Ethernet für die Saia PCD® Serie

Dokument 26-776 | Ausgabe GER13 | 2019-08-08

0

0 Inhaltsverzeichnis

0

0

0.1 Dokumentversionen

Version	Datum	Publiziert	Anmerkungen
DE03	-	-	Erste freigegebene Ausgabe
DE04	2008-01-15	2008-02-12	Überarbeitete Ausgabe
DE05	2010-01-15	2010-02-17	Vorbereitung zum Übersetzen
DE06	2011-07-27		Kapitel 2: Fullduplex und MDIX ergänzt
DE07	2012-04-04		Kapitel 4.3.9: ***„OpenUDP“*** ersetzt mit ***„OpenClientTCP“***
DE08	2013-11-05	2013-11-13	Neues Logo und neuer Firmenname
DE09	2014-02-18 2014-04-03	2014-04-03 2014-04-03	Kapitel 4 - Flussdiagramme geändert Kapitel 4 - Länge der gesendeten Nutzdaten UDP
GER10	2016-02-26	2016-02-26	Kapitel 4.2.1 - Fehler in Anwendungsbeispiel
GER11	2017-09-01	2017-09-01	alle ***„IPD“*** durch ***„IP“*** ersetzt
GER12	2018-06-05	2017-06-05	Kapitel 2.2.5 - Tabellen Kommentare geändert
GER13	2019-08-08	2019-08-08	Tippfehler korrigiert „CTP“ --> „TCP“

0.2 Handelsmarken und Warenzeichen

Saia PCD® und Saia PG5®
sind registrierte Warenzeichen der Saia-Burgess Controls AG.

Technische Veränderungen basieren auf dem aktuellen technischen Stand.

Publiziert in der Schweiz

1 Inbetriebnahme

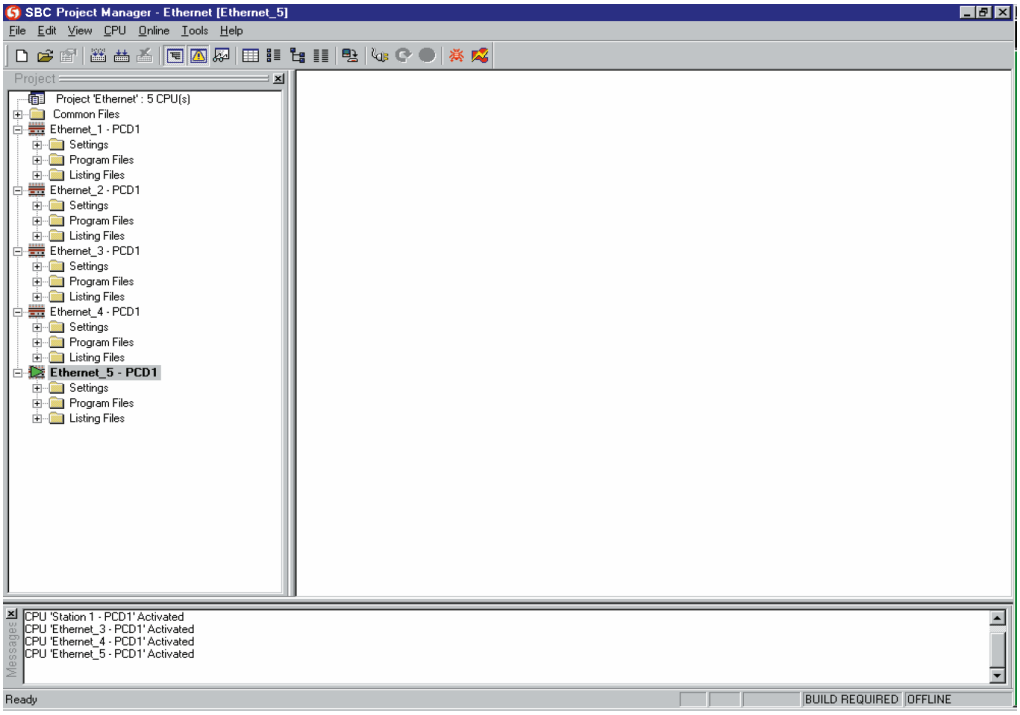
1.1 Wichtige Hinweise

- Pro Saia PCD®-System kann, ausser auf der PCD2.M480, nur ein Ethernet-Modul verwendet werden
- Pro System ist nur ein S-Bus PGU-Port erlaubt. Wenn das Ethernet-Modul als S-Bus PGU konfiguriert ist, kann kein anderer Port als S-Bus PGU-Port benutzt werden
- Das Ethernet-Modul kann auch für das reduzierte S-Bus-Protokoll konfiguriert werden (siehe die folgenden Richtlinien). In diesem Fall kann jeder andere Port als S-Bus PGU benutzt werden
- Eine Ethernet-Station benötigt eine Zuordnungstabelle mit der IP-Address und dem IP-Node für alle Stationen im Netz. Diese Zuordnungstabelle wird durch den Hardware-Konfigurator automatisch erzeugt und in einem DBX in die Saia PCD® geladen. Die Hardwarekonfiguration aller PCD-Systeme im Ethernet muss daher im Projektmanager im selben Projekt definiert werden
- Wenn mehrere Personen verschiedene Saia PCD®-Stationen im gleichen Pro jekt programmieren, wird empfohlen, dass die gesamte Netzkonfiguration zu erst auf einem Programmier-PC definiert wird. Diese Konfiguration sollte dann auf die anderen Programmier-PCs kopiert oder von den Programmier-PCs importiert werden
- Die Gateway-Kommunikation ist nur in Richtung Ethernet zu einem S-Bus-Sub-netz, jedoch nicht in umgekehrter Richtung möglich. Pro Gateway ist nur ein S-Bus-Subnetz erlaubt.

1.2 Konfiguration eines Ethernet-Ports der Saia PCD® mit Saia PG5®

Weitere Informationen sind in der Saia PG5® Hilfe zu finden.

1.2.1 Ein neues Projekt erzeugen, das alle CPUs im Ethernet umfasst

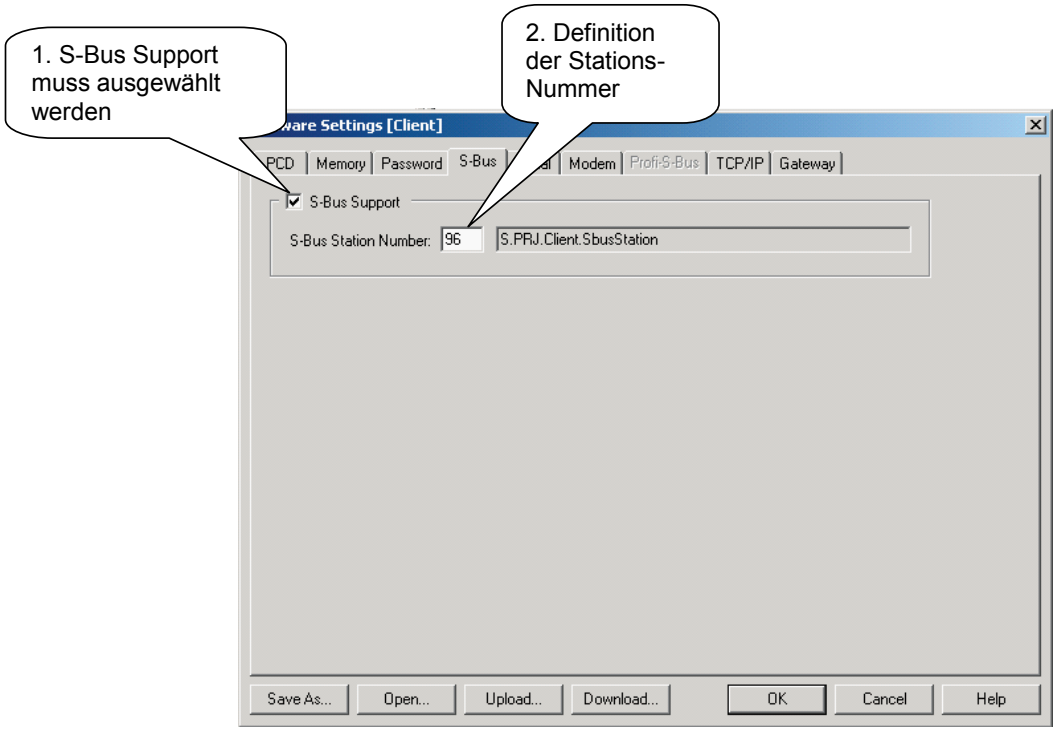


Die Hardware-Einstellungen müssen für alle Saia PCD®-Stationen eingegeben werden.

1.2.2 Hardware-Einstellungen vornehmen

Für die Konfiguration des Ethernets sind die folgenden Fenster relevant:

- **SBC S-Bus Einstellungen**

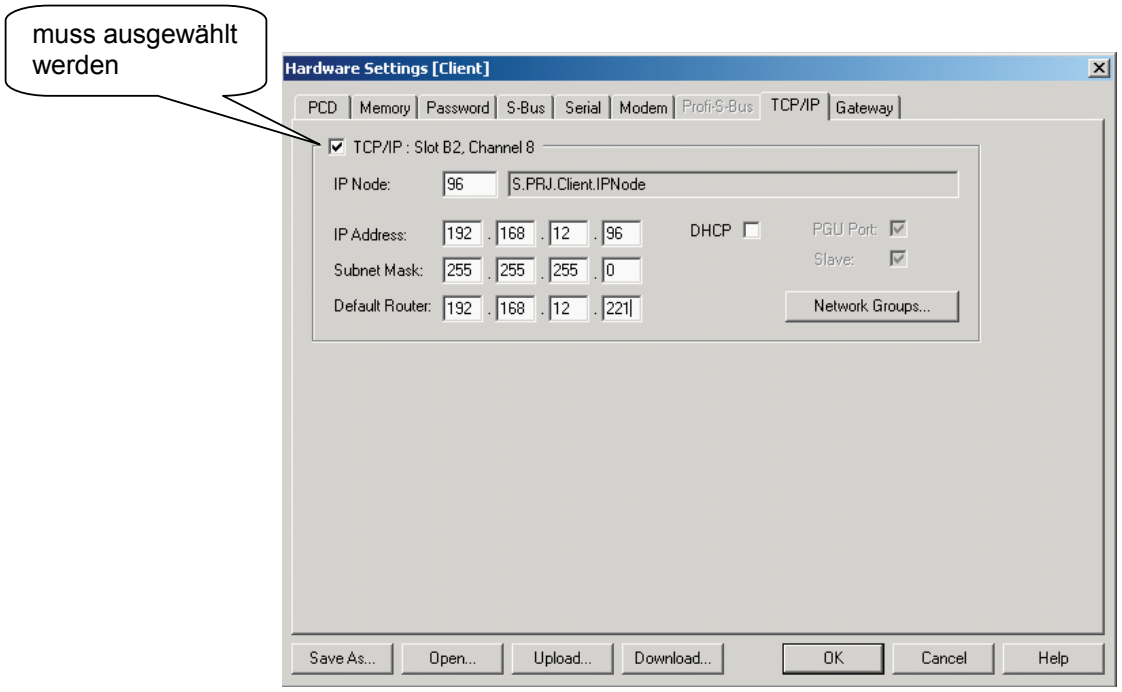


- **Gateway-Einstellung**

Die Definition ist nur für eine Gateway-Station notwendig.

Die Einstellungen sind dieselben wie für die Standard-S-Bus-Kommunikation.

- **Einstellungen des Ethernet TCP/IP-Moduls**



Eingabe der Werte für IP Node , IP Address , Subnet Mask und Default Router.

In den Fällen mit nur einem Subnetz ist die Eingabe für den Default Router nicht zwingend notwendig.

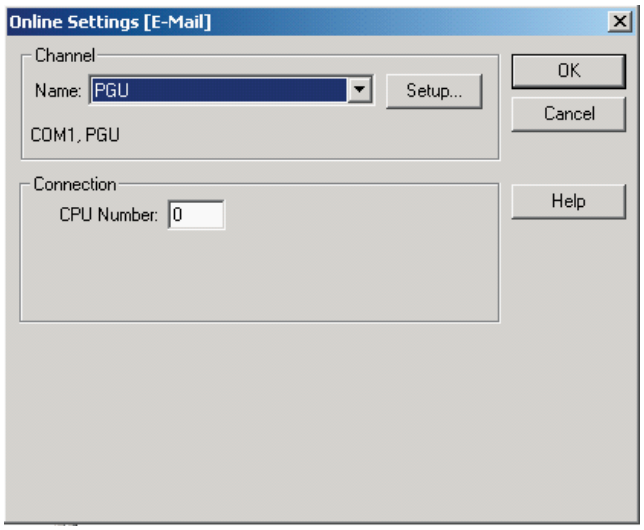
Die Einstellungen Channel and Slot (Kanal und Steckplatz) sind abhängig davon, wo das Ethernet-Modul installiert wurde.

Für einfache Anwendungen ist es nicht notwendig, die Einstellung für die Network Groups (Netzgruppen) anzupassen.

Ausführliche Information finden Sie auch in der Saia PG5® Hilfe.

1.2.3 Download der Konfiguration

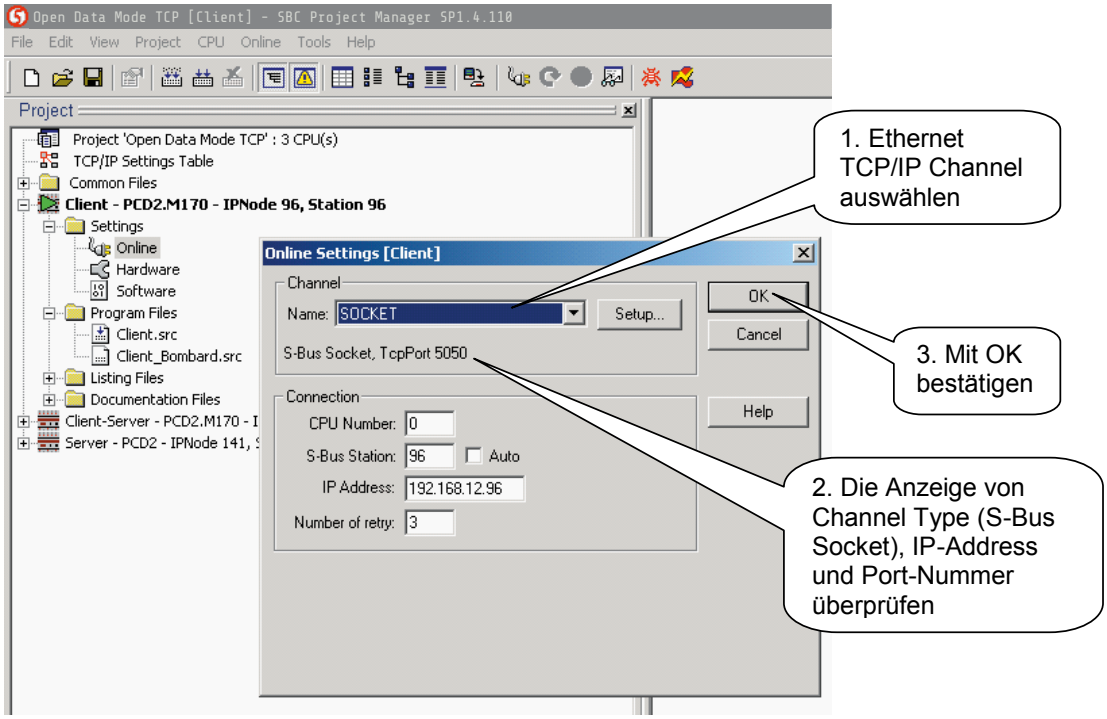
Beim ersten Mal muss die Konfiguration über den PGU-Port unter Benutzung des PGU-Kanals heruntergeladen werden.



Nach der Auswahl des PGU Channel (PGU-Kanals) lassen sich die Konfigurations-einstellungen aus dem Menü Hardware Settings (Hardware-Einstellungen) herunter-laden.

1.3 Online-Verbindung via Ethernet

1.3.1 Auswahl der Online Settings (Online-Einstellungen)

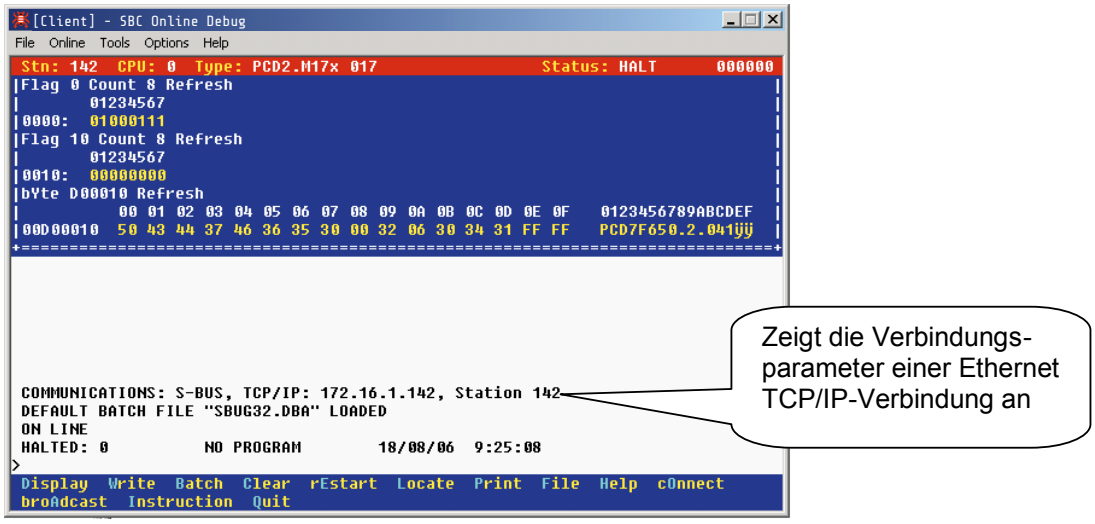


Die Werte für die CPU Number (CPU-Nummer), S-Bus Station und IP Address werden automatisch entsprechend der Hardware-Einstellungen der ausgewählten Station gesetzt; sie lassen sich bei Bedarf jedoch anpassen.

1.3.2 Aufbau einer Online-Verbindung via Ethernet

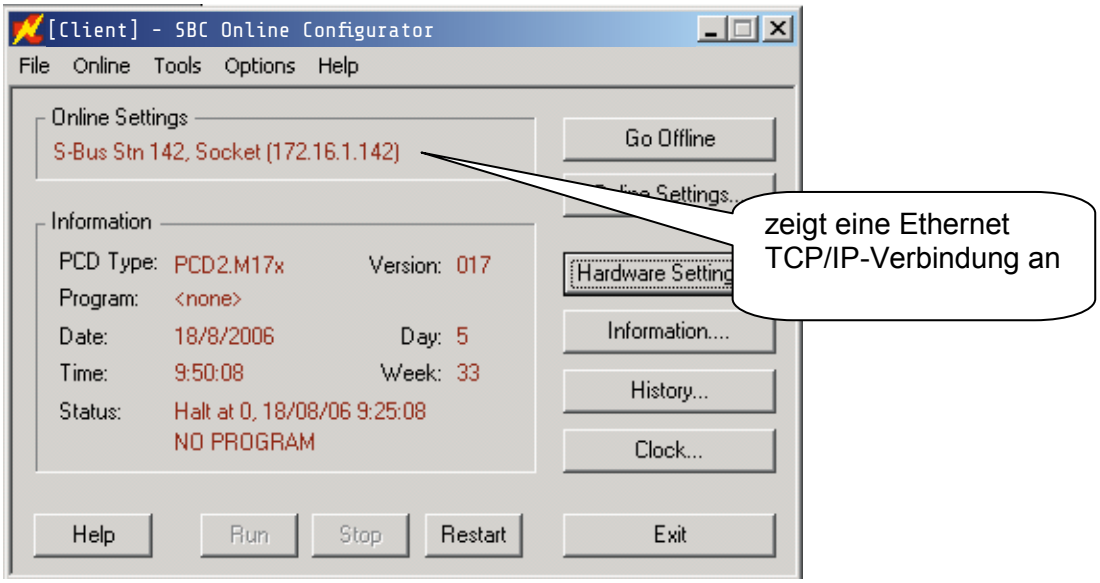
Die Verbindung zu den Saia PCD®s kann mit Hilfe eines der folgenden Online-Werkzeuge aufgebaut werden.

Online-Verbindung mit Hilfe des Debuggers:

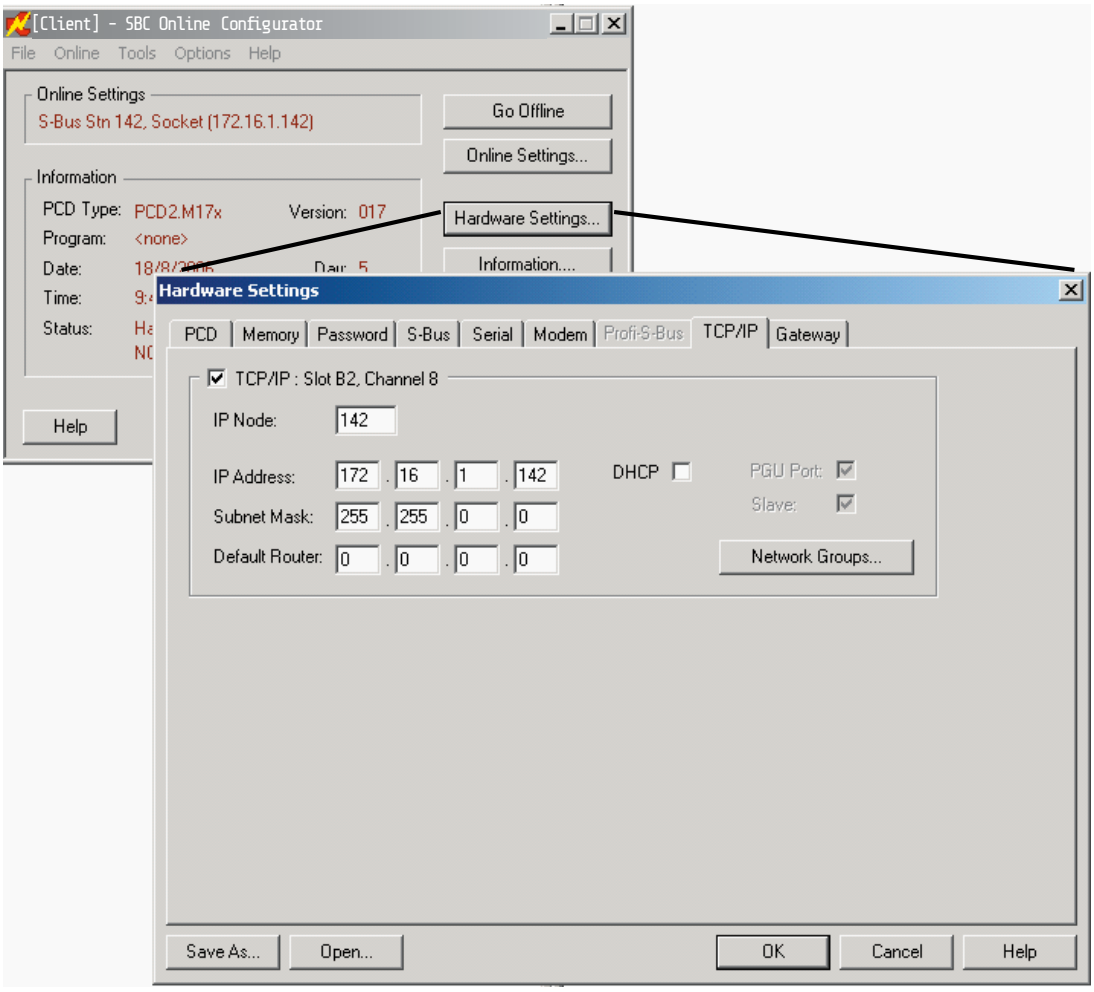


1

Online-Verbindung mit Hilfe des Online Konfigurators:



Die Einstellungen für Ethernet TCP/IP in der Saia PCD® lassen sich auch mit Hilfe der fol- genden Menübefehle online überprüfen oder ändern:



Wenn die Online-Verbindung trotz einer korrekt konfigurierten Saia PCD® nicht funktioniert, siehe Kapitel 5: Diagnose und Fehlersuche und -behebung.

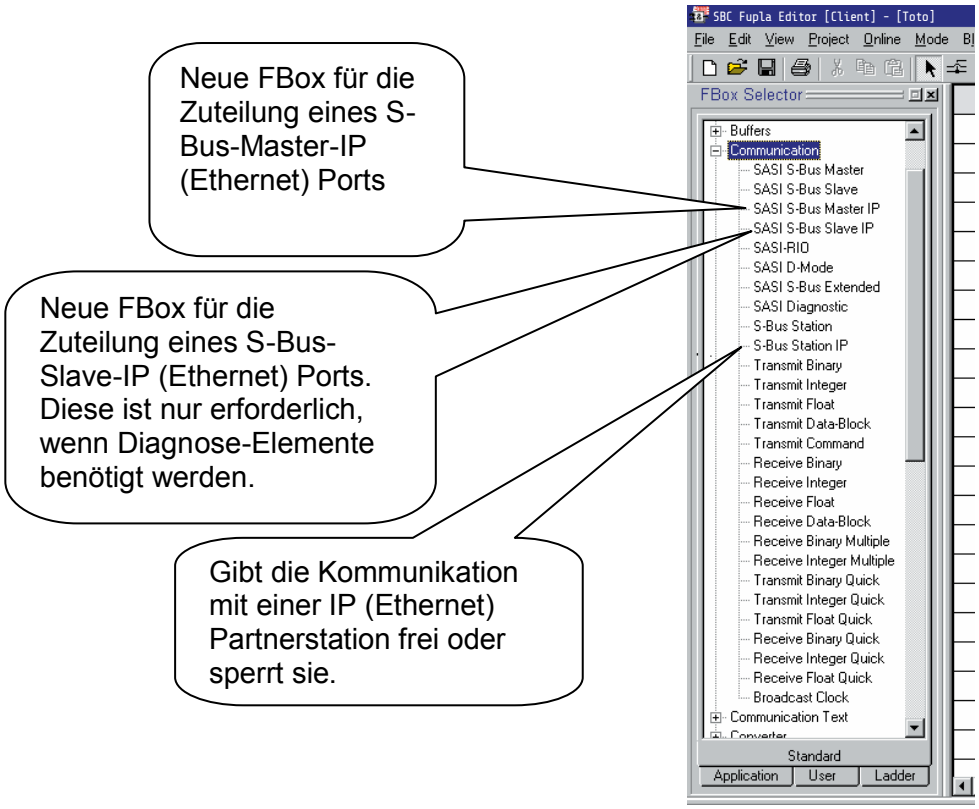
1.4 Erstellen eines Anwenderprogramms in Fupla

Für die Programmierung einer Kommunikation via Ethernet wird eine aktualisierte Kommunikations-Bibliothek benötigt.

Ethernet TCP/IP wird ab Version $2.2.003 unterstützt.

Überprüfen Sie die korrekte Installation der neuen Bibliothek.

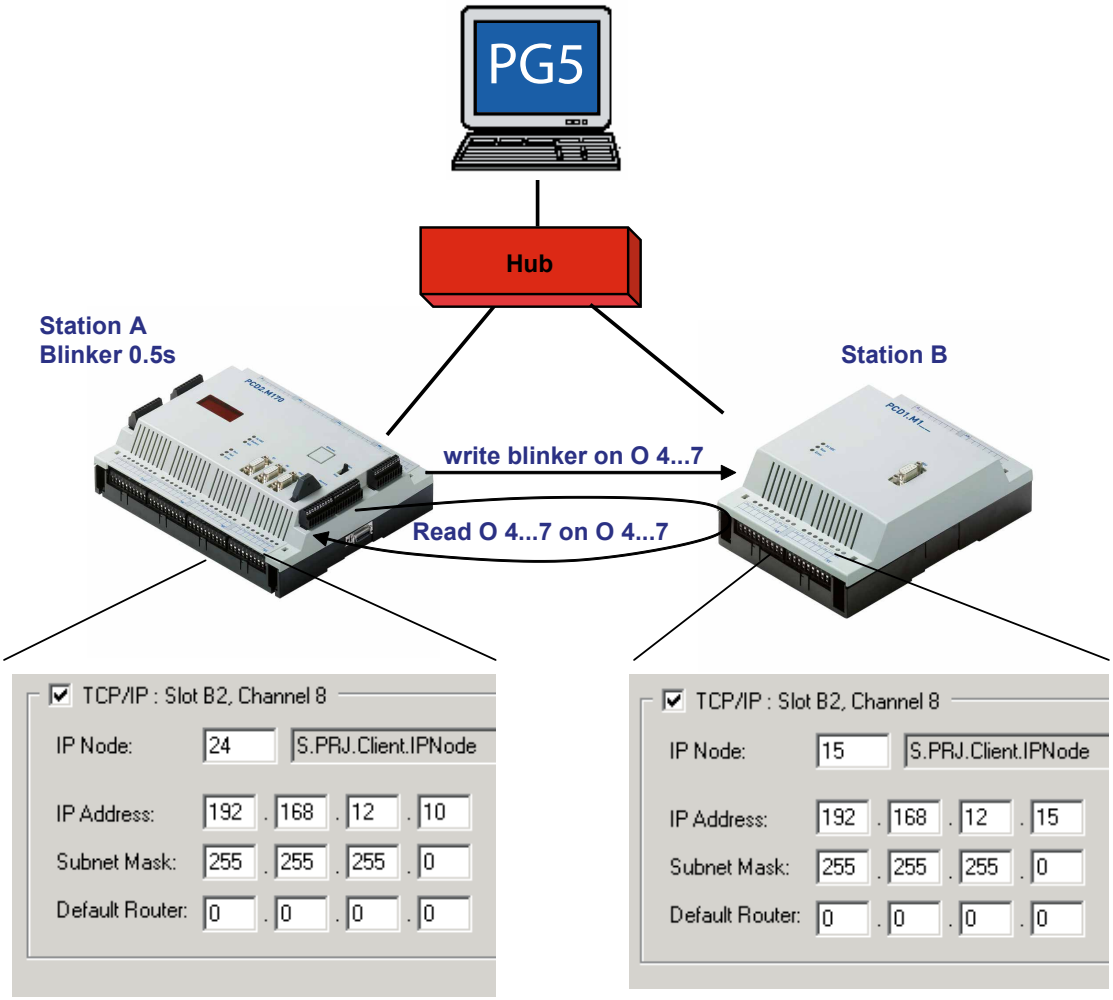
Für alle Sende/Empfangs-Saia PG5® FBoxen (zum Beispiel Transmit/Receive Binary) muss ein neuer zusätzlicher Parameter definiert werden: die IP Node Number (IP-Node-Nummer).



Die Ethernet TCP/IP-Kommunikation wird gleich wie eine Standard-S-Bus-Kommunikation programmiert.

1

Programmierungsbeispiel in Fupla:

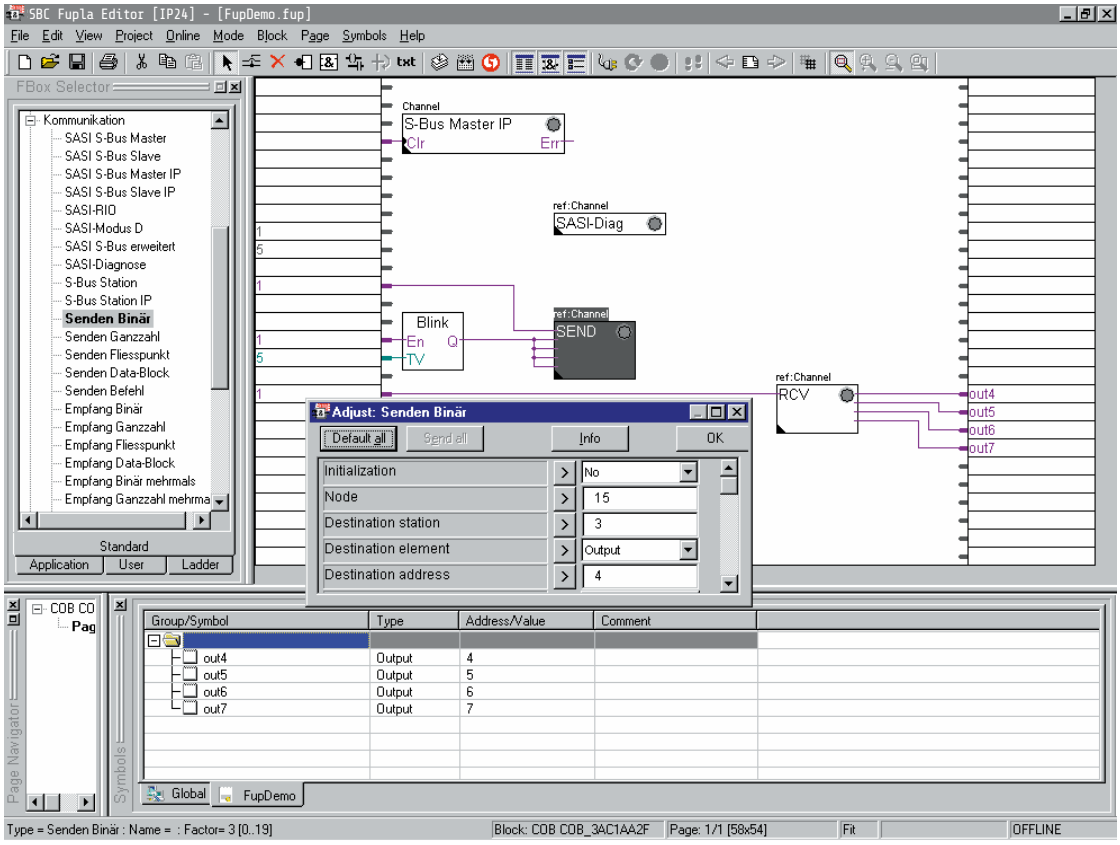


Aufgabe:

Auf Station A wird ein Blinksignal erzeugt. Das Blinksignal wird via Ethernet TCP/IP in die Station B auf die Ausgänge 4...7 kopiert. Diese Ausgänge werden zurück gelesen und auf die Ausgänge 4...7 von Station A kopiert.

1

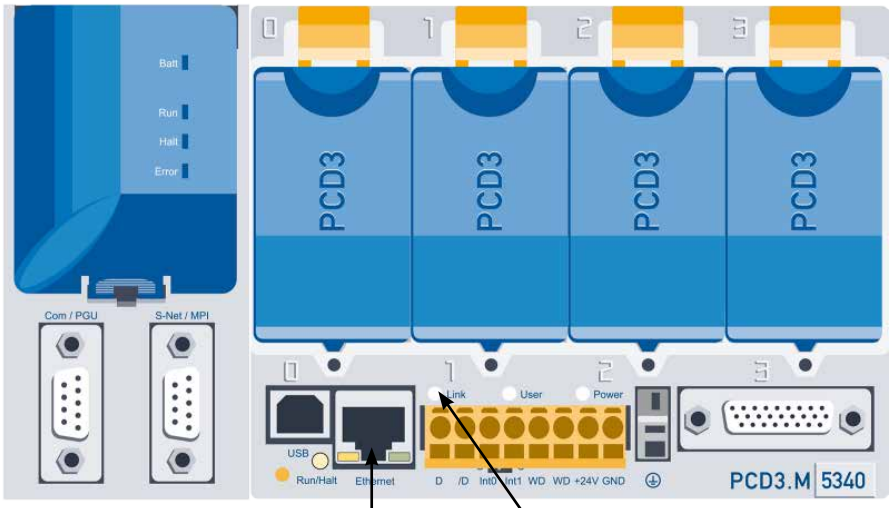
Benutzerprogramm in Fupla:



2 Hardware

2.1 Saia PCD® Systeme mit aufsteckbarem Ethernet Modulen

Die PCD3.Mxx Familie hat Geräte mit aufsteckbaren Ethernet Modulen. Ethernet kommuniziert auf diesen Geräten jeweils über den Channel 9.



2.1.1 PCD3.M3xx0 und PCD3.M5xx0

PCD3.M3xx0 und **PCD3.M5xx0** mit Hardware-Versionen F und neuere Versionen unterstützen den Full-Duplex-Modus und Auto-MDIX (auto-crossing der Signale). Ob eine PCD3.Mxxx0 Fullduplex-und Auto-MDIX unterstützt, kann anhand der LEDS des RJ45 Anschlusses gesehen werden. Wenn der Stecker mit LEDs ausgestattet ist, wird Fullduplex-und Auto-MDIX unterstüstzt.

Bestellinformationen:

PCD3.M5540	Ethernet-fähige PCD3	
PCD3.M3120	Ethernet-fähige PCD3	(ohne Batterie, nicht erweiterbar)
PCD3.M3330	Ethernet-fähige PCD3	(ohne Battierie, erweiterbare E/A)
PCD3.M6340	Ethernet-fähige PCD3	(mit LAN Modul)
PCD3.M6540	Ethernet-fähige PCD3	(mit Profibus DP Master Modul)
PCD3.M2330A4Tx	PCD3 Wide Area Controller	(WAC)
PCD3.M2130V6	PCD3 Compact	
PCD3.M5560	PCD3 Power CPU	
PCD3.M6360	PCD3 Power CPU	(mit LAN Modul)
PCD3.M5560	PCD3 Power CPU	(mit Profibus DP Master Modul)

2.1.2 PCD2.M5540

Die **PCD2.M5540** verfügt über einen integrierten Ethernet-Schalter mit zwei Anschlüssen. Die Ethernet-Kommunikation der PCD2.M5540 nutzt den Kanal 9. Full-Duplex-Modus und Auto-MDIX (Auto-Crossing der Signale) werden von allen Hardware-Versionen der PCD2.M5540 unterstützt.

2.1.3 PCD1.M2120

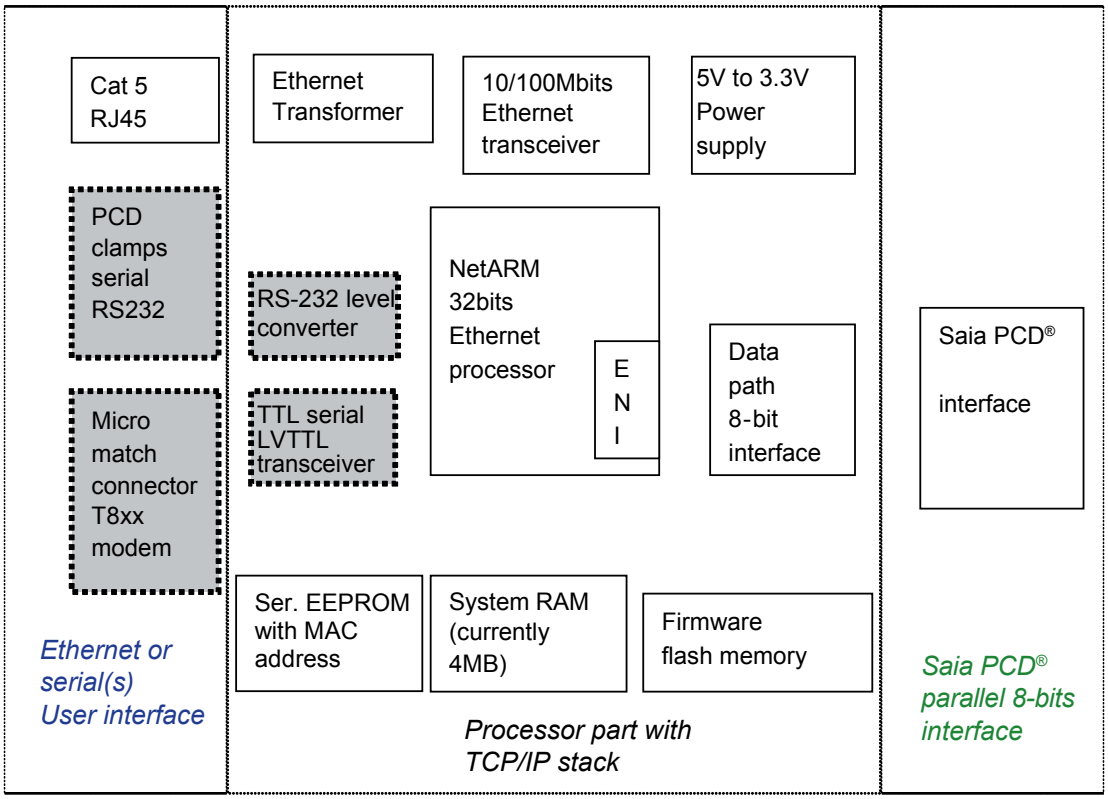
Die **PCD1.M2120** verfügt über einen integrierten Ethernet-Schalter mit zwei Anschlüssen. Die Ethernet-Kommunikation der **PCD1.M2120** nutzt den Kanal 9. Full-Duplex-Modus und Auto-MDIX (Auto-Crossing der Signale) werden von allen Hardware-Versionen der PCD1.M2120 unterstützt.

2.1.4 PCD3.T665/T666 Ethernet RIO

Die **PCD3.T665/T666 Ethernet RIO** verfügt über eine integrierte Ethernet-Schnittstelle mit zwei Anschlüssen. Die Ethernet-Kommunikation nutzt den Kanal 9 der PCD3.T665/T666. Full-Duplex-Modus und Auto-MDIX (Auto-Crossing die Signale) werden durch die PCD3.T665/T666 von allen Hardware-Versionen unterstützt.

2.2 Ethernet TCP/IP-Modul PCD7.F65x

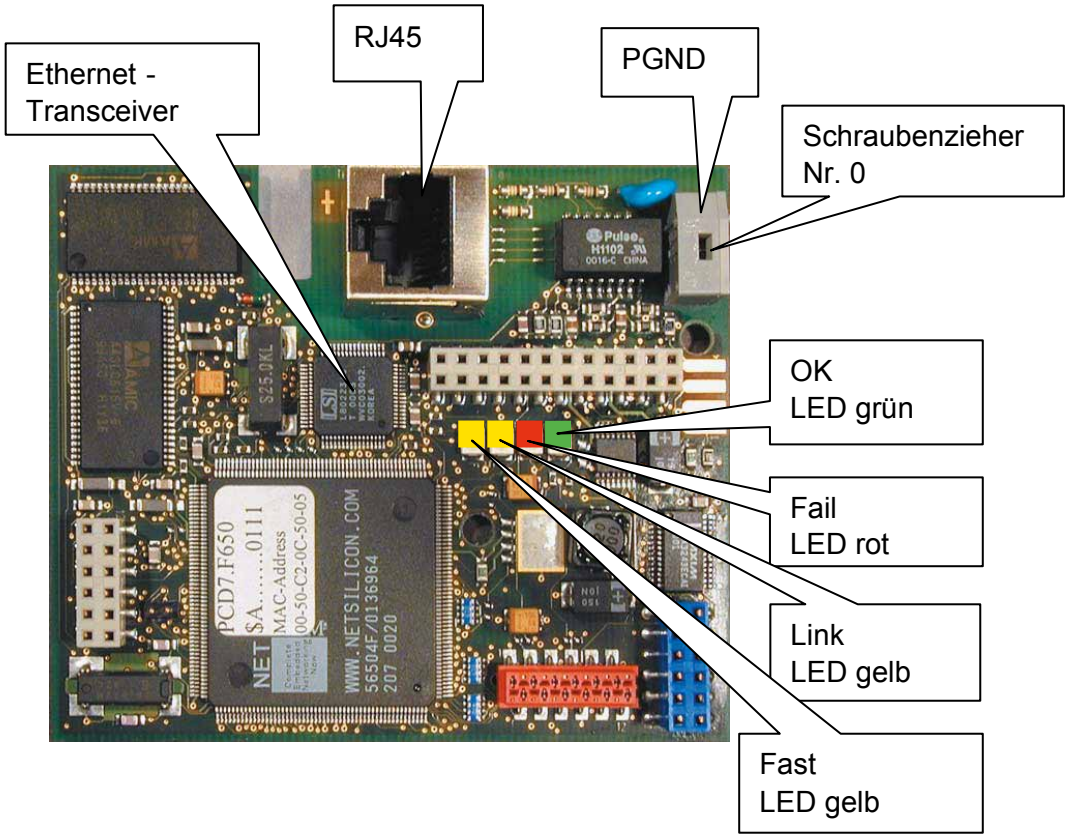
2.2.1 Blockschaltbild



Anmerkung: Die schattierten Blöcke werden gegenwärtig durch die Firmware nicht unterstützt oder können in Ihrer Hardware fehlen.

Die seriellen Schnittstellen werden durch den Ethernet-Prozessor verwaltet und können nicht als reguläre Saia PCD® Schnittstellen benutzt werden. Diese beiden Schnittstellen sind für die IP-Verbindung (PPP, SLIP,…) via Modem oder für die Fehlersuche und -behebung vorgesehen.

2.2.2 Layout PCD7.F650/F655



2.2.3 Layout PCD7.F651/F652

Das Layout entspricht jenem der PCD7.F650/F655 mit der Ausnahme, der verschiedenen RJ45-Stecker Varianten.

2.2.4 PGND-Verbindung

Für die Einhaltung der EMV-Werte muss PGND (Schutzerde) immer mit der Chassis-Erde verbunden sein.

Anschliessen der Schutzerde:

- Mit einem Schraubenzieher (Nr. 0) durch den Schlitz die Federkraftklemme öffnen
- Erdungsdraht in die PGND-Öffnung stecken
- Schraubenzieher wieder herausziehen
- An Schutzerdeleiter ziehen, um zu prüfen ob er korrekt richtig angeschlossen ist

2.2.5 LED-Funktionen

Fast (gelb) Diese LED leuchtet, wenn eine schnelle 100 Mbit/s-Verbindung erkannt wird, und ist dunkel bei 10 Mbit/s

Link (gelb) Diese LED leuchtet, wenn eine Verbindung erkannt wird und blinkt bei Ethernet-Verkehr (aus bei Aktivität, ein bei keiner Aktivität)

Fail (rot) Diese LED blinkt, wenn ein Hardware- oder Firmware-Fehler erkannt wurde

OK (grün) Diese LED zeigt den fehlerfreien Betrieb an (blinkt mit einer Frequenz von 2 Hz)

Die gelben LEDs haben während der Boot-Sequenz des Prozessors eine andere Bedeutung als im Normalbetrieb. Unmittelbar nach einem Reset (beim Einschalten) zeigt die Fast LED eine Verbindung mit 10 Mbit/s an, während die Link LED eine 100-Mbit/s-Verbindung signalisiert.

Link LED gelb	Fast LED gelb	Kommentar
LINK 100	LINK10	Nach des Resets
LINK + ACT	10/100	Während des Betriebs

Die Fail LED (rot) zeigt an, dass ein Fehler erkannt wurde. Die Blinkfrequenz liefert Information über den erkannten Fehler. Die OK LED kann ebenfalls blinken

Abhängig vom erkannten Fehler können die grünen und roten LEDs blinken. Wenn die grüne LED alleine blinkt, ist kein Fehler vorhanden.

Verletzungsgefahr! Der Ethernet-Transceiver wird ziemlich heiss. Ohne Verbindung liegt seine Temperatur etwa 30° über der Umgebungstemperatur. Dies ist normal.

2.2.6 RJ45-Pinbelegung

Pin	Name	Adernpaar verdrillt	Drahtfarbe	Ethernet	Fast Ethernet	Bemerkung
1	TPO+	Paar 2	weiss/orange	TPO+	TPO+	
2	TPO-	Paar 2	orange	TPO-	TPO-	
3	TPI+	Paar 3	weiss/grün	TPI+	TPI+	
4	Term. 1a	Paar 1	blau	-	Paar 1	abgeschlossen mit 75 Ω*
5	Term. 1b	Paar 1	weiss/blau	-	Paar 1	zusammen mit Pin 4*
6	TPI-	Paar 3	grün	TPI-	TPI-	
7	Term. 4a	Paar 4	weiss/braun	-	Paar 4	abgeschlossen mit 75 Ω*
8	Term. 4b	Paar 4	braun	-	Paar 4	zusammen mit Pin 8*
* Bob Smith-Terminierung						

Für Ethernet (10 Mbit/s) können Adernpaar 1 und Adernpaar 4 fehlen.
Die Fast Ethernet-Kabelbelegung (CAT5 Kabel) ist kompatibel zu 10 Mbit/s Ethernet.

2.2.7 Verkabelung

Kabeltyp:

Dieses Modul ist für den Einsatz mit abgeschirmtem und nicht abgeschirmtem 100-Ohm-Kabel (UTP oder STP) ausgelegt. Bei elektromagnetischen Störungen ist abgeschirmtes Kabel für einen höheren Durchsatz sehr zu empfehlen. Wenn eine 100-Mbit/s-Verbindung benutzt wird, ist Kabeltyp CAT 5 die Minimalanforderung. Für eine 10-Mbit/s-Verbindung muss mindestens der Kabeltyp CAT 3 verwendet werden.

Dieses Modul benutzt ein Autonegotiationsverfahren zur Bestimmung der Übertragungsgeschwindigkeit und der Betriebsart. Um die Verbindung mit 100 Mbit/s und Voll-Duplex aufzubauen, müssen beide Enden die Autonegotiation unterstützen. Ist dies nicht der Fall, wird das Modul die Verbindung mit 10 Mbit/s und Half-Duplex Mode herstellen. Das Autonegotiationsverfahren erkennt den verwendeten Kabeltyp jedoch nicht. Das bedeutet, dass Sie die Verbindung zwar mit einem CAT 3-Kabel mit 100 Mbit/s aufbauen können, aber eine solche Verbindung eventuell nicht zuverlässig arbeiten wird (dies kann zum Beispiel mit einem Überkreuzungskabel auftreten).

Eine spezielle Call System Function (CSF) erlaubt es die PCD7.F65x in Voll-Duplex oder Half-Duplex Mode zu versetzen und entweder mit 10 Mbit/s oder 100 Mbit/s zu kommunizieren. Mehr Details hierzu finden sie im Kapitel über die speziellen CSFs.

Kabellänge:

Maximal 100 m

Biegeradius:

Für den Biegeradius gelten gewisse Einschränkungen.

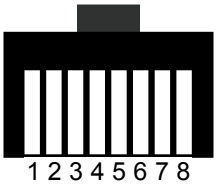
Gemäss EIA SP-2840A beträgt der minimale Biegeradius das 10-fache des Aussendurchmessers des Kabels. Gemäss ISO DIS 11801 beträgt er das 8-fache des Aussendurchmessers.

Verdrahtung:

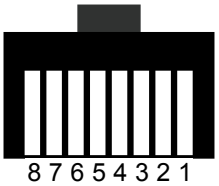
Die Verkabelung kann entweder Peer-to-Peer mit einem Überkreuzungskabel oder mit einem normalen Patchkabel via Hub oder einem Switch erfolgen.

Pinbelegung:

Stecker (Ansicht Steckerende, von vorne, Kabel verläuft nach hinten weg)



Buchse (Ansicht Wandsteckdose, von vorne)



2.2.8 Kabel

Kabelbelegung Überkreuzungskabel (Peer-to-Peer):

Pin	Farbe		Pin
1	or/ws	◄——►	3
2	orange	◄——►	6
3	gn/ws	◄——►	1
4	blau	◄——►	4
5	ws/bl	◄——►	5
6	gn	◄——►	2
7	ws/br	◄——►	7
8	braun	◄——►	8

Kabelbelegung Patchkabel (über Hub):

Pin	Farbe		Pin
1	or/ws	◄——►	1
2	orange	◄——►	2
3	gn/ws	◄——►	3
4	blau	◄——►	4
5	ws/bl	◄——►	5
6	gn	◄——►	6
7	ws/br	◄——►	7
8	braun	◄——►	8

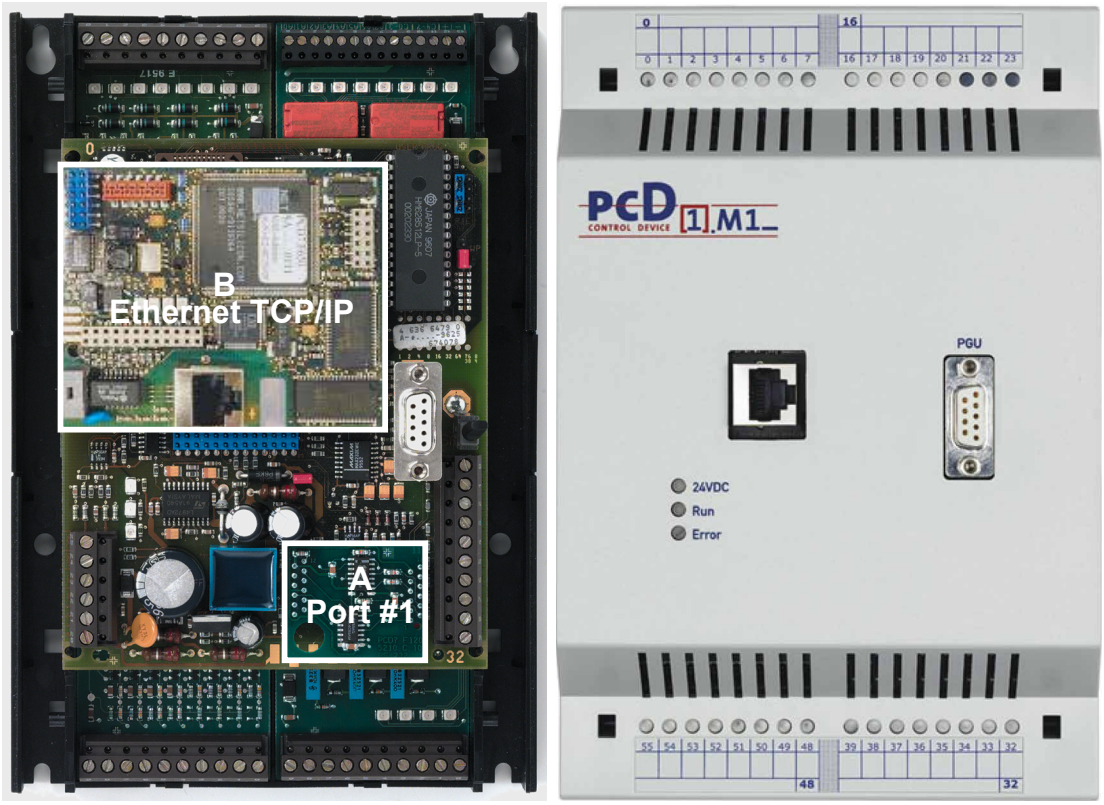
Farben definiert gemäss EIA/TIA T568B.

2.3 Konfigurierte Systeme mit PCD7.F65x

2

2.3.1 PCD1.M135F65x

PCD7.F65x auf Steckplatz B / Channel 9 PCD1.M135F65x



PGND-Leitung von PCD7.F65x muss mit Klemme 23 der PCD1 verbunden werden.

Bestellangaben:

- Als konfiguriertes System:

  PCD1.M135F655 PCD1 konfiguriertes System mit Ethernet-Modul

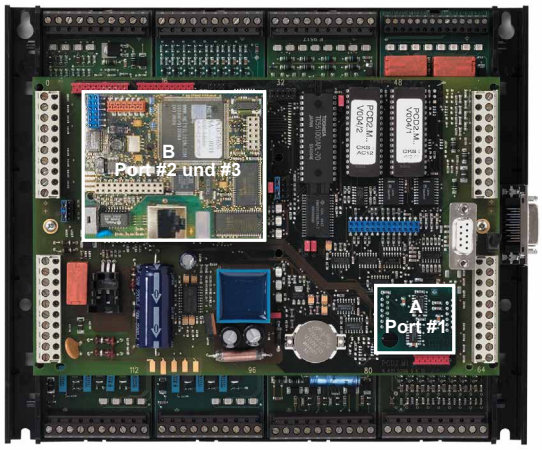
- Als Zusatz:

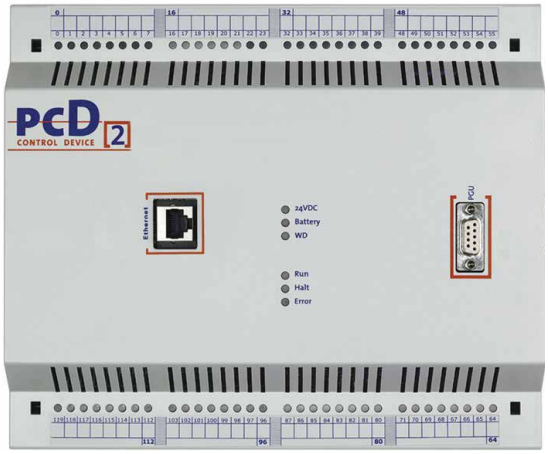
  PCD7.F655 Ethernet-Modul für PCD1/PCD2
  4‘104‘7409‘0 Deckel für PCD1.M135 mit Ausschnitt für RJ45-Stecker

Mit älteren PCD1-Hardware-Versionen kann es Einschränkungen geben.

2.3.2 PCD2.M150F65x

PCD7.F65x auf Steckplatz B / Channel 9 PCD2.M150F65x





PGND-Leitung von PCD7.F65x muss mit Klemme 27 der PCD2 verbunden werden. Die Saia PCD® Klemmen 30...39 für RS-232 reservieren (RS-232 in Vorbereitung).

Bestellangaben:

- Als konfiguriertes System:

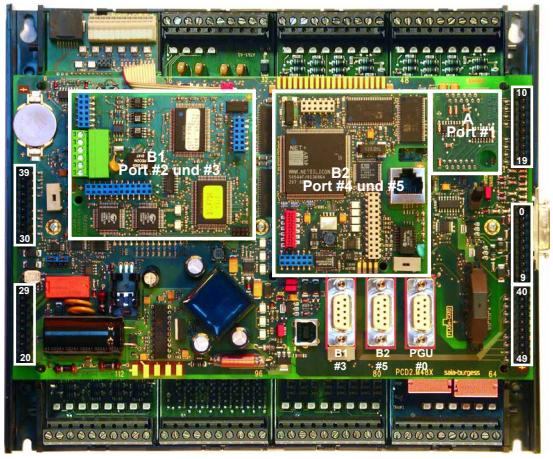
  PCD2.M150F655 PCD2 konfiguriertes System mit Ethernet-Modul

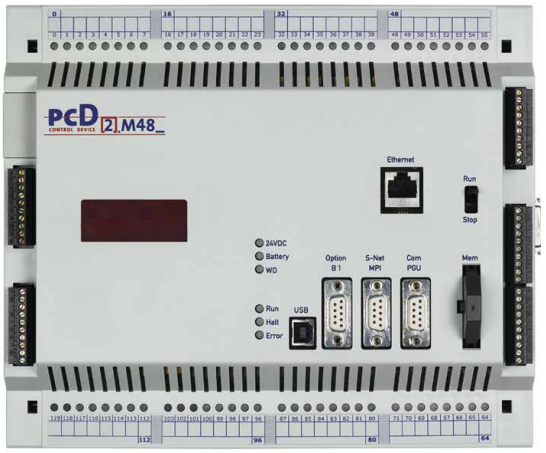
- Als Zusatz:

  PCD7.F655 Ethernet-Modul für PCD1/PCD2
  4‘104‘7410‘0 Deckel für PCD2.M150 mit Ausschnitt für RJ45-Stecker

2.3.3 PCD2.M480F65x-2

PCD7.F65x auf Steckplatz B2 / Channel 8 und zusätzlich auch auf Steckplatz B1 / Channel 9





PGND-Leitung von PCD7.F65x muss mit PGND-Klemme neben B1 DB9 verbunden werden. Die Saia PCD® Klemmen 40...49 für RS-232 reservieren (RS-232 in Vorbereitung).

Bestellangaben:

- Als konfiguriertes System:

  PCD2.M480F655-2 PCD2 konfiguriertes System mit zwei Ethernet-Modulen

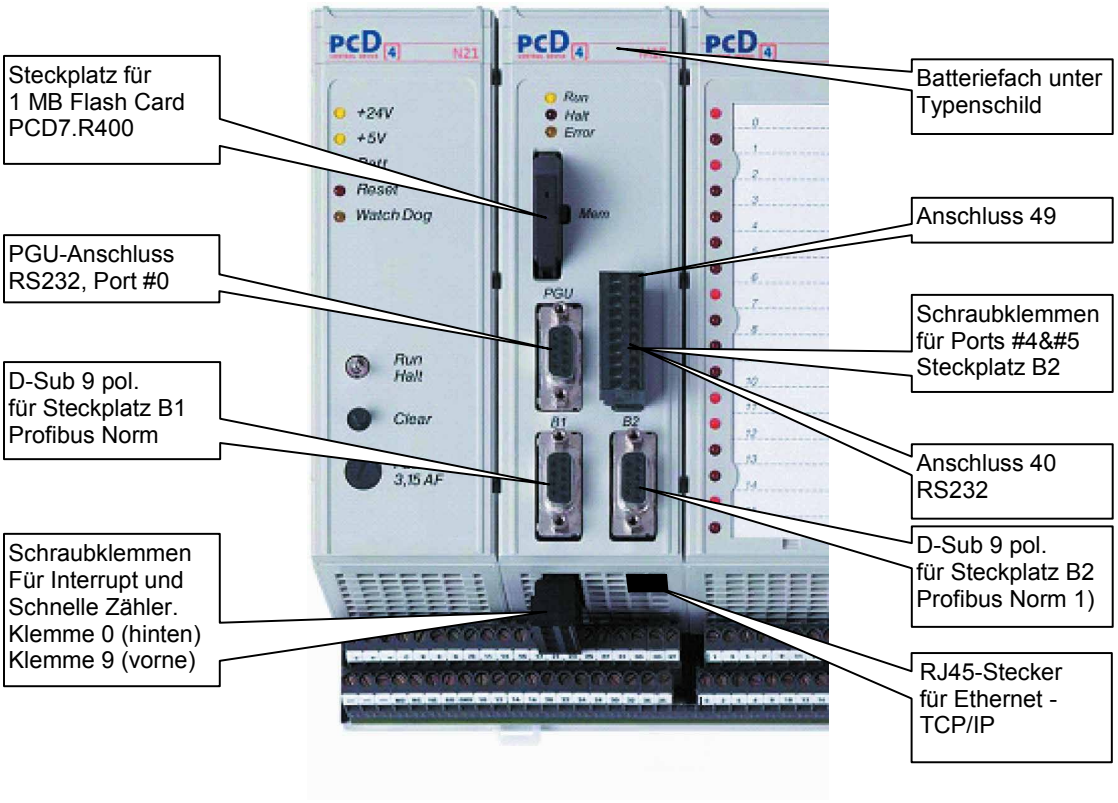
- Als Zusatz:

  PCD7.F655 Ethernet-Modul für PCD1/PCD2
  4‘104‘7503‘0 Deckel für PCD2.M480F655-2 mit Ausschnitt für zwei RJ45-Stecker

2.3.4 PCD4.M170Fx9

Die Ethernet-Verbindung erfolgt über den am unteren Rand der PCD4.M17x angebrachten RJ 45 Stecker. Ist nur als konfiguriertes System PCD4.M170Fx9 (wobei x den Code der auf B1 aufgesteckten Schnittstelle bezeichnet) verfügbar.

PCD7.F65x auf Steckplatz B1 / Channel 9



PGND-Leitung von PCD7.F65x muss mit PGND verbunden werden. Die Saia PCD® Klemmen 40...49 für RS-232 reservieren (RS-232 in Vorbereitung).

Bestellangaben:

- Als konfiguriertes System:

  PCD4.M170Fx9 PCD4 konfiguriertes System mit Ethernet Modul

2.3.5 PCD7.F65x auf xx7

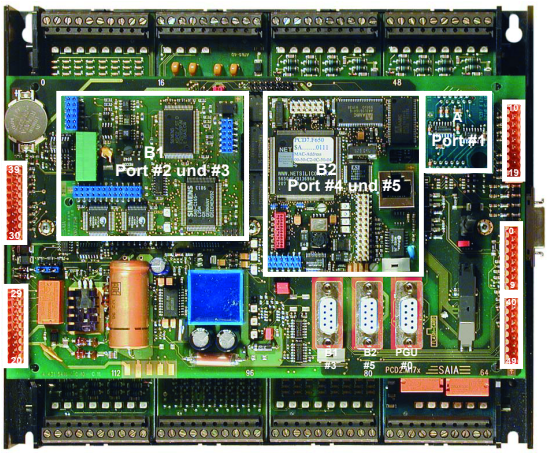
Für die ethernetfähige xx7 mit PCD7.F65x gibt es konfigurierte und nicht konfigurierte Systeme. Wir verweisen hierzu auf das xx7 Ethernet-Handbuch 26-791.

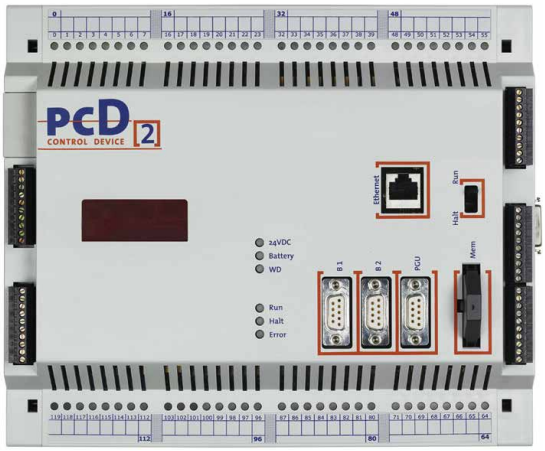
2.4 Nichtkonfigurierte ethernetfähige Systeme

2.4.1 PCD2.M170 mit PCD7.F65x

2

PCD7.F65x auf Steckplatz B2 / Channel 8





PGND-Leitung von PCD7.F65x muss mit PGND-Klemme neben B1 DB9 verbunden werden. Die Saia PCD® Klemmen 40...49 für RS-232 reservieren (RS-232 in Vorbereitung).

Bestellangaben:

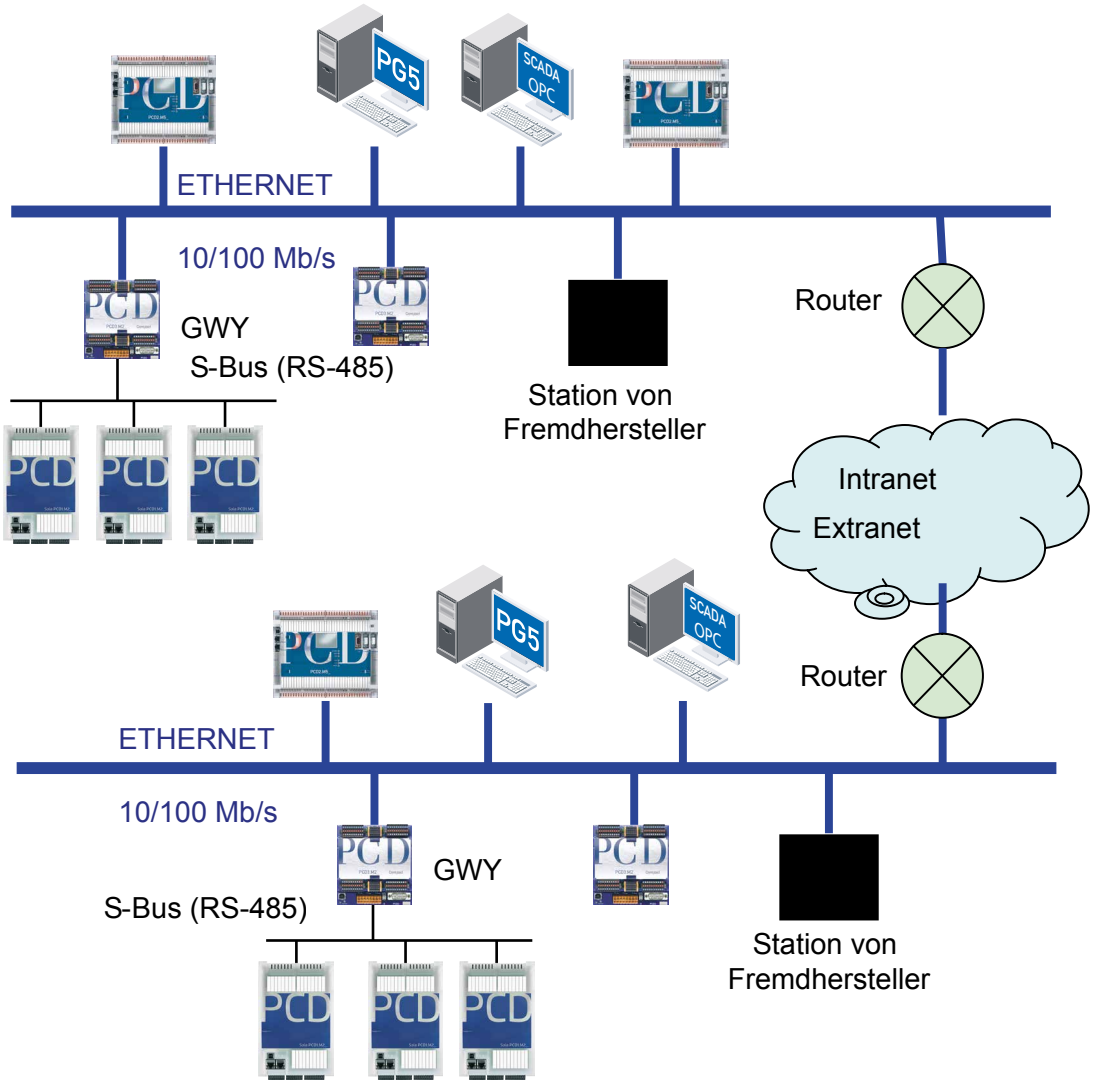
PCD7.F655	Ethernet-Modul für PCD1/PCD2

2.4.2 PCD7.F65x auf xx7

Für die ethernetfähige xx7 mit PCD7.F65x gibt es konfigurierte und nicht konfigurierte Systeme. Wir verweisen hierzu auf das xx7 Ethernet-Handbuch 26-791.

3 Eigenschaften und Funktionen

3.1 Mögliche Verbindungen und Netz-Topologien



Ein Ethernet erlaubt normalerweise alle Arten von Verbindungen zwischen den an das Netz angeschlossenen Stationen. Das Saia PG5® Programmierwerkzeug und ein SCADA-System können direkt über das Ethernet auf eine PCD und eine Station eines Fremdherstellers zugreifen. Das Saia PG5® Programmierwerkzeug und das SCADA-System, das auf der SCOM.dll basiert, sind Clients.

Ethernet-TCP/IP-Module von Saia Burgess Controls benötigen kein proprietäres Netz und lassen sich in Standard-Netzen mit Standard-Komponenten wie Hubs, Switches, Router usw. einsetzen. Die Ethernet TCP/IP-Module von Saia Burgess Controls unterstützen alle aktuellen Netz-Topologien.

Es kann ein S-Bus-Netz aufgebaut und unter einer an das Ethernet angeschlossenen Gateway-Station positioniert werden. Auf die PCD in diesem S-Bus-Subnetz wird indirekt über eine PCD zugegriffen, die als Ethernet-Gateway-Station konfiguriert wurde. In dieser Eigenschaft leitet sie die vom Ethernet empfangenen Meldungen an das untergeordnete S-Bus-Netz weiter. Auf diese Weise lassen sich einfach mehrere S-Bus-Netze in das Ethernet integrieren.

In einem Ethernet TCP/IP-Netz gibt es zwei Protokolle:

- S-Bus-Protokoll (UDP/IP, Port 5050)
- Open Data-Mode-Protokoll (UDP/IP oder TCP/IP und vom Anwender definierte Ports), das zur Implementierung eines Anwender-Protokolls eine Socket-Schnittstelle benutzt.

Übersicht der Funktionen und wie sie erreicht werden können

Funktionen	Erreicht mit		
	S-Bus-Ethernet UDP/IP Port 5050	Open Data Mode Ethernet UDP/IP	Open Data Mode Ethernet TCP/IP
Programmierung der PCD und Fehlersuche und -behebung mit Saia PG5®	✓		
PCD-Multimaster-S-Bus-Kommunikation	✓		
Anschluss von PCD-Stationen an ein SCADA-System	✓	✓	✓
Kommunikation zwischen PCD-Stationen und einem Fremdsystem			
Implementierung von Anwender- Protokollen		✓	✓
S-Bus-Gateway-Funktionalität (vom Ethernet zum Standard-S-Bus)	✓		

S-Bus UDP, Open Data Mode UDP und Open Data Mode TCP können gleichzeitig mit derselben PCD betrieben werden.

3.2 Ether-S-Bus

Das Ether-S-Bus-Protokoll wird benutzt für die Kommunikation zwischen
- zwei PCDs
- einer PCD und dem Saia PG5® Programmierwerkzeug
- einer PCD und anderen Stationen (SCADA-System, OPC-Server oder einer andern PLC, die das Ether-S-Bus-Protokoll unterstützt).

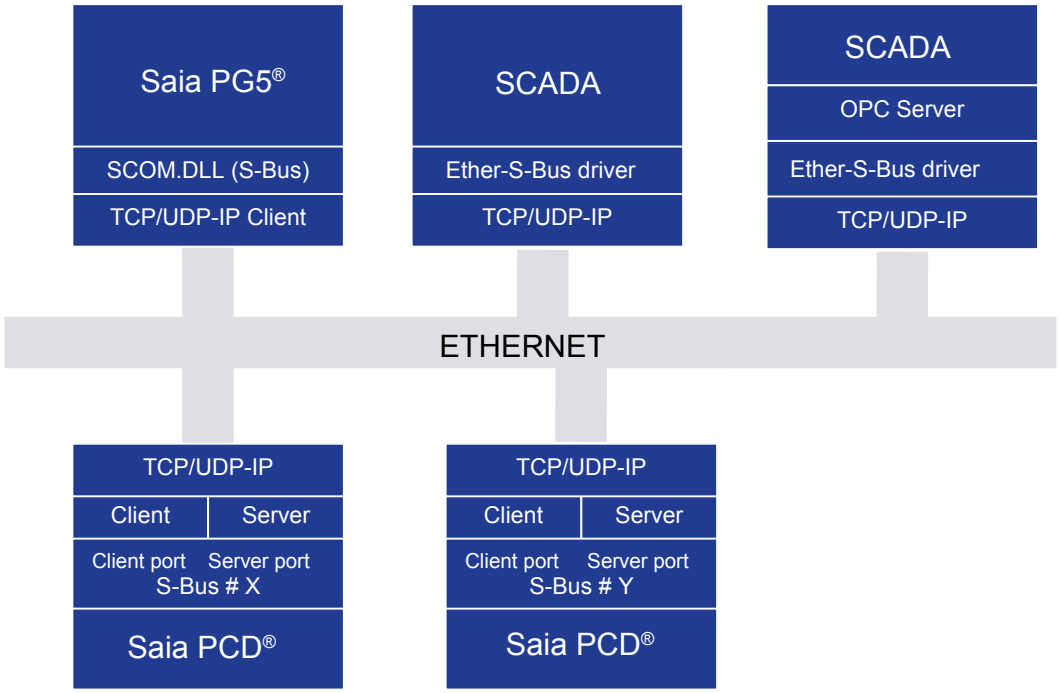
Die Daten werden unter Verwendung der bekannten STXM/SRXM-Befehle in IL oder der komfortablen FUPLA-FBoxen ausgetauscht. Die Syntax ist dem vorhandenen S-Bus-Telegramm sehr ähnlich.

S-Bus via IP ist mit UDP-Sockets über den festen **Eingangs-**Port 5050 implementiert. Wenn Sie Verbindungen über Firewalls verwalten, müssen Sie sicherstellen, dass dieser Port in der Firewall-Konfiguration **freigeschaltet** ist.

Die Multimaster-Kommunikation zwischen PCDs wird im Ether-S-Bus unterstützt. Daher verfügt jede PCD über einen Server- und einen Client-Port und kann gleichzeitig als Client wirken oder als Server-Station passiv bleiben.

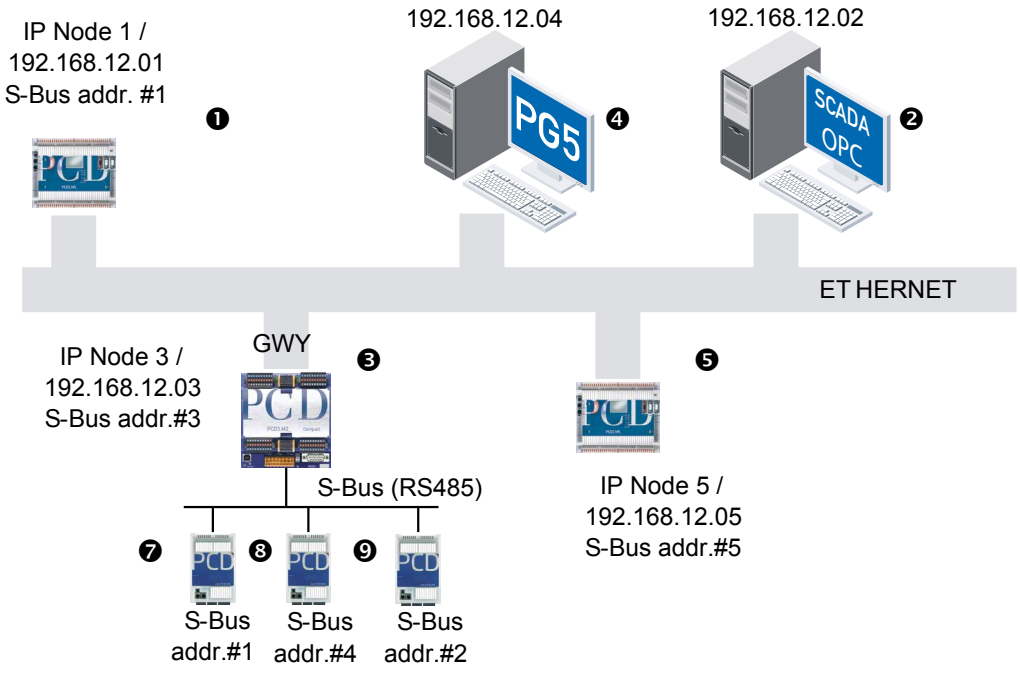
Darstellung der Anwendungsschichten

- Saia PG5® basierend auf SCOM.dll
- SCADA-System unterstützt Ether-S-Bus
- SCADA-System verbunden mit einem OPC-Server
- PCD mit Ether-S-Bus und Server-Ports



Ein SCADA-System kann entweder Ether-S-Bus unterstützen oder mit einem OPC-Server verbunden sein, welcher Ether-S-Bus unterstütz.

3.2.1 Netz-Topologie und -Adressierung



Jedes IP-Modul im Ethernet wird durch eine weltweit eindeutige, feste MAC-Adresse identifiziert. Danach wird die Station mit einer IP-Adresse konfiguriert. Diese ist oft innerhalb des Unternehmens standortabhängig und wird im Internet nicht registriert (private IP-Adresse). Ein IP-Node wird nach den Spezifikationen von SBC definiert.

Die Umsetzungstabelle (MAC-Adresse ↔ IP-Adresse) wird intern durch die TCP/IP-ARP- Stacks verwaltet.

Um die Adressbehandlung für den Benutzer so einfach wie möglich zu gestalten, wurde mit dem IP-Node eine zusätzliche Abstraktionsschicht eingeführt. Das IP- Modul wird im Saia PG5® Hardware-Konfigurator durch die Zuteilung einer IP-Adresse, eines einfachen IP-Node und einer S-Bus-Adresse konfiguriert. Oft wird dieselbe Nummer für die S-Bus-Adresse und den IP-Node einer Station benutzt.

Später, im Programm, wird für die Kommunikation auf dem Ethernet nur noch die IP-Node-Nummer und die S-Bus-Adresse der Station benutzt. Nach der Konfiguration einer Station, muss sich der Anwender nicht mehr um die IP-Adresse kümmern.

Die Tabelle mit allen Kombinationen von IP-Adressen, IP-Nodes und S-Bus-Adressen für das ganze Projekt wird durch Saia PG5® generiert und in der PCD gespeichert.

Tabelle der Stationen aus Sicht der PCD von IP-Knoten #1 aus (IP-Adresse 192.168.12.01, S-Bus-Addresse #1).

Station	IP-Adresse	IP-Knoten	S-Bus-Adresse
❺	192.168.12.05	5	5
❸	192.168.12.03	3	3
❼	192.168.12.03	3	1
❽	192.168.12.03	3	4
❾	192.168.12.03	3	2

Weitere Einzelheiten über die Behandlung der Adressentabelle ist in der Saia PG5® Hilfe zu finden: Hardware-Einstellungen - TCP/IP-Seite.

Vor dem Aufruf des Befehls STXM/SRXM muss die Zieladresse wie bei jedem Standard S-Bus-Protokoll in das Address-Register geladen werden. Wie in den nachstehenden Beispielen gezeigt, erfolgt die Adressierung in zwei Adressfeldern.

Adressierende Station ❺	LDL R 100 5 ;S-Bus-Adresse LDH R 100 5 ;IP-Knoten STXM 9 10 F 500 O 32
Adressierende Station ❸	LDL R 100 3 ;S-Bus-Adresse LDH R 100 3 ;IP-Knoten STXM 9 10 F 500 O 32
Adressierende Einheit ❽	LDL R 100 4 ;S-Bus-Adresse LDH R 100 3 ;IP-Knoten STXM 9 10 F 500 O 32

3.2.2 Programmierung, Fehlersuche und -behebung via Ethernet

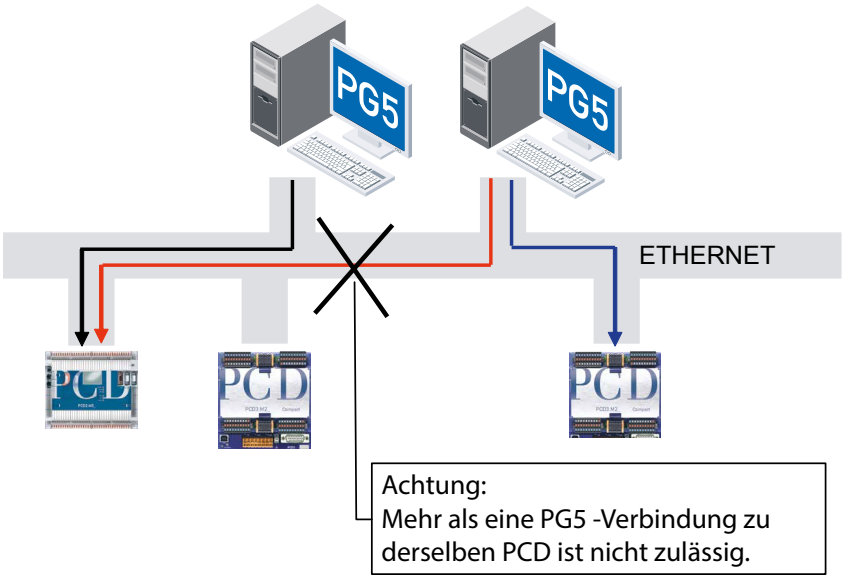
Konfigurieren, Programmieren, Downloaden und Debuggen einer PCD-Station können via Ethernet mit dem Programmierwerkzeug Saia PG5® und dem bekannten S- Bus-Protokoll ausgeführt werden. Via Ethernet erfolgt das Programm-Download viel schneller als mit dem S-Bus über eine serielle Schnittstelle. Dies ist bei umfangreichen Benutzerprogrammen sehr nützlich.

Die Programmierung via Ethernet mit S-Bus-Protokoll ist ab Saia PG5® Version 1.1 möglich.Dazu werden verwendet:

- die Instruktionsliste (IL) oder
- die komfortablen FUPLA FBoxen.

Die Daten werden in der IL mit den üblichen STXM/SRXM-Befehlen ausgetauscht. Die Syntax ist dem Standard S-Bus-Telegramm sehr ähnlich. Die Zugriffssicherheit ist durch den bekannten S-Bus-Passwortschutz gewährleistet.

Die Benutzung mehrerer Saia PG5®-Programmierwerkzeuge auf einem Ethernet ist möglich. Dies erlaubt die parallele Entwicklung und Inbetriebnahme grosser Projekte durch mehrere Programmierer oder Teams, um Zeit zu gewinnen. Hingegen ist gleichzeitig nur eine S-Bus UDP Full-Protokoll-Verbindung auf die PCD zulässig.

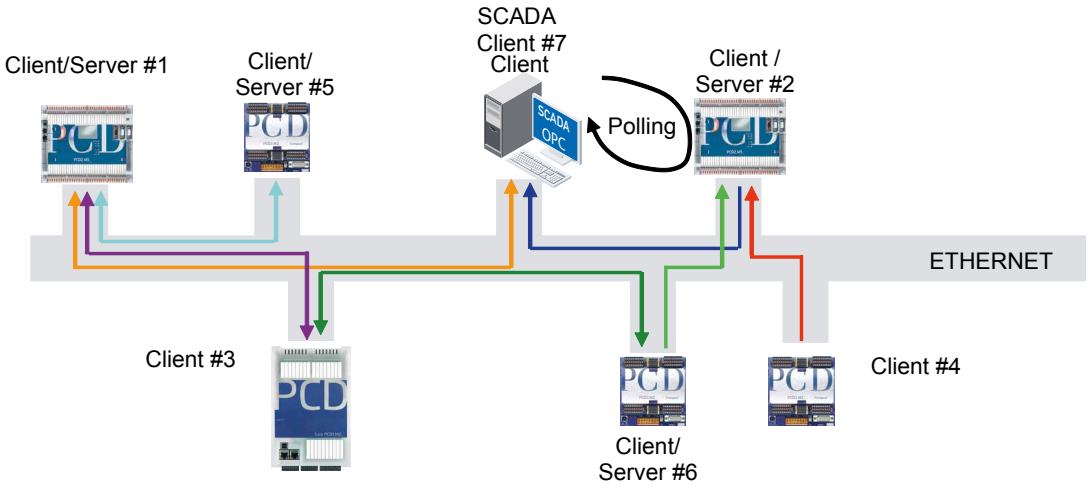


3.2.3 Multimaster-Kommunikation

Verglichen mit dem Standard S-Bus-Protokoll ist die neue Master-Master-Funktionalität für Ethernet eine wesentliche Verbesserung. Der Standard S-Bus über RS-485 erlaubte nur einen Client pro Netz. Bei S-Bus via Ethernet können alle Stationen, als Clients und Server arbeiten.

Durch die Auslegung des Netzes für den Multimaster-Betrieb ist es möglich, leistungsfähige, ereignisgesteuerte Kommunikationsverbindungen zwischen PCD-Stationen aufzubauen. Aus diesem Grund ist in jede PCD ein Client- und ein Server-Port implementiert.

So empfängt zum Beispiel eine PCD (Client/Server #2) Information von mehreren andern PCDs (Client/Server #6 und Client #4) und wird durch ein SCADA-System oder einen OPC-Server periodisch abgefragt.



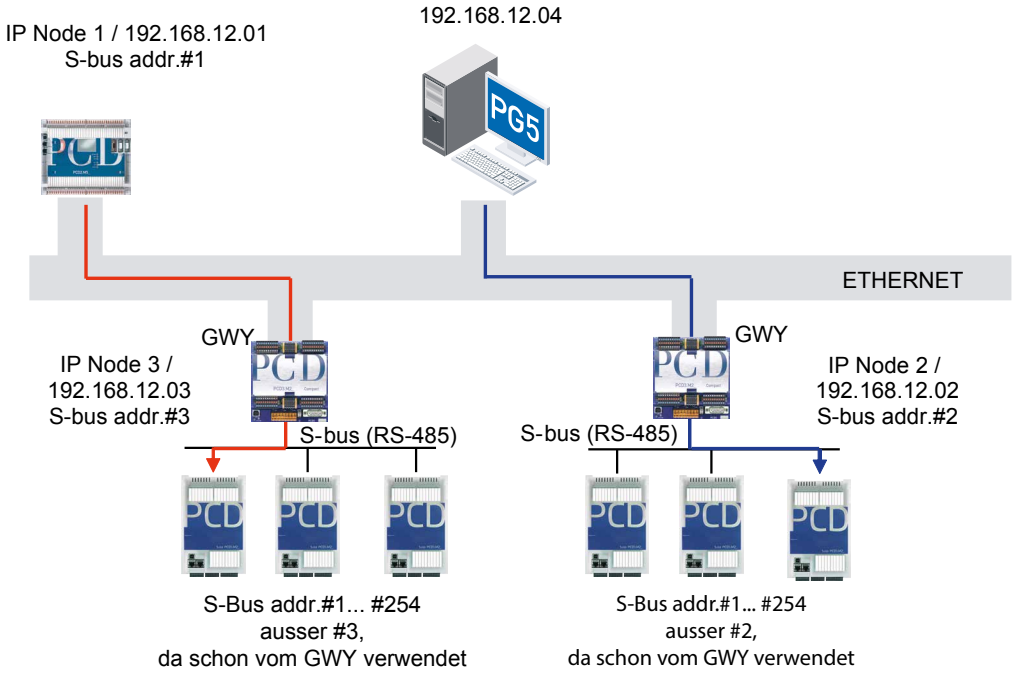
3.2.4 S-Bus-Gateway und S-Bus-Subnetze

Unter einer am Ethernet angeschlossenen Gateway-Station (GWY) kann ein S-Bus-Subnetz aufgebaut werden.

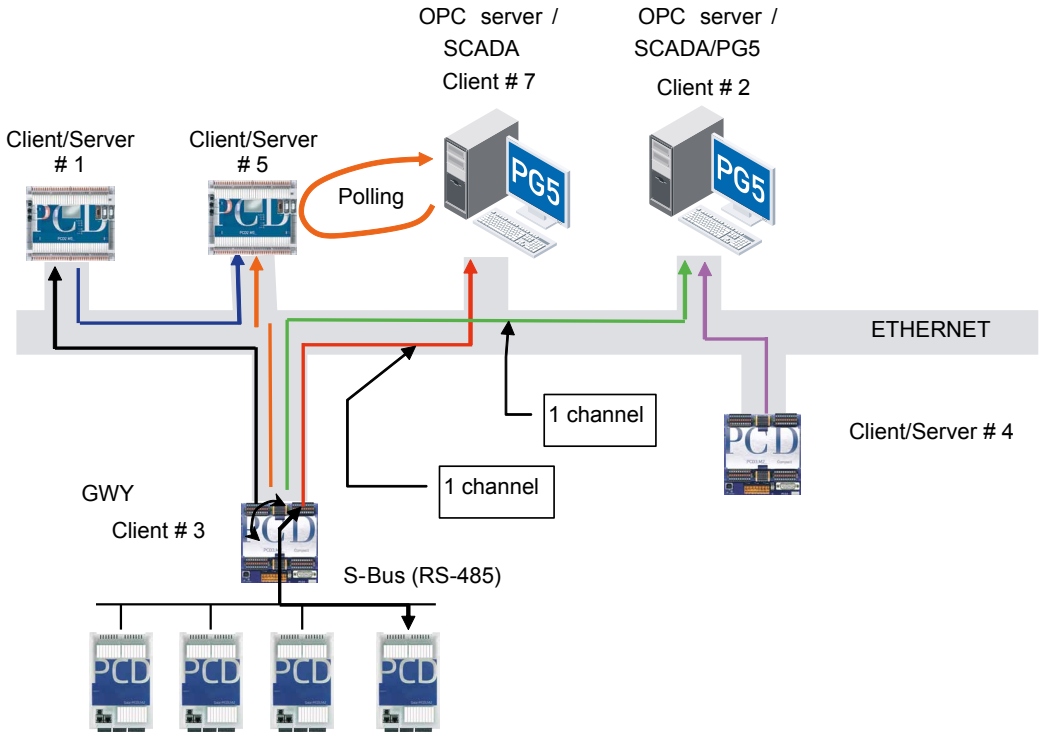
Auf die PCDs in diesem S-Bus-Subnetz wird indirekt über die als Ethernet-Gateway-Station (GWY) konfigurierte PCD zugegriffen. In dieser Eigenschaft leitet sie die vom Ethernet empfangenen Meldungen an das untergeordnete S-Bus-Netz weiter. Diese Gateway-Station ist die einzige zulässige Client-Station im S-Bus Subnetz.

Auf diese einfacheWeise lassen sich mehrere S-Bus-Netze in ein Ethernet integrieren. Es ist daher in einem Ethernet möglich, 65 535 IP-Knoten oder 65 535 x 254 PCDs (254 PCDs in einem S-Bus Subnetz inklusive der als Gateway-Station konfigurierten PCD) zu adressieren.

Eine PCD in einem S-Bus-Subnetz benötigt nur eine S-Bus-Adresse (keine IP-Adresse).



3.2.5 Regeln für die S-Bus-Gateway-Kommunikation



In einer Gateway-Station ist eine begrenzte Anzahl Puffer für die Behandlung ankommender Telegramme reserviert.

PCD1 GWY:	1 Puffer	(= 1 Telegramm; max. 1 Client)
PCD2 GWY:	4 Puffer	(= 4 Telegramme; max. 4 Clients)

Die Telegramme werden sequenziell behandelt.

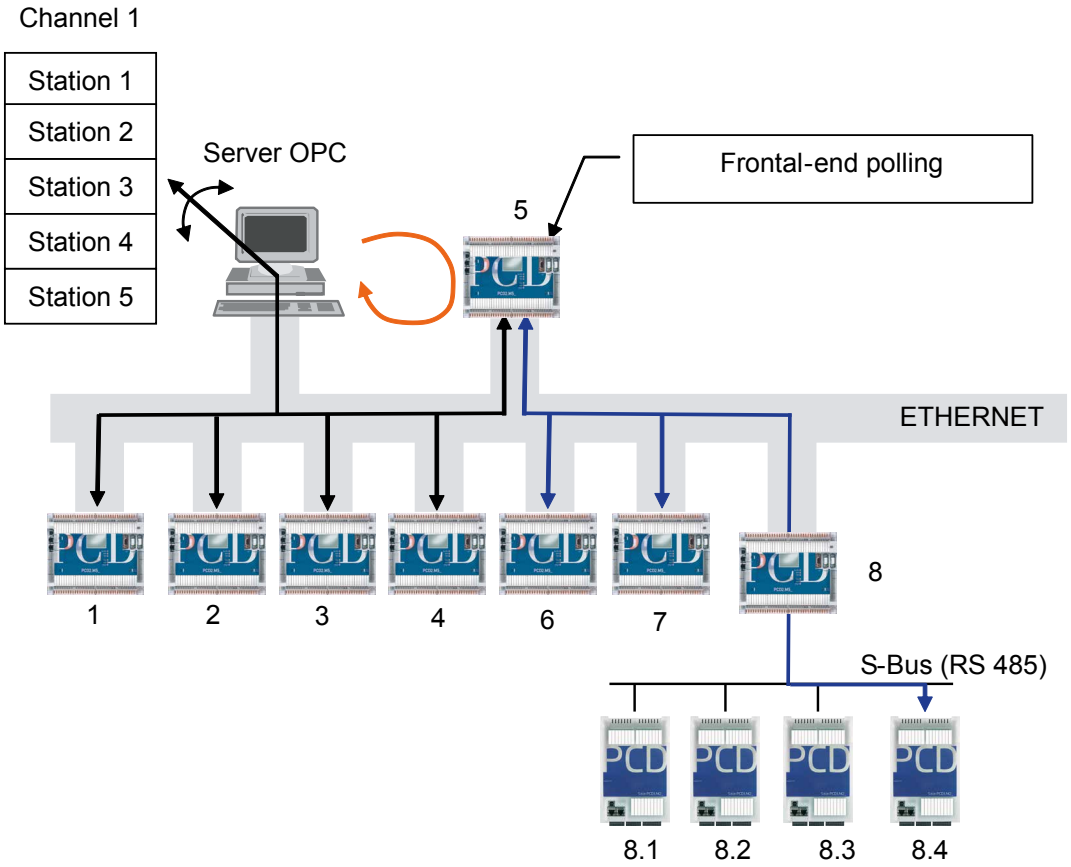
Für die Kommunikation mit den S-Bus-Stationen unter einer Gateway-Station ist daher die Benutzung einer einzigen Kanalverbindung pro OPC-Server oder pro

SCADA-System vorgeschrieben.

Wenn bei einer PCD2 Gateway-PCD mehr als vier Clients gleichzeitig auf PCD-Stationen zugreifen, die unter der Gateway-Station angeschlossen sind, können Telegramme verloren gehen. Für die PCD1 als Gateway gilt diese Restriktion bereits für einen Client.

3.2.6 Regeln für OPC-Server

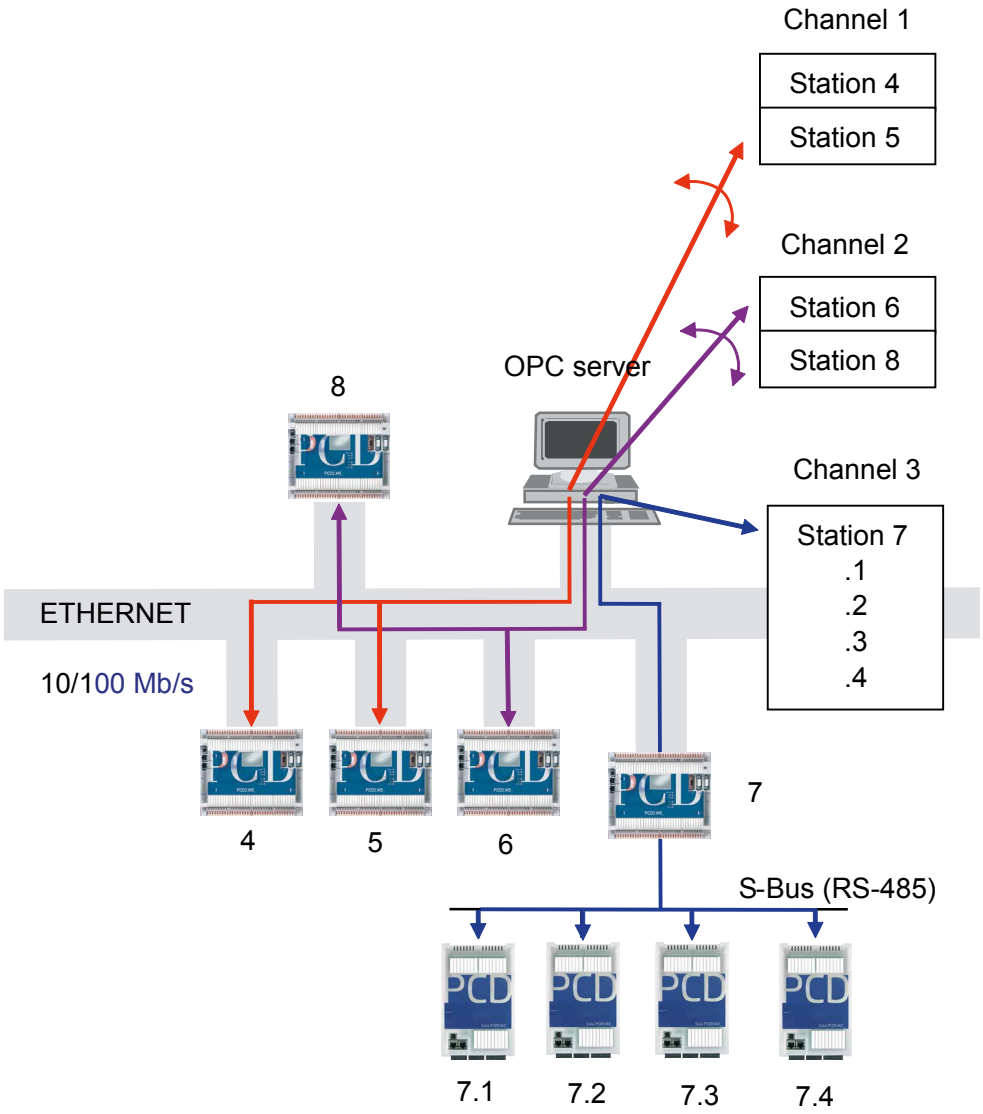
Standard-Lösung mit einem Kanal



Die Standard-Lösung besteht darin, mehrere PCD-Stationen durch einen einzigen Kanal auf dem OPC-Server zu verarbeiten. Die PCD-Stationen 1, 2, 3, 4 und 5 werden eine nach der anderen seriell verbunden.

Es ist auch möglich, eine „Front-End-PCD“ abzufragen, die Daten von mehreren andern PCD-Stationen, z.B. 6, 7 und 8.4 empfängt.

Lösung mit mehreren Kanälen



Wenn die Standard-Lösung nicht möglich ist, weil die Daten der PCD-Stationen schneller verarbeitet werden müssen, können mehrere parallele Kanäle auf dem OPC-Server eingerichtet werden. So können die Daten der PCD-Stationen schneller verarbeitet werden.

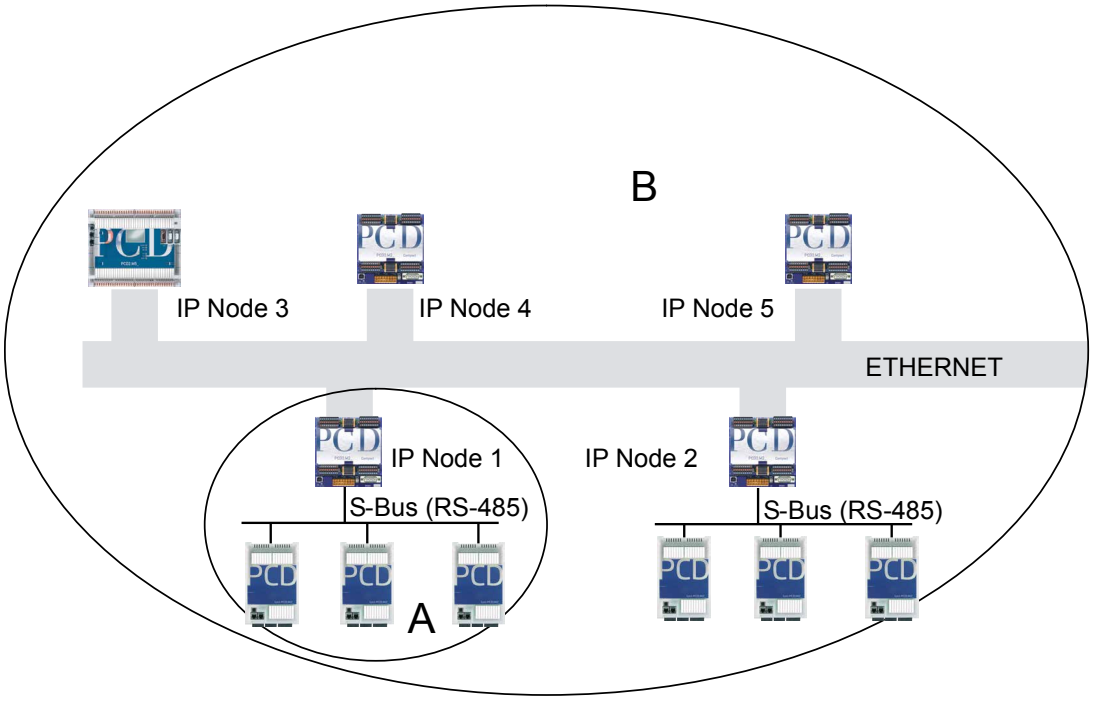
Jeder Kanal bedeutet eine Programm-Task auf dem OPC-Server. Je mehr Tasks, desto höher ist die CPU-Belastung.

PCD1 Stationen in einem S-Bus-Netz unter einer Gateway-Station dürfen nur durch einen Kanal mit dem OPC-Server verbunden sein, weil nur ein 1-Telegramm-Puffer besteht.

3.2.7 Broadcast-Telegramme

Die Übermittlung von Broadcast-Telegrammen, für die Synchronisierung zu einer Client-Station, ist auf zwei verschiedene Arten möglich.

A beschränkt auf das S-Bus-Netz
B auf dem ganzen Ethernet, einschliesslich aller S-Bus-Stationen, die unter einer Gateway-Station adressiert sind.



IP-Knoten 65 535 (0xFFFF) ist für die Übermittlung von Broadcast-Telegrammen via IP reserviert. Er kann in der IL und mit FUPLA FBoxen adressiert werden.

A	Broadcast auf ein S-Bus-Netz beschränkt	Senden an IP-Knoten S-Bus	X<65’535 255
B	Broadcast auf dem ganzen Ethernet, einschliesslich des S-Bus-Netzes unter der Gateway-Station	Senden an IP-Knoten S-Bus	65’535 255
C	Diese Art von Broadcast ist nicht erlaubt	Senden an IP-Knoten S-Bus	65’535 X<255

Client-Seite:

Wenn Broadcast-Telegramme vom Typ C gesendet werden, gibt die Diagnose der Client-Station Fehlermeldungen aus. Das Telegramm wird von der Client-Station nicht gesendet.

- Das NEXE-Flag wird gesetzt
- Die Flags 29 und 30 in der Sende-Diagnose des Diagnose-Registers werden gesetzt
- Die Error-LED leuchtet

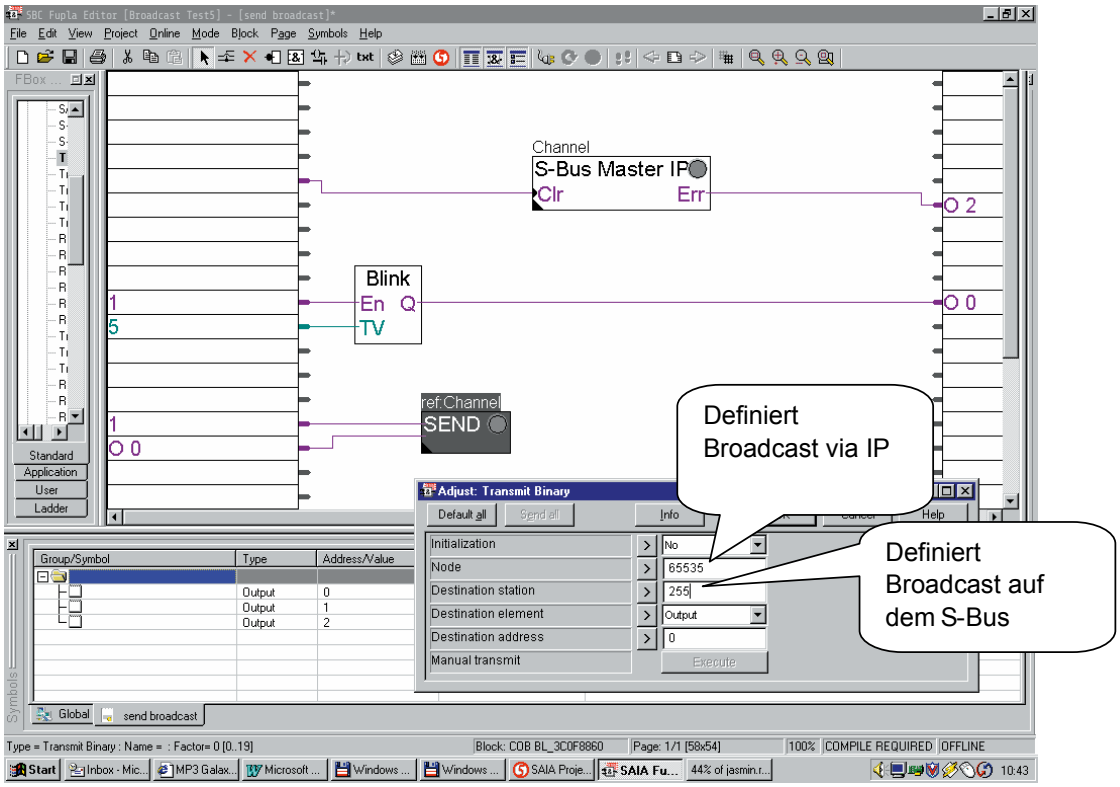
Ausserdem wird Saia PG5® das Senden von Broadcast-Telegrammen vom Typ C nicht zulassen.

Server-Seite:

Auf der Server-Seite werden Broadcast-Telegramme der Typen A und B ohne Beantwortung des Telegramms ausgeführt.

Der Server kann nicht zwischen Standard S-Bus-Telegrammen und Broadcast-Telegrammen vom Typ C unterscheiden. Daher wird die PCD-Server-Station auf diese Art empfangener Telegramme antworten.

Siehe folgendes Beispiel, bei dem “Send Binary“ als Broadcast-Telegramm des Typs B gesendet wird.



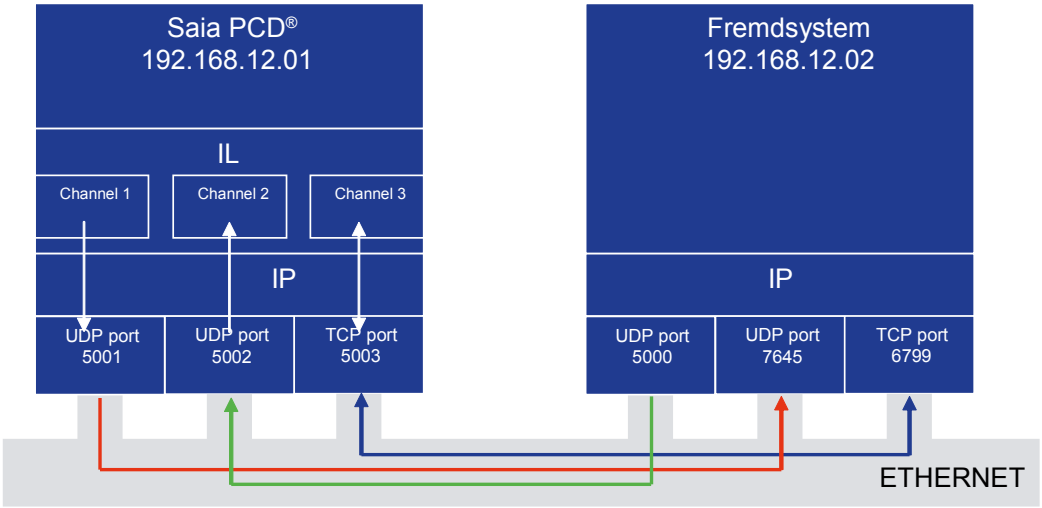
3.3 Open Data Mode via TCP/IP oder UDP/IP

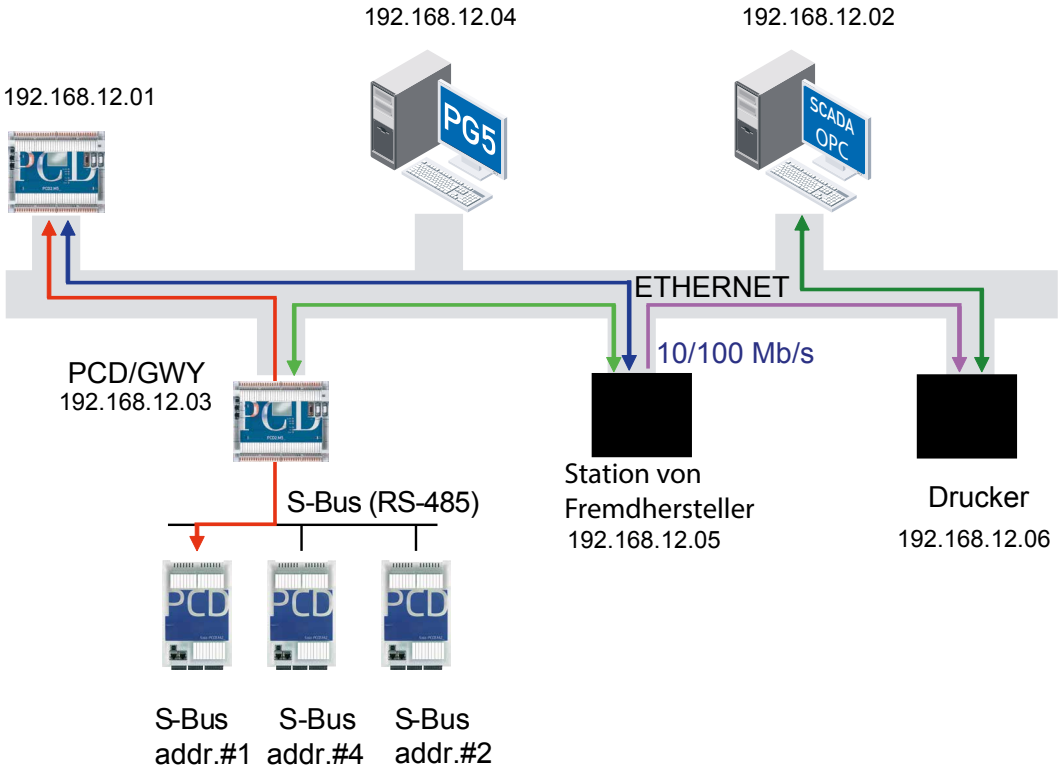
Die Benutzung des Open Data Mode (offener Datenmodus) ermöglicht einer PCD mit einer Station eines Fremdherstellers zu kommunizieren, die das S-Bus-Protokoll nicht unterstützt. Es können aber auch zwei PCDs (falls erforderlich) im Open Data Mode miteinander kommunizieren.

Stationen von Fremdherstellern (z.B. Drucker, andere PLCs usw.) unterstützen das S-Bus-Protokoll nicht; es können daher nur unverarbeitete Daten (Zeichen, Strings) ausgetauscht werden. Eine PCD kann unverarbeitete, transparente TCP/IP- oder UDP/IP-Datenpakete senden. Die Implementierung des Protokolls wird dann durch die Benutzeranwendung praktisch ohne Begrenzungen oder Einschränkungen behandelt. Die Implementierungsregeln der Anwendung werden vollständig dem Benutzer überlassen. Zu den Möglichkeiten gehören die Multimaster-Kommunikation, die ereignisgesteuerte Kommunikation usw.

Der Benutzer kann zwischen der TCP/IP- oder UDP/IP-Kommunikation wählen. Der UDP-Typ arbeitet verbindungslos und der TCP-Typ verbindungsorientiert. Unter TCP/IP wird beim Aufbau einer Verbindung zwischen Client und Server unterschieden.

In diesem Protokolltyp benutzt der Programmierer den SBC API-Socket für den direkten Zugriff auf die UDP- oder TCP/IP-Socket-Schicht. Ein Kommunikationskanal auf der PCD wird mit der IP-Adresse des Geräts und dem Kommunikationsport (UDP-Port / TCP-Port) definiert.




192.168.12.04
192.168.12.02
192.168.12.01
PG5
SCADA
OPC
ETHERNET
10/100 Mb/s
PCD/GWY
192.168.12.03
S-Bus (RS-485)
Station von
Fremdhersteller
192.168.12.05
Drucker
192.168.12.06
S-Bus
addr.#1
S-Bus
addr.#4
S-Bus
addr.#2

3.4 Auslegung und Struktur des Netzes

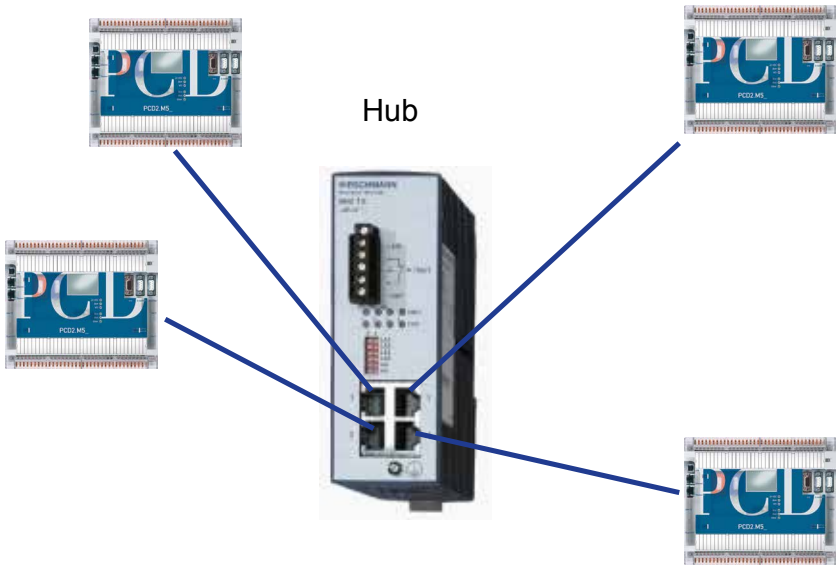
Neben dem PCD7.F65x TCP/IP Ethernet-Modul werden für den Aufbau der Ethernet-Netze weitere Komponenten benötigt.

3.4.1 Hubs (Sternnetz)

Das Schema eines Hub-Netzes gleicht einem Stern - ein zentraler Hub mit Leitungen, die wie Strahlen zu jedem angeschlossenen Host führen. Ein Hub überträgt lediglich das Signal, das er auf der Leitung empfängt. Es gibt keine logische Verbindung. Im Gegensatz zu einem Bus- oder Ringnetz reduziert ein Netz mit Stern-Topologie die Verletzlichkeit des gesamten Netzes auf jede einzelne Unterbrechung in einem Kabel. Wenn die Verbindungsleitung eines einzelnen Hosts zum Netz be-schädigt oder getrennt wird, geht nur der Zugriff auf diesen Host verloren.

Gründe für den Einsatz von Hubs:

- preisgünstige Verbindung zu den Einheiten
- kurze Verzögerungen beim Weiterleiten von Daten
- Mit einem Hub kann mit dem Ethernet-Analyzer der gesamte Datenverkehr gesehen werden. Im Gegensatz zum Switch welcher die Kommunikation kanalisiert.



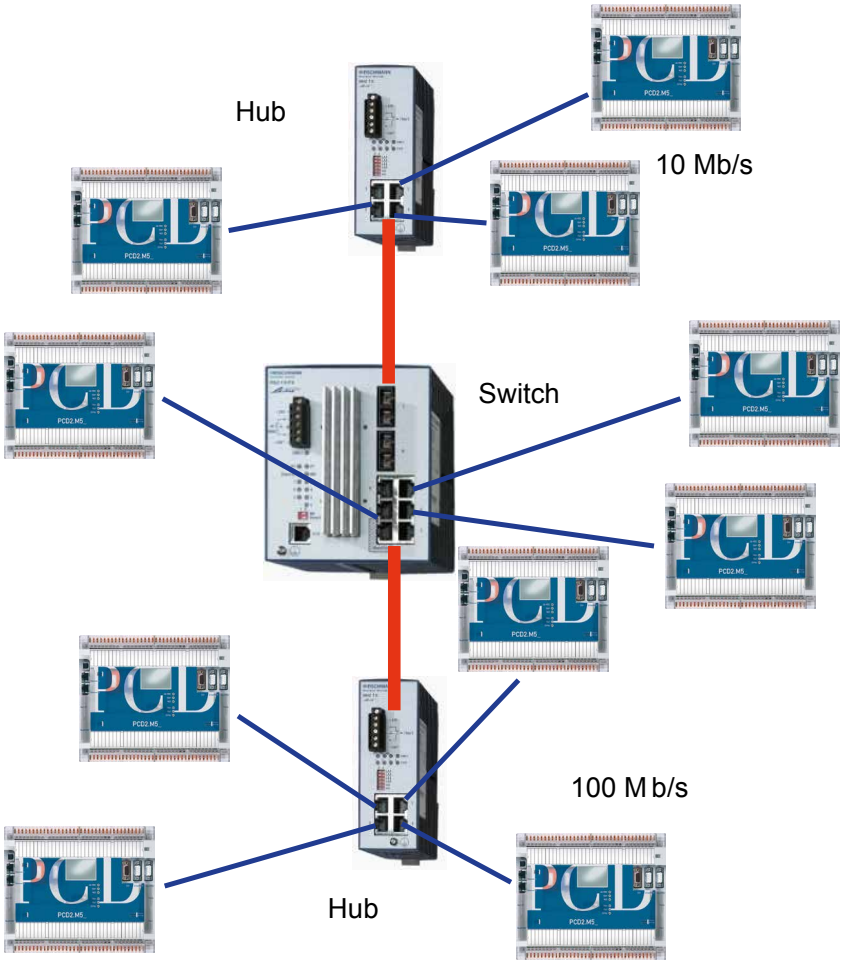
3.4.2 Switches

Switches übertragen oder wiederholen nicht einfach die auf dem einzelnen Kabel gesendeten Signale, sondern empfangen alle Signale, verarbeiten sie und senden sie anschliessend weiter. Das Ergebnis ist, dass nur gültige Ethernet-Rahmen an das andere Netz gesendet werden. Diese Filterung erfolgt auf der Ethernet-Schicht durch den Vergleich der MAC-Adresse. Die Leitweg-Entscheidung wird für jede IP-Adresse nur einmal gefällt und in der Folge wird jeder Rahmen mit der gleichen IP-Adresse gestützt auf diese Entscheidung weitergeleitet.
Datenverkehr mit lokalem Ziel bleibt lokal. Dadurch wird die Anzahl Datenkollisionen minimiert und die Netzleistung optimiert. Es entstehen kollisionsfreie Bereiche.
Zwei Ethernets können mit Switches verbunden werden, die den Verkehr von einem Netz in das andere übertragen.

Ein Switch kann mehrere Telegramme gleichzeitig verarbeiten.

Gründe für den Einsatz von Switches:

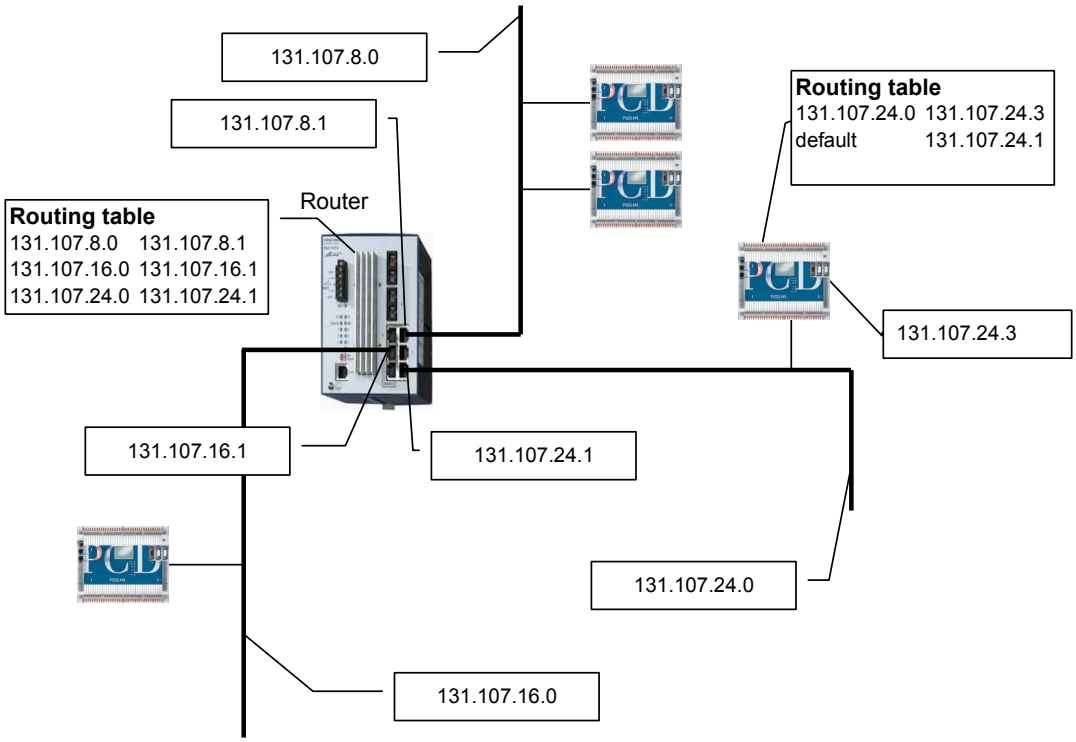
- Schaffung kollisionsfreier Bereiche: Verbesserung des deterministischen Verhaltens des Netzes wegen der geringeren Anzahl Kollisionen
- Kombination unterschiedlicher Verbindungen (10/100 Mbit/s, Halbduplex HDX / Vollduplex FDX)
- effizienterer Datenfluss wegen Punkt-zu-Punkt-Verbindungen und Vollduplex-Betrieb
- verbesserte Netzleistung wegen Filtern (Broadcast- und Multicast-Filtern) und Priorisierungsverfahren.



3.4.3 Router

Ein Router ist eine periphere Verbindungseinheit, die Telegramme von einem physischen Netz in ein anderes überträgt. Der Sende-Host und der Router müssen für das Telegramm ein Übertragungsziel auswählen. Diese Leitweg-Auswahl erfolgt, wenn der IP-Stack die Leitweg-Tabelle konsultiert. Die Leitweg-Tabelle enthält die IP-Adresseinträge und identifiziert die Netzschnittstellen für den Router. Als Vorgabe sendet ein Router nur Telegramme an jene Netze, für die er über eine konfigurierte Schnittstelle verfügt.

- Wenn ein Host mit einem andern Host kommunizieren möchte, entscheidet das IP-Protokoll zuerst, ob der Ziel-Host sich im lokalen oder abgesetzten Netz befindet
- Wenn sich der Ziel-Host im abgesetzten Netz befindet, sucht das IP-Protokoll in seiner Leitweg-Tabelle nach einem Leitweg zum abgesetzten Netz
- Wenn kein expliziter Leitweg gefunden wird, wird die Vorgabeadresse des Routers benutzt
- Die Leitweg-Tabelle auf dem Router wird konsultiert, um das Telegramm an das spezifizierte Netz weiterzuleiten.



3.4.4 Netzkomponenten

Saia Burgess Controls liefert keine spezifischen Ethernet-Netzkomponenten. Gute Erfahrungen wurden mit den Netzkomponenten der Ethernet Rail Family von Hirsch mann gemacht.

Die Industrial Line ETHERNET Rail Family wurde speziell für die Benutzung in industriellen Automationsanwendungen entwickelt. Sie unterstützt Redundanz-Funktionen (alternativer Datenpfad durch HIPER-Ring-Strukturen, redundante Kopplung von Netzsegmenten) und garantiert einen hohen Grad an Netzverfügbarkeit.

Hirschmann stellt auch Unterstützung für die Netzentwicklung in Kundenprojekten zur Verfügung: www.hirschmann.com

3.4.5 Steigerung der Leistungsfähigkeit

Vollduplex, Autonegotiation, Autosensing

Das Ethernet TCP/IP-Modul PCD7.F65x unterstützt die Betriebsarten „Vollduplex“ und „Halbduplex“. Duplex ist keine Netz-Topologie, sondern eine Methode für den Datenaustausch zwischen zwei Nodes.

Das Ethernet TCP/IP-Modul PCD7.F65x benutzt Autonegotiation und Autosensing, um zwischen zwei Nodes einen kompatiblen Modus zu installieren. Sobald zwei Nodes miteinander verbunden sind, wird nach der folgenden Liste der bevorzugte Datenkommunikations-Modus installiert:

- 100Base TX und Vollduplex-Modus
- 100Base TX und Halbduplex-Modus
- 10Base T und Vollduplex-Modus
- 10Base T und Halbduplex-Modus

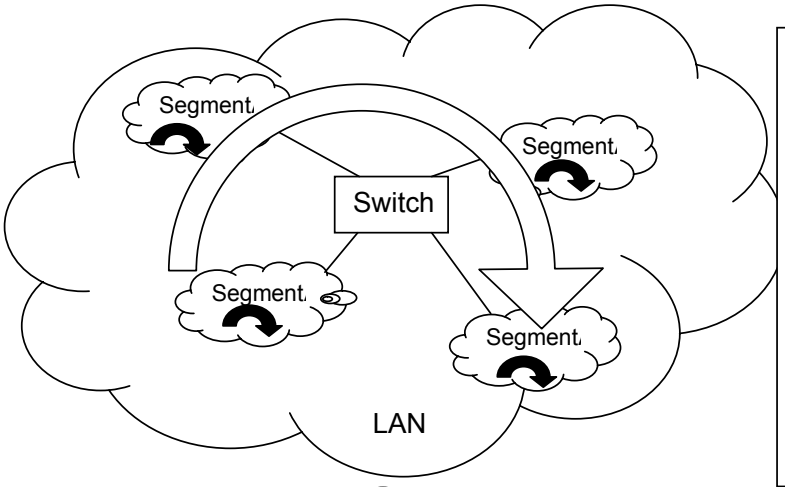
Wenn von der PCD aus mit der Partnerstation keine Autonegotiation möglich ist, wird der Modus 10Base T und Halbduplex-Modus installiert.

Autosensing ist das Leistungsmerkmal, das die Datenrate eines Signals (10 Mbit/s oder 100 Mbit/s) erkennt. Autosensing ist auch ohne Autonegotiation möglich.

Geschaltetes LAN im Vergleich zum gemeinsam benutzten LAN

Es ist auch möglich, das Netz in unterschiedliche Gruppen aufzuteilen, um ein isoliertes Ethernet zu erreichen, ohne den gesamten Netzverkehr unnötig zu belasten. Aus diesem Grund sollte die Technologie mit Standard-Switches benutzt werden, um kollisionsfreie Bereiche zu schaffen.

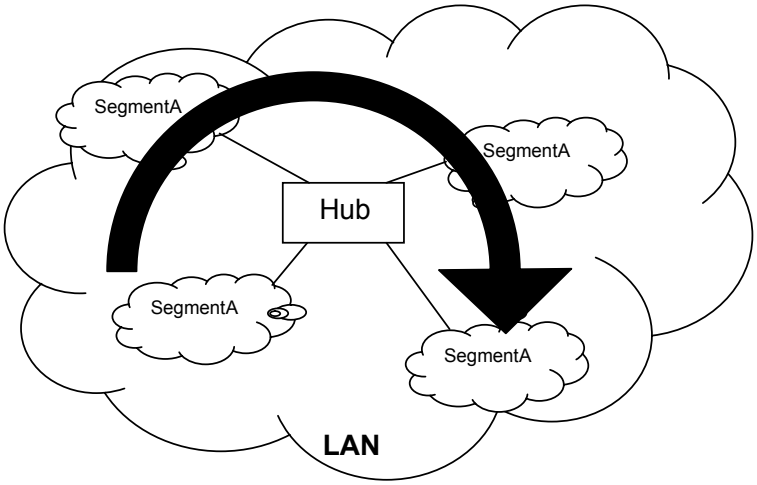
Geschaltetes LAN



- Jedes Segment verfügt über die volle Leistungsfähigkeit / Datenrate
- Gleichzeitiger Datenverkehr in mehreren Segmenten
- Lokaler Datenverkehr bleibt lokal. Nur ausgewählte Telegramme verlassen das lokale Segment

Hohes Verkehrsaufkommen
Niedriges Verkehrsaufkommen

Gemeinsam genutztes LAN



- Alle Segmente nutzen die Leistungsfähigkeit / Datenrate gemeinsam
- Alle Telegramme durchlaufen alle Segmente.
- Es befindet sich immer nur ein Telegramm im Netz
- Kollisionen reduzieren die Effizienz auf 40%.

Hohes Verkehrsaufkommen

Maximierung der Nutzdaten (Payload)

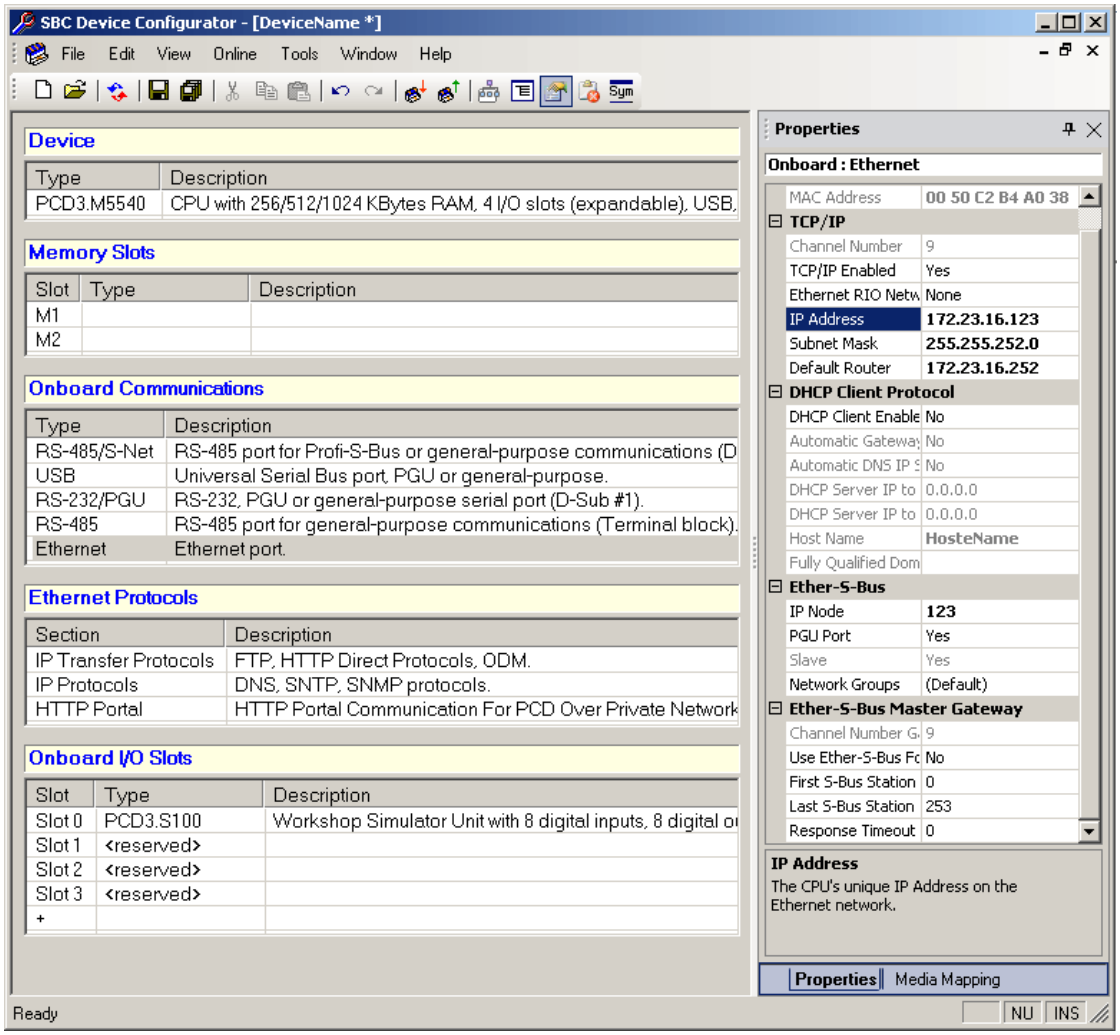
Es ist zu beachten, dass sowohl TCP/IP als auch UDP/IP (aber insbesondere TCP/IP) einen ziemlich grossen Overhead aufweisen. Diese Overheads fallen bei kleinen Nutzdaten in Telegrammen besonders ins Gewicht. Es wird daher empfohlen, grosse Nutzdatenmengen zusammenzufassen und zu senden (d.h. 32 Register auf dem S-Bus via UDP/IP), um von der Leistung des Ethernet profitieren zu können.

4 Konfiguration und Programmierung

4.1 Konfiguration und Adressierung

4.1.1 Konfiguration des S-Bus-IP-Port (Server)

Das IP-Modul wird in Saia PG5® → Hardware Settings konfiguriert:

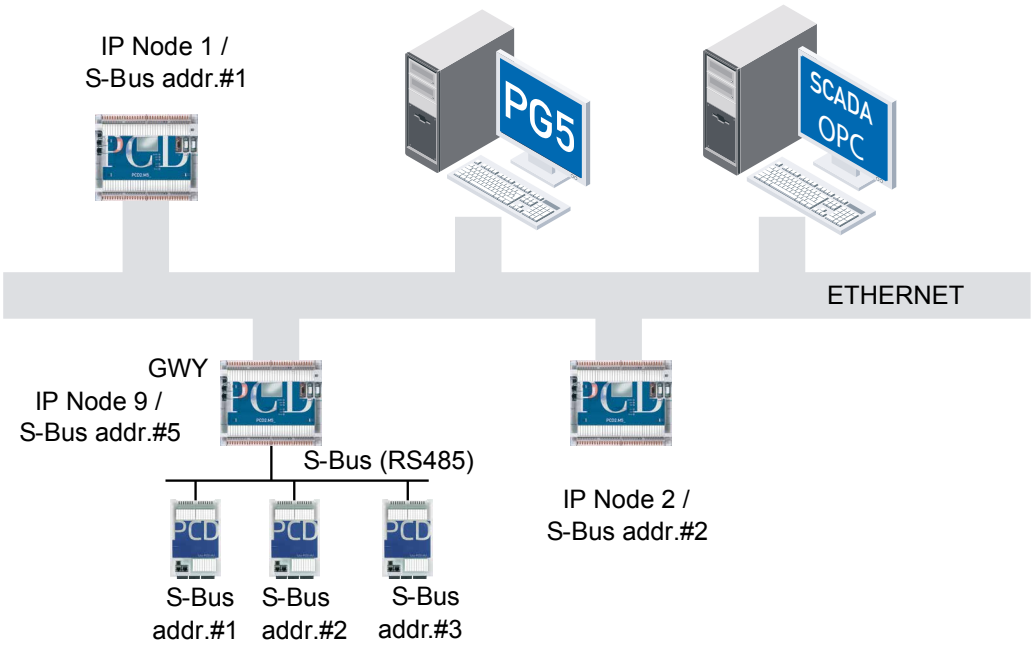


Weitere Einzelheiten über die Hardware-Konfigurationseinstellungen sind in der PG5 → Hilfe zu finden.

4.1.2 Adressierung der IP-Server-Station

Im Prinzip ist jedes IP-Modul durch eine weltweit eindeutige Ethernet-Adresse (MAC) und eine standortbezogene IP-Adresse definiert. Die Umsetzung (MAC-Adresse ↔ IP-Adresse) erfolgt automatisch durch das ARP (Address Resolution Protocol) im TCP/IP stack.

PG5 generiert eine IP-Settings-Tabelle (DBX), um die IP-Adressen, IP-Knoten und S-Bus Adressen mit der gesamten CPU-Hardwareinformation des Projekts zu verbinden. Der Benutzer benötigt zur Programmierung nur die S-Bus-Adressen und IP-Knoten. Die DBX wird während der Projekterzeugung berechnet und mit Hilfe des Benutzerprogramms in die PCD geladen. Wenn eine IP-Adresse, S-Bus-Adresse oder ein IP-Node im Projekt geändert wird, muss das Programm sämtlicher betroffener OPMs wiederhergestellt (Generierung eines neuen DBx) und in die PCD geladen werden. Weitere Informationen sind in der PG5-Hilfe zu finden.
Eine Word-codierte Adressierung für IP-Knoten und eine Byte-codierte S-Bus-Adresse bieten die geeigneten Adressierungsmöglichkeiten im Ethernet. Sie ermöglichen die Adressierung von 65 536 IP-Knoten oder 65 536 x 253 PCDs (253 PCDs in einem S-Bus-Netz unter einem Ethernet-S-Bus-Gateway, siehe Abbildung). PG5 generiert eine DBx mit der Beziehung zwischen IP-Knoten und IP-Adresse.



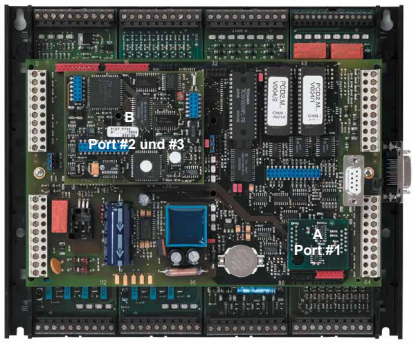
Für Stationen in einem S-Bus-Netz unter dem Gateway im IP-Node #9 darf die S-Bus-Adresse #5nicht mehr benutzt werden, da bereits vom Gateway verwendet.
Umsetzungstabelle IP-Adresse - IP-Knoten - S-Bus-Adresse

IP-Adresse ↔...	...IP-Node	...S-Bus-Adresse
192.168.2.151	2	2
192.168.2.155	3	9
192.168.2.168	6	--
192.168.2.201	7	--
192.168.2.105	9	1...254 exkl. 5

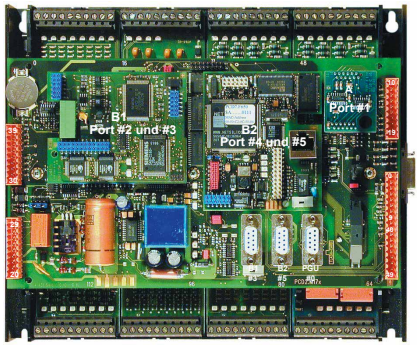
4.1.3 Steckplätze und Kanalnummern

Die Steckplätze/Kanäle für die PCD7.F65x sind auf der Saia PCD® wie folgt definiert (siehe auch Kapitel 2):

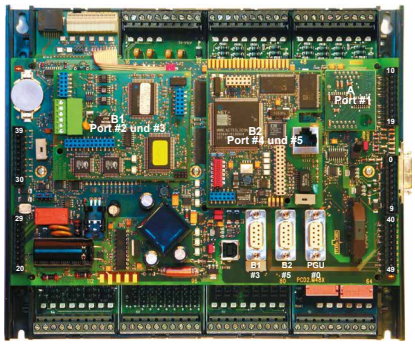
PCD1.M130
Kanal 9
Steckplatz B



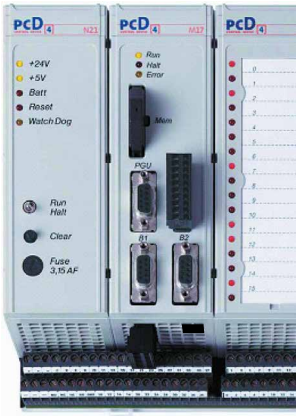
PCD2.M15x
Kanal 9
Steckplatz B



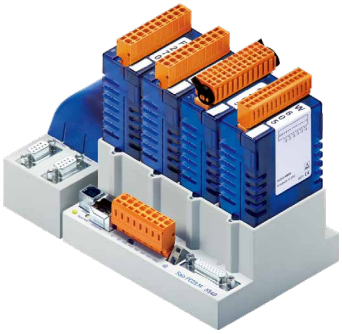
PCD2.M17x
Kanal 8
Steckplatz B2



PCD2.M480
Kanal 9
Steckplatz B1
und zusätzlich auch
Kanal 8



PCD4.M17x
Kanal 9
Steckplatz B1



Ohne PCD7.F65x
ethernetfähig
PCD3.M5540
PCD3.M3120
PCD3.M3330
PCD3.M6340
PCD3.M6540
Kanal 9

4.2 Programmierung des S-Bus via Ethernet

Das Ethernet-S-Bus-Protokoll wird für die Kommunikation zwischen zwei PCDs, einer PCD und einem Programmierwerkzeug (PGX) oder einer Fremdeinheit (PLS mit scom.dll oder PLC mit S-Bus-Unterstützung) benutzt.
Die neue Master-Master-Funktionalität in einem Netz ist eine Verbesserung des Standard-S-Bus-Protokolls. Der Standard-S-Bus erlaubt nur einen Master pro Netz. Im S-Bus via Ethernet können alle Stationen als Masters arbeiten.
Diese Art Kommunikation wird mit den bekannten STXM/SRXM-Befehlen gesteuert. Die Syntax ist den Standard S-Bus-Telegrammen ähnlich.

Eigenschaften	
Ablauf	Befehlsanforderung Anforderung (Client) ● Anzeige (Server) ● Antwort (Server) ● Bestätigung (Client)
Kommunikations-sicherheit	Maximal 3 wiederholte Versuche im Hintergrund. Es wird die S-Bus-CRC-16-Fehlerprüfung angewendet. Via IP wird keine spezielle sichere Schicht benutzt.
Protokoll via IP	Für die Kommunikation auf dem S-Bus via Ethernet wird das UDP-Protokoll verwendet. Der Empfangs-Kommunikations-Socket ist offen und permanent mit Port 5050 verknüpft. Der Ausgangs-Kommunikations-Socket kann entweder mit Port 5050 verknüpft sein (default) oder einen, im System freien Port benutzen

4.2.1 Beschreibung der Saia PCD® Befehle

Mit dem IP werden die folgenden Instruktionen unterstützt:

SASI(I)	Serielle Schnittstelle zuweisen
STXM(X)	Serielles Medium für Daten senden
SRXM(X)	Serielles Medium für Daten empfangen

SASI-Initialisierung

Zusätzlich zur Konfiguration im Offline-Hardware-Konfigurator wurde ein SASI für Client und Slave zur Verfügung gestellt. Im Interesse der Aufwärtskompatibilität bleibt die Anzahl der SASI-Parameter unverändert. Weil der S-Bus via Ethernet einen festen, vordefinierten Eingangs-Kommunikationsport benutzt (5050), muss kein spezieller Kommunikationsport definiert werden. Als Ausgangsport wird für die Kommunikation ein vom System freier Port zwischen 1024 und 4999 gewählt. Der Ausgangsport kann aber auch im SASI-Text, wie unten beschrieben, mit dem Port 5050 verknüpft werden.

```
Verwendung:     SASI  Channel     ;Steckplatz (8 oder 9)
                      Text nb     ;Definitionstext-Nr. 0...7999

Beispiel:       SASI  8           ;Steckplatz 8 initialisieren
                      Text IP     ;Definitionstext für SASI
```

Definition von SASI-Text:

Master	“<mode_def>,<dest_reg>; <diag_def>; <option_def>“
Beispiel	“MODE:EM,R100; DIAG:F1000,R1000; TOUT:500,PORT:5050
Slave	“<mode_def>,<dest_reg>; <diag_def>;
Beispiel	“MODE:ES; DIAG:F1000,R1000;

”<mode_def>,<dest_reg>; Definiert den Modus
Im Ethernet-Master-Modus wird die Adresse der abgesetzten Station dem EM angefügt.

Master	“MODE:EM,<dest_reg>;“	EM: Ethernet-Master-Modus. <dest_reg>: Dieses Register definiert die Adresse der abgesetzten Station.

Slave	“MODE:ES;	ES: Ethernet-Slave-Modus (nur für die Slave-Diagnose verwendet). Hinweis: der Ethernet-Slave wird automatisch konfiguriert, wenn das TCP/IP Modul in den PG5 “Hardware Settings“ konfiguriert wird.

<dest_reg>: Stellt das Register dar, das die Partner-Station definiert. (Typ: Rxxxx)

Bevor STXM/SRXM-Befehle gesendet werden können, muss wie beim Standard-S-Bus die Zieladresse in einem Adressregister gespeichert werden. Diese Adressierung benutzt zwei Adressfelder.

Register-Adressfeld (32 Bit)		
MS Word	**LS Word**	
IP-Knotennummer der Ziel-PCD	Nicht benutzt	S-Bus-Adresse des Ziel-Slaves

<diag_def>; Definiert die Diagnoseelemente für die Ethernet-Kommunikation.
Format: “DIAG:<diag_flag>,<diag_reg>;“

Typ		**Beschreibung**
<diag_flag>	Fxxxx Oxxxx	Basisadresse von 8 aufeinanderfolgenden Flags oder Ausgängen
<diag_flag>	Rxxxx	Basisadresse des Diagnose-Registers

<option_def>“ Definiert die Optionselemente für die Ethernet-Kommunikation.
Format: “[<tout_def>],[<port_def>];“

Typ		**Beschreibung**
<tout_def>	TOUT: xxx	Stellt den Timeout-Wert des EM-Modus in ms dar. Der Default- Timeout-Wert ist 500 ms. (Unterer Grenzwert = 100 ms)
<port_def>	PORT: xxx	Wahl des Ausgang-Ports für die S-Bus UDP Kommunikation Default: xxx = 0 0 = ein im System freier Port zwischen 1024 und 4999 wird gewählt xxx = 5000 bis 65‘535, bedeutet feste Zuweisung durch den Benutzer

Flags: Wenn der Definitionstext fehlt oder ungültig ist oder die Firmware keine IP-Unterstützung bietet, wird das Error (E) Flag gesetzt.

Diagnose

Diagnose-Flags

Adresse	Name	Beschreibung
xxxx	RBSY	Receiver busy
xxxx+1	RFUL	Receiver buffer full
xxxx+2	RDIA	Receiver diagnostic
xxxx+3	TBSY	Transmitter busy
xxxx+4	TFUL	Transmitter full
xxxx+5	TDIA	Transmitter diagnostic
xxxx+6	XBSY	Physical link
xxxx+7	NEXE	Not executed

Transmitter Busy (TBSY) ↑
Zeigt an, dass eine Übertragung in Gang ist.

Bedeutung für Master-Station: Ist während der Ausführung einer STXM- oder SRXM-Instruktion auf “H“ (high) gesetzt Das Flag wird zurückgesetzt, sobald eine gültige Antwort empfangen wurde.

Slave-Station: Ist auf “H“ gesetzt, während die Antwort gesendet wird.

Transmitter Diagnostic (TDIA) ↑
Zeigt an, dass während der Telegrammübermittlung ein Fehler erkannt wurde. Eine ausführliche Beschreibung des Fehlers ist im Diagnose-Register (Bits 16.bis.31) zu finden. Das Flag wird zurückgesetzt, sobald alle Sender-Diagnosebits (16.bis.31) im Diagnose-Register zurückgesetzt wurden.

Not Executed (NEXE) ↑
Zeigt an, dass eine Instruktion (STXM oder SRXM) nach drei Versuchen nicht ausgeführt wurde. Das Flag wird durch die nächste S-Bus-Instruktion zurückgesetzt.

Physical Link (XBSY) ↑
Zeigt an, ob ein physikalischer Link vorhanden ist oder nicht. Ist auf “H“ (high) gesetzt, wenn das Ethernet TCP/IP Modul einen physikalischen Link detektiert und ist auf “L“ (low) gesetzt, wenn der physikalische Link verloren geht. Sobald der SASI (Master oder Slave) Befehl ausgeführt wird, ist dieses Flag assigniert und kann als Diagnose be nutzt werden. Schickt die PCD Telegramme ohne etablierten physikalischen Link, so wird das Bit 23 im Diagnose-Register gesetzt.

Diagnose-Register

	Bit	Bezeichnung	Beschreibung
EMPFÄNGER	0	Overrun error	Überlauf im internen Empfangspuffer
	1	Nicht benutzt	
	2	Framing error	Wird normalerwise durch eine nicht korrekte Baudrate verursacht
	3	Break error	Unterbrechung in der Datenleitung *)
	4	BCC error	Bad Block Check Code oder CRC-16
	5	Nicht benutzt	
	6	Nicht benutzt	
	7	Nicht benutzt	
	8	Length error	Die Telegrammlänge ist ungültig
	9	Nicht benutzt	
	10	Inv tlg	Ungültiges oder unzulässiges Telegramm
	11	Nicht benutzt	
	12	Range error	Ungültige Elementadresse
	13	Value error	Fehler im empfangenen Wert
	14	Missing Media	Media ungültig oder inexistent
	15	Nicht benutzt	
SENDER	16	Retry count	Zeigt die Anzahl (binär) wiederholter Versuche an. (Telegrammwiederholungen in binärer Darstellung)
	17		
	18		
	19		
	20	NAK response	Negative Antwort (NAK) erhalten
	21	Missing response	Keine Antwort nach dem Timeout erhalten
	22	Inv. response	ACK/NAK statt Daten empfangen, oder Daten statt ACK/NAK
	23	Target not present	Zielstation nicht präsent oder physikalischen Link verloren
	24	Nicht benutzt	
	25	Nicht benutzt	
	26	Nicht benutzt	
	27	Nicht benutzt	
	28	Range error	Ungültige Elementadresse
	29	TxNode_error	Knoten ist nicht vorhanden
	30	TxBroadcast_error	Fehler beim Senden eines Broadcast-Telegramms über IP
	31	Program error	Unzulässiger Sendeversuch

*) Hat im SM0/SS0-Modus keine Bedeutung

Jedes Bit, das im Diagnose-Register “H“ (high) wurde, bleibt gesetzt, bis es durch das Anwenderprogramm oder den Debugger manuell zurückgesetzt wird.

Overrun Error (Bit 0)

Wird auf “H“ gesetzt, wenn im internen DUART-Puffer ein Überlauf auftritt.

Ursache:
Die zugewiesene Übertragungsrate ist zu hoch

→
Die CPU kann nicht mehr alle empfangenen Zeichen verarbeiten. Dies kann eintreten, wenn eine CPU an Kommunikatonsverbindungen beteiligt ist, die eine hohe Datenübertragungsrate über mehrere Schnittstellen geleichzeitig erfordern. Es ist theoretisch für alle Schnittstellen einer CPU (mit Ausnahme der 20-mA-Stromschleife) möglich, gleichzeitig die maximale Übertragungsrate von 19 200 Bit/s zugewiesen zu bekommen. In der Praxis kann dieser Fehler jedoch auftreten, wenn über mehrere Schnittstellen ein hohes Kommunikationsaufkommen vorhanden ist. Das Systemprogramm behandelt die Schnittstellen mit unterschiedlichen Prioritäten.

Die höchste Priorität wird der Schnittstelle 0 zugeordnet und nimmt hin zu Schnittstelle 3 ab.

Abhilfe:
- reduzieren
- Für die schnelle Kommunikation wenn möglich eine Schnittstelle hoher Priorität benutzen.

Framing Error (Bit 2)

Wird gesetzt, wenn ein Zeichen mit einem Rahmenbildungsfehler empfangen wird (fehlendes Stoppbit). Dies wird normalerweise durch eine falsch gesetzte Übertragungsrate verursacht.

Break Error (Bit 3)

Wird auf “H“ gesetzt, wenn während des Empfangs eines Zeichens eine Unterbrechung auftritt.

Ursache: Datenleitung unterbrochen oder Übertragunsrate falsch gesetzt.

BCC or CRC-16 Error (Bit 4)

Wird auf “H“ gesetzt, wenn ein CRC-16-Fehler im ankommenden Telegramm festgestellt wird. Das ankommende Telegramm wird zurückgewiesen.

Reaktion des Slave: Das empfangene Telegramm wird ignoriert

Reaktion des Masters: Das empfangene Telegramm wird ignoriert und das letzte Telegramm wird nochmals gesendet.

Ursache: Störung auf der Datenleitung.

Abhilfe: Elektrische Installation überprüfen.

Length Error (Bit 8)

Wird auf “H“ gesetzt, wenn ein Telegramm mit ungültiger Länge empfangen wird. Dieser Fehler kann in einem Netz aus ausschliesslich PCD-Stationen (keine Stationen fremder Hersteller) nicht auftreten. Der Fehler zeigt an, dass ein ungültiges Telegramm von einem externen System empfangen wurde. Dies führt zu einer NAK-Antwort.

Address Error (Bit 10)

Wird auf “H“ gesetzt, wenn ein ungültiges Telegramm empfangen wird (falscher Befehlscode).

Ursache: Dieselbe wie für Length Error (es erfolgt keine NAK-Antwort).

Range Error (Bit 12)

Wird auf “H“ gesetzt, wenn ein ankommendes Telegramm eine ungültige PCD-Elementadresse enthält. Dieser Fehler kann nicht in einem Netz auftreten, das ausschliesslich aus PCD-Stationen besteht, da die Master-PCD den Bereich der Elementadressen beim Senden überwacht. Die Slave-Station antwortet auf diesen Fehler mit NAK.

Value Error (Bit 13)

Wird beim Empfang eines ungültigen Datenwertes auf “H“ gesetzt.

Beispiel: Die STXM-Instruktion wird benutzt, um die Uhr zu laden. Der für die Stunde empfangene Wert ist 30. Der zulässige Bereich für die Stunde ist jedoch nur 0...23. Die Slave-Station antwortet auf diesen Fehler mit NAK.

RxBroadcast Error (Bit 14)

Wird gesetzt, wenn ein ungültiges Broadcast-Tele-

gramm empfangen wird (IP-Broadcast → IP-Node = 65 535 und S-Bus-Adresse < 255).

Retry Count (Bits 16 bis 19)
Zeigt die Anzahl wiederholter Telegramme als Binär wert an, die während der Ausführung einer SRXM- oder STXM- Instruktion gesendet wurden. Bit 16 stellt dabei das LS-Bit (niederwertigstes Bit) dar. Die Qualität eines S-Bus-Netzes kann durch die Überwachung dieser beiden Bits beurteilt werden.

Negative Response (Bit 20)
Wird auf “H“ gesetzt, wenn eine NAK-Antwort von einem Slave empfangen wird. Das bedeutet, dass der Master zuvor ein ungültiges Telegramm gesendet hat. Die folgenden Fehler überprüfen: Value Error, Range Error and Length Error.

Missing Response (Bit 21)
Wird auf “H“ gesetzt, wenn von der Slave-Station nach Ablauf des Timeout keine Antwort empfangen wurde. In diesem Fall wird das Telegramm erneut gesendet (maximal zweimal).

Mögliche Ursachen:
Die adressierte Slave-Station ist nicht vorhanden.
Installationsfehler im Netz (Verdrahtung).
Die Slave-Station hat ein unverständliches Telegramm mit einem CRC-16-Fehler empfangen.

Abhilfen:
Slave-Station überprüfen (Verbindungen, Stationsnummer) Wurde der korrekte Leitungsabschluss und wurden die Pull-up/down-Widerstände auf der Busleitung an der ersten und letzten Station angeschlossen?

Multiple NAK (Bit 22)
Wird auf “H“ gesetzt, wenn anstelle der erwarteten ACK oder NAK eine andere Antwort von einer Slave-Station empfangen wird.

Mögliche Ursachen:
- Mehr als ein Slave mit derselben Stations-Nummer.
- Mehr als ein Master im Netz.
- Störung auf der Busleitung.

Abhilfen:
Wie für den Fehler Missing Response

Target not present (Bit 23)
Wird auf “H“ gesetzt, wenn die Zielstation auf dem Netz nicht angesprochen werden kann. Entweder nach drei Versuchen oder wenn der physikalische Link nicht besteht.

Mögliche Ursachen:
Defektes Verbindungskabel oder Stromversorgung der Zielstation unterbrochen

Range Error (Bit 28)
Wird auf “H“ gesetzt, wenn die SRXM- oder STXM-Instruktionen eine Element-Adresse (Ursprungs- oder Zieladresse) ausserhalb des erlaubten Bereichs an zeigen.

Mögliche Ursache:
Fehler im Anwenderprogramm
Überwachte Bereiche:

Eingänge/Ausgänge	0...8191
Flags	0...8191
Timer/Zähler	0...1599
Register	0...4095

Beispiel:

Während der Ausführung der folgenden STXM-In struktion wird das Bit für Range Error auf “H“ gesetzt.

```
STXM    1       ; Kanal 1
        25      ; 25 Register
        R 1000  ; Ursprung der Basisadresse
        R 4072  ; Ziel der Basisadresse
```

Es wird versucht, die Inhalte der Register 1000 bis 1024 in der Master-Station an die Register 4072 bis 4096 in der Slave-Station zu senden.

TxNode Error (Bit 29)

Wird auf “H“ gesetzt, wenn der Knoten in der Knoten-Liste nicht vorhanden ist, wenn er nicht konfiguriert oder wenn ein ungültiges Broadcast-Telegramm gesendet wurde (IP- Broadcast → IP-Knoten = 65‘535 und S-Bus-Adresse < 255).

TxBroadcast Error (Bit 30)

Wird auf “H“gesetzt, wenn ein ungültiges Broadcast-Telegramm gesendet wurde (IP-Broadcast → IP-Knoten = 65‘535 und S-Bus-Adresse < 255).

Program Error (Bit 31)

Wird während der Ausführung einer STXM- oder SRXM-Instruktion auf “H“ gesetzt, wenn die Schnittstelle im SS1-Modus zugewiesen wurde oder eine ähnliche Instruktion bereits ausgeführt wird (TBSY-Flag wurde vor der Ausführung der Instruktion nicht abgefragt).

STXM/SRXM-Daten an die/von der Slave-Station senden/empfangen

Die Instruktionen sind dieselben wie auf dem S-Bus. Der einzige Unterschied ist die Kanalnummer (8 oder 9) und die IP-Knoten-Adresse der Partner-Station.

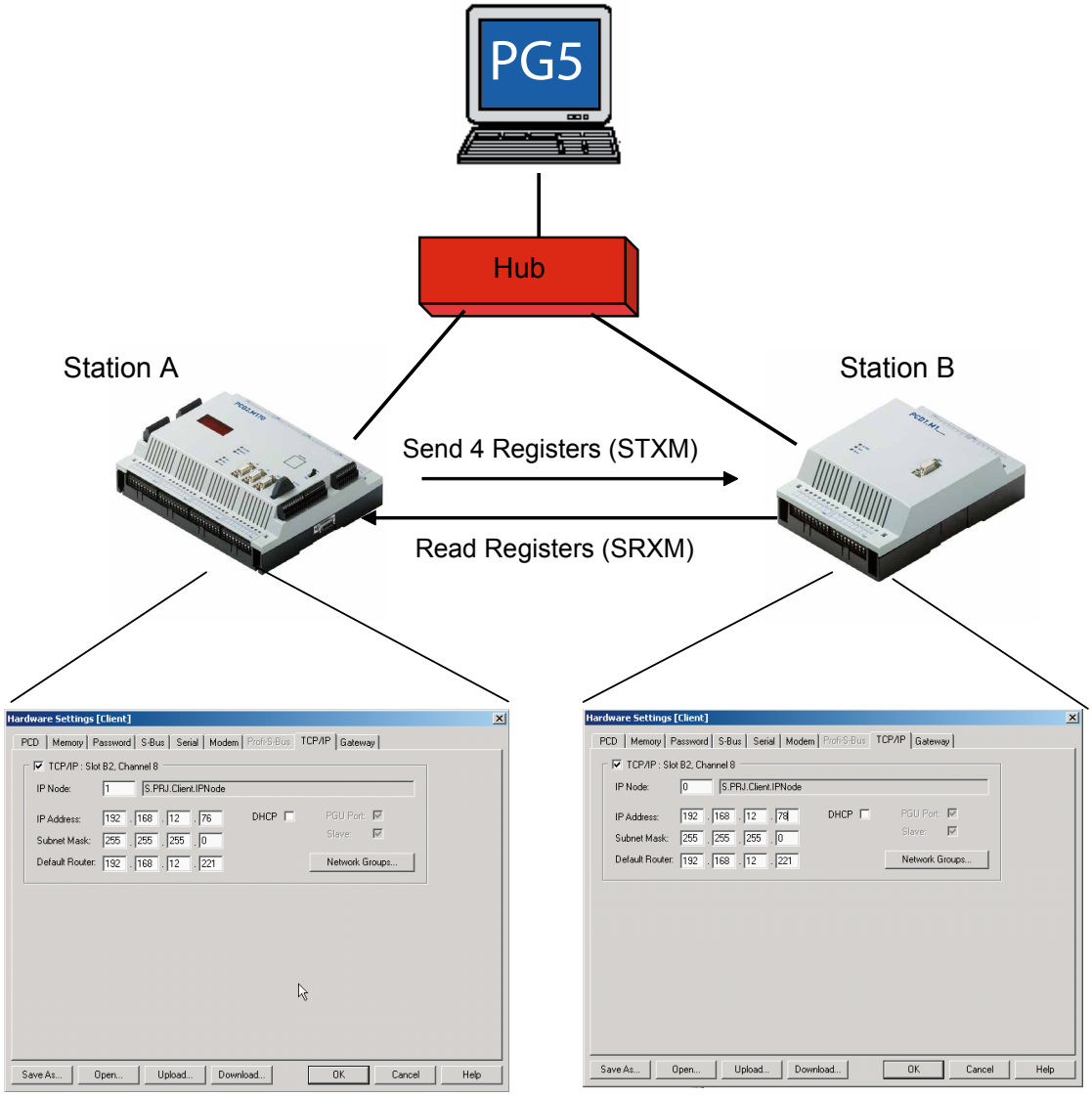
Weitere Einzelheiten im S-Bus-Handbuch.

Anwendungsbeispiel:

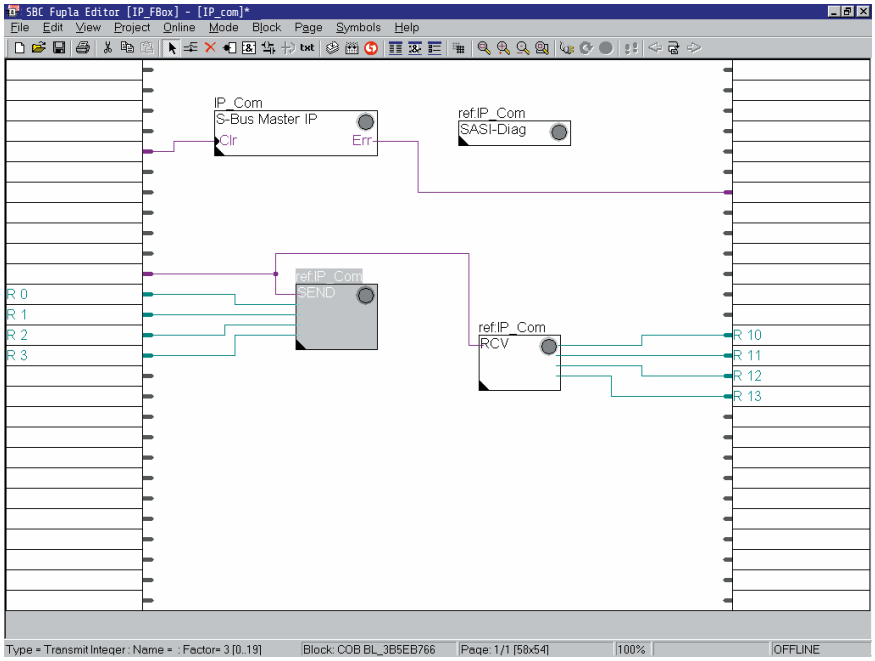
```
LDL     R 100       ; Zieladressen-Register
        3           ; S-Bus-Adresse 3
LDH     R 100
        15          ; IP-Knoten 15
STH     TBSY        ; Busy-Flag für Senden überprüfen
JR      H
next

STXM    9           ; Steckplatz 9 / B1
        4           ; vier Elemente
        F 500       ; Ursprungsadresse
        O 32        ; Zieladresse
next:
```


4.2.2 Programmierung des Ethernet-S-Bus mit Saia PG5® FBoxen

Beispiel:

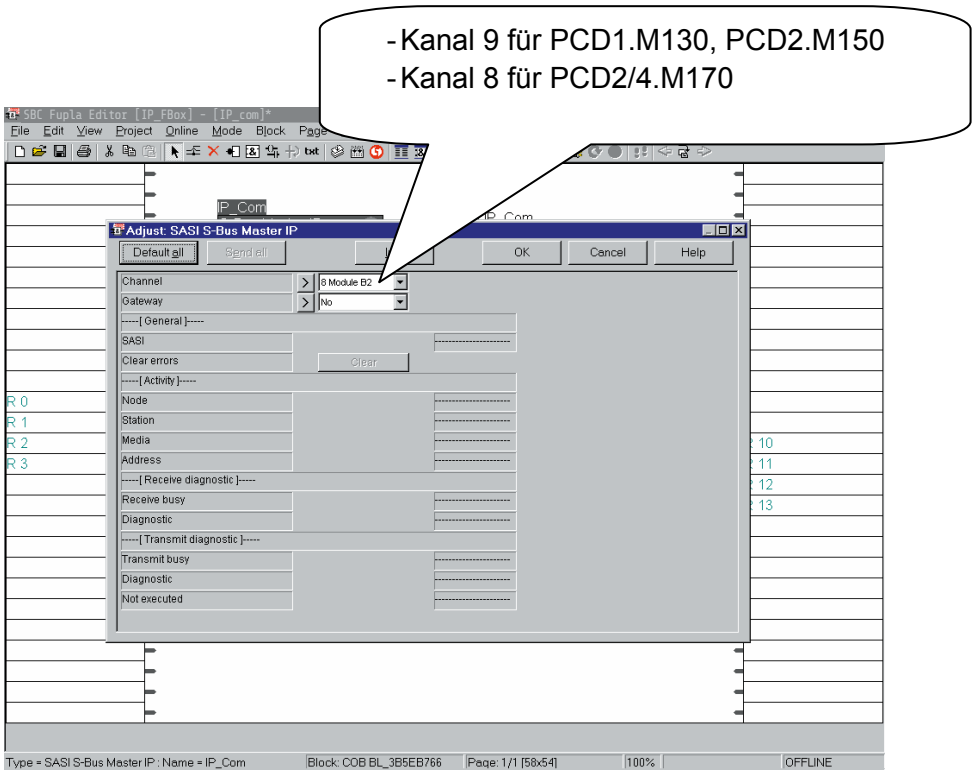


Station A sendet an / empfängt von Station B praktisch gleichzeitig 4 Register.

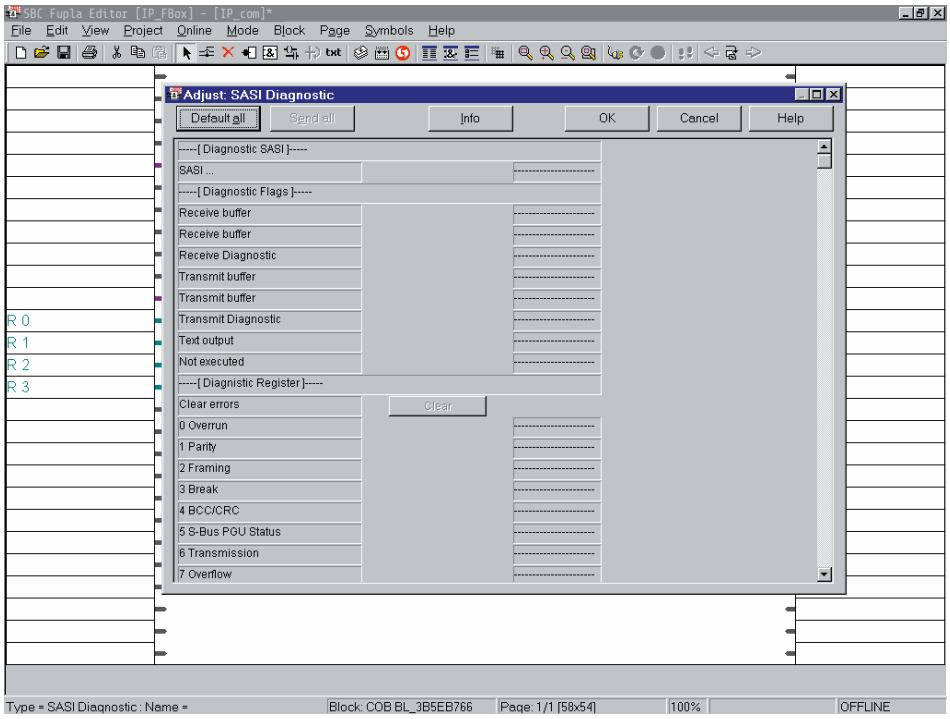


SASI IP Master-Einstellung

Kanalauswahl in der SASI-Einstellung:

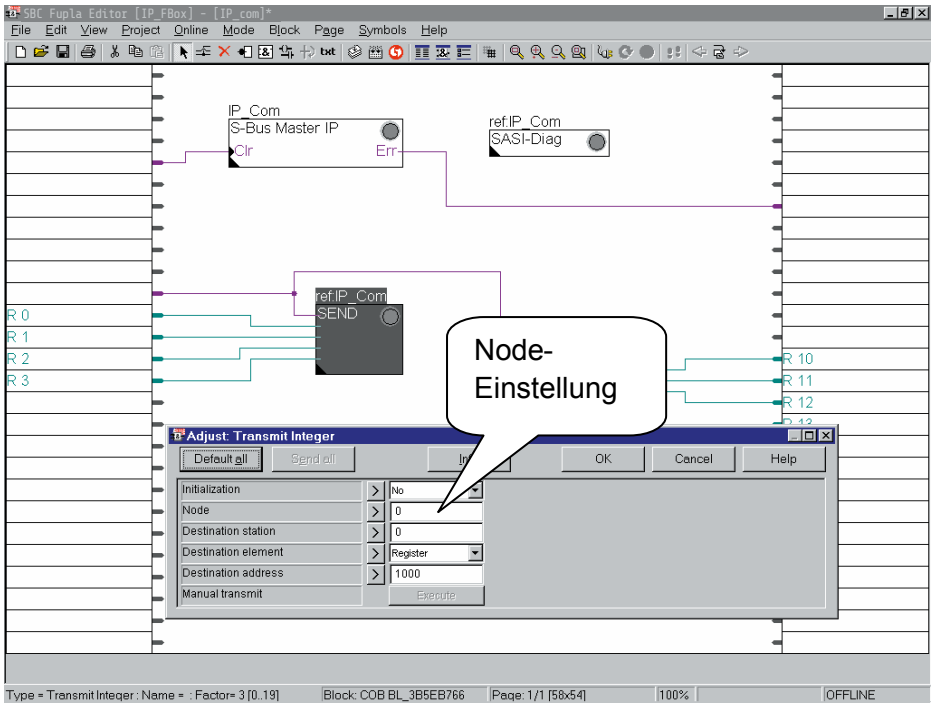


SASI IP Diag-Einstellung



Sende/Empfangs-Einstellung

Der einzige Unterschied zwischen dem IP-S-Bus und dem seriellen S-Bus ist der Node in der Adressierungs-Einstellung.



4.2.3 S-Bus IP Multimaster

Mehrere Master können an einen S-Bus IP-Slave-Port angeschlossen werden. Jeder Master-Gateway (MGWY) verfügt über 4 Anforderungspuffer (PCD1, 1 Puffer), die für die Verarbeitung 4 paralleler Anforderungen an den MGWY reserviert sind.

Weitere Einzelheiten sind in Kapitel 3 “Eigenschaften und Funktionen“ zu finden.

4.2.4 Broadcast-Telegramme via S-Bus IP

Zwei unterschiedliche Übermittlungsarten von Broadcast-Telegrammen sind erlaubt (für die Synchronisierung mit einer Client-Station):

- Broadcast auf das S-Bus-Netz beschränkt (ohne Ethernet)
- Broadcast über das ganze Ethernet einschliesslich aller S-Bus-Stationen unter der adressierten Gateway-Station.

Weitere Einzelheiten sind in Kapitel 3 “Eigenschaften und Funktionen“ zu finden.

4.3 Programmierung des Open Data Mode via Ethernet

Der Open Data Mode wird generell für die PCD-Kommunikation mit einer Station eines Fremdherstellers verwendet, die den S-Bus nicht unterstützt. Jegliche Fremprotokolle können somit implementiert werden. Zwei PCDs können, bei Bedarf, ebenfalls im Open Data Mode kommunizieren.

Stationen fremder Hersteller unterstützen keine proprietären Protokolle (wie S-Bus). Daher sollten via IP nur unverarbeitete Datenblöcke (Zeichen, Strings ohne Header) übertragen werden.

Die PCD kann zwar Daten an eine abgesetzte Station senden, aber im Client-Modus keine Daten direkt von der abgesetzten Station anfordern. Die im Open Data Mode empfangenen Daten werden an die Anwendungsschicht gesendet.

Wenn mit dem Übertragungsprotkoll UDP gearbeitet wird, kann nicht festgestellt werden, ob die Verbindung zwischen zwei Stationen unterbrochen wurde. Dieses Leistungsmerkmal muss durch den Benutzer in der Anwendungsschicht implementiert werden. Ein Kommunikationsüberwachungsverfahren, das Unterbrechungen in der Kommunikation feststellen kann, wurde in das TCP-Protokoll implementiert.

4.3.1 Beschreibung des Open Data Mode

Im Open Data Mode werden die unverarbeiteten Daten an den UDP- oder TCP-Header angefügt und dann gesendet. Im S-Bus auf UDP werden die Daten immer an den UDP-Header angehängt.

4.3.2 Konfiguration

Das IP-Modul muss mit der IP-Adresse, der Subnet-Maske und dem Default-Routerals S-Bus auf IP unter Verwendung der PG5-Hardware-Einstellungen konfiguriert werden. Keine weitere Konfiguration ist notwendig. Der Open Data Mode wird durch den Befehl InitODM im Benutzerprogramm initialisiert.

4.3.3 Wichtige Merkmale in UDP und TCP im Open Data Mode

Generelles Merkmal:

Im Open Data Mode UDP und TCP ist wegen der Grösse der Mailbox zwischen dem Ethernet TCP/IP Modul und der PCD zu beachten, dass die maximal zulässige Länge der gesendeten Nutzdaten 1536 Bytes (720 Bytes bei PCD7F650/655) pro Telegramm nicht übersteigt. Überzählige Daten gehen beim Senden verloren.

TCP Merkmale:

TCP ist ein verbindungsorientertes und flussorientiertes Protokoll. Es kann somit vorkommen, dass Einzelteile eines Telegrammes oder einige zusammengefasste Telegramme über das Ethernet verschickt werden. Es ist die Pflicht des Benutzers beim Empfang die Telegramme, wenn nötig wieder zusammenzusetzen. TCP garantiert jedoch, dass die Daten in der richtigen Reihenfolge ankommen. TCP hat eine Kontrollschicht implementiert, welche allfällige Wiederholungen beim Versenden garantiert. Es gehen in TCP auf dem Ethernet keine Daten verloren.

TCP gibt Auskunft über den Verbindungsstatus.

In TCP herrscht das Client-Server-Prinzip. Es wird vor der Verbindung der beiden Kommunikationsendpunkte in Client und Server unterschieden. Der Server wartet auf eine Verbindung eines Clients. Ein Server kann sich über denselben Eingangsport mit verschiedenen Clients verbinden lassen. Der Client kann sich über denselben Ausgangsport jedoch nur mit einem Server verbinden. Ist die Verbindung zwischen Client und Server hergestellt, so können beide Seiten Daten senden.Beide Seiten sind dann quasi Client, beide können auch eine Verbindung beenden.

UDP Merkmale:

UDP ist ein verbindungsloses und telegrammorientiertes Protokoll. Es werden die ganzen Telegramme einzeln und auf einmal verschickt. Es gibt jedoch keine Kontrollschicht über den Datenverkehr wie für Wiederholungen in TCP. Es kann nicht garantiert werden, dass die Telegramme den Endpunkt erreichen. Der Benutzer muss selbständig Kontrollmechanismen, wie zum Beispiel Handshake, programmieren. Jede Station kann jeder Station Telegramme senden. Es gibt also keine Client-Server Beziehung. Auch können in UDP keine grösseren Telegramme als 1536 Bytes Nutzdaten (720 Bytes bei PCD7F650/655) (Grösse der Kommunikations-Mailbox) empfangen werden.

Ein globales Diagnose-Flag gibt Auskunft ob die Verbindung zum nächstliegenden Hub/Switch besteht. Diese Verbindung kann jederzeit getestet werden. Zum Beispiel bevor ein UDP Telegramm verschickt wird.

4.3.4 Programmierung mit IL

Der Open Data Mode wird mit den Systemfunktionen programmiert.

Call system function

Dies ist eine Art Instruktionserweiterung mit verschiedenen Bibliotheken. Der Aufruf gleicht einem FB-Aufruf. Für das Ethernet-TCP/IP-Modul steht eine Bibliothek zur Verfügung. Diese Bibliothek ist Bestandteil der Firmware. Die Funktionen werden mit der CSF-Instruktion aufgerufen. Zwischen dem Ethernet-TCP/IP-Modul und der PCD werden mit diesen CSF-Instruktionen über eine gemeinsame Mailbox (Sendepuffer, Empfangspuffer) kommuniziert. Der Benutzer dieser CSF-Instruktionen steuert die Kommunikation des Open Data Modes dementsprechend selbständig und ist verantwortlich für den korrekten Kommunikationsfluss über die Mailbox. So, dass zum Beispiel keine Telegramme in der Mailbox hängen bleiben.

Definition der CSF (Call System Function):

```
CSF [cc]    <Library>
            <Function>
            [<parameter 1>]
            [<parameter 2>]
            ....
            [<parameter n>]
```

Beispiel:

```
CSF [cc]    S.IP.Library
            S.IP.InitODM
            F 1000
            R 1000
            1000
```

Die PCD2.M480 kann mit zwei PCD7.F65x bestückt werden. Je nach Steckplatz (B1 oder B2) muss die Bezeichnung der Library für die Initialisation des Open Data Mode angepasst werden.

Beispiel für Steckplatz B1:

```
CSF [cc]    S.IP.Library_SlotB1
            S.IP.InitODM
            F 1000
            R 1000
            1000
```

Beispiel für Steckplatz B2:

```
CSF [cc]    S.IP.Library_SlotB2
            S.IP.InitODM
            F 2000
            R 2000
            1000
```

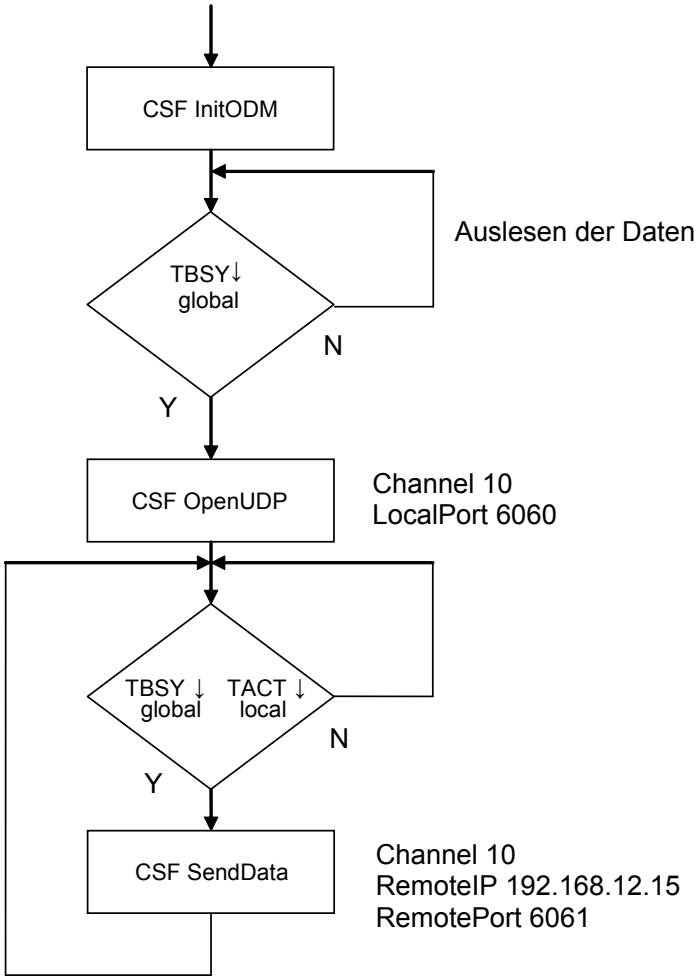
Übersicht über die zur Verfügung stehenden CSF für den Offenen Data Mode. Die Details zu den Funktionen werden auf den folgenden Seiten erläutert.

CSF 0	IPODMInit
CSF 1	IPODMOpenUDP
CSF 2	IPODMOpenClientTCP
CSF 3	IPODMOpenServerTCP
CSF 4	IPODMClose
CSF 5	IPODMConnectTCP
CSF 6	IPODMDisconnectTCP
CSF 7	IPODMGetConnectionTCP
CSF 8	IPODMAcceptTCP
CSF 9	IPODMSend
CSF 10	IPODMRead
CSF 11	IPODMSendRev
CSF 12	IPODMReadRev

Anbei zwei Beispiele in Blockdiagramm-Logik zur Erklärung wie mit diesen CSFs gearbeitet wird. Zur Vereinfachung der Darstellung wurden bei den JA/NEIN Abfragen “JR -1“ Schritte benutzt.

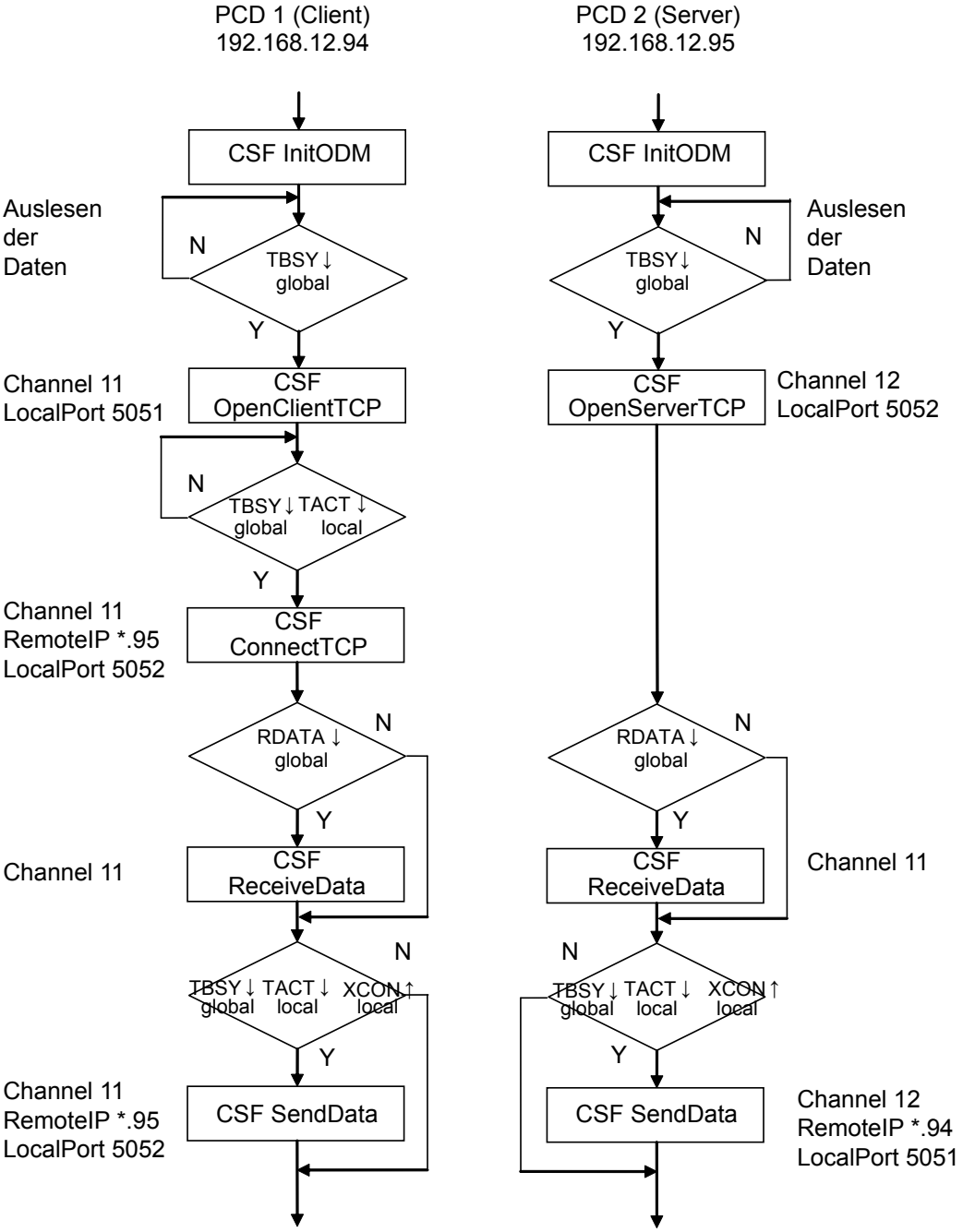
Zunächst eine UDP Kommunikation: Der CSF “Open Data Mode“ ist nötig um den Open Data Mode zu initialisieren. Da UDP verbindungslos ist, wird keine Verbindung zwischen den Kommunikations-Teilnehmern aufgebaut. Es gibt somit keine Client-Server Beziehung und jeder Teilnehmer kann mit jedem beliebigen anderen UDP Teilnehmer kommunizieren. Voraussetzung ist, dass der Ziel-Teilnehmer ein UDP Socket auf einem dem Sende-Teilnehmer bekannten Port geöffnet hat. Im folgenden Beispiel wird vorausgesetzt, dass sich auf dem Netzwerk ein anderer Teilnehmer mit der IP-Adresse 192.168.12.15 und einem geöffneten UDP Socket auf Port 6061 befindet, den wir adressieren können. Wie später noch eingehend erwähnt wird, ist der Behandlung des Diagnose Flags “RDATA“ (Daten sind angekommen) höchste Priorität zuzuordnen.

UDP



Beispiel der TCP Kommunikation: Der CSF “Open Data Mode“ ist nötig um den Open Data Mode zu initialisieren. Da TCP verbindungsorientiert ist, muss der Client auf seinen Server eine Verbindung aufbauen, über welche dann Telegramme ausge-tauscht werden. Der Client öffnet ein TCP-Client Socket mit “OpenCLientTCP“ und der Server ein TCP-Server Socket mit “OpenServerTCP“. Auf diesem Port horcht der Server auf eine Verbindungsanfrage. Deshalb wird an dieser Stelle oft vom “Listener Socket“ gesprochen. Mit dem CSF “ConnectTCP“ verbindet sich der Client auf den Server Teilnehmer mit 192.168.12.95 und Port 5052. Ist die Verbindung einmal erfolgt, dann gibt es eigentlich keinen Unterschied mehr zwischen Client und Server. Beide Teilnehmer können dann die Initiative ergreifen und einander Telgramme und Anfragen schicken. Auch können jederzeit beide die Verbindung beenden. Wie später noch eingehend erwähnt wird ist der Behandlung des Diagnose Flags “RDATA“ (Daten sind angekommen) höchste Priorität zuzuordnen.

TCP


PCD 1 (Client)
192.168.12.94
PCD 2 (Server)
192.168.12.95
CSF InitODM
CSF InitODM
Auslesen der Daten
N
TBSY ↓ global
Y
Auslesen der Daten
N
TBSY ↓ global
Y
Channel 11
LocalPort 5051
CSF OpenClientTCP
CSF OpenServerTCP
Channel 12
LocalPort 5052
N
TBSY ↓ global
TACT ↓ local
Y
Channel 11
RemoteIP *.95
LocalPort 5052
CSF ConnectTCP
N
RDATA ↓ global
Y
N
RDATA ↓ global
Y
Channel 11
CSF ReceiveData
CSF ReceiveData
Channel 11
N
TBSY ↓ global
TACT ↓ local
XCON ↑ local
Y
N
TBSY ↓ global
TACT ↓ local
XCON ↑ local
Y
Channel 11
RemoteIP *.95
LocalPort 5052
CSF SendData
CSF SendData
Channel 12
RemoteIP *.94
LocalPort 5051

4.3.5 InitODM

Unter PG5 1.4 wurde ein erster provisorischer Mechanismus eingeführt, um die ODM-Modi zwischen SBC- und Engiby Bibliotheken zu teilen. Da nur wenige User-Anwendungen von diesem Fall (unter PG5 1.4) betroffen sind, wird dieses ältere Prinzip hier nicht beschrieben.

Für PG5 2.0 wurde ein allgemein gültigeres Prinzip umgesetzt und sollte nun von allen Entwicklern von Kommunikationsanwendungen unter ODM-Modus respektiert werden. Diese Version ist in diesem Dokument beschriebenen und erlaubt die Verwendung von ODM in mehreren Bibliotheken gleichzeitig.

4.3.5.1 Initialisierung des ODM-Modus

Der ODM-Modus muss in der CPU einmal für alle Anwendungen initialisiert werden, durch den Mittelwert der SCF-Anweisung. Diese Anweisung kann weder in der Anwender-Applikation noch in einer Kommunikations-Bibliothek platziert werden, da bei einer geteilten ODM nur eine gelingen wird.

Zur Lösung dieses Problems wird die ODM Initialisierung nun durch die SPM ausgeführt, wenn die entsprechende Option im Device Configurator unter den Ethernet-Optionen in der TCP/IP-Gruppe eingestellt ist.

Durch Einstellen der Option „Yes“ werden die folgenden 3 Massnahmen ausgeführt:

- Der erforderliche Code um die ODM zu initialisieren wird erzeugt
- Das Auslese Timeout (Reading timeout) wird für alle Anwendungen definiert
- Eine Reihe von System-Symbolen werden veröffentlicht

4.3.5.2 Telegramm Auslese Timeout

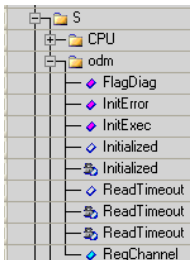
Der Standardwert (und empfohlene Wert) für das Auslese Timeout liegt bei 1000 Millisekunden. Dieser Wert gilt für alle Anwendungsbereiche der ODM. Dabei handelt es sich um die maximal erlaubte Zeit für das Lesen eines empfangenen Datagramms. Jede Anwendung ist verantwortlich für das Lesen der adressierten Pakete. Solange ein Paket nicht aus dem Empfangspuffer gelesen wurde, können keine anderen (auch für andere Anwendungen) gelesen werden. Daher ist das Auslese Timeout ein Sicherheitsmechanismus, um sicherzustellen, dass der Empfangspuffer nicht durch eine einzelne Anwendung blockiert werden kann.

Allerdings, wenn eine Anwendung nicht mehr empfangene Datagramme liest, wird die Kommunikation der anderen Anwendungen drastisch verlangsamt und kann sogar dazu führen, dass Pakete verloren gehen.

Daher muss eine Anwendung mit gemeinsamen ODM die eingehenden immer unverzüglich Pakete überprüfen und sie so schnell wie möglich lesen. Falls die sofortige Verarbeitung des Pakets nicht möglich ist, sollte das Paket gelesen und in einem temporären Puffer gespeichert werden (oder aber ignoriert werden mit einer Fehleranzeige).

4.3.5.3 System Symbole

Das ist eine Liste mit den publizierten System Symbolen.



Beschreibung

s.odm.FlagDiag	Basisadresse von 8 Diagnose-Flags. Siehe Ethernet-Handbuch
s.odm.FlagDiag+0	Übertragung beschäftigt
s.odm.FlagDiag+1	Empfang beschäftigt
s.odm.FlagDiag+2	Verbindungs-Event empfangen
s.odm.InitError	Initialisierung fehlgeschlagen. Nur gültig, wenn InitExec eingestellt
s.odm.InitExec	Initialisierung wurde ausgeführt
s.odm.Initialized	ODM Initialisierung Option gesetzt
s.odm.ReadTimeout	Lese Timeout-Wert in Millisekunden
s.odm.RegChannel	Register zur Angabe des betroffenen Kanals durch ein empfangenes Paket

Eine Anwendung mit gemeinsamer ODM sollte zunächst prüfen, ob die ODM-Option festgelegt wurde und dann eine Nachricht ($warning, $error oder sogar $fatal) ausgeben, wenn dies nicht der Fall ist. Assembler-Richtlinien sollten auch verwendet werden, um weitere Assembler Fehler zu vermeiden, wenn ODM nicht initialisiert wurde. Dies macht das Verständnis und die Problembehebung einfacher für die Benutzer.

Beispiel:

```
$ifndef s.odm.initialized
  $error Driver xyz, ODM not initialized. See Device configurator.
  exitm    ;exit macro to avoid further errors
$else
$if s.odm.initialized = 0
  $error Driver xyz, ODM not initialized. See Device configurator.
  exitm    ;exit macro to avoid further errors
$endif
$endif
```

4.3.5.4 Initialisierung ausführen und Fehler

Bevor die gemeinsame ODM verwendet werden kann, muss die Anwendung sicherstellen, dass die Initialisierung ohne Fehler ausgeführt wurde. Weil nicht vorhergesagt werden kann, wo der Code im $init-Abschnitt platziert wird. Diese Überprüfung muss im COB-Segment erfolgen mit den beiden System-Symbolen: s.odm.InitExec und s.odm.InitError.

Beispiel:

```
$cobseg
     sth    s.odm.InitExec     ;;Executed
     anl    s.odm.InitError    ;;wihtout error
     jr     l l_skip     ;;->Not yet initialized or error
                 ;;Else ready to use ODM
$endcobseg
```


4.3.5.5 Wahl der ODM Kanäle

Die Anzeige der S-Bus-Kanäle (8 oder 9) darf im Device Configurator nicht verwechselt werden. Diese Kanäle sind nur gültig für den S-Bus und wählen den Zugriff auf die Ethernet-Karte.

Für den ODM-Modus hat jede CPU eine begrenzte Anzahl von logischen Kanälen zur Verfügung. Jeder Treiber muss einen anderen Kanal benutzen. Um Kollisionen zu vermeiden, muss die Anzahl der ODM Kanalnummer durch den Benutzer einstellbar sein.

Die Anzahl der Kanäle sind in Kapitel 4.3.7 aufgelistet.

Zurzeit wird ein Mechanismus für die dynamische Zuweisung von ODM-Kanälen untersucht. Dies wird in diesem Dokument noch nicht vorgestellt.

In der Zwischenzeit gelten die folgenden Regeln:

- Kanäle 1 bis 10 (einstellbar) sollten für Benutzeranwendungen und Drittanbieter-Bibliotheken eingesetzt werden.
- Kanäle 11 bis 19 reserviert für zukünftige Nutzung
- Kanäle 20 bis 29 (nicht einstellbar) werden für SBC-Standard Saia PG5® FBoxen (WAA-Bibliothek) verwendet.

4.3.5.6 Empfang von Paketen

Wie oben erläutert, muss jede Anwendung, die die gemeinsame ODM benutzt,

zyklisch den Empfang von Paketen prüfen und die empfangenen Pakete (und nur Pakete, die an seinen Kanal geschickt wurden) so schnell wie möglich lesen.

Die Anwendung überprüft den Kanal mit dem System Symbol s.odm.RegChannel.

Beispiel:

```
sth    s.odm.FlagDiag+1
jr     l l_no_data ;;->No reception

cmp    .s.odm.RegChannel
       My_Channel
acc    z
jr     l l_no_data ;;->Not for my channel

CSF    ReceiveData
```


4.3.5.7 Empfang von Anschluss Ereignissen

Die gleichen Regeln gelten für den Empfang von Verbindungsereignissen, wenn der TCP-Client-Verbindungsfilter verwendet wird.

Die Anwendung muss zyklisch den Empfang von Ereignissen prüfen und die empfangenen Ereignisse und nur Ereignisse, die an seinen Kanal gesandt wurden, so schnell wie möglich lesen.

Die Anwendung muss den Kanal, an welchen ein Ereignis adressiert wurde, mit dem System Symbol s.odm.RegChannel überprüfen.

Beachten Sie, dass für die Anzeige des Kanals für die Daten-Pakete und für Verbindungsereignisse die gleichen Register verwendet werden. Die FW stellt sicher, dass nur eines der beiden Flags (Datenempfang und Event-Empfang) zur gleichen Zeit eingestellt ist. Dazu ist es wichtig, die Verbindungsereignisse immer zu lesen, auch wenn die Ereignisse in einigen Situationen ignoriert werden können.

Beispiel:

```
sth    s.odm.FlagDiag+2
jr     l l_no_event      ;;->No event reception

cmp    s.odm.RegChannel
       My_Channel
acc    z
jr     l l_no_event      ;;->Not for my channel

CSF    Get connection event
```

4.3.6 Diagnose

Globale Diagnose-Flags

Addresse	Name	Beschreibung:
Xxxx	TBSY	Transmitter busy
xxxx + 1	RDATA	Receive data Server
xxxx + 2	RCON	Connection
xxxx + 3	Reserviert	
xxxx + 4	Reserviert	
xxxx + 5	Reserviert	
xxxx + 6	XBSY	Physical Link
xxxx + 7	Reserviert	

Transmitter busy (TBSY)↑

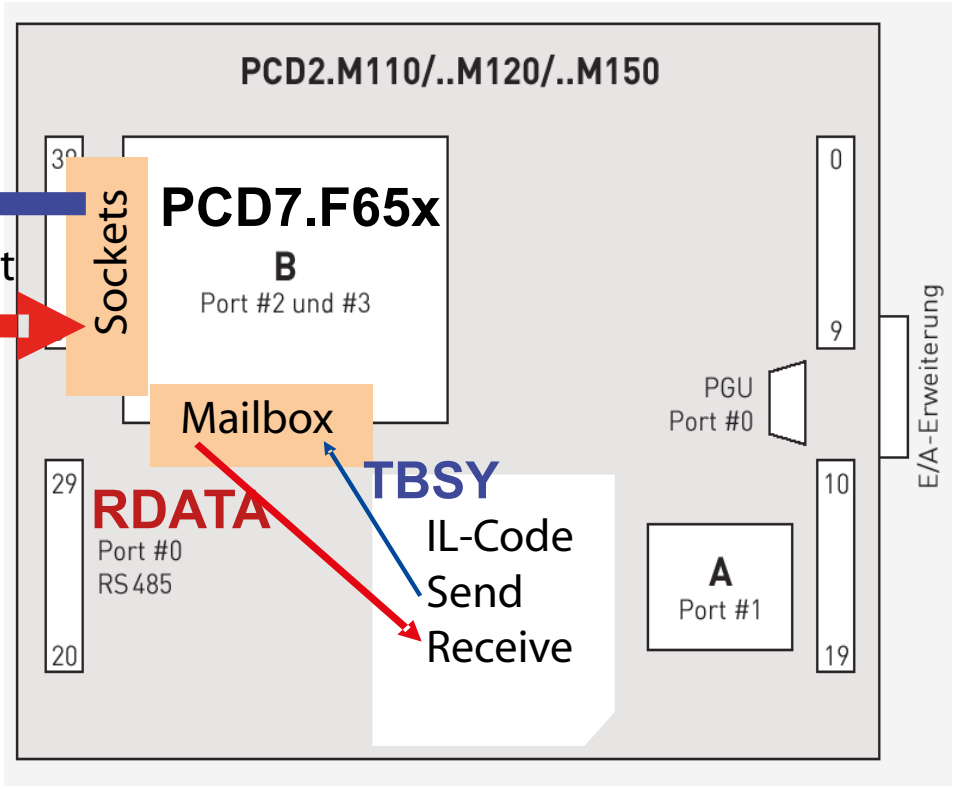
Wird gesetzt, wenn sich ein Telegramm im Sendepuffer befindet. Es ist nicht möglich, Daten an irgendeinen beliebigen anderen Kanal zu senden, wenn dieses Flag gesetzt ist.

Receive data (RDATA)↑

Wird gesetzt, wenn sich ein Telegramm im Empfangspuffer befindet. Dies ist das zentrale Flag im Open Data Mode und muss dementsprechend mit erster Priorität behandelt werden. Sobald sich Daten in der Empfangsmailbox der PCD befinden -dies wird mit RDATA “H“ (high) angezeigt- muss die Mailbox seitens PCD gelesen, die Daten bearbeitet und die Mailbox mittels Aufruf des CSF Receive Data für andere Kanäle wieder freigegeben werden. Der Sende- und Empfangspuffer kann per Transaktion (Senden/Empfangen) maximal 1536 Bytes Nutzdaten (720 Bytes bei PCD7F650/655) aufnehmen.

Beim Empfang eines Datentelegrammes, RDATA “H“ (high), zeigt der Inhalt des “Channel-Registers“ (globale Diagnose bei InitODM) für welchen Kanal die Daten an gekommen sind. Hierzu gibt es einen wichtigen Hinweis: Wenn die lokale Station mit mehr als einer Gegenstation kommuniziert (wie zum Beispiel ein TCP Server, der mit mehreren TCP Clients verbunden ist) so muss der Inhalt des “Channel Registers“ vor dem Aufruf des CSF “ReceiveData / ReceiveDataRev“ zwischengespeichert werden, um nach dem Abholen der Daten mit der Kopie des “Channel Registers“ weiter zu arbeiten. Denn, gleich nach dem Aufruf des CSF “ReceiveData / ReceiveDataRev“ rückt bereits das nächste ankommende Telegramm nach, und der Inhalt des “Channel Registers“ kann verändert sein.

Siehe dazu auch den Eintrag FAQ #100584 “Receive data in Open Data Mode“.



Receiver connection (RCON)↑

Wird gesetzt, wenn auf einem TCP-Server eine Verbindungsänderung erfolgt und ist gleichzeitig die Aufforderung, die “GetConnectionTCP-Funktion“ auszuführen erfolgt und ist gleichzeitig die Aufforderung, die “GetConnection” TCP-Funktion auszuführen.

Physical Link (XBSY)

Zeigt an, ob ein physikalischer Link zum nächsten HUB/Switch vorhanden ist oder nicht. Ist auf “H“ (high) gesetzt, wenn das Ethernet TCP/IP Module einen physikalischen Link detektiert und ist auf “L“ (low) gesetzt, wenn der physikalische Link verloren geht.

Kanal-Diagnose-Flags

Im Open Data Mode programmiert der Benutzer praktisch direkt in einer Abstraktionsebene über den Berkeley Sockets. Um die Programmierung zu vereinfachen und zu abstrahieren wird mit sogenannten Kanälen gearbeitet. Für praktisch jedes Socket auf dem Ethernet TCP/IP Module (Open Data Mode UDP und TCP) gibt‘s einen virtuellen Kanal über welchen kommuniziert wird. Jeder Kanal besitzt seine eigenen Diagnose-Flags und ein Diagnose-Register. Auf diese Kanäle wird in den Funktionen “OpenUDP“, “OpenClientTCP“, “OpenServerTCP“, “Close“, “ConnectTCP“, “DisconnectTCP“, “AcceptConnectionTCP“, “SendData“, “SendDataRev“, “ReceiveData“ und “ReceiveDataRev“ zugegriffen.

Adresse	Name	Beschreibung:
xxxx	RDIA	Receiver diagnostic
xxxx + 1	TACT	Transmitter active
xxxx + 2	TDIA	Transmitter diagnostic
xxxx + 3	XCON	Port connected
xxxx + 4	NEXE	Not executed
xxxx + 5	Reserviert	
xxxx + 6	Reserviert	
xxxx + 7	Reserviert	

Receiver diagnostic (RDIA)↑
Im Diagnose-Register ist eine Empfänger-Diagnose vorhanden. Das Flag wird auf “H“ gesetzt, wenn beim Empfang eines Telegramms ein Fehler festgestellt wird. Eine ausführliche Beschreibung des Fehlers ist im Diagnose-Register (Bits 0 bis 15) zu finden. Das Flag wird zurückgesetzt, sobald alle Receiver-Diagnosebits im Diagnose-Register zurückgesetzt wurden

Transmitter active (TACT)↑
Ist gesetzt, wenn sich ein Telegramm im Sendepuffer befindet. Es ist nicht möglich, andere Daten an diesen Kanal zu senden oder diesen Kanal zu verbinden / zu trennen bis TACT nicht mehr aktiv ist.

Transmitter diagnostic (TDIA)↑
Im Diagnose-Register ist eine Sender-Diagnose vorhanden. Es wird auf “H“ gesetzt, wenn beim Senden eines Telegramms ein Fehler festgestellt wird. Eine ausführliche Beschreibung des Fehlers ist im Diagnose-Register
(Bits 16 bis 31) zu finden. Das Flag wird zurückgesetzt, sobald alle Sender-Diagnosebits im Diagnose-Register zurückgesetzt wurden.

Port connected (XCON)↑
Der TCP-Port ist mit einer anderen TCP Station verbunden.

Not executed (NEXE)↑
Wird gesetzt, wenn eine Instruktion nicht vollständig ist oder nicht ausgeführt wurde. Das Flag wird durch die nächste erfolgreiche Kommunikations-Instruktion zurückgesetzt.

Kanal-Diagnose-Register

	Bit	Bezeichnung	Beschreibung
EMPFÄNGER	0	Read timeout error	Ein Telegramm wurde nicht aus dem Puffer ausgelesen
	1	Nicht benutzt	
	2	Nicht benutzt	
	3	Nicht benutzt	
	4	Nicht benutzt	
	5	Nicht benutzt	
	6	Nicht benutzt	
	7	Nicht benutzt	
	8	Length error	Die Empfangslänge ist grösser als der Puffer
	9	Nicht benutzt	
	10	Address error	Fehler bei der Umsetzung der IP-Adresse aus dem Puffer
	11	Nicht benutzt	
	12	Range error	Ungültige Elementadresse
	13	Status error	Ungültiger Status
	14	Rx Mailbox error	Mailbox enthält keine Daten
	15	Rx Channel error	Falsche Kanalnummer
SENDER	16	Nicht benutzt	
	17	Nicht benutzt	
	18	Nicht benutzt	
	19	Nicht benutzt	
	20	Nicht benutzt	
	21	Nicht benutzt	
	22	INicht benutzt	
	23	Stat. not present	Zielstation nicht präsent
	24	Nicht benutzt	
	25	Nicht benutzt	
	26	Nicht benutzt	
	27	Nicht benutzt	
	28	Range error	Ungültige Elementadresse
	29	Add error	Port nicht vorhanden oder Port und Adresse passen nicht
	30	No connection	Die Verbindung zu dieser Station ist nicht geöffnet
	31	Program error	Unzulässiger Sendeversuch

Jedes Bit, das im Diagnose-Register auf “H“ (high) gesetzt wurde, bleibt solange gesetzt, bis es durch das Benutzerprogramm oder den Debugger manuell zurückgesetzt wird.

Read timeout error (Bit 0)
Wird auf “H“ gesetzt, wenn ein Telegramm nicht aus dem Puffer ausgelesen wurde. Puffer-Timeout ist zu kurz oder Kanal wurde nicht gelesen.

Length error (Bit 8)
Wird auf “H“ gesetzt, wenn die Empfangslänge die Pufferlänge übersteigt.

Address error (Bit 10)
Wird auf “H“ gesetzt, wenn bei der Umsetzung der IP-Adresse aus dem Puffer ein Fehler auftritt:
- Umwandlung in Register
- Text zu kurz

Range error (Bit 12)
Wird auf “H“ gesetzt, wenn die Elementadresse ungültig ist.

Mailbox error (Bit 14)	Wird auf “H“ gesetzt, wenn die ausgelesene Mailbox keine Daten enthält.
Channel error (Bit 15)	Wird auf “H“ gesetzt, wenn die ausgelesene Kanalnummer falsch ist.
Target not present (Bit 23)	Wird auf “H“ gesetzt, wenn die Zielstation auf dem Netz nicht angesprochen werden kann. Zum Beispiel wenn der physikalische Link zum nächsten HUB/Switch nicht besteht und per UDP ein Telegramm verschickt wird (nicht PCD7.F655). Oder wenn das sendende Socket im TCP/IP Stack einen Fehler meldet und das Telegramm somit nicht auf das Ethernet geschickt wurde.
Range error (Bit 28)	Wird auf “H“ gesetzt, wenn die Elementadresse ungültig ist.
TxNode error (Bit 29)	Wird auf “H“ gesetzt, wenn der Node nicht vorhanden ist.
No connection (Bit 30)	Wird auf “H“ gesetzt, wenn die Verbindung zu dieser Station nicht geöffnet ist.

4.3.7 Anzahl der ODM Kanäle

Die Gesamtzahl der pro PCD (TCP und UDP) erlaubten Kanäle hängt vom PCD-System ab. Es sind dies:

PCD System	**PCD1.M130**	**PCD2.M150**	**PCDx.M170**	**PCD2.M480**
Anzahl der Kanäle	16	32	32	32
PCD System	**PCD3.Mxxx**	**PCD1.M2xx0**	**PCD2.M5**	
Anzahl der Kanäle	32	32	32	

Wobei erwähnt werden muss, dass S-Bus UDP einen Kanal benötigt, wenn Eingangs- (S-Bus Server) und Ausgangs-Port (S-Bus Client) auf 5050 gewählt wird.Benutzen S-Bus Server und S-Bus Client verschiedene Ports, so werden vom System zwei Kanäle benötigt. Im Open Data Mode können dann die weiteren Kanäle benutzt werden. Wobei für diese Tabelle alle mit “OpenUDP“, “OpenTCPClient“ und “OpenTCPServer“ geöffneten Kanäle summiert berücksichtigt werden.

4.3.8 OpenUDP

Öffnet einen UDP-Kanal im IP Open Data Mode (entspricht einem “socket“ und “bind“ im Berkeley Standard). Nach diesem Aufruf können über diesen UDP-Kanal Telegramme versendet werden.

Diese Funktion kann nur ausgeführt werden, wenn TBSY auf “L“ (low) gesetzt ist.CSF

```
[cc]    S.IP.Library          ; IP-Bibliothek
        S.IP.OpenUDP          ; Funktion Open UDP channel
→       LocalPort             ; der lokale IP-Port über welchen Daten
                              ; gesendet/ empfangen werden (R/K)
→       Ch_Diag_Flag          ; Basis der Kanal-Diagnose-Flags
                              ; (8 Flags) (F)
→       Ch_Diag_Register      ; Kanal-Diagnose-Register (R)
```

LocalPort: Wahl des lokalen Ports für die Open Data Mode UDP Kommunikation
0 = ein vom System freier Port wird gewählt
X = direkte Zuweisung des lokalen Ports durch den Benutzer
Der LocalPort 5050 ist für den S-Bus-UDP Modus reserviert.

TACT↑ wenn der Befehl “OpenUDP“ für das Ethernet-TCP/IP Module in die Kommunikations-Mailbox abgelegt wurde.

TACT↓ nach dem ACK des Ethernet-TCP/IP Modules. Ein neuer Open Data Mode-Befehl kann ausgeführt werden.

NEXE↑ Wenn zum Beispiel der lokale IP Port bereits benutzt wird.

Beispiel:

```
CSF     [cc]    S.IP.Library          ; Bibliothek
                S.IP.OpenUDP          ; Funktion
                10                    ; Kanal 10
                5000                  ; Port 5000
                F 1010                ; Basis der Kanal-Diag
                                      ; nose-Flags
                R 1                   ; Kanal Diagnose-Register
```

Flags: Wenn die Firmware keine IP Open Data Mode-Unterstützung bietet, wird das Error (E) Flag gesetzt.

4.3.9 OpenClientTCP

Öffnet einen Client-TCP-Kanal im IP Open Data Mode (entspricht einem “socket“ und “bind“ im Berkeley Standard). Nach diesem Aufruf kann sich der TCP Client auf einen TCP Server verbinden.

Diese Funktion kann nur ausgeführt werden, wenn TBSY auf “L“ (low) gesetzt ist.


```
CSF     [cc]    S.IP.Library          ; IP-Bibliothek
                S.IP.OpenClientTCP    ; Funktion Open TCP channel
        →       LocalPort             ; der lokale IP-Port über welchen Daten
                                      ; gesendet/ empfangen werden (R/K)
        →       Ch_Diag_Flag          ; Basis der Kanal-Diagnose-Flags
                                      ; (8 Flags) (F)
        →       Ch_Diag_Register      ; Kanal-Diagnose-Register (R) Conn_Tout
        →       Conn_Tout             ; Verbindungs-Timeout: 0=kein, x=Sek.
                                      ; (R/K)
```

LocalPort: Wahl des lokalen Ports für die Open Data Mode TCP Kommunikation
0 = ein vom System freier Port wird gewählt
X = direkte Zuweisung des lokalen Ports durch den Benutzer
Obwohl der LocalPort 5050 für die S-Bus UDP Kommunikation reserviert ist, kann dieser Port 5050 in TCP trotzdem benutzt werden.

Conn_Tout: Wenn Conn_Tout (Sekunden) überschritten wird, ohne dass über den Kanal ein Telegramm empfangen wurde, wird der Kanal geschlossen (empfangsabhängig).

- 0 = keine Timeout-Kontrolle
- x = Timeout-Kontrolle alle x Sekunden

TACT↑ wenn der Befehl „OpenClient TCP“ für das Ethernet-TCP/IP Module in die Kommunikations-Mailbox abgelegt wurde.

TACT↓ nach dem ACK des Ethernet-TCP/IP Modules. Ein neuer Open Data Mode-Befehl kann ausgeführt werden.

NEXE↑ Wenn zum Beispiel der lokale IP Port bereits benutzt wird.

Beispiel:

```
CSF     [cc]    S.IP.Library          ; Bibliothek
                S.IP.OpenClientTCP    ; Funktion
                10                    ; Kanal 10
                5555                  ; Port 5555
                F 1010                ; Basis der Kanal-Diag
                                      ; nose-Flags
                R 1                   ; Kanal Diagnose-Register
                R 2                   ; Verbindungstimeout
```

Flags: Wenn die Firmware keine IP Open Data Mode-Unterstützung bietet, wird das Error (E) Flag gesetzt.

4.3.10 OpenServerTCP

Öffnet einen TCP-Server-Kanal im IP Open Data Mode (entspricht einem “socket“, “bind“ und “accept“ im Berkeley Standard). Nach diesem Aufruf ist der TCP Server bereit eine Verbindung eines TCP Clients anzunehmen.

Diese Funktion kann nur ausgeführt werden, wenn TBSY auf “L“ (low) gesetzt ist.


```
CSF  [cc]   S.IP.Library              ; IP-Bibliothek
            S.IP.OpenServerTCP  ; Funktion Open UDP channel
       →    Channel                  ; Kanal-Nr. (R/K)
       →    LocalPort                ; der lokale IP-Port über welchen Daten
                                     ; gesendet/ empfangen werden (R/K)
       →    Ch_Diag_Flag             ; Basis der Kanal-Diagnose-Flags
                                     ; (8 Flags) (F)
       →    Ch_Diag_Register         ; Kanal-Diagnose-Register (R) Conn_Tout
       →    Connection_filter        ; Verbindungs-Filter (R/K)
       →    Conn_Tout                ; Verbindungs-Timeout: 0=kein, x=Sek.
                                     ; (R/K)
```

LocalPort: Wahl des lokalen Ports für die Open Data Mode UDP Kommunikation
0 = ein vom System freier Port wird gewählt
X = direkte Zuweisung des lokalen Ports durch den Benutzer
Obwohl der LocalPort 5050 für die S-Bus UDP Kommunikation reservi ert ist, kann dieser Port 5050 in TCP trotzdem benutzt werden.

Connection _filter (nur auf TCP-Server)

- 0 = kein Filter, keine Verbindungsinfo erhältlich (automatische Annahme jeglicher Clients)
- 1 = kein Filter, Verbindungsinfo erhältlich (automatische An- nahme jeglicher Clients)
- 2 = Filter-Anforderung zur Annahme des Clients, Verbindungsin- fo erhältlich

Conn_Tout: Wenn Conn_Tout (Sekunden) überschritten wird, ohne dass über den Kanal ein Telegramm empfangen wurde, wird der Kanal geschlossen (empfangsabhängig).

- 0 = keine Timeout-Kontrolle
- x = Timeout-Kontrolle alle x Sekunden

TACT↑ wenn der Befehl „OpenUDP“ für das Ethernet-TCP/IP Module in die Kommunikations-Mailbox abgelegt wurde.

TACT↓ nach dem ACK des Ethernet-TCP/IP Modules. Ein neuer Open Data Mode-Befehl kann ausgeführt werden.

NEXE↑ Wenn zum Beispiel der lokale IP Port bereits benutzt wird.

```
Beispiel:   CSF   [cc]   S.IP.Library              ; Bibliothek
                         S.IP.OpenServerTCP ; Funktion
                         10                        ; Kanal 10
                         5432                      ; Port 5432
                         F 1010                    ; Basis der Kanal-Diag
                                                   ; nose-Flags
                         R 1                       ; Kanal Diagnose-Register
                         0                         ; kein Filter
                         R 2                       ; Verbindungstimeout
```

Flags: Wenn die Firmware keine IP Open Data Mode-Unterstützung bietet, wird das Error (E) Flag gesetzt.

4.3.11 Close

Schliesst einen Kanal im IP Open Data Mode (entspricht einem „closesocket“ im Berkeley Standard).

Diese Funktion kann nur ausgeführt werden, wenn TACT auf “L“ (low) gesetzt ist.

```
CSF   [cc]   S.IP.Library              ; IP-Bibliothek
             S.IP.Close                ; Funktion Close-channel
      →      Channel                   ; Kanal-Nr. (R/K)
```

```
Beispiel:   CSF   [cc]   S.IP.Library              ; Bibliothek
                         S.IP.Close        ; Funktion
                         10                        ; Kanal 10
```

Flags: Wenn die Firmware keine IP Open Data Mode-Unterstützung bietet, wird das Error (E) Flag gesetzt.

Die vorgehend für diesen Kanal definierten Diagnose-Flags und das Diagnose-Register dürfen nach dem “Close“ nicht mehr benutzt werden bis sie wieder neu initalisiert und assigniert wurden.Der CSF “Close“ trennt die Verbindung zwischen den beiden Stationen auf PCD7.F650/1/2 sofort. Die PCD7.F655 verhält sich in diesem Fall ein wenig anders. Der Unterschied wird im folgenden Beispiel beschrieben:
Ein TCP Client verbindet sich auf einen TCP Server und sendet einTelegramm Das globale Diagnose-Flag RDATA↑ wird auf dem Server gesetzt. Der Server liest das Telegramm momentan jedoch nicht. Macht nun der Client ein “Close“, so wird auf dem Client das Socket sogleich geschlossen. Auf dem Server wird das Socket erst geschlossen, wenn das anstehende Telegramm gelesen wurde und das globale DiagnoseFlag RDATA↑ somit “low“ ist. So gehen im TCP-Server beim Schliessen der Verbindung keine Telegramme verloren.

4.3.12 ConnectTCP

Verbindet einen Client-TCP-Kanal im IP Open Data Mode mit einem Server-TCP-Kanal (entspricht einem „connect“ im Berkeley Standard). Nach diesem Aufruf kann über die TCP Verbindung kommuniziert werden.

Diese Funktion kann nur ausgeführt werden, wenn TACT und TBSY auf “L“ (low) gesetzt sind.

```
CSF  [cc]   S.IP.Library              ; IP-Bibliothek
            S.IP.ConnectTCP           ; Funktion Connect TCP-channel
       →    Channel                   ; Kanal-Nr. (R/K)
       →    RemoteIP/Node             ; IP-Adresse des abgesetzten Servers
                                      ; (R/K/X)*
       →    Remote Port               ; IP-Port abgesetzten Servers (R/K)
*) siehe Kapitel: IP-Adressen-Decodierung
```

TBSY↑ wenn der Befehl “ConnectTCP“ für das Ethernet-TCP/IP Module in die Kommunikations-Mailbox abgelegt wurde.

TACT↑ wenn der Befehl “ConnectTCP“ für das Ethernet-TCP/IP Module in die Kommunikations-Mailbox abgelegt wurde.

TBSY↓ nachdem das Ethernet-TCP/IP Module das “ConnectTCP“ aus der Kommunikations-Mailbox gelesen hat. Ein ein neuer Open Data Mode-Befehl kann ausgeführt werden.

TACT↓ nach dem ACK des Ethernet-TCP/IP Modules. Ein neuer Open Data Mode-Befehl kann ausgeführt werden.

XCON↑ wenn vom Ethernet-TCP/IP Module das “connected-Event“ zurückkommt. Jetzt steht die Verbindung zwischen dem TCP Client und dem TCP Server.

Beispiel:

```
CSF  [cc]   S.IP.Library              ; Bibliothek
            S.IP.ConnectTCP           ; Funktion
            10                        ; Kanal 10
            R5                        ; Abgesetzte IP-Adresse
            5555                      ; Port 5555
```

Flags: Wenn die Firmware keine IP Open Data Mode-Unterstützung bietet, wird das Error (E) Flag gesetzt.

Solange der CSF ConnectTCP nicht abgeschlossen ist dürfen seine Parameter “RemoteIP/Node“ und “RemotePort“ nicht verändert werden.

4.3.13 DisconnectTCP

Trennt einen TCP-Kanal im IP Open Data Mode. (entspricht einem “closesocket“ im Berkeley Standard). Diese Funktion kann auf einem TCP-Client- und -Server-Kanal ausgeführt werden.

Diese Funktion kann nur ausgeführt werden, wenn TACT und TBSY auf “L“ (low) gesetzt sind.


```
CSF  [cc]   S.IP.Library                ; IP-Bibliothek
            S.IP.DisconnectTCP  ; Funktion Disconnect TCP-channel
       →    Channel                    ; Kanal-Nr. (R/K)
       →    RemoteIP/Node              ; IP-Adresse des abgesetzten Servers
                                       ; (R/K/X)*
       →    Remote Port                ; IP-Port abgesetzten Servers (R/K)
            *) siehe Kapitel: IP-Adressen-Decodierung
```

TBSY↑ wenn der Befehl “DisconnectTCP“ für das Ethernet-TCP/IP Module in die Kommunikations-Mailbox abgelegt wurde.

TACT↑ wenn der Befehl “DisconnectP“ für das Ethernet-TCP/IP Module in die Kommunikations-Mailbox abgelegt wurde.

TBSY↓ nachdem das Ethernet-TCP/IP Module das “ConnectTCP“ aus der Kommunikations-Mailbox gelesen hat. Ein ein neuer Open Data Mode-Befehl kann ausgeführt werden.

TACT↓ nach dem ACK des Ethernet-TCP/IP Modules. Ein neuer Open Data Mode-Befehl kann ausgeführt werden.

XCON↑ wenn vom Ethernet-TCP/IP Module das “connected-Event“ zurückkommt. Jetzt steht die Verbindung zwischen dem TCP Client und dem TCP Server.

Beispiel:

```
CSF   [cc]   S.IP.Library              ; Bibliothek
             S.IP.DisconnectTCP  ; Funktion
             10                        ; Kanal 10
             R5                        ; Abgesetzte IP-Adresse
             5555                      ; Port 5555
```

Flags: Wenn die Firmware keine IP Open Data Mode-Unterstützung bietet, wird das Error (E) Flag gesetzt.

Der CSF “Disconnect“ trennt die Verbindung zwischen den beiden Stationen auf PCD7.F650/1/2 sofort und das Diagnose-Flag XCON ↓ zeigt den Status der Verbindung sogleich an. Die PCD7.F655 verhält sich in diesem Fall ein wenig anders. Der Unterschied wird im folgenden Beispiel beschrieben: Ein TCP Client verbindet sich auf einen TCP Server und sendet einTelegramm Das globale Diagnose-Flag RDATA ↑ wird auf dem Server gesetzt. Der Server liest das Telegramm momentan jedoch nicht. Macht nun der Client ein “Disconnect“, so wird auf dem Client das Diagnose-Flag XCON↓ sogleich “low“ gesetzt. Auf dem Server wird das XCON↓ erst “low“ gesetzt, wenn das anstehende Telegramm gelesen wurde und das globale Diagnose-Flag RDATA↓ somit auch “low“ ist. So gehen im TCP Server beim Trennen der Verbindung keine Telegramme verloren.

4.3.14 GetConnectionTCP

Liest die Verbindungsdaten eines TCP-Kanals im IP Open Data Mode aus

RCON↑ Zeigt eine Änderung eines TCP-Clients oder Servers an. Die Änderung kann mit dem Befehl “GetConnectionTCP“ vom Server aus gelesen werden. Die Information über den Status “Anforderung Verbindung/verbunden/getrennt“ kann nur benutzt werden, wenn im zuvor definierten TCP Server das Filter auf 1 oder 2 gesetzt wurde. “GetConnectionTCP“ kann nur auf einem TCP-Server-Kanal ausgeführt werden. Wird mit mehreren Kanälen gearbeitet, so soll auf den Kanal, welcher im Channel-Register angegeben ist, zugegriffen werden.

```
CSF  [cc]   S.IP.Library              ; IP-Bibliothek
            S.IP.GetconnectionTCP     ; Funktion Verbindungsdaten TCP-channel
       →    Channel                   ; Kanal-Nr. (R/K)
       →    RemoteIP/Node             ; IP-Adresse des abgesetzten Servers
                                      ; (R/K/X)*
       →    Remote Port               ; IP-Port abgesetzten Servers (R/K)
            Status                    ; Status der Verbindung**
```

*) siehe Kapitel: IP-Adressen-Decodierung

**) Status: 0=Client fordert Verbindung an und erwartet Annahme des Servers
1=verbunden
2=getrennt

RCON↓ Nach der Ausführung des Befehls GetconnectionTCP

Beispiel:

```
CSF  [cc]   S.IP.Library              ; Bibliothek
            S.IP.GetconnectionTCP     ; Funktion
            10                        ; Kanal 10
            R100                      ; Abgesetzte IP-Adresse
            R101                      ; Abgesetzter Port
            R102                      ; Status
```

Flags: Wenn die Firmware keine IP Open Data Mode-Unterstützung bietet, wird das Error (E) Flag gesetzt.

4.3.15 AcceptConnectionTCP

Annahme einer Verbindung auf einem Server TCP-Kanal im IP Open Data Mode
Diese Funktion kann nur in Verbindung mit einer TCP Server Station benutzt werden, welche zuvor mit der Verbindungsfilter-Option “1“ oder “2“ in “OpenServerTCP“ definiert wurde.

Diese Funktion kann nur ausgeführt werden, wenn TACT und TBSY auf “L“ (low) gesetzt sind.


```
CSF  [cc]   S.IP.Library                  ; IP-Bibliothek
            S.IP.AcceptconnectionTCP      ; Funktion Verbindungsdaten TCP-channel
       →    Channel                       ; Kanal-Nr. (R/K)
       →    Status                        ; Status der Verbindung*
            *) Status:   0=Verbindung annehmen
                         1=Verbindung ablehnen
```

Beispiel:

```
CSF  [cc]   S.IP.Library                  ; Bibliothek
            S.IP.AcceptconnectionTCP      ; Funktion
            10                            ; Kanal 10
            0                             ; Status=Verbindung an
                                          ; nehmen
```

Flags: Wenn die Firmware keine IP Open Data Mode-Unterstützung bietet, wird das Error (E) Flag gesetzt.

4.3.16 SendData

Sendet Daten über einen Kanal im IP Open Data Mode (entspricht einem “send“ für TCP oder einem “sendto“ für UDP im Berkeley Standard).

Die Funktion kann nur ausgeführt werden, wenn TBSY und TACT auf “L“ (low) gesetzt sind.


```
CSF  [cc]   S.IP.Library              ; IP-Bibliothek
            S.IP.SendData       ; Funktion Daten sendenl
       →    Channel                  ; Kanal-Nr. (R/K)
       →    RemoteIP/Node            ; IP-Adresse für das Senden von Daten
                                     ; (R/K/X)*
       →    Remote Port              ; IP-Port abgesetzten Servers (R/K)
       →    Datalength               ; Länge der zu sendenden Daten (R/K)**
       →    Data                     ; Datenpuffer (R/X/DB)
            *) siehe Kapitel: IP-Adressen-Decodierung
            **) Die max. Länge der gesendeten Daten ist 1536 Bytes
```

TACT↑ wenn der Befehl “SendData“ für das Ethernet-TCP/IP Module in die Kommunikations-Mailbox abgelegt wurde.

TBSY↑ wenn der Befehl “SendData“ für das Ethernet-TCP/IP Module in die Kommunikations-Mailbox abgelegt wurde.

TACT↓ nach dem ACK des Ethernet-TCP/IP Modules.

TBSY↓ nachdem das Ethernet-TCP/IP Module das “SendData“ aus der Kommunikations-Mailbox gelesen hat. Ein ein neuer Open Data Mode-Befehl kann ausgeführt werden.

Beispiel:

```
CSF   [cc]   S.IP.Library          ; Bibliothek
             S.IP.SendData; Funktion
             10                    ; Kanal 10
             R100                  ; Abgesetzte IP-Adresse
             R101                  ; Abgesetzter Port
             100                   ; Länge der zu sendenden Daten
             R1000                 ; Start des Sendpuffers
```

Flags: Wenn die Firmware keine IP Open Data Mode-Unterstützung bietet, wird das Error (E) Flag gesetzt.

Der Datenpuffer darf nicht verändert werden bevor der Befehl “SendData“ vollständig abgeschlossen ist.

4.3.17 SendDataRev

Sendet Daten in umgekehrter Byte-Reihenfolge über einen Kanal im IP Open Data Mode (entspricht einem “send“ für TCP oder einem “sendto“ für UDP im Berkeley Standard).

SendDataRev vertauscht die Bytes, wenn die abgesetzte Station *“Intel“*-Format aufweist (siehe nachstehende Tabelle).

Die Funktion kann nur ausgeführt werden, wenn TBSY und TACT auf “L“ (low) ge-setzt sind.


```
CSF  [cc]   S.IP.Library          ; IP-Bibliothek
            S.IP.SendDataRev      ; Funktion Daten senden
       →    Channel               ; Kanal-Nr. (R/K)
       →    RemoteIP/Node         ; IP-Adresse für das Senden von Daten
                                  ; (R/K/X)*
       →    Remote Port           ; IP-Port abgesetzten Servers (R/K)
       →    Datalength            ; Länge der zu sendenden Daten (R/K)**
       →    Data                  ; Datenpuffer (R/X/DB)
       *) siehe Kapitel: IP-Adressen-Decodierung
       **) Die max. Länge der gesendeten Daten ist 1536 Bytes
```

TACT↑ wenn der Befehl “SendData“ für das Ethernet-TCP/IP Module in die Kommunikations-Mailbox abgelegt wurde.

TBSY↑ wenn der Befehl “SendData“ für das Ethernet-TCP/IP Module in die Kommunikations-Mailbox abgelegt wurde.

TACT↓ nach dem ACK des Ethernet-TCP/IP Modules.

TBSY↓ nachdem das Ethernet-TCP/IP Module das “SendData“ aus der Kommunikations-Mailbox gelesen hat. Ein ein neuer Open Data Mode-Befehl kann ausgeführt werden.

Beispiel:

```
CSF  [cc]   S.IP.Library          ; Bibliothek
            S.IP.SendDataRev      ; Funktion
            10                    ; Kanal 10
            R100R                 ; Abgesetzte IP-Adresse
            R101                  ; Abgesetzter Port
            100                   ; Länge der zu sendenden Daten
            R1000                 ; Start des Sendpuffers
```

Flags: Wenn die Firmware keine IP Open Data Mode-Unterstützung bietet, wird das Error (E) Flag gesetzt.

Der Datenpuffer darf nicht verändert werden bevor der Befehl “SendData“ vollständig abgeschlossen ist.

4.3.18 ReceiveData

Empfängt Daten über einen Kanal im IP Open Data Mode

Diese Funktion kann nur ausgeführt werden, wenn RDATA auf “H“ (high) gesetzt ist. Wird mit mehreren Kanälen gearbeitet, so muss im Kanal-Parameter der Kanal angegeben werden, auf welchem die Daten angekommen sind (Kanal-Register). Den Inhalt des Kanal-Registers (im InitODM konfiguriert) direkt vor dem Aufruf des CSF “RevceiveData“ lesen und seine Kopie im Befehl als “Channel“ anwenden.

```
CSF  [cc]   S.IP.Library              ; IP-Bibliothek
            S.IP.ReceiveData          ; Funktion
       →    Channel                   ; Kanal-Nr. (R/K)
       ←    RemoteIP/Node             ; IP-Adresse für das Senden von Daten
                                      ; (R/K/X)*
       ←    Remote Port               ; IP-Port abgesetzten Servers (R/K)
       →    Max_ Datalength           ; Max. Grösse des Datenpuffers in Bytes
                                      ; (0=Keine Überprüfung)**
       →    Datalength                ; Länge der zu sendenden Daten (R/K)**
       →    Data                      ; Datenpuffer (R/X/DB)
            *) siehe Kapitel: IP-Adressen-Decodierung
            **) Die max. Länge der gesendeten Daten ist 1536 Bytes
```

RDATA↓ nach der Ausführung des Befehls ReceiveData

Beispiel:

```
CSF  [cc]   S.IP.Library              ; Bibliothek
            S.IP.ReceiveData          ; Funktion
            10                        ; Kanal 10
            R100R                     ; Abgesetzte IP-Adresse
            R101                      ; Abgesetzter Port
            100                       ; Länge der Daten
            R1                        ; Empfangene Daten in Bytes
            R1000                     ; Start des Sendpuffers
```

Flags: Wenn die Firmware keine IP Open Data Mode-Unterstützung bietet, wird das Error (E) Flag gesetzt.

4.3.19 ReceiveDataRev

Empfängt Daten in umgekehrter Byte-Reihenfolge über einen IP-Kanal im “Open Data Mode“.
ReceiveDataRev vertauscht die Bytes, wenn die abgesetzte Station *“Intel“*-Format aufweist (siehe nachfolgende Tabelle).
Diese Funktion kann nur ausgeführt werden, wenn RDATA auf “H“ (high) gesetzt ist. Wird mit mehreren Kanälen gearbeitet, so muss im Kanal-Parameter der Kanal angegeben werden, auf welchem die Daten angekommen sind (Kanal-Register). Den Inhalt des Kanal-Registers (im InitODM konfiguriert) direkt vor dem Aufruf des CSF “RevceiveData“ lesen und seine Kopie im Befehl als “Channel“ anwenden.CSF [cc]

```
S.IP.Library                ; IP-Bibliothek
        S.IP.ReceiveDataRev ; Funktion Daten empfangen
  →     Channel             ; Kanal-Nr. (R/K)
  ←     RemoteIP/Node       ; IP-Adresse für das Senden von Daten
                            ; (R/K/X)*
  ←     Remote Port         ; IP-Port abgesetzten Servers (R/K)
  →     Max_ Datalength     ; Max. Grösse des Datenpuffers in Bytes
                            ; (0=Keine Überprüfung)**
  →     Datalength          ; Länge der empfangenen Daten (R/K)**
  →     Data                ; Empfangsdatenpuffer (R/X/DB)
```

*) siehe Kapitel: IP-Adressen-Decodierung
**) Die max. Länge der gesendeten Daten ist 1536 Bytes

RDATA↓ nach der Ausführung des Befehls ReceiveData

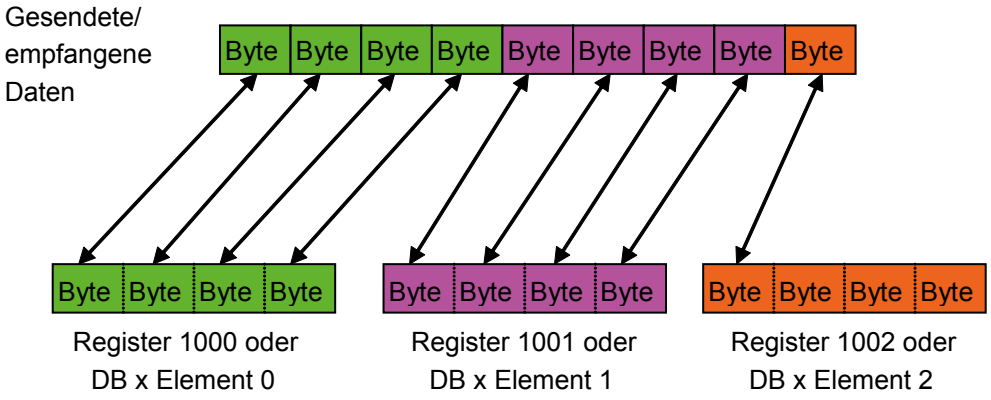
Beispiel:

```
CSF  [cc]  S.IP.Library                ; Bibliothek
           S.IP.ReceiveDataRev ; Funktion
           10                          ; Kanal 10
           R100R                       ; Abgesetzte IP-Adresse
           R101                        ; Abgesetzter Port
           100                         ; Länge der Daten
           R1                          ; Empfangene Daten in
                                       ; Bytes
           R1000                       ; Start des Empfangspuff-
                                       ; ers
```

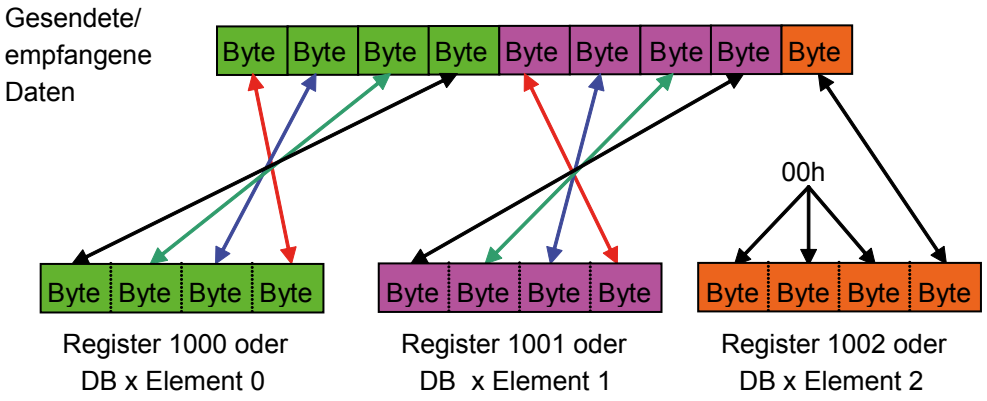
Flags: Wenn die Firmware keine IP Open Data Mode-Unterstützung bietet, wird das Error (E) Flag gesetzt.

4.3.20 Byte Swapping

Beispiel: Es sind 9 Bytes zu senden/ zu empfangen.
→ Ohne Byte Swapping (SendData / ReceiveData):



→ Mit Byte Swapping (SendData / ReceiveData):



Wenn sich Text im Puffer befindet, werden die Bytes nie vertauscht!

4.3.21 IP-Adressen-Decodierung

Die IP-Adresse kann als Wert in einem Text, Register oder einer Konstanten eingegeben werden. Die IP-Adresse kann auch einen Node in einem Register oder einen konstanten Wert bezeichnen. Ein konstanter Wert kann nur ein Node sein.

IP address in text:

Im Medium “Text“, wird die IP-Adresse als Text in der Form von 4 Dezimalzahlen dargestellt, die durch Punkte getrennt sind, z.B. „192.168.12.14“

IP address in register:

Die IP-Adresse kann auch als Medium “Register“ dargestellt werden. Ist das höhere Wort der 4 Bytes Null, dann wird ein Node dargestellt. Ist das höhere Wort der 4 Bytes verschieden Null, dann codiert das „Register“ eine Adresse und ist als 4 hexadezimale Zahlen codiert:

aa	bb	cc	dd

Die IP-Adresse wird in der Empfangs- und Verbindungs-Information dieses obenbeschriebene Format aufweisen.

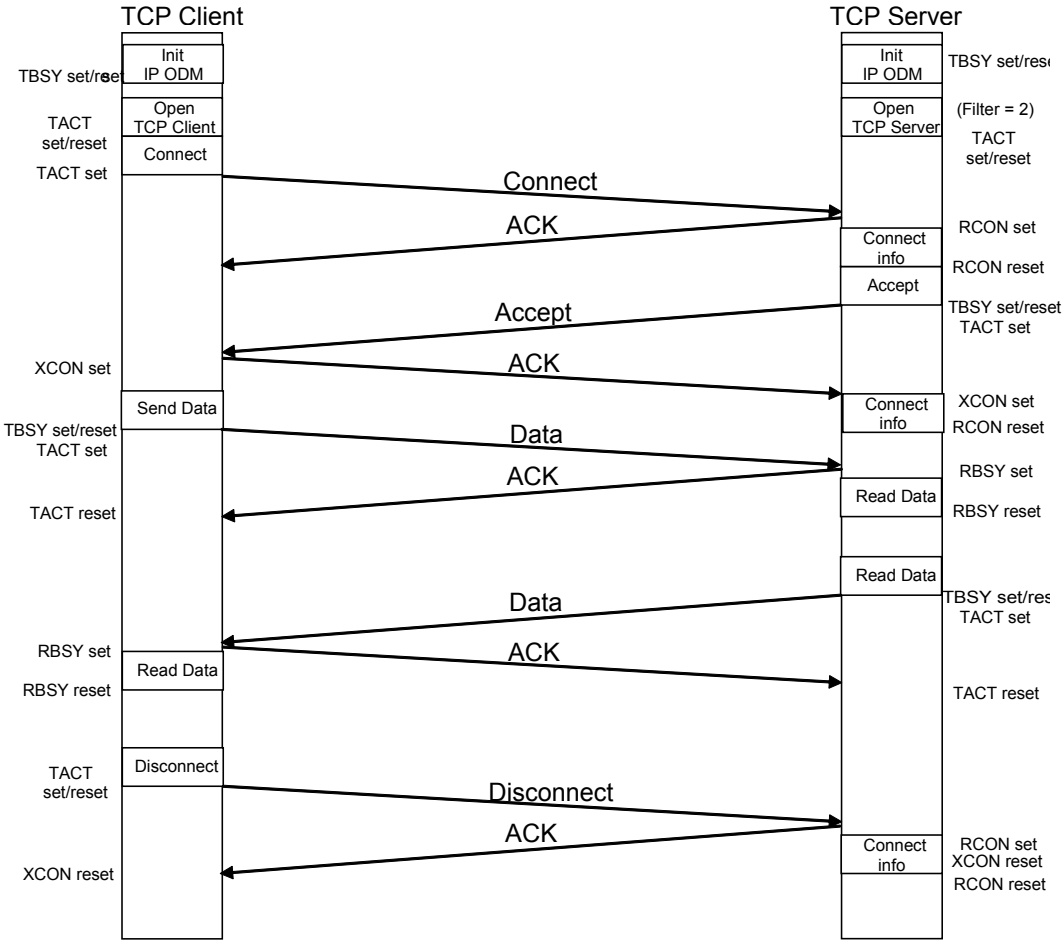
Beispiel: 0C0A80C0Eh für die IP-Adresse 192.168.12.14

IP address in constant:

Codiert als Medium “Konstante“ stellt die IP-Adresse immer die Node-Nummer dar.

4.3.22 Typischer TCP-Verbindungsfluss

Verbindung eines TCP Clients (172.16.1.142) auf einen TCP Server (172.16.1.141) und Verschicken von zwei Telegrammen.



Oder in der Ansicht eines Ethernet Analyzers (in diesem Fall Wireshark). Hier wurde nur ein Telegramm verschickt.

No.	Time	Source	Destination	Protocol	Info
1	0.000000	172.16.1.142	172.16.1.141	TCP	5050 > 5050 [SYN] Seq=0 Ack=0 Win=5840 Len=0 MSS=1460
2	0.001275	172.16.1.141	172.16.1.142	TCP	5050 > 5050 [SYN, ACK] Seq=0 Ack=1 Win=720 Len=0 MSS=1460
3	0.002324	172.16.1.142	172.16.1.141	TCP	5050 > 5050 [ACK] Seq=1 Ack=1 Win=720 Len=0
4	8.359572	172.16.1.142	172.16.1.141	TCP	5050 > 5050 [PSH, ACK] Seq=1 Ack=1 Win=720 Len=4
5	8.733767	172.16.1.141	172.16.1.142	TCP	5050 > 5050 [ACK] Seq=1 Ack=5 Win=720 Len=0
6	17.921981	172.16.1.142	172.16.1.141	TCP	5050 > 5050 [FIN, ACK] Seq=5 Ack=1 Win=720 Len=0
7	17.923051	172.16.1.141	172.16.1.142	TCP	5050 > 5050 [ACK] Seq=1 Ack=6 Win=720 Len=0
8	17.924448	172.16.1.141	172.16.1.142	TCP	5050 > 5050 [FIN, ACK] Seq=1 Ack=6 Win=720 Len=0
9	17.925515	172.16.1.142	172.16.1.141	TCP	5050 > 5050 [ACK] Seq=6 Ack=2 Win=720 Len=0

4.4 Zusätzliche CSFs

Wie bereits im Open Data Mode aufgeführt gibt es für spezielle Funktionalitäten im Zusammenhang mit der PCD7.F65x “Call System Functions“. Hier nun die Auflistung der zusätzlichen CSFs.

4.4.1 CSF NA-Reset

Dieser CSF löst einen Hard-Reset der PCD7.F65x aus. Dies kommt einem Power Off/On der PCD7.F65x gleich.

```
CSF     [cc]    S.IP.Library            ; IP-Bibliothek
                S.IP.NAReset  ; Funktion NAReset
        ←       Status                  ; Status der PCD7.F65x (R)
```

Der Status kann folgende Werte annehmen:

0=Während der Ausführung des Resets
1=Reset ist ausgeführt und die PCD7.F65x ist wieder bereit
2=PCD7.F65x ist nicht präsent, keine Signatur verfügbar
3=Status-Register ist ausserhalb der Register-Limiten
4=Timeout in der Mailbox-Kommunikation
5=Firmware ist zu alt / nicht kompatibel.

4.4.2 CSF SetLocalIPNode

Dieser CSF konfiguriert die PCD7.F65x auf eine neue IP-Konfiguration

```
CSF  [cc]    S.IP.Library                ; IP-Bibliothek
             S.IP.SetLocalIPNode  ; Funktion SetLocalIPNode
       →     IP_Address                  ; Neue IP-Adresse der PCD7.F65x (R)
       →     Subnet_Mask                 ; Neue Subnet Maske der PCD7.F65x (R)
       →     Default_Gateway             ; Neues Default Gateway der PCD7.F65x
                                         ; (R)
```

Die neue IP-Adresse wird auf der PCD7.F650/1/2 ohne Power Off / On aktualisiert. Um mit einer neuen Subnet Maske und einem neuen Default Gateway arbeiten zu können, muss die PCD7.F650/1/2 neu gestartet werden.
Auf der PCD7.F655 können alle drei IP-Parameter während des Betriebs geändert und aktualisiert werden.

4.4.3 CSF IPPhyConfig

Dieser CSF konfiguriert den Ethernet Physical Chip auf der PCD7.F65x.

```
CSF  [cc]   S.IP.Library          ; IP-Bibliothek
            S.IP.IPPhyConfig      ; Funktion IPPhyConfig
       →    Autonegotiate         ; Ob Autonegotiate Mode, TRUE/FALSE (R)
       →    Speed                 ; Kommunikationsgeschw., 10/100 Mb/s (R)
       →    Duplex_mode           ; Duplex Modus, Full-/Half Duplex (R)
       ←    Status                ; Status Fehlermeldung (R)
```

Es ist möglich, die Konfiguration des Physical Chip auf der PCD7.F65x mit einem CSF zu verändern. Standardmässig arbeitet der Physical Chip im Autonegotiation Modus. Das heisst, er ist fähig in 10Mb/s oder 100Mb/s, Half-Duplex oder Full-Duplex Mode zu arbeiten.

Mögliche Konfigurationen des Physical Chips sind entweder Autonegotiate Mode oder Modi mit fixierter Kommunikationsgeschwindigkeit und fixiertem Duplex Modus. Der CSF gibt einen Status zurück.

Autonegotiate:	“1”	um den Physical Chip in den Autonegotiation Modus zu setzen
	“0”	um nicht im Autonegotiation Modus zu arbeiten
Speed:	“1”	um den Physical Chip auf 10Mb/s zu fixieren
	“2”	um den Physical Chip auf 100Mb/s zu fixieren
	“3”	der Physical Chip sucht sich selber die passende Kommunikationsgeschwindigkeit (10 oder 100Mb/s). Ist nur im Autonegotiation Modus erlaubt.
Duplex Mode	“1”	um den Physical Chip auf Halb-Duplex Modus zu fixieren
	“2”	um den Physical Chip auf Voll-Duplex Modus zu fixieren
	“3”	der Physical Chip sucht sich selber den passenden Kommunikationsmodus (Halb- oder Voll-Duplex) aus. Ist nur im Autonegotiation Modus erlaubt
Error Status	“0”	kein Fehler ist aufgetreten
	“1”	ein Fehler ist aufgetreten ● im Autonegotiation Modus: Autonegotiation war nicht erfolgreich ● im Nicht-Autonegotiation Modus: es war nicht möglich den Physical Chip in die entsprechende Konfiguration zu versetzen.

Parameter “Autonegotiate“ = “1“

Wenn der Parameter Autonegotiate gesetzt ist, muss dem Physical Chip mitgeteilt werden, welche möglichen Kommunikationsgeschwindigkeiten und Kommunikations-Modi unterstützen werden sollen. Somit wird sein “Advertisement Capability Register“ konfiguriert.

Der Inhalt des Kastens gibt die Konfiguration des „Advertisement Capability Register“ an und definiert welche Kommunikations-Modi erlaubt sind		Wert des Parameters für die Kommunikationsge-schwindigkeit		
		1	2	3
Wert des Pa- rameters für denDuplex Modus	1	10 Mb/s, Halb Duplex, fixiert	100 Mb/s, Halb Duplex, fixiert	10 Mb/s + 100 Mb/s, Halb-Duplex erlaubt
	2	10 Mb/s, Voll Duplex, fixiert	100 Mb/s, Voll Duplex, fixiert	10 Mb/s + 100 Mb/s, Voll- Duplex erlaubt
	3	10 Mb/s, Halb- + Voll Duplex erlaubt	100 Mb/s, Halb- + Voll Duplex erlaubt	10 Mb/s + 100 Mb/s, Halb- + Voll Duplex erlaubt

Nachdem der Physical Chip per CSF in den Autonegotiation Modus gesetzt wurde, macht die PCD7.F65x einen Autonegotiations Zyklus mit der Station am anderen Ende des Kabels um den Link korrekt zu installieren. Wenn die Station am anderen Ende des Kables kein Autonegotiation unterstützt, wird die Kommunikation nicht sicher und stabil laufen und es kann Probleme geben. Deshalb gibt der CSF in diesem Fall eine Fehlermeldung mit dem “Status“ zurück. Wenn die Station am anderen Ende des Kabels jedoch Autonegotiate unterstützt, dann handeln sich die beiden Geräte den bestmöglichsten Kommunikationsmodus aus. Nach einem Verlust des physikalischen Links und einem neuen Link (wenn die PCD7.F65x zum Beispiel an einen anderen Port des Switchs gehängt wird), wird ein erneuter Autonegotiation Zyklus gestartet.

Parameter “Autonegotiate“ = „0“

Wenn der Parameter Autonegotiate nicht gesetzt ist, dann muss der Physical Chip in eine klare und fixe Konfiguration gesetzt werden. Es kann nur 10Mb/s oder 100Mb/s und Halb- **oder** Voll-Duplex eingestellt werden.

Der Inhalt des Kastens gibt die Konfiguration des ”Advertisement Capability Register“ an und definiert welche Kommunikations-Modi erlaubt sind		Wert des Parameters für die Kommunikationsge-schwindigkeit		
		1	2	3
Wert des Pa- rameters für denDuplex Modus	1	10 Mb/s, Halb Duplex, fixiert	100 Mb/s, Halb Duplex, fixiert	nicht erlaubt
	2	10 Mb/s, Voll Duplex, fixiert	100 Mb/s, Voll Duplex, fixiert	nicht erlaubt
	3	nicht erlaubt	nicht erlaubt	nicht erlaubt

Nachdem der Physical Chip per CSF in einen fixen Modus gesetzt wurde, macht die PCD7.F65x keinen Autonegotiation Zyklus mit der Station am anderen Ende des Kabels. Es ist aber unbedingt notwendig, dass die Station am anderen Ende des Kabels dieselbe Konfiguration besitzt wie jene des lokalen Physical Chips. Sonst kann keine stabile Kommunikationsweise garantiert werden.

4.4.4 CSF SendEtherSBUS

Dieser CSF ist wie ein STXM Befehl (Senden von Medien). Anstatt des IP-Nodes wird direkt die IP-Adresse der Station angegeben, welche die Medien erhalten soll. Sendet zu einer Slave-Station die Quellenelemente der eigenen PCD (Master).

```
CSF  [cc]    S.IP.Library                ; IP-Bibliothek
             S.IP.SendEtherSBUS ; Funktion SendEtherSBUS
        →    Channel                     ; Kanal-Nr. (R/K)
        →    RemoteIP                    ; Abgesetzte IP-Adresse für das Senden
                                         ; von Daten (R/K)*
        →    SBUSAddr                    ; Abgesetzte S-Bus Adresse (R)
        →    Count                       ; Anzahl Elemente
        →    Source                      ; Quelle I/O/F/R/T/C/DB/K
        →    Destination                 ; Ziel I/O/F/R/T/C/DB/K
        ←    Status                      ; Status Fehlermeldung (R)
```

* Siehe Kapitel: IP-Adressen-DecodierungCount 1...32 Anzahl Elemente R/T/C welche gesendet werden 1...128 Anzahl Elemente I/O/F welche gesendet werden

	0	Spezial Funktions-Code. Hinweise im “Befehlssatz für die PCD-Familie“ oder PG5-Hilfe beachten
Source	0...8191	Basis Adresse der Elemente I/O/F in der Master PCD
	0...4095	Basis Adresse der Elemente R in der Master PCD
	0...1599	Basis Adresse der Elemente T/C in der Master PCD
	0...7999	Basis Adresse der Elemente DB in der Master PCD
	4000	K, Spezial Funktions-Code. Hinweise im “Befehlssatz für die PCD-Familie“ oder PG5-Hilfe beachten
Destination	0...8191	Basis Adresse der Elemente I/O/F in der Slave PCD
	0...4095	Basis Adresse der Elemente R in der Slave PCD
	0...1599	Basis Adresse der Elemente T/C in der Slave PCD
	0...7999	Basis Adresse der Elemente DB in der Slave PCD
	1000	K, Schreibe Uhr in der Slave PCD
	17, 18, 19	K, Spezial Funktions-Code. Hinweise im “Befehlssatz für die PCD-Familie“ oder PG5-Hilfe beachten
Status	0...4095	Status des CSF in R 0 = Kein Fehler 1 = Fehler im Gebrauch des Kanals oder IP-Module nicht präsent. 2 = Master nicht assigniert 3 = Fehler in der Übersetzung der IP-Adresse 4 = Diese Broadcast-Art ist nicht erlaubt 5 = Media Fehler

Beispiel:

```
$INCLUDE SNetLib.inc      ; Einbinden der benötigten SNET Bibliothek
        SASI   9          ; SASI Master wie für das traditionelle S-Bus mit
                          ; Diagnose
        SASI_Master       ; Text mit Diagnose Flags und Register
        LD     R 10       ; Abgesetzte IP-Adresse ist 192.168.12.95
        0C0A80C5CH
        LD     R 20       ; Abgesetzte S-Bus-Adresse ist 95
        95

        CSF    S.SNET.Library
               S.SNET.SendEtherSBUS
               9                ; Kanal 9
               R 10             ; Abgesetzte IP-Adresse
               R 20             ; Abgesetzte S-Bus-Adress
               4                ; 4 Elemente
               R 0              ; Ab Register 0...3 der Master Station
               R 0              ; Auf Register 0...3 auf 192.168.12.95, S-Bus 95
               R 30             ; Status
```

Die folgende Tabelle zeigt welche Elemente von der lokalen Quellstation auf die dementsprechenden Elemente der Zielstation kopiert werden können.

		Slave PCD, Destination						
		O	F	R	C	T	DB	Clock
Master PCD, Source	I	x	x					
	O	x	x					
	F	x	x					
	R			x	x	x	x	x
	C			x	x	x	x	
	T			x	x	x	x	
	DB			x	x	x		

4.4.5 CSF RecvEtherSBUS

Dieser CSF ist wie ein SRXM Befehl (Empfangen von Medien). Anstatt des IP-Nodes wird direkt die IP-Adresse der Station von welcher die Medien gelesen werden soll angegeben. Liest von der Partner-Station die Quellelemente und lädt diese in der eigenen PCD auf die Zielelemente.

```
CSF  [cc]    S.IP.Library                ; IP-Bibliothek
             S.IP.RecvEtherSBUS ; Funktion RecvEtherSBUS
        →    Channel                     ; Kanal-Nr. (R/K)
        →    RemoteIP                    ; Abgesetzte IP-Adresse für das Lesen
                                         ; von Daten (R/K)*
        →    SBUSAddr                    ; Abgesetzte S-Bus Adresse (R)
        →    Count                       ; Anzahl Elemente
        →    Source                      ; Quelle I/O/F/R/T/C/DB/K
        →    Destination                 ; Ziel I/O/F/R/T/C/DB/K
        ←    Status                      ; Status Fehlermeldung (R)
        * Siehe Kapitel: IP-Adressen-Decodierung
```

Count	1...32	Anzahl Elemente R/T/C welche gesendet werden
	1...128	Anzahl Elemente I/O/F welche gesendet werden
	0	Spezial Funktions-Code. Hinweise im “Befehlssatz für die PCD-Familie“ oder PG5-Hilfe beachten
	R nnnn	Wird benutzt für Data Block Transfer
Source	0...8191	Basis Adresse der Elemente I/O/F in der Slave PCD
	0...4095	Basis Adresse der Elemente R in der Slave PCD
	0...1599	Basis Adresse der Elemente T/C in der Slave PCD
	0...7999	Basis Adresse der Elemente DB in der Slave PCD
	6000	K, Spezial Funktions-Code. Hinweise im “Befehlssatz für die PCD-Familie“ oder PG5-Hilfe beachten
Destination	0...8191	Basis Adresse der Elemente I/O/F in der Master PCD
	0...4095	Basis Adresse der Elemente R in der Master PCD
	0...1599	Basis Adresse der Elemente T/C in der Master PCD
	0...7999	Basis Adresse der Elemente DB in der Master PCD
Status	0...4095	Status des CSF in R 0 = Kein Fehler 1 = Fehler im Gebrauch des Kanals oder IP-Module nicht präsent. 2 = Master nicht assigniert 3 = Fehler in der Übersetzung der IP-Adresse 4 = Diese Broadcast-Art ist nicht erlaubt 5 = Media Fehler

Beispiel:

```
$INCLUDE SNetLib.inc    ; Einbinden der benötigten SNET Bibliothek
        SASI  9         ; SASI Master wie für das traditionelle S-Bus mit
                        ; Diagnose
        SASI_Master     ; Text mit Diagnose Flags und Register
        LD    R 10      ; Abgesetzte IP-Adresse ist 192.168.12.95
        0C0A80C5CH
        LD    R 20      ; Abgesetzte S-Bus-Adresse ist 95
        95
```

```
CSF   S.SNET.Library
      S.SNET.RecvEtherSBUS
      9                 ; Kanal 9
      R 10              ; Abgesetzte IP-Adresse
      R 20              ; Abgesetzte S-Bus-Adress
      4                 ; 4 Elemente
      R 0               ; Ab Register 0...3 der Master Station
      R 0               ; Auf Register 0...3 auf 192.168.12.95, S-Bus 95
      R 30              ; Status
```


Die folgende Tabelle zeigt welche Elemente von der lokalen Quellstation auf die dementsprechenden Elemente der Zielstation kopiert werden können.

		Master PCD, Destination					
		O	F	R	C	T	DB
Slave PCD, Source	I	x	x				
	O	x	x				
	F	x	x				
	R			x	x	x	x
	C			x	x	x	x
	T			x			x
	K						
	DB			x	x	x	

4.5 Ethernet-TCP/IP-Fehlermeldungen

Fehlermeldung	**Beschreibung:**	**Massnahme**
IPM NOT PRESENT	Es ist eine IP-Konfiguration, aber kein IP-Modul vorhanden.	Ein IP-Modul am konfigurierten Steckplatz einstecken oder IP- Konfiguration löschen.
IPM DOES NOT RESTART	Die PCD hat einen Restart ausgeführt, aber das IP-Modul antwortet nicht.	Speisung aus- und wieder einschalten.
IPM HAS OLD FW	Die FW des IP-Moduls ist zur PCD FW nicht kompatibel.	Update der FW im IP-Modul ausführen.
IP FAIL SASITEXT	Im SASI-Text ist ein Fehler vorhanden	SASI-Text überprüfen.
IP FAIL SASI DBX	In der Knotenliste-Konfigurations-dbx ist ein Fehler vorhanden	Knoten-Konfiguration überprüfen Achtung: In der Knotenliste muss mindestens ein IP-Knoten vorhanden sein! IP-Modul konfigurieren und in die PCD einstecken.
IP FAIL NO IPM	Eine IP-Funktion wurde ausgeführt; es ist jedoch keine IP-Konfiguration oder kein IP-Modul vorhanden.	

5 Diagnose, Fehlersuche und -behebung

5.1 Zusammenfassung

Jeder Ansatz für die Fehlersuche und-behebung umfasst zwei wesentliche Phasen, die Diagnose-Phase und die Phase, in der die Lösung implementiert wird. Die Analyse des Problems umfasst das schrittweise Eingrenzen potenzieller Ursachen eines Fehlers, indem Fragen gestellt werden, die diese Ursachen eliminieren können. Es ist wichtig, die Ursachen aller Fehler zu isolieren.

5.2 TCP/IP-Kommunikationsprozess

Der Kommunikationsprozess zwischen zwei PCD-Stationen in einem Netz kann in drei Stufen unterteilt werden. Wenn ein Telegramm von einer PCD zu einer andern gesendet wird, durchläuft es diese Stufen wie folgt.

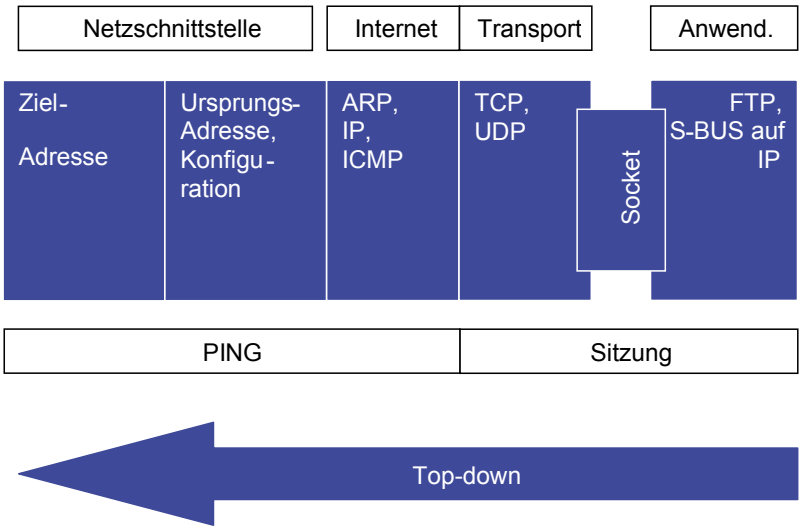
- Umsetzung der Kennzeichnung des IP-Knotens in die lokale IP-Adresse (konfi guriert in den HW-Einstellungen).
- Umsetzung der Übertragungs-IP-Adresse in eine MAC-Adresse (durch die ARP-Tabelle). Beim Senden von Broadcast-Meldungen wird immer die MAC-Adresse im folgenden Format benutzt:

  0xFF-FF-FF-FF-FF-FF
- Übertragung des IP-Datagramms an die vom ARP gelieferte MAC-Adresse.

Jeder TCP/IP-Stack folgt beim Senden eines Telegramms von einem Punkt zu einem andern dieser Sequenz.

5.3 Top-down-Direktive

Bei der Fehlersuche in TCP/IP empfehlen wir, mit der obersten Ebene (Anwendungsschicht) zu beginnen und schrittweise nach unten zu den tieferen IP-Schichten zu gehen. Die Fehlersuche und -behebung umfasst die Prüfung, ob die Protokolle jeder Schicht mit den Schichten ober- und unterhalb dieser Schicht kommunizieren können.



Nachdem das Ethernet-TCP/IP-Modul in den PG5-HW-Einstellungen konfiguriert wurde (IP-Adresse, Subnet-Maske, Default-Gateway, Stationsnummer (S-Bus-Adresse) und IP-Knoten), erfolgt die Fehlersuche in zwei Stufen:

1) Eine Verbindung zwischen einem PG5-Client und einem PCD-Slave aufbauen.

Verbinden des PG5 Online Debuggers:

- in den PG5 Online Settings den IP Channel Type SOCKET auswählen.
- Einstellungen ändern (nur bei Bedarf)
- PG5 Online Debugger online bringen.

Wenn dies funktioniert, wurde eine Verbindung zwischen der Netz-Schnittstellen-Schicht und der Anwendungsschicht aufgebaut. Alle dazwischen notwendigen Protokolle arbeiten einwandfrei.

Wenn es nicht funktioniert, ist die Verbindung zwischen der Netz-Schnittstellen-Schicht und der Internet-Schicht zu überprüfen.

2) Der Top-down-Direktive folgen und mit dem PING-Befehl überprüfen, ob das ARP korrekt arbeitet, d.h.ob für jede Echo-Anforderung und jede Antwort auf eine Echo-Anforderung die IP-Adresse in die richtige MAC-Adresse umgesetzt wird.

5.3.1 PING

Wenn ein PING-Befehl (Echo-Anforderung) erfolgreich ist, bedeutet dies, dass TCP/IP auf dem Host-Computer und auf dem Ethernet-TCP/IP-Modul korrekt installiert ist, und dass die ARP-Cache-Liste die PCD-Stationen enthält.

Beispiele:

PING 127.0.0.1	“Look up address“ prüft, ob der TCP/IP-Stack auf dem Host-Computer korrekt installiert ist. Die Echo-Anforderung bleibt im lokalen TCP/IP-Stack.
PING 192.168.12.60	Eigene IP-Adresse: Prüft, ob die Station korrekt in die ARP-Cache-Tabelle (für die Umsetzung der IP-Adresse in die MAC-Adresse verwendet) eingetragen wurde.

PING 192.168.12.60	IP-Adresse der abgesetzten PCD-Einheit: Prüft, ob der TCP/IP-Stack (einschliesslich ARP-Cache) auf der abgesetzten PCD-Station korrekt installiert wurde.
PING 192.168.12.60	Die IP-Adresse einer PCD-Station lässt sich nur über einen Router erreichen: Prüft, ob die konfigurierte Default-Gateway-Adresse stimmt.

PING <IP-Address> -t	Sendet kontinuierlich Echo-Anforderungen
PING <IP-Address> -n 10	Sendet 10 aufeinanderfolgende Echo-Anforderungen
PING <IP-Address> -L 320	Sendet eine Anforderung mit 320 Bytes Länge

```
C:\WINNT\system32>ping 192.168.12.60

Pinging 192.168.12.60 with 32 bytes of data:

Reply from 192.168.12.60: bytes=32 time<10ms TTL=30
Reply from 192.168.12.60: bytes=32 time<10ms TTL=30
Reply from 192.168.12.60: bytes=32 time<10ms TTL=30
Reply from 192.168.12.60: bytes=32 time<10ms TTL=30
```

Time	Reaktionszeit für die Echoanforderung
TTL	Time-to-live (Gültigkeitsdauer):Diese Zahl wird bei jedem Durchgang durch einen Switch oder Router dekrementiert. Sobald die TTL null erreicht, wird das Telegramm nicht mehr weiter gesendet.

Wenn ein PING-Befehl nicht funktioniert, ist die ARP-Cache-Liste zu überprüfen, um festzustellen, ob die PCD-Station enthalten und ob die Adressumsetzung möglich ist

(IP-Adresse in MAC-Adresse).

5.3.2 ARP

Der ARP-Befehl listet den Inhalt der ARP-Cache-Tabelle auf dem lokalen Host-Computer auf. Ein ARP-Eintrag in der Cache-Tabelle enthält die abgesetzten IP- und MAC-Adressen.

Wenn zwei Stationen im selben Subnetz keine PING-Anforderungen austauschen können, ist zu verifizieren, dass die Station in der ARP-Cache-Tabelle aufgeführt ist.

ARP -a	Zeigt alle Einträge im ARP-Cache an
ARP -d <IP-Address>	Löscht einen Eintrag im ARP-Cache.

```
C:\WINNT\system32>arp -a

Interface: 192.168.12.46 on Interface 2
  Internet Address      Physical Address      Type
  192.168.12.60         00-50-c2-0c-50-24     dynamic
  192.168.12.200        00-10-83-f9-1d-fe     dynamic
  192.168.12.202        00-30-6e-00-03-50     dynamic
  192.168.12.221        00-80-3e-7f-15-fa     dynamic
```

5.3.3 Weitere nützliche Befehle für den Host-Computer

IPCONFIG Zeigt die TCP/IP-Konfigurationsparameter des Host-Computers an

```
C:\WINNT\system32>ipconfig

Windows NT IP Configuration

Ethernet adapter E100B1:

        IP Address. . . . . . . . . . : 192.168.12.46
        Subnet Mask . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . : 192.168.12.221

C:\WINNT\system32>_
```

```
C:\WINNT\system32>ipconfig/all

Windows NT IP Configuration

        Host Name . . . . . . . . . . : ch01w234.saia-burgess.ch
        DNS Servers . . . . . . . . . : 192.168.6.204
                                        192.168.6.28
        Node Type . . . . . . . . . . : Hybrid
        NetBIOS Scope ID. . . . . . . :
        IP Routing Enabled. . . . . . : No
        WINS Proxy Enabled. . . . . . : No
        NetBIOS Resolution Uses DNS   : No

Ethernet adapter E100B1:

        Description . . . . . . . . . : Intel(R) PRO Adapter
        Physical Address. . . . . . . : 00-90-27-BC-ED-8A
        DHCP Enabled. . . . . . . . . : No
        IP Address. . . . . . . . . . : 192.168.12.46
        Subnet Mask . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . : 192.168.12.221
        Primary WINS Server . . . . . : 192.168.6.204
        Secondary WINS Server . . . . : 192.168.6.209
```

NETSTAT Zeigt die Statistiken und Protokolle von vorhandenen TCP/IP-Verbindungen auf dem Host-Computer an

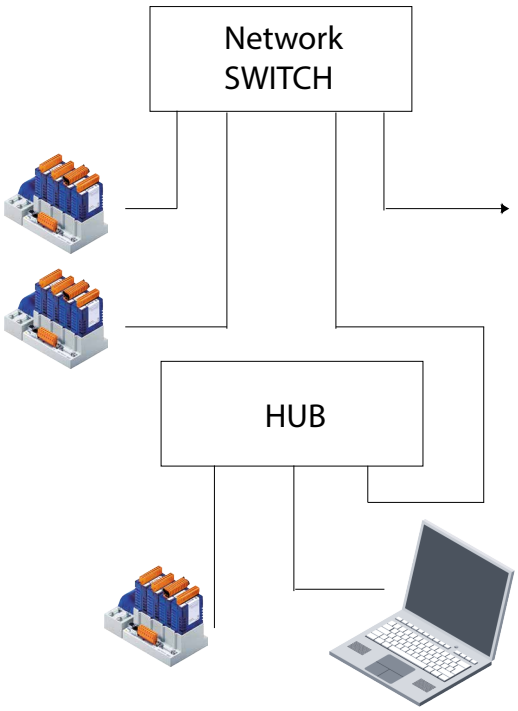
NETSTAT -a	Zeigt in einer Liste alle aktiven Verbindungen und Ports an
NETSTAT -r	Zeigt in einer Liste den Inhalt der Leitweg-Tabelle an
NETSTAT -p UDP	Zeigt eine Liste mit allen aktiven UDP-Verbindungen an
NETSTAT -p UDP	Zeigt eine Liste mit allen aktiven TCP-Verbindungen an
NETSTAT -p ICMP	Zeigt eine Liste mit allen aktiven ICMP-Verbindungen an

5.4 Ethernet Protocol Analyzer Wireshark

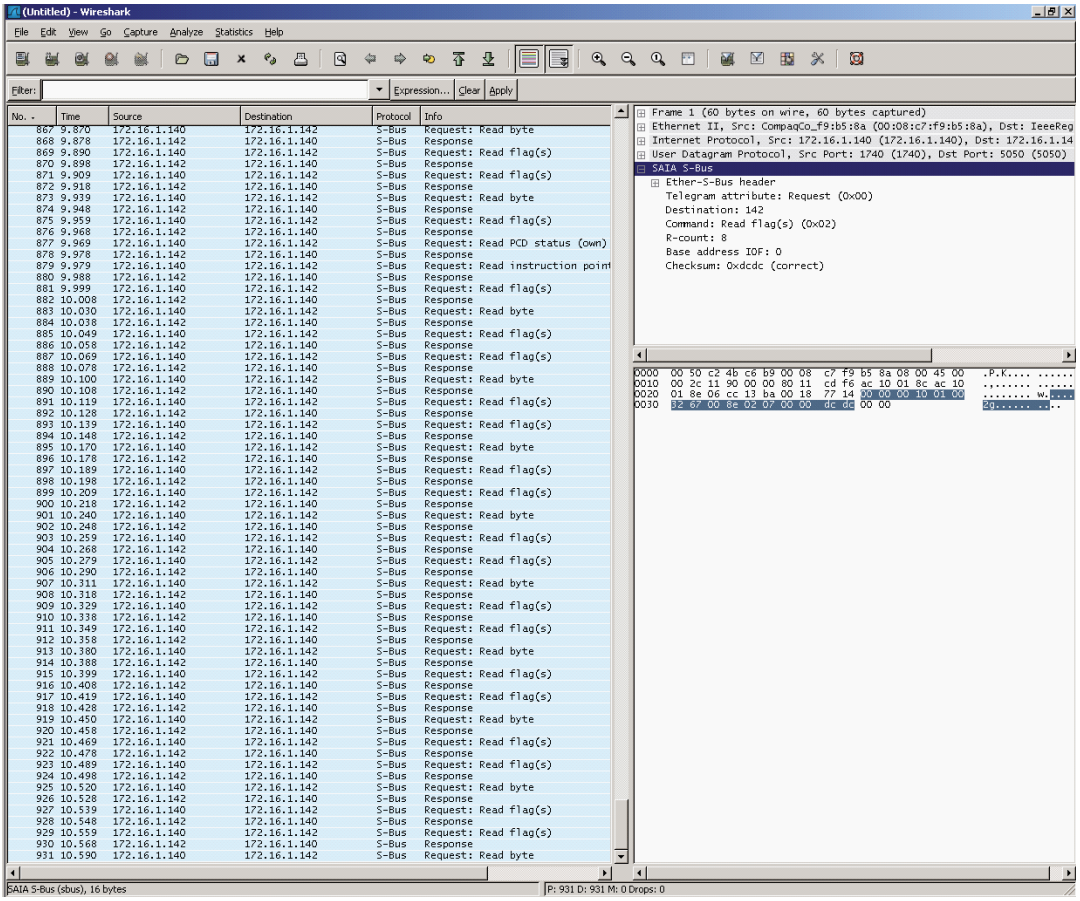
Um die proprietäre Ether-S-Bus Kommunikation und generelle Ethernet Kommunikation zu analysieren empfehlen wir die Benutzung der Freeware Wireshark (www.wireshark.org).

Dies ist ein Open Source Analyzer inklusive dem Ethernet-Karten Logger WinPcap. Mit diesem Programm gemachte Ethernet-Aufzeichnungen können wir analysieren und interpretieren. Der Wireshark Analyzer, ab Version 1.1, interpretiert auch gleich die Ether-S-Bus Telegramme. Genauere Informationen zu diesem Thema entnehmen Sie bitte der betreffenden FAQ #100569.

Um möglichst die ganze Kommunikation einer betroffenen Station aufzuzeichnen, wird ihr ein Hub vorgeschaltet. Der Analyzer horcht dann an diesem Hub. Switches kanalisieren die Kommunikation und eignen sich nicht für Analysen und Aufnahmen.



Dies ist ein interpretiertes S-Bus UDP Telegramm (Red Flags), aufgenommen in Wireshark

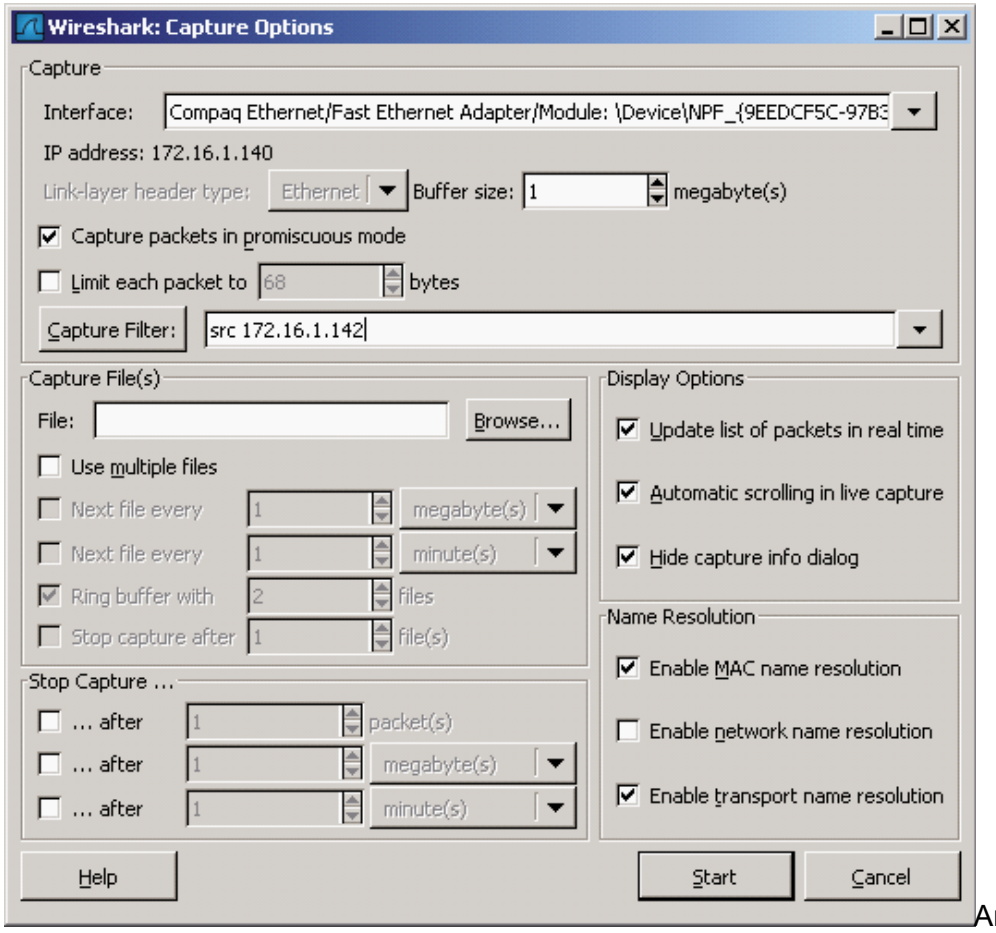


Auch andere Protokolle, wie zum Beispiel ModBus TCP oder ModBus UDP, werden interpretiert angezeigt.

Wireshark bietet sehr nützliche Filterfunktionen. Sie können direkt beim Datenloggen, oder später, beim Analysieren, angewandt werden. Oftmals ist es ratsam die Kommunikation ohne “Capture Filter aufzuzeichnen und die Filter später, als “Display Filter”, bei der Analyse einzusetzen.

Hierzu einige Beispiele:

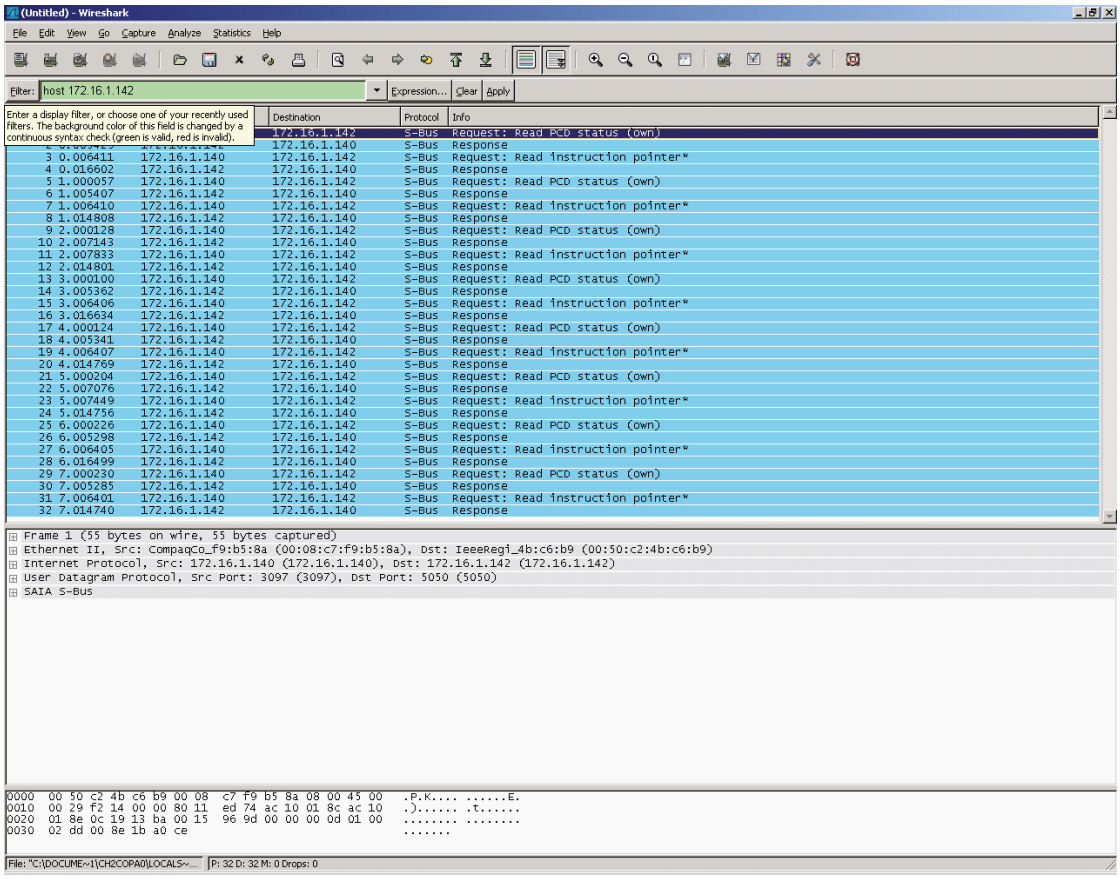
“Capture Filter”-nur die Telegramme, welche die Filterbedingung erfüllen, werden auch aufgezeichnet. Der Capture Filter kann unter “Capture → Options...” oder mit der Tastenkombination “Ctrl+K” aufgerufen werden. Mit der Einstellung unten im Bild werden alle Telegramme abgehend oder ankommend auf die IP-Adresse 172.16.1.142 aufgezeichnet.



Andere Aufzeichnungsfilter sind zum Beispiel:

Host 172.18.5.4	zeichnet nur den Verkehr von der oder zur IP-Adresse 172.18.5.4 auf
Port 5050	zeichnet den Verkehr über Port 5050 auf
SRC 172.18.5.4	zeichnet alle von dieser IP-Adresse abgehenden Telegramme auf
DST Port 135 und TCP Port 135	zeichnet den Verkehr auf Zielport 135 auf in TCP

“Display Filters”-dieser Filter wird nur auf die Anzeige angewandt und verändert den Inhalt der Log-Datei nicht. Es sind Sortierfilter. Der Display-Filter wird direkt im Hauptfenster definiert.



Beispiele von Display-Filtern:

Udp Tcp	zeigt den gesamten UDP Verkehr zeigt den gesamten TCP Verkehr
Host 172.18.5.4	zeigt den Verkehr der IP-Adresse 172.18.5.4
Host 172.18.5.4 und Port 5050	zeigt den Verkehr der IP-Adresse 172.18.5.4 über den Port 5050

Verknüpfungen mit Vergleichsoperatoren:

tp.addr == 172.18.5.4	zeigt den Verkehr der IP-Adresse 172.18.5.4
tp.addr != 172.18.5.4	zeigt den ganzen Verkehr ausser jenem über die IP-Adresse 172.18.5.4
!(ip.addr == 172.18.5.4)	zeigt den ganzen Verkehr ausser jenem über die IP-Adresse 172.18.5.4
Ip.src == 172.18.5.4 and ip.dst == 172.18.5.5	zeigt alle Telegramme von 172.18.5.4 nach 172.18.5.5
Frame.pkt_len < 128	zeigt alle Pakete, kleiner als 128 Bytes
Tcp.port == 25 or icmp	zeigt alle Telegramme über TCP Port 25 und ICMP Telegramme
Tcp.window_size == 0&& tcp.flags.reset!=1	TCP Window Size ist Null. Buffer ist voll

Erlaubt sind folgende Vergleiche:

eq	==	and	&&
ne	!=	or	||
gt	>	xor	^^
lt	<	not	!
ge	>=		
le	<=		

Boolean Operationen, wie etwa für die sechs TCP Control Bits, sind auch erlaubt.
Zum Beispiel die Bedingung: tcp.flags.syn

URG: Urgent Pointer field significant
ACK: Acknowledgement field significant
PSH: Push Function (daten aus dem Stack senden)
RST: Reset the connection (z.B. Antwort, wenn ein Telegramm auf einem Port eingeht, der nicht geöffnet ist)

No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.12.92	Broadcast	ARP	Who has 192.168.12.93? Tell 192.168.12.92
2	0.000379	IeeeRegi_0c:54:0c	192.168.12.92	ARP	192.168.12.93 is at 00:50:c2:0c:54:0c
3	0.001539	192.168.12.92	192.168.12.93	TCP	6000 > 6000 [SYN] Seq=1603176 Ack=0 Win=720 Len=0 MSS=1458
4	0.002056	192.168.12.93	192.168.12.92	TCP	6000 > 6000 [RST, ACK] Seq=0 Ack=1603177 Win=0 Len=0
5	1.008559	192.168.12.92	192.168.12.93	TCP	6000 > 6000 [SYN] Seq=1795557 Ack=0 Win=720 Len=0 MSS=1458
6	1.009080	192.168.12.93	192.168.12.92	TCP	6000 > 6000 [RST, ACK] Seq=0 Ack=1795558 Win=0 Len=0
7	2.018454	192.168.12.92	192.168.12.93	TCP	6000 > 6000 [SYN] Seq=1987938 Ack=0 Win=720 Len=0 MSS=1458
8	2.018962	192.168.12.93	192.168.12.92	TCP	6000 > 6000 [RST, ACK] Seq=0 Ack=1987939 Win=0 Len=0
9	3.028428	192.168.12.92	192.168.12.93	TCP	6000 > 6000 [SYN] Seq=2180319 Ack=0 Win=720 Len=0 MSS=1458
10	3.028932	192.168.12.93	192.168.12.92	TCP	6000 > 6000 [RST, ACK] Seq=0 Ack=2180320 Win=0 Len=0
11	4.038395	192.168.12.92	192.168.12.93	TCP	6000 > 6000 [SYN] Seq=2372700 Ack=0 Win=720 Len=0 MSS=1458
12	4.038904	192.168.12.93	192.168.12.92	TCP	6000 > 6000 [RST, ACK] Seq=0 Ack=2372701 Win=0 Len=0
13	5.048493	192.168.12.92	192.168.12.93	TCP	6000 > 6000 [SYN] Seq=2565081 Ack=0 Win=720 Len=0 MSS=1458
14	5.048993	192.168.12.93	192.168.12.92	TCP	6000 > 6000 [RST, ACK] Seq=0 Ack=2565082 Win=0 Len=0
15	6.058333	192.168.12.92	192.168.12.93	TCP	6000 > 6000 [SYN] Seq=2757462 Ack=0 Win=720 Len=0 MSS=1458
16	6.058841	192.168.12.93	192.168.12.92	TCP	6000 > 6000 [RST, ACK] Seq=0 Ack=2757463 Win=0 Len=0
17	7.068332	192.168.12.92	192.168.12.93	TCP	6000 > 6000 [SYN] Seq=2949843 Ack=0 Win=720 Len=0 MSS=1458
18	7.068844	192.168.12.93	192.168.12.92	TCP	6000 > 6000 [RST, ACK] Seq=0 Ack=2949844 Win=0 Len=0

SYN: Synchronize sequence numbers (Start einer TCP-Verbindung)

No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.12.92	Broadcast	ARP	Who has 192.168.12.93? Tell 192.168.12.92
2	0.000408	IeeeRegi_0c:54:0c	192.168.12.92	ARP	192.168.12.93 is at 00:50:c2:0c:54:0c
3	0.001583	192.168.12.92	192.168.12.93	TCP	6000 > 6000 [SYN] Seq=769525 Ack=0 Win=720 Len=0 MSS=1458
4	0.002373	192.168.12.93	192.168.12.92	TCP	6000 > 6000 [SYN, ACK] Seq=384001 Ack=769526 Win=720 Len=0 MSS=1458
5	0.004847	192.168.12.92	192.168.12.93	TCP	6000 > 6000 [ACK] Seq=769526 Ack=384002 Win=720 Len=0

FIN: No more data from sender (Ende einer TCP-Verbindung)

No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.12.92	192.168.12.93	TCP	6000 > 6000 [FIN, ACK] Seq=833653 Ack=768002 Win=720 Len=0
2	0.000609	192.168.12.93	192.168.12.92	TCP	6000 > 6000 [ACK] Seq=768002 Ack=833654 Win=720 Len=0
3	0.001954	192.168.12.93	192.168.12.92	TCP	6000 > 6000 [FIN, ACK] Seq=768002 Ack=833654 Win=720 Len=0
4	0.003014	192.168.12.92	192.168.12.93	TCP	6000 > 6000 [ACK] Seq=833654 Ack=768003 Win=720 Len=0

Die FAQ 100535 gibt weitere Auskunft über die Filterfunktionen.

5.5 PCD7.F655 IP-Stack Debugging über RS-232 der Saia PCD®

Der TCP/IP Stack einer PCD7.F655 kann über die serielle Schnittstelle RS-232 der PCD debugged werden. Es können zum Beispiel Statistiken im Ethernet-Verkehr analysiert, die ARP-Tabelle der PCD7.F655 eingesehen oder die offenen Sockets betrachtet werden. Details über die verschiedenen Funktionen sind unten aufgelistet.

Die PCD7.F655 kann über folgende PCD-Ports debugged werden:

PCD...	PCD7.F655 auf...	Debug über...
PCD2.M150	Slot B1	Port 2 (Pin 30...34)
PCD2.M170	Slot B2	Port 4 (Pin 40...44)
PCD4.M170	Slot B2	Port 4 (Pin 40...44)
PCD2.M480	Slot B1, Slot B2	Port 2 (Pin 30...34) Port 4 (Pin 40...44)

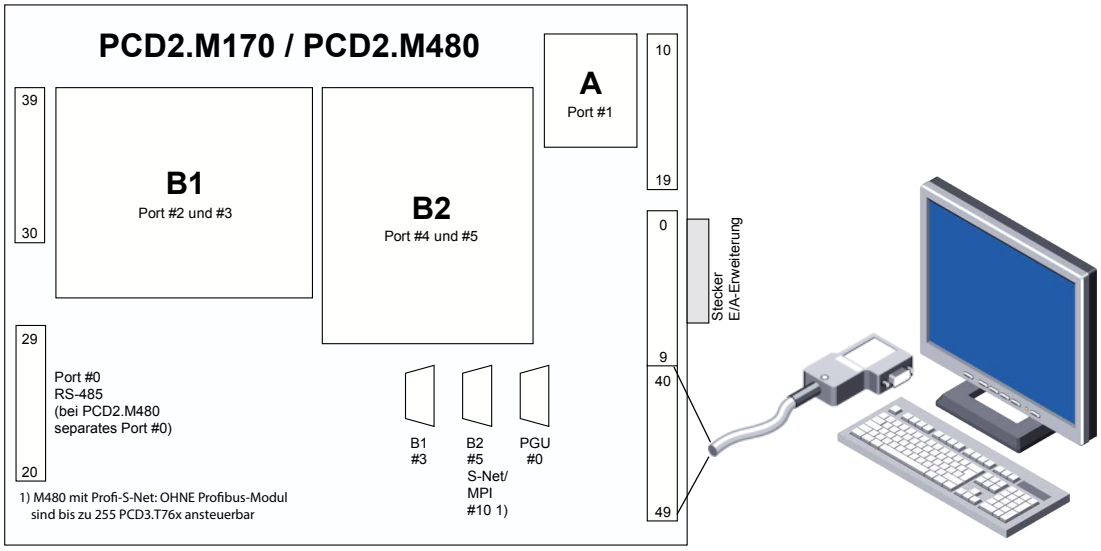
Die PCD7.F655 auf einer PCD1.M13x kann mangels fehlender serieller Schnittstelle nicht debugged werden.

Verbinden sie die serielle Schnittstelle der PCD mit einem Null-Modem Kabel (benutzt werden nur GND, TxD und RxD) mit ihrem PC und öffnen sie zum Beispiel eine Hyperterminal-Verbindung mit folgenden Parametern:

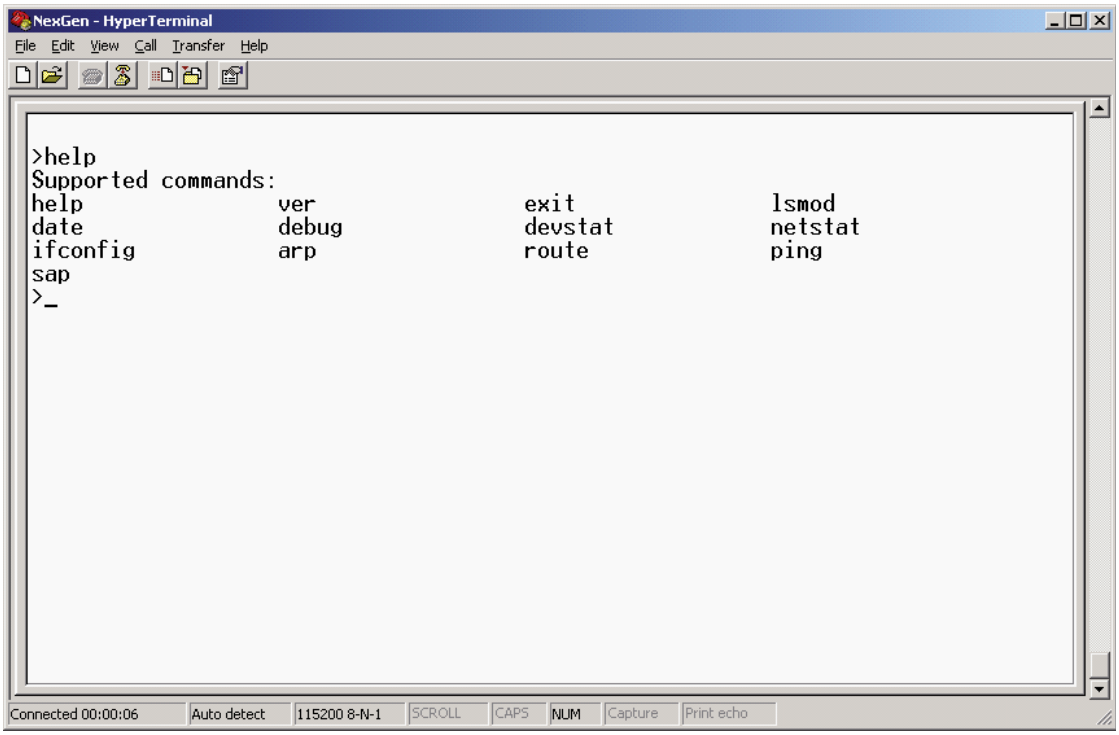
- Bits per second: 115’200
- Data Bits: 8
- Parity: None
- Stop Bits: 1
- Flow Control: None

Nach dem Betätigen der Enter-Taste erscheint der Prompt “>” des Shell. Nun können Sie über das Shell die Debug-Befehle starten.

Im Beispiel unten steckt die PCD7.F655 auf Steckplatz B2 einer PCD2.M170. Verbinden sie also Port 4 (Pin 40...44) der PCD2.M170 mit ihrem PC zum Starten des Kommunikationsshell.



Zu Beginn des Shell lassen sich mit “help” alle unterstützten Funktionen zum Debuggen des TCP/IP Stacks auflisten.



“ver” Version des Stacks

“netstat” Aktive Netzwerkverbindungen
-a
-s Genaue Statistik über IP-, ICMP-, IGMP-, TCP- und UDP-Telegramme. Fehltelegramme können hier ermittelt werden.
-b Management des zur Verfügung stehenden Speichers auf der PCD7.F655

“ifconfig” Ethernet Interface der Station

“arp” ARP-Tabelle der Station

“ping” Pingen der Remote Station

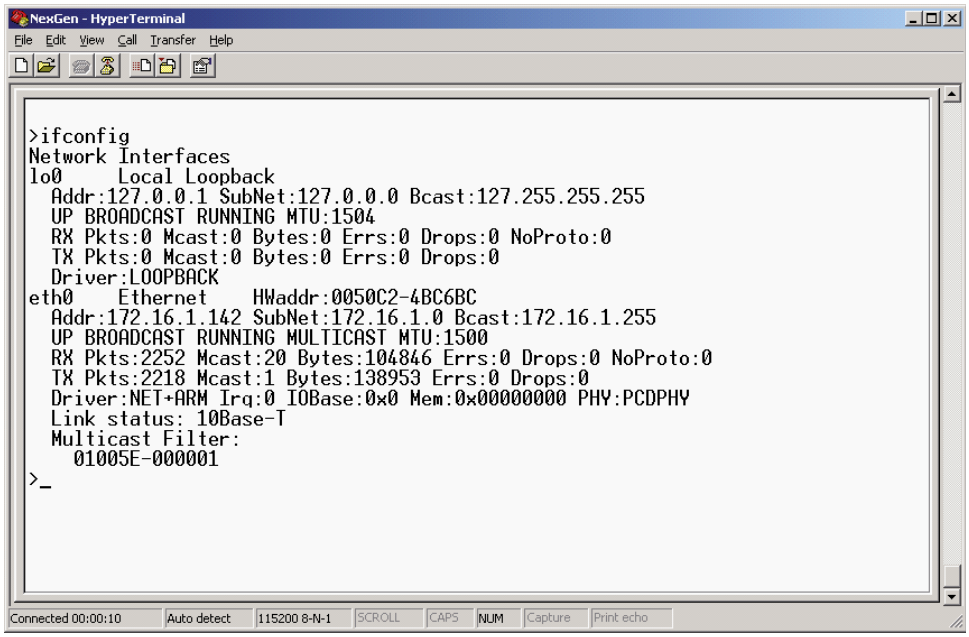
“sap” Sockets, die auf dem Modul offen sind
PCD1: **immer** Sap 16 für S-Bus UDP Port 5050
Für alle anderen PCDs, welche IP unterstützen immer Sap 32 für S-Bus UDP Port 5050

help <Befehl> Zeigt die möglichen Aufrufargumente der Funktion an.
Z.B.:”>help netstat”

Im Folgenden einige Beispiele der Benutzung der Debug-Funktion während einer laufenden Kommunikation.

Der Shell-Befehl **“ifconfig”** zeigt das Ethernet Interface der PCD7.F655 an. Die IP-Adresse ist in diesem Fall 172.16.1.142, das Subnet ist 172.16.1.0 und die Broadcast Adresse ist für das konfigurierte Netzwerk 172.16.1.255.

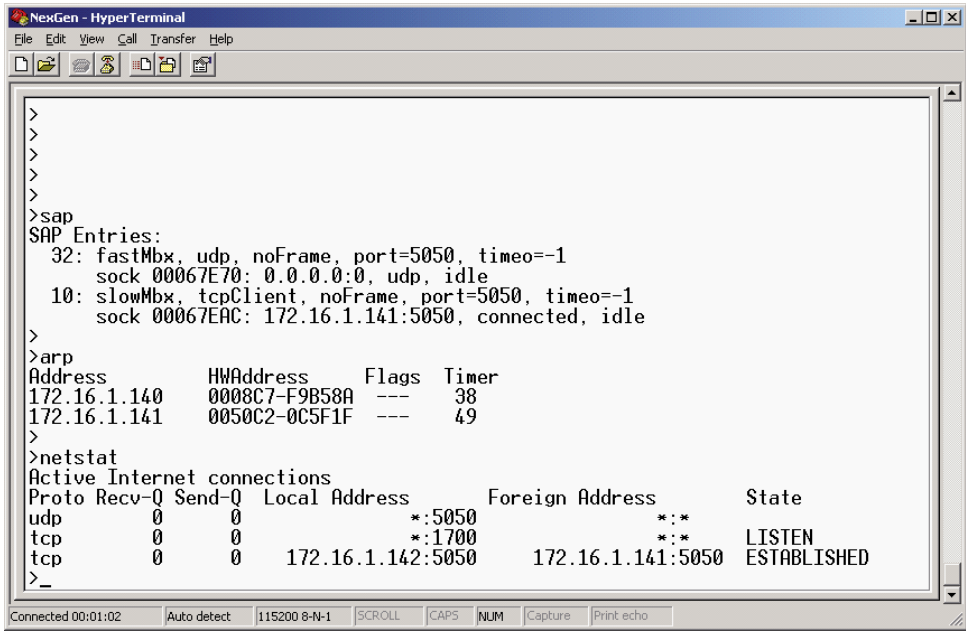
Auch die MAC-Adress ist ersichtlich (00 50 C2 4B C6 BC) sowie eine grobe Statistik der Anzahl der ein- und ausgegangenen Telegramme.



Der Shell-Befehl **“sap”** zeigt die vorhandenen Knoten-Punkte (Saps) an, welche auf der PCD7.F655 geöffnet sind. Der UDP-Port 5050 (S-Bus UDP9 ist immer der Sap 32 (ausser PCD1.M13x, Sap 16). Am Sap 10 ist eine TCP Client Verbindung geöffnet auf die Station 172.16.1.141 am entfernten Port 5050. Status ist “verbunden/ connected”. Dies zeigt auch, dass UDP Kommunikation auf Port 5050 und TCP Kommunikation auf Port 5050 gleichzeitig funktionieren.

Der Shell-Befehl **“arp”** zeigt die interne ARP-Tabelle der lokalen Station. Im Beispiel sind der PC, auf welchem der PG5 Online Debugger läuft (172.16.1.140) und die entfernte Station (172.16.1.141)-mit welcher eine TCP Verbindung auf Port 5050 besteht-zu sehen.

Der Shell-Befehl **“netstat”** zeigt die verschiedenen Sockets, welche auf der lokalen Station geöffnet sind. Es sind dies **immer** der UDP Socket auf Port 5050 (S-Bus UDP) und TCP Socket auf Port 1700 (FTP Server Socket für den Firmware Update der PCD7.F655). Im Beispiel ist noch der vorher bereits angesprochene TCP Client Socket geöffnet.



Der Shell-Befehl **"netstat -s"** zeigt eine genaue Statistik über die IP-, ICMP-, IGMP-, TCP- und UDP-Telegramme. Fehltelegramme können hier ermittelt werden.

```
>netstat -s


Ip:
     8424 total packets received
     0 with invalid header
     0 with invalid address
     0 forwarded
     0 with unknown protocol
     0 incoming packets discarded
     8424 incoming packets delivered
     8390 requests sent out
     0 outgoing packets dropped
     0 dropped because of missing route
     0 reassemblies required
     0 packets reassembled ok
     0 packets reassembles failed
     0 packets fragmented ok
     0 fragments failed
     0 fragments created
Icmp:
     0 messages received
     0 messages received with error
     Input histogram:
     0 messages sent
     0 messages not sent
     Output histogram:
Igmp:
     0 messages received
     0 messages received with error
     0 membership queries received
     0 membership reports received
     0 membership reports received for our groups
     0 membership reports sent
Tcp:
     1 active opens
     0 passive opens
     0 embryonic connections dropped
     0 established connections dropped
     10 packets sent (0 retransmitted)
     0 RESET packets sent
     9 packets received
     0 packets received with errors
     0 duplicate ACKs
     0 out-of-order packets
Udp:
     8400 packets received
     35 packets to unkown port received
     0 packets received with errors
     8380 packets sent
```

Beispiel einer reellen Applikation bestehend aus drei PCDs (lokal 192.100.100.106 sowie Remote 192.100.100.104 und 192.100.100.100) die miteinander in S-Bus UDP (Port 5050) kommunizieren. Dauer der Aufzeichnung ca. eine Woche.

```
>sap
SAP Entries:
     32:   fastMbx, udp, noFrame, port=5050, timeo=-1
           sock 000680C8: 0.0.0.0:0, udp, idle

>arp
Address          HWAddress         Flags       Timer
192.100.100.104  0050C2-4BC21D     ---         119
192.100.100.100  000D56-941547     ---         120

>ifconfig
Network Interfaces
loO  Local Loopback
     Addr:127.0.0.1 SubNet:127.0.0.0 Bcast:127.255.255.255
     UP BROADCAST RUNNING MTU:1504
     RX Pkts:0 Mcast:0 Bytes:0 Errs:0 Drops:0 NoProto:0
     TX Pkts:0 Mcast:0 Bytes:0 Errs:0 Drops:0
     Driver:LOOPBACK

ethO       Ethernet          HWaddr:0050C2-4BC6B4
     Addr:192.100.100.106 SubNet:192.100.100.0
Bcast:192.100.100.255
     UP BROADCAST RUNNING MULTICAST MTU:1500
     RX Pkts:52921868 Mcast:81423 Bytes:2723134990 Errs:39 Drops:0
NoProto:0
     TX Pkts:52846356 Mcast:15481 Bytes:2016432650 Errs:2 Drops:0
     Driver:NET+ARM lrq:0 IOBase:0x0 Mem:0x00000000 PHY:PCDPHY
     Link Status:  lOOBase-TX
     Multicast Filter:
           01005E-000001

>netstat -b
Network message buffers
total:     126 free:   116, 15560/196056 bytes in use
Socket control blocks
total:     200 free:   198, 336/33600 bytes in use
TCP control blocks
total:     200 free:   199, 212/42400 bytes in use

>netstat -s
Ip:
     52851798 total packets received
     28 with invalid header
     207 with invalid address
     0 forwarded
     0 with unknown protocol
     0 incoming packets discarded
     52851563 incoming packets delivered
     53295668 requests sent out
     0 outgoing packets dropped
     0 dropped because of missing route
     0 reassemblies required
     0 packets reassembled ok
     0 packets reassembles failed
     0 packets fragmented ok
     0 fragments failed
     0 fragments created
```

```
Icmp:
     191684 messages received
     0 messages received with error
     Input histogram:
           echo reply: 4
           echo request: 191680
     191684 messages sent
     0 messages not sent
     Output histogram:
           echo reply: 191680
           echo request: 4

Igmp:
     0 messages received
     0 messages received with error
     0 membership queries received
     0 membership reports received
     0 membership reports received for our groups
     0 membership reports sent

Tcp:
     0 active opens
     0 passive opens
     0 embryonic connections dropped
     0 established connections dropped
     0 packets sent (0 retransmitted)
     0 RESET packets sent
     0 packets received
     0 packets received with errors
     0 duplicate ACKs
     0 out-of-order packets

Udp:
     52659652 packets received
     25391 packets to unkown port received
     0 packets received with errors
     53103984 packets sent
```

6 Programmierbeispiele

6.1 Verweis auf Ethernet-Links

Auf Verweise auf Ethernet-Literatur wird hier verzichtet, denn zu diesem Thema existiert eine riesige Vielfalt und einige davon herauszupicken bringt nichts. Im weiteren ist sehr viel Information auch im Internet zu finden.

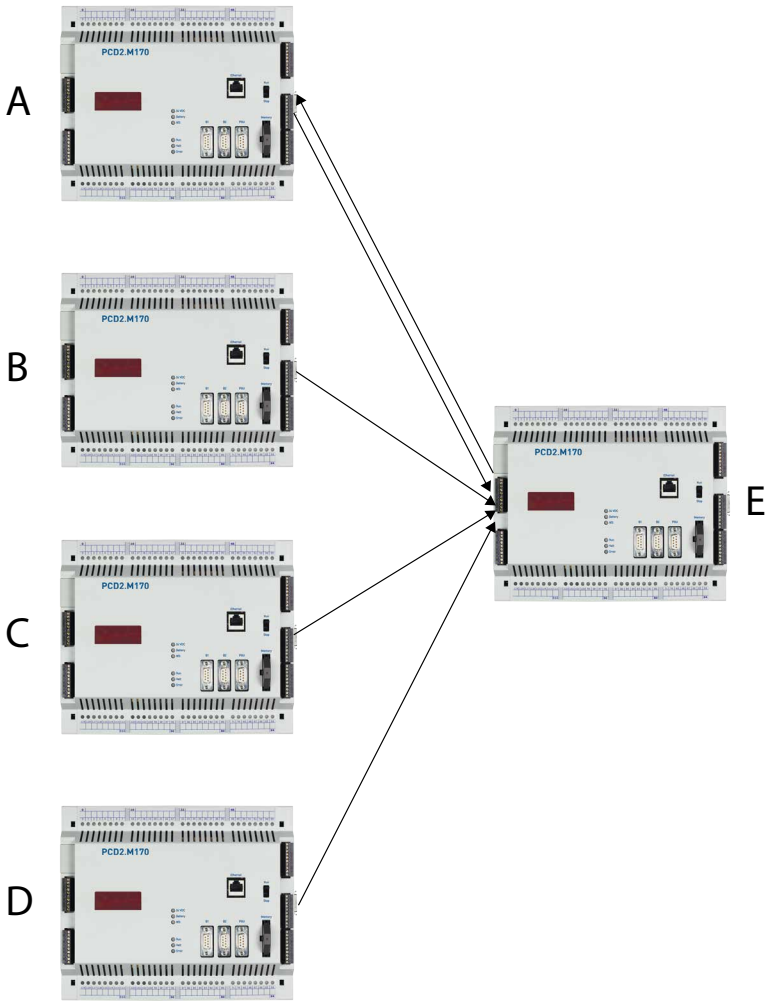
IANA (International Assigned Numbers Authority). Volltextsuche für die RFC Sammlung usw. http://www.iana.org

Hirschmann: Für unsere internen Tests benutzen wir meist industrielles Material der Firma Hirschmann und machten damit gute Erfahrungen. Deren homepage mit gut funktionierender Volltextsuche sowie guten Handbüchern ist zu finden unter http://www.hirschmann.com

Ein kostenloser **Ethernet Analyzer** ist unter http://www.wireshark.org zu finden. Genauere Informationen sind der FAQ #100569 zu entnehmen.

6.2 Leistungsmessung

Die Resultate basieren auf Messungen in einem lokalen, separaten Netzwerk mit Hub. Es wurden PCD2.M170 benutzt und die PCD7.F65x war mit der FW Version 041 ausgerüstet. Getestet wurde mit S-Bus UDP Kommunikation.



Saia PCD® E in Stopp

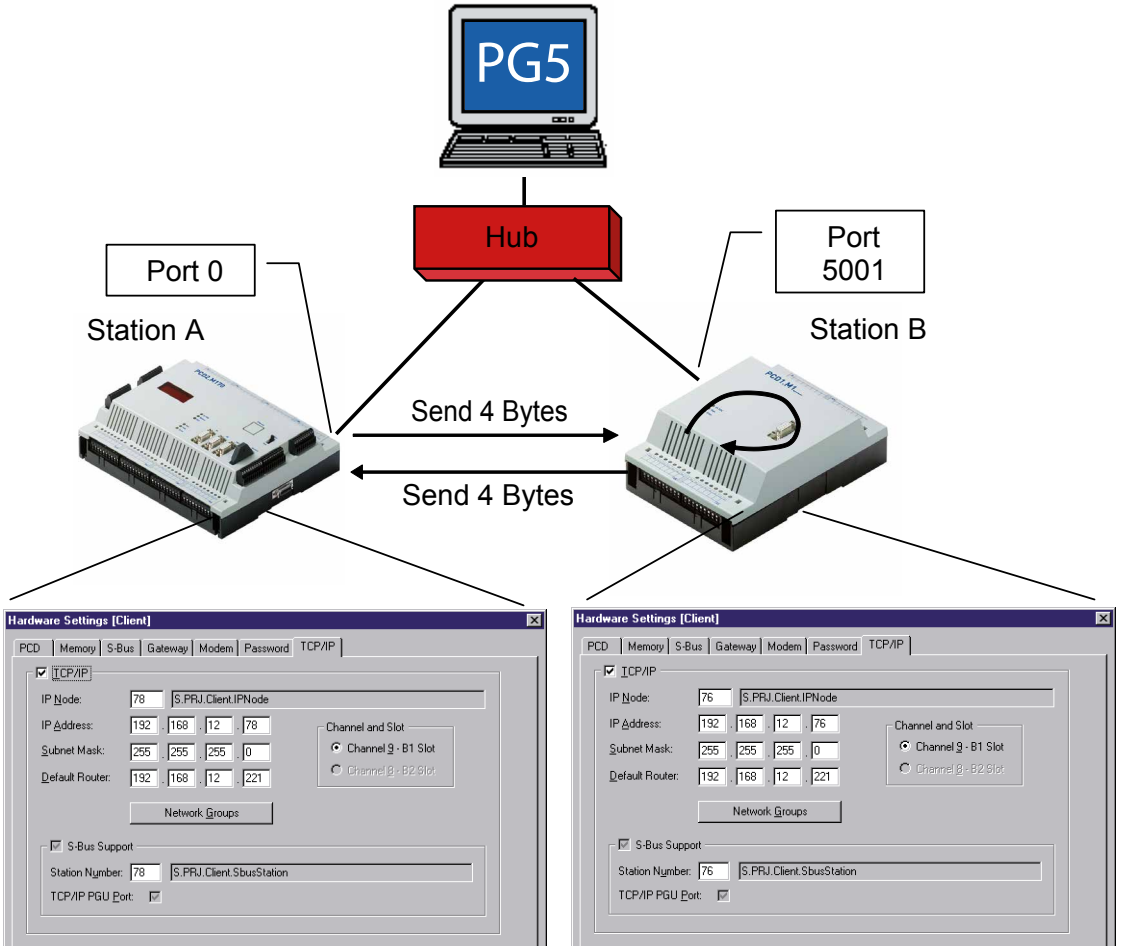
A → E	A: 124 S-Bus Tlgs/s d.h.,1 S-Bus Telegramm Benötigt 8 ms
A → E + B → E	A + B: 112 + 112 = 224 S-Bus Tlgs/s
A → E + B → E +C → E	A + B + C: 103 + 103 +103 = 309 S-Bus Tlgs/s
A → E + B → E +C → E + D → E	A + B + C +D: 86 + 85 +85 + 86 = 342 S-Bus Tlgs/s In dieser Konstellation wird die Bandbreite annähernd erreicht

Saia PCD® E schickt S-Bus Telegramme auf Saia PCD® A

A → E + B → E +C → E + E → A	E: 79 S-Bus Tlgs/s
A → E + B → E +C → E + D → E + E → A	E: 63 S-Bus Tlgs/s

6.3 Programmbeispiel: Open Data Mode TCP/IP

Nachstehend wird ein programmiertes Beispiel einer TCP/IP-Verbindung im Open Data Mode gezeigt. Es implementiert einen Client-Echo-Server in TCP/IP. Ein zuvor initialisiertes Register wird an den Server gesendet, der es zurücksendet. Der Client inkrementiert das Register und sendet die nächste Anforderung.



6.3.1 Server

Flussdiagramm des Servers

XOB 16 Initialisiert den Open Data Mode
Öffnet einen TCP-Server-Socket auf Port 5001
Schaltet in den Modus “automatische Client-Annahme”.
Wartet auf die Verbindung zum Client.

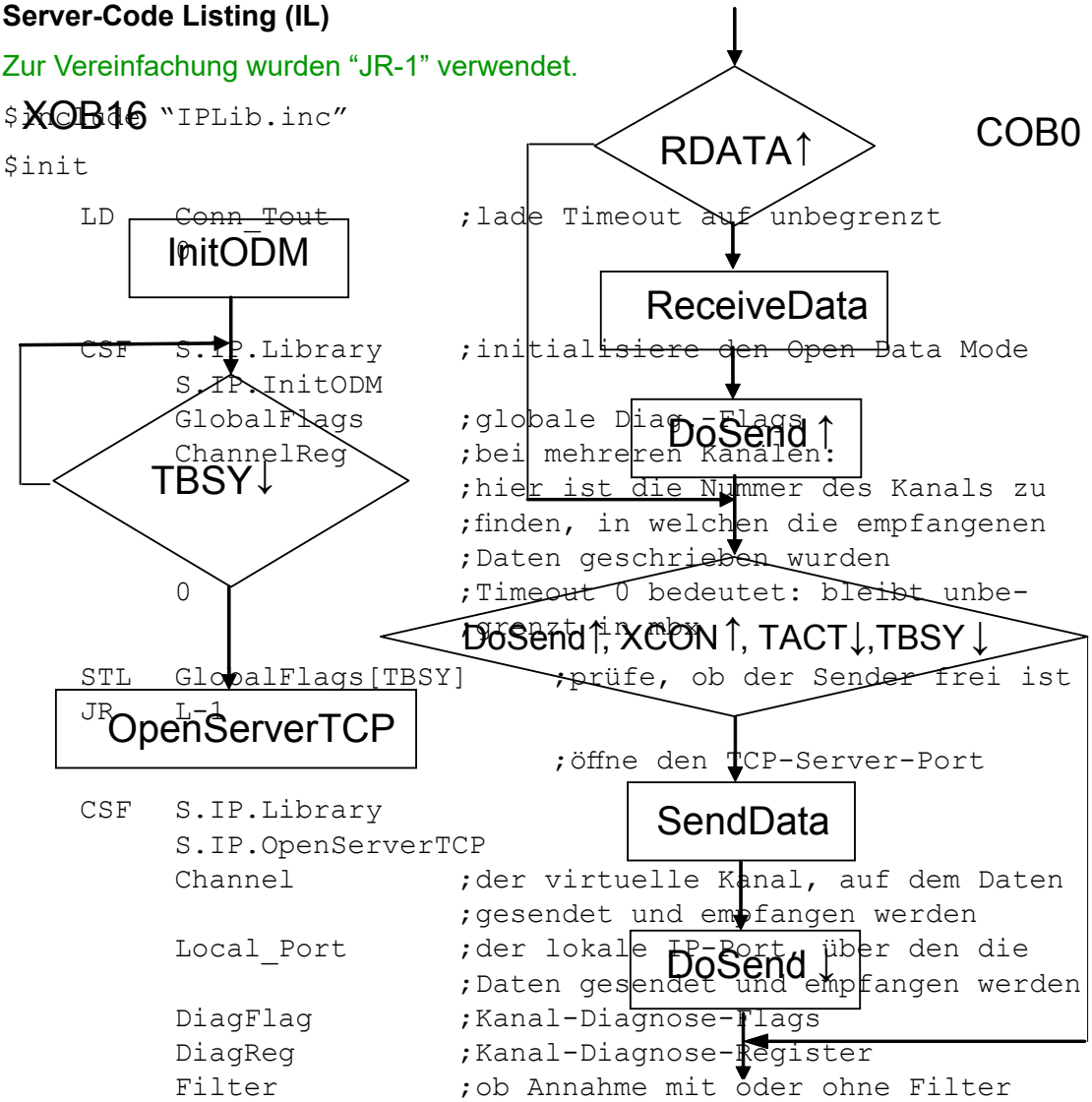
COB 0 Empfängt Daten
Sendet die empfangenen Daten zurück

Zur Vereinfachung der Darstellung wurden bei den JA/NEIN Abfragen “JR-1” Schritte benutzt.

Server-Code Listing (IL)

Zur Vereinfachung wurden “JR-1” verwendet.

```
$include "IPLib.inc"
$init
     LD    Conn_Tout      ;lade Timeout auf unbegrenzt
           0
     CSF   S.IP.Library   ;initialisiere den Open Data Mode
           S.IP.InitODM
           GlobalFlags    ;globale Diag.-Flags
           ChannelReg     ;bei mehreren Kanälen:
                          ;hier ist die Nummer des Kanals zu
                          ;finden, in welchen die empfangenen
                          ;Daten geschrieben wurden
           0              ;Timeout 0 bedeutet: bleibt unbe-
                          ;grenzt in mbx
     STL   GlobalFlags[TBSY]    ;prüfe, ob der Sender frei ist
     JR    L_1
                          ;öffne den TCP-Server-Port
     CSF   S.IP.Library
           S.IP.OpenServerTCP
           Channel        ;der virtuelle Kanal, auf dem Daten
                          ;gesendet und empfangen werden
           Local_Port     ;der lokale IP-Port, über den die
                          ;Daten gesendet und empfangen werden
           DiagFlag       ;Kanal-Diagnose-Flags
           DiagReg        ;Kanal-Diagnose-Register
           Filter         ;ob Annahme mit oder ohne Filter
           Conn_Tout      ;Verbindungs-Timeout: 0 für kein,
                          ;andernfalls x Sekunden
     RES   DoSend         ;Trigger für das Senden
     RES   DoDisconnect
$endinit
```



```
      COB   0
            0
read: STH   GlobalFlags[RDATA]    ;wird gesetzt, wenn ein Telegramm
                                  ;eintrifft
      JR    L nodata

      CSF H S.IP.Library          ;Daten empfangen
            S.IP.ReceiveData
            Channel               ;Nummer des Empfangskanals angeben,
                                  ;auf welchem die Daten eingegangen
                                  ;sind. Bei mehreren Kanälen:
                                  ;Wert des Kanal-Registers.
            RecvIP                ;Daten empfangen von dieser IP-
                                  ;Adresse
            RecvPort              ;Daten empfangen über diesen Port
            4                     ;max. Datenlänge unterstützt
                                  ;(Puffer)
            RecvLnth              ;empfangene Datenlänge effektiv
            RecvData              ;empfangene Daten

      SET   DoSend

nodata STH DoSend
      ANH   DiagFlag[XCON]        ;Port verbunden (Kanal)
      ANL   DiagFlag[TACT]        ;und Sender inaktiv (Kanal)
      ANL   GlobalFlags[TBSY]     ;und Sender frei (global)
      JR    L nosend              ;senden nicht möglich
      CSF H S.IP.Library          ;Daten senden
            S.IP.SendData
            Channel               ;virtueller Kanal
            RecvIP                ;Partner IP-Adresse
            RecvPort              ;Partner-Port
            4                     ;4 Bytes senden
            RecvData              ;die zuvor empfangenen 4 Bytes zu-
                                  ;rück senden

      RES   DoSend

      STH   DoDisconnect
      JR    L nosend

      CSF   S.IP.Library
            S.IP.DisconnectTCP
            Channel
            RecvIP
            RecvPort
      RES   DoDisconnect

nosend: ECOB
```

Server-Code: Variablendeklaration und -initialisierung

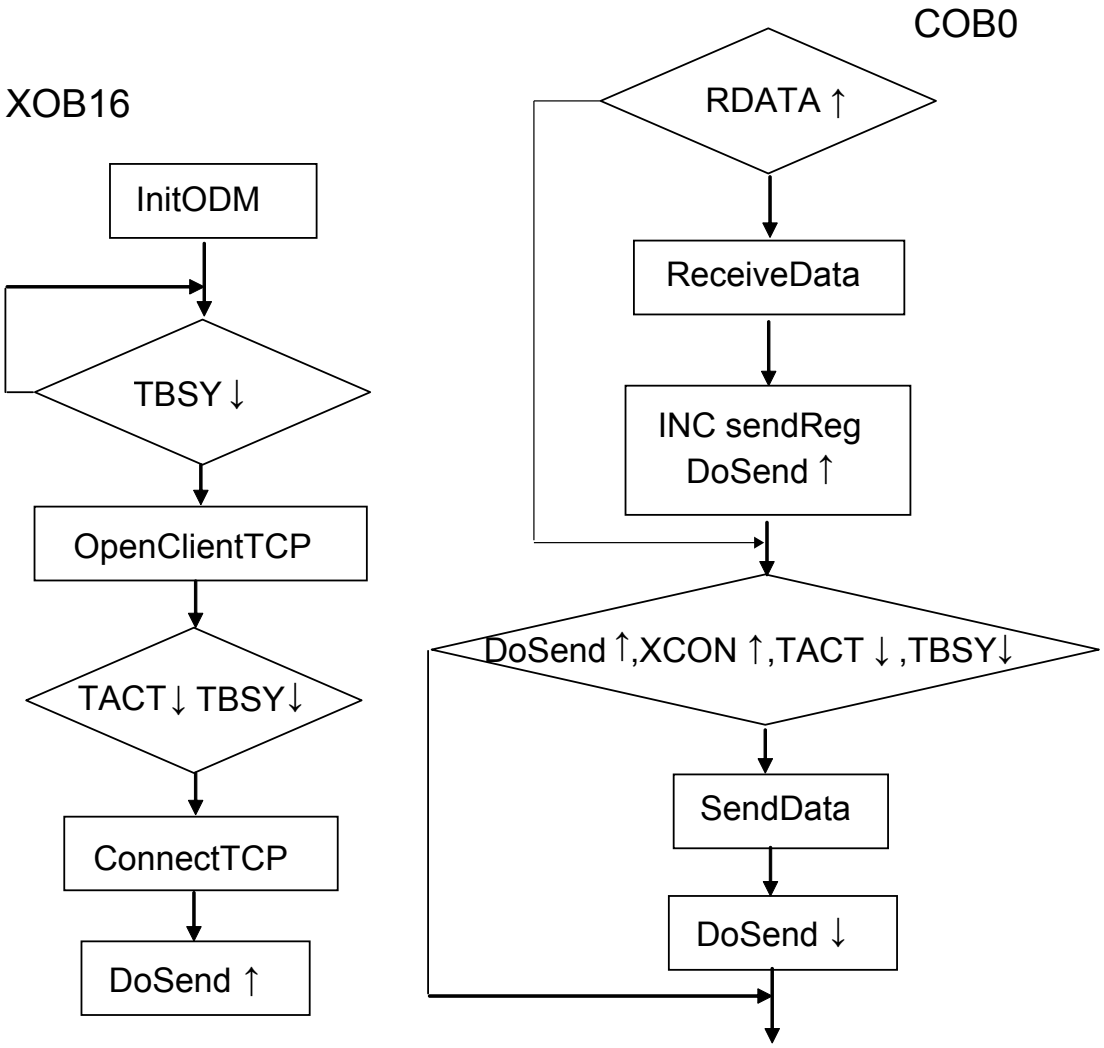
Group/Symbol	Type	Address/Value	Comment
RDIA		0	
TBSY		0	
RDATA		1	
TACT		1	
RCON		2	
TDIA		2	
XCON		3	
NEXE		4	
RemotePort		65535	
GlobalFlags	F	0 [3]	
DoSend	F	5	
DiagFlag	F	10 [5]	
DoDisconnect	F	100	
Filter	K Constant	0	
Channel	K Constant	10	
Local_Port	K Constant	5001	
Conn_Tout	R		
ChannelReg	R	0	
DiagReg	R	1	
SendReg	R	2	
RecvPort	R	3	
RecvLnth	R	4	
RecvData	R	5	
RecvIP	Text RAM		
RemoteIP	Text RAM		

6.3.2 Client

Flussdiagramm des Client

XOB16 Initialisiert den Open Data Mode
Öffnet einen TCP-Client-Socket auf Port 0
Stellt die Verbindung zum Server auf Port 5001 her

COB0 Überprüft Daten
Empfängt Daten
Inkrementiert Register (Zähler)
Sendet Daten



Client-Code Listing (IL)

Zur Vereinfachung wurden “JR-1” und “JR-2” verwendet.

```
$include "IPLib.inc"
$init
     LD    SendReg          ;initialisiere das Senderegister
           0
     LD    Conn_Tout        ;lade Timeout auf unbegrenzt
           0

     CSF   S.IP.Library     ;initialisiere den Open Data Mode
           S.IP.InitODM
           GlobalFlags      ;globale Diag.-Flags
           ChannelReg       ;bei mehreren Kanälen:
                            ;hier ist die Nummer des Kanals zu
                            ;finden, in welchen die empfangenen
                            ;Daten geschrieben wurden
           0                ;Timeout 0 bedeutet: bleibt unbe-
                            ;grenzt in mbx
     STL   GlobalFlags[TBSY]     ;prüfe, ob der Sender frei ist
     JR    L -1
                                 ;öffne den TCP-Server-Port
     CSF   S.IP.Library
           S.IP.OpenClientTCP
           Channel          ;der virtuelle Kanal, auf dem Daten
                            ;gesendet und empfangen werden
           Port             ;der lokale IP-Port, über den die
                            ;Daten gesendet und empfangen werden
           DiagFlag         ;Kanal-Diagnose-Flags
           DiagReg          ;Kanal-Diagnose-Register
           Filter           ;ob Annahme mit oder ohne Filter
           Conn_Tout        ;Verbindungs-Timeout: 0 für kein,
                            ;andernfalls x Sekunden
     STL   DiagFlag[TACT]
     ANL   GlobalFlags[TBSY]
     JR    L -2

     CSF   S.IP.Library     ;Verbindung zum TCP-Server aufbauen
           S.IP.ConnectTCP
           Channel          ;virtueller Kanal
           RemoteIP         ;Server IP-Adresse
           RemotePort       ;Server-Port

     SET   DoSend

     RES   DoDisconnect
$endinit
```

```
      COB   0
            0
read: STH   GlobalFlags[RDATA]    ;wird gesetzt, wenn ein Telegramm
                                  ;eintrifft
      JR    L nodata

      CSF H S.IP.Library          ;Daten empfangen
            S.IP.ReceiveData
            Channel               ;Nummer des Empfangskanals angeben,
                                  ;auf welchem die Daten eingegangen
                                  ;sind. Bei mehreren Kanälen:
                                  ;Wert des Kanal-Registers.
            RecvIP                ;Daten empfangen von dieser IP-
                                  ;Adresse
            RecvPort              ;Daten empfangen über diesen Port
            4                     ;max. Datenlänge unterstützt
                                  ;(Puffer)
            RecvLnth              ;empfangene Datenlänge effektiv
            RecvData              ;empfangene Daten

      SET   DoSend

nodata STH DoSend
      ANH   DiagFlag[XCON]        ;Port verbunden (Kanal)
      ANL   DiagFlag[TACT]        ;und Sender inaktiv (Kanal)
      ANL   GlobalFlags[TBSY]     ;und Sender frei (global)
      JR    L nosend              ;senden nicht möglich
      CSF H S.IP.Library          ;Daten senden
            S.IP.SendData
            Channel               ;virtueller Kanal
            RecvIP                ;Partner IP-Adresse
            RecvPort              ;Partner-Port
            4                     ;4 Bytes senden
            RecvData              ;die zuvor empfangenen 4 Bytes zu-
                                  ;rück senden

      RES   DoSend

      STH   DoDisconnect
      JR    L nosend

      CSF   S.IP.Library          ;Verbindung trennen
            S.IP.DisconnectTCP
            Channel
            RecvIP
            RecvPort
      RES   DoDisconnect

nosend: ECOB
```

Client-Code: Variablendeklaration und -initialisierung

Group/Symbol	Type	Address/Value	Comment
RDIA		0	
TBSY		0	
RDATA		1	
TACT		1	
RCON		2	
TDIA		2	
XCON		3	
NEXE		4	
RemotePort		5001	
GlobalFlags	F	0 [8]	
DiagFlag	F	8 [8]	
DoSend	F	16	
DoDisconnect	F	100	
Channel	K Constant	10	
Port	K Constant	5005	
Conn_Tout	R		
ChannelReg	R	0	
DiagReg	R	1	
SendReg	R	2	
RecvPort	R	3	
RecvLnth	R	4	
RecvData	R	5	
RecvIP	Text RAM		
RemoteIP	Text RAM		

A Anhang

A.1 Icons

	Dieses Symbol weist auf weitere Informationen hin, die in diesem oder einem anderen Handbuch oder in technischen Unterlagen zu diesem Thema existieren. Zu solchen Dokumenten gibt es keine direkten Verweise.
	Dieses Symbol warnt den Leser, dass Komponenten durch elektrostatische Entladung bei Berührung beschädigt werden können. Empfehlung: berühren Sie zumindest den Minuspol des Systems (Gehäuse PGU-Stecker) bevor Sie mit den elektronischen Teilen in Kontakt kommen. Noch besser ist es, ein geerdetes Band am Handgelenk zu tragen, das mit dem Minuspol des Systems verbunden ist.
	Dieses Symbol bezeichnet Anweisungen, die streng befolgt werden müssen.
	Erklärungen neben diesem Symbol sind nur für die Saia PCD® Classic-Serie gültig.
	Erklärungen neben diesem Symbol sind nur für die Saia PCD® xx7-Serie gültig.

A

A.2 Kontakt

Saia-Burgess Controls AG
Bahnhofstrasse 18
3280 Murten / Schweiz

Telephon ..... +41 26 580 30 00
Fax..... +41 26 580 34 99

E-Mail Support: ..... support@saia-pcd.com
Supportseite: ..... www.sbc-support.com
SBC Seite: ..... www.saia-pcd.com
Internationale Vertretungen &
SBC Verkaufsgesellschaften: ..... www.saia-pcd.com/contact

Postadresse für Rücksendungen von Produkten, durch Kunden des Verkaufs Schweiz:

Saia-Burgess Controls AG
Service Après-Vente
Bahnhofstrasse 18
3280 Murten / Schweiz

A