



ID FECOM

Version 2.08.04

Software-Support für serielle Schnittstelle

**Für 32-Bit Betriebssysteme
Windows 2000/XP/Vista
und Windows CE
und Linux**

Hinweis

© Copyright 1998-2008 by FEIG ELECTRONIC GmbH
Lange Straße 4
D-35781 Weilburg-Waldhausen
eMail: obid@feig.de

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

Weitergabe sowie Vervielfältigung dieses Dokuments, Verwertung und Mitteilung ihres Inhalts sind nicht gestattet, soweit nicht ausdrücklich zugestanden. Zuwiderhandlung verpflichtet zu Schadenersatz. Alle Rechte für den Fall der Patenterteilung oder Gebrauchsmuster-Eintragung vorbehalten.

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

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.

Die in diesem Handbuch gemachten Installationsempfehlungen gehen von günstigsten Rahmenbedingungen aus. FEIG ELECTRONIC GmbH übernimmt keine Gewähr für die einwandfreie Funktion einer OBID®-Anlage in systemfremden Umgebungen.

OBID® and OBID i-scan® are registered trademarks of FEIG ELECTRONIC GmbH.

Windows is a registered trademark of Microsoft Corporation in the United States and other countries.

Windows Vista is either a registered trademark or trademark of Microsoft Corporation in the United States and/or other countries

Linux® is a registered Trademark of Linus Torvalds.

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 FECOM 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 FECOM.DLL, FECOMCE.DLL oder LIBFECOM.SO.x.y.z¹ ohne Abgabe von Lizenzgebühren vertreiben, unter der Voraussetzung, daß diese Anwendungsprogramme oder Programmbibliotheken dazu dienen, Geräte und / oder Anlagen anzusteuern oder zu betreiben, die von FEIG entwickelt und / oder vertrieben 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 dekompileieren, 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 wirtschaftlichen Zweck der zu ersetzenden Bestimmung am nächsten kommt.
3. Dieser Vertrag unterliegt dem Recht der Bundesrepublik Deutschland. Gerichtsstand ist, sofern Sie Vollkaufmann sind, Weilburg.

Inhalt:

1. Einleitung	6
2. Installation.....	7
2.1. 32-Bit Windows 2000/XP/Vista	7
2.2. Windows CE	7
2.3. 32-Bit Linux.....	8
3. Einbindung in das Anwendungsprogramm.....	9
4. Änderungen gegenüber der Vorversion	9
5. Programmierschnittstelle.....	10
5.1. Übersicht.....	10
5.2. Liste der Funktionen	11
5.3. Ereignissignalisierung für Steuerleitungen.....	12
5.3.1. FECOM_OpenPort	13
5.3.2. FECOM_ClosePort	14
5.3.3. FECOM_DetectPort.....	15
5.3.4. FECOM_GetPortList.....	16
5.3.5. FECOM_GetDLLVersion	17
5.3.6. FECOM_GetErrorText	17
5.3.7. FECOM_GetLastError	18
5.3.8. FECOM_GetPortHnd.....	19
5.3.9. FECOM_GetPortPara.....	20
5.3.10. FECOM_SetPortPara	21
5.3.11. FECOM_DoPortCmd	22
5.3.12. FECOM_AddEventHandler.....	23
5.3.13. FECOM_DelEventHandler.....	25
5.3.14. FECOM_Transceive	26
5.3.15. FECOM_Transmit.....	27
5.3.16. FECOM_Receive	28
6. Anhang	29
6.1. Fehlercodes	29
6.2. Liste der Parameterkennungen.....	31
6.3. Liste der Konstanten für die FECOM_EVENT_INIT-Struktur	33
6.4. Änderungshistorie	34

1. Einleitung

Das Supportpaket ID FECOM 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 zur seriellen Schnittstelle des PC unter 32-Bit Windows 2000, XP und Vista, sowie Windows CE und 32-Bit Linux, angeboten und ist speziell für die gemeinsame Verwendung mit weiteren Supportpaketen (z.B. ID FERW, ID FERWA, ID FEISC) entwickelt worden.

Das Supportpaket für Windows 2000, XP und Vista besteht aus folgenden Komponenten:

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

Das Supportpaket für Windows CE besteht aus folgenden Komponenten:

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

Das Supportpaket für 32-Bit Linux besteht aus folgenden Komponenten:

File	Use
LIBFECOM.SO.x.y.z ¹	Funktions-Bibliothek mit allen Funktionen
FECOM.H	Header-Datei für C/C++-Projekte

Anmerkung: Die Bibliothek wurde unter SuSE Linux 9.1 mit der GNU Compiler Collection V3.3.3 erstellt.

¹ x.y.z repräsentiert die Versionsnummer der Bibliotheksdatei

2. Installation

2.1. 32-Bit Windows 2000/XP/Vista

Die Installation muß von Hand ausgeführt werden:

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

2.2. Windows CE

Die Installation muß von Hand ausgeführt werden:

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

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

2.3. 32-Bit Linux

Die Installation gestaltet sich sehr einfach:

- Kopieren Sie die Dateien in Verzeichnisse Ihrer Wahl.
- Erzeugen Sie einen symbolischen Link auf die Bibliotheksdatei `libfecom.so.x.y.z`¹ im Verzeichnis `/usr/lib` durch folgende Aufrufe:

```
cd /usr/lib
```

```
ln -s /<Verzeichnis>/libfecom.so.x.y.z libfecom.so.x
```

```
ln -s /<Verzeichnis>/libfecom.so.x libfecom.so
```

```
ldconfig
```

Anmerkung: Die Bibliothek wurde unter SuSE Linux 9.1 mit der GNU Compiler Collection V3.3.3 erstellt.

¹ x.y.z repräsentiert die Versionsnummer der Bibliotheksdatei

3. Einbindung in das Anwendungsprogramm

Wenn die LIB-Datei (nur Windows) dem Entwicklungswerkzeug bekannt gemacht wurde, kann sofort jede Funktion verwendet werden. Voraussetzung ist natürlich die Deklaration der DLL/SO-Funktionen mit einer include-Anweisung innerhalb jeder Source-Datei, die FECOM-Funktionen aufrufen.

4. Änderungen gegenüber der Vorversion

- Windows: Verbesserungen für den Remote-Einsatz unter Windows Server
- Linux:

Bitte beachten Sie auch die Änderungshistorie im Anhang.

5. Programmierschnittstelle

5.1. Übersicht

Die FECOM kapselt für den Anwender alle notwendigen Funktionen und Parameter zum Verwalten von einer oder mehreren gleichzeitig geöffneten seriellen Schnittstellen. 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 FECOM führt eine Liste mit allen erzeugten Port-Objekten und jedes Port-Objekt verwaltet alle für seine Schnittstelle relevanten Einstellungen innerhalb seines lokalen Speichers.

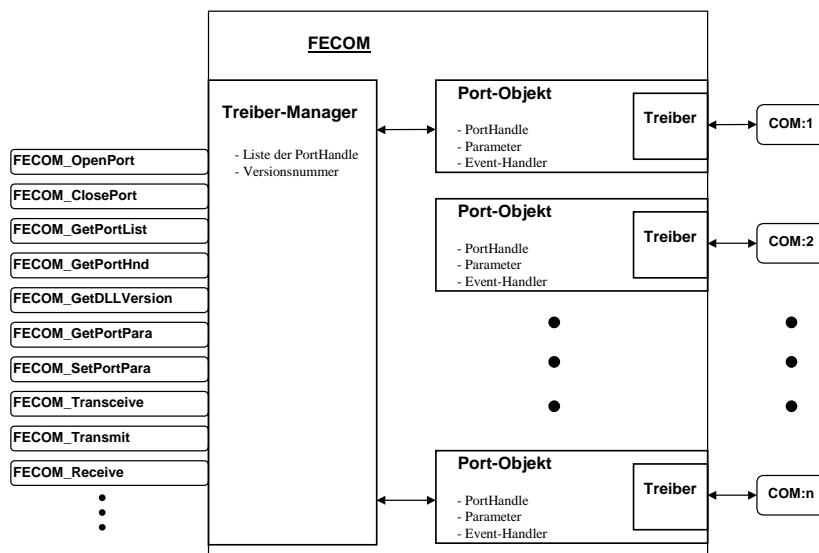


Abbildung 1: Interner Aufbau von FECOM

Vor der ersten Kommunikation muß ein Port-Objekt angelegt werden. Dies wird von der Funktion **FECOM_OpenPort** 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 Port-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 COM-Ports. Diese Liste kann mit der Funktion **FECOM_GetPortList** abgerufen werden. Mit den Handle, die man damit sukzessive erhält, kann man anschließend mit der Funktion **FECOM_GetPortPara** alle diesen Port betreffenden Einstellungen, einschließlich der Portnummer, auslesen.

Ein mit **FECOM_OpenPort** erzeugtes Port-Objekt muß unbedingt wieder mit der Funktion **FECOM_ClosePort**, die auch den COM-Port schließt, aus dem Speicher entfernt werden.

Wird ein Anwendungsprogramm mehrfach aufgerufen, erhält jedes Programm (Instanz) mit dem Funktionsaufruf **FECOM_GetPortList** eine leere Port-Liste. Dadurch wird eine Vermischung von Zugriffsrechten unter verschiedenen Programm-Instanzen verhindert. Allerdings kann ein COM-Port nur einmal geöffnet werden, da er physisch auch nur einmal vorhanden ist.

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

5.2. Liste der Funktionen

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

- **int FECOM_OpenPort(char* cPortNr)**
- **int FECOM_ClosePort(int iPortHnd)**
- **int FECOM_DetectPort(int iPortNr)**
- **int FECOM_GetPortList(int iNext)**
- **void FECOM_GetDLLVersion(char* cVersion)**
- **int FECOM_GetErrorText(int iErrorCode, char* cErrorText)**
- **int FECOM_GetLastError(int iPortHnd , int* iErrorCode, char* cErrorText)**
- **int FECOM_GetPortHnd(char* cPortNr)**
- **int FECOM_GetPortPara(int iPortHnd, char* cPara, char* cValue)**
- **int FECOM_SetPortPara(int iPortHnd, char* cPara, char* cValue)**
- **int FECOM_DoPortCmd(int iPortHnd, char* cCmd, char* cValue)**
- **int FECOM_AddEventHandler(int iPortHnd, FECOM_EVENT_INIT* pInit)**
- **int FECOM_DelEventHandler(int iPortHnd, FECOM_EVENT_INIT* pInit)**
- **int FECOM_Transceive(int iPortHnd, UCHAR* cSendProt, int iSendLen, UCHAR* cRecProt, int iRecLen)**
- **int FECOM_Transmit(int iPortHnd, UCHAR* cSendProt, int iSendLen)**
- **int FECOM_Receive(int iPortHnd, UCHAR* cRecProt, int iRecLen)**

5.3. Ereignissignalisierung für Steuerleitungen

Für die Steuerleitungen DTR, RTS, CTS, DCD und DSR können, getrennt für jede Steuerleitung eines jeden geöffneten Ports, Ereignisbehandlungsmaßnahmen installiert werden. Sobald eine Steuerleitung seinen Zustand ändert, wird die entsprechende Signalisierung ausgeführt. Auf diese Weise kann man einer Applikation asynchron zum Programmablauf das Ereignis mitteilen.

Eine Ereignisbehandlungsmaßnahme muß mit der Funktion **FECOM_AddEventHandler** installiert werden. Man kann zwischen drei verschiedenen Signalisierungsmethoden wählen: Nachricht an aufrufenden Prozeß, Nachricht an ein Fenster oder Verwendung einer Callback-Funktion.

Eine installierte Ereignisbehandlungsmaßnahme muß mit der Funktion **FECOM_DeEventHandler** wieder entfernt werden.

Die Struktur **FECOM_EVENT_INIT** enthält die für die Signalisierung notwendigen Parameter:

```
typedef struct _FECOM_EVENT_INIT
{
    UINT uiUse;      // Definiert den Event (z.B. FECOM_CTS_EVENT)
    UINT uiMsg;      // Message-Code für dwThreadID und hwndWnd (z.B. WM_USER_xyz)
    UINT uiFlag;      // Spezifiziert die Verwendung der union (z.B. FECOM_WND_HWND)
    union
    {
        DWORD    dwThreadID;      // für Thread-ID
        HWND     hwndWnd;         // für Window-Handle
        void      (*cbFct)(int, int); // für Callback-Funktion
        HANDLE    hEvent;         // für Event-Handle
    }Method1;
} FECOM_EVENT_INIT;
```

Kernelement der Struktur ist die **union**, die entweder die ID eines Prozesses, das Handle eines Fensters, einen Funktionszeiger oder das Handle eines Windows-API Events enthält. Die Auswahl der Signalisierungsform wird mit dem Parameter *uiFlag* vorgenommen. Im Parameter *uiUse* hinterlegt man eine Kennung der Steuerleitung, der man die Behandlungsmethode zuordnen möchte. Für die Nachrichtenmethoden muß man in *uiMsg* den Messagecode hinterlegen.

Man kann zu einer Steuerleitung mehrere Behandlungsmethoden installieren. Aber jede *dwThreadID*, *hwndWnd*, *cbFct* oder *hEvent* kann nur einmal pro Steuerleitung und Port verwendet werden.

Unabhängig der Ereignissignalisierung kann man den Zustand jeder Steuerleitung mit der Funktion **FECOM_DoPortCmd** abfragen.

¹ Die Benennung der union mit Method ist ausschließlich für C-Programmierer. C++-Programmierer greifen auf die union direkt über die Struktur zu.

5.3.1. FECOM_OpenPort

Funktion	Öffnet eine serielle Schnittstelle zur Kommunikation mit einem OBID-Leser.
Syntax	int FECOM_OpenPort(char* cPortNr)
Beschreibung	<p>Die Funktion öffnet mit Standardparametern eine serielle Schnittstelle und legt intern eine Schnittstellenstruktur zur Verwaltung der Parameter an. Zur nachträglichen Änderung dieser Parameter kann die Funktion FECOM_SetPortPara verwendet werden. Mit FECOM_GetPortPara können diese Parameter ausgelesen werden. Der zurückgelieferte Handle <i>iPortHnd</i> identifiziert die Schnittstelle von außen.</p> <p><i>cPortNr</i> ist eine nullterminierte Zeichenkette mit der Adresse der seriellen Schnittstelle (z. B. "1" für COM:1). Zulässig sind Werte von "1"..."256"¹.</p> <p>Die mit FECOM_OpenPort geöffnete serielle Schnittstelle muß (!) mit der Funktion FECOM_ClosePort wieder geschlossen werden. Andernfalls wird der von der DLL reservierte Speicher nicht wieder freigegeben.</p>
Rückgabewert	Wenn die serielle Schnittstelle fehlerfrei geöffnet werden konnte, wird ein Handle (>0) zurückgeliefert. Im Fehlerfall liefert die Funktion einen Wert kleiner als Null zurück. Die Liste der Fehlercodes findet sich im Anhang.
Standardparameter	Die Standardparameter der seriellen Schnittstelle sind: Baud: 9600; Frame: 8E1; Timeout: 600ms
Beispiel	<pre> ... #include "fecom.h" ... char cPortNr[4]; itoa(1, cPortNr, 10); // Integer in Char wandeln ... int handle = FECOM_OpenPort(cPortNr); // COM:1 soll geöffnet werden if(handle < 0) { // hier Code für den Fehlerfall } else { // Kommunikation über COM:1, die Empfangsdaten liegen im Erfolgsfall in RecBuf // hier Code für Kommunikation oder anderes } </pre>

¹ Anmerkung für Linux: intern wird z.B. aus der Port-Nummer „1“ ein „ttyS0“ generiert.

5.3.2. FECOM_ClosePort

Funktion	Schließt eine serielle Schnittstelle.
Syntax	int FECOM_ClosePort(int iPortHnd)
Beschreibung	Die Funktion schließt die durch den Parameter <i>iPortHnd</i> angegebene serielle Schnittstelle und gibt den reservierten Speicher wieder frei.
Rückgabewert	Der Rückgabewert ist 0, wenn die serielle Schnittstelle geschlossen wurde. Im Fehlerfall liefert die Funktion einen Wert kleiner als Null zurück. Die Liste der Fehlercodes findet sich im Anhang.
Beispiel	<pre>... #include "fecom.h" int Err; char cPortNr[4]; ... itoa(1, cPortNr, 10); // Integer in Char wandeln int handle = FECOM_OpenPort(cPortNr); // COM:1 soll geöffnet werden if(handle < 0) { // hier Code für den Fehlerfall } if(handle > 0) { Err = FECOM_ClosePort(handle); ... }</pre>

5.3.3. FECOM_DetectPort

Funktion	Prüft, ob eine serielle Schnittstelle physisch vorhanden ist.
Syntax	int FECOM_DetectPort(int iPortNr)
Beschreibung	<p>Die Funktion prüft die serielle Schnittstelle mit der Nummer iPortNr, ob sie physisch vorhanden ist. Ist die Schnittstelle gefunden worden, wird eine 0 zurückgegeben, andernfalls FECOM_ERR_PORT_NOT_FOUND.</p> <p>Diese Funktion eignet sich idealerweise für Anwendungsprogramme, die dem Benutzer eine Liste der möglichen seriellen Schnittstellen anbieten will.</p>
Rückgabewert	Ist die Schnittstelle gefunden worden, wird eine 0 zurückgegeben, andernfalls FECOM_ERR_PORT_NOT_FOUND. Im Fehlerfall liefert die Funktion einen Wert kleiner als Null zurück. Die Liste der Fehlercodes findet sich im Anhang.
Beispiel	<pre>... #include "fecom.h" ... for(int iPortNr=1; iPortNr<257, ++iPortNr) { if(0 == FECOM_DetectPort(iPortNr); { // Port ist physisch vorhanden } }</pre>

5.3.4. FECOM_GetPortList

Funktion	Ermittelt in Abhängigkeit vom Parameter <i>iNext</i> den ersten oder den nachfolgenden PortHandle aus der internen Liste der geöffneten seriellen Schnittstellen.
Syntax	int FECOM_GetPortList(int iNext)
Beschreibung	Die Funktion gibt ein PortHandle aus der internen Liste der PortHandles zurück. Übergibt man für <i>iNext</i> eine 0, wird der erste Eintrag aus der Liste zurückgegeben. Übergibt man mit <i>iNext</i> ein in der Liste geführtes PortHandle, wird der dem PortHandle 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 PortHandle geliefert. Ist das Ende der internen Liste erreicht, also der übergebene PortHandle keinen Nachfolger hat, wird eine 0 zurückgegeben. Ist kein Port geöffnet, wird FECOM_ERR_EMPTY_LIST zurückgeliefert. Im Fehlerfall liefert die Funktion einen Wert kleiner als Null zurück. Die Liste der Fehlercodes findet sich im Anhang.
Beispiel	<pre>#include "fecom.h" ... // Funktion ermittelt die Einstellungen aller geöffneten COM-Ports void COMList(void) { int iNextHnd = FECOM_GetPortList(0); // den ersten Handle ermitteln while(iNextHnd > 0) { // hier z. B. Code zum Auslesen der COM-Parameter mit FECOM_GetPortPara(...) ... iNextHnd = FECOM_GetPortList(iNextHnd); // nächsten Handle ermitteln } ... // hier z. B. Code zum Anzeigen der Liste ... }</pre>
Tip	<p>Beim Schließen aller geöffneten COM-Ports bedient man sich gerne einer Schleife, ähnlich der im oberen Beispiel. Nur muß man bedenken, daß man von einem geschlossenen Port keinen Nachfolger mehr ermitteln kann. In dem folgenden Codefragment wird gezeigt, wie man in einer Schleife alle geöffneten Ports schließen kann:</p> <pre>... iNextHnd = FECOM_GetPortList(0); // den ersten Handle ermitteln while(iNextHnd > 0) { iCloseHnd = iNextHnd; iNextHnd = FECOM_GetPortList(iNextHnd); // erst nächsten Handle ermitteln iError = FECOM_ClosePort(iCloseHnd); // jetzt erst Port schließen } ...</pre>

5.3.5. FECOM_GetDLLVersion

Funktion	Ermittelt die Versionsnummer der DLL/SO.
Syntax	void FECOM_GetDLLVersion(char* cVersion)
Beschreibung	<p>Die Funktion gibt die Versionsnummer der DLL/SO zurück.</p> <p><i>cVersion</i> ist eine leere, nullterminierte Zeichenkette zur Rückgabe der Versionsnummer. Die Zeichenkette sollte wenigstens 256 Zeichen aufnehmen können.</p> <p>In der aktuellen Version wird die Zeichenkette mit "02.08.03" gefüllt. Neuere Versionen könnten aber weitere Informationen liefern.</p>
Rückgabewert	ohne
Beispiel	<pre>... #include "fecom.h" char cVersion[256]; FECOM_GetDLLVersion(cVersion) // hier Code zum Anzeigen der Versionsnummer</pre>

5.3.6. FECOM_GetErrorText

Funktion	Ermittelt Fehlertext zum Fehlercode
Syntax	int FECOM_GetErrorText(int iErrorCode, char* cErrorText)
Beschreibung	<p>Die Funktion übergibt in <i>cErrorText</i> den zum <i>iErrorCode</i> zugehörigen englischen Fehlertext.</p> <p>Der Puffer für <i>cErrorText</i> sollte 256 Zeichen aufnehmen können.</p>
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	<pre>... #include "fecom.h" #include "fecomdef.h" char cErrorText[256]; ... int iBack = FECOM_GetErrorText(FECOM_ERR_EMPTY_LIST, cErrorText) // hier Code zum Anzeigen des Textes</pre>

5.3.7. FECOM_GetLastError

Funktion	Ermittelt den letzten Fehlercode und übergibt Fehlertext
Syntax	int FECOM_GetLastError(int iPortHnd , int* iErrorCode, char* cErrorText)
Beschreibung	<p>Die Funktion übergibt in <i>iErrorCode</i> den letzten Fehlercode der mit <i>iPortHnd</i> ausgewählten Schnittstelle zurück und übergibt in <i>cErrorText</i> den zugehörigen englischen Fehlertext.</p> <p>Der Puffer für <i>cErrorText</i> sollte 256 Zeichen aufnehmen können.</p>
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	<pre>... #include "fecom.h" ... char cErrorText[256]; int iErrorCode = 0; ... int iBack = FECOM_GetLastError(iPortHnd, &iErrorCode, cErrorText) // hier Code zum Anzeigen des Textes</pre>

5.3.8. FECOM_GetPortHnd

Funktion	Ermittelt von einer mit FECOM.DLL geöffneten seriellen Schnittstelle den Port-Handle.
Syntax	int FECOM_GetPortHnd(char* cPortNr)
Beschreibung	<p>In der Regel stellt man in einem Programm die Nummer des COM-Ports ein, dagegen wird programmintern mit einem Handle gearbeitet. Mit dieser Funktion kann auf einfache Weise der PortHandle einer zuvor geöffneten seriellen Schnittstelle ermittelt werden.</p> <p>Diese Funktion ist eine "Umkehrfunktion" zu FECOM_GetPortPara(handle, "PortNr", Value), die zum Port-Handle die Nummer des COM-Ports ermittelt.</p> <p><i>cPortNr</i> ist eine nullterminierte Zeichenkette mit der Adresse der seriellen Schnittstelle (z. B. "1" für COM:1). Zulässig sind Werte von "1"..."256".</p>
Rückgabewert	Wenn die serielle Schnittstelle zur übergebenen <i>cPortNr</i> gefunden wurde, wird der PortHandle (>0) zurückgeliefert. Konnte in der Port-Liste die gesuchte PortNummer <i>cPortNr</i> nicht gefunden werden, wird eine 0 zurückgegeben. Im Fehlerfall liefert die Funktion einen Wert kleiner als Null zurück. Die Liste der Fehlercodes findet sich im Anhang.
Beispiel	<pre>... #include "fecom.h" ... char cPortNr[4]; ... itoa(1, cPortNr, 10); // Integer in Char wandeln int handle = FECOM_OpenPort(cPortNr); // COM:1 soll geöffnet werden if(handle < 0) { // hier Code für den Fehlerfall } else { // handle wird mittels PortNr erneut ermittelt handle = FECOM_GetPortHnd(cPortNr); }</pre>

5.3.9. FECOM_GetPortPara

Funktion	Ermittelt von der mit <i>iPortHnd</i> bestimmten seriellen Schnittstelle einen Parameter.
Syntax	Int FECOM_GetPortPara(int iPortHnd, char* cPara, char* cValue)
Beschreibung	<p>Die Funktion ermittelt den aktuellen Wert eines Parameters.</p> <p><i>cPara</i> ist eine nullterminierte Zeichenkette mit der Parameterkennung</p> <p><i>cValue</i> ist eine leere, nullterminierte Zeichenkette zur Rückgabe des Parameterwerts. Die Zeichenkette sollte wenigstens 128 Zeichen aufnehmen können.</p>
Parameterkennungen	<p>Die Parameterkennungen sind: Baud, Frame, Timeout, ErrCode, ErrStr, TxTimeControl, TxDelayTime, CharTimeoutMpy, Performance, Language, PortNr. Letzterer gibt die physische Nummer der seriellen Schnittstelle zurück.</p> <p>Der Parameter Language setzt die Sprache in der DLL und ist ein globaler, also nicht an ein Port-Handle gebundener Wert. Man setzt deshalb iPortHnd in diesem Fall auf 0.</p>
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: 6.2. Liste der Parameterkennungen .
Beispiel	<pre>... #include "fecom.h" ... char cValue[128]; ... if(!FECOM_GetPortPara(handle, "Baud", cValue)) { // hier Code zum Anzeigen des COM-Parameters ... } }</pre>

5.3.10. FECOM_SetPortPara

Funktion	Setzt einen Parameter einer seriellen Schnittstelle auf einen neuen Wert.			
Syntax	int FECOM_SetPortPara(int iPortHnd, char* cPara, char* cValue)			
Beschreibung	Die Funktion übergibt an die mit <i>iPortHnd</i> benannte serielle Schnittstelle einen neuen Parameter. Dadurch wird die betroffene serielle Schnittstelle neu initialisiert und Send- und Empfangspuffer werden gelöscht.			
	<i>cPara</i> ist eine nullterminierte Zeichenkette mit der Parameterkennung.			
	<i>cValue</i> ist eine nullterminierte Zeichenkette mit dem neuen Parameterwert.			
	Parameterkennung	Wertebereich	Defaultwert	Einheit
	Baud	300...115200	9600	bit/s
	Frame	7N1, 7E1, 7O1, 7N2, 7E2, 7O2, 8N1, 8E1, 8O1	8E1	
	Timeout	0...99999	600	ms
	TxTimeControl	0, 1	1	
	TxDelayTime	0...999	5	ms
	CharTimeoutMpy	1...99	1	-
	Performance	0, 1	1	-
	Language	7, 9	9	-
	UseOBID (nur Linux)	0, 1	0	-
Rückgabewert	Wenn die serielle Schnittstelle mit dem neuen Parameterwert fehlerfrei initialisiert werden konnte, wird eine 0 zurückgeliefert. Im Fehlerfall liefert die Funktion einen Wert kleiner als Null zurück. Die Liste der Fehlercodes findet sich im Anhang.			
Querverweis	Weitere Informationen in: 6.2. Liste der Parameterkennungen .			
Beispiel	<pre> ... #include "fecom.h" int Err; char cPortNr[4]; ... itoa(1, cPortNr, 10); // Integer in Char wandeln int handle = FECOM_OpenPort(cPortNr); // COM:1 soll geöffnet werden if(handle > 0) { Err = FECOM_SetPortPara(handle, "Baud", "4800"); ... } </pre>			

5.3.11. FECOM_DoPortCmd

Funktion	Führt ein Kommando an einer seriellen Schnittstelle aus.	
Syntax	int FECOM_DoPortCmd(int iPortHnd, char* cCmd, char* cValue)	
Beschreibung	Die Funktion führt an der mit <i>iPortHnd</i> benannten seriellen Schnittstelle ein Kommando aus.	
	<i>cCmd</i> ist eine nullterminierte Zeichenkette mit der Kommandokennung.	
	<i>cValue</i> ist eine nullterminierte Zeichenkette mit dem neuen Parameterwert bzw. für den Rückgabewert (z.B. Status einer Steuerleitung).	
	Wenn in <i>cValue</i> ein Rückgabewert erwartet wird, sollte der Puffer wenigstens 16 Zeichen aufnehmen können.	
	Kommando	Funktion
	FlushInQ	Eingangspuffer wird gelöscht
	FlushOutQ	Ausgangspuffer wird gelöscht
	SetDTR	setzt DTR-Leitung „ON“ oder „OFF“
	SetRTS	setzt RTS-Leitung „ON“ oder „OFF“
	GetDTR	ermittelt DTR-Status
	GetRTS	ermittelt RTS-Status
	GetCTS	ermittelt CTS-Status
	GetDCD	ermittelt DCD-Status
	GetDSR	ermittelt DSR-Status
Rückgabewert	Wenn das Kommando fehlerfrei ausgeführt wurde, wird eine 0 zurückgeliefert. Im Fehlerfall liefert die Funktion einen Wert kleiner als Null zurück.	
1. Beispiel	<pre>#include "fecom.h" ... int Err; char cPortNr[4]; ... itoa(1, cPortNr, 10); // Integer in Char wandeln int handle = FECOM_OpenPort(cPortNr); // COM:1 soll geöffnet werden if(handle > 0) { Err = FECOM_DoPortCmd(handle, "FlushInQ", ""); ... } ...</pre>	
2. Beispiel	<pre>#include "fecom.h" ... int Err; char cValue[16]; ... Err = FECOM_DoPortCmd(handle, "GetCTS", cValue); if(strcmp(cValue, „ON“)==0) // Vergleich von Zeichenketten { // CTS ist gesetzt } ...</pre>	

5.3.12. FECOM_AddEventHandler

Funktion	Eine Ereignisbehandlungsmaßnahme wird installiert
Syntax	int FECOM_AddEventHandler(int iPortHnd, FECOM_EVENT_INIT* pInit)
Beschreibung	<p>Die Funktion installiert eine von vier möglichen Ereignisbehandlungsmethode. Diese Methode kommt dann zur Anwendung, wenn sich der Zustand der Steuerleitung ändert, für die die Methode installiert wurde. Auf diese Weise ist eine asynchrone Reaktion auf Ereignisse in einem Applikationsprogramm möglich.</p> <p>Die Ereignisbehandlungsmethode wird nur für die mit <i>iPortHnd</i> identifizierten Schnittstelle eingerichtet. Das bedeutet, daß man bei Bedarf für jede geöffnete Schnittstelle diese Installation durchführen muß.</p> <p><u>1. Methode: Nachricht an Thread (nicht für Linux)</u></p> <p>Diese Methode verwendet man für den Nachrichtenaustausch zwischen Threads¹. Der Thread ermittelt mit der API-Funktion GetCurrentThreadID() den Thread-Identifizier und übergibt diesen als Parameter dwThreadID in der FECOM_EVENT_INIT-Struktur. Der Thread muß für den Empfang der Nachricht, die von FECOM mit der API-Funktion PostThreadMessage(..) verschickt wurde, eine Nachrichtenbehandlungsfunktion bereitstellen. Der Nachrichtencode ist frei wählbar.</p> <p>Die FECOM_EVENT_INIT-Struktur wird wie folgt ausgefüllt:</p> <pre> uiUse = FECOM_xyz_EVENT // siehe Defines FECOM.H uiMsg = WM_USER + ... // frei wählbar, aber oberhalb von WM_USER² uiFlag = FECOM_THREAD_ID dwThreadID = GetCurrentThreadID() </pre> <p>Die MessageMap-Funktion in der Applikation bekommt im 1. Parameter (WPARAM) die Portnummer und im 2. Parameter (LPARAM) den Zustand der Steuerleitung (0 = nicht gesetzt; 1 = gesetzt) übergeben.</p> <p><u>2. Methode: Nachricht an Fenster (nicht für Linux)</u></p> <p>Diese Methode verwendet man, wenn die Nachricht direkt an ein Fenster geschickt werden soll. Von dem betreffenden Fenster wird mit der API-Funktion GetWindow(..)³ der Handle ermittelt und als Parameter hwndWnd in der FECOM_EVENT_INIT-Struktur übergeben. Das Fenster muß für den Empfang der Nachricht, die von FECOM mit der API-Funktion PostMessage(..) verschickt wurde, eine Nachrichtenbehandlungsfunktion bereitstellen. Der Nachrichtencode ist frei wählbar.</p> <p>Die FECOM_EVENT_INIT-Struktur wird wie folgt ausgefüllt:</p> <pre> uiUse = FECOM_xyz_EVENT // siehe Defines FECOM.H uiMsg = WM_USER + ... // frei wählbar, aber oberhalb von WM_USER² uiFlag = FECOM_WND_HWND hwndWnd = GetWindow(...) </pre> <p>Die MessageMap-Funktion in der Applikation bekommt im 1. Parameter (WPARAM) die Portnummer und im 2. Parameter (LPARAM) den Zustand der Steuerleitung (0 = nicht</p>

¹ Paralleler, vom Applikationsprogramm unabhängiger Ausführungspfad. Auch das Applikationsprogramm ist ein Thread.

² Siehe Windows-Dokumentation zur Platform-SDK

³ Bei Verwendung der MFC-Klasse CWnd kann auch die Methode GetSafeHwnd() benutzt werden

	<p>gesetzt; 1 = gesetzt) übergeben.</p> <p><u>3. Methode: Aufruf einer Callback-Funktion</u></p> <p>Mit der Callback-Methode wird ein Funktionszeiger für ein Ereignis installiert. Diese Funktion muß wie folgt deklariert sein: void IhrFunktionsName(int, int); Tritt eine Zustandsänderung an der betreffenden Steuerleitung ein, wird die Funktion von FECOM aufgerufen. Der Inhalt der Funktion kann frei bestimmt werden. Die Übergabeparameter sind allerdings festgelegt: im 1. Parameter (int) wird die Portnummer und im 2. Parameter (int) der Zustand der Steuerleitung (0 = nicht gesetzt; 1 = gesetzt) übergeben.</p> <p>Die FECOM_EVENT_INIT-Struktur wird wie folgt ausgefüllt:</p> <pre> uiUse = FECOM_xyz_EVENT // siehe Defines FECOM.H uiMsg wird nicht benötigt uiFlag = FECOM_CALLBACK cbFct = (void*)&IhrFunktionsName¹ </pre> <p><u>4. Methode: Setzen eines Events (nicht für Linux)</u></p> <p>Mit der Event-Methode wird ein Event-Handle für ein Ereignis installiert. Tritt eine Zustandsänderung an der betreffenden Steuerleitung ein, wird der Event von FECOM mit der API-Funktion SetEvent(...) gesetzt. Auf Seiten der Anwendung wartet man mit der API-Funktion WaitForSingleObject(...) auf das Ereignis. Da man nicht unterscheiden kann, wie sich der Zustand der betreffenden Steuerleitung geändert hat, muß man mit der Funktion FECOM_DoPortCmd den Zustand abfragen. Der gesetzte Event muß vom Anwendungsprogramm mit der API-Funktion ResetEvent(...) wieder zurückgesetzt werden.</p> <p>Die FECOM_EVENT_INIT-Struktur wird wie folgt ausgefüllt:</p> <pre> uiUse = FECOM_xyz_EVENT // siehe Defines FECOM.H uiMsg wird nicht benötigt uiFlag = FECOM_EVENT hEvent = CreateEvent(...); </pre> <p>Eine installierte Ereignisbehandlungsmethode muß wieder mit der Funktion FECOM_DeEventHandler entfernt werden.</p> <p>Beim Schließen einer Schnittstelle gehen alle für diese Schnittstelle installierten Ereignisbehandlungsmethoden verloren.</p>
Querverweis	<p>Weitere Informationen in: 5.3. Ereignissignalisierung für Steuerleitungen, 5.3.13. FECOM DeEventHandler, 6.3. Liste der Konstanten für die FECOM EVENT INIT-Struktur.</p>
Rückgabewert	<p>Im fehlerfreien Fall liefert die Funktion Null und im Fehlerfall einen Wert kleiner als Null zurück. Die Liste der Fehlercodes findet sich im Anhang.</p>

¹ Die Funktion hat den Prototyp: void IhrFunktionsName(int, int)

5.3.13. FECOM_DelEventHandler

Funktion	Eine Ereignisbehandlungsmaßnahme wird entfernt
Syntax	int FECOM_DelEventHandler(int iPortHnd, FECOM_EVENT_INIT* plnit)
Beschreibung	<p>Die Funktion entfernt eine zuvor mit FECOM_AddEventHandler installierte Ereignisbehandlungsmaßnahme. In der FECOM_EVENT_INIT-Struktur spezifiziert man die zu entfernende Ereignisbehandlungsmaßnahme im Detail.</p> <p><u>Entfernung der 1. Methode: Nachricht an Thread (nicht für Linux)</u> Die FECOM_EVENT_INIT-Struktur wird wie folgt ausgefüllt: uiUse = FECOM_xyz_EVENT // siehe Defines in FECOM.H uiMsg wird nicht benötigt uiFlag = FECOM_THREAD_ID dwThreadID = GetCurrentThreadID()</p> <p><u>Entfernung der 2. Methode: Nachricht an Fenster (nicht für Linux)</u> Die FECOM_EVENT_INIT-Struktur wird wie folgt ausgefüllt: uiUse = FECOM_xyz_EVENT // siehe Defines in FECOM.H uiMsg wird nicht benötigt uiFlag = FECOM_WND_HWND hwndWnd = GetWindow(...)</p> <p><u>Entfernung der 3. Methode: Aufruf einer Callback-Funktion</u> Die FECOM_EVENT_INIT-Struktur wird wie folgt ausgefüllt: uiUse = FECOM_xyz_EVENT // siehe Defines FECOM.H uiMsg wird nicht benötigt uiFlag = FECOM_CALLBACK cbFct = (void*)&IhrFunktionsName</p> <p><u>Entfernung der 4. Methode: Setzen eines Events (nicht für Linux)</u> Die FECOM_EVENT_INIT-Struktur wird wie folgt ausgefüllt: uiUse = FECOM_xyz_EVENT // siehe Defines FECOM.H uiMsg wird nicht benötigt uiFlag = FECOM_EVENT hEvent = IhrEventHandle;</p>
Querverweis	Weitere Informationen in: 5.3. Ereignissignalisierung für Steuerleitungen , 5.3.12. FECOM_AddEventHandler , 6.3. Liste der Konstanten für die FECOM_EVENT_INIT-Struktur .
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.

5.3.14. FECOM_Transceive

Funktion	Funktion zur Kommunikation (Transmit and Receive) über den Port.
Syntax	int FECOM_Transceive(int iPortHnd, UCHAR* cSendProt, int iSendLen, UCHAR* cRecProt, int iRecLen)
Beschreibung	<p>Die Funktion schickt die in <i>cSendProt</i> enthaltenen Daten über die serielle Schnittstelle an ein angeschlossenes Gerät und hinterlegt in <i>cRecProt</i> die empfangenen Daten.</p> <p>Im Parameter <i>iSendLen</i> muß die Anzahl der Zeichen in <i>cSendProt</i> übergeben werden.</p> <p>Mit dem Parameter <i>iRecLen</i> muß die maximale Länge des Puffers <i>cRecProt</i> angegeben werden. Übersteigt die Anzahl der empfangenen Zeichen den in <i>iRecLen</i> übergebenen Wert, wird die Funktion mit einem Fehler beendet. Die bis zum Abbruch empfangenen Zeichen werden in <i>cRecProt</i> hinterlegt.</p> <p>Vor der Kommunikation werden Sende- und Empfangspuffer gelöscht.</p> <p>Mit dem Parameter <i>TxDelayTime</i>¹ kann man das Sendeprotokoll solange verzögern, bis die Zeit <i>TxDelayTime</i> seit dem letzten Empfangsprotokoll vergangen ist.</p>
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	<pre>#include "fecom.h" ... int iSendLen; int iRecProtLen; char cPortNr[4]; ... itoa(1, cPortNr, 10); // Integer in Char wandeln UCHAR cSendBuf[256]; // Puffergröße ist eventuell an die Sendedaten anzupassen UCHAR cRecBuf[256]; // Puffergröße ist eventuell an die Empfangsdaten anzupassen ... int handle = FECOM_OpenPort(cPortNr); // COM:1 soll geöffnet werden if(handle < 0) { // hier Code für den Fehlerfall } else { // das Sendprotokoll wird z.B. mit einer Funktion ermittelt und in SendBuf abgelegt iSendLen = GetSendProtocol(cSendBuf); // Kommunikation über COM:1, die Empfangsdaten liegen im Erfolgsfall in RecBuf iRecProtLen = FECOM_Transceive(handle, cSendBuf, iSendLen, cRecBuf, 256); if(cRecProtLen < 0) { // Kommunikationsfehler } }</pre>

¹ s. Kapitel [5.3.10. FECOM_SetPortPara](#)

5.3.15. FECOM_Transmit

Funktion	Funktion zur Ausgabe eines Protokolls über den Port.
Syntax	int FECOM_Transmit(int iPortHnd, UCHAR* cSendProt, int iSendLen)
Beschreibung	<p>Die Funktion schickt die in <i>cSendProt</i> enthaltenen Daten über die serielle Schnittstelle an ein angeschlossenes Gerät und wartet <u>nicht</u> auf ein Antwortprotokoll.</p> <p>Im Parameter <i>iSendLen</i> muß die Anzahl der Zeichen in <i>cSendProt</i> übergeben werden.</p> <p>Vor dem Senden des Protokolls wird der Sendepuffer gelöscht. Zeichen, die noch auf die Ausgabe warten, gehen dadurch verloren.</p> <p>Die Funktion kehrt erst zurück, wenn alle Zeichen über die Schnittstelle ausgegeben wurden.</p>
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	<pre> ... #include "fecom.h" int iErr; int iSendLen; char cPortNr[4]; ... itoa(1, cPortNr, 10); // Integer in Char wandeln UCHAR cSendBuf[256]; // Puffergröße ist eventuell an die Senddaten anzupassen ... int handle = FECOM_OpenPort(cPortNr); // COM:1 soll geöffnet werden if(handle < 0) { // hier Code für den Fehlerfall } else { // das Sendprotokoll wird z.B. mit einer Funktion ermittelt und in SendBuf abgelegt iSendLen = GetSendProtocol(cSendBuf); // Kommunikation über COM:1 iErr = FECOM_Transmit(handle, cSendBuf, iSendLen); if(iErr < 0) { // Kommunikationsfehler } } </pre>

5.3.16. FECOM_Receive

Funktion	Funktion zum Empfang eines Protokolls über den Port.
Syntax	int FECOM_Receive(int iPortHnd, UCHAR* cRecProt, int iRecLen)
Beschreibung	<p>Die Funktion erwartet innerhalb der Zeit Timeout (s. 6.2. Liste der Parameterkennungen) über die serielle Schnittstelle empfangene Daten, liest sie aus und hinterlegt sie im Empfangspuffer <i>cRecProt</i>.</p> <p>Mit dem Parameter <i>iRecLen</i> muß die maximale Länge des Puffers <i>cRecProt</i> angegeben werden. Übersteigt die Anzahl der empfangenen Zeichen den in <i>iRecLen</i> übergebenen Wert, wird die Funktion mit einem Fehler beendet. Die bis zum Abbruch empfangenen Zeichen werden in <i>cRecProt</i> hinterlegt.</p> <p>Die Funktion löscht <u>nicht</u> den Empfangspuffer. Dadurch ist gewährleistet, daß zuvor eingetroffene Zeichen nicht verloren gehen.</p>
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	<pre> ... #include "fecom.h" #include "fecomdef.h" int iRecProtLen; char cPortNr[4]; ... itoa(1, cPortNr, 10); // Integer in Char wandeln UCHAR cRecBuf[256]; // Puffergröße ist eventuell an die Empfangsdaten anzupassen ... int handle = FECOM_OpenPort(cPortNr); // COM:1 soll geöffnet werden if(handle < 0) { // hier Code für den Fehlerfall } else { // Kommunikation über COM:1, die Empfangsdaten liegen im Erfolgsfall in RecBuf iRecProtLen = FECOM_Receive(handle, cRecBuf, 256); if(iRecProtLen < 0) { // Kommunikationsfehler oder Pufferüberlauf if(iRecProtLen == FECOM_ERR_OVL_RECBUF) { // Pufferüberlauf: Daten in RecBuf sind gültige Empfangsdaten ... } } } } </pre>

6. Anhang

6.1. Fehlercodes

Fehler-Konstante	Wert	Beschreibung
FECOM_ERR_NEWPORT_FAILURE	-1000	Fehler beim Erzeugen eines neuen Port-Objekts. Eventuell ist Speichermangel eine Ursache.
FECOM_ERR_EMPTY_LIST	-1001	Port-Handleliste ist leer (keine Port-Objekte angelegt)
FECOM_ERR_POINTER_IS_NULL	-1002	ein Zeiger ist Null, also ungültig
FECOM_ERR_NO_MEMORY	-1003	Speichermangel
FECOM_ERR_UNSUPPORTED_HARDWARE	-1004	nicht unterstützte Hardware. Der Fehler wird gemeldet, wenn die verwendete Hardware keinen Zähler mit hoher Auflösung unterstützt
FECOM_ERR_PORT_NOT_FOUND	-1005	Rückgabe von FECOM_DetectPort, wenn der angegebene Port nicht vorhanden ist.
FECOM_ERR_UNSUPPORTED	-1006	nicht unterstützte Funktion oder Option
FECOM_ERR_NO_PORT	-1010	Port konnte nicht geöffnet werden
FECOM_ERR_NO_CONNECT	-1011	Timeout beim Öffnen des Ports. Port ist nicht geöffnet worden.
FECOM_ERR_LINK_ID	-1012	der Parameter cPortNr in der Funktion FECOM_OpenPort ist fehlerhaft
FECOM_ERR_PORT_IS_OPEN	-1013	der Port ist bereits geöffnet
FECOM_ERR_UNKNOWN_HND	-1020	der übergebene Port-Handle ist unbekannt
FECOM_ERR_HND_IS_NULL	-1021	der übergebene Port-Handle ist 0
FECOM_ERR_HND_IS_NEGATIVE	-1022	der übergebene Port-Handle ist negativ
FECOM_ERR_NO_HND_FOUND	-1023	kein Port-Handle in Port-Handleliste gefunden
FECOM_ERR_TIMEOUT	-1030	Timeout beim Lesen vom Port
FECOM_ERR_NO_SENDPROTOCOL	-1031	kein Sendeprotokoll übergeben
FECOM_ERR_RECEIVE_PROCESS	-1032	Fehler im Empfangsprozess
FECOM_ERR_INIT_COMM_PROCESS	-1033	Fehler beim Initialisieren des Ports
FECOM_ERR_FLUSH_INPUT_BUFFER	-1034	Fehler beim Löschen des Empfangspuffers
FECOM_ERR_FLUSH_OUTPUT_BUFFER	-1035	Fehler beim Löschen des Sendepuffers
FECOM_ERR_CHANGE_PORT_PARA	-1036	Fehler beim Ändern eines Port-Parameters
FECOM_ERR_TRANSMIT_PROCESS	-1037	Fehler im Sendeprozess
FECOM_ERR_RECEIVE_NOISE_DATA	-1038	Checksummen- und/oder Paritätsfehler

Fehler-Konstante	Wert	Beschreibung
		oder nicht identifizierbare Daten
FECOM_ERR_UNKNOWN_PARAMETER	-1050	Übergabeparameter ist nicht bekannt
FECOM_ERR_PARAMETER_OUT_OF_RANGE	-1051	Übergabeparameter zu groß oder zu klein
FECOM_ERR_ODD_PARAMETERSTRING	-1052	eine nicht unterstützte Option wurde per Übergabeparameter aufgerufen
FECOM_ERR_PORTNR_OUT_OF_RANGE	-1053	die übergebene Portnummer liegt nicht im gültigen Wertebereich von 1..256
FECOM_ERR_UNKNOWN_ERRORCODE	-1054	unbekannter Fehlercode
FECOM_ERR_OVL_RECBUF	-1070	Überlauf des Empfangspuffers

6.2. Liste der Parameterkennungen

Parameterkennung	Wertebereich	Default	Einheit	Beschreibung
Baud	300...115200	9600	bit/s	Baudrate der Schnittstelle
Frame	7N1, 7E1, 7O1, 7N2, 7E2, 7O2, 8N1, 8E1, 8O1	8E1	-	Zeichenrahmen (Datenbits, Parität, Stoppbits)
Timeout	0...99999	600	ms	Maximale Wartezeit auf Empfangsprotokoll
PortNr	1...256	0	-	Nummer des COM-Ports
TxTimeControl	0, 1	1	-	Wenn gesetzt (1), dann wird intern die Ausgabe des nächsten Sendeprotokolls solange verzögert, bis mindestens TxDelayTime (ms) nach dem letzten Empfangsprotokoll vergangen sind. Wenn ungesetzt (0), dann wird das Sendeprotokoll immer schnellstmöglich ausgegeben.
TxDelayTime	0...999	5	ms	Minimale Zeitspanne zwischen letztem Empfangs- und nächstem Sendeprotokoll. Wird nur berücksichtigt, wenn TxTimeControl=1
CharTimeoutMpy	1...99	1	-	Seit der V2.00.00 wird die Zeichen-Timeout intern berechnet. Die Zeichen-Timeout bestimmt, nach welcher Zeit nach dem Empfang des letzten Zeichens der Empfangsprozess beendet wird. Mit einigen PCs kann es vermehrt zu Protokollängenfehlern kommen, weil die Wartezeit zu gering ist. In diesem Fall kann mit diesem Parameter die Wartezeit multipliziert werden.
Performance	0, 1	1	-	Bestimmt die CPU-Rechenzeit für die interne Kommunikationsroutine. Standardmäßig wird mit geringer CPU-Beanspruchung gearbeitet. Setzt man den Wert auf 0, dann arbeitet die interne Kommunikation zwar mit maximaler Performance, andere Programme und Threads werden allerdings deutlich gebremst.
Language	7 - Deutsch 9 - Englisch	9	-	Auswahl der Sprache für interne Textressourcen.
UseOBID	0, 1	0	-	nur für Linux: Aktivierung der auf OBID-Protokolle angepasste Empfangsroutine

anstelle der nachfolgenden Parameterkennungen sollte die Funktion **FECOM_GetLastError** benutzt werden

ErrCode	-	-	-	liefert letzten Fehlercode
ErrStr	-	-	-	liefert zum letzten Fehler einen Text

Obsolete Parameter				
CharTimeout	0...99999	20	ms	Maximale Wartezeit auf nächstes Zeichen im Empfangsprotokoll
SleepTime	0...999	0	ms	Wartezeit nach dem Sendeprotokoll und vor dem Einlesen des Empfangsprotokolls ¹
PortOpenTimeout	0...99999	5000	ms	Maximale Wartezeit für das Öffnen eines COM-Ports

¹ Siehe [5.3.14. FECOM Transceiver](#)

6.3. Liste der Konstanten für die FECOM_EVENT_INIT-Struktur

Die Konstantendefinitionen sind in der Datei FECOM.H bzw. FECOM.BAS bzw. FECOM.PAS enthalten.

Konstante	Wert	Verwendung	Beschreibung
FECOM_THREAD_ID	1	uiFlag	Ereignissignalisierung mit Thread-Nachricht
FECOM_WND_HWND	2	uiFlag	Ereignissignalisierung mit Window-Nachricht
FECOM_CALLBACK	3	uiFlag	Ereignissignalisierung mit Callback-Funktion
FECOM_EVENT	4	uiFlag	Ereignissignalisierung mit Windows-API-Event
FECOM_CTS_EVENT	1	uiUse	Signalisierung bei CTS-Änderung
FECOM_DCD_EVENT	2	uiUse	Signalisierung bei DCD-Änderung
FECOM_DSR_EVENT	3	uiUse	Signalisierung bei DSR-Änderung
FECOM_RTS_EVENT	4	uiUse	Signalisierung bei RTS-Änderung
FECOM_DTR_EVENT	5	uiUse	Signalisierung bei DTR-Änderung

6.4. Änderungshistorie

V2.08.02

- Windows: Verbesserte Thread-Sicherheit beim Öffnen und Schliessen von Schnittstellen
- Linux: Verbesserte Performanz in der Kommunikation durch spezielle, auf OBID-Protokolle abgestimmte Empfangsroutine bei Verwendung des neuen Parameters UseOBID.
- Obere Grenze des Parameters CharTimeoutMpy auf 99 erhöht.

V2.08.00

- Die Linux-Version wurde mit dem Compiler GCC 3.3.3 unter SuSE Linux 9.1 erstellt.

V2.07.00

- Neue Baudraten 230400 und 460800

V2.06.08

- Kleine Fehlerkorrekturen

V2.06.06

- Erstes Linux Release (SuSE Linux 8.2, GNU Compiler Collection V3.3-23, glibc V2.3.2-6)

V2.06.00

- Integration einer Java-Schnittstelle für die OBID® Java-Bibliothek
- neuer Fehlercode: -1006

V2.05.00

- Die neue Version ist zu 100% rückwärtskompatibel zur Vorgängerversion.
- Erste Version für Windows CE.

V2.04.04

- Die neue Version ist zu 100% rückwärtskompatibel zur Vorgängerversion.
- Alle Konstanten der Headerdatei FEComDef.h sind in die Datei FECOM.h verschoben worden. FEComDef.h ist damit überflüssig.
- Die Funktion **FECOM_GetPortHnd** gibt bei Portnummer 256 keinen Fehler mehr zurück.

V2.04.00

- Die neue Version ist zu 100% rückwärtskompatibel zur Vorgängerversion.
- Neue Funktion: **FECOM_DetectPort**.

V2.03.00

- Die neue Version ist zu 100% rückwärtskompatibel zur Vorgängerversion.
- interne Version.

V2.02.00

- Die neue Version ist zu 100% rückwärtskompatibel zur Vorgängerversion.
- Neue Parameter: Performance und Language.
- Die Portnummer 256 führt nicht mehr zum Fehler.

V2.00.01

- Die neue Version ist zu 100% rückwärtskompatibel zur Vorgängerversion.
- Neuer Parameter CharTimeoutMpy.

V2.00.00

- Die neue Version ist zu 100% rückwärtskompatibel zur Vorgängerversion.
- FECOM ist nicht mehr von anderen Dateien abhängig.
- Die Performance der Empfangsroutine in der DLL wurde wesentlich gesteigert: Die Funktionen FECOM_Receive und FECOM_Transceive kehren nun mit minimaler Verzögerung nach dem Empfang des Protokolls zurück.
- Neue Steuerparameter: TxTimeControl und TxDelayTime zur gezielten Verzögerung von Sendeprotokollen.
- Die Parameter CharTimeout, PortOpenTimeout und SleepTime sind obsolet.
- Das Öffnen von Schnittstellen mit einer Nummer größer als 9 ist jetzt möglich.
- Die Ereignissignalisierung ist für Events erweitert worden.
- FECOM kann unter C/C++ dynamisch gelinkt werden. In der Header-Datei FECOM.H finden sich die Funktionsdeklarationen.
- Umbenennung des Fehlercodes:
FECOM_ERR_READ_PROTOCOL in FECOM_ERR_RECEIVE_PROCESS

V1.01.00

- Die Funktion **FECOM_DoPortCmd** erlaubt das Setzen bzw. Abfragen folgender Steuerleitungen:

Steuerleitung	setzen	Status abfragen
DTR	X	X
RTS	X	X
CTS	-	X
DCD	-	X
DSR	-	X

- Zusätzlich kann man die Zustandsänderung einer Steuerleitung mit einer Signalisierung verknüpfen. Näheres dazu findet sich im Kapitel [5.3. Ereignissignalisierung für Steuerleitungen](#).
- Die Funktion **FECOM_GetLastError** gibt den letzten Fehlercode und einen Fehlertext zurück.
- Die Funktion **FECOM_GetErrorText** gibt zu einem beliebigen Fehlercode den Fehlertext zurück.

V1.00.09

- neue Parameter für **FECOM_GetPortPara**: ERRCODE, ERRSTR

die Versionen 1.00.07 und 1.00.08 waren interne Zwischenversionen

V1.00.06

- Die Version 1.00.06 wurde nur intern hinsichtlich verbesserter Performance und Stabilität modifiziert.
- Die Version 1.00.06 ist in allen Funktionen Aufruf-kompatibel zu den Vorversionen 1.00.02 – 1.00.05
- Die Version 1.00.06 ist in folgenden Funktionen nicht mehr Aufruf-kompatibel zur Version 1.00.01:

1. FECOM_OpenPort
2. FECOM_GetPortHnd

Beide Funktionen erwarten jetzt einen Zeiger auf eine Zeichenkette gegenüber einem Bytewert in der Vorversion!