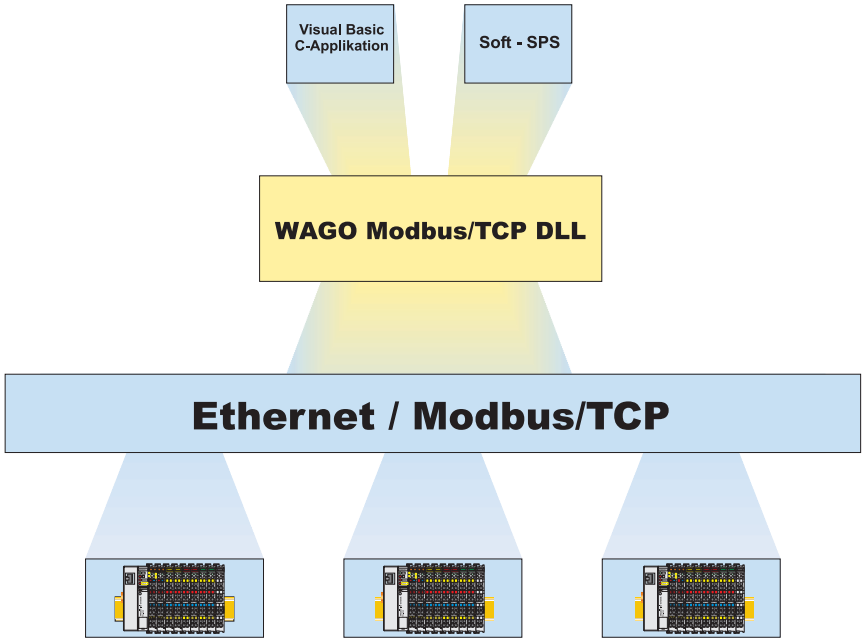
WAGO I/O SYSTEM 759

API Modbus/TCP DLL



Handbuch

Technische Beschreibung

Version 1.2.0

WAGO®
INNOVATIVE CONNECTIONS

Copyright © 2003 by WAGO Kontakttechnik GmbH
Alle Rechte vorbehalten.

WAGO Kontakttechnik GmbH
Hansastraße 27
D-32423 Minden

Tel.: +49 (0) 571/8 87 – 0
Fax: +49 (0) 571/8 87 – 1 69

E-Mail: info@wago.com

Web: http://www.wago.com

Technischer Support
Tel.: +49 (0) 571/8 87 – 5 55
Fax: +49 (0) 571/8 87 – 85 55

E-Mail: support@wago.com

Es wurden alle erdenklichen Maßnahmen getroffen, um die Richtigkeit und Vollständigkeit der vorliegenden Dokumentation zu gewährleisten. Da sich Fehler, trotz aller Sorgfalt, nie vollständig vermeiden lassen, sind wir für Hinweise und Anregungen jederzeit dankbar.

E-Mail: documentation@wago.com

Wir weisen darauf hin, dass die im Handbuch verwendeten Soft- und Hardwarebezeichnungen und Markennamen der jeweiligen Firmen im Allgemeinen einem Warenzeichenschutz, Markenschutz oder patentrechtlichem Schutz unterliegen.

INHALTSVERZEICHNIS

1 Wichtige Erläuterungen 1
1.1 Rechtliche Grundlagen 1
1.1.1 Urheberschutz 1
1.1.2 Personalqualifikation 1
1.1.3 Bestimmungsgemäßer Gebrauch 1
1.2 Symbole 2
1.3 Schriftkonventionen 3
1.4 Darstellungen der Zahlensysteme 3
1.5 Gültigkeitsbereich 4
1.6 Abkürzungen 4

2 Modbus/TCP DLL 5
2.1 Überblick 5
2.2 Installation 6
2.3 Funktionen 6
2.3.1 MBTInit 7
2.3.2 MBTExit 8
2.3.3 MBTConnect 9
2.3.4 MBTDisconnect 11
2.3.5 MBTReadRegisters 12
2.3.6 MBTReadCoils 14
2.3.7 MBTReadExceptionStatus 16
2.3.8 MBTReadCompleted 18
2.3.9 MBTWriteRegisters 19
2.3.10 MBTWriteCoils 21
2.3.11 MBTWriteCompleted 23
2.3.12 MBTSwapWord 24
2.3.13 MBTSwapDWord 25
2.3.14 MODBUSTCP_TABLE_xxx 26
2.4 Beispiele 27
2.4.1 VBA 27
2.4.1.1 Kurzbeschreibung 27
2.4.1.2 Deklaration der Konstanten und Funktionen der DLL 27
2.4.1.3 Deklaration der Variablen und Ereignisprozeduren des Formulars 28
2.4.2 C 30
2.4.2.1 Kurzbeschreibung 30
2.4.2.2 Interface 30
2.4.2.3 Programm 32

3 Index 40

1 Wichtige Erläuterungen

Um dem Anwender eine schnelle Installation und Inbetriebnahme der in diesem Handbuch beschriebenen Geräte zu gewährleisten, ist es notwendig, die nachfolgenden Hinweise und Erläuterungen sorgfältig zu lesen und zu beachten.

1.1 Rechtliche Grundlagen

1.1.1 Urheberschutz

Dieses Handbuch, einschließlich aller darin befindlichen Abbildungen, ist urheberrechtlich geschützt. Jede Weiterverwendung dieses Handbuches, die von den urheberrechtlichen Bestimmungen abweicht, ist nicht gestattet. Die Reproduktion, Übersetzung in andere Sprachen, sowie die elektronische und fototechnische Archivierung und Veränderung bedarf der schriftlichen Genehmigung der WAGO Kontakttechnik GmbH, Minden. Zuwiderhandlungen ziehen einen Schadenersatzanspruch nach sich.

Die WAGO Kontakttechnik GmbH behält sich Änderungen, die dem technischen Fortschritt dienen, vor. Alle Rechte für den Fall der Patenterteilung oder des Gebrauchmusterschutzes sind der WAGO Kontakttechnik GmbH vorbehalten. Fremdprodukte werden stets ohne Vermerk auf Patentrechte genannt. Die Existenz solcher Rechte ist daher nicht auszuschließen.

1.1.2 Personalqualifikation

Der in diesem Handbuch beschriebene Produktgebrauch richtet sich ausschließlich an Fachkräfte mit einer Ausbildung in der SPS-Programmierung, Elektrofachkräfte oder von Elektrofachkräften unterwiesene Personen, die außerdem mit den geltenden Normen vertraut sind. Für Fehlhandlungen und Schäden, die an WAGO-Produkten und Fremdprodukten durch Missachtung der Informationen dieses Handbuches entstehen, übernimmt die WAGO Kontakttechnik GmbH keine Haftung.

1.1.3 Bestimmungsgemäßer Gebrauch

Die Komponenten werden ab Werk für den jeweiligen Anwendungsfall mit einer festen Hard- und Softwarekonfiguration ausgeliefert. Änderungen sind nur im Rahmen der in den Handbüchern dokumentierten Möglichkeiten zulässig. Alle anderen Veränderungen an der Hard- oder Software sowie der nicht bestimmungsgemäße Gebrauch der Komponenten bewirken den Haftungsausschluss der WAGO Kontakttechnik GmbH.

Wünsche an eine abgewandelte bzw. neue Hard- oder Softwarekonfiguration richten Sie bitte an WAGO Kontakttechnik GmbH.

1.2 Symbole



Gefahr
Informationen unbedingt beachten, um Personen vor Schaden zu bewahren.



Achtung
Informationen unbedingt beachten, um am Gerät Schäden zu verhindern.



Beachten
Randbedingungen, die für einen fehlerfreien Betrieb unbedingt zu beachten sind.



ESD (Electrostatic Discharge)
Warnung vor Gefährdung der Komponenten durch elektrostatische Entladung. Vorsichtsmaßnahme bei Handhabung elektrostatisch entladungsgefährdeter Bauelemente beachten.



Hinweis
Routinen oder Ratschläge für den effizienten Geräteeinsatz und die Softwareoptimierung.



Weitere Informationen
Verweise auf zusätzliche Literatur, Handbücher, Datenblätter und INTERNET Seiten.

1.3 Schriftkonventionen

kursiv	Namen von Pfaden und Dateien sind als kursive Begriffe gekennzeichnet. z. B.: *C:\Programme\WAGO-IO-CHECK*
kursiv	Menüpunkte sind als kursive Begriffe fett gekennzeichnet. z. B.: ***Speichern***
\	Ein Backslash zwischen zwei Namen bedeutet die Auswahl eines Menüpunktes aus einem Menü. z. B.: ***Datei \ Neu***
ENDE	Schaltflächen sind fett und mit Kapitälchen dargestellt z. B.: **EINGABE**
< >	Tasten-Beschriftungen sind in spitzen Klammern eingefasst und fett dargestellt z. B.: **<F5>**
`Courier`	Programmcodes werden in der Schriftart Courier gedruckt. z. B.: `END_VAR`

1.4 Darstellungen der Zahlensysteme

Zahlensystem	Beispiel	Bemerkung
Dezimal	100	normale Schreibweise
Hexadezimal	0x64	C-Notation
Binär	'100' '0110.0100'	in Hochkomma, Nibble durch Punkt getrennt

1.5 Gültigkeitsbereich

Dieses Handbuch beschreibt folgende Komponenten:

Artikel-Nr.	Beschreibung
759-312	API Modbus/TCP DLL

1.6 Abkürzungen

DI Digitaleingang (Digital Input)

DO Digitalausgang (Digital Output)

I/O [Input/Output] Ein- / Ausgang

ID Identifier, Identifikation, eindeutige Kennzeichnung

2 Modbus/TCP DLL

2.1 Überblick

Die DLL implementiert das Modbus/TCP Protokoll.

Die Modbus/TCP DLL unterstützt die Betriebssysteme Windows NT 4.0 (ab SP5), Windows 2000, Windows 95 (mit Windows Socket 2.0 Update), Windows 98.

Es wird die im jeweiligen Windows System enthaltene Windows Socket 2.0 Schnittstelle für die TCP/IP Kommunikation benutzt.

Die DLL unterstützt synchrones und asynchrones Lesen und Schreiben von Werten.

Als Transportprotokoll kann wahlweise TCP oder UDP gewählt werden.

Sie läßt sich von den Programmiersprachen C und Visual Basic aus nutzen.

Bei Aufrufen der DLL aus Visual Basic werden lediglich synchrone Aufrufe unterstützt.

Bei Aufrufen der DLL aus C werden sowohl synchrone als auch asynchrone Aufrufe unterstützt.

Diese Library unterstützt vom Open Modbus/TCP Protokoll V1.0 nur die Kommandos: FC1, FC2, FC3, FC4, FC7, FC15 und FC16.

Die Werte der Modbus/TCP Tabellen, die mit diesen Kommandos gelesen bzw. geschrieben werden können sind in der nachfolgenden Tabelle aufgeführt:

	Lesen	**Schreiben**
Output Register	FC 3	FC 16
Input Register	FC 4	
Output Coil	FC 1	FC 15
Input Coil	FC 2	
Exception Status	FC 7	

Die Library achtet darauf, daß pro Verbindung zu einem Modbus/TCP Gerät nur ein Kommando parallel verarbeitet werden kann.

Für jede geöffnete Verbindung kann eine Zeit angegeben werden, nach der eine I/O Anfrage abgebrochen wird.

Die Entwicklung der DLL wurde mit der Microsoft Visual C++ 6.0 Entwicklungsumgebung durchgeführt. Alle Module der DLL sind als ASCII Komponenten mit statisch gebundener C-Runtime übersetzt.

2.2 Installation

Die Datei *.DLL sollte in das Verzeichnis \system32 im Windows-Standardverzeichnis kopiert werden. Wird ein anderes Verzeichnis gewählt, so muß in der Windows-Systemsteuerung bei den Umgebungsvariablen der Eintrag für den Pfad entsprechend angepasst werden.

2.3 Funktionen

Alle Funktionen der MBT Library haben Rückgabewerte, die dem HRESULT Format entsprechen. Die Funktionen des Socket APIs geben keine Rückgabewerte dieses Formates zurück. Die MBT Library konvertiert diese Rückgabewerte mittels des Makros HRESULT_FROM_WIN32. Bei der nachfolgenden Beschreibung wird dies mittels "HR von" gekennzeichnet.

Folgende Funktionen sind in der Modbus/TCP.DLL enthalten:

MBTInit
MBTExit
MBTConnect
MBTDisconnect
MBTReadRegisters
MBTReadCoils
MBTReadExeptionStatus
MBTReadCompleted
MBTWriteRegisters
MBTWriteCoils
MBTWriteCompleted
MBTSwapWord
MBTSwapDWord
MODBUSTCP_TABLE_xxx

2.3.1 MBTInit

WAGO-I/O-*PRO* 32 Elemente der Dynamic Link Library	
Kategorie:	Diese Funktion initialisiert die Modbus/TCP Library.
Name:	MBTInit
Typ:	Funktion
Name der DLL:	ModbusTCP.DLL
Anwendbar für:	Modbus/TCP-Protokoll

Deklaration:

```
C:      LONG MBTInit(void);

VB:     Public Declare Function MBTInit Lib "MBT" () As Long
```

Parameter:	Kommentar:
keine	

Rückgabewerte:	HEX-Wert:	Kommentar:
S_OK	0	MBT Library wurde erfolgreich initialisiert
MBT_THREAD_ CREATION_ERROR	0xEF010000	Der Abeitsthread der MBT Library konnte nicht erzeugt werden
HR von WSASYSNOTREADY	0x8007276B	Das Netzwerk Subsystem ist nicht bereit für die Netzwerk-Kommunikation
HR von WSAVERNOT SUPPORTED	0x8007276C	Die benötigte Windows Socket Version 2 wird vom System nicht angeboten
HR von WSAEINPROGRESS	0x80072734	Eine blockierende Windows Socket 1.1 Operation ist aktiv
HR von WSAEPROCLIM	0x80072753	Die maximale Anzahl an Threads, die von der von der Socketimplementierung unterstützt wird ist erreicht

Bemerkung:

Erst nach dem Aufruf der Funktion MBTInit können die anderen Funktionen der MBT Library genutzt werden.

2.3.2 MBTExit

WAGO-I/O-*PRO* 32 Elemente der Dynamic Link Library		
Kategorie:	Diese Funktion terminiert die Modbus/TCP Library.	
Name:	MBTExit	
Typ:	Funktion	
Name der DLL:	ModbusTCP.DLL	
Anwendbar für:	Modbus/TCP-Protokoll	
Deklaration:		
C: LONG MBTExit(void); VB: Public Declare Function MBTExit Lib "MBT" () As Long		
Parameter:	**Kommentar:**	
keine		
Rückgabewerte:	**HEX-Wert:**	**Kommentar:**
S_OK	0	MBT Library wurde erfolgreich initialisiert
MBT_EXIT_TIMEOUT_ERROR	0xEF010001	Der Arbeitsthread der MBT Library konnte nicht innerhalb des Timeouts beendet werden
MBT_UNKNOWN_THREAD_EXIT_ERROR	0xEF010002	Der Arbeitsthread der MBT Library hat sich mit einem Fehler-Code beendet
Bemerkung:		
Die Funktion MBTExit ist für ein kontrolliertes Beenden der Modbus/TCP Library aufzurufen.		

2.3.3 MBTConnect

WAGO-I/O-*PRO* 32 Elemente der Dynamic Link Library	
Kategorie:	Diese Funktion erzeugt eine Verbindung mit dem übergebenen Port des Gerätes.
Name:	MBTConnect
Typ:	Funktion
Name der DLL:	ModbusTCP.DLL
Anwendbar für:	Modbus/TCP-Protokoll

Deklaration:

```
C:      LONG MBTConnect(
                IN LPCTSTR szHostAddress,
                IN WORD port,
                IN BOOL useTCPorUDP,
                IN DWORD requestTimeout,
                OUT HANDLE *hSocket);

VB:     Public Declare Function MBTConnect Lib "MBT" (
                ByVal szHostAddress As String,
                ByVal port As Integer,
                ByVal useTCPorUDP As Long,
                ByVal requestTimeout As Long,
                hSocket As Long) As Long
```

Parameter:	**Kommentar:**
szHostAddress	IP-Adresse (z.B. "172.17.5.91") oder DNS Name des Modbus/TCP Gerätes
port	IP-Port über den die Kommunikation mit dem Gerät geführt werden soll
useTCPorUDP	Kommunikation über TCP (TRUE) oder UDP (FALSE)
requestTimeout	Timeout in Millisekunden für eine I/O Operation zum Gerät
hSocket	Handle für die erzeugte Verbindung

Rückgabewerte:	**HEX-Wert:**	**Kommentar:**
S_OK	0	Die Verbindung wurde aufgebaut
MBT_NO_ENTRY_ ADDABLE_ERROR	0xEF010004	Fehler in der Socket-Verwaltung der Library
HR von WSANOTINITIALISED	0x8007276D	MBTInit wurde nicht aufgerufen
HR von WSAENETDOWN	0x80072742	Fehler im Netzwerk-Subsystem
HR von WSANO_ RECOVERY	0x80072AFB	Ein Fataler Fehler ist aufgetreten

HR von WSAHOST_NOT_FOUND	0x80072AF9	Das Gerät für den übergebenen DNS Name wurde nicht gefunden oder ungültige IP Adresse
HR von WSATRY_AGAIN	0x80072AFA	Fehler im DNS Server
HR von WSAEPROTONOSUPPORT WSAEAFNOSUPPORT	0x8007273B 0x8007273F	Das TCP oder UPD Protokoll wird nicht unterstützt.
HR von WSAEMFILE	0x80072728	Das System hat keine weiteren Socket Descriptors zur Verfügung
HR von WSAEADDRNOTAVAIL	0x80072741	Die Remote Adresse ist ungültig
HR von WSAECONNREFUSED	0x8007274D	Der Verbindungsversuch wurde abgewiesen
HR von WSAETIMEDOUT	0x8007274C	Timeout während des Verbindungsaufbaus

Bemerkung:

Diese Funktion erzeugt einen Windows Socket Objekt. Dieses wird für das TCP Protokoll mit dem Prot des Modbus/TCP Gerätes verbunden. Beim UDP Protokoll wird der Port des Gerätes in der Library gespeichert.

Erst nachdem eine Verbindung zu einem Gerät aufgebaut wurde, können Werte des Gerätes gelesen und geschrieben werden.

2.3.4 MBTDisconnect

WAGO-I/O-*PRO* 32 Elemente der Dynamic Link Library		
Kategorie:	Diese Funktion bricht die Verbindung zu einem Gerät ab.	
Name:	MBTDisconnect	
Typ:	Funktion	
Name der DLL:	ModbusTCP.DLL	
Anwendbar für:	Modbus/TCP-Protokoll	
Deklaration:		
C: LONG MBTDisconnect(IN HANDLE hSocket); VB: Public Declare Function MBTDisonnect Lib "MBT" (ByVal hSocket As Long) As Long		
Parameter:	**Kommentar:**	
hSocket	Handle der Verbindung zum Gerät	
Rückgabewerte:	**HEX-Wert:**	**Kommentar:**
S_OK	0	Die Verbindung wurde abgebaut
MBT_HANDLE_INVALID_ERROR	0xEF010006	Das übergebene Handle ist ungültig
HR von WSANOTINITIALISED	0x8007276D	MBTInit wurde nicht aufgerufen
HR von WSAENETDOWN	0x80072742	Fehler im Netzwerk-Subsystem
Bemerkung:		
Für ein kontrolliertes Beenden einer Verbindung zu einem Gerät ist die Funktion MBTDisconnect aufzurufen.		

2.3.5 MBTReadRegisters

WAGO-I/O-*PRO* 32 Elemente der Dynamic Link Library	
Kategorie:	Diese Funktion liest Werte von Registern.
Name:	MBTReadRegisters
Typ:	Funktion
Name der DLL:	ModbusTCP.DLL
Anwendbar für:	Modbus/TCP-Protokoll

Deklaration:

```
C:      LONG MBTReadRegisters(
                IN HANDLE hSocket,
                IN BYTE tableType,
                IN WORD dataStartAddress,
                IN WORD numWords,
                OUT LPBYTE pReadBuffer,
                OPTIONAL IN MBTReadCompleted fpReadCompletedCallback
                OPTIONAL IN DWORD callbackContext);

VB:     Public Declare Function MBTReadRegisters Lib "MBT" (
                ByVal hSocket As Long,
                ByVal tableType As Byte,
                ByVal dataStartAddress As Integer,
                ByVal numWords As Integer,
                pReadBuffer As Any,
                ByVal fpReadCompletedCallback As Long,
                ByVal callbackContext As Long) As Long
```

Parameter:	**Kommentar:**
hSocket	Handle der Verbindung zum Gerät
tableType	Modbus/TCP Tabellen Typ (MODBUS_TABLE_xxx) Hier gültig: Input Register oder Output Register
dataStartAddress	Startadresse der zu lesenden Register
numWords	Anzahl der zu lesenden Register
pReadBuffer	Speicherbereich, in den die Register gelesen werden sollen. Bei asynchronem Aufruf ist NULL zu übergeben.
FpReadCompletedCallback	C-Callback Funktion, die nach dem Abschluß eines asynchronen Lesens aufgerufen wird. Bei synchronem Aufruf (default) ist NULL zu übergeben.
callbackContext	Kontext, der der asynchronen Callback Funktion mitgegeben wird. Bei synchronem Aufruf (default) ist 0 zu übergeben.

Rückgabewerte:	**HEX-Wert:**	**Kommentar:**
S_OK	0	Die Verbindung wurde abgebaut

MBT_HANDLE_INVALID_ ERROR	0xEF010006	Das übergebene Handle ist ungültig
MBT_NO_JOB_ ADDABLE_ERROR	0xEF010005	Fehler in der Job Verwaltung der Library
MBT_EXIT_ERROR	0xEF01000B	Der I/O-Auftrag wurde durch ein MBTExit Call abgebrochen
MBT_SOCKET_ TIMEOUT_ERROR	0xEF010008	Ein Modbus/TCP Telegramm für den I/O-Auftrag wurde abgeschickt. Innerhalb des Request-Timeouts wurde jedoch keine Antwort empfangen.
HR von WSANOTINITIALISED	0x8007276D	MBTInit wurde nicht aufgerufen
HR von WSAENETDOWN	0x80072742	Feher im Netzwerk-Subsystem
HR von WSAENETRESET WSAECONNABORTED WSAEDISCON	0x80072744 0x80072745 0x80072746	Die Verbindung zum Gerät wurde unterbrochen
HR von WSAEWOULDBLOCK	0x80072733	Zu viele I/O Operationen sind zur Zeit ausständig
HR von WSAEFAULT	0x8007271E	Der übergebene ReadBuffer zeigt auf keinen gültigen Speicherbereich
HR von WSAENOBUFS	0x80072747	Ein Auftrag an einem Socket konnte aufgrund unzureichendem Pufferspeicher oder einer vollen Warteschlange nicht durchgeführt werden.

Bemerkung:

Der Parameter fpReadCompletedCallback entscheidet, ob die Funktion asynchron oder synchron ausgeführt wird. Ist dieser Parameter NULL, so wird die Funktion synchron, ansonsten asynchron ausgeführt. Nach der Beendigung der asynchronen I/O Operation wird die übergebene Callback Funktion mit dem Ergebnis der I/O Operation aufgerufen. Beim asynchronen Aufruf der Funktion wird ein übergebener Buffer (pReadBuffer) ignoriert und das Ergebnis in einem Speicherbereich der DLL geliefert.

Für das Lesen von Output Registern wird das Modbus/TCP Kommando FC3 für die Input Register das Kommando FC4 benutzt.

2.3.6 MBTReadCoils

WAGO-I/O-*PRO* 32 Elemente der Dynamic Link Library	
Kategorie:	Diese Funktion liest Werte von Coils.
Name:	MBTReadCoils
Typ:	Funktion
Name der DLL:	ModbusTCP.DLL
Anwendbar für:	Modbus/TCP-Protokoll

Deklaration:

```
C:      LONG MBTReadCoils(
                IN HANDLE hSocket,
                IN BYTE tableType
                IN WORD dataStartAddress,
                IN WORD numBits,
                OUT LPBYTE pReadBuffer,
                OPTIONAL IN MBTReadCompleted fpReadCompletedCallback
                OPTIONAL IN DWORD callbackContext);

VB:     Public Declare Function MBTReadCoils Lib "MBT" (
                ByVal hSocket As Long,
                ByVal tableType As Byte,
                ByVal dataStartAddress As Integer,
                ByVal numBits As Integer,
                pReadBuffer As Any,
                ByVal fpReadCompletedCallback As Long,
                ByVal callbackContext As Long) As Long
```

Parameter:	**Kommentar:**
hSocket	Handle der Verbindung zum Gerät
tableType	Modbus/TCP Tabellen Typ (MODBUS_TABLE_xxx) Hier gültig: Input Coil oder Output Coil
dataStartAddress	Startadresse der zu lesenden Coils
numBits	Anzahl der zu lesenden Coils
pReadBuffer	Speicherbereich, in den die Register gelesen werden sollen. Bei asynchronem Aufruf ist NULL zu übergeben.
fpReadCompletedCallback	C-Callback Funktion, die nach dem Abschluß eines asynchronen Lesens aufgerufen wird. Bei synchronem Aufruf (default) ist NULL zu übergeben.
callbackContext	Kontext, der der asynchronen Callback Funktion mitgegeben wird. Bei synchronem Aufruf (default) ist 0 zu übergeben.

Rückgabewerte:	**HEX-Wert:**	**Kommentar:**
S_OK	0	Die Verbindung wurde abgebaut

MBT_HANDLE_INVALID_ ERROR	0xEF010006	Das übergebene Handle ist ungültig
MBT_NO_JOB_ ADDABLE_ERROR	0xEF010005	Fehler in der Job Verwaltung der Library
MBT_EXIT_ERROR	0xEF01000B	Der I/O-Auftrag wurde durch ein MBTExit Call abgebrochen
MBT_SOCKET_ TIMEOUT_ERROR	0xEF010008	Ein Modbus/TCP Telegramm für den I/O-Auftrag wurde abgeschickt. Innerhalb des Request-Timeouts wurde jedoch keine Antwort empfangen.
HR von WSANOTINITIALISED	0x8007276D	MBTInit wurde nicht aufgerufen
HR von WSAENETDOWN	0x80072742	Fehler im Netzwerk-Subsystem
HR von WSAENETRESET WSAECONNABORTED WSAEDISCON	0x80072744 0x80072745 0x80072746	Die Verbindung zum Gerät wurde unterbrochen
HR von WSAEWOULDBLOCK	0x80072733	Zu viele I/O Operationen sind zur Zeit ausständig
HR von WSAEFAULT	0x8007271E	Der übergebene ReadBuffer zeigt auf keinen gültigen Speicherbereich
HR von WSAENOBUFS	0x80072747	Ein Auftrag an einem Socket konnte aufgrund unzureichendem Pufferspeicher oder einer vollen Warteschlange nicht durchgeführt werden.

Bemerkung:

Der Parameter fpReadCompletedCallback entscheidet, ob die Funktion asynchron oder synchron ausgeführt wird. Ist dieser Parameter NULL, so wird die Funktion synchron, ansonsten asynchron ausgeführt. Nach der Beendigung der asynchronen I/O Operation wird die übergebene Callback Funktion mit dem Ergebnis der I/O Operation aufgerufen. Beim asynchronen Aufruf der Funktion wird ein übergebener Buffer (pReadBuffer) ignoriert und das Ergebnis in einem Speicherbereich der DLL geliefert.

Für das Lesen von Output Coils wird das Modbus/TCP Kommando FC1 für die Input Register das Kommando FC2 benutzt.

2.3.7 MBTReadExceptionStatus

WAGO-I/O-*PRO* 32 Elemente der Dynamic Link Library	
Kategorie:	Diese Funktion liest Werte von Coils.
Name:	MBTReadExceptionStatus
Typ:	Funktion
Name der DLL:	ModbusTCP.DLL
Anwendbar für:	Modbus/TCP-Protokoll

Deklaration:

```
C:      LONG MBTReadCoils(
                IN HANDLE hSocket,
                OUT LPBYTE pReadBuffer,
                OPTIONAL IN MBTReadCompleted fpReadCompletedCallback
                OPTIONAL IN DWORD callbackContext);

VB:     Public Declare Function MBTReadCoils Lib "MBT" (
                ByVal hSocket As Long,
                pReadBuffer As Any,
                ByVal fpReadCompletedCallback As Long,
                ByVal callbackContext As Long) As Long
```

Parameter:	**Kommentar:**
hSocket	Handle der Verbindung zum Gerät
pReadBuffer	Speicherbereich, in den die Register gelesen werden sollen. Bei asynchronem Aufruf ist NULL zu übergeben.
fpReadCompletedCallback	C-Callback Funktion, die nach dem Abschluß eines asynchronen Lesens aufgerufen wird. Bei synchronem Aufruf (default) ist NULL zu übergeben.
callbackContext	Kontext, der der asynchronen Callback Funktion mitgegeben wird. Bei synchronem Aufruf (default) ist 0 zu übergeben.

Rückgabewerte:	**HEX-Wert:**	**Kommentar:**
S_OK	0	Die Verbindung wurde abgebaut
MBT_HANDLE_INVALID_ERROR	0xEF010006	Das übergebene Handle ist ungültig
MBT_NO_JOB_ADDABLE_ERROR	0xEF010005	Fehler in der Job Verwaltung der Library
MBT_EXIT_ERROR	0xEF01000B	Der I/O-Auftrag wurde durch ein MBTExit Call abgebrochen

MBT_SOCKET_ TIMEOUT_ERROR	0xEF010008	Ein Modbus/TCP Telegramm für den I/O-Auftrag wurde abgeschickt. Innerhalb des Request-Timeouts wurde jedoch keine Antwort empfangen.
HR von WSANOTINITIALISED	0x8007276D	MBTInit wurde nicht aufgerufen
HR von WSAENETDOWN	0x80072742	Fehler im Netzwerk-Subsystem
HR von WSAENETRESET WSAECONNABORTED WSAEDISCON	0x80072744 0x80072745 0x80072746	Die Verbindung zum Gerät wurde unterbrochen
HR von WSAEWOULDBLOCK	0x80072733	Zu viele I/O Operationen sind zur Zeit ausständig
HR von WSAEFAULT	0x8007271E	Der übergebene ReadBuffer zeigt auf keinen gültigen Speicherbereich
HR von WSAENOBUFS	0x80072747	Ein Auftrag an einem Socket konnte aufgrund unzureichendem Pufferspeicher oder einer vollen Warteschlange nicht durchgeführt werden.

Bemerkung:

Der Parameter fpReadCompletedCallback entscheidet, ob die Funktion asynchron oder synchron ausgeführt wird. Ist dieser Parameter NULL, so wird die Funktion synchron, ansonsten asynchron ausgeführt. Nach der Beendigung der asynchronen I/O Operation wird die übergebene Callback Funktion mit dem Ergebnis der I/O Operation aufgerufen. Beim asynchronen Aufruf der Funktion wird ein übergebener Buffer (pReadBuffer) ignoriert und das Ergebnis in einem Speicherbereich der DLL geliefert.

Für das Lesen des Exception Status wird das Modbus/TCP Kommando FC7 benutzt.

2.3.8 MBTReadCompleted

WAGO-I/O-*PRO* 32 Elemente der Dynamic Link Library		
Kategorie:	Die Callback Funktion wird beim Abschluß eines asynchronen Lesens aufgerufen.	
Name:	MBTReadCompleted	
Typ:	Funktion	
Name der DLL:	ModbusTCP.DLL	
Anwendbar für:	Modbus/TCP-Protokoll	
Deklaration:		

```
C:      void MBTReadCompleted(
               IN HANDLE hSocket,
               IN DWORD callbackContext,
               IN LONG errorCode,
               IN BYTE tableType,
               IN WORD dataStartAddress,
               IN WORD numRead,
               IN WORD numBytes,
               IN LPBYTE pReadBuffer);
```

Parameter:	**Kommentar:**
hSocket	Handle der Verbindung zum Gerät
callbackContext	Kontext, der beim asynchronen Aufruf der Funktion mitgegeben wurde
errorCode	Ergebnis des Lesevorgangs
tableType	Modbus/TCP Tabellen Typ (MOBUSTCP_TABLE_xxx)
dataStartAddress	Startadresse der zu lesenden Register oder Coils
numRead	Anzahl der zu lesenenden Register oder Coils
numBytes	Anzahl der gelesenen Bytes
pReadBuffer	Werte der gelesenen Bytes

Rückgabewerte:	**HEX-Wert:**	**Kommentar:**
keine		

Bemerkung:

2.3.9 MBTWriteRegisters

WAGO-I/O-*PRO* 32 Elemente der Dynamic Link Library	
Kategorie:	Diese Funktion schreibt Werte von Registern.
Name:	MBTWriteRegisters
Typ:	Funktion
Name der DLL:	ModbusTCP.DLL
Anwendbar für:	Modbus/TCP-Protokoll

Deklaration:

```
C:      LONG MBTWriteRegisters(
                IN HANDLE hSocket,
                IN WORD dataStartAddress,
                IN WORD numWords,
                IN LPBYTE pWriteBuffer,
                OPTIONAL IN MBTWriteCompleted fpWriteCompletedCallback
                OPTIONAL IN DWORD callbackContext);

VB:     Public Declare Function MBTWriteRegisters Lib "MBT" (
                ByVal hSocket As Long,
                ByVal dataStartAddress As Integer,
                ByVal numWords As Integer,
                pWriteBuffer As Any,
                ByVal fpWriteCompletedCallback As Long,
                ByVal callbackContext As Long) As Long
```

Parameter:	Kommentar:
hSocket	Handle der Verbindung zum Gerät
dataStartAddress	Startadresse der zu schreibenden Register
numWords	Anzahl der zu schreibenden Register
pWriteBuffer	Speicherbereich mit den Werten der zu schreibenden Register
fpWriteCompletedCallback	C-Callback Funktion, die nach dem Abschluß eines asynchronen Schreibens aufgerufen wird. Bei synchronem Aufruf (default) ist NULL zu übergeben.
callbackContext	Kontext, der der asynchronen Callback Funktion mitgegeben wird. Bei synchronem Aufruf (default) ist 0 zu übergeben.

Rückgabewerte:	HEX-Wert:	Kommentar:
S_OK	0	Die Verbindung wurde abgebaut
MBT_HANDLE_INVALID_ERROR	0xEF010006	Das übergebene Handle ist ungültig
MBT_NO_JOB_ADDABLE_ERROR	0xEF010005	Fehler in der Job Verwaltung der Library

MBT_EXIT_ERROR	0xEF01000B	Der I/O-Auftrag wurde durch ein MBTExit Call abgebrochen
MBT_SOCKET_ TIMEOUT_ERROR	0xEF010008	Ein Modbus/TCP Telegramm für den I/O-Auftrag wurde abgeschickt. Innerhalb des Request-Timeouts wurde jedoch keine Antwort empfangen.
HR von WSANOTINITIALISED	0x8007276D	MBTInit wurde nicht aufgerufen
HR von WSAENETDOWN	0x80072742	Fehler im Netzwerk-Subsystem
HR von WSAENETRESET WSAECONNABORTED WSAEDISCON	0x80072744 0x80072745 0x80072746	Die Verbindung zum Gerät wurde unterbrochen
HR von WSAEWOULDBLOCK	0x80072733	Zu viele I/O Operationen sind zur Zeit ausständig
HR von WSAEFAULT	0x8007271E	Der übergebene WriteBuffer zeigt auf keinen gültigen Speicherbereich
HR von WSAENOBUFS	0x80072747	Ein Auftrag an einem Socket konnte aufgrund unzureichendem Pufferspeicher oder einer vollen Warteschlange nicht durchgeführt werden.

Bemerkung:

Der Parameter fpWriteCompletedCallback entscheidet, ob die Funktion asynchron oder synchron ausgeführt wird. Ist dieser Parameter NULL, so wird die Funktion synchron, ansonsten asynchron ausgeführt. Nach der Beendigung der asynchronen I/O Operation wird die übergebene Callback Funktion mit dem Ergebnis der I/O Operation aufgerufen.

Für das Schreiben von Output Registern wird das Modbus/TCP Kommando FC16 benutzt.

2.3.10 MBTWriteCoils

WAGO-I/O-*PRO* 32 Elemente der Dynamic Link Library	
Kategorie:	Diese Funktion schreibt Werte von Coils.
Name:	MBTWriteCoils
Typ:	Funktion
Name der DLL:	ModbusTCP.DLL
Anwendbar für:	Modbus/TCP-Protokoll

Deklaration:

```
C:      LONG MBTWriteCoilss(
                IN HANDLE hSocket,
                IN WORD dataStartAddress,
                IN WORD numBits,
                IN LPBYTE pWriteBuffer,
                OPTIONAL IN MBTWriteCompleted fpWriteCompletedCallback
                OPTIONAL IN DWORD callbackContext);

VB:     Public Declare Function MBTWriteCoils Lib "MBT" (
                ByVal hSocket As Long,
                ByVal dataStartAddress As Integer,
                ByVal numBits As Integer,
                pWriteBuffer As Any,
                ByVal fpWriteCompletedCallback As Long,
                ByVal callbackContext As Long) As Long
```

Parameter:	**Kommentar:**
hSocket	Handle der Verbindung zum Gerät
dataStartAddress	Startadresse der zu schreibenden Coils
numBits	Anzahl der zu schreibenden Coils
pWriteBuffer	Speicherbereich mit den Werten der zu schreibenden Coils
fpWriteCompletedCallback	C-Callback Funktion, die nach dem Abschluß eines asynchronen Schreibens aufgerufen wird. Bei synchronem Aufruf (default) ist NULL zu übergeben.
callbackContext	Kontext, der der asynchronen Callback Funktion mitgegeben wird. Bei synchronem Aufruf (default) ist 0 zu übergeben.

Rückgabewerte:	**HEX-Wert:**	**Kommentar:**
S_OK	0	Die Verbindung wurde abgebaut
MBT_HANDLE_INVALID_ERROR	0xEF010006	Das übergebene Handle ist ungültig
MBT_NO_JOB_ADDABLE_ERROR	0xEF010005	Fehler in der Job Verwaltung der Library

MBT_EXIT_ERROR	0xEF01000B	Der I/O-Auftrag wurde durch ein MBTExit Call abgebrochen
MBT_SOCKET_ TIMEOUT_ERROR	0xEF010008	Ein Modbus/TCP Telegramm für den I/O-Auftrag wurde abgeschickt. Innerhalb des Request-Timeouts wurde jedoch keine Antwort empfangen.
HR von WSANOTINITIALISED	0x8007276D	MBTInit wurde nicht aufgerufen
HR von WSAENETDOWN	0x80072742	Fehler im Netzwerk-Subsystem
HR von WSAENETRESET WSAECONNABORTED WSAEDISCON	0x80072744 0x80072745 0x80072746	Die Verbindung zum Gerät wurde unterbrochen
HR von WSAEWOULDBLOCK	0x80072733	Zu viele I/O Operationen sind zur Zeit ausständig
HR von WSAEFAULT	0x8007271E	Der übergebene WriteBuffer zeigt auf keinen gültigen Speicherbereich
HR von WSAENOBUFS	0x80072747	Ein Auftrag an einem Socket konnte aufgrund unzureichendem Pufferspeicher oder einer vollen Warteschlange nicht durchgeführt werden.

Bemerkung:

Der Parameter fpWriteCompletedCallback entscheidet, ob die Funktion asynchron oder synchron ausgeführt wird. Ist dieser Parameter NULL, so wird die Funktion synchron, ansonsten asynchron ausgeführt. Nach der Beendigung der asynchronen I/O Operation wird die übergebene Callback Funktion mit dem Ergebnis der I/O Operation aufgerufen.

Für das Schreiben von Output Registern wird das Modbus/TCP Kommando FC15 benutzt.

2.3.11 MBTWriteCompleted

WAGO-I/O-*PRO* 32 Elemente der Dynamic Link Library		
Kategorie:	Die Callback Funktion wird beim Abschluß eines asynchronen Schreibens aufgerufen.	
Name:	MBTWriteCompleted	
Typ:	Funktion	
Name der DLL:	ModbusTCP.DLL	
Anwendbar für:	Modbus/TCP-Protokoll	

Deklaration:

```
C:      void MBTWriteCompleted(
                IN HANDLE hSocket,
                IN DWORD callbackContext,
                IN LONG errorCode,
                IN BYTE tableType,
                IN WORD dataStartAddress,
                IN WORD numWrite,
                IN LPBYTE pWriteBuffer);
```

Parameter:	**Kommentar:**
hSocket	Handle der Verbindung zum Gerät
callbackContext	Kontext, der beim asynchronen Aufruf der Funktion mitgegeben wurde
errorCode	Ergebnis des Schreibvorgangs
tableType	Modbus/TCP Tabellen Typ (MOBUSTCP_TABLE_xxx)
dataStartAddress	Startadresse der zu schreibenden Register oder Coils
numWrite	Anzahl der zu schreibenden Register oder Coils
pWriteBuffer	Werte der zu schreibenden Register oder Coils

Rückgabewerte:	**HEX-Wert:**	**Kommentar:**
keine		

Bemerkung:

2.3.12 MBTSwapWord

WAGO-I/O-*PRO* 32 Elemente der Dynamic Link Library	
Kategorie:	Diese Funktion vertauscht das untere und das obere Byte des Arguments vom Typ WORD.
Name:	MBTSwapWord
Typ:	Funktion
Name der DLL:	ModbusTCP.DLL
Anwendbar für:	Modbus/TCP-Protokoll

Deklaration:

```
C:      WORD MBTSwapWord(
                const WORD wData);

VB:     Public Declare Function MBTSwapWord Lib "MBT" (
                ByVal wData As Integer) As Integer
```

Parameter:	**Kommentar:**
wData	Word das geswappt werden soll

Rückgabewerte:	**Kommentar:**
	Diese Funktion liefert den geswappten Wert zurück.

Bemerkung:

2.3.13 MBTSwapDWord

WAGO-I/O-*PRO* 32 Elemente der Dynamic Link Library	
Kategorie:	Diese Funktion vertauscht das 1. und 4. und das 2. und 3. Byte des DWord Arguments.
Name:	MBTSwapDWord
Typ:	Funktion
Name der DLL:	ModbusTCP.DLL
Anwendbar für:	Modbus/TCP-Protokoll

Deklaration:

```
C:      DWORD MBTSwapDWord(
                const DWORD dwData);

VB:     Public Declare Function MBTSwapDWord Lib "MBT" (
                ByVal dwData As Long) As Long
```

Parameter:	**Kommentar:**
dwData	DWord das geswappt werden soll

Rückgabewerte:	**Kommentar:**
	Diese Funktion liefert den geswappten Wert zurück.

Bemerkung:

2.3.14 MODBUSTCP_TABLE_xxx

Das Define MODBUSTCP_TABLE_xxx bestimmt die Modbus/TCP Tabelle, auf die eine Aktion ausgeführt werden soll.

	Define	Wert
Output Register	MODBUSTCP_TABLE_OUTPUT_REGISTER	4
Input Register	MODBUSTCP_TABLE_INPUT_REGISTER	3
Output Coil	MODBUSTCP_TABLE_OUTPUT_COIL	0
Input Coil	MODBUSTCP_TABLE_INPUT_COIL	1
Exception Status	MODBUSTCP_TABLE_EXCEPTION_STATUS	7

2.4 Beispiele

2.4.1 VBA

2.4.1.1 Kurzbeschreibung

Basis für die Steuerung mit Visual Basic ist ein Formular mit den Steuerungselementen btnReadCoil, btnWriteCoil, btnReadRegister und btnWriteRegister vom Typ Button sowie den Steuerungselementen AdrCoil, edtReadCoil, edtWriteCoil, AdrRegister, edtReadRegister und edtWriteRegister vom Typ Eingabefeld.

Mit dem Öffnen des Formulars wird die Modbus/TCP DLL initialisiert und eine Verbindung zum Knoten aufgebaut.

Im Eingabefeld AdrCoil bzw. AdrRegister wird die anzusprechende Adresse eingetragen.
Nach Anklicken des Buttons btnReadCoil bzw. btnReadRegister wird der Status des Eingangs im Eingabefeld edtReadCoil bzw. edtReadRegister angezeigt.
Nach Anklicken des Buttons btnWriteCoil bzw. btnWriteRegister wird der im Eingabefeld edtWriteCoil bzw. edtWriteRegister eingetragene Wert zum Ausgang geschrieben.
Die Ereignisprozeduren werden durch das Ereignis "Click" der Buttons aufgerufen.

Mit dem Schliessen des Formulars wird die Verbindung zum Knoten abgebaut und die Modbus/TCP DLL terminiert.

2.4.1.2 Deklaration der Konstanten und Funktionen der DLL

Konstanten

```
Const MODBUSTCP_TABLE_OUTPUT_REGISTER = 4
Const MODBUSTCP_TABLE_INPUT_REGISTER = 3
Const MODBUSTCP_TABLE_OUTPUT_COIL = 0
Const MODBUSTCP_TABLE_INPUT_COIL = 1
Const MODBUSTCP_TABLE_EXCEPTION_STATUS = 7
```

DLL-Funktionen

```
Public Declare Function MBTInit Lib "MBT" () As Long
Public Declare Function MBTExit Lib "MBT" () As Long
Public Declare Function MBTConnect Lib "MBT" (ByVal szHostAddress As String, ByVal port As Integer, ByVal useTCPorUDP As Long, ByVal requestTimeout As Long, hSocket As Long) As Long
Public Declare Function MBTDisconnect Lib "MBT" (ByVal hSocket As Long) As Long
Public Declare Function MBTReadRegisters Lib "MBT" (ByVal hSocket As Long, ByVal tableType As Byte, ByVal dataStartAddress As Integer, ByVal numWords As Integer, pReadBuffer As Any, ByVal fpReadCompletedCallback As Long, ByVal callbackContext As Long) As Long
Public Declare Function MBTReadCoils Lib "MBT" (ByVal hSocket As Long, ByVal tableType As Byte, ByVal dataStartAddress As Integer, ByVal numBits As Integer, pReadBuffer As Any, ByVal fpReadCompletedCallback As Long, ByVal callbackContext As Long) As Long
Public Declare Function MBTReadExceptionStatus Lib "MBT" (ByVal hSocket As Long, pExceptionStatus As Byte, ByVal fpReadCompletedCallback As Long, ByVal callbackContext As Long) As Long
Public Declare Function MBTWriteRegisters Lib "MBT" (ByVal hSocket As Long, ByVal
```

```
dataStartAddress As Integer, ByVal numWords As Integer, pWriteBuffer As Any, By-
Val fpWriteCompletedCallback As Long, ByVal callbackContext As Long) As Long
Public Declare Function MBTWriteCoils Lib "MBT" (ByVal hSocket As Long, ByVal
dataStartAddress As Integer, ByVal numBits As Integer, pWriteBuffer As Any, ByVal
fpWriteCompletedCallback As Long, ByVal callbackContext As Long) As Long
Public Declare Function MBTSwapWord Lib "MBT" (ByVal wData As Integer) As Integer
Public Declare Function MBTSwapDWord Lib "MBT" (ByVal dwData As Long) As Long
```

2.4.1.3 Deklaration der Variablen und Ereignisprozeduren des Formulars

Globale Variablen und Konstanten

```
Const g_MBusIP As String = "172.17.5.91"
Const g_Port As Long = 502
Dim g_hSocket As Long
```

btnReadCoil

```
Private Sub btnReadCoil_Click()
Dim inpVal As Byte
Dim adr As Integer
    inpVal = 0
    adr = CInt(AdrCoil.Text)
    ret = MBTReadCoils(g_hSocket, _
                    MODBUSTCP_TABLE_OUTPUT_COIL, _
                    adr, _
                    1, _
                    inpVal, _
                    0, _
                    0)
    edtReadCoil.Text = inpVal
End Sub
```

btnReadRegister

```
Private Sub btnReadRegister_Click()
Dim inpVal As Integer
Dim adr As Integer
    inpVal = 0
    adr = CInt(AdrRegister.Text)
    ret = MBTReadRegisters(g_hSocket, _
                    MODBUSTCP_TABLE_OUTPUT_REGISTER, _
                    adr, _
                    1, _
                    inpVal, _
                    0, _
                    0)
    inpVal = MBTSwapWord(inpVal)
    edtReadRegister.Text = inpVal
End Sub
```

btnWriteCoil

```
Private Sub btnWriteCoil_Click()
Dim outVal As Byte
Dim adr As Integer
    outVal = CByte(edtWriteCoil.Text)
    adr = CInt(AdrCoil.Text)
    ret = MBTWriteCoils(g_hSocket, _
                     adr, _
                     1, _
                     outVal, _
                     0, _
                     0)
End Sub
```

btnWriteRegister

```
Private Sub btnWriteRegister_Click()
```

```
Dim outVal As Integer
Dim adr As Integer
    outVal = CInt(edtWriteRegister.Text)
    outVal = MBTSwapWord(outVal)
    adr = CInt(AdrRegister.Text)
    ret = MBTWriteRegisters(g_hSocket, _
                    adr, _
                    1, _
                    outVal, _
                    0, _
                    0)
End Sub
```

Form

```
Private Sub Form_Load()
Dim ret As Long
    MBTInit
    ret = MBTConnect(g_MBusIP, g_Port, True, 1000, g_hSocket)
    If (ret <> 0) Then
        MBTExit
        MsgBox "Couldn't connect to MB Device: 0x" & Hex(ret)
        Unload Me
        Exit Sub
    End If
End Sub

Private Sub Form_Unload(Cancel As Integer)
    MBTDisconnect (g_hSocket)
    MBTExit
End Sub
```

2.4.2 C

2.4.2.1 Kurzbeschreibung

Im folgenden C-Beispielprogramm wird mit der Funktion MBTInit die DLL initialisiert und danach mit MBTConnect eine Verbindung aufgebaut.

Nach erfolgreichem Verbindungsaufbau werden nacheinander die Funktionen MBTWriteRegisters, MBTReadRegisters, MBTWriteCoils, MBTReadCoils und MBTReadExceptionStatus zum Schreiben und Lesen der Aus- und Eingänge asynchron aufgerufen. Anschließend werden die Funktionen nacheinander synchron aufgerufen.
Nach jedem Funktionsaufruf wird der Status auf dem Monitor angezeigt.

Am Programmende wird mit der Funktion MBTDisconnect die Verbindung abgebaut und dann mit MBTExit die DLL terminiert.

2.4.2.2 Interface

```
#ifndef _MBT_H_
#define _MBT_H_

#ifdef __cplusplus
extern "C"
{
#endif


#define MODBUSTCP_TABLE_OUTPUT_REGISTER   4
#define MODBUSTCP_TABLE_INPUT_REGISTER    3
#define MODBUSTCP_TABLE_OUTPUT_COIL       0
#define MODBUSTCP_TABLE_INPUT_COIL        1
#define MODBUSTCP_TABLE_EXCEPTION_STATUS  7


//--------------------------------------------------------------------------
// Type Definitions                                                        |
//--------------------------------------------------------------------------

typedef void (WINAPI *MBTReadCompleted)(
   IN HANDLE hSocket,          // socket handle
   IN DWORD callbackContext,   // callback context, handed over at the call
   IN LONG errorCode,          // result of the read operation
   IN BYTE tableType,          // type of MODBUS/TCP tables(MODBUSTCP_TABLE_xxx)
   IN WORD dataStartAddress,   // start address of the registers or coils to be
                               // read
   IN WORD numRead,            // number of the registers or coils to be read
   IN WORD numBytes,           // number of the bytes to be read
   IN LPBYTE pReadBuffer       // memory section with the data to be written
   );

typedef void (WINAPI *MBTWriteCompleted)(
   IN HANDLE hSocket,          // socket handle
   IN DWORD callbackContext,   // callback context, handed over at the call
   IN LONG errorCode,          // result of the write operation
   IN BYTE tableType,          // type of MODBUS/TCP tables(MODBUSTCP_TABLE_xxx)
                               // output registers or output coils
   IN WORD dataStartAddress,   // start address of the registers or coils to be
                               // written
   IN WORD numWrite,           // number of the registers or coils to be written
```

```
   IN LPBYTE pWriteBuffer     // memory section with the data to be written
   );


//----------------------------------------------------------------------------
// Prototypes                                                                 |
//----------------------------------------------------------------------------

// initializes the MODBUS/TCP library
LONG WINAPI MBTInit();

// terminates the MODBUS/TCP library
LONG WINAPI MBTExit();

// creates a socket and connects it to the given device port
LONG WINAPI MBTConnect(
   IN LPCTSTR szHostAddress,  // TCP/IP address of device
   IN WORD port,              // TCP port in device for communication
   IN BOOL useTCPorUDP,       // TRUE - TCP; FALSE - UDP
   IN DWORD requestTimeout,   // maximal time for managing an I/O request (ms)
   OUT HANDLE *hSocket        // handle of the connected socket
   );

// aborts the connection to a device and releases the socket
LONG WINAPI MBTDisconnect(
   IN HANDLE hSocket          // handle of the connected socket
   );

// read from a connected socket
LONG WINAPI MBTReadRegisters(
   IN HANDLE hSocket,         // handle of the connected socket
   IN BYTE tableType,         // Modbus/TCP Tabellen Typ (MODBUSTCP_TABLE_xxx)
                              // (here: input register or output register
   IN WORD dataStartAddress,  // start address of the registers to be read
   IN WORD numWords,          // number of the registers to be read
   OUT LPBYTE pReadBuffer,    // memory section from which the data are read
                              // (NULL at asynchronous call)
   OPTIONAL IN MBTReadCompleted fpReadCompletedCallback = NULL,
                              // C-callback function, called after termination
                              // of asynchronous reading (NULL at synchronous
                              // call)
   OPTIONAL IN DWORD callbackContext = 0
                              // context, handed over to the asynchronous
                              // (callback function (0 at synchronous call)
   );

// write to a connected socket
LONG WINAPI MBTWriteRegisters(
   IN HANDLE hSocket,         // handle of the connected socket
   IN WORD dataStartAddress,  // start address of the registers to be written
   IN WORD numWords,          // number of the registers to be written
   IN LPBYTE pWriteBuffer,    // memory section from which the data are written
                              // (NULL at asynchronous call)
   OPTIONAL IN MBTWriteCompleted fpWriteCompletedCallback = NULL,
                              // C-callback function, called after termination
                              // of asynchronous writing (NULL at synchronous
                              // call)
   OPTIONAL IN DWORD callbackContext = 0
                              // context, handed over to the asynchronous
                              // (callback function (0 at synchronous call)
   );

// read from a connected socket
LONG WINAPI MBTReadCoils(
   IN HANDLE hSocket,         // handle of the connected socket
```

```
   IN BYTE tableType,         // Modbus/TCP Tabellen Typ (MODBUSTCP_TABLE_xxx)
                              // (here: input coil or output coil
   IN WORD dataStartAddress,  // start address of the coils to be read
   IN WORD numBits,           // number of the coils to be read
   OUT LPBYTE pReadBuffer,    // memory section from which the data are read
                              // (NULL at asynchronous call)
   OPTIONAL IN MBTReadCompleted fpReadCompletedCallback = NULL,
                              // C-callback function, called after termination
                              // of asynchronous reading (NULL at synchronous
                              // call)
   OPTIONAL IN DWORD callbackContext = 0
                              // context, handed over to the asynchronous
                              // (callback function (0 at synchronous call)
   );

// write to a connected socket
LONG WINAPI MBTWriteCoils(
   IN HANDLE hSocket,         // handle of the connected socket
   IN WORD dataStartAddress,  // start address of the coils to be written
   IN WORD numBits,           // number of the coils to be written
   IN LPBYTE pWriteBuffer,    // memory section from which the data are written
                              // (NULL at asynchronous call)
   OPTIONAL IN MBTWriteCompleted fpWriteCompletedCallback = NULL,
                              // C-callback function, called after termination
                              // of asynchronous writing (NULL at synchronous
                              // call)
   OPTIONAL IN DWORD callbackContext = 0
                              // context, handed over to the asynchronous
                              // (callback function (0 at synchronous call)
   );

// read from a connected socket
LONG WINAPI MBTReadExceptionStatus(
   IN HANDLE hSocket,         // handle of the connected socket
   OUT LPBYTE pExceptionStatus,
                              // memory section from which the data are read
                              // (NULL at asynchronous call)
   OPTIONAL IN MBTReadCompleted fpReadCompletedCallback = NULL,
                              // C-callback function, called after termination
                              // of asynchronous reading (NULL at synchronous
                              // call)
   OPTIONAL IN DWORD callbackContext = 0
                              // context, handed over to the asynchronous
                              // (callback function (0 at synchronous call)
   );

WORD WINAPI MBTSwapWord(const WORD wData);

DWORD WINAPI MBTSwapDWord(const DWORD dwData);

#ifdef __cplusplus
}
#endif

#endif
```

2.4.2.3 Programm

```
#include <windows.h>
#include <tchar.h>
#include <assert.h>
```

```
#include <stdio.h>
#include "MBT.h"

#define DEFAULT_PORT             502
#define DEFAULT_SERVER_NAME      _T("172.17.5.90")
#define DEFAULT_PROTOCOL         TRUE       /* TCP */
#define DEFAULT_REQUEST_TIMEOUT  1000   /* in ms */

LONG ret;
char pWriteBuffer[] = "This is a small test message.\n";
bool received;

void WINAPI rcb(
   IN HANDLE hSocket,          // socket handle
   IN DWORD callbackContext,   // callback context, handed over at the call
   IN LONG errorCode,          // result of the read operation
   IN BYTE tableType,          // type of MODBUS/TCP tables(MODBUSTCP_TABLE_xxx)
   IN WORD dataStartAddress,   // start address of the registers or coils to be
                               // read
   IN WORD numRead,            // number of the registers or coils to be read
   IN WORD numBytes,           // number of the bytes to be read
   IN LPBYTE pReadBuffer       // memory section with the data to be written
   )
{
  int i;

  printf( "\nread callback called:\n"
          "\tcontext: %lu\n"
          "\terror code: %ld\n"
          "\ttable type: %d\n"
          "\tdata start address: %d\n"
          "\tnumber of read registers/coils: %d\n"
          "\tnumber of read bytes: %d\n"
          "\tread bytes: ",
          callbackContext,
          errorCode,
          tableType,
          dataStartAddress,
          numRead,
          numBytes
          );

  if( errorCode == S_OK )
  {
    for( i = 0; i < numBytes; ++i )
    {
      printf( "%d ", pReadBuffer[i] );
    }
  }

  printf( "\n" );

  received = true;
}

void WINAPI wcb(
   IN HANDLE hSocket,          // socket handle
   IN DWORD callbackContext,   // callback context, handed over at the call
   IN LONG errorCode,          // result of the write operation
   IN BYTE tableType,          // type of MODBUS/TCP tables(MODBUSTCP_TABLE_xxx)
                               // output registers or output coils
   IN WORD dataStartAddress,   // start address of the registers or coils to be
                               // written
   IN WORD numWrite,           // number of the registers or coils to be written
   IN LPBYTE pWriteBuffer      // memory section with the data to be written
```

```
   )
{
  int i;
  int numBytes;

  printf( "\nwrite callback called:\n"
          "\tcontext: %lu\n"
          "\terror code: %ld\n"
          "\ttable type: %d\n"
          "\tdata start address: %d\n"
          "\tnumber of written registers/coils: %d\n"
          "\twritten bytes: ",
          callbackContext,
          errorCode,
          tableType,
          dataStartAddress,
          numWrite
          );

  if( errorCode == S_OK )
  {
    switch( tableType )
    {
        case MODBUSTCP_TABLE_OUTPUT_REGISTER:
            numBytes = 2 * numWrite;
            break;
        case MODBUSTCP_TABLE_OUTPUT_COIL:
            numBytes = (numWrite + 7) / 8;
            break;
    }

    for( i = 0; i < numBytes; ++i )
    {
      printf( "%d ", pWriteBuffer[i] );
    }
  }

  printf( "\n" );

  received = true;
}

void WINAPI esrcb(
   IN HANDLE hSocket,          // socket handle
   IN DWORD callbackContext,   // callback context, handed over at the call
   IN LONG errorCode,          // result of the read operation
   IN BYTE tableType,          // type of MODBUS/TCP tables(MODBUSTCP_TABLE_xxx)
   IN WORD dataStartAddress,   // start address of the registers or coils to be
                               // read
   IN WORD numRead,            // number of the registers or coils to be read
   IN WORD numBytes,           // number of the bytes to be read
   IN LPBYTE pReadBuffer       // memory section with the data to be written
   )
{
  int i;

  printf( "\nexception status read callback called:\n"
          "\tcontext: %lu\n"
          "\terror code: %ld\n"
          "\ttable type: %d\n"
          "\tdata start address: %d\n"
          "\tnumber of read registers/coils: %d\n"
          "\tnumber of read bytes: %d\n"
          "\tread bytes: ",
          callbackContext,
```

```
          errorCode,
          tableType,
          dataStartAddress,
          numRead,
          numBytes
          );

  if( errorCode == S_OK )
  {
    for( i = 0; i < numBytes; ++i )
    {
      printf( "%d ", pReadBuffer[i] );
    }
  }

  printf( "\n" );

  received = true;
}


int main(int argc, char* argv[])
{
  HANDLE hSocket;
  int i = 1234;
  int ch;
  BYTE pWriteBuffer[6] = { 0xff, 0xff, 0xff, 0xff, 0xff, 0xff };
  BYTE pWriteBuffer2[6] = { 0x33, 0x33, 0x33, 0x33, 0x33, 0x33 };
  BYTE pReadBuffer[6];

  ret = MBTInit();

  if( ret == S_OK )
  {
    ret = MBTConnect( DEFAULT_SERVER_NAME, DEFAULT_PORT, DEFAULT_PROTOCOL,
                      DEFAULT_REQUEST_TIMEOUT, &hSocket );
  }

  if( ret == S_OK )
  {
    received = false;
    ret = MBTWriteRegisters(
                hSocket,
                0x0000,
                3,
                (LPBYTE) pWriteBuffer,
                wcb,
                i
                );

  }

  if( ret == S_OK )
  {
    while( !received )
    {
      Sleep( 100 );
    }
  }


  if( ret == S_OK )
  {
    received = false;
    ret = MBTReadRegisters(
```

```
                hSocket,
                MODBUSTCP_TABLE_OUTPUT_REGISTER,
                0x0200,
                3,
                NULL,
                rcb,
                i
                );
}

if( ret == S_OK )
{
  while( !received )
  {
    Sleep( 100 );
  }
}

if( ret == S_OK )
{
  received = false;
  ret = MBTWriteCoils(
                hSocket,
                0x0000,
                6,
                (LPBYTE) pWriteBuffer,
                wcb,
                i
                );

}

if( ret == S_OK )
{
  while( !received )
  {
    Sleep( 100 );
  }
}

if( ret == S_OK )
{
  received = false;
  ret = MBTReadCoils(
                hSocket,
                MODBUSTCP_TABLE_OUTPUT_COIL,
                0x0200,
                6,
                NULL,
                rcb,
                i
                );
}

if( ret == S_OK )
{
  while( !received )
  {
    Sleep( 100 );
  }
}

if( ret == S_OK )
{
  received = false;
```

```
  // read from a connected socket
  ret = MBTReadExceptionStatus(
    hSocket,          // handle of the connected socket
    NULL,             // memory section from which the data are read
                      // (NULL at asynchronous call)
    esrcb,            // C-callback function, called after termination
                      // of asynchronous reading (NULL at synchronous
                      // call)
    i                 // context, handed over to the asynchronous
                      // (callback function (0 at synchronous call)
 );
}

if( ret == S_OK )
{
  while( !received )
  {
    Sleep( 100 );
  }
}

printf( "Geben Sie <RETURN> ein!\n" );
scanf( "%c", &ch );

//****************** Sync calls ****************************************

if( ret == S_OK )
{
  printf( "\n\n\n****************** Sync calls ******************\n\n" );

  ret = MBTWriteRegisters(
              hSocket,
              0x0000,
              3,
              (LPBYTE) pWriteBuffer2,
              NULL,
              0
              );
}

if( ret == S_OK )
{
  ret = MBTReadRegisters(
              hSocket,
              MODBUSTCP_TABLE_OUTPUT_REGISTER,
              0x0200,
              3,
              pReadBuffer,
              NULL,
              0
              );
}

if( ret == S_OK )
{
  printf( "\nWrite/Read registers: " );

  for( i = 0; i < 2 * 3; ++i )
  {
    printf( "%d ", pReadBuffer[i] );
  }

  ret = MBTWriteCoils(
              hSocket,
```

```
                0x0000,
                6,
                (LPBYTE) pWriteBuffer2,
                NULL,
                0
                );
}

if( ret == S_OK )
{
  ret = MBTReadCoils(
                hSocket,
                MODBUSTCP_TABLE_OUTPUT_COIL,
                0x0200,
                6,
                pReadBuffer,
                NULL,
                0
                );
}

if( ret == S_OK )
{
  printf( "\nWrite/Read coils: " );

  for( i = 0; i < (6 + 7) / 8; ++i )
  {
    printf( "%d ", pReadBuffer[i] );
  }

  ret = MBTReadExceptionStatus(
    hSocket,          // handle of the connected socket
    pReadBuffer,      // memory section from which the data are read
                      // (NULL at asynchronous call)
    NULL,             // C-callback function, called after termination
                      // of asynchronous reading (NULL at synchronous
                      // call)
    0                 // context, handed over to the asynchronous
                      // (callback function (0 at synchronous call)
  );
}

if( ret == S_OK )
{
  printf( "\nRead exception status: " );

  for( i = 0; i < 1; ++i )
  {
    printf( "%d ", pReadBuffer[i] );
  }
}


if( ret == S_OK )
{
  ret = MBTDisconnect( hSocket );
}

if( ret == S_OK )
{
  ret = MBTExit();
}

if( ret != S_OK )
{
```

```
    fprintf( stderr, "### Error No %ld\n", ret );
  }

  printf( "\nGeben Sie <RETURN> ein!\n" );
  scanf( "%c", &ch );

  return 0;
}
```

3 Index

B

Beispiele 27

D

Deklaration der Konstanten und Funktionen der DLL 27
Deklaration der Variablen und Ereignisprozeduren des Formulars 28

F

Funktionen 6

I

Installation 6
Interface 30

K

Kurzbeschreibung 27, 30

M

MBTConnect 9
MBTDisconnect 11
MBTExit 8
MBTInit 7
MBTReadCoils 14
MBTReadCompleted 18
MBTReadExceptionStatus 16
MBTReadRegisters 12
MBTSwapDWord 25
MBTSwapWord 24
MBTWriteCoils 21
MBTWriteCompleted 23
MBTWriteRegisters 19
Modbus/TCP DLL 5
MODBUSTCP_TABLE_xxx 26

P

Programm 33

Ü

Überblick 5

V

VBA 27