ibaPDA-Interface-Modbus-TCP-Server

Datenschnittstelle Modbus-TCP-Server

Handbuch

Ausgabe 3.1

Messsysteme für Industrie und Energie

www.iba-ag.com

Hersteller

iba AG

Königswarterstraße 44

90762 Fürth

Deutschland

Kontakte

Zentrale	+49 911 97282-0
Support	+49 911 97282-14
Technik	+49 911 97282-13
E-Mail	iba@iba-ag.com
Web	www.iba-ag.com

Version	Datum	Revision	Autor	Version SW
3.1	10-2024	Neue Version ibaPDA v8	rm, nm	8.6.0

Inhalt

1 Zu dieser Dokumentation

Diese Dokumentation beschreibt die Funktion und Anwendung der Software-Schnittstelle *ibaPDA-Interface-Modbus-TCP-Server*.

Andere Dokumentation

Diese Dokumentation ist eine Ergänzung zum *ibaPDA*-Handbuch. Informationen über alle weiteren Eigenschaften und Funktionen von *ibaPDA* finden Sie im *ibaPDA*-Handbuch bzw. in der Online-Hilfe.

1.1 Zielgruppe und Vorkenntnisse

Diese Dokumentation wendet sich an ausgebildete Fachkräfte, die mit dem Umgang mit elektrischen und elektronischen Baugruppen sowie der Kommunikations- und Messtechnik vertraut sind. Als Fachkraft gilt, wer auf Grund der fachlichen Ausbildung, Kenntnisse und Erfahrungen sowie Kenntnis der einschlägigen Bestimmungen die übertragenen Arbeiten beurteilen und mögliche Gefahren erkennen kann.

Im Besonderen wendet sich diese Dokumentation an Personen, die mit Projektierung, Test, Inbetriebnahme oder Instandhaltung von Speicherprogrammierbaren Steuerungen der unterstützten Fabrikate befasst sind. Für den Umgang mit *ibaPDA-Interface-Modbus-TCP-Server* sind folgende Vorkenntnisse erforderlich bzw. hilfreich:

- Betriebssystem Windows
- Grundkenntnisse *ibaPDA*
- Kenntnis von Projektierung und Betrieb des betreffenden Messgeräts/-systems

1.2 Schreibweisen

In dieser Dokumentation werden folgende Schreibweisen verwendet:

Aktion	Schreibweise
Menübefehle	Menü *Funktionsplan*
Aufruf von Menübefehlen	*Schritt 1 – Schritt 2 – Schritt 3 – Schritt x* Beispiel: Wählen Sie Menü *Funktionsplan – Hinzufügen – Neuer Funktionsblock*
Tastaturtasten	<Tastenname> Beispiel: <Alt>; <F1>
Tastaturtasten gleichzeitig drücken	<Tastenname> + <Tastenname> Beispiel: <Alt> + <Strg>
Grafische Tasten (Buttons)	<Tastenname> Beispiel: <OK>; <Abbrechen>
Dateinamen, Pfade	`Dateiname`, `Pfad` Beispiel: `Test.docx`

1.3 Verwendete Symbole

Wenn in dieser Dokumentation Sicherheitshinweise oder andere Hinweise verwendet werden, dann bedeuten diese:

Gefahr!

Wenn Sie diesen Sicherheitshinweis nicht beachten, dann droht die unmittelbare Gefahr des Todes oder der schweren Körperverletzung!

- Beachten Sie die angegebenen Maßnahmen.

Warnung!

Wenn Sie diesen Sicherheitshinweis nicht beachten, dann droht die mögliche Gefahr des Todes oder schwerer Körperverletzung!

- Beachten Sie die angegebenen Maßnahmen.

Vorsicht!

Wenn Sie diesen Sicherheitshinweis nicht beachten, dann droht die mögliche Gefahr der Körperverletzung oder des Sachschadens!

- Beachten Sie die angegebenen Maßnahmen.

Hinweis

Hinweis, wenn es etwas Besonderes zu beachten gibt, wie z. B. Ausnahmen von der Regel usw.

Tipp

Tipp oder Beispiel als hilfreicher Hinweis oder Griff in die Trickkiste, um sich die Arbeit ein wenig zu erleichtern.

Andere Dokumentation

Verweis auf ergänzende Dokumentation oder weiterführende Literatur.

2 Systemvoraussetzungen

Folgende Systemvoraussetzungen sind für die Verwendung der Datenschnittstelle Modbus-TCP-Server erforderlich:

- *ibaPDA* v8.0.0 oder höher
- Lizenz für *ibaPDA-Interface-Modbus-TCP-Server*
- Netzwerkanschluss 10/100 Mbit

In der *ibaPDA*-Dokumentation finden Sie weitere Anforderungen an die Computer-Hardware und die unterstützten Betriebssysteme.

Hinweis

iba empfiehlt, die TCP/IP-Kommunikation auf einem separaten Netzwerksegment durchzuführen, um eine gegenseitige Beeinflussung durch sonstige Netzwerkkomponenten auszuschließen.

Systemeinschränkungen

- Die maximale Länge einer Modbus TCP/IP-Nachricht ist auf 244 Bytes beschränkt.
- Unterschiedliche Behandlung des TCP/IP-Acknowledge, siehe ↗ *Probleme mit TCP-Performance durch Delayed Acknowledge*, Seite 36

Lizenzen

Bestell-Nr.	Produktbezeichnung	Beschreibung
31.001020	ibaPDA-Interface-Modbus-TCP-Server	Erweiterungslizenz für ein *ibaPDA*-System um eine Modbus-TCP-Server Schnittstelle Anzahl der Verbindungen: 64
31.101020	one-step-up-Interface-Modbus over TCPIP-Server	Lizenz für die Erweiterung einer vorhandenen Schnittstelle um 64 weitere Modbus-TCP-Server-Verbindungen, maximal 3 Erweiterungslizenzen zulässig

3 Datenschnittstelle Modbus-TCP-Server

3.1 Allgemeine Informationen

3.1.1 Modbus TCP/IP

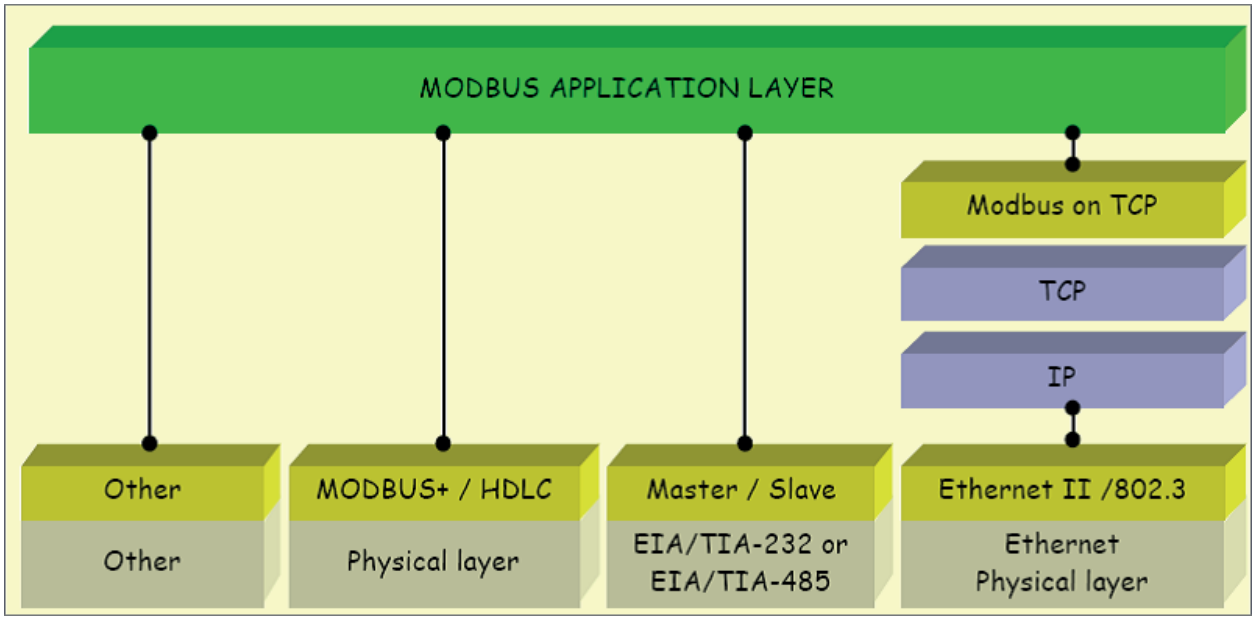
Das Transmission Control Protocol (TCP) ist eines der Kernprotokolle der Internetprotokoll-Familie.

IP arbeitet auf der unteren Vermittlungsschicht und ist für die Vermittlung von Nachrichten zwischen Rechnern im Internet zuständig. TCP ist auf einer höheren Schicht (Transportschicht) angeordnet und befasst sich mit den beiden Endsystemen. TCP sorgt für einen zuverlässigen Datenstrom von einem Programm auf einem Rechner zu einem anderen Programm auf einem anderen Rechner. TCP ist in RFC1180 und in RFC768 beschrieben.

Modbus ist ein Protokoll für eine Client/Server-Kommunikation zwischen Geräten an unterschiedlichen Bussen oder Netzwerken.

Modbus ist derzeit in den folgenden Bussen oder Netzwerken implementiert, wie in folgender Abbildung dargestellt:

- TCP/IP über Ethernet
- Asynchrone serielle Übertragung über unterschiedliche Medien
- Modbus PLUS (High-Speed-Kommunikation über ein Token-Passing-Netzwerk)



ibaPDA bietet die Möglichkeit, Signale mit dem Modbus-Protokoll über serielle Verbindungen (Modbus ASCII und Modbus RTU) und über TCP/IP zu messen. Diese Dokumentation beschreibt die Kommunikation über TCP/IP und als Variante die Übertragung des Modbus RTU Protokolls über TCP/IP, wobei *ibaPDA* als Client agiert.

Jedes System, das in der Lage ist, Nachrichten mit dem MODBUS-TCP-Protokoll als Server zu empfangen und zu beantworten, kann mit *ibaPDA* kommunizieren.

Andere Dokumentation

Verweis auf ergänzende Dokumentation oder weiterführende Literatur:

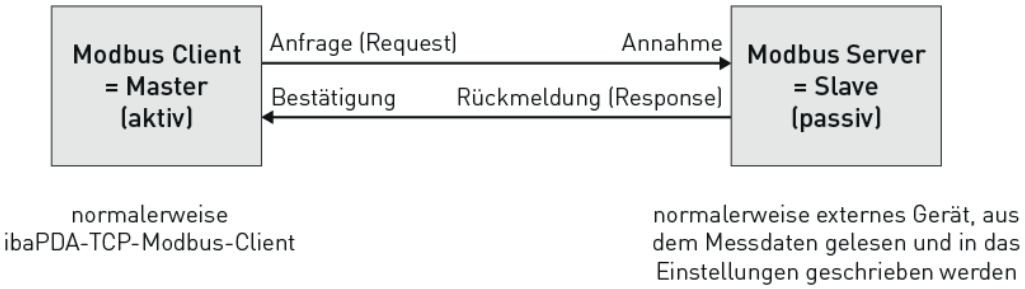
- *ibaPDA*-Handbuch (https://www.iba-ag.com/de/downloads)
- TCP/IP Tutorial, RFC1180 (https://www.rfc-editor.org/rfc/rfc1180)
- Transmision Control Protocol, RFC793 (https://www.rfc-editor.org/rfc/rfc793)
- Modbus Messaging Implementation Guide V1 (https://modbus.org)
- Modbus Application Protocol V1.1 (https://modbus.org)
- Modbus Protocol Reference Guide Rev J, Modicon

3.1.2 Client-Server-Architektur

Der Modbus-Dienst unterstützt eine Client-Server-Kommunikation für Geräte, die über Ethernet TCP/IP verbunden sind.

Das Client/Server-Modell basiert auf 4 Nachrichtentypen:

- Anfrage (Request)
- Annahme (Indication)
- Rückmeldung (Response)
- Bestätigung (Confirmation)



Daten lesen: Der Modbus-TCP-Client (*ibaPDA*) baut die Verbindung zu dem Modbus-Server auf, sendet periodisch die Anfrage und wartet auf die Rückmeldung, welche die angeforderten Daten enthält.

Daten schreiben: Der Modbus-TCP-Client (*ibaPDA*) baut die Verbindung zu dem Modbus-Server auf, sendet die Anfrage, welche die Ausgabedaten enthält, und wartet auf die Rückmeldung.

Standardmäßig wird für die Modbus-TCP/IP-Kommunikation der Port 502 verwendet, jedoch besteht in *ibaPDA*die Möglichkeit, andere Portnummern einzugeben.

Mit einer *ibaPDA-Interface-Modbus-TCP-Server*-Lizenz kann *ibaPDA* bis zu 64 Verbindungen empfangen, d. h. bis zu 64 Modbus-Server können Verbindungen zu *ibaPDA* aufbauen. Die Anzahl kann durch mehrfaches Laden der Lizenz auf max. 256 Verbindungen erweitert werden.

3.1.3 Modbus-Protokoll

Im Folgenden finden Sie Informationen zum Aufbau und Inhalt der Modbus-Protokolle.

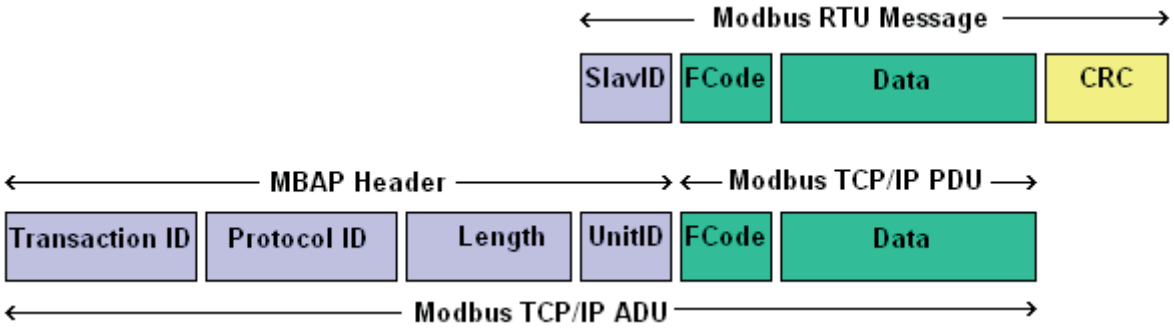
Byte-Reihenfolge

Modbus verwendet "BIG ENDIAN", d. h. in den Telegrammen werden die höherwertigen Bytes zuerst gesendet und folglich in den niederwertigen Adressen im Speicher abgelegt,

ibaPDA wandelt alle empfangenen 16- und 32-Bit-Werte in das Intel-Format "LITTLE ENDIAN" um ("Swapping"). Für Daten, die nicht von einer ursprünglichen Modbus-Steuerung kommen, bietet *ibaPDA* die Möglichkeit, die Swapping-Methode einzustellen. Siehe ↗ *Allgemeine Einstellungen der Schnittstelle*, Seite 17.

Modbus RTU / Modbus TCP

Die folgende Darstellung zeigt den grundsätzlichen Aufbau des Modbus-Protokolls und die Unterschiede zwischen Modbus RTU und Modbus TCP.



RTU	Remote Terminal Unit
MBAP	Modbus Application Protocol
ADU	Application Data Unit
PDU	Protocol Data Unit

Bei Modbus TCP wird dem Funktionscode der MBAP Header vorangestellt, die Unit-ID entspricht der Slave-ID des RTU-Protokolls und der CRC-Code entfällt.

MBAP Header

Der MBAP Header wird bei der Kommunikation mit TCP/IP verwendet, um die Modbus-Applikationsdaten zu identifizieren.

Der Header enthält folgende Felder:

Felder	Bytes	Beschreibung
Transaktions-ID	2	Kennzeichnung einer Request/Response-Aktion
Protokoll-ID	2	0 = Modbus-Protokoll
Länge	2	Anzahl der folgenden Bytes
Unit-ID	1	Adressierung eines Remote-Slaves der mit dem Modbus-Server verbunden ist.

- Transaktions-ID: Kennzeichnet die Zuordnung der Aktionen.
  Sie wird vom Modbus-Client in der Anfrage (request) gesendet, der Modbus-Server kopiert die Transaktions-ID in das Antwort-Telegramm (response).
  ibaPDA wertet dieses Feld als Sequenzzähler aus und erwartet, dass bei jedem Zyklus der Wert um 1 inkrementiert wird. Bei Überlauf muss der Wert von 32767 auf -32768 (0x7FFF → 0x8000) bzw. von 65535 auf 0 (0xFFFF → 0x0000) springen.
- Protokoll-ID:
  Wird bei Multiplex-Verfahren verwendet, das Modbus-Protokoll hat den Wert 0.
- Länge:
  Gibt die Anzahl der folgenden Bytes wieder, inklusive Unit-ID, Funktionscode und Datenfelder. Maximalwert ist 251 (max. Länge der Nutzdatenbytes 244 + 7).
- Unit-ID (Geräteadresse):
  Das Feld wird vom Modbus-Client in der Anfrage (request) gesendet und vom Server mit demselben Wert in seiner Antwort zurückgeschickt.
  Dieses Feld wird von *ibaPDA* nicht ausgewertet.

Funktionscode

Ein Byte enthält den Funktionscode, der festlegt, welche Funktion aufgrund eines Requests durch den Server ausgeführt werden soll.

Der *ibaPDA-Interface-Modbus-TCP-Server*-Treiber unterstützt nur die Funktion

- 0x10: Write Multiple Registers

Datenfelder

Der Nutzdatenbereich ist in mehrere Unterbereiche unterteilt, wie Startadresse, Anzahl der Register, Anzahl der Bytes und die aktuellen Daten. Der Inhalt dieser Felder hängt vom verwendeten Funktionscode ab. Bei Funktionscode 0x10 enthalten die Datenfelder folgende Werte:

Felder	Bytes	Beschreibung
Startadresse	2	Startadresse des verwendeten Speicherbereichs
Anzahl der Objekte	2	Anzahl der verwendeten Register oder Coils
Anzahl der Bytes	1	Anzahl der Daten-Bytes
Datenbereich	n	n Daten-Bytes

- Startadresse
  Der *ibaPDA-Interface-Modbus-TCP-Server*-Treiber verwendet die Modbus-Startadresse zur Adressierung eines Datenmoduls. Die Startadresse, in *ibaPDA* als "Modulindex" bezeichnet, ist eine Nummer, in dem die Daten einem Datenmodul zugeordnet werden.

  In *ibaPDA* sind 4 Modultypen definiert:

  - Integer: 32 Analogwerte (Integer) und 32 Binärsignale
  - Real: 32 Analogwerte (Real) und 32 Binärsignale
  - Allgemein: beliebige Datenstruktur mit maximaler Länge von 244 Bytes
  - Dig512: 512 Binärsignale (32 Statusworte mit je 16 Bits)

Der Modulindex wird durch eine laufende Nummer 00….63 und einem dem Modultyp und der Lizenz entsprechenden Offset gebildet.

Modultyp	1. Lizenz	2. Lizenz	3. Lizenz	4. Lizenz
Integer und Dig512	0-63	1000-1063	2000-2063	3000-3063
Real	100-163	1100-1163	2100-2163	3100-3163
Allgemein	200-263	1200-1263	2200-2263	3200-3263

Der Modulindex entspricht dem Index in der *ibaPDA*-Moduleinstellung. Der Wert muss eindeutig sein und darf während der Datenübertragung nicht verändert werden.

- Anzahl der Objekte: Dieses Feld enthält die Anzahl der Register, die in der Nachricht übertragen werden. Mit *Modbus Integer* und *Modbus Dig512* werden 34 Register gesendet. Mit *Modbus Real* werden 66 Register gesendet. Im Modul *Allgemein* ist die Anzahl der Register variabel, aber auf maximal 122 begrenzt. In diesem Fall muss die Anzahl der Register, die gesendet werden, eingefügt werden.
- Anzahl der Bytes: Dieser Wert ist immer die Anzahl der Register multipliziert mit 2, da die Register Word-basiert sind (2 Bytes). Maximale Anzahl ist 244.
- Datenbereich: Das Feld enthält die aktuellen Daten, die an *ibaPDA* gesendet werden. Der Datentyp hängt vom verwendeten Modbus-Modul ab. Jeder Modultyp ist beschränkt auf eine feste Anzahl von Werten, die gesendet werden können, mit Ausnahme des Allgemein-Moduls. Im Allgemein-Modul können alle verfügbaren Datentypen nebeneinander verwendet werden. In *ibaPDA* muss nur die Adresse des Signals und der verwendete Datentyp konfiguriert werden.

3.1.4 Modbus TCP/IP – Telegrammaufbau

Die Modbus-Telegramme haben folgenden, dem Modultyp entsprechenden Telegrammaufbau.

3.1.4.1 Modbus Integer und Modbus Dig512

Bei Modultyp *Modbus Integer* sind die 32 analogen Werte vom Typ Integer (16-Bit) und die 32 digitalen Werte sind dicht gepackt als DWORD.

Bei Modultyp *Modbus Dig512* werden die 32 Analogwerte als 16-Bit-Statusworte ausgewertet, das DWORD wird nicht verwendet.

Request Modbus Client → ibaPDA (Modbus Server)

	Offs	Bytes	Typ	Modbus Beschreibung	Inhalte (hex)	Anmerkung
	00	2	UINT	Transaktions-ID	xx xx	Wird von *ibaPDA* als Sequenzzähler ausgewertet, d. h. der ID muss in jedem Zyklus inkrementiert werden
MBAP	02	2	UINT	Protokoll-ID	00 00	0
	04	2	UINT	Cmd Länge	4B	75
	06	1	BYTE	Unit-ID	xx	nicht verwendet
Fcode	07	1	BYTE	Funktionscode	10	"Write Multiple Registers"
Daten	08	2	UINT	Startadresse	xx xx	Modulindex i000 bis i063
	10	2	UINT	Anzahl Objekte	22	34: Anzahl Register
	12	1	BYTE	Anzahl Bytes	44	68: Anzahl Bytes
Nutzdaten	13	64	INT	Daten	xx	32 Analogwerte
	77	4	DWORD	Daten	xx	32 Digitalwerte

3.1.4.2 Modbus Real

Die analogen Werte sind vom Typ FLOAT (IEEE Format) und die 32 digitalen Werte sind dicht gepackt als DWORD.

Request Modbus Client → ibaPDA (Modbus Server)

	Offs	Bytes	Typ	Modbus Beschreibung	Inhalte (hex)	Anmerkung
MBAP	00	2	UINT	Transaktions-ID	xx xx	Wird von *ibaPDA* als Sequenzzähler ausgewertet
	02	2	UINT	Protokoll-ID	00 00	0
	04	2	UINT	Cmd Länge	8B	139
	06	1	BYTE	Unit-ID	xx	nicht verwendet
Fcode	07	1	BYTE	Funktionscode	10	"Write Multiple Registers"
Daten	08	2	UINT	Startadresse	xx xx	Modulindex i100 bis i163
	10	2	UINT	Anzahl Objekte	42	66: Anzahl Register
	12	1	BYTE	Anzahl Bytes	84	132: Anzahl Bytes
Nutzdaten	13	128	FLOAT	Daten	xx	32 Analogwerte
	141	4	DWORD	Daten	xx	32 Digitalwerte

3.1.4.3 Modbus Allgemein

Der Nutzdatenbereich kann eine beliebige Datenstruktur mit unterschiedlichen Datentypen enthalten. Von *ibaPDA* werden folgende Datentypen unterstützt:

BYTE, WORD, DWORD, INT, DINT und FLOAT

In *ibaPDA* muss die hier definierte Datenstruktur nachgebildet werden. Dabei können die BYTE-, WORD- und DWORD-Variablen auch als 8, 16 oder 32 einzelne Bits interpretiert werden (und umgekehrt).

Request Modbus Client → ibaPDA (Modbus Server)

	Offs	Bytes	Typ	Modbus Beschreibung	Inhalte (hex)	Anmerkung
MBAP	00	2	UINT	Transaktions-ID	xx xx	Wird von *ibaPDA* als Sequenzzähler ausgewertet
	02	2	UINT	Protokoll-ID	00 00	0
	04	2	UINT	Cmd Länge	xx	n + 7
	06	1	BYTE	Unit-ID	xx	nicht verwendet
Fcode	07	1	BYTE	Funktionscode	10	"Write Multiple Registers"
Daten	08	2	UINT	Startadresse	xx xx	Modulindex i200 bis i263
	10	2	UINT	Anzahl Objekte	42	n/2: aufgerundet
	12	1	BYTE	Anzahl Bytes	84	n (maximal 122)
Nutzdaten	13	nn	nnnn	Daten	xx	beliebige Datenstruktur (max. 244 Bytes)

3.1.4.4 Response

Standardmäßig wird jedes Telegramm vom Modbus-Server mit einem Response-Telegramm beantwortet. In *ibaPDA* kann dieses unterdrückt werden.

Response ibaPDA (Modbus-Server) → Modbus Client:

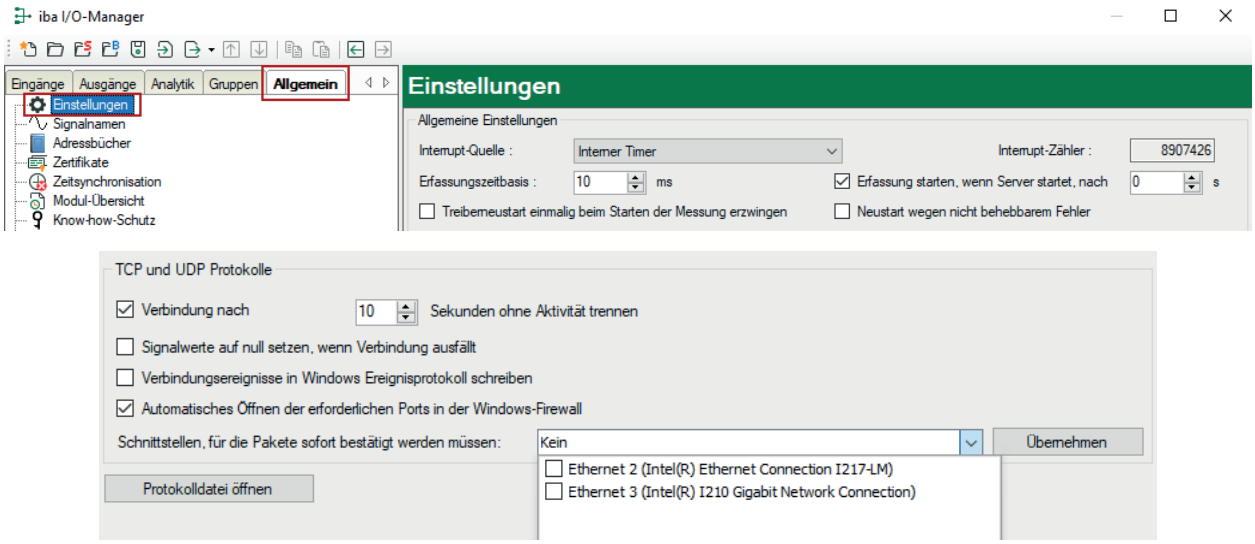
	Offs	Bytes	Typ	Modbus Beschreibung	Inhalt (hex)	ibaPDA Beschreibung
MBAP	00	2	UINT	Transaktions-ID	xx xx	Sequenzzähler, Spiegel der Anfrage
	02	2	UINT	Protokoll-ID	00 00	0
	04	2	UINT	Cmd Länge	00 06	6
	06	1	BYTE	Unit-ID	xx	Spiegel der Anfrage
Fcode	07	1	BYTE	Funktionscode	10	Spiegel der Anfrage
Daten	08	2	UINT	Startadresse	xx	Spiegel der Anfrage
	10	2	UINT	Anzahl Objekte	xx	Spiegel der Anfrage

3.2 Konfiguration und Projektierung ibaPDA

Nachfolgend ist die Projektierung in *ibaPDA* beschrieben. Wenn alle Systemvoraussetzungen erfüllt sind, bietet *ibaPDA* im Schnittstellenbaum des I/O-Managers die Schnittstelle *Modbus TCP Server* an.

3.2.1 Allgemeine Einstellungen

Das "Totmann-Timeout" konfigurieren Sie für alle von *ibaPDA* unterstützen TCP- und UDP-Protokolle gemeinsam.



Verbindung nach … Sekunden ohne Aktivität trennen
Verhalten und Timeout-Zeit ist vorgebbar.

Signalwerte auf null setzen, wenn Verbindung ausfällt
Wenn deaktiviert, bleibt der zuletzt gelesene Wert erhalten.

Verbindungsereignisse in Windows Ereignisprotokoll schreiben
Aktuelle Ereignisse werden in Windows protokolliert.

Automatisches Öffnen der erforderlichen Ports in der Windows-Firewall
Wenn aktiviert, werden vom *ibaPDA*-Server-Dienst alle Ports, die für die aktuell lizenzierten Schnittstellen benötigt werden, automatisch in der Firewall freigeschaltet.

Wenn deaktiviert, können die benötigten Ports im I/O-Manager der lizenzierten Schnittstellen über <Port in Firewall zulassen> freigeschaltet werden.

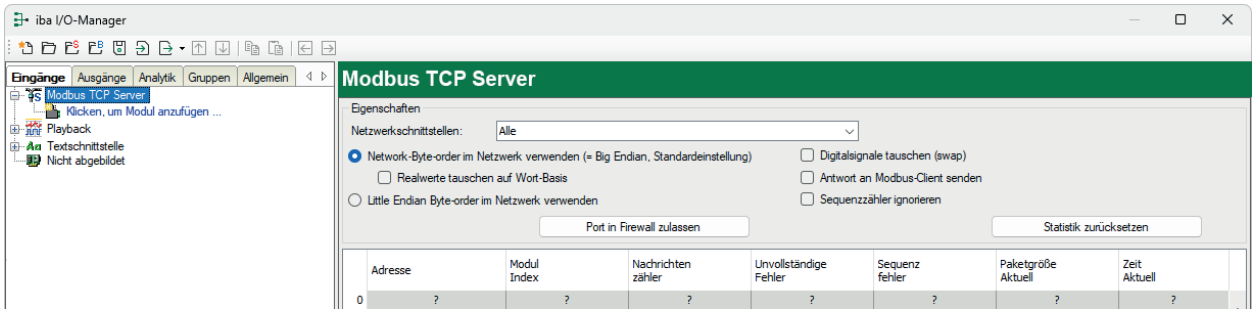
Schnittstellen, für die Pakete sofort bestätigt werden müssen
Auswahl der erforderlichen Schnittstellen

Hinweis

Ist *ibaPDA*der aktive Partner (Client), baut *ibaPDA*nach wenigen Sekunden die Verbindung wieder auf, um dem passiven Partner die Möglichkeit zu geben, wieder Daten zu senden.

3.2.2 Allgemeine Einstellungen der Schnittstelle

Die Schnittstelle selbst hat folgende Funktionen und Konfigurationsmöglichkeiten:



Netzwerkschnittstellen
Mit dieser Drop-down-Liste können Sie bestimmen, welcher Netzwerkadapter des betreffenden Rechners für diese Schnittstelle verwendet wird. Nur auf den ausgewählten Netzwerkadaptern werden die Ports für die Kommunikation geöffnet. Falls auf einem Netzwerkadapter mehrere IP-Adressen konfiguriert sind, wird für all diese IP-Adressen ein Socket geöffnet. Damit die Schnittstellenkonfiguration validiert werden kann, muss mindestens ein Netzadapter ausgewählt sein. Wenn Sie die Auswahl *kein* treffen, wird bei der Validierung der I/O-Konfiguration eine Fehlermeldung angezeigt. Standardmäßig sind alle Netzwerkadapter ausgewählt.

Netzwerk-Byte-Order im Netzwerk verwenden (= Big Endian, Standardeinstellung)
Alternative:

Little Endian Byte-Order im Netzwerk verwenden
siehe ↗ *Modbus-Protokoll*, Seite 11.

Realwerte tauschen auf Wort-Basis
Für Daten, die nicht von original Modbus-Geräten kommen, kann hier die Byte-Reihenfolge geändert werden. Die Bytes werden nach dem Muster ABCD → CDAB getauscht. Diese Option betrifft nur Werte im Modultyp Real, für den Modultyp Allgemein sind spezifische Einstellmöglichkeiten definiert.

Digitalsignale tauschen (swap)
Für Daten, die nicht von original Modbus-Geräten kommen, kann hier die Byte-Reihenfolge geändert werden. Die Bytes werden nach dem Muster ABCD → BADC getauscht.

Antwort an Modbus-Client senden
Jedes Telegramm wird mit einem Antworttelegramm bestätigt, siehe ↗ *Response*, Seite 15.
Durch Deaktivieren dieser Option kann das Antworttelegramm unterdrückt werden.

Sequenzzähler ignorieren
Wenn aktiviert, wird die Spalte *Sequenzfehler* in der Verbindungtabelle ausgeblendet.

<Ports in Firewall zulassen>
Bei der Installation von *ibaPDA* werden die Standard-Portnummern der verwendeten Protokolle automatisch in der Firewall eingetragen. Wenn Sie die Portnummer hier verändern oder das Interface nachträglich freischalten, müssen Sie über diesen Button diesen Port in der Firewall zulassen.

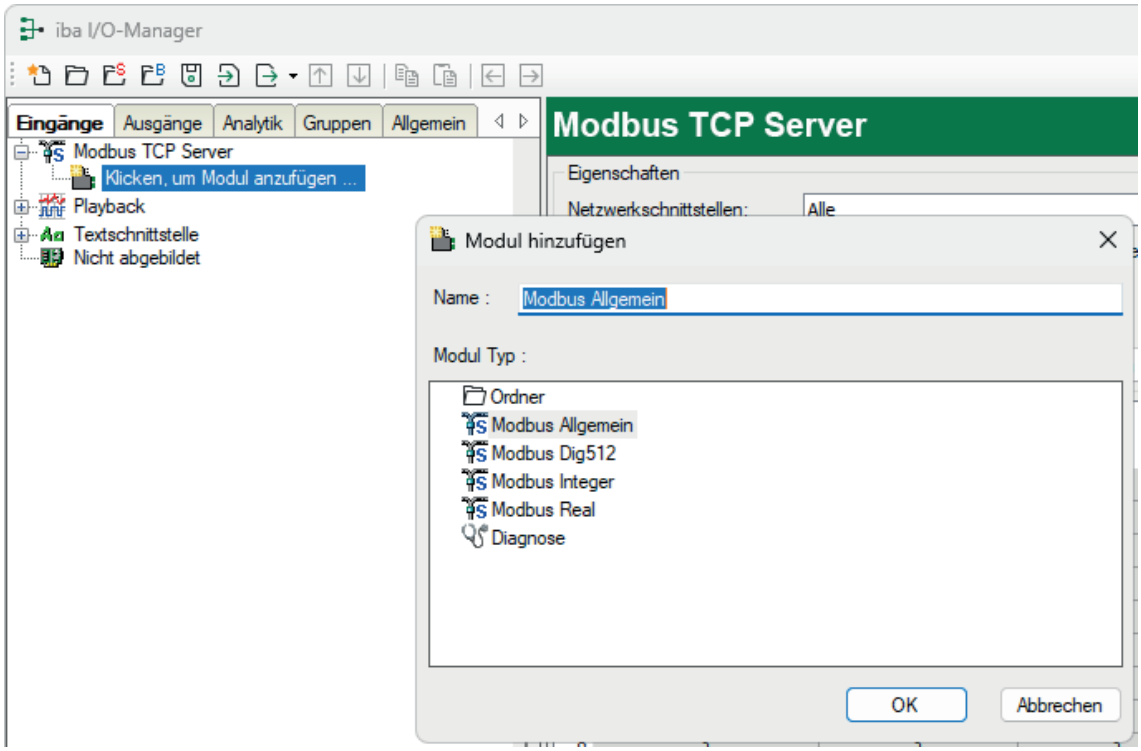
<Statistik zurücksetzen>
Über diesen Button können Sie die berechneten Zeitwerte und den Fehlerzähler in der Tabelle auf 0 setzen.

Verbindungstabelle
Sobald eine Verbindung aufgebaut ist, werden Live-Daten in der Tabelle angezeigt werden. Siehe ↗ *Überprüfung der Verbindung*, Seite 28.

3.2.3 Modul hinzufügen

Vorgehen

1. Klicken Sie auf den blauen Link *Klicken, um Modul anzufügen*, der sich unter jeder Datenschnittstelle im Register *Eingänge* oder *Ausgänge* befindet.
2. Wählen Sie im Dialogfenster den gewünschten Modultyp aus und vergeben Sie bei Bedarf einen Namen über das Eingabefeld.
3. Bestätigen Sie Ihre Auswahl mit <OK>.



Tipp

Wenn bereits TCP/IP-Verbindungen von Modbus-Clients bestehen, klicken Sie mit der rechten Maustaste auf das Interface und wählen Sie *Autom. Erkennung*. Dann werden für alle vorhandenen Verbindungen automatisch die richtigen Module angelegt.

Modultypen
Folgenden Modultypen stehen zur Auswahl:

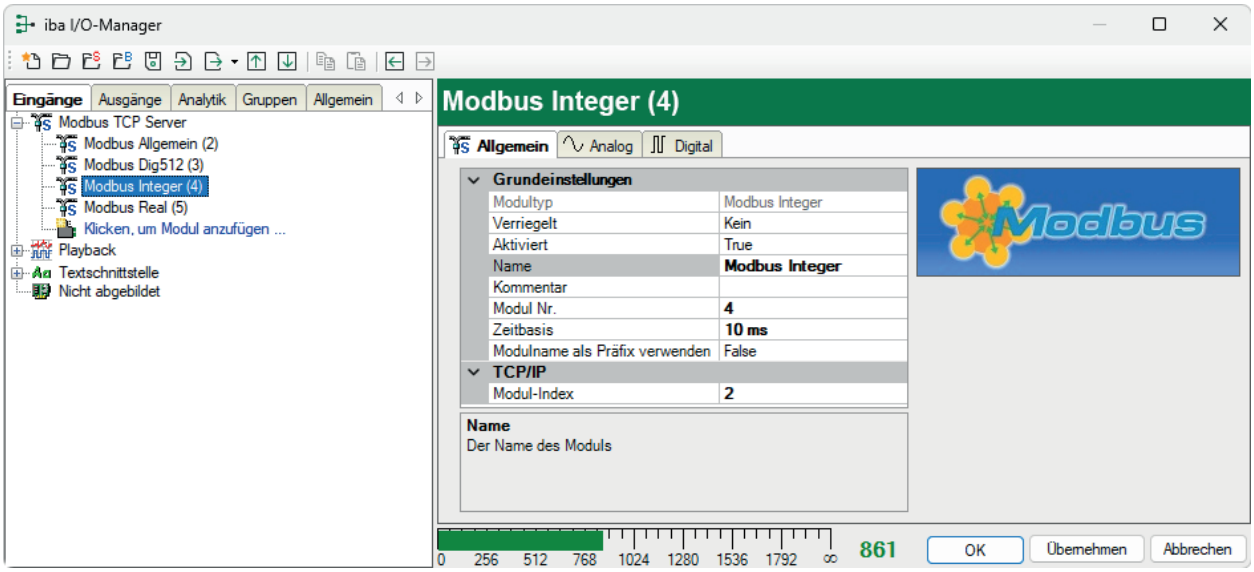
- Allgemein
- Dig512
- Integer
- Real

Weiterführende Informationen finden Sie in den jeweiligen Kapiteln unter ↗ *Allgemeine Informationen*, Seite 9 und ↗ *Konfiguration und Projektierung ibaPDA*, Seite 16

3.2.3.1 Allgemeine Moduleinstellungen

Um ein Modul zu konfigurieren, markieren Sie es in der Baumstruktur.

Alle Module haben die folgenden Einstellmöglichkeiten.



Grundeinstellungen

Modultyp (nur Anzeige)
Zeigt den Typ des aktuellen Moduls an.

Verriegelt
Sie können ein Modul verriegeln, um ein versehentliches oder unautorisiertes Ändern der Einstellungen zu verhindern.

Aktiviert
Aktivieren Sie das Modul, um Signale aufzuzeichnen.

Name
Hier können Sie einen Namen für das Modul eintragen.

Kommentar
Hier können Sie einen Kommentar oder eine Beschreibung zum Modul eintragen. Dies wird dann als Tooltip im Signalbaum angezeigt.

Modul Nr.
Diese interne Referenznummer des Moduls bestimmt die Reihenfolge der Module im Signalbaum von *ibaPDA*-Client und *ibaAnalyzer*.

Zeitbasis
Alle Signale dieses Moduls werden mit dieser Zeitbasis erfasst.

Modulname als Präfix verwenden
Diese Option stellt den Modulnamen den Signalnamen voran.

TCP/IP

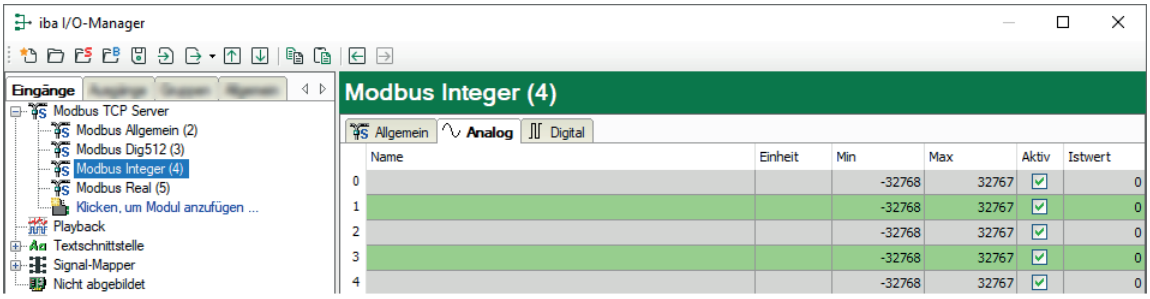
Modul-Index
Die Modulindizes werden durch eine laufende Nummer 00…63 und einem dem Modultyp und der Lizenz entsprechenden Offset gebildet, siehe ↗ *Modbus-Protokoll*, Seite 11

Für eine detaillierte Beschreibung der Parameter wird auf das*ibaPDA*-Handbuch verwiesen.

3.2.3.2 Generelle Signalkonfiguration

Die Auswahl der zu messenden Daten erfolgt auf der Modbus-Client-Seite durch Rangieren der Daten in die entsprechenden Datenbereiche.

Je nach Modultyp können Sie die Signale mit Namen, Einheit, Skalierungsfaktor und Kommentaren versehen, sowie aktiv oder inaktiv setzen.



Name
Eingabe eines aussagekräftigen Klartextnamens als Signalbezeichnung

Einheit (nur Analogsignale)
Eingabe einer physikalischen Einheit für das Signal

Sie können maximal 11 Zeichen eingeben, das Feld wird nur als Kommentarfeld betrachtet. Die Einheit erscheint immer in Verbindung mit einer numerischen Anzeige der Werte.

Gain, Offset (nur Analogsignale)
Angabe von Gain (Verstärkung) und Offset (Signalwert im Nullpunkt) zur Skalierung der eingehenden Werte

Diese Werte beschreiben eine lineare Kennlinie für die Skalierung zur Umrechnung in physikalische Einheiten. Wenn eingehende Werte in physikalischen Einheiten angegeben werden, können Sie diese Funktion ignorieren, also Gain = 1 und Offset = 0 setzen.

Aktiv
Aktivierung oder Deaktivierung des jeweiligen Signals

Istwert
Anzeige des aktuellen Istwerts des Signals

Andere Dokumentation

Ausführliche Beschreibungen zu den Spalten und zum Ausfüllen der Signaltabellen finden Sie in der Dokumentation zu *ibaPDA*.

Hinweis

Der Modultyp *TDC TCP/UDP Allgemein* unterstützt auch die Erfassung und Verarbeitung von Texten. Hierzu kann im Register *Analog* der Datentyp STRING[32] ausgewählt werden. Zur Wandlung eines Textsignals bzw. Unterteilung in mehrere Textsignale verwenden Sie ein *Texttrenner*-Modul unter der Schnittstelle *Virtuell*.

3.2.3.3 Modultyp Integer

Mit dem Modultyp *Integer* können Sie bis zu 32 Analogwerte (Integer) und 32 Binärsignale erfassen.

Das Modul hat keine modulspezifischen Einstellungen, für allgemeine Einstellungen, siehe ↗ *Generelle Signalkonfiguration*, Seite 21.

3.2.3.4 Modultyp Dig512

Mit dem Modultyp *Dig512* können Sie bis zu 512 Digitalwerte erfassen, organisiert als 32 Statusworte (Typ Integer) zu je 16 Bits.

Das Modul hat keine modulspezifischen Einstellungen, es gibt nur die Register *Allgemein* und *Digital*. Für Informationen zu den allgemeinen Einstellungen, siehe ↗ *Generelle Signalkonfiguration*, Seite 21.

3.2.3.5 Modultyp Real

Mit dem Modultyp *Real* können Sie bis zu 32 Analogwerte (Real) und 32 Binärsignale erfassen.

Modulspezifische Einstellungen

- Anzahl Analogsignale: Die zu erfassende Anzahl der Analogsignale ist einstellbar (1 bis 32).

  Achten Sie auf Folgendes:
  Wenn Sie die Digitalsignale verwenden wollen, muss das Telegramm den in ↗ *Modbus Real*, Seite 14, beschriebenen Aufbau haben. Wenn Sie keine Digitalsignale verwenden, kann das Telegramm entsprechend gekürzt werden.

- In der Analogtabelle wird die Skalierung mit *Gain*/*Offset* durchgeführt

3.2.3.6 Modultyp Allgemein

Mit dem Modultyp *Allgemein* können Sie beliebige Datenstrukturen mit maximaler Länge von 244 Bytes erfassen.

Modulspezifische Einstellungen

- Analogsignale swappen, Digitalsignale swappen

  Sie können die Auswertereihenfolge der Byte ändern. (Die Einstellungen unter der Schnittstelle gelten hier nicht!)

- Anzahl der Analogsignale, Anzahl der Digitalsignale

  Anzahl der maximal konfigurierbaren Analogsignale und Digitalsignale.

- In der Analogtabelle wird die Skalierung mit *Gain*/*Offset* durchgeführt
- Für jede Variable müssen Sie die Adresse, d. h. den Offset im Telegrammpuffer, sowie den Datentyp eintragen. Achten Sie darauf, dass hier ab Nutzdatenanfang ohne Header gezählt wird.

Hinweis

Der Modultyp *Modbus Allgemein* unterstützt auch die Erfassung und Verarbeitung von Texten. Hierzu kann im Register *Analog* der Datentyp STRING[32] ausgewählt werden. Zur Wandlung eines Textsignals bzw. Unterteilung in mehrere Textsignale verwenden Sie ein *Texttrenner*-Modul unter der Schnittstelle *Virtuell*.

Beschribung der Spalten

Name, Einheit, Gain, Offset, Aktiv
siehe ↗ *Generelle Signalkonfiguration*, Seite 21

Adresse
Stelle im Nutzdatenbereich eines Telegramms oder Dual-Port-Speichers (angegeben in Bytes), an der das gewünschte Signal liegt

Der Adressraum hängt vom Datentyp der vorausgehenden Daten ab. Nach dem Ändern von Datentypen kann so eine Anpassung der Adresseinträge erforderlich sein.

Datentyp (nur Analogsignale)
Auswahl des Datentyps des Signals

Der Datentyp bestimmt jeweils die Adresse des nächsten Signals.

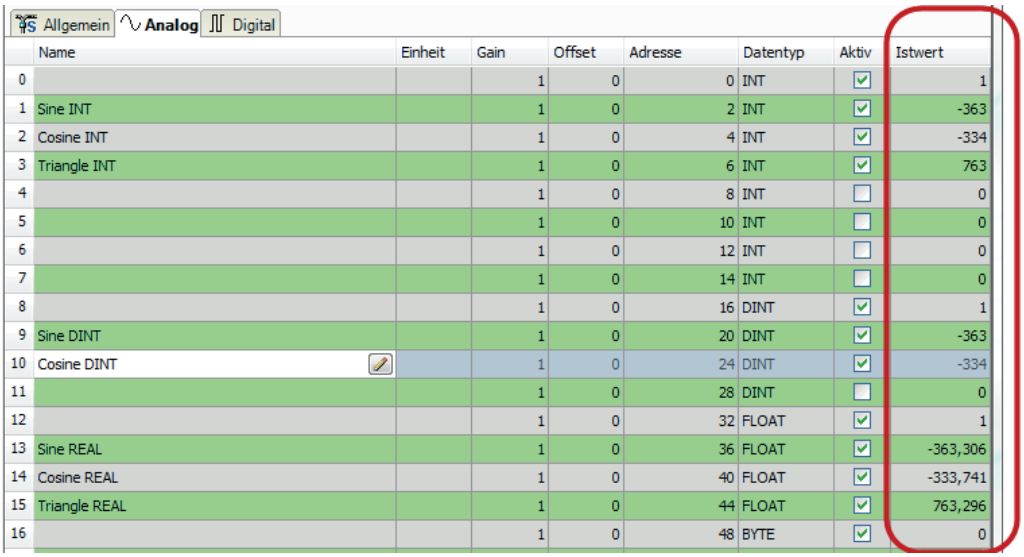
Die folgenden Datentypen werden unterstützt: SINT, BYTE, INT, WORD, DWORD, DINT, FLOAT, DOUBLE, STRING[32].

Tipp

Achten Sie darauf, dass die Adresse abhängig von den Datentypen der vorausgehenden Daten ist. Deshalb empfiehlt es sich, zuerst die Datentypen, dann die Adressen unter Verwendung der Ausfüll-Automatik einzustellen.

3.2.4 Moduldiagnose

In den Registern *Analog* und *Digital* der Module werden die Inhalte der Telegramme angezeigt.



	Name	Einheit	Gain	Offset	Adresse	Datentyp	Aktiv	Istwert
0			1	0	0	INT	☑	1
1	Sine INT		1	0	2	INT	☑	-363
2	Cosine INT		1	0	4	INT	☑	-334
3	Triangle INT		1	0	6	INT	☑	763
4			1	0	8	INT	☐	0
5			1	0	10	INT	☐	0
6			1	0	12	INT	☐	0
7			1	0	14	INT	☐	0
8			1	0	16	DINT	☑	1
9	Sine DINT		1	0	20	DINT	☑	-363
10	Cosine DINT		1	0	24	DINT	☑	-334
11			1	0	28	DINT	☐	0
12			1	0	32	FLOAT	☑	1
13	Sine REAL		1	0	36	FLOAT	☑	-363,306
14	Cosine REAL		1	0	40	FLOAT	☑	-333,741
15	Triangle REAL		1	0	44	FLOAT	☑	763,296
16			1	0	48	BYTE	☑	0

Folgende Fehler können auftreten:

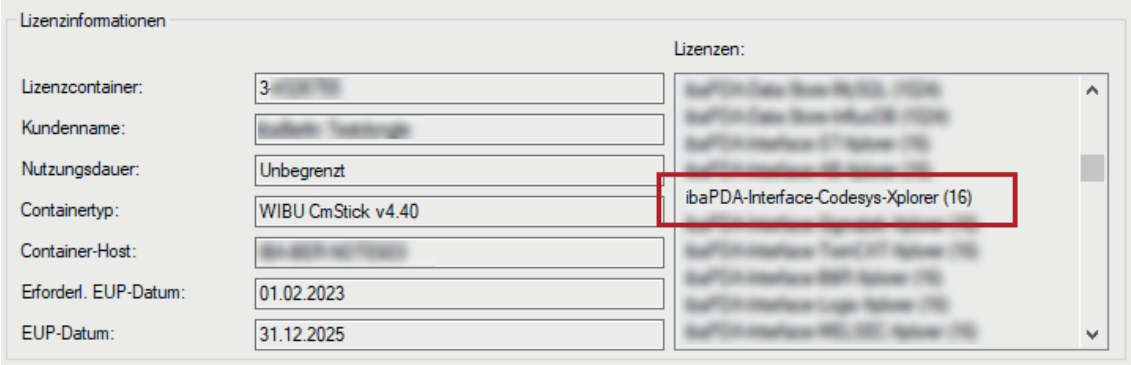
- Es werden keine Daten angezeigt:
  - Der Telegrammpuffer auf der Steuerungsseite ist nicht gefüllt
  - Die Anschlüsse des Sendbausteins sind falsch beschaltet
- Es werden falsche Werte angezeigt:
  - Der Telegrammpuffer auf der Steuerungsseite ist nicht richtig gefüllt (Offset-Fehler)
  - Bytereihenfolge ist falsch eingestellt, siehe ↗ *Allgemeine Moduleinstellungen*, Seite 20
  - Es gibt mehrere Module mit dem gleichen Modulindex
- Die Digitalsignale sind falsch sortiert:
  - Bytereihenfolge ist falsch eingestellt, siehe ↗ *Allgemeine Moduleinstellungen*, Seite 20
- Die Telegramme kommen nicht schneller als ca. 200 ms mit Sequenzfehler:
  - Problem mit "Delayed Acknowledge", siehe ↗ *Probleme mit TCP-Performance durch Delayed Acknowledge*, Seite 36
  - Probleme durch "Nagle-Algorithmus", siehe ↗ *Unbrauchbare TCP-Daten als Folge des Nagle-Algorithmus*, Seite 38

4 Diagnose

4.1 Lizenz

Falls die gewünschte Schnittstelle nicht im Signalbaum angezeigt wird, können Sie entweder in *ibaPDA* im I/O-Manager unter *Allgemein – Einstellungen* oder in der *ibaPDA* Dienststatus-Applikation überprüfen, ob Ihre Lizenz für die Schnittstelle *ibaPDA-Interface-Modbus-TCP-Server* ordnungsgemäß erkannt wird. Die Anzahl der lizenzierten Verbindungen ist in Klammern angegeben.

Die folgende Abbildung zeigt beispielhaft die Lizenz für die Schnittstelle *Codesys-Xplorer*.



4.2 Sichtbarkeit der Schnittstelle

Ist die Schnittstelle trotz gültiger Lizenz nicht zu sehen, ist sie möglicherweise verborgen.

Überprüfen Sie die Einstellung im Register *Allgemein* im Knoten *Schnittstellen*.

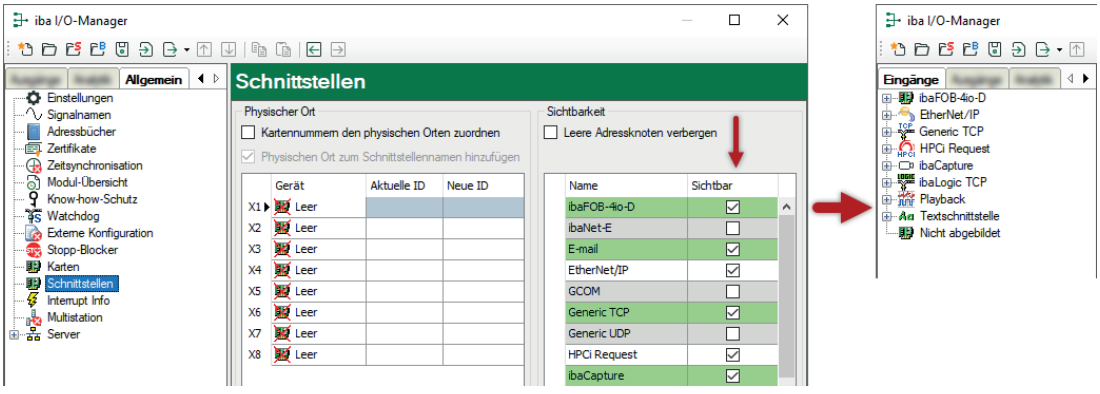
Sichtbarkeit

Die Tabelle *Sichtbarkeit* listet alle Schnittstellen auf, die entweder durch Lizenzen oder installierte Karten verfügbar sind. Diese Schnittstellen sind auch im Schnittstellenbaum zu sehen.

Mithilfe der Häkchen in der Spalte *Sichtbar* können Sie nicht benötigte Schnittstellen im Schnittstellenbaum verbergen oder anzeigen.

Schnittstellen mit konfigurierten Modulen sind grün hinterlegt und können nicht verborgen werden.

Ausgewählte Schnittstellen sind sichtbar, die anderen Schnittstellen sind verborgen:



4.3 Protokolldateien

Wenn Verbindungen zu Zielsystemen bzw. Clients hergestellt wurden, dann werden alle verbindungsspezifischen Aktionen in einer Textdatei protokolliert. Diese (aktuelle) Datei können Sie z. B. nach Hinweisen auf mögliche Verbindungsprobleme durchsuchen.

Die Protokolldatei können Sie über den Button <Protokolldatei öffnen> öffnen. Der Button befindet sich im I/O-Manager:

- bei vielen Schnittstellen in der jeweiligen Schnittstellenübersicht
- bei integrierten Servern (z. B. OPC UA-Server) im Register Diagnose.

Im Dateisystem auf der Festplatte finden Sie die Protokolldateien von *ibaPDA*-Server (…\ProgramData\iba\ibaPDA\Log). Die Dateinamen der Protokolldateien werden aus der Bezeichnung bzw. Abkürzung der Schnittstellenart gebildet.

Dateien mit Namen `Schnittstelle.txt` sind stets die aktuellen Protokolldateien. Dateien mit Namen `Schnittstelle_yyyy_mm_dd_hh_mm_ss.txt` sind archivierte Protokolldateien.

Beispiele:

- `ethernetipLog.txt` (Protokoll von EtherNet/IP-Verbindungen)
- `AbEthLog.txt` (Protokoll von Allen-Bradley-Ethernet-Verbindungen)
- `OpcUAServerLog.txt` (Protokoll von OPC UA-Server-Verbindungen)

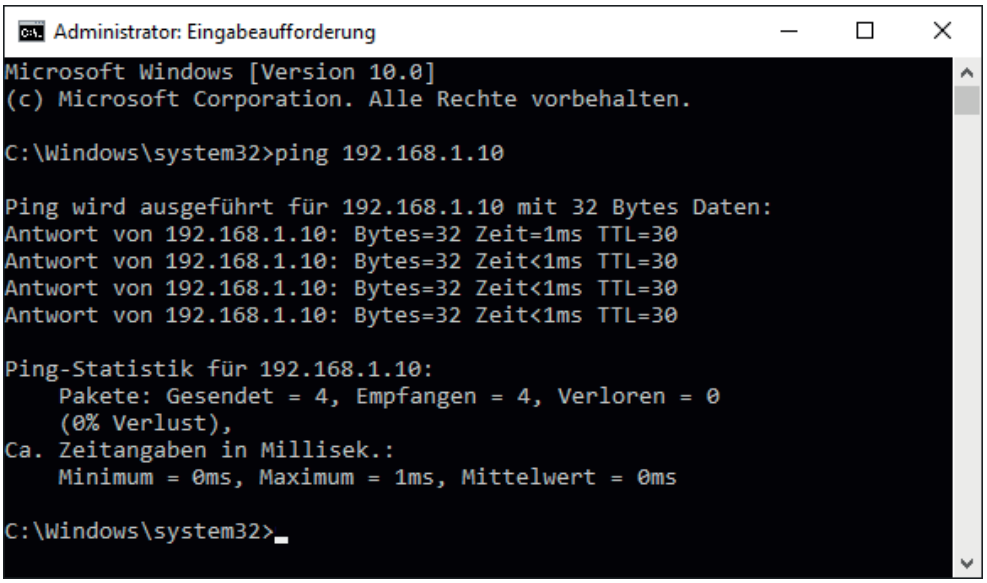
4.4 Verbindungsdiagnose mittels PING

Ping ist ein System-Befehl, mit dem Sie überprüfen können, ob ein bestimmter Kommunikationspartner in einem IP-Netzwerk erreichbar ist.

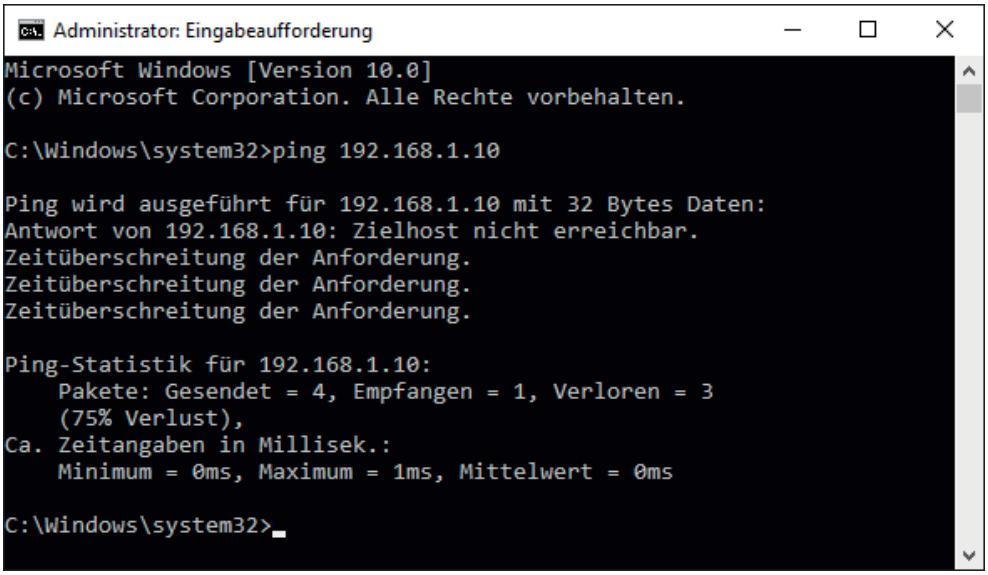
1. Öffnen Sie eine Windows Eingabeaufforderung.

2. Geben Sie den Befehl "ping" gefolgt von der IP-Adresse des Kommunikationspartners ein und drücken Sie <ENTER>.

→ Bei bestehender Verbindung erhalten Sie mehrere Antworten.

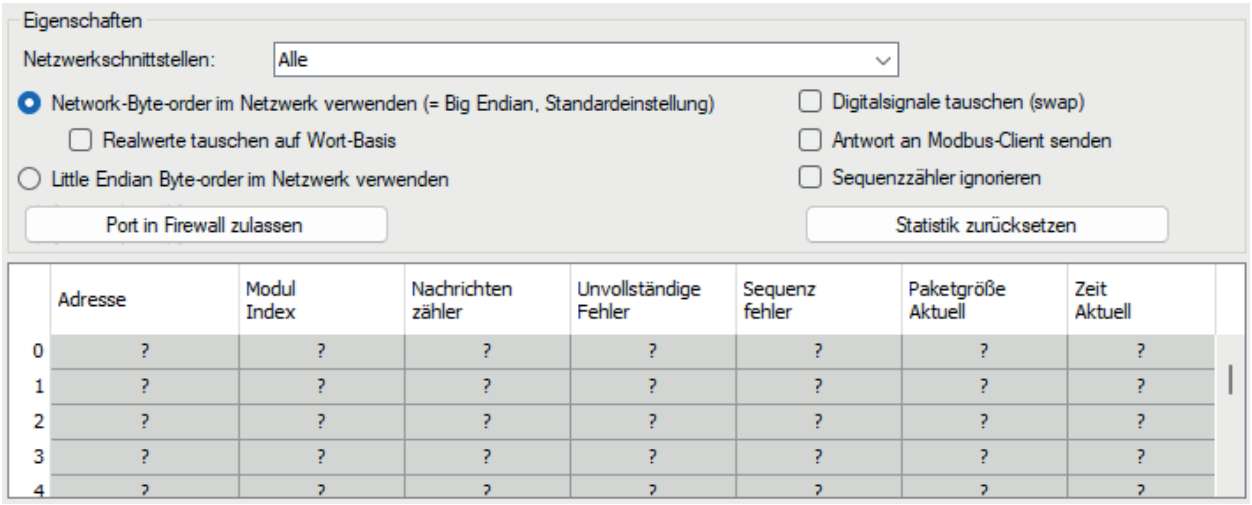


→ Bei nicht bestehender Verbindung erhalten Sie Fehlermeldungen.



4.5 Überprüfung der Verbindung

Wenn Sie im Signalbaum des I/O-Managers die Datenschnittstelle *Modbus TCP Server* markieren, zeigt die Tabelle eine Übersicht über alle Verbindungen dieser Schnittstelle.



Die Tabellenspalten und ihre Bedeutung:

- Adresse: Adresse des Modbus-Clients
- Modulindex: Feld *Startadresses* aus dem Telegramm-Header
- Nachrichtenzähler: Anzahl der empfangenen Telegramme
- Unvollständige Fehler: Wird hochgezählt, wenn die Länge des Telegramms nicht den Längenangaben im Telegramm-Header entspricht
- Sequenzfehler: Wird hochgezählt, wenn in dem Feld *Transaction-ID* des Telegramm-Headers der Zähler nicht in jedem Zyklus um 1 inkrementiert wird
- Paketgröße Aktuell: Telegrammlänge insgesamt
- Zeit Aktuell: Zyklus mit dem die Telegramme vom Modbus-Client eintreffen

Zusätzliche Informationen liefert die Hintergrundfarbe der Zeilen:

Farbe	Bedeutung
Grün	Die Verbindung ist OK, die *Zeit Aktuell* entspricht ungefähr der Modulzeitbasis.
Orange	Die Verbindung ist OK, die Modulzeitbasis ist wesentlich schneller als die *Zeit Aktuell*". Zur Optimierung sollte die Modulzeitbasis angepasst werden.
Grau	Es ist keine Verbindung konfiguriert.

Mögliche Ursachen für einen Verbindungsabbruch

- Modbus-Client ist in Stopp
- keine Ethernet-Verbindung zwischen *ibaPDA*-PC und dem Modbus-PLC
- Fehler in der Verbindungsprojektierung:
  - falsche Remote-IP-Adresse
  - *ibaPDA*-Portnummer stimmt nicht mit der Verbindungsprojektierung überein.

Weitere Fehler:

- Zählen Werte in den Spalten *Unvollständig* und *Sequenzfehler* hoch, deutet das auf einen der folgenden Fehler hin:
  - Fehler im Telegramm-Header
  - Fehler in der Byte-Reihenfolge
  - Das "Delayed Ackowledge"-Problem tritt auf, siehe ↗ *Probleme mit TCP-Performance durch Delayed Acknowledge*, Seite 36

4.6 Diagnosemodule

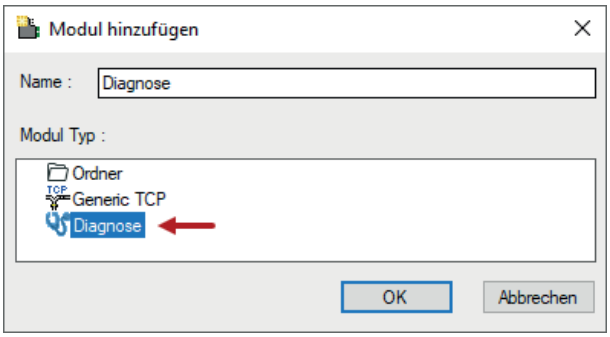
Diagnosemodule sind für die meisten Ethernet-basierten Schnittstellen und Xplorer-Schnittstellen verfügbar. Mit einem Diagnosemodul können Informationen aus den Diagnoseanzeigen (z. B. Diagnoseregister und Verbindungstabellen einer Schnittstelle) als Signale erfasst werden.

Ein Diagnosemodul ist stets einem Datenerfassungsmodul derselben Schnittstelle zugeordnet und stellt dessen Verbindungsinformationen zur Verfügung. Durch die Nutzung eines Diagnosemoduls können die Diagnoseinformationen auch im *ibaPDA*-System durchgängig aufgezeichnet und ausgewertet werden. Diagnosemodule verbrauchen keine Verbindung der Lizenz, da sie keine Verbindung aufbauen, sondern auf ein anderes Modul verweisen.

Nutzungsbeispiele für Diagnosemodule:

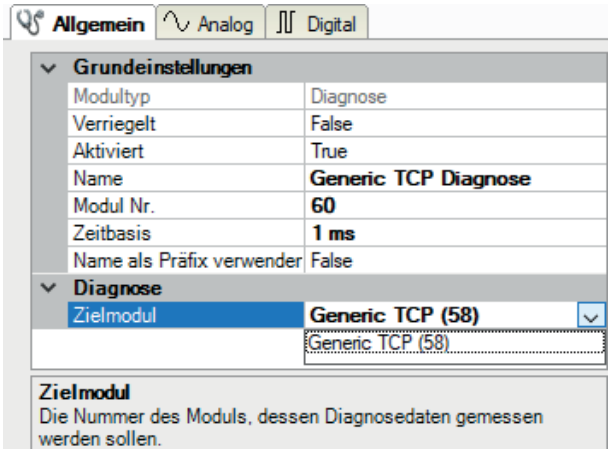
- Wenn der Fehlerzähler einer Kommunikationsverbindung einen bestimmten Wert überschreitet oder eine Verbindung abbricht, kann eine Benachrichtigung generiert werden.
- Bei einem Störungsfall können die aktuellen Antwortzeiten im Telegrammverkehr in einem Störungsreport dokumentiert werden.
- Der Status der Verbindungen kann in *ibaQPanel* visualisiert werden.
- Diagnoseinformationen können über den in *ibaPDA* integrierten SNMP-Server oder OPC DA/UA-Server an übergeordnete Überwachungssysteme wie Netzwerkmanagement-Tools weitergegeben werden.

Wenn für eine Schnittstelle ein Diagnosemodul verfügbar ist, wird im Dialog "Modul hinzufügen" der Modultyp "Diagnose" angezeigt (Beispiel: Generic TCP).



Moduleinstellungen Diagnosemodul

Bei einem Diagnosemodul können Sie folgende Einstellungen vornehmen (Beispiel: Generic TCP):



Die Grundeinstellungen eines Diagnosemoduls entsprechen denen der anderen Module. Es gibt nur eine für das Diagnosemodul spezifische Einstellung, die vorgenommen werden muss: das Zielmodul.

Mit der Auswahl des Zielmoduls weisen Sie das Diagnosemodul dem Modul zu, dessen Verbindungsinformationen erfasst werden sollen. In der Auswahlliste der Einstellung stehen die unterstützten Module derselben Schnittstelle zur Auswahl. Pro Diagnosemodul kann genau ein Datenerfassungsmodul zugeordnet werden. Wenn Sie ein Modul ausgewählt haben, werden in den Registern *Analog* und *Digital* umgehend die verfügbaren Diagnosesignale hinzugefügt. Welche Signale das sind, hängt vom Schnittstellentyp ab. Im nachfolgenden Beispiel sind die Analogwerte eines Diagnosemoduls für ein Generic TCP-Modul aufgelistet.

Allgemein | **Analog** | Digital

	Name	Einheit	Gain	Offset	Aktiv	Istwert
0	IP-Adresse (Teil 1)		1	0	☑	
1	IP-Adresse (Teil 2)		1	0	☑	
2	IP-Adresse (Teil 3)		1	0	☑	
3	IP-Adresse (Teil 4)		1	0	☑	
4	Port		1	0	☑	
5	Telegrammzähler		1	0	☑	
6	Unvollständig		1	0	☑	
7	Paketgröße (aktuell)	Bytes	1	0	☑	
8	Paketgröße (max)	Bytes	1	0	☑	
9	Zeit zwischen Daten (aktuell)	ms	1	0	☑	
10	Zeit zwischen Daten (min)	ms	1	0	☑	
11	Zeit zwischen Daten (max)	ms	1	0	☑	

Die IP(v4-)-Adresse eines Generic TCP-Moduls, z. B. (siehe Abbildung), wird entsprechend der 4 Bytes bzw. Oktetts in 4 Teile zerlegt, um sie leichter lesen und vergleichen zu können. Andere Größen, wie Portnummer, Zählerstände für Telegramme und Fehler, Datengrößen und Laufzeiten für Telegramme werden ebenfalls ermittelt. Im nachfolgenden Beispiel sind die Digitalwerte eines Diagnosemoduls für ein Generic TCP-Modul aufgelistet.

Allgemein | Analog | **Digital**

	Name	Aktiv	Istwert
0	Aktiver Verbindungsmodus	☑	
1	Ungültiges Paket	☑	
2	Verbinde	☑	
3	Verbunden	☑	

Diagnosesignale

Abhängig vom Schnittstellentyp stehen folgende Signale zur Verfügung:

Signalname	Bedeutung
Aktiv	Nur für redundante Verbindungen relevant. Aktiv bedeutet, dass die Verbindung zur Messung der Daten verwendet wird, d. h. bei redundanten Standby-Verbindungen steht der Wert 0. Bei normalen/nicht redundanten Verbindungen steht immer der Wert 1.
Aktualisierungszeit (Istwert/konfiguriert/max/min/Mittelwert)	Gibt die Aktualisierungszeit an, in der die Daten aus der SPS, der CPU oder vom Server abgerufen werden sollen (konfiguriert). Standard ist gleich dem Parameter "Zeitbasis". Während der Messung kann die reale aktuelle Aktualisierungszeit (Istwert) höher sein als der eingestellte Wert, wenn die SPS mehr Zeit zur Übertragung der Daten benötigt. Wie schnell die Daten wirklich aktualisiert werden, können Sie in der Verbindungstabelle überprüfen. Die minimal erreichbare Aktualisierungszeit wird von der Anzahl der Signale beeinflusst. Je mehr Signale erfasst werden, desto größer wird die Aktualisierungszeit. Max/min/Mittelwert: statische Werte der Aktualisierungszeit seit dem letzten Start der Erfassung bzw. Rücksetzen der Zähler
Anforderungen Sendewiederholung	Anzahl der nochmals angeforderten Datentelegramme (in) bei Verlust oder Verspätung
Antwortzeit (aktuell/konfiguriert/max/min/Mittelwert)	Antwortzeit ist die Zeit zwischen Messwertanforderung von *ibaPDA* und Antwort von der SPS bzw. Empfang der Daten. Aktuell: Istwert Max/min/Mittelwert: statische Werte der Antwortzeit seit dem letzten Start der Erfassung bzw. Rücksetzen der Zähler
Anzahl Anforderungsbefehle	Zähler für Anforderungstelegramme von *ibaPDA* an die SPS/CPU
Aufgebaute Verbindungen (in)	Anzahl der aktuell gültigen Datenverbindungen für den Empfang
Aufgebaute Verbindungen (out)	Anzahl der aktuell gültigen Datenverbindungen für das Senden
Ausgangsdatenlänge	Länge der Datentelegramme mit Ausgangssignalen in Bytes (*ibaPDA* sendet)
Datenlänge	Länge der Datentelegramme in Bytes

Signalname	Bedeutung
Datenlänge des Inputs	Länge der Datentelegramme mit Eingangssignalen in Bytes (*ibaPDA* empfängt)
Datenlänge O->T	Größe des Output-Telegramms in Byte
Datenlänge T->O	Größe des Input-Telegramms in Byte
Definierte Topics	Anzahl der definierten Topics
Empfangene Telegramme seit Konfiguration	Anzahl der empfangenen Datentelegramme (in) seit Beginn der Erfassung
Empfangene Telegramme seit Verbindungsstart	Anzahl der empfangenen Datentelegramme (in) seit Beginn des letzten Verbindungsaufbaus
Empfangszähler	Anzahl der empfangenen Telegramme
Exchange ID	ID des Datenaustauschs
Falscher Telegrammtyp	Anzahl der Empfangstelegramme mit falschem Telegrammtyp
Fehlerzähler	Zähler der Kommunikationsfehler
Gepufferte Anweisungen	Anzahl der noch nicht ausgeführten Anweisungen im Zwischenspeicher
Gepufferte Anweisungen sind verloren	Anzahl der gepufferten aber nicht ausgeführten und verlorenen Anweisungen
Gesendete Telegramme seit Konfiguration	Anzahl der gesendeten Datentelegramme (out) seit Beginn der Erfassung
Gesendete Telegramme seit Verbindungsstart	Anzahl der gesendeten Datentelegramme (out) seit Beginn des letzten Verbindungsaufbaus
ID der Verbindung O->T	ID der Verbindung für Output-Daten (vom Zielsystem an *ibaPDA*) Entspricht der Assembly-Instanznummer
ID der Verbindung T->O	ID der Verbindung für Input-Daten (von *ibaPDA* an Zielsystem) Entspricht der Assembly-Instanznummer
IP-Adresse (Teil 1-4)	4 Oktets der IP-Adresse des Zielsystems
IP-Quelladresse (Teil 1-4) O->T	4 Oktets der IP-Adresse des Zielsystems Output-Daten (vom Zielsystem an *ibaPDA*)
IP-Quelladresse (Teil 1-4) T->O	4 Oktets der IP-Adresse des Zielsystems Input-Daten (von *ibaPDA* an Zielsystem)
IP-Zieladresse (Teil 1-4) O->T	4 Oktets der IP-Adresse des Zielsystems Output-Daten (vom Zielsystem an *ibaPDA*)
IP-Zieladresse (Teil 1-4) T->O	4 Oktets der IP-Adresse des Zielsystems Input-Daten (von *ibaPDA* an Zielsystem)
KeepAlive-Zähler	Anzahl der vom OPC UA-Server empfangenen KeepAlive-Telegramme
Lesezähler	Anzahl der Lesezugriffe/Datenanforderungen
Multicast Anmeldefehler	Anzahl der Fehler bei Multicast-Anmeldung
Nachrichtenzähler	Anzahl der empfangenen Telegramme
Paketgröße (aktuell)	Größe der aktuell empfangenen Telegramme

Signalname	Bedeutung
Paketgröße (max)	Größe des größten empfangenen Telegramms
Ping-Zeit (Istwert)	Antwortzeit für ein Ping-Telegramm
Port	Portnummer für die Kommunikation
Producer ID (Teil 1-4)	Producer-ID als 4 Byte unsigned Integer
Profilzähler	Anzahl der vollständig erfassten Profile
Pufferdateigröße (aktuell/mittl./max)	Größe der Pufferdatei zum Zwischenspeichern der Anweisungen
Pufferspeichergröße (aktuell/mittl./max)	Größe des belegten Arbeitsspeichers zum Zwischenspeichern der Anweisungen
Schreibverlustzähler	Anzahl missglückter Schreibzugriffe
Schreibzähler	Anzahl erfolgreicher Schreibzugriffe
Sendezähler	Anzahl der Sendetelegramme
Sequenzfehler	Anzahl Sequenzfehler
Synchronisation	Gerät wird für die isochrone Erfassung synchronisiert
Telegramme pro Zyklus	Anzahl der Telegramme im Zyklus der Aktualisierungszeit
Telegrammzähler	Anzahl der empfangenen Telegramme
Topics aktualisiert	Anzahl der aktualisierten Topics
Trennungen (in)	Anzahl der aktuell unterbrochenen Datenverbindungen für den Empfang
Trennungen (out)	Anzahl der aktuell unterbrochenen Datenverbindungen für das Senden
Unbekannter Sensor	Anzahl unbekannter Sensoren
Ungültiges Paket	Ungültiges Datenpaket erkannt
Unvollständig	Anzahl unvollständiger Telegramme
Unvollständige Fehler	Anzahl unvollständiger Telegramme
Verarbeitete Anweisungen	Anzahl der ausgeführten SQL-Anweisungen seit dem letzten Start der Erfassung
Verbinde	Verbindung wird aufgebaut
Verbindungsphase (in)	Zustand der ibaNet-E Datenverbindung für den Empfang
Verbindungsphase (out)	Zustand der ibaNet-E Datenverbindung für das Senden
Verbindungsversuche (in)	Anzahl der Versuche, die Empfangsverbindung (in) aufzubauen
Verbindungsversuche (out)	Anzahl der Versuche, die Sendeverbindung (out) aufzubauen
Verbunden	Verbindung ist aufgebaut
Verbunden (in)	Eine gültige Datenverbindung für den Empfang (in) ist vorhanden
Verbunden (out)	Eine gültige Datenverbindung für das Senden (out) ist vorhanden

Signalname	Bedeutung
Verlorene Images	Anzahl der verlorenen Images (in), die selbst nach einer Sendewiederholung nicht empfangen wurden
Verlorene Profile	Anzahl unvollständiger/fehlerhafter Profile
Zeilen (letzte)	Anzahl der Ergebniszeilen der letzten SQL-Abfrage (innerhalb der projektierten Anzahl von Ergebniszeilen)
Zeilen (Maximum)	Höchste Anzahl der Ergebniszeilen einer SQL-Abfrage seit dem letzten Start der Erfassung (maximal gleich der projektierten Anzahl von Ergebniszeilen)
Zeit zwischen Daten (aktuell/max/min)	Zeit zwischen zwei korrekt empfangenen Telegrammen Aktuell: zwischen den letzten zwei Telegrammen Max/min: statistische Werte seit Start der Erfassung oder Rücksetzen der Zähler
Zeit-Offset (Istwert)	Gemessene Zeitdifferenz der Synchronität zwischen dem ibaNet-E-Gerät und *ibaPDA*

5 Anhang

5.1 Fehlerbehebung

Im Folgenden finden Sie Hilfestellung zu möglichen Fehlern bei der Anwendung mit *ibaPDA-Interface-Modbus-TCP-Server*. Wenden Sie sich bei weitergehenden Fragen oder im Zweifelsfall an den iba-Support.

5.1.1 Probleme mit TCP-Performance durch Delayed Acknowledge

ibaPDA-Messungen von Automatisierungsgeräten mit TCP/IP funktionieren manchmal nicht mit Zykluszeiten < 200 ms.

Fehlerbild in ibaPDA

Unvollständige Telegramme und/oder Ausreißer in den Datenwerten (je nach Controller-Typ des Senders)

Ursache

Es gibt im TCP/IP-Protokoll verschiedene Varianten, wie das "Acknowledge" behandelt wird.

Der Standard WinSocket arbeitet nach RFC1122 mit dem "Delayed Acknowledge"-Mechanismus (Delayed ACK). Dieser sagt aus, dass das "Acknowledge" verzögert wird, bis weitere Telegramme eintreffen, um diese dann gemeinsam zu quittieren. Falls keine weiteren Telegramme kommen, wird spätestens nach 200 ms (abhängig vom Socket) das ACK-Telegramm gesendet.

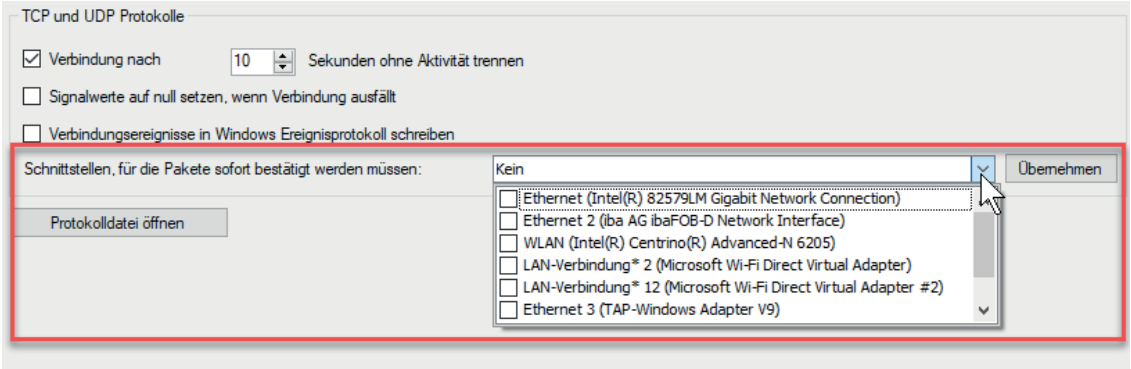
Der Datenfluss wird durch ein "Sliding Window" (Parameter Win=nnnn) gesteuert. Der Empfänger gibt an, wie viele Bytes er empfangen kann, ohne eine Quittung zu senden.

Manche Controller akzeptieren dieses Verhalten nicht, sondern erwarten nach jedem Datentelegramm eine Quittung. Falls dieses nicht innerhalb einer bestimmten Zeit (200 ms) kommt, wiederholt er das Telegramm und packt evtl. neu zu sendende Daten dazu, was beim Empfänger zu einem Fehler führt, da das vorherige Telegramm korrekt empfangen wurde.

Abhilfe

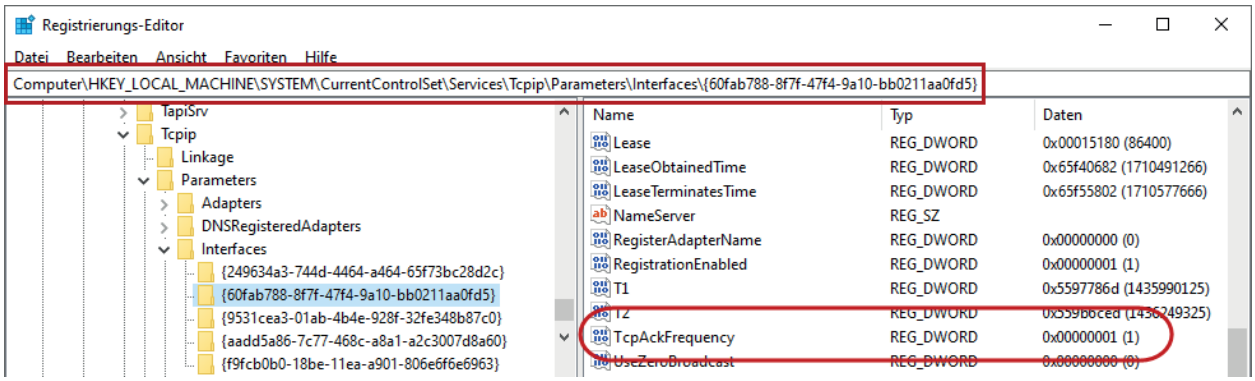
Das "Delayed Acknowledge" lässt sich einzeln pro Netzwerkadapter über einen Eintrag in der Windows Registry abschalten. Zur einfachen Änderung bietet *ibaPDA* im I/O-Manager unter *Allgemein* im Register *Einstellungen* einen entsprechenden Dialog.

Wählen Sie in der Liste der Netzwerkadapter diejenigen aus, für die das "Delayed Acknowledge" deaktiviert werden soll, und klicken Sie danach auf <Übernehmen>.



Der Parameter "TcpAckFrequency" (REG_DWORD = 1) wird dadurch im Registry-Pfad der ausgewählten Netzwerkadapter angelegt:

`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\{InterfaceGUID}`



Hinweis

Grundsätzlich können Sie derartige, TCP-spezifische Probleme umgehen, indem Sie *UDP* anstelle von *TCP* nutzen.

Das User Datagram Protocol (UDP) ist ein minimales, nicht verbindungsorientiertes und gegen Telegrammverluste ungesichertes Netzwerkprotokoll. Dabei wird u. a. auf Empfangsquittierung der gesendeten Daten verzichtet. In stabilen und performanten Netzwerken fällt dies jedoch nicht nennenswert ins Gewicht und kann aufgrund der bei *ibaPDA* üblichen zyklischen Datenübertragung vernachlässigt werden.

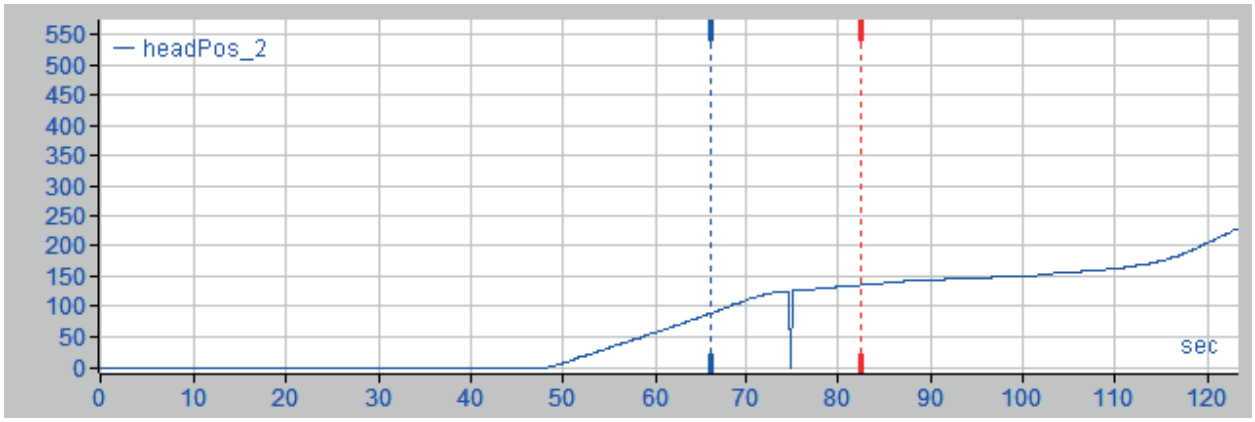
5.1.2 Unbrauchbare TCP-Daten als Folge des Nagle-Algorithmus

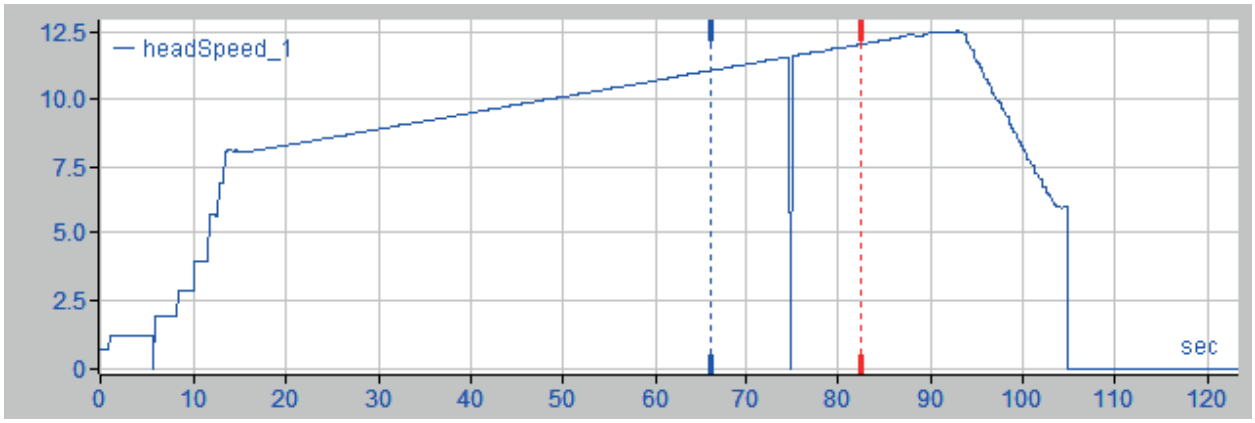
Symptome

ibaPDA-Messungen von Automatisierungsgeräten mit TCP/IP zeigen Ausreißer in den Messwerten.

Fehlerbild in ibaPDA

Unvollständige Telegramme und/oder Ausreißer in den Datenwerten (siehe Beispiele in den folgenden Abbildungen)





Ursache

Der Nagle-Algorithmus (Nagle's algorithm) ist ein Mechanismus zur Verbesserung der TCP-Effizienz, indem er die Anzahl der über das Netz gesendeten kleinen Pakete reduziert und mehrere Datenblöcke sammelt, bevor die Daten über das Netz gesendet werden.

Da die Schnittstelle "Generic-TCP" kein Protokoll auf Anwendungsebene verwendet, kann *ibaPDA* als Empfänger diese zusammengefassten Nachrichten nicht korrekt verarbeiten. Die Schnittstelle *Generic TCP* erwartet nur ein Datagramm pro TCP-Telegramm mit stets gleichem Layout und gleicher Länge.

Aber der Nagle-Algorithmus und die Option *Delayed ACK* spielen in einem TCP/IP-Netzwerk nicht gut zusammen, siehe ↗ *Probleme mit TCP-Performance durch Delayed Acknowledge*, Seite 36:

Der Delayed ACK-Mechanismus versucht, mehr Daten pro Segment zu senden, wenn er kann. Ein Teil des Nagle-Algorithmus hängt aber von einem ACK ab, um Daten zu senden. Delayed ACKs warten also darauf, das ACK zu senden, während der Nagle-Algorithmus darauf wartet, das ACK zu empfangen.

Dies führt zu zufälligen Verzögerungen von 200 ms bis 500 ms bei Segmenten, die sonst sofort gesendet und an den empfangsseitigen Stack von *ibaPDA* als Anwendung übergeben werden könnten.

Abhilfe

Es wird empfohlen, zunächst den *Delayed ACK*-Mechanismus zu deaktivieren, siehe Kapitel ↗ *Probleme mit TCP-Performance durch Delayed Acknowledge*, Seite 36 erläutert. In einer typischen Echtzeitanwendung schickt der Sender dann die neuen Daten mit einer bestimmten Zykluszeit an *ibaPDA*, da die vorherigen Daten sofort quittiert wurden. Je nach Implementierung des TCP/IP-Stacks auf der Senderseite kann der Nagle-Algorithmus dennoch aktiv werden und automatisch eine Reihe kleiner Puffernachrichten aggregieren, wodurch der Algorithmus die Übertragung absichtlich verlangsamt.

Dies kann auch sporadisch durch eine kurzzeitige Überlastung auf der Senderseite geschehen, die den Stack dazu veranlasst, einige Nachrichten zusammenzulegen.

Um den puffernden Nagle-Algorithmus zu deaktivieren, verwenden Sie die Socket-Option *TCP_NODELAY*. Die Socket-Option *TCP_NODELAY* ermöglicht es dem Netzwerk, die durch den Nagle-Mechanismus verursachten Delays zu umgehen, indem der Nagle-Algorithmus deaktiviert wird und die Daten gesendet werden, sobald sie verfügbar sind.

Die Aktivierung von *TCP_NODELAY* zwingt einen Socket, die Daten in seinem Puffer zu senden, unabhängig von der Paketgröße. Das *TCP_NODELAY*-Flag ist eine Option, die für jeden einzelnen Socket aktiviert werden kann und beim Erstellen eines TCP-Sockets angewendet wird.

(Siehe Eigenschaft *Socket.NoDelay* in .NET-Anwendungen im Namespace *System.Net.Sockets*)

Hinweis

Grundsätzlich können Sie derartige, TCP-spezifische Probleme umgehen, indem Sie *UDP* anstelle von *TCP* nutzen.

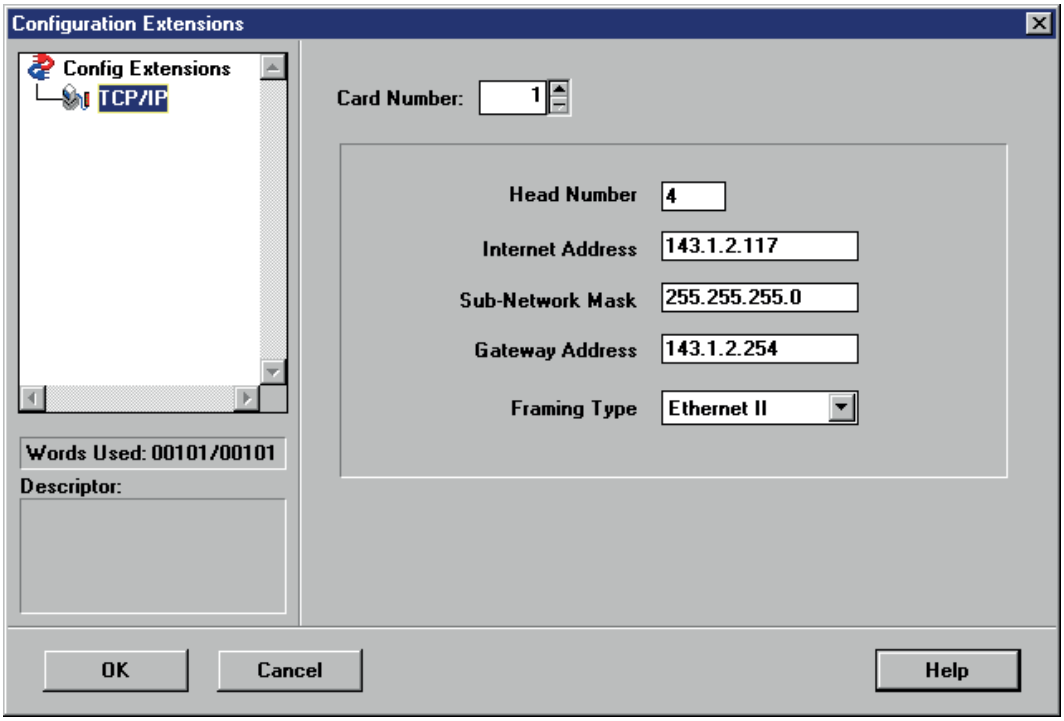
Das User Datagram Protocol (UDP) ist ein minimales, nicht verbindungsorientiertes und gegen Telegrammverluste ungesichertes Netzwerkprotokoll. Dabei wird u. a. auf Empfangsquittierung der gesendeten Daten verzichtet. In stabilen und performanten Netzwerken fällt dies jedoch nicht nennenswert ins Gewicht und kann aufgrund der bei *ibaPDA* üblichen zyklischen Datenübertragung vernachlässigt werden.

5.2 Projektierungsbeispiele

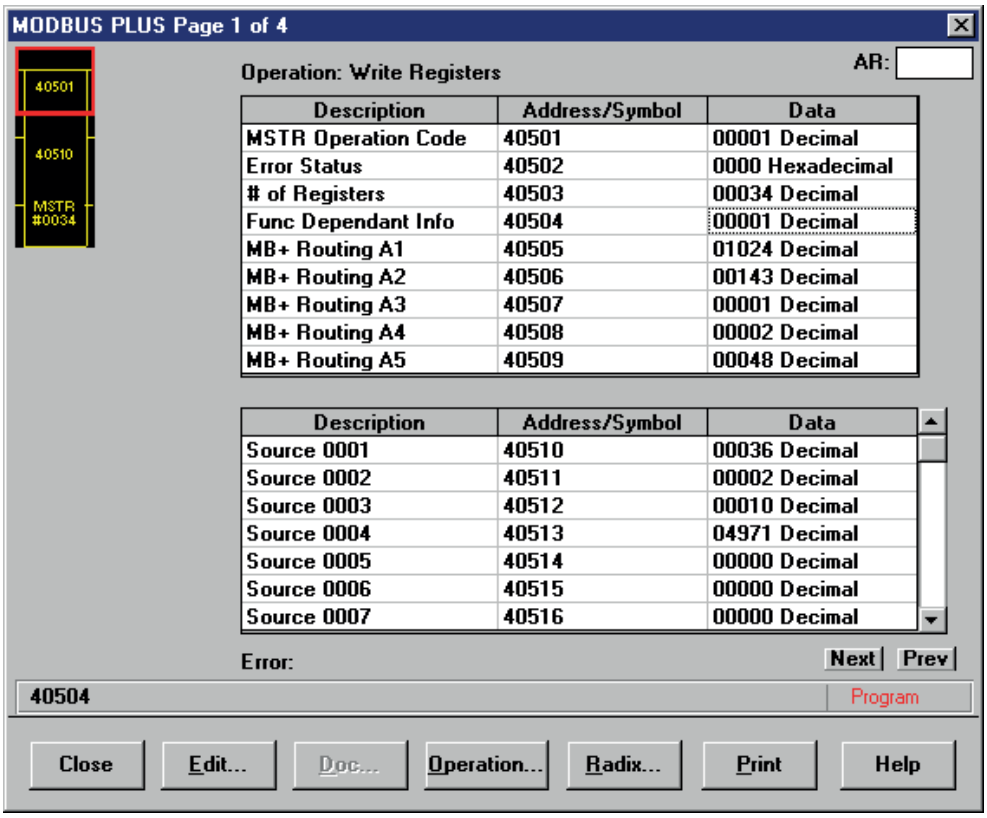
5.2.1 Projektierungsbeispiel Modicon Quantum

5.2.1.1 Konfiguration der TCP/IP-Schnittstelle in ProWORX NxT

1. Installieren Sie die Ethernet-Module (NOE).
2. Konfigurieren Sie die IP-Adresse für die NOE (siehe folgende Abbildung).



3. Erstellen Sie ein Ladder- oder ConCept-Programm mit einem MSTR-Baustein und konfigurieren Sie ihn, wie in der nächsten Abbildung zu sehen.



- MSTR operation Code:
  1 Decimal bedeutet Schreibbefehl.
- Error Status:
  0 hex steht für mögliche Kommunikationsfehler.
- # of Registers:
  34 Decimal; steht für 34 an *ibaPDA* zu sendende Register (32 Analogwerte + 32 Digitalwerte). Wenn Sie den Modultyp *Modbus Real* verwenden, müssen 66 Register gesendet werden. Wenn Sie den Modultyp *Modbus Allgemein* verwenden, können bis zu 122 Register gesendet werden.

Hinweis

Wenn andere Zahlen als 34 bzw. 66 konfiguriert werden (bei Integer, Dig512 und Float), führt dies zu Fehlermeldungen, sowohl in *ibaPDA* als auch am MSTR-Baustein, und es wird keine Kommunikation aufgebaut.

- Func. Dependant Info:
  1 Decimal; steht für *ibaPDA*-Modulnummer 1. Verwenden Sie verschiedene Nummern für verschiedene *ibaPDA*-Modulnummern. Addieren Sie 100 für Module vom Typ Modbus Real, addieren Sie 200 für Allgemein-Module.

- MB+Routing A1:
  In dem hier gezeigten Beispiel wurde die NOE im Slot 4 installiert. Dementsprechend wurde hier der Wert 1024 dezimal eingetragen, was 0x400 entspricht. Die 4 im hexadezimalen Wert steht für die Slot-Nummer. Für ein NOE mit der Slot-Nummer 5 wäre der Wert 0x500, entsprechend 1280 dezimal. Dieser Screenshot wurde vom Programm ProWORX NxT aufgenommen.
- MB+Routing A2..A5:
  Hier im Beispiel IP-Adresse 143.1.2.48; diese vier Felder enthalten die IP-Adresse von *ibaPDA*.

5.2.1.2 Ladder-Programm für die SPS

Es gibt zwei Arten, die SPS zu programmieren:

ibaPDA **sendet keine Quittierung**

- Vorteil:
  Schnellere Kommunikation, geringerer Datenverkehr auf dem LAN
- Nachteil:
  Funktioniert nicht mit NOE Serie 700
  Wenn zu schnell eingestellt, kann die SPS blockiert werden.

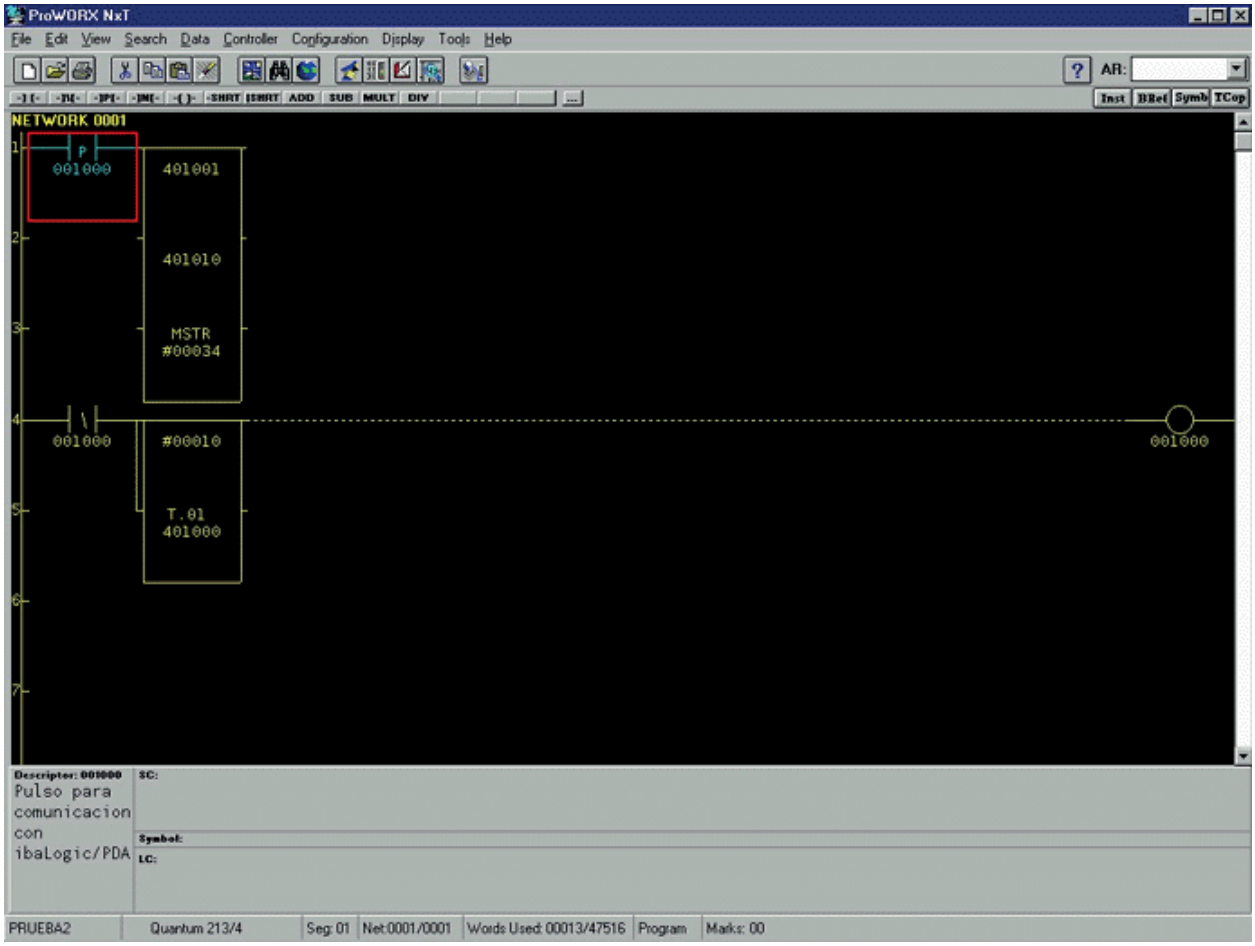
ibaPDA **sendet eine Quittierung**

- Vorteil:
  NOE Serie 700 unterstützt
- Nachteil:
  Langsamere Kommunikation

Falls Sie die zweite Option wählen, aktivieren Sie „Antwort an Modbus-Master senden" in den allgemeinen Einstellungen von Modbus-TCP-Server, siehe ↗ *Allgemeine Einstellungen der Schnittstelle*, Seite 17. Wenn Sie die erste Option verwenden wollen, deaktivieren Sie das Kästchen einfach.

5.2.1.2.1 ibaPDA sendet keine Quittierung (Beispiel)

Vervollständigen Sie das Programm wie in der Abbildung unten zu sehen (Beispiel: Ladder-Programm in ProWORX NxT).



T.01 ist hierbei ein Timer, der Pulse generiert, in diesem Fall alle 10 ms.

Beachten Sie, dass der markierte Schalter "P 001000" in dem roten Rechteck der Pulsausgang ist, der von T.01 generiert wird.

Beachten Sie weiterhin, dass *ibaPDA* keine Quittierung sendet und der MSTR-Baustein somit alle 10 ms die Register sendet, sogar wenn Kommunikationsfehler auftreten oder keine Antwort erfolgt. Erwarten Sie keine Antwort seitens der Kommunikation!

Wenn die SPS gestartet wird, sollte der MSTR-Baustein die Kommunikation mit *ibaPDA* aufbauen. Anschließend sollte in der *ibaPDA*-Diagnose, bzw. in der Verbindungsübersicht, die entsprechende Verbindung mit einem grünen Symbol gekennzeichnet sein. Beobachten Sie einige Minuten, wie sich die Kommunikation verhält. Wenn die Kommunikation mehr als zwei Minuten aufrechterhalten wird, dann sollte sie in Ordnung sein.

Hinweis

Ein (1) Sequenzfehler, der angezeigt wird, ist OK. Dies deutet auf den ersten Versuch hin, mit *ibaPDA* zu kommunizieren.

Jedes Mal wenn der MSTR-Baustein Daten sendet, zählt ein Telegrammzähler in der Verbindungsübersicht hoch (I/O-Manager in *ibaPDA*).

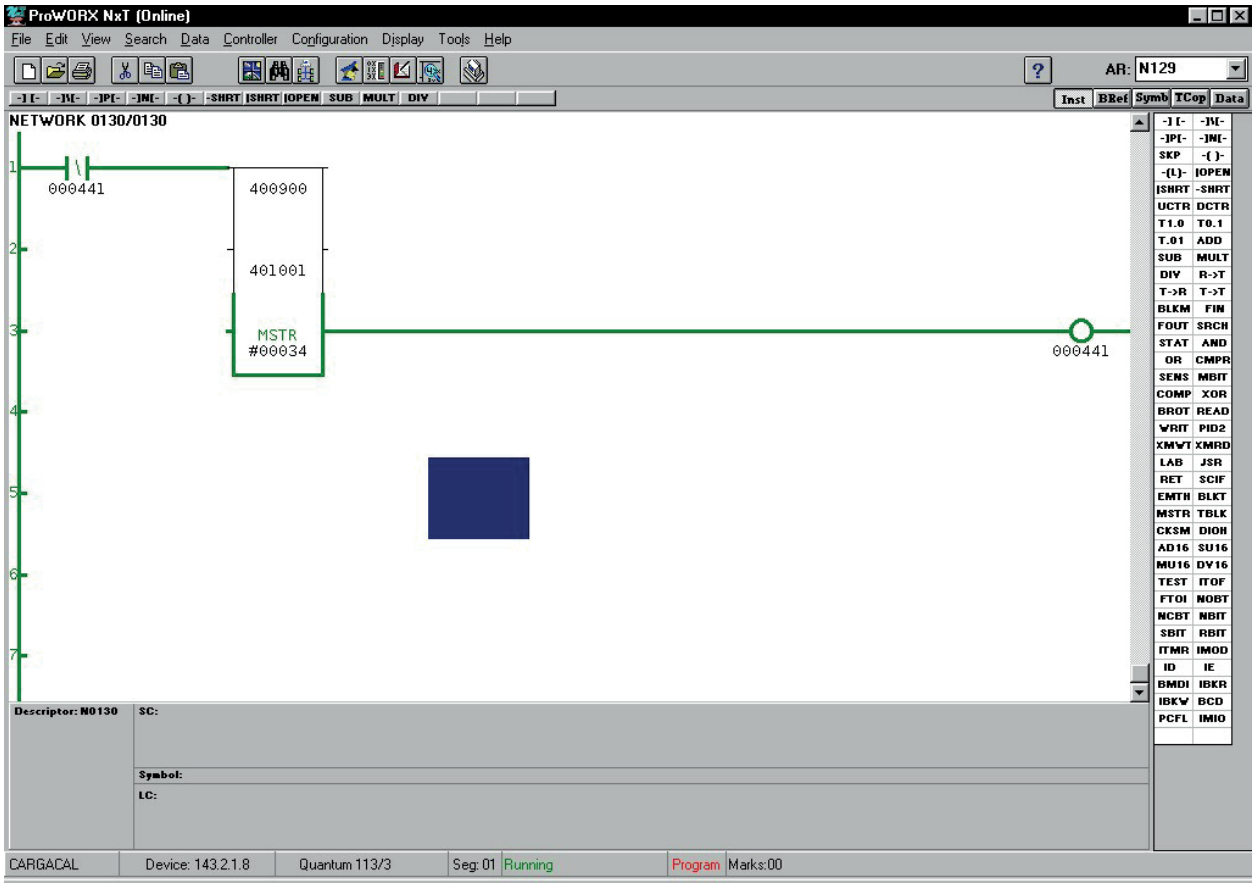
Um "Live"-Daten zu testen, empfehlen wir die Projektierung eines Zählers in der SPS-Applikation, dessen Wert in das erste Register geschrieben wird, das vom MSTR-Baustein übertragen wird.

Wenn Sie mehr als 32 Analog- und 32 Digitalsignale verwenden wollen, gehen Sie bei der MSTR-Konfiguration wie oben beschrieben vor, nur dass Sie den Parameter *Func. Dependant Info* (*ibaPDA* Modulnummer) ändern.

In *ibaPDA* müssen Sie jeweils Module des gewünschten Typs hinzufügen.

5.2.1.2.2 ibaPDA sendet eine Quittierung (Beispiel)

Vervollständigen Sie das Programm wie in der folgenden Abbildung (Beispiel: Ladder-Programm in ProWORX NxT).



Beachten Sie, dass diesmal kein Timer vorhanden ist, der den MSTR-Baustein zum regelmäßigen Schreiben zwingt. Stattdessen wird nach jeder erfolgreichen Übermittlung von *ibaPDA* eine "ACK"-Antwort gesendet, die wiederum zu einer "SUCCESSFUL"-Ausgabe führt und den MSTR-Baustein für den nächsten Sendezyklus vorbereitet.

5.2.1.3 ConCept-Programm für die SPS

Die durchzuführenden Schritte mit der Programmiersoftware ConCept entsprechen prinzipiell den in den beiden vorhergehenden Abschnitten beschriebenen.

Bei der Parametrierung des MSTR-Bausteins werden unter ConCept andere Parameterbezeichnungen verwendet:

ProWORX NxT	ConCept
MSTR operation Code	w1
Error Status	w2
# of Registers	w3
Func. Dependant Info	w4
MB+Routing A1	w5
MB+Routing A2	w6
MB+Routing A3	w7
MB+Routing A4	w8
MB+Routing A5	w9

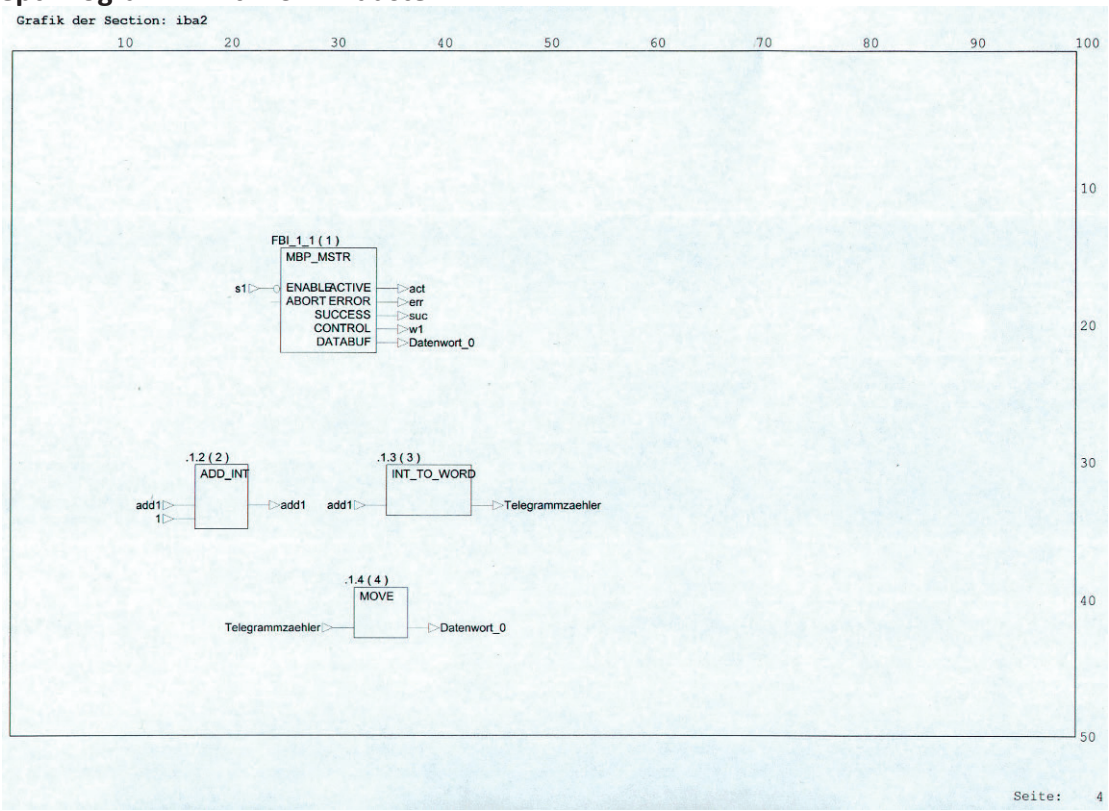
Beispiel für ConCept-Konfiguration und -Programm

Variablenliste (Name: Alles, Typ: Alles, Datentyp: Alles, Sortiert nach: Name)

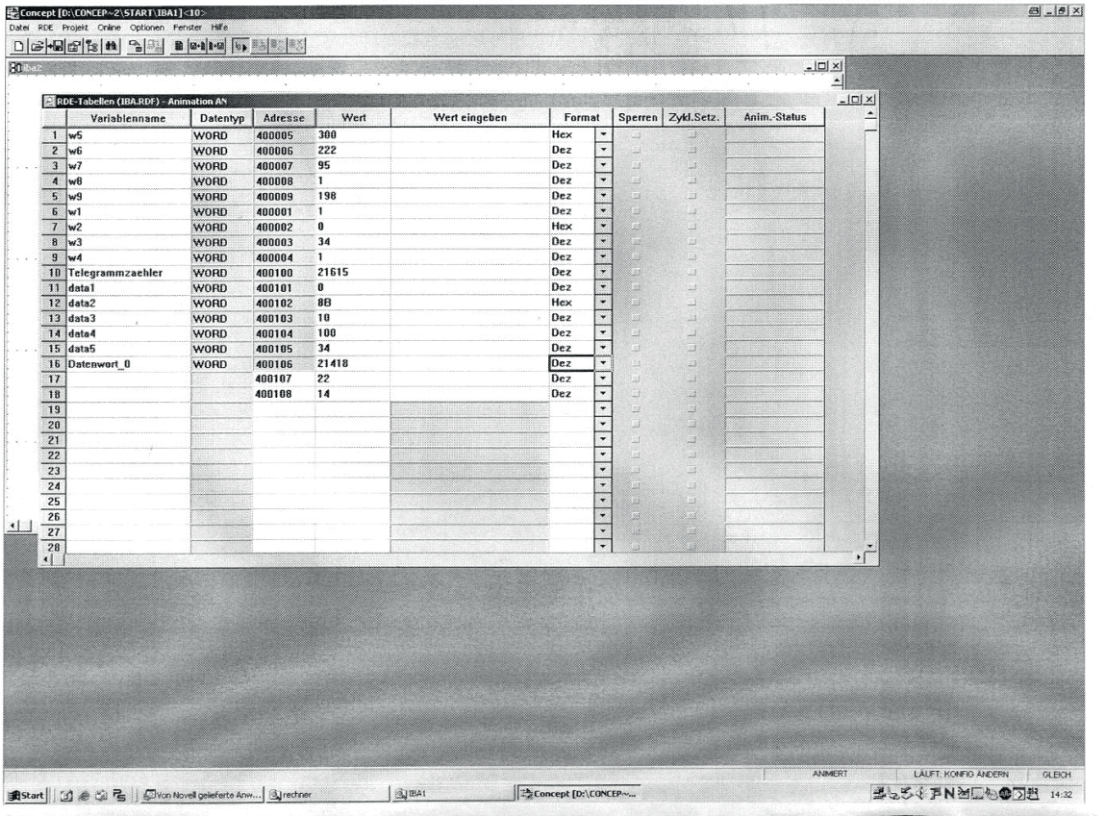
Variablenname	Typ	DTyp	Adresse	Initial-Wert	Kommentar	Verw
act	VAR	BOOL				1
add1	VAR	INT				3
data1	VAR	WORD	400101			0
data2	VAR	WORD	400102			0
data3	VAR	WORD	400103			0
data4	VAR	WORD	400104			0
data5	VAR	WORD	400105			0
Datenwort_0	VAR	WORD	400106			2
err	VAR	BOOL				1
iba2	IVAR	SECT_CTRL				0
s1	VAR	BOOL				1
suc	VAR	BOOL				1
Telegrammzaehler	VAR	WORD	400100			2
w1	VAR	WORD	400001	1		1
w2	VAR	WORD	400002			0
w3	VAR	WORD	400003	10		0
w4	VAR	WORD	400004	139		0
w5	VAR	WORD	400005	0300		0
w6	VAR	WORD	400006	222		0
w7	VAR	WORD	400007	95		0
w8	VAR	WORD	400008	1		0
w9	VAR	WORD	400009	198		0

Eine ausführlichere Beschreibung zum Thema ConCept-Projektierung (Beispiel) ist in Vorbereitung.

ConCept-Programm mit MSTR-Baustein:



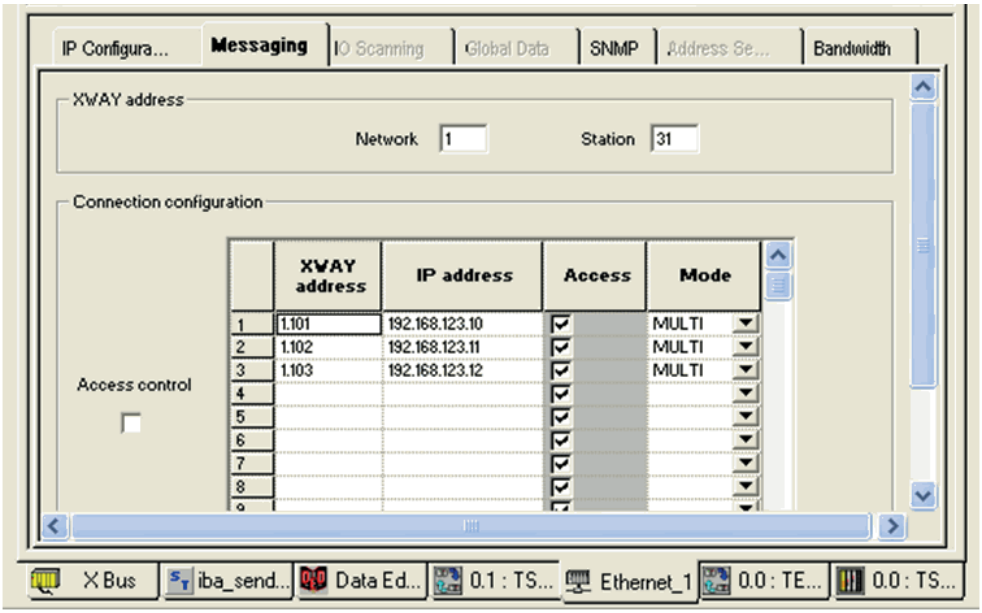
ConCept RDE-Tabelle:



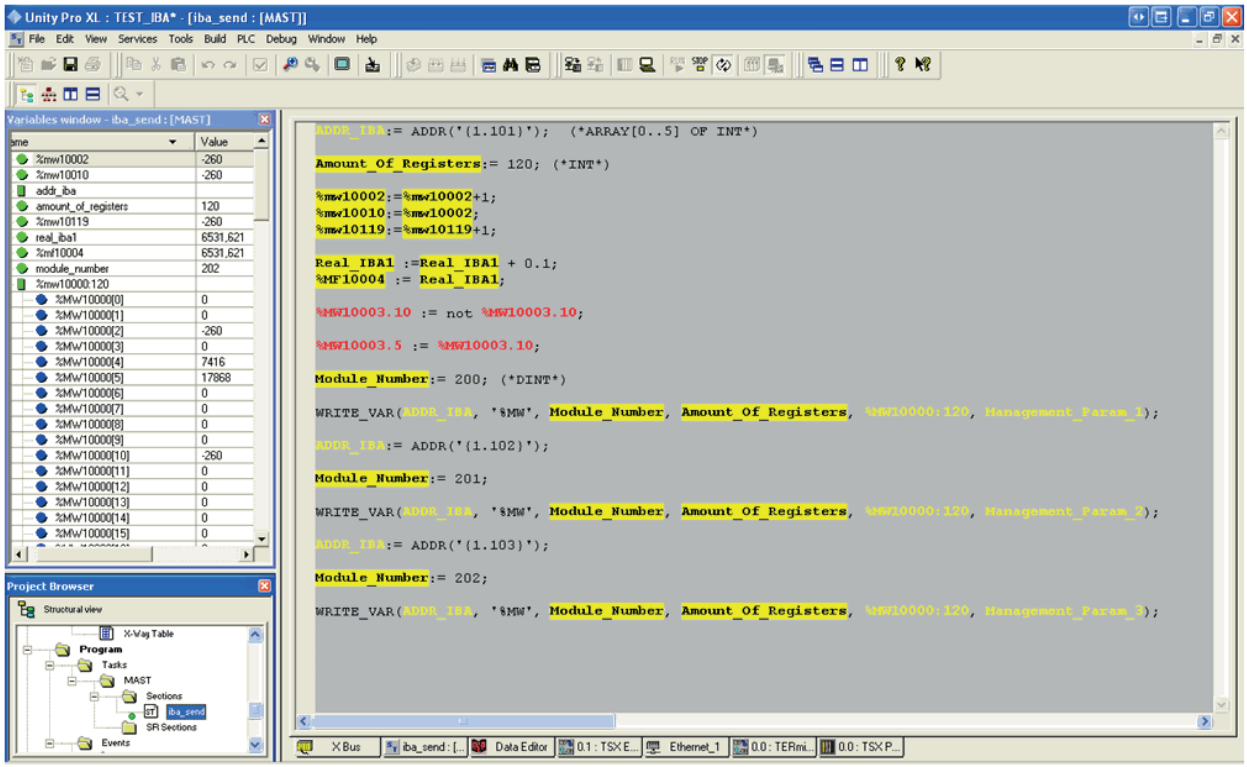
5.2.1.4 Unity Pro XL-Programm für die SPS (Beispiel Modul Allgemein)

1. Installieren Sie das Ethernet-Modul.
2. Konfigurieren Sie die IP-Adresse für das Ethernet-Modul.

   Das Ethernet-Modul finden Sie im Projekt-Browser unter: Station\Communication\networks\



3. Erstellen Sie ein neues Ladder(LAD)- oder Structure-Text(ST)-Programm. Ein neues Programm kann unter Station\Program\Tasks\MAST\Sections hinzugefügt werden (Beispiel: ST-Programm in Unity Pro XI):



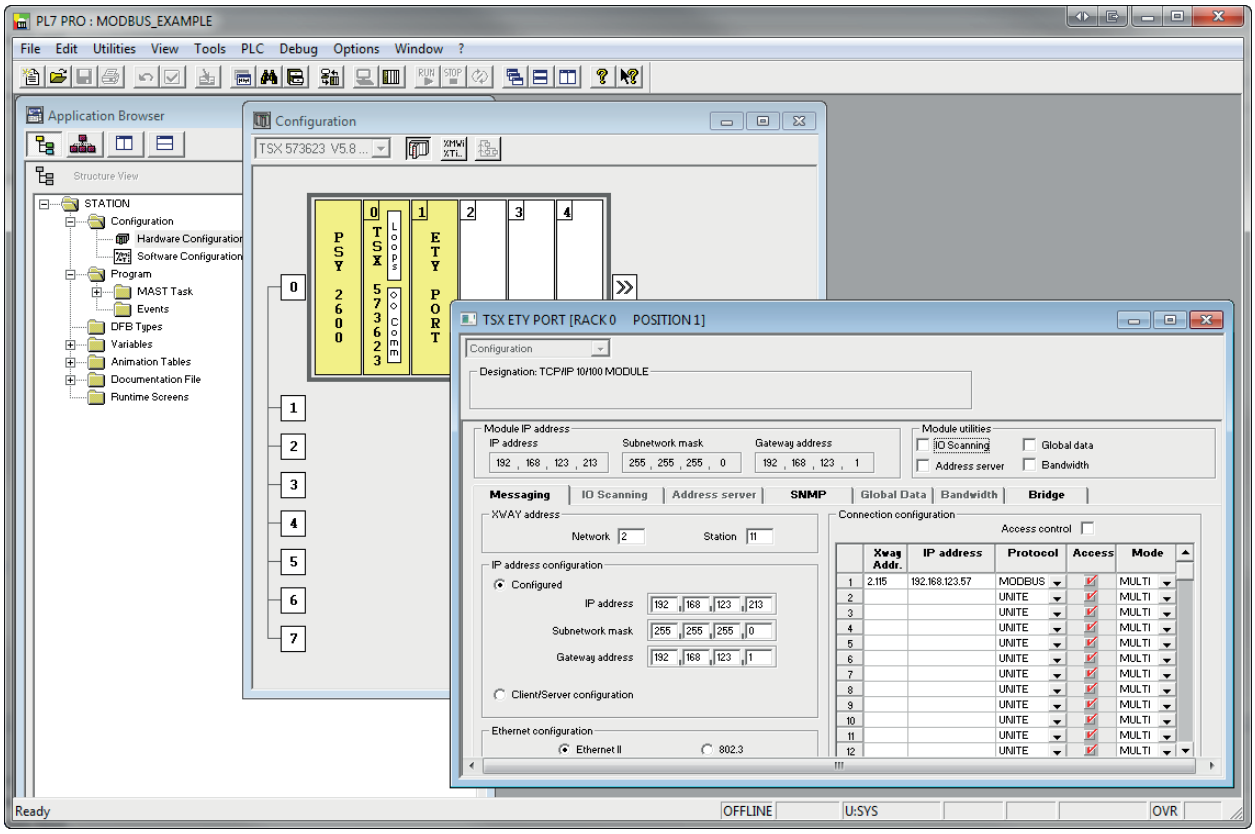
Um eine Nachricht über MODBUS TCP/IP zu senden, wird die WRITE_VAR Funktion verwendet. Diese Funktion benötigt einige Parameter wie IP-Adresse, Variablen\-typ, Modulnummer, Anzahl der Register, etc.

Die IP-Adresse wird mit der ADDR_IBA Variablen dargestellt. Diese Variable ist die XWAY-Adresse, die im Ethernet-Modul konfiguriert ist. Für jedes Modul wird eine eigene IP-Adresse benötigt. Das bedeutet, für 3 Module müssen 3 IP-Adressen konfiguriert werden.

Im Beispiel wurden 3 unterschiedliche Variablen erstellt, um einen besseren Überblick über das Programm zu behalten. Als Modulnummern wurden die Nummern 200, 201 und 202 verwendet, da Allgemein-Module an *ibaPDA* gesendet werden sollen. Jedes Modul enthält Float-, Integer- und Bool-Werte mit einer Gesamtlänge von 120 Registern.

5.2.2 Projektierungsbeispiel mit PL7 Pro

5.2.2.1 Netzwerkkonfiguration



In der Hardware-Konfiguration der ETY (Netzwerk)-Karte, muss eine XWAY-Adresse eingetragen werden, um Daten an *ibaPDA* mit dem MODBUS-Protokoll senden zu können.

Wie im *Connection configuration*-Feld oben zu sehen (Bespiel: Netzwerk-Konfiguration in PL7 Pro), besteht XWAY aus:

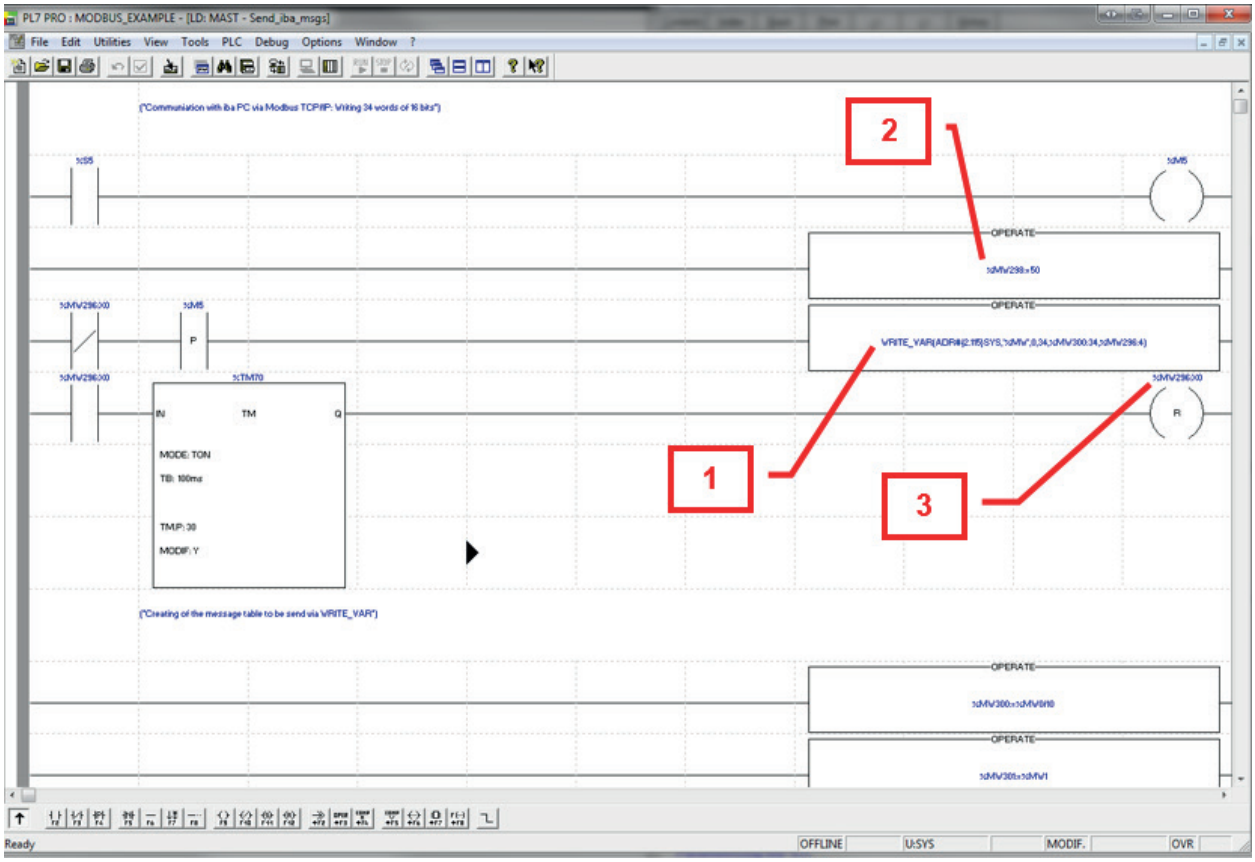
- XWAY Address: Diese Adresse wird vom Programm als "Gateway" verwendet.
- IP address: Hier muss die IP-Adresse des *ibaPDA*-Systems eingetragen werden.
- Protocol: Hier sollte das MODBUS-Protokoll ausgewählt werden, um Nachrichten über MODBUS zu senden.

- Access: Dieses Feld muss ausgewählt werden
- Mode: Multimode auswählen.

Wenn alle Einträge gemacht sind und die IP-Adresse der Karte konfiguriert ist, muss die Konfiguration bestätigt werden.

5.2.2.2 Nachrichten-Konfiguration (Beispiel)

Erstellen Sie ein Programm im "Mast"-Arbeitsbereich (Beispiel: Programm in PL7 Pro).



Die Nachricht soll zyklisch an *ibaPDA* gesendet werden. Deshalb wird eine Systemvariable "%S5" benutzt. Diese Systemvariable ist ein digitales Signal, das mit einer Periode von 100 ms umschaltet. An der steigenden Flanke des Digitalsignals wird die Nachricht an *ibaPDA* gesendet (Netzwerk 3).

Mit der WRITE_VAR Funktion (1) werden Daten an *ibaPDA* gesendet. Diese Funktion verwendet folgende Einstellungen:

```
WRITE_VAR(ADR#{2.115}SYS,'%MW',0,34,%MW300:34,%MW296:4)
```

- XWAY Addr.: Die zuvor erzeugte XWAY Addr. muss hier eingefügt werden ADR#{<XWAY Addr} SYS
- Type: Hier ist der Variablentyp '%MW' ausgewählt
- Modulindex: Hier soll der gleiche Modulindex wie in *ibaPDA* verwendet werden.
- Anzahl der Register: 34 ist die Anzahl der Register (Word mit 16 Bit), die an *ibaPDA* gesendet werden

- Startadresse: Die Adressen %MW300 bis %MW334 werden gesendet
- Management Words: Diese Worte enthalten den Status, Sendeeinstellungen, etc.

Die Funktion (2) mit der %MW298 auf 50 eingestellt ist, wird verwendet, um einen Timeout der WRITE_VAR Funktion zu setzen.

Der Wert MW269:X0 (3) zeigt, ob die Variable gesendet wurde oder nicht.

Der Timer (%TM70) wird verwendet, um zu verhindern, dass die Nachricht mehr als 1 Minute (im Normalfall auf z. B. 100 ms gesetzt) benötigt, um an das *ibaPDA*-System gesendet zu werden.

Als Ergebnis wird eine Tabelle mit Signalen erzeugt, die den konfigurierten Adressen zugeordnet sind und die in die Signaltabelle von *ibaPDA* kopiert werden. Die Tabelle steht unter dem Timer.

6 Support und Kontakt

Support

Tel.: +49 911 97282-14

E-Mail: support@iba-ag.com

Hinweis

Wenn Sie Support benötigen, dann geben Sie bitte bei Softwareprodukten die Nummer des Lizenzcontainers an. Bei Hardwareprodukten halten Sie bitte ggf. die Seriennummer des Geräts bereit.

Kontakt

Hausanschrift

iba AG
Königswarterstraße 44
90762 Fürth
Deutschland

Tel.: +49 911 97282-0

E-Mail: iba@iba-ag.com

Postanschrift

iba AG
Postfach 1828
90708 Fürth

Warenanlieferung, Retouren

iba AG
Gebhardtstraße 10
90762 Fürth

Regional und weltweit

Weitere Kontaktadressen unserer regionalen Niederlassungen oder Vertretungen finden Sie auf unserer Webseite:

www.iba-ag.com