OBID®
Welcome to RFID

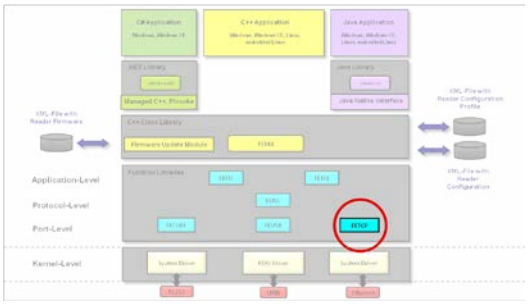
HANDBUCH

ID FETCP

Version 2.02.00

Software-Support für

TCP/IP-Schnittstelle



Betriebssystem	Ausführung		Anmerkungen
	32-Bit	64-Bit	
Windows XP	X	(X)	bei 64-Bit nur mit 32-Bit Laufzeitsystem
Windows Vista / 7 / 8	X	X	
Windows CE	X	-	
Linux	X	X	
Apple Max OS X	-	X	ab V10.7.3, Architektur x86_64

final
public (B)
2014-03-17
H30802-8d-ID-B.doc

FEIG ELECTRONIC

Hinweis

 FEIG ELECTRONIC GmbH
Lange Straße 4
D-35781 Weilburg-Waldhausen
Tel.: +49 6471 3109-0
eMail: obid@feig.de

Alle früheren Ausgaben verlieren mit dieser Ausgabe ihre Gültigkeit.
Die Angaben in diesem Dokument können ohne vorherige Ankündigung geändert werden.

Die Zusammenstellung der Informationen in diesem Dokument erfolgt nach bestem Wissen und Gewissen. FEIG ELECTRONIC GmbH übernimmt keine Gewährleistung für die Richtigkeit und Vollständigkeit der Angaben in diesem Dokument. Insbesondere kann FEIG ELECTRONIC GmbH nicht für Folgeschäden auf Grund fehlerhafter oder unvollständiger Angaben haftbar gemacht werden. Da sich Fehler, trotz aller Bemühungen nie vollständig vermeiden lassen, sind wir für Hinweise jederzeit dankbar.

Die in diesem Dokument gemachten Installationsempfehlungen gehen von günstigsten Rahmenbedingungen aus. FEIG ELECTRONIC GmbH übernimmt keine Gewähr für die einwandfreie Funktion in systemfremden Umgebungen.

FEIG ELECTRONIC GmbH übernimmt keine Gewährleistung dafür, dass die in diesem Dokument enthaltenden Informationen frei von fremden Schutzrechten sind. FEIG ELECTRONIC GmbH erteilt mit diesem Dokument keine Lizenzen auf eigene oder fremde Patente oder andere Schutzrechte.

Lizenzvertrag über die Nutzung der Software

Dies ist ein Vertrag zwischen Ihnen und der FEIG ELECTRONIC GmbH (nachfolgend "FEIG") über die Nutzung der überlassenen Programmbibliothek ID FETCP und die vorliegende Dokumentation, nachfolgend Lizenzmaterial genannt. Mit der Installation und Benutzung der Software erklären Sie sich mit allen Bestimmungen dieses Vertrages ausnahmslos und ohne Einschränkung einverstanden. Wenn Sie mit den Bestimmungen dieses Vertrages nicht oder nicht vollständig einverstanden sind, dürfen Sie das Lizenzmaterial nicht installieren oder anderweitig benutzen. Das überlassene Lizenzmaterial ist Eigentum der FEIG ELECTRONIC GmbH und ist international urheberrechtlich geschützt.

§1 Vertragsgegenstand und Vertragsumfang

1. FEIG gewährt Ihnen das Recht, das überlassene Lizenzmaterial zu installieren und zu den nachstehenden Bedingungen zu nutzen.
2. Sie dürfen sämtliche Bestandteile des Lizenzmaterials auf einer Festplatte oder einem sonstigen Speichermedium installieren. Die Installation und Nutzung darf auch auf einem Netzwerk-Fileserver erfolgen. Sie dürfen Sicherheitskopien des Lizenzmaterials anfertigen.
3. FEIG gewährt Ihnen das Recht die dokumentierte Programmbibliothek für die Entwicklung eigener Anwendungsprogramme oder Programmbibliotheken zu verwenden und Sie dürfen die Laufzeitdatei FETCP.DLL, FETCPCE.DLL, LIBFETCP.x.y.z.DYLIB[1] oder LIBFETCP.SO.x.y.z[1] ohne Abgabe von Lizenzgebühren vertreiben, unter der Voraussetzung, dass diese Anwendungsprogramme oder Programmbibliotheken nur zusammen mit von FEIG entwickelten Geräten verwendet werden.

§2 Schutz des Lizenzmaterials

1. Das Lizenzmaterial ist geistiges Eigentum von FEIG und seinen Lieferanten. Es ist gemäß Urheberrecht, internationalen Verträgen und einschlägigen Gesetzen des Landes geschützt, in dem sie genutzt wird. Struktur, Organisation und Code der Software sind wertvolles Geschäftsgeheimnis und vertrauliche Information von FEIG und seinen Lieferanten.
2. Sie verpflichten sich, die Programmbibliothek sowie die Dokumentation nicht zu ändern, anzupassen, zu übersetzen, rückzuentwickeln, zu dekompilieren, zu disassemblieren oder auf andere Weise zu versuchen, den Quellcode dieser Software herauszufinden.
3. Soweit FEIG im Lizenzmaterial Schutzvermerke, wie Copyright-Vermerke und andere Rechtsvorbehalte angebracht hat, sind Sie verpflichtet, diese unverändert beizubehalten sowie in alle von Ihnen hergestellten vollständigen oder teilweisen Kopien in unveränderter Form zu übernehmen.
4. Die Weitergabe von Lizenzmaterial ist weder vollständig noch auszugsweise gestattet, solange dazu keine explizite anderslautende Vereinbarung zwischen Ihnen und FEIG getroffen wurde. Nicht betroffen von dieser Regelung sind solche Anwendungsprogramme oder Programmbibliotheken, die gem. §1 Absatz 3. dieser Vereinbarung erstellt und vertrieben werden.

§3 Gewährleistung und Haftungsbeschränkungen

1. Sie stimmen mit FEIG darüber überein, dass es nicht möglich ist, EDV-Programme so zu entwickeln, dass sie für alle Anwendungsbedingungen fehlerfrei sind. FEIG weist Sie ausdrücklich darauf hin, dass die Installation eines neuen Programms bereits vorhandene Software beeinflussen kann, und zwar auch solche Software, die nicht gleichzeitig mit der neuen Software ausgeführt wird. FEIG haftet in keinem Fall für direkte oder indirekte Schäden, für Folgeschäden oder Sonderschäden, Einschließlich entgangenen Geschäftsgewinn oder entgangener Einsparungen. Wenn Sie sicherstellen wollen, dass es zu keinerlei Beeinflussung eines bereits installierten Programms kommt, dürfen Sie die vorliegende Software nicht installieren.
2. FEIG weist ausdrücklich darauf hin, dass mit der Software irreversible Einstellungen und Anpassungen an Geräten vorgenommen werden können, wodurch diese Geräte zerstört oder unbrauchbar gemacht werden können. FEIG übernimmt für derartiges Handeln unabhängig davon ob dies bewusst oder unbewusst erfolgte keinerlei Gewährleistung.
3. FEIG liefert Ihnen die Software "wie besehen" ohne jegliche Gewährleistung. FEIG kann für die Leistung oder die Ergebnisse, die Sie durch die Nutzung der Software erzielen, nicht garantieren. FEIG übernimmt keine Gewährleistung oder Garantie dafür, dass keine Schutzrechte Dritter verletzt werden, auch nicht dafür, dass die Software für irgendeinen bestimmten Zweck geeignet ist.
4. FEIG weist ausdrücklich darauf hin, dass das Lizenzmaterial nicht für den Einsatz mit oder in medizinischen Geräten oder für Geräte für lebenserhaltende Maßnahmen konzipiert ist, bei denen ein Fehler eine Gefahr für menschliches Leben oder für die gesundheitliche Unversehrtheit zur Folge haben kann.
   Der Anwender des Lizenzmaterials ist dafür verantwortlich, geeignete Maßnahmen zu ergreifen um Gefahren, Schäden oder Verletzungen zu vermeiden.

[1] *x.y.z repräsentiert die aktuelle Versionsnummer*

§4 Schlußbestimmungen

1. Dieser Vertrag enthält die vollständigen Lizenzbestimmungen und ersetzt alle eventuell vorangegangenen Regelungen und Absprachen. Änderungen und Ergänzungen bedürfen der Schriftform.
2. Sollte eine der in diesem Vertrag enthaltenen Bestimmungen unwirksam sein oder werden, so wird die Gültigkeit der übrigen Bestimmungen hierdurch nicht berührt. Beide Vertragsparteien verpflichten sich, die unwirksame Bestimmung durch eine solche wirksame Bestimmung zu ersetzen, die dem wirtschaftlichem Zweck der zu ersetzenden Bestimmung am nächsten kommt.
3. Dieser Vertrag unterliegt dem Recht der Bundesrepublik Deutschland. Gerichtsstand ist Weilburg.

Inhalt

1. Einleitung

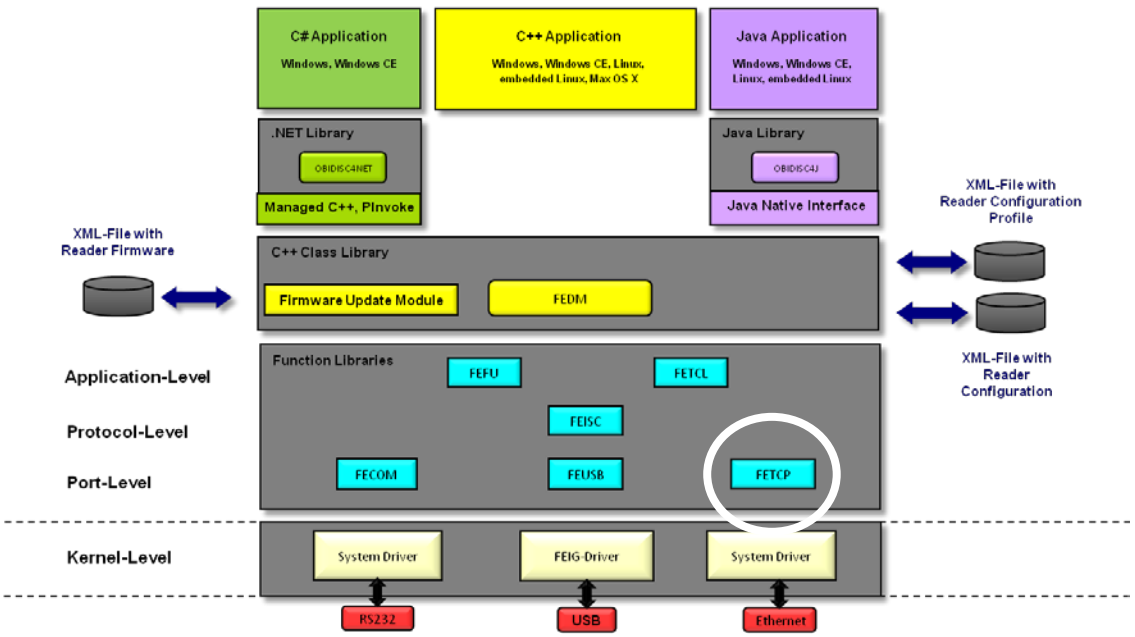
Das Supportpaket ID FETCP dient zur Unterstützung bei der Programmierung von kommunikations-orientierter Software und unterstützt die Sprachen ANSI-C, ANSI-C++ und prinzipiell jede andere Sprache, die C-Funktionen aufrufen kann.

Mit dem Supportpaket wird eine einfache Funktionsschnittstelle zum Socket API der unterstützten Betriebssysteme angeboten und ist speziell für die gemeinsame Verwendung mit weiteren Supportpaketen (z.B. ID FEISC) entwickelt worden.

Verwendet werden kann die Bibliothek mit folgenden Betriebssystemen:

Betriebssystem	Ausführung		Anmerkungen
	32-Bit	64-Bit	
Windows XP	X	(X)	bei 64-Bit nur mit 32-Bit Laufzeitsystem
Windows Vista / 7 / 8	X	X	
Windows CE	X	-	
Linux	X	X	
Apple Max OS X	-	X	ab V10.7.3, Architektur x86_64

Die Bibliothek FETCP bildet die erste Ebene in dem mehrschichtigen, hierarchisch strukturierten Aufbau von FEIG-Bibliotheken. Mit ihr wird ausschließlich die Transportschicht zum TCP/IP-Treiber des Betriebssystems realisiert. Das nachfolgende Bild zeigt eine Übersicht über alle Bibliotheken.



Programmierer, die sich für diese Schicht als Integrationsoption entscheiden, müssen das Protokollhandling (Aufbau/Zerlegung von Protokollrahmen, CRC-Prüfung, Längenprüfung) in Ihrer

Applikation implementieren. Dadurch entsteht erheblicher Programmieraufwand und es gilt abzuwägen, ob der Einstieg auf diesem Level zwingend notwendig ist.

Wer nur auf die Funktionsbibliotheken zurückgreifen muss oder möchte, sollte die Bibliothek FEISC als nächst höheres API wählen.

1.1. Lieferumfang

Dieses Supportpaket besteht aus den nachfolgend aufgelisteten Dateien. In der Regel wird das Paket mit anderen Bibliotheken in einem speziell für das jeweilige Betriebssystem zusammengestellten Software Development Kit (SDK) – z.B. ID ISC.SDK.Win - ausgeliefert.

1.1.1. Windows XP / Vista / 7 / 8

Datei	Verwendung
FETCP.DLL	DLL mit allen Funktionen
FETCP.LIB	LIB-Datei zum Linken für C/C++-Projekte
FETCP.H	Header-Datei für C/C++-Projekte

1.1.2. Windows CE

Datei	Verwendung
FETCPCE.DLL	DLL mit allen Funktionen
FETCPCE.LIB	LIB-Datei zum Linken für C/C++-Projekte
FETCP.H	Header-Datei für C/C++-Projekte

1.1.3. Linux

Datei	Verwendung
LIBFETCP.SO.x.y.z[2]	Funktions-Bibliothek mit allen Funktionen
FETCP.H	Header-Datei für C/C++-Projekte

1.1.4. Mac OS X

Datei	Verwendung
LIBFETCP.x.y.z.dylib[2]	Funktions-Bibliothek mit allen Funktionen
FETCP.H	Header-Datei für C/C++-Projekte

[2] *x.y.z repräsentiert die Versionsnummer der Bibliotheksdatei*

2. Änderungen gegenüber der Vorversion

- Bugfix: Pufferüberlauf beim Empfang von Daten wird abgefangen, wenn der bereitgestellt Puffer zu klein ist
- Neuer Fehlercode: FETCP_ERR_BUFFER_OVERFLOW
- Verbesserte Threadsicherheit
- Windows:
  1. Migration der Entwicklungsumgebung von Visual Studio 2008 zu Visual Studio 2010.
  2. DLL jetzt ohne MFC
  3. Erstes Release der 64-Bit Version
  4. Anbindung an Log-Manager
- Erste Release-Version für Mac OS X, ab V10.7.3
- Linux:

  Version für 64-Bit

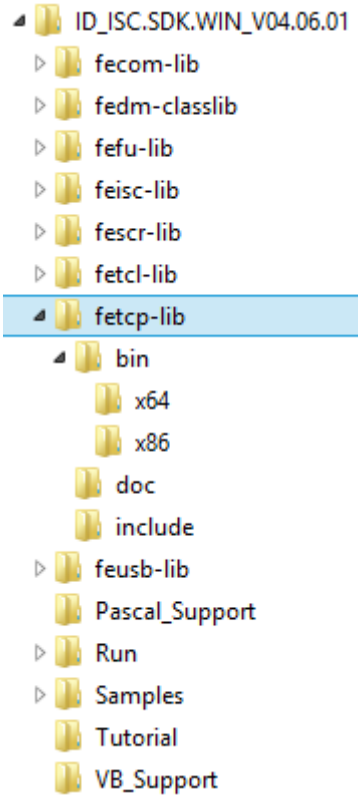
Bitte beachten Sie auch die Änderungshistorie im Anhang.

3. Installation

Das Supportpaket wird in der Regel mit einem Software Development Kit (SDK) ausgeliefert. Kopieren Sie das SDK in ein Verzeichnis Ihrer Wahl.

Die Dateien dieses Supportpakets finden sich im Verzeichnis fetcp-lib.

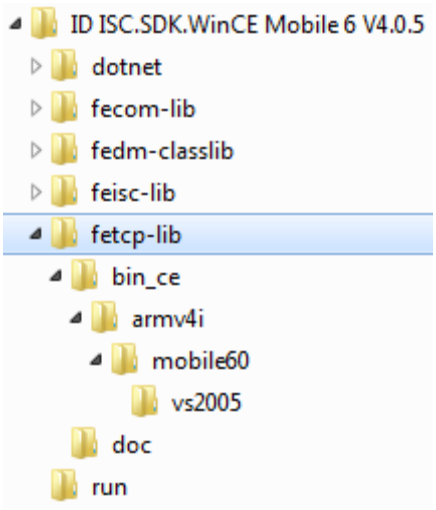
3.1. 32- und 64-Bit Windows XP/Vista/7/8



Wenn eigene Projekte nicht im SDK-Verzeichnis angelegt werden sollen, dann empfiehlt sich folgende Vorgehensweise:

- Kopieren Sie FETCP.DLL in das Verzeichnis des Anwendungsprogramms (empfohlen) oder in das Systemverzeichnis von Windows.
- Kopieren Sie FETCP.LIB in das Projekt- oder LIB-Verzeichnis
- Kopieren Sie FETCP.H in das Projekt- oder INCLUDE-Verzeichnis

3.2. Windows CE

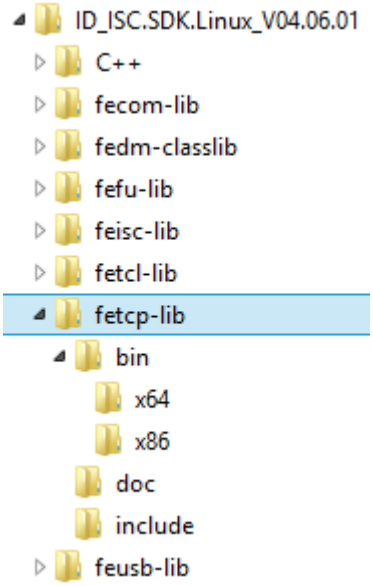


Wenn eigene Projekte nicht im SDK-Verzeichnis angelegt werden sollen, dann empfiehlt sich folgende Vorgehensweise:

- Kopieren Sie die Datei FETCPCE.DLL in das Systemverzeichnis des Windows CE Rechners.
- Kopieren Sie FETCPCE.LIB in das Projekt- oder LIB-Verzeichnis.
- Kopieren Sie FETCP.H in das Projekt- oder INCLUDE-Verzeichnis

Hinweis: die DLL kann nicht mit eMbedded Visual Basic 3.0 verwendet werden.

3.3. 32- und 64-Bit Linux



Zur Installation gibt es zwei Optionen:

Option 1: Falls eine install.sh im SDK-Verzeichnis vorliegt, führen Sie diese aus. Damit werden alle Bibliotheken in das Verzeichnis /usr/lib bzw. /usr/lib64 kopiert und alle symbolischen Links angelegt. Die Headerdatei können Sie in ein Verzeichnis Ihrer Wahl kopieren.

Option 2: Kopieren Sie die Dateien dieses Supportpakets in Verzeichnisse Ihrer Wahl und Erzeugen Sie symbolische Links auf die Bibliotheksdatei libfetcp.so.x.y.z[3] im Verzeichnis /usr/lib bzw. /usr/lib64 durch folgende Aufrufe:

cd /usr/lib (für 64 Bit : /usr/lib64)

ln –s /<Verzeichnis>/libfetcp.so.x.y.z libfetcp.so.x

ln –s /<Verzeichnis>/libfetcp.so.x libfetcp.so

ldconfig

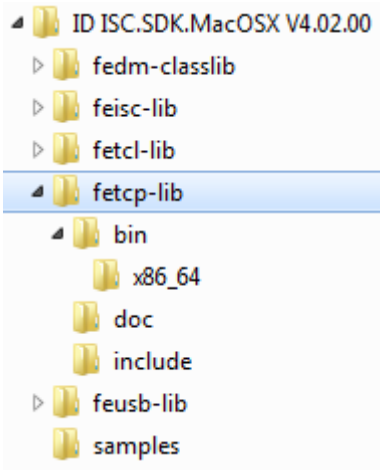
Anmerkung:

X86: Die Bibliothek wurde unter SuSE Linux 11.1 mit der GNU Compiler Collection V4.3.2 erstellt.

X64: Die Bibliothek wurde unter SuSE Linux 11.2 mit der GNU Compiler Collection V4.4.1 erstellt.

[3] *x.y.z repräsentiert die Versionsnummer der Bibliotheksdatei*

3.4. 64-Bit Mac OS X



Zur Installation gibt es zwei Optionen:

Option 1: Falls eine install.sh im SDK-Verzeichnis vorliegt, führen Sie diese aus. Damit werden alle Bibliotheken in das Verzeichnis /usr/local/lib kopiert und alle symbolischen Links angelegt. Die Headerdatei können Sie in ein Verzeichnis Ihrer Wahl kopieren.

Option 2: Kopieren Sie die Dateien dieses Supportpakets in Verzeichnisse Ihrer Wahl und Erzeugen Sie symbolische Links auf die Bibliotheksdatei libfetcp.x.y.z.dylib[4] im Verzeichnis /usr/local/lib durch folgende Aufrufe:

cd /usr/local/lib

ln –s libfetcp.x.y.z.dylib libfetcp.x.dylib

ln –s libfetcp.x.dylib libfetcp.dylib

Anmerkung: Die Bibliothek wurde unter Mac OS X V10.7.3 mit Xcode V4.3.2 erstellt. Die Bibliothek ist mit der Architektur x86_64 kompatibel.

[4] *x.y.z repräsentiert die Versionsnummer der Bibliotheksdatei*

4. Einbindung in das Anwendungsprogramm

4.1. Unterstützte Entwicklungsumgebungen

Betriebssystem	Entwicklungsumgebung	Unterstützung
Windows XP / Vista / 7 / 8	Visual Studio	ja
	Borland C++ Builder	ja
	Embarcadero C++ Builder	ja
Windows CE	eMbedded Visual C++ 4	ja
	Visual Studio 2005 / 2008	ja
Linux	GCC	ja
Mac OS X	GCC	ja, für Projekte mit x86_64 Architektur
	Xcode ≥ V4.3.2	ja, für Projekte mit x86_64 Architektur

4.2. Einbindung in Visual Studio

1. Include-Pfad zur Headerdatei in den Projekteinstellungen (Kategorie C/C++) hinzufügen
2. die LIB-Datei in den Projekteinstellungen (Kategorie Linker) eintragen

4.3. Einbindung in Xcode

1. Pfad zur Headerdatei in den Projekteinstellungen (Kategorie Search Paths und dort für User Header Search Paths) hinzufügen
2. die DYLIB-Datei per Drag-and-Drop dem Projekt hinzufügen

5. Programmierschnittstelle

5.1. Übersicht

Die FETCP Bibliothek kapselt für den Anwender alle notwendigen Funktionen und Parameter zum Verwalten von einem oder mehreren gleichzeitig geöffneten TCP/IP-Kanälen (Sockets). Der objektorientierte innere Aufbau (s. Abb. 1) ist nach außen hin bewusst als eine Funktionsschnittstelle herausgeführt. Dies hat den Vorteil der Sprachunabhängigkeit.

Die Bibliothek hat eine Selbstverwaltung, die ein Anwendungsprogramm davon befreit, irgendwelche Werte, Einstellungen oder Sonstiges zwischenspeichern zu müssen: Der Treiber-Manager in FETCP führt eine Liste mit allen erzeugten Socket-Objekten und jedes Socket-Objekt verwaltet alle für seine Schnittstelle relevanten Einstellungen innerhalb seines lokalen Speichers.

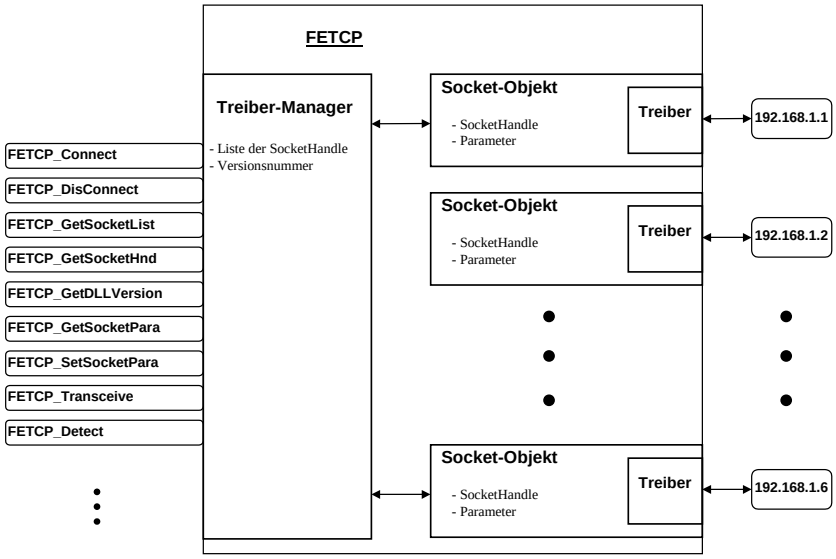


Abbildung 1: Interner Aufbau von FETCP

Vor der ersten Kommunikation muss ein Socket-Objekt angelegt werden. Dies wird von der Funktion **FETCP_Connect** automatisch ausgeführt. Wenn diese Funktion fehlerfrei ausgeführt werden konnte, erhält man mit dem Rückgabewert einen Handle, der vom Anwendungsprogramm verwaltet werden kann. Nur mit diesem Handle ist eine eindeutige Identifikation des geöffneten Socket-Objekts möglich. Der oder die Handle müssen aber nicht im Anwendungsprogramm gespeichert werden, denn der Treiber-Manager der Bibliothek verwaltet intern eine Liste aller geöffneten Sockets. Diese Liste kann mit der Funktion **FETCP_GetSocketList** abgerufen werden. Mit den Handle, die man damit sukzessive erhält, kann man anschließend mit der Funktion **FETCP_GetSocketPara** alle diesen Socket betreffenden Einstellungen, einschließlich der Hostadresse und der Portnummer, auslesen.

Ein mit **FETCP_Connect** erzeugtes Socket-Objekt muss unbedingt wieder mit der Funktion **FETCP_DisConnect**, die auch den Socket schließt, aus dem Speicher entfernt werden.

Wird ein Anwendungsprogramm mehrfach aufgerufen, erhält jedes Programm (Instanz) mit dem Funktionsaufruf **FETCP_GetSocketList** eine leere Socket-Liste. Dadurch wird eine Vermischung von Zugriffsrechten unter verschiedenen Programm-Instanzen verhindert.

Bei Kommunikationsfehlern mit **FETCP_Transceive**, **FETCP_Transmit** oder **FETCP_Receive** muss die Verbindung mit **FETCP_DisConnect** beendet und mit **FETCP_Connect** neu aufgebaut werden.

Jede Bibliotheks-Funktion (Ausnahme: **FETCP_GetDLLVersion**) hat einen Rückgabewert, der im Fehlerfall immer negativ ist.

Die Kommunikation über TCP/IP ist nicht trivial. Insbesondere die Fehlerbehandlung kann sehr aufwändig sein. Gute Unterlagen zu Grundlagen der TCP/IP-Kommunikation finden sich im Internet.

5.2. Threadsicherheit

Alle FEIG-Bibliotheken sind prinzipiell nicht vollständig threadsicher. Unter Beachtung einiger Regeln kann man dennoch Parallelität in der Ausführung von Kommunikationsaufgaben und damit praktische Threadsicherheit erreichen. Man muss auch wissen, dass alle OBID® RFID-Leser immer nur eine Aktion ausführen können, also synchron arbeiten.

Auf der Ebene der Transportschicht (FECOM, FEUSB, FETCP) kann über jede Verbindung nur synchron kommuniziert werden, weil auch die OBID-Leser nur synchron arbeiten. Threadsicher sind die Port-Objekte untereinander, weil diese unabhängig voneinander sind. Es ist demnach möglich, dass z. B. zwei Threads mit zwei OBID-Lesern über zwei verschiedene TCP-Verbindungen kommunizieren.

5.3. Fehlerbehandlung bei Kommunikationsproblemen

TCP/IP basierte Kommunikation ist normalerweise unproblematisch. Tritt jedoch ein Kommunikationsproblem auf, unterscheidet sich die Fehlerbehandlung von der bei seriell oder USB basierter Kommunikation.

Die folgenden Problemfälle werden diskutiert:

- Kommunikationsfehler
- Fehler beim Verbindungsaufbau
- Fehler beim Verbindungsabbau
- Verhalten bei unterbrochener Verbindung

5.3.1. Kommunikationsfehler

Wenn die Funktion `FETCP_Transceive` mit den Fehlercodes -1230 (Timeout), -1232 (Error in read process) oder -1237 (error in send process) zurückkehrt, muss die Verbindung sofort geschlossen und erneut wieder aufgebaut werden.

Wenn ein Timeout ignoriert wird und ein weiteres OBID-Protokoll an den Leser gesendet wird, kann in der Zwischenzeit das vorhergehende Antwortprotokoll empfangen worden sein. Dies führt zu einem dauerhaften Versatz der Protokolle und kann nur durch einen Verbindungsabbau mit anschließendem Verbindungsaufbau gelöst werden.

Die voreingestellte Timeout von 3000ms ist normalerweise ausreichend dimensioniert. In seltenen Fällen kann man mit der Funktion `FETCP_SetPortPara` diese Timeout vergrößern.

5.3.2. Fehler beim Verbindungsaufbau

Fehler beim Verbindungsaufbau mit der Funktion `FETCP_Connect` müssen in Abhängigkeit des Fehlercodes im Detail analysiert werden.

Fehlercode	Fehlerbehandlung
-1211	Timeout beim Verbindungsaufbau zum OBID-Leser. Dies ist ein normales Laufzeitproblem. Ursache könnte eine andere bestehende Verbindung sein. OBID-Leser können prinzipiell immer nur eine Verbindung annehmen. Andere Ursachen, die von einem Installationsteam gelöst werden müssen: - Die Stromversorgung des OBID-Lesers ist nicht eingeschaltet - Der OBID-Leser befindet sich in einem anderen Subnetz - Der OBID-Leser ist bezüglich seiner LAN-Parameter falsch eingestellt.
-1212	Der Parameter cHostAdr enthält eine ungültige Zeichenkette. Dieser Fehler kann nur vom Entwicklungsteam analysiert und gelöst werden.
-1251	Der Parameter iPortNr ist außerhalb des Wertebereichs. Dieser Fehler kann nur vom Entwicklungsteam analysiert und gelöst werden.

5.3.3. Fehler beim Verbindungsabbau

Der Verbindungsabbau wird intern mit der Systemfunktion closesocket (Windows) bzw. close (Linux, Mac OS X) realisiert. Diese Systemfunktionen arbeiten asynchron und kehren vor dem Schließen der Verbindung zurück. Deshalb ist nicht sichergestellt, dass der finale TCP-Status TIME_WAIT nach dem Rücksprung aus der Funktion `FETCP_DisConnect` eingenommen wird. Mit dem Rückgabewert von `FETCP_DisConnect` wird deshalb u.a. der letzte TCP-Status der Applikationsebene angezeigt. Ist der Verbindungsabbau noch nicht abgeschlossen, darf kein neuer Verbindungsaufbau erfolgen. Man muss in diesem Fall mit wiederholten Aufrufen der Funktion `FETCP_GetSocketState(cHostAdr, iPortNr)` das Schließen der Verbindung beobachten.

Zwei Fehlersituationen können mit der Funktion `FETCP_DisConnect` auftreten:

Fehlercode	Fehlerbehandlung
-1213	Der Verbindungsabbau konnte vom Betriebssystem nicht ausgeführt werden. Die Verbindung bleibt bestehen. Lösung: Der Verbindungsabbau muss wiederholt warden, bis er erfolgt ist.
1 - 10	Der Verbindungsabbau konnte vom Betriebssystem eingeleitet werden, aber der letzte TCP-Status ist nicht TIME_WAIT. Die Rückgabewerte 1 bis 10 signalisieren den zuletzt eingenommenen TCP-Status (siehe auch: 5.7. Liste der TCP-Status)

5.3.4. Verhalten bei unterbrochener Verbindung – die Keep-Alive Option

Bei einer Unterbrechung der aktiven Verbindung durch z.B. Kabelbruch kann die Serverseite (OBID-Leser) keinen Fehler feststellen. Auf der anderen Seite kann auch die Clientseite (Host) ohne Kommunikation keinen Fehler detektieren. Beides sieht die TCP-Spezifikation nicht vor. Erst mit dem nächsten OBID-Protokoll wird der Host in einen Fehler laufen und die Verbindung schließen und wieder öffnen wollen. Doch der OBID-Leser wird den Verbindungsaufbau nie mitgeteilt bekommen, auch wenn die Ursache behoben ist, denn der OBID-Leser hält eine halb-offene Verbindung.

Die Lösung für dieses sehr realistische Szenario ist die Aktivierung der Keep-Alive Option im OBID-Leser. Damit sendet der Leser während der Kommunikationspausen auf der IP-Ebene zyklisch kleine Pakete, die der Host beantworten muss. Werden durch z.B. Kabelbruch die Antworten verhindert, wird der OBID-Leser nach einer einstellbaren Zeit seine Verbindung selbst schließen. Jeder OBID i-*scan*® und OBID® *classic-pro* Leser mit Ethernet Schnittstelle hat Keep-Alive Parameter und es wird dringend empfohlen, diese Option zu aktivieren.

5.4. Liste der Funktionen

Hinweis: UCHAR wird als Abkürzung (#define) für "unsigned char" verwendet.

- **int FETCP_Connect(** char* cHostAdr, int iPortNr **)**
- **int FETCP_DisConnect(** int iSocketHnd **)**
- **int FETCP_GetSocketState(** char* cHostAdr, int iPortNr **)**
- **int FETCP_Detect(**char* cHostAdr, int iPortNr **)**
- **int FETCP_GetSocketList(** int iNext **)**
- **void FETCP_GetDLLVersion(** char* cVersion **)**
- **int FETCP_GetErrorText(** int iErrorCode, char* cErrorText **)**
- **int FETCP_GetLastError(** int iSocketHnd , int* iErrorCode, char* cErrorText **)**
- **int FETCP_GetSocketHnd(**char* cHostAdr, int iPortNr **)**
- **int FETCP_GetSocketPara(** int iSocketHnd, char* cPara, char* cValue **)**
- **int FETCP_SetSocketPara(** int iSocketHnd, char* cPara, char* cValue **)**
- **int FETCP_AddEventHandler**[5]**(** int iSocketHnd, FETCP_EVENT_INIT* pInit **)**
- **int FETCP_DelEventHandler**[4]**(** int iSocketHnd, FETCP_EVENT_INIT* pInit **)**
- **int FETCP_Transceive(** int iSocketHnd, UCHAR* cSendProt, int iSendLen, UCHAR* cRecProt, int iRecLen **)**
- **int FETCP_Transmit(** int iSocketHnd, UCHAR* cSendProt, int iSendLen **)**
- **int FETCP_Receive(** int iSocketHnd, UCHAR* cRecProt, int iRecLen **)**

[5] für zukünftige Erweiterungen

5.4.1. FETCP_Connect

Funktion	Öffnet einen TCP/IP-Kanal (Socket) zur Kommunikation mit einem OBID®-Leser (TCP/IP-Server).
Syntax	**int FETCP_Connect(char* cHostAdr, int iPortNr)**
Beschreibung	Die Funktion öffnet einen TCP/IP-Kanal, stellt eine Verbindung zum TCP/IP-Server her und legt intern eine Schnittstellenstruktur zur Verwaltung der Parameter an. Zur nachträglichen Änderung dieser Parameter kann die Funktion **FETCP_SetSocketPara** verwendet werden. Mit **FETCP_GetSocketPara** können diese Parameter ausgelesen werden. Der zurückgelieferte Handle *iSocketHnd* identifiziert die Schnittstelle von außen. *cHostAdr* ist eine nullterminierte Zeichenkette mit der Host-Adresse (z.B. "192.168.1.1"). *iPortNr* ist die Nummer des Ports. Der mit **FETCP_Connect** geöffnete Socket muß (!) mit der Funktion **FETCP_DisConnect** wieder geschlossen werden. Andernfalls wird der von der Bibliothek reservierte Speicher nicht wieder freigegeben. Bei Kommunikationsfehlern mit **FETCP_Transceive**, **FETCP_Transmit** oder **FETCP_Receive** muss die Verbindung mit **FETCP_DisConnect** beendet und mit **FETCP_Connect** neu aufgebaut werden.
Rückgabewert	Wenn der Socket fehlerfrei geöffnet und eine Verbindung zum TCP/IP-Server hergestellt wurde, wird ein Handle (>0) zurückgeliefert. Im Fehlerfall liefert die Funktion einen Wert kleiner als Null zurück. Die Liste der Fehlercodes findet sich im Anhang.

Beispiel	...
	#include "fetcp.h"
	...
	...
	int handle = FETCP_Connect("192.168.1.1", 4000);
	if(handle < 0)
	{
	// hier Code für den Fehlerfall
	}
	else
	{ // Kommunikation möglich
	// hier Code für Kommunikation oder anderes
	}

5.4.2. FETCP_DisConnect

Funktion	Beendet eine TCP/IP-Verbindung zu einem OBID®-Leser und schließt den Socket.
Syntax	**int FETCP_DisConnect(int iSocketHnd)**
Beschreibung	Die Funktion beendet eine TCP/IP-Verbindung zu einem OBID®-Leser, schließt den durch den Parameter *iSocketHnd* angegebenen Socket und gibt den reservierten Speicher wieder frei. Nach dem Schließen des Socket wird geprüft, ob der Socket den Status TIME_WAIT erreicht hat (s. 5.7. Liste der TCP-Status).
Rückgabewert	Der Rückgabewert ist 0, wenn der Socket geschlossen und der Status TIME_WAIT erreicht wurde. Konnte der Socket geschlossen werden, TIME_WAIT aber nicht innerhalb von 500ms erreicht werden, wird der ermittelte TCP-Status (>0) zurückgegeben. Im Fehlerfall liefert die Funktion einen Wert kleiner als Null zurück. Die Liste der Fehlercodes findet sich im Anhang.

Beispiel	...
	#include "fetcp.h"
	...
	...
	int Err;
	...
	int handle = FETCP_Connect("192.168.1.1", 4000);
	if(handle < 0)
	{
	// hier Code für den Fehlerfall
	}
	...
	...
	if(handle > 0)
	{
	Err = FETCP_DisConnect(handle);
	if(Err == 0)
	{
	// OK
	}
	else if(Err > 0)
	{
	// Status TIME_WAIT nicht erreicht. Err enthält letzten TCP-Status
	}
	else
	{
	// <0 bedeutet Fehlerfall
	}
	}
	...
	...

5.4.3. FETCP_Detect

Funktion	Prüft, ob ein TCP/IP-Server erreichbar ist.
Syntax	**int FETCP_Detect(char* cHostAdr, int iPortNr)**
Beschreibung	Die Funktion prüft, ob ein TCP/IP-Server erreichbar ist. Die Verbindung, sofern sie zustande kommt, wird sofort wieder beendet. *cHostAdr* ist eine nullterminierte Zeichenkette mit der Host-Adresse (z.B. "192.168.1.1"). *iPortNr* ist die Nummer des Ports.
Rückgabewert	Bei erfolgreichem Verbindungstest wird eine 0 zurückgegeben, andernfalls FETCP_ERR_SERVER_NOT_FOUND. Im Fehlerfall liefert die Funktion einen Wert kleiner als Null zurück. Die Liste der Fehlercodes findet sich im Anhang.

Beispiel	...
	#include "fetcp.h"
	...
	if(0 == FETCP_Detect("192.168.1.1", 4000);
	{
	// Verbindung zum TCP/IP-Server möglich
	}

5.4.4. FETCP_GetSocketState

Funktion	Ermittelt den Status einer Verbindung.
Syntax	**int FETCP_GetSocketState(char* cHostAdr, int iPortNr)**
Beschreibung	Die Funktion prüft für einen TCP/IP-Kanal, welchen Status die Verbindung zum TCP/IP-Server eingenommen hat. Nach einem Verbindungsaufbau mit **FETCP_Connect** sollte der Status ESTABLISHED sein (s 5.7. Liste der TCP-Status). Nach einem Verbindungsabbau mit **FETCP_DisConnect** sollte der Status TIME_WAIT sein. Diese Funktion kann nach Kommunikationsfehlern eingesetzt um zu prüfen, ob die Verbindung noch besteht. *cHostAdr* ist eine nullterminierte Zeichenkette mit der Host-Adresse (z.B. "192.168.1.1"). *iPortNr* ist die Nummer des Ports.
Rückgabewert	Der Status der Verbindung (>0). Im Fehlerfall liefert die Funktion einen Wert kleiner als Null zurück. Die Liste der Fehlercodes findet sich im Anhang.

Beispiel	...
	#include "fetcp.h"
	...
	...
	int status = FETCP_GetSocketState("192.168.1.1", 4000);
	if(status < 0)
	{
	// hier Code für den Fehlerfall
	}
	else
	{ // Status wurde ermittelt
	// status enthält einen Wert zwischen 1 und 12
	}

5.4.5. FETCP_GetSocketList

Funktion	Ermittelt in Abhängigkeit vom Parameter *iNext* den ersten oder den nachfolgenden Socket-Handle aus der internen Liste der geöffneten Sockets.
Syntax	**int FETCP_GetSocketList(int iNext)**
Beschreibung	Die Funktion gibt ein Socket-Handle aus der internen Liste der Socket-Handles zurück. Übergibt man für *iNext* eine 0, wird der erste Eintrag aus der Liste zurückgegeben. Übergibt man mit *iNext* ein in der Liste geführten Socket-Handle, wird der dem Socket-Handle nachfolgende Eintrag ermittelt und zurückgegeben. Man kann auf diese Weise durch sukzessives Einsetzen des Rückgabewertes die Liste von vorne nach hinten durchlaufen und alle Einträge abrufen.
Rückgabewert	Wenn ein Eintrag gefunden wurde, wird mit dem Rückgabewert der Socket-Handle geliefert. Ist das Ende der internen Liste erreicht, also der übergebene Socket-Handle keinen Nachfolger hat, wird eine 0 zurückgegeben. Ist kein Socket geöffnet, wird FETCP_ERR_EMPTY_LIST zurückgeliefert. Im Fehlerfall liefert die Funktion einen Wert kleiner als Null zurück. Die Liste der Fehlercodes findet sich im Anhang.

Beispiel	#include "fetcp.h"
	...
	// Funktion ermittelt die Einstellungen aller geöffneten Sockets
	void TcpList(void)
	{ int iNextHnd = FETCP_GetSocketList(0); // den ersten Handle ermitteln
	while(iNextHnd > 0)
	{ // hier beliebiger Code
	...
	iNextHnd = FETCP_GetSocketList(iNextHnd); // nächsten Handle ermitteln
	}
	...
	// hier z. B. Code zum Anzeigen der Liste
	...
	}

Tip	Beim Schließen aller geöffneten Sockets bedient man sich gerne einer Schleife, ähnlich der im oberen Beispiel. Nur muß man bedenken, dass man von einem geschlossenen Socket keinen Nachfolger mehr ermitteln kann. In dem folgenden Codefragment wird gezeigt, wie man in einer Schleife alle geöffneten Sockets schließen kann:
	...
	iNextHnd = FETCP_GetSocketList(0); // den ersten Handle ermitteln
	while(iNextHnd > 0)
	{ iCloseHnd = iNextHnd;
	iNextHnd = FETCP_GetSocketList(iNextHnd); // erst nächsten Handle ermitteln
	iError = FETCP_DisConnect(iCloseHnd); // jetzt erst Socket schließen
	}
	...

5.4.6. FETCP_GetDLLVersion

Funktion	Ermittelt die Versionsnummer der DLL/SO.
Syntax	**void FETCP_GetDLLVersion(char* cVersion)**
Beschreibung	Die Funktion gibt die Versionsnummer der DLL/SO/DYLIB zurück. *cVersion* ist eine leere, nullterminierte Zeichenkette zur Rückgabe der Versionsnummer. Die Zeichenkette sollte wenigstens 256 Zeichen aufnehmen können. In der Zeichenkette wird die aktuellen Versionsnummer zurückgegeben (z.B. "02.02.00"). Neuere Versionen könnten aber weitere Informationen liefern.
Rückgabewert	ohne
Beispiel	... #include "fetcp.h" ... ... char cVersion[256]; FETCP_GetDLLVersion(cVersion); // hier Code zum Anzeigen der Versionsnummer ... ...

5.4.7. FETCP_GetErrorText

Funktion	Ermittelt Fehlertext zum Fehlercode
Syntax	**int FETCP_GetErrorText(int iErrorCode, char* cErrorText)**
Beschreibung	Die Funktion übergibt in *cErrorText* den zum *iErrorCode* zugehörigen englischen Fehlertext. Der Puffer für *cErrorText* sollte 256 Zeichen aufnehmen können.
Rückgabewert	Im fehlerfreien Fall liefert die Funktion Null und im Fehlerfall einen Wert kleiner als Null zurück. Die Liste der Fehlercodes findet sich im Anhang.
Beispiel	... #include "fetcp.h" ... ... char cErrorText[256]; ... int iBack = FETCP_GetErrorText(FETCP_ERR_TIMEOUT, cErrorText) // hier Code zum Anzeigen des Textes ... ...

5.4.8. FETCP_GetLastError

Funktion	Ermittelt den letzten Fehlercode und übergibt Fehlertext
Syntax	**int FETCP_GetLastError(int* iErrorCode, char* cErrorText)**
Beschreibung	Die Funktion übergibt in *iErrorCode* den letzten Fehlercode und in *cErrorText* den zugehörigen englischen Fehlertext zurück. Der Puffer für *cErrorText* sollte 256 Zeichen aufnehmen können.
Rückgabewert	Im fehlerfreien Fall liefert die Funktion Null und im Fehlerfall einen Wert kleiner als Null zurück. Die Liste der Fehlercodes findet sich im Anhang.

Beispiel	...
	#include "fetcp.h"
	...
	...
	char cErrorText[256];
	int iErrorCode = 0;
	...
	
	int iBack = FETCP_GetLastError(&iErrorCode, cErrorText)
	// hier Code zum Anzeigen des Textes
	...
	...

5.4.9. FETCP_GetSocketHnd

Funktion	Ermittelt von einem geöffneten TCP/IP-Port den Socket-Handle.
Syntax	**int FETCP_GetSocketHnd(char* cHostAdr, int iPortNr)**
Beschreibung	Mit dieser Funktion kann auf einfache Weise der Socket-Handle eines zuvor geöffneten TCP/IP-Ports ermittelt werden. *cHostAdr* ist eine nullterminierte Zeichenkette mit der Host-Adresse (z.B. "192.168.1.1"). *iPortNr* ist die Nummer des Ports.
Rückgabewert	Wenn der angegebene TCP/IP-Port in der internen Socket-Liste gefunden wurde, wird der Socket-Handle (>0) zurückgeliefert. Konnte in der Socket-Liste der gesuchte TCP/IP-Port nicht gefunden werden, wird FETCP_ERR_NO_HND_FOUND zurückgegeben. Im Fehlerfall liefert die Funktion einen Wert kleiner als Null zurück. Die Liste der Fehlercodes findet sich im Anhang.

Beispiel	...
	#include "fetcp.h"
	...
	...
	int handle = FETCP_Connect("192.168.1.1", 4000);
	if(handle < 0)
	{
	// hier Code für den Fehlerfall
	}
	else
	{ // handle wird mittels Host-Adresse und Port-Nummer erneut ermittelt
	handle = FETCP_GetSocketHnd("192.168.1.1", 4000);
	}

5.4.10. FETCP_GetSocketPara

Funktion	Ermittelt von dem mit *iSocketHnd* bestimmten TCP/IP-Port einen Parameter.
Syntax	**Int FETCP_GetSocketPara(int iPortHnd, char* cPara, char* cValue)**
Beschreibung	Die Funktion ermittelt den aktuellen Wert eines Parameters. *cPara* ist eine nullterminierte Zeichenkette mit der Parameterkennung *cValue* ist eine leere, nullterminierte Zeichenkette zur Rückgabe des Parameterwerts. Die Zeichenkette sollte wenigstens 128 Zeichen aufnehmen können.
Parameter-kennungen	Die Parameterkennungen sind: HostAdr, PortNr, Timeout, CharTimeout, Language. Der Parameter Language setzt die Sprache in der DLL und ist ein globaler, also nicht an ein Socket-Handle gebundener Wert. Man setzt deshalb *iSocketHnd* in diesem Fall auf 0.
Rückgabewert	Im fehlerfreien Fall liefert die Funktion den Wert 0 und im Fehlerfall einen Wert kleiner als Null zurück. Die Liste der Fehlercodes findet sich im Anhang.
Querverweis	Weitere Informationen in: 5.6. Liste der Parameterkennungen.

Beispiel	...
	#include "fetcp.h"
	...
	...
	char cValue[128];
	...
	if(!FETCP_GetSocketPara(handle, "HostAdr", cValue))
	{
	// hier Code zum Anzeigen der Host-Adresse
	...
	}
	...
	...
	}

5.4.11. FETCP_SetSocketPara

<table>
<tr><td>Funktion</td><td colspan="4">Setzt einen Parameter für einen TCP/IP-Port auf einen neuen Wert.</td></tr>
<tr><td>Syntax</td><td colspan="4">int FETCP_SetSocketPara(int iPortHnd, char* cPara, char* cValue)</td></tr>
<tr><td>Beschreibung</td><td colspan="4">Die Funktion übergibt an den mit iSocketHnd benannten TCP/IP-Port einen neuen Parameter.

cPara ist eine nullterminierte Zeichenkette mit der Parameterkennung.

cValue ist eine nullterminierte Zeichenkette mit dem neuen Parameterwert.</td></tr>
<tr><td></td><td>Parameterkennung</td><td>Wertebereich</td><td>Defaultwert</td><td>Einheit</td></tr>
<tr><td></td><td>Timeout</td><td>0...99999</td><td>3000</td><td>ms</td></tr>
<tr><td></td><td>CharTimeout</td><td>1…999</td><td>25</td><td>ms</td></tr>
<tr><td></td><td>Language</td><td>7, 9</td><td>9</td><td>-</td></tr>
<tr><td></td><td>UseOBID</td><td>0, 1</td><td>1</td><td>-</td></tr>
<tr><td>Rückgabewert</td><td colspan="4">Im fehlerfreien Fall liefert die Funktion den Wert 0 und im Fehlerfall einen Wert kleiner als Null zurück. Die Liste der Fehlercodes findet sich im Anhang.</td></tr>
<tr><td>Querverweis</td><td colspan="4">Weitere Informationen in: 5.6. Liste der Parameterkennungen.</td></tr>
<tr><td>Beispiel</td><td colspan="4">...
#include "fetcp.h"
...
...
int Err;
...
int handle = FETCP_Connect("192.168.1.1", 4000);
if(handle > 0)
{
    Err = FETCP_SetSocketPara(handle, "Timeout", "5000");
    ...
}
...
...</td></tr>
</table>

5.4.12. FETCP_Transceive

Funktion	Funktion zur Kommunikation (Transmit and Receive).
Syntax	**int FETCP_Transceive(int iPortHnd, UCHAR* cSendProt, int iSendLen, UCHAR* cRecProt, int iRecLen)**
Beschreibung	Die Funktion schickt die in *cSendProt* enthaltenen Daten über den TCP/IP Port an ein angeschlossenes Gerät und hinterlegt in *cRecProt* die empfangenen Daten. Im Parameter *iSendLen* muß die Anzahl der Zeichen in *cSendProt* übergeben werden. Mit dem Parameter *iRecLen* muß die maximale Länge des Puffers *cRecProt* angegeben werden. Übersteigt die Anzahl der empfangenen Zeichen den in *iRecLen* übergebenen Wert, wird die Funktion sofort beendet. Die bis zum Abbruch empfangenen Zeichen werden in *cRecProt* hinterlegt. Vor der Kommunikation werden Sende- und Empfangspuffer gelöscht.
Rückgabewert	Im fehlerfreien Fall liefert die Funktion die Länge des Empfangsprotokolls und im Fehlerfall einen Wert kleiner als Null zurück. Die Liste der Fehlercodes findet sich im Anhang.

Beispiel	#include “fetcp.h”
	...
	int iSendLen;
	int iRecProtLen;
	char* cHost = “192.168.1.1”;
	int iPortNr = 10001;
	UCHAR cSendBuf[256]; // Adjust buffer size to transmit data if needed
	UCHAR cRecBuf[256]; // Adjust buffer size to receive data if needed
	…
	int handle = FETCP_Connect(cHost, iPortNr);
	if(handle < 0)
	{
	// code here for error condition
	}
	else
	{
	// the transmit protocol is gotten for example with a function and stored in SendBuf
	iSendLen = GetSendProtocol(cSendBuf);
	// Communication through the socket, if successful, the receive data are located in RecBuf
	iRecProtLen = FETCP_Transceive(handle, cSendBuf, iSendLen, cRecBuf, 256);
	if(cRecProtLen < 0)
	{
	// Communication error
	}
	}

5.4.13. FETCP_Transmit

Funktion	Funktion zur Ausgabe eines Protokolls.
Syntax	**int FETCP_Transmit(int iPortHnd, UCHAR* cSendProt, int iSendLen)**
Beschreibung	Die Funktion schickt die in *cSendProt* enthaltenen Daten über den TCP/IP Port an ein angeschlossenes Gerät und wartet nicht auf ein Antwortprotokoll. Im Parameter *iSendLen* muß die Anzahl der Zeichen in *cSendProt* übergeben werden. Vor dem Senden des Protokolls wird der Sendepuffer gelöscht. Zeichen, die noch auf die Ausgabe warten, gehen dadurch verloren. Die Funktion kehrt erst zurück, wenn alle Zeichen über die Schnittstelle ausgegeben wurden.
Rückgabewert	Im fehlerfreien Fall liefert die Funktion 0 und im Fehlerfall einen Wert kleiner als Null zurück. Die Liste der Fehlercodes findet sich im Anhang.

Beispiel	…
	#include “fetcp.h”
	...
	...
	int iErr;
	int iSendLen;
	char* cHost = “192.168.1.1”;
	int iPortNr = 10001;
	UCHAR cSendBuf[256]; // Buffer size may need to be adjusted to the send data
	…
	int handle = FETCP_Connect(cHost, iPortNr);
	if(handle < 0)
	{
	// code here for error condition
	}
	else
	{ // the transmit protocol is gotten for example with a function and stored in SendBuf
	iSendLen = GetSendProtocol(cSendBuf);
	// Communication through the socket
	iErr = FETCP_Transmit(handle, cSendBuf, iSendLen);
	if(iErr < 0)
	{
	// Communication error
	}
	}

5.4.14. FETCP_Receive

Funktion	Funktion zum Empfang eines Protokolls.
Syntax	**int FETCP_Receive(int iPortHnd, UCHAR* cRecProt, int iRecLen)**
Beschreibung	Die Funktion erwartet innerhalb der Zeit Timeout (s. 5.6. Liste der Parameterkennungen) über den TCP/IP Port Daten, liest sie aus und hinterlegt sie im Empfangspuffer *cRecProt*. Mit dem Parameter *iRecLen* muß die maximale Länge des Puffers *cRecProt* angegeben werden. Übersteigt die Anzahl der empfangenen Zeichen den in *iRecLen* übergebenen Wert, wird die Funktion sofort beendet. Die bis zum Abbruch empfangenen Zeichen werden in *cRecProt* hinterlegt. Die Funktion löscht nicht den Empfangspuffer. Dadurch ist gewährleistet, dass zuvor eingetroffene Zeichen nicht verloren gehen.
Rückgabewert	Im fehlerfreien Fall liefert die Funktion die Länge des Empfangsprotokolls und im Fehlerfall einen Wert kleiner als Null zurück. Die Liste der Fehlercodes findet sich im Anhang.

Beispiel	…
	#include “fetcp.h”
	…
	…
	int iRecProtLen;
	char* cHost = “192.168.1.1”;
	int iPortNr = 10001;
	UCHAR cRecBuf[256]; // Buffer size may need to be adjusted to the receive data
	…
	int handle = FETCP_Connect(cHost, iPortNr);
	if(handle < 0)
	{
	// code here for error condition
	}
	else
	{ // Communication through the socket, if successful the receive data will be located in RecBuf
	iRecProtLen = FETCP_Receive(handle, cRecBuf, 256);
	if(iRecProtLen < 0)
	{
	// Communication error
	}
	}

ANHANG

5.5. Fehlercodes

Fehler-Konstante	Wert	Beschreibung
FETCP_ERR_NEWSOCKET_FAILURE	-1200	Fehler beim Erzeugen eines neuen Socket-Objekts. Eventuell ist Speichermangel eine Ursache.
FETCP_ERR_EMPTY_LIST	-1201	Socket-Handleliste ist leer (keine Socket-Objekte angelegt)
FETCP_ERR_POINTER_IS_NULL	-1202	ein Zeiger ist Null, also ungültig
FETCP_ERR_NO_MEMORY	-1203	Speichermangel
FETCP_ERR_SERVER_NOT_FOUND	-1205	Rückgabe von FETCP_Detect, wenn zum angegebenen Server keine Verbindung möglich war.
FETCP_ERR_NO_CONNECTION	-1211	Timeout beim Verbindungsaufbau zum TCP/IP-Server. Ursache kann auch sein, dass ein anderer Client die Verbindung blockiert.
FETCP_ERR_SERVER_ADDR	-1212	der Parameter cHostAdr in den Funktionen FETCP_Detect oder FETCP_Connect ist strukturell fehlerhaft
FETCP_ERR_UNKNOWN_HND	-1220	der übergebene Socket-Handle ist unbekannt
FETCP_ERR_HND_IS_NULL	-1221	der übergebene Socket-Handle ist 0
FETCP_ERR_HND_IS_NEGATIVE	-1222	der übergebene Socket-Handle ist negativ
FETCP_ERR_NO_HND_FOUND	-1223	kein Socket-Handle in Socket-Handleliste gefunden
FETCP_ERR_TIMEOUT	-1230	Timeout beim Lesen vom Socket
FETCP_ERR_RECEIVE_PROCESS	-1232	Fehler im Empfangsprozess
FETCP_ERR_CHANGE_PORT_PARA	-1236	Fehler beim Ändern eines Socket-Parameters
FETCP_ERR_TRANSMIT_PROCESS	-1237	Fehler im Sendeprozess
FETCP_ERR_GET_CONNECTION_STATE	-1238	Fehler beim Abfragen des Verbindungsstatus
FETCP_ERR_UNKNOWN_PARAMETER	-1250	Übergabeparameter ist nicht bekannt
FETCP_ERR_PARAMETER_OUT_OF_RANGE	-1251	Übergabeparameter zu groß oder zu klein
FETCP_ERR_ODD_PARAMETERSTRING	-1252	eine nicht unterstützte Option wurde per Übergabeparameter aufgerufen
FETCP_ERR_UNKNOWN_ERRORCODE	-1254	unbekannter Fehlercode
FETCP_ERR_BUFFER_OVERFLOW	-1270	Bereitgestellter Empfangspuffer ist zu klein

5.6. Liste der Parameterkennungen

Parameterkennung	Wertebereich	Default	Einheit	Beschreibung
HostAdr				Host-Adresse (z.B. "192.168.1.1") Adresse kann nur gelesen werden.
PortNr	0…65535		-	Port-Nummer Port-Nummer kann nur gelesen werden.
Timeout	0...99999	3000	ms	Maximale Wartezeit auf Empfangsprotokoll
CharTimeout	1...999	25	ms	Die Zeichen-Timeout bestimmt, nach welcher Zeit nach dem Empfang des letzten Zeichens der Empfangsprozeß beendet wird.
Language	7 - Deutsch 9 - Englisch	9	-	Auswahl der Sprache für interne Textresourcen.
UseOBID	0, 1	1	-	Aktivierung der auf OBID-Protokolle angepasste Empfangsroutine

5.7. Liste der TCP-Status

TCP-Status	Wert
FETCP_STATE_CLOSED	1
FETCP_STATE_LISTEN	2
FETCP_STATE_SYN_SENT	3
FETCP_STATE_SYN_RCVD	4
FETCP_STATE_ESTABLISHED	5
FETCP_STATE_FIN_WAIT1	6
FETCP_STATE_FIN_WAIT2	7
FETCP_STATE_CLOSE_WAIT	8
FETCP_STATE_CLOSING	9
FETCP_STATE_LAST_ACK	10
FETCP_STATE_TIME_WAIT	11
FETCP_STATE_DELETE_TCB	12

5.8. Änderungshistorie

V2.00.00

- Neue Funktion: **FETCP_GetSocketState**
- Die Funktion **FETCP_DisConnect** hat jetzt auch positive Rückgabewerte. Dies kann Anpassungen in Applikationen notwendig machen. Es wird empfohlen, jede Codezeile mit **FETCP_DisConnect** genau zu untersuchen. Weitere Informationen finden sich im Kapitel **5.4.2. FETCP_DisConnect**
- Neuer Fehlercode -1238 (Fehler beim Lesen des Verbindungszustands)
- Windows / Windows CE:
  5. Migration der Entwicklungsumgebung von Visual Studio 6 zu Visual Studio 2008.
  6. Anpassung der Callback Funktionsdeklaration in ***struct _FETCP_EVENT_INIT*** bzgl. der Calling-Konvention. Daher ist diese Version nicht kompatibel zur Vorgängerversion und nicht kompatibel mit Anwendungen, die mit der Vorgängerversion kompiliert wurden. Code-anpassungen bzgl. dieser Änderung sind nicht notwendig, aber Anwendungen müssen neu kompiliert werden.

V1.02.05

- Interne Modifikation

V1.02.04

- Version für Windows CE
- Verbesserte Performanz in der Kommunikation durch spezielle, auf OBID-Protokolle abgestimmte Empfangsroutine. Abschaltbar durch den neuen Parameter UseOBID.

V1.02.03

- Modifizierte Lizenbestimmung
- Keine Längenlimits mehr in **FETCP_Transceive** und **FETCP_Transmit**
- **FETCP_Transmit** und **FETCP_Receive** für Visual Basic 6 verfügbar
- Die Linux-Version wurde mit dem Compiler GCC 3.3.3 unter SuSE Linux 9.1 erstellt

V1.02.00

- Neue Funktionen: **FETCP_Transmit**, **FETCP_Receive**
- Erstes Linux Release (SuSE Linux 8.2, GNU Compiler Collection V3.3-23, glibc V2.3.2-6)

V1.00.00

- Dies ist die erste Release-Version.