

# IP-S7-LINK

## RFC 1006

### für

# PC – SIMATIC S7

S7 200 – 1200 – 300 – 400

MS-Windows 2000/NT/XP/Vista/7, Linux

C/C++, c#, VB, VB.net, Delphi, Excel, Access, PHP und andere



Version 1.36 (10.5.2010)

**Voraussetzungen:**

<b>Betriebssysteme:</b>	MS-Windows 95 ,98, NT, 2000, XP,Vista, Windows 7 Linux
<b>Programmiersprachen</b>	C, C++, Delphi, VisualBasic, C-Sharp, VB.Net, Access, Excel, PHP
<b>Hardware</b>	PC mit installiertem TCP/IP-Protokoll und Netzwerkkarte
<b>SPS</b>	Simatic S7 200/300/400 mit CP 243, CP343-1, CP-443-1, sowie LeanCP außerdem ProfiNet - CPU und S7-LAN

**Lieferumfang:**

Folgende Dateien können sich im Lieferumfang befinden:

<b>Hauptverzeichnis</b>	
Ips7lnk.pdf	diese Datei – Dokumentation
Version.htm	Datei über Fehlerbeseitigung

<b>Verzeichnis 'CPP'</b>	<b>Dateien für Visual CPP C++</b>
IPS7LNK.H	Header-Datei für C / C++
IPS7LNK.DLL	Treiber DLL
IPS7LNK.LIB	Lib-Datei zum Linken mit C++
IPS7DEMO.DSP	Projektdatei für Visual C++ V 6.00
IPS7DEMO.CPP	Beispielprogramm in 'C++' einer Konsolenapplikation
IPS7DEMO.EXE	EXE-Datei der CPP-Demo

<b>Verzeichnis 'Delphi'</b>	<b>Dateien für Delphi</b>
IPS7LNK.PAS	Delphi-Header TPU im Quellcode
IPS7LNK.DLL	Treiber DLL
IPS7DEMO.exe	EXE-Datei der Delphidemo
IPS7DEMO.cfg	Delphi - Projektdateien
IPS7DEMO.dof	
IPS7DEMO.dpr	
IPS7DEMO.res	
IPS7LNK.dcu	
main.dcu	
main.dfm	
main.pas	
OEM.BMP	

<b>Verzeichnis ' VisualBasic'</b>	<b>Dateien für Visual Basic</b>
IPS7LNK.DLL	Treiber DLL, <b>Achtung: Für Visual Basic und Excel diese Datei ins Windowsverzeichnis kopieren!</b>
IPS7LNK.BAS	Header / Moduldatei für Visual Basic
IPS7DEMO.EXE	EXE-Datei der VB-Demo
IPS7DEMO.FRM IPS7DEMO.FRX IPS7DEMO.VBP IPS7DEMO.VBW	Visual Basic Projektdateien

<b>Verzeichnis Excel</b>	<b>Dateien für Excel</b>
IPS7LNK.DLL	Treiber DLL, <b>Achtung: Für Visual Basic und Excel diese Datei ins Windowsverzeichnis kopieren!</b>
IPS7LNK.BAS	Header/Moduldatei für VisualBasic
IPS7DEMO.XLS	Excel-Datei mit Makro für Demo

<b>Verzeichnis ' PHP'</b>	<b>Dateien für PHP</b>
ips7lnk_php.so	Modul für die PHP-Erweiterung, dieses Modul in das Extensionverzeichnis der betreffenden PHP-Installation kopieren

<b>Verzeichnis ' DotNet'</b>	<b>Dateien für .Net</b>
DemoCSharp	Verzeichnis Demoprogramm C#
DemoVB.Net	Verzeichnis Demoprogramm VB.Net
NetFiles	Verzeichnis für die Assemblies der verschiedenen Frameworks (2.0/3.0/3.5)
ips7lnk.chm	Hilfdatei / Programmieranleitung für .Net Assemblies

### **Installation:**

**Windows: Die DLL ins Verzeichnis des Programms oder ins Systemverzeichnis kopieren.**

**Linux: Die o-Datei zu Ihrem Programm linken .Net:**

Für .Net entnehmen Sie die Dokumentation der ips7lnk.chm-Datei. Für .Net (C# und VB.Net) wurden alle Funktionen in eine Klasse eingebettet. Es liegen Assemblies für Framework 2.0/3.0/3.5 bei. Diese Assemblies verwenden die ,ips7lnk.dll'. Sorgen Sie dafür, dass ,ips7lnk.dll' entweder im Programmverzeichnis Ihrer Applikation oder im Windowssystemverzeichnis zur Verfügung steht. Fügen Sie das gewünschte Assembly

ips7lnknet.dll einfach als Verweis in Ihre Applikation ein. Schon stehen sämtliche Funktionen zur Verfügung.

#### **PHP:**

#### **Die Extension ips7lnk\_php.so in das Extensionverzeichnis der jeweiligen 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');`

#### **Funktionsweise:**

IP-S7-LINK ist eine DLL für MS-Windows (95/98/2000/NT/XP/Vista) bzw. eine Library oder eine Erweiterung für Linux (C,C++,PHP), welche die Anbindung eines PC an Industrial Ethernet der SIMATIC S7 von Siemens ermöglicht. Mit einfachen Funktionen kann der Anwender schnell mit C, C++, Delphi, Visual Basic, Excel oder auch PHP auf die Daten der SPS'en im Netz zugreifen. Zur Kopplung wird nur die IP-Adresse des CP sowie der Steckplatz der CPU im SPS-Rack benötigt. Sofort kann auf Merker, Eingänge, Ausgänge und auch Datenbausteine der SPS lesend oder auch schreiben zugegriffen werden.

### Funktionsbeschreibung 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:

### Funktionen zur Initialisierung:

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

### Aufrufparameter:

<b>Nr</b>	<b>Datentyp</b>	<b>Datentyp (PHP)</b>	<b>Name</b>	<b>Funktion</b>
1	32-Bit Zeiger auf C-String	string	IPAdr	IP-Adresse der SPS im Format: xxx.xxx.xxx.xxx. Beispiel: "192.169.0.100"
2	32-Bit Wert 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 Wert unsigned	long	Slot	Nummer des Steckplatzes der CPU beginnend mit „1“, normalerweise „2“. Bei S7-200 egal

<b>Nr</b>	<b>Datentyp</b>	<b>Datentyp (PHP)</b>	<b>Name</b>	<b>Funktion</b>
4	32-Bit Wert unsigned	long	RxTimeout	Timeout in Millisekunden für Warten auf TCP/IP-Paket von der SPS 0 bedeutet Standardeinstellung = 500 ms
5	32-Bit Wert unsigned	long	TxTimeout	Timeout in Millisekunden für Senden eines TCP/IP-Paketes an die SPS 0 bedeutet Standardeinstellung = 500 ms
6	32-Bit Wert unsigned	long	ConTimeout	Timeout in Millisekunden für Warten auf Verbindungsaufbau mit SPS 0 bedeutet Standardeinstellung = 5000 ms ( 5sec.) muss bei Bedarf verlängert werden.

<i>Funktion</i>	<i>Funktion (PHP)</i>	<i>Beschreibung / Zweck</i>
<b>IPS7OpenEx</b>	<b>ips7_openex</b>	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. <b>Es können durch die Wahl der Parameter OP/PG S7200 oder auch Verbindungen über ein Subnetz hergestellt werden.</b>

**Aufrufparameter:**

<b>Nr</b>	<b>Datentyp</b>	<b>Datentyp (PHP)</b>	<b>Name</b>	<b>Funktion</b>
1	32-Bit Zeiger auf C-String	string	IPAdr	IP-Adresse der SPS im Format: xxx.xxx.xxx.xxx. Beispiel: 192.169.0.100
2	32-Bit Wert 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 Wert unsigned	long	Slot	Nummer des Steckplatzes der CPU beginnend mit „1“, normalerweise „2“. Bei S7-200 egal
4	32-Bit Wert unsigned	long	SubNet-Id	Subnetz-ID, wenn über ein Subnet zu zugegriffen werden soll. In der Step-S7 Software wird die Adresse z.B. so dargestellt: 0035 – 0001 geben Sie dann an 0x00350001 Wird nur bei AccessMode 10 oder 11 verwendet

<b>Nr</b>	<b>Datentyp</b>	<b>Datentyp (PHP)</b>	<b>Name</b>	<b>Funktion</b>
5	32-Bit Wert unsigned	long	DstMPIAdr	Ziel-MPI-Adresse, wenn über ein Subnetz die Verbindung aufgebaut werden soll. Siehe AccessMode 10 und 11!
6	32-Bit Wert 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 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.
7	32-Bit Wert unsigned	long	RxTimeout	Timeout in Millisekunden für Warten auf TCP/IP-Paket von der SPS 0 bedeutet Standardeinstellung = 500 ms
8	32-Bit Wert unsigned	long	TxTimeout	Timeout in Millisekunden für Senden eines TCP/IP-Paketes an die SPS 0 bedeutet Standardeinstellung = 500 ms
9	32-Bit Wert unsigned	long	ConTimeout	Timeout in Millisekunden für Warten auf Verbindungsaufbau mit SPS 0 bedeutet Standardeinstellung = 5000 ms ( 5sec.) muss bei Bedarf verlängert werden.

### **Rückgabewerte für die Open-Funktionen:**

Die Funktionen liefert einen 32-Bit Wert mit Vorzeichen als Rückgabewert mit folgender Bedeutung:

<b>Wert</b>	<b>Fehlerbeschreibung</b>	<b>Bedeutung</b>
>= 0	Alles OK	Die Rückgabe 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

<i>Funktion</i>	<i>Funktion</i>	<i>Beschreibung / Zweck</i>
<b>IPS7Connect</b>	<b>ips7_connect</b>	Führt einen expliziten Verbindungsaufbau zur SPS aus. Ab Version 1.35! Damit kann ohne Lese Schreibauftrag die Verbindung zur SPS hergestellt werden.

**Aufrufparameter:**

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

**Rückgabewert:**

Die Funktion IPS7Connect liefert einen 32-Bit Wert mit Vorzeichen als Rückgabewert mit folgender Bedeutung:

<b>Wert</b>	<b>Fehlerbeschreibung</b>
1	Verbindung wurde hergestellt
<= 0	Verbindung konnte nicht hergestellt werden. Die genaue Bedeutung finden Sie bei den Returnwerten für die Lese- / Schreibfunktionen

<i>Funktion</i>	<i>Funktion</i>	<i>Beschreibung / Zweck</i>
<b>IPS7GetConnect</b> <b>Status</b>	<b>ips7_getconnectst</b> <b>atus</b>	Prüft den TCP/IP Verbindungsstatus zur SPS.

**Aufrufparameter:**

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

**Rückgabewert:**

Die Funktion IPS7Connect liefert einen 32-Bit Wert mit Vorzeichen als Rückgabewert mit folgender Bedeutung:

<b>Wert</b>	<b>Fehlerbeschreibung</b>
1	Verbindung wurde hergestellt und besteht.
<= 0	Verbindung ist unterbrochen. Evtl. IPS7Connect, Lese oder Schreibfunktionen aufrufen.

<i>Funktion</i>	<i>Funktion</i>	<i>Beschreibung / Zweck</i>
<b>IPS7SetKeepAlive</b>	<b>ips7_setkeepalive</b>	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:**

<b>Nr</b>	<b>Datentyp</b>	<b>Datentyp (PHP)</b>	<b>Name</b>	<b>Funktion</b>
1	32-Bit Wert unsigned	long	Ref	Die Referenz der Verbindung, welche mit IPS7Open generiert wurde. Dient zur internen Identifikation der Verbindung
2	32-Bit Wert unsigned	long	AliveTime	Findet innerhalb der Zeit "AliveTime" (ms) kein Datenverkehr auf der TCP/IP-Verbindung statt, so wird ein KeepAlive-Telegramm 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-Telegramm. Dies wird einige male innerhalb der Zeit "AliveInterval" wiederholt (bei Windows 6 mal). War der Vorgang nicht erfolgreich, wird die Verbindung beendet.
3	32-Bit Wert 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.

**Rückgabewert:**

Die Funktion IPS7SetKeepAlive liefert einen 32-Bit Wert mit Vorzeichen als Rückgabewert mit folgender Bedeutung:

<b>Wert</b>	<b>Fehlerbeschreibung</b>
0	Setzen der Werte war erfolgreich.
< 0	Das setzen der KeepAlive-Zeit konnte nicht ausgeführt werden.

<i>Funktion</i>	<i>Funktion</i>	<i>Beschreibung / Zweck</i>
<b>IPS7Close</b>	<b>ips7_close</b>	Dient zur Deinitialisierung der Verbindung, Speicher wird freigegeben und die TCP/IP-Verbindung wird getrennt.

**Aufrufparameter:**

<b>Nr</b>	<b>Datentyp</b>	<b>Datentyp (PHP)</b>	<b>Name</b>	<b>Funktion</b>
1	32-Bit Wert unsigned	long	Ref	Die Referenz der Verbindung, welche mit IS7Open generiert wurde. Dient zur internen Identifikation der Verbindung

### **Rückgabewert:**

Die Funktionen liefert einen 32-Bit Wert mit Vorzeichen als Rückgabewert mit folgender Bedeutung:

<b>Wert</b>	<b>Fehlerbeschreibung</b>	<b>Bedeutung / Reaktion</b>
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.

### **Funktionen zum Lesen und Schreiben**

<i>Funktion</i>	<i>Funktion (PHP)</i>	<i>Beschreibung / Zweck</i>
<b>IPS7RdW</b>	<b>ips7_rdw</b>	wortweise lesen aus der SPS (E, A, M, DB )
<b>IPS7RdPlcW</b>	<b>ips7_rdplcw</b>	Wortweise lesen aus der SPS (E, A, M, DB) jedoch Startadresse nach SPS-Adressierung (ab 1.17) damit ein Zugriff auf ungerade Startadressen möglich
<b>IPS7RdB</b>	<b>ips7_rdb</b>	byteweise lesen aus der SPS (E, A, M, DB)
<b>IPS7WrW</b>	<b>ips7_wrw</b>	wortweise schreiben in die SPS (E, A, M, DB, Z)
<b>IPS7WrPlcW</b>	<b>ips7_wrplcw</b>	wortweise schreiben in die SPS (E, A, M, DB, Z) jedoch Startadresse nach SPS-Adressierung (ab 1.17) damit ein Zugriff auf ungerade Startadressen möglich
<b>IPS7WrB</b>	<b>ips7_wrb</b>	byteweise schreiben in die SPS ((E, A, M, DB, Z)
<b>IPS7RdDW</b>	<b>ips7_rdrw</b>	doppelwortweise lesen aus der SPS (E, A, M, DB, T)
<b>IPS7WrDW</b>	<b>ips7_wrdw</b>	doppelwortweise schreiben in die SPS (E, A, M, DB, T)
<b>IPS7RdReal</b>	<b>ips7_rdreal</b>	Real (Fließpunktzahl lesen (E, A, M, DB)
<b>IPS7WrReal</b>	<b>ips7_wrreal</b>	Real (Fließpunktzahl schreiben (E, A, M, DB)

**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. Näheres finden Sie weiter unten erklärt. Siehe dazu **Neu in Version 1.17!**

### **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.

**Aufrufparameter:****Die Lese- und Schreibfunktionen besitzen die selben Eingangsparameter:**

<b>Nr</b>	<b>Datentyp</b>	<b>Datentyp (PHP)</b>	<b>Name</b>	<b>Funktion</b>
1	32-Bit Wert unsigned	long	Ref	Die Referenz der Verbindung, welche mit IS7Open generiert wurde. Dient zur internen Identifikation der Verbindung
2	32-Bit Wert unsigned	long	Typ	<p>Die Auswahl des Speicherbereichs in der SPS (DB, Eingang, Ausgang, Merker), welcher bearbeitet werden soll:</p> <p>'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)</p> <p>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</p> <p>'Z' = 90 dez. steht für Zähler (nur mit Wortfunktionen möglich)</p> <p>Auch die Zähler sind in der SPS BCD-Codiert abgelegt. Die Zählerwerte reichen von 0 – 999.</p>
3	32-Bit Wert unsigned	32-Bit Wert unsigned	DBNr	Datenbausteinnummer, diese wird nur beim Typ 'D' verwendet. Ansonsten steht dort der Wert „0“
4	32-Bit Wert 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.
5	32-Bit Wert unsigned	32-Bit Wert unsigned	Anz	Anzahl der Einheiten (Byte, Worte, Doppelworte Real, oder Einheiten, z.B. Timer), die gelesen bzw. geschrieben werden sollen.

<b>Nr</b>	<b>Datentyp</b>	<b>Datentyp (PHP)</b>	<b>Name</b>	<b>Funktion</b>
6	32-Bit Pointer	mixed	Buffer	<p>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.</p> <p><b>Hinweis für PHP:</b> Bei PHP geben Sie als hier die Referenz einer Variablen an also: Bemerkung zu ips7_rdplcw ips7_rdw ips7_rddw ips7_rdreal Aufruf z.B.: &amp;Result = ips7_rdplcw (\$Ref, ord ("M"), 0, 6,5, &amp;\$W); Werden mehr als 1 Element gelesenen, 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).</p>

<b>Nr</b>	<b>Datentyp</b>	<b>Datentyp (PHP)</b>	<b>Name</b>	<b>Funktion</b>
7	-----	long (optional)	bSigned bei ipss7_rdplcw ips7_rdw ipss7_wrplcw ips7_wrdw  bLong bei ips7_rdb	<b>Hinweis für PHP:</b> <b>ipss7_rdplcw</b> <b>ips7_rdw</b> <b>ipss7_wrplcw</b> <b>ips7_wrdw</b> 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). Wird der Parameter übergeben gilt: 0 = ohne Vorzeichen (signed) 1 = mit Vorzeichen Außerdem stehen für eine nachträgliche Konvertierung einzelner Werte die Funktionen <b>ips7_i2w</b> und <b>ips7_w2i</b> zu Verfügung. Näherer finden Sie dort. <b>ips7_rdb</b> 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.

### **Funktionen zum Bit-Lesen, Setzen und Rücksetzen**

<i>Funktion</i>	<i>Funktion (PHP)</i>	<i>Beschreibung / Zweck</i>
<b>IPS7RdBit</b>	<b>ips7_rdbit</b>	ein Bit lesen SPS (E,A,M, DB )
<b>IPS7SetBit</b>	<b>ips7_setbit</b>	ein Bit setzen lesen aus der SPS (E,A,M,DB)
<b>IPS7ResetBit</b>	<b>ips7_resetbit</b>	ein Bit zurücksetzen in der SPS (E,A,M,DB )

### **Aufrufparameter:**

#### **Die Lese- und Schreibfunktionen besitzen die selben Eingangsparameter:**

<b>Nr</b>	<b>Datentyp</b>	<b>Datentyp (PHP)</b>	<b>Name</b>	<b>Funktion</b>
1	32-Bit Wert unsigned	long	Ref	Die Referenz der Verbindung, welche mit IS7Open generiert wurde. Dient zur internen Identifikation der Verbindung

<b>Nr</b>	<b>Datentyp</b>	<b>Datentyp (PHP)</b>	<b>Name</b>	<b>Funktion</b>
2	32-Bit Wert 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
3	32-Bit Wert unsigned	long	DBNr	Datenbausteinnummer, diese wird nur beim Typ 'D' verwendet. Ansonsten steht dort der Wert „0“
4	32-Bit Wert unsigned	long	Ab	Byte Adresse z.B. M 10.0, dann steht hier 10. Beachten Sie hier Unterschied zwischen IPS7RdW und IPS7RdPlcW bzw. IPS7WrW und IPS7WrPlcW. Siehe weiter unten !
5	32-Bit Wert unsigned	long	Bit	BitNr muss zwischen 0 und 7 liegen. z.B. bei M5.4 steht hier 4.
6	32-Bit Adresse	mixed	Buffer	<b>Dieser Parameter ist nur für IPS7RdBit.</b> 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); <b>Hinweis für PHP:</b> Bei PHP geben Sie als hier die Referenz einer Variablen an also: 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.

## Optimiertes Lesen verschiedener Datenbereiche in einem Stück (ab Version 1.35)

<i>Funktion</i>	<i>Funktion (PHP)</i>	<i>Beschreibung / Zweck</i>
<b>IPS7RdMulti</b>	<b>ips7_rdmulti</b>	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. <b>ACHTUNG!</b> Die Lesereihenfolge ist eine andere als die Reihenfolge der übergebenen 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 sein als der in der SPS sein. Der Aufrufer hat einen entsprechend großen Zielbereich zur Verfügung zu stellen. Was 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) zur 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.

## Der Aufbau des Request Records/Structs für IPS7RdMulti

Name	Typ	Beschreibung / Zweck																														
<i>DataArea</i>	<i>32-Bit unsigned</i>	<p>Die Auswahl des Speicherbereichs in der SPS (DB, Eingang, Ausgang, Merker), welcher bearbeitet werden soll:</p> <p>‘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)</p> <p>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</p> <p>‘Z’ = 90 dez. steht für Zähler</p> <p>Auch die Zähler sind in der SPS BCD-Codiert abgelegt. Die Zählerwerte reichen von 0 – 999.</p>																														
<i>DataType</i>	<i>32-Bit unsigned</i>	<p>Datentyp in der SPS</p> <table border="1"> <thead> <tr> <th>Name</th> <th>Wert</th> <th>SPS-Datentyp / Bitbreite</th> </tr> </thead> <tbody> <tr> <td><i>IPS7_BIT</i></td> <td>0</td> <td>Ein Bit / Boolean</td> </tr> <tr> <td><i>IPS7_BYTE</i></td> <td>1</td> <td>Byte (8 Bit)</td> </tr> <tr> <td><i>IPS7_WORD</i></td> <td>2</td> <td>WORD 16 Bit unsigned integer (ohne Vorzeichen)</td> </tr> <tr> <td><i>IPS7_INT</i></td> <td>3</td> <td>INT 16 Bit signed (mit Vorzeichen)</td> </tr> <tr> <td><i>IPS7_DWORD</i></td> <td>4</td> <td>DWORD 32 Bit unsigned integer (ohne Vorzeichen)</td> </tr> <tr> <td><i>IPS7_DINT</i></td> <td>5</td> <td>long 32 Bit signed integer (mit Vorzeichen)</td> </tr> <tr> <td><i>IPS7_REAL</i></td> <td>6</td> <td>S7Real</td> </tr> <tr> <td><i>IPS7_TIMER</i></td> <td>7</td> <td>Timer in der S7</td> </tr> <tr> <td><i>IPS7_COUNTER</i></td> <td>8</td> <td>Zähler in der S7</td> </tr> </tbody> </table>	Name	Wert	SPS-Datentyp / Bitbreite	<i>IPS7_BIT</i>	0	Ein Bit / Boolean	<i>IPS7_BYTE</i>	1	Byte (8 Bit)	<i>IPS7_WORD</i>	2	WORD 16 Bit unsigned integer (ohne Vorzeichen)	<i>IPS7_INT</i>	3	INT 16 Bit signed (mit Vorzeichen)	<i>IPS7_DWORD</i>	4	DWORD 32 Bit unsigned integer (ohne Vorzeichen)	<i>IPS7_DINT</i>	5	long 32 Bit signed integer (mit Vorzeichen)	<i>IPS7_REAL</i>	6	S7Real	<i>IPS7_TIMER</i>	7	Timer in der S7	<i>IPS7_COUNTER</i>	8	Zähler in der S7
Name	Wert	SPS-Datentyp / Bitbreite																														
<i>IPS7_BIT</i>	0	Ein Bit / Boolean																														
<i>IPS7_BYTE</i>	1	Byte (8 Bit)																														
<i>IPS7_WORD</i>	2	WORD 16 Bit unsigned integer (ohne Vorzeichen)																														
<i>IPS7_INT</i>	3	INT 16 Bit signed (mit Vorzeichen)																														
<i>IPS7_DWORD</i>	4	DWORD 32 Bit unsigned integer (ohne Vorzeichen)																														
<i>IPS7_DINT</i>	5	long 32 Bit signed integer (mit Vorzeichen)																														
<i>IPS7_REAL</i>	6	S7Real																														
<i>IPS7_TIMER</i>	7	Timer in der S7																														
<i>IPS7_COUNTER</i>	8	Zähler in der S7																														
<i>DBNr</i>	<i>32-Bit unsigned</i>	<p>Datenbausteinnummer, diese wird nur beim Typ ‘D’ verwendet.  Ansonsten steht dort der Wert „0“</p>																														
<i>Cnt</i>	<i>32-Bit unsigned</i>	Anzahl der Datenelemente, die gelesen / geschrieben werden sollen																														
<i>Start</i>	<i>32-Bit unsigned</i>	Startbyte in der SPS.																														

Name	Typ	Beschreibung / Zweck																								
<i>StartBit</i>	32-Bit <i>unsigned</i>	Die Nummer des ersten Bits in der SPS, Werte (0 – 7), wird nur bei Bit-Zugriffen verwendet, ansonsten 0																								
<i>PCDatyType</i>	32-Bit <i>unsigned</i>	Datentyp im PC																								
		<table border="1"> <thead> <tr> <th>Name</th> <th>Wert</th> <th>Datentyp in der SPS</th> </tr> </thead> <tbody> <tr> <td><i>PC_BYTE</i></td> <td>0</td> <td>Byte (8 Bit)</td> </tr> <tr> <td><i>PC_WORD16</i></td> <td>1</td> <td>16 Bit unsigned integer</td> </tr> <tr> <td><i>PC_INT16</i></td> <td>2</td> <td>16 Bit signed integer</td> </tr> <tr> <td><i>PC_WORD32</i></td> <td>3</td> <td>32 Bit unsigned integer</td> </tr> <tr> <td><i>PC_INT32</i></td> <td>4</td> <td>32 Bit signed integer</td> </tr> <tr> <td><i>PC_FLOAT</i></td> <td>5</td> <td>32 Bit Fließpunktzahl im PC (float)</td> </tr> <tr> <td><i>PC_DOUBLE</i></td> <td>6</td> <td>64-Bit Fließpunktzahl im OPC (double)</td> </tr> </tbody> </table>	Name	Wert	Datentyp in der SPS	<i>PC_BYTE</i>	0	Byte (8 Bit)	<i>PC_WORD16</i>	1	16 Bit unsigned integer	<i>PC_INT16</i>	2	16 Bit signed integer	<i>PC_WORD32</i>	3	32 Bit unsigned integer	<i>PC_INT32</i>	4	32 Bit signed integer	<i>PC_FLOAT</i>	5	32 Bit Fließpunktzahl im PC (float)	<i>PC_DOUBLE</i>	6	64-Bit Fließpunktzahl im OPC (double)
Name	Wert	Datentyp in der SPS																								
<i>PC_BYTE</i>	0	Byte (8 Bit)																								
<i>PC_WORD16</i>	1	16 Bit unsigned integer																								
<i>PC_INT16</i>	2	16 Bit signed integer																								
<i>PC_WORD32</i>	3	32 Bit unsigned integer																								
<i>PC_INT32</i>	4	32 Bit signed integer																								
<i>PC_FLOAT</i>	5	32 Bit Fließpunktzahl im PC (float)																								
<i>PC_DOUBLE</i>	6	64-Bit Fließpunktzahl im OPC (double)																								
<i>Result</i>	32-Bit <i>unsigned</i>	Dort steht nach Aufruf das Ergebnis für diesen Auftrag. Die einzelnen Werte finden Sie weiter unter bei der Beschreibung: <b><i>Rückgabewert für Read/Writefunktionen bzw. Result im Record IPS7RdMulti</i></b>																								
<i>UserData_0</i>	32-Bit <i>unsigned</i>	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.																								
<i>UserData_1</i>	32-Bit <i>unsigned</i>																									
<i>Data</i>	32-Bit <i>Pointer</i>	Der eigentliche, Zeiger auf den Speicherbereich im PC für diesen Auftrag.																								
<i>pUserData</i>	32-Bit <i>Pointer</i>	wie <i>UserData_0</i> und <i>UserData_1</i> , jedoch ein <i>Pointer</i>																								

### **Hier ein Beispiel für IPS7RdMulti in C /C++**

```
// kleine Funktion zur Initialisierung eines einzelnen Requests
void InitRq (IPS7_RQ_MULTII *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, 0, sizeof (Rq));

Cnt = 0;

// lese ab E 4.0 32 Bit
InitRq (&Rq[Cnt++], 'E', IPS7_BIT, PC_BYTE, 0, 4, 0, 32, EBits);

// lese ab EB0 20 Byte und lege diese ab [20] ab
InitRq (&Rq[Cnt++], 'E', IPS7_BYTE, PC_BYTE, 0, 0, 0, 20, &EBytes[20]);

//lese ab MB 20 10 Worte
InitRq (&Rq[Cnt++], 'M', IPS7_WORD, PC_WORD16, 0, 20, 0, 1, MWords);

//lese DB10 ab Datenbyte 0 150 Worte
InitRq (&Rq[Cnt++], 'D', IPS7_WORD, PC_WORD16, 10, 0, 0, 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, 0, 0, 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, 0, 60, DB20RealAsFloat);

//lese ab Timer 5 10 Timer und lege diese als int ab
InitRq (&Rq[Cnt++], 'T', IPS7_TIMER, PC_WORD32, 0, 5, 0, 10, TimerAsInt);
// T5 10 Timer

Res = IPS7RdMulti(Ref, Rq, Cnt);

for (int i = 0; i < Cnt; i++)
{
    if (Rq[i].Result == 0) // alles in Ordnung
    {
        //.. Daten auswerten und verarbeiten
    }
    else
    {
        // Fehler auswerten
    }
}
}

```

**Rückgabewert für Read/Writefunktionen bzw. Result im Record IPS7RdMulti:**

Die Funktionen liefert einen 32-Bit Wert mit Vorzeichen als Rückgabewert mit folgender Bedeutung:

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.

Wert	Fehlerbeschreibung	Reaktion
-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	Unbenötigte 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	<b>Nur bei MultiRead:</b> PC und S7-Datentyp stehen falscher Relation z.B. PC = BYTE SPS = Wort	PC-Datenbereich anpassen

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

## 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

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

## Neu in Version 1.17 !!!!

### **Unterschied S7RdPlcW <-> IPS7RdW und S7WrPlcW <-> S7WrW**

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 vor 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);**

### **!!! Beachte 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.

<b>Wortadresse</b>	<b>zugeordnete Bytes</b>
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:

<b>PC</b>	<b>SPS</b>
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

### **Programmbeispiele:**

#### **a) Aufruf von C oder C++ aus:**

**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, 0, 10, WordBuffer);**

// Aufruf der Bytefunktion z.B. Lese MB 0 , 10 Bytes

**IPS7RdB (Ref, 'M', 0, 0, 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

#### **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

**IPS7RdW (Ref, LongWord ('D'), 10, 0, 10, @WordBuffer[0]);**

// Aufruf der Bytefunktion z.B. Lese MB 0 , 10 Bytes

**IPS7RdB (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

**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:

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

### Funktionen zum Bit-Lesen, Setzen und Rücksetzen

<i>Funktion</i>	<i>Funktion (PHP)</i>	<i>Beschreibung / Zweck</i>
<b>IPS7RdBit</b>	<b>ips7_rdbit</b>	ein Bit lesen SPS (E,A,M, DB )
<b>IPS7SetBit</b>	<b>ips7_setbit</b>	ein Bit setzen lesen aus der SPS (E,A,M,DB)
<b>IPS7ResetBit</b>	<b>ips7_resetbit</b>	ein Bit zurücksetzen in der SPS (E,A,M,DB )

<i>Funktion</i>	<i>Funktion (PHP)</i>	<i>Beschreibung / Zweck</i>
<b>IPS7GetSockErr</b>	<b>ips7_getsockerr</b>	Liefert den letzten Socket-Fehler zurück

### Aufrufparameter:

<b>Nr</b>	<b>Datentyp</b>	<b>Datentyp (PHP)</b>	<b>Name</b>	<b>Funktion</b>
1	32-Bit Wert unsigned	long	Ref	Die Referenz der Verbindung, welche mit IS7Open generiert wurde. Dient zur internen Identifikation der Verbindung

### Rückgabewert:

Die Funktionen liefert einen 32-Bit Wert mit Vorzeichen als Rückgabewert mit folgender Bedeutung:

<b>Wert</b>	<b>Fehlerbeschreibung</b>	<b>Bedeutung / Reaktion</b>
0	alles OK	Es liegt kein Fehler an

Wert	Fehlerbeschreibung	Bedeutung / Reaktion
-3	Mit der angegebenen Referenznummer wurde kein IPS7Open durchgeführt	Haben Sie IPS7Open aufgerufen ?.
-99	Die Referenznummer ist ungültig	-----
Sonstige	Socketerror	Erklärung siehe Liste unterhalb.

### Socketfehler: Diese Liste erhebt keinen Anspruch auf Vollständigkeit

Name	Code	Bedeutung
WSAEINTR	10004	Aufruf wurde abgebrochen
WSAEBADF	10009	
WSAEACCES	10013	Zugriffsfehler
WSAEFAULT	10014	Parameter sind falsch
WSAEINVAL	10022	<ol style="list-style-type: none"> <li>1. Andere Funktion muss vorher aufgerufen werden Socket ist schon an Adresse gebunden</li>   <li>2. Socket noch nicht an Adresse gebunden bzw. schon verbunden</li> </ol>
WSAEMFILE	10024	Ressourcen fehlen (Dateien, Warteschlangen)
WSAEWOULDBLOCK	10035	Aufruf würde blockieren
WSAEINPROGRESS	10036	Parallele Aufrufe nicht erlaubt
WSAEALREADY	10037	Abgebrochene Routine trotzdem schon fertig
WSAENOTSOCK	10038	Kein gültiger Socket angegeben
WSAEDESTADDRREQ	10039	Zieladresse benötigt
WSAEMSGSIZE	10040	Datagramm zu groß, wurde abgeschnitten
WSAEPROTOTYPE	10041	
WSAENOPROTOOPT	10042	Unbekannte Socket-Option
WSAEPROTONOSUPPORT	10043	Protokoll wird nicht unterstützt
WSAESOCKTNOSUPPORT	10044	Sockettyp wird in angegebener Adressfamilie nicht unterstützt
WSAEOPNOTSUPP	10045	Dieser Sockettyp wird nicht unterstützt
WSAEPFNOSUPPORT	10046	Protokollfamilie wird nicht unterstützt
WSAEAFNOSUPPORT	10047	Adressfamilie wird nicht unterstützt
WSAEADDRINUSE	10048	IP-Adresse bzw. Port werden schon/noch benutzt
WSAEADDRNOTAVAIL	10049	Port/Adresse nicht verfügbar
WSAENETDOWN	10050	Netzwerk reagiert nicht
WSAENETUNREACH	10051	Netzwerk kann nicht erreicht werden
WSAENETRESET	10052	Verbindung durch TCP/IP zurückgesetzt
WSAECONNABORTED	10053	Verbindung durch TCP/IP abgebrochen
WSAECONNRESET	10054	Partner hat Verbindung zurückgesetzt
WSAENOBUFS	10055	Ressourcen fehlen (Interner Pufferspeicher)
WSAEISCONN	10056	Socket ist schon verbunden
WSAENOTCONN	10057	Socket ist noch nicht verbunden
WSAESHUTDOWN	10058	Andere Seite hat Verbindung einseitig beendet
WSAETOOMANYREFS	10059	

<b>Name</b>	<b>Code</b>	<b>Bedeutung</b>
WSAETIMEDOUT	10060	Aufruf dauert zu lange, daher Abbruch
WSAECONNREFUSED	10061	Angerufener möchte keinen Verbindungsaufbau
WSAELOOP	10062	
WSAENAMETOOLONG	10063	
WSAEHOSTDOWN	10064	
WSAEHOSTUNREACH	10065	Host nicht erreichbar
WSAENOTEMPTY	10066	
WSAEPROCLIM	10067	
WSAEUSERS	10068	
WSAEDQUOT	10069	
WSAESTALE	10070	
WSAEREMOTE	10071	
WSASYSNOTREADY	10091	Netzwerk nicht zur Kommunikation bereit
WSAVERNOTSUPPORTED	10092	gewünschte Winsock-Version wird nicht unterstützt
WSANOTINITIALISED	10093	Socket.Initialize muß aufgerufen werden
WSAHOST_NOT_FOUND	11001	DNS-Server nicht gefunden
WSATRY_AGAIN	11002	Gesuchter Rechner nicht gefunden
WSANO_RECOVERY	11003	Nicht behebbarer Fehler
WSANO_DATA	11004	Keine Namensdaten vorhanden
WSANO_ADDRESS	11004	

Die Funktionsdeklarationen entnehmen Sie den betreffenden Headerdateien:

für C/C++: ips7lnk.h

für Delphi: ips7lnk.pas

für VB: ips7lnk.bas

für .Net-Sprachen (C#, VB.Net) Assembly aus dem Verzeichnis NetFiles einbinden. Die Hilfe/Dokumentation finden Sie in ips7lnk.chm.