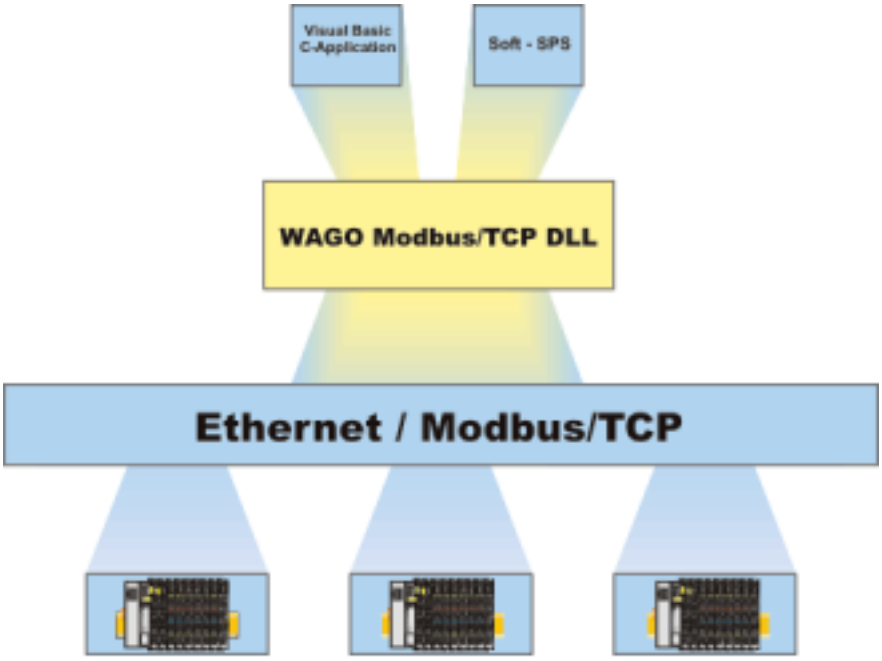
WAGO-I/O-SYSTEM 750

API Modbus/TCP DLL



取り扱い説明書

Version 1.2.0.J

WAGO®
INNOVATIVE CONNECTIONS

WAGO Kontakttechnik GmbH
Hansastraße 27
D-32423 Minden

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

E-Mail: info@wago.com

Web: http://www.wago.com

Technical Support
Phone: +49 (0) 571/8 87 – 5 55
Fax: +49 (0) 571/8 87 – 85 55

E-Mail: support@wago.com

本書の作成には万全を期しておりますが、お気づきの点やご意見がございましたら下記までお知らせください。
〒136-0071　東京都江東区亀戸 1-5-7 日鐵 ND タワー
ワゴジャパン株式会社　I/O グループ
TEL,03-5627-2059　FAX,03-5627-2055

この取り扱い説明書において使用される会社名、ソフトウェアおよびハードウェアの名称は、一般的に商標法または特許法により保護されています。

目次

1 重要事項

本書が対象とする製品のインストールおよびスタートアップを迅速に行うために、以下の情報と説明を十分に読んで理解し、その内容を順守してください。

1.1 法的原則

1.1.1 著作権

本書は図表を含めてすべて著作権で保護されています。本書に明記された著作権条項に抵触する使用は禁じられています。複製、翻訳、電子的手段または複写による保存および修正を行うには、ワゴコンタクトテクニック社（ドイツ）の同意書が必要です。これに違反した場合、当社には損害賠償を請求する権利が生じます。

1.1.2 使用者の資格基準

本書で説明する製品は、PLC プログラミングの資格を有する技術者、電気機器の専門技術者、または適用規格を熟知している電気機器の専門技術者の指導を受けた者が必ず操作してください。不適切な作業による損害、または本書の内容を順守しないために発生したワゴ製品および他社製品の損害について、ワゴコンタクトテクニック社（ドイツ）は一切の責任を負いかねますのでご了承ください。

1.1.3 用途

使用されるコンポーネントは各用途に応じて、専用のハードウェアおよびソフトウェアコンフィグレーションで動作するようになっています。変更する場合は、必ず本書で記述された範囲内で行ってください。ハードウェアやソフトウェアに対してそれ以外の変更を加えた場合や、コンポーネントが規格に準じて使用されなかった場合は、ワゴコンタクトテクニック社（ドイツ）の責任範囲外となりますのでご注意ください。改造版および／または新規のハードウェアまたはソフトウェアコンフィグレーションに関する要件については、ワゴジャパン株式会社まで直接お問い合わせください。

1.2 記号

危険
ケガをしないために必ず守ってください。

警告
製品にダメージを与えない為に必ず守ってください。

注意
円滑な動作を確保するため、限界条件を必ず守ってください。

ESD (Electrostatic Discharge)
静電気により製品にダメージを与える恐れがある場合の警告です。

Note
装置の効果的な使用およびソフトウェア最適化のための手順やヒントです。

詳細について
追加文書、取り扱い説明書、データシート、インターネットホームページを参照してください。

1.3 フォントの表記

Italic	ファイル名及びパス名は斜字体で表します。 例: *C:\programs\WAGO-IO-CHECK*
Italic	メニューアイテムは太字斜字体で表します。 例: ***Save***
\	メニューアイテムを順番に行う場合にバックスラッシュで表します。 例: ***File\New***
END	ボタンを押す操作は太字で小さな大文字で表します。 例: **ENTER**
< >	キーを押す操作は< >で囲って表します。 例: **<F5>**
`Courier`	プログラムコードは `Courier` フォントで表します。 例: `END_VAR`

1.4 数字の表記

コード	例	備考
Decimal	100	通常表記
Hexadecimal	0x64	C 言語での表記
Binary	'100' '0110.0100'	シングルクォーテーションで囲う 1/2 バイトづつドットで区切る

1.5 有効範囲

製品名	内容
759-312	API Modbus/TCP DLL

1.6 略語

AI	アナログ入力
AO	アナログ出力
DI	デジタル入力
DO	デジタル出力
I/O	入出力

2 Modbus/TCP DLL

2.1 概要

この DLL は Modbus/TCP Protocol を実行する為のものです。

Modbus/TCP DLL は次の OS に対応しています。Windows NT 4.0 (from version SP5), Windows 2000, Windows 95 (with Windows Socket 2.0 Update), Windows 98.

TCP/IP 通信を行う為に Windows システムの Windows Socket 2.0 を使用します。

この DLL はシンクロナス（同期）とアシンクロナス（非同期）で値の読み書きを行います。

伝送プロトコルは TCP と UDP が選択できます。

プログラミング言語は C と VisualBASIC です。

Visual BASIC からは DLL のシンクロナス（同期）コールだけが行えます。

C からは DLL のシンクロナス（同期）、アンシンクロナス（非同期）コールの両方が行えます。

この DLL は Modbus/TCP protocol V1.0.の FC1, FC2, FC3, FC4, FC7, FC15 , FC16 に対応しています。

Modbus/TCP テーブルの値は次の表で表します。

	Read	**Write**
Output Register	FC 3	FC 16
Input Register	FC 4	
Output Coil	FC 1	FC 15
Input Coil	FC 2	
Exception Status	FC 7	

この DLL は１つの Modbus/TCP デバイスに対し複数のコネクションから同時にコマンドを送らないように管理しています。

I/O への問い合わせが中断されたあとコネクションをオープンする間のインターバルタイムを設定できます。

この DLL は Microsoft Visual C++ 6.0 の開発環境で開発されました。DLL の全てのモジュールは C ランタイムと静的リンクするアスキーコンポーネンツです。

2.2 インストール

DLL のファイル名は MBT.DLL です。Windows の system32 フォルダにコピーしてください。もし他のフォルダにコピーする場合は Windows の環境変数に適切な登録しなければなりません。

2.3 関数

MBT ライブラリの全ての関数の戻り値は HRESULT フォーマットに対応します。API ソケット関数はどんな戻り値でもこのフォーマットでは返しません。MBT ライブラリがその戻り値を HRESULT_FROM_WIN32 に変換します。以降、本マニュアルで述べる HRESULT_FROM_WIN32 を "HR from"と記述します。

Modbus/TCP.DLL には次の関数が含まれています。

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 Elements of Dynamic Link Library		
種類:	この関数は ModbusTCP ライブラリを初期化します	
関数名:	MBTInit	
タイプ:	Function	
DLL 名:	ModbusTCP.DLL	
適用:	Modbus/TCP-Protocol	
宣言:		
C: LONG MBTInit(void); VB: Public Declare Function MBTInit Lib "MBT" () As Long		
パラメータ:	**コメント:**	
無		
戻り値:	**HEX 値:**	**コメント:**
S_OK	0	MBT ライブラリの初期化に成功しました。
MBT_THREAD_ CREATION_ERROR	0xEF010000	MBT ライブラリのワークスレッドは作成できませんでした。
HR from WSASYSNOTREADY	0x8007276B	ネットワークサブシステムはネットワーク接続の 準備ができていません。
HR from WSAVERNOT SUPPORTED	0x8007276C	必要な Windows socket version 2 がシステム上にありません。
HR from WSAEINPROGRESS	0x80072734	Windows socket 1.1 の動作が障害になっています。
HR from WSAEPROCLIM	0x80072753	サポートされたソケットの実行スレッド数が最大値に達しています。
注意:		
他の MBT ライブラリ関数は MBTInit 関数をコールしたあとでなければ使用できません。		

2.3.2 MBTExit

WAGO-I/O-*PRO* 32 Elements of Dynamic Link Library		
種類:	この関数は Modbus/TCP ライブラリを終了します。	
関数名:	MBTExit	
タイプ:	Function	
DLL 名:	ModbusTCP.DLL	
適用:	Modbus/TCP-Protocol	
宣言:		
C: LONG MBTExit(void); VB: Public Declare Function MBTExit Lib "MBT" () As Long		
パラメータ:	**コメント:**	
無		
戻り値:	**HEX 値:**	**コメント:**
S_OK	0	MBT ライブラリの終了に成功しました。
MBT_EXIT_TIMEOUT_ ERROR	0xEF010001	タイムアウトエラー
MBT_UNKNOWN_ THREAD_EXIT_ERROR	0xEF010002	未知のエラーで終了しました。
注意:		
Modbus/TCP ライブラリの使用を終了するときに必ずコールしてください。		

2.3.3 MBTConnect

WAGO-I/O-*PRO* 32 Elements of Dynamic Link Library		
種類:	この関数は Modbus/TCP 機器を制御する為にポートを開き接続を行います。	
関数名:	MBTConnect	
タイプ:	Function	
DLL 名:	ModbusTCP.DLL	
適用:	Modbus/TCP-Protocol	
宣言:		

```
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
```

パラメータ:	**コメント:**
szHostAddress	Modbus/TCP 機器の IP-アドレス (例 "172.17.5.91") か DNS 名
port	Modbus/TCP 機器と接続する為の IP ポート（通常は 502）
useTCPorUDP	接続方法　TCP (TRUE) か UDP (FALSE)
requestTimeout	タイムアウト(msec)
hSocket	コネクションハンドラ(これをもとに接続を管理します)

戻り値:	**HEX 値:**	**コメント:**
S_OK	0	接続が確立されました
MBT_NO_ENTRY_ ADDABLE_ERROR	0xEF010004	ライブラリのソケット管理エラーが発生しました
HR from WSANOTINITIALISED	0x8007276D	MBTInit がまだコールされていません
HR from WSAENETDOWN	0x80072742	ネットワークシステムエラーが発生しました
HR from WSANO_ RECOVERY	0x80072AFB	致命的エラーが発生しました
HR from WSAHOST_NOT_FOUND	0x80072AF9	指定された IP アドレスの機器が存在しないか DNS 名が間違えている可能性があります

HR from WSATRY_AGAIN	0x80072AFA	DNS サーバーエラーです
HR from WSAEPROTONOSUPPORT WSAEAFNOSUPPORT	0x8007273B 0x8007273F	指定した TCP か UDP プロトコルがサポートされていません。
HR from WSAEMFILE	0x80072728	システムがこの種類のソケットを 利用できません
HR from WSAEADDRNOTAVAIL	0x80072741	リモートアドレスが無効です
HR from WSAECONNREFUSED	0x8007274D	接続が拒絶されました
HR from WSAETIMEDOUT	0x8007274C	コネクションタイムアウト

注意:

この関数は Windows socket オブジェクトを作成します。TCP プロトコルで MODBUS/TCP デバイスのポートと接続します。このライブラリには UDP プロトコルも用意されています。

接続が確立してからデバイスのリード、ライトを行ってください。

2.3.4 MBTDisconnect

WAGO-I/O-*PRO* 32 Elements of Dynamic Link Library		
種類:	この関数はデバイスとの接続を切断します。	
関数名:	MBTDisconnect	
タイプ e:	Function	
DLL 名:	ModbusTCP.DLL	
適用:	Modbus/TCP-Protocol	
宣言:		
C: LONG MBTDisconnect(IN HANDLE hSocket); VB: Public Declare Function MBTDisonnect Lib "MBT" (ByVal hSocket As Long) As Long		
パラメータ:	**コメント:**	
hSocket	コネクションハンドラ	
戻り値:	**HEX 値:**	**コメント:**
S_OK	0	切断されました.
MBT_HANDLE_INVALID_ERROR	0xEF010006	コネクションハンドラが無効です.
HR from WSANOTINITIALISED	0x8007276D	MBTInit 関数がコールされていません.
HR from WSAENETDOWN	0x80072742	ネットワークシステムエラーが発生しました
注意:		
MBTDisconnect はデバイスとの接続を切断する関数です。		

2.3.5 MBTReadRegisters

WAGO-I/O-*PRO* 32 Elements of Dynamic Link Library	
種類:	この関数はレジスタからデータを読み出します。.
関数名:	MBTReadRegisters
タイプ:	Function
DLL 名:	ModbusTCP.DLL
適用:	Modbus/TCP-Protocol

宣言:

```
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
```

パラメータ:	コメント:
hSocket	コネクションハンドラ
tableType	Modbus/TCP tables Type (MODBUS_TABLE_xxx) INPUT レジスタ及び OUTPUT レジスタに対応（FC3,FC4）
dataStartAddress	読み出し開始位置
numWords	読み出し数
pReadBuffer	読み出したレジスタを格納するエリア アシンクロナスコールの時は NULL にして下さい
FpReadCompletedCallback	アシンクロナスリードファンクションを完了したあとにコールされる C のコールバックファンクション シンクロナスコールの時は NULL にして下さい(デフォルト).
callbackContext	アシンクロナスコールバックファンクションの状態 シンクロナスコールの時は 0 にして下さい(デフォルト).

戻り値:	HEX 値:	コメント:
S_OK	0	正常終了

MBT_HANDLE_INVALID_ ERROR	0xEF010006	コネクションハンドラが無効です.
MBT_NO_JOB_ ADDABLE_ERROR	0xEF010005	ライブラリのソケット管理エラーが発生しました.
MBT_EXIT_ERROR	0xEF01000B	MBTExit コールによりジョブが中断されました
MBT_SOCKET_ TIMEOUT_ERROR	0xEF010008	I/O に対して Modbus/TCP コマンドを送ったがレスポンスが返らずリクエストタイムアウトに なりました
HR from WSANOTINITIALISED	0x8007276D	MBTInit がコールされていません
HR from WSAENETDOWN	0x80072742	ネットワークエラーです.
HR from WSAENETRESET WSAECONNABORTED WSAEDISCON	0x80072744 0x80072745 0x80072746	デバイスとの接続はすでに中止されています.
HR from WSAEWOULDBLOCK	0x80072733	大量の I/O 操作が未処理になっています.
HR from WSAEFAULT	0x8007271E	リードバッファが有効な格納先を参照していません.
HR from WSAENOBUFS	0x80072747	操作を実行する為の十分なシステムバッファが不足しています.

注意:

パラメータ fpReadCompletedCallback は関数がアシンクロナスまたはシンクロナスで実行されるかどうか決定します。このパラメータが NULL ならシンクロナス、NULL でなければアシンクロナスで実行されます。アシンクロナス I/O 操作のあと、コールバックファンクションはコールと一緒に I/O 操作の結果を返します。関数をアシンクロナスでコールした場合 pReadBuffer は無視され結果は DLL の 格納先に入ります。

The Modbus/TCP コマンド FC3 は output レジスタ、 コマンド FC4 は input レジスタの読出しに使用されます。

2.3.6 MBTReadCoils

WAGO-I/O-*PRO* 32 Elements of Dynamic Link Library	
種類:	This function reads data from Coils.
関数名:	MBTReadCoils
タイプ:	Function
DLL 名:	ModbusTCP.DLL
適用:	Modbus/TCP-Protocol

宣言:

```
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
```

パラメータ:	**コメント:**
hSocket	コネクションハンドラ
tableType	Modbus/TCP tables Typ (MODBUS_TABLE_xxx) Applicable here: Input coil or output coil
dataStartAddress	読み出し開始位置
numBits	読み出し数
pReadBuffer	読み出した値を格納するエリア アシンクロナスコールの時は NULL にして下さい
fpReadCompletedCallback	アシンクロナスリードファンクションを完了したあとにコールされる C のコールバックファンクション シンクロナスコールの時は NULL にして下さい(デフォルト).
callbackContext	アシンクロナスコールバックファンクションの状態 シンクロナスコールの時は 0 にして下さい(デフォルト).

戻り値:	**HEX 値:**	**コメント:**
S_OK	0	正常終了.

MBT_HANDLE_INVALID_ ERROR	0xEF010006	コネクションハンドラが無効です.
MBT_NO_JOB_ ADDABLE_ERROR	0xEF010005	ライブラリのソケット管理エラーが発生しました.
MBT_EXIT_ERROR	0xEF01000B	MBTExit コールによりジョブが中断されました.
MBT_SOCKET_ TIMEOUT_ERROR	0xEF010008	I/O に対して Modbus/TCP コマンドを送ったがレスポンスが返らずリクエストタイムアウトに なりました ..
HR from WSANOTINITIALISED	0x8007276D	MBTInit がコールされていません .
HR from WSAENETDOWN	0x80072742	ネットワークエラーです.
HR from WSAENETRESET WSAECONNABORTED WSAEDISCON	0x80072744 0x80072745 0x80072746	デバイスとの接続はすでに中止されています.
HR from WSAEWOULDBLOCK	0x80072733	大量の I/O 操作が未処理になっています.
HR from WSAEFAULT	0x8007271E	リードバッファが有効な格納先を参照していません.
HR from WSAENOBUFS	0x80072747	操作を実行する為の十分なシステムバッファが不足しています.

注意:

パラメータ fpReadCompletedCallback は関数がアシンクロナスまたはシンクロナスで実行されるかどうか決定します。このパラメータが NULL ならシンクロナス、NULL でなければアシンクロナスで実行されます。アシンクロナス I/O 操作のあと、コールバックファンクションはコールと一緒に I/O 操作の結果を返します。関数をアシンクロナスでコールした場合 pReadBuffer は無視され結果は DLL の 格納先に入ります。

Modbus/TCP コマンド FC1 は output コイル 、コマンド FC2 は input コイルの読出しに使用します。

2.3.7 MBTReadExceptionStatus

WAGO-I/O-*PRO* 32 Elements of Dynamic Link Library	
種類:	この関数は例外ステータスを読み出します
関数名:	MBTReadExceptionStatus
タイプ:	Function
DLL 名:	ModbusTCP.DLL
適用:	Modbus/TCP-Protocol

宣言:

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

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

パラメータ:	コメント:
hSocket	コネクションハンドラ
pReadBuffer	読み出した値を格納するエリア アシンクロナスコールの時は NULL にして下さい
fpReadCompletedCallback	アシンクロナスリードファンクションを完了したあとにコールされる C のコールバックファンクション シンクロナスコールの時は NULL にして下さい(デフォルト).
callbackContext	アシンクロナスコールバックファンクションの状態 シンクロナスコールの時は 0 にして下さい(デフォルト).

戻り値:	HEX 値:	コメント:
S_OK	0	正常終了.
MBT_HANDLE_INVALID_ERROR	0xEF010006	コネクションハンドラが無効です
MBT_NO_JOB_ADDABLE_ERROR	0xEF010005	ライブラリのソケット管理エラーが発生しました.
MBT_EXIT_ERROR	0xEF01000B	MBTExit コールによりジョブが中断されました.

MBT_SOCKET_ TIMEOUT_ERROR	0xEF010008	I/O に対して Modbus/TCP コマンドを送ったがレスポンスが返らずリクエストタイムアウトに なりました ..
HR from WSANOTINITIALISED	0x8007276D	MBTInit がコールされていません .
HR from WSAENETDOWN	0x80072742	ネットワークエラーです.
HR from WSAENETRESET WSAECONNABORTED WSAEDISCON	0x80072744 0x80072745 0x80072746	デバイスとの接続はすでに中止されています.
HR from WSAEWOULDBLOCK	0x80072733	大量の I/O 操作が未処理になっています.
HR from WSAEFAULT	0x8007271E	リードバッファが有効な格納先を参照していません.
HR from WSAENOBUFS	0x80072747	操作を実行する為の十分なシステムバッファが不足しています.

注意:

パラメータ fpReadCompletedCallback は関数がアシンクロナスまたはシンクロナスで実行されるかどうか決定します。このパラメータが NULL ならシンクロナス、NULL でなければアシンクロナスで実行されます。アシンクロナス I/O 操作のあと、コールバックファンクションはコールと一緒に I/O 操作の結果を返します。関数をアシンクロナスでコールした場合 pReadBuffer は無視され結果は DLL の 格納先に入ります。

Modbus/TCP コマンド FC7 は exception status の読出しに使用されます。

2.3.8 MBTReadCompleted

WAGO-I/O-*PRO* 32 Elements of Dynamic Link Library		
種類:	このコールバック関数はアシンクロナス読出し関数が完了したあとに呼び出されます。	
関数名:	MBTReadCompleted	
タイプ:	Function	
DLL 名:	ModbusTCP.DLL	
適用:	Modbus/TCP-Protocol	
宣言:		

```
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);
```

パラメータ:	**コメント:**
hSocket	コネクションハンドラ
callbackContext	関数のアシンクロナスコール間の状態
errorCode	エラーコード
tableType	Modbus/TCP tables Typ (MOBUSTCP_TABLE_xxx)
dataStartAddress	読み出し開始位置
numRead	読み出し数
numBytes	読み出しバイト数
pReadBuffer	読み出し値の格納先

戻り値:	**HEX 値:**	**コメント:**
none		

注意:

2.3.9 MBTWriteRegisters

WAGO-I/O-*PRO* 32 Elements of Dynamic Link Library	
種類:	この関数はレジスタにデータを書込みます。
関数名:	MBTWriteRegisters
タイプ:	Function
DLL 名:	ModbusTCP.DLL
適用:	Modbus/TCP-Protocol

宣言:

```
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
```

パラメータ:	コメント:
hSocket	コネクションハンドラ
dataStartAddress	書き込み開始アドレス
numWords	書き込み数
pWriteBuffer	書き込み値の格納先
fpWriteCompletedCallback	アシンクロナスライトファンクションを完了したあとにコールされる C のコールバックファンクション シンクロナスコールの時は NULL にして下さい(デフォルト).
callbackContext	アシンクロナスコールバックファンクションの状態 シンクロナスコールの時は 0 にして下さい(デフォルト).

戻り値:	HEX 値:	コメント:
S_OK	0	正常終了.
MBT_HANDLE_INVALID_ERROR	0xEF010006	コネクションハンドラが無効です.
MBT_NO_JOB_ADDABLE_ERROR	0xEF010005	ライブラリのソケット管理エラーが発生しました.

MBT_EXIT_ERROR	0xEF01000B	MBTExit コールによりジョブが中断されました.
MBT_SOCKET_ TIMEOUT_ERROR	0xEF010008	I/O に対して Modbus/TCP コマンドを送ったがレスポンスが返らずリクエストタイムアウトに なりました ..
HR from WSANOTINITIALISED	0x8007276D	MBTInit がコールされていません .
HR from WSAENETDOWN	0x80072742	ネットワークエラーです.
HR from WSAENETRESET WSAECONNABORTED WSAEDISCON	0x80072744 0x80072745 0x80072746	デバイスとの接続はすでに中止されています.
HR from WSAEWOULDBLOCK	0x80072733	大量の I/O 操作が未処理になっています.
HR from WSAEFAULT	0x8007271E	ライトバッファが有効な格納先を参照していません.
HR from WSAENOBUFS	0x80072747	操作を実行する為の十分なシステムバッファが不足しています.

注意:

パラメータ fpWriteCompletedCallback は関数がアシンクロナスまたはシンクロナスで実行されるかどうか決定します。このパラメータが NULL ならシンクロナス、NULL でなければアシンクロナスで実行されます。アシンクロナス I/O 操作のあと、コールバックファンクションはコールと一緒に I/O 操作の結果を返します。関数をアシンクロナスでコールした場合 pReadBuffer は無視され結果は DLL の 格納先に入ります。

Modbus/TCP コマンド FC16 は output レジスタの書込みに使用されます。.

2.3.10 MBTWriteCoils

WAGO-I/O-*PRO* 32 Elements of Dynamic Link Library		
種類:	この関数はコイルにデータを書込みます	
関数名:	MBTWriteCoils	
タイプ:	Function	
DLL 名:	ModbusTCP.DLL	
適用:	Modbus/TCP-Protocol	

宣言:

```
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
```

パラメータ:	**コメント:**
hSocket	コネクションハンドラ
dataStartAddress	書き込み開始アドレス
numBits	書き込み数
pWriteBuffer	書き込み値の格納先
fpWriteCompletedCallback	アシンクロナスライトファンクションを完了したあとにコールされる C のコールバックファンクション シンクロナスコールの時は NULL にして下さい(デフォルト).
callbackContext	アシンクロナスコールバックファンクションの状態 シンクロナスコールの時は 0 にして下さい(デフォルト).

戻り値:	**HEX 値:**	**コメント:**
S_OK	0	正常終了.
MBT_HANDLE_INVALID_ERROR	0xEF010006	コネクションハンドラが無効です.
MBT_NO_JOB_ADDABLE_ERROR	0xEF010005	ライブラリのソケット管理エラーが発生しました.

MBT_EXIT_ERROR	0xEF01000B	MBTExit コールによりジョブが中断されました.
MBT_SOCKET_ TIMEOUT_ERROR	0xEF010008	I/O に対して Modbus/TCP コマンドを送ったがレスポンスが返らずリクエストタイムアウトに なりました ..
HR from WSANOTINITIALISED	0x8007276D	MBTInit がコールされていません .
HR from WSAENETDOWN	0x80072742	ネットワークエラーです.
HR from WSAENETRESET WSAECONNABORTED WSAEDISCON	0x80072744 0x80072745 0x80072746	デバイスとの接続はすでに中止されています.
HR from WSAEWOULDBLOCK	0x80072733	大量の I/O 操作が未処理になっています.
HR from WSAEFAULT	0x8007271E	ライトバッファが有効な格納先を参照していません
HR from WSAENOBUFS	0x80072747	操作を実行する為の十分なシステムバッファが不足しています.

注意:

パラメータ fpWriteCompletedCallback は関数がアシンクロナスまたはシンクロナスで実行されるかどうか決定します。このパラメータが NULL ならシンクロナス、NULL でなければアシンクロナスで実行されます。アシンクロナス I/O 操作のあと、コールバックファンクションはコールと一緒に I/O 操作の結果を返します。関数をアシンクロナスでコールした場合 pReadBuffer は無視され結果は DLL の 格納先に入ります。

Modbus/TCP コマンド FC15 は output レジスタの書込みに使用されます。

2.3.11 MBTWriteCompleted

WAGO-I/O-*PRO* 32 Elements of Dynamic Link Library		
種類:	このコールバック関数はアシンクロナス書込み関数が完了したあとに呼び出されます。	
関数名:	MBTWriteCompleted	
タイプ:	Function	
DLL 名:	ModbusTCP.DLL	
適用:	Modbus/TCP-Protocol	
宣言:		
C: void MBTWriteCompleted(IN HANDLE hSocket, IN DWORD callbackContext, IN LONG errorCode, IN BYTE tableType, IN WORD dataStartAddress, IN WORD numWrite, IN LPBYTE pWriteBuffer);		
パラメータ:	**コメント:**	
hSocket	コネクションハンドラ	
callbackContext	関数のアシンクロナスコール間の状態	
errorCode	エラーコード	
tableType	Modbus/TCP tables Typ (MOBUSTCP_TABLE_xxx)	
dataStartAddress	書込み開始位置	
numWrite	書込み数	
pWriteBuffer	書込み値の格納先	
戻り値:	**HEX 値:**	**コメント:**
none		
注意:		

2.3.12 MBTSwapWord

WAGO-I/O-*PRO* 32 Elements of Dynamic Link Library	
種類:	この関数はワードの上位バイトと下位バイトをスワップするものです。
関数名:	MBTSwapWord
タイプ:	Function
DLL 名:	ModbusTCP.DLL
適用:	Modbus/TCP-Protocol
宣言:	
C: WORD MBTSwapWord(const WORD wData); VB: Public Declare Function MBTSwapWord Lib "MBT" (ByVal wData As Integer) As Integer	
パラメータ:	**コメント:**
wData	ワードデータ
戻り値:	**コメント:**
	この関数はスワップした値を返します。
注意:	

2.3.13 MBTSwapDWord

WAGO-I/O-*PRO* 32 Elements of Dynamic Link Library	
種類:	この関数はダブルワードの１バイト目と４バイト目、２バイト目と３バイト目をスワップします。
関数名:	MBTSwapDWord
タイプ:	Function
DLL 名:	ModbusTCP.DLL
適用:	Modbus/TCP-Protocol
宣言:	
C: DWORD MBTSwapDWord(const DWORD dwData); VB: Public Declare Function MBTSwapDWord Lib "MBT" (ByVal dwData As Long) As Long	
パラメータ:	**コメント:**
dwData	ダブルワードデータ
戻り値:	**コメント:**
	この関数はスワップした値を返します。
注意:	

2.3.14 MODBUSTCP_TABLE_xxx

MODBUSTCP_TABLE_xxx については以下の表を参照してください。

	定義	値
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 プログラム例（VB,C）

2.4.1 VBA

2.4.1.1 解説

Visual BASIC のサンプルプログラムは 1 つのフォームと btnReadCoil, btnWriteCoil, btnReadRegister and btnWriteRegister のコマンドボタン と AdrCoil, edtReadCoil, edtWriteCoil, AdrRegister, edtReadRegister ,edtWriteRegister のテキストボックスで構成されています。

Modbus/TCP DLL の初期設定とノードへの接続はフォームのオープンによってセットアップされます。

アクティブにしたいアドレスはテキストボックス AdrCoil 及び AdrRegister に入れてください。
コマンドボタン btnReadCoil 及び btnReadRegister をクリックすると テキストボックス edtReadCoil 及び edtReadRegister に表示されます。

コマンドボタン btnWriteCoil 及び BtnWriteRegister をクリックするとテキストボックス edtWriteCoil 及び edtWriteRegister に書かれた値を出力します。
これらのイベントプロシージャはそれぞれのボタンクリックイベントに書いてあります。

フォームを閉じるとノードとの接続は切断され Modbus/TCP DLL も終了します。

2.4.1.2 定数と DLL 関数の宣言

定数

```
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 関数

```
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, By-
Val 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 ta-
bleType 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 callback-
Context 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 変数宣言と イベントプロシージャー

global variables and constants

```
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 解説

C の次のプログラム例では、DLL は MBTInit 関数 で初期化され接続は MBTConnect 関数でセットアップされます。

接続が成功した後、入出力を行う為の書込み読出し関数 MBTWriteRegisters 、MBTReadRegisters 、MBTWriteCoils 、MBTReadCoils および MBTReadExceptionStatus はアシンクロナスコールされます。その後で書込み読出し関数はシンクロナスコールされます。状態は各機能をコールした後モニターに表示されます。

プログラムの最後で MBTDisconnect によりノードとの接続を切断し MBTExit で DLL を終了します。

2.4.2.2 インターフェイス

```
#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 プログラム

```
#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( "press <RETURN> !\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( "\npress <RETURN> !\n" );
  scanf( "%c", &ch );

  return 0;
}
```

3 索引