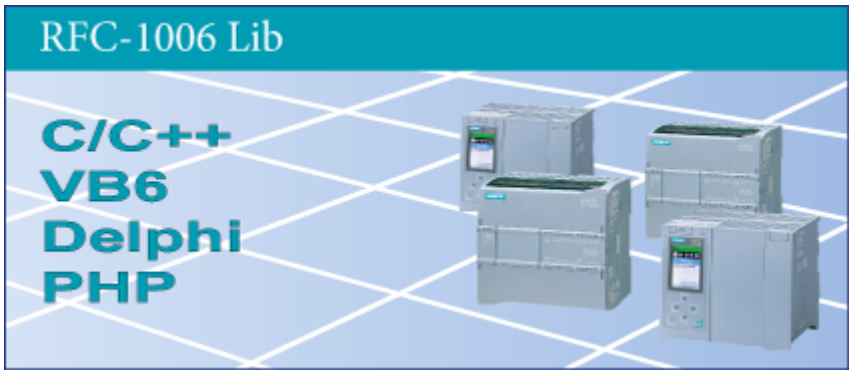
RFC 1006 Lib

RFC 1006 Client- und Serverentwicklung leicht gemacht



Software Version 1.40

Betriebssystem

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

Programmiersprachen

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

Versionshistorie - Die Liste der Verbesserungen pro Version

Voraussetzungen

Hardware

- PC mit installiertem TCP/IP-Protokoll und Netzwerkkarte

Installation

Windows

Kopieren Sie die DLL-Datei in das Verzeichnis Ihres Programms. Um die Bibliothek systemweit zur Verfügung zu stellen, legen Sie sie im Systemverzeichnis `%SystemRoot%\system32` ab.

Linux

Linken Sie die Object-Datei (.o) beim Linkeraufruf zu Ihrem Programm.

Funktionsweise

RFC1006-Lib ist eine DLL für MS-Windows, welche die Anbindung eines PC an Industrial Ethernet übre RFC-1006 ermöglicht.
Mit einfachen Funktionen kann der Anwender schnell mit C, C++, Delphi, Visual Basic oder auch Excel RFC 1006 Verbindungen aufbauen und Daten senden und Empfangen.
Zur Kopplung wird nur die IP-Adresse, DSAP, SSAP des Partners benötigt. Zur Kommunikation für RFC1006 wird standardmäßig Port 102 verwendet.

Funktionsbeschreibung im Detail

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

Serverbetrieb

Zum Betrieb als Server beachte man die Funktionen:
Rfc1006OpenServer
Rfc1006GetStatus
Rfc1006StartServer

Ansonsten können die Funktionen **Rfc1006Rx** und **Rfc1006Tx** für die Kommunikation verwendet werden. Es ist ratsam in regelmäßigen Zeitabständen mit **Rfc1006GetStatus** zu prüfen, ob eine Verbindung besteht und dann erst die Sende und Empfangsroutinen verwenden.

Folgende Funktionen stehen zur Verfügung:

Initialisierung

Rfc1006Open

zur Initialisierung der Verbindung, dort wird nur Speicher vorbereitet. Beim ersten Aufruf der Lese- oder Schreibfunktionen wird, die TCP/IP-Verbindung / RFC 1006 Verbindung automatisch gestartet.
(Nur bei Betrieb als Client verwenden!)

Aufrufparameter

Nr.	Speicherbreite	Bezeichnung	Funktion
1	Zeiger auf „0„-terminierte Zeichenkette (C-String, 32-Bit Zeiger	IPAdr	IP-Adresse der SPS im Format: xxx.xxx.xxx.xxx. Beispiel: 192.169.0.100
2	Zeiger auf „0“-terminierte Zeichenkette (C-String, 32-Bit Zeiger	SSAP	eigener SAP Es werden maximal 255 Zeichen verwendet.
3	32-Bit Wert ohne Vorzei.	SSAPLen	Länge des eigenen SAP. Es werden maximal 255 Zeichen verwendet
4	Zeiger auf „0„-terminierte Zeichenkette (C-String, 32-Bit Zeiger	DSAP	SAP des Partners. Es werden maximal 255 Zeichen verwendet.
5	32-Bit Wert ohne Vorzei.	DSAPLen	Länge des SAP des Partners. Es werden maximal 255 Zeichen verwendet
6	32-Bit Wert ohne Vorzei.	PortNr	Nummer des TCP/IP-Ports der für diese Verbindung verwendet werden soll. Standard ist 102. Wird hier 0 angegeben so wird intern 102 verwendet

Nr.	Speicherbreite	Bezeichnung	Funktion
7	32-Bit Wert ohne Vorzei.	ConnectTimeout	Timeout in Millisekunden für Warten auf Verbindungsaufbau mitdem Partner. 0 bedeutet Standardeinstellung = 5000 ms (5sec.) muss bei Bedarf verlängert werden.

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

Legt eine neue Verbindung für den Serverbetrieb an. Der Start der Verbindung wird mit **Rfc1006StartServer** gesteuert.
Wurde bereits „Rfc1006StartServer“ ausgeführt, so wird die „neue Verbindung“ sofort als verfügbar eingetragen.
Falls der Port bereits von einer anderen Anwendung verwendet wird kommt es zu einem Socketfehler. Mit Rfc1006GetSockErr, Rfc1006GetSockErrString kann er genaue Grund ermittelt werden.

Ab Version 1.31 wurde „Rfc1006OpenExServer“ implementiert. Hier kann der TCP-Port für die gewünschte Verbindung angegeben werden. So ist es möglich, auf verschiedenen Ports gleichzeitig Verbindungen entgegenzunehmen. **(Nur bei Betrieb als Server verwenden!)**

Aufrufparameter

Nr.	Speicherbreite	Bezeichnung	Funktion
1	Zeiger auf „0“-terminierte Zeichenkette (C-String, 32-Bit Zeiger	DstIPAdr	IP-Adresse des Clients, welcher eine Verbindung zum Server aufnehmen darf. Beispiel: „192.169.0.100„. Ist der String Leer also „“, wird jede Absendeadresse akzeptiert.
2	Zeiger auf „0„-terminierte Zeichenkette (C-String, 32-Bit Zeiger	SSAP	eigener SAP Es werden maximal 255 Zeichen verwendet.
3	32-Bit Wert ohne Vorzei.	SSAPLen	Länge des eigenen SAP. Es werden maximal 255 Zeichen verwendet

Nr.	Speicherbreite	Bezeichnung	Funktion
4	Zeiger auf „0“-terminierte Zeichenkette (C-String, 32-Bit Zeiger	DSAP	SAP des Partners. Es werden maximal 255 Zeichen verwendet.
5	32-Bit Wert ohne Vorzei.	DSAPLen	Länge des SAP des Partners. Es werden maximal 255 Zeichen verwendet
6	32-Bit Wert ohne Vorzei.	Port	TCP/IP Port (nur bei Rfc1006OpenExServer)

Rückgabewerte

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

Wert	Fehlerbeschreibung	Bedeutung
>= 0	Alles OK	Die Rückgabe ist die Referenznummer für diese Verbindung und muss bei allen anderen Funktionen als Eingangsparameter Ref verwendet werden.
-2	Keine Resourcen mehr frei.	Maximale Anzahl an verfügbaren Verbindungen erreicht.
-7	Es ist ein Socketfehler aufgetreten	Das kann passieren, wenn der Server bereits gestartet ist und eine neue Verbindung hinzugefügt wird und der angegebene Port bereits von einer anderen Anwendung verwendet wird.

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

(Nur bei Betrieb als Server verwenden!)

Startet den Server mit den mit **Rfc1006OpenServer** / **Rfc1006OpenExServer** angelegten Verbindungen.

Nach dem Start können Verbindungen mit den Methoden **Rfc1006OpenServer** und **Rfc1006CloseServer** dynamisch hinzugefügt oder entfernt werden.

Sollte der Port einer angelegten Verbindung bereits durch eine andere Anwendung verwendet werden, tritt ein Socketfehler auf (Returnwert `E_RFC1006_SOCKERR`). Mit **Rfc1006GetSockErr** und **Rfc1006GetSockErrString** kann der genaue Grund ermittelt werden - verwenden Sie bei der Abfrage `Ref = -1`.

Aufrufparameter

Nr.	Speicherbreite	Bezeichnung	Funktion
1	32-Bit Wert	Port	**ACHTUNG** Dieser Parameter wird intern nicht mehr verwendet. Verwenden Sie **Rfc1006OpenExServer**, um den Port einer Verbindung festzulegen.
2	Zeiger auf „0„-terminierte Zeichenkette (C-String, 32-Bit Zeiger	IPBindAdr	IP-Adresse der Netzwerkkarte auf welcher die Verbindung laufen soll. NULL oder ein Leerstring läßt die Verbindung auf allen Netzwerkkarten zu.
3	32-Bit Wert	MaxCon	Maximale Anzahl der Verbindungen, die der Server unterhalten kann.

Rückgabewerte

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

Wert	Fehlerbeschreibung	Bedeutung
> 0	Alles OK	Server wurde gestartet.
-10	Server läuft bereits.	Rfc1006StartServer wurde bereits aufgerufen.
-11	Server kann nicht gestartet werden	Fehler beim Starten des Serverthreads aufgetreten.
-15	Der angegebene Port ist bereits in Verwendung.	Binding kann nicht durchgeführt werden.

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)
```

Deinitialisierung

Rfc1006StopServer

Beendet den Server.
(Nur bei Betrieb als Server verwenden!)

Rückgabewerte

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

Wert	Fehlerbeschreibung	Bedeutung
> 0	Alles OK	Server wurde beendet.
-12	Server war nicht gestartet.	RFC1006StartServer wurd nicht aufgerufen.

C/C++

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

Delphi

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

VB

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

Rfc1006Close

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

Aufrufparameter

Nr.	Speicherbreite	Bezeichnung	Funktion
1	32-Bit Wert ohne Vorzei.	Ref	Die Referenz der Verbindung, welche mit Rfc1006Open generiert wurde. Dient zur internen Identifikation der Verbindung.

Rückgabewerte

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

Wert	Fehlerbeschreibung	Bedeutung / Reaktion
0	alles OK	Speicher wieder freigegeben und Verbindung, wenn vorhanden geschlossen
-3	Mit der angegebenen Referenznummer wurde kein Rfc1006Open durchgeführt	Haben Sie Rfc1006Open aufgerufen?
-99	Die Referenznummer ist ungültig	————————

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

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

C/C++

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

Delphi

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

VB

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

Empfangen und Senden

Rfc1006Rx / Rfc1006Tx

Funktion	Beschreibung / Zweck
Rfc1006Rx	Versucht Daten über RFC 1006 zu empfangen. Besteht noch keine Verbindung zum Partner, so wir diese vorher aufgebaut (Nur im Clientmodus). Besteht im Serverbetrieb keine Verbindung für, so kehrt die Funktion sofort mit –1 zurück.
Rfc1006Tx	Sendet Daten über RFC 1006 an den Partner. Besteht noch keine Verbindung zum Partner, so wir diese vorher aufgebaut(Nur im Clientmodus). Besteht im Serverbetrieb keine Verbindung für, so kehrt die Funktion sofort mit –1 zurück.

Aufrufparameter

Die Lese- und Schreibfunktionen besitzen die selben Eingangparameter:

Nr.	Speicherbreite	Bezeichnung	Funktion
1	32-Bit Wert ohne Vorzei.	Ref	Die Referenz der Verbindung, welche mit RFC1006Open generiert wurde. Dient zur internen Identifikation der Verbindung

Nr.	Speicherbreite	Bezeichnung	Funktion
2	32-Bit Adresse	Buffer	Die Adresse auf den Quell- bzw. Zielspeicher im PC.
3	32-Bit Wert ohne Vorzei.	Cnt	Bei Rx, Anzahl der maximal zu empfangenden Datenbytes. Bei Tx Anzahl der zu sendenden Datenbytes.
4	32-Bit Wert mit Vorzei.	Timeout	Timeout in ms für Senden bzw. Empfangen 0 = warte nicht -1 = warte bis alles gesendet, bis irgendetwas empfangen wir

Rückgabewerte

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

Wert	Fehlerbeschreibung	Reaktion
>=0	Anzahl der gesendeten / empfangenen Datenbytes	
-1	Zeitüberlauf, gewünschter Partner offensichtlich nicht oder nicht mehr vorhanden	Einfach weitere Schreib- und Leseaufträge absetzten der Treiber baut die Verbindung automatisch auf. Evtl. die Timeoutzeiten insbesondere die Connect-Timeoutzeit verlängern.
-5	Allgemeiner Fehler	Prüfen ob Netzwerk richtig im PC installiert ist: TCP/IP aktiviert ? Winsocket installiert ?
-6	Partner nicht gefunden	Portnummer in Rfc1006Open nicht korrekt oder es ist keine Verbindung mehr zu diesem Port frei. Konfiguratioin prüfen
-7	Socketfehler aufgetreten	Rfc1006GetSockErr aufrufen und Fehler auswerten
-13	Noch kein Client mit dem Server verbunden.	Client verbinden, evtl. SSAP DSAP prüfen.
-14	Die Größe des Empfangsbuffers ist zu klein.	Empfangsbuffer vergrössern
-99	Die Referenznummer ist ungültig	Haben Sie Rfc1006Open aufgerufen ?
-1234	Demozeit ist abgelaufen	Vollversion erwerben

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&)
```

Verbindung

Rfc1006GetSockErr

Liefert den letzten Socket-Fehler zurück. Dieser wird Verbindungsbezogen gespeichert. Tritt eine Socket-Fehler beim Aufruf von **Rfc1006OpenServer** / **Rfc1006OpenExServer** / **RfcStartServer** auf, so gibt es noch keine Verbindungsreferenz. In diesem Falle wird der Socket-Fehler „globalen“ Bereich gespeichert. Zur Abfrage dieses Wertes verwenden Sie als Ref den Wert -1.

Aufrufparameter

Nr.	Speicherbreite	Bezeichnung	Funktion
1	32-Bit Wert ohne Vorzei.	Ref	Die Referenz der Verbindung, welche mit Rfc1006Open generiert wurde. Dient zur internen Identifikation der Verbindung. Zur Abfrage des globalen Socketfehlers den Wert -1 verwenden.

Rückgabewerte

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

Wert	Fehlerbeschreibung	Bedeutung / Reaktion
0	alles OK	Es liegt kein Fehler an
-3	Mit der angegebenen Referenznummer wurde kein Rfc1006Open durchgeführt	Haben Sie Rfc1006Open aufgerufen ?.
-99	Die Referenznummer ist ungültig	————————
Sonstige	Socketerror	Erklärung siehe Liste unterhalb.

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

Liefert den Socketfehler in Textform zurück

Aufrufparameter

Nr.	Speicherbreite	Bezeichnung	Funktion
1	32-Bit Wert	SockErr	Die Socke-Fehlernummer, welche von der Funkion Rfc1006GetSockErr geliefert wurde.
2	Pointer auf einen C-String	ErrStr	Die Adresse des C-Strings, wo der Text abgelegt werden soll.
3	32-Bit Wert unsigned	MaxLen	Maximale Länge des „ErrStr“.

Rückgabewerte

Die Funktionen liefert den Pointer auf den C-String des 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

Liefert den Status der Verbindung

Aufrufparameter

Nr.	Speicherbreite	Bezeichnung	Funktion
1	32-Bit Wert ohne Vorzei.	Ref	Die Referenz der Verbindung, welche mit Rfc1006Open generiert wurde. Dient zur internen Identifikation der Verbindung

Rückgabewerte

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

Wert	Fehlerbeschreibung	Bedeutung / Reaktion
0	Verbindung nicht vorhanden	
1	Verbindung vorhanden	
-3	Mit der angegebenen Referenznummer wurde kein Rfc1006Open durchgeführt	Haben Sie Rfc1006Open aufgerufen ?.
-99	Die Referenznummer ist ungültig	————————

Wert	Fehlerbeschreibung	Bedeutung / Reaktion
Sonstige	Socketerror	Erklärung siehe Liste unterhalb.

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

Führt die Verbindung zum Partner aus

Aufrufparameter

Nr.	Speicherbreite	Bezeichnung	Funktion
1	32-Bit Wert ohne Vorzei.	Ref	Die Referenz der Verbindung, welche mit Rfc1006Open generiert wurde. Dient zur internen Identifikation der Verbindung

Rückgabewerte

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

Wert	Fehlerbeschreibung	Bedeutung / Reaktion
0	Verbindung konnte nicht hergestellt werden.	
1	Verbindung hergestellt	
-3	Mit der angegebenen Referenznummer wurde kein Rfc1006Open durchgeführt	Haben Sie Rfc1006Open aufgerufen ?.
-99	Die Referenznummer ist ungültig	————————
Sonstige	Socketerror	Erklärung siehe Liste unterhalb.

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

Trennt die Verbindung zum Partner.

Aufrufparameter

Nr.	Speicherbreite	Bezeichnung	Funktion
1	32-Bit Wert ohne Vorzei.	Ref	Die Referenz der Verbindung, welche mit Rfc1006Open generiert wurde. Dient zur internen Identifikation der Verbindung

Rückgabewerte

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

Wert	Fehlerbeschreibung	Bedeutung / Reaktion
0	Verbindung getrennt.	
1	Verbindung nicht getrennt	
-3	Mit der angegebenen Referenznummer wurde kein Rfc1006Open durchgeführt	Haben Sie Rfc1006Open aufgerufen ?.
-99	Die Referenznummer ist ungültig	————————
Sonstige	Socketerror	Erklärung siehe Liste unterhalb.

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

Aktiviert/daktiviert FastAcknowlege Antwort, default: deaktiviert

Aufrufparameter

Nr.	Speicherbreite	Bezeichnung	Funktion
1	32-Bit Wert ohne Vorzei.	Ref	Die Referenz der Verbindung, welche mit Rfc1006Open generiert wurde. Dient zur internen Identifikation der Verbindung
2	32-Bit Wert ohne Vorzei	Mode	1 = ein 0 = aus

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

Setzt individuelle TCP/IP KeepAlive Zeiten für die mit Ref angegebene Verbindung. Muss nur verwendet werden, wenn die Standardwerte nicht gelten sollen.

Sie sollten diese Funktion unmittelbar nach dem „Open“-Aufruf ausführen.

Aufrufparameter

Nr	Datentyp	Datentyp (PHP)	Name	Funktion
1	32-Bit Wert unsigned	long	Ref	Die Referenz der Verbindung, welche mit Rfc1006Open generiert wurde. Dient zur internen Identifikation der Verbindung
2	32-Bit Wert unsigned	long	AliveTime	Findet innerhalb der Zeit „AliveTime“ (ms) kein Datenverkehr auf der TCP/IP-Verbindung statt, so wird ein KeepAlive-Telegramm gesendet, um die Verbindung zu prüfen. Wird bei dieser Prüfung ein Fehler festfestellt, sendet der IP-Stack innerhalb der Zeit „AliveInterval“ (ms) ein nächstes KeepAlive-Telegramm. Dies wird einige male innerhalb der Zeit **AliveInterval** wiederholt (bei Windows 6 mal). War der Vorgang nicht erfolgreich, wird die Verbindung beendet.
3	32-Bit Wert unsigned	long	AliveInterval	Das Intervall in ms, in welchem KeepAlive Telegramme wiederholt werden. Dieses wird aktiv, wenn ein Fehler beim Senden / Empfangen eines KeepAlive-Telegramms aufgetreten ist.

Rückgabewerte

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

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

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

Rfc1006TxUnlocked sendet Daten an den Partner ohne die Verbindungdaten zu locken. Diese Funktion kann bei MultiThreading Anwednung finden, wenn ein Receive-Thread die Verbindung gerade blockiert. VORSICHT! Der Aufrufer hat dafür zu sorgen, dass während der Ausführung das Objekt mit der angegebenen Referenz geöffnet bleibt!
Die Funktion und Returnwerte wie bei 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 liefert zurück, ob ein Paket zum Empfangen in der Empfangsqueue vorhanden ist.

Aufrufparameter

Nr.	Speicherbreite	Bezeichnung	Funktion
1	32-Bit Wert ohne Vorzei.	Ref	Die Referenz der Verbindung, welche mit Rfc1006Open generiert wurde. Dient zur internen Identifikation der Verbindung

Rückgabewerte

Ein Returnwert > 0 bedeutet, es liegt mindestens ein Packet für die gewählte Verbindung zur Abholung bereit.

Rfc1006Rx aufrufen.

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&)
```

Socketfehler

Diese Liste erhebt keinen Anspruch auf Vollständigkeit

Name	Code	Bedeutung
WSAEINTR	10004	Aufruf wurde abgebrochen
WSAEBADF	10009	
WSAEACCES	10013	Zugriffsfehler
WSAEFAULT	10014	Parameter sind falsch
WSAEINVAL	10022	- Andere Funktion muß vorher aufgerufen werden
		- Socket ist schon an Adresse gebunden
		- Socket noch nicht an Adresse gebunden bzw. schon verbunden
WSAEMFILE	10024	Resourcen fehlen (Dateien, Warteschlangen)
WSAEWOULDBLOCK	10035	Aufruf würde blockieren
WSAEINPROGRESS	10036	Parallele Aufrufe nicht erlaubt
WSAEALREADY	10037	Abgebrochene Routine trotzdem schon fertig
WSAENOTSOCK	10038	Kein gültiger Socket angegeben
WSAEDESTADDRREQ	10039	Zieladresse benötigt
WSAEMSGSIZE	10040	Datagram zu groß, wurde abgeschnitten
WSAEPROTOTYPE	10041	
WSAENOPROTOOPT	10042	Unbekannte Socket-Option
WSAEPROTONOSUPPORT	10043	Protokoll wird nicht unterstützt
WSAESOCKTNOSUPPORT	10044	Sockettyp wird in angegebener Adressfamilie nicht unterstützt
WSAEOPNOTSUPP	10045	Dieser Sockettyp wird nicht unterstützt
WSAEPFNOSUPPORT	10046	Protokollfamilie wird nicht unterstützt
WSAEAFNOSUPPORT	10047	Adressfamilie wird nicht unterstützt
WSAEADDRINUSE	10048	IP-Adresse bzw. Port werden schon/noch benutzt
WSAEADDRNOTAVAIL	10049	Port/Adresse nicht verfügbar
WSAENETDOWN	10050	Netzwerk reagiert nicht
WSAENETUNREACH	10051	Netzwerk kann nicht erreicht werden
WSAENETRESET	10052	Verbindung durch TCP/IP zurückgesetzt
WSAECONNABORTED	10053	Verbindung durch TCP/IP abgebrochen
WSAECONNRESET	10054	Partner hat Verbindung zurückgesetzt

Name	Code	Bedeutung
WSAENOBUFS	10055	Resourcen fehlen (Interner Pufferspeicher)
WSAEISCONN	10056	Socket ist schon verbunden
WSAENOTCONN	10057	Socket ist noch nicht verbunden
WSAESHUTDOWN	10058	Andere Seite hat Verbindung einseitig beendet
WSAETOOMANYREFS	10059	
WSAETIMEDOUT	10060	Aufruf dauert zu lange, daher Abbruch
WSAECONNREFUSED	10061	Angerufener möchte keinen Verbindungsaufbau
WSAELOOP	10062	
WSAENAMETOOLONG	10063	
WSAEHOSTDOWN	10064	
WSAEHOSTUNREACH	10065	Host nicht erreicbar
WSAENOTEMPTY	10066	
WSAEPROCLIM	10067	
WSAEUSERS	10068	
WSAEDQUOT	10069	
WSAESTALE	10070	
WSAEREMOTE	10071	
WSASYSNOTREADY	10091	Netzwerk nicht zur Kommunikation bereit
WSAVERNOTSUPPORTED	10092	gewünschte Winsock-Version wird nicht unterstützt
WSANOTINITIALISED	10093	Socket.Initialize muß aufgerufen werden
WSAHOST_NOT_FOUND	11001	DNS-Server nicht gefunden
WSATRY_AGAIN	11002	Gesuchter Rechner nicht gefunden
WSANO_RECOVERY	11003	Nicht behebbarer Fehler
WSANO_DATA	11004	Keine Namensdaten vorhanden
WSANO_ADDRESS	11004	

Versionshistorie

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

Inhaltsverzeichnis