	Day	32 Bit Wert für den Tag (1 -31)
2	32-Bit Wert	long	Hour	32 Bit Wert für die Stunde (0 – 23)
2	32-Bit Wert	long	Min	32 Bit Wert für die Minuten (0 – 59)
2	32-Bit Wert	long	Sec	32 Bit Wert für die Sekunden (0-59)
2	32-Bit Wert	long	Ms	32 Bit Wert für die Millisekunden (0-999)

C/C++ Funktionsdeklaration

```
long WINAPI
IPS7SetPLCTime (long Ref, long Year, long Month, long Day,
  long Hour, long Min, long Sec, long Ms); // 1.54
```

Delphi / Pascal Funktionsdeklaration

```
{ -- Version 1.54 --}
FUNCTION
IPS7SetPLCTime (Ref : LongInt; Year : LongInt; Month : LongInt; Day : LongInt;
  Hour : LongInt; Min : LongInt; Sec : LongInt; Ms : LongInt) : LongInt;
stdcall; external 'IPS7LNK.DLL';
```

Es gelten dieselben Return-Werte wie für die Lese/Schreibfunktionen!

IPS7GetPLCName

IPS7GetModuleName

IPS7GetPlantIdName

IPS7GetCopyrightEntry

IPS7GetModuleSNr

IPS7GetModuleTypeName

IPS7GetMMCSNr

IPS7GetLocationDesignation

IPS7GetOEMId

In der SPS stehen verschiedene Systeminformationen zur Verfügung. Die meisten jedoch erst ab FW 2.2 der S7. Folgende Informationen können in einem String empfangen werden.

Name	Name (PHP)	Beschreibung / Zweck
IPS7GetPLCName	ips7_getplcname	Liest den Namen der SPS
IPS7GetModuleName	ips7_getmodulename	Liest den Namen der Baugruppe (z.B CPU 315-2 DP/PN - Test)
IPS7GetPlantIdName	ips7_getplantidname	Liest den Anlagenamen der SPS
IPS7GetCopyrightEntry	ips7_getcopyrightentry	Liest den Copyright Eintrag der SPS
IPS7GetModuleSNr	ips7_getmodulesnr	Liest die Seriennummer der SPS
IPS7GetModuleTypeName	ips7_getmoduletypename	Liest den Baugruppentyp der SPS (z.B CPU 315-2 DP/PN)
IPS7GetMMCSNr	ips7_getmmcsnr	Liest die Seriennummer der MMC
IPS7GetLocationDesignation	ips7_getlocationdesignation	Liest die Ortsbeschreibung in der SPS
IPS7GetOEMId	ips7_getoemid	Liest die OEM-ID der CPU

Aufrufparameter

Nr	Datentyp	Datentyp (PHP)	Name	Funktion
1	32-Bit unsigned	long	Ref	Die Referenz der Verbindung, welche mit IPS7Open generiert wurde. Dient zur internen Identifikation der Verbindung
2	32-Bit Pointer auf C-String	string	strInfo	Pointer auf den String, der die Info erhalten soll. Muss mindestens 64 Zeichen aufnehmen können

C/C++ Funktionsdeklaration

```
extern long WINAPI
IPS7GetPLCName(long Ref, char *Str);

extern long WINAPI
IPS7GetModuleName(long Ref, char *Str); // 1.55

extern long WINAPI
IPS7GetPlantIdName(long Ref, char *Str); // 1.55

extern long WINAPI
IPS7GetCopyrightEntry(long Ref, char *Str); // 1.55

extern long WINAPI
IPS7GetModuleSNr(long Ref, char *Str); // 1.55

extern long WINAPI
IPS7GetModuleTypeName(long Ref, char *Str); // 1.55

extern long WINAPI
IPS7GetMMCSNr(long Ref, char *Str); // 1.55

extern long WINAPI
IPS7GetOEMId(long Ref, char *Str); // 1.55

extern long WINAPI
IPS7GetLocationDesignation (long Ref, char *Str); // 1.55
```

Delphi / Pascal Funktionsdeklaration

```
{ -- Version 1.55--}
FUNCTION
IPS7GetPLCName(Ref : LongInt; Str : PAnsiChar) : LongInt;
  stdcall; external 'IPS7LNK.DLL';

{ -- Version 1.55 --}
FUNCTION
IPS7GetModuleName(Ref : LongInt; Str : PAnsiChar) : LongInt;
  stdcall; external 'IPS7LNK.DLL';

{ -- Version 1.55 --}
FUNCTION
IPS7GetPlantIdName(Ref : LongInt; Str : PAnsiChar) : LongInt;
  stdcall; external 'IPS7LNK.DLL';

{ -- Version 1.55 --}
FUNCTION
IPS7GetCopyrightEntry(Ref : LongInt; Str : PAnsiChar) : LongInt;
  stdcall; external 'IPS7LNK.DLL';

{ -- Version 1.55 --}
IPS7GetModuleSNr(Ref : LongInt; Str : PAnsiChar) : LongInt;
  stdcall; external 'IPS7LNK.DLL';

{ -- Version 1.55 --}
FUNCTION
IPS7GetModuleTypeName(Ref : LongInt; Str : PAnsiChar) : LongInt;
  stdcall; external 'IPS7LNK.DLL';

{ -- Version 1.55 --}
FUNCTION
IPS7GetMMCSNr(Ref : LongInt; Str : PAnsiCHar) : LongInt;
  stdcall; external 'IPS7LNK.DLL';

{ -- Version 1.55 --}
FUNCTION
IPS7GetOEMId(Ref : LongInt; Str : PAnsiChar) : LongInt;
  stdcall; external 'IPS7LNK.DLL';

{ -- Version 1.55 --}
FUNCTION
IPS7GetLocationDesignation (Ref : LongInt; Str : PAnsiChar) : LongInt;
  stdcall; external 'IPS7LNK.DLL';
```

Es gelten die selben Return-Werte wie für die Lese/Schreibfunktionen!

Datenbausteinlänge auslesen

IPS7GetDBLen

Ab Version 1.78

Name	Name (PHP)	Beschreibung / Zweck
IPS7GetDBLen		Liest die Datenbausteinlänge der übergebenen Datenbausteinnummer aus.

Aufrufparameter

Nr	Datentyp	Datentyp (PHP)	Name	Funktion
1	32-Bit unsigned	long	Ref	Die Referenz der Verbindung, welche mit IPS7Open generiert wurde. Dient zur internen Identifikation der Verbindung
2	32-Bit unsingend	long	DBNr	gewünschte Datenbausteinnummer

C/C++ Funktionsdeklaration

```
extern long WINAPI
IPS7GetDBLen(long Ref, unsigned long DBNr);
```

Delphi / Pascal Funktionsdeklaration

```
FUNCTION
IPS7GetDBLen(Ref : LongInt, DBNr : LongWord) : LongInt;
  stdcall; external 'IPS7LNK.DLL';
```

Return-Werte Read/Write

Wert	Fehlerbeschreibung	Bedeutung/Reaktion
> 0	ausgelesene Datenbausteinlänge	
-35	Datenbaustein existiert nicht	Datenbausteinnummer prüfen
-36	Funktion wird nicht von der SPS unterstützt	
Es gelten die selben Return-Werte wie für die Lese/Schreibfunktionen!		

Programmbeispiele

C/C++

```
unsigned char ByteBuffer[512];
unsigned short int WordBuffer[512];

//Aufruf der Bytefunktion z.B. Lese DB 10, ab DW0, 10 Worte
IPS7RdW (Ref,'D',10, , 10, WordBuffer);

//Aufruf der Bytefunktion z.B. Lese MB 0 , 10 Bytes
IPS7RdB (Ref,'M' ,, , 10, ByteBuffer);
```

Nach erfolgreichem Aufruf gilt:

PC	=	SPS
WordBuffer[0]	=	DB10.DBW0
WordBuffer[1]	=	DB10.DBW2
WordBuffer[2]	=	DW10.DBW4
ByteBuffer[0]	=	MB 0
ByteBuffer[1]	=	MB 1

Delphi

```
ByteBuffer array [..511] of Byte;
WordBuffer array [..511] of Word;

//Aufruf der Wortfunktion z.B. Lese DB 10, ab DW0, 10 Worte
IPS7RdW (Ref, LongWord ('D'), 10, , 10, @WordBuffer[]);

//Aufruf der Bytefunktion z.B. Lese MB 0 , 10 Bytes
IPS7RdB (Ref,'M' ,, , 10, @ByteBuffer[]);
```

VB

```
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
IPS7RdW (Ref, 68, 10, 0, 10, WordBuffer(0))

//Aufruf der Bytefunktion z.B. Lese MB 0 , 10 Bytes
IPS7RdB (Ref, 77, 0, 0, 10, ByteBuffer(0));
```

Nach erfolgreichem Aufruf gilt:

PC	**=**	**SPS**
WordBuffer[0]	=	DB10.DBW0
WordBuffer[1]	=	DB10.DBW2
WordBuffer[2]	=	DW10.DBW4
ByteBuffer[0]	=	MB 0
ByteBuffer[1]	=	MB 1

Socket Fehler Liste

Win Socket Fehlerliste

Release Notes

Release Notes

IP-S7-LINK Release Notes

Version 1.78.00 04.10.2018

- IPS7GetDBLen - auslesen der Datenbausteinlänge

Version 1.77.00 12.12.17

- Logo 8.1 konnte nicht verbinden
- für mehrer Verbindungen IPS7OpenExWithTSAP eingeführt

Version 1.76.00 20.5.17

- Paket für „S7200 und Logo“ läuft nun auch für Logo 8.1

Version 1.75.00 11.10.16

- Paket für nur S71200 und Logo erstellt

Version 1.74.00 6.7.15

- Windows CE konnte keine Timeoutzeiten auf Socketebene
    - Timeoutüberwachung implementiert
- bei Connect-Timeout wird nun SocketError „host not reachable“ gemeldet
- Prüfung, ob Socketverbindung noch besteht, verbessert

Version 1.73 vom 16.6.15

- Lint und ULInt Datentypen implementiert

Version 1.72.93 vom 25.3.15

- LinkTo - Anpassung

Version 1.71.92 vom 8.12.14

- S7-Logo IPS7RdReal

Version 1.71.92 vom 8.12.14

- S7-Logo IPS7RdReal

Version 1.70.91. vom 2.7.14

- Peripheriezugriff implementiert

Version 1.69.90 vom 04.06.14

- bei Multiread interne Speicherverwaltung optimiert
- für Bit/Byte grundsätzlich bytealigned, ansonsten 4-Byte aligned
- beim erstmaligen MultiRead-Zugriff mit grossen Paketen kam es zu Speicherkonflikten
- der Effekt bestand seit V 1.67.88

Version 1.68.89 vom 09.05.14

- IPS7ReadDiagBuffer implementiert

Version 1.67.88 vom 29.04.14

- Multiread bei Bit, wurden mit Wert 0 gelesen

Version 1.66.87 vom 04.04.14

- Multireadfehler(2) bei String, wenn der Stringeintrag fragmentiert war
- Multiread Speicher optimiert

Version 1.65.86 vom 31.03.14

- Multireadfehler bei String, wenn der Stringeintrag durch 2 Aufträge fragmentiert war

Version 1.64 vom 04.12.13

- Multiread ungünstige Kombination von Readaufträgen erzeugte Fehler

Version 1.63

- Multiread der Result wurde in den Einzelaufträgen nicht sauber weitergegeben

Version 1.62.83 vom 09.09.13

- Kommunikation mit Logo-SPS bei nur Logo/S7200 angepasst

Version 1.62.82 vom 08.07.2013

- IPS7RdStr Fehler bei Multiread beseitigt

Version 1.61.80 vom 19.06.2013

- IPS7RdStr, IPS7WrStr implementiert
- String bei IPS7RdMulti implementiert

Version 1.60.78 vom 02.04.2013

- PDU-Size optimiert, Zugriff erfolgt mit maximaler Grösse
- Zugriff auf Logo SPS eingefügt→ Accessmode 3 bei IPS7OpenEx

Version 1.58 vom 17.07.12

- betrifft nur die ARM-Version
- beim Lesen von Realwerten wurden wg. ALIGNMENT falsche Werte gelesen

Version 1.57

- IPS7RdW
- Offset bei Start != 0 wurde mit Faktor 4 statt 2 berechnet

Version 1.56 vom 21.05.12

- Lite Version für Privatanwender generiert

Version 1.55 vom 12.04.12
neue Funktionen:

- IPS7GetPLCName(long Ref, char *Str);
- IPS7GetModuleName(long Ref, char *Str);
- IPS7GetPlantIdName(long Ref, char *Str);
- IPS7GetCopyrightEntry(long Ref, char *Str);
- IPS7GetModuleSNr(long Ref, char *Str);
- IPS7GetModuleTypeName(long Ref, char *Str);
- IPS7GetMMCSNr(long Ref, char *Str);
- IPS7GetOEMId(long Ref, char *Str);
- IPS7GetLocationDesignation (long Ref, char *Str); LPCSTR in LPCTSTR geändert wg. WINCE

Version 1.54 vom 05.04.12

- LPCSTR in LPCTSTR geändert wg. WINCE
- S7-200 / Logo -Zugriff eingebaut, bei Logo immer TSAP 02.00 verwenden
- purer S7-200 Logo Lizenz eingeführt, kann mit S7-LAN-LINK kombiniert werden
- bei S7-LAN-LINK wurde bei nichtvorhanden sein des S7-LAN oder S5-LAN Socket nicht geschlossen
- Setzen / lesen der SPS-Uhrzeit implementiert
- IPS7GetPLCTime (long Ref, long *pYear, long *pMonth, long *pDay, long *pHour, long *pMin, long *pSec, long *pMs);
- IPS7SetPLCTime (long Ref, long Year, long Month, long Day, long Hour, long Min, long Sec, long Ms);

Version 1.53 vom 22.02.12

- RdMultiSimplex, wenn z.B. während Debug die IP-Verbindung zurückgesetzt wurde, wurde evtl. ein nicht definierter positiver Fehler zurückgemeldet

Version 1.52 vom 30.01.12

- IPS7RdBit, Bit-Lesen ab 1.51 es wurde immer Bit 0 gelesen
- Im Source Änderungen für IAR-Compiler eingefügt

Version 1.51 vom 30.11.11
bei 1.50 MultiRead Fehler (PDUSize)

Version 1.50 vom 25.11.11

- MultiRead mit Counter war nicht korrekt
- Anpassungen für Embedded Systeme gemacht
- Leseroutinen optimiert

Version 1.49 vom 11/11

- Bei Linux wird für Critial Sections nun pthreads verwendet

Version 1.48 29.11.10

- für S7-LAN-LINK auch S5-LAN integriert
- unter Linux für Erstellung shared Libs mit Compileroption -„fPIC“ übersetzt

Version 1.47 vom 24.09.10

- Unterstützung für ARM-Processoren implementiert (Alignment-Trap behoben)

Version 1.46 vom 08.09.10

- Bit-Zugriffe mit MultiRead Funktion
- Beim Lesen von Bit's mit Start-Bitadresse > 0 trat bei einigen SPS der Fehler auf
- Datenbereich nicht vorhanden (Bitadresse wurde bei Bytezugriff mit übergeben)

Version 1.45 vom 07.09.10

- PDU-Size für CPU 400 etc. optimiert

Version 1.44 vom 18.08.10

- Unterstützung / Erkennung S5-LAN mit S7-TCP/IP
- Bei MultiRead-Zugriff auf S5-LAN wird die Real-Konvertierung nicht im S5-LAN sondern im Treiber vorgenommen. Der Treiber muss jedoch wissen, ob ein S5-LAN angeschlossen ist.
- ab S5-LAN ++ V 1.20 kann dies der Treiber automatisch erkennen.
- bei Modulen < 1.20, bei der Funktion IPS7OpenEx als AccessMode „20“ angeben.

Version 1.43 vom 22.07.10

- MultiReadzugriff: Demoversion eingebaut
- sizeof - Vergleiche berichtigt

Version 1.42 vom 14.07.10

- MultiReadzugriff: Trat vor ausführen des ersten Read-Auftrags ein Fehler auf (z.B Timeout etc.) Wurde im Auftrag als Result -88 /Auftrag nicht bearbeitet gesetzt nun wird dort der tatsächliche Fehlerwert angegeben.

- Linux: war kein connect möglich, so wurde Fehler -5 (genereller Fehler) gesetzt nun wird Socketfehler gesetzt, so kann die eigentliche Ursache mit errno bwz. strerrno() ermittelt werden

Version 1.41 vom 13.07.10

- MultiReadzugriff: lesen von Ausgängen wurde nicht unterstützt

Version 1.40 vom 07.07.10

- MultiReadzugriff: bei Blöcken > 220 Byte kam es zu Überschreibungen
- .Net MultiReadzugriff: Da der da Garbage Collection die Variablen unvorhergesehen verschieben kann, musste die Zugriffsweise überarbeitet werden.
- Programme in c# oder VB.Net sollten die Funktiion RdMultiBuffered verwenden. Näheres in der .chm Datei!

Version 1.39 vom 17.06.10

- MultiReadzugriff: neue Funktion „ IPS7RdMultiCalcPacketCnt “ Liefert die Anzahl der benötigten Pakete zum lesen aller angegebenen MultiRead-Aufträge

Version 1.38 vom 24.05.10

- MultiReadzugriff: Int16 und Int32 (signed) wurde bei Übergabe von PC_INT32, und PC:DWORD in einen unsigned konvertiert nun erfolgt die Konvertiereung richtig in einen signed-Wert

Version 1.37 vom 18.05.10

- .Net-Interface: für Multiread-Zugriff, Int16 und Int32 (signed) Zugriff implementiert.
- .Net-Interface: für Multiread-Zugriff, bei verwenden von Arrays, wird die Grösse geprüft, ist ein Array zu klein, wird der Fehler -20 erzeugt.
- MultiReadzugriff: Umwandlung von Bit in DWORD oder Real führte zur Schutzverletzung

Version 1.36 vom 03.05.10

- IPS7RdMulti, Kopierfehler es kam zur Schutzverletzung
- Demo für Delphi überarbeitet, Outfit wie C++/C#/VB.net

Version 1.35 vom 14.04.10
Neu Funktionen:

- IPS7Connect – führt explezite IP-Verbindung aus
- IPS7GetConnectStatus – prüft den IP-Verbindungsstatus
- IPS7SetKeepAlive – setzt individuelle KeepAlive-Zeiten
- IPS7RdMulti – liest verschiedene Datenbereiche an einem Stück aus der SPS

Version 1.34 vom 02/10

- Zwischenversion

Version 1.33 vom 02.02.10

- beim Lesen der Timer konnte es bei laufendem Timer zu falschen Ergebnissen kommen. Die Basis wurde falsch berechnet.

Version 1.32 vom 27.08.09

- Lesen / Schreiben der Real / Float-Werte mit S7-Code so ist auch ein Betrieb an S5-Lan++ mit Realwerten möglich.

Version 1.31 vom 20.08.09

- in den .Net-Assemblies Strong-Names eingefügt (20.8.09)

Version 1.30 vom 17.07.09

- V 1.29 auf Linux portiert, S7-LAN-Link hat jetzt selbes Interface wie IP-S7-LINK, läuft aber nur mit S7-LAN,
- so ist der Umstieg auf IP-S7-LINK für den Anwender einfacher. (17.7.09)

Version 1.29 vom 20.04.09

- Assembly Interface zu .Net Rd Methode mit 32 Bit Integer hat Bit gelesen, nun 32 Bit

Version 1.28 vom 09.02.09

- ips7lnk.lib verweiste auf s7lanlnk.dll, dadurch kam es zu Linkfehlern beim VC++ Compiler, bzw. zur Medlung, S7lanknk.dll wird nicht gefunden

Version 1.27 vom 26.08.08

- Zusätzliche Prüfungen für gültigen Speicher eingefügt. Behandlung der Critical Sections verbessert

Version 1.26 vom 12.08.08

- IPS7WrBit war nicht exportiert

Version 1.25 vom 16.07.08
Beim Aufruf von IPS7Open mit mehreren Threads gleichzeitig kam es gelegentlich zu Traps. Problem mit CriticalSection behoben

Version 1.24 vom 10.07.08

- .Net und PHP Eingepflegt

Version 1.23 vom 02.06.06

- Zugriff über Routing per SubnetID eingefügt

Version 1.22 vom 14.10.05

- Betrieb mit Slot-SPS und Soft-SPS ging nicht (FAST-ACK wurde nicht korrekt verarbeitet)
- Probleme beim Empfang von fragmentierten Daten

Version 1.21 vom 04.08.05

- Betrieb mit CP 243 (S7 200) implementiert

Version 1.20 vom 08.07.05

- Sonderversion für Fachhochschule begrenzt auf DB1 und DB2 erzeugt

Version 1.19 vom 20.05.05

- Die Änderung 1.18 war nur für IPS7RdPlcW und IPS7WrPlcW bei Blöcken > 111 Worte gedacht. Mit 1.18.funktionerten IPS7RdW und IPS7WrW in diesen Bereichen nicht mehr

Version 1.18 vom 17.05.05

- falsche Berechnung der StartAdresse bei IPS7RdW und IPS7WrW bei Blöcken > 111 Worte

- Die Startadresse der Folgeblöcke wurde falsch bestimmt

Version 1.17 vom 04.03.05

- Neue Funktion eingefügt IPOpenPG, damit ist es möglich, eine Verbindung über den PG-Kanal der SPS herzustellen
- Sinnvoll, wenn keine OP-Kanäle mehr frei sind.
- Zum worteweise Lesen und Schreiben mit ungeraden Startadressen Funktion IPS7RdPlcW und S7WrPlcW

Version 1.16 vom 01.12.04

- Wir hatten grundsätzlich den PG.Kanal verwendet, ab nun wird der HMI/OP-Kanal verwendet

Version 1.15 vom 11.11.04

- Timeoutüberwachung zum Empfang des gesamten Blocks eingefügt. U.u. kam es zu Problemen mit Berthel SPS

Version 1.14 vom 01.08.04

- Maximale Anzahl geöffneter Kanäle auf 256 erhöht

Version 1.13 vom 26.07.04

- Maximale Anzahl geöffneter Kanäle auf 128 erhöht

Version 1.12 vom 07.07.04

- Um höhere Performance zu erreichen wurde der Nagle-Algorithmus ausgeschaltet. D.h. TCP_NODELAY wurd auf 1 gesetzt.

Version 1.11 vom 13.05.04

- Beim 'close' der Sockets hat Windows den gewünschten Port teils erst nach 20 Minuten wieder freigegeben.
- Dabei kam es zum Effekt, daß erst nach Neustart der SPS oder des PC eine neue Verbindung zu SPS möglich war

Version 1.10 vom 13.05.04

- Maximale Anzahl geöffneter Kanäle auf 64 erhöht

Version 1.09 vom 19.03.04

- Schreiben in Eingänge erlaubt

Version 1.08 vom 23.01.02

- Beim Verbindungsversuch mit Teilnehmern, die nicht im Netz waren, wurden Handles im System belegt und nicht mehr freigegeben.
- Problem behoben!

Version 1.07 vom 12.12.01

- DLL-Aufruf von mehreren Applikationen aus gab Fehler nun beseitigt

Version 1.06 vom 19.09.01

- Timer / Zähler Funktionen eingefügt

- Doppelwortfunktionen eingefügt
- Realzahlenzugriffe (Fließpunktarithmetik) eingefügt

Version 1.05 vom 12.07.01

- Bit-Lese und Schreibfunktionen eingefügt

Version 1.04 vom 19.06.01

- Byteweise Lesen und Schreiben bei DB eingefügt

Version 1.03 vom 17.05.01

- Beim Lesen von Blöcken, die durch 222 teilbar sind
- z.B. 444 Byte oder 222 Worte
- z.B. 666 Byte oder 333 Worte ..
- wurde der letzten 222 Byte nicht gelesen, es erfolgte keine Fehlermeldung
- Beim Schreiben von Blöcken, die durch 212 teilbar sind
- z.B. 424 Byte oder 212 Worte
- z.B. 636 Byte oder 318 Worte ..
- wurde der letzten 212 Byte nicht geschrieben, es erfolgte keine Fehlermeldung

Version 1.02 vom 29.01.01

- Fehlernummer -6 eingefügt
- Funktion IPS7GetSockErr eingefügt (siehe Dokumentation)

Version 1.01 vom 21.12.00

- Lesen von Bausteinbereichen, die nicht exisitierten wurden als OK bestätigt, obwohl der angegebene Bereich nicht existierte
- Dies trat auf bei DB > 111 Worte

Version 1.00 vom 14.12.00

Inhaltsverzeichnis