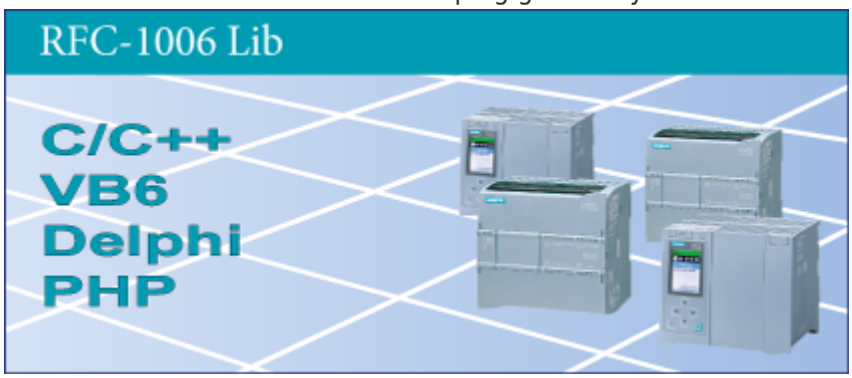
RFC 1006 Lib

RFC 1006 Client- and Server developing goes easy



Software Version 1.40

Operation System

- Windows 10 / 8 / 7 / Vista / XP 32/64 Bit
- Linux x86 32/64 Bit
- Linux ARM / Embedded 32/64Bit

Programming Languages

- C
- C++
- C#
- VB
- VB.net
- Delphi

Versions History - The List of improvements of each Version

Requirements

Hardware

- PC with installed TCP/IP-Protocol and network card

Installation

Windows

Copy the DLL into the directory of your the program. To make the library system-wide available place the DLL into the windows system directory `%SystemRoot%\system32`.

Linux

Link the object file (.o) to your application.

Operation

RFC1006-Lib is a DLL for MS-Windows, which allows connection of a PC to Industrial Ethernet over RFC1006.
With simple features, users can quickly realize RFC 1006 connection with C, C++, Delphi, Visual Basic or Excel.
For communication only the IP address, DSAP, SSAP of the partner is required. RFC1006 uses TCP Port 102 by default.
The Library can be uses to provide as Client or/and Server funtionality.

Function description in detail

Please note: The functions are performed with the standard socket interface, which has the consequence that the function only after the fulfilment of the task returns to the caller. For asynchronous operation call these functions easily from a separate thread on which is responsible for the communication of the system.

Server operation

To operate as a server note the functions:
Rfc1006OpenServer
Rfc1006GetStatus
Rfc1006StartServer

Otherwise the function **Rfc1006Rx** and **Rfc1006Tx** can be used for communication. It is advisable to check with **Rfc1006GetStatus** in a regular interval whether there is a connection and then use only the transmit and receive routines.

The following functions are available:

Initialisation

Rfc1006Open

To initialize the connection, there only memory is prepared. On the first call of the read or write functions the TCP/IP connection / RFC 1006 connection is started automatically.
(Use only when operating as a client!)

Call parameters

Nr.	Memory width	Description	Function
1	Pointer to a “0”-terminated string (C-String, 32-Bit pointer	IPAdr	IP-Address of the PLC in format: xxx.xxx.xxx.xxx. Sample: 192.169.0.100
2	Pointer to a “0”-terminated string (C-String, 32-Bit pointer	SSAP	own SAP. It can be used up to 255 characters.
3	32-Bit unsigned value	SSAPLen	Length of own SAP. A maximum of 255 characters used
4	Pointer to a “0”-terminated string (C-String, 32-Bit Zeiger	DSAP	
5	32-Bit unsigned value	DSAPLen	Length of the SAP partner. A maximum of 255 characters used
6	32-Bit unsigned value	PortNr	Number of TCP/IP ports to be used for this connection. Default is 102. Set to 0 to so is used internally 102
7	32-Bit unsigned value	ConnectTimeout	Timeout in milliseconds for waiting for connection with the partner. 0 means default = 5000 ms (5 sec.) Must be extended if necessary.

C/C++

```
long WINAPI
Rfc1006Open (LPCSTR IPAdr,BYTE *SSAP, DWORD LenSSAP, BYTE *DSAP, DWORD LenDSAP,
             DWORD Port, DWORD ConnectTimeout);
```

Delphi

```
FUNCTION
Rfc1006Open (IPAdr : PAnsieChar; SSAP : Pointer; LenSSAP : LongWord; DSAP : Pointer;
             LenDSAP : LongWord; Port LongWord; ConnectTimeout : LongWord): LongInt;
            stdcall; external 'Rfc1006Lib.dll';
```

VB

```
Declare Function Rfc1006Open& Lib "Rfc1006Lib.dll" (ByVal IPAdr As String, _
                                                     SSAP as Byte, _
                                                     ByVal LenSSAP&, _
                                                     DSAP As Byte,  _
                                                     ByVal LenDSAP&, _
                                                     ByVal Port&, _
                                                     ByVal ConnectTimeout&)
```

Rfc1006OpenServer / Rfc1006OpenExServer

Creates a new connection for server operation. The start of the connection is controlled with **Rfc1006StartServer** or **Rfc1006StartExServer**.
If the server is already running, the new connection will be immediately registered as available.
If the desired port is already in use by an other application, there will be returned the value -7 (Socket error occured). Use **Rfc1006GetSockErr** / **Rfc1006GetSockErrString** to determine the reason.

From version 1.31 “Rfc1006StartExServer” was implemented. Here, the TCP port can be specified for the desired compound. So it is possible to simultaneously receive on different ports connections.
(Use only when operating as a server!)

Call parameters

Nr.	Memory width	Description	Function
1	Pointer to a “0”-terminated String (C-String, 32-Bit Pointer	DstIPAdr	IP address of the client, which must establish a connection to the server. Example: “192.169.0.100”. Is the string empty like “”, each sender address is accepted
2	Pointer to a “0”-terminated String (C-String, 32-Bit Pointer	SSAP	own SAP. A maximum of 255 characters used
3	32-Bit unsigned value	SSAPLen	Length of the own SAP. A maximum of 255 characters used
4	Pointer to a „0“-terminated String (C-String, 32-Bit pointer	DSAP	SAP des Partners. A maximum of 255 characters used
5	32-Bit unsigned value	DSAPLen	Length of the SAP from the partner. A maximum of 255 characters used
6	32-Bit unsigned value	Port	TCP/IP Port (only at Rfc1006OpenExServer)

Return value

The function returns a 32-bit signed value as a return value with the following meaning:

Wert	Error description	Meaning
>= 0	OK	The return is the reference number for this connection and must be used for all other functions as an input parameter Ref.
-2	No more resources.	Reached maximum number of available connections.
-15	The desired port is in use.	Binding couldn't be established.

C/C++

```
long WINAPI
Rfc1006OpenServer (LPCSTR DstIPAdr, BYTE *SSAP, DWORD LenSSAP, BYTE *DSAP, DWORD LenDSAP);

long WINAPI
Rfc1006OpenExServer (LPCSTR DstIPAdr, BYTE *SSAP, DWORD LenSSAP, BYTE *DSAP, DWORD LenDSAP,
DWORD Port);
```

Delphi

```
FUNCTION
Rfc1006OpenServer (DstIPAdr : PAnsiChar; SSAP : Pointer; LenSSAP : LongWord; DSAP : Pointer;
LenDSAP : LongWord): LongInt;
            stdcall; external 'Rfc1006Lib.dll';

FUNCTION
Rfc1006OpenExServer (DstIPAdr : PAnsiChar; SSAP : Pointer; LenSSAP : LongWord; DSAP :
Pointer; LenDSAP : LongWord; Port : LongWord)): LongInt;
  stdcall; external 'Rfc1006Lib.dll';
```

VB

```
Declare Function Rfc1006OpenServer& Lib "Rfc1006Lib.dll"(ByVal DstIPAdr as String, _
                                                         SSAP as Byte, _
                                                         ByVal LenSSAP&, _
                                                         DSAP as Byte, _
                                                         ByVal LenDSAP&)

Declare Function Rfc1006OpenExServer& Lib "Rfc1006Lib.dll"(ByVal DstIPAdr as String, _
                                                         SSAP as Byte, _
                                                         ByVal LenSSAP&, _
                                                         DSAP as Byte, _
                                                         ByVal LenDSAP&_
                                                         ByVal Port&)
```

Rfc1006StartServer

(Use only when operating as a server!)

Starts the server with the connections applied with **Rfc1006OpenServer** / **Rfc1006OpenExServer**.

When server is running connections can be dynamically added and removed by calling **Rfc1006OpenServer** / **Rfc1006CloseServer**.

If the desired port is already in use by another application, value E_RFC1006_SOCKERR (-7) will be returned. Use **Rfc1006GetSockErr** / **Rfc1006GetSockErrString** with `Ref = -1` to determine the reason.

Call parameters

Nr.	Memory width	Description	Function
1	32-Bit value	Port	**ATTENTION** This parameter is internally not used any more. Use **Rfc1006OpenExServer** instead to specify the port of a connection.
2	Pointer to a “0”-terminated String (C-String, 32-Bit pointer	IPBindAdr	IP address of the network card where to run the connection. NULL or an empty string allows for the connection to all NICs.
3	32-Bit value	MaxCon	Maximum number of connections that can maintain the server.

Return value

The function returns a signed 32-bit value as return value with the following meaning:

value	Error description	Meaning
> 0	OK	Server is started.
-10	Server still running.	Rfc1006StartServer already called.
-11	Server couldn´t be started	Errors occurred when starting the server threads.
-15	The desired port is in use.	Binding couldn't be established.

C/C++

```
long WINAPI
Rfc1006StartServer (int Port, LPCSTR BindIPAdr, int MaxCon);
```

Delphi

```
FUNCTION
Rfc1006StartServer (Port : LongWord; BindIPAdr : PAnsiChar; MaxCon : LongWord): LongInt;
            stdcall; external 'Rfc1006Lib.dll';
```

VB

```
Declare Function Rfc1006StartServer& Lib "Rfc1006Lib.dll"(int Port, _
                                                          ByVal BindIPAdr as String, _
                                                          int MaxCon)
```

Deinitialisation

Rfc1006StopServer

Stops the Server.
(Use only as Server!)

Return value

The function returns a 32-bit signed value as a return value with the following meaning:

value	Error description	meaning
> 0	OK	Server stopped.
-12	Server wasn´t started. StartServer was not called.	

C/C++

```
long WINAPI
Rfc1006StopServer (void);
```

Delphi

```
FUNCTION
Rfc1006StopServer (): LongInt;
            stdcall; external 'Rfc1006Lib.dll';
```

VB

```
Declare Sub Rfc1006StopServer& Lib "Rfc1006Lib.dll"  ()
```

Rfc1006Close

to initialize the connection, memory is released and the TCP/IP connection is disconnected.

Call parameters

Nr.	Memory width	Description	Function
1	32-Bit unsigned value	Ref	The connection reference, which was generated with RFC1006 Open. Used for internal identification of the connection.

Return value

The function returns a 32-bit signed value as a return value with the following meaning:

Value	Error description	Meaning / Reaction
0	OK	Memory is released again and connection, if any, closed
-3	With the specified reference no RFC1006Open was conducted	Did you called Rfc1006Open?
-99	The reference number is invalid	————————

C/C++

```
long WINAPI
Rfc1006Close (long Ref);
```

Delphi

```
FUNCTION
Rfc1006Close (Ref : LongInt) : LongInt;
            stdcall; external 'Rfc1006Lib.dll';
```

VB

```
Declare Function  Rfc1006Close& Lib "Rfc1006Lib.dll" (ByVal Ref&)
```

Rfc1006CloseAll

to initialize the connection, memory is released and the TCP/IP connection is disconnected.

C/C++

```
void WINAPI
Rfc1006CloseAll (void);
```

Delphi

```
PROCEDURE
Rfc1006CloseAll ();
            stdcall; external 'Rfc1006Lib.dll';
```

VB

```
Declare Sub Rfc1006CloseAll Lib "Rfc1006Lib.dll" ()
```

Receiving and transmitting

Rfc1006Rx / Rfc1006Tx

Function	Description / Purpose
Rfc1006Rx	Attempts to receive data via RFC 1006 yet there is no connection to the partner, so we built this earlier (Only in client mode). If the server operating no connection, the function returns immediately with -1.
Rfc1006Tx	Sends data on RFC 1006 to the partner. yet there is no connection to the partner, so we built this earlier (Only in client mode). If the server operating no connection, the function returns immediately with -1.

Call parameters

The read and write functions have the same input parameters:

Nr.	Memory width	Description	Function
1	32-Bit unsigned value	Ref	The reference of the connection, which was generated with RFC1006Open. Used for internal identification of the connection
2	32-Bit Address	Buffer	The address of the Source or target memory in the PC.
3	32-Bit unsigned Return value	Cnt	At Rx, the maximum number of data bytes to be received. When Tx Number of sent data bytes.
4	32-Bit signed value	Timeout	Timeout in ms for sending and receiving 0 = do not wait -1 = wait until everything sent until we receive anything

Return value

The function returns a 32-bit signed value as a return value with the following meaning:

value	Error description	Reaction
>=0	Count of the sent / received Databytes	
-1	Timeout, desired partners obviously do not or no longer exists	deposed Simply more read and write requests, the driver automatically sets up the connection. Possibly the timeouts in particular extend the Connect Timeout.
-5	General Errror	Check whether the network installed in the PC properly: TCP/IP enabled? Winsocket installed?
-6	Partner not found	Portnumber in Rfc1006Open is not correct or there is no connection to this port free. Check configuration

value	Error description	Reaction
-7	Socket error occurred	Call RFC1006GetSockErr and evaluate errors
-13	Not a client connected to the server.	Client connect, check any SAP or DSAP.
-14	The size of the receive buffer is too small.	Enlarge receive buffer
-99	Reference invalid	Did you called Rfc1006Open?
-1234	Demo evaluation time expired	Buy the full version or restart

C/C++

```
long WINAPI
Rfc1006Rx (long Ref, void *Buf, DWORD MaxCnt, long RxTimeout);

long WINAPI
Rfc1006Tx (long Ref, void *Buf, DWORD Cnt, long Timeout);
```

Delphi

```
FUNCTION
Rfc1006Rx (Ref : LongInt; Buf : Pointer; MaxCnt : LongWord; RxTimeout : LongWord): LongInt;
            stdcall; external 'Rfc1006Lib.dll';

FUNCTION
Rfc1006Tx (Ref : LongInt; Buf : Pointer; Cnt : LongWord; Timeout : LongWord): LongInt;
            stdcall; external 'Rfc1006Lib.dll';
```

VB

```
Declare Function Rfc1006Rx& Lib "Rfc1006Lib.dll"  (ByVal Ref&, _
                                                    Buf as Byte, _
                                                    DWORD MaxCnt, _
                                                    long RxTimeout)

Declare Function Rfc1006Tx& Lib "Rfc1006Lib.dll"  (ByVal Ref&, _
                                                    Buf as Byte, _
                                                    ByVal Cnt&, _
                                                    ByVal Timeout&)
```

Connection

Rfc1006GetSockErr

Returns the last Socket error

Call parameters

Nr.	Memory width	Description	Function
1	32-Bit unsigned value	Ref	The connection reference, which was generated with RFC1006Open. Used for internal identification of the connection

Return value

The function returns a 32-bit signed value as a return value with the following meaning:

Value	Error description	Meaning / Reaction
0	OK	No error occurred
-3	With the specified reference no RFC1006 Open was conducted	Have you called Rfc1006Open?
-99	Reference number invalid	————————
Other	Socket error	Explanation see list below

C/C++

```
long WINAPI
Rfc1006GetSockErr (long Ref);
```

Delphi

```
FUNCTION
Rfc1006GetSockErr (Ref : LongInt) : LongInt;
            stdcall; external 'Rfc1006Lib.dll';
```

VB

```
Declare Function Rfc1006GetSockErr& Lib "Rfc1006Lib.dll" (ByVal Ref&)
```

Rfc1006GetSockErrString

With this function you get the the socket error readable in text form.

Aufrufparameter

Nr.	Memory width	Description	Function
1	32-Bit value	SockErr	Socket error number returned by **Rfc1006GetSockErr**.
2	Pointer to a String (C-String)	ErrStr	Pointer of C-String, where text is to be stored.
3	32-Bit unsigned value	MaxLen	max length of “ErrStr”.

Return vaue

returns the pointer to C-String “ErrStr”.

C/C++

```
const char * WINAPI
Rfc1006GetSockErrString(long SockErr, LPSTR ErrStr, DWORD MaxLen);
```

Delphi

```
FUNCTION
Rfc1006GetSockErrString (SockErr : LongInt; ErrStr : PAnsiChar; MaxLen LongWord) : PAnsiStr;
            stdcall; external 'Rfc1006Lib.dll';
```

VB

```
Declare Function Rfc1006GetSockErrString& Lib "Rfc1006Lib.dll" (ByVal SockErr&, _
                                                    ByVal ErrStr As String, _
                                                    ByVal MaxLen&) as String
```

Rfc1006GetStatus

Returns the status of the connection

Call parameters

Nr.	Memory width	Description	Function
1	32-Bit unsigned value	Ref	The connection reference, which was generated with RFC1006Open. Used for internal identification of the connection

Return value

The function returns a 32-bit signed value as a return value with the following meaning:

Value	Error description	Meaning / Reaction
0	Connection does not exist	
1	Connection exist	
-3	With the specified reference no RFC1006Open was called	Have you called Rfc1006Open?
-99	Reference number invalid	————————
Other	Socket error	Explanation see list below.

C/C++

```
long WINAPI
Rfc1006GetStatus (long Ref);
```

Delphi

```
FUNCTION
Rfc1006GetStatus (Ref : LongInt) : LongInt;
            stdcall; external 'Rfc1006Lib.dll';
```

VB

```
Declare Function Rfc1006GetStatus& Lib "Rfc1006Lib.dll"(ByVal Ref&)
```

Rfc1006Connect

Executes the connection to the partner from

Call parameters

Nr.	Memory width	Description	Function
1	32-Bit unsigned value	Ref	The connection reference, which was generated with RFC1006Open. Used for internal identification of the connection

Return value

The function returns a 32-bit signed value as a return value with the following meaning:

Value	Error description	Meaning / Reaction
0	Connection couldn´t be established.	
1	Connection established.	
-3	With the specified reference no RFC1006Open was called	Have you called Rfc1006Open?
-99	Reference number invalid	————————

Value	Error description	Meaning / Reaction
Other	Socket error	Explanation see list below.

C/C++

```
long WINAPI
Rfc1006Close (long Ref);
```

Delphi

```
FUNCTION
Rfc1006Connect (Ref : LongInt) : LongInt;
            stdcall; external 'Rfc1006Lib.dll';
```

VB

```
Declare Function Rfc1006Connect& Lib "Rfc1006Lib.dll"(ByVal Ref&)
```

Rfc1006Disconnect

Disconnect the connection of the partner.

Call parameters

Nr.	Memory width	Description	Function
1	32-Bit unsigned value	Ref	The connection reference, which was generated with RFC1006Open. Used for internal identification of the connection

Return value

The function returns a 32-bit signed value as a return value with the following meaning:

value	Error description	Meaning / Reaction
0	Connection closed.	
1	Connection not closed	
-3	With the specified reference no RFC1006Open was called	Have you called Rfc1006Open?
-99	Reference number invalid	————————
Other	Socket error	Explanation see list below.

C/C++

```
long WINAPI
Rfc1006Disconnect (long Ref);
```

Delphi

```
FUNCTION
Rfc1006Disconnect (Ref : LongInt) : LongInt;
            stdcall; external 'Rfc1006Lib.dll';
```

VB

```
Declare Function Rfc1006Disconnect& Lib "Rfc1006Lib.dll"(ByVal Ref&)
```

Rfc1006SetFastAck

Activates/deactivates FastAcknowlege answer, default: deactivated

Call parameters

Nr.	Memory width	Description	Function
1	32-Bit unsigned value	Ref	The connection reference, which was generated with RFC1006Open. Used for internal identification of the connection
2	32-Bit unsigned value	Mode	1 = on 0 = off

C/C++

```
long WINAPI
Rfc1006SetFastAck (long Ref, DWORD Mode);
```

Delphi

```
FUNCTION
Rfc1006SetFastAck (Ref : LongInt; Mode : LongWord): LongInt;
            stdcall; external 'Rfc1006Lib.dll';
```

VB

```
Declare Function Rfc1006SetFastAck& Lib "Rfc1006Lib.dll" (ByVal Ref&, _
                                    ByVal Mode&)
```

Rfc1006SetKeepAlive

Set custom TCP / IP keepalive time for the connection specified with Ref. Must be used only if the standard value not to apply.

You should perform this function immediately after the “open” call.

Call parameters

Nr	Data type	Data type (PHP)	Name	Function
1	32-Bit value unsigned	long	Ref	The connection reference, which was generated with RFC1006Open. Used for internal identification of the connection
2	32-Bit value unsigned	long	AliveTime	When there is no traffic on the TCP/IP connection within the time “Alive Time” (ms) instead, as a keep-alive message is sent to check the connection. If an error is found during this check, the IP stack sends a next keepalive message within the time “AliveInterval” (ms). This is several times repeated within the time **AliveInterval** (in Windows 6 times). The process was not successful, the connection is terminated.
3	32-Bit value unsigned	long	AliveInterval	The interval in ms, in which keepalive messages repeatedly. This is active when a sending / receiving a keepalive telegram has occurred.

Return value

The Function IPS7SetKeepAlive returns a 32-Bit signed value as return value with following meaning:

value	Error description
0	Put of the values was successful.
< 0	Setting the Keep Alive time could not be executed.

C/C++

```
long WINAPI
Rfc1006SetKeepAlive (long Ref, DWORD AliveInterval, DWORD AliveTime);
```

Delphi

```
FUNCTION
Rfc1006SetKeepAlive (Ref : LongInt; AliveInterval : LongWord; AliveTime : LongWord):
LongInt;
            stdcall; external 'Rfc1006Lib.dll';
```

VB

```
Declare Function Rfc1006SetKeepAlive& Lib "Rfc1006Lib.dll"(ByVal Ref&, _
                                        ByVal AliveInterval&, _
                                        ByVal AliveTime&)
```

Rfc1006TxUnlocked

Rfc1006TxUnlockedsends data to the partner without the connection data to entice. This feature can be found at MultiThreading application when a receive-thread the connection just blocked.
CAUTION! The caller must ensure that during the execution of the object remains open with the specified reference!
The function and return values as Rfc1006Tx

C/C++

```
long WINAPI
Rfc1006TxUnlocked (long Ref, void *Buf, DWORD Cnt, long Timeout);
```

Delphi

```
FUNCTION
Rfc1006TxUnlocked (Ref : LongInt; Buf : PChar; Cnt : LongWord; Timeout : LongWordt):
LongInt;
            stdcall; external 'Rfc1006Lib.dll';
```

VB

```
Declare Function Rfc1006TxUnlocked& Lib "Rfc1006Lib.dll"(ByVal Ref&, _
                                    void *Buf, _
                                    ByVal Cnt&, _
                                    ByVal Timeout&)
```

Rfc1006PacketInQ

Rfc1006PacketInQ returns, if a receive packet is in the receive queue.

Call parameters

Nr.	Memory width	Description	Function
1	32-Bit unsigned value	Ref	The connection reference, which was generated with RFC1006Open. Used for internal identification of the connection

Return value

A return value > 0 means, it is at least prepared a packet for the selected connection for pickup.
Call Rfc1006Rx.

C/C++

```
long WINAPI
Rfc1006PacketInQ (long Ref);
```

Delphi

```
FUNCTION
Rfc1006PacketInQ (Ref : LongInt) : LongInt;
            stdcall; external 'Rfc1006Lib.dll';
```

VB

```
Declare Function Rfc1006PacketInQ& Lib "Rfc1006Lib.dll" (ByVal Ref&)
```

Socket error

This list is not exhaustive

Name	Code	meaning
WSAEINTR	10004	Call cancelled
WSAEBADF	10009	
WSAEACCES	10013	access error
WSAEFAULT	10014	Parameters are wrong
WSAEINVAL	10022	- Other function must first be called
		- Socket is already bound to address
		- Socket is not bound by address or already connected
WSAEMFILE	10024	Lack of resources (files, queues)
WSAEWOULDBLOCK	10035	Call would block
WSAEINPROGRESS	10036	Parallel Views not allowed
WSAEALREADY	10037	Terminated routine nevertheless already finished
WSAENOTSOCK	10038	No valid socket specified
WSAEDESTADDRREQ	10039	Destination address required
WSAEMSGSIZE	10040	Datagram too large, was cut
WSAEPROTOTYPE	10041	
WSAENOPROTOOPT	10042	Unknown socket option
WSAEPROTONOSUPPORT	10043	Protocol not supported

Name	Code	meaning
WSAESOCKTNOSUPPORT	10044	Socket type is not supported in the specified address family
WSAEOPNOTSUPP	10045	This socket type is not supported
WSAEPFNOSUPPORT	10046	Protocol family not supported
WSAEAFNOSUPPORT	10047	Address family is not supported
WSAEADDRINUSE	10048	IP-Address or Port already used
WSAEADDRNOTAVAIL	10049	Port/Address not available
WSAENETDOWN	10050	Network does not respond
WSAENETUNREACH	10051	Network can not be reached
WSAENETRESET	10052	Connection through TCP/IP reset
WSAECONNABORTED	10053	Connection through TCP/IP cancelled
WSAECONNRESET	10054	Partner has reset connection
WSAENOBUFS	10055	Lack of resources (internal buffer)
WSAEISCONN	10056	Socket is already connected
WSAENOTCONN	10057	Socket is not connected
WSAESHUTDOWN	10058	Other side has unilaterally terminated compound
WSAETOOMANYREFS	10059	
WSAETIMEDOUT	10060	Call takes too long, so cancelled
WSAECONNREFUSED	10061	Callee wants no connection
WSAELOOP	10062	
WSAENAMETOOLONG	10063	
WSAEHOSTDOWN	10064	
WSAEHOSTUNREACH	10065	Host not reachable
WSAENOTEMPTY	10066	
WSAEPROCLIM	10067	
WSAEUSERS	10068	
WSAEDQUOT	10069	
WSAESTALE	10070	
WSAEREMOTE	10071	
WSASYSNOTREADY	10091	Not ready for network communication
WSAVERNOTSUPPORTED	10092	Desired Winsock version is not supported
WSANOTINITIALISED	10093	Socket.Initialize must be called
WSAHOST_NOT_FOUND	11001	DNS-Server couldn´t be found
WSATRY_AGAIN	11002	Not found searched PC
WSANO_RECOVERY	11003	Unrecoverable error
WSANO_DATA	11004	No name data available
WSANO_ADDRESS	11004	

Version History

V 1.45 26.9.2019

- KeepAlive wurde bei Linux nicht richtig behandelt, Linux kann die Zeit nur in Sekunden angeben, Windows in ms nun werden die angegebenen Zeiten in Sekunden (/1000) umgerechnet
- KeepAlive in Demoprogramm eingebaut
- Unterbrechungserkennung wurde verbessert

V 1.44 23.9.2019

- die Anzahl der möglichen Serververbindungen wurde auf 256 gesetzt

V 1.43 11.4.2019

V 1.42 10.04.2019

- Stabilitätsverbesserung unter Linux durch Umstellung des Taskmanagements auf pthreads
- Kompatibilität zu Linux Kernel 2.4 in separatem Deployment
- Neue Funktionen um Version der Library abzufragen

V 1.41 31.10.2018

- Beim gleichzeitigen Betrieb von Server und Client wurde in der Version 1.41 der Server nicht gestartet

V 1.40 23.10.2018

- Server: Wenn auf den Server TCP/IP Port bereits ein Binding z.B. durch ein anderes Program bestand, erfolgte keine Rückmeldung Rfc1006StartServer meldet in diesem Fall nun RFC1006_BINDING_SERVER_PORT = -15
- Server: SO_REUSEADDR ist jetzt auf “false” gesetzt
- Rfc1006GetSockErrString implementiert

V 1.39 12.4.18

- Tx Funktion thread safe gemacht

V 1.38 5.10.17

- FastAck von 1.37 rückgängig gemacht, nun wieder nach jedem Empfang eines jeden Fragments Siemens SPS mit kleiner PDU-Size bekamen beim senden grosser Pakete Timeoutprobleme
- Linux: fixed: undefined reference to `__stack_chk_fail_local' , mit -fno-stack-protector compiliert

V 1.37 20.9.17

- wenn FastAck aktiv war, wurde diese nach Empgfanges eines jeden Fragmentes gesendet, nun erst nach Empfang des letzen Fragments
- Rfc1006Rx: für Timeout -1 funktioniert bei Windows nicht richtig. Windows kommt sofort zurück, mit Workaround behoben.
- Linux ist mit -fPIC kompiliert, so ist rfc1006lib.o mit “Position independent Code”
- Demo für Linux implementiert
- Demo expired Code ist nun -1234 (vorher 0x1234)

V 1.36

- Linux: Lock Mechanismus mit pthreads implementiert
- Linux: Änderungen seit V 1.31 nachgezogen

V 1.35

- internal Version

V 1.34 - 25.5.17

- Bug: V 1.33 wenn bei Rfc1006Rx in der Timeoutzeit kein Paket empfangen werden konnte, wurde die Verbindung immer geschlossen

BugFix: wenn bei Rfc1006Rx in der Timeoutzeit ein Fragment nicht komplett empfangen werden kann,

wird die Verbindung geschlossen

V 1.33 - 17.5.17

- wenn fragmentierte Packet mit einem Zeitversatz > Timeout beim Rx emfangen werden sollten, wurde nur das letzte Fragment empfangen
  - der RxTimeout bei Rfc1006Rx ist die Zeit die gewartet wird für den Empfang eines Fragmentes
  - Empfehlung: Rfc1006PacketInQ() aufrufen, wenn ein Paket vorhanden, mit entsprechendem RxTimeout Rfc1006Rx aufrufen

```
if (Rfc1006PacketInQ (m_Ref))
{
  Rfc1006Rx (m_Ref, m_RxData), sizeof (m_RxData) - 1, RxTimeout);
}
```

- tritt während des Empfangs von fragmentierten Daten ein Timeout auf, so wird nun die Verbindung geschlossen. Und es wird -1 (Timeout) als Fehler zurück gegeben

V 1.32 - 24.6.15

- Connect von Client und Server des selben Processes (Localhost/selbe Maschine) ging nicht
- paralleles ausführen von “StartServer” hatte DeadLock-Problem

V 1.31 - 28.5.15

- Rfc1006OpenExServer implementiert
- Der Server kann nun auf veschiedenen Ports verbunden werden
- Anzahl maximale Verbindungen auf 64 erhöht (vorher 32)

V 1.30 - 18.12.14

- Aufruf von SetKeepAlive nach Rfc1006Close führte zum Returnwert -7

V 1.29 - 22.8.14

- Verbindungsabbrucherkennung verbessert

V 1.28 - 6.8.14

- Im Serverbetrieb kam es bei geöffneten Verbindungen zur Verzögerung des Connection Request, wenn auf geöffnete Verbindung ein Rx mit langen Timeout lief

V 1.27 - 28.11.12

- TCP/IP NO_DELAY gesetzt

V 1.26 - 24.9.12

- (intern) IsIPConnected überarbeitet, es wurde nicht immer der korrekte Status zurückgegeben
- Default Port bei Angabe Port 0 auf 102 korrigiert
- Anzahl maximaler Verbindungen auf 256 erhöht
- Rfc1006PacketInQ (long Ref) implementiert
  prüft, ob ein Paket zum Empfang bereit ist
  Verwendung:
  if (Rfc1006PacketInQ (Ref))
  {

Rfc1006Rx (mit entsprechendem Timeout)
}

nach Aufruf von Rfc1006Connect und Rfc1006GetStatus kann bei Rückgabe false zusätzlich Rfc1006GetSockErr (Ref) aufgerufen werden, um den Grund auf Socketebene zu ermitteln

V 1.25 - 12.9.12

- RxRFC1006 mit Timeout 0 wartete länger als 0 ms nun behoben

V 1.24

- Rfc1006GetStatus lieferte nicht immer den richtigen Connectstatus

V 1.23 - 5.12.11

- PDU-Size auf mindestens 128 Byte abgebprüft

V 1.22 - 28.7.11

- FastAck mit Funktion Rfc1006SetFastAck einstellbar grundsätzlich aus

V 1.21 - 11.7.11

- FastAck eingebaut

V 1.20 - 28.6.11

- Max PDUSize auf 8 K gesetzt

V 1.19

- Rfc1006TxUnlocked eingefügt

V 1.18 - 16.5.11

- SetKeepAlive eingefügt
- GetStatus intern, den echten Status der Verbindung geprüft

Table of Contents