Softwaredokumentation

ELA-COMM V3.5.8.1

elrest Automationssysteme GmbH
Leibnizstraße 10
73230 Kirchheim unter Teck
Germany
Telefon: +49 (0) 7021 / 92025-0
www.elrest.de

Impressum

elrest Automationssysteme GmbH

Leibnizstraße 10
73230 Kirchheim unter Teck
Germany

Tel.: + 49 (0) 7021 / 92025-0
Fax: + 49 (0) 7021 / 92025-29

E-Mail: vertrieb@elrest.de
Web: http://www.elrest.de

Technischer Support

Tel.: +49 (0) 7021 / 92025-33
Fax. +49 (0) 7021 / 92025-29
E-mail: support@elrest.de

Dieses Dokument wurde sorgfältig erstellt, um die Richtigkeit und Vollständigkeit der Dokumentation zu gewährleisten.
Da sich jedoch Fehler nie ausnahmslos vermeiden lassen, sind wir für ihre Anregungen und Mithilfe immer dankbar.

Inhaltsverzeichnis

1 Allgemein 5
1.1 Vorwort 5
1.2 Haftungsbedingungen 5
1.3 Sicherheitsrichtlinien und Schutzmaßnahmen 6
1.4 Copyright 6
1.5 Symbole 7
1.6 Gewährleistung 7
2 Übersicht der benötigten Komponenten 8
Komponente 1: Compiled Library 8
Komponente 2: Benutzerschnittstelle (Interface) 8
Komponente 3: Aufruf der Schnittstellen im Anwenderprogramm 8
Komponente 4: Systemevents (Aufruf in Taskkonfiguration) 8
3 Kommunikation Modbus 9
3.1 Generell zu Modbus 9
Funktionsbeschreibung 9
Übersicht der unterstützenden Funktionscodes 9
Anzahl der übertragbaren Coil & Register nach FunktionCode 10
Änderungen die zur Laufzeit erfolgen können: 11
Allgemeine Methoden 13
3.2 TCP Modbus 23
Allgemeines zum CLIENT 23
Generell zum SERVER 24
Anzahl der Kommunikationsinstanzen 25
Anzahl der Kommunikationsbefehle durch den Client 25
SetStatusRegister() SetStatusCoil() 26
Client Übertragungsfunktionen 28
ResetCounter() 33
Steuer Methoden auf dem Server 34
Informations- Methoden auf dem Server 36
Serverkonfiguration 37
Clientkonfiguration 41
Fehlermeldungen 44
3.3 RTU Modbus 46
Generell 46

Allgemeine Methoden Konfiguration 48
4 Kommunikation CAN- Layer2 55
4.1 Generelles zu Can- Layer2 55
Funktionsbeschreibung 55
5 Hilfe bei Störungen 60
5.1 Service und Support 60
6 Historie 61

1 Allgemein

1.1 Vorwort

Dieses Handbuch enthält Texte, Abbildungen und Erläuterungen zur korrekten Installation und Bedienung. Vor der Installation und dem Einsatz der Geräte muss dieses Handbuch gelesen und beachtet werden.

Es wendet sich ausschließlich an ausgebildete Fachkräfte der Steuerungs- und Automationstechnik. Diese müssen mit den aktuellen Normen und Richtlinien vertraut sein.

Bei Fragen zur Installation, Anwendung und Bedienung wenden sie sich bitte an die elrest-Kunden-Hotline:

Tel.:07021/92025-33

Fax:07021/92025-59

E-Mail: hotline@elrest.de

oder an Ihre zuständige Vertretung.

Dieses Handbuch wird vorbehaltlich etwaiger Änderungen herausgegeben. Änderungen können ohne Hinweis vorgenommen werden.

1.2 Haftungsbedingungen

Die Dokumentation wurde sorgfältig erstellt.

Alle Beispiele und Abbildungen in diesem Handbuch dienen nur als Hilfe zum Verstehen des Textes. Es können Änderungen ohne Hinweise vorgenommen werden. Für die Richtigkeit der dargestellten Bedienvorgänge kann keine Gewährleistung übernommen werden. An Hand von den Texten, Erläuterungen und Abbildungen in diesem Handbuch können keine Ansprüche auf schon gelieferte Produkte gemacht werden. elrest Automationssysteme GmbH übernimmt keine Verantwortung für eine Produktanwendung, die sich auf die dargestellten Beispiele (z.B. in eStudio Demo) bezieht.

elrest Automationssysteme GmbH übernimmt unter keinen Umständen die Haftung oder Verantwortung für Schäden, die aus einer unsachgemäßen Installation bzw. Anwendung der Geräte oder des Zubehörs entstanden sind.

Das Fachpersonal hat sicherzustellen, dass die Montage und die Anwendung der Produkte alle Sicherheitsanforderungen, Gesetzen, Bestimmungen und Normen entsprechen

Die nationalen Vorschriften und jeweils gültigen Sicherheitsbestimmungen sind zu beachten.

Eingriffe und Veränderungen an den Geräten führen zum Erlöschen der Gewährleistung.

1.3 Sicherheitsrichtlinien und Schutzmaßnahmen

Dieses Handbuch wurde für geschultes und kompetentes Personal erstellt. Die Qualifizierung wird durch die europäischen Richtlinien für Maschinen, Niederspannungen und EMV definiert. Bei Spannungen, die grösser als die Schutzkleinspannung sind, muss die Montage der Geräte durch eine Elektrofachkraft erfolgen.

Die nationalen Vorschriften und jeweils gültigen Sicherheitsbestimmungen sind zu beachten. Eingriffe und Veränderungen an den Geräten führen zum Erlöschen der Gewährleistung.

Aufgrund der großen Anzahl von verschiedenen Anwendungsmöglichkeiten dieser Geräte müssen sie die Anpassung für Ihren speziellen Anwendungsfall selbst vornehmen.

Wenn Schaltungskomponenten ausfallen sollten, müssen entsprechende Sicherheitseinrichtungen dafür sorgen, dass die angeschlossene Peripherie angehalten wird.

Versuchen sie nicht, die Geräte selbst zu reparieren oder elektrische Teile auszutauschen. Wenden sie sich hierfür ausschließlich an die elrest Service Abteilung. Kontakt können sie über den elrest- Support aufnehmen.

Beachten sie bei Installation und Einsatz der Geräte die lokalen und nationalen Normen und Vorschriften

Die einschlägigen Vorschriften (VDE etc.) beim Umgang mit elektrischen Anlagen sind zu beachten:

- Freischalten
- Gegen Wiedereinschalten sichern
- Spannungsfreiheit feststellen
- Erden und Kurzschließen
- Keine Erdschleifen
- Benachbarte unter Spannung stehende Teile sind abzudecken oder abzuschranken

1.4 Copyright

Die in diesem Handbuch erwähnten Produktnamen sind Warenzeichen oder eingetragene Warenzeichen der jeweiligen Herstellerfirmen und werden hiermit anerkannt.

Die in diesem Dokument enthaltenen Informationen können ohne Vorankündigung geändert werden und stellen keine Verpflichtung seitens elrest dar.

1.5 Symbole

In diesem Handbuch werden zur Hervorhebung von bestimmten Informationen verschiedene Symbole verwendet. Hiermit erhält das Bedienpersonal notwendige Hinweise zu den Sicherheits- und Schutzmaßnahmen. Bei jedem Auftreten der Symbole muss der zugehörige Hinweis gelesen werden

Bezeichnet eine möglicherweise auftretende Gefahr, die zu einem Personen- oder Sachschaden führen kann.

Bezeichnet Hinweise, damit die Handhabung einfacher wird.

HINWEIS

Wichtiger Hinweis!
Kennzeichnet eine mögliche Fehlfunktion, die aber keinen Sachschaden zur Folge hat, wenn sie nicht vermieden wird.

INFORMATION

Weitere Information
Weist auf weitere Informationen hin, die kein wesentlicher Bestandteil dieser Dokumentation sind (z. B. Internet).

1.6 Gewährleistung

Ein Gewährleistungsanspruch setzt eine fachgerechte Montage und Inbetriebnahme nach der für das Gerät gültigen Montage-, Inbetriebnahme- und Bedienungsanleitung voraus. Die erforderlichen Montage-, Inbetriebnahme- und Wartungsarbeiten dürfen nur von sachkundigen und autorisierten Personen durchgeführt werden. Sehen sie hierzu unsere EULA Bestimmungen.

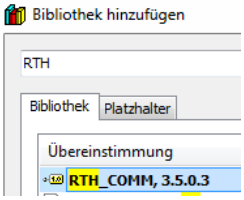
Hersteller	**Handelsmarke**	**Ursprungsland**
elrest Automationssysteme GmbH Leibnizstraße 10 D-73230 Kirchheim unter Teck		Germany

Telefon:+49 (0) 7021/92025-0
Fax: +49 (0) 7021/92025-29

2 Übersicht der benötigten Komponenten

Komponente 1: Compiled Library

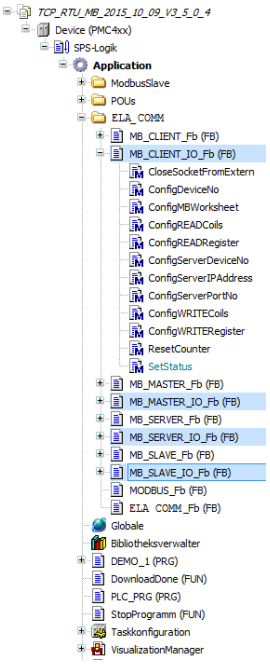
Die Compiled-Library erhalten Sie in der Regel als geschützte Bibliothek die einzubinden ist.



Quelle:
Compiled Library

Komponente 2: Benutzerschnittstelle (Interface)

Diese Schnittstelle vermittelt zwischen der Bibliothek und Ihrem Anwenderprogramm.

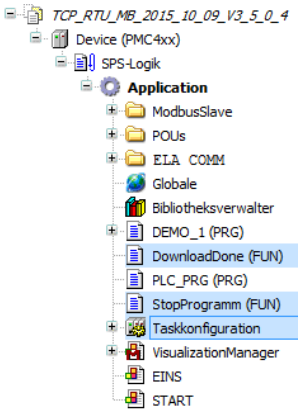


Quelle:
Kopieren aus Demo- Applikation

Komponente 3: Aufruf der Schnittstellen im Anwenderprogramm

Quelle:
Aufruf gemäß Handbuch, oder kopieren & anpassen aus Demoapplikation.

Komponente 4: Systemevents (Aufruf in Taskkonfiguration)



Quelle:
Anlegen gemäß Handbuch, oder kopieren & anpassen aus der Demoapplikation.

3 Kommunikation Modbus

3.1 Generell zu Modbus

*F*unktionsbeschreibung

Die ELA_COMM ab Version 3.5.0.4 erlaubt die Benutzung von:

- TCP Modbus als Client und Server, sowie
- RTU Modbus als Master und Server.

Der Modbus kann während der Laufzeit konfiguriert und geändert werden.

Übersicht der unterstützenden Funktionscodes

Ab der Version V3.5.0.2 werden folgende FC unterstützt:

[X] FC01 - READ COIL STATUS
[ ] FC02 - READ DISCRET INPUTS
[X] FC03 - READ HOLDING REGISTERS
[ ] FC04 - READ INPUT REGISTER
[X] FC05 - WRITE SINGLE COIL
[X] FC06 - WRITE SINGLE REGISTER
[ ] FC07 - READ EXEPTION STATUS (nur Seriell)
[ ] FC08 - DIAGNOSTICS (nur Seriell)
[ ] FC11 - GET COM EVENT COUNTER (nur Seriell)
[ ] FC12 - GET COM EVENT LOG (nur Seriell)
[X] FC15 - WRITE MULTIPLE COILS
[X] FC16 - WRITE MULTIPLE REGISTER
[ ] FC17 - REPORT SLAVE ID (nur Seriell)
[ ] FC20 - READ FILE RECORD
[ ] FC21 - WRITE FILE RECORD
[ ] FC22 - MASK WRITE REGISTER
[ ] FC23 - READ/WRITE MULTIPLE REGISTER
[ ] FC24 - READ FIFO QUEUE
[ ] FC43 - ENCAPSULATED INTERFACE TRANSPORT

Anzahl der übertragbaren Coil & Register nach FunktionCode

Slave(RTU)	Register	Anzahl
	FC3	125
	FC5	1
	FC16	123

	Coils	Anzahl
	FC1	2000
	FC6	1
	FC15	1968

Server(TCP)	Register	Anzahl
	FC3	125
	FC5	1
	FC16	123

	Coils	Anzahl
	FC1	2000
	FC6	1
	FC15	1968

Änderungen die zur Laufzeit erfolgen können:

- Anzahl der benutzten Kommunikationsinstanzen
- IP- Adresse & Portnummer
- Anzahl und Art der Kommunikationsbefehle inklusive der Polltime
- COM- Port und dessen Einstellungen
- Diagnosemöglichkeiten
- Teilnehmer aktivieren / deaktivieren

Wirkt sich ein Parameter auf die Socket- oder COMPort- Verbindung aus, so wird diese sofort geschlossen und mit den geänderten Parametern unverzüglich neu geöffnet.
Erfolgt dies an einem kommunizierenden Server, kann ein Reconnect durch den Client bis zu 35 Sekunden betragen, da dieser weiterhin versucht auf bisherigen Socketverbindung Daten zu übertragen.

Änderungen durch erneute Codeerzeugung

Erweitern der Maximalgrößen für:

- Maximale Anzahl von Worksheets
- Maximale Anzahl von Kommunikationsinstanzen

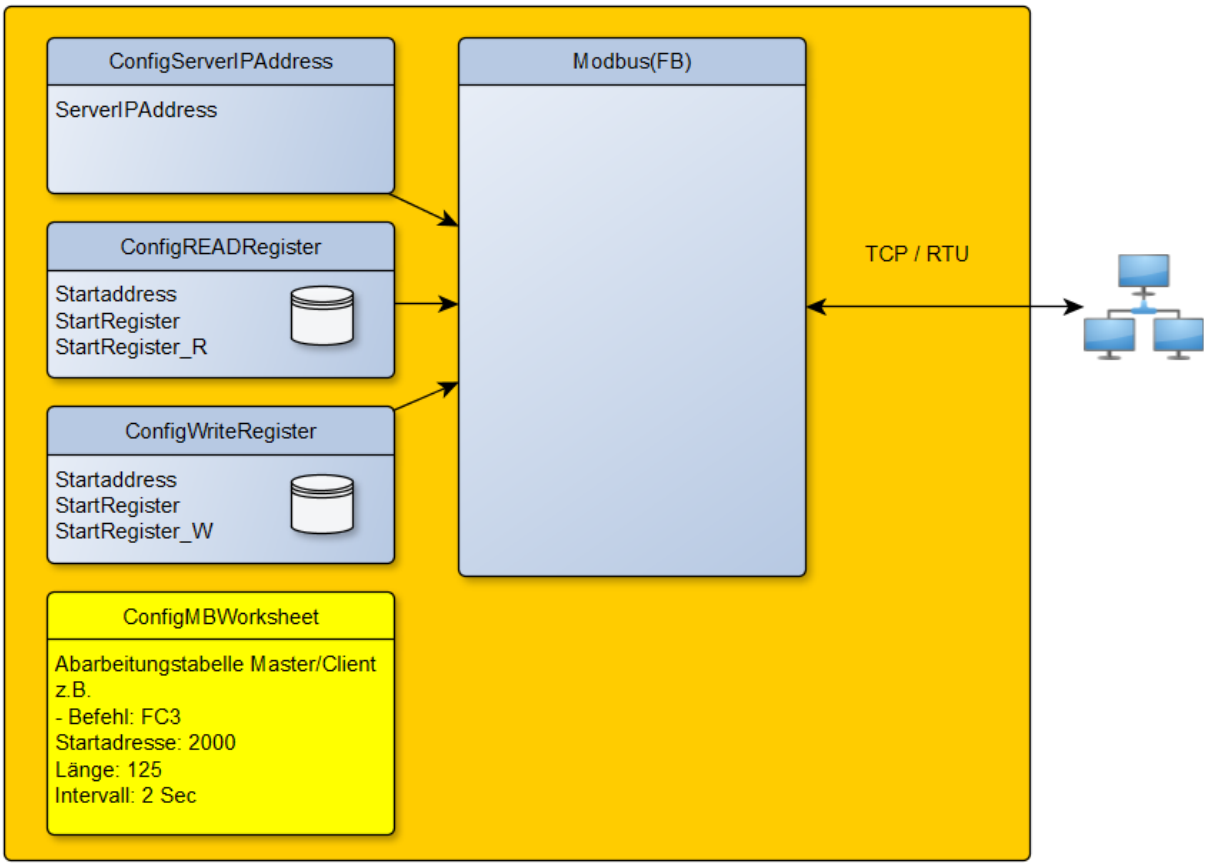
HINWEIS

Diese Maximaleinstellungen liegen als VAR_CONSTANT vor, und benötigen eine Recompile.
Darauf ist zu achten, dass die Konstante den maximalen Werten entspricht.

Übersicht der unterstützten Methoden

Methode	Slave	Master	Server	Client
CloseCOMPortFromExtern	X	X		
CloseSocketFromExtern			X	X
ConfigCOMSettings	X	X		
ConfigDeviceNo	X	X	X	X
ConfigReadCoils	X	X	X	X
ConfigReadRegister	X	X	X	X
ConfigWriteCoils	X	X	X	X
ConfigWriteRegister	X	X	X	X
ResetCounter	X	X	X	X
SetStatusCoil	X		X	
SetStatusRegister	X		X	
ShowDamageRecv	X	X		
ConfigMBWorksheet		X		X
ConfigMyIPAddress			X	
ConfigServerPortNo			X	X
ConfigServerIPAddress				X
GetClientIPAddress			X	

X = unterstützende Methode
orange hinterlegt = nur zur Informationsgewinnung



Allgemeine Methoden

ConfigDeviceNo()

Funktion:
Setzen der Stationsnummer

Geltungsbereich:
[X] Client/Master → Setzt die eigene Stationsnummer (Dummy)
[X] Server/Slave → Setzt die eigene Stationsnummer

Beispiel:
ELA_COMM.MODBUS.CLIENT[0].ConfigDeviceNo(DeviceNo:=1);

	Eigene Teilnehmernummer	**Fremde Teilnehmernummer**
Master	Nicht notwendig	notwendig
Slave	Notwendig	Nicht notwendig

Variable IN	**Typ**	**Bedeutung**
DeviceNo	BYTE	CLIENT / MASTER: Bei der Verwendung als Client oder Master gelten lediglich folgende beide Einstellungen: 0 → Kommunikation ist ausgeschalten >0 → Kommunikation ist ausgeschalten Die Devicenummer mit der Kommuniziert werden soll wird im Worksheet vorgegeben, bzw ist beim Client TCP- Basierend. SLAVE: Setzen der Stationsnummer (1..247) Die Stationsnummern 248.. 255 sind reserviert. SERVER: Der TCP-Modbus Server antwortet auf alle ServerDeviceNo, da die Adressierung tatsächlich über die IP-Adresse erfolgt. Wird Stationsnummer =0 gewählt, wird die Kommunikationsinstanz nicht abgearbeitet. → Disable

ConfigServerPortNo()

Funktion:
Wahl des Kommunikationsports

Geltungsbereich:
[X] Client → Angabe, über welchen Port sich der Client am Server anmeldet
[X] Server → Angabe, unter welchem Port der Server ansprechbar ist

Beispiel:
ELA_COMM.MODBUS.CLIENT[0].ConfigServerPortNo(PortNo := 502);

Variable IN	Typ	Bedeutung
PortNo	WORD	Port des zu kontaktierenden Servers. Wird kein Port angegeben gilt Port 502.

ConfigReadRegister()

Funktion:
Konfigurieren des Speicherbereichs für Read-Register (FC3)

Geltungsbereich:
[X] CLIENT / MASTER
[X] SERVER / SLAVE

Beispiel:

```
ELA_COMM.MODBUS.CLIENT[0].ConfigREADRegister (
        Startaddress    := ADR(R_REALS) ,
        StartRegister   := StartRegister_R,
        Size            := SIZEOF(R_REALS),
        ExtendedVar     := 1
);
```

Variable IN	**Typ**	**Bedeutung**
Startaddress	POINTER TO WORD	Adresse des angelegten Array Beispiel für ExtendedVar =0 *R_WORDS : ARRAY[0..MAXREGISTER] OF WORD;* *Eintrag hierfür:* *ADR(R_WORDS)* Beispiel für ExtendedVar =1 *R_REALS : ARRAY[0..MAXREGISTER] OF REAL;* *Eintrag hierfür:* *ADR(R_REALS)*
StartRegister	WORD	Adresse unter welcher das erste Register im Speicher angesprochen wird. Soll das erste Register im Speicher unter der Adresse 2000 angesprochen werden, so ist im Client Folgendes einzutragen: ELA_COMM.MODBUS.CLIENT[0].ConfigREADRegister (Startaddress := ADR(R_WORDS), StartRegister := **2000**, Size := SIZEOF(R_WORDS));
Size	WORD	Anzahl der Register z.B. SIZEOFF(R_WORDS)

<table>
<tr>
<td>ExtendedVar</td>
<td>Byte</td>
<td>Wahl des Quell- und Zielformat im Array:

0 = Modbus Standard
Der Wert wird als 16bit Signed gelesen, geschrieben und übertragen.

1 = REAL Formated
Der Wert wird als 32 bit REAL gelesen und geschrieben. Die Übertragung erfolgt, wie gewohnt als 16 bit Signed. Hierzu wird der Wert mit einem Scalingfaktor von 10 bearbeitet.

<u>Beispiel:</u>
Modbus Faktor Register
33 -- /10 -> 3.3
-282 <- *10 -- -28.2

Während der Modbus- Übertragung wird der Wert auf das Format INT begrenzt. Daher können nur Werte zwischen −32.768 und 32.767 übertragen werden.

2 = REAL (Experimental, nicht überall implementiert)
Der Wert wird als 32 bit REAL gelesen, geschrieben und übertragen.
Ein Wert belegt daher immer 2 16 bit-Register.</td>
</tr>
</table>

SIZEOF

Dieser arithmetische Operator ist nicht von der Norm IEC61131-3 vorgeschrieben.
Er kann verwendet werden, um die Anzahl Bytes zu bestimmen, die von der angegebenen Variablen *x* benötigt werden.
Der SIZEOF Operator liefert immer einen vorzeichenlosen Wert.

Der Typ der Rückgabevariablen paßt sich der gefundenen Größe von Variable x an.

Rückgabewert von SIZEOF(x)	Datentyp der Konstanten, die implizit für die gefundene Größe verwendet wird
0 <= Größe von x < 256	USINT
256 <= Größe von x < 65536	UINT
65536 <= Größe von x < 4294967296	UDINT
4294967296 <= Größe von x	ULINT

Die Registergröße sollte mit einer eigenen Variablen übergeben werden.

ConfigWriteRegister()

Funktion:
Konfigurieren des Speicherbereiches für Write-Register (FC6,FC16)

Geltungsbereich:
[X] CLIENT / MASTER
[X] SERVER / SLAVE

Beispiel:

```
ELA_COMM.MODBUS.CLIENT[0].ConfigWRITERegister(
Startaddress   := ADR(W_REALS),
StartRegister  := StartRegister_W,
Size           := SIZEOF(W_REALS),
ExtendedVar    := 1

);
```

Variable IN	**Typ**	**Bedeutung**
Startaddress	POINTER TO WORD	Adresse des angelegten Array Beispiel für ExtendedVar =0 *W_WORDS : ARRAY[0..MAXREGISTER] OF WORD;* *Eintrag hierfür:* *ADR(W_WORDS)* Beispiel für ExtendedVar =1 *W_REALS : ARRAY[0..MAXREGISTER] OF REAL;* *Eintrag hierfür:* *ADR(W_REALS)*
StartRegister	WORD	Adresse, unter der das erste Register im Speicher angesprochen wird. Soll das erste Register im Speicher unter der Adresse 2000 angesprochen werden, so ist Folgendes einzutragen. ELA_COMM.MODBUS.CLIENT[0].ConfigWRITERegister(Startaddress := ADR(W_REALS), StartRegister := 2000, Size := SIZEOF(W_REALS), ExtendedVar := 1);
Size	WORD	Anzahl der Register z.B. SIZEOFF(W_WORDS)

<table>
<tr>
<td>ExtendedVar</td>
<td>BYTE</td>
<td>Wahl des Quell- und Zielformat im Array.

0 = Modbus Standard
Der Wert wird als 16 bit signed gelesen, geschrieben und übertragen.

1 = REALFormated
Der Wert wird als 32 bit REAL gelesen und geschrieben. Die Übertragung erfolgt, wie gewohnt als 16 bit signed. Hierzu wird der Wert mit einem Scalingfaktor von 10 bearbeitet.

<u>Beispiel:</u>
Modbus Faktor Register
33 -- /10 -> 3.3
-282 <- *10 -- -28.2

Während der Modbus- Übertragung wird der Wert auf das Format INT begrenzt. Daher können nur Werte zwischen −32.768 und 32.767 übertragen werden.

2 = REAL (Experimental, nicht überall implementiert)
Der Wert wird als 32bit REAL gelesen, geschrieben und übertragen.
Ein Wert belegt daher immer zwei 16-bit-Register.</td>
</tr>
</table>

ConfigREADCoils()

Funktion:
Konfigurieren des Speicherbereiches für Read-Coils (FC1)

Geltungsbereich:
[X] CLIENT / MASTER
[X] SERVER / SLAVE

Beispiel:
```
ELA_COMM.MODBUS.MASTER[0].ConfigREADCoils(
Startaddress  := ADR(InputMemory),
StartRegister  := 0,
Size           := SIZEOF(InputMemory),
ExtendedVar  := 1
);
```

Variable IN	**Typ**	**Bedeutung**
Startaddress	POINTER TO WORD	Adresse des angelegten Datenarray Beispiel für ExtendedVar =0 *InputMemory : ARRAY [0..249] OF BYTE;* Eintrag hierfür: *ADR(InputMemory)* Beispiel für ExtendedVar =1 *InputbMemory : ARRAY [0..MAXREGISTER] OF BOOL;* Eintrag hierfür: *ADR(InputbMemory)*
StartRegister	WORD	Adresse unter welcher das erste Bit im Speicher angesprochen wird. Für ExtendedVar =0 müß der Wert immer Null sein. Für ExtendedVar =1 Angabe unter welcher Registeradresse das erste Feld im Array zur Verfügung gestellt werden soll.
Size	WORD	Größe des Speichers z.B. SIZEOFF(InputMemory)
ExtendedVar	BYTE	Neu ab Version V3.5.0.4 Übertragungsart: 0: Coils werden im Speicher aus Bits in einem Bytearray gelesen. 1: Coils werden im Speicher aus einem Array of BOOL gelesen.

ConfigWRITECoils()

Funktion:
Konfiguriere den Speicherbereich für Write-Coil (FC5,FC15)

Geltungsbereich:
[X] CLIENT / MASTER
[X] SERVER / SLAVE

Beispiel:

```
ELA_COMM.MODBUS.CLIENT[0].ConfigWRITECoils(
StartAddress        := ADR(OutputMemory),
StartRegister       := 0 ,
Size                := SIZEOF(OutputMemory),
ExtendedVar         := 1
);
```

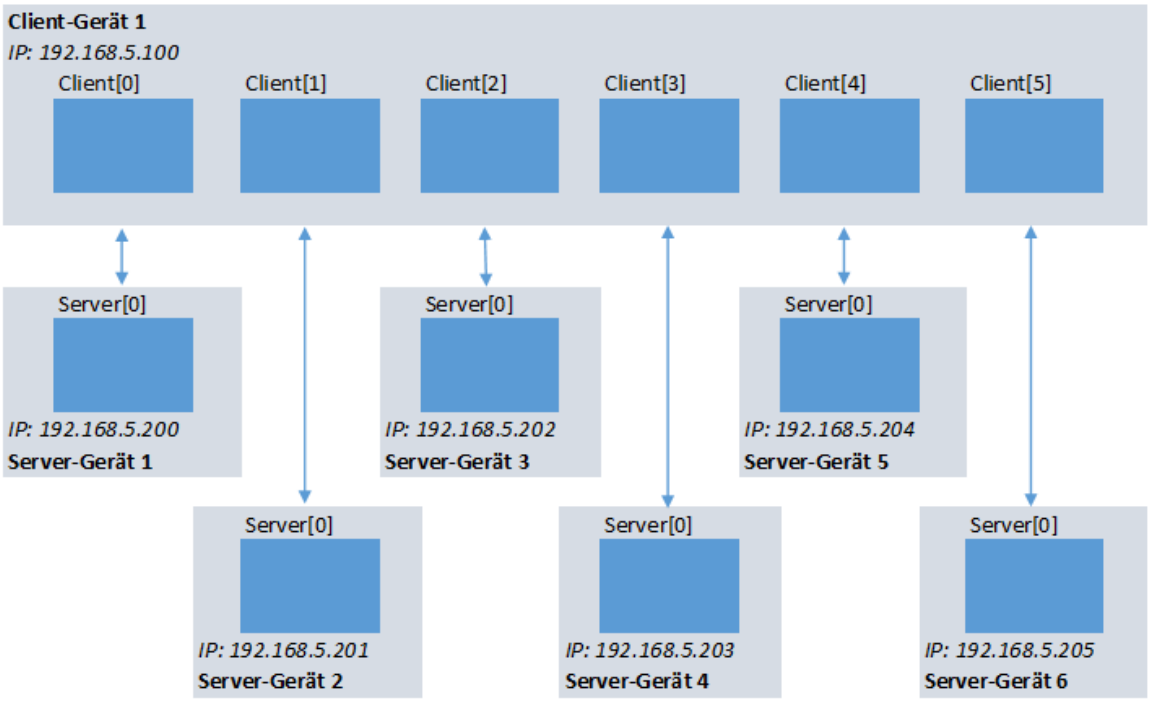
Variable IN	Typ	Bedeutung
StartAddress	POINTER TO WORD	Adresse des angelegten Datenarray Beispiel für ExtendedVar =0 *OutputMemory : ARRAY [0..249] OF BYTE;* *Eintrag hierfür*: *ADR(OutputMemory)* Beispiel für ExtendedVar =1 *OutputMemory : ARRAY [0..MAXREGISTER] OF BOOL;* *Eintrag hierfür*: *ADR(OutputMemory)*
StartRegister	WORD	Adresse unter welcher das erste Bit im Speicher angesprochen wird. Für ExtendedVar =0 müß der Wert immer Null sein. Für ExtendedVar =1 Angabe unter welcher Registeradresse das erste Feld im Array zur Verfügung gestellt werden soll.
Size	WORD	Größe des Speichers z.B. SIZEOFF(OutputMemory)
ExtendedVar	BYTE	Neu ab Version V3.5.0.4 Übertragungsart: 0: Coils werden im Speicher aus Bits in einem Bytearray gelesen. 1: Coils werden im Speicher aus einem Array of BOOL geschrieben.

3.2 TCP Modbus

Allgemeines zum CLIENT

Jede Verbindung läuft in einer eigenen Kommunikationsinstanz.
Dies gewährleistet, dass alle Verbindungen unabhängig voneinander kommunizieren können.
Wird kein Port vorgewählt, so gilt als Default der Port 502.
Es kann jedoch ein anderer Port gewählt werden, bzw. für jede Verbindung ein Individueller.

Projktierbeispiel:



Kommunikation jeweils Port 502

Verbindungs Timeout:
Erhält ein Client ab der Version V3.5.0.3 nach 35 Sekunden keine Antwort, so gilt der Server als „verloren“. Es ist daher notwendig innerhalb einer halben Minute mindestens 1x den Server mit einem beliebigen Telegramm anzusprechen.
Die Socketverbindung wird mit deaktiviertem Linger geschlossen. Das Schließen wirkt sofort.
Keine „toten“ Verbindungen bleiben auf der Steuerung erhalten.
Nun wird eine neue Socketverbindung zum Server geöffnet

HINWEIS

- Es muss sichergestellt sein, daß nicht benötigte Instanzen abgeschalten sind.
- Es muss vermieden werden, dass z.B. die Steuerung als Client auf sich selbst als Server zugreift.

HINWEIS

Jede weitere Kommunikationsinstanz muss sich von der Vorhergehenden entweder in der IP-Adresse oder der Portnummer unterscheiden, damit die Kommunikationen nicht kollidieren.

ConfigServerIPAddress()

Funktion:

Angabe zu welchem Server die Verbindung hergestellt werden soll.

Geltungsbereich:

[X] Client → Angabe zu welchem Server die Verbindung hergestellt werden soll.
[ ] Server

Beispiel:

```
ELA_COMM.MODBUS.CLIENT[0].ConfigServerIPAddress(ServerIPAddress :='192.168.5.201');
```

Variable IN	**Typ**	**Bedeutung**
ServerIPAddress	String(19)	IP Adresse des Servers zu welchem der Client die Verbindung aufbauen soll.
Variable OUT	**Typ**	**Bedeutung**
	BOOL	Nicht verwendet

Generell zum SERVER

Jede Server-Instanz kann bis zu 16 Verbindungen durch Clients bedienen.
Alle Anfragen können innerhalb eines PLC-Zyklus empfangen, bearbeitet und beantwortet werden.

Timeout:
Erhält der Server ab der Version V3.5.0.3 nach 65 Sekunden keine beliebige Anfrage, so gilt der Client als „verloren“. Es ist daher notwendig innerhalb einer Minute mindestens 1x den Server mit einem beliebigen Telegramm anzusprechen.
Der Socket wird mit deaktiviertem Linger geschlossen, das Schließen wird sofort wirksam und keine „toten“ Verbindungen bleiben auf der Steuerung erhalten.
Der Kanal steht unverzüglich für weitere Clientanfragen zur Verfügung.

Anzahl der Kommunikationsinstanzen

Die Anzahl der Kommunikationsinstanzen lässt sich durch die Konstanten MAXCLIENTS und MAXSERVERS im MODBUS_Fb abändern.

```
FUNCTION_BLOCK MODBUS_Fb
VAR_INPUT
END_VAR

VAR_OUTPUT
END_VAR

VAR
//CLIENT  : MB_CLIENT_IO_Fb;            // Für eine Server- Instanz
CLIENT  : ARRAY [0.. MAXCLIENTS-1] OF MB_CLIENT_IO_Fb;

// Für n- Server- Instanzen
//SERVER  : MB_SERVER_IO_FB;
SERVER  :  ARRAY [0.. MAXSERVERS-1] OF MB_SERVER_IO_FB;
          // Für n- Client- Instanzen
i:INT;
END_VAR

  VAR CONSTANT
    MAXCLIENTS       : INT:= 6;    // Anz. der zu kontaktierenden Servern / Anz. der Instanzen
    MAXSERVERS     : INT:= 6;  // Anzahl der Clients zu einem Server
  END_VAR
```

Device.Application.DEMO_1

Ausdruck	Datentyp
RTH_COMM	RTH_COMM_Fb
MODBUS	MODBUS_Fb
CLIENT	ARRAY [0..(MAXCLIENTS - 1)] OF MB_CLIENT_IO_Fb
CLIENT[0]	MB_CLIENT_IO_Fb
CLIENT[1]	MB_CLIENT_IO_Fb
CLIENT[2]	MB_CLIENT_IO_Fb
CLIENT[3]	MB_CLIENT_IO_Fb
CLIENT[4]	MB_CLIENT_IO_Fb
CLIENT[5]	MB_CLIENT_IO_Fb
SERVER	ARRAY [0..(MAXSERVERS - 1)] OF MB_SERVER_IO_FB
SERVER[0]	MB_SERVER_IO_FB
SERVER[1]	MB_SERVER_IO_FB
SERVER[2]	MB_SERVER_IO_FB
SERVER[3]	MB_SERVER_IO_FB
SERVER[4]	MB_SERVER_IO_FB
SERVER[5]	MB_SERVER_IO_FB

Anzahl der Kommunikationsbefehle durch den Client

Die Anzahl der möglichen Kommunikationsbefehle ist ab der Version 3.5.0.2 auf 100 begrenzt, kann aber selbst angepasst werden.

Generell können jedoch bis zu *n*- Befehle implementiert werden.
Die Anzahl wird vorgegeben durch die Konstante MAXWORKSHEET im Client- Kommunikations-FB.

Device.Application.DEMO_1.RTH_COMM.MODBUS.CLIENT[0].CLIENT_Itf

Ausdruck	Datentyp	Wert
DebugSendString	STRING(MAXSTRINGLENGTH)	'c0114:012-->00 72 00 00 00 06 01 01 00 01 07 D0 '
DebugRecvString	STRING(MAXSTRINGLENGTH)	'c0114:009-->00 72 00 00 00 03 01 81 01 '
DebugTimeDuration	TIME	T#0ms
DebugTimeStart	TIME	T#48m7s180ms
MBClientWorksheet	ARRAY [0..MAXWORKSHEET] OF DATA	
MBClientWorksheet[0]	DATA	
Activate	BOOL	TRUE
SlaveNo	BYTE	1
FC	TFUNCTIONCODE	FC03_READ_HOLDING_REGISTERS
Register	WORD	0
Length	WORD	123
Polltime_ms	WORD	1000
DoTransfer	BOOL	FALSE
tLastPoll	TIME	T#48m6s280ms
tLastRecv	TIME	T#48m6s300ms
CountAnswer	WORD	38
CountAnswerOld	WORD	38
ErrorCode	BYTE	0
MBClientWorksheet[1]	DATA	
MBClientWorksheet[2]	DATA	
MBClientWorksheet[3]	DATA	
MBClientWorksheet[4]	DATA	
MBClientWorksheet[5]	DATA	
MBClientWorksheet[6]	DATA	
MBClientWorksheet[7]	DATA	
MBClientWorksheet[8]	DATA	
MBClientWorksheet[9]	DATA	
MBClientWorksheet[10]	DATA	

SetStatusRegister()
SetStatusCoil()

Funktion:
Attribute zu den jeweiligen Registern.
Diese Funktion ist eine Erweiterung zur Coil- & Registerkonfiguration, und kann nur zusammen angewandt werden.
Sie gelten unter TCP und RTU gleicher Maßen.

Die Attribute sind nur wirksam bei Aktivierung von „REAL Formated Register“ bei Registern und „BOOL- ARRAY“ bei Coil.

Hierzu muß ExtendedVAR = 1 in
ConfigReadRegister() / ConfigWriteRegister(), bzw
ConfigReadCoil() / ConfigWriteCoil() gewählt werden.

Geltungsbereich:

[ ] CLIENT / MASTER → derzeit keine Verwendung.
[X] SERVER / SLAVE → Datenpunkt Gültig / Schreiben erlaubt

Beispiel:

```
ELA_COMM.MODBUS.SLAVE[0].SetStatusRegister(
                pzStatusRegister_R    := ADR(StatusRegister_R),
                pzStatusRegister_W    := ADR(StatusRegister_W),
                StartStatusRegister_R := StartRegister_R,
                StartStatusRegister_W := StartRegister_W,
                SizeStatusRegister_R  := SIZEOF(StatusRegister_R),
                SizeStatusRegister_W  := SIZEOF(StatusRegister_W)
                 );
```

Variable IN	Typ	Bedeutung
pzStatusRegister_R bzw. pzStatusCoil_R	POINTER_TO_BYTE	Übergabe der Adresse des Bytearray zum Setzen von Attributen. <u>Bit-Attribute im Byte</u>: B0 = ACTIVE (Datenpunkt gültig) Trifft ein Schreib- oder Lesetelegramm auf ein Register oder Coil, welches über KEINE gesetztes ACTIVE-BIT verfügt, wird die Anfrage mit einem Fehlertelegramm Code 2 = „Ungültiger Adresse“ beantwortet B1 = WRITABLE (Schreibbar) B2... B7 = Reserve Trifft ein Schreibtelegramm auf ein Register oder Coil, welches über KEINE Schreibberechtigung verfügt, wird dieses Register explizit NICHT beschrieben. In Folge wird ein gültiges Antworttelegramm versandt.
pzStatusRegister_W bzw. pzStatusCoil_W	POINTER_TO_BYTE	
StartStatusRegister_R bzw. StartStatusCoil_R StartStatusRegister_W bzw. StartStatusCoil_W	WORD	Adresse, unter der das erste Register oder Bit im Speicher angesprochen wird.
SizeStatusRegister_R bzw. SizeStatusCoil_R	DWORD	Größe des Speichers z.B. SIZEOF(StatusRegister_R),

Variable OUT	Typ	Bedeutung
	BOOL	Nicht verwendet

Client Übertragungsfunktionen

ConfigMBWorksheet()

Funktion:
Setzt Übertragungsbefehl

Geltungsbereich:
[X] CLIENT / MASTER
[ ] SERVER / SLAVE

Beispiel:

```
Result4[0]:=
ELA_COMM.MODBUS.CLIENT[3].ConfigMBWorksheet(
   ActionNo      := 0,Activate :=TRUE,
   SlaveNo       := 1, FC :=FC03_READ_HOLDING_REGISTERS,
   Register      := 0,
   Length        := 20,
   Polltime_ms :=Polltime1
);
```

Der Befehl muss nur einmal aufgerufen werden damit die Übertragung dauerhaft funktioniert. Möchte man den Rückgabewert in Erfahrung bringen, muss der Befehl zyklisch abgefragt werden.

Variable IN	Typ	Bedeutung
ActionNo	WORD	Aktionsnummer Frei einstellbar, jedoch begrenzt auf 100 Aktionen.
Activate	BOOL	Befehl aktivieren / deaktivieren
DeviceNo	BYTE	Nummer des anzufragenden Geräts
FC	INT	Funktionscode
Register	WORD	Nummer des ersten Registers
Length	WORD	Anzahl der Register / Anzahl der Coils

Polltime_ms	WORD	Abstand der Befehlsausführung in Millisekunden Da die Pollzeit auch für der TimeOut- Überwachung verwendet wird, ist daraufzu achten daß dieser Wert nicht zu klein ist. Die Pollzeit muß so groß gewählt werden, daß im Regelbetrieb alle anderen Anfragen ebenfalls erfolgen können. Die Minimal mögliche Pollzeit ist abhängig von: 1.) Geschwindigkeit der Antwort 2.) der Zykluszeit in welcher die ELA-COMM aufgerufen wird. 3.) Anzahl weiterer Aufrufe Dabei besitzt folgendes besonders Größen Einfluß: bei TCP: Antwortzeit des Servers Diese kann nach Hardware zwischen 16ms und 700ms variieren. bei RTU: Die Übertragungsgeschwindkeit (Baudrate)

Variable OUT	Typ	Bedeutung
ReturnValue	Byte	0 = OK 1 = ILLEGAL FUNCTION Funktionscode vom Server nicht unterstützt 2 = ILLEGAL DATA ADDRESS Illegale Adresse 3 = ILLEGAL DATA VALUE Illegaler Wert 4 = SLAVE DEVICE FAILURE Slave Gerätefehler 10 = GATEWAY PATH UNAVAILABLE Interner Fehler - Konnte Anfrage nicht senden 11 = GATEWAY TARGET DEVICE FAILED TO RESPOND externer Fehler keine Antwort von Gegenstelle 12 = CRC Fehler Erhält man auf eine Anfrage nach Ablauf der Zeit Polltime_ms keine Antwort, so wird die Anfrage 1x wiederholt. Nach Ablauf der Polltime + 3 Sekunden wird ein Fehler ausgelöst.

Beispiel eines einmaligen Senden

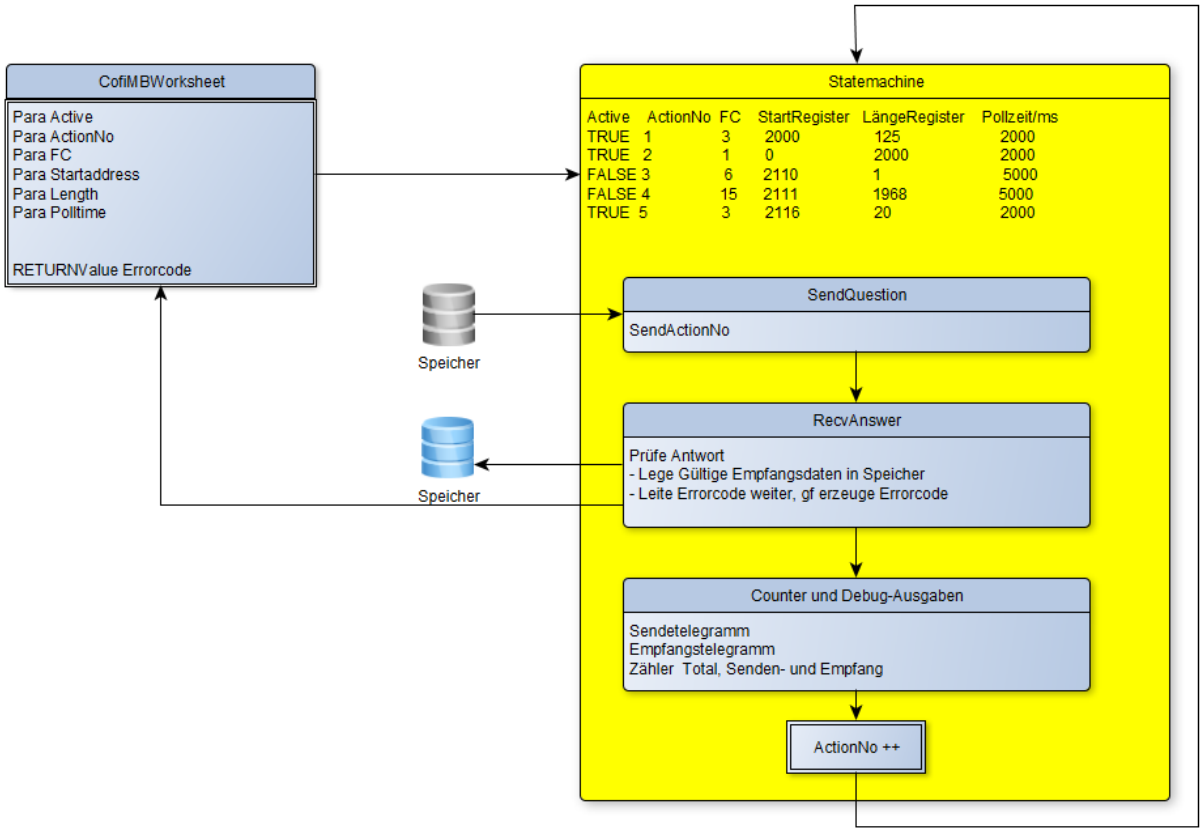
Funktion:

In nachfolgendem Codebeispiel wird von extern das Activ- Bit gesetzt.
Nachdem im Worksheet für diese Aktion das Antworttelegramm eingegangen ist, wird das Activ- Bit zurückgesetzt.

Vorteil dieser „einmaliger Übertragung“ gegenüber etwaiger anderen Methoden:
Das Fehlerhandling und die Rückgaberoutine arbeitet korrekt, da eine für das Fehlermanagement korrekte Pollzeit übergeben und die Server- bzw Slave- Antwort abgewartet wird.

Bedingung:
- Ab Version 3.5.0.5
- Zum Teilnehmer muß denoch zyklisch irgendein Telegramm gesendet werden, da bei mehr als 35 Sekunde inaktivität der Teilnehmer als „verloren“ gewertet wird.

```
// Für einmaliges Senden. Activ wird zurückgesetzt, wenn zum Sendetelegramm die Antwort vorliegt.
// Dies erlaubt die Auswertung von Fehlercodes. Im Beispiele hier Fehlercode 2 = "Illegal Address"
Result[4] 2 := ELA_COMM.MODBUS.MASTER[0].ConfigMBWorksheet(  ActionNo    := 4,
                                                             Activate    := Activ FALSE ,
                                                             DeviceNo    := 1,
                                                             FC          := 15,
                                                             Register    := RegisterStart 0 ,
                                                             Length      := DummyCoilLength 8 ,
                                                             Polltime_ms := Polltime1 600
                                                            );
IF ELA_COMM.MODBUS.MASTER[0].Master_Itf.MBWorksheet[4].CountAnswer 4 <>
   ELA_COMM.MODBUS.MASTER[0].Master_Itf.MBWorksheet[4].CountAnswerOld 4
  THEN
  Activ FALSE :=FALSE;
END_IF
```


CofiMBWorksheet
Para Active
Para ActionNo
Para FC
Para Startaddress
Para Length
Para Polltime
RETURNValue Errorcode
Statemachine
Active ActionNo FC StartRegister LängeRegister Pollzeit/ms
TRUE 1 3 2000 125 2000
TRUE 2 1 0 2000 2000
FALSE 3 6 2110 1 5000
FALSE 4 15 2111 1968 5000
TRUE 5 3 2116 20 2000
Speicher
SendQuestion
SendActionNo
Speicher
RecvAnswer
Prüfe Antwort
- Lege Gültige Empfangsdaten in Speicher
- Leite Errorcode weiter, gf erzeuge Errorcode
Counter und Debug-Ausgaben
Sendetelegramm
Empfangstelegramm
Zähler Total, Senden- und Empfang
ActionNo ++

ResetCounter()

Funktion:
Die Zähler der Debug-Stringausgaben werden auf Null gesetzt.

Geltungsbereich:
[X] CLIENT / MASTER
[X] SERVER / SLAVE

Beispiel:
ELA_COMM.MODBUS.SLAVE[0].ResetCounter(Reset);

Variable IN	Typ	Bedeutung
Reset	Bool	Die Zähler der Debug-Stringausgaben werden auf Null gesetzt. Es empfieht sich Reset für einen Zyklus anstehen zu lassen. Dies bedeutet, daß eine Zeile später der Reset durch *Reset:=FALSE;* bereits wieder zurückgesetzt werden kann.

Steuer Methoden auf dem Server

ConfigMyIPAddress ()

Funktion:
Setzt die zu verwendende IP-Adresse

Geräte mit einer Ethernet-Schnittstelle:
Wird als IP-Adresse 0.0.0.0 angegeben, oder die Funktion nicht aufgerufen, so ermittelt das Gerät selbst seine eigene IP-Adresse.

Geräte mit mehreren Ethernet-Schnittstellen:
Geben Sie für Jede KommunikationsInstanz an auf welche IP-Adresse der Server antworten soll.

Geltungsbereich:
[ ] Client
[X] Server

Beispiel:
ELA_COMM.MODBUS.SERVER[0].ConfigMyIPAddress('192.168.5.230');

Variable IN	**Typ**	**Bedeutung**
MyIPAddress	String(19)	Besitzt die Steuerung mehrere LAN-Schnittstellen ist hier anzugeben welche IP- Adresse benutzt werden soll. Bei Geräten mit einer LAN-Schnittstelle kann hier entweder: 1.) Die eigene IP-Adresse eingetragen, 2.) 0.0.0.0 zur Auto-Erkennung, oder 3.) der Befehl nicht ausgeführt.

Variable OUT	**Typ**	**Bedeutung**
	BOOL	Nicht verwendet

Besitzt ein Gerät mehrere LAN-Adapter, können sie zuvor anhand des Adapternamen die IP-Adresse in Erfahrung bringen und diese dann zuweisen.

```
/// + ========================================================================
/// + Abfrage LAN-Adapter
/// +
/// +
/// + ========================================================================

//SearchName:='EMACB1'; // LAN0 an CM211
//SearchName:='eth0';   // LAN0 an RMC5xx
  SearchName 'eth1' :='eth1';   // LAN0 an RMC5xx

instGetIPAdressfromAdaperName(
  SearchName 'eth1'       := SearchName 'eth1',
  StartSearch TRUE      := StartSearch TRUE,
  AdapterFound TRUE     => AdapterFound TRUE,
  IPAdress '192.168.2.'         => IPAdressDetected '192.168.2.',
  AdaptersNoTotal 4 => NoOfLANAdapter 4 ,
  AdapterNames     => LANAdapterName ,
  AdapterIPs       => LANAdapterIP
  );

ELA_COMM.MODBUS.SERVER[0].ConfigMyIPAddress(IPAdressDetected '192.168.2.');
                                             '192.168.2.254'
// Debug
LANAdapterName[0] 'lo'   ; LANAdapterIP[0] '127.0.0.1' ;
LANAdapterName[1] 'eth0' ; LANAdapterIP[1] '192.168.5.' ;
LANAdapterName[2] 'sit0' ; LANAdapterIP[2] '0.0.0.0' ;
LANAdapterName[3] 'eth1' ; LANAdapterIP[3] '192.168.2.' ;
LANAdapterName[4] ''     ; LANAdapterIP[4] '' ;
```

Informations- Methoden auf dem Server

GetClientIPAddress()

Funktion:
Erhalten des IP Adresse von dem angemeldeten Client am Server

Sendet ein Client keine aktive Anfragen mehr, bleibt die Socketverbindung zu ihm noch 65 Sekunden geöffnet, ehe sie geschlossen und die Adresse „0.0.0.0“ angezeigt wird.

Geltungsbereich:
[ ] Client
[X] Server

Beispiel:
ShowClientIPAddress[0]:=ELA_COMM.MODBUS.SERVER[0].GetClientIPAddress();

Variable IN	**Typ**	**Bedeutung**
Connection	BYTE	Fortlaufende Nummer der Verbindung.

Variable OUT	**Typ**	**Bedeutung**
ReturnValue	String(19)	IP-Adresse des am Server angemeldeten Clients. Ist kein Client angemeldet, erhält man die IP 0.0.0.0

Serverkonfiguration

Der Server verfügt über keine speziellen Steuerungsfunktionen.
Es muss lediglich angegeben werden:

- Stationskennung (DeviceNo)
- Portnummer
- Konfiguration des Speicherbereichs.

Die Funktionsweise, ***wie, wann und wo*** Daten abgeholt, bzw. geschrieben werden, wird durch den Client vorgegeben.

Beispiel Deklaration und Code

Deklaration

```
VAR
  ELA_COMM              : ELA_COMM_Fb;
  // ExtendedVar = 0  (Modbus Standard) gewählt wurde
  R_WORDS               : ARRAY[0..MAXREGISTER] OF WORD;
  W_WORDS               : ARRAY[0..MAXREGISTER] OF WORD;

  // ExtendedVar = 0  (Modbus Standard) gewählt wurde
  R_REALS                : ARRAY[0..MAXREGISTER] OF REAL;
  W_REALS                : ARRAY[0..MAXREGISTER] OF REAL;

  StartRegister_R       : WORD:= 0;
  StartRegister_W       : WORD:= 0;
  LENGTH_R              : WORD:= 16#FFFF;
  LENGTH_W              : WORD:= 16#FFFF;
  InputMemory           : ARRAY [0..249] OF BYTE;                 // Simulate INPUT-    VARIABLES for coil communication, ExtendedVAR =0;
  OutputMemory          : ARRAY [0..249] OF BYTE;                 // Simulate OUTPUT- VARIABLES for coil communication, ExtendedVAR =0;
  bInputMemory          : ARRAY [0..MAXREGISTER] OF BOOL;  // Simulate INPUT-    VARIABLES for coil communication, ExtendedVAR =1;
  bOutputMemory         : ARRAY [0..MAXREGISTER] OF BOOL;  // Simulate OUTPUT- VARIABLES for coil communication, ExtendedVAR =1;


  Writable              : ARRAY[0..MAXREGISTER] OF BOOL;
  Datapointlist_R       : ARRAY[0..MAXREGISTER] OF BOOL;
  Datapointlist_W       : ARRAY[0..MAXREGISTER] OF BOOL;

END_VAR
VAR CONSTANT
  MAXREGISTER       : WORD:= 16#FFFF;
END_VAR
```

Code

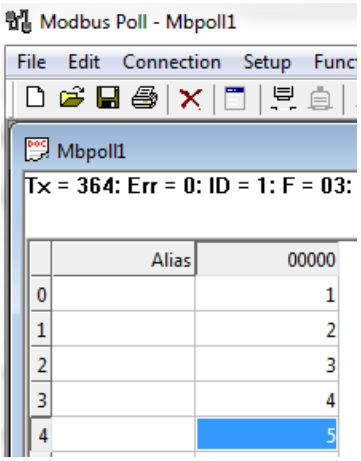
```
// ===============================================================================================
// MODBUS- SERVER
// --> Daten werden zur Verfügung gestellt
// INFO: Die Angaben müssen nicht zyklisch erfolgen. Der einmalige Aufruf genügt.
// ===============================================================================================
// Konfiguration  SERVER #1
FBVersion:=ELA_COMM.MODBUS.SERVER[0].Server_Itf.FBVersion;
ELA_COMM.MODBUS.SERVER[0].ConfigDeviceNo(DeviceNo:=DeviceNo);
ELA_COMM.MODBUS.SERVER[0].ConfigServerPortNo(PortNo:= TestPortNo );
ELA_COMM.MODBUS.SERVER[0].ConfigREADRegister (  Startaddress :=ADR(R_REALS),
                                                StartRegister  :=StartRegister_R,
                                                Size:= SIZEOF(R_REALS) ,
                                                ExtendedVar:= 1
                                               );
ELA_COMM.MODBUS.SERVER[0].ConfigWRITERegister(  Startaddress :=ADR(W_REALS),
                                                StartRegister  :=StartRegister_W,
                                                Size:= SIZEOF(W_REALS) ,
                                                ExtendedVar:= 1
                                               );

ELA_COMM.MODBUS.SERVER[0].SetStatusRegister(    pzStatusRegister_R    := ADR(StatusRegister_R),
                                                pzStatusRegister_W    := ADR(StatusRegister_W),
                                                StartStatusRegister_R := StartRegister_R,
                                                StartStatusRegister_W := StartRegister_W,
                                                SizeStatusRegister_R  := SIZEOF(StatusRegister_R),
                                                SizeStatusRegister_W  := SIZEOF(StatusRegister_W)
                                               );
ELA_COMM.MODBUS.SERVER[0].ConfigREADCoils(      Startaddress :=ADR(bInputMemory),
                                                StartRegister  :=0,
                                                Size:= SIZEOF(bInputMemory),
                                                ExtendedVar:= 1
                                                );

ELA_COMM.MODBUS.SERVER[0].ConfigWRITECoils(     Startaddress :=ADR(bOutputMemory),
                                                StartRegister :=0 ,
                                                Size:= SIZEOF(bOutputMemory),
                                                ExtendedVar:= 1
                                                );
ELA_COMM.MODBUS.SERVER[0].SetStatusCoil(        pzStatusCoil_R    := ADR(StatusCoil_R),
                                                pzStatusCoil_W    := ADR(StatusCoil_W),
                                                StartStatusCoil_R := 0,
                                                StartStatusCoil_W := 0,
                                                SizeStatusCoil_R  := SIZEOF(StatusCoil_R),
                                                SizeStatusCoil_W  := SIZEOF(StatusCoil_W)
                                                );
DebugRecvString:= ELA_COMM.MODBUS.SERVER[0].Server_Itf.DebugRecvString;
DebugSendString:= ELA_COMM.MODBUS.SERVER[0].Server_Itf.DebugSendString;

ShowClientIPAddress[ 0]:=ELA_COMM.MODBUS.SERVER[0].GetClientIPAddress(Connection:= 0);
ShowClientIPAddress[ 1]:=ELA_COMM.MODBUS.SERVER[0].GetClientIPAddress(Connection:= 1);
ShowClientIPAddress[ 2]:=ELA_COMM.MODBUS.SERVER[0].GetClientIPAddress(Connection:= 2);
ShowClientIPAddress[ 4]:=ELA_COMM.MODBUS.SERVER[0].GetClientIPAddress(Connection:= 4);
ShowClientIPAddress[ 5]:=ELA_COMM.MODBUS.SERVER[0].GetClientIPAddress(Connection:= 5);
ShowClientIPAddress[ 6]:=ELA_COMM.MODBUS.SERVER[0].GetClientIPAddress(Connection:= 6);
ShowClientIPAddress[ 7]:=ELA_COMM.MODBUS.SERVER[0].GetClientIPAddress(Connection:= 7);
ShowClientIPAddress[ 8]:=ELA_COMM.MODBUS.SERVER[0].GetClientIPAddress(Connection:= 8);
ShowClientIPAddress[ 9]:=ELA_COMM.MODBUS.SERVER[0].GetClientIPAddress(Connection:= 9);
ShowClientIPAddress[10]:=ELA_COMM.MODBUS.SERVER[0].GetClientIPAddress(Connection:= 10);
ShowClientIPAddress[11]:=ELA_COMM.MODBUS.SERVER[0].GetClientIPAddress(Connection:= 11);
ShowClientIPAddress[12]:=ELA_COMM.MODBUS.SERVER[0].GetClientIPAddress(Connection:= 12);
ShowClientIPAddress[13]:=ELA_COMM.MODBUS.SERVER[0].GetClientIPAddress(Connection:= 13);
ShowClientIPAddress[14]:=ELA_COMM.MODBUS.SERVER[0].GetClientIPAddress(Connection:= 14);
ShowClientIPAddress[15]:=ELA_COMM.MODBUS.SERVER[0].GetClientIPAddress(Connection:= 15);

// ===============================================================================================
//   MODBUS- AUFRUF
// Info:  Der Bausteinaufruf sollte NACH der Konfiguration erfolgen, da sonst z.B.
//        die Socket-Verbindung ohne Gültige IP-Adresse geöfnet wird.
// ===============================================================================================

ELA_COMM(); // MODBUS-AUFRUF
```

Screenshot - Applikation als Server

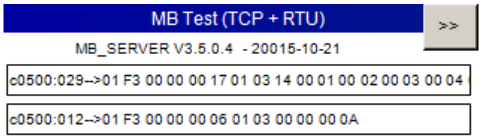
Datenbereitstellung

R_REALS	ARRAY [0..MAXREGISTER] OF R...	
R_REALS[0]	REAL	0.1
R_REALS[1]	REAL	0.2
R_REALS[2]	REAL	0.3
R_REALS[3]	REAL	0.4
R_REALS[4]	REAL	0.5

Datenabholung durch PC-Client



HMI



Parametrierungsbeispiel

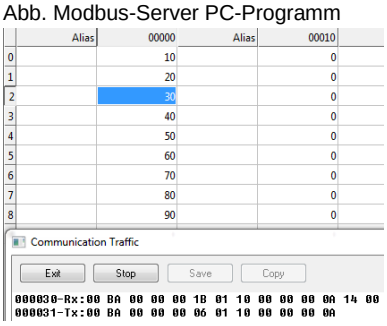
```
// ============================================================================================================
// MODBUS- SERVER
// --> Daten werden zur Verfügung gestellt
// INFO: Die Angaben müssen nicht zyklisch erfolgen. Der einmalige Aufruf genügt.
// ============================================================================================================

// Konfiguration  SERVER #1
FBVersion 'MB_SERVER' :=ELA_COMM.MODBUS.SERVER[0].Server_Itf.FBVersion 'MB_SERVER';

//RTH_COMM.MODBUS.SERVER[0].ConfigServerDeviceNo(ServerDeviceNo:=1);
ELA_COMM.MODBUS.SERVER[0].ConfigDeviceNo(DeviceNo:=1);
ELA_COMM.MODBUS.SERVER[0].ConfigServerPortNo(PortNo:= TestPortNo 502 );
//RTH_COMM.MODBUS.SERVER[0].ConfigMyIPAddress('192.168.5.230');

ELA_COMM.MODBUS.SERVER[0].ConfigREADRegister (Startaddress :=ADR(R_REALS)    ,StartRegister  :=StartRegister_R 0 , Size:= SIZEOF(R_REALS) , ExtendedVar:= 1);
ELA_COMM.MODBUS.SERVER[0].ConfigWRITERegister(Startaddress :=ADR(W_REALS)    ,StartRegister  :=StartRegister_W 0 , Size:= SIZEOF(W_REALS) , ExtendedVar:= 1);

ELA_COMM.MODBUS.SERVER[0].SetStatusRegister(
                                    pzStatusRegister_R    := ADR(StatusRegister_R),
                                    pzStatusRegister_W    := ADR(StatusRegister_W),
                                    StartStatusRegister_R := Star  VAR DEMO_1.StatusRegister_W : ARRAY [0..MAXREGISTER] OF BYTE
                                    StartStatusRegister_W := StartRegister_W 0 ,
                                    SizeStatusRegister_R  := SIZEOF(StatusRegister_R),
                                    SizeStatusRegister_W  := SIZEOF(StatusRegister_W)
                                    );

ELA_COMM.MODBUS.SERVER[0].ConfigREADCoils   (Startaddress :=ADR(bInputMemory),StartRegister  :=0 , Size:= SIZEOF(bInputMemory),ExtendedVar:= 1);
ELA_COMM.MODBUS.SERVER[0].ConfigWRITECoils  (Startaddress :=ADR(bOutputMemory),StartRegister :=0 , Size:= SIZEOF(bOutputMemory),ExtendedVar:= 1);

ELA_COMM.MODBUS.SERVER[0].SetStatusCoil(
                                    pzStatusCoil_R    := ADR(StatusCoil_R),
                                    pzStatusCoil_W    := ADR(StatusCoil_W),
                                    StartStatusCoil_R := 0,
                                    StartStatusCoil_W := 0,
                                    SizeStatusCoil_R  := SIZEOF(StatusCoil_R),
                                    SizeStatusCoil_W  := SIZEOF(StatusCoil_W)
                                    );

DebugRecvString 'c0146:012-' := ELA_COMM.MODBUS.SERVER[0].Server_Itf.DebugRecvString 'c0146:012-';
DebugSendString 'c0146:029-' := ELA_COMM.MODBUS.SERVER[0].Server_Itf.DebugSendString 'c0146:029-';

ShowClientIPAddress[0] '192.168.5.' :=ELA_COMM.MODBUS.SERVER[0].GetClientIPAddress(Connection:= 0);
ShowClientIPAddress[1] '0.0.0.0' :=ELA_COMM.MODBUS.SERVER[0].GetClientIPAddress(Connection:= 1);
ShowClientIPAddress[2] '0.0.0.0' :=ELA_COMM.MODBUS.SERVER[0].GetClientIPAddress(Connection:= 2);
ShowClientIPAddress[4] '0.0.0.0' :=ELA_COMM.MODBUS.SERVER[0].GetClientIPAddress(Connection:= 4);
ShowClientIPAddress[5] '0.0.0.0' :=ELA_COMM.MODBUS.SERVER[0].GetClientIPAddress(Connection:= 5);
ShowClientIPAddress[6] '0.0.0.0' :=ELA_COMM.MODBUS.SERVER[0].GetClientIPAddress(Connection:= 6);
ShowClientIPAddress[7] '0.0.0.0' :=ELA_COMM.MODBUS.SERVER[0].GetClientIPAddress(Connection:= 7);
```

Screenshot – PC Software als Client

Abb. Screenshot FB-Client Instanz

ServerIPAddress	STRING	'192.168.5.201'
PortNo	WORD	502
Timeout	WORD	1000
sInfo	STRING(80)	'Status = RUN'
FBVersion	STRING(80)	'MB_CLIENT V3.5.0.4 - 20015-10-12'
DEBUG	BOOL	TRUE
DebugSendString	STRING(MAXSTRINGLENGTH)	'c0092:033-->00 5C 00 00 00 1B 01 10 00 00 00 0A 14 00 0A 00 14 00 1E 00 28 00 32 00 3C 00 46 00 50 00 5A 00 00
DebugRecvString	STRING(MAXSTRINGLENGTH)	'c0092:012-->00 5C 00 00 00 06 01 10 00 00 00 0A '
DebugTimeDuration	TIME	T#10ms
DebugTimeStart	TIME	T#7h48m55s979ms
MBWorksheet	ARRAY [0..MAXWORKSHEET] OF...	
DeviceNo	BYTE	1

Abb. Modbus-Server PC-Programm



Clientkonfiguration

Beispiel Deklaration und Code

Deklaration

```
VAR
  ELA_COMM            : ELA_COMM_Fb;
  // ExtendedVar = 0  (Modbus STANDARD) gewählt wurde
  R_WORDS             : ARRAY[0..MAXREGISTER] OF WORD;
  W_WORDS             : ARRAY[0..MAXREGISTER] OF WORD;

  // ExtendedVar = 1  (Modbus REAL FORMATED) gewählt wurde
  R_REALS             : ARRAY[0..MAXREGISTER] OF REAL;
  W_REALS             : ARRAY[0..MAXREGISTER] OF REAL;

  StartRegister_R     : WORD:= 0;
  StartRegister_W     : WORD:= 0;
  LENGTH_R            : WORD:= 16#FFFF;
  LENGTH_W            : WORD:= 16#FFFF;

  // ExtendedVar = 0  (Modbus STANDARD) gewählt wurde
  InputMemory         : ARRAY [0..249] OF BYTE; // Simulate INPUT-    VARIABLES for coil communication, Extended Var = 0;
  OutputMemory        : ARRAY [0..249] OF BYTE; // Simulate OUTPUT- VARIABLES for coil communication, Extended Var = 0;

  // ExtendedVar = 1  (Modbus BOOLARRAY) gewählt wurde
  bInputMemory        : ARRAY [0..MAXREGISTER] OF BOOL; // Simulate INPUT-    VARIABLES for coil communication, Extended Var = 1;
  bOutputMemory       : ARRAY [0..MAXREGISTER] OF BOOL; // Simulate OUTPUT- VARIABLES for coil communication, Extended Var = 1;


  Writable            : ARRAY[0..MAXREGISTER] OF BOOL;
  Datapointlist_R     : ARRAY[0..MAXREGISTER] OF BOOL;
  Datapointlist_W     : ARRAY[0..MAXREGISTER] OF BOOL;

END_VAR
VAR CONSTANT
  MAXREGISTER     : WORD:= 16#FFFF;   // Anzahl der Maximalregister. Der Wert sollte auf den tatsächlichen Bedarf angepasst werden.
END_VAR
```

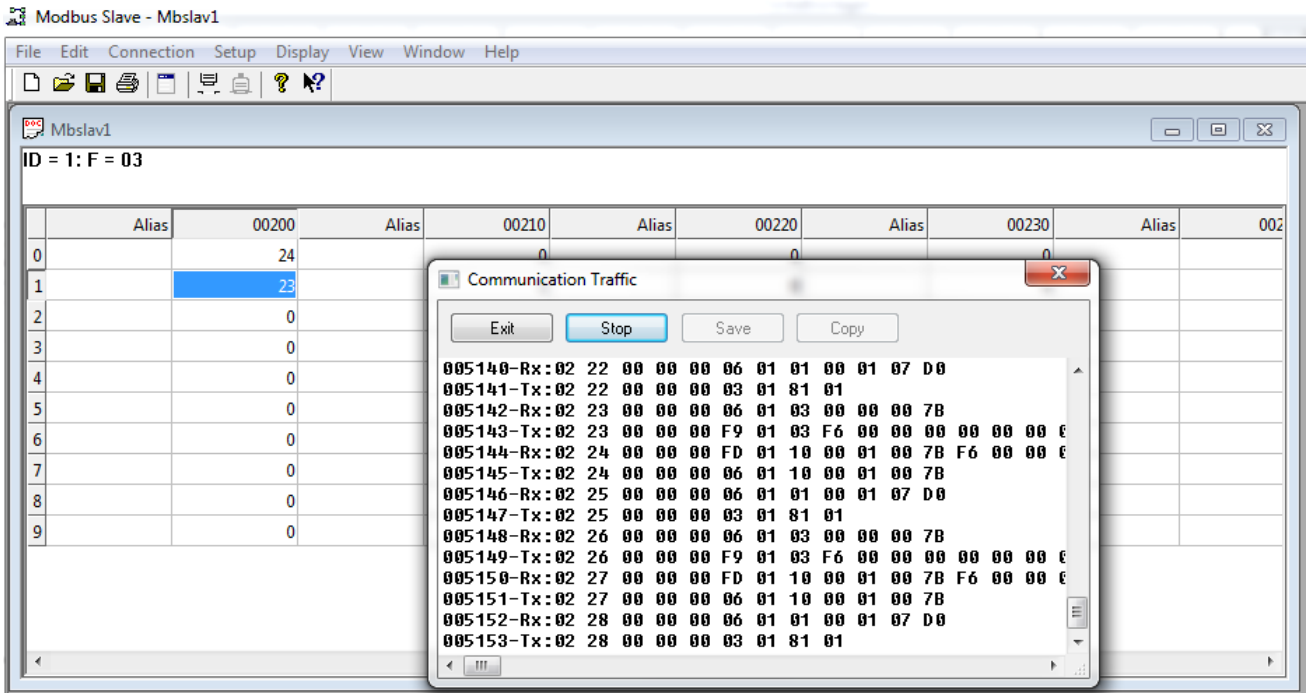
Code

```
// ===============================================================================================
// MODBUS- CLIENT
// --> Ruft Daten ab
// INFO: Die Angaben müssen nicht zyklisch erfolgen. Der einmalige Aufruf genügt.
// ===============================================================================================

//Konfiguration
// CLIENT INSTANZ #1
FBVersion:=ELA_COMM.MODBUS.CLIENT[0].Client_Itf.FBVersion;   // FB-Name der HMI bereitstellen
ELA_COMM.MODBUS.CLIENT[0].ConfigServerIPAddress(ServerIPAddress :='192.168.5.201');
ELA_COMM.MODBUS.CLIENT[0].ConfigServerPortNo(PortNo := 502 );
ELA_COMM.MODBUS.CLIENT[0].ConfigServerDeviceNo(ServerDeviceNo:=1);

ELA_COMM.MODBUS.CLIENT[0].ConfigREADRegister (Startaddress :=ADR(R_REALS)    ,StartRegister  :=StartRegister_R, Size:= SIZEOF(R_REALS), ExtendedVar:= 1);
ELA_COMM.MODBUS.CLIENT[0].ConfigWRITERegister(Startaddress :=ADR(W_REALS)    ,StartRegister  :=StartRegister_W, Size:= SIZEOF(W_REALS), ExtendedVar:= 1);

ELA_COMM.MODBUS.CLIENT[0].ConfigREADCoils    (Startaddress :=ADR(bInputMemory) ,StartRegister :=4    , Size:= SIZEOF(bInputMemory) ,ExtendedVar:=1);
ELA_COMM.MODBUS.CLIENT[0].ConfigWRITECoils   (Startaddress :=ADR(bOutputMemory),StartRegister :=4    , Size:= SIZEOF(bOutputMemory),ExtendedVar:=1);

//Datenabruf
Result[0]:= ELA_COMM.MODBUS.CLIENT[0].ConfigMBWorksheet(ActionNo:= 0,Activate :=FALSE ,DeviceNo :=1, FC :=3, Register:=10, Length:=DummyNoOfRegisters, Polltime_ms:=Polltime1);
Result[1]:= ELA_COMM.MODBUS.CLIENT[0].ConfigMBWorksheet(ActionNo:= 1,Activate :=FALSE ,DeviceNo :=1, FC :=3, Register:=2000, Length:=DummyNoOfRegisters, Polltime_ms:=Polltime2);
Result[2]:= ELA_COMM.MODBUS.CLIENT[0].ConfigMBWorksheet(ActionNo:= 2,Activate :=FALSE ,DeviceNo :=1, FC :=06, Register:=DummyNoOfRegisters, Length:=1, Polltime_ms:=Polltime2);
Result[3]:= ELA_COMM.MODBUS.CLIENT[0].ConfigMBWorksheet(ActionNo:= 3,Activate :=TRUE ,DeviceNo :=1, FC :=16, Register:=0, Length:=DummyNoOfRegisters, Polltime_ms:=Polltime2);
Result[4]:= ELA_COMM.MODBUS.CLIENT[0].ConfigMBWorksheet(ActionNo:= 4,Activate :=FALSE ,DeviceNo :=1, FC :=01, Register:=10, Length:=DummyNoOfRegisters, Polltime_ms:=Polltime2);
Result[5]:= ELA_COMM.MODBUS.CLIENT[0].ConfigMBWorksheet(ActionNo:= 5,Activate :=FALSE ,DeviceNo :=1, FC :=15, Register:=1, Length:=DummyNoOfRegisters, Polltime_ms:=Polltime1);
Result[6]:= ELA_COMM.MODBUS.CLIENT[0].ConfigMBWorksheet(ActionNo:= 6,Activate :=FALSE ,DeviceNo :=1, FC :=5, Register:=10, Length:=1, Polltime_ms:=Polltime1);

DebugRecvString:= ELA_COMM.MODBUS.CLIENT[0].Client_Itf.DebugRecvString;
DebugSendString:= ELA_COMM.MODBUS.CLIENT[0].Client_Itf.DebugSendString;

// ===============================================================================================
//   MODBUS- AUFRUF
// Info:  Der Bausteinaufruf sollte NACH der Konfiguration erfolgen, da sonst z.B.
//        die Socket-Verbindung ohne Gültige IP-Adresse geöfnet wird.
// ===============================================================================================

ELA_COMM(); // MODBUS-AUFRUF
```

Screenshot - Applikation als Client

```
// ===============================================================================================
// MODBUS- CLIENT
// --> Ruft Daten ab
// INFO: Die Angaben müssen nicht zyklisch erfolgen. Der einmalige Aufruf genügt.
// ===============================================================================================

//Konfiguration

// CLIENT INSTANZ #1

FBVersion 'MB_CLIENT' :=ELA_COMM.MODBUS.CLIENT[0].Client_Itf.FBVersion 'MB_CLIENT' ;   // FB-Name der HMI bereitstellen

ELA_COMM.MODBUS.CLIENT[0].ConfigServerIPAddress(ServerIPAddress :='192.168.5.201');
ELA_COMM.MODBUS.CLIENT[0].ConfigServerPortNo(PortNo := 502 );
ELA_COMM.MODBUS.CLIENT[0].ConfigDeviceNo(DeviceNo:=1);

ELA_COMM.MODBUS.CLIENT[0].ConfigREADRegister (Startaddress :=ADR(R_REALS)    ,StartRegister  :=StartRegister_R 0 , Size:= SIZEOF(R_REALS), ExtendedVar:= 1);
ELA_COMM.MODBUS.CLIENT[0].ConfigWRITERegister(Startaddress :=ADR(W_REALS)    ,StartRegister  :=StartRegister_W 0 , Size:= SIZEOF(W_REALS), ExtendedVar:= 1);

ELA_COMM.MODBUS.CLIENT[0].ConfigREADCoils    (Startaddress :=ADR(bInputMemory) ,StartRegister :=4    , Size:= SIZEOF(bInputMemory) ,ExtendedVar:=1);
ELA_COMM.MODBUS.CLIENT[0].ConfigWRITECoils   (Startaddress :=ADR(bOutputMemory),StartRegister :=4    , Size:= SIZEOF(bOutputMemory),ExtendedVar:=1);

//Datenabruf
Result[0] 0 := ELA_COMM.MODBUS.CLIENT[0].ConfigMBWorksheet(ActionNo:= 0,Activate :=FALSE ,DeviceNo :=1, FC :=3, Register:=10, Length:=DummyNoOfRegisters 10 , Polltime_ms:=Polltime1 600 );
Result[1] 0 := ELA_COMM.MODBUS.CLIENT[0].ConfigMBWorksheet(ActionNo:= 1,Activate :=FALSE ,DeviceNo :=1, FC :=3, Register:=2000, Length:=DummyNoOfRegisters 10 , Polltime_ms:=Polltime2 1400 );
Result[2] 0 := ELA_COMM.MODBUS.CLIENT[0].ConfigMBWorksheet(ActionNo:= 2,Activate :=FALSE ,DeviceNo :=1, FC :=06, Register:=DummyNoOfRegisters 10 , Length:=1, Polltime_ms:=Polltime2 1400 );
Result[3] 0 := ELA_COMM.MODBUS.CLIENT[0].ConfigMBWorksheet(ActionNo:= 3,Activate :=TRUE ,DeviceNo :=1, FC :=16, Register:=0, Length:=DummyNoOfRegisters 10 , Polltime_ms:=Polltime2 1400 );
Result[4] 0 := ELA_COMM.MODBUS.CLIENT[0].ConfigMBWorksheet(ActionNo:= 4,Activate :=FALSE ,DeviceNo :=1, FC :=01, Register:=10, Length:=DummyNoOfRegisters 10 , Polltime_ms:=Polltime2 1400 );
Result[5] 0 := ELA_COMM.MODBUS.CLIENT[0].ConfigMBWorksheet(ActionNo:= 5,Activate :=FALSE ,DeviceNo :=1, FC :=15, Register:=1, Length:=DummyNoOfRegisters 10 , Polltime_ms:=Polltime1 600 );
Result[6] 0 := ELA_COMM.MODBUS.CLIENT[0].ConfigMBWorksheet(ActionNo:= 6,Activate :=FALSE ,DeviceNo :=1, FC :=5, Register:=10, Length:=1, Polltime_ms:=Polltime1 600 );

DebugRecvString 'c0012:012-' := ELA_COMM.MODBUS.CLIENT[0].Client_Itf.DebugRecvString 'c0012:012-' ;
DebugSendString 'c0012:033-' := ELA_COMM.MODBUS.CLIENT[0].Client_Itf.DebugSendString 'c0012:033-' ;
```

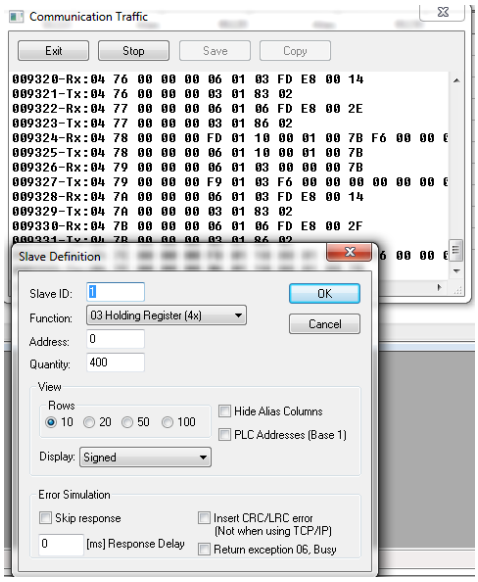
Screenshot – PC Software als Server



Fehlermeldungen

Beispiele

```
//Datenabruf
Result[0] 10 := ELA_COMM.MODBUS.CLIENT[0].SetMBClientWorksheet(ActionNo:= 0,Activate :=TRUE ,SlaveNo :=1, FC :=FC03_READ_HOLDING_REGISTERS, Register:=0, Length:=123, Polltime_ms:=Polltime1 1000 );
Result[1] 2 := ELA_COMM.MODBUS.CLIENT[0].SetMBClientWorksheet(ActionNo:= 1,Activate :=TRUE ,SlaveNo :=1, FC :=FC03_READ_HOLDING_REGISTERS, Register:=65000, Length:=20, Polltime_ms:=Polltime2 1000 );
Result[2] 2 := ELA_COMM.MODBUS.CLIENT[0].SetMBClientWorksheet(ActionNo:= 2,Activate :=TRUE ,SlaveNo :=1, FC :=06, Register:=65000, Length:=1, Polltime_ms:=Polltime2 1000 );
Result[3] 11 := ELA_COMM.MODBUS.CLIENT[0].SetMBClientWorksheet(ActionNo:= 3,Activate :=TRUE ,SlaveNo :=1, FC :=16, Register:=1, Length:=123, Polltime_ms:=Polltime2 1000 );
Result[4] 0 := ELA_COMM.MODBUS.CLIENT[0].SetMBClientWorksheet(ActionNo:= 4,Activate :=FALSE ,SlaveNo :=1, FC :=01, Register:=1, Length:=DummyCoilLength 2000 , Polltime_ms:=Polltime2 1000 );
Result[5] 0 := ELA_COMM.MODBUS.CLIENT[0].SetMBClientWorksheet(ActionNo:= 5,Activate :=FALSE ,SlaveNo :=1, FC :=15, Register:=1, Length:=24, Polltime_ms:=Polltime1 1000 );
Result[6] 0 := ELA_COMM.MODBUS.CLIENT[0].SetMBClientWorksheet(ActionNo:= 6,Activate :=FALSE ,SlaveNo :=1, FC :=5, Register:=2000, Length:=1, Polltime_ms:=Polltime1 1000 );
DummyCoilLength 2000 ;
```



Errorcode 1 (Function Code not supported)

Der verwendete FunktionCode wird entweder vom Client/Master oder vom Server/Slave nicht unterstützt.

Errorcode 2 (Illegal address value)

Die angegebene Adresse verweist nicht auf einen Speicheradresse
Ein schreibender oder lesender Zugriff ist nicht möglich.

Errorcode 3 (Illegal data value)

Der zu lesende oder zu schreibene Bereich liegt nicht mehr komplett im Speicherbereich.

Errorcode 4 (slave device failure)

Wird nicht verwendet.

Errorcode 5 (Acknowledge)

Wird nicht verwendet.

Errorcode 6 (slave device busy)

Wird nicht verwendet.

Errorcode 8 (memory parity error)

Wird nicht verwendet.

Errorcode 11 (no answer)

Der Server / Slave hat innerhalb der vorgegebenen Zeit NICHT geantwortet.

Errorcode 12 (CRC error)

Client: Die CRC- Prüfung findet hier bereits im Ethernet-Stack statt.
Dies bedeutet dass die Antwort bereits auf TCP/IP-Ebene gelöscht wird. → Error Code 11

Master: Das fehlerhafte Telegramm wird in der Master-Instanz berechnet,
als fehlerhaft markiert und verworfen. → Error Code 12

3.3 RTU Modbus

Generell

Der RTU- Modbus besteht aus 2 Teilen

MASTER	→	Fragt aktiv die Daten an.
SLAVE	→	Datenbereitstellung, Anwortet auschliesslich auf Anfragen des Masters, wird selbst nie aktiv.

Jede Komponente, bzw. jede Instanz einer Komponente benötigt einen eigenen COM-Port.

Beispiel:

COM2 als RS485 als SLAVE	→	Kann Anfragen mehrere Master beantworten. (Achtung, Gefahr von Anfragekollisionen)
COM2 als RS485 als MASTER	→	Der Master kann mehere Slaves anfragen.
COM1 als RS232 als MASTER	→	Kann auf Grund der Spezifikation von RS232 (Peer to Peer) nur mit einem Slave kommunizieren.

Beschreibung der Methoden

Die ELA_COMM Version ab Version 3.5.0.3 erlaubt die Benutzung von RTU Modbus sowohl als Master, wie auch als Slave.
Hierbei kann der Modbus während der Laufzeit konfiguriert und geändert werden.

Änderungen die zur Laufzeit erfolgen können:

- Anzahl der benutzten Kommunikationsinstanzen, Ein- und Ausschaltenen der Instanzen.
- Alle COMPort- Einstellungen.
- Ändern der Gerätenummer (DeviceNo)

Änderungen mit Code:

Erweitern der Maximalgrößen für:

- Maximale Anzahl von Worksheets
- Maximale Anzahl von Kommunikationsinstanzen

HINWEIS

Diese Maximaleinstellungen liegen als VAR_CONSTANT vor und benötigen eine Recompile. Darauf ist zu achten, dass die Konstanten den maximalen Werten entsprechen.

HINWEIS

Es ist Sorge zu tragen, daß nicht mehrere Instanzen auf denselben COM-Port zugreifen. Greift z.B eine Master- und eine Slaveinstanz auf denselben COM-Port zu, kann jede Instanz nicht mehr alle Empfangstelegramme erhalten. Die Kommunikation ist gestört.

Allgemeine Methoden Konfiguration

ConfigCOMSettings()

Funktion:
Öffnen des COM-Port

Die Parameter werden im laufenen Betrieb überwacht. Sollte sich ein Parameter ändern, so wird der Com-Port geschlossen und mit den neuen Parametern neu geöffnet.

Geltungsbereich:
[X] MASTER
[X] SLAVE

Beispiel:

```
ELA_COMM.MODBUS.MASTER[0].ConfigCOMSettings
(COMPort          := SYS_COMPORT1,
COMBaud           := SYS_BR_38400,
COMParity         := SYS_NOPARITY,
COMStopBits       := SYS_ONESTOPBIT,
COMTimeout        := SYS_NOWAIT,
COMBufferSize     := 200,
COMExBufferSize   := 8,
COMRs485          := FALSE
);
```

<table>
<tr><th>Variable IN</th><th>Typ</th><th>Bedeutung</th></tr>
<tr><td>COMPort</td><td>INT</td><td>Wahl des COM-Ports gemäß CODESYS SysCOM
<table><tr><th>Name</th><th>Datentyp</th><th>Initialwert</th></tr><tr><td>SYS_COMPORT_NONE</td><td>INT</td><td>0</td></tr><tr><td>SYS_COMPORT1</td><td>INT</td><td>1</td></tr><tr><td>SYS_COMPORT2</td><td>INT</td><td></td></tr><tr><td>SYS_COMPORT3</td><td>INT</td><td></td></tr><tr><td>SYS_COMPORT4</td><td>INT</td><td></td></tr></table></td></tr>
<tr><td>COMBaud</td><td>DWORD</td><td>Wahl der Übertragungsgeschwindigkeit gemäß CODESYS SysCOM
<table><tr><th>Name</th><th>Datentyp</th><th>Initialwert</th></tr><tr><td>SYS_BR_4800</td><td>DWORD</td><td>4800</td></tr><tr><td>SYS_BR_9600</td><td>DWORD</td><td>9600</td></tr><tr><td>SYS_BR_19200</td><td>DWORD</td><td>19200</td></tr><tr><td>SYS_BR_38400</td><td>DWORD</td><td>38400</td></tr><tr><td>SYS_BR_57600</td><td>DWORD</td><td>57600</td></tr><tr><td>SYS_BR_115200</td><td>DWORD</td><td>115200</td></tr></table></td></tr>
<tr><td>COMParity</td><td>BYTE</td><td>Wahl der Parität gemäß CODESYS SysCOM
<table><tr><th>Name</th><th>Datentyp</th><th>Kommentar</th></tr><tr><td>SYS_NOPARITY</td><td>BYTE</td><td>No parity</td></tr><tr><td>SYS_ODDPARITY</td><td>BYTE</td><td>Odd parity</td></tr><tr><td>SYS_EVENPARITY</td><td>BYTE</td><td>Even parity</td></tr></table></td></tr>
<tr><td>COMStopBits</td><td>BYTE</td><td>Anzahl der Stopbits gemäß CODESYS SysCOM
<table><tr><th>Name</th><th>Datentyp</th><th>Initialwert</th></tr><tr><td>SYS_ONESTOPBIT</td><td>BYTE</td><td>1</td></tr><tr><td>SYS_ONE5STOPBITS</td><td>BYTE</td><td></td></tr><tr><td>SYS_TWOSTOPBITS</td><td>BYTE</td><td></td></tr></table></td></tr>
<tr><td>COMTimeout</td><td>UDINT</td><td>Wahl des Schnittstellen-Timout in Millisekunden
gemäß CODESYS SysCOM
<table><tr><th>Name</th><th>Datentyp</th><th>Initialwert</th></tr><tr><td>SYS_NOWAIT</td><td>UDINT</td><td>0</td></tr><tr><td>SYS_INFINITE</td><td>UDINT</td><td>16#FFFFFFFF</td></tr></table>Als Default sollte SYS_NOWAIT verwendet werden.

Zeit, welche bei Handhabung des COM-Portes im Fehlerfall blockierend vergehen kann, ehe ein Fehlercode ausgegebe wir.
Bei SYS_NOWAIT wird „nicht blockierend“ unmittelbar ein Fehlercode ausgegeben.</td></tr>
</table>

COMBufferSize	UDINT	DEFAULT = 1024 Dieser Wert ist gerätespezifisch und z.B. auf 1.) Visio PMC410 über die Windows Registry auf 4096 eingestellt. 2.) RMC 503 fix auf 4096 eingestellt. Die Vorgabe aus IEC wird daher ignoriert.
COMExBufferSize	BYTE	Default = 8 Anzahl von Bits/Byte Hinweis: Dieser Parameter wird vom Laufzeitsystem derzeit nicht ausgewertet.
COMRs485	BOOL	Setzen der automatischen Umschaltung von senden und empfangen. Hierfür wird ComSettingsEx.bRtsControl auf den Wert 3 gesetzt. FALSE := Kein automatischen umschalten (RS232) TRUE := Automatisches umschalten (RS485)

Variable OUT	**Typ**	**Bedeutung**
	BOOL	Nicht verwendet

CloseCOMPortFromExtern()

Funktion:
Schließen des COM- Port durch ein Systemereignis.

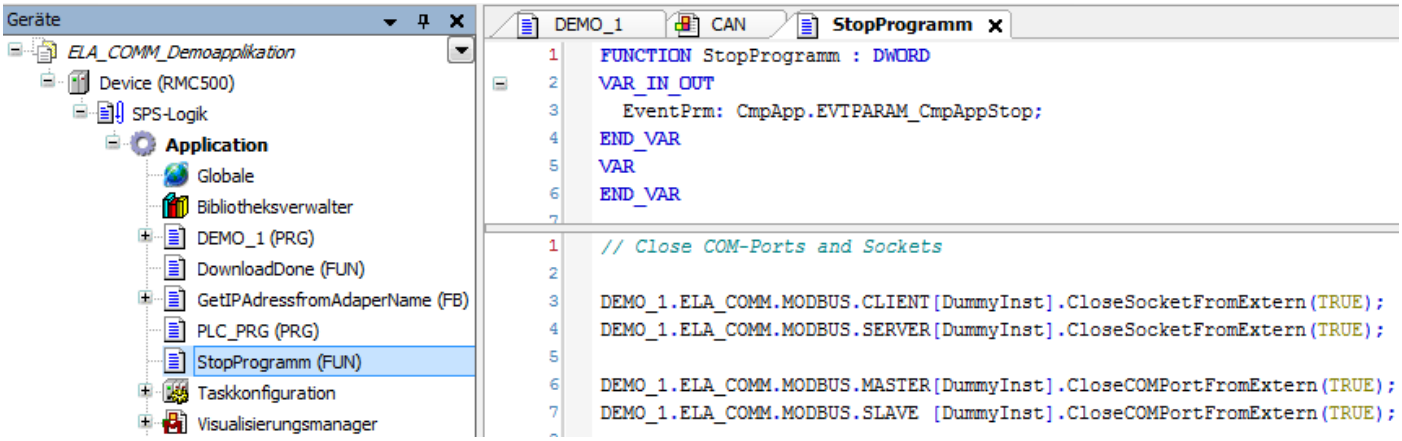
Geltungsbereich:
[X] MASTER
[X] SLAVE

Beispiel:
DEMO_1.ELA_COMM.MODBUS.MASTER[0].CloseCOMPortFromExtern(CloseCOMPort := TRUE);

Variable IN	Typ	Bedeutung
CloseCOMPort	BOOL	Schliesst den COM- Port in den jeweiligen Instanzen.

Variable OUT	Typ	Bedeutung
	BOOL	Nicht verwendet

Um diese Funktionalität sinnvoll zur Anwendung bringen zu können, sind folgende applikative Schritte notwendig.



Führen Sie innerhalb dieser Funktion nun Ihre Befehle zum Schließen der COM- Ports oder zum Schließen der Socketverbindung aus.

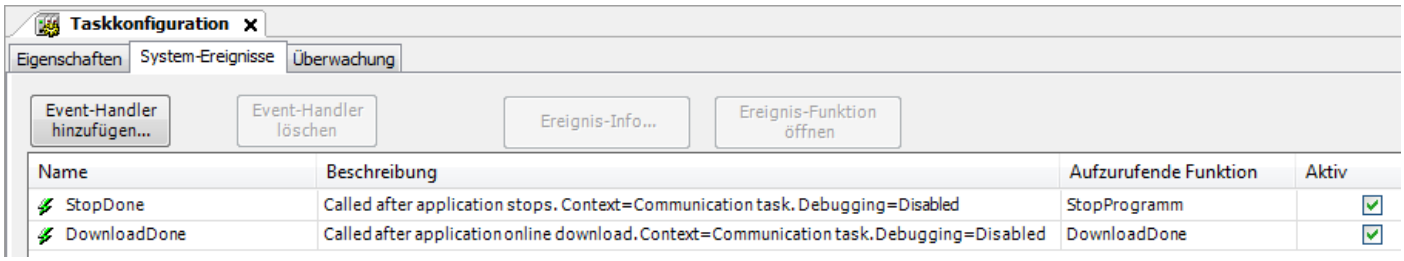
Anlegen von Systemereignissen in der Taskkonfiguration, welche die Funktionen zum Schließen der Sockets und der COM-Ports aufrufen.

Es wird empfohlen die beiden Systemereignisse zu verwenden:

- Stop (StopDone)

- Download beendet (DownloadDone)

In beiden Fällen werden in nachfolgendem Beispiel die COM-Ports, bzw. die Socketverbindungen geschlossen.



Nachfolgende Methoden sind im allgemeinen Teil beschrieben.
(siehe KapitelTCP)

ConfigDeviceNo()

SetStatusRegister()

SetStatusCoil()

ConfigReadRegister()

ConfigWriteRegister()

ResetCounter()

ConfigREADCoils()

ConfigWRITECoils()

Master Übertragungsfunktionen

ConfigMBWorksheet()

Slave Methoden

ShowDamageRecv()

Funktion:
Zeigt in der Debugausgabe auch ungültige Telegramme und Zeichen an, welche zur Auswertung nicht berücksichtigt werden. (Inbetriebnahmemodus)

Geltungsbereich:
[X] Master
[X] Slave

Beispiel:
ELA_COMM.MODBUS.SLAVE[0].ShowDamageRecv(ShowDamage := ShowDamageRecvTelegramms);

Variable IN	Typ	Bedeutung
ShowDamage	Bool	FALSE (Default) Zeigt nur ein Empfangstelegramme an, wenn die Stationskennung im Telegramm und die Telegrammlänge stimmen Diese Option ist für ein funktionierendes System empfohlen. TRUE Zeigt auch unvollständige oder falsche Telegramme im Empfangspuffer an. Diese Option ist für Inbetriebnahmen empfohlen.

Bool-Array (Versionen < V3.5.0.4)

In späteren Versionen ist die Anbindung an ein Bool-Array geplant.

Sollte bereits heute die Verwendung von BOOL-Arrays gegenüber dem BYRE-Array Vorteile bringen, können diese extern gewandelt werden.

Für diese Formel muss MAXREGISTER durch 8 teilbar sein

Codebeispiel:

Für lesenden Zugriff → (FC1)
ByteArray → Boolarray

```
FOR i:=0 TO (MAXREGISTER/8) BY 8 DO
bInputMemory[i]   := InputMemory[i].0;
bInputMemory[i+1] := InputMemory[i].1;
bInputMemory[i+2] := InputMemory[i].2;
bInputMemory[i+3] := InputMemory[i].3;
bInputMemory[i+4] := InputMemory[i].4;
bInputMemory[i+5] := InputMemory[i].5;
bInputMemory[i+6] := InputMemory[i].6;
```

```
bInputMemory[i+7]    := InputMemory[i].7;
END_FOR
```

4 Kommunikation CAN- Layer2

4.1 Generelles zu Can- Layer2

Funktionsbeschreibung

Die ELA_COMM ab Version 3.5.8.0 wurde um eine CAN- Layer2 Kommunikation erweitert.

Merkmale:
- Bis zu 63 Teilnehmer
- Synchronisatzion von Variablen in einem Array vom Typ BOOL/BYTE/INT/DINT/REAL
- Attribute für jede Variable (Senden / Empfangen)
- Automatische Lastanpassung an den CAN-Bus
- Teilnehmererkennung über Empfangstelegramme

ConfigDeviceNo()

Funktion:
Vergabe der NodeID des Gerätes

Geltungsbereich:
[X] CAN

Beispiel:
ELA_COMM.CAN.CANMASTER[0].ConfigDeviceNo(DeviceNo:=DeviceNo);

Variable IN	Typ	Bedeutung
DeviceNo	BYTE	Node ID (Devicenummer) unter welcher die Instanz künftig Sendetelegramme absetzt. 0= Instanz ist deaktiviert

ConfigCANSettings()

Funktion:
Vergabe der NodeID des Gerätes

Wird die Funktion nicht aufgrufen, so gelten folgende Default- Werte:
CANNo := T#10S ,
CANBaud := T#50MS ,

Geltungsbereich:
[X] CAN

Beispiel:
ELA_COMM.CAN.CANMASTER[0].ConfigCANSettings(CANNo:=0 ,CANBaud :=50);

Variable IN	Typ	Bedeutung
CANNo	BYTE	Setzt den gewünschten CAN-Controller CAN0/CAN1
CANBaud	WORD	Setzen der zu verwendenden Baudrate in kB.

ConfigTime()

Funktion:
Vergabe der NodeID des Gerätes

Wird die Funktion nicht aufgrufen, so gelten folgende Default- Werte:
PollVariableMax := T#10S ,
PollVariableMin := T#50MS ,
TimeOutDetectDevices := T#30S

Geltungsbereich:
[X] CAN

Beispiel:
```
ELA_COMM.CAN.CANMASTER[0].ConfigTime
(
PollVariableMax := T#10S ,
 PollVariableMin := T#50MS ,
TimeOutDetectDevices := T#30S
);
```

Variable IN	Typ	Bedeutung
PollVariableMax	TIME	Zeit nach der ein Wert ohne Werteänderung zyklisch gesandt wird. Bei Variablenwertänderung innerhalb < PollVariableMax und > PollVariableMin wird der Wert unverzüglich gesandt.
PollVariableMin	TIME	Sperrzeit für Werteänderungen. Ein Telegramm wegen Variablenänderung wird erst nach Ablauf dieser Zeit gesandt.
TimeOutDetectDevices	TIME	Zeit nachdem ein detektierter Kommunikationspartner nach Ausbleiben von Telegrammen als "verloren" gilt. (Prüfung alle 3 Sekunden).

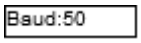
Funktion:
Abfrage ob Partnergeräte vorhanden sind.
ReturnValue: Bool

Geltungsbereich:
[X] CAN

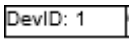
Beispiel:
xDeviceDetect_01:= DEMO_1.ELA_COMM.CAN.CANMASTER[0]._CANData.DeviceOK[1]
xDeviceDetect_02:= DEMO_1.ELA_COMM.CAN.CANMASTER[0]._CANData.DeviceOK[2]

Erklärung der Funktionsweise am Demobeispiel mit 2 Geräten

Allgemein:

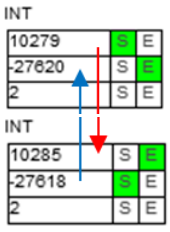


Beide Steuerungen unterhalten sich mit derselben Baudrate. Dies ist zwingend erforderlich.

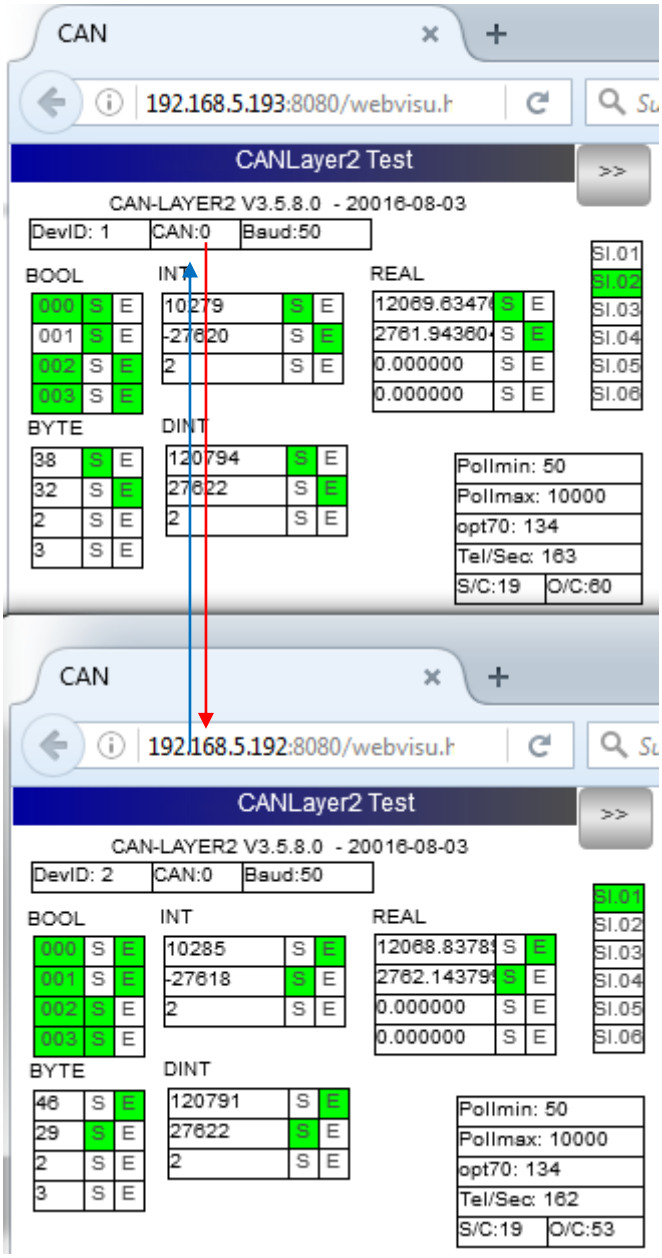


Jede Steuerung muß eine Individuelle ID besitzen.

Ein Partner kann detektiert werden, wenn dieser ein Telegramm absetzt. Variable zur Erkennung:
DEMO_1.ELA_COMM.CAN.CANMASTER[0].
_CANData.DeviceOK[1]



Übertragung von Variablenwerten:
Bei Device 1 wurde als erster Integer-Wert Senden aktiviert. Zyklisch und bei Wertänderung stellt dieses Gerät den Wert über den CAN-Bus bereit. Der zweite Integer-Wert wurde als Empfangswert definiert. Somit wird dieser Wert nur über denCAN-Bus aktualisiert. Das gleichzeitige Senden- und Empfangen eines Wertes ist möglich.



5 Hilfe bei Störungen

5.1 Service und Support

Hotline

Für zusätzliche Unterstützung und Informationen können sie unsere Hotline zu folgenden Zeiten erreichen:

Mo-Fr: 8.00- 12.00 und 13.00 - 16.30

Tel.: +49 (0) 7021 / 92025-33

Außerhalb dieser Zeiten, können sie uns per e-mail oder Fax erreichen:

Fax.: +49 (0) 7021 / 92025-29
e-mail: support@elrest.de

Training und Workshops

Wir bieten Ausbildung oder Projekt bezogene Workshops zu allen elrest Produkten an.

Für weitere Informationen kontaktieren sie bitte unsere Vertriebsabteilung:

Telefon: +49 (0) 7021/92025-0
Fax: +49 (0) 7021/92025-29
E-mail: vertrieb@elrest.de

6 Historie

Datum	*Name*	*Version*	*Änderung*
12.07.2015	Ne	V3.5.0.0	Neu erstellt
16.07.2015	Ne	V3.5.0.1	Namensänderungen in Methoden, Variablengröße
23.07.2015	Ne	V3.5.0.2	Korrekturen, Server antwortet auf alle Stationsnummern
14.09.2015	Ne	V3.5.0.3	Implementation RTU- Modbus
21.10.2015	Ne	V3.5.0.4	Nachträge / Optimierungen
06.11.2015	Ne	V3.5.0.5	Korrekturen, ACTIVE-Bit, Statemachine
19.11.2015	Ne	V3.5.0.5_1	Fehlerkorrektur
23.11.2015	Ne	V3.5.0.5_2	Korrekturen in SetStatusRegister() / SetStatusCoil()
25.11.2015	Ne	V3.5.0.5_3	2 Grafiken hinzugefügt
11.02.2016	Ne	V3.5.0.6	Fehlerkorrekturen und Änderungen
03.08.2016	Ne	V3.5.8.0	Implementation CAN-Layer2
09.08.2016	Ne	V3.5.8.1	Fehlerkorrekturen MBServer FC16 Typ0
12.09.2016	Ne	V3.5.8.1	Anpassung Dokumentation

Versionshinweis:
Ziffer 1: 3S- Hauptnummer (3s)
Ziffer 2: 3S- Versionsnummer (3s)
Ziffer 3: 3S- Servicepack (3s)
Ziffer 4: Fortlaufende Versionsnummer bei unveranderten Ziffern 1-3
gültig ab 06/16