Applikationsschrift 5030:
Kommunikation mit der PA-CONTROL Touch über ModbusTCP

Kurzfassung	Diese Applikationsschrift zeigt, wie mit dem Modbus-TCP-Protokoll über die TCP/IP-Schnittstelle mit der PAC-CONTROL Touch kommuniziert werden kann. Folgendes wird beschrieben ■ Modbus-TCP-Protokoll ■ Zugriff auf die PA-CONTROL ■ Programmbeispiel Microsoft Visual C# 2008 Express ■ Programmbeispiel Beckhoff TwinCAT ■ Programmbeispiel PA-CONTOL

Warenzeichen und Warennamen sind ohne Gewährleistung der freien Verwendbarkeit benutzt. Bei der Erstellung der Texte und Beispiele wurde mit großer Sorgfalt vorgegangen. Trotzdem können Fehler nicht ausgeschlossen werden. Die IEF-Werner GmbH kann für fehlende oder fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine Haftung übernehmen.

Die IEF-Werner GmbH behält sich das Recht vor, ohne Ankündigung die Software oder Hardware oder Teile davon, sowie die mitgelieferten Druckschriften oder Teile davon zu verändern oder zu verbessern.

Alle Rechte der Vervielfältigung, der fotomechanischen Wiedergabe, auch auszugsweise sind ausdrücklich der IEF-Werner GmbH vorbehalten.

Für Verbesserungsvorschläge und Hinweise auf Fehler sind wir jederzeit dankbar.

© Januar 2018; IEF-Werner GmbH

IEF-Werner GmbH | Wendelhofstraße 6 | DE-78120 Furtwangen | www.ief-werner.de | Zentrale: +49 7723/925-0

Inhaltsverzeichnis

1 Änderungshistorie 5
2 Allgemein 6
2.1 Port Nummer 6
3 Modbus TCP 7
3.1 Einführung 7
3.2 Protokoll 8
3.2.1 MBAP Header-Beschreibung 8
3.2.2 Function Codes 9
4 Kommunikation mit der PA-CONTROL 10
4.1 PA-CONTROL allgemein 11
4.1.1 Status- und Betriebszustandsmeldungen 12
4.2 Achsdaten 13
4.2.1 Achsdaten lesen 14
4.2.2 Status der Achse 16
4.3 N-Register 17
4.4 R-Register 17
4.5 PA-CONTROL Strings 18
4.6 Digitale Eingänge 18
4.7 Digitale Ausgänge 19
4.8 Merker 19
4.9 System Merker 20

Erstellt von: Grieshaber Edmund | APP5030_DE_1120445_PA-CONTROL_ModbusTCP_Kommunikation_R1d.doc

4.10 System N-Register 20
4.11 System R-Register 20

5 Beispiel C#-Programm 21
5.1 Visual Studio Project 22
5.2 Klasse Class_PaControlModbusTcpClient 23
5.2.1 Klasse Class_PaControlModbusTcpClient.ClassSystemPaControl 24
5.2.2 Klasse Class_PaControlModbusTcpClient.ClassAxe 25
5.2.3 Klasse Class_PaControlModbusTcpClient.ClassInput 26

6 Beispiel C++ 27
6.1 Visual Studio Project 28
6.2 Klasse Class_ModbusTCP 29
6.2.1 Klasse Class_ModbusTcp.SystemPaControl 30
6.2.2 Klasse Class_ModbusTcp.Axe 31
6.3 Skizze zur Kommunikation (Übersicht) 32
6.3.1 Kommunikation in der Betriebsart AUTOMATIK 33
6.3.2 Kommunikation in der Betriebsart SLAVE 34

7 Beispiel Beckhoff TwinCAT Programm 35
7.1 Beispiel: „PAC_ModbusTCP_Vx-xx.pro“ 36
7.1.1 Funktionsbaustein ReadPacNRegister 37
7.1.2 Funktionsbaustein WritePacNRegister 38
7.1.3 Funktionsbaustein ReadPacRRegister 39
7.1.4 Funktionsbaustein WritePacRRegister 40
7.1.5 Funktionsbaustein ReadPacMerker 41
7.1.6 Funktionsbaustein WritePacMerker 42
7.1.7 Funktionsbaustein ReadOnePacString 43

7.1.8 Funktionsbaustein WriteOnePacString 44

8 Beispiel von PA-CONTROL zu PA-CONTROL 45

9 Beispiel PA-CONTROL zu einem MODBUS-Server (Fa. Hilscher) 46

9.1 Konfiguration der Teilnehmer 47
9.1.1 IP-Adresse der PA-CONTROL 47
9.1.2 Einstellung der Hilscher-Karte 48
9.2 Befehlsübersicht 49
9.2.1 Befehlsbeispiele Merker 49
9.2.2 Befehlsbeispiele N-Register 50
9.3 Kommunikationsbeispiel 51
9.3.1 Programm in der PA-CONTROL 51
9.3.2 Hilscher-Karte Diagnose E/A-Monitor 52
9.3.2.1 G322-Befehl 52
9.3.2.2 G332-Befehl 53
9.3.3 Hilscher Karte Diagnose 54

10 Slave Betrieb der PA-CONTROL 55

10.1 Slave Betrieb Interface 55
10.2 Slave Kommando Achsdaten 56
10.2.1 Kommandos 57
10.2.2 Statusinformation 58
10.3 Slave IO-Knoten Daten 59
10.3.1 Slave IO-Knoten Daten schreiben 59
10.3.2 Slave IO-Knoten Daten lesen 60
10.4 Steuerungsprinzip für die Achsen über die Kommandos 61
10.4.1 Ablaufdiagramm für ein Kommando für eine Achse 62

1 Änderungshistorie

Dokumentenänderungen und Lebenslauf:

Dokumentencode	Datum	Erstellung und Änderung
APP5030_DE_1120445_PAC_ModbusTCP_Kommunikation_R1a.doc	Juli 2014	Neuanlage dieses deutschen Dokuments
APP5030_DE_1120445_PAC_ModbusTCP_Kommunikation_R1b.doc	April 2015	Kommunikation von einer PA-CONTROL mit einem MODBUS-Server
APP5030_DE_1120445_PAC_ModbusTCP_Kommunikation_R1c.doc	Januar 2017	Beispiel für C# und C++ eingefügt (siehe Abschnitt 5, 🗎 21 und 6, 🗎 27) Beschreibung zur Betriebsart "SLAVE" der PA-CONTROL eingefügt (siehe Abschnitt 10, 🗎 55)
APP5030_DE_1120445_PAC_ModbusTCP_Kommunikation_R1d.doc	Januar 2018	Ergänzung um Abschnitt Steuerungsprinzip für die Achsen über die Kommandos, ab Seite 61.

2 Allgemein

Die PA-CONTROL Touch ist mit einer TCP/IP-Schnittstelle ausgestattet. Mit WINPAC wird über diese Schnittstelle die Inbetriebnahme und Programmierung durchgeführt.

Auf der PA-CONTROL Touch ist ein Modbus-TCP-Server implementiert. Dieser kann über die TCP/IP-Schnittstelle (IP-Adresse, Port=11001 oder Port=502, ...) angesprochen werden.

Die Modbus-TCP-Schnittstelle erlaubt folgende Zugriffe:

- PA-CONTROL allgemein (Starte Betriebsart AUTOMATIK, Stoppen ... , Abbrechen ... , ...)
- Achsen (nur lesen)
- digitale Ein- und Ausgänge, Merker
- N-Register, R-Register
- ...

Über diese Schnittstelle kann eine übergeordnete Steuerung (SPS, PC, ...) die Betriebsart AUTOMATIK der PA-CONTROL starten, stoppen und bei Bedarf abbrechen.

Der Status der Maschine (Produktinformationen, Stückzahlen, Positionen, ...), welcher in Registern oder Merkern abgelegt ist, kann gelesen werden.

Durch Schreiben von Werten in die Register oder Merker kann dann der Ablauf der Maschine gesteuert werden.

2.1 Port Nummer

Portnummer	Beschreibung
502	Default für MODBUS, reserviert von der „Internet Community“
11001	PA-CONTROL

3 Modbus TCP

3.1 Einführung

Modbus-TCP hat sich neben Profinet IO, Ethernet/IP, Ethercat und Powerlink als Defacto-Standard für die industrielle Kommunikation über Ethernet etabliert.

Die Basis hierfür bildet eine stabile Spezifikation, verfügbare Basistechnologie und eine Vielzahl industrieller Basisgeräte.

Modbus-TCP ist heute ein offener Internet-Draft-Standard. Die seit der Ursprungsvariante bewährten Modbus-Dienste und das Objektmodell wurden unverändert beibehalten und auf TCP/IP als Übertragungsmedium abgebildet.

Modbus-TCP ist in der Norm IEC61158 festgelegt und wird in IEC 61784-2 als CPF15/1 referenziert.

Für die Modbus Kommunikation ist der TCP Port 502 reserviert.

Hinweis: Da bei der PA-CONTROL die TSwin-Oberfläche über Modbus-TCP auf dem Port 502 schon kommuniziert, ist die freie Modbus-TCP-Schnittstelle auf den Port 11001 gelegt worden.

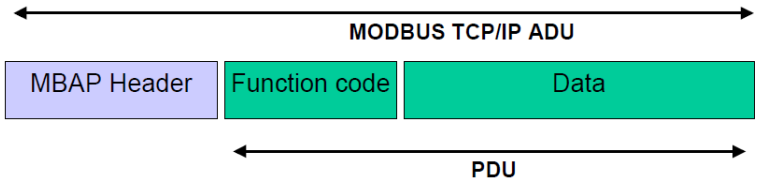
Weitere Informationen finden Sie unter: http://www.modbus.org

- Modbus_over_serial_line_V1_02.pdf
- Modbus_Messaging_Implementation_Guide_V1_0b.pdf
- ModbusApplicationProtocol_v1_1.pdf

3.2 Protokoll

Ein Modbus-TCP-Telegramm (Modbus TCP/IP ADU) besteht aus:

- MBAP Header (Modbus Application Protocol header)
- Function code
- Data



3.2.1 MBAP Header-Beschreibung

Der MBAP Header ist 7 Bytes lang.

Benennung	Länge	Beschreibung
Transaction Identifier	2 Bytes	Wird für die Zuordnung der Telegramme verwendet. Der Server kopiert den Wert in die Antwort.
Protocol Identifier	2 Bytes	Bei Modbus-TCP immer 0.
Length	2 Bytes	Anzahl der folgenden Bytes inklusive Unit Identifier, Functioncode und Data
Unit Identifier	1 Byte	Dieses Feld wir für „Intra-System-Routing“ verwendet. Da die PA-CONTROL mehr als 65 536 Datenworte für die Kommunikation hat (Ein- /Ausgänge, Merker, N-/R-Register, Achsdaten, ...), wird über dieses Feld in der PA-CONTROL auf die verschiedenen Elemente verzweigt. z.B.: ■ 1 ... 16 : Daten der Achse 1 bis 16 ■ 100 : Zugriff auf die N-Register ■ 110 : Zugriff auf digitale Eingänge

Erstellt von: Grieshaber Edmund | APP5030_DE_1120445_PA-CONTROL_ModbusTCP_Kommunikation_R1d.doc

3.2.2 Function Codes

In der PA-CONTROL wurden nicht alle Function codes umgesetzt.

Benennung	Code [hex]	Unit Identifier	Beschreibung
Read Discrete Inputs	02	110	Liest die digitalen Eingänge
Write Single Coil	05		
Write Multiple Coil	15		
Read Input Register	04	1 ... 16	Liest Daten der Achse [1...16]
		99	Liest Statusdaten der PA-CONTROL [Betriebsart, Serien-Nummer, ...]
Read Holding Register	03	1 ... 16	Liest Daten der Achse [1...16]
		99	Liest Statusdaten der PA-CONTROL [Betriebsart, Serien-Nummer, ...]
		120	Liest digitale Ausgänge
		130	Liest Merker
		100	Liest N-Register
		101	Liest R-Register
Write Single Register	06	99	Beeinflusst die Betriebsart der PA-CONTROL [Start, Stop, ...]
		120	Setzt einen digitalen Ausgang
		130	Setzt einen Merker
Write Multiple Register	16	99	Beeinflusst die Betriebsart der PA-CONTROL [Start, Stop, ...]
		120	Setzt mehrere digitale Ausgänge
		130	Setzt mehrere Merker
		100	Schreibt ein oder mehrere N-Register
		101	Schreibt ein oder mehrere R-Register

4 Kommunikation mit der PA-CONTROL

Der Modbus stellt für die Adressierung der Elemente (Eingänge, Ausgänge, Register, ...) ein Adresswort mit 65536 Möglichkeiten für Datenworte bereit.
Da für die PA-CONTROL aber mehr als 65536 Möglichkeiten benötigt werden, wird das Feld „Unit Identifier“ aus dem Modbus Application Protocol Header (MBAP Header) benutzt.

Unit-Identifier	Zugriff auf	Verwendung in den Betriebsarten		
		Grundstellung	AUTOMATIK	ONLINE
1 ... 16	Achse 1 bis Achse 16	lesen	lesen	lesen / schreiben
17 ... 32	CAN-IO-Knoten	lesen	lesen	lesen / schreiben
99	PA-CONTROL allgemein	lesen / schreiben	lesen / schreiben	lesen / schreiben
100	N-Register	lesen / schreiben	lesen / schreiben	nicht sinnvoll
101	R-Register	lesen / schreiben	lesen / schreiben	nicht sinnvoll
103	PA-CONTROL-Strings	lesen / schreiben	lesen / schreiben	nicht sinnvoll
105	PA-CONTROL-Strings (wortorganisiert)	lesen / schreiben	lesen / schreiben	nicht sinnvoll
104	N-Register-Byte-Wert (nur Bit 0 ... 7)	lesen / schreiben	lesen / schreiben	nicht sinnvoll
110	Digitale Eingänge	lesen / schreiben	lesen / schreiben	nicht sinnvoll
120	Digitale Ausgänge	lesen / schreiben	lesen / schreiben	nicht sinnvoll
130	Merker	lesen / schreiben	lesen / schreiben	nicht sinnvoll
131	Merkerwort	lesen / schreiben	lesen / schreiben	nicht sinnvoll
160	System-Merker	lesen / schreiben	lesen / schreiben	nicht sinnvoll
161	System-N-Register	lesen / schreiben	lesen / schreiben	nicht sinnvoll
162	System-R-Register	lesen / schreiben	lesen / schreiben	nicht sinnvoll

4.1 PA-CONTROL allgemein

Für den Zugriff auf die allgemeinen Daten der PA-CONTROL ist der Unit Identifier 99 festgelegt worden.

Adresse	Function code	Beschreibung	Format / Type
0		Versionsinfo	16 Bit Integer
1		Betriebsart	16 Bit Integer
2		Systemfehlernummer	16 Bit Integer
3		Ablauffehlernummer	16 Bit Integer
4		1 = starte Betriebsart AUTOMATIK	16 Bit Integer
5		1 = stoppe Betriebsart AUTOMATIK	16 Bit Integer
6		1 = abbrechen Betriebsart AUTOMATIK	16 Bit Integer
7		1 = quittieren Systemfehler	16 Bit Integer
18		1 = starte Betriebsart SLAVE	16 Bit Integer
19		1 = abbrechen Betriebsart SLAVE	16 Bit Integer
1000		Text des Systemfehlers	Text, max. 255 Zeichen
2000		Text des Ablauffehlers	Text, max. 255 Zeichen
3000		Seriennummer	Text, max. 255 Zeichen
4000		Version des Betriebssystems	Text, max. 255 Zeichen
5000		In der Betriebsart Automatik, der Programmname des ersten TASKs	Text, max. 255 Zeichen
6000		In der Betriebsart Automatik, die aktuelle Programmzeile des ersten TASKs	Text, max. 255 Zeichen

4.1.1 Status- und Betriebszustandsmeldungen

Betriebsart	Betriebszustand	Weitere Informationen - Fehlerbeseitigung
0	keine Kommunikation	Steuerung im undefinierten Zustand
1	Grundstellung	Nach dem Einschalten
2	manuelle Bedienung	Bild: Diagnose, System , ... aktiv
3	manuelle Bedienung	Verfahren der Achsen
4	manuelle Bedienung	Editor oder Tool aktiv
9	Grundstellung mit Systemfehler	Ist in Grundstellung und ein Systemfehler steht an
10	Automatik gestartet	START-Befehl ausgelöst
11	Automatik läuft	Das als „START-Programm“ definierte Programm wird abgearbeitet
12	Automatik und „STOP“ erkannt	
13	Automatik und bearbeite „STOP“- Programm	Wartet auf ein Ereignis im STOP-Programm
15	Automatik und gestoppt	
15	Automatik und bearbeite Programm „START“ nach „STOP“	Wartet auf ein Ereignis im START-nach-STOP-Programm
17	Automatik und Störung erkannt	
18	Automatik und bearbeite „Störungs-Programm“	Wartet auf ein Ereignis im Störungs-Programm
19	Automatik mit Störung und gestoppt	Das als „START-Programm“ definierte Programm ist gestoppt, kein weiteres Programm wird abgearbeitet.
20	manuelles Verfahren über Serielle Schnittstelle	Motor bzw. Achse wird über WINPAC manuell gefahren
31	SLAVE-Betriebsart läuft	

Erstellt von: Grieshaber Edmund | APP5030_DE_1120445_PA-CONTROL_ModbusTCP_Kommunikation_R1d.doc

4.2 Achsdaten

Für den Zugriff auf die Achsen ist der Unit Identifier 1 bis 16 festgelegt worden.

Die Achsdaten können unabhängig von der Betriebsart der PA-CONTROL immer gelesen werden. Das Schreiben der Achsdaten ist nur in der Betriebsart ONLINE sinnvoll.

4.2.1 Achsdaten lesen

Adresse	Function code	Beschreibung	Format / Type
0	Read Input Register Read Holding Register	Type der Achse	16 Bit Integer
1		1 = Achse bereit	16 Bit Integer
2		Status positiver Endschalter	16 Bit Integer
3		Status negativer Endschalter	16 Bit Integer
4		Achse ist referenziert	16 Bit Integer
5		Status der Achse	16 Bit Integer
6		Aktuelle Position der Achse ()	32 Bit, Kommazahl
8		Achsparameter Verfahrbereichsminimum ()	32 Bit, Kommazahl
10		Achsparameter Verfahrbereichsmaximum ()	32 Bit, Kommazahl
12		Achsparameter Verfahrgeschwindigkeit	32 Bit, Kommazahl
14		Achsparameter Referenzgeschwindigkeit ()	32 Bit, Kommazahl
16		Achsparameter Manuellgeschwindigkeit ()	32 Bit, Kommazahl
18		Achsparameter Schleichganggeschwindigkeit ()	32 Bit, Kommazahl
20		Achsparameter Anzeigeeinheit	16 Bit Integer
21		Achsparameter aktuelle Verfahrgeschwindigkeit ()	32 Bit, Kommazahl
23		SLAVE-Betriebsart: Achse-Fehler-Nummer	16 Bit Integer
24		SLAVE-Betriebsart: Kommando-Fehler Nummer	16 Bit Integer
25		SLAVE-Betriebsart: aktuelles Kommando	16 Bit Integer
26		SLAVE-Betriebsart: Status des aktuellen Kommandos	16 Bit Integer

Adresse	Function code	Beschreibung	Format / Type
27		Status der Achseingänge A1.I4097, A2.I4161 ...	16 Bit Unsigned Int.
28		Status der Achseingänge A1.I4113, A2.I4177	16 Bit Unsigned Int.
29		Status der Achseingänge A1.I4129	16 Bit Unsigned Int.
30		Status der Achseingänge A1.I4145	16 Bit Unsigned Int.
31		Hardwarefreigabe der Achse vorhanden (AchsEA-Eingang Input OFF+ Reglerfreigabe + Endstufenfreigabe)	16 Bit Integer (bool)
32		Achse ist "Initialisiert"	16 Bit Integer (bool)
33		Achse ist "Disabled"	16 Bit Integer (bool)
34		Achse ist "OFF"	16 Bit Integer (bool)
35		Achse ist "ON"	16 Bit Integer (bool)
36		Achse "Läuft"	16 Bit Integer (bool)
37		Achse ist "In Position"	16 Bit Integer (bool)
38		Achse ist "Gestoppt"	16 Bit Integer (bool)
39		Achse ist "Gestört"	16 Bit Integer (bool)

4.2.2 Status der Achse

Wert	Status der Achse	Weitere Informationen
0	INIT	
1	DISABLED	
2	SAFE	
3	SAFE_W_DISABLED	
4	SAFE_W_POERATIONAL	
5	SAFE_W_ACTIVE	
6	IDLE	
7	IDLE_W_DIABLED	
8	IDLE_W_OPERATIONAL	
9	IDLE_W_ACTIVE	
10	OPERATIONAL	
11	ACTIVE	
12	HALT	
13	HALT_W_DISABLED	
14	HALT_W_OPERATIONAL	
15	HALT_W_ACTIVE	
16	FAULT	

4.3 N-Register

Für den Zugriff auf die N-Register ist der Unit Identifier 100 festgelegt worden.

Adresse	Function code	Beschreibung	Format / Type
0	Read Holding Register Write Multiple Register	N1	32 Bit Integer
2		N2	
4		N3	
16382		N8192	

4.4 R-Register

Für den Zugriff auf die R-Register ist der Unit Identifier 101 festgelegt worden.

Adresse	Function code	Beschreibung	Format / Type
0	Read Holding Register Write Multiple Register	R1	32 Kommazahl
2		R2	
4		R3	
16382		R8192	

4.5 PA-CONTROL Strings

Für den Zugriff auf die PA-CONTROL-Strings ist der Unit Identifier 103 festgelegt worden.
Für den wortorganisierten Zugriff (Beckhoff-SPS, ...) auf die PA-CONTROL-Strings ist der Unit Identifier 105 festgelegt worden.

Adresse	Function code	Beschreibung	Format / Type
1000	Read Holding Register Write Multiple Register	S1	Text, max. 80 Zeichen
2000		S2	
3000		S3	
16000		S16	

4.6 Digitale Eingänge

Für den Zugriff auf die digitalen Eingänge ist der Unit Identifier 110 festgelegt worden.

Adresse	Function code	Beschreibung	Format / Type
0	Read Discrete Inputs	I1	16 Bit Integer
1		I2	
2		I3	
3		...	

Erstellt von: Grieshaber Edmund | APP5030_DE_1120445_PA-CONTROL_ModbusTCP_Kommunikation_R1d.doc

4.7 Digitale Ausgänge

Für den Zugriff auf die digitalen Ausgänge ist der Unit Identifier 120 festgelegt worden.

Adresse	Function code	Beschreibung	Format / Type
0	Read Holding Register Write Single Register Write Multiple Register	O1	16 Bit Integer
1		O2	
2		O3	
3		...	

4.8 Merker

Für den Zugriff auf die Merker ist der Unit Identifier 130 festgelegt worden.

Adresse	Function code	Beschreibung	Format / Type
0	Read Holding Register Write Single Register Write Multiple Register	M1	16 Bit Integer
1		M2	
2		M3	
3		...	

4.9 System Merker

Für den Zugriff auf die System-Merker ist der Unit Identifier 160 festgelegt worden.

Adresse	Function code	Beschreibung	Format / Type
0	Read Holding Register	SM1	16 Bit Integer
1		SM2	

4.10 System N-Register

Für den Zugriff auf die System-N-Register ist der Unit Identifier 161 festgelegt worden.

Adresse	Function code	Beschreibung	Format / Type
0	Read Holding Register	SN1	32 Bit Integer
1		SN2	

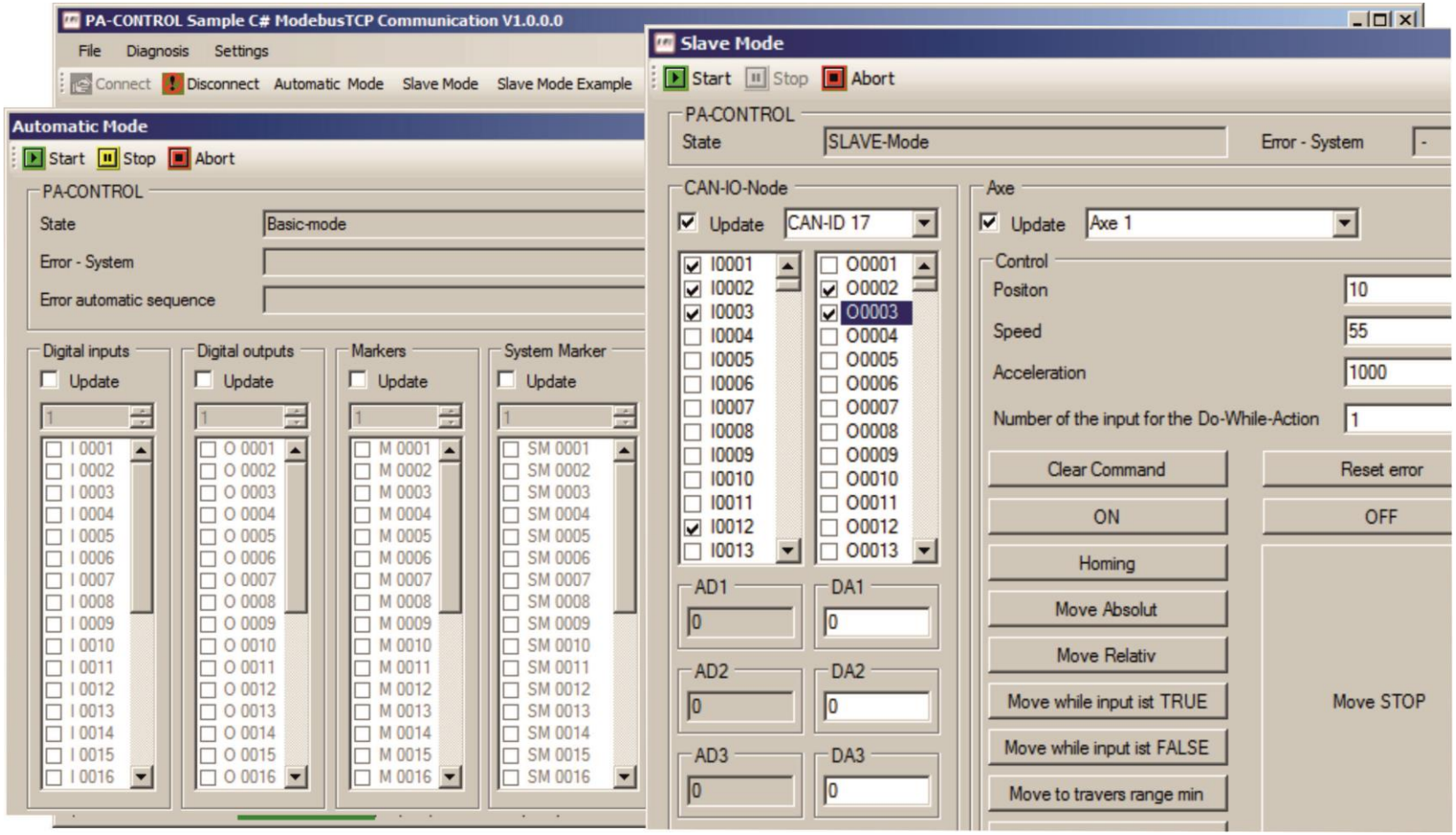
4.11 System R-Register

Für den Zugriff auf die System-R-Register ist der Unit Identifier 162 festgelegt worden.

Adresse	Function code	Beschreibung	Format / Type
0	Read Holding Register	SR1	32 Bit Kommazahl
1		SR2	

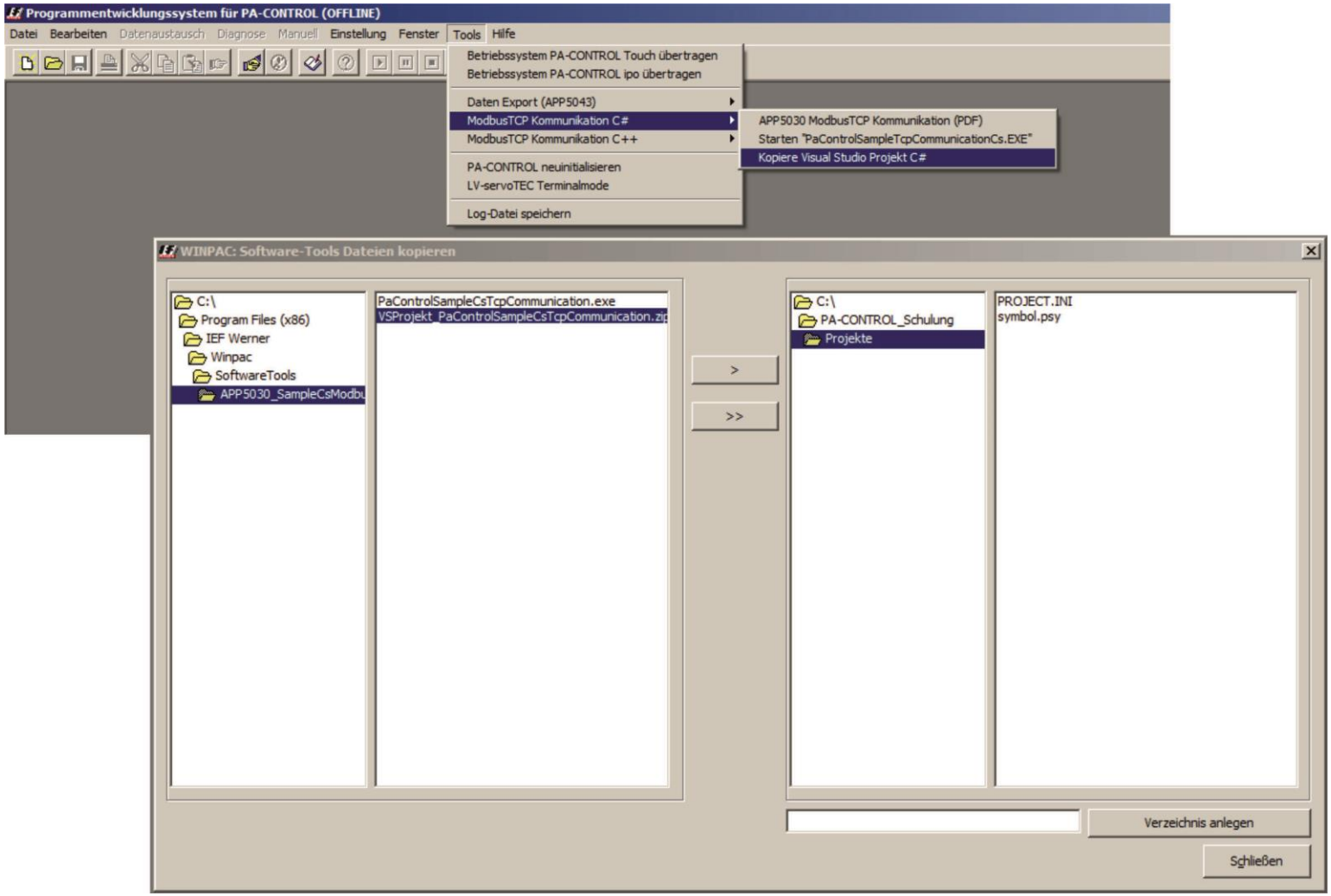
5 Beispiel C#-Programm

Das C# Projekt " PA-CONTROLSampleC#ModebusTCPCommunication" dient als Beispiel für die Kommunikation mit der PA-CONTROL.

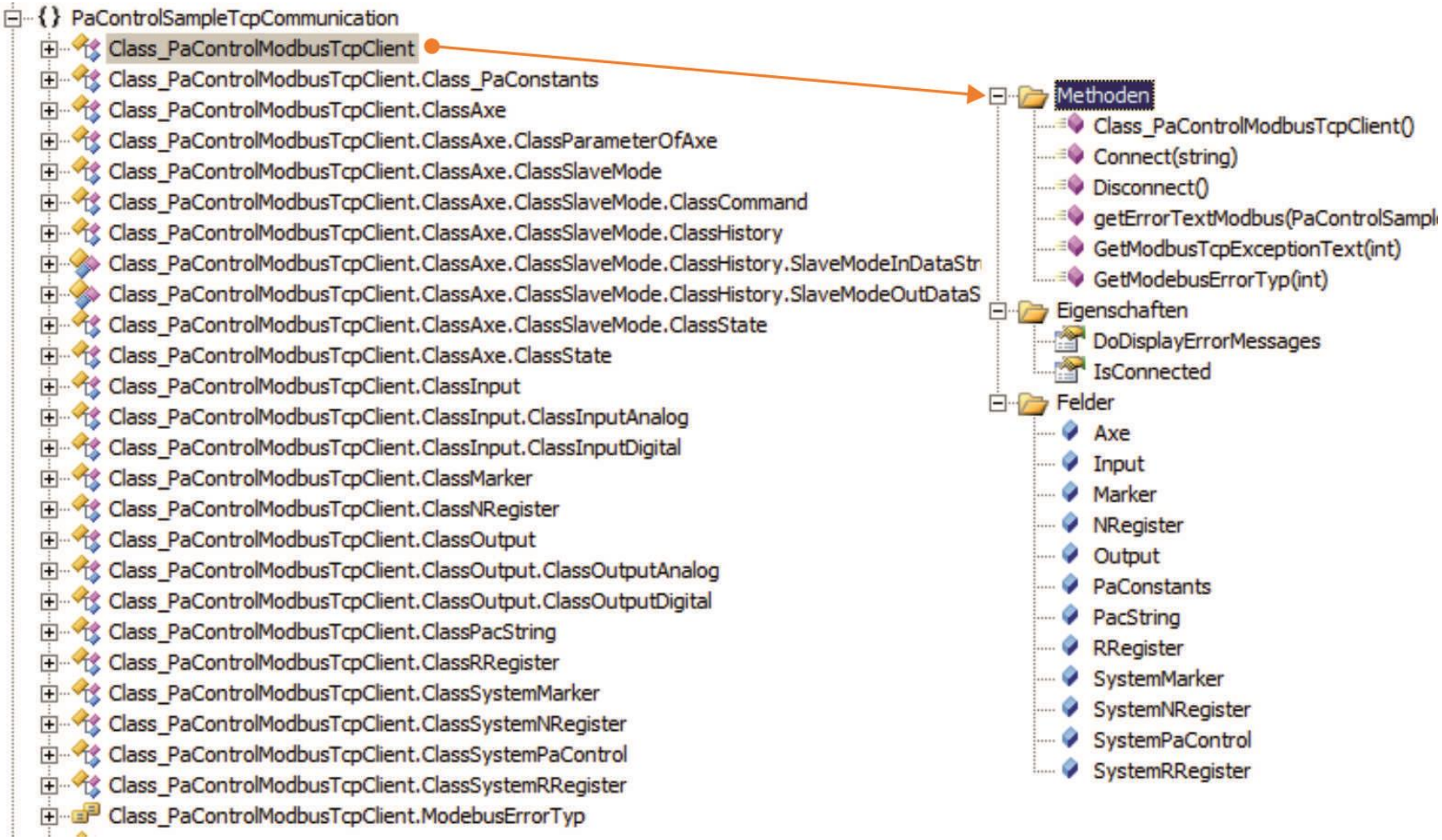


5.1 Visual Studio Project

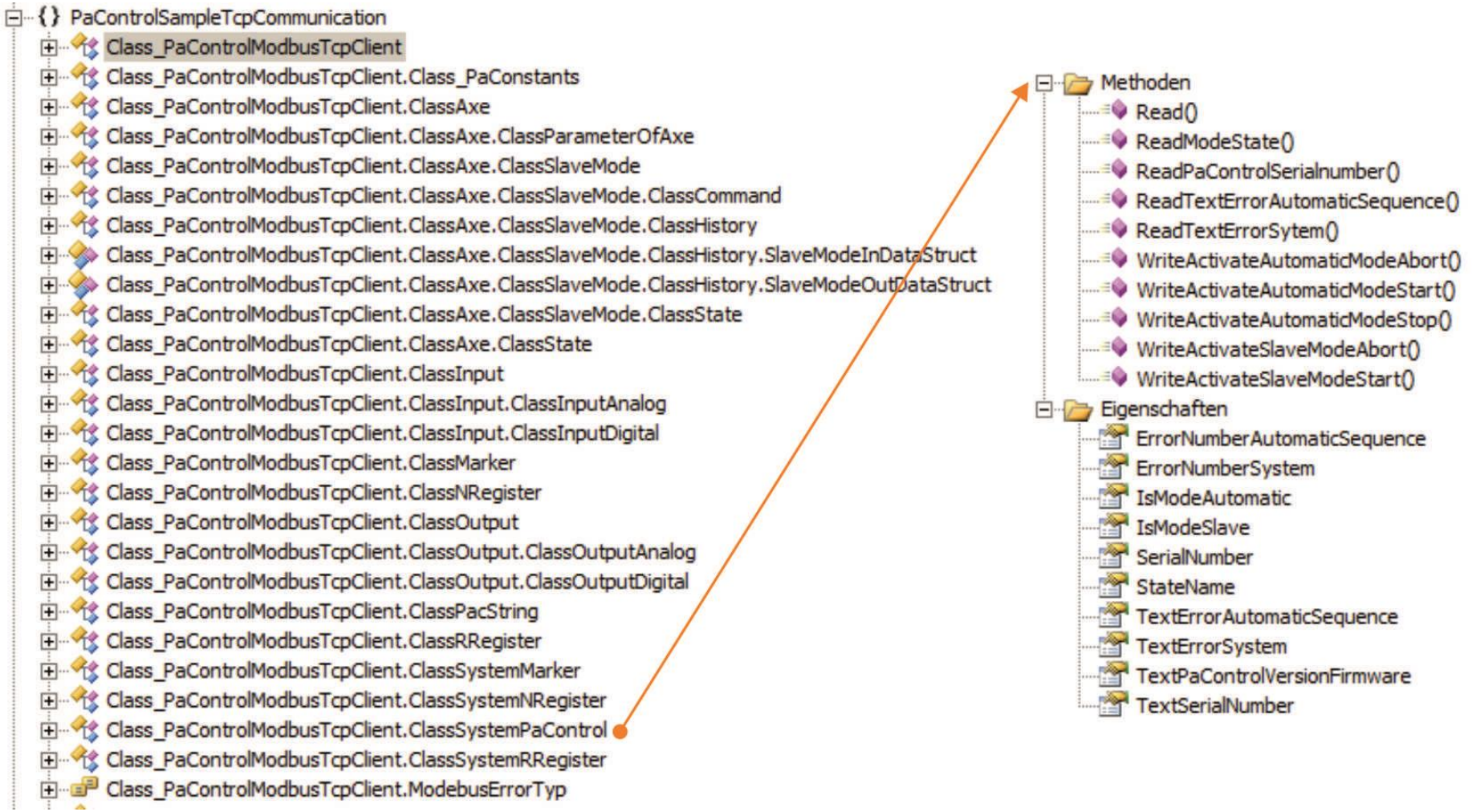
Bei der Installation von WINPAC wurde die EXE-Datei und das Visual-Studio-Projekt als ZIP-Datei mit aufgespielt und kann kopiert werden.



5.2 Klasse Class_PaControlModbusTcpClient

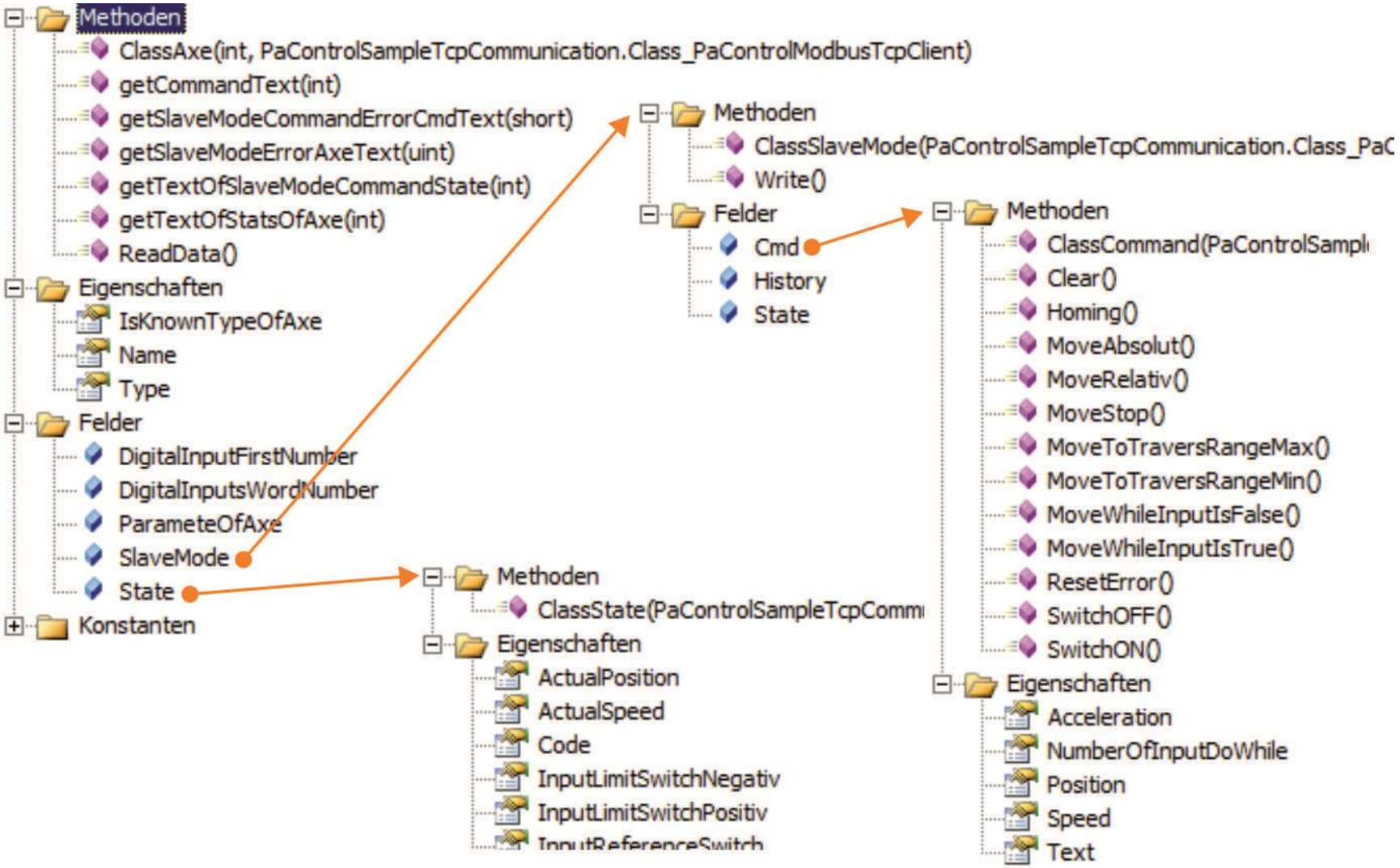


5.2.1 Klasse Class_PaControlModbusTcpClient.ClassSystemPaControl



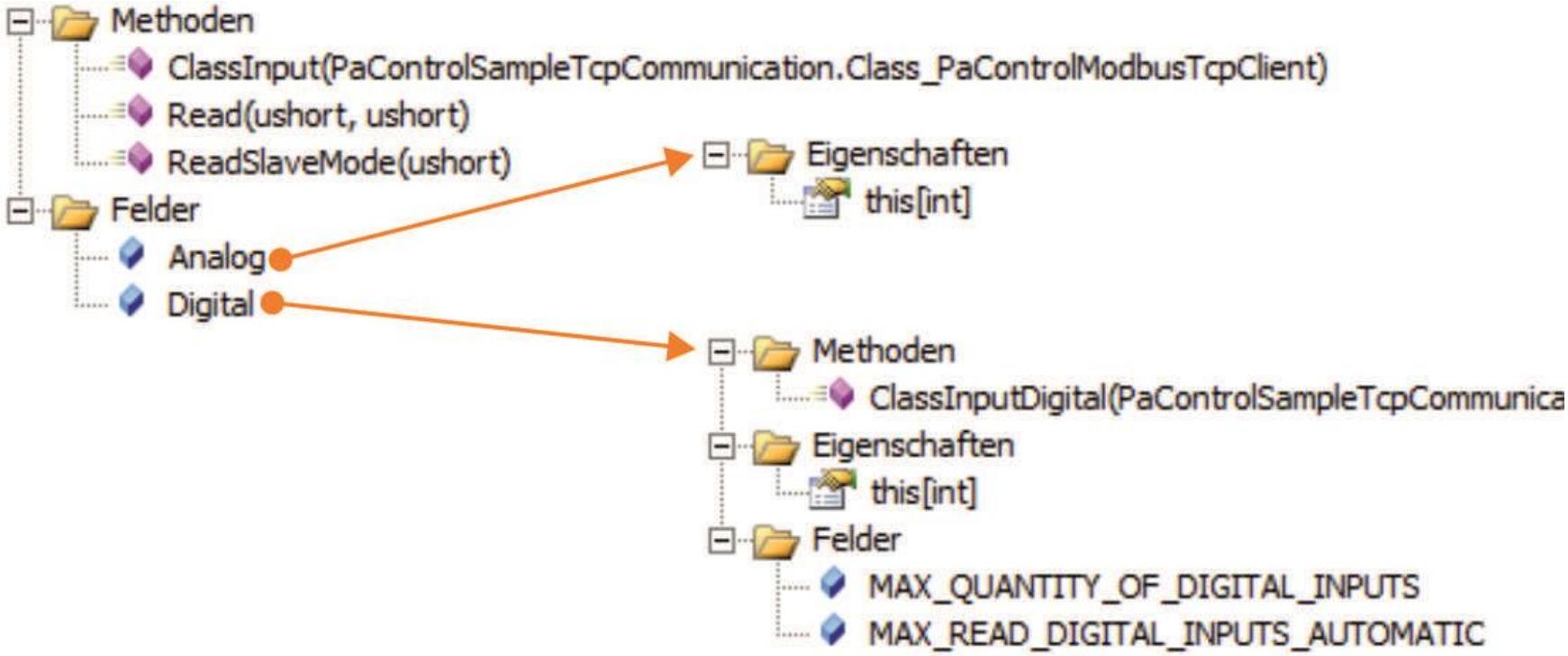
Erstellt von: Grieshaber Edmund | APP5030_DE_1120445_PA-CONTROL_ModbusTCP_Kommunikation_R1d.doc

5.2.2 Klasse Class_PaControlModbusTcpClient.ClassAxe



Erstellt von: Grieshaber Edmund | APP5030_DE_1120445_PA-CONTROL_ModbusTCP_Kommunikation_R1d.doc

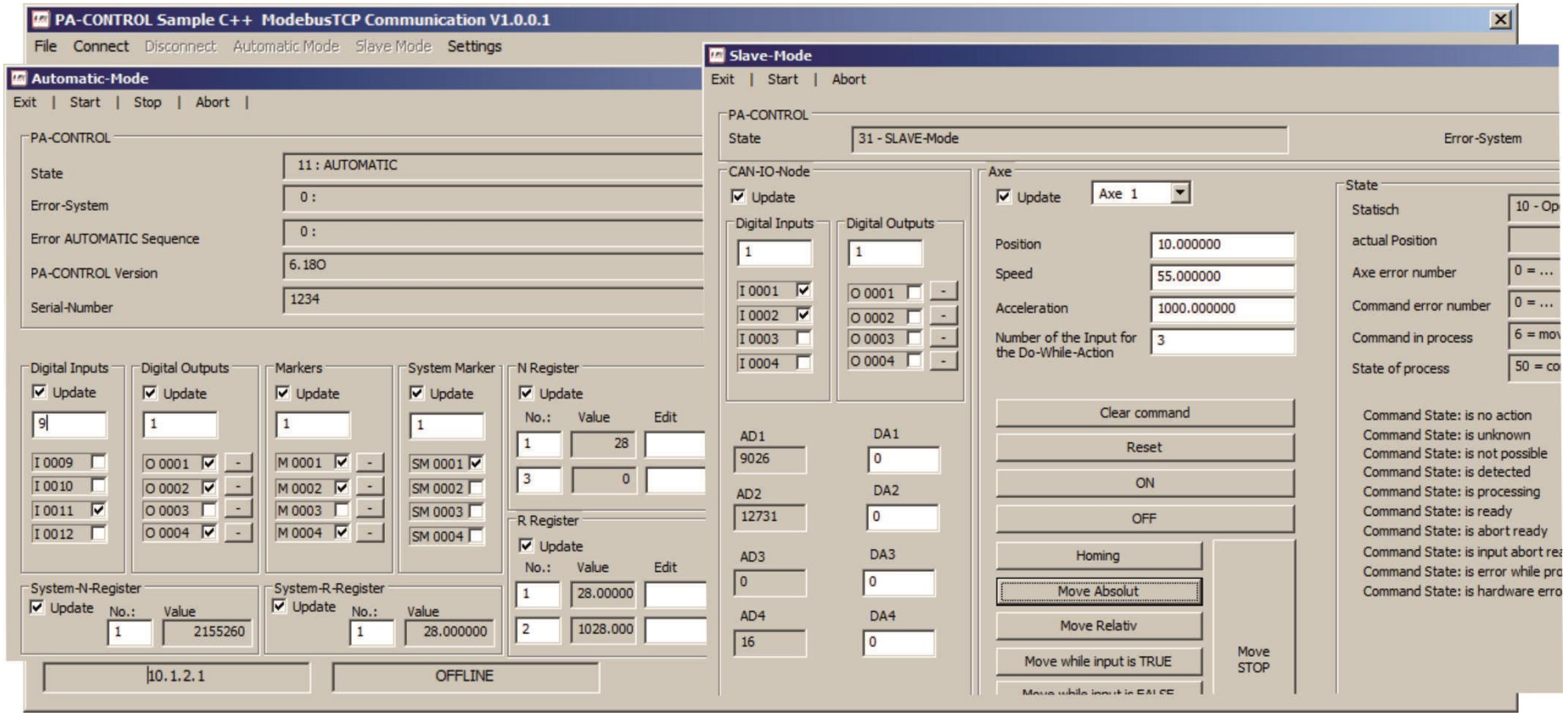
5.2.3 Klasse Class_PaControlModbusTcpClient.ClassInput



Erstellt von: Grieshaber Edmund | APP5030_DE_1120445_PA-CONTROL_ModbusTCP_Kommunikation_R1d.doc

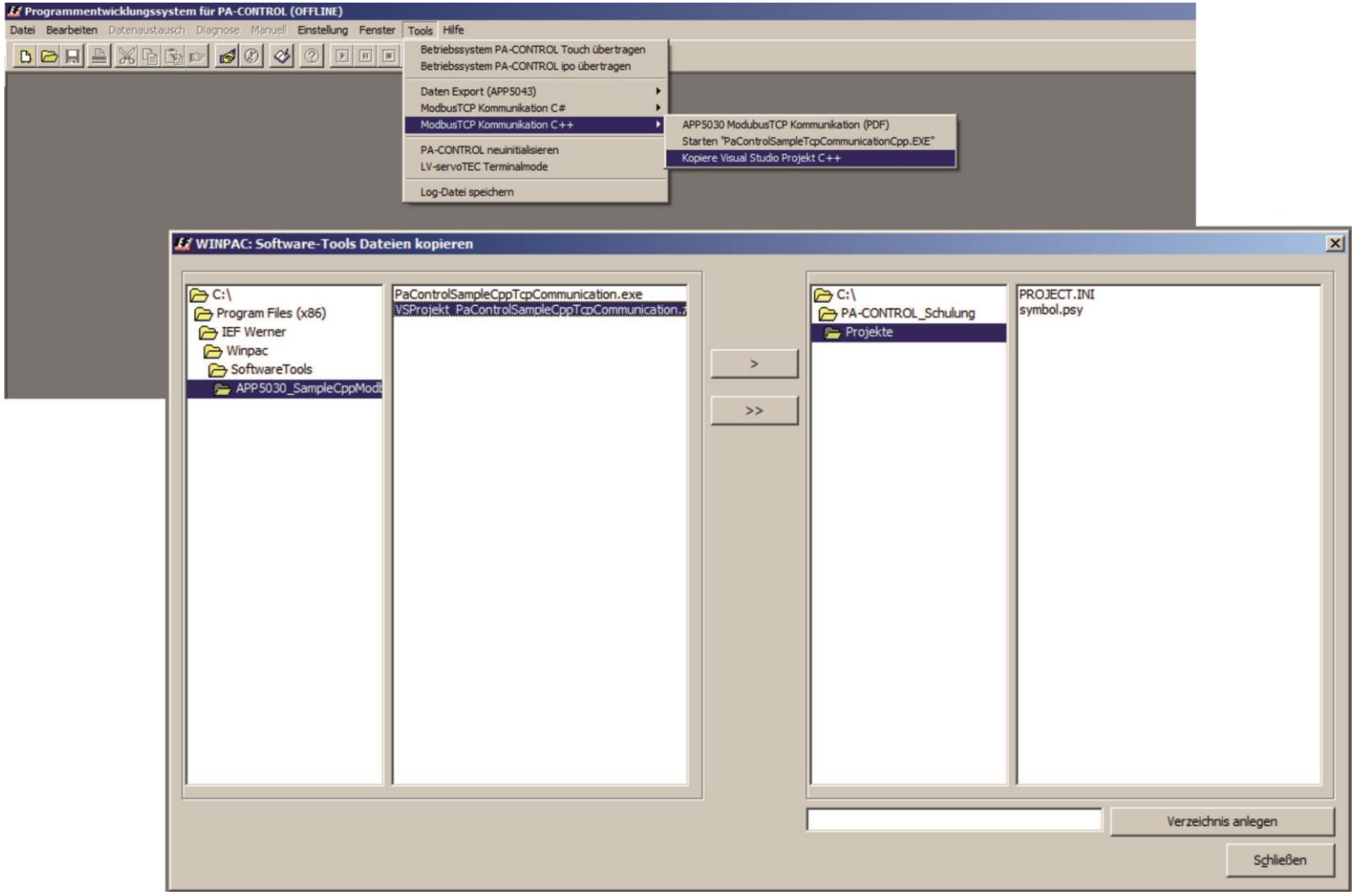
6 Beispiel C++

Das C++ Projekt "PA-CONTROLSampleC++TCPCommunication" dient als Beispiel für die Kommunikation mit der PA-CONTROL.



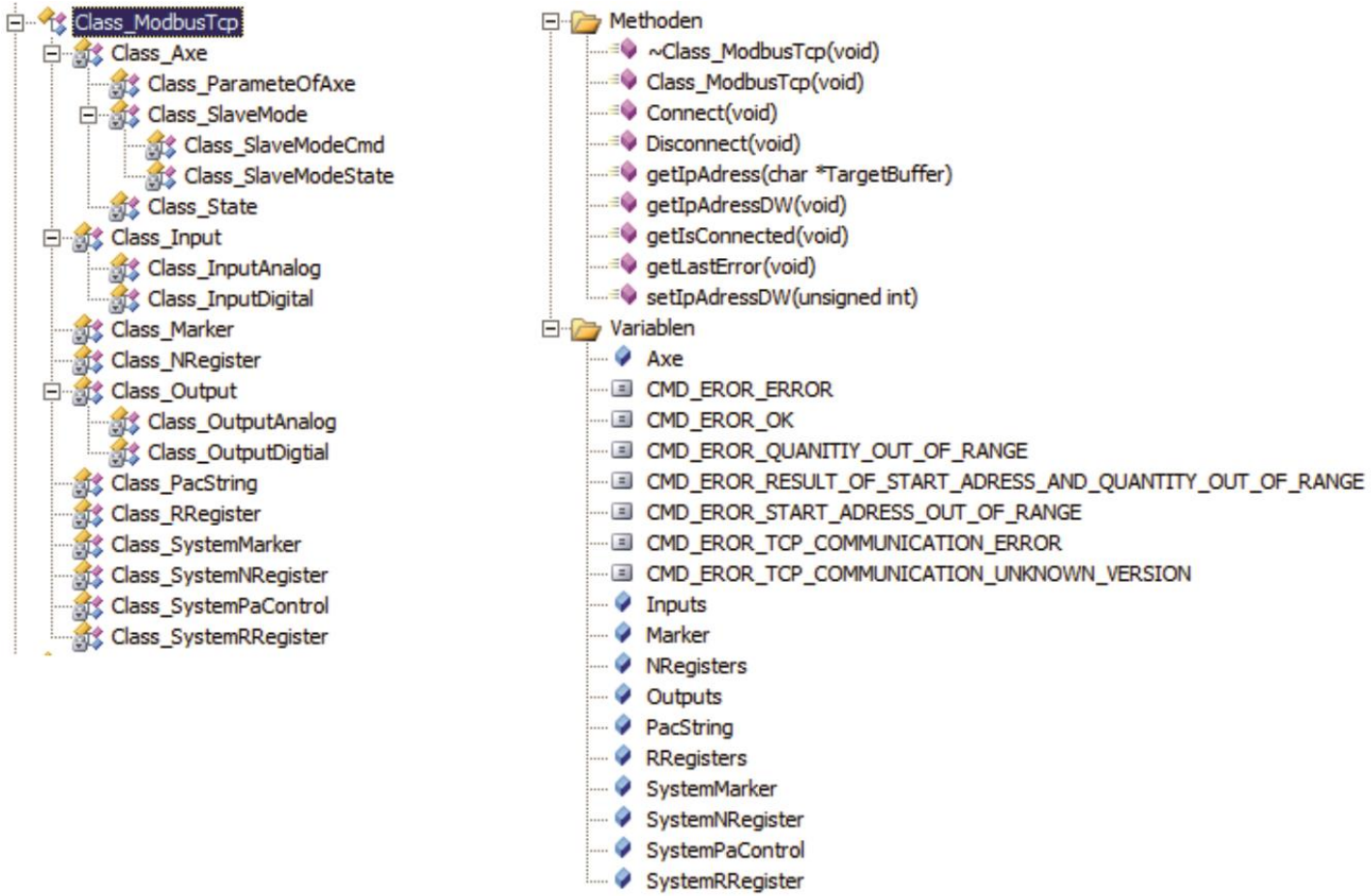
6.1 Visual Studio Project

Bei der Installation von WINPAC wurde die EXE-Datei und das Visual-Studio-Projekt als ZIP-Datei mit aufgespielt und kann kopiert werden.



6.2 Klasse Class_ModbusTCP

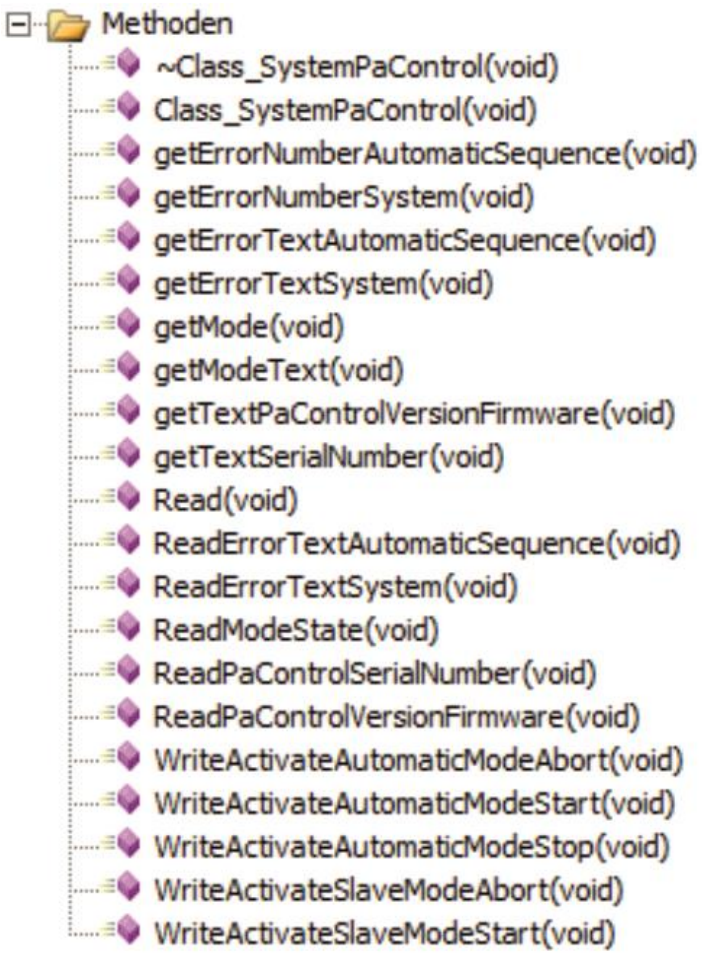
Die Klasse "Class_ModbusTCP" kann in ein C++ Programm eingebunden werden.



Als Beispiel für die Verwendung der Klasse dient das C++ Projekt "PaControlSampleC++TcpCommunication".

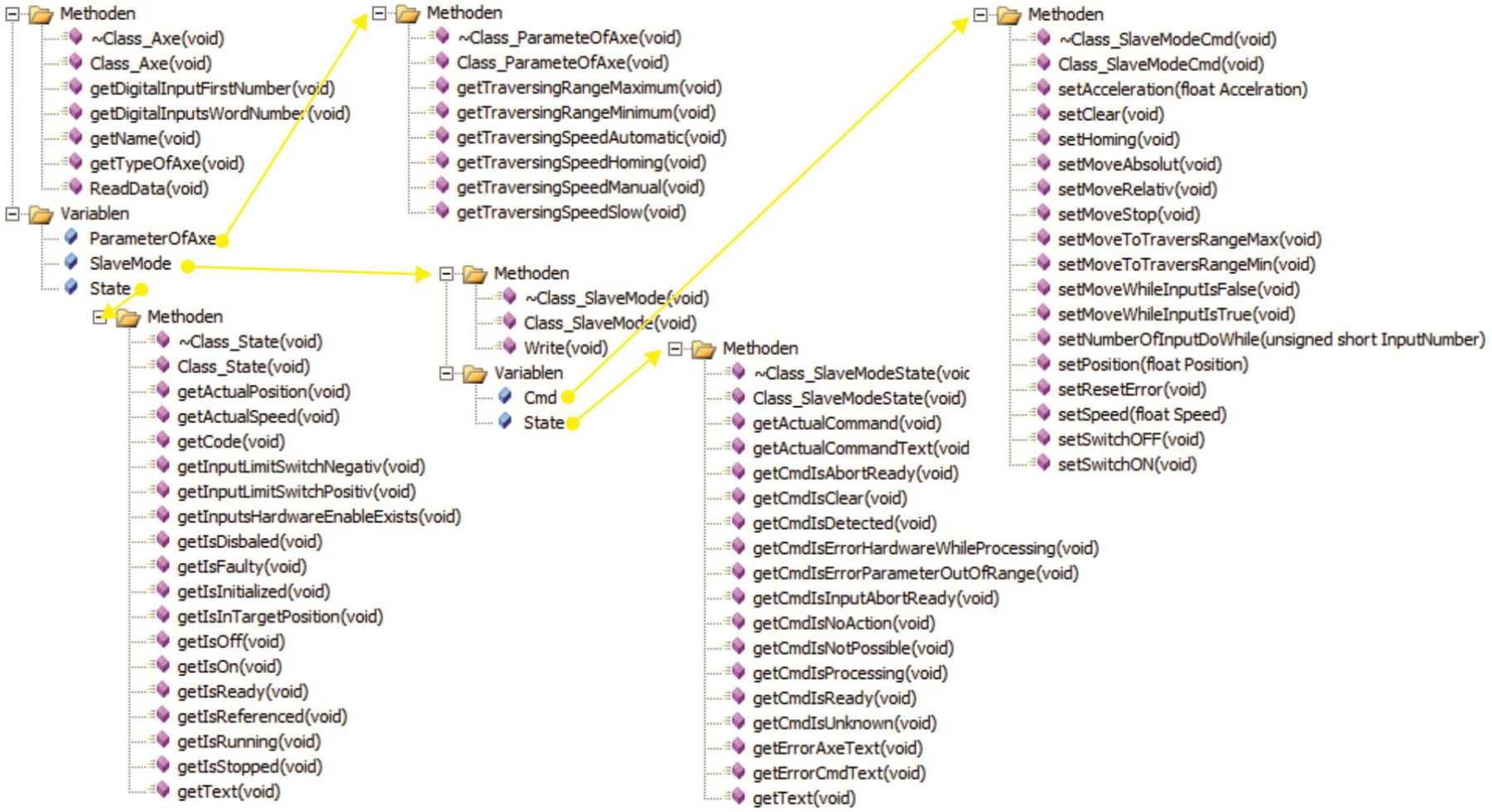
6.2.1 Klasse Class_ModbusTcp.SystemPaControl

Über die Unterklasse "SystemPaControl" wird zum Beispiel die Betriebsart "AUTOMATIK" gestartet, gestoppt und abgebrochen.



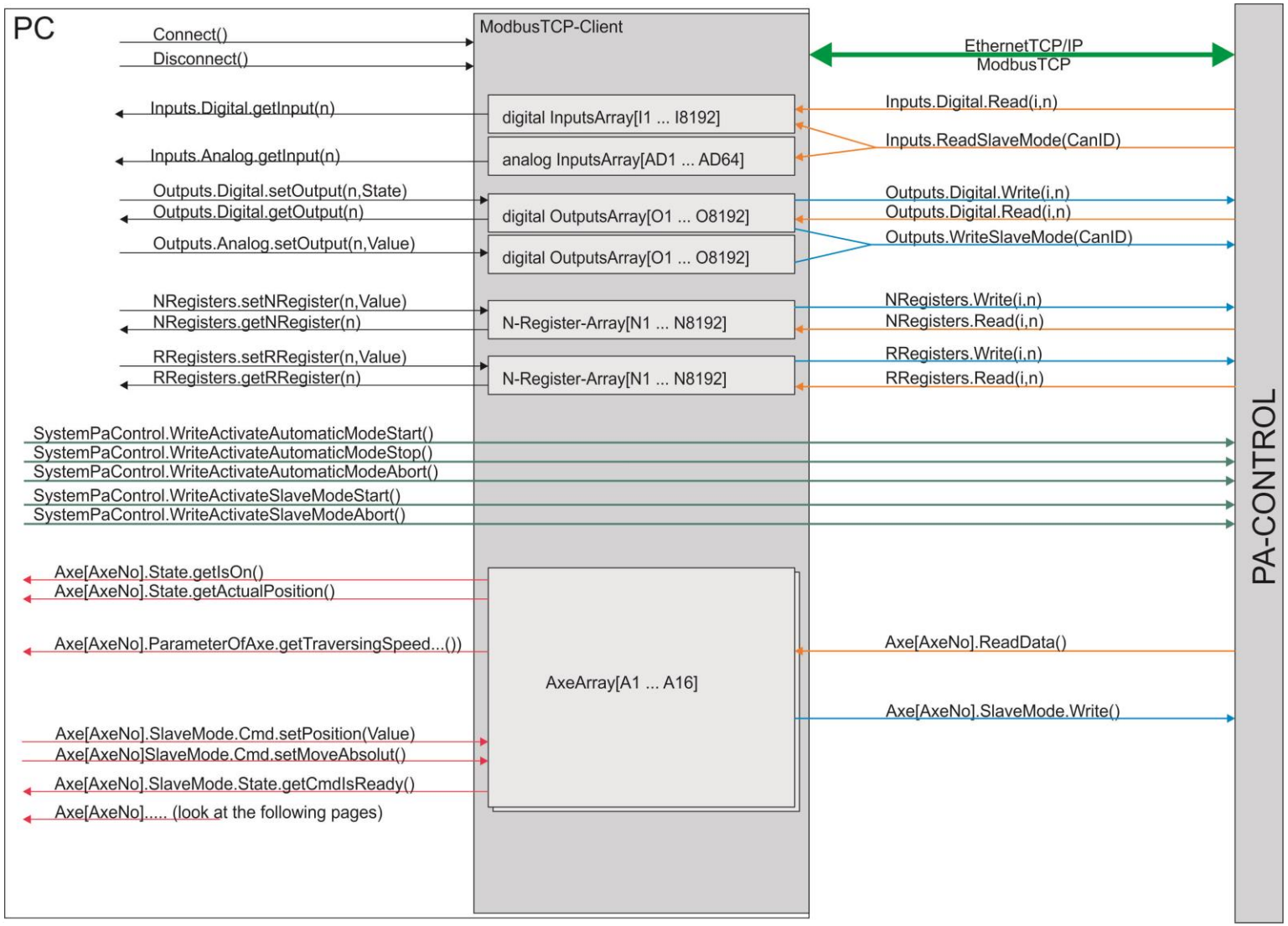
6.2.2 Klasse Class_ModbusTcp.Axe

Die Kommunikation mit den Achsen erfolgt über die Unterklasse "Axe".



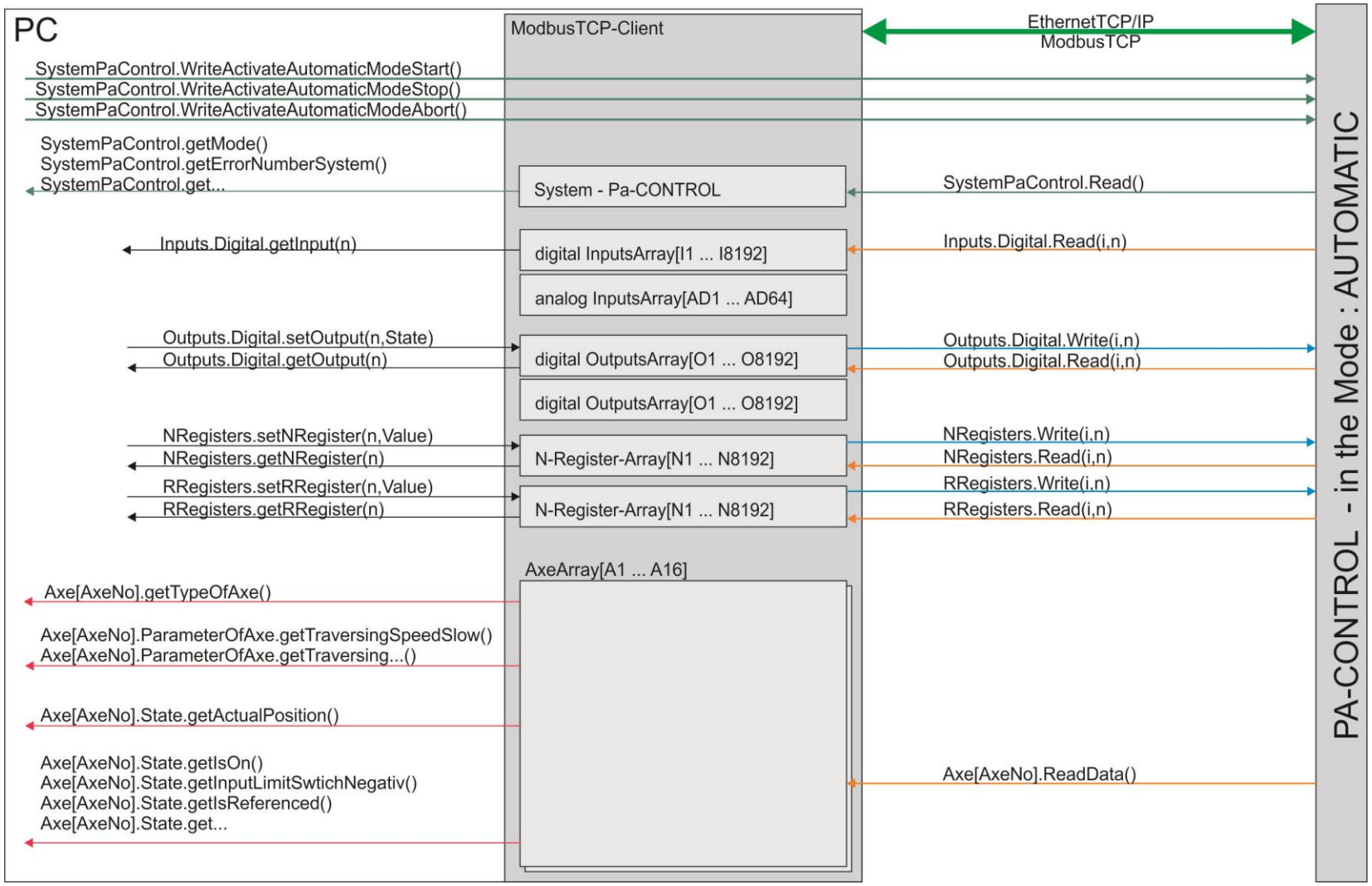
Erstellt von: Grieshaber Edmund | APP5030_DE_1120445_PA-CONTROL_ModbusTCP_Kommunikation_R1d.doc

6.3 Skizze zur Kommunikation (Übersicht)



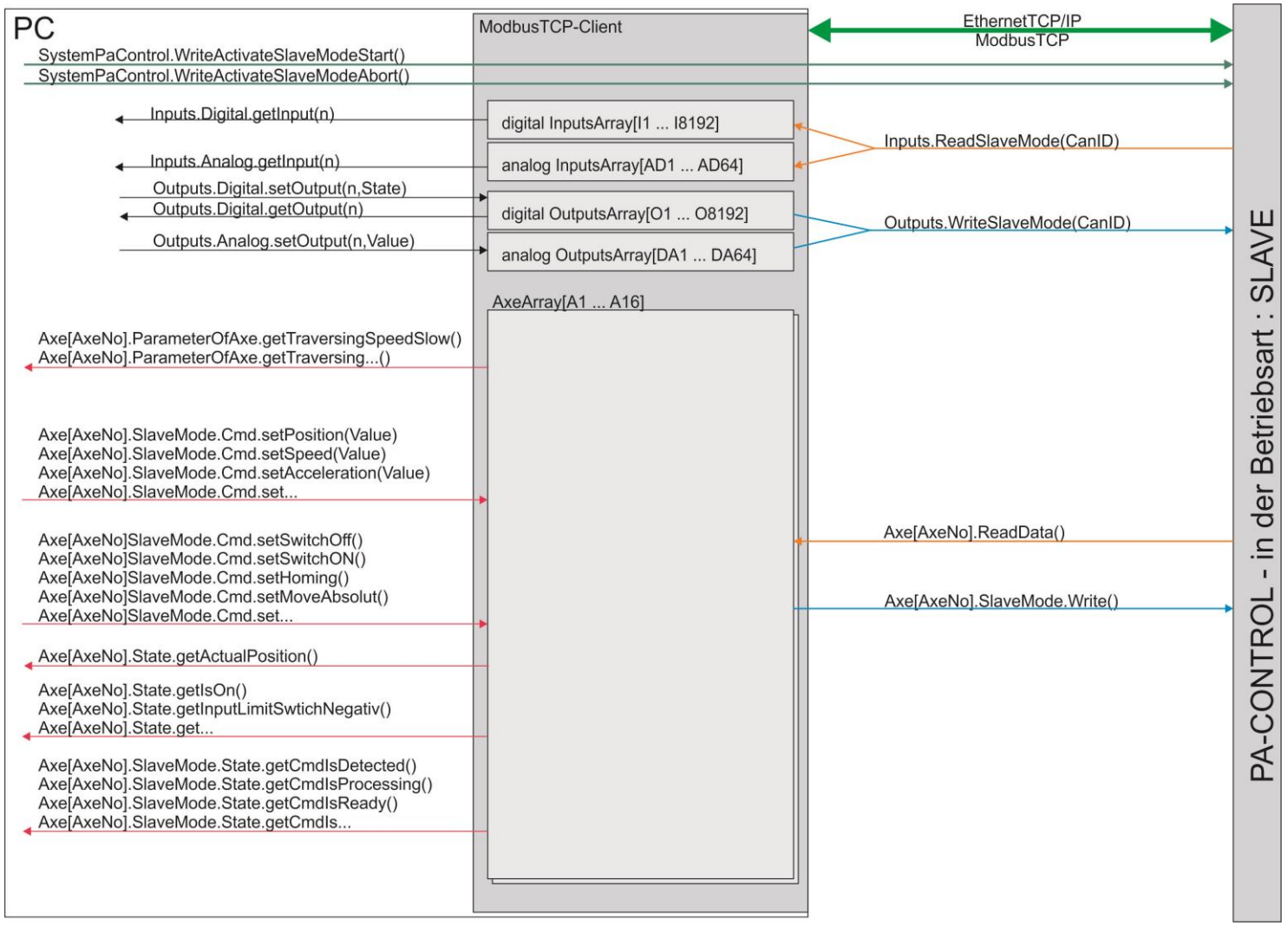
Erstellt von: Grieshaber Edmund | APP5030_DE_1120445_PA-CONTROL_ModbusTCP_Kommunikation_R1d.doc

6.3.1 Kommunikation in der Betriebsart AUTOMATIK



Erstellt von: Grieshaber Edmund | APP5030_DE_1120445_PA-CONTROL_ModbusTCP_Kommunikation_R1d.doc

6.3.2 Kommunikation in der Betriebsart SLAVE



Erstellt von: Grieshaber Edmund | APP5030_DE_1120445_PA-CONTROL_ModbusTCP_Kommunikation_R1d.doc

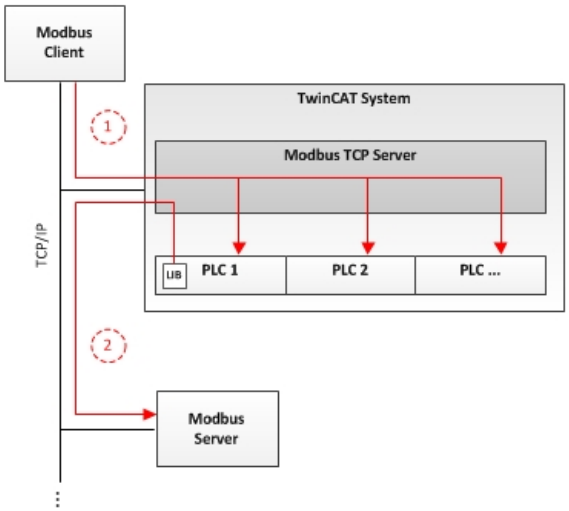
7 Beispiel Beckhoff TwinCAT Programm

Übersicht

Das Supplement ermöglicht über eine Netzwerkverbindung (TCP/IP) über das offene Protokoll Modbus zu kommunizieren.

Modbus ist ein offener Industriestandard der von der unabhängigen Modbus Organization gepflegt und verwaltet wird.

Das Protokoll basiert auf einer Client/Server-Architektur. Das Supplement bietet die Möglichkeiten als Modbus Client oder Server zu dienen:



Server-Funktionalität:

(1) Der TwinCAT Modbus TCP Server ermöglicht über das Netzwerk auf den vollständigen Speicherbereich der TwinCAT SPS zu zugreifen.

Client-Funktionalität:

(2) Die mitgelieferte SPS-Bibliothek ermöglicht es, mit weiteren Modbus-Teilnehmern zu kommunizieren (2) , um Informationen auszutauschen (z.B. Messwerte, Zustände) und zu steuern.

TwinCAT MODBUS TCP Server : Modbus Client-Funktionalität

Übersicht

Die im Modbus-Protokoll definierten Funktionen sind als SPS-Bausteine in der Bibliothek TcModbusSrv.lib realisiert.

Modbus TCP Funktion	Funktions-Code	SPS-Baustein
Read Coils	1	FB_MBReadCoils
Read Inputs	2	FB_MBReadInputs
Read Registers	3	FB_MBReadRegs
Read Input Registers	4	FB_MBReadInputRegs
Write Single Coil	5	FB_MBWriteSingleCoil
Write Single Register	6	FB_MBWriteSingleReg
Write Multiple Coils	15	FB_MBWriteCoils
Write Multiple Registers	16	FB_MBWriteRegs
Read/Write Multiple Registers	23	FB_MBReadWriteRegs
Diagnose	8	FB_MBDiagnose

Modbus UDP Funktion	Funktions-Code	SPS-Baustein
Read Coils	1	FB_MBUdpReadCoils
Read Inputs	2	FB_MBUdpReadInputs
Read Registers	3	FB_MBUdpReadRegs
Read Input Registers	4	FB_MBUdpReadInputRegs
Write Single Coil	5	FB_MBUdpWriteSingleCoil
Write Single Register	6	FB_MBUdpWriteSingleReg
Write Multiple Coils	15	FB_MBUdpWriteCoils
Write Multiple Registers	16	FB_MBUdpWriteRegs
Read/Write Multiple Registers	23	FB_MBUdpReadWriteRegs
Diagnose	8	FB_MBUdpDiagnose

7.1 Beispiel: „PAC_ModbusTCP_Vx-xx.pro“

Im Beispielprogramm sind für den Zugriff auf die Register und Merker der PA-CONTROL Funktionsblöcke vorhanden.

Für die Werte der Register und Merker wurden Arrays bereitgestellt.

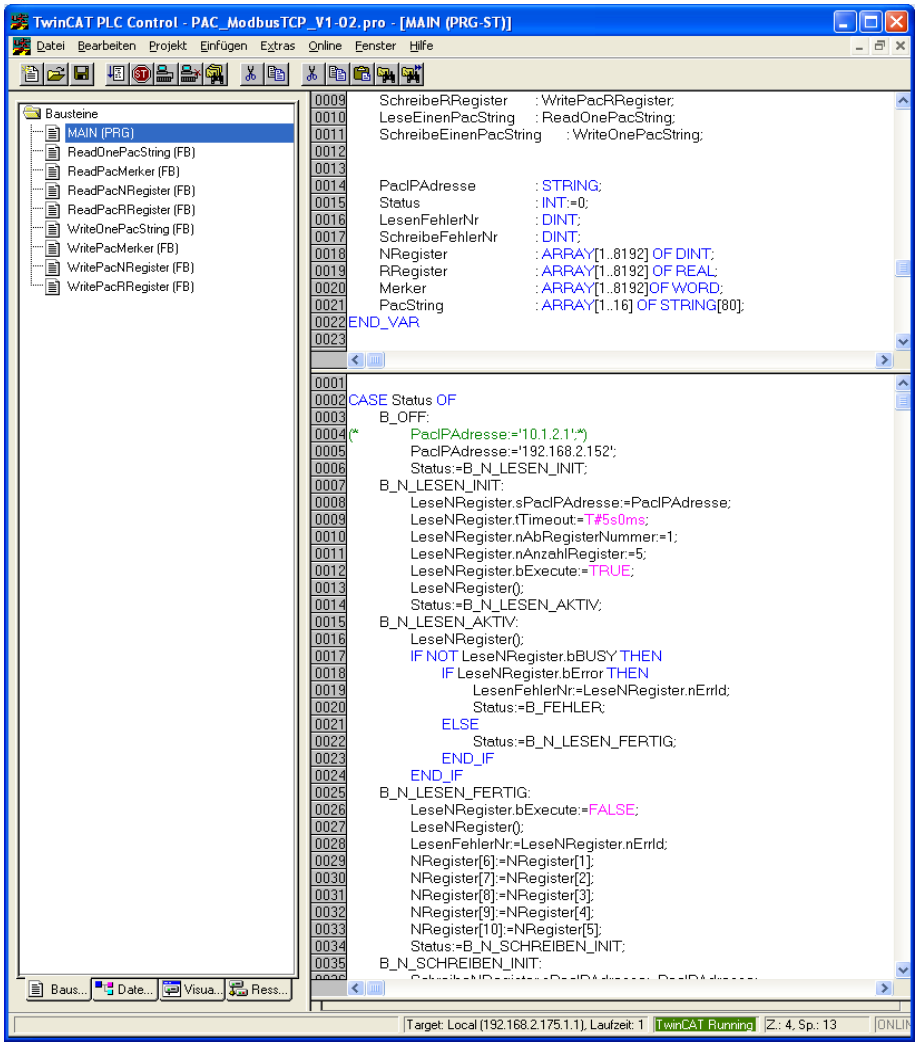
NRegister: ARRAY[1..8192] OF DINT;

RRegister: ARRAY[1..8192] OF REAL;

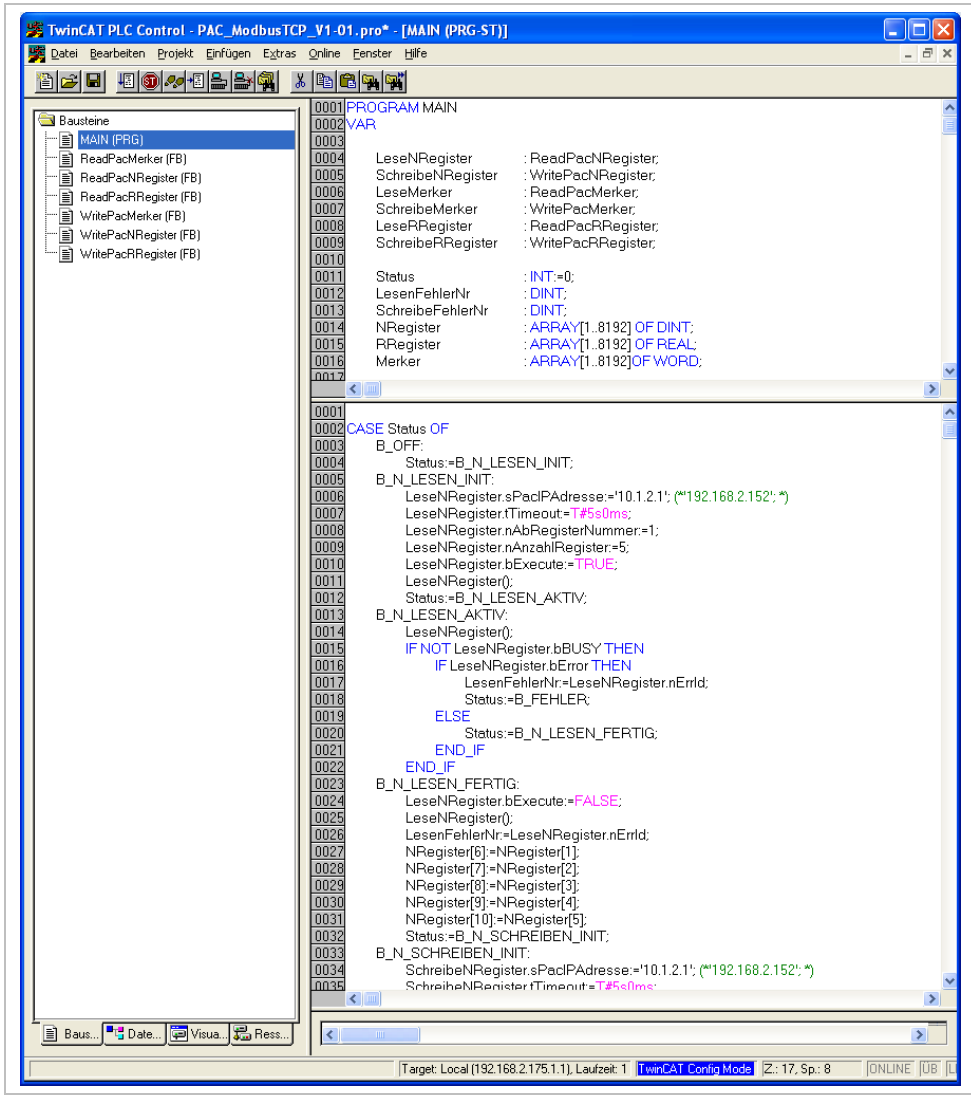
Merker: ARRAY[1..8192]OF WORD;

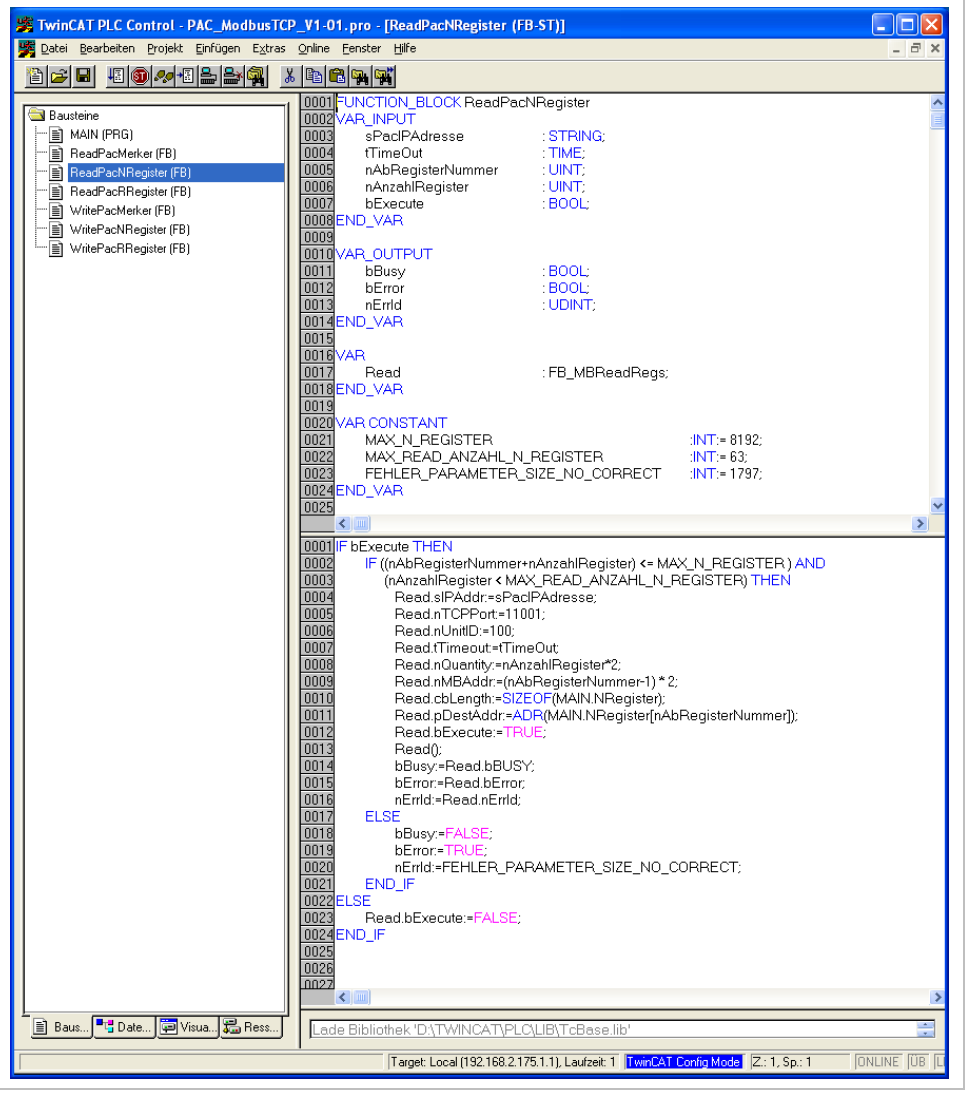
PacString: ARRAY[1..16]OF STRING[80];

Die Werte werden beim Lesen in diese Arrays eingetragen und beim Schreiben aus diesen Arrays herausgeholt.

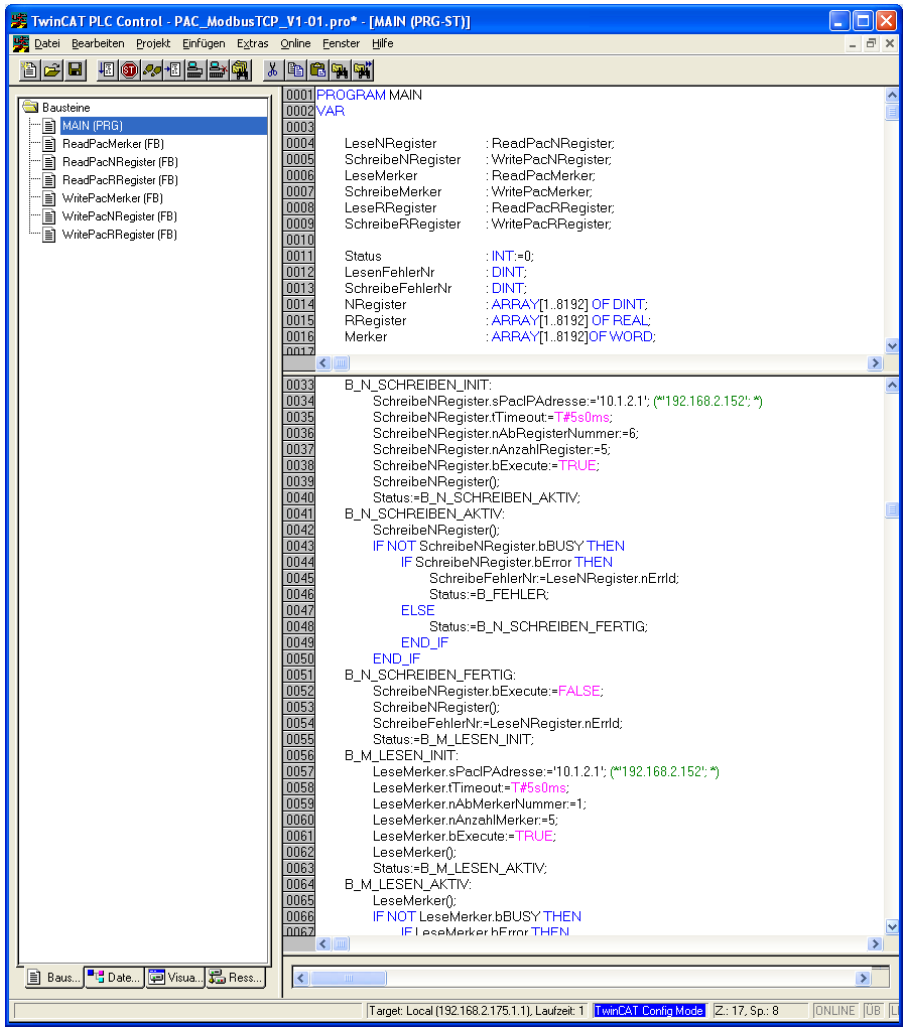


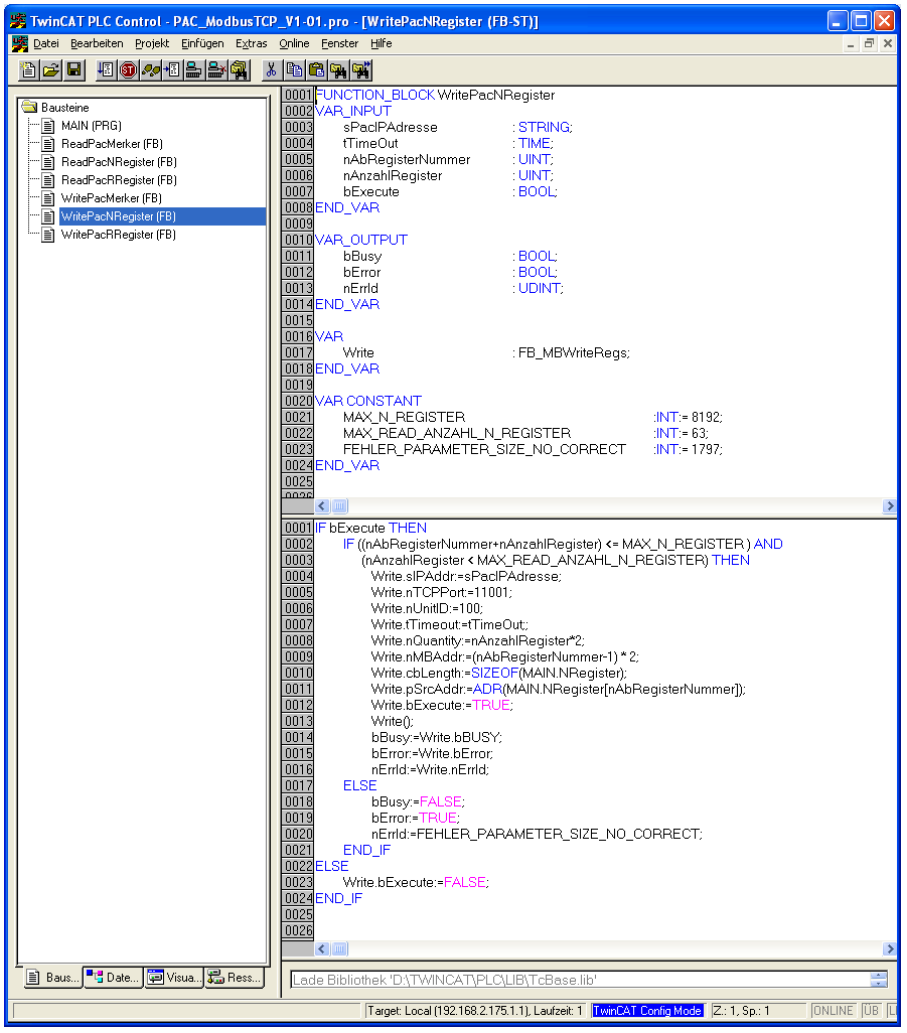
7.1.1 Funktionsbaustein ReadPacNRegister





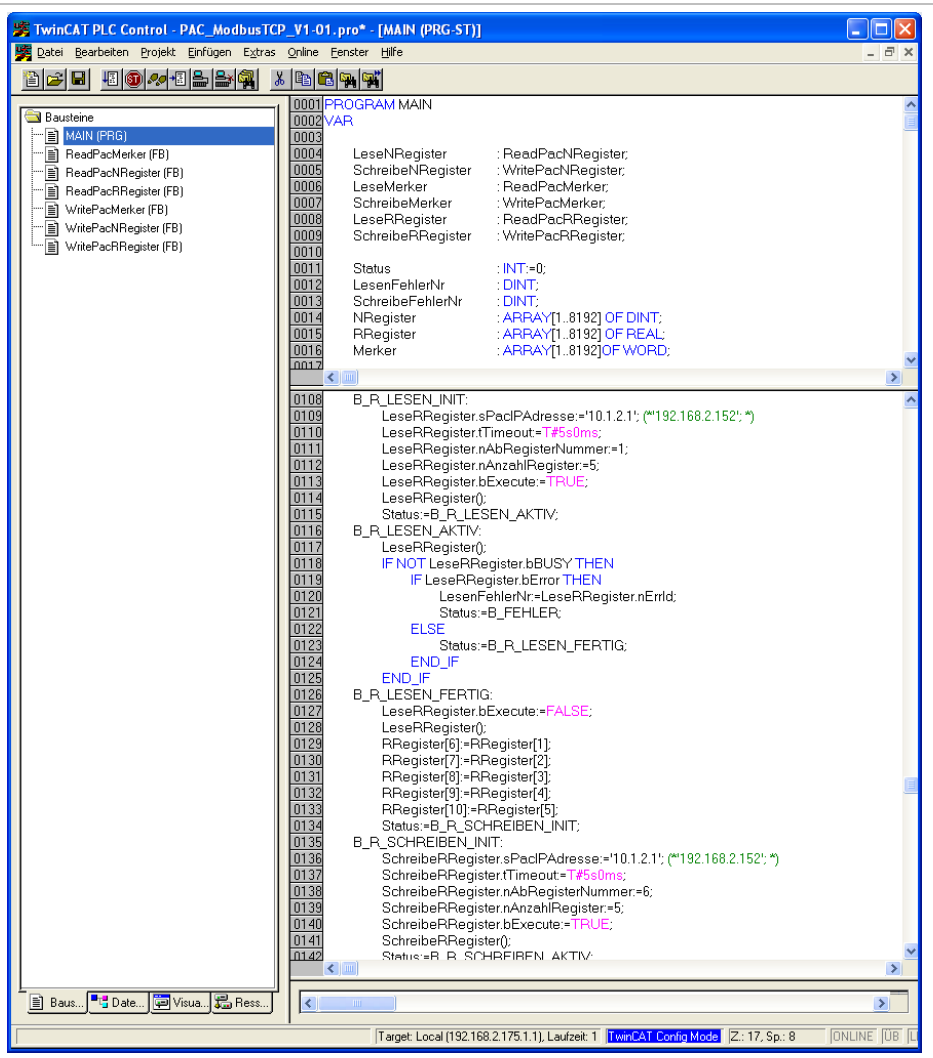
7.1.2 Funktionsbaustein WritePacNRegister

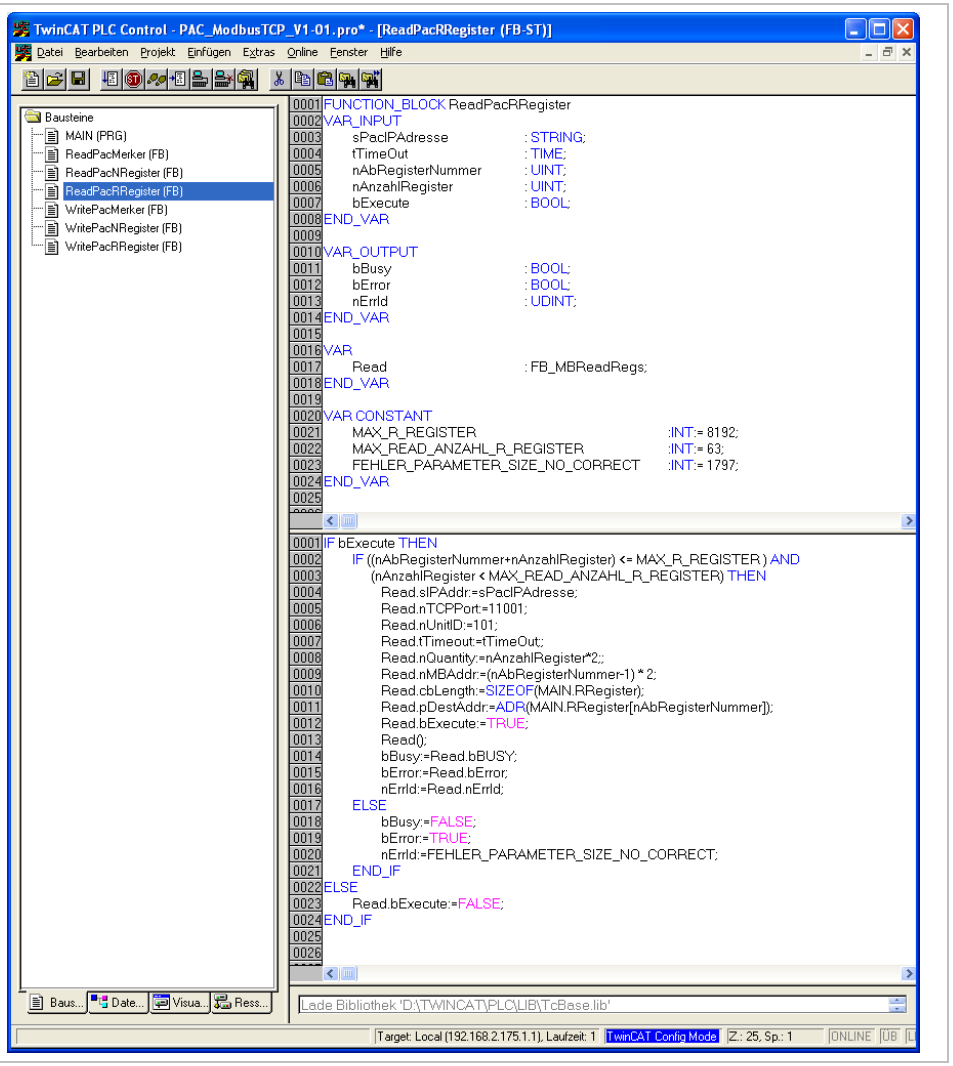




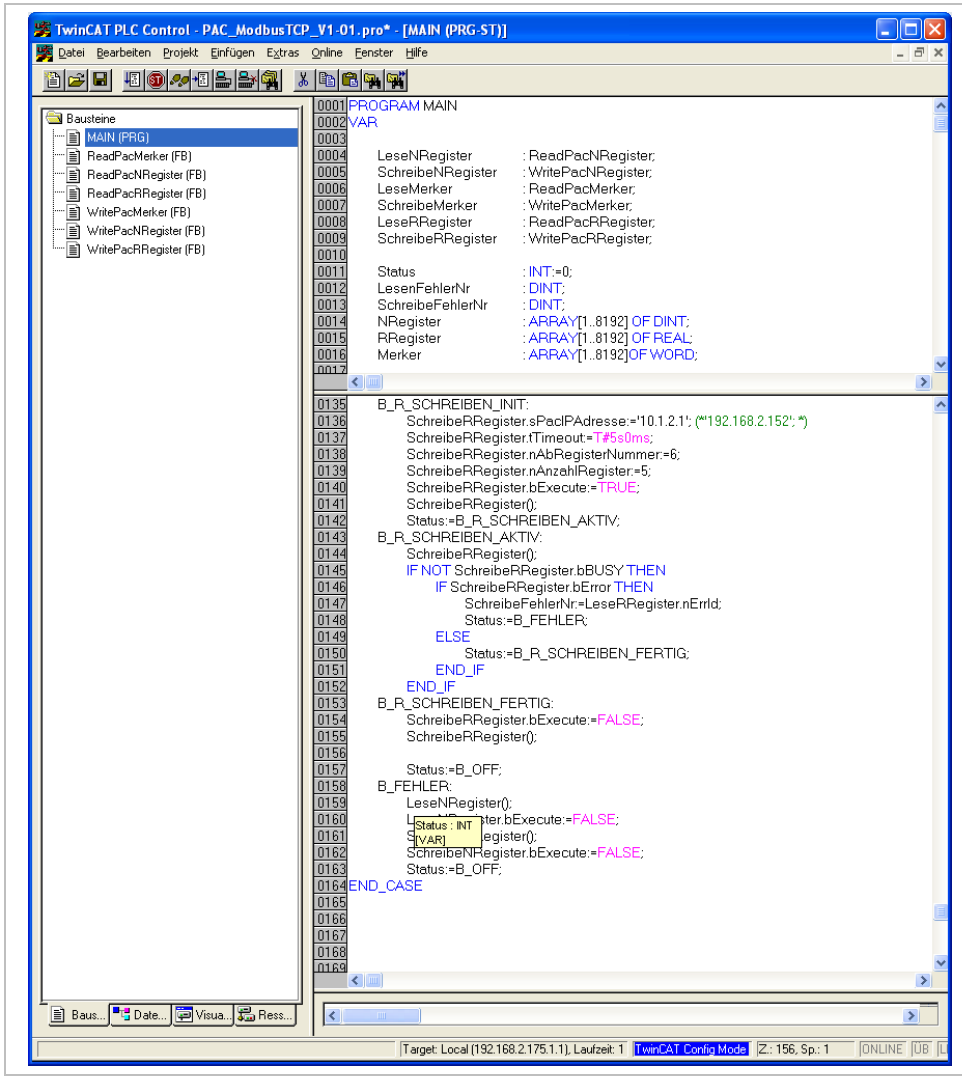
Erstellt von: Grieshaber Edmund | APP5030_DE_1120445_PA-CONTROL_ModbusTCP_Kommunikation_R1d.doc

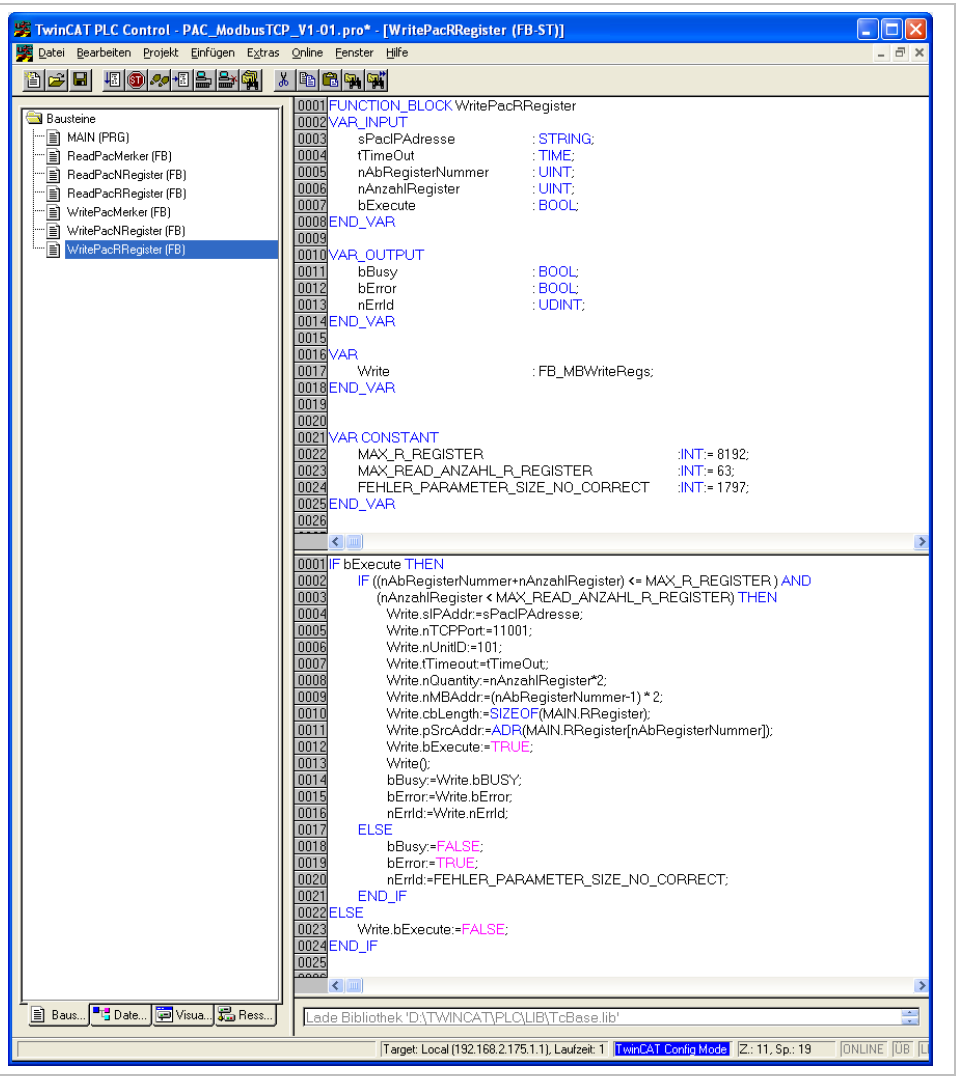
7.1.3 Funktionsbaustein ReadPacRRegister



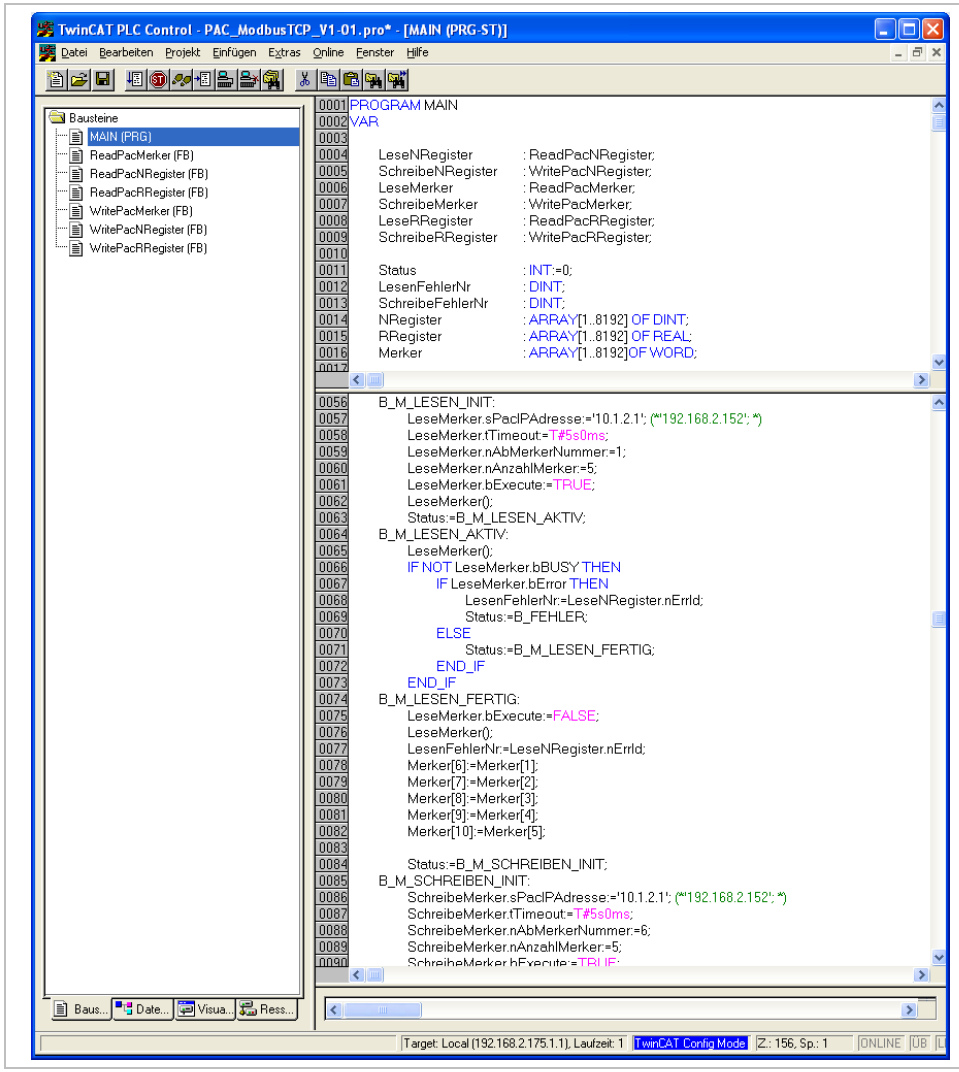


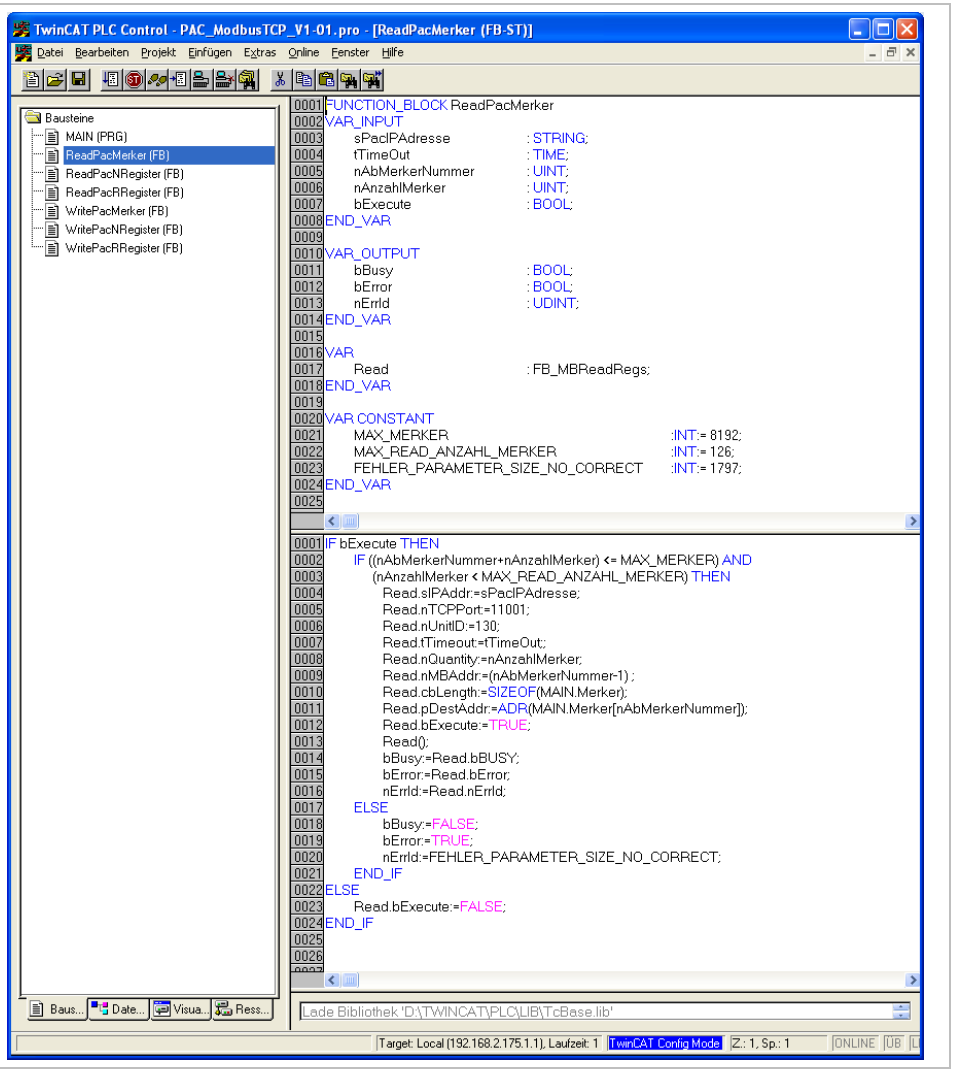
7.1.4 Funktionsbaustein WritePacRRegister



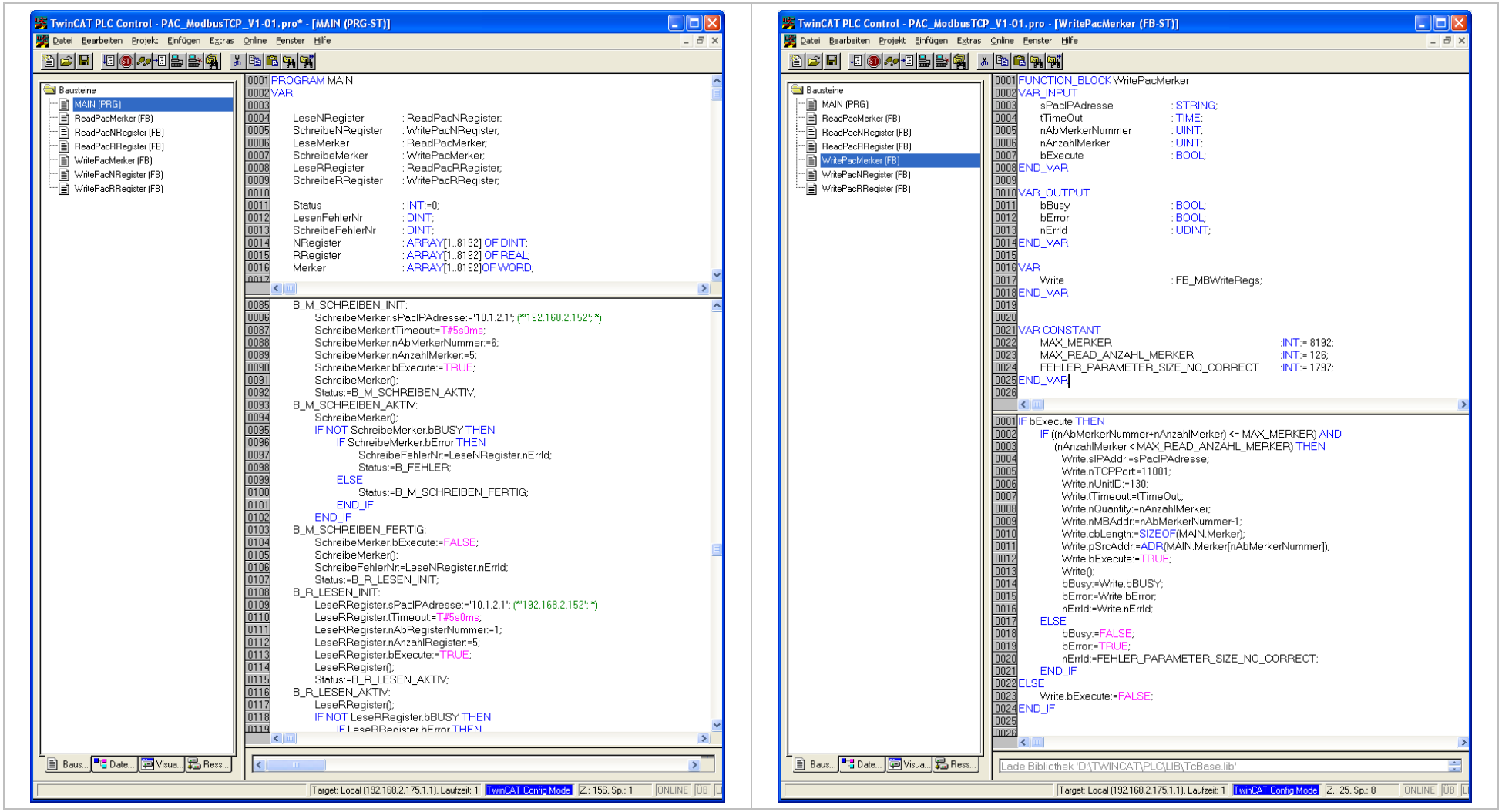


7.1.5 Funktionsbaustein ReadPacMerker

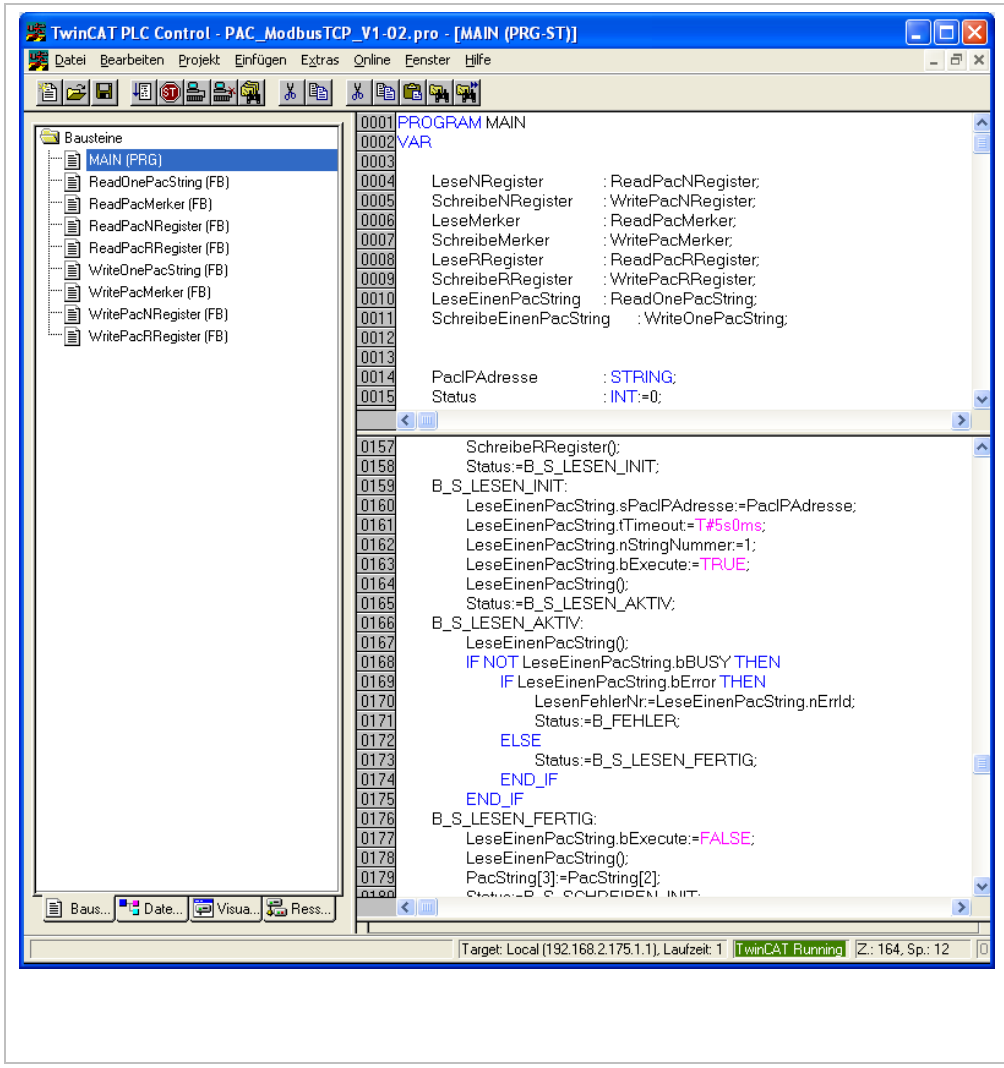


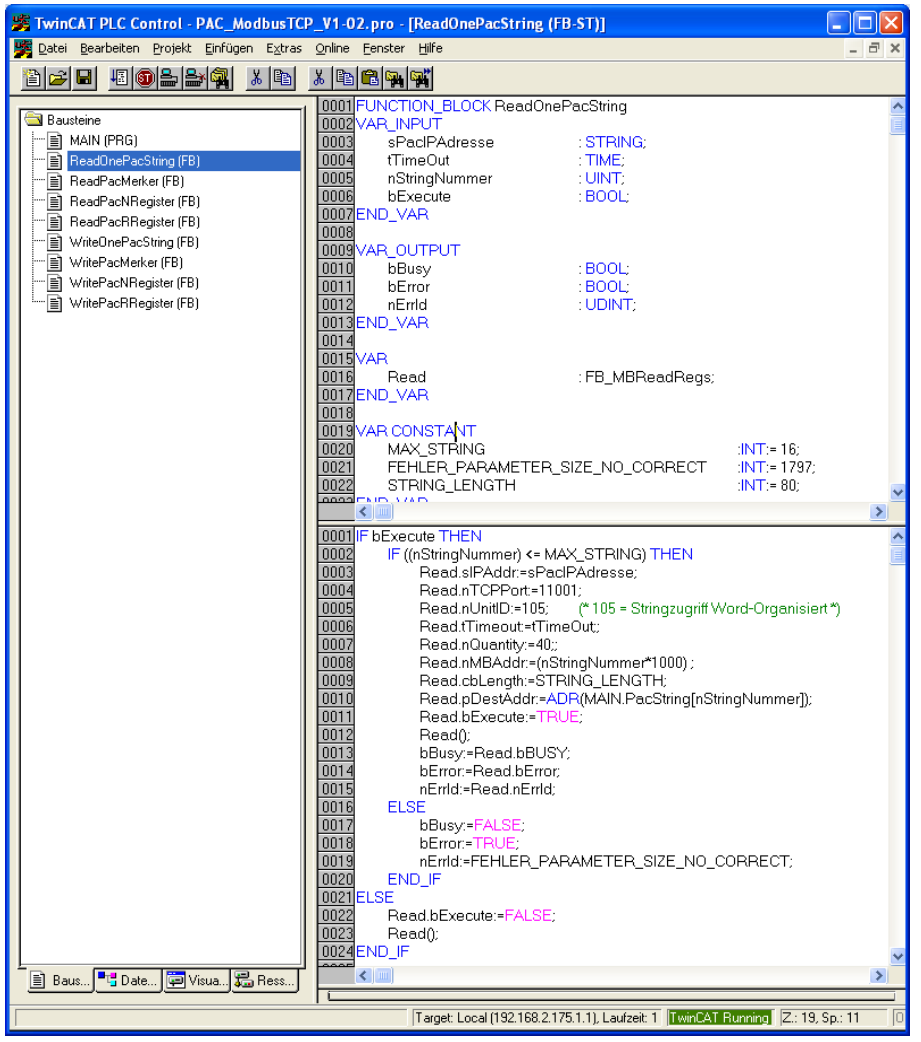


7.1.6 Funktionsbaustein WritePacMerker



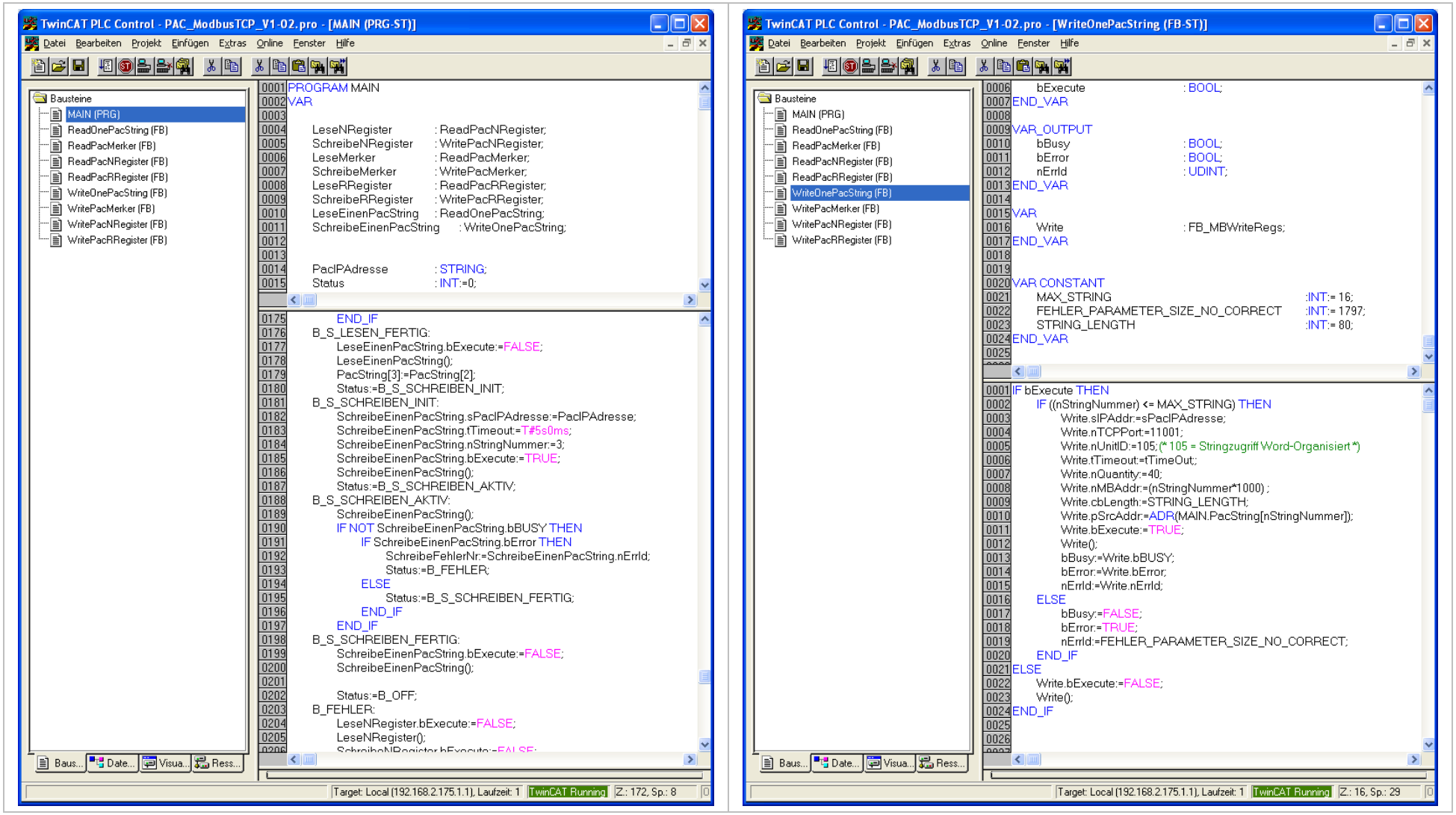
7.1.7 Funktionsbaustein ReadOnePacString





Erstellt von: Grieshaber Edmund | APP5030_DE_1120445_PA-CONTROL_ModbusTCP_Kommunikation_R1d.doc

7.1.8 Funktionsbaustein WriteOnePacString



Erstellt von: Grieshaber Edmund | APP5030_DE_1120445_PA-CONTROL_ModbusTCP_Kommunikation_R1d.doc

8 Beispiel von PA-CONTROL zu PA-CONTROL

Die PA-CONTROL Touch hat Befehle (G300-Gruppe), um über Ethernet mit dem ModbusTCP-Server einer anderen PA-CONTROL zu kommunizieren.

Befehl allgemein:

- G3xx.<IP-HH>.<IP-HL>.<IP-LH>.<IP-LL>.<Ab-Quelle-Element>.<Anzahl>.<Ab-Ziel-Element>.<Fehler-Register>.<Sprungziel bei Fehler>

Befehl	Beschreibung / Funktion	Beispiel
G301	Überträgt (kopiert) Zustände von Merker in die adressierte PA-CONTROL	G301.10.1.2.3.M1.16.M257.N111.Fehler
G311	Holt Zustände von Merker aus der adressierten PA-CONTROL	G311.10.1.2.3.M200.1.M200.N111.Fehler
G302	Überträgt (kopiert) Zustände von N-Registern in die adressierte PA-CONTROL	G302.10.1.2.3.N10.2.N20.N111.Fehler
G312	Holt Zustände von N-Registern aus der adressierten PA-CONTROL	G312.10.1.2.3.N100.5.N100.N111.Fehler
G303	Überträgt (kopiert) Zustände von R-Registern in die adressierte PA-CONTROL	G303.10.1.2.3.R10.2.R20.N111.Fehler
G313	Holt Zustände von R-Registern aus der adressierten PA-CONTROL	G313.10.1.2.3.R100.5.R100.N111.Fehler
G304	Überträgt (kopiert) den Inhalt von einem PAC-String in die adressierte PA-CONTROL	G302.10.1.2.3.S2.1.S3.N111.Fehler
G312	Holt den Inhalt eines PAC-String aus der adressierten PA-CONTROL	G312.10.1.2.3.S5.1.S6.N111.Fehler

9 Beispiel PA-CONTROL zu einem MODBUS-Server (Fa. Hilscher)

Die PA-CONTROL Touch hat Befehle (G300-Gruppe), um über Ethernet mit einem ModbusTCP-Server zu kommunizieren.

Befehl allgemein beim Schreiben in den MODBUS-Teilnehmer

- G3xx.<IP-HH>.<IP-HL>.<IP-LH>.<IP-LL>. <Port-Nummer>.<Unitidentifier>.<Ab dem PA-CONTOL-Element>.<Anzahl der PA-CONTROL-Elemente>.<Ablage ab der Start-Adresse in dem MODBUS-Teilnehmer>.<Fehler-N-Register>.<Sprungziel bei Fehler>

Befehl allgemein beim Lesen aus dem MODBUS-Teilnehmer

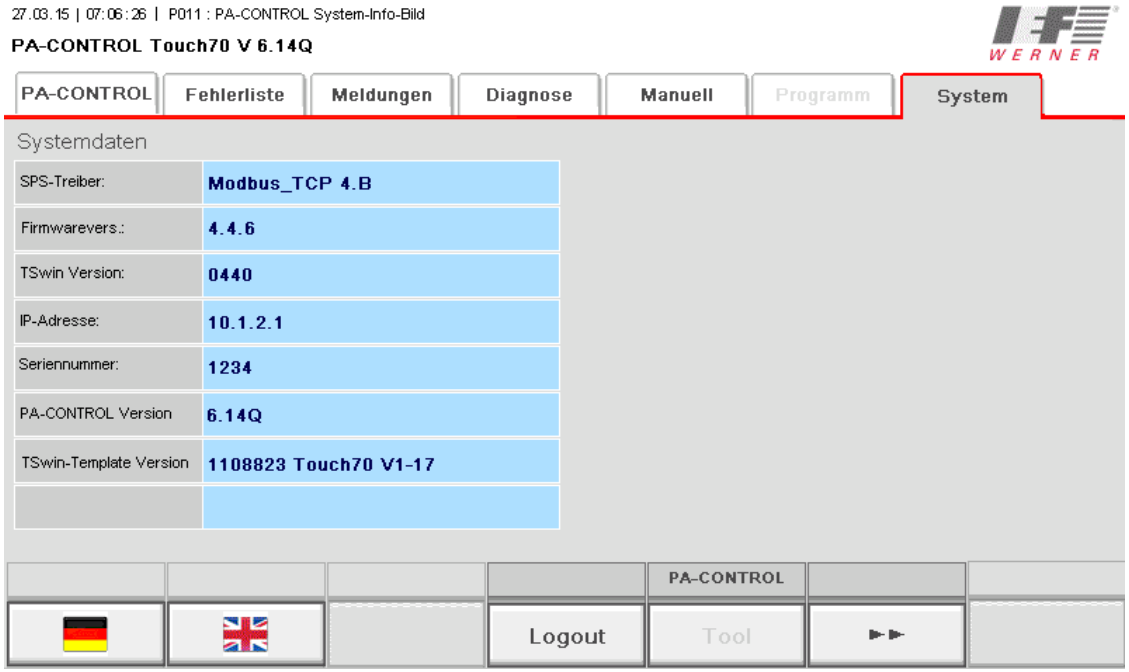
- G3xx.<IP-HH>.<IP-HL>.<IP-LH>.<IP-LL>. <Port-Nummer>.<Unitidentifier>.<Start-Adresse in dem MODBUS-Teilnehmer>.<Anzahl der Register in dem MODBUS-Teilnehmer>.<Ablage ab dem PA-CONTOL-Element>.<Fehler-N-Register>.<Sprungziel bei Fehler>

HINWEIS:

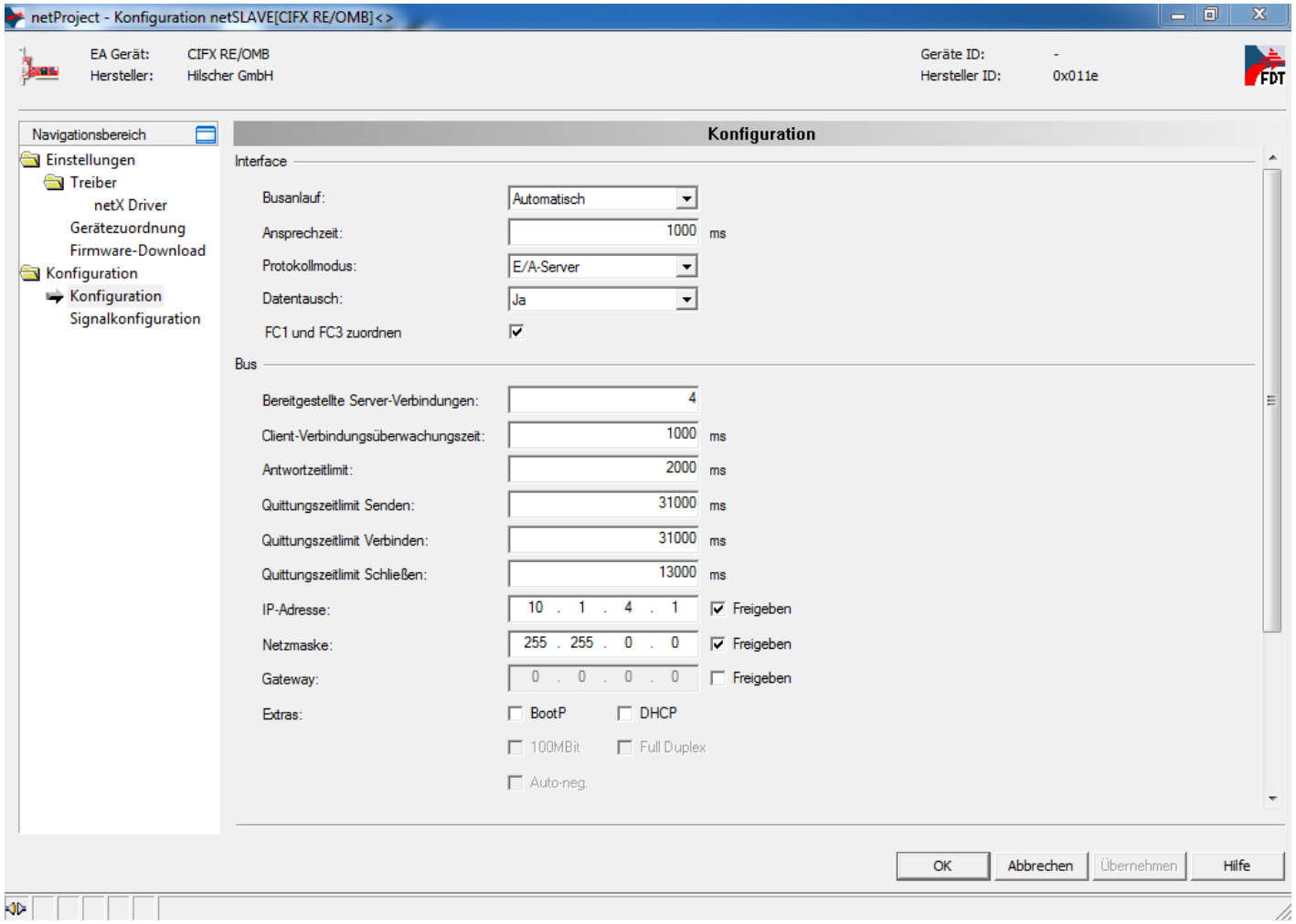
- In den MODBUS-Teilnehmern werden die Daten von Start-Adresse 0 bis Start-Adresse 65535 angesprochen.
- Die Unitidentifier kennzeichnet einen „Remote-Slave“, der über z-B. RS232 angeschlossen ist. Wenn er nicht verwendet wird, sollte er auf „0“ gestellt werden.
- Die Angabe für die Anzahl orientiert sich an der Quelle, von der Kopiert wird.
  Werden die Daten aus der PA-CONTROL geholt und in den MODBUS-Teilnehmer übertragen, wird die Anzahl durch die PA-CONTROL-Element bestimmt.
  Werden die Daten aus dem MODBUS-Teilnehmer geholt und in der PA-CONTROL abgelegt, so wird die Anzahl durch die MODBUS-Register bestimmt. (siehe Befehlsbeschreibung im Programmierhandbuch der PA-CONTROL).

9.1 Konfiguration der Teilnehmer

9.1.1 IP-Adresse der PA-CONTROL



9.1.2 Einstellung der Hilscher-Karte



Erstellt von: Grieshaber Edmund | APP5030_DE_1120445_PA-CONTROL_ModbusTCP_Kommunikation_R1d.doc

9.2 Befehlsübersicht

Befehl	Beschreibung
G321	Schreibt einen oder mehrere Merker aus der PA-CONTROL in den adressierten MODBUS-TCP-Teilnehmer
G322	Schreibt einen oder mehrere N-Register aus der PA-CONTROL in den adressierten MODBUS-TCP-Teilnehmer
G323	Schreibt einen oder mehrere R-Register aus der PA-CONTROL in den adressierten MODBUS-TCP-Teilnehmer
G324	Schreibt einen PA-CONTROL-String aus der PA-CONTROL in den adressierten MODBUS-TCP-Teilnehmer
G331	Liest einen oder mehrere Holding-Register aus dem adressierten MODBUS -TCP-Teilnehmer und speichert in den Merker
G332	Liest einen oder mehrere Holding-Register aus dem adressierten MODBUS -TCP-Teilnehmer und speichert ins N-Register
G333	Liest einen oder mehrere Holding-Register aus dem adressierten MODBUS -TCP-Teilnehmer und speichert ins R-Register
G334	Liest mehrere Holding-Register aus dem adressierten MODBUS -TCP-Teilnehmer und speichert in einen PA-CONTROL-String
G390	Beendet Socketverbindung, schließt Socketverbindung

9.2.1 Befehlsbeispiele Merker

Obwohl Merker nur den Zustand „0“ oder „1“ haben können, werden sie bei der Übertragung wie 16-Bit-Werte betrachtet. Es werden die MODBUS-Funktionen „Read-Holding-Register“ und „Write-Multiple-Register“ verwendet.

- G321.10.1.2.12.502.0.M5.3.0.N900.FEHLER
  schreibt die Merker M5, M6 und M7 in die Register 0 bis 2 (Start-Adresse=0) in den adressierten MODBUS-TCP-Teilnehmer (10.1.2.12)
- G321.192.168.2.152.502.0.M1.2.5.N900.FEHLER
  schreibt die Merker M1und M2 in die Register 5 und 6 (Start-Adresse=5) in den adressierten MODBUS-TCP-Teilnehmer (192.168.2.151)
- G331.10.1.2.12.502.0.0.3.M5.N900.FEHLER
  liest die Holding-Register 1, 2, und 3 des MODBUS-Teilnehmers (10.1.2.12) und schreibt sie in die PA-CONTROL in die Merker M5, M6 und M7

9.2.2 Befehlsbeispiele N-Register

N-Register werden in der PA-CONTROL als 32-Bit-Wert (31Bit, plus Vorzeichen) behandelt. Bei der Übertragung müssen also 2 Holding-Register verwendet werden.

- G322.10.1.2.12.502.0.N1.4.0.N900.FEHLER

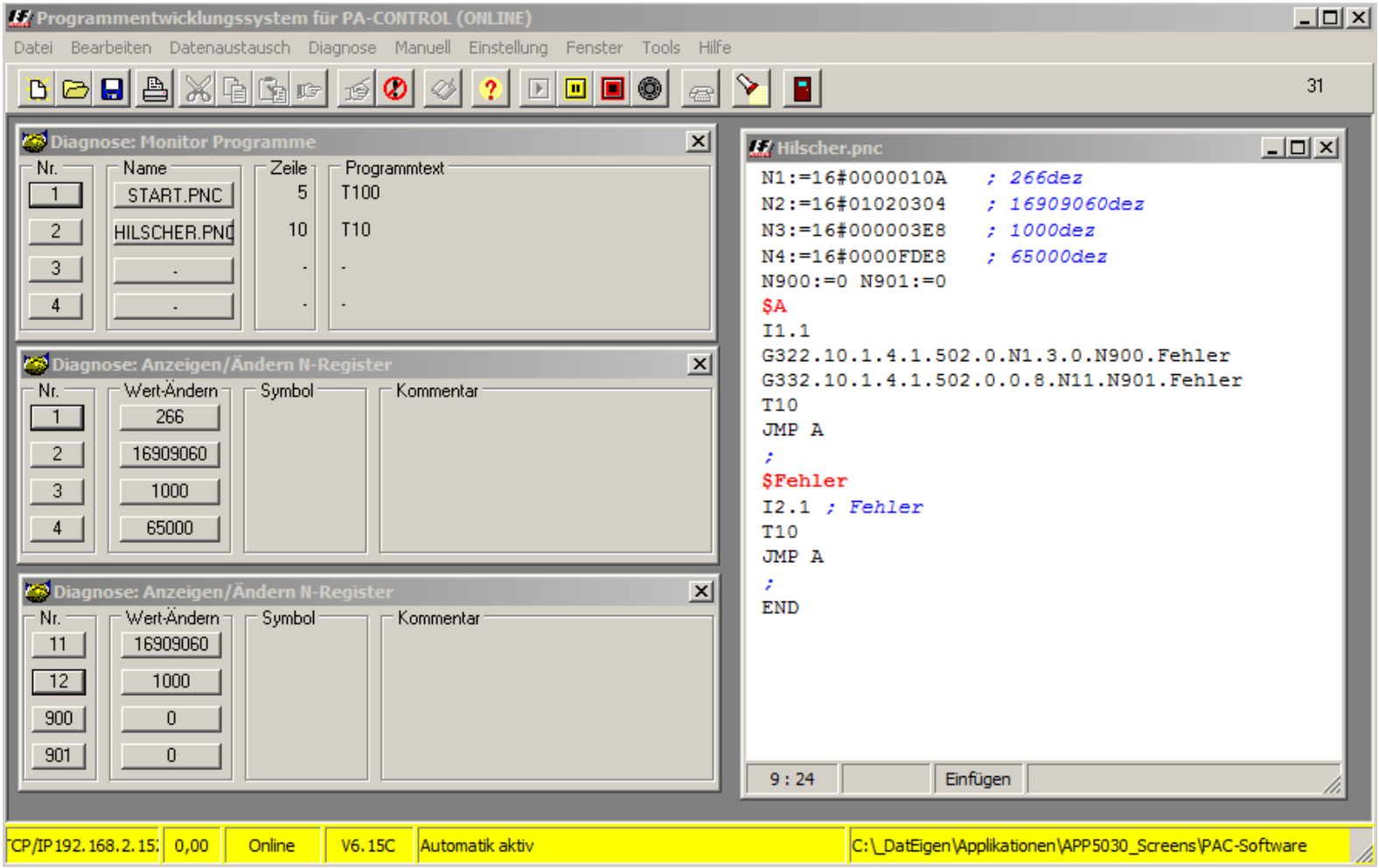
  schreibt den Inhalt der N-Register N1, N2, N3 und N4 in den MODBUS-Teilnehmer ab der Start-Adresse 10.
  Es werden 4 x 32Bit, also 8 Register(16-Bit) beschrieben (Holding Register[1 ...8]).
- G322.10.1.2.12.502.0.N100.10.2.N900.FEHLER

  schreibt den Inhalt der N-Register N100 bis N110 in den MODBUS-Teilnehmer ab der Start-Adresse 2.
  Es werden 10 x 32Bit, also 20 Register(16-Bit) beschrieben (Holding Register[3 ...22]).
- G331.10.1.2.12.502.0.0.8.N1.N900.FEHLER

  liest die Holding-Register 1 bis 8 des MODBUS-Teilnehmers und schreibt sie in die PA-CONTROL in die N-Register N1 bis N4

9.3 Kommunikationsbeispiel

9.3.1 Programm in der PA-CONTROL



9.3.2 Hilscher-Karte Diagnose E/A-Monitor

9.3.2.1 G322-Befehl

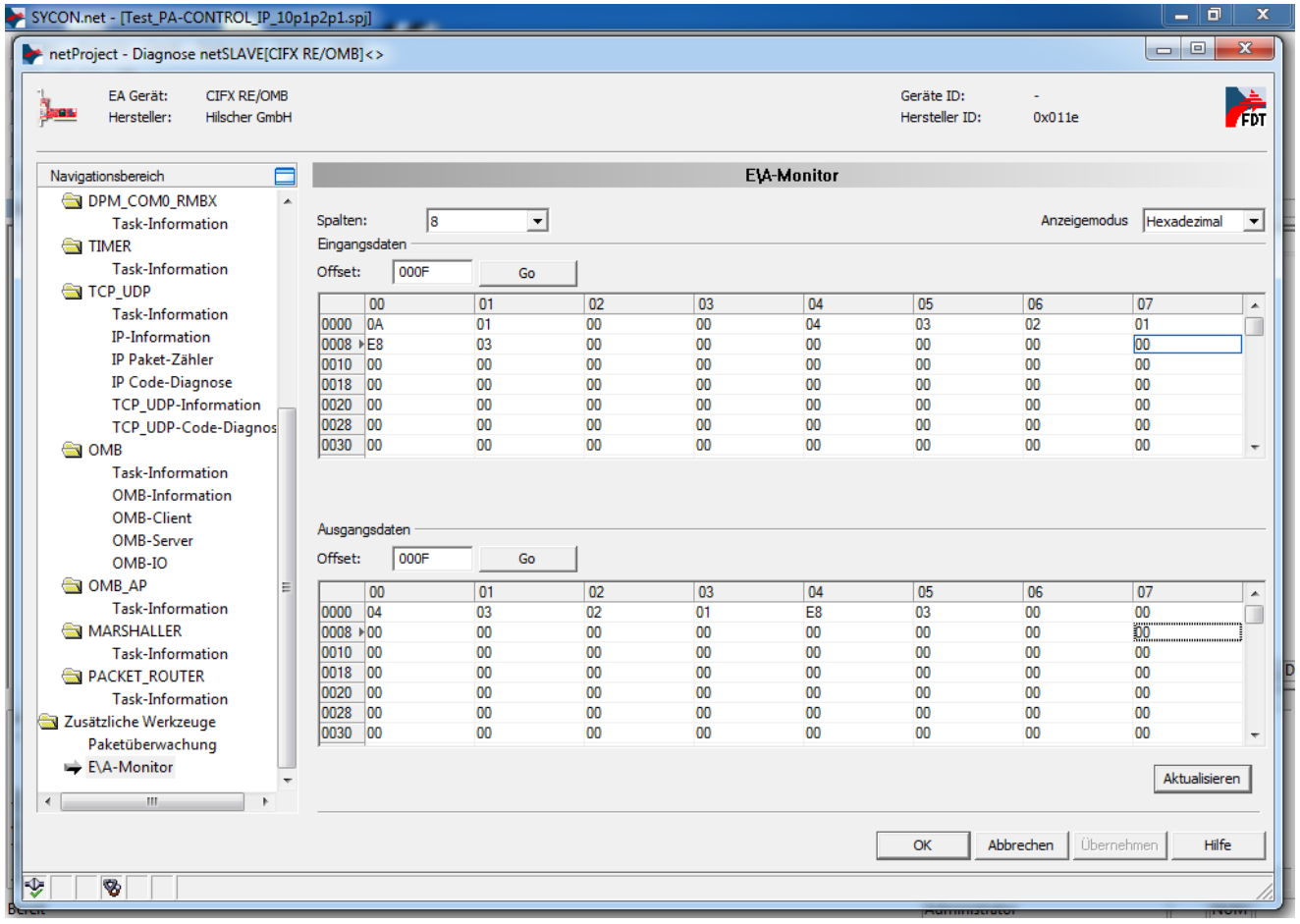
Der G322-Befehl schreibt 3 N-Register (N1, N2, N3) ab der Offset-Adresse 0 in die Eingangsdaten der Hilscherkarte.
Dabei werden pro N-Register 4 Bytes übertragen

N1:=16#0000010A / 266dez
→ Zeile 0000, Spalte 00 (lowest Byte), 01, 02, 03 (highest Byte)

N2:=16#01020304 / 16909060dez
→ Zeile 0000, Spalte 04(lowest Byte), 05, 06, 07 (highest Byte)

N3:=16#000003E8 / 1000dez
→ Zeile 0008, Spalte 00 (lowest Byte), 01, 02, 03 (highest Byte)

N4:=16#0000FDE8 / 65000dez
→ Zeile 0008, Spalte 04 (lowest Byte), 05, 06, 07 (highest Byte)



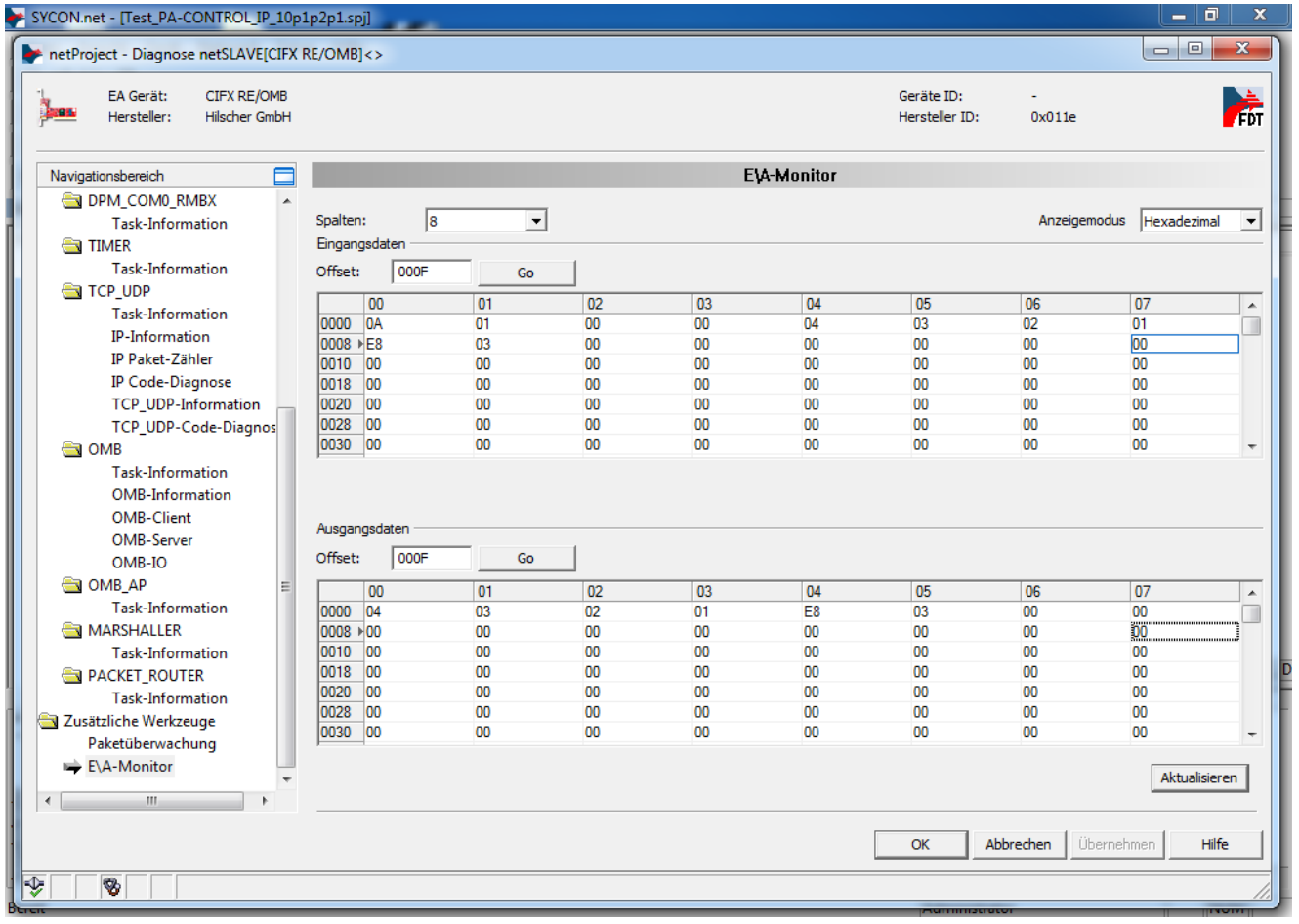
9.3.2.2 G332-Befehl

Der G322-Befehl holt 8 Datenworte aus den Ausgangsdaten ab der Offset-Adresse 0 aus der Hilscherkarte und legt dies in N-Register ab.

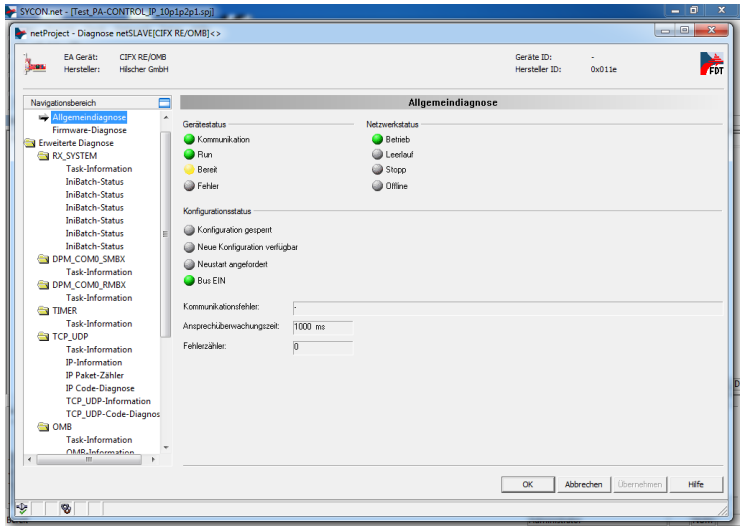
Dabei werden in der PA-CONTROL 2 N-Register (N11, N12) beschrieben.

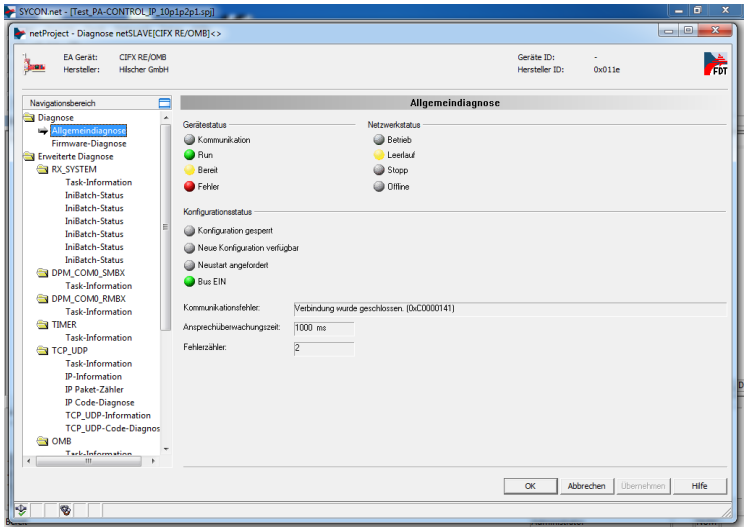
N11:=16#01020304 / 16909060dez
→ Zeile 0000, Spalte 00(lowest Byte), 01, 02, 03 (highest Byte)

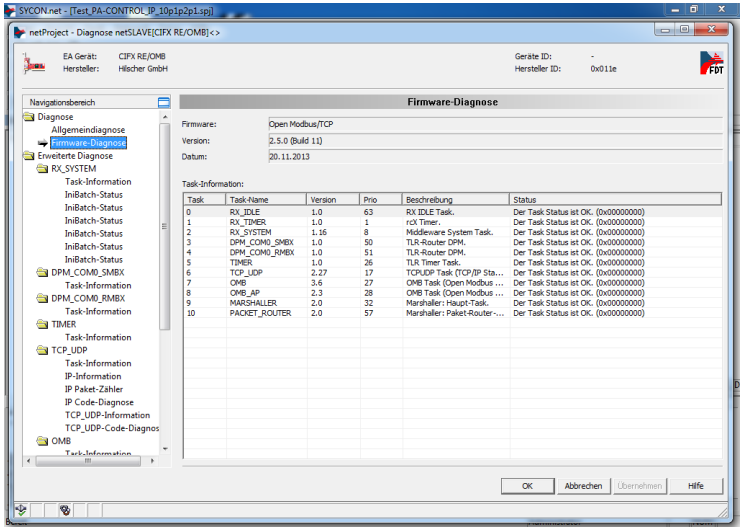
N12:=16#000003E8 / 1000dez
→ Zeile 0000, Spalte 04 (lowest Byte), 05, 06, 07 (highest Byte)

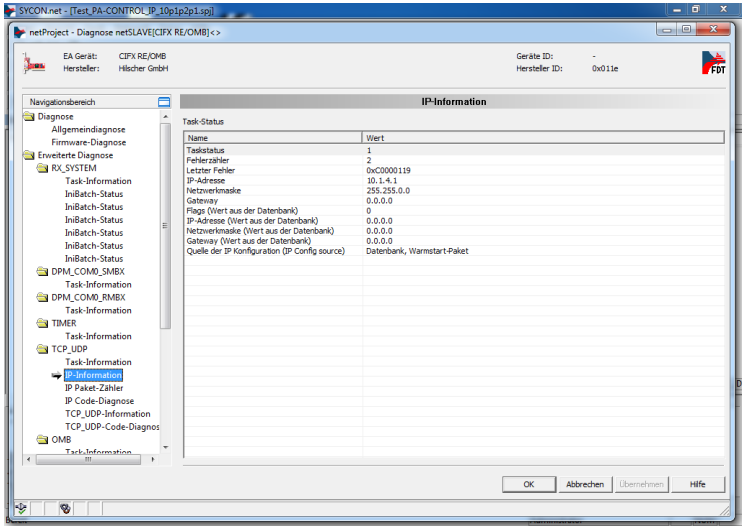


9.3.3 Hilscher Karte Diagnose









Erstellt von: Grieshaber Edmund | APP5030_DE_1120445_PA-CONTROL_ModbusTCP_Kommunikation_R1d.doc

10 Slave Betrieb der PA-CONTROL

In der Betriebsart SLAVE können die Achsen, digitale Ein- und Ausgänge sowie analoge Ein- und Ausgänge über verschiedene Befehle über die Ethernet-Schnittstelle verfahren, gesteuert und abgefragt werden. Es werden keine PA-CONTROL-Programme benötigt.

Die Steuerung der Achsen, Ein- und Ausgängen erfolgt über die Datenfelder des ModbusTCP-Interfaces.

10.1 Slave Betrieb Interface

Unit Identifier	Element	Schreiben	Lesen
1 ... 16	Achsen	Die "SLAVE Kommando Achsdaten" schreiben: - Achse ON - Startbefehl - Mache Referenzfahrt - Fahrbefehl absolut auf Position - ... - Position - Geschwindigkeit - ...	Den Status der Achse lesen: - Achsentype - Achse bereit - Status positiver Endschalter - Status negativer Endschalter - ... - aktuelle Position der Achse - ...
17 ... 32	IO-Knoten → Eingänge → Ausgänge	Alle digitalen und analogen Ausgänge dieses CAN-IO-Knotens setzen oder rücksetzen → bis zu 256 digitale Ausgänge → bis zu 4 analoge Ausgänge (8 Bit, ..., 16-Bit)	Der Zustand aller digitalen und analogen Eingänge dieses CAN-IO-Knotens lesen → bis zu 256 digitale Eingänge → bis zu 4 analoge Eingänge (8Bit, ..., 16 Bit)

10.2 Slave Kommando Achsdaten

Adresse	Funktion	Bemerkung	Format / Type
0	Kommando		16 Bit Integer
1	Nummer des Einganges für die Kommandos ■ Fahre so lange Eingang auf 1 ■ Fahre so lange Eingang auf 0 ■ ...		16 Bit Integer
2	Position [AE]		32 Bit Kommazahl
4	Verfahrgeschwindigkeit [AE/s]		32 Bit Kommazahl
6	Beschleunigung [AE/s^2]		32 Bit Kommazahl

10.2.1 Kommandos

Nummer (Code)	Funktion	Bemerkung
0	Keine Aktion	
1	Rücksetzen Fehler der Achse	
2	Kommando abbrechen	
3	Schalte Achse ON	
4	Schalte Achse OFF	
5	Mache Referenzfahrt	
6	Fahre Achse auf absolute Position	
7	Fahre Achse um relative Position	
8	Fahre Achse auf absolute Position, solange der Eingang "1" bestromt ist	
9	Fahre Achse auf absolute Position, solange der Eingang "0" nicht bestromt ist	
10	Fahre bis zur Verfahrbereichsgrenze minimum	
11	Fahre bis zur Verfahrbereichsgrenze maximum	

10.2.2 Statusinformation

Nummer (Code)	Statusinformation	Bemerkung
0	Keine Aktion	
10	Unbekannt	
20	Kommando nicht möglich	
30	Kommando erkannt	
40	Kommando wird bearbeitet	
50	Kommando ist fertig und fehlerfrei ausgeführt	
60	Kommando wurde abgebrochen und die Bearbeitung ist beendet.	
70	Kommando ist fertig und fehlerfrei ausgeführt, der Fahrauftrag wurde über den definierten Eingang beendet.	
80	Fehler, ein Parameter (Position, Geschwindigkeit, ...) war außerhalb des Bereiches, die Bearbeitung ist beendet.	Geschwindigkeit zu groß, ...
90	Fehler, Fehler von der Hardware, während der Bearbeitung des Kommandos.	Endschalter, Schleppfehler, ...

10.3 Slave IO-Knoten Daten

10.3.1 Slave IO-Knoten Daten schreiben

Beim Schreiben können alle Ausgangsdaten eines CAN-IO-Knotens in einem Schreibvorgang übertragen werden.

Adresse	Funktion	CAN-ID 17	CAN-ID 18	CAN-ID 19	CAN-ID 20	Format / Type
0	Digitale Ausgänge, Wort 0	O1 ... O16	O257 ... O572	O513 ...O528	O769 ...O784	16 Bit Integer
1	Digitale Ausgänge, Wort 1	O17 ... O32	O273 ...	O529 ...	O785	16 Bit Integer
2	Digitale Ausgänge, Wort 2	O33 ... O48				16 Bit Integer
3	Digitale Ausgänge, Wort 3	O49 ... O64				16 Bit Integer
4	Digitale Ausgänge, Wort 4	O65 ... O80				16 Bit Integer
5	Digitale Ausgänge, Wort 5	O81 ... O96				16 Bit Integer
...	...	...				
14	Digitale Ausgänge, Wort 14	O225 ... O239				16 Bit Integer
15	Digitale Ausgänge, Wort 15,	O240 ...O256	... O512	... O768	... O1024	16 Bit Integer
16	Analoger Ausgang 1	DA1	DA5	DA9	DA13	16 Bit Integer
17	Analog Ausgang 2	DA2	DA6	DA10	DA14	16 Bit Integer
18	Analog Ausgang 3	DA3	DA7	DA11	DA15	16 Bit Integer
19	Analog Ausgang 4	DA4	DA8	DA12	DA16	16 Bit Integer

Erstellt von: Grieshaber Edmund | APP5030_DE_1120445_PA-CONTROL_ModbusTCP_Kommunikation_R1d.doc

10.3.2 Slave IO-Knoten Daten lesen

Beim Lesen können alle Eingangsdaten eines CAN-IO-Knotens gelesen werden.

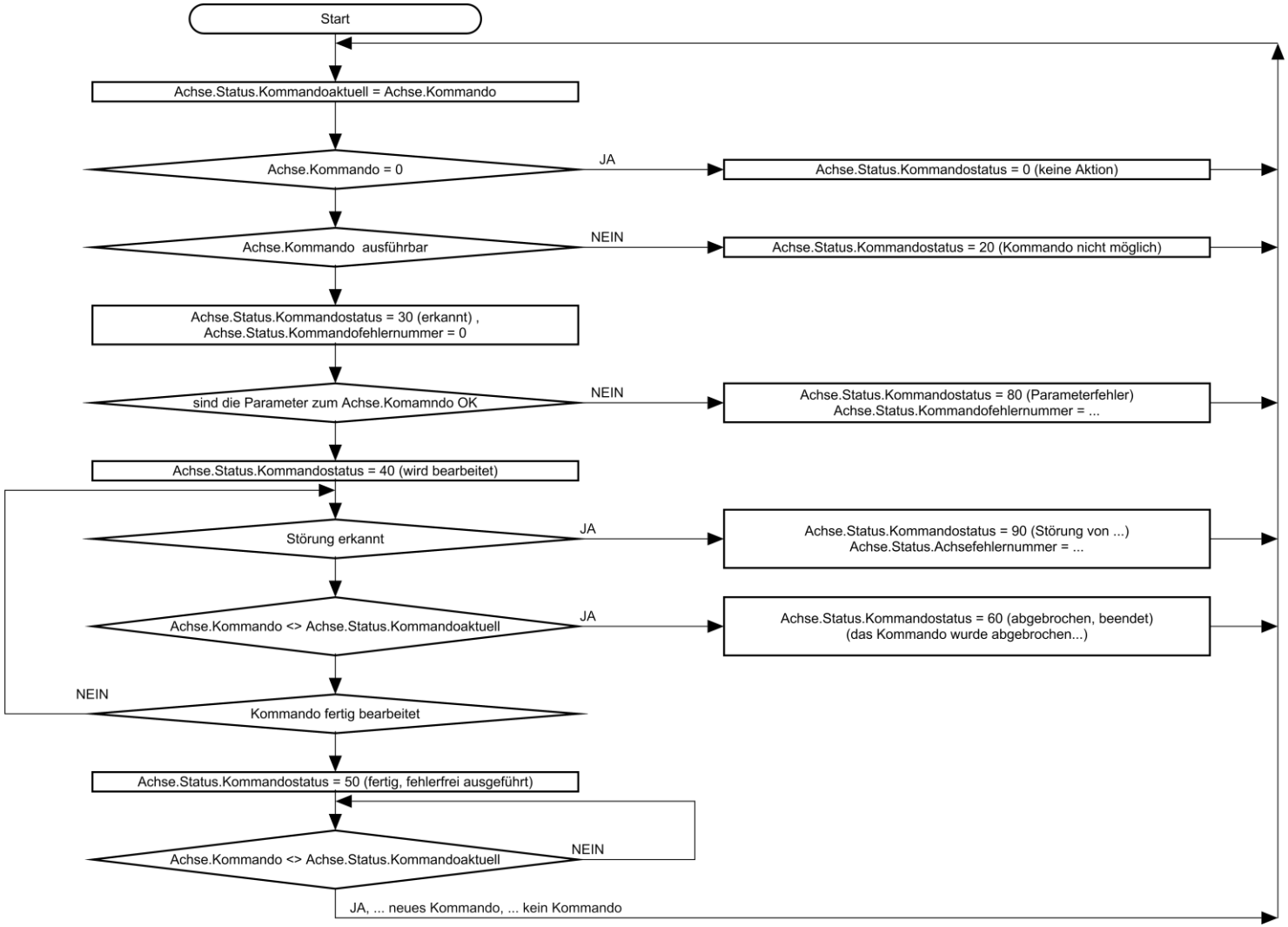
Adresse	Funktion	CAN-ID 17	CAN-ID 18	CAN-ID 19	CAN-ID 20	Format / Type
0	Digitale Eingänge, Wort 0	I1 ... I16	I257 ... I572	I513 ...I528	I769 ...I784	16 Bit Integer
1	Digitale Eingänge, Wort 1	I17 ... I32	I273 ...	I529 ...	I785	16 Bit Integer
2	Digitale Eingänge, Wort 2	I33 ... I48				16 Bit Integer
3	Digitale Eingänge, Wort 3	I49 ... I64				16 Bit Integer
4	Digitale Eingänge, Wort 4	I65 ... I80				16 Bit Integer
5	Digitale Eingänge, Wort 5	I81 ... I96				16 Bit Integer
...	...	...				
14	Digitale Eingänge, Wort 14	II225 ... I239				16 Bit Integer
15	Digitale Eingänge, Wort 15	I240 ...I256	... I512	... I768	... I1024	16 Bit Integer
16	Analog Eingang 1	AD1	AD5	AD9	AD13	16 Bit Integer
17	Analog Eingang 2	AD2	AD6	AD10	AD14	16 Bit Integer
18	Analog Eingang 3	AD3	AD7	AD11	AD15	16 Bit Integer
19	Analog Eingang 4	AD4	AD8	AD12	AD16	16 Bit Integer

10.4 Steuerungsprinzip für die Achsen über die Kommandos

Eingangsdaten	Achse	Ausgangsdaten
Kommandoparameter: ■ Position [AE] ■ Verfahrgeschwindigkeit [AE/s] ■ Beschleunigung [AE/s2]		
Achse.Kommando	ACHSE	Achse.Status.Kommandoaktuell
		Achse.Status.Kommandostatus
		Achse.Status.Kommandofehlernummer
		Achse.Status.Achsefehlernummer

Erstellt von: Grieshaber Edmund | APP5030_DE_1120445_PA-CONTROL_ModbusTCP_Kommunikation_R1d.doc

10.4.1 Ablaufdiagramm für ein Kommando für eine Achse



Erstellt von: Grieshaber Edmund | APP5030_DE_1120445_PA-CONTROL_ModbusTCP_Kommunikation_R1d.doc