WAGO I/O SYSTEM 750

Nutzung der Ethernet.lib

für Client/Server Verbindungen

Anwendungshinweis

A112200, Deutsch
Version 1.0.0

WAGO®
INNOVATIVE CONNECTIONS

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.

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

INHALTSVERZEICHNIS

1 Wichtige Erläuterungen

Um dem Anwender eine schnelle Installation und Inbetriebnahme der 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 Dokument, einschließlich aller darin befindlichen Abbildungen, ist urheberrechtlich geschützt. Jede Weiterverwendung dieses Dokumentes, 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 Dokument 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 Dokumentes 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 dem Dokument aufgezeigten 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 Gültigkeitsbereich

Dieser Anwendungshinweis basiert auf die genannte Hard- und Software der jeweiligen Hersteller sowie auf die zugehörige Dokumentation. Daher gilt dieser Anwendungshinweis nur für die beschriebene Installation.
Neue Hard- und Softwareversionen erfordern eventuell eine geänderte Handhabung.

Beachten Sie die ausführliche Beschreibung in den jeweiligen Handbüchern.

1.3 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.

2 Beschreibung

Im Rahmen dieser Application note wird der Einsatz der Ethernet.lib für den Wago Ethernet Controller 750-842 in Verbindung mit Client/Server Anwendungen erläutert. Dem Anwender steht damit die Möglichkeit offen, eigene Protokolle über Ethernet IP zu implementieren. Es wird sowohl der Verbindungsaufbau über TCP als auch der Datenaustausch über UDP erläutert.

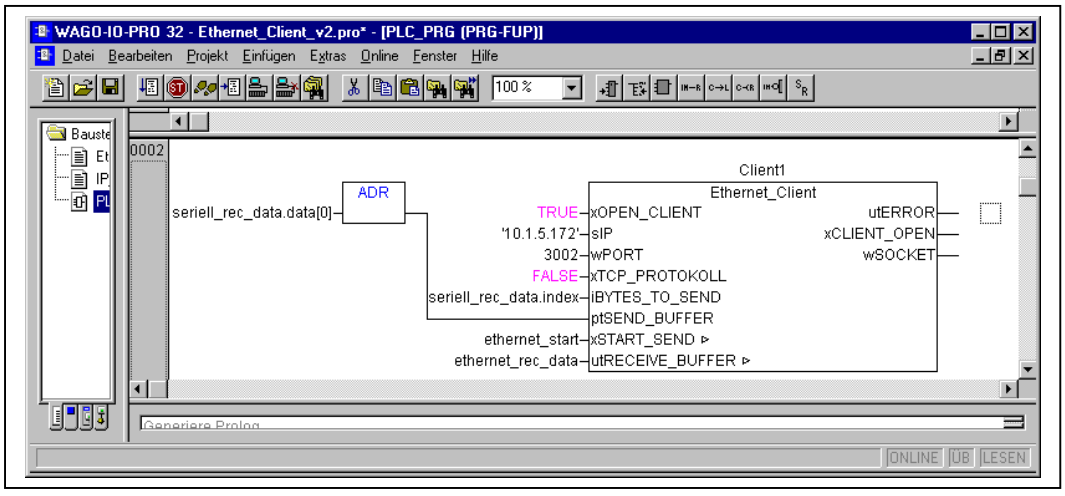
2.1 Material

Die Anbindung ist unter den folgenden Voraussetzungen getestet worden:

Bezeichnung	Hersteller	Type
Ethernet-Controller	WAGO	750-842 (02.04.00(06))
Ethernet.lib	WAGO	

3 Ethernet Client

Es wird die Programmierung eines Clients erläutert, der entweder eine TCP oder aber eine UDP Verbindung zu einem entsprechenden Server aufbauen kann. Als Datenübergabebereich sind zwei Datenfelder definiert. Im utRECEIVE_BUFFER werden die Daten, die empfangen wurden, gespeichert. In einem zweiten Bereich, auf den der Zeiger ptSEND_BUFFER zeigt, können die Daten eingegeben werden, die über das Ethernet Netzwerk verschickt werden sollen. Über den Eintrag iBytes_TO_SEND wird festgelegt, wie viele Bytes gesendet werden sollen. Der Sendevorgang wird über die boolsche Variable xSTART_SEND eingeleitet.



3.1 Die Client Zustandsmaschine

Intern wird der Client durch eine Zustandsmaschine realisiert.

3.1.1 Zustand 0:

Initialisierung diverser Variablen und Funktionsblöcke.

3.1.2 Zustand 1:

Prüfen ob der Ethernet Stack initialisiert ist.

```
1:(*check ethernet stack*)
getConfig(EN:=TRUE );
IF getConfig.ENO THEN
        IF getConfig.IP_ADR<>0 THEN
                status:=4;(*stack initialized*)
        ELSE
                status:=0;(*try again*)
        END_IF
END  IF
```

3.1.3 Zustand 4

Initialisierung der Funktionsbausteine MyEthernetRead, MyEthernetClose, MyEthernetOpen. Anschließend wird das Kommando zum Öffnen einer Verbindung abgesetzt.

```
4:(*open connection*)
MyEthernetRead(EN:= 0, SOCKET:=0 ,  DATA:=EthernetBuffer );
MyEthernetClose(EN:= 0);
MyEthernetOpen(EN:=0);


MyEthernetOpen(EN:=1 , TYP:=SOCK_STREAM , PROTO:=IPPROTO_TCP ,
                        IP_ADR:= IP_ADRESSE(sIP), PORT:=wPort );
status:=5;
```

3.1.4 Zustand 5

Warten ob sich eine Verbindung mit dem Server herstellen läßt. Schlägt dieser Versuch fehl, wird erneut probiert, eine Verbindung herzustellen.

```
5:(*open connection*)

MyEthernetOpen(EN:=1 , TYP:=SOCK_STREAM , PROTO:=IPPROTO_TCP ,

                        IP_ADR:= IP_ADRESSE(sIP), PORT:=wPort );

IF MyEthernetOpen.ENO = 1 THEN

        IF MyEthernetOpen.ERROR = 0 THEN

                Status  := 10; (*socket available*)

                xCLIENT_OPEN:=TRUE;

                wSOCKET     :=MyEthernetOpen.socket;

        ELSE

                Status := 0; (*no socket available, try it again *)

        END_IF;

        utERROR:=MyEthernetOpen.ERROR;

END_IF;
```

3.1.5 Zustand 10

Prüfen ob Daten empfangen wurden oder ein Sendeauftrag ansteht.

```
10:
MyEthernetRead(EN:= 1, SOCKET:=MyEthernetOpen.SOCKET ,
                        DATA:=EthernetBuffer );
MyEthernetRead(EN:= 0, SOCKET:=0 ,  DATA:=EthernetBuffer );
utERROR:=MyEthernetREAD.ERROR;


IF(MyEthernetRead.ERROR<>0) THEN
        Status:=40; (* Error reading the socket, close it *)
        ELSIF (MyEthernetRead.LEN_OUT <>0) THEN
                Status:=20; (* Process the data received via the ethernet port *)
                ELSIF (iBytes_TO_SEND > 0 AND xSTART_SEND) THEN
                        status:=30; (* Process the data received via the serial port *)
                        MyEthernetWrite(EN:= 0 , DATA:=EthernetBuffer );
END_IF
```

3.1.6 Zustand 20

Empfangene Daten in den Empfangspuffer kopieren.

```
20:
FOR i := 1 TO MyEthernetRead.LEN_OUT DO
        utReceive_buffer.index :=(utReceive_buffer.index MOD 1500)+1;
        utReceive_buffer.data[utReceive_buffer.index] := EthernetBuffer[i];
END_FOR
Status:=10;
```

3.1.7 Zustand 30

Daten über Ethernet senden.

```
30: (* Process the data in the send buffer *)
EthernetBuffer :=ptSEND_BUFFER^;
MyEthernetWrite(EN:= 1,SOCKET:= MyEthernetOpen.SOCKET,
                        LEN_IN:=iBYTES_TO_SEND,DATA:=EthernetBuffer );

IF MyEthernetWrite.eno THEN
    IF(MyEthernetWrite.ERROR <>0) THEN
        Status:=40;(* Error writing to the socket, close it *)
    ELSE
        Status:=10; (* Communication complete, read more data *)
    END_IF;
    xStart_send:=FALSE;
    utERROR:=MyEthernetWrite.ERROR;
END_IF
```

3.1.8 Zustand 40

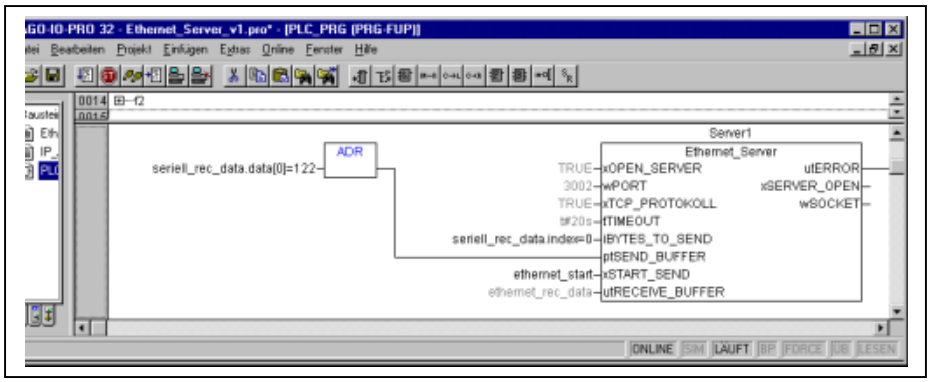
Verbindung schließen.

```
40:    (* Close the client *)
MyEthernetClose(EN:= 1, SOCKET:= MyEthernetOpen.SOCKET);
IF MyEthernetClose.eno then
    MyEthernetClose(EN:= 0);
    xCLIENT_OPEN:=FALSE;
    Status := 0; (* Try to open the client again *)
End  if
```

4 Ethernet Server

Es wird die Programmierung eines Servers erläutert, der entweder über eine TCP oder aber über eine UDP Verbindung mit einem entsprechenden Client kommuniziert. Als Datenübergabebereich sind zwei Datenfelder definiert. Im utRECEIVE_BUFFER werden die Daten, die Empfangen wurden ausgegeben. In einem zweiten Bereich, auf den der Zeiger ptSEND_BUFFER zeigt, können die Daten eingegeben werden, die über das Ethernet verschickt werden sollen. Über den Eintrag iBytes_TO_SEND wird festgelegt, wie viele Bytes gesendet werden sollen. Der Sendevorgang wird über die boolsche Variable xSTART_SEND gestartet.

Ferner verfügt der Baustein über eine Zeitüberwachung, die nach der eingestellten Zeit den Server kurzfristig schließt und anschließend sofort wieder öffnet. Durch diesen Mechanismus wird sichergestellt, daß im Falle eines gestörten Clients der Server nicht durch eine halboffenen Verbindung blockiert wird.



4.1 Die Server Zustandsmaschine

Intern wird der Server durch eine Zustandsmaschine realisiert.

4.1.1 Zustand 0

Initialisierung der Funktionsbausteine.

4.1.2 Zustand 1

Prüfen ob der Ethernet Stack initialisiert ist. Dazu wird der Funktionsblock Ethernet_Get_Network_Konfiguration benutzt. Wenn der Ethernet Stack initialisiert ist, liefert der Funktionsblock die aktuelle IP Adresse des Servers zurück.

```
1:(*check ethernet stack*)

getConfig(EN:=TRUE );

IF getConfig.ENO THEN

        IF getConfig.IP_ADR<>0 THEN

                status:=4;(*stack initialized*)

                MyEthernetOpen (EN:=0);

                MyEthernetRead(EN:= 0, SOCKET:=0 , DATA:=EthernetBuffer);

                MyEthernetClose(EN:= 0);

        ELSE

                status:=0;(*try again*)

        END_IF

END_IF
```

4.1.3 Zustand 4

Öffnen einer Verbindung entweder gemäß dem TCP oder als UDP Protokoll.

```
4:      (* try to open a socket*)
IF xTCP_PROTOKOLL THEN
        MyEthernetOpen(EN:=1 , TYP:=SOCK_STREAM,
                                PROTO := IPPROTO_TCP, PORT := wPort);
ELSE
        MyEthernetOpen(EN:=1 , TYP:=SOCK_DGRAM,
                                PROTO := IPPROTO_UDP, PORT := wPort);
END_IF


IF ((MyEthernetOpen.ERROR = 0) AND (MyEthernetOpen.ENO = 1)) THEN
        Status  := 10; (* socket created, wait for data to process *)
        MyEthernetOpen (EN:=0);
        xServer_open:=TRUE;
        wSocket:=MyEthernetOpen.socket;
ELSE
        Status := 0; (*socket not available, try it again *)
        MyEthernetOpen (EN:=0);
        xServer_open:=FALSE;
        wSocket:=0;
END_IF;
```

4.1.4 Zustand 10

Lesen der eingetroffenen Daten.

```
10:     (* Monitor ethernet port for new data *)
MyEthernetRead(EN:= 1, SOCKET:=MyEthernetOpen.SOCKET ,  DATA:=EthernetBuffer );
MyEthernetRead(      EN:= 0, SOCKET:=0 ,  DATA:=EthernetBuffer );


IF(MyEthernetRead.ERROR<>0) THEN
        Status:=40; (* Error reading the socket, close it *)
ELSIF (MyEthernetRead.LEN_OUT <>0) THEN
                Status:=20; (* Process the data received via the ethernet port *)
                client_SRC_IP:=MyEthernetRead.SRC_IP;
                client_SRC_PORT:=MyEthernetRead.SRC_PORT;
                watchdog_timer1(IN:=FALSE , PT:=tTIMEOUT );
                ELSIF (iBytes_TO_SEND > 0 AND xSTART_SEND) THEN
                        status:=30; (* Process the data received via the serial port *)
                        MyEthernetWrite(      EN:= 0 , DATA:=EthernetBuffer );
END_IF
```

4.1.5 Zustand 20

Empfangene Daten in den Receive_Buffer kopieren.

```
20:     (* Process the data received via the ethernet port *)
FOR i := 1 TO MyEthernetRead.LEN_OUT DO
        utReceive_buffer.index :=(utReceive_buffer.index MOD 1500)+1;
        utReceive_buffer.data[utReceive_buffer.index] := EthernetBuffer[i];
END_FOR
Status:=10;
```

4.1.6 Zustand 30

Daten über Ethernet senden.

```
30:     (*send data*)
EthernetBuffer :=ptSEND_BUFFER^;
IF xTCP_PROTOKOLL THEN
        MyEthernetWrite(    EN:= 1,SOCKET:= MyEthernetOpen.SOCKET,
                            LEN_IN:=iBYTES_TO_SEND, DATA:=EthernetBuffer );
ELSE
        IF client_SRC_IP>0 AND client_SRC_PORT>0 THEN
                MyEthernetWrite(EN:=1, SOCKET:=MyEthernetOpen.SOCKET
                                LEN_IN:=iBYTES_TO_SENd,ST_IP:=client_SRC_IP,
                                DST_PORT:=client_SRC_PORT,DATA:=EthernetBuffer );
        ELSE
                status:=10;
        END_IF
END_IF

IF MyEthernetWrite.eno THEN
        IF(MyEthernetWrite.ERROR <>0) THEN
                Status:=40;(* Error writing to the socket, close it *)
        ELSE
                Status:=10; (* Comms complete, read more data *)
        END_IF;
                xStart_send:=FALSE;
END_IF
```

4.1.7 Zustand 40

Schließen der Verbindung.

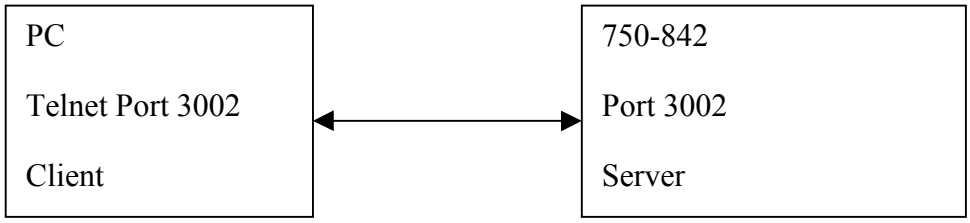
```
40:     (* Close the server *)
        MyEthernetClose(EN:= 1, SOCKET:= MyEthernetOpen.SOCKET);
        IF MyEthernetClose.eno then
                MyEthernetClose(EN:= 0);
                xServer_open:=FALSE;
                utERROR:=0;
                wSocket:=0;
                Status := 0; (* Try to open the server again *)
        End_if
```

5 Beispiele

Im folgenden werden zwei Beispiele erläutert, die den Einsatz der Client Server Funktionalität verdeutlichen. Die Beispiele greifen dabei auf die Bausteine Ethernet_Client bzw. Ethernet_Server zurück. Diese Bausteine stehen im Quellcode zur Verfügung und können ggf. eigenständig erweitert werden.

5.1 Ein Client in Verbindung mit einem 750-842 als Server

Auf einem PC steht das Programm Telnet zur Verfügung, welches als Client eingesetzt werden kann. Es wird eine TCP Verbindung auf Port 3002 eingerichtet.

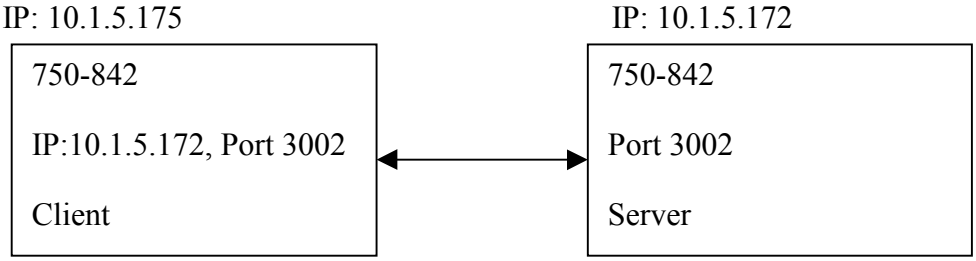


Wird über Telnet eine „1“ eingegeben, wird der erste digitale Ausgang eingeschaltet. Wird eine „0“ eingegeben, wird der Ausgang ausgeschaltet. Im Server ist ein kleines Programm hinterlegt, daß die Client Anforderungen ausführt und eine entsprechende Antwort an den Client meldet.

Beispielprogramm: EthernetServerExample.pro

5.2 Ein 750-842 Client in Verbindung mit einem 750-842 als Server

Es wird eine TCP Verbindung auf Port 3002 eingerichtet.



Es wird der Server aus Beispiel 4.2.1 benutzt. Für den Client wird in einem 750-842 Controller der Baustein Ethernet_Client aufgerufen. Ein weiterer Baustein stößt die Datenübertragung an. Wird der erste digitale Eingang betätigt, wird der erste digitale Ausgang auf dem Server eingeschaltet.

Beispielprogramm: EthernetClientExample.pro

WAGO Kontakttechnik GmbH
Postfach 2880 • D-32385 Minden
Hansastraße 27 • D-32423 Minden
Telefon: 05 71/8 87 – 0
Telefax: 05 71/8 87 – 1 69
E-Mail: info@wago.com

Internet: http://www.wago.com