TRAEGER.DE
INDUSTRY COMPONENTS

IP-S7-LINK

Via TCP/IP to the SIMATIC S7 via HMI/PG protocol (RFC1006)
Documentation for Version 1.72

Supported Systems

- S7-1500
- S7-1200
- S7-300/400/200
- !Logo
- WIN AC RTX
- VIPA S7
- each S7-compatible PLC

Operating system

- MS-Windows Desktop XP/7/8/10 32/64-Bit
- all MS-Windows Server 32/64-Bit
- all Linux 32/64-Bit
- Embedded Linux 32/64Bit

Programming languages

- C
- C++
- C#
- VB
- Delphi
- Excel
- Access
- PHP and other

Release Notes

Installation

Windows

- C/C++, Delphi VB etc.:
  - copy the DLL to the program directory or to the system directory
- PHP
  - copy the extension ips7lnk_php.dll into the extension directory of the PHP-installation
  - load the module over PHP.ini or in the program
    php.ini: extension = ips7lnk_php.dll
    in the program: ld ('ips7lnk_php.dll');

Linux

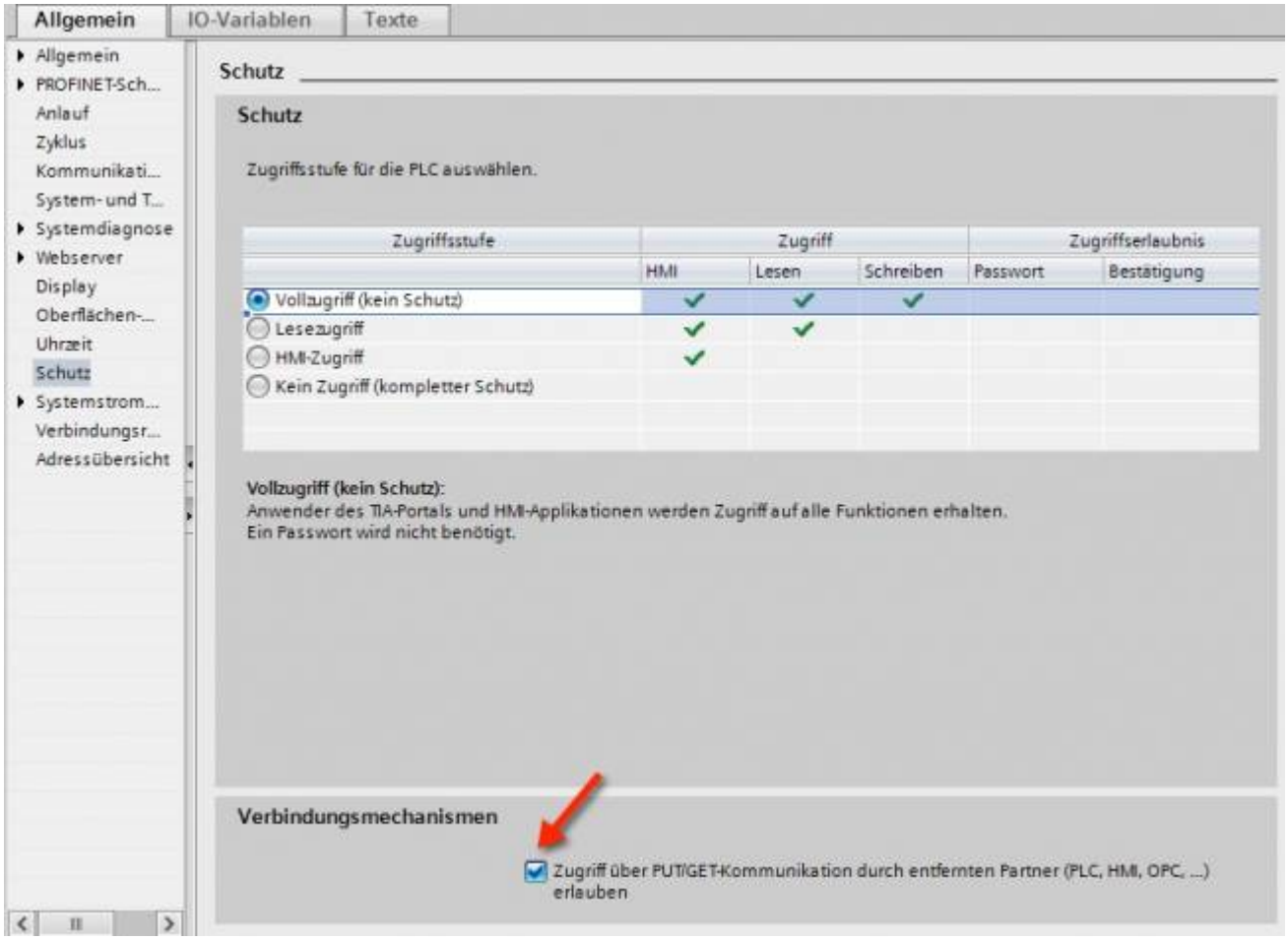
- C/C++ and other
  - Link the o. file to your program or create a .so file
- PHP under Linux
  - copy the extension ips7lnk_php.so into the extension directory of the PHP-installation
  - load the module over PHP.ini or in the program
    php.ini: extension = ips7lnk_php.so
    in the program: ld (´ips7lnk_php.so');

PLC - settings

Settings for S7 1200/1500

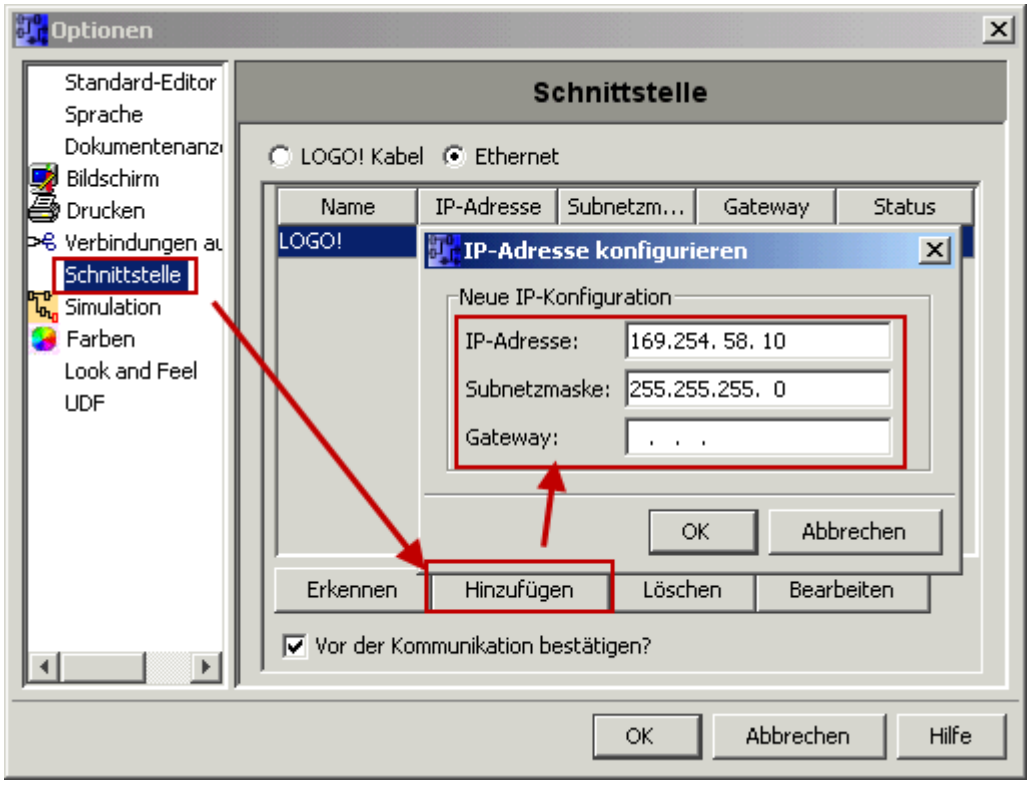
The optimized block access needs to be deactivated in the data block attributes for access to the S7-1500 and S7-1200.

In the S7-1500 must be enabled in the communication setting in addition to the PUT / GET access . How this works you see here (snapshot from TIA Portal) .

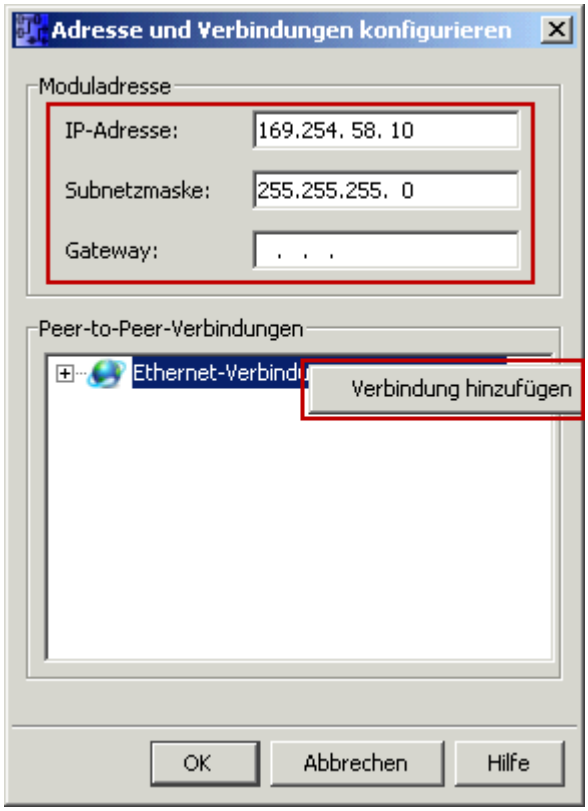


Settings Logo

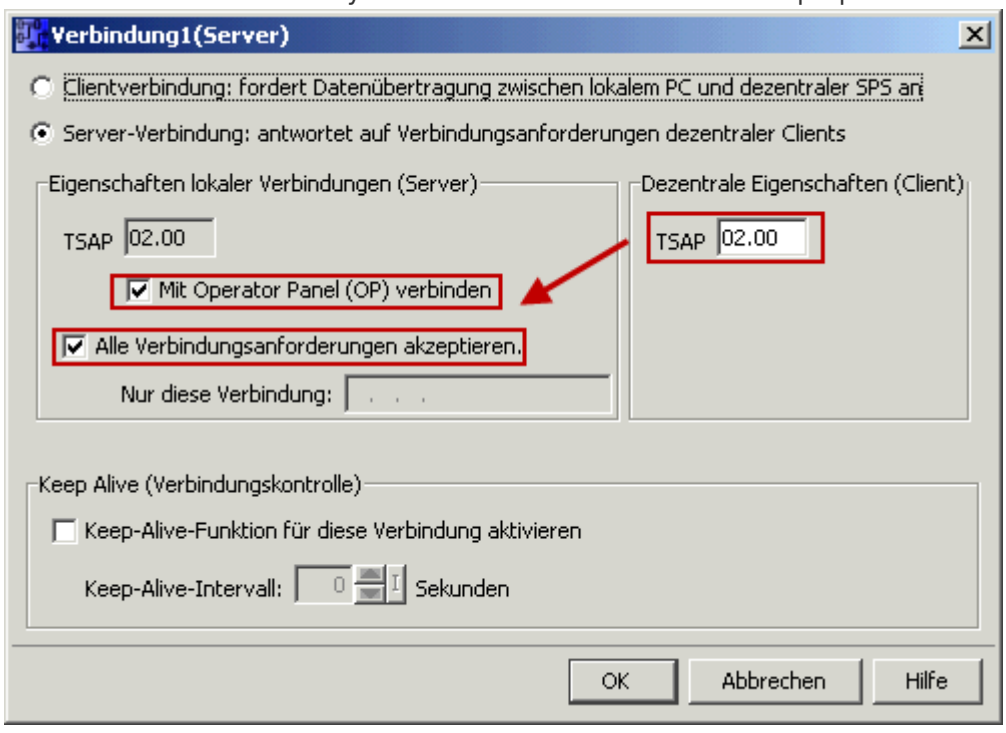
1. Use the Logo Soft Comfort the IP address of a logo! PLCs:



2. Configure PLCs so that connections from an HMI device accepted the Logo!. To do so, go to “Tools- > Ethernet Connections” and then add a new connection.



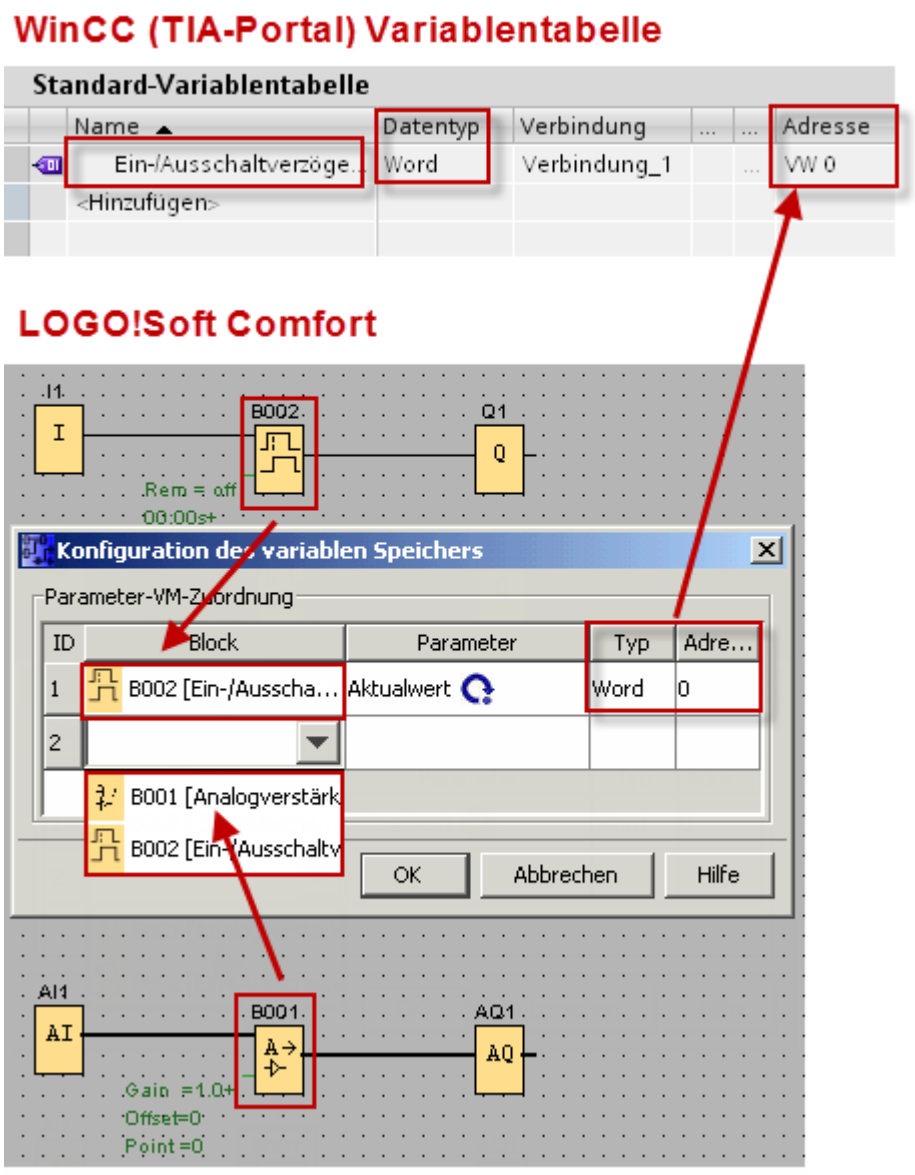
3. Double-click on the newly created connection to access the properties.



Select:

1. Server Connection
2. Local TSAP: 02:00 - 02:00 decentralized TSAP
3. accept all connections.

You can access DB1, inputs , outputs, flags , counters and timers with IP -S7 -LINK . Now put on “ Tools- > VM parameter map ” the variables that are to be transferred to the DB1.



Functionality

IP-S7-LINK implements the connection of a PC to Industrial Ethernet of the SIMATIC S7. The library is available for various programming languages, operating systems, architectures and platforms. The tool is developed in pure C / C ++ code. It can be ported to any platform / architecture with minimal effort. The library provides the necessary functions for communication. The connection to the PLC is independently controlled by IP-S7-LINK and automatically restored in the event of a fault. Only the IP address of the PLC / CP and the slot of the CPU in the PLC rack are required for the coupling. Immediately, flags, inputs, outputs, data blocks, PLC time etc. can be read and written.

Functions in Detail

Please note: The functions are executed with the standard socket interface, which means that the function returns to the caller only after the task has been completed.

For asynchronous operation, simply call these functions from a separate thread, which is responsible for communicating with the system. The following functions are available:

Initialization

IPS7Open / IPS7OpenPG / IPS7OpenS7200

Functions in Detail

Name	Name (PHP)	Description / Purpose
IPS7Open	ips7_open	to initialize the connection, there will be only memory prepared. On the first call of the read or write functions the TCP / IP connection is started automatically. The connection will be established over the OP-Channel
IPS7OpenPG	ips7_openpg	Since version 1.17, to initialize the connection, there will be only memory prepared. On the first call of the read or write functions the TCP / IP connection is started automatically. The connection will be established over the PG-Channel
IPS7OpenS7200	ips7_opens7200	Since version 1.21, to initialize the connection, there will be only memory prepared. On the first call of the read or write functions the TCP / IP connection is started automatically. The connection will be established to a S7-200

Call parameters

Nr	data type	PHP-data type	Name	Function
1	32-Bit Pointer to C-String	string	IPAdr	IP-address of the PLC in the Format: xxx.xxx.xxx.xxx. Sample: “192.169.0.100”
2	32-Bit unsigned	long	Rack	Number of the Rack, in which the PLC-CPU plugged in. Counting starts with „0“. General 0. No matter when S7-200
3	32-Bit unsigned	long	Slot	Number of the slot of the CPU starting with „1„, general „2“ at S7-300-400 or „1“ at S7-1200/1500. No matter when S7-200
4	32-Bit unsigned	long	RxTimeout	Timeout in Milliseconds for waiting for TCP/IP-Packets from the PLC, 0 is standard settings = 500 ms
5	32-Bit unsigned	long	TxTimeout	Timeout in Milliseconds for sending the TCP/IP-Packets to the PLC, 0 is standard settings = 500 ms
6	32-Bit unsigned	long	ConTimeout	Timeout in Milliseconds waiting for establishing a connection to the PLC, 0 is standard settings = 5000 ms (5sec.) must be extended if necessary

C/C++ Functional Declaration

```
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 Functional Declaration

```
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 Functional Declaration

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

IPS7OpenEx

Name	Name (PHP)	Description / Purpose
IPS7OpenEx	ips7_openex	From version 1.23, to initialize the connection, only memory is prepared. On the first call of the read or write functions the TCP / IP connection is started automatically. t can establish a connection, by choosing the parameters, to a OP/PG S7200 or connections over a subnet

Call parameter

Nr	data type	data type (PHP)	Name	Function
1	32-Bit pointer auf C-String	string	IPAdr	IP-Address of the PLC in the format: xxx.xxx.xxx.xxx. Example: 192.169.0.100
2	32-Bit unsigned	long	Rack	Number of the Rack, in which the PLC-CPU plugged in. Counting starts with „0“. General 0. No matter when S7-200

Nr	data type	data type (PHP)	Name	Function
3	32-Bit unsigned	long	Slot	Number of the slot of the CPU starting with „1„, general „2“ at S7-300-400 or „1“ at S7-1200/1500. No matter when S7-200
4	32-Bit unsigned	long	SubNet-Id	Subnet-ID, to be when accessed over a subnet. In the Step-S7 Software is the address as example: 0035 - 0001, then specify 0x00350001. Used only when Access Mode 10 or 11
5	32-Bit unsigned	long	DstMPIAdr	Target-MPI-Address, if the connection should be established via a subnet. See Access Mode 10 and 11!
6	32-Bit unsigned	long	AccessMode	Type of access: 0 = OP-connection, establishing a connection to the specified CPU by the rack and slot number 1 = PG-connection, establishing a connection to the specified CPU by the rack and slot number 2 = connection to a S7-200 over pluged TCP/IP-CP 3 = connection to a Siemens-Logo-PLC (since 1.60.78) 4 = connection over the channel „other“ (since 1.60.79) 10 = OP-connection on subnet, which is connected to the path indicated by the rack and slot CPU build, subnet ID and DstMPI address to be specified 11 = PG-connection on subnet, which is connected to the path indicated by the rack and slot CPU build subnetID and DstMPI address to be specified 20 = connection to S5-LAN++. The conversion of the real values are then also at MultiRead accesses
7	32-Bit unsigned	long	RxTimeout	Timeout in Milliseconds for waiting for the TCP/IP-Packets from the PLC, 0 is standard settings = 500 ms
8	32-Bit unsigned	long	TxTimeout	Timeout in Milliseconds for sending the TCP/IP-Packets to the PLC, 0 is standard settings = 500 ms
9	32-Bit unsigned	long	ConTimeout	Timeout in Milliseconds waiting for establishing a connection to the PLC, 0 is standard settings = 5000 ms (5sec.) must be extended if necessary.

C/C++ Functional Declaration

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

Delphi / Pascal Functional Declaration

```
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 Functional Declaration

```
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

Works as IPS7OpenEx. But this functions provides the feature to pass an LocalTSAP and RemoteTSAP. So any connections to the PLC with custom TSAPs can be used. The length of TSAPs is fixed to 2 bytes. This feature can be used e.g. for connect to Logo8. There the TSAP for a channel can be configured in the PLC. Example: In the PLC is configured:

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

Note: Local / Remote are in relation to the partner.

Name	Name (PHP)	purpose
IPS7OpenExWithTSAP	ips7_openexwithtsap	

parameters

Nr	data type	data type (PHP)	Name	Function
1	32-Bit pointer auf C-String	string	IPAdr	IP-Address of the PLC in the format: xxx.xxx.xxx.xxx. Example: 192.169.0.100
2	32-Bit unsigned	long	Rack	Number of the Rack, in which the PLC-CPU plugged in. Counting starts with „0“. General 0. No matter when S7-200
3	32-Bit unsigned	long	Slot	Number of the slot of the CPU starting with „1„, general „2“ at S7-300-400 or „1“ at S7-1200/1500. No matter when S7-200
4	32-Bit unsigned	long	SubNet-Id	Subnet-ID, to be when accessed over a subnet. In the Step-S7 Software is the address as example: 0035 - 0001, then specify 0x00350001. Used only when Access Mode 10 or 11
5	32-Bit unsigned	long	DstMPIAdr	Target-MPI-Address, if the connection should be established via a subnet. See Access Mode 10 and 11!

Nr	data type	data type (PHP)	Name	Function
6	32-Bit unsigned	long	AccessMode	Type of access: 0 = OP-connection, establishing a connection to the specified CPU by the rack and slot number 1 = PG-connection, establishing a connection to the specified CPU by the rack and slot number 2 = connection to a S7-200 over pluged TCP/IP-CP 3 = connection to a Siemens-Logo-PLC (since 1.60.78) 4 = connection over the channel „other“ (since 1.60.79) 10 = OP-connection on subnet, which is connected to the path indicated by the rack and slot CPU build, subnet ID and DstMPI address to be specified 11 = PG-connection on subnet, which is connected to the path indicated by the rack and slot CPU build subnetID and DstMPI address to be specified 20 = connection to S5-LAN++. The conversion of the real values are then also at MultiRead accesses
7	32-Bit unsigned	long	RxTimeout	Timeout in Milliseconds for waiting for the TCP/IP-Packets from the PLC, 0 is standard settings = 500 ms
8	32-Bit unsigned	long	TxTimeout	Timeout in Milliseconds for sending the TCP/IP-Packets to the PLC, 0 is standard settings = 500 ms
9	32-Bit unsigned	long	ConTimeout	Timeout in Milliseconds waiting for establishing a connection to the PLC, 0 is standard settings = 5000 ms (5sec.) must be extended if necessary.
10	pointer auf BYTE Array der Länge 2	BYTE *	LocalTSAP	
11	pointer auf BYTE Array der Länge 2	BYTE *	RemoteTSAP	

C/C++ Function Declaration

```
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 Function Declaration

```
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 Function Declaration

```
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 Values

Return Values

Value	Error description	Description
>= 0	OK	The return value is the reference number for this connection and must be used as input parameter Ref for all other functions
-2	No free resource	Maximum reach of available connections
-10	AccessMode not possible (since 1.23)	if use of a illegal number for AccessMode. See IPS7OpenEx AccessMode

Deinitialization

IPS7Close

Disables the connection, clears memory and disconnects the TCP/IP connection.

Call parameter

Nr	data type	data type (PHP)	Name	Function
1	32-Bit value unsigned	long	Ref	The reference of the connection that was generated with IPS7Open. Used to identify the connection internally

C/C++ Functional Declaration

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

Delphi / Pascal Functional Declaration

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

Visual Basic Functional Declaration

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

Return value The function provides a signed 32-bit return value with the following meaning:		
Value	**Error description**	**Meaning/Reaction**
0	OK	Memory again enabled and connection closed, if available
-3	No IPS7Open was performed with the specified reference number	Did you called IPS7Open?
-99	The reference number is invalid	
-30	only PHP	The number or type of passed parameters is incorrect
-31	only PHP	The internal conversion of the data could not be performed, e.g. A string was passed where long is necessary

Connection

IPS7Connect

Executes an explicit connection to the PLC. From version 1.35 onwards! This allows the connection to the PLC to be established without a read/write request.

Call parameter

Nr	data type	data type (PHP)	Name	Function
1	32-Bit unsigned	long	Ref	The reference of the connection that was generated with IPS7Open. Used to identify the connection internally

C/C++ Functional Declaration

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

Delphi / Pascal Functional Declaration

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

Return value

Value	Error description
1	Connection established
⇐ 0	Connection could not be established. The exact meaning can be found in the return values for the read / write functions

IPS7GetConnectStatus

Checks the TCP/IP connection status to the PLC.

Call parameter

Nr	data type	data type (PHP)	Name	Function
1	32-bit unsigned	long	Ref	The reference of the connection that was generated with IPS7Open. Used to identify the connection internally

C/C++ Functional Declaration

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

Delphi / Pascal Functional Declaration

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

Visual Basic Functional Declaration

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

Return value

Value	Error description
1	Connection established and alive
⇐ 0	Connection is interrupted. Possibly call IPS7Connect, read or write function

IPS7SetKeepAlive

Since V 1.35! Sets individual TCP/IP KeepAlive times for the connection specified by Ref. Must only be used if the default values do not apply.
You should execute this function immediately after the “Open” call.

Call parameter

Nr	data type	data type (PHP)	Name	Function
1	32-bit unsigned	long	Ref	The reference of the connection that was generated with IPS7Open. Used to identify the connection internally
2	32-bit unsigned	long	AliveTime	If no data traffic on the TCP/IP connection takes place within the time **AliveTime** (ms), a KeepAlive telegram is sent to check the connection. If an error is detected during this check, the IP stack sends a next KeepAlive telegram within the time **AliveInterval** (ms). This is repeated several times within the time **AliveInterval** (Win 6 times). If the operation is not successful, the connection is terminated
3	32-bit unsigned	long	AliveInterval	The interval in ms, in which KeepAlive telegrams are repeated. This is activated if an error occurred while sending / receiving a KeepAlive telegram

C/C++ Functional Declaration

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

Delphi / Pascal Functional Declaration

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

Visual Basic Functional Declaration

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

Return value

Value	Error description
0	Setting the values was successful
< 0	Setting the keep-alive time could not be executed

IPS7SetTCPPort

By default, TCP/IP port 102 (RFC 1006) is used. With IPS7SetTCPPort it can be changed. You should execute this function immediately after the “Open” call.

Call parameter

Nr	data type	data type (PHP)	Name	Function
1	32-Bit unsigned	long	Ref	The reference of the connection that was generated with IPS7Open. Used to identify the connection internally
2	32-Bit unsigned	long	Port	Number of the new TCP/IP-Port 1 - 65565

C/C++ Functional Declaration

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

Delphi / Pascal Functional Declaration

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

Return value

Value	Error description	Meaning/Reaction
0	OK	Memory again enabled and connection closed, if available
-3	No IPS7Open was performed with the specified reference number	Did you called IPS7Open?
-99	The reference number is invalid	
-30	only PHP	The number or type of passed parameters is incorrect

Value	Error description	Meaning/Reaction
-31	only PHP	The internal conversion of the data could not be performed, e.g. A string was passed where long is necessary

IPS7GetSockErr

Name	Name (PHP)	Description / Purpose
IPS7GetSockErr	ips7_getsockerr	Returns the last socket error

Call parameter

Nr	data type	data type (PHP)	Name	Function
1	32-Bit unsigned	long	Ref	The reference of the connection that was generated with IS7Open. Used to identify the connection internally

C/C++ Functional Declaration

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

Delphi / Pascal Functional Declaration

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

Visual Basic Functional Declaration

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

Return Values

Value	Error description	Meaning/Reaction
0	OK	There are no errors
-3	No IPS7Open was performed with the specified reference number	Did you called IPS7Open?
-99	The reference number is invalid	
Others	Socket error	For explanation see list below

Read and Write

IP-S7-LINK provides read and write functions for each data type.
The data type functions differ in the call parameters. In the following, the methods with the same parameters are grouped together.

Byte

Name	Name (PHP)	Description/purpose
IPS7RdB	ips7_rdb	read byte-wise (I,O,F,P,DB)
IPS7WrB	ips7_wrb	write byte-wise (I,O,F,P,DB,Z)

Word

Name	Name (PHP)	Description/purpose
IPS7RdW	ips7_rdw	read word-wise (I,O,F,P,DB)
IPS7RdPlcW	ips7_rdplcw	read word-wise (I,O,F,P,DB) but start address after PLC addressing to access unsigned start addresses
IPS7WrW	ips7_wrw	write word-wise (I,O,F,P,DB,Z)
IPS7WrPlcW	ips7_wrplcw	write word-wise (I,O,F,P,DB,Z) but start address after PLC addressing to access unsigned start addresses

DWord

Name	Name (PHP)	Description/purpose
IPS7RdDW	ips7_rddw	read double word-wise (I,O,F,P,DB,T)
IPS7WrDW	ips7_wrdw	write double word-wise (I,O,F,P,DB,T)

Real

Name	Name (PHP)	Description/purpose
IPS7RdReal	ips7_rdreal	read real (floating point) (I,O,F,P,DB))
IPS7WrReal	ips7_wrreal	write real (floating point (I,O,F,P,DB))

LInt

Name	Name (PHP)	Description/purpose
IPS7RdLInt	ips7_rdlint	read LInt (64 Bit) (I,O,F,P,DB)
IPS7WrLInt	ips7_wrlint	write LInt (64 Bit) (I,O,F,P,DB)

LUInt

Name	Name (PHP)	Description/purpose
IPS7RdULInt	ips7_rdulint	read ULInt (64 Bit) (I,O,F,P,DB)
IPS7WrULInt	ips7_wrulint	write ULInt (64 Bit) (I,O,F,P,DB)

String

Name	Name (PHP)	Description/purpose
IPS7RdString	ips7_rdstring	read String (I,O,F,P,DB) Start address after PLC addressing 1.Byte possible length 2. Byte actual length from 3. Byte data (sample address: DB0.DBB0, from DB0.DBB2 stand the actual data). IP-S7-LINK reads the possible length of the string → Shortens to actual length → adds NULL-CHAR ('\0') at the end

Name	Name (PHP)	Description/purpose
IPS7WrString	ips7_wrstring	write String (I,O,F,P,DB) Start address after PLC addressing. Length of the transferred string is determined and written to the contents of the content. **Attention!** Define your data area large enough so that no existing process data can be overwritten

Call parameters

Call parameter

Nr	data type	data type (PHP)	Name	Function
1	32-Bit unsigned	long	Ref	The reference of the connection that was generated with IS7Open. Used to identify the connection internally
2	32-Bit unsigned	long	Type	The selection of the memory area in the PLC (DB, input, output, flag), which is to be edited: 'D' = 68 dec. Stands for data block 'E' = 69, dec. Stands for inputs 'A' = 65 dec. Stands for outputs 'M' = 77 dec. Stands for flag 'P' = 80 dec. Stands for peripheral 'T' = 84 dec. Stands for timers (only possible with double word functions). The timers are stored in the PLC with time base and value in the BCD format. In order to process this format immediately in the PC, the driver performs an automatic conversion in milliseconds. The smallest possible raster is 10 ms. When writing to the PLC, the driver automatically selects a suitable time base. This can lead to rounding. The time range is from 0 to 9990000 ms 'Z' = 90 dec. Stands for counter (only possible with word functions). The counters are stored in the PLC BCD-coded. The counter values range from 0 - 999
3	32-Bit unsigned	32-Bit value unsigned	DBNr	Data block number, this is only used for type 'D'. Otherwise, the value “0”
4	32-Bit unsigned	32-Bit value unsigned	From	First word or byte from which to read or write. For word operations Start word For byte, double word and real functions Start byte For timer or counter, this is the number of the first element to be read
5	32-Bit unsigned	32-Bit value unsigned	Count	Number of units (byte, words, double words, real or units, e.g. timers) to be read or written

Nr	data type	data type (PHP)	Name	Function
6	32-Bit Pointer	mixed	Buffer	The address to the source or target memory in the PC. For word functions, this is a pointer to a field of 16-bit-wide words. For the Byte functions, this is an address on a field with 8-bit-wide bytes. For double-word pointers to Long For real pointers to a Double
				Note for PHP: For PHP, specify the reference of a variable
				Comment on ips7_rdplcw, ips7_rdw, ips7_rddw, ips7_rdreal}} Call for example.: &Result = ips7_rdplcw ($Ref, ord (“M”), 0, 6,5, &$W);
				If more than one element is read, the variable is converted into an array of type long or double. If only one value is read and the variable is not an array, the value is stored as long. If the variable is already an array and only one value is read, the result is stored in the first element of the array. Duplicates (ips7_rddw) are always processed with signs (signed)
7	———	long (optional)	bSigned at ipss7_rdplcw, ips7_rdw, ips7_wrplcw, ips7_wrdw bLong at ips7_rdb	**Note for PHP:** ips7_rdplcw, ips7_rdw, ips7_wrplcw, ips7_wrdw Optionally, it can be determined whether the values should be read as signed or unsigned 16-bit integers. If the parameter is not specified, the operation is always signed. We pass the parameters: 0 = without sign (unsigned) 1 = with sign In addition, the functions **ips7_i2w** and **ips7_w2i** are available for a subsequent conversion of individual values. You can find more information here. **ips7_rdb** Ips7_rdb saves the result as a string. However, if you want to use the values simply as an array, you can use bLong = 1 to store the result as long.Array.

Please pay attention to word-wise access

For word-by-word access, please check whether you want to use the initial address according to PLC syntax, or the computationally correct.

You must use IPS7RdPlcW or IPS7RdW or IPS7WrPlcW or IPS7RdW.

The PC and the PLC have different addressing modes. In the S7, the memory area is byte-wise oriented. This is how you address the MB0 and MB1 from the viewpoint of the PLC programmer with MW 0, but the MB1 and MB2 with MW1. You can see that MW0 and MW1 overlap in MB1.

Prior to version 1.17, it was only possible to access straight start addresses with the word functions. As of V 1.17, access to unsigned start addresses is possible with the functions **S7RdPlcW and S7WrPlcW** . If you now want to read MW 1, as the PLC programmer sees, call:

IPS7RdPlcW (Ref, 'M', 0, 1, 1, word buffer); *!!! Note this with word operations with S7RdW and S7WrW !!!*

Example of flag. This also applies to inputs, outputs and data words. The word addressing in the PLC uses the following bytes:

Word address	Assigned bytes
MW0	MB 0 and MB 1
MW1	MB 1 and MB 2

Word address	Assigned bytes
MW2	MB 2 and MB 3

You can see that the use of unsigned word addresses can result in a double assignment. Therefore the word functions (IPS7RdW and IPS7WrW) only support access to even word addresses. This means that the start word number in the driver is always multiplied by 2. This method allows a simple image of the PLC memory in the PC. So a word step in the PC are 16 bits in the PC and 16 bits in the PLC. Example:
WORD Buf [64];
The call ***IPS7RdW (Ref, Type, DBNr, 0, 5, Buf)*** has the following effect:

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

So you have to halve the starting word number in order to access with the PC correctly. This also applies to data blocks → unsigned word addresses of the PLC can not be read or written word by word.
If you still want to address unsigned start addresses, use the IPS7RdPlcW and IPS7WrPlcW functions.

Additional functions for PHP

long ips7_w2i(mixed Buffer,long Count);
Convert unsigned 16-Bit - value in signed 16-Bit value

long ips7_i2w(mixed Buffer, long Count);
Convert signed 16-Bit - value in unsigned 16-Bit value

Parameter	Description / Purpose
Buffer	Reference to the long values (array) or the long value to be converted
Count	Count of the values

C/C++ Functional Declaration

```
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 Functional Declaration

```
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 Functional Declaration

```
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++ Functional Declaration

```
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 Functional Declaration

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

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

Visual Basic Functional Declaration

```
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)
```

Bit

Name	Name(PHP)	Description/purpose
IPS7RdBit	ips7_rdbit	read a Bit (I,O,F,P,DB)
IPS7SetBit	ips7_setbit	set a Bit (I,O,F,P,DB)
IPS7ResetBit	ips7_resetbit	reset a Bit in the PLC (I,O,F,P,DB)

Call parameter

Nr	data type	data type (PHP)	Name	Function
1	32-Bit unsigned	long	Ref	The reference of the connection that was generated with IPS7Open. Used to identify the connection internally
2	32-Bit unsigned	long	Typ	The selection of the memory area in the PLC (DB, input, output, flag), which is to be edited: 'D' = 68 dec. Stands for data block 'E' = 69, dec. Stands for inputs 'A' = 65 dec. Stands for outputs 'M' = 77 dec. Stands for flag 'P' = 80 dec. Stands for peripheral
3	32-Bit unsigned	long	DBNr	Data block number, this is only used for type 'D'. Otherwise, the value “0”
4	32-Bit unsigned	long	Ab	Byte address e.g. M 10.0, then here is 10. Note the difference between IPS7RdW and IPS7RdPlcW or IPS7WrW and IPS7WrPlcW. See below!
5	32-Bit unsigned	long	Bit	BitNr must be between 0 and 7, e.g. At M 5.4, here is 4
6	32-Bit Adresse	mixed	Buffer	**This parameter is only for IPS7RdBit** The address to the destination memory in the PC. Pointer to a byte. If bBit is set content 1 else 0. Example: read M 6.5 BYTE W; IPS7RdBit (Ref, 'M', 0, 6.5 & W); **Note for PHP:** For PHP, specify the reference of a variable as follows: ips7_rdbit (Ref, ord (“M”), 0, 6,5, & $ W); This variable is automatically converted to a “long”. Therefore, use a variable that has not yet been used. The state of the bit (0 or 1) is thus stored as long

C/C++ Functional Declaration

```
<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 Functional Declaration

```
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 Functional Declaration

```
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 Functional Declaration

Because there are no 16-bit values in PHP, the 16-bit data is stored as long. The word functions store the result in a long array; if only one unit is read, the result is saved as a single long if the variable is not yet an array. The reading and writing of words (16 bits) is always done with a sign. That the value is interpreted as a 16-bit integer. If the values are handled as unsigned 16-bit, then observe the optional parameter 7 (bSigned). The ByteFunctions store the result basically as a string, however, if you want to simply call the values as a long array, you can use bLong=1 to store the result as a long array.

```
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
 IPS7RdMLInt (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';
```

Read mixed data areas

IPS7RdMulti

Name	Name (PHP)	Description / Purpose
IPS7RdMulti	ips7_rdmulti	Performs a mixed read job. For the jobs, a list / array of records / structures of the type “IPS7_RQ_MULTI” is filled out and the function is passed on. The function sorts and bundles these jobs and optimizes them. **ATTENTION!** The reading sequence is often different than the order of the handed list

IPS7RdMulti automatically performs data conversion between S7 and PC. The conversion is to be compared with **data type casting** in the programming language. The bit width of the dataType in the PC must be equal to or greater than that in the PLC.

Explanation of PHP

The caller has to provide a correspondingly large target area.

In PHP the variables are basically not assigned to a fixed datatype. The determination of the data type therefore takes the expansion module. Basically, the target or source variable for the read / write buffer (= parameter 6) must be passed as a reference. So use the “&” character. E.g. $Res = ips7_rdplcw ($Ref, ord (“M”), 0, 0, 2, &$Values);

This means: For example, Two 16-bit values (total 32 bits) are read and converted into double values (64 bits) in the PC, two double values (total 128 bits) must also be provided in the PC.

The convertation looks as in the example:

Value in the PLC	Value in the PC
6	6.0
1	1.0

The PLC and PC data types are selected in the request. Coded using constants as described below.

IPS7RdMultiCalcPacketCnt

Name	Name (PHP)	Description / Purpose
IPS7RdMultiCalcPacketCnt	ips7_rdmulticalcpacketcnt	The return value is the number of communication packets required to read all multiread jobs. This can be used to check whether all desired variables can be read in one piece. This is the case if the return value == 1. Please note the return value: **> = 0** the number of packets **<0** an error has occurred, evaluation as below

Return Values

Return Values Read/Write

When the IPS7RdMulti function is used, the return value is stored in the Struct element “Res”.

Value	Error description	Reaction
0	OK	Evaluate data
2	Block or data area does not exist, e.g. Access to DB that is not present or too small	Check whether the desired data area is present in the PLC
-1	Time overflow, desired PLC apparently not or no longer present	The driver automatically sets up additional write and read jobs. Possibly the timeout times extend in particular the Connect timeout time

Value	Error description	Reaction
-2	The maximum number (256) of the possible connections is reached	Close unnecessary connections
-3	Can occur with Close. No Open was executed with the specified reference	Check your source code to see if the variable for the reference has not been overwritten. Or a close has already been executed for this reference
-5	General Error occurred	Check if network is properly installed in the PC: TCP/IP activated? Winsocket installed?
-6	Destination CPU not found	Rack or Slot number is wrong. There is no connection to this slot anymore. Check CP configuration
-7	Socket error occurred	Call IPS7GetSockErr and evaluate the error
-8	memory error	Requested memory in the PC is not available
-9	out of range	e.g. Timer > 9990000 ms
-10	Data type not allowed or not supported	Check if the code is correct for datatype
-11	The specified PC data type is not possible for the specified PLC data range	This may e.g. If the counter is to be accessed and the PC data area is specified as BYTE. Remedy: Change the data area in the PC
-20	The specified memory in the PC is too small (for example, array is too small), can only occur with .net or PHP	Data area in the PC enlarge or correct
-21	Only .Net! An Open has already been executed for this instance of the class	You may want to call Close
-31	**Only MultiRead:** PC and S7-datatype are in the wrong relation, e.g. PC = BYTE PLC = word	Adjust the PC data area
-32	**Only MultiRead:** S7 provides incorrect number of data for the specified datatype	
-88	**Only MultiRead:** The corresponding task has not yet been processed	
-99	The reference number is invalid	Have you called IPS7Open?
4660	Demo time has expired	by full version

C/C++ Functional Declaration

```
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 Functional Declaration

```
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';
```

The structure of the request record / structure for IPS7RdMulti

<table>
<tr><th>Name</th><th>Type</th><th colspan="3">Description / Purpose</th></tr>
<tr><td>DataArea</td><td>32-Bit unsigned</td><td colspan="3">The selection of the memory area in the PLC (DB, input, output, flag), which is to be edited:
D = 68 dec. Stands for data block
E = 69 dec. Stands for inputs
A = 65 dec. Stands for outputs
M = 77 dec. Stands for flag
T = 84 dec. Stands for Timer
(only possible with double word functions)
The timers are stored in the PLC with time base and value in the BCD format. In order to process this format immediately in the PC, the driver performs an automatic conversion in milliseconds. The smallest possible raster is 10 ms. When writing to the PLC, the driver automatically selects a suitable time base. This can lead to rounding.
The time range is from 0 to 9990000 ms
Z = 90 dec. Stands for counter
The counters are also stored in the PLC BCD-coded. The counter values range from 0 - 999</td></tr>
<tr><td rowspan="12">DataType</td><td rowspan="12">32-Bit unsigned</td><td colspan="3">data type in the PLC</td></tr>
<tr><td>Name</td><td>Value</td><td>SPS-data type / bit width</td></tr>
<tr><td>IPS7_BIT</td><td>0</td><td>A 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</td></tr>
<tr><td>IPS7_INT</td><td>3</td><td>INT 16 Bit signed</td></tr>
<tr><td>IPS7_DWORD</td><td>4</td><td>DWORD 32 Bit unsigned integer</td></tr>
<tr><td>IPS7_DINT</td><td>5</td><td>long 32 Bit signed integer</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 the S7</td></tr>
<tr><td>IPS7_COUNTER</td><td>8</td><td>Counter in the S7</td></tr>
<tr><td>IPS7_LINT</td><td>9</td><td></td></tr>
<tr><td>IPS7_ULINT</td><td>10</td><td>UINT 64 Bit unsigned integer</td></tr>
</table>

<table>
<tr><td>DBNr</td><td>32-Bit unsigned</td><td colspan="3">Data block number, this is only used for type 'D'. Otherwise the value 0</td></tr>
<tr><td>Cnt</td><td>32-Bit unsigned</td><td colspan="3">Number of data elements to be read / written</td></tr>
<tr><td>Start</td><td>32-Bit unsigned</td><td colspan="3">Start byte in the PLC</td></tr>
<tr><td>StartBit</td><td>32-bit unsigned</td><td colspan="3">The number of the first bit in the PLC, values (0 - 7), is only used for bit accesses, otherwise 0</td></tr>
<tr><td rowspan="11">PCDataType</td><td rowspan="11">32-Bit unsigned</td><td colspan="3">data type in the PC</td></tr>
<tr><td>Name</td><td>Value</td><td>data type in the PLC</td></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 floating point in the PC (float)</td></tr>
<tr><td>PC_DOUBLE</td><td>6</td><td>64 Bit floating point in the 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>
<tr><td>Result</td><td>32-Bit unsigned</td><td colspan="3">Result for this order. The individual values can be found below in the description: Return values for Read / Write functions</td></tr>
<tr><td>UserData_0</td><td>32-Bit unsigned</td><td colspan="3" rowspan="2">This entry can be used by the caller to access its own information. The value is neither evaluated nor changed by the driver. Thus, e.g. Additional information about the properties of the variable can be stored in the PC</td></tr>
<tr><td>UserData_1</td><td>32-Bit unsigned</td></tr>
<tr><td>Data</td><td>32-Bit Pointer</td><td colspan="3">The actual pointer to the memory area in the PC for this job</td></tr>
<tr><td>pUserData</td><td>32-Bit Pointer</td><td colspan="3">Such as UserData_0 and UserData_1, but as a pointer</td></tr>
</table>

IPS7RdMulti Sample C/C++

```
// Small function for initializing a single request
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 Requests;

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

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

    // read from EB0 20 Bytes and put it down [20]
    InitRq (&Rq[Cnt++], 'E', IPS7_BYTE, PC_BYTE, , , , 20, &EBytes[20]);

    //read from MB 20 10 words
    InitRq (&Rq[Cnt++], 'M', IPS7_WORD, PC_WORD16, , 20, , 1, MWords);

    //read DB10 from data byte 0 150 words
    InitRq (&Rq[Cnt++], 'D', IPS7_WORD, PC_WORD16, 10, , , 150, DB10Words);

    //read DB20 from data byte 0 150 words put these but as double in the PC
    InitRq (&Rq[Cnt++], 'D', IPS7_WORD, PC_DOUBLE, 10, , , 150, DB10WordsAsDouble);

    //read DB20 from data byte 6 60 real values and place these as float values in the PC
    InitRq (&Rq[Cnt++], 'D', IPS7_REAL, PC_FLOAT, 10, 6, , 60, DB20RealAsFloat);

    //read from Timer 5 10 Timer and set it as int
    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 == ) // OK
       {
           //.. evaluate and process data
       }
       else
       {
            // handle exception
       }
    }
    }
}
```

Read diagnostic buffer

IPS7ReadDiagBuffer

Name	Name (PHP)	Description/purpose
IPS7ReadDiagBuffer		Read diagnostic buffer

Call parameter

Nr	data type	data type (PHP)	Name	Function
1	32-Bit unsigned	long	Ref	The reference of the connection that was generated with IPS7Open. Used to identify the connection internally
2	S7_EVENTENTRY Pointer	mixed	pEvent	Pointer on the structure of the event, see above. Data are written to each other in a large format (16-bit Word, 8-Bit Priority ...)
3	32-Bit		MaxEvents	Number of events to be read
4	32-Bit		pRxEventCnt	Pointer to the actual count of selected events

C/C++ Functional Declaration

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 Functional Declaration

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

Read System Information

IPS7GetPLCTime

Name	Name (PHP)	Description / Purpose
IPS7GetPLCTime	ips7_getplctime	Reads the system time of the PLC

Call parameter

Nr	data type	data type (PHP)	Name	Function
1	32-Bit unsigned	long	Ref	The reference of the connection that was generated with IPS7Open. Used to identify the connection internally
2	32-Bit Pointer to 32 Bit value	long	Year	Pointer to the 32-bit value for the year (after read you will find for example 2012)
2	32-Bit Pointer to 32 Bit value	long	Month	Pointer to the 32 bit value for the month (1-12)
2	32-Bit Pointer to 32 Bit value	long	Day	Pointer to the 32 bit value for the day (1 -31)
2	32-Bit Pointer to 32 Bit value	long	Hour	Pointer to the 32 bit value for the hour (0 – 23)
2	32-Bit Pointer to 32 Bit value	long	Min	Pointer to the 32 bit value for the minute (0 – 59)
2	32-Bit Pointer to 32 Bit value	long	Sec	Pointer to the 32 bit value for the second (0-59)
2	32-Bit Pointer to 32 Bit value	long	Ms	Pointer to the 32 bit value for the millisecond (0-999)

C/C++ Functional Declaration

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

Delphi / Pascal Functional Declaration

```
{ -- 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';
```

The same return values apply as for the Read / Write functions!

IPS7SetPLCTime

Name	Name (PHP)	Description / Purpose
IPS7SetPLCTime	ips7_setplctime	Set the system time of the PLC

Call parameter

Nr	data type	data type (PHP)	Name	Function
1	32-Bit value	long	Ref	The reference of the connection that was generated with IPS7Open. Used to identify the connection internally
2	32-Bit value	long	Year	32 Bit value for the year (1999 – 21xx)
2	32-Bit value	long	Month	32 Bit value for the month (1-12)
2	32-Bit value	long	Day	32 Bit value for the day (1 -31)
2	32-Bit value	long	Hour	32 Bit value for the hour (0 – 23)
2	32-Bit value	long	Min	32 Bit value for the minute (0 – 59)
2	32-Bit value	long	Sec	32 Bit value for the second (0-59)
2	32-Bit value	long	Ms	32 Bit value for the milliseconds (0-999)

C/C++ Functional Declaration

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

Delphi / Pascal Functional Declaration

```
{ -- 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';
```

The same return values apply as for the Read / Write functions!

IPS7GetPLCName

IPS7GetModuleName

IPS7GetPlantIdName

IPS7GetCopyrightEntry

IPS7GetModuleSNr

IPS7GetModuleTypeName

IPS7GetMMCSNr

IPS7GetLocationDesignation

IPS7GetOEMId

Various system information are available in the PLC. Most of them, however, only start at FW 2.2 of the S7. The following information can be received in a string.

Name	Name (PHP)	Description / Purpose
IPS7GetPLCName	ips7_getplcname	Reads the name of the PLC
IPS7GetModuleName	ips7_getmodulename	Reads the name of the module (e.g. CPU 315-2 DP/PN - Test)
IPS7GetPlantIdName	ips7_getplantidname	Reads the plant name of the PLC
IPS7GetCopyrightEntry	ips7_getcopyrightentry	Reads the Copyright entry of the PLC
IPS7GetModuleSNr	ips7_getmodulesnr	Reads the serial number of the PLC
IPS7GetModuleTypeName	ips7_getmoduletypename	Reads the type name of the PLC (e.g. CPU 315-2 PN/DP)
IPS7GetMMCSNr	ips7_getmmcsnr	Reads the serial number of the MMC
IPS7GetLocationDesignation	ips7_getlocationdesignation	Reads the given location description of the PLC
IPS7GetOEMId	ips7_getoemid	Reads the OEM-ID of the CPU

Call parameter

Nr	data type	data type (PHP)	Name	Function
1	32-Bit unsigned	long	Ref	The reference of the connection that was generated with IPS7Open. Used to identify the connection internally
2	32-Bit Pointer auf C-String	string	strInfo	Pointer of the String, which receives the information. Must be at least 64 characters

C/C++ Functional Declaration

```
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 Functional Declaration

```
{ -- 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';
```

The same return values apply as for the Read / Write functions!

Program samples

C/C++

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

//Call ByteFunction e.g. read DB 10, from DW0, 10 words
IPS7RdW (Ref,'D',10, , 10, WordBuffer);

//Call ByteFunction e.g. read MB 0 , 10 bytes
IPS7RdB (Ref,'M' ,, , 10, ByteBuffer);
```

After successful call, the data is be stored as:

PC	=	PLC
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;

  //Call WordFunction e.g. read DB 10, from DW0, 10 words
  IPS7RdW (Ref, LongWord ('D'), 10, , 10, @WordBuffer[]);

  //Call ByteFunction e.g. read MB 0 , 10 bytes
  IPS7RdB (Ref,'M' ,, , 10, @ByteBuffer[]);
```

VB

```
  Dim ByteBuffer (0 to 511) as Byte;
  Dim WordBuffer (0..511) as Word;

  //Call WordFunction e.g. read DB 10, from DW0, 10 words
  IPS7RdW (Ref, 68, 10, 0, 10, WordBuffer(0))

  //Call ByteFunction e.g. read MB 0 , 10 bytes
  IPS7RdB (Ref, 77, 0, 0, 10, ByteBuffer(0));
```

After successful call, the data is be stored as:

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

Socket error list

Win Socket error list

Release Notes

Release Notes

IP-S7-LINK version history

Version 1.73 from 16.6.15

- Lint and ULInt data types implemented

Version 1.72.93 from 25.3.15

- LinkTo - correction

Version 1.71.92 from 8.12.14

- S7-Logo IPS7RdReal

Version 1.71.92 from 8.12.14

- S7-Logo IPS7RdReal

Version 1.70.91. from 2.7.14

- Periphery access implemented

Version 1.69.90 from 04.06.14

- internal memory management optimized for MultiRead
- for Bit/Byte absolutely byte aligned, otherwise 4-Byte aligned
- the first time MultiRead access with large packages it came to memory conflicts
- this effect are since V 1.67.88

Version 1.68.89 from 09.05.14

- IPS7ReadDiagBuffer implemented

Version 1.67.88 from 29.04.14

- MultiRead with Bit, read with the value 0

Version 1.66.87 from 04.04.14

- MultiRead error(2) with String, if the string entry was fragmented
- MultiRead memory optimised

Version 1.65.86 from 31.03.14

- MultiRead error with String, if the string entry was fragmented by 2 orders

Version 1.64 from 04.12.13

- MultiRead unfavorable combination of print jobs generated error

Version 1.63

- MultiRead the Result was not passed properly in the individual orders

Version 1.62.83 from 09.09.13

- adjust communication with Logo-PLC Logo/S7200

Version 1.62.82 from 08.07.2013

- IPS7RdStr error corrected at MultiRead

Version 1.61.80 from 19.06.2013

- IPS7RdStr, IPS7WrStr implemented
- String in IPS7RdMulti implemented

Version 1.60.78 from 02.04.2013

- PDU-Size optimised, access with maximum size
- inserted accessing Logo PLC → Access Mode 3 at IPS7OpenEx

Version 1.58 from 17.07.12

- only for the ARM-Version
- reading wrong real values because of the ALIGNMENT

Version 1.57

- IPS7RdW
- Offset at Start != 0 was calculated with the factor 4 and not 2

Version 1.56 from 21.05.12

- Lite Version for private use

Version 1.55 from 12.04.12
new functions:

- 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 changed because of WINCE

Version 1.54 from 05.04.12

- LPCSTR changed in LPCTSTR because of WINCE
- S7-200 / Logo - access implemented, with Logo always use TSAP 02.00
- license only for S7-200 Logo generated, can be combined with S7-LAN-LINK
- with S7-LAN-LINK socket wasn´t closed, when when S7-LAN or S5-LAN where not available
- Set / read the PLC-time implemented
- 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 from 22.02.12

- RdMultiSimplex, if e.g. while debugging, the IP-connection was reset, was possibly an undefined positive error returned

Version 1.52 from 30.01.12

- IPS7RdBit, read Bit since version 1.51 always a Bit 0 was read
- inserted in the source changes for IAR Compiler

Version 1.51 from 30.11.11
with 1.50 MultiRead error (PDUSize)

Version 1.50 from 25.11.11

- MultiRead with Counter was incorrect
- adjustments for Embedded Systems
- read routines optimised

Version 1.49 from 11/11

- In Linux for Critical Sections now pthreads are used

Version 1.48 29.11.10

- for S7-LAN-LINK also S5-LAN integrated
- under Linux for generate shared Libs with Compileroption -“fPIC” compiled

Version 1.47 from 24.09.10

- Support for ARM-Processors implemented (Alignment-Trap fixed)

Version 1.46 from 08.09.10

- Bit-accesses with MultiRead function
- At read from Bit's with Start-Bit address > 0 at some PLC's a error occurred
- Data area not reachable (Bit address was passed with byte access)

Version 1.45 from 07.09.10

- PDU-Size for CPU 400 etc. optimised

Version 1.44 from 18.08.10

- Support / identification S5-LAN with S7-TCP/IP
- with MultiRead-access to S5-LAN the real conversion is not carried out in S5-LAN but the driver. The driver needs to know if a S5-LAN is connected.
- since S5-LAN ++ V 1.20 the driver can automatically detect it
- at modules < 1.20, at function IPS7OpenEx use AccessMode “20”

Version 1.43 from 22.07.10

- MultiReadzugriff: demo version implemented
- sizeof - corrected comparisons

Version 1.42 from 14.07.10

- MultiRead access: before execution of the first read-contract an error (e.g. timeout etc.) was commissioned as a result -88 / order not edited set now the actual error value is there.
- Linux: no connect are possible, so error -5 (general error) was set, now the socket error will be set,

so the original problem can be checked with error number or strerrno()

Version 1.41 from 13.07.10

- MultiRead access: read from outputs was not supported

Version 1.40 from 07.07.10

- MultiRead access: at blocks > 220 Byte it came to overrides
- .Net MultiRead access: because of the Garbage Collection the variables can be shifted accidentally, the access way had to be revised
- Programs in C# or VB.Net should use the function RdMultiBuffered. More you will find in the .chm file!

Version 1.39 from 17.06.10

- MultiRead access: new function " IPS7RdMultiCalcPacketCnt " returns the count of packets for read all of the given MultiRead-assignments

Version 1.38 from 24.05.10

- MultiRead access: Int16 and Int32 (signed) was on collection of PC_INT32 and PC:DWORD was converted in unsigned now conversion will be done right to a signed-value

Version 1.37 from 18.05.10

- .Net-Interface: for MultiRead access, Int16 and Int32 (signed) access implemented.
- .Net-Interface: for MultiRead access, at use of arrays, size will be checked, is the array to small, error -20 will be generated
- MultiRead access: Conversion of bits in DWORD or Real resulted in the protection violation

Version 1.36 from 03.05.10

- IPS7RdMulti, copy error protection violation occured
- Demo for Delphi updated, Outfit also like C++/C#/VB.net

Version 1.35 from 14.04.10
New functions:

- IPS7Connect – takes explicit IP connection from
- IPS7GetConnectStatus – checks the IP connection state
- IPS7SetKeepAlive – sets individual keepalive times
- IPS7RdMulti – reads various data areas in one piece from the PLC

Version 1.34 from 02/10

- between version

Version 1.33 from 02.02.10

- when reading the timer it could lead to erroneous results when the timer is running. The base was miscalculated.

Version 1.32 from 27.08.09

- Read / write the real / float values with S7 code as well as an operation to S5 Lan++ is possible with real values.

Version 1.31 from 20.08.09

- in .Net-Assemblies Strong-Names implemented

Version 1.30 from 17.07.09

- V 1.29 ported to Linux, S7-LAN-Link have now the same interface as IP-S7-LINK, but runs only with S7-LAN, so the transition to IP-S7-LINK for the user is a simple

Version 1.29 from 20.04.09

- Assembly Interface for .Net Rd method with 32 Bit Integer was reading Bit, now 32 Bit

Version 1.28 from 09.02.09

- ips7lnk.lib referneced to s7lanlnk.dll, so a link error at VC++ Compiler, or a message, that S7lanknk.dll couldn´t be found, occurred

Version 1.27 from 26.08.08

- Additional tests for valid memory inserted. improved treatment of Critical Sections

Version 1.26 from 12.08.08

- IPS7WrBit was not exported

Version 1.25 from 16.07.08
When calling IPS7Open with multiple threads, there were occasional traps. Fixed problem with Critical Section

Version 1.24 from 10.07.08

- .Net and PHP implemented

Version 1.23 from 02.06.06

- access for Routing over SubnetID implemented

Version 1.22 from 14.10.05

- Operation with slot PLC and soft PLC did not worked (FAST ACK was not properly processed)
- Problems at receiving fragmented data

Version 1.21 from 04.08.05

- operation with CP 243 (S7 200) implemented

Version 1.20 from 08.07.05

- special version for university limited to DB1 and DB2 generated

Version 1.19 from 20.05.05

- The change of 1.18 was intended only for IPS7RdPlcW and IPS7WrPlcW in blocks > 111 words. With 1.18 the function IPS7RdW and IPS7WrW in these areas no longer worked

Version 1.18 from 17.05.05

- wrong calculation of start address with IPS7RdW and IPS7WrW at blocks > 111 words

- The start address of the sequence blocks were incorrectly determined

Version 1.17 from 04.03.05

- New function added IPOpenPG so it is possible to connect through the PG-PLC channel
- Useful if no OP channels are free
- For word-reading and writing with unsigned start addresses function IPS7RdPlcW and S7WrPlcW

Version 1.16 from 01.12.04

- We basically used the PG channel, from now the HMI / OP channel is used

Version 1.15 from 11.11.04

- Timeout monitoring inserted for reception of the entire block. Maybe, it caused problems with Berthel PLC

Version 1.14 from 01.08.04

- Maximum number of open channels increased to 256

Version 1.13 from 26.07.04

- Maximum number of open channels increased to 128

Version 1.12 from 07.07.04

- In order to achieve higher performance of the Nagle algorithm has been turned off. That TCP_NODELAY was set to 1

Version 1.11 from 13.05.04

- At 'close' Windows socket has the required port partly unblocked only after 20 minutes
- there by the effect was that only after restarting the PLC or the PC, a new connection to the PLC could be established

Version 1.10 from 13.05.04

- Maximum count opened channels raised up to 64

Version 1.09 from 19.03.04

- allowed writing inputs

Version 1.08 from 23.01.02

- When trying to connect with users who were not in the network, handles were occupied in the system and no longer released
- Problem fixed!

Version 1.07 from 12.12.01

- fixed DLL-call from multiple applications

Version 1.06 from 19.09.01

- Timer / Counter function implemented
- double word function implemented
- real access (Floating point arithmetic) implemented

Version 1.05 from 12.07.01

- Bit- read and write function implemented

Version 1.04 from 19.06.01

- Bytewise read and write in DB implemented

Version 1.03 from 17.05.01

- When reading blocks, which are divisible by 222
- e.g. 444 Byte or 222 words
- e.g. 666 Byte or 333 words ..
- the last 222 bytes has not been read, no error message was created
- When writing blocks by 212 divisible
- e.g. 424 Byte or 212 words
- e.g. 636 Byte or 318 words ..
- the last 212 bytes has not been written, there was no error message

Version 1.02 from 29.01.01

- added error number -6
- Function IPS7GetSockErr implemented (see documentation)

Version 1.01 from 21.12.00

- read of block areas that have not been exist were confirmed as OK, although the specified range does not exist
- this happened with DB > 111 words

Version 1.00 from 14.12.00

Table of Contents