Applikation zur Kommunikation

applications
& TOOLS

Subnetz übergreifendes Datensatz-Routing über eine Gateway CPU mit S7-Kommunikation (BSEND/BRECEIVE)

SIEMENS

Präambel

Anforderung

Aufgrund heterogener Netzstrukturen gewinnt die subnetzübergreifende Kommunikation auch in der Automatisierungswelt immer mehr an Bedeutung. Um Transparenz und Durchgängigkeit zu ermöglichen, müssen immer größer werdende Datenmengen über diese Subnetzgrenzen hinweg transportiert und analysiert werden.
Da diese Anforderung nicht nur an die Beziehung PC ←→ AS, sondern auch für die Beziehung AS←→AS gestellt wird, sind besondere Vorkehrungen zu treffen.

Lösungsmöglichkeiten

Die SIMATIC beherrscht auf den verschiedenen Bussystemen eine große Anzahl relevanter Protokolle. Das speziell für das SIMATIC-S7-Umfeld entwickelte S7-Protokoll bietet besonders für die hier vorgestellte Lösung die besten Voraussetzungen, um die gestellten Anforderungen mit geringem Programmieraufwand zu realisieren.

Beispielapplikation

Exemplarisch wird hier anhand einer Mittlerstation (auch Gateway genannt) die Kommunikation zwischen AS-Systemen gezeigt, die sich in unterschiedlichen Subnetzen befinden. Dabei fungiert die Mittlerstation als „Datenumsetzer“ zwischen den unterschiedlichen Subnetzen. Speziell wird auf die notwendigen Mechanismen eingegangen, die ein solches „Daten umsetzen“ (auch Datensatz-Routing genannt) ermöglichen.

Nutzen für den Anwender

Mit der komplett vorbereiteten und ablauffähigen Applikationssoftware sowie dieser Dokumentation ergibt sich für Sie folgender Nutzen:

- Erwerb von Hintergrundwissen zum Thema S7-Kommunikation und der Anwendung der Programmierschnittstelle BSEND / BRECEIVE.
- Verwendung von Modulen, die als Basiselement unabhängig von der Gesamtapplikation in anderen Applikationen eingesetzt werden können, z.B. Fehlerauswertung und –behandlung sowie Ringpuffer.
- Zeitersparnis bei Eigenentwicklungen durch einfache Nutzung der in der Beispielapplikation verwendeten Funktionselemente.

Lösungsgrundansatz der vorliegenden Applikation

Für die Applikation werden unterschiedliche Komponenten (Hard- und Software) benötigt. Diese werden zum Teil mit dieser Applikation mitgeliefert, teilweise stellen Sie die Komponenten zur Verfügung.

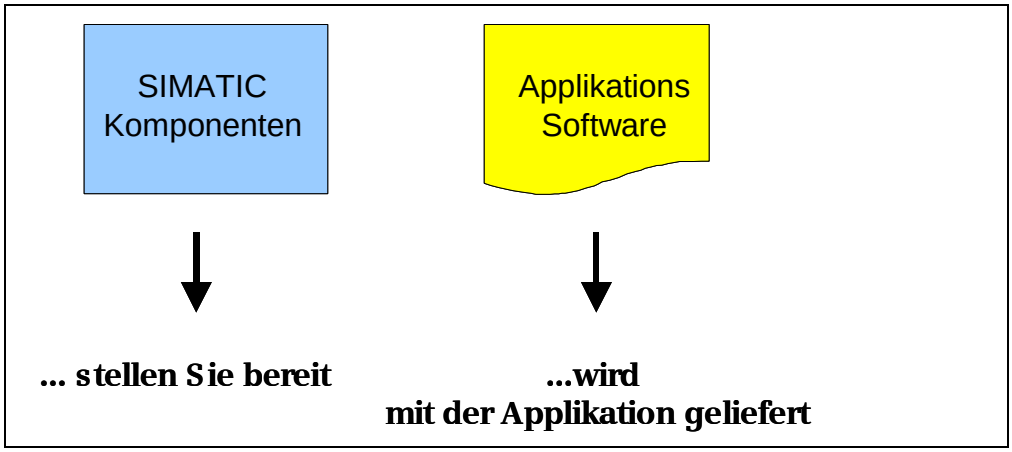


Bild 0-1

Um welche Komponenten es dabei im Einzelnen geht und wie deren Zusammenspiel miteinander funktioniert, wird in dieser Dokumentation vermittelt.

Vorwort

Aufbau des Dokuments

Die Dokumentation der vorliegenden Applikation ist in folgende Hauptteile gegliedert.

Teil	Beschreibung	Hinweis
A1	Im Teil A1 erfahren Sie alles, um sich einen Überblick zu verschaffen. Sie lernen die verwendeten Komponenten (Standard Hard- und Softwarekomponenten und die zusätzlich entwickelte Software) kennen. Die dargestellten Funktionseckdaten zeigen die Leistungsfähigkeit der vorliegenden Applikation.	
A2	Im Teil A2 wird auf die detaillierten Funktionsabläufe der beteiligten Hard- und Softwarekomponenten eingegangen. Sie benötigen diesen Teil nur, wenn Sie den Detailablauf und das Zusammenspiel der Lösungskomponenten kennen lernen wollen.	Sie können diesen Teil überspringen, wenn Sie die Applikation zunächst anhand der Step-by-Step-Anweisungen einmal testen wollen.
B	Der Teil B führt Sie Schritt für Schritt durch den Aufbau und die Inbetriebnahme der Applikation auf Basis der mitgelieferten Software und stellt, wo notwendig, die Kernprojektierungsschritte vor.	
C	Der Teil C ist dann interessant, wenn Sie auf Basis der vorliegenden Software eine Erweiterung/Anpassung an Ihre Anlage vornehmen möchten.	

Gewährleistung, Haftung und Support

Für die in diesem Dokument enthaltenen Informationen übernehmen wir keine Gewähr.

Unsere Haftung, gleich aus welchem Rechtsgrund, für durch die Verwendung der in diesem Dokument beschriebenen Beispiele, Hinweise, Programme, Projektierungs- und Leistungsdaten usw. verursachte Schäden ist ausgeschlossen, soweit nicht z.B. nach dem Produkthaftungsgesetz in Fällen des Vorsatzes, der grober Fahrlässigkeit, wegen der Verletzung des Lebens, des Körpers oder der Gesundheit, wegen einer Übernahme der Garantie für die Beschaffenheit einer Sache, wegen des arglistigen Verschweigens eines Mangels oder wegen Verletzung wesentlicher Vertragspflichten zwingend gehaftet wird. Der Schadensersatz wegen Verletzung wesentlicher Vertragspflichten ist jedoch auf den vertragstypischen, vorhersehbaren Schaden begrenzt, soweit nicht Vorsatz oder grobe Fahrlässigkeit vorliegt oder wegen der Verletzung des Lebens, des Körpers oder der Gesundheit zwingend gehaftet wird. Eine Änderung der Beweislast zu Ihrem Nachteil ist hiermit nicht verbunden.

Die Applikationsbeispiele sind unverbindlich und erheben keinen Anspruch auf Vollständigkeit hinsichtlich Konfiguration und Ausstattung sowie jeglicher Eventualitäten. Sie stellen keine kundenspezifische Lösungen dar, sondern sollen lediglich Hilfestellung bieten bei typischen Aufgabenstellungen. Sie sind für den sachgemäßen Betrieb der beschrieben Produkte selbst verantwortlich. Diese Applikationsbeispiele entheben Sie nicht der Verpflichtung zu sicherem Umgang bei Anwendung, Installation, Betrieb und Wartung. Durch Nutzung dieses Applikationsbeispiels erkennen Sie an, dass Siemens über die oben beschriebene Haftungsregelung hinaus nicht für etwaige Schäden haftbar gemacht werden kann. Wir behalten uns das Recht vor, Änderungen an diesem Applikationsbeispiel jederzeit ohne Ankündigung durchzuführen. Bei Abweichungen zwischen den Vorschlägen in diesem Applikationsbeispiel und anderen Siemens Publikationen, wie z.B. Katalogen, hat der Inhalt der anderen Dokumentation Vorrang.

Bei Fragen zu diesem Beitrag wenden Sie sich bitte über folgende E-Mail-Adresse an uns:

csweb@ad.siemens.de

Inhaltsverzeichnis

Teil A1 : Applikationsbeschreibung

Ziel Teil A1

Der Teil A1 dieses Dokuments soll dem Leser

- die Automatisierungsaufgabe klarmachen
- eine Lösungsmöglichkeit aufzeigen
- die Leistungsfähigkeit der Gesamtapplikation aufzeigen

Inhalt Teil A1

1 Automatisierungsaufgabenstellung

Einordnung

Dieses Kapitel beschreibt die Automatisierungsaufgabenstellung. Es bildet die Grundlage für die vorgestellte Automatisierungslösung.

Technologische Aufgabenstellung/ Übersicht

Datenrouting über Subnetzgrenzen ist im Gegensatz zum PG-Routing zum jetzigen Zeitpunkt nicht im Betriebssystem der SIMATIC-CPUs enthalten.

Diese Applikation soll zeigen, wie anwenderspezifische Daten von Station A zu Station B über Subnetzgrenzen hinweg gesendet werden können. Um dies realisieren zu können, ist eine „Mittlerstation“ – ein Gateway – notwendig, welches Mitglied in beiden Subnetzen ist.

Dabei erfolgt das Weiterleiten (Routing) der Daten bidirektional; d.h., sowohl Station A als auch Station B können gleichzeitig Datenquellen und –senken sein.

Folgendes Bild veranschaulicht die Aufgabenstellung:

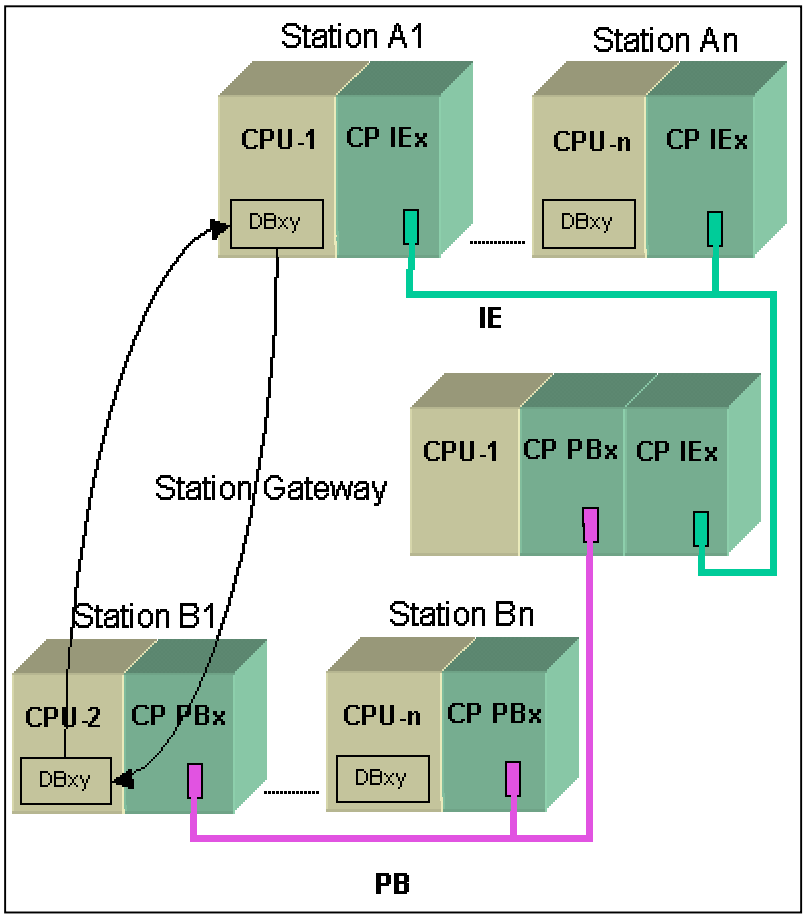


Bild 1-1

Konkrete Anforderungen an die Applikation

- Quittierte und bidirektionale Übertragung von anwenderspezifischen Daten (bis 64kB) über Subnetzgrenzen hinweg (IE→PB und PB→IE)
- Unterstützung von bis zu 8 Stationen (z.B. 4 Stationen im IE-Subnetz und 4 Stationen im PB-Subnetz)
- Übertragung durch das S7-Protokoll mittels S7-Kommunikation (BSEND/BRECV)
- Bereitstellung einer universellen, wiederverwendbaren Anwenderschnittstelle für die Kernfunktion „Datenrouting“
- Proprietäre Telegrammlaufzeitermittlung

2 Automatisierungslösung

Einleitung

In diesem Kapitel erfahren Sie konkret, wie die vorliegende Applikation die in Kapitel 1 genannte Automatisierungsaufgabe löst. Dabei wird aufgezeigt, was die Applikation leisten kann, welche Teilmodule sie enthält und wie diese funktionieren. Die Beschreibung des Funktionsmechanismus hält sich dabei bewusst auf einem globalen Niveau. Ein tiefergehendes Verständnis wird Ihnen im Teil A2 dieser Dokumentation vermittelt, den Sie allerdings nur dann benötigen, wenn Sie Hintergrundwissen, den Detailablauf und das Zusammenspiel der Lösungskomponenten kennen lernen wollen.

Inhalt Kapitel Automatisierungslösung

Tabelle 2-1

Kap.-Nr.	Kapitelüberschrift	Hier erfahren Sie...
2.1	Übersicht zur Gesamtlösung	... welche Komponenten an der Applikation beteiligt sind.
2.2	Beschreibung der Funktionalitäten der Applikation	... wie die Applikation arbeitet und welche Kernelemente implementiert wurden.
2.3	Alternativlösungen	... welche Möglichkeiten noch in der SIMATIC realisiert sind, um diese Funktionalität zu ermöglichen.
2.4	Erforderliche Komponenten	... welche Hard- und Softwarekomponenten in dieser Applikation verwendet werden, weiterhin erhalten Sie einen Überblick über die verwendeten Applikationsbausteine und deren Funktion.

2.1 Übersicht zur Gesamtlösung

Darstellung der beteiligten Komponenten

Das folgende Übersichtsbild zeigt den HW-Aufbau der Beispielapplikation und der darauf befindlichen Anwendersoftware-Komponenten.

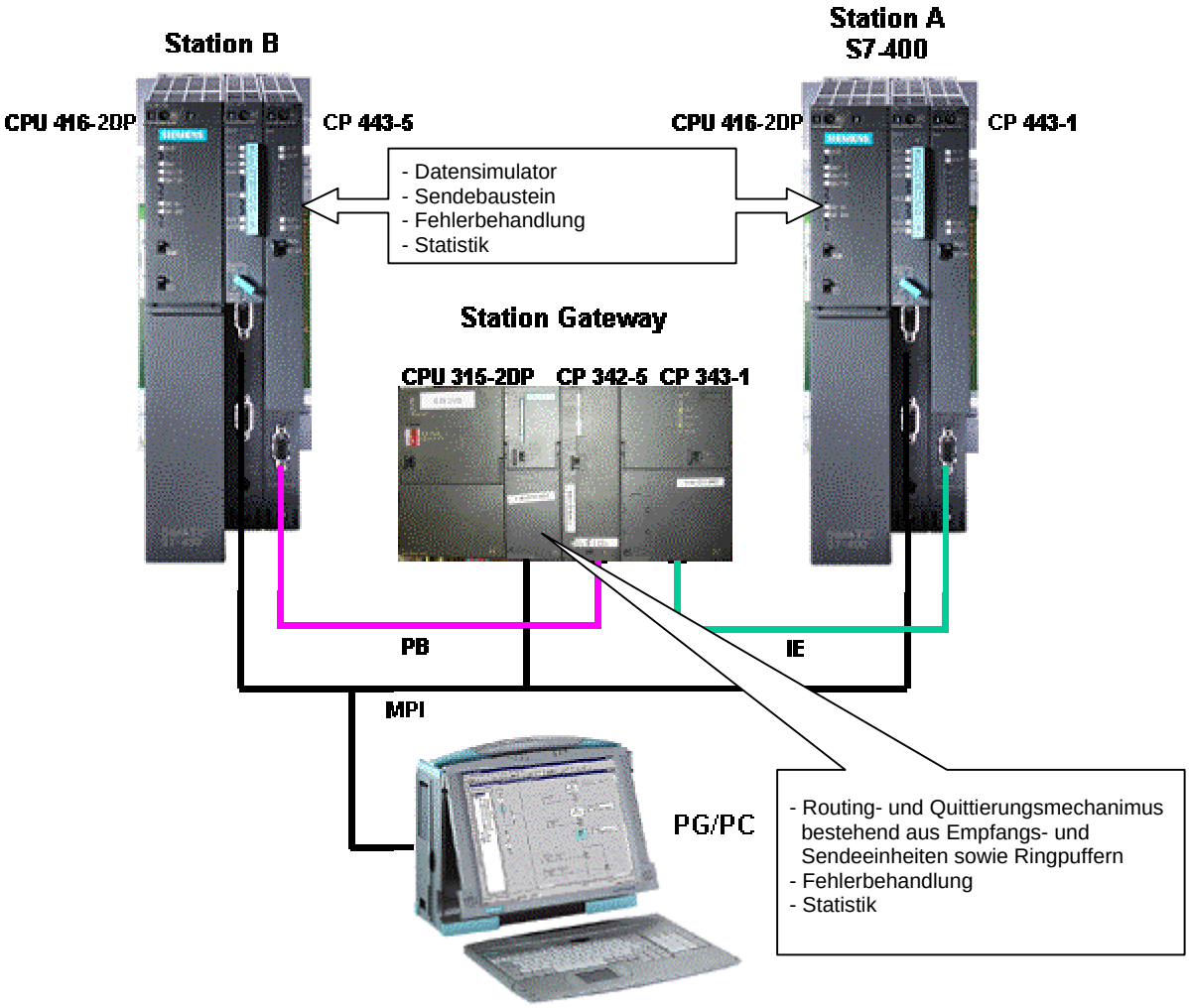


Bild 2-1

Hinweis

Die Beispielapplikation wurde für je eine Station in jedem Subnetz erstellt. Intern sind die Strukturen auf 8 Stationen angelegt.

2.2 Beschreibung der Funktionalitäten der Applikation

Was ist die Kernfunktionalität der Applikation?

Die Applikation leitet anwenderspezifische Daten einer Station in einem Subnetz (PB oder IE) zu einer Station in einem anderen Subnetz weiter und quittiert dies.

Die Quell-/Zielstationen senden nur dann, wenn sie eine Quittung erhalten haben.

Beschreibung des Programmablaufs

Die folgende Tabelle beschreibt, welche Aktionen die Applikation durchführt, um Daten von Station B nach Station A zu senden:

Tabelle 2-2 Ablauf der Applikation

Ort	Schritt	Aktion	Erläuterung
Station B – CPU 416-2DP	1	Sobald der `Sendebaustein` frei ist, wird mit dem `Datensimulator` ein Datensatz generiert. Dabei wird die aktuelle CPU-Zeit gespeichert.	Für die erstmalige Erzeugung eines Datensatzes wird ein Triggerereignis benötigt, welches über die Variablentabelle gesetzt werden kann.
	2	Dem `Sendebaustein` werden die generierten Daten sowie die Quell-und Zielinformation übergeben.	Die Quell- und Zielinformationen sind über die Variablentabelle vorzugeben. Diese Informationen entsprechen den Local IDs, die im Gateway für die jeweilige Station projektiert wurden. Logisch gesehen repräsentieren diese Informationen die Routing-Informationen.
	3	Der `Sendebaustein` generiert aus den übergebenen Informationen einen Telegrammkopf und sendet die zusammengesetzten Daten an das Gateway.	Das Gateway benötigt den Telegrammkopf, um das Ziel der Daten bestimmen zu können.
Gateway – CPU 315-2DP	4	Sobald das Gateway die Daten empfangen hat, analysiert es den Telegrammkopf und trägt die Daten in den jeweiligen Ziel-`Ringpuffer` ein.	Jede projektierte Verbindung (1-8), d.h. jeder zugeordnete Kommunikationskanal, hat im Gateway genau einen zugeordneten `Ringpuffer`. Durch das Eintragen des empfangenen Telegramms in den Ziel-Ringpuffer wird der Routing-Mechanismus realisiert.
	5	Die `Sendeinheit` (=Sendeeinheit 1) des Gateways liest das älteste Datum aus ihrem `Ringpuffer` (=Ringpuffer 1) aus und sendet die Daten an die Zielstation (=Station A).	Jede Sendeeinheit benutzt einen ihr zugeordneten Ringpuffer und sendet die Daten an die ihr zugeordnete Zielstation. **Hinweis:** Die Gateway-CPU hat 8 Sendeeinheiten.
Station A	6	Station A empfängt die Daten	**Hinweis:** Die Zielstation sendet kein explizites Quittungstelegramm, da dies von der S7-Kommunikation automatisch durchgeführt wird.

Ort	Schritt	Aktion	Erläuterung
Gateway – CPU 315-2DP	7	Nach Beendigung des Sendeauftrages wird ein Quittungstelegramm generiert und in den `Ringpuffer` der Quellstation (=Ringpuffer 2) eingetragen.	Das Senden von Telegrammen jedweder Art wird ausschließlich von der jeweiligen `Sendeeinheit` ausgeführt.
	8	Die `Sendeeinheit` (=Sendeeinheit 2) liest das Quittungstelegramm aus dem ihr zugeordneten `Ringpuffer` (=Ringpuffer 2) aus und sendet das Telegramm an die Quellstation (=Station B).	Das ordnungsgemäße Senden der Quittung wird nicht weiter berücksichtigt. Der Routing-Vorgang ist im Gateway damit beendet.
Station B – CPU 416-2DP	9	Der `Sendebaustein` der Quellstation wertet das Quittungstelegramm aus.	Der `Sendebaustein` der Ziel-/Quellstation enthält neben einem BSEND auch einen BRCV-Aufruf, der ständig auf Quittungstelegramme wartet.
	10	Bei negativer Quittung werden Fehlerstati nach außen gegeben.	Fehler können durch den BSEND- und den BRCV-Aufruf auftreten. Hinzu kommen Fehler der BSEND-Aufrufe im Gateway, die im `Sendebaustein` durch Analyse des Kopfes des Quittungstelegramms ermittelt werden.
	11	Der `Sendebaustein` der Quellstation wird nach Ankommen des Quittungstelegramms oder Fehler in BSEND/BRECV-Aufrufen wieder in den Zustand „Frei“ gesetzt.	Das Senden neuer Daten ist möglich.
	12	Die aktuelle Zeitspanne wird gemessen und fließt in den Mittelwert (`Statistik`) mit ein.	Es werden immer die letzten 100 Werte gemittelt und ausgegeben.

Hinweis

- Die Quell- und Zielstation sind identisch. Für den Ablauf spielt es also keine Rolle, ob von Station A zu Station B oder von Station B zu Station A Daten gesendet werden.
- Im Gateway wird in jedem Programmzyklus eine Zykluszeitmessung durchgeführt. Dabei wird ein Mittelwert über alle 100 Werte errechnet.

Beschreibung der Funktionselemente

Die folgende Tabelle beschreibt die einzelnen Funktionselemente, die maßgeblich zur Lösung der Aufgabenstellung beitragen:

Tabelle 2-3 Funktionselemente

Nr.	Funktionselemente	Aufgaben
1	Datensimulator	• Füllt bei jedem Aufruf ein Feld mit definierter Größe mit Characters von ‚A' bis ‚Z'.
2	Sendebaustein	• Übernimmt die Nutzdaten • Generiert ein Telegramm, welches aus Kopf- und Nutzdaten besteht • Sendet das Telegramm mit dem Baustein BSEND und verwendet dabei eine feste Parametrierung der ID und R_ID • Empfängt das Quittungstelegramm des Gateways • Verriegelt den Baustein, bis ein Quittungstelegramm vom Gateway ankommt
3	Headergenerator	• Erstellt aus gegebenen Kopf- und Nutzdaten ein zusammengesetztes Telegramm
4	Pointergenerator	• Erstellt aus einer Datenbaustein-Nummer, einer Startadresse und einer Länge einen ANY-Pointer auf einen Datenbaustein
5	Empfangseinheit(en)	• Empfangene Datentelegramme der Quell-/Zielstationen (mit BRCV-Bausteinen) • Auswertung des Telegrammkopfes und Eintragung der Daten in den jeweiligen Ringpuffer **Hinweis:** Ist der Ringpuffer voll, so werden keine weiteren Daten mehr empfangen.
6	Sendeeinheit(en)	• Liest den ältesten Eintrag aus ihrem Ringpuffer aus, sobald er frei ist • Sendet das Datentelegramm aus dem Ringpuffer an die ihm zugeordnete Station • Erstellung eines Quittungstelegramms, nachdem das Versenden des Datentelegramm abgeschlossen ist • Eintragen des Quittungstelegramms in den Ringpuffer der Quellstation
7	Ringpuffer	• Datenschnittstelle zwischen Empfangs- und Sendeeinheiten • Pufferung aller Telegramme, die an die jeweilige Station gesendet werden sollen • Realisierung als FIFO-Puffer
8	Fehlerbehandlung	• Fehlerbehandlung für alle auftretenden Fehler bei BSEND- / BRCV-Aufrufen • Fehlerprotokollierung in einem Protokoll-Datenbaustein • Auslesen einer projektierten Fehlerreaktion
9	Statistik	• Bildung eines Mittelwertes aus 100 Messwerten • In den Quell-/Zielstationen wird die Telegrammlaufzeit und die Zykluszeit gemessen. • Im Gateway wird die Zykluszeit gemessen.

Vorteile dieser Lösung

- Hoher Wiederverwendbarkeits-, Integrations- und Migrationswert aufgrund modularer Programmierstruktur
- Übermittlung großer Datenmengen (bis zu 64 kB) möglich, da S7-Kommunikation mit BSEND/BRCV verwendet wurde
- Subnetzunabhängig wegen S7-Kommunikation
- Geringer Programmier- und Projektieraufwand, da sich der Anwender nicht um IP- und/oder Profibusadressen kümmern muss. Ein Ablesen der Verbindungsprojektierung genügt.
- Automatisierter Quittierungsmechanismus. Daraus folgt eine gesicherte Übertragung sowie Fehleranalyse.
- Durch die Ringpuffer wird eine zeitliche Entkopplung der Quell- und Zielstationen erreicht, die die Verwendung mehrerer Quell- und Zielstationen in jedem Subnetz ermöglicht
- Zusätzliche Fehleranalyse ist wegen bereits vorhandener Strukturen möglich.

2.3 Alternativlösungen

2.3.1 Hardware-Alternativen

Quell-/Zielstationen

Als Quell-/Zielstationen können auch SPSen der S7-300-Reihe verwendet werden, sofern diese S7-Kommunikation mit BSEND/BRCV unterstützen.

Gateway

Als Gateway kann auch eine SPS der S7-400-Familie genutzt werden.

Dies ist vor allem bei einer größeren Anzahl an zu verwaltenden Stationen vorteilhaft (ab 8 Stationen), da hier eine signifikante Verlängerung der Zykluszeit der S7-300-CPU zu beobachten ist. Als Resultat reduziert sich der Telegrammdurchsatz.

Aufgrund der Nutzung einer S7-300 als Gateway können auf der Senderseite im PB-Netz sporadische Fehler auftreten, die auf den großen Zykluszeitunterschied zwischen S7-400-Sender und S7-300-Gateway zurückzuführen sind.

Mit einer performanteren CPU lassen sich diese Mali beseitigen (siehe dazu Kap. 3).

Die Hardwarekosten steigen dadurch natürlich ebenfalls an.

2.3.2 Bus-Alternativen

Profibus-Netz

Das Profibus-Netz kann auch mit höheren Baudraten betrieben werden.

Der Telegrammdurchsatz erhöht sich dadurch aber nicht wesentlich. Leistungsengpässe sind sowohl die Performance der CPU als auch der auf 187,5kBaud beschränkte Rückwandbus.

Eine andere Netzwahl (IE) erscheint nicht sinnvoll, da das Partnernetz auch ein IE-Netz ist und ein Gateway sodann nicht unbedingt nötig ist.

Industrial-Ethernet-Netz

Das Industrial-Ethernet-Netz kann mit anderen Modi (Halbduplex/10 MBit bis Vollduplex/100 MBit) betrieben werden.

Der Telegrammdurchsatz erhöht sich dadurch aber nicht wesentlich. Leistungsengpässe sind sowohl die Performance der CPU als auch der auf 187,5kBaud beschränkte Rückwandbus.

Eine andere Netzwahl (PB) erscheint nicht sinnvoll, da das Partnernetz auch ein PB-Netz ist und ein Gateway sodann nicht unbedingt nötig ist.

2.3.3 Protokoll-Alternativen

Protokoll

Bei Verwendung anderer Protokolle als dem S7-Protokoll sind entsprechende Kommunikationsbaugruppen notwendig.

Zudem muss die Implementierung der Anwenderschnittstellen neu erstellt werden. Bei Protokollen, die unterhalb der Ebene7 angesiedelt sind, sollte eine Ebene7-Quittierung aufgesetzt werden.

Dies führt zu erheblichem Mehraufwand.

2.4 Erforderliche Komponenten

2.4.1 Hardware-Komponenten

Folgende Hardware-Komponenten sind zur Realisierung der Applikation nötig.

Tabelle 2-4 Verwendete S7–400 Komponenten

Komponente	MLFB / Bestellnummer	Hinweis
Rack S7 400 UR2 (2 Stück)	6ES7400-1JA01-0AA0	Hier kann jedes UR oder CR verwendet werden.
PS 407 10 A (2 Stück)	6ES7407-0KA01-0AA0	Hier kann jede PS verwendet werden, die ausreichend für den Strombedarf ausgelegt ist.
CPU 416-2 DP (2 Stück)	6ES7416-2XK02-0AB0	FW V 1.2, o.ä. CPU
CP 443-5 Extended	6GK7443-5DX00-0XE0	Hier kann jeder Profibus CP der S7-400-Serie verwendet werden (aktuelle MLFB: 6GK7443-5DX02-0XE0).
CP 443-1	6GK7443-1EX11-0XE0	Hier kann jeder IE CP der S7-400-Serie verwendet werden.

Tabelle 2-5 Verwendete S7–300 Komponenten

Komponente	MLFB / Bestellnummer	Hinweis
PS 307 5A	6ES7307-1EA00-0AA0	Hier kann jede PS verwendet werden die ausreichend für den Strombedarf ausgelegt ist.
CPU 315-2 DP	6ES7315-2AG10-0AB0	FW V 2.0, o.ä. CPU
CP 342-5	6GK7342-5DA02-0XE0	Hier können CPs ab 6GK7342-5DA02-xxxx verwendet werden.
CP 343-1	6GK7343-1EX20-0XE0	Hier können ausschließlich CPs ab 6GK7343-1EX11-xxxx verwendet werden.
Micro Memory Card	6ES7953-8LM00-0AA0	Min 64k MMC

Tabelle 2-6 Verwendete PG Komponenten

Komponente	MLFB / Bestellnummer	Hinweis
Power PG	6ES7750-2CA52-4FB4	
On Board CP 5611	N/A	Teil der PG Hardware.

2.4.2 Software-Komponenten

Folgende Software-Komponenten sind zur Realisierung der Applikation nötig.

Tabelle 2-7: Verwendete Applikationskomponenten

Komponente	MLFB / Bestellnummer	Hinweis
Step 7 V 5.2	6ES7810-4CC06-0YX0	
NCM S7 Profibus	N/A	Teil der Step 7 SW.
SCL V5.1	6ES7811-1CC04-0YX0	Ist bei Anpassung der Nutzdatenlänge innerhalb des Projektes empfohlen, jedoch nicht unbedingt notwendig (s. Kap. 8.1)

2.4.3 Applikationssoftware-Komponenten

Die Applikation besteht aus dem Projekt „20983154_S7_Data_Gateway_CODE_v10.zip“ und beinhaltet folgende S7-Stationen:

- SIMATIC 400 – Station A
- SIMATIC 400 – Station B
- SIMATIC 300 - Gateway

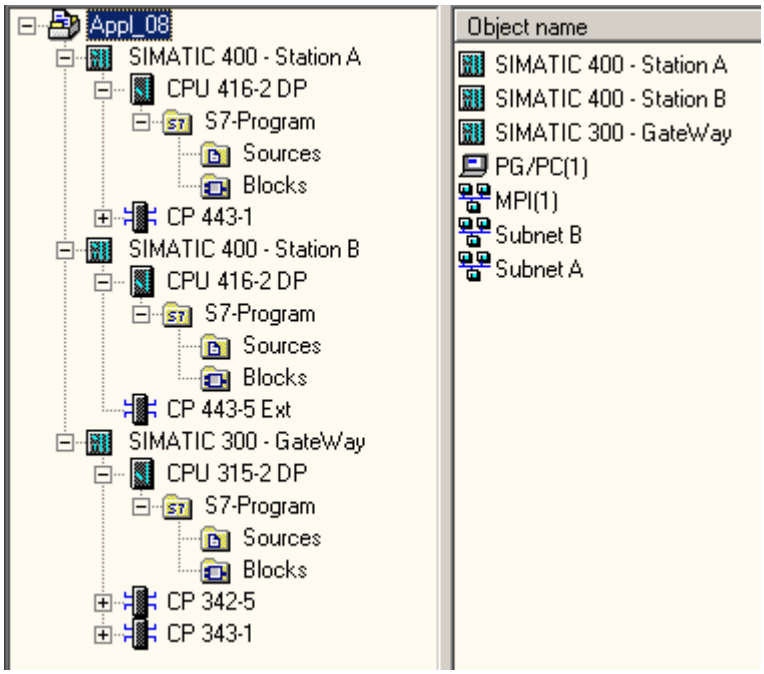


Bild 2-2

Die Programme in den S7-400-Stationen sind absolut identisch. Lediglich in der Variablentabelle sind aufgrund unterschiedlicher Verbindungsprojektierungen Unterschiede vorhanden.

Die S7-300 übernimmt die Aufgabe des Gateways. Sie unterscheidet sich also grundsätzlich von den S7-400-Stationen.

Nachfolgend finden Sie Tabellen, die die verwendeten Funktionselemente den Bausteinen zuordnen und dabei die Funktionalität stichpunktartig beschreiben.

Eine detaillierte Funktionsbeschreibung finden Sie im Teil A2.

Bausteine der Quell-/Zielstationen (FBs, FCs, SFBs, SFCs)

Tabelle 2-8

Funktionselement	Name	Funktionsbeschreibung	Technische Daten	Ablaufumgebung
Zentrale Ablaufumgebung	FB1 „Manager“	Steuert die Quell-/Zielstationen und ruft unter Nutzung von DBs folgende Funktionen auf: • Statistik: Erstellt die durchschnittlichen Telegrammlauf- und Zykluszeiten: - FC 22 „TimeStamp“ - FB3 „CalcStatistic“ • Simulation der Daten: Daten werden erstellt und in einen anwenderspezifischen Datenbaustein eingetragen: - DB4 „UserData“ - FC21 „GenPointerOnDB“ - FB 28 „DataSimulator“ • Senden der Daten: Erstellt aus vorgegebenen Zieldaten einen Telegrammkopf und sendet Kopf mit Nutzdaten zum Gateway: - FB25 „SendBlock“ • Empfang der Daten: Empfängt die Nutzdaten: - DB5 „RcvDB“ - FC21 „GenPointerOnDB“ - SFB 13 „BRCV“ - FB 33 „Error Control“	• Erstellsprache: AWL • Ladespeicher: 1036 Bytes • Arbeitsspeicher: 932 Bytes	OB1
	DB1	Instanz-DB zum FB1 „Manager“	• Erstellsprache: DB • Ladespeicher: 128 Bytes • Arbeitsspeicher: 54 Bytes	

Funktionselement	Name	Funktionsbeschreibung	Technische Daten	Ablaufumgebung
	SFB 13 „BRCV“	System-Baustein zum Empfangen der Daten. Empfängt Datentelegramme.	• Erstellsprache: AWL • Ladespeicher: - Bytes • Arbeitsspeicher: - Bytes	FB1 „Manager“
	DB 14	Instanz-DB zum SFB 14 „BRCV“	• Erstellsprache: DB • Ladespeicher: 124 Bytes • Arbeitsspeicher: 62 Bytes	
	FB33 „Error Control“	Protokolliert aufgetretene Fehler des BRCV-Aufrufes und erstellt eine Reaktionsmeldung.	• Erstellsprache: AWL • Ladespeicher: 658 Bytes • Arbeitsspeicher: 540 Bytes	FB1 „Manager“
	DB33	Instanz-DB zum FB 33 „Error Control“	• Erstellsprache: DB • Ladespeicher: 156 Bytes • Arbeitsspeicher: 64 Bytes	
	DB5 „RcvDB“	Empfangsfach für empfangene Datentelegramme	• Erstellsprache: DB • Ladespeicher: 722 Bytes • Arbeitsspeicher: 252 Bytes	FB1 „Manager“
Statistik	FC22 „TimeStamp“	Ermittelt mit den Funktionen SFC1 „READ_CLK“ und FC8 „DT_TOD“ die aktuelle Zeit der CPU. Es wird die Anfangs- und die Endzeit vor dem Senden und nach dem Empfangen von Daten ermittelt.	• Erstellsprache: AWL • Ladespeicher: 183 Bytes • Arbeitsspeicher: 128 Bytes	FB1 „Manager“
	SFC1 „READ_CLK“	System-Funktion: Liest aktuelle CPU-Zeit aus	• Erstellsprache: AWL • Ladespeicher: - Bytes • Arbeitsspeicher: - Bytes	FC 22 „TimeStamp“
	FC8 „DT_TOD“	Ladbare Funktion: Wandelt Datum und Zeit in Zeit um.	• Erstellsprache: AWL • Ladespeicher: 312 Bytes • Arbeitsspeicher: 242 Bytes	FC 22 „TimeStamp“

Funktionselement	Name	Funktionsbeschreibung	Technische Daten	Ablaufumgebung
	FB3 „CalcStatistic“	Es wird die Zeitdifferenz aus 2 Zeiten errechnet und über einen Mittelwertbildner (FB2 „STATISTIC“) der Mittelwert, der höchste und der niedrigste Wert aus 100 Werten, bestimmt.	• Erstellsprache: AWL • Ladespeicher: 464 Bytes • Arbeitsspeicher: 374 Bytes	FB1 „Manager“
	DB8	Instanz-DB zum FB3 „CalcStatistic“	• Erstellsprache: DB • Ladespeicher: 160 Bytes • Arbeitsspeicher: 78 Bytes	
	FB2 „STATISTIC“	Es wird der Mittelwert, der höchste und der niedrigste aus einer Reihe von Werten, bestimmt.	• Erstellsprache: SCL • Ladespeicher: 388 Bytes • Arbeitsspeicher: 306 Bytes	FB3 „CalcStatistic“
	DB6	Instanz-DB zum FB2 „STATISTIC“. Enthält Daten für Telegrammlaufzeiten.	• Erstellsprache: DB • Ladespeicher: 146 Bytes • Arbeitsspeicher: 78 Bytes	
	DB7	Instanz-DB zum FB2 „STATISTIC“. Enthält Daten für Zykluszeiten.	• Erstellsprache: DB • Ladespeicher: 146 Bytes • Arbeitsspeicher: 78 Bytes	
Datensimulator	DB4 „UserData“	Datenbaustein, der die Nutzdaten (hier: simulierte Daten) enthält	• Erstellsprache: DB • Ladespeicher: 680 Bytes • Arbeitsspeicher: 236 Bytes	FB1 „Manager“
	FC21 „GenPointe-rOnDB“	Erzeugt einen Any pointer auf einen gegebenen Daten-baustein mit der Funktion FC20 „PointerGenerator“	• Erstellsprache: AWL • Ladespeicher: 214 Bytes • Arbeitsspeicher: 152 Bytes	FB1 „Manager“
	FB 28 „DataSimula-tor“	Trägt in einen gegebenen Datenbaustein Characters von ‚A’ bis ‚Z’ ein.	• Erstellsprache: AWL • Ladespeicher: 266 Bytes • Arbeitsspeicher: 190 Bytes	FB1 „Manager“
	DB 28	Instanz-DB zum FB 28 „DataSimulator“	• Erstellsprache: DB • Ladespeicher: 110 Bytes • Arbeitsspeicher: 52 Bytes	

Funktionselement	Name	Funktionsbeschreibung	Technische Daten	Ablaufumgebung
	UDT 101 „UserData“	Beschreibt die Struktur der Nutzdaten	• Erstellsprache: AWL • Ladespeicher: - Bytes • Arbeitsspeicher: - Bytes	FB1 „Manager“
Sendebaustein	FB25 „SendBlock“	Der Sendebaustein erstellt aus übergebenen Daten ein Telegramm mit Kopf- und Nutzerdaten und sendet dieses zum Ziel. Der Sendebaustein ist erst nach Erhalt einer Quittung oder einem Fehler wieder sendebereit.	• Erstellsprache: AWL • Ladespeicher: 1514 Bytes • Arbeitsspeicher: 1340 Bytes	FB1 „Manager“
	DB25	Instanz-DB zum FB 25 „SendBlock“	• Erstellsprache: DB • Ladespeicher: 220 Bytes • Arbeitsspeicher: 80 Bytes	
	DB2 “AckDB”	Empfangsfach für Quittungstelegramme	• Erstellsprache: DB • Ladespeicher: 122 Bytes • Arbeitsspeicher: 52 Bytes	FB25 „SendBlock“
	DB3 “SendDB”	Sendefach der Telegramms	• Erstellsprache: DB • Ladespeicher: 722 Bytes • Arbeitsspeicher: 252 Bytes	FB25 „SendBlock“
	SFC 20 „BlkMov“	Kopiert Telegrammdaten in eine definierte Struktur.	• Erstellsprache: AWL • Ladespeicher: - Bytes • Arbeitsspeicher: - Bytes	FB25 „SendBlock“
	FC20 „PointerGene-rator“	Erzeugt einen Any pointer auf einen gegebenen Daten-baustein mit Hilfe einer Länge und einer Startadresse	• Erstellsprache: AWL • Ladespeicher: 166 Bytes • Arbeitsspeicher: 106 Bytes	FC21 „GenPointe-rOnDB“
	FB21 “Headergene-rator”	Generiert Kopfdaten und fügt sie an den Anfang eines existierenden Datenbereichs.	• Erstellsprache: AWL • Ladespeicher: 612 Bytes • Arbeitsspeicher: 474 Bytes	FB25 „SendBlock“

Funktionselement	Name	Funktionsbeschreibung	Technische Daten	Ablaufumgebung
	DB 21	Instanz-DB zum FB21	• Erstellsprache: DB • Ladespeicher: 200 Bytes • Arbeitsspeicher: 84 Bytes	
	SFB 12 „BSEND“	System-Baustein zum Senden der Daten	• Erstellsprache: AWL • Ladespeicher: - Bytes • Arbeitsspeicher: - Bytes	FB25 „SendBlock“
	DB 12	Instanz-DB zum SFB 12 „BSEND“	• Erstellsprache: DB • Ladespeicher: 126 Bytes • Arbeitsspeicher: 62 Bytes	
	SFB 13 „BRCV“	System-Baustein zum Empfangen von Quittungstele-grammen	• Erstellsprache: AWL • Ladespeicher: - Bytes • Arbeitsspeicher: - Bytes	FB25 „SendBlock“
	DB 13	Instanz-DB zum SFB 13 „BRCV“	• Erstellsprache: DB • Ladespeicher: 124 Bytes • Arbeitsspeicher: 62 Bytes	
	FB33 „Error Control“	Protokolliert aufgetretene Fehler mit den BSEND/BRCV-Aufrufen und erstellt eine Reaktionsmeldung.	• Erstellsprache: AWL • Ladespeicher: 658 Bytes • Arbeitsspeicher: 540 Bytes	FB25 „SendBlock“
	DB33	Instanz-DB zum FB 33 „Error Control“	• Erstellsprache: DB • Ladespeicher: 156 Bytes • Arbeitsspeicher: 64 Bytes	
	DB99 “Glob-Constants”	Enthält die Werte für die Fehlerstati, die vom Sendeblock zurückgeliefert werden	• Erstellsprache: DB • Ladespeicher: 100 Bytes • Arbeitsspeicher: 42 Bytes	FB25 „SendBlock“
	UDT100 „TelHeader”	Definiert die Struktur der Kopfdaten	• Erstellsprache: AWL • Ladespeicher: - Bytes • Arbeitsspeicher: - Bytes	

Funktionselement	Name	Funktionsbeschreibung	Technische Daten	Ablaufumgebung
	UDT 102 „Sendstruct“	Definiert die Struktur des Sendefachs	• Erstellsprache: AWL • Ladespeicher: - Bytes • Arbeitsspeicher: - Bytes	
	UDT 103 „IDData“	Definiert die Struktur der Identifikation von Quelle und Ziel	• Erstellsprache: AWL • Ladespeicher: - Bytes • Arbeitsspeicher: - Bytes	
Headergenerator	FB 21 „Headergene-rator“	Generiert Kopfdaten und fügt sie an den Anfang eines existierenden Datenbereichs.	• Erstellsprache: AWL • Ladespeicher: 612 Bytes • Arbeitsspeicher: 474 Bytes	FB 21 „Headergene-rator“
	DB 21	Instanz-DB zum FB 21 „Headergenerator“	• Erstellsprache: DB • Ladespeicher: 200 Bytes • Arbeitsspeicher: 84 Bytes	
	SFC1 „READ_CLK“	System-Funktion: Liest aktuelle CPU-Zeit aus	• Erstellsprache: AWL • Ladespeicher: - Bytes • Arbeitsspeicher: - Bytes	FB 21 „Headergene-rator“
	FC8 „DT_TOD“	Ladbare Funktion: Wandelt Datum und Zeit in Zeit um.	• Erstellsprache: AWL • Ladespeicher: 312 Bytes • Arbeitsspeicher: 242 Bytes	FB 21 „Headergene-rator“
	SFC 20 „BlkMov“	Kopiert Headerdaten in den Telegrammdatenbereich	• Erstellsprache: AWL • Ladespeicher: - Bytes • Arbeitsspeicher: - Bytes	FB 21 „Headergene-rator“
	UDT100 „TelHeader”	Definiert die Struktur der Kopfdaten	• Erstellsprache: AWL • Ladespeicher: - Bytes • Arbeitsspeicher: - Bytes	
Pointergenerator	FC21 „GenPointe-rOnDB“	Erzeugt einen Any pointer auf einen gegebenen Daten-baustein mit der Funktion FC20 „PointerGenerator“	• Erstellsprache: AWL • Ladespeicher: 214 Bytes • Arbeitsspeicher: 152 Bytes	unterschiedlich

Funktionselement	Name	Funktionsbeschreibung	Technische Daten	Ablaufumgebung
	FC20 „PointerGenerator“	Erzeugt einen Any pointer auf einen gegebenen Datenbaustein mit Hilfe einer Länge und einer Startadresse	• Erstellsprache: AWL • Ladespeicher: 166 Bytes • Arbeitsspeicher: 106 Bytes	FC21 „GenPointerOnDB“
Fehlerbehandlung	FB33 „Error Control“	Protokolliert aufgetretene Fehler mit BSEND/BRCV-Aufrufen und erstellt eine Reaktionsmeldung. Es werden die letzten 20 Fehler protokolliert.	• Erstellsprache: AWL • Ladespeicher: 658 Bytes • Arbeitsspeicher: 540 Bytes	unterschiedlich
	DB33	Instanz-DB zum FB 33 „Error Control“	• Erstellsprache: DB • Ladespeicher: 156 Bytes • Arbeitsspeicher: 64 Bytes	
	SFC1 „READ_CLK“	System-Funktion: Liest aktuelle CPU-Zeit aus	• Erstellsprache: AWL • Ladespeicher: - Bytes • Arbeitsspeicher: - Bytes	FB33 „Error Control“
	FC8 „DT_TOD“	Ladbare Funktion: Wandelt Datum und Zeit in Zeit um.	• Erstellsprache: AWL • Ladespeicher: 312 Bytes • Arbeitsspeicher: 242 Bytes	FB33 „Error Control“
	DB 110 „Protocoll DB“	Enthält die letzten 20 Fehler.	• Erstellsprache: DB • Ladespeicher: 882 Bytes • Arbeitsspeicher: 396 Bytes	FB33 „Error Control“
	DB 111 „REACT_DB_BSEND“	Enthält die Projektierdaten, um festzustellen, wie auf welchen Fehlerstatus zu reagieren ist. Der Fehlerstatus entspricht den Stati des BSEND.	• Erstellsprache: DB • Ladespeicher: 214 Bytes • Arbeitsspeicher: 84 Bytes	FB33 „Error Control“
	DB 112 „REACT_DB_BRCV“	Enthält die Projektierdaten, um festzustellen, wie auf welchen Fehlerstatus zu reagieren ist. Der Fehlerstatus entspricht den Stati des BRCV.	• Erstellsprache: DB • Ladespeicher: 182 Bytes • Arbeitsspeicher: 72 Bytes	FB33 „Error Control“
	UDT 109 „Errorlocationstruct“	Definiert, wie der Fehlerort zu spezifizieren ist.	• Erstellsprache: AWL • Ladespeicher: - Bytes • Arbeitsspeicher: - Bytes	

Funktionselement	Name	Funktionsbeschreibung	Technische Daten	Ablaufumgebung
	UDT 110 „Errorstruct“	Definiert die Struktur im Fehler-DB	• Erstellsprache: AWL • Ladespeicher: - Bytes • Arbeitsspeicher: - Bytes	
	UDT 111 “Table_Status_Reaction”	Definiert die Struktur der Daten, wie auf bestimmte Fehler zu reagieren ist	• Erstellsprache: AWL • Ladespeicher: - Bytes • Arbeitsspeicher: - Bytes	

Anwenderbausteine des Gateways

Tabelle 2-9

Funktionselement	Name	Funktionsbeschreibung	Technische Daten	Ablaufumgebung
Zentrale Ablaufumgebung	FC1 „Manager“	• Ausführen der Statistikfunktionalität: Alle 100 Zyklen werden die Ergebnisse der Statistik-Funktion ausgegeben: FB6 „STATISTIC“ • Status der Ringpuffer: FB 5 „Buffermanager“ Wenn ein Ringpuffer voll ist, so wird nur noch gesendet und nicht mehr empfangen. • Aufruf der Empfangseinheiten: FB4 „ReceiveUnit“ • Aufruf der Sendeeinheiten: FC 3 „SndUnits“	• Erstellsprache: AWL • Ladespeicher: 862 Bytes • Arbeitsspeicher: 720 Bytes	OB1
	DB99 „Glob-Constants“	Enthält die Konstanten der Applikation.	• Erstellsprache: DB • Ladespeicher: 862 Bytes • Arbeitsspeicher: 720 Bytes	
Statistik	FB6 „STATISTIC“	Es wird der Mittelwert, der höchste und der niedrigste aus einer Reihe von Werten, bestimmt. Als Werte werden die Zykluszeiten erfasst.	• Erstellsprache: SCL • Ladespeicher: 388 Bytes • Arbeitsspeicher: 306 Bytes	FC1 „Manager“
	DB6	Instanz-DB zum FB6 „STATISTIC“. Enthält Daten für Zykluszeiten.	• Erstellsprache: DB • Ladespeicher: 146 Bytes • Arbeitsspeicher: 78 Bytes	
Ringpuffer	FB5 „Buffermanager“	Der Puffermanager wertet die Aktion aus, die auf einen bestimmten Puffer ausgeführt werden soll und verteilt dementsprechend die Eingangsparameter and den FB3 „Bufferhandler“	• Erstellsprache: AWL • Ladespeicher: 618 Bytes • Arbeitsspeicher: 532 Bytes	Unterschiedlich
	DB5	Instanz-DB zum FB5 „Buffermanager“	• Erstellsprache: DB • Ladespeicher: 116 Bytes • Arbeitsspeicher: 56 Bytes	

Funktionselement	Name	Funktionsbeschreibung	Technische Daten	Ablaufumgebung
	FB3 „Bufferhandler“	Der Bufferhandler wertet anhand eines switch-case den Ringpuffer aus, auf dem eine bestimmte Aktion ausgeführt werden soll.	• Erstellsprache: AWL • Ladespeicher: 1186 Bytes • Arbeitsspeicher: 1068 Bytes	FB5 „Buffermanager“
	DB 3	Instanz-DB zum FB3 „Bufferhandler“	• Erstellsprache: DB • Ladespeicher: 116 Bytes • Arbeitsspeicher: 56 Bytes	
	FB2 „Ringbuffer“	Funktion verwaltet einen Ringpuffer. Je nach geforderter Aktion werden Daten gelesen, geschrieben, gelöscht oder der Status ausgelesen.	• Erstellsprache: SCL • Ladespeicher: 2194 Bytes • Arbeitsspeicher: 1652 Bytes	FB3 „Bufferhandler“
	DB 60, 61, 62, 63, 64, 65, 66, 67 „Ringbuffer_Datx“	Instanz-DBs zu den Ringpuffern	• Erstellsprache: DB • Ladespeicher: 2730 Bytes • Arbeitsspeicher: 2224 Bytes	
	FB1 „VAR_LEN“	Berechnet die Länge einer Variablen in Bytes	• Erstellsprache: SCL • Ladespeicher: 468 Bytes • Arbeitsspeicher: 376 Bytes	FB 2 „Ringbuffer“
	DB1 „VAR_LEN_DAT“	Instanz-DB zum FB 1 „VAR_LEN“	• Erstellsprache: DB • Ladespeicher: 100 Bytes • Arbeitsspeicher: 50 Bytes	
	FC 10 “GetDBNr”	Ermittelt die Nummer eines DBs	• Erstellsprache: AWL • Ladespeicher: 106 Bytes • Arbeitsspeicher: 52 Bytes	FB 2 „Ringbuffer“
	UDT 102 „Telstruct“	Definiert die Struktur des Sende-/Empfangsfachs, also die Größe der zu speichernden Daten.	• Erstellsprache: AWL • Ladespeicher: - Bytes • Arbeitsspeicher: - Bytes	

Funktionselement	Name	Funktionsbeschreibung	Technische Daten	Ablaufumgebung
Empfangseinheiten	FB 4 „ReceiveUnit”	Ruft den BRCV-Baustein auf, der einer bestimmten Empfangseinheit zugeordnet ist. Trägt die empfangenen Daten in den entsprechenden Ringpuffer ein.	• Erstellsprache: AWL • Ladespeicher: 1626 Bytes • Arbeitsspeicher: 1478 Bytes	
	DB 4	Instanz-DB zum FB 4 “ReceiveUnit”	• Erstellsprache: DB • Ladespeicher: 140 Bytes • Arbeitsspeicher: 52 Bytes	
	FC21 „GenPointerOnDB“	Erzeugt einen Any pointer auf einen gegebenen Datenbaustein (das Empfangsfach) mit der Funktion FC20 „PointerGenerator“.	• Erstellsprache: AWL • Ladespeicher: 214 Bytes • Arbeitsspeicher: 152 Bytes	FB4 „ReceiveUnit“
	DB99 „Glob-Constants“	Enthält die Konstanten der Applikation.	• Erstellsprache: DB • Ladespeicher: 862 Bytes • Arbeitsspeicher: 720 Bytes	
	FB 13 „BRCV300”	Empfängt die Datentelegramme.	• Erstellsprache: AWL • Ladespeicher: 6098 Bytes • Arbeitsspeicher: 5258 Bytes	FB 4 “ReceiveUnit”
	DB 20, 21, 22, 23, 24, 25, 26, 27 „BRCV_DataIDBx“	Instanz-DBs für BRCV-Aufrufe. Jeder Instanz-DB repräsentiert eine Empfangseinheit.	• Erstellsprache: DB • Ladespeicher: 662 Bytes • Arbeitsspeicher: 374 Bytes	
	SFC 20 „BlkMov“	Kopiert Telegrammdaten in einen Telegrammkopf, damit das Telegrammziel ermittelt werden kann.	• Erstellsprache: AWL • Ladespeicher: - Bytes • Arbeitsspeicher: - Bytes	FB 4 “ReceiveUnit”
	DB 50, 51, 52, 53, 54, 55, 56, 57 “ReceivedDatax”	Empfangsfächer der BRCV-Bausteine	• Erstellsprache: DB • Ladespeicher: 722 Bytes • Arbeitsspeicher: 252 Bytes	FB 4 “ReceiveUnit”

Funktionselement	Name	Funktionsbeschreibung	Technische Daten	Ablaufumgebung
	FB 5 „Buffermanager“	Trägt Daten in den angegebenen Ringpuffer ein.	• Erstellsprache: AWL • Ladespeicher: 618 Bytes • Arbeitsspeicher: 532 Bytes	FB 4 “ReceiveUnit”
	DB 5	Instanz-DB zum FB 5 „Buffermanager“	• Erstellsprache: DB • Ladespeicher: 116 Bytes • Arbeitsspeicher: 56 Bytes	
	UDT100 „TelHeader”	Definiert die Struktur der Kopfdaten	• Erstellsprache: AWL • Ladespeicher: - Bytes • Arbeitsspeicher: - Bytes	
	UDT 102 „Telstruct“	Definiert die Struktur des Sende-/Empfangsfachs	• Erstellsprache: AWL • Ladespeicher: - Bytes • Arbeitsspeicher: - Bytes	
	UDT 103 „IDData“	Definiert die Struktur der Identifikation von Quelle und Ziel	• Erstellsprache: AWL • Ladespeicher: - Bytes • Arbeitsspeicher: - Bytes	
Sendeeinheiten	FC 3 „SndUnits“	Wählt die gewünschte Sendeeinheit aus	• Erstellsprache: AWL • Ladespeicher: 1146 Bytes • Arbeitsspeicher: 978 Bytes	FC1 „Manager“
	FB 7 „SndUnit“	Dieser Baustein stellt die zentrale Funktionalität für den Routing- und Quittierungsmechanismus dar. Je nach Zustand und Ringpuffer-Eintrag werden sowohl Daten- als auch Quittungstelegramme generiert.	• Erstellsprache: AWL • Ladespeicher: 3768 Bytes • Arbeitsspeicher: 3504 Bytes	FC 3 „SndUnits“
	DB 70, 71, 72, 73, 74, 75, 76, 77	Instanz-DBs zum FB 7 „SndUnit“. Diese entsprechen der Instanz einer Sendeeinheit.	• Erstellsprache: AWL • Ladespeicher: 142 Bytes • Arbeitsspeicher: 54 Bytes	

Funktionselement	Name	Funktionsbeschreibung	Technische Daten	Ablaufumgebung
	FC21 „GenPointerOnDB“	Erzeugt einen Any pointer auf einen gegebenen Datenbaustein (das Sendefach) mit der Funktion FC20 „PointerGenerator“	• Erstellsprache: AWL • Ladespeicher: 214 Bytes • Arbeitsspeicher: 152 Bytes	FB 7 „SndUnit“
	FB 5 „Buffermanager“	Liest Daten aus dem angegebenen Ringpuffer aus. Generiert ein Quittungstelegramm und schreibt dieses in den korrespondierenden Ringpuffer.	• Erstellsprache: AWL • Ladespeicher: 618 Bytes • Arbeitsspeicher: 532 Bytes	FB 7 „SndUnit“
	DB 5	Instanz-DB zum FB 5 „Buffermanager“	• Erstellsprache: DB • Ladespeicher: 116 Bytes • Arbeitsspeicher: 56 Bytes	
	SFC 20 „BlkMov“	Kopiert Telegrammdaten in eine Telegrammkopfstrukur, um ein Quittungstelegramm zu generieren.	• Erstellsprache: AWL • Ladespeicher: - Bytes • Arbeitsspeicher: - Bytes	FB 7 “SndUnit”
	DB99 „GlobConstants“	Enthält die Konstanten der Applikation.	• Erstellsprache: DB • Ladespeicher: 862 Bytes • Arbeitsspeicher: 720 Bytes	FB 7 “SndUnit”
	DB 90 „HeadDB“	Enthält temporär die Kopfdaten als Schmiermerker.	• Erstellsprache: DB • Ladespeicher: 122 Bytes • Arbeitsspeicher: 52 Bytes	FB 7 “SndUnit”
	FB 12 „BSEND300“	Ladbarer Funktionsbaustein zum Senden von Daten mittels BSEND. Je nach Zustand werden Daten- oder Quittungstelegramme gesendet.	• Erstellsprache: AWL • Ladespeicher: 6110 Bytes • Arbeitsspeicher: 5284 Bytes	FB 7 “SndUnit”
	DB 10, 11, 12, 13, 14, 15, 16, 17 „BsendACK_IDBx“	Instanz-DBs für die BSEND-Aufrufe, die Datentelegramme senden.	• Erstellsprache: DB • Ladespeicher: 656 Bytes • Arbeitsspeicher: 372 Bytes	

Funktionselement	Name	Funktionsbeschreibung	Technische Daten	Ablaufumgebung
	DB 30, 31, 32, 33, 34, 35, 36, 37 „BsendDATA_IDBx“	Instanz-DB für die BSEND-Aufrufe, die Quittungstelegramme senden.	• Erstellsprache: DB • Ladespeicher: 656 Bytes • Arbeitsspeicher: 372 Bytes	
	DB 40, 41, 42, 43, 44, 45, 46, 47 „SendDatax“	Sendefach der jeweiligen Sendeeinheit	• Erstellsprache: DB • Ladespeicher: 722 Bytes • Arbeitsspeicher: 252 Bytes	FB 7 “SndUnit”
	UDT100 „TelHeader”	Definiert die Struktur der Kopfdaten	• Erstellsprache: AWL • Ladespeicher: - Bytes • Arbeitsspeicher: - Bytes	
	UDT 102 „Telstruct“	Definiert die Struktur des Sende-/Empfangsfachs	• Erstellsprache: AWL • Ladespeicher: - Bytes • Arbeitsspeicher: - Bytes	
	UDT 103 „IDData“	Definiert die Struktur der Identifikation von Quelle und Ziel	• Erstellsprache: AWL • Ladespeicher: - Bytes • Arbeitsspeicher: - Bytes	
Pointergenerator	Siehe Quell-/Zielstation			
Fehlerbehandlung	Siehe Quell-/Zielstation			

Projektierungselemente des Gateways und der Quell-/Zielstation

Tabelle 2-10

Element	Name	Beschreibung	Engineering-Tool
Kommunikationsverbindung	S7-Verbindung	Die S7-Verbindung ist die projektierte logische Verknüpfung zwischen 2 Kommunikationspartnern. Mit ihr wird der Startpunkt, der Kommunikationsweg und der Zielpunkt der Kommunikation festgelegt.	SIMATIC Manager NetPro
CPU und MPI-Bus	Uhrzeitsynchronisation	Zur Uhrzeitsynchronisation zwischen den genutzten CPU Baugruppen wird ein möglichst zeitgenauer Verlauf benötigt.	SIMATIC Manager HW-Config.

Aufwand Programmierung/Projektierung

- Um eine vergleichbare Applikation nach zu implementieren, wird bei mittleren SIMATIC-Vorkenntnissen ein Aufwand von10-15MT geschätzt.
- Zur Einarbeitung in die vorliegende Applikation sollte mit 1-2MT gerechnet werden.
- Um eine tiefer gehende Fehlerbehandlung (wie z.B. das Wiederholen eines Sendeauftrags nach einem Fehler; s. Kap. 8 respektive 8.4) zu realisieren, ist in etwa ein Aufwand von 2-3MT zu erwarten.
  Dabei wird vorausgesetzt, dass sich der Bearbeiter in die vorliegende Applikation eingearbeitet hat.
- Der Projektieraufwand ist gering, da er sich lediglich auf die Verbindungsprojektierung in NETPro und die Uhrzeitsynchronisation in HWConfig beschränkt.
- Alle zu verwendenden Engineering-Tools sind bei Verwendung standardisierter Programmiergeräte bereits im SIMATIC-Basispaket enthalten.

3 Leistungseckdaten

Einleitung

In diesem Kapitel geben wir Ihnen einen Überblick über die Leistungsfähigkeit dieser Applikation und der verwendeten Komponenten.

Inhalt Kapitel Leistungsdaten

Tabelle 3-1

Kap.-Nr.	Kapitelüberschrift	Hier erfahren Sie...
3.1	Leistungseckdaten der Hardware ohne applikative Komponenten	... welche Leistungseckdaten der Hardware zu beachten sind.
3.2	Leistungseckdaten des Protokolls	... welche Leistungseckdaten bzgl. des Protokolls zu beachten sind.
3.3	Leistungseckdaten der Gesamtapplikation	... welche Leistungseckdaten die Gesamtapplikation aufweist..

3.1 Leistungseckdaten der Hardware ohne applikative Komponenten

Da die Kommunikationslast die Gateway-CPU trifft, wird diese explizit aufgeführt.
Besondere Beachtung gilt der Anzahl an Verbindungen sowie dem Speicherausbau, da vor allem die Ringpuffer einen recht großen Speicherbedarf haben können.
Auf die Leistungsdaten der CPUs in den Quell-/Zielstationen und den Kommunikationsprozessoren wird nicht näher eingegangen. Die zugehörigen Daten können Sie den entsprechenden Handbüchern entnehmen.

Als weiterer Leistungsengpass ist die Physik der verwendeten Netze zu sehen.
Da der MPI-Bus ausschliesslich für die Uhrzeitsynchronisation und den PG-AS-Betrieb (Online gehen) genutzt wird, ist die Busbelastung des MPI-Bus gering. Die MPI-Bus-Belastung wird somit nicht näher betrachtet.

Hinweis zur Leistungsoptimierung

Um eine Leistungsoptimierung seitens der Telegrammlaufzeit zu erreichen, sollte das Gateway eine hochwertige S7-400-CPU sein.

In der derzeitigen Gateway-Programmstruktur werden alle Stationen in einem Zyklus behandelt. D.h. die BSEND-/BRCV-Bausteine werden alle in **einem** Zyklus aufgerufen, was die Zykluszeit belastet. Daraus resultiert u.a. die hohe Telegrammlaufzeit.

Ändert man die Programmstruktur im Gateway dahingehend, dass die BSEND-/BRCV-Aufrufe über mehrere Zyklen verteilt werden, so ist nicht unbedingt eine Erhöhung der Telegrammlaufzeit zu erwarten, da

- jeder BSEND-Aufruf mindestens 2 Zyklen benötigt
- jeder Station 2 BSEND-Aufrufe (Daten- und Quittungs-BSEND) zugeordnet sind
- jeder Station ein BRCV-Aufruf zugeordnet ist und
- zwischen den BSEND-/BRCV-Aufrufen zusätzlicher Verwaltungsaufwand entsteht

Bei 2 Stationen würden somit für ein Telegramm etwa 8-10 Zyklen benötigt werden, um ein Telegramm von Station1 zu Station2 zu senden. Bei einer durchschnittlichen Zykluszeit von etwa 7-10ms und gleicher Busphysik resultiert eine ähnliche Telegrammlaufzeit.

Daraus folgt, dass im Gateway zwar die Zykluszeit, aber nicht die Telegrammlaufzeit signifikant verringert werden kann.

Die in der Applikation verwendeten Parameter sind mit gekennzeichnet.

Leistungseckdaten des Profibus

Tabelle 3-2

Kriterium	Leistungseckdaten	Zusätzlicher Hinweis
Übertragungsgeschwindigkeit	9,6 kbit/s 19,2 kbit/s 45,45 kbit/s 93,75 kbit/s 187,5 kbit/s 500 kbit/s 1,5 Mbit/s 3 Mbit/s 6 Mbit/s 12 Mbit/s	Die Übertragungsgeschwindigkeit ist durch das Projektierungstool einstellbar. Das Projektierungstool ist im Fall der S7 die SIMATIC Step 7 Software. In der Applikation ist eine Baudrate von 1,5MBaud eingestellt.
Übertragungsprinzip	Token / Token Passing	Standardisiert in der EN 50170
Netzausdehnung	0,2m - >100 km	Die Netzausdehnung ist durch die Verwendung von optischen Komponenten erreichbar.
Verwendete Busphysik	RS 485 (2 draht Leiter), LWL	
Anzahl der Teilnehmer am Bus / Segment	126 Teilnehmer pro Bus / 32 Teilnehmer pro Segment	
Nutzbare Protokolle	DP, FDL, FMS, S7, TF	

Leistungseckdaten des Industrial Ethernet

Tabelle 3-3

Kriterium	Leistungseckdaten	Zusätzlicher Hinweis
Übertragungsgeschwindigkeit	10 Mbit/s Halb- / Vollduplex 100 Mbit/s Halb- / Vollduplex 11 Mbit/s (im Wireless Bereich)	In der Applikation ist eine Baudrate von 100MBaud bei Vollduplex eingestellt.
Übertragungsprinzip	CSMA/CD / CSMA/CA	Standardisiert in der IEEE 802.3 und IEEE 802.11
Netzausdehnung	1,5m - >100 km	Die Netzausdehnung ist durch die Verwendung von optischen Komponenten erreichbar.
Verwendete Busphysik	• Co- bzw. Triaxialnetzwerk (bis 10 MBaud) • Twisted Pair (RJ45) • LWL	
Anzahl der Teilnehmer am Bus	Bis zu 1024 an einer Collision Domain.	Die Anzahl der Teilnehmer ist durch Kopplung verschiedener Domains über Router oder Switches erweiterbar.
Nutzbare Protokolle / Stacks	Die SIMATIC unterstützt den • ISO Stack Mit den Protokollen: • ISO Transport • S7 Protokoll sowie den: • TCP Stack Mit den Protokollen: • UDP • TCP • RFC 1006 • PPP • FTP • SMTP • S7 Protokoll auf Basis des RFC 1006	Die Auswahl der Protokolle ist durch die CPs beschränkt.

3.2 Leistungseckdaten des S7-Protokolls

Tabelle 3-4

Kriterium	Leistungseckdaten	Zusätzlicher Hinweis
Kommunikationsebene	Ebene 7 nach ISO-OSI-Referenzmodell	
Kopplungsart	Peer to Peer	
Kopplungsfunktionalität	Client / Client bzw. Client / Server	Dies ist abhängig vom genutzten Dienst. Bei PUT / GET liegt immer ein Client / Server - Verhältnis vor.

Kriterium	Leistungseckdaten	Zusätzlicher Hinweis
Übertragbare Datenmengen	Zwischen 1 Byte und 64 kByte.	Auch dies ist abhängig vom gewählten Dienst, in der vorliegenden Applikation ist das gesamte Spektrum ausnutzbar.
Mögliche Dienste	• Blockorientiertes Senden / Empfangen • Ungeblocktes Senden / Empfangen • Read Write Dienst	Das blockorientierte Senden/Empfangen wird in den folgenden Kapiteln noch einzeln behandelt.

3.3 Leistungseckdaten der Gesamtapplikation

Tabelle 3-5

Kriterium	Leistungseckdaten		
Programmgröße	S7-300	54692 Byte	Arbeitsspeicher (bei 200 Bytes Nutzdaten)
	S7-400	2140 / 4984 Byte	Arbeitsspeicher Code / Daten (bei 200 Bytes Nutzdaten)
Maximale Zykluszeit	S7-300	24 ms	Durchschnittlich etwa 5ms
	S7-400	4 ms	Durchschnittlich etwa 1ms

Kriterium	Leistungseckdaten	Zusätzlicher Hinweis
max. Datenübertragungsrate (PB)	12 Mbit/s	Eingestellt sind 1,5 Mbit/s
max. Datenübertragungsrate (IE)	100 Mbit/s Vollduplex	Eingestellt sind 100 Mbit/s Vollduplex
max. Telegrammlänge	Ca. 800 Bytes	Die Größe ist von der Gateway-CPU begrenzt. Eine CPU 315-2DP kann eine max. Datenbausteingröße von 16kB verwalten. Da der ein Ringpuffer 10 Telegramme beinhalten kann, kann ein Telegramm max. 16 / 10 kB groß sein. Davon muss noch die Headerinformation abgezogen werden. Bei einer S7-400 entspricht die max. Telegrammlänge etwa 6,2 kByte Eingestellt sind 200 Bytes Nutzdaten.
max. Anzahl Verbindungen	Gateway: 1 Verbindung pro zu routender Station	Die Verbindungen müssen aufsteigend und lückenlos vergeben sein.
	Quell-/Zielstation: 1 Verbindung zum Gateway	
Telegrammlaufzeit bei gegebener Hardware (bidirektional)	Min: 180 msec Avg: 240 msec Max: 420 msec	Diese Angaben beziehen sich auf 100 gemessene Werte bei bidirektionalem Datenverkehr
Telegrammlaufzeit bei gegebener Hardware (unidirektional: IE → PB)	Min: 110 msec Avg: 180 msec Max: 250 msec	Diese Angaben beziehen sich auf 100 gemessene Werte bei unidirektionalem Datenverkehr

Kriterium	Leistungseckdaten	
Telegrammlaufzeit bei gegebener Hardware (unidirektional: PB → IE)	Min: 160 msec Avg: 220 msec Max: 290 msec	Diese Angaben beziehen auf 100 gemessene Werte bei unidirektionalem Datenverkehr
Anzahl an unterstützten Stationen in der vorliegenden Applikation	2 bis 8 **Hinweis:** Durch Anpassung kann die Anzahl der unterstützten Stationen erhöht werden.	Beispiel: • 4 Stationen im Subnetz A (IE) • 4 Stationen im Subnetz B (PB) In der Applikation genutzt: • 1 Station im Subnetz A (IE) • 1 Station im Subnetz B (PB)
Datenrichtung	• unidirektional • bidirektional	Dies ist einstellbar

Teil A2 : Funktionsmechanismen

Ziel Teil A2

Der Teil A2 dieses Dokuments soll dem Leser

- alle vorkommenden Funktionselemente verständlich machen
- die Komponenten aufzeigen, die einfach in eigene Anwendungen integrierbar sind.
- Entsprechendes Hintergrundwissen vermitteln

Inhalt Teil A2

4 Funktionsmechanismen

Einleitung

Hier finden Sie Informationen

- zur Lösungsstruktur der Gesamt-Applikation und der Einzelelemente
- zum verwendeten Bus-/Protokoll
- zu den verwendeten Funktionselementen und ihrer Arbeitsweise
- zur allgemeinen Vorgehensweise bei Kommunikationsaufgaben mit BSEND/BRCV

Weitere Details entnehmen Sie dem Teil C.

Was können Sie damit anfangen?

Grundsätzlich ist die vorliegende Applikation sofort einsatzfähig. Mit der Installationsanleitung können Sie die Applikation in Betrieb nehmen, ohne dieses Kapitel durchgearbeitet zu haben. Wollen Sie jedoch z.B. bestimmte Module der Applikation für Ihre Anforderungen variieren, benötigen Sie tiefergehende Informationen.

Inhalt Kapitel Funktionsmechanismen

Nachfolgend finden Sie zur Orientierung den Aufbau des Kapitels 4.

Tabelle 4-1

Kapitel	Kapitelüberschrift	Hier erfahren Sie...
4.1	Datenflussmodell in dieser HW-Konstellation	... welchen Weg die Daten innerhalb der Applikation nehmen.
4.2	Strukturelle Komponenten dieser Kommunikationsaufgabe	... mehr über den strukturierten Aufbau des Applikationsfalls
4.3	Physikalisches Bus-Medium	... Detailinformationen über die möglichen Busmedien
4.4	Protokollspezifische Mechanismen des S7-Protokolls	... tiefere Einblicke in die Arbeitsweise und Methoden des S7-Protokolls
4.5	Applikative Struktur in diesem Beispiel	... einen Überblick der verwendeten Module
4.6	Funktionselemente in der Applikationssoftware	... Informationen über die Funktionsweise der einzelnen Funktionsbausteine (Teilfunktionen).

4.1 Datenflussmodell in dieser HW-Konstellation

Darstellung des Datenflussmodells

Im folgenden Schema ist das Datenflussmodell der Applikation aus SIMATIC-Komponentensicht dargestellt.

Um die Darstellung möglichst übersichtlich zu halten, werden die Anwenderprogramme nur für einen unidirektionalen Datenfluss aufgeführt.

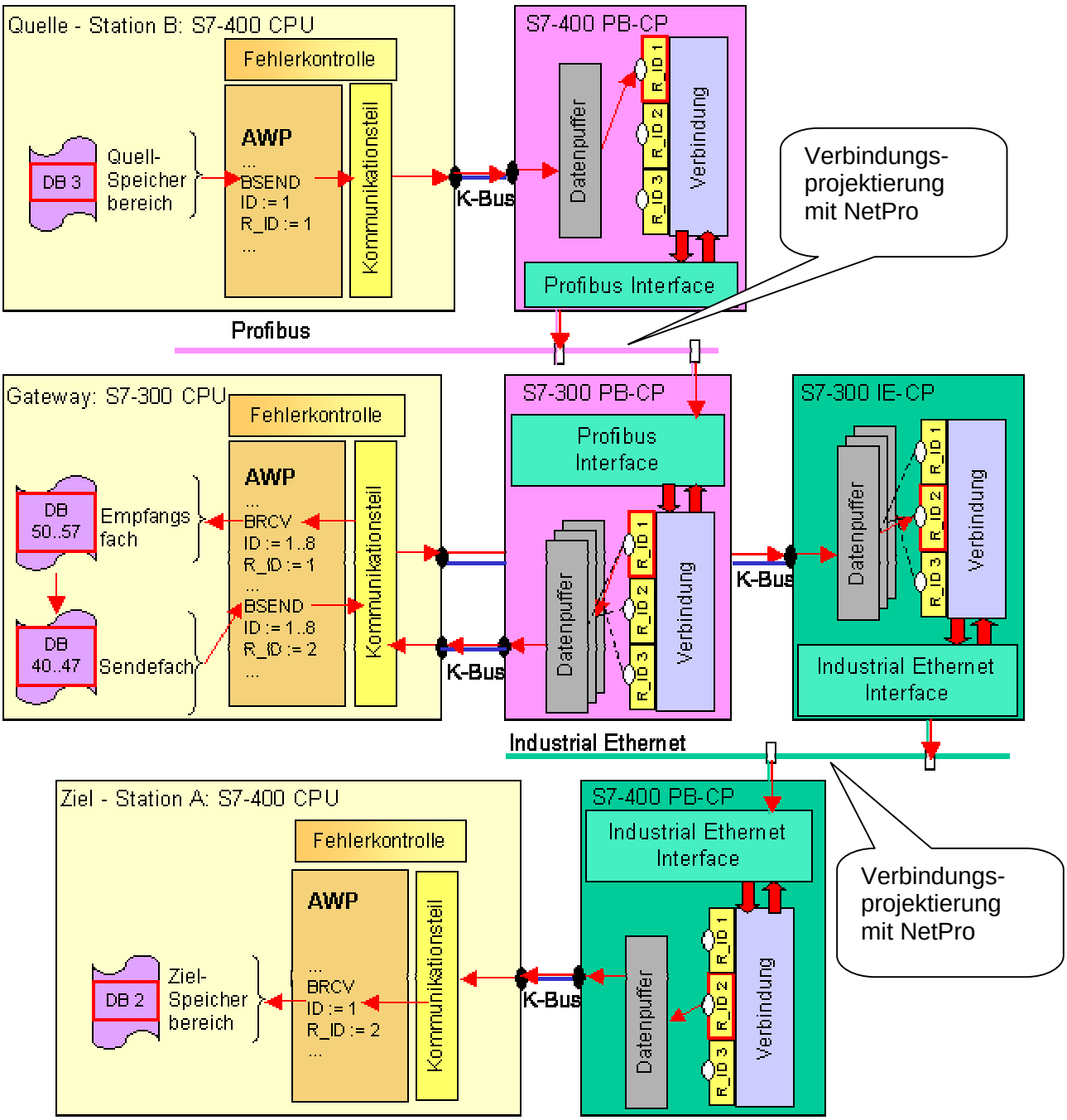


Bild 4-1 Unidirektionales Datenflussdiagramm ohne Quittung

Beschreibung des Datenflussmodells

Tabelle 4-2

Pos.	Beschreibung
1	Der Anwenderschnittstelle (BSEND) im Anwenderprogramm wird der zu sendende Datenbereich per S7-Any-Pointer übergeben. Mit einer steigenden Flanke werden die Rohdaten über den K-Bus an den CP geschickt. Dieser sendet die Daten gemäß Verbindungsprojektierung an den CP des Gateways.
2	Nachdem der BRCV-Baustein im Gateway den erfolgreichen Empfang der Daten signalisiert hat, werden die Daten im Anwenderprogramm der Gateway-CPU analysiert, um das Ziel der Daten zu bestimmen. Je nach Ziel werden die Daten nun in den jeweiligen Ringpuffer und somit in das jeweilige Sendefach geschrieben und der dazugehörige BSEND mit einer steigenden Flanke aufgerufen. Nun werden die Daten an den jeweiligen CP weitergeleitet. Dieser verpackt die Daten in ein S7-Protokoll Rahmenpaket und sendet es gemäß Verbindungsprojektierung an den CP der Zielstation.
3	Die Zielstation signalisiert über das NDR-Bit des BRCV-Bausteins den Empfang der Daten.
4	**Folgendes ist im Datenflussdiagramm nicht dargestellt:** Die Gateway-CPU generiert nach dem Senden der Daten an die Zielstation ein Quittungstelegramm und sendet es an die Quellstation. Dazu wird zunächst der Kopf des Sendefachs untersucht, um die ID der Quellstation zu extrahieren. Nun wird das Quittungstelegramm generiert und in den zur Datenquelle zugehörigen Ringpuffer eingetragen. Die Quittung wird sodann mit dem zur Quellstation zugeordneten BSEND-Baustein versendet, wodurch die Rohdaten an den anderen CP weitergeleitet werden. Dieser verpackt die Daten in ein S7-Protokoll Rahmenpaket und sendet es gemäß Verbindungsprojektierung an den CP der Quellstation. Damit wird eine Ebene7+ -Quittung realisiert.
5	Die Quellstation signalisiert mit dem Empfang des Quittungstelegramms den Abschluss des Sendevorgangs.

4.2 Strukturelle Komponenten einer Kommunikationsaufgabe

Einleitung

Im einfachsten Sinn lautet eine Kommunikationsaufgabe: „Transportiere meine Daten von Station A nach Station B“. Rund um diesen banalen Satz sind für den Praktiker aber eine Reihe von Entscheidungen zu fällen und Strukturen zu realisieren.

Wir zeigen Ihnen in den folgenden Abschnitten einen allgemeingültigen Ansatz, der je nach Granularität der Aufgabenstellung, Wahl des Kommunikationsweges-/netzes, Trägerprotokoll, Anwendermodulen, etc. mehr oder weniger ausgearbeitet werden muss.

Darstellung der Struktur-Komponenten

Die folgende Grafik stellt die einzelnen Schichten und Module der Kommunikationsbeziehung zwischen den drei Stationen dar, wobei hier der besseren Übersicht wegen nur die für die unidirektionale Datenübertragung von Station B zu Station A relevanten Module berücksichtigt werden.

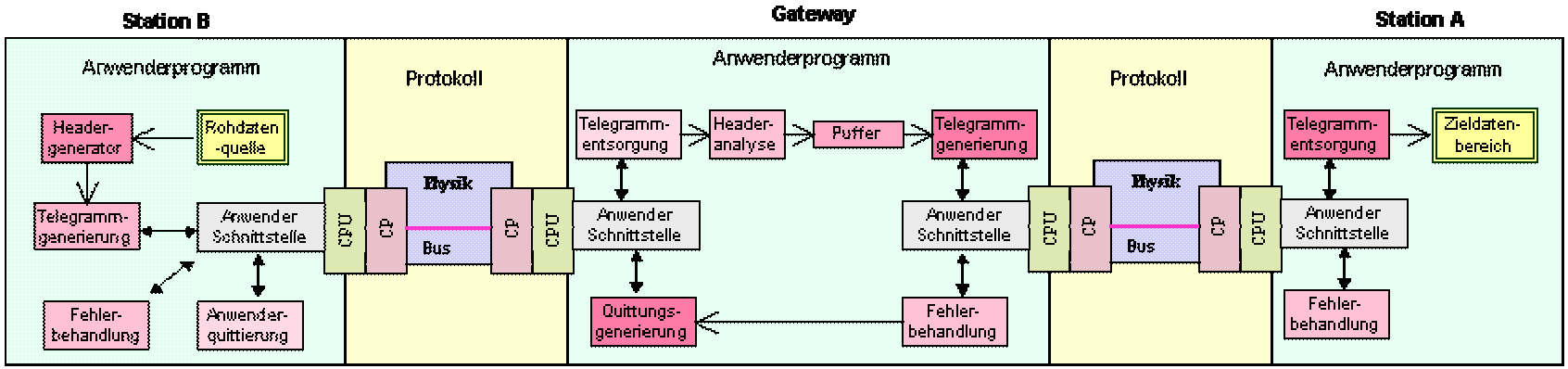


Bild 4-2

Beschreibung

Die unidirektionale Kommunikation besteht aus

- einer physikalischen Kopplung dreier Stationen über verschiedene Busmedien-/Subnetze und entsprechenden Anschaltungen
- der Benutzung eines Übertragungs-Protokolls
- den Anwendermodulen, die die Rohdaten über die Anwenderschnittstelle des Protokolls verpackt, übergibt und analysiert.

Im Folgenden werden die Einzelschichten und die möglichen Implementierungen vorgestellt.

Dabei beschränken wir uns auf die in der Applikation verwendeten Busmedien, Protokolle und Anwenderschnittstellen.

4.3 Physikalisches Bus-Medium

Eine SIMATIC Steuerung unterstützt, auch durch Einsatz entsprechender Kommunikationsprozessoren, eine großes Spektrum von Bussystemen:

z.B.:

- MPI, durch die CPU Schnittstelle bereitgestellt,
- Ind. Ethernet sowie
- PROFIBUS

4.3.1 Profibus Netzwerk

Einleitung

Der PROFIBUS (PROcess Field BUS) wurde als offener, firmenneutraler Feldbusstandard in der DIN 19245 bzw. später in der EN 50170 verabschiedet. Zahlreiche Firmen haben an der Spezifikation und der Umsetzung des Standards mitgearbeitet und bieten im Rahmen ihrer Produktserien ein großes Produktspektrum an PROFIBUS Komponenten und Baugruppen an.

Die Busphysik

Die physikalische Ankopplung des Profibus erfolgt über die RS485 Schnittstelle. Mit Ausnahme des PROFIBUS PA, der wegen seiner Eigensicherheit eine andere physikalische Grundlage hat, nutzt der PROFIBUS den Standard RS 485 zur elektrischen Ankopplung.

Die RS 485 Schnittstelle ist eine potentialfreie erdsymmetrische Schnittstelle für Mehrpunktverbindungen. Das Verfahren der Datenübertragung basiert auf der Differenzspannungsmessung. Hierbei werden die logischen Zustände in unterschiedlichen Spannungsniveaus definiert (log. 0 entspricht 1,5 bis 5 V, log. 1 entspricht –1,5 bis – 5 V DC). Die maximale Spannung auf einem RS 485 Netzwerk ist unterhalb des TTL Spannungspegels, also 5 V DC.

Neben der elektrischen Variante für den PROFIBUS bestehen auch optische Varianten. Diese erlauben es, sowohl Distanzen von bis zu 15 km zwischen zwei Teilnehmern zu überbrücken als auch Potentialdifferenzen vollständig abzulegen.

Token-Prinzip / Token-Passing

Um auf dem PROFIBUS ein gleichzeitiges Senden mehrerer Teilnehmer zu verhindern, wird hier das sogenannte Token-Prinzip bzw. das Token-Passing angewendet. Hierbei wird innerhalb des Netzwerkes eine Sendeberechtigung (Token) an den nächsten „**aktiven**“ Teilnehmer weitergeleitet. Dieser Token stellt die Erlaubnis dar, mit den anderen Teilnehmern am PROFIBUS zu kommunizieren. Dies betrifft allerdings nur den aktiven Sendebetrieb. Angeforderte Daten bzw. Quittungen werden auch während die-

ser Zeit von den Zielstationen beantwortet. Passive Stationen, auch Slaves genannt, erhalten keinen Token, da sie einem Master zugeordnet sind und von diesem in einem regelmäßigen Zeitraster gepollt werden.

Durch die dem PROFIBUS zugrunde liegenden Busparameter wird weiterhin festgelegt, wann der Token spätestens wieder den ersten Teilnehmer am PROFIBUS erreichen muss. Deshalb ist Deterministik gewährleistet.

Weitere Informationen können Sie auch der Profibus Spezifikation entnehmen.

Profibus Spezifikation

Der Profibus ist gemäß der Spezifikation in 2 Protokollebenen unterteilt worden.

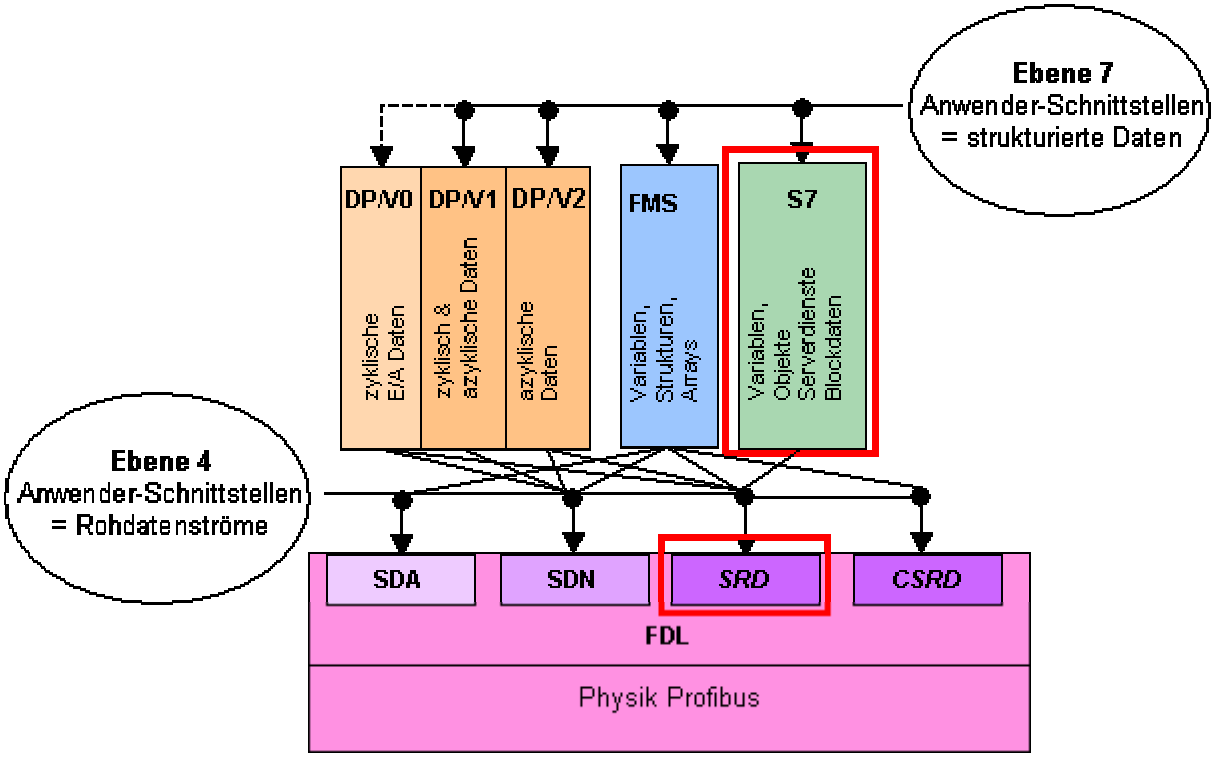


Bild 4-3

Auf Ebene 7 (Applikationsebene) befinden sich gemäß der Spezifikation die folgenden 2 Protokolle:

- FMS (Fieldbus Message Specification) zur Übertragung von Daten auf Zellenebene
- DP (Distributed Peripheral) zur Steuerung bzw. Auswertung von dezentral positionierten Aktoren und Sensoren (PROFIBUS PA nutzt die identischen Mechanismen wie DP, basiert aber als eigensicherer Bus physikalisch auf der IEC 1158-2 Spezifikation).

Auf der unteren Ebene (Sicherungsschichtebene) befindet sich das Basisprotokoll FDL. Dies ist Basis für jedes implementierte Protokoll.

Das FDL-Protokoll bietet 4 Dienste zur Datenübertragung an:

Tabelle 4-3: FDL Dienste

Dienst	Beschreibung
SDN	Send Data with No Acknowledge (Daten-Sendung ohne Quittung)
SDA	Send Data with Acknowledge (Daten-Sendung mit Quittung)
SRD	Send and Request Data (Sendung mit Anforderung und Rückantwort)
CSRD	Cyclic Send and Request Data (Zyklische Sendung mit Anforderung und Rückantwort)

Weitere Protokolle

Alle auf Profibus basierenden Protokolle nutzen diese Dienste, auch in unterschiedlicher Kombination.

Von der Normung unabhängig haben sich auch andere Protokolle entwickelt, die nicht in einer Norm offengelegt sind. Beispiele hierfür sind:

- das S7-Protokoll bzw.
- das TF-Protokoll (Nur bei SIMATIC S5 durch den CP 5430 verfügbar)

Das S7-Protokoll wird, da es in der vorliegenden Applikation verwendet wird, weitergehend betrachtet.

4.3.2 Industrial Ethernet Netzwerk

Einleitung

Am 22 Mai 1973 schrieb Dr. Bob Metcalfe eine kurze Notiz mit dem Titel „Alto Ethernet“, die den Beginn der Entwicklung des heutigen Netzstandards Ethernet darstellt.
Seit dem wurde von großen Firmen wie 3COM, IBM, Cisco, DEC oder XEROX die Entwicklung des Hardwarestandards Ethernet vorangetrieben.

Hardware Basis

Ethernet ist in der Basisversion als Triaxial- bzw. Koaxialnetzwerk konzipiert worden. Dieses Hardwarekonzept erlaubte Linienstrukturen, die Distanzen von bis zu 2500m zu überbrücken vermochten. Dabei wurden Datenübertragungsraten von 10 MBit/s genutzt.

Im Laufe der Jahre wurden durch Neuentwicklungen, wie zum Beispiel der Glasfaser Technologie, diese Grenzen überschritten. Durch steigende Ansprüche an Datendurchsatz und Verfügbarkeit wurde das Konzept von Triaxialnetzen nicht weiter verfolgt. Heutige Netze werden in sternförmiger Twisted Pair Technik mit RJ45 Anschlüssen ausgelegt. Heutige Netzwerke arbeiten in der Regel mit Baudraten von 100 MBit/s.

Aktuelle Entwicklungen zielen in Richtung Wireless Ethernet. Dieses wird in absehbarer Zeit die Performance des heutigen verkabelten Ethernets erreichen. Das kabelgebundene Ethernet wird sich hingegen in den Gigabit Bereich begeben.

Das Zugriffsverfahren CSMA

Im Gegensatz zum Profibus nutzt das Ethernet keinen Token. Es sind alle Teilnehmer am Bus gleichberechtigt, Telegramme zu versenden. Um daraus entstehende Probleme bei der Datenkommunikation zu regeln, wird bei Ethernet ein Zugriffsverfahren namens CSMA (Carrier Sens Multiple Access) verwendet. Dies ist in verschiedenen Versionen gängig. Für kabelgebundenes Ethernet wird in der Regel CSMA/CD (with Collision Detection) verwendet, wogegen bei Wireless Ethernet mit CSMA/CA (with Collision Avoidance) gearbeitet wird.

Im Rahmen dieses Dokumentes wird nicht näher auf das Verfahren, das hinter diesem Prinzip steckt, eingegangen.

Aufgrund des Zugriffsverfahrens ist keine direkte Deterministik möglich. Dies bedeutet, dass Ethernet nicht erlaubt auszusagen, ob ein Telegramm zu einem bestimmten Zeitpunkt übertragen ist.

Protokolle

Im Laufe der Jahre haben sich im Rahmen der Ethernet Entwicklung eine Vielzahl von Protokollen entwickelt. Diese sind teilweise auch von anderen Netzen auf Ethernet portiert worden. Das berühmteste Beispiel hierfür stellt wohl das Protokoll TCP/IP dar.

In der Automatisierungswelt hat sich daneben noch das auf dem ISO/OSI – Referenzmodell aufbauende ISO – Transport Protokoll sowie das TF/MAP Protokoll entwickelt.
Jedoch haben sich diese Protokolle im Rahmen der globalen Zusammenarbeit nicht durchsetzen können, da sie sehr spezifisch und teilweise auch auf Siemens Produkte ausgelegt sind.

Viele Anwendungen werden heute über das TCP, UDP oder dem vom ISO-Transport abgeleiteten RFC 1006 Protokoll (auf TCP Basis) ausgeführt.

Innerhalb der S7-Welt hat sich ebenso das S7-Protokoll auf Basis des ISO-Transport oder des TCP Stacks etabliert.

4.4 Protokollspezifische Mechanismen des S7-Protokolls

Einleitung

Das hier vorgestellte S7-Protokoll ist innerhalb des Spektrums der von SIMATIC unterstützten Protokolle etwas herausragend, da es als einziges Protokoll physikunabhängig arbeitet. Es kann sowohl auf MPI als auch auf Profibus oder Ethernet Basis verwendet werden. Innerhalb der Steuerun-

gen dient es ebenso zur Übertragung von Daten an Komponenten wie auch an FM's oder serielle CP's.

Aufgabe eines Protokolls

Ein Protokoll ist eine Sammlung von Standards, die beschreiben, wie Daten übertragen werden.

In der Regel werden aus der Gesamtheit der Daten einzelne Pakete mit festgelegten Strukturen geschnürt und diese werden in einem bestimmten Format bzw. einer bestimmten Codierung übertragen.

Ein sinnvoller Vergleich im alltäglichen Bereich ist eine Sprache. Sie enthält Regeln und Mittel, um eine Information zu übertragen. Hält sich eine Seite nicht an diese Regeln oder verwendet andere Mittel, so versteht die andere Seite die Information nicht.

Übersicht über die verfügbaren Protokolle

Zur Übersicht und besseren Einordnung des S7-Protokolls folgt eine kurze Übersicht der im Moment im SIMATIC Umfeld üblichen Protokolle auf Profibus bzw. Ethernet im Hinblick auf das ISO-OSI Referenzmodell.

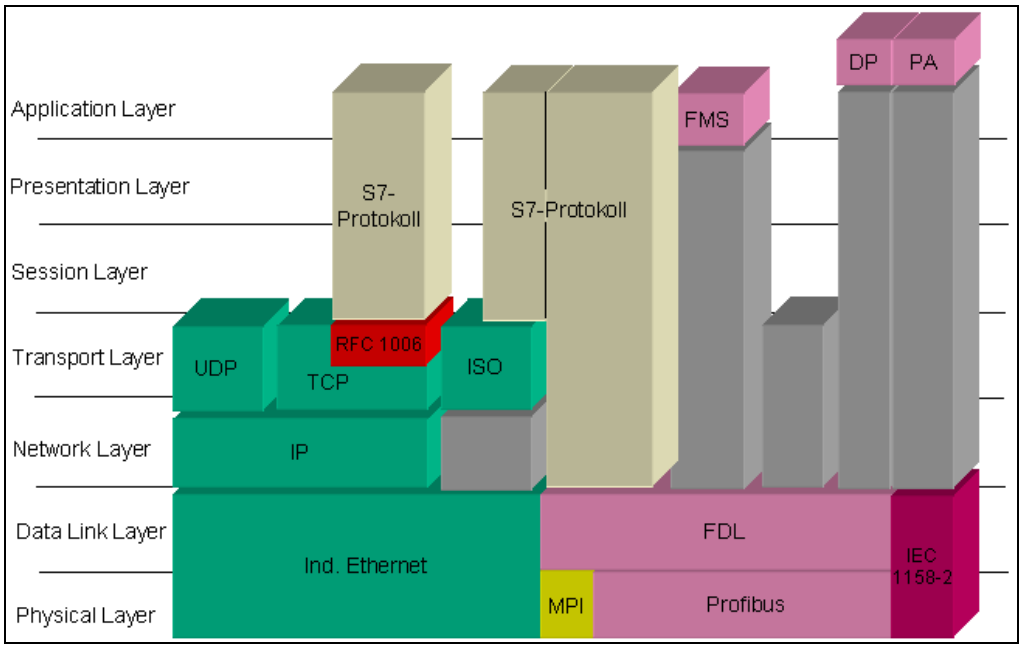


Bild 4-4

4.4.1 Kerneigenschaften des S7-Protokolls

Einleitung

In den folgenden Abschnitten werden die spezifischen Eigenschaften des S7 Protokolls beschrieben. Detaillierte Beschreibungen der Aufrufschnittstellen folgen in einem der nächsten Kapitel.

Dienste des S7 Protokolls

Das S7 Protokoll arbeitet mit dem Kommunikationsdienst S7-Kommunikation. Dieser hat 3 Dienstklassen.

Tabelle 4-4

Dienst	Beschreibung
PUT / GET	Ein unidirektionaler Schreib- / Lese-Dienst zur Übertragung kleiner Datenmengen an bzw. von einer Station.
USEND / URECEIVE	Ein bidirektionaler unkoordinierter Dienst zur Übertragung mittlerer Datenmengen zwischen zwei Stationen.
BSEND / BRECEIVE	Ein bidirektionaler blockorientierter Dienst zur Übertragung großer Datenmengen zwischen zwei Stationen

Datenmengengerüst

Das S7-Protokoll erlaubt es, Daten von 1 Byte bis zu 64 KByte zu übertragen. Das Mengenspektrum ist abhängig von dem genutzten Dienst und der verwendeten Hardware (siehe auch Tabelle am Ende dieses Kapitels).

Verwendbare Bussysteme

Das S7-Protokoll nutzt außerhalb einer S7-Steuerung folgende Busmedien:

- MPI
- PROFIBUS
- Ethernet (sowohl ISO als auch TCP Stack)

Systeme die das S7-Protokoll unterstützen

Folgende Systeme unterstützen das S7-Protokoll:

- S7-Steuerungen inklusive der verfügbaren Kommunikationsprozessoren
- PC-Systeme mit entsprechenden Anschaltungen und Treiberausrüstung

Steuerungssysteme wie S5, TI, GE oder AB lassen sich nicht über das S7-Protokoll an die S7-Steuerung koppeln.

Übersicht der Dienste und Eigenschaften

Tabelle 4-5

Dienste / Eigenschaften	BSEND / BRCV(1	USEND / URCV	PUT / GET
Max. Datenlänge S7-300 / S7-400	32 KB / 64 KB (2	440 Byte / 440 Byte(3	164 Byte / 400 Byte(3
Mögliche Adressbereiche S7-300 / S7-400	M, D / M, T, Z, E, A, D	M, D / M, T, Z, E, A, D	M, D / M, T, Z, E, A, D
Datenkonsistenz S7-300 / S7-400	Die gesamte Länge pro Auftrag	Die gesamte Länge pro Auftrag	8 – 32 Byte / 32 Byte bis gesamte Länge (4
Kommunikationsprinzip	Client / Client	Client / Client	Client / Server

Dienste / Eigenschaften	BSEND / BRCV(1	USEND / URCV	PUT / GET
Max. Anzahl der Verbindungen	Siehe CPU Spezifikation.	Siehe CPU Spezifikation.	Siehe CPU Spezifikation.
Bausteintypen	SFB / FB 12 „BSEND“ SFB / FB 13 „BRCV“	SFB / FB 8 „USEND“ SFB / FB 9 „URCV“	SFB / FB 15 „PUT“ SFB / FB 14 „GET“

(1 In dieser Applikation genutzt
(2 Entspricht der Gesamtgröße der Nutzdaten für den SFB / FB im Fall von Industrial Ethernet.
(3 Entspricht der maximalen Länge eines Datenbausteines des jeweiligen Systems.
(4 Abhängig von der benutzten CPU.

4.4.2 Projektierungsmodell des S7-Protokolls

Die S7-Verbindung

Unter einer S7-Verbindung ist eine kommunikative Verknüpfung zwischen zwei Endpunkten zu verstehen, die **beide** das S7-Protokoll unterstützen. Diese Endpunkte können sowohl 2 CPU Baugruppen als auch andere Baugruppen, wie z.B. ein PC, ein CP, eine FM Baugruppe oder ähnliches sein.

Ferner unterscheidet man zwischen ein- und zweiseitigen Verbindungen:

- Einseitige Verbindung:
  Bei einer einseitigen Verbindung liegt eine Client-Server-Beziehung vor. Nur dem Client ist der Server bekannt, während der Server keinerlei Informationen über den Client hat.
  Bei einer Kommunikation über den Kommunikationsdienst S7-Kommunikation mit PUT/GET liegt eine einseitige Verbindung vor.
- Zweiseitige Verbindung:
  Bei einer zweiseitigen Verbindung sind in beiden Endpunkten die Informationen über den Partner bekannt.
  Bei einer Kommunikation über den Kommunikationsdienst S7-Kommunikation mit BSEND/BRCV bzw. USEND/URCV liegt eine zweiseitige Verbindung vor. Man spricht in diesem Zusammenhang auch von Client-Client-Verbindung.

Das Projektierungstool Netpro

Die Projektierung jedes Verbindungstyps erfolgt im Projektierungstool NetPro. Es unterstützt den Anwender bei der Verbindung zweier Stationen innerhalb eines STEP-7 Projekts, indem in diesen Fällen die notwendigen Detaileinstellungen wie TSAP’s bzw. Ressourcenverteilung durch die Software vorgenommen werden. Eine Beschreibung der Projektierung einer S7-Verbindung finden Sie im Kapitel 5.3.1.

NetPro unterstützt auch eine Projektierung von Verbindungen zwischen Kommunikationspartnern, die sich in zwei unterschiedlichen Projekten befinden.

S7-Verbindungsressourcen

! **Wichtig**

Die folgenden Erläuterungen beziehen sich auf die in der Applikation verwendeten Hardware-Komponenten.

Insbesondere ist das Verhalten bei der S7-300 nur bei CPs ab 6GK7342-5DA02-xxxx und 6GK7343-1EX11-xxxx gegeben.
Ältere CPs können ein anderweitiges Verhalten zeigen!

Die Anzahl der S7 Verbindungsressourcen beschreibt die Fähigkeit einer Baugruppe, wie viele Kommunikationsverbindungen mit einem anderen Partner aufgebaut werden können. Die Anzahl dieser Verbindungsressourcen ist stark vom S7-System bzw. der spezifischen Baugruppe abhängig.

Die beschränkte Anzahl an S7 Verbindungsressourcen lässt sich bei der S7-300 durch Verwendung von CPs beeinflussen. Die CPU selbst ist nicht in der Lage eine aktive, statische S7-Kommunikation zu führen. Für passive, statische S7-Kommunikation hält die CPU auch nur eine sehr geringe Anzahl von Verbindungsressourcen vor. Um hier eine Möglichkeit zu schaffen, sowohl eine größere Zahl passiver als auch eine größere Zahl aktiver Verbindungen betreiben zu können, wurde die S7-Kommunikationsfunktionalität mit in die Kommunikationsprozessoren der S7-300 implementiert.

Die Einschränkung der Anzahl der S7 Verbindungsressourcen lässt sich bei der S7-400 grundsätzlich **nicht** beeinflussen. Die S7-400 trägt eine definierte Anzahl von Verbindungsressourcen im Betriebssystem.

Das folgende Bild veranschaulicht das Ressourcenhandling:

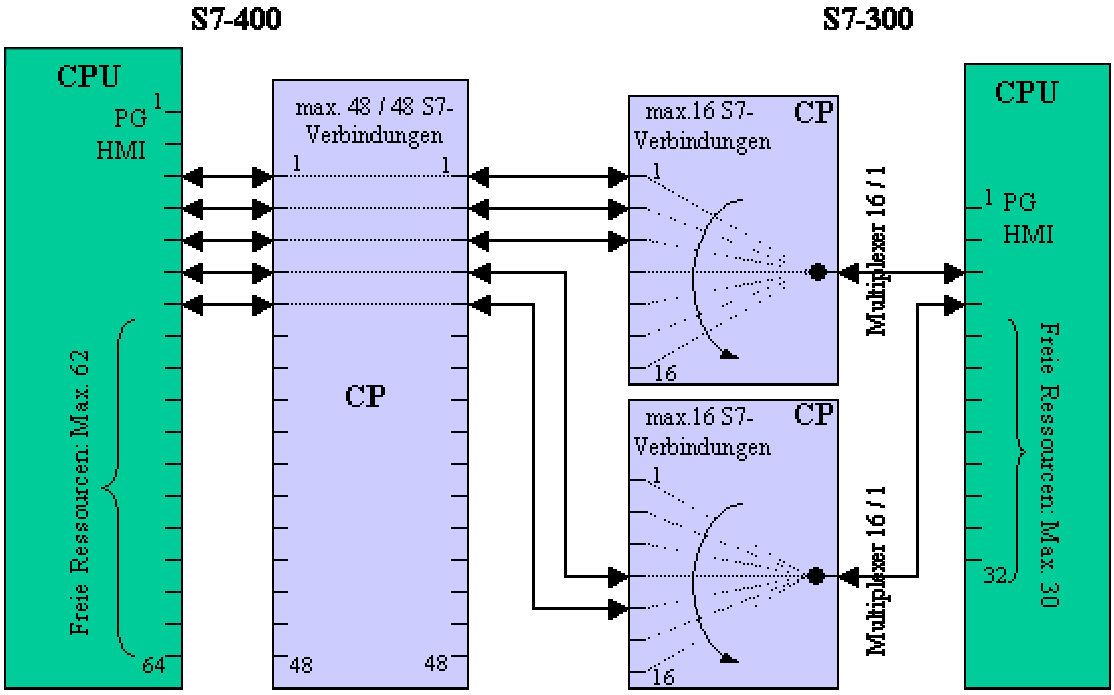


Bild 4-5 Ressourcenhandling der S7-400 und S7-300

Der Kommunikationsprozessor der S7-300 arbeitet somit als Multiplexer für aktive und passive S7-Kommunikationsverbindungen. Es können auf diese Weise bis zu 16 S7- sowie 16 TD/OP-Verbindungen über den CP gemultiplext werden. Der CP an sich verwendet zur Kopplung mit der CPU nur eine Kommunikationsressource der CPU, um deren Belastung so gering wie möglich zu halten.

Eine besondere Projektierung einer Verbindung, die diesen Mechanismus nutzt, ist innerhalb eines Projektes oder innerhalb von 2 STEP-7-Projekten auf einem PC nicht erforderlich. Diese Aufgabe wird vollständig von STEP-7 übernommen.

Die Anzahl steckbarer CPs ist bei der S7-400 von der CPU limitiert, liegt jedoch in der Regel bei aktuellen Baugruppen bei etwa 10 CPs.

Bei der S7-300 ist hier noch der Faktor der freien S7 Kommunikationsressource wichtig; denn sind alle Kommunikationsressourcen belegt, so kann **kein** weiterer CP gesteckt werden.
Die Anzahl nutzbarer Kommunikationsressourcen für CPU Baugruppen der S7-300 beschränkt sich in der Regel auf die freien Ressourcen, deren Anzahl - abhängig von CPU-Typ, CP und Projektierung - zwischen 0 und 30 variieren kann. Um das Multiplexen von S7-Verbindungen auch außerhalb eines STEP-7-Projektes nutzen zu können, ist eine spezielle Projektierung des Endpunktes notwendig. Hier spielen die S7-TSAPs eine Rolle, die sonst durch STEP-7 bzw. NetPro direkt vergeben werden.

S7 – TSAP

Die Adressen einer Kommunikationsverbindung, egal ob es sich um eine S7-Verbindung oder eine andere Verbindung handelt, sind durch zwei Parameter festgelegt:

- Der erste Parameter ist die Netzwerkadresse des lokalen und des remoten Partners.
- Der zweite Parameter ist eine Detailadresse innerhalb der jeweiligen Station.

Bei TCP wird diese Detailadresse durch den Port, bei ISO Transport bzw. Profibus Verbindungen durch den TSAP realisiert. Der TSAP (**T**ransport **S**ervice **A**ccess **P**oint) stellt die Adresse der Ressource innerhalb des Kommunikationsprozessors dar.

Ein TSAP wird auch bei S7-Verbindungen verwendet, er hat hier aber eine andere Funktion, denn er beschreibt den **Endpunkt der Kommunikationsverbindung**.

Der TSAP des S7-Protokolls besteht aus 2 Teilen:

- der Rack/Steckplatz Adresse sowie
- der Verbindungsressource.

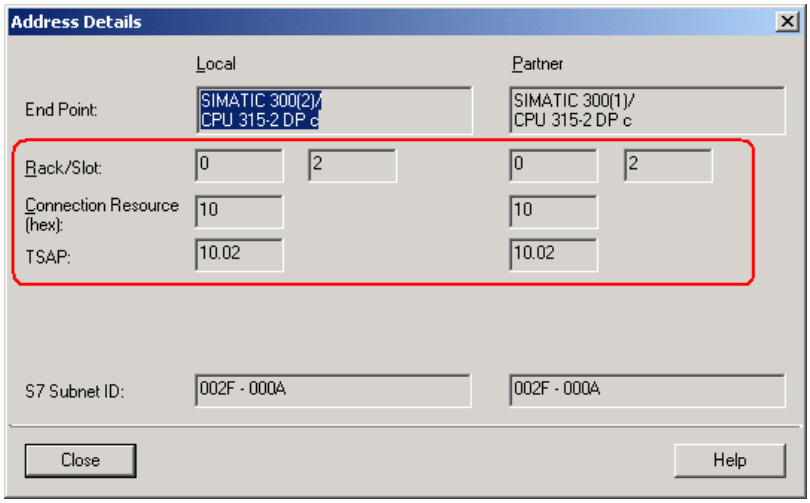


Bild 4-6

Rack / Steckplatz Adresse

Mit der Rack / Steckplatz Adresse wird der Steckplatz des Kommunikationszielpunktes, in der Regel der CPU, adressiert, da sie normalerweise der Endpunkt der Verbindung ist.

Bei der S7-300 ist die CPU immer im Steckplatz 2, deshalb ist hier die Adresse immer XX.02. Eine Ausnahme von dieser Regel stellt die Projektierung einer Verbindung über einen multiplexenden S7-CP der S7-300 dar.

Hinweis: Verbindungsprojektierung „außerhalb“ eines STEP-7-Projektes

Falls Sie eine Kommunikationsverbindung über einen CP der S7-300 mit der S7-Multiplexing-Funktion verwenden möchten und die Projektierung von „außerhalb“ eines STEP-7-Projektes vorgenommen wird, z.B. von COML S7 aus, so ist als Endpunkt der Verbindung nicht die CPU, sondern der Kommunikationsprozessor vorzusehen. Siehe hierzu auch den FAQ 17628518

Verbindungsressource

Folgende Verbindungsressourcen sind in der SIMATIC S7 vorgesehen:

Tabelle 4-6

Verbindungs-ressource	Bedeutung	Typ	Für S7-Verbindungen nutzbar	Bedeutung
0x01 („PG“)	PG-Verbindung	Freie Verbindung	Nein	Eine für PG Funktionen fest in jeder CPU reservierte Verbindung.
0x02 („OP“)	OP-Verbindung	Freie Verbindung	Nein	Eine für HMI-Verbindungen fest in jeder CPU reservierte Verbindung
0x03	Sonstige	Freie Verbindung	Ja	Diese Verbindungsressource ist zur freien Verfügung und kann für verschiedene Anwendungen genutzt werden, z.B. bei der Kopplung einer einseitig projektierten Verbindung auf passiver Seite oder dem Einsatz eines CP’s.
0x10 ... 0xDF	Verbindungen mit statischem oder dynamischem Verbindungsaufbau	Projektierte Verbindungen	Ja	Eine dieser Verbindungsressourcen kann genau eine Verbindung bedienen. Verwendung: zweiseitige Verbindungen

In der vorliegenden Applikation werden „projektierte Verbindungen“ genutzt.

4.4.3 Anwenderschnittstelle zu diesem Protokoll

Die Anwenderschnittstellen des S7 Protokolls lassen sich in 3 Gruppen einteilen:

- Schreib- und Lesedienste (PUT / GET)
- Unkoordinierte Sende- und Empfangsdienste (USEND / URECEIVE)
- Blockorientierte Sende- und Empfangsdienste (BSEND / BRECEIVE)

Im folgenden Kapitel wird der in der vorliegenden Applikation genutzte „blockorientierte Sende- und Empfangsdienst“ betrachtet.

4.4.4 Blockorientierte Sendefunktion „BSEND“ SFB / FB 12

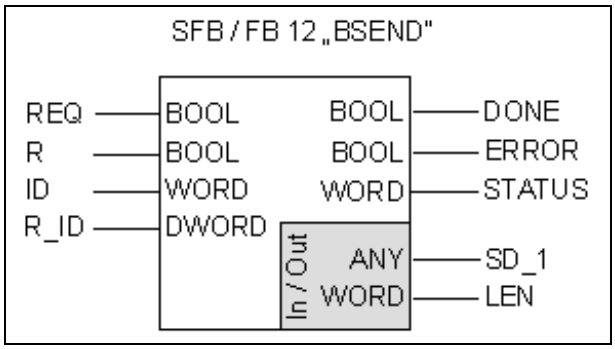


Bild 4-7

Tabelle 4-7: Parameter des SFB / FB 12

Parameter	Beschreibung
REQ	Der Steuerparameter REQ dient zum Aktivieren des Datenaustausches bei positiver Flanke.
ID	Der Adressierungsparameter ID wird aus der Verbindungsprojektierung entnommen.
R	Der Steuerparameter Reset aktiviert den Abbruch eines noch laufenden Datenaustauschs bei positiver Flanke.
R_ID	Der Adressierungsparameter R_ID ist für eine Unteradressierung innerhalb einer S7 Verbindung zuständig. Ein Sende- / Empfangspärchen wird eineindeutig mittels ID und R_ID definiert.
DONE	Der Zustandsparameter DONE informiert mit einer positiven Flanke über den erfolgreichen Abschluss der Funktion.
ERROR	Der Zustandsparameter ERROR zeigt zusammen mit dem Parameter STATUS einen auftretenden Fehler an.
STATUS	Der Zustandsparameter STATUS zeigt die Detailinformation des aktuellen Zustandes des Bausteins an. Ein Wert ungleich 0h oder 19h ist ohne gleichzeitige ERROR Anzeige als Warnung zu interpretieren.
SD_1	Der Zeiger SD deutet auf den Quellbereich in der lokalen CPU.
LEN	Der Ein- / Ausgabeparameter Länge gibt die Länge des zu sendenden Datenblockes vor.

Hinweise

- Der Baustein BSEND der S7 300 ist der SIMATIC-Net-CP Bibliothek innerhalb des S7300 Programmcontainers als FB 12 zu entnehmen.
- Im Gegensatz zu den S7 400 SFBs können die FBs der S7-300 auch dynamisch mit Parametern versorgt werden.
- Der Parameter R_ID darf **innerhalb** einer CPU bei einer gegebenen ID entweder bei einem BRCV oder bei einem BSEND-Baustein verwendet werden. Anderenfalls ist die Eindeutigkeit des ID-R_ID-Paares zwischen den korrespondierenden stationsübergreifenden BSEND/BRCV-Aufrufen nicht mehr gegeben und es wird ein Fehlerstatus ausgegeben.

Beschreibung zu „BSEND“

Der SFB / FB 12 sendet Daten an einen entfernten Partner – SFB / FB vom Typ „BRCV“. Bei diesem Datentransfer kann eine größere Datenmenge zwischen den Kommunikationspartnern transportiert werden als dies mit allen anderen Kommunikations- SFB’s / FB’s für projektierte S7 Verbindungen möglich ist.

Die übertragenen Datenbereiche werden in Segmente aufgeteilt, die den Rahmenbedingungen des genutzten Subnetzes entsprechen. Jedes Segment wird einzeln an den Partner gesendet und von dem dort laufenden Funktionsbaustein SFB / FB „BRCV“ quittiert.

Einschränkend ist in der S7 300, im Fall einer aktiven Ausführung der Funktion, nur ein SD Parameter möglich.

Hinweis

Bitte beachten Sie, dass Sie bei den über ID und R_ID verknüpften Bausteinen die Parameter SD_1 und RD_1 in der

- Länge (LEN) und im
- Datentyp

übereinstimmend definiert haben.

4.4.5 Blockorientierte Empfangsfunktion „BRCV“ SFB / FB 13

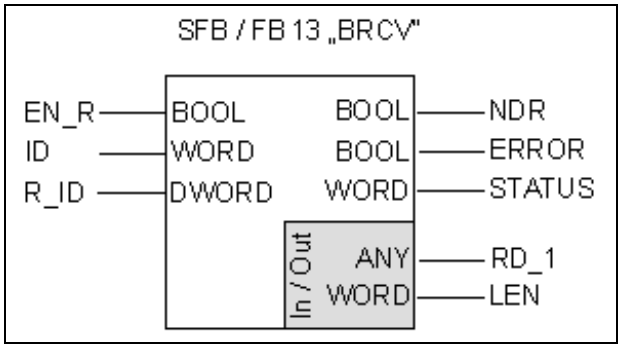


Bild 4-8

Tabelle 3-12: Parameter des SFB / FB 13

Parameter	Beschreibung
EN_R	Steuerparameter EN_R signalisiert Empfangsbereitschaft, wenn der Eingang gesetzt ist.
ID	Adressierungsparameter ID entnommen aus der Verbindungsprojektierung.
R_ID	Adressierungsparameter R_ID ist für eine Unteradressierung innerhalb einer S7 Verbindung zuständig. Ein Sende / Empfangspärchen wird eindeutig mittels ID und R_ID definiert.
NDR	Zustandsparameter NDR informiert mit einer positiven Flanke über den erfolgreichen Abschluss der Funktion.
ERROR	Zustandsparameter ERROR zeigt zusammen mit dem Parameter STATUS einen auftretenden Fehler an.
STATUS	Der Zustandsparameter STATUS zeigt die Detailinformation des aktuellen Zustandes des Bausteines an. Ein Wert ungleich 0h oder 19h ohne gleichzeitige ERROR Anzeige ist als Warnung zu interpretieren.
RD_1	Zeiger auf den Zielbereich in der lokalen CPU.
LEN	Länge der empfangenen Daten.

Hinweise

- Der Baustein BRCV der S7 300 ist der SIMATIC-Net-CP Bibliothek innerhalb des S7300 Programmcontainers als FB 13 zu entnehmen.
- Im Gegensatz zu den S7 400 SFBs können die FBs der S7-300 auch dynamisch mit Parametern versorgt werden.
- Der Parameter R_ID darf **innerhalb** einer CPU bei einer gegebenen ID entweder bei einem BRCV oder bei einem BSEND-Baustein verwendet werden. Anderenfalls ist die Eindeutigkeit des ID-R_ID-Paares zwischen den korrespondierenden stationsübergreifenden BSEND/BRCV-Aufrufen nicht mehr gegeben und es wird ein Fehlerstatus ausgegeben.

Beschreibung des „BRCV“

Der SFB / FB 13 „BRCV“ empfängt Daten von einem entfernten Partner – SFB / FB vom Typ „BSEND“.

Jedes empfangene Datensegment wird mit einer Quittung an den Partner - SFB / FB beantwortet und der Parameter LEN wird aktualisiert.

Der Baustein wird erst nach Aufruf des Steuerparameters EN_R mit einem Wert von 1 (TRUE) empfangsbereit. Ein Setzen des Parameters auf 0 (FALSE) bricht den aktuell laufenden Auftrag ab oder unterbindet einen neuen Sendeanstoß auf Senderseite mit einer entsprechenden Fehlermeldung (Partner SFB / FB befindet sich in falschem Zustand).

Einschränkend ist in der S7 300, im Fall einer aktiven Ausführung der Funktion, nur ein RD Parameter möglich.

Hinweis

Bitte beachten Sie, dass Sie bei den über ID und R_ID verknüpften Bausteinen die Parameter SD_1 und RD_1 in der

- Länge (LEN) und im
- Datentyp

übereinstimmend definiert haben.

Konsistenzbetrachtung bei blockorientierten Sende- / Empfangsfunktionen

Konsistent sind Daten, sobald sie den für einen Datenbestand vorgegebenen Konsistenzbedingungen genügen.

Die Konsistenzbedingung im Fall der Datenübertragung ist **Vollständigkeit**.

Daten, die vollständig übertragen wurden, können als konsistent betrachtet werden.

Da im Fall der blockorientierten Sende- / Empfangsfunktionen auf beiden Seiten Funktionen genutzt werden, ist hier eine getrennte Betrachtung der Sende- und Empfangsseite erforderlich.

Tabelle 4-8

Seite	Konsistenz
Sendeseite	Um die Datenkonsistenz der zu übertragenen Daten zu gewährleisten, darf der Datenbereich, der sich im Sendebereich SD_1 befindet, nicht beschrieben werden, solange der aktuelle Sendevorgang läuft. Der Abschluss des aktuellen Sendevorganges wird durch den Zustandsparameter DONE == TRUE angezeigt.
Empfangsseite	Die übertragenen Daten liegen konsistent im Zielbereich vor, sobald der Zustandsparameter NDR den Wert TRUE hat. Vor dem nächsten Start des Empfangsbausteines ist ein Sichern der Daten bzw. eine Auswertung der empfangenen Daten notwendig. Sobald der Empfangsbaustein wieder mit einem EN_R := TRUE angestoßen wird, kann der Zielbereich, zumindest zum Teil, wieder überschrieben worden sein.

4.4.6 Sequenzbeschreibung des in der Applikation verwendeten Dienstes BSEND / BRECEIVE

Im folgenden Bild sehen Sie anschaulich den sequenziellen Ablauf **eines** BSEND / BRCV Auftrags:

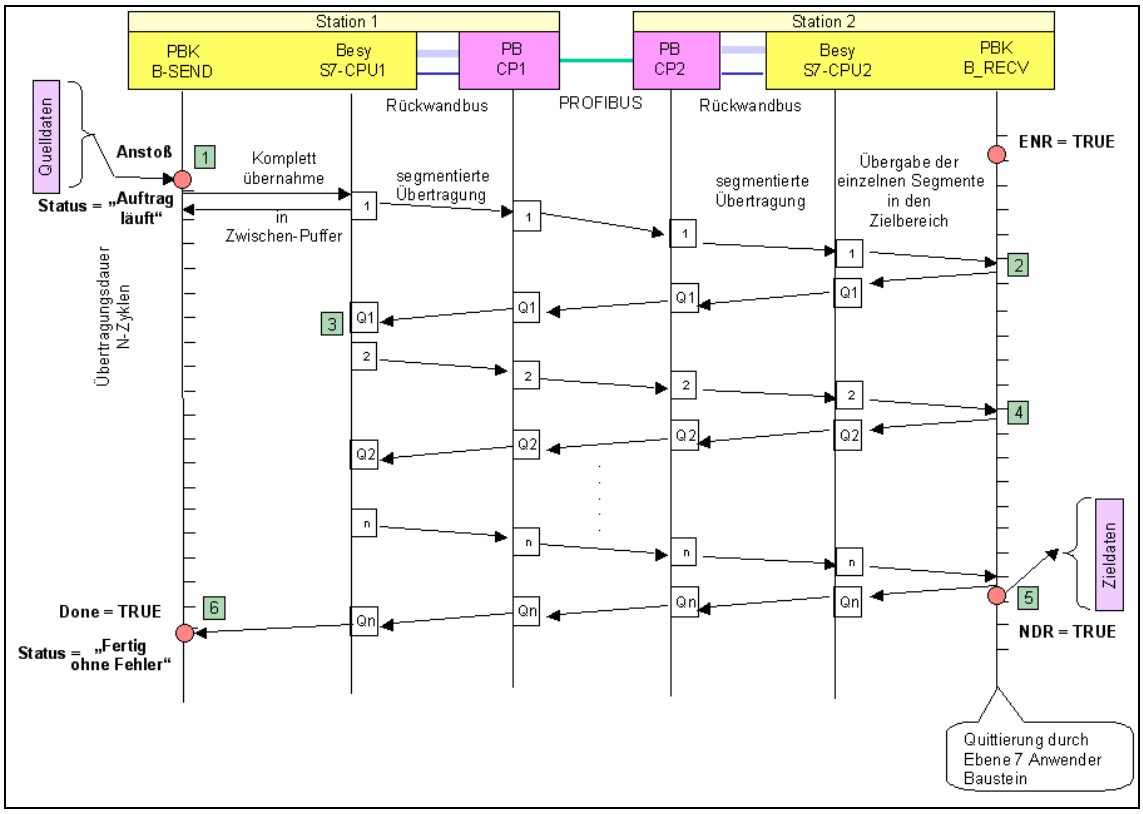


Bild 4-9

Beschreibung des sequenziellen Vorgangs

Folgende Tabelle enthält eine Beschreibung des sequenziellen Ablaufs einer BSEND / BRECEIVE Kopplung.

Tabelle 4-9

Pos.	Beschreibung
1	Durch die positive Flanke am Eingang REQ des „BSEND“ wird die Datenübertragung gestartet. Die Daten, die mit dem Datenpointer SD angezeigt werden, werden in der Länge LEN ausgelesen und dem Betriebssystem der CPU / CP übergeben. Das Betriebssystem (Besy) segmentiert die Daten und schickt das erste Datensegment über den CP, der dieses verpackt und über den angeschlossenen Bus überträgt.
2	Auf der Empfängerseite werden die Daten empfangen. Das Betriebssystem der CPU / CP identifiziert die Daten, worauf die CP die Daten an die CPU weiterreicht. Die CPU übergibt diese an den Funktionsbaustein „BRCV“ der Applikation. Der durch den Aufruf mit EN_R gestartete Funktionsbausteinservice bestätigt das Telegramm mit einer Quittung, die den Rückweg über den CP beginnt.
3	Durch den Empfang der Quittung nimmt das Betriebssystem des Senders das nächste Datensegment und überträgt es an die Zielstation.
4	Die Zielstation erhält das Datensegment und fügt es dem bereits empfangenen Datensegment zu. Die Quittierung des Datensegments wird vorbereitet und übertragen.
5	Nach dem Erhalt des letzten Datensegmentes in der Zielstation quittiert der BRCV Service die Daten mit einem letzten Quittierungstelegramm. Daraufhin wird beim nächsten Aufruf des „BRCV“ Bausteins durch die Ausgabe des Parameters NDR die erfolgreiche Übertragung der Daten angezeigt.
6	Die Sendestation zeigt nach erfolgreichem Empfang der Quittung beim nächsten Aufruf der Sendefunktion „BSEND“ durch den Ausgabeparameter DONE an, dass der Sendeauftrag erfolgreich ausgeführt wurde.

4.5 Struktur der Applikation

Einleitung

Dieses Kapitel beschreibt den strukturellen Aufbau der vorliegenden Applikation.

Dazu wird das Datenflussmodell inkl. Adressiermechanismus zunächst grob und danach verfeinert in der Quell-/Zielstation und im Gateway dargestellt.

4.5.1 Gesamtübersicht

Um den Datenfluss der Gesamtapplikation übersichtlich zu halten, wird nachfolgend der Datenfluss folgender Konstellation gezeigt:

Festlegungen

- In NetPro wurde folgende Verbindungsprojektierung vorgenommen:

Tabelle 4-10

Projektierung in	Projektierung zu	ID
Station A	Gateway	1
Gateway	Station A	1
Gateway	Station B	2
Station B	Gateway	1

- Aufgrund der Gegebenheiten der BSEND/BRCV-Bausteine (s. Kap. 4.4.4) und um die Telegrammtypen zu unterscheiden, wurden die R_IDs folgendermaßen festgelegt:

Tabelle 4-11

Aktion	R_ID beim BSEND- bzw. BRCV-Bausteinaufruf
Quellstation sendet Daten	1
Gateway empfängt Daten	1
Gateway sendet Daten weiter	2
Zielstation empfängt Daten	2
Gateway sendet Quittung	3
Quellstation empfängt Quittung	3

Durch diese Festlegungen lässt sich eine eineindeutige Zuordnung zwischen ID und R_ID realisieren.

Datenflussmodell der Applikation

Im folgenden Datenflussmodell zeigen wir auf einer abstrakten Ebene:

- Adressierschema und Routing-Mechanismus des Gateways
- Unidirektionaler Sendevorgang von Station B zu Station A inkl. Quittung

Dabei gelten folgende Bezeichnungen:

Tabelle 4-12

Abkürzung	Bedeutung
Ex	Empfangseinheit x
Sx	Sendeeinheit x
Rx	Ringpuffer x
ID	Entspricht dem Parameter „ID“ am BSEND/BRCV-Baustein
R_ID	Entspricht dem Parameter „R_ID“, der dem jeweiligen BSEND/BRCV-Baustein zugeordnet ist
x	(für diesen Fall nicht relevant)

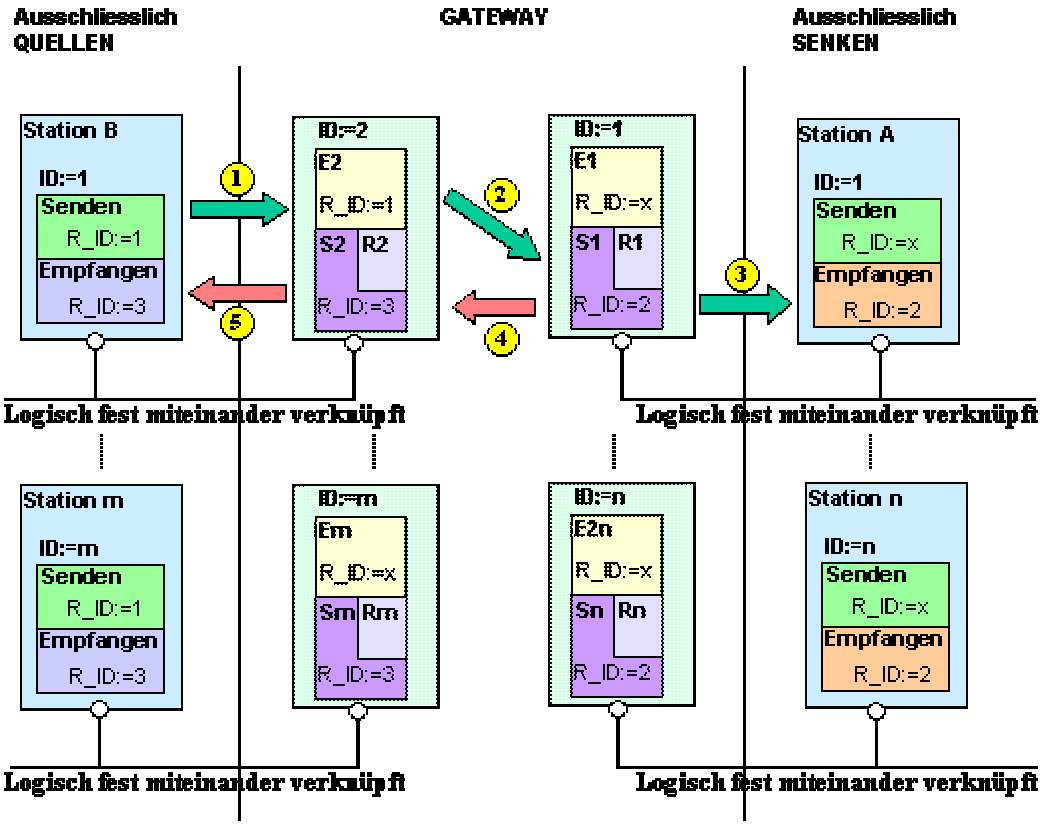


Bild 4-10 Datenflussmodell für unidirektionales Senden/Empfangen

Erläuterung zum Datenflussmodell

Das Gateway besteht aus 2 Kernelementen:

- Empfangseinheiten und
- Sendeeinheiten mit fest zugeordnetem Ringpuffer

Festlegungen:

- Jede Empfangseinheit ist über die Verbindungprojektierung und die ID-Vergabe logisch fest mit einer Quellstation verbunden
- Jede Sendeeinheit ist über die Verbindungprojektierung und die ID-Vergabe logisch fest mit einer Zielstation verbunden
- Jede Quellstation benutzt zum Senden der Daten die R_ID:=1
- Jede Zielstation benutzt zum Empfangen der Daten die R_ID:=2
- Jede Quellstation benutzt zum Empfangen der Quittung die R_ID:=3

➔ Daraus resultiert eine feste Verschaltung zwischen den BSEND/BRCV-Bausteinen in den Quell-/Zielstationen und im Gateway.

Zeitlicher Ablauf des Datenflusses

Tabelle 4-13

Schritt	Aktion	Erläuterung
1	Station B sendet Daten an das Gateway	Die mit der Station B fest verbundene Empfangseinheit (hier: E2) empfängt das Datentelegramm
2	Empfangseinheit E2 leitet die Daten an die Sendeeinheit S1 weiter.	Die Empfangseinheit E2 analysiert den Telegrammkopf und trägt das Telegramm in den Ringpuffer R1 der Sendeeinheit S1 ein
3	Sendeeinheit S1 sendet Telegramm	Die Sendeeinheit S1, die fest mit der Station A verschaltet ist, liest das Telegramm aus ihrem Ringpuffer R1 aus und sendet es an die Zielstation Station A.
4	Sobald das Senden an die Station A beendet ist, generiert die Sendeeinheit S1 ein Quittungstelegramm und leitet es an die Sendeeinheit S2 weiter.	Damit die Station B eine Quittung über den Sendevorgang erhält, muss die Sendeeinheit S1 ein Quittungstelegramm generieren und dieses in den Ringpuffer R2 der Sendeeinheit S2 eintragen, die fest mit der Station B verschaltet ist.
5	Sendeeinheit S2 sendet das Quittungstelegramm	Die mit der Station B fest verbundene Sendeeinheit S2 sendet das Telegramm in ihrem Ringpuffer R2 an die Station B.
6	Station B empfängt Quittung.	

4.5.2 Darstellung der Datenflussmodelle in der Quell-/Zielstation und im Gateway

Nachfolgend werden nun die verfeinerten Datenflussmodelle in der Quell-/Zielstation und dem Gateway dargestellt.

Dabei wird das bidirektionale Senden und Empfangen dargestellt.

Quell-/Zielstation

Folgendes Bild zeigt den Datenfluss in der Quell-/Zielstation:

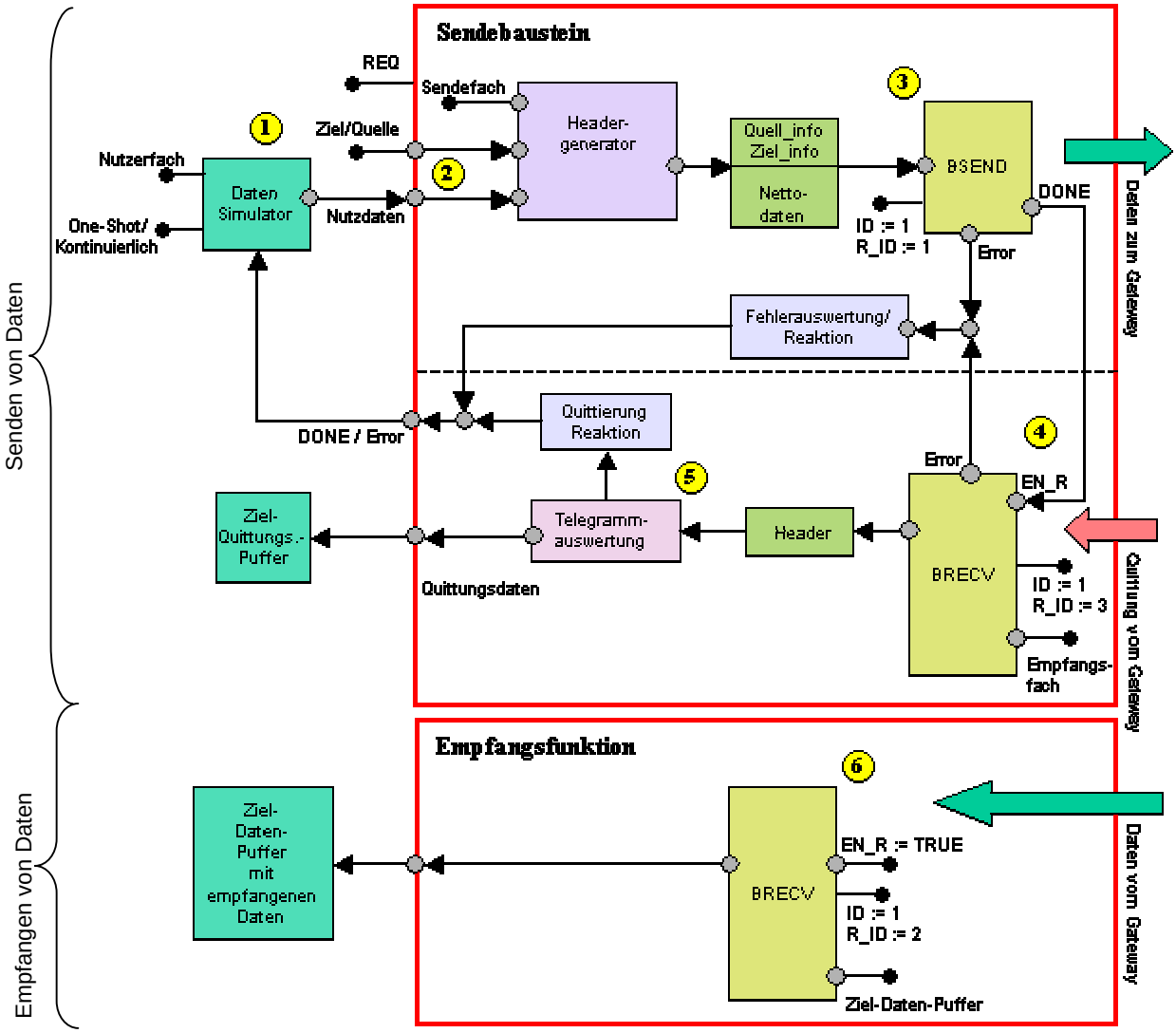


Bild 4-11

Beschreibung der Abläufe in der Quell-/Zielstation

Tabelle 4-14

Schritt	Beschreibung
1	Ein Datensimulator beschreibt einen vorgegebenen Speicherbereich.
2	Die so simulierten Nutzdaten und die Quell-/Zielinformationen werden dem Sendebaustein mit REQ=1 übergeben. Der Sendebaustein wird sodann für weitere Aufrufe verriegelt.
3	Der Sendebaustein erstellt mit dem Headergenerator ein aus Nutzdaten und Quell-/Zielinformationen zusammengesetztes Telegramm, welches dem BSEND-Baustein mit einer positiven Flanke übergeben wird. Auftretende Fehler werden mit einem Fehlerbaustein erfasst. Im Fehlerfall wird dies der Schnittstelle gemeldet und der Baustein wird wieder frei geschaltet.
4	Sobald der Sendevorgang abgeschlossen ist, wird ein BRCV-Baustein gestartet, um auf das Quittungstelegramm zu warten. Auftretende Fehler werden mit einem Fehlerbaustein erfasst. Im Fehlerfall wird dies der Schnittstelle gemeldet und der Baustein wird wieder frei geschaltet.
5	Sobald das Quittungstelegramm empfangen wurde, werden die Kopfdaten ausgewertet. Auftretende Fehler werden mit einem Fehlerbaustein erfasst. Im Fehlerfall wird dies der Schnittstelle gemeldet und der Baustein wird wieder frei geschaltet. Für den Fall, dass die Quittungsdaten eingehender untersucht werden sollen, wird ein Any-Pointer auf das Empfangsfach zurückgeliefert.
6	Um Daten zu empfangen, wird lediglich ein BRCV-Baustein aufgerufen.

Hinweis

Da jede Station mit dem Gateway über eine feste Verbindungsprojektierung verbunden ist, sind die Parameter ID und R_ID fest vorgegeben. Der Parameter ID muss bei einer anderweitigen Projektierung angepasst werden, wohingegen der Parameter R_ID nicht verändert werden darf (siehe Kap. 4.5.1).

Gateway

Das folgende Datenflussdiagramm stellt die Situation im Gateway dar, wenn Station x über das Gateway Daten an Station y sendet.

Dabei wird jeweils nur die betroffene Empfangs- und Sendeeinheit gezeigt (aus Übersichtsgründen wird das Element „Pointergenerator“ nicht dargestellt):

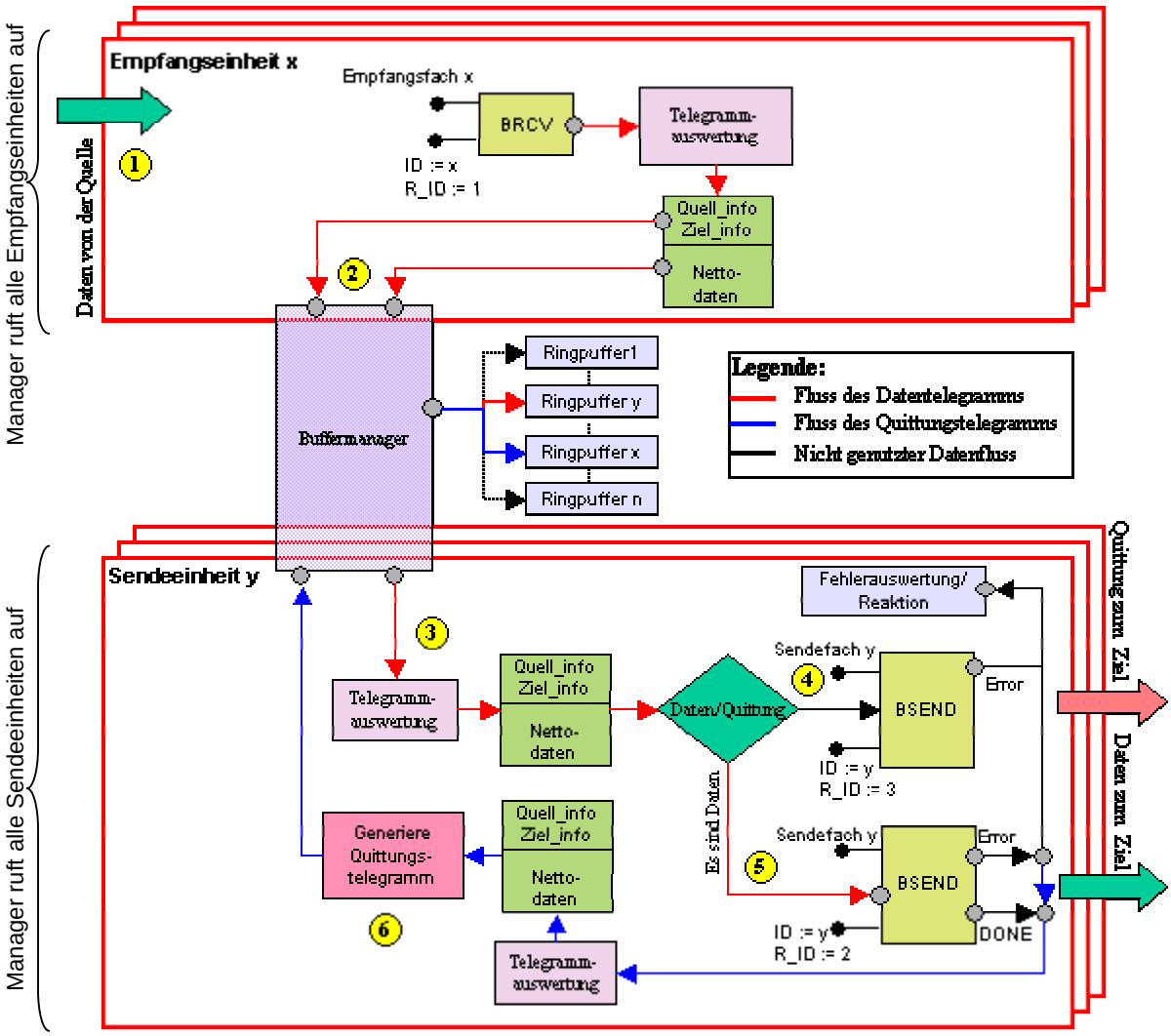


Bild 4-12 Datenflussmodell im Gateway

Beschreibung der Abläufe im Gateway

Tabelle 4-15

Schritt	Beschreibung
1	Jede Empfangseinheit empfängt auf der ihr zugewiesenen Verbindung Datentelegramme. In diesem Fall empfängt die Empfangseinheit x das Datentelegramm der Station x.
2	Die Datentelegramme werden analysiert und über den Buffermanager in den Ringpuffer der Ziel-Sende-Einheit geschrieben – dies entspricht der Routing-Analyse. In diesem Fall ist das Ziel Station y. Dementsprechend muss das Datentelegramm in den Ringpuffer y eingetragen werden. **Hinweis:** Bei einer fehlerhaften Zielinformation (z.B. der Eintrag 0) wird das empfangene Datentelegramm verworfen und es wird **keine** Fehlermeldung generiert. Es muss sicher gestellt sein, dass die Ziel- und Quellinformationen im Telegrammkopf korrekt sind (siehe dazu auch Kap. 8.4).
3	Jede Sendeeinheit liest das älteste Datum (FIFO-Ringpuffer) aus ihrem Ringpuffer aus. In diesem Fall liest die Sendeeinheit y den ältesten Eintrag ihres Ringpuffers y aus. Durch die Telegrammanalyse wird festgestellt, ob es sich um ein Daten- oder ein Quittungstelegramm handelt.
4	Handelt es sich um ein Quittungstelegramm, so wird eine Quittung gesendet. Es findet kein weiterer Quittierungsmechnismus statt. Lediglich auftretende Fehler werden protokolliert. Der Algorithmus endet. Im anderen Fall handelt es sich um ein Datentelegramm → Schritt 5.
5	Die Daten werden an das Ziel gesendet. Auftretende Fehler werden protokolliert.
6	Unabhängig vom Erfolg des Sendevorgangs wird ein Quittungstelegramm generiert, indem der Telegrammkopf des Sendefachs leicht modifiziert wird. Dabei wird der Status des BSEND-Bausteins in das Quittungstelegramm eingetragen. Das Quittungstelegramm wird sodann über den Buffermanager in den Ringpuffer der jeweiligen Sendeeinheit eingetragen. In diesem Fall war die Quelle des Datentelegramms Station x. Dementsprechend wird das Quittungstelegramm in den Ringpuffer x eingetragen. Der Algorithmus endet.

Das folgende Kapitel stellt die einzelnen Funktionselemente, die den Ablauf maßgeblich beeinflussen, vor.

4.6 Funktionselemente in der Applikationssoftware

Inhalt des Kapitels

In diesem Kapitel erhalten Sie eine Übersicht über die verwendeten Funktionselemente.

Zunächst werden die zentralen Steuerfunktionen, die „Manager“-Bausteine, vorgestellt, ehe Sie dann über die einzelnen Funktionselemente auf schematisierter Ebene einen umfassenden Überblick darüber erhalten, wie die Applikation letztendlich realisiert wurde.

Eine detailliertere Darstellung der wichtigsten Elemente erhalten Sie auf Code-Ebene im Teil C bzw. Kap. 7.

Das Kapitel enthält folgende Unterkapitel:

Tabelle 4-16

Kap.	Name des Kapitels	Primäre Aufgabe	Anwendung in Quell-/Zielstation	Anwendung in Gateway
4.6.1	FB1 „Manager“	Steuert das Senden und Empfangen von Daten	X	
4.6.2	FC1 „Manager“	Koordiniert die Aufrufe der Empfangs- und Sendeeinheiten		X
4.6.3	Pointergenerator	Erstellt einen Any-Pointer auf einen Datenbaustein	X	X
4.6.4	Datensimulator	Füllt einen Datenbaustein mit simulierten Nutzdaten	X	
4.6.5	Headergenerator	Erstellt aus den Eingabedaten einen Kopf und fügt diesen einem gegebenen Datenbaustein hinzu.	X	
4.6.6	Fehlerbehandlung	Protokollierung von Fehlern bei BSEND/BRCV-Aufrufen und Ermittlung einer Fehlerreaktion	X	X
4.6.7	Statistik	Ermittlung des höchsten, niedrigsten und Mittelwertes aus einer Reihe von Messwerten.	X	X
4.6.8	Sendebaustein	Sendet ein Datentelegramm	X	
4.6.9	Ringpuffer	Speichert Daten in einer Ringstruktur		X
4.6.10	Empfangeinheit(en)	Empfängt Nutzdaten und bereitet das Routing vor		X
4.6.11	Sendeeinheit(en)	Sendet Daten- und Quittungstelegramme zu den Quell-/Zielstationen		X

4.6.1 FB1 „Manager“ (Quell-/Zielstation)

Bausteinhierarchie

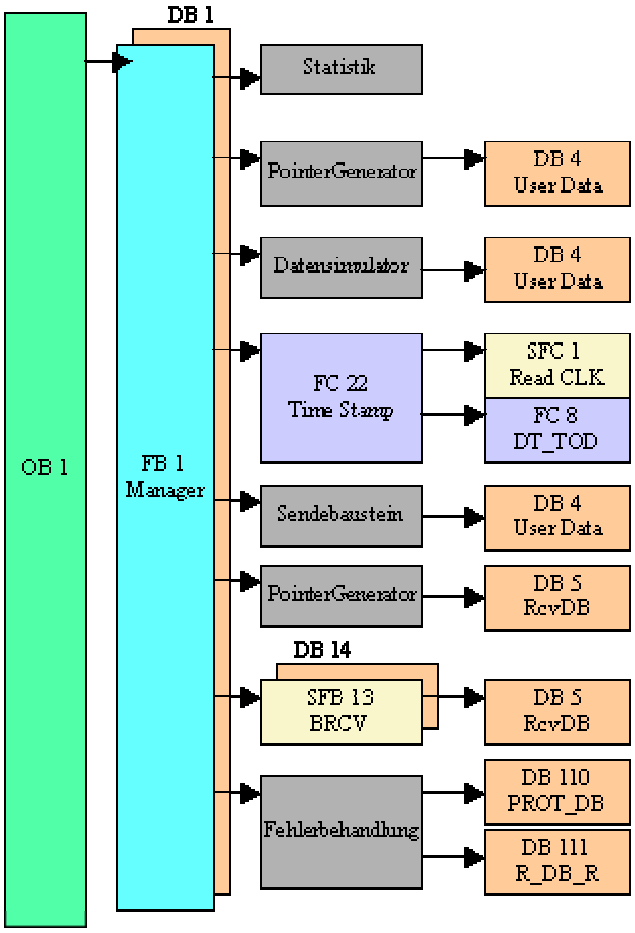


Bild 4-13

Aufgabenstellung

Der FB1 „Manager“ hat folgende Aufgaben:

- Koordinierter Aufruf der einzelnen Funktionselemente:
  - Ausführen der Statistik-Funktionalität
  - Simulation der Nutzdaten
  - Anstoß des Sendebausteins
  - Empfangen von Datentelegrammen
- Steuerung der Simulation:
  - Senden mit Einmaltriggerung
  - Senden mit automatischer und wiederholter Triggerung

Anwenderschnittstelle

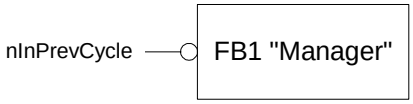


Bild 4-14

Tabelle 4-17

Name	Typ	Erklärung
nInPrevCycle	Int	Zykluszeit des vorhergehenden Zyklus

4.6.2 FC1 „Manager“ (Gateway)

Bausteinhierarchie

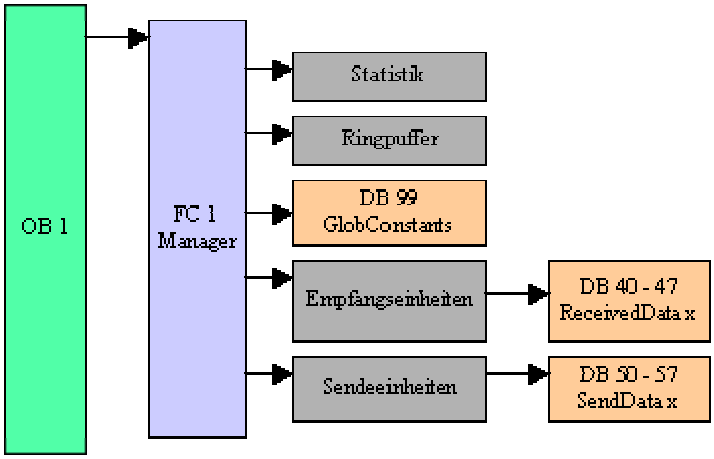


Bild 4-15

Aufgabenstellung

Der FC1 „Manager“ hat folgende Aufgaben:

- Ausführen der Statistik-Funktionalität.
- Prüfen, ob ein Ringpuffer voll ist.
  Wenn ein Ringpuffer voll ist, so wird nicht mehr empfangen.
- Zyklischer Aufruf aller Empfangseinheiten, sofern kein Ringpuffer voll ist.
- Zyklischer Aufruf aller Sendeeinheiten.

Lösungsansatz

Der FC1 „Manager“ liest zunächst den Status aller Ringpuffer aus. Zeigt der Status eines einzigen Ringpuffer an, dass er voll ist, so wird mit den Sendeeinheiten fortgefahren. Dadurch wird verhindert, dass innerhalb des Gateways ein Datentelegramm „verloren“ gehen kann. Der Sendevorgang auf der Quellstation verlängert sich entsprechend.

Sowohl die Empfangs- als auch die Sendeeinheiten werden innerhalb einer Schleife aufgerufen. Der Schleifenzähler entspricht der Anzahl zu verwaltender Stationen.
Innerhalb einer Schleife führen Fehler, die die Empfangs- oder Sendeeinheit liefert (z.B. ein Fehler beim Schreiben in den Ringpuffer), zu einem Übergang zur nächsten Empfangs- bzw. Sendeeinheit. Ein Fehlerhandling ist hierzu nicht implementiert, da die Fehlerwahrscheinlichkeit aufgrund der Architektur sehr unwahrscheinlich ist (siehe dazu Kap. 8.4.2).

Weiteres Fehlerhandling, wie z.B. bei BSEND/BRCV-Aufrufen, findet in den untergeordneten Bausteinen über das Funktionselement „Fehlerbehandlung„ (siehe Kap. 4.6.6) statt.

Anwenderschnittstelle

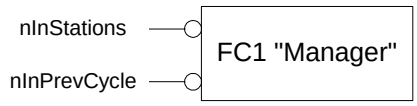


Bild 4-16

Tabelle 4-18

Name	Typ	Erklärung
nInStations	Int	Anzahl der zu verwaltenden Stationen. Das Maximum dieses Parameters ist durch die Anzahl an Empfangs- und Sendeeinheiten sowie projektierter Verbindungen beschränkt. Bei einem fehlerhaften Zahlenwert geht das Datentelegramm verloren (für weiteres Fehlerhandling siehe Kap. 8.4).
nInPrevCycle	Int	Zykluszeit des vorhergehenden Zyklus

4.6.3 Pointergenerator (Quell-/Zielstation und Gateway)

Bausteinhierarchie

Bild 4-17

Aufgabenstellung

Um Datenbausteine dynamisch adressieren zu können, müssen Any-Pointer verwendet werden, die die Datenbausteinnummer, die Anfangsadresse und die Länge des zu adressierenden Datenbereichs spezifizieren. Da die Erstellung solcher Any-Pointer relativ mühsam ist und diese häufig benutzt werden müssen, wurde ein Pointergenerator implementiert.

Lösungsansatz

Der Pointergenerator erstellt zu einer gegebenen Datenbausteinnummer einen Any-Pointer, der an den Anfang des Datenbausteins und auf die gesamte Länge desselbigen zeigt.

Im Gateway wird der Baustein v.a. für Adressierung der Sende- und Empfangsfächer benutzt.
In der Quell-/Zielstation wird er bei der Erstellung der simulierten Nutzdaten sowie bei der Adressierung des Sende- und Empfangsfaches verwendet.

Anwenderschnittstelle

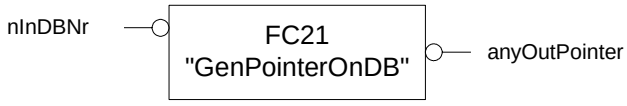


Bild 4-18

Tabelle 4-19

Name	Typ	Erklärung
nInDBNr	Int	Input-Parameter für die DB-Nummer. **Hinweis:** Der DB muss existieren!
anyOutPointer	Any-Pointer	Any-Pointer auf den Datenbaustein

4.6.4 Datensimulator (Quell-/Zielstation)

Bausteinhierarchie

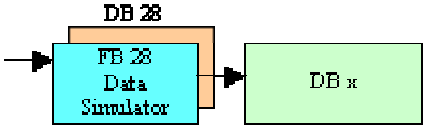


Bild 4-19

Aufgabenstellung

Das Funktionselement „Datensimulator“ generiert aus der Vorgabe des durch den Pointergenerator erstellten Zieldatenpointers und der vorgegebenen Datenlänge die zu übertragenden Nutzdaten.

Lösungsansatz

Um diese Funktionalität zu realisieren, wurde der FB 28 „Data_Simulator_S7_400“ erstellt. Er generiert selbstständig den geforderten Datenbereich.

Hierzu beginnt der Funktionsbaustein damit, den Startpunkt der Daten um die Headergröße zu verschieben. Danach werden die Daten in ASCII Wer-

ten der Buchstaben alphabetisch von „A“ bis „Z“ in den Zielbereich geschrieben. Wurde der Buchstabe „Z“ erreicht, wird wieder bei „A“ begonnen.

Anwenderschnittstelle

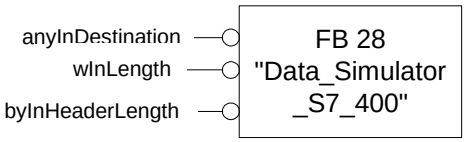


Bild 4-20

Tabelle 4-20

Name	Typ	Erklärung
anyInDestination	Any-Pointer	Ziel-Datenbereich, der beschrieben werden soll
wInLength	Word	Länge des Datenbereichs
byInHeaderLength	Byte	Länge des Bereichs, der für Kopfdaten freigelassen werden soll. Dieser Parameter ist für diese Simulation mit 0 belegt, um einen Datenbereich mit reinen Nutzdaten zu simulieren.

4.6.5 Headergenerator (Quell-/Zielstation)

Bausteinhierarchie

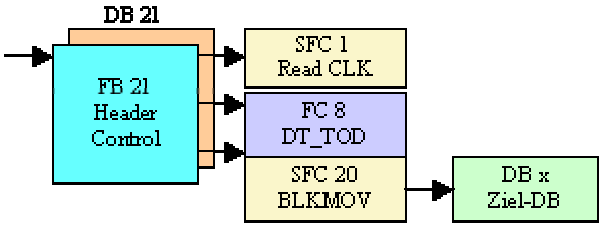


Bild 4-21

Aufgabenstellung

Damit das Gateway entscheiden kann, wohin das Telegramm gesendet werden soll, sind Quell-/Zielinformationen notwendig.
Um zusätzlich den aufgesetzten Quittungsmechanismus realisieren zu können, sind weitere Datenstrukturen unerlässlich.

Der Header hat folgenden Aufbau:

Address	Name	Type	Initial val	Comment
0.0		STRUCT		
+0.0	byType	BYTE	B#16#0	is it data or an acknowledgement
+2.0	TimeStamp	TIME_OF_DAY	TOD#0:0:0.0	Timestamp
+6.0	tSourceInfo	"IDData"		Source information
+8.0	tDestInfo	"IDData"		Destination information
+10.0	wTelegrammID	WORD	W#16#0	ID of telegram for sending different telegramms on one port
+12.0	wError	WORD	W#16#0	Error Code of used "BSEND" in Gateway
+14.0	wStatus	WORD	W#16#0	Status of routing
=16.0		END_STRUCT		

Bild 4-22

(Nähere Information zur Kopfstruktur erhalten Sie im Kapitel 7.1.4 unter „Notwendige Datenstrukturen“)

Lösungsansatz

Der Headergenerator erstellt nun aus gegebenen Eingangsparametern einen Telegrammkopf mit vorgegebener Struktur.

Dabei wird die erstellte Struktur an den Anfang eines gegebenen Datenbereichs geschrieben.

Hinweis

In dieser Applikation sind die Nutzdaten bereits im gegebenen Datenbereich vorhanden.

Anwenderschnittstelle

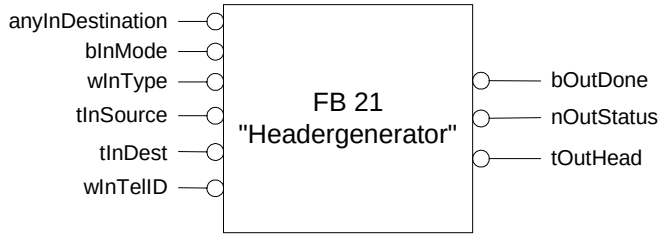


Bild 4-23

Tabelle 4-21

Name	Typ	Erklärung
anyInDestination	Any-Pointer	Zu modifizierender Datenbaustein
bInMode	Int	FALSE: Füge Header ein TRUE: Entferne Header und beschreibe Headerbereich mit Dummy-Daten In dieser Applikation wird der Baustein immer mit FALSE belegt. Ein Header wird also ausschließlich eingefügt.
wInType	Int	Telegrammtyp 1: Daten 2: Quittung
tInSource	Int	Local ID der Quellstation, gesehen vom Gateway aus

Name	Typ	Erklärung
tInDest	Int	Local ID der Zielstation, gesehen vom Gateway aus
wInTelID	Int	Telegramm-ID, um mehrere verschiedene Telegramme senden zu können, da diese Möglichkeit durch die feste Vergabe der R_ID sonst wegfallen würde.
bOutDone	Bool	Done-Bit
nOutStatus	Int	Status der Headergenerierung
tOutHead	Kopfstruktur	Enthält den Kopf, wenn bInMode := TRUE

4.6.6 Fehlerbehandlung (Quell-/Zielstation und Gateway)

Bausteinhierarchie

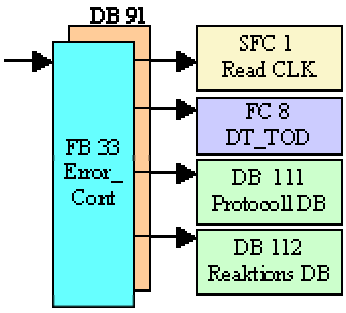


Bild 4-24

Aufgabenstellung

Das Funktionselement Fehlerbehandlung hat folgende Aufgaben:

- Protokollierung eines auftretenden Fehlers bei BSEND/BRCV-Aufrufen mit dem dazugehörigen Status (Protokoll-DB).
- Überprüfung, ob der vorgefundene Fehlerstatus bekannt ist und ob darauf regiert werden kann (Reaktions-DB).
- Ermittlung der projektierten Fehlerreaktion und Rückgabe derselbigen an die aufrufende Funktion.

Lösungsansatz

Das Funktionselement „Fehlerbehandlung“ ist nach jedem BSEND/BRCV-Aufruf positioniert. Dadurch werden auftretende Fehler sofort analysiert und protokolliert. Des weiteren wird eine Fehlerreaktion ausgegeben.

Zur Protokollierung der Fehler steht ein globaler Protokoll-Datenbaustein zur Verfügung. Dieser enthält ein Array von Datenstrukturen, die es ermöglichen, einen Fehler zeitlich und örtlich zurückverfolgen zu können.

Um eine Fehlerreaktion zu ermitteln, wurden 2 weitere Fehlerreaktions-Datenbausteine erstellt, die jedem Status eines BSEND oder BRCV-

Bausteins eine Reaktion zuordnen.
Das Funktionselement „Fehlerbehandlung“ sucht in dem entsprechenden Fehlerreaktions-Datenbaustein den jeweiligen Status und gibt die zugeordnete Reaktion aus.
Dadurch kann vom Anwender projektiert werden, auf welchen Fehlerstatus welche Reaktion (z.B. Restart oder Stop) ausgeführt werden soll.

Weitere Informationen zu diesem Funktionselement können Sie dem Kapitel 7.1.2 entnehmen.

Anwenderschnittstelle

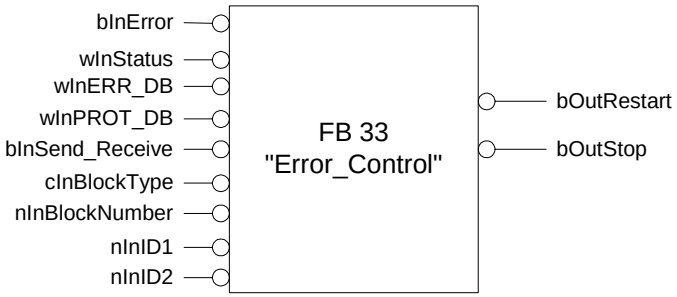


Bild 4-25

Tabelle 4-22

Name	Typ	Erklärung
bInError	bool	Falls TRUE, so wird der Baustein aktiv
wInStatus	Word	Abzuspeichernder Status des BSEND/BRCV
wInERR_DB	Word	Datenbaustein mit den projektierten Reaktionen
wInPROT_DB	Word	Datenbaustein, der die letzten 20 Fehler beinhaltet
bInSend_Receive	Bool	Spezifiziert, ob der Fehler bei einem BSEND- oder einem BRCV-Aufruf auftrat. TRUE: BRCV FALSE: BSEND
cInBlockType	Char	Optional: Angabe, in welcher Bausteinart der Fehler auftrat (Aufruf aus FB, FC etc.)
nInBlocknumber	Int	Optional: Angabe, aus welcher Bausteinnummer der BSEND/BRCV aufgerufen wurde
nInID1	Int	Optional: Freie Nummer In dieser Applikation wird hier die Local ID gespeichert
nInID2	Int	Optional: Freie Nummer In dieser Applikation wird hier die R_ID gespeichert
bOutRestart	Bool	Gibt an, ob der Auftrag wiederholt werden soll. Dies wird in dieser Applikation nicht ausgewertet.
bOutStop	Bool	Gibt an, ob der Auftrag gestoppt (Reset) werden soll. Dies wird in dieser Applikation nicht ausgewertet.

4.6.7 Statistik (Quell-/Zielstation und Gateway)

Bausteinhierarchie

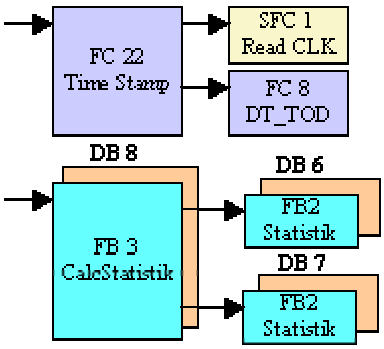


Bild 4-26

Aufgabenstellung

Das Funktionselement „Statistik“ soll aus einer Anzahl von Messwerten den höchsten, niedrigsten und mittleren Wert (bzgl. 100 Messwerten) berechnen.

Die Messwerte dieser Applikation entsprechen der Telegrammlaufzeit und der Zykluszeit. Dabei werden immer die letzten 100 Werte berücksichtigt.

Lösungsansatz

Das Funktionselement Statistik besteht im wesentlichen aus dem Mittelwertbildner FB2 „Statistik“.

Dieser Baustein ist nochmals durch den Baustein FB3 „CalcStatistik“ gekapselt, so dass folgendes Vorgehen ermöglicht wird:

- Merke die CPU-Zeit unmittelbar vor dem Senden von Daten
- Merke die CPU-Zeit unmittelbar nach dem Empfangen der Quittung
- Errechne Differenz der beiden Zeiten (→ daraus resultiert eine Zeitdauer)
- Ermittle dabei sogleich die geforderten Werte
- Setze den Mittelwertbildner nach 100 Messungen zurück.

Hinweise

- Der Zeitunterschied wird durch Endzeit – Anfangszeit errechnet.
  Sollte Anfangszeit > Endzeit gelten, so wird ein falscher Wert berechnet. Dieser Fall kann nur dann eintreten, wenn die Triggerung zum Starten des Sendevorgangs asynchron erfolgt.
- Da im Gateway lediglich die Zykluszeit gemessen wird, ist eine Kapselung des Mittelwertbildners nicht notwendig gewesen. Dort wird der Mittelwertbildner (dort: FB6 „Statistik“) direkt aus dem FC1 „Manager“ aufgerufen.

Anwenderschnittstelle

Baustein FC22 „TimeStamp“:

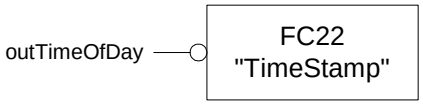


Bild 4-27

Tabelle 4-23

Name	Typ	Erklärung
outTimeOfDay	TOD	Dies ist die aktuelle Tageszeit in der SIMATIC-CPU

Baustein FB3 „CalcStatistic“:

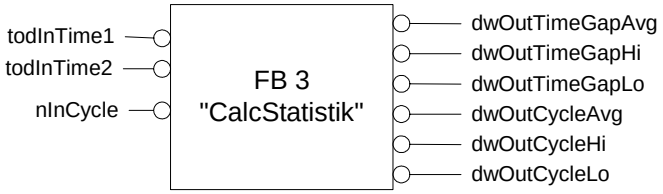


Bild 4-28

Tabelle 4-24

Name	Typ	Erklärung
todInTime1	TOD	Anfangszeit
todInTime2	TOD	Endzeit
nInCycle	Int	Letzte Zykluszeit
dwOutTimeGapAvg	DWORD	Durchschnittlicher Zeitunterschied
dwOutTimeGapHi	DWORD	Größter Zeitunterschied
dwOutTimeGapLo	DWORD	Kleinster Zeitunterschied
dwOutCycleAvg	DWORD	Durchschnittliche Zykluszeit
dwOutCycleHi	DWORD	Größte Zykluszeit
dwOutCycleLo	DWORD	Kleinste Zykluszeit

4.6.8 Sendebaustein (Quell-/Zielstation)

Bausteinhierachie

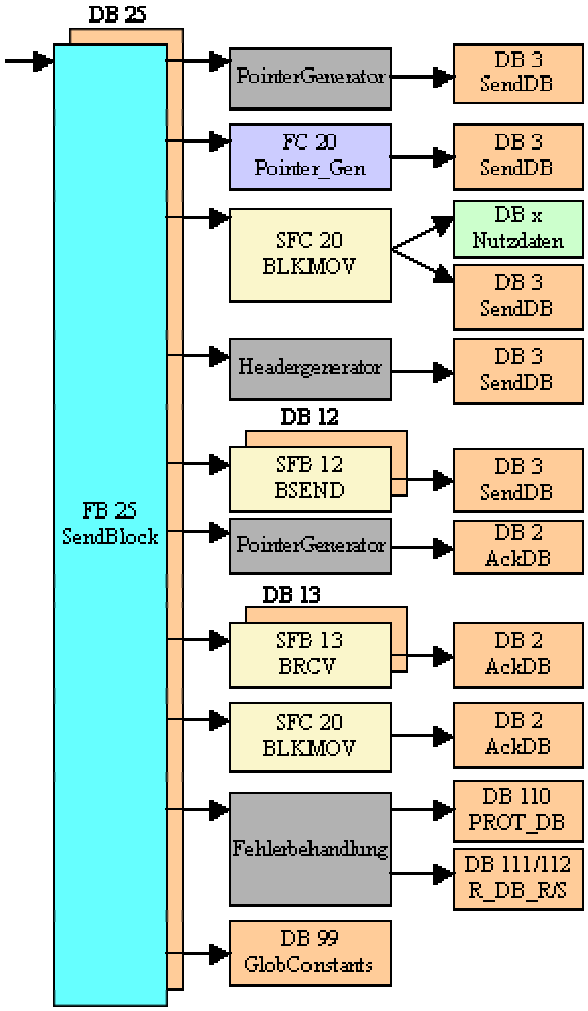


Bild 4-29

Aufgabenstellung

Der FB25 „SendBlock“ ist das zentrale Element in der Quell-/Zielstation.

Er hat folgende Aufgaben:

- Ausgehend von Nutzdaten muss ein Telegramm erstellt werden, das vom Gateway weitergeleitet werden kann
- Der BSEND-Baustein muss korrekt versorgt werden
- Der Sendebaustein muss auf die Quittung des Gateways warten und diese auswerten

- Solange kein Fehler auftritt und solange der Baustein sendet oder auf eine Quittung wartet, darf er kein weiteres Telegramm verschicken. Dies entspricht einer Verriegelungsfunktion während eines Sendevorgangs.

Der Sendebaustein stellt also die Kommunikationsschnittstelle dar, um über das Gateway Nutzdaten zu einer anderen Station zu senden.

Funktionsprinzip

Dem Sendebaustein liegt folgendes vereinfachtes Zustandsdiagramm zu Grunde:

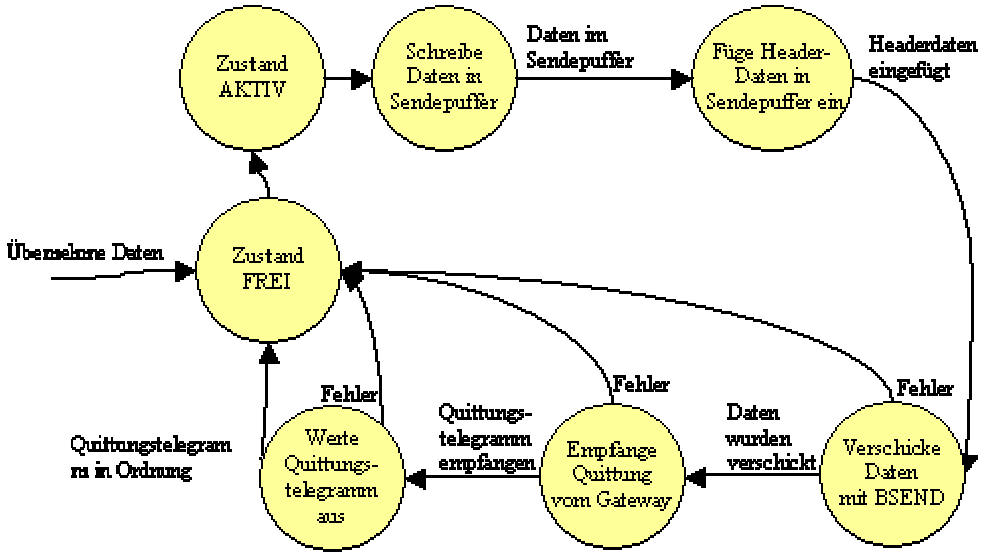


Bild 4-30

Sobald der Baustein über bIOBSEND_REQ eine 1 (TRUE) bekommt, startet der Algorithmus.

! **Wichtig**

- Der Parameter bIOBSEND_REQ wird vom Sendebaustein immer auf 0 (FALSE) zurückgesetzt
- Es sei nochmals darauf hingewiesen, dass der Baustein nur dann Daten übernimmt, wenn er sich um Zustand „FREI“ befindet.
  Dies kann zu Problemen führen, falls ein Quittungstelegramm nicht empfangen wird, da dann der Sendebaustein blockiert ist. Für nähere Informationen siehe Kap. 8.4.

Anwenderschnittstelle

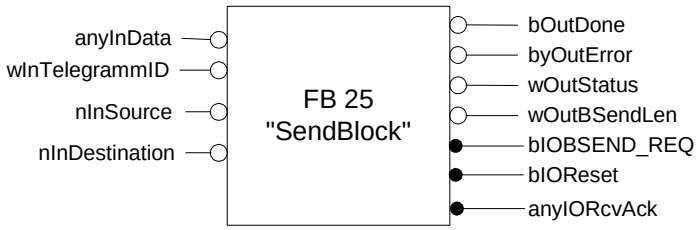


Bild 4-31

Tabelle 4-25

Name	Typ	Erklärung
anyInData	Any-Pointer	Zeiger auf die Nutzdaten
wInTelegrammID	Word	Telegramm-ID, um den Wegfall der R_ID zu kompensieren
nInSource	Int	Local ID der Quellstation, gesehen vom Gateway aus
nInDestination	Int	Local ID der Zielstation, gesehen vom Gateway aus
bOutDone	Bool	Bit, welches angibt, ob der Sendebaustein die Daten erfolgreich gesendet hat
byOutError	Byte	Gibt an, in welchem Bereich ein Fehler auftrat: 1: bei BSEND 2: Bei BRCV 3: Auswertung des Quittungstelegramms ergab einen Fehler
wOutStatus	Word	Status zum angegebenen Fehler
wOutBSendLen	Word	Länge der gesendeten Daten
bIOBSEND_REQ	Bool	Gibt an, dass Daten gesendet werden sollen. Wird vom Sendebaustein wieder zurückgesetzt.
bIOReset	Bool	Setzt den Baustein zurück.
anyIORcvAck	Any-Pointer	Zeiger auf das Quittungstelegramm

4.6.9 Ringpuffer (Gateway)

Bausteinhierarchie

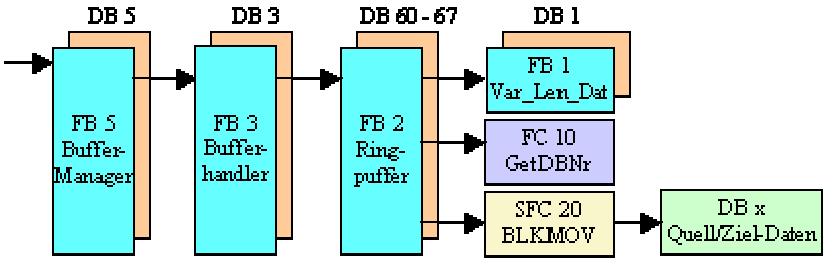


Bild 4-32

Aufgabenstellung

Das Funktionselement „Ringpuffer“ hat folgende Aufgaben:

- Zeitliche Entkopplung des Datenflusses zwischen Quell- und Zielstation. Die Entkopplung ist notwendig, da die Subnetze unterschiedliche Spezifikationen hinsichtlich Schnelligkeit und Deterministik besitzen (siehe Kap. 4.3).
- Feste Assoziation zur entsprechenden Sendeeinheit.
  Der Ringpuffer enthält alle Telegramme, die die damit assoziierte Sendeeinheit verschicken soll. Dabei ist es unerheblich, ob es sich um ein Daten- oder ein Quittungstelegramm handelt.
- Bereitstellen einer Anwenderschnittstelle, um ein einfaches Datenhandling zwischen Empfangs- und Sendeeinheiten realisieren zu können.
- Speicherung von bis zu 10 Telegrammen.

Lösungsansatz

Der Ringpuffer speichert Datenbereiche vorgegebener Größe in einer Ringstruktur.

Dabei gilt:

- Die Ringstruktur ist statisch, d.h. die Größe wird zur Kompilierzeit vorgegeben und kann nicht angepasst werden.
  Jede Ringstruktur hat eine Größe von 10.
- Es wird immer das älteste Element ausgelesen (→ FIFO-Prinzip)
- Ein neues Element wird immer hinter dem jüngsten Element gespeichert
- Ist der Puffer voll, so wird wieder an der 1. Stelle des Puffers angefangen (→ Ringstruktur)
- Der Ringpuffer unterstützt folgende Modi:
  - Schreiben
  - Lesen

- Überschreiben (überschreibt das älteste Element, auch wenn es noch nicht ausgelesen wurde)
- Status (liest den Status des Ringpuffers aus; z.B. BUFFER_FULL)
- Löschen (löscht die Daten im Ringpuffer)

- Der Baustein FB5 „Buffermanager“ dient als Schnittstelle zu den Ringpuffern. Durch die Eingabeparameter wird der entsprechende Ringpuffer ausgewählt und die gewünschte Operation ausgeführt.
  Der Baustein stellt darüber hinaus Ausgabeparameter zur Verfügung, um den Status des jeweiligen Ringpuffers zurückzuliefern.

Anwenderschnittstellen

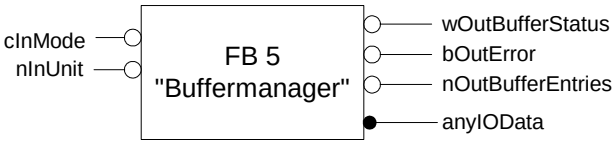


Bild 4-33

Tabelle 4-26

Name	Typ	Erklärung
cInMode	Char	Operation, die der Ringpuffer ausführen soll: • ‚C’: Ringpuffer löschen • ‚S’: Status auslesen • ‚O’: Überschreiben • ‚R’: Lesen • ‚W’: Schreiben
nInUnit	Int	Nummer des zu bearbeitenden Ringpuffers. In der Applikation werden max. 8 Ringpuffer verwendet.
wOutBufferStatus	Word	Status des Ringpuffers: • S_OK: 0 • BLK_ERR (Fehler bei BlkMov): 4 • S_OV (Overflow): 1 • BUFFER_EMPTY: 3 • BUFFER_FULL: 2
bOutError	Bool	Zeigt an, ob ein Fehler aufgetreten ist
nOutBufferEntries	Int	Anzahl der Einträge im Ringpuffer
anyIOData	Any-Pointer	Zeiger auf die Quell-/Zieldaten

4.6.10 Empfangseinheiten (Gateway)

Bausteinhierarchie

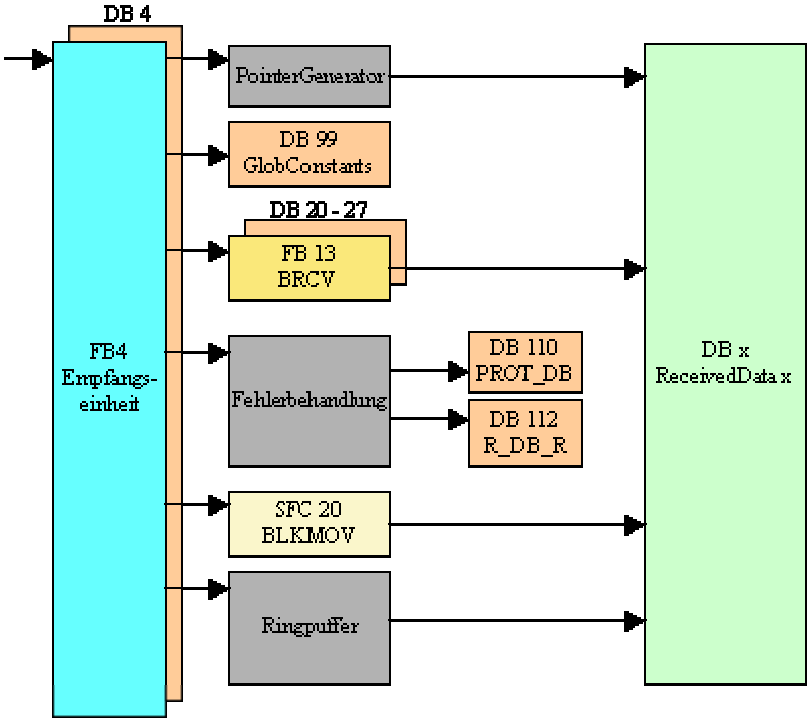


Bild 4-34

Aufgabenstellung

Eine Empfangseinheit soll mittels BRCV bzgl. einer fest verschalteten Verbindung ständig Telegramme empfangen.

Sobald ein Telegramm vollständig empfangen wurde, sollen die Daten in den Ringpuffer der jeweiligen Sendeeinheit eingetragen werden (s. Kap. 4.5).

Lösungsansatz

Jede Empfangseinheit beinhaltet genau einen BRCV-Aufruf mit einer fest vorgegebenen ID und R_ID.

Nach dem Aufruf der Empfangseinheit wird zunächst mit dem Pointergenerator ein Any-Pointer auf das jeweilige Empfangsfach generiert und dem BRCV-Baustein übergeben.

Sobald das NDR-Bit des BRCV-Bausteins den erfolgreichen Empfang der Daten meldet, wird das Telegramm analysiert.
Dabei wird der Telegrammkopf untersucht. Anhand der enthaltenen Zielinformation wird das Telegramm in den Ringpuffer der assoziierten Sende-

einheit eingetragen (s. Kap. 4.5).
Dadurch wird das Datensatz-Routing in Zielrichtung realisiert.

Der Baustein liefert umfangreiche Fehlerstati zurück, die jedoch in der aktuellen Applikation nicht ausgewertet werden.

! **Wichtig**

- Die Empfangseinheiten sind untereinander unabhängig. Sie arbeiten asynchron.
- Jeder BRCV-Baustein ist immer aktiv. EN_R ist also immer TRUE. Damit keine inkonsistenten Daten empfangen werden können, muss die Quellstation einen entsprechenden Algorithmus verwenden und die Empfangseinheit die empfangenen Daten sichern. Der benutzte Baustein FB25 „SendBlock“ in der Quellstation stellt dies sicher.

Anwenderschnittstelle

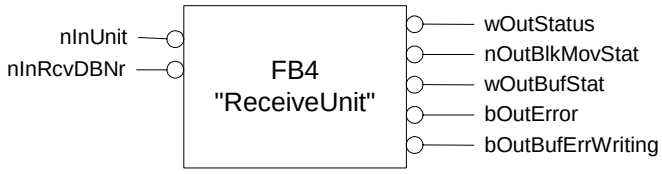


Bild 4-35

Tabelle 4-27

Name	Typ	Erklärung
nInUnit	Int	Spezifiziert die Nummer der Empfangseinheit und damit auch die Local ID, über die empfangen wird
nInRcvDBNr	Int	Empfangsfach der jeweiligen Empfangseinheit
wOutStatus	Word	Liefert den Status des BRCV-Bausteins
nOutBlkMovStat	Int	Liefert den Status des BlkMov-Bausteins
wOutBufStat	Word	Liefert den Status des Ringpuffers an
bOutError	Bool	Zeigt an, dass ein Fehler auftrat
bOutBufErrWriting	bool	Zeigt an, ob ein Fehler beim Schreiben in den Ringpuffer auftrat

4.6.11 Sendeeinheiten (Gateway)

Bausteinhierarchie

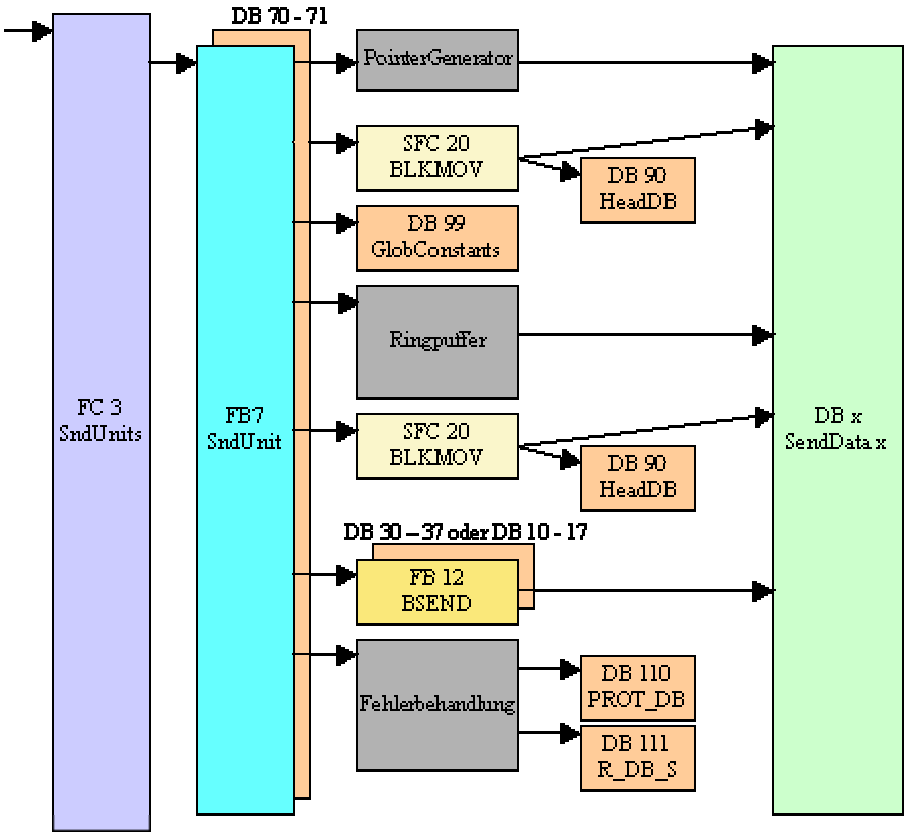


Bild 4-36

Aufgabenstellung

Die Sendeeinheit hat die Aufgabe, das älteste Datum ihres Ringpuffer auszulesen und an das Ziel zu verschicken.

Ist der Sendevorgang beendet, so soll, falls ein Datentelegramm gesendet wurde, ein Quittungstelegramm generiert werden, welches in den Ringpuffer der Sendeeinheit eingetragen werden soll, die wiederum an die jeweilige Quellstation sendet.

Funktionsprinzip

Jede Sendeeinheit des Gateways schickt immer nur über einen Kommunikationskanal (eine Local ID) (s. Kap. 4.5).
Dieses Vorgehen ist aufgrund der Anwenderschnittstelle der BSEND-Bausteine nötig (s. Kap. 4.4.4).

Jede Sendeeinheit besitzt einen assoziierten Ringpuffer.
Dabei gilt, dass jede Sendeeinheit ausschliesslich Daten sendet, die sie aus ihrem Ringpuffer ausgelesen hat.

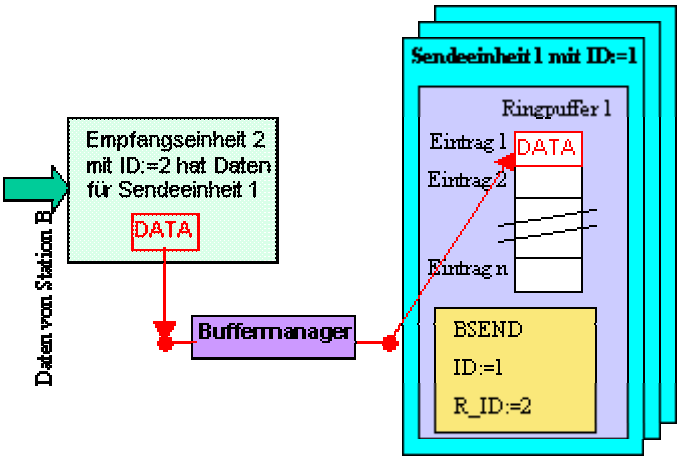
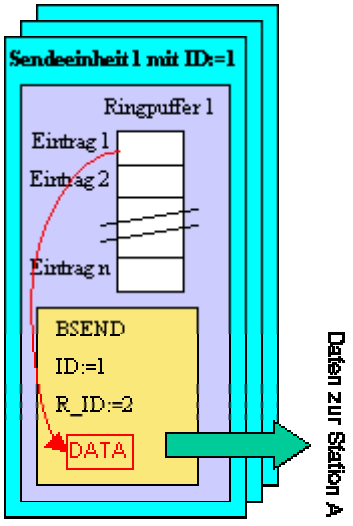
Als Resultat muss eine Sendeeinheit das Quittungstelegramm in einen Ringpuffer schreiben, der der Sendeeinheit zugeordnet ist, die zur jeweiligen Quellstation sendet.

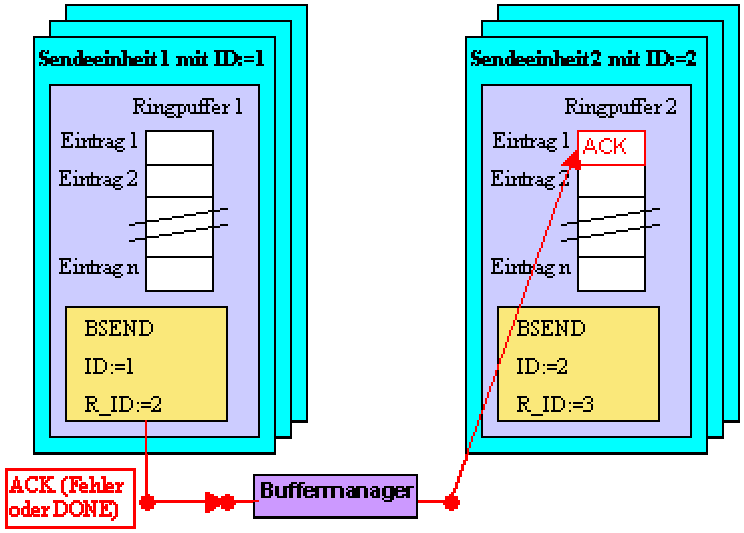
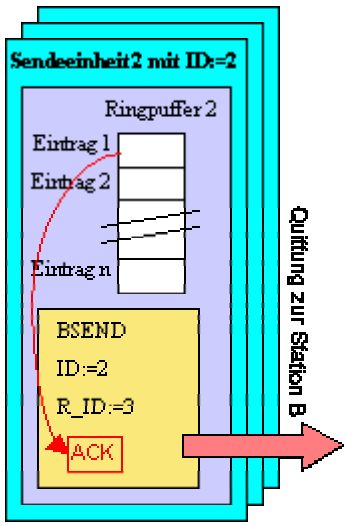
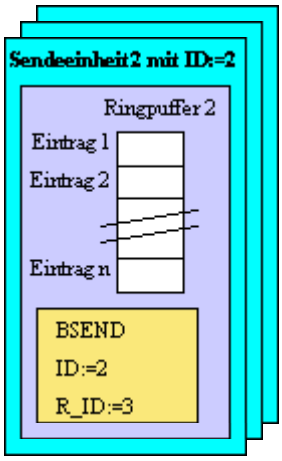
Beispiel

Folgendes Beispiel illustriert die Vorgänge:

Es sollen Daten von der Station B (Local ID:=2 (Quelle)) an die Station A (Local ID:=1 (Ziel)) gesendet werden. (Die IDs sind immer aus Sicht des Gateways) (siehe auch Kap. 4.5).

Tabelle 4-28

Schritt	Erklärung	Diagramm
1	Die Empfangseinheit 2 hat Daten in den Ringpuffer 1 eingetragen. Also hat die Sendeeinheit 1 Daten in ihrem Ringpuffer.	
2	Die Daten werden ausgelesen und verschickt. Die Sendeeinheit 1 verschickt die ausgelesenen Daten immer an die Station, die an Local ID:=1 verbunden ist. Da es sich um ein Datentelegramm handelt, ist die R_ID:=2.	

Schritt	Erklärung	Diagramm
3	Der BSEND-Auftrag wurde abgeschlossen; entweder ist das Error oder das Done-Bit gesetzt. Nun werden die verschickten Daten analysiert. Dabei wird die Station B (Local ID:=2)als Quellstation identifiziert und ein Quittungstelegramm wird erzeugt. Da die Quellinformation:=2 ist (denn die Quellstation hat die Local_ID:=2), wird das erzeugte Quittungstelegramm über den Buffermanager in den Ringpuffer 2 eingetragen. Die Sendeeinheit 1 wird sodann beendet und ist frei für den nächsten Sendevorgang.	
4	Die Sendeeinheit 2, die mit dem Ringpuffer 2 assoziiert ist, liest das Quittungstelegramm aus ihrem Ringpuffer aus und sendet es an ihr Ziel. Da die Sendeeinheit 2 mit der Station B fest verschaltet ist, wird das Quittungstelegramm an Station B geschickt. Da es sich bei dem Telegramm um ein Quittungstelegramm handelt, ist die R_ID:=3.	
5	Da die Sendeeinheit 2 festgestellt hat, dass sie eine Quittung und keine Daten gesendet hat, ist ihr Auftrag mit dem Abschluss des BSEND-Auftrages beendet. Sie ist frei für den nächsten Sendeauftrag.	

Folgendes Zustandsdiagramm zeigt die Abläufe **einer** Sendeinheit anhand eines Zustandsdiagramms:

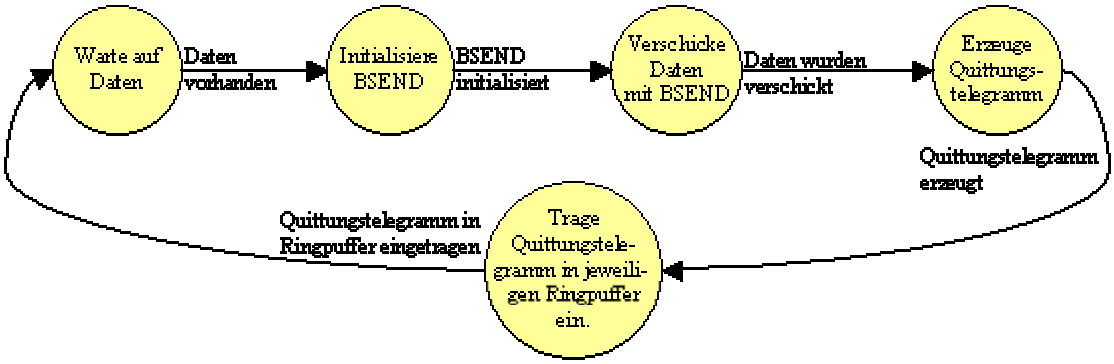


Bild 4-37

Anwenderschnittstelle

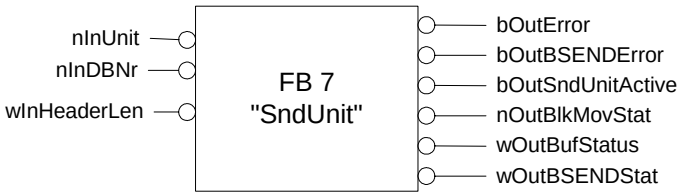


Bild 4-38

Tabelle 4-29

Name	Typ	Erklärung
nInUnit	Int	Nummer der Sendeinheit und damit auch Local ID des jeweiligen BSEND-Aufrufs
nInDBNr	Int	Nummer des Sendefachs
wInHeaderLen	Word	Länge der Kopfdaten
bOutError	Bool	Fehler-Bit
bOutBSENDError	Bool	Fehler-Bit des BSEND-Bausteins
bOutSndUnitActive	Bool	Zeigt an, ob die Sendeeinheit aktiv ist
nOutBlkMovStat	Int	Status des BlkMov-Aufrufs
wOutBufStatus	Word	Status des Ringpuffers
wOutBSENDStat	Word	Status des BSEND-Bausteins

Teil B: Installation der Beispielapplikation

Ziel Teil B

Der Teil B dieses Dokuments soll dem Leser

- die Installation des Beispiels mit allen HW-/SW-Komponenten erklären
- notwendige Projektierungsschritte erläutern
- die Bedienung der Applikation zeigen

Inhalt Teil B

5 Installation der Hard- und Software

Einleitung

In diesem Kapitel beschreiben wir, wie Sie die Applikation in Betrieb nehmen.

Inhalt Kapitel Installation der Hard- und Software

Tabelle 5-1

Kap.-Nr.	Kapitelüberschrift	Hier erfahren Sie...
5.1	Hardwareaufbau	... wie Sie die Hardware aufbauen müssen.
5.2	Installation der Software	... wie Sie die Software in Betrieb nehmen.
5.3	Projektierungen der Applikation	... wie Sie die in der Applikation vorhandenen Projektierungen durchführen können..

5.1 Hardwareaufbau

Die Komponenten des Hardwareaufbaus entnehmen Sie bitte dem Kapitel 2.1 „Übersicht zur Gesamtlösung“.

Gehen Sie für die Hardwareinstallation nach folgender Tabelle vor:

! **Hinweis**

Schalten Sie die Spannungsversorgung erst nach dem letzten Schritt zu.

Tabelle 5-2

Schritt	Fokus	Aktion
1	Station A	Bauen Sie die Station A analog zur Abbildung in Kap. 2.1 „Darstellung der beteiligten Komponenten“ auf.
2	Station B	Bauen Sie die Station B analog zur Abbildung in Kap. 2.1 „Darstellung der beteiligten Komponenten“ auf.
3	Gateway	Bauen Sie die Gateway-Station analog zur Abbildung in Kap. 2.1 „Darstellung der beteiligten Komponenten“ auf.
4	Industrial Ethernet	Verbinden Sie die Industrial Ethernet CPs der Station A und der Gateway-Station.
5	Profibus	Verbinden Sie die Profibus CPs der Station B und der Gateway-Station
6	MPI	Verbinden Sie die integrierten MPI-Schnittstellen in den CPUs mit einem (mehreren) MPI-Kabeln. Verbinden Sie die PG-Schnittstelle mit einem Stecker in einer der 3 Stationen.

5.2 Installation der Software

STEP7

Auf die Beschreibung der Installation von STEP7 wird an dieser Stelle verzichtet. Die Installation findet in gewohnter Windows-Umgebung statt und ist selbsterklärend.

Installation des STEP7-Projektes

Falls Sie das Projekt an einem vorhandenen Bus betreiben wollen, müssen Sie die vom Projekt belegten Adressen beachten:

Tabelle 5-3

Station	Baugruppe	MPI-Adresse	Profibus-Adresse	IP-Adresse
Station A	CPU 416-2DP	2	2 (nicht relevant)	-
	CP 443-1	-	-	140.80.0.1
Station B	CPU 416-2DP	3	2 (nicht relevant)	-
	CP 443-5 Ext	-	6	-
Gateway	CPU 315-2DP	4	2 (nicht relevant)	-
	CP 342-5	5	5	-
	CP 343-1	6	-	140.80.0.2
PG	CP 1613	7	-	-

Gehen Sie nun wie folgt vor:

Tabelle 5-4

Schritt-Nr.	Aktion	Hinweis / Erklärung
1	Öffnen Sie den Step 7 Manager.	
2	Dearchivieren Sie das Projekt über das Menü Datei > Dearchivieren...	Suchen Sie das Projekt **20983154_S7_Data_Gateway_CODE_v10.zip** mittels der Browserfunktion und bestätigen Sie mit **OK**.

Schritt-Nr.	Aktion	Hinweis / Erklärung
3	Wählen Sie ein Verzeichnis, in das die Projektdateien dearchiviert werden sollen.	Am Ende der Dearchivierung werden Sie gefragt, ob Sie das Projekt mit Step 7 öffnen wollen, bestätigen Sie diese Frage mit **Ja**.
4	Nach dem Öffnen des Projektes öffnen Sie den Projektbaum des Projekts.	
5	Klicken Sie im Projektbaum auf `SIMATIC 400 - Station A` und laden Sie mit `Zielsystem -> Laden` das Programm in die `Station A`.	Achten Sie gegebenenfalls darauf, dass die richtige MPI-Adresse (2) ausgewählt ist.
6	Klicken Sie im Projektbaum auf `SIMATIC 400 - Station B` und laden Sie mit `Zielsystem -> Laden` das Programm in das `Station B`.	Achten Sie gegebenenfalls darauf, dass die richtige MPI-Adresse (3) ausgewählt ist.
7	Klicken Sie im Projektbaum auf `SIMATIC 300 - GateWay` und laden Sie mit `Zielsystem -> Laden` das Programm in das `Gateway`.	Achten Sie gegebenenfalls darauf, dass die richtige MPI-Adresse (4) ausgewählt ist.

Hinweis

Wenn STEP 7 eine Station nicht laden kann, kann es sein, dass die MPI-Adressen nicht richtig konfiguriert sind. Lösen Sie PROFIBUS-, Ethernet- und MPI-Kabel von den Stationen und stecken Sie das MPI-Kabel des PGs in die MPI-Buchse der zu ladenden CPU. Führen Sie anschließend den Ladevorgang für diese Station nochmals durch. Verbinden Sie dann wieder die Stationen mit den Kabeln.

5.3 Projektierungen der Applikation

Hinweis

Die komplette Projektierung ist bereits im Projekt enthalten, die Applikation ist sofort ablauffähig.

Wenn Sie bereits mit der Vorgehensweise bei der Projektierung von S7-Verbindungen sowie der Uhrzeitsynchronisation zwischen mehreren Stationen vertraut sind, können Sie dieses Kapitel überspringen.

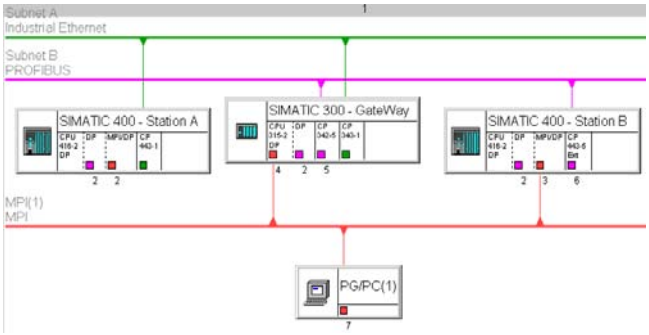
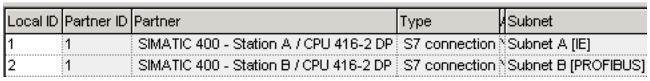
5.3.1 Projektierung von S7-Verbindungen

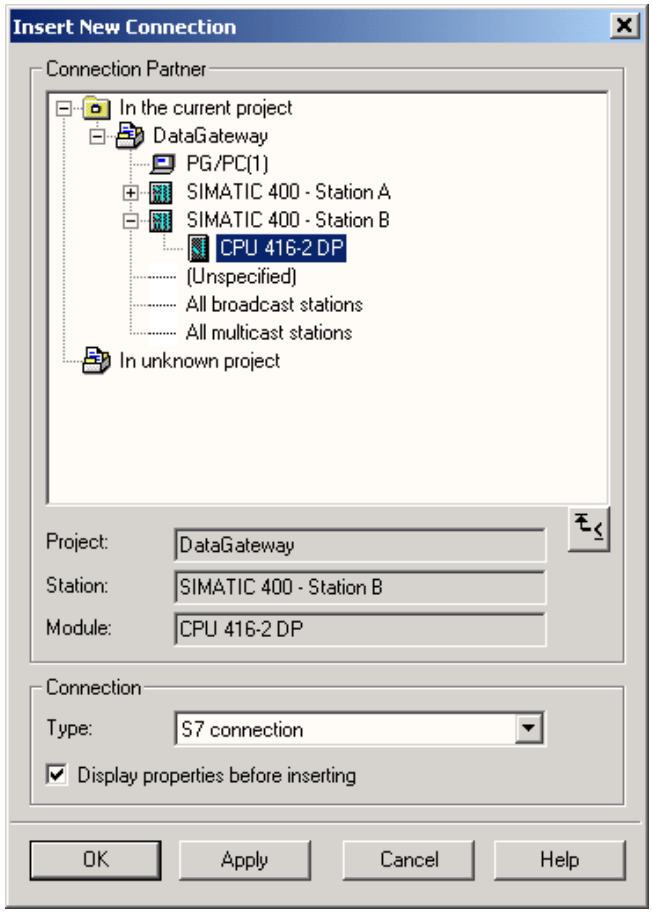
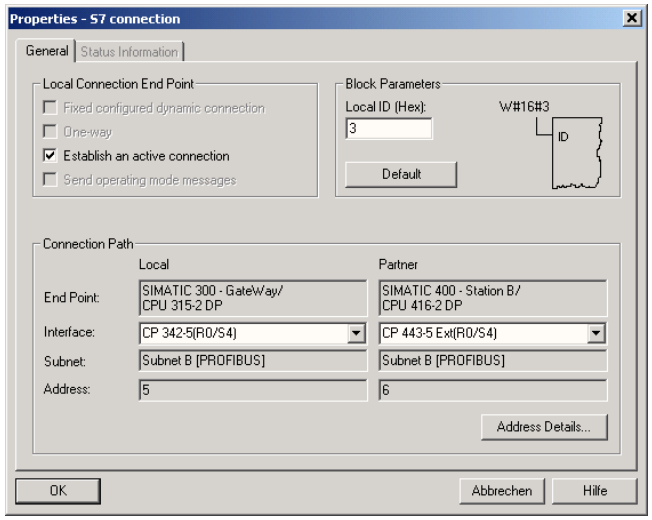
Allgemeines zu S7-Verbindungen

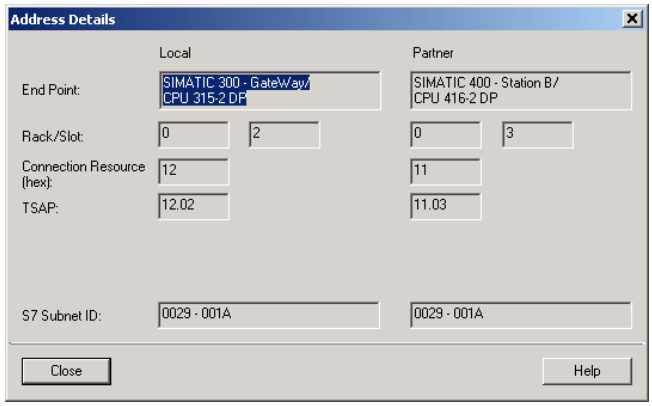
S7-Verbindungen beschreiben Kommunikationswege zwischen SIMATIC-Kommunikationskomponenten. Eine S7-Verbindung verwendet hier das S7-Protokoll. Als Kommunkationsdienst wird hier die S7-Kommunkation mit BSEND/BRCV verwendet.

Um S7-Verbindungen zu projektieren, müssen Sie folgendermaßen vorgehen:

Tabelle 5-5

Nr.	Aktion	Hinweis/Bild
1	Öffnen Sie die NetPro, indem Sie im Simatic Manager  anwählen.	
2	In der Netzansicht sind die drei Netzwerke in Rot (MPI), Purpur (Profibus) und Grün (Industrial Ethernet) sichtbar. Die im Projekt verbundenen Steuerungen sind mit ihren Schnittstellen an die Netzwerke geknüpft. Durch Selektieren einer CPU, in diesem Fall die CPU 315-2DP, wird die Verbindungstabelle in der unteren Hälfte des Fensters sichtbar.	 In der Verbindungstabelle sind alle für die jeweilige Steuerung projektierten Verbindungen aufgelistet.
3	Um den prinzipiellen Verlauf einer solchen Verbindungsprojektierung aufzuzeigen, wird am Beispiel der ersten Verbindung diese erläutert. Durch Betätigen von Verbindung einfügen  wird der Verbindungsdialog gestartet.	Alternativ kann hier auch der Menüpunkt Einfügen > neue Verbindung oder die Tastenkombination CTRL + N genutzt werden.

4	Im oberen Fensterteil sind alle im Projekt enthaltenen Kommunikationspartner sichtbar. Als Option sind auch unspezifizierte Verbindungspartner oder Referenzen aus anderen Projekten möglich. Im unteren Teil des Fensters ist der Verbindungstyp spezifiziert, hier eine S7-Verbindung. Durch Bestätigen mit OK oder Übernehmen öffnet sich das Eigenschaftsfenster der Verbindung.	
5	Das Eigenschaftsfenster teilt sich in 3 Bereiche. Der erste Bereich ist der lokale Verbindungsendpunkt. Hier wird der Verbindungstyp, ein- oder zweiseitige Verbindung sowie das Verbindungshandling, aktiver oder passiver Verbindungsaufbau, definiert. Der zweite Bereich ist der Bereich Bausteinparameter. Hier wird die am SFB bzw. FB anzugebende Verbindungs-ID dargestellt. Der dritte Bereich ist der Verbindungsweg. Hier können mehrere Verbindungswege möglich sein. Dadurch kann eine Auswahl getroffen werden, über welches Netzwerk die Daten übertragen werden sollen. Mittels des Buttons Adressdetails können weitere Informationen über die Verbindung angezeigt werden.	 In diesem Fall wird eine beidseitig projektierte Verbindung verwendet. Der aktive Verbindungsaufbau ist auf einer Seite zwingend erforderlich, aber frei vergebbar. Bitte lassen Sie die Lokale ID in der Vorgabe unverändert, nur bei Verbindungen über Projektgrenzen hinweg sind Einstellungen an dieser Stelle notwendig. Über die Einstellung der Schnittstelle können Sie den Übertragungsweg ändern.

		Achtung: Die Local IDs im Gateway müssen aufsteigend und lückenlos sein (z.B. 1, 2, 3, ..., 8)
	Die Adressdetails lassen sich in drei logische Teile aufteilen. • Der Verbindungsendpunkt lokal und remote. Hier werden die adressierten Baugruppen, welchen die S7-Verbindung zugeordnet ist, angegeben. • Mit der Ressourcenadresse (diese enthält die Rack / Slot Adresse sowie die Verbindungsressource) und den daraus ermittelten TSAP wird nochmals eine eindeutige Referenzierung auf die Lokalität des Verbindungsendpunktes durchgeführt (s. 4.4.2 „Projektierungsmodell des S7-Protokolls“). • Die S7 Subnetz ID stellt eine von Step 7 vergebene ID für das genutzte Netzwerk dar.	 Änderungen sind in diesem Fenster nur möglich, wenn Sie eine Verbindung zu einem unspezifizierten Partner projektiert haben.

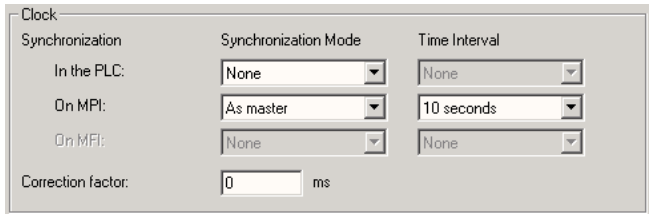
5.3.2 Projektierung einer Uhrzeitsynchronisation

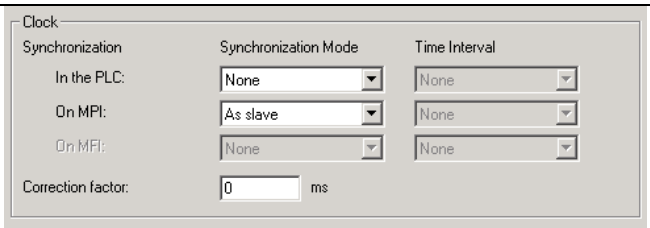
Allgemeines zur Uhrzeitsynchronisation

Um sicherzustellen, dass die Uhrzeit in allen Modulen eines Netzwerks gleich ist, gibt es die Möglichkeit, das Gateway des Netzwerks als Uhrzeit-Master zu konfigurieren, so dass die Zeit der Teilnehmer in bestimmten Zeitabständen an die Zeit des Gateways angeglichen wird.

Vorgehensweise bei der Projektierung

Tabelle 5-6

Nr.	Aktion	Hinweis/Bild
1	Öffnen Sie die HW-Konfiguration der Gateway-Station	
2	Öffnen Sie über das Kontextmenü der CPU den Dialog „Objekteigenschaften“, dann die Lasche „Diagnose/Uhrzeit“.	
3	Im unteren Teil des Dialogs konfigurieren Sie die CPU wie im Bild gezeigt als Uhrzeit-Master.	Die Aktualisierung der Zeit der Teilnehmer findet via MPI im Abstand von 10 Sekunden statt.
4	Quittieren Sie den Dialog mit OK und speichern und übersetzen Sie die HW-Konfig.	

5	Öffnen Sie die HW-Konfiguration der Station A.	
6	Öffnen Sie über das Kontextmenü der CPU den Dialog „Objekteigenschaften“, dann die Lasche „Diagnose/Uhrzeit“.	
7	Im unteren Teil des Dialogs konfigurieren Sie wie im Bild gezeigt die CPU als Uhrzeit-Slave.	Die Aktualisierung der Zeit des Teilnehmers findet via MPI in dem Abstand statt, der auf der Master-Seite konfiguriert wurde.
8	Quittieren Sie den Dialog mit OK und speichern und übersetzen Sie die HW-Konfig.	
9	Für die Station B gehen Sie die Schritte 5 bis 8 analog vor.	

6 Bedienen und Beobachten der Applikation

Einleitung

In diesem Kapitel werden die Schritte erklärt, die Sie mit der Bedienung der Applikation vertraut machen sollen.

Inhalt Kapitel Bedienen und Beobachten der Applikation

Tabelle 6-1

Kap.-Nr.	Kapitelüberschrift	Hier erfahren Sie...
6.1	Funktionen der Anwenderschnittstelle	... wie die Anwenderschnittstelle der Applikation aufgebaut ist.
6.2	Steuerung der Datenübertragung	... wie Sie die Datenübertragung steuern können.
6.3	Simulation von Übertragungsfehlern	... wie Sie Übertragungsfehler simulieren können.
6.4	Systemparameter / Messungen am System	... welche Messungen in der Applikation durchgeführt werden können.
6.5	Diagnosemöglichkeiten	... welche Diagnosemöglichkeiten in der Applikation bestehen.

Voraussetzungen

Die Applikation kann in Betrieb genommen werden, wenn folgende Voraussetzungen erfüllt sind:

- Die Bedingungen aus dem Kapitel 5 „Installation der Hard- und Software“ sind erfüllt
- Alle Geräte sind eingeschaltet
- Das STEP7-Projekt befindet sich online auf der S7-CPU

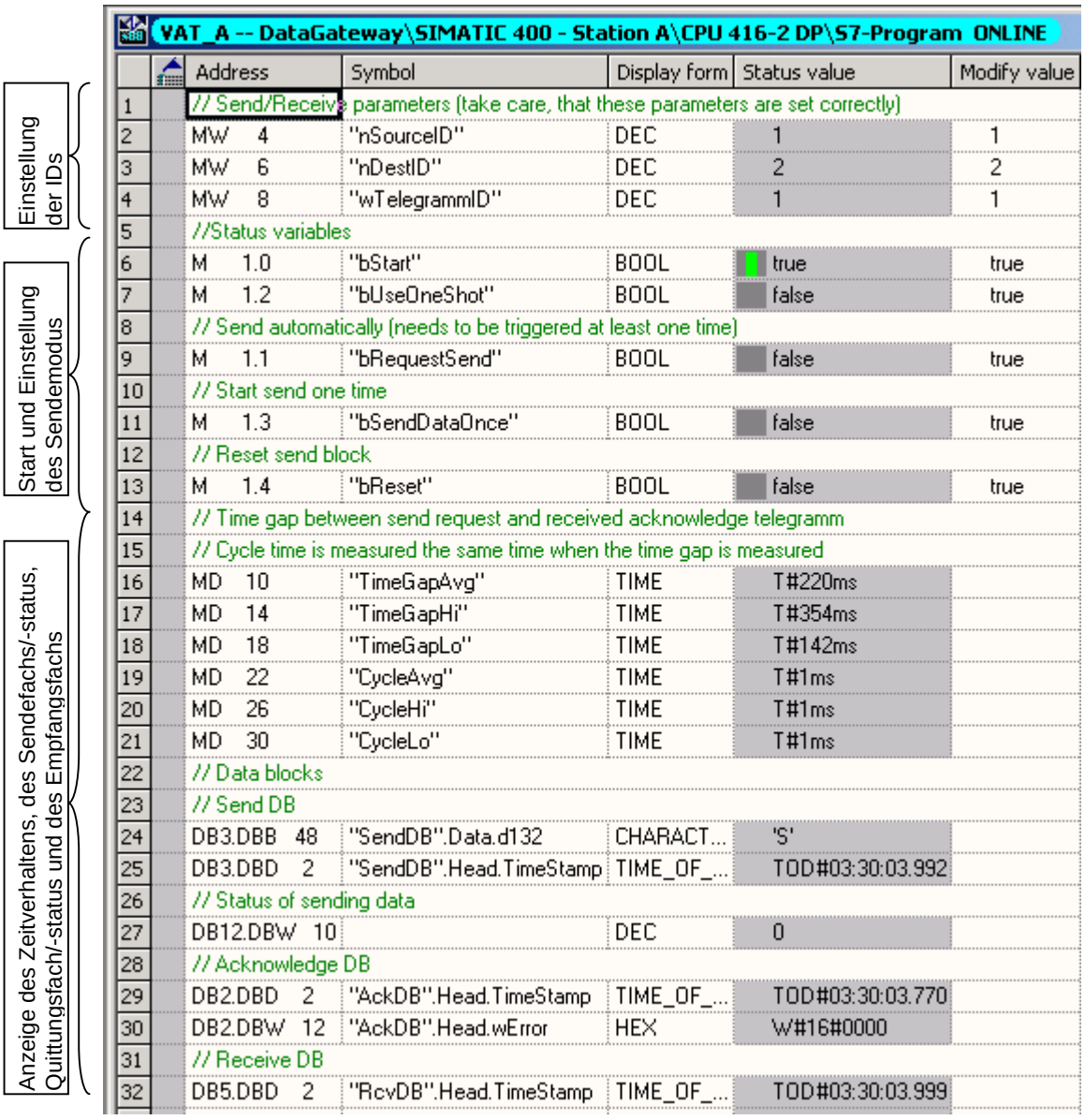
6.1 Funktionen der Anwenderschnittstelle

Die Anwenderschnittstelle ist in Form von drei Variablentabellen (VAT) in Step7 ausgeführt. Diese stellen sich wie folgt dar:

- VAT_A/B: Anzeige/Steuerung der wichtigsten Parameter der Station A/B.
  Die Variablentabellen für A und B sind bis auf die eingestellten Steuerwerte identisch.
- VAT_C: Anzeige/Steuerung der wichtigsten Parameter des Gateways.

Variablentabelle VAT_A/B der Station A/B

Das folgende Bild zeigt die Variablentabelle für die Station A. In der nachfolgenden Tabelle werden die einzelnen Einträge erläutert.



VAT_A -- DataGateway\SIMATIC 400 - Station A\CPU 416-2 DP\S7-Program ONLINE

	Address	Symbol	Display form	Status value	Modify value
1	// Send/Receive parameters (take care, that these parameters are set correctly)				
2	MW 4	"nSourceID"	DEC	1	1
3	MW 6	"nDestID"	DEC	2	2
4	MW 8	"wTelegrammID"	DEC	1	1
5	//Status variables				
6	M 1.0	"bStart"	BOOL	true	true
7	M 1.2	"bUseOneShot"	BOOL	false	true
8	// Send automatically (needs to be triggered at least one time)				
9	M 1.1	"bRequestSend"	BOOL	false	true
10	// Start send one time				
11	M 1.3	"bSendDataOnce"	BOOL	false	true
12	// Reset send block				
13	M 1.4	"bReset"	BOOL	false	true
14	// Time gap between send request and received acknowledge telegramm				
15	// Cycle time is measured the same time when the time gap is measured				
16	MD 10	"TimeGapAvg"	TIME	T#220ms	
17	MD 14	"TimeGapHi"	TIME	T#354ms	
18	MD 18	"TimeGapLo"	TIME	T#142ms	
19	MD 22	"CycleAvg"	TIME	T#1ms	
20	MD 26	"CycleHi"	TIME	T#1ms	
21	MD 30	"CycleLo"	TIME	T#1ms	
22	// Data blocks				
23	// Send DB				
24	DB3.DBB 48	"SendDB".Data.d132	CHARACT...	'S'	
25	DB3.DBD 2	"SendDB".Head.TimeStamp	TIME_OF_...	TOD#03:30:03.992	
26	// Status of sending data				
27	DB12.DBW 10		DEC	0	
28	// Acknowledge DB				
29	DB2.DBD 2	"AckDB".Head.TimeStamp	TIME_OF_...	TOD#03:30:03.770	
30	DB2.DBW 12	"AckDB".Head.wError	HEX	W#16#0000	
31	// Receive DB				
32	DB5.DBD 2	"RcvDB".Head.TimeStamp	TIME_OF_...	TOD#03:30:03.999	

Bild 6-1

Tabelle 6-2

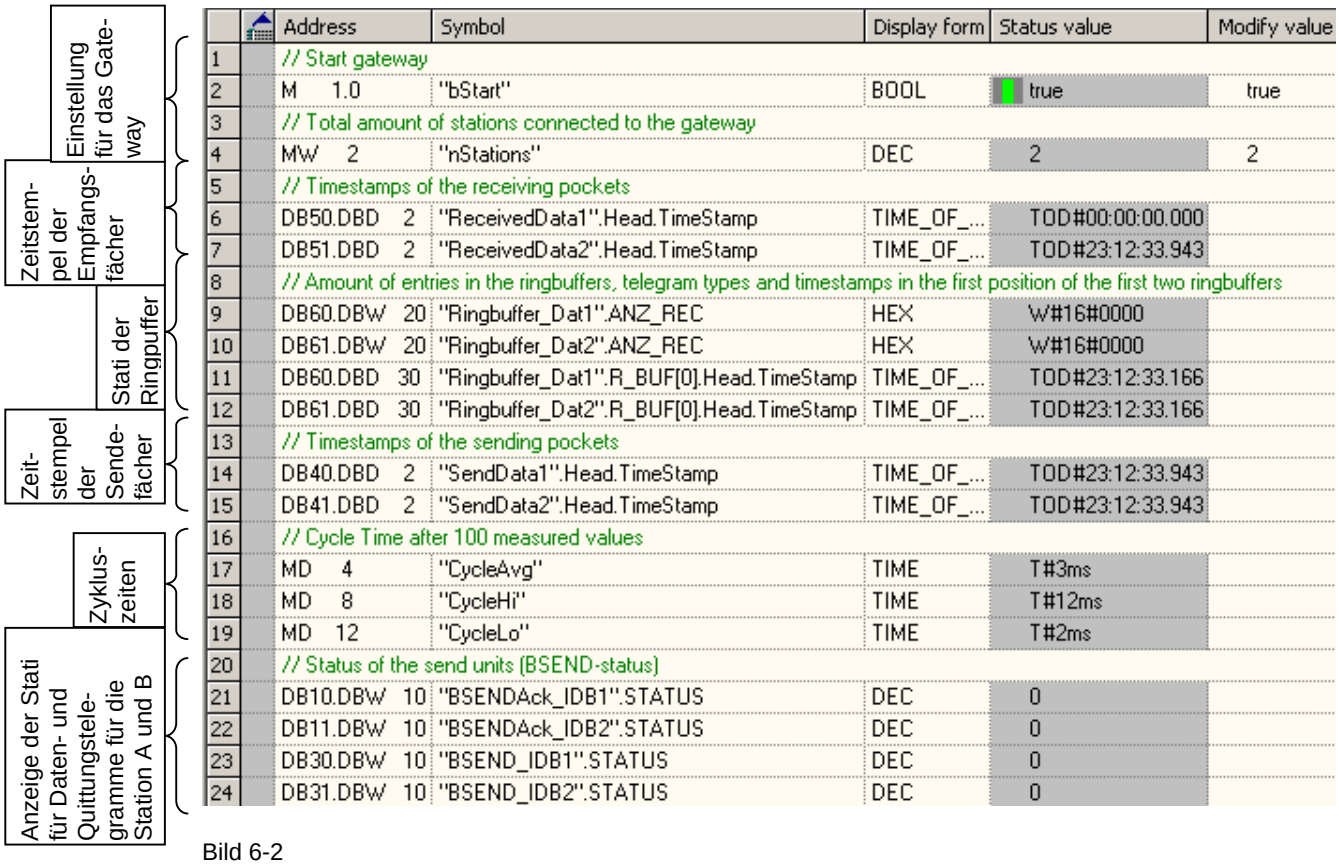
Bereich	Operand	Symbol	Erklärung
Einstellung der IDs	MW 4	nSourceID	Bezeichnet die Local ID der Quellstation aus der Sicht des Gateways. Station A ist also am Gateway über Local ID :=1 verbunden.
	MW 6	nDestID	Bezeichnet die Local ID der Zielstation aus der Sicht des Gateways. Station B ist also am Gateway über Local ID:=2 verbunden.
	MW 8	wTelegrammID	Legt die Telegramm-ID fest, die in der vorliegenden Applikation nicht ausgewertet wird.
Start und Einstellung	M1.0	bStart	Falls TRUE, so ist die Applikation in dieser Station aktiv.

Bereich	Operand	Symbol	Erklärung
des Sendemodus	M1.2	bUseOneShot	Steuert, ob das Senden der Daten automatisch (FALSE) oder manuell getriggert werden soll (TRUE).
	M1.1	bRequestSend	Muss im Falle des automatischen Sendens mindestens einmal auf TRUE gesetzt werden (positive Flanke). Der Merker wird von der Applikation automatisch wieder zurückgesetzt.
	M1.3	bSendDataOnce	Im Falle des manuellen Sendens wird mit einer positiven Flanke (FALSE → TRUE) genau ein Datenpaket verschickt. Der Merker wird von der Applikation automatisch wieder zurückgesetzt.
	M1.4	bReset	Mit diesem Parameter lässt sich der Sendebaustein zurücksetzen. Dies ist z.B. dann notwendig, wenn die falschen „Quell-/Zielinformationen“ angegeben wurden und der Sendebaustein aufgrund einer fehlenden Quittung nicht mehr in den freien Sendezustand gelangen kann. Sobald der Sendebaustein eine steigende Flanke bekommt und bReset := TRUE ist, wird er zurückgesetzt.
Anzeige des Zeitverhaltens, des Sendefachs/-status, Quittungssfach/-status und des Empfangsfachs	MD10	TimeGapAvg	Durchschnittliche Telegrammlaufzeit *
	MD14	TimeGapHi	Höchste Telegrammlaufzeit *
	MD18	TimeGapLo	Niedrigste Telegrammlaufzeit *
	MD22	CycleAvg	Durchschnittliche Zykluszeit *
	MD26	CycleHi	Höchste Zykluszeit *
	MD30	CycleLo	Niedrigste Zykluszeit *
	DB3.DBB48	SendDB.Data.d132	Zeigt das aktuell verschickte Nutzdatum im Nutzdatenbyte 48
	DB3.DBD2	SendDB.Head.TimeStamp	Zeigt den Zeitstempel der aktuell zu sendenden Daten
	DB12.DBW10	-	Zeigt den Status des Sendens (25 bedeutet, dass der Auftrag läuft; s. STEP7-Hilfe zum Baustein BSEND)
	DB2.DBB2	AckDB.Head.TimeStamp	Zeigt den Zeitstempel des Telegramms, welches aktuell quittiert wurde. Der hier zu sehende Zeitstempel ist im Normalfall immer der Zeitstempel des zuletzt geschickten Datentelegramms.
	DB2.DBD12	AckDB.Head.wError	Fehlerstatus, der im Gateway auftrat. Bei 0 trat kein Fehler auf. (Die hier aufgeführten Stati entsprechen den Fehlerstati des BSEND-Bausteins; s. STEP7-Hilfe)
	DB5.DBD2	RcvDB.Head.TimeStamp	Zeigt den Zeitstempel des zuletzt empfangenen Datentelegramms.

(*) Gemessen bei 100 Werten

Variablentabelle der Gateway-CPU

Das folgende Bild zeigt die Variablentabelle für das Gateway. In der nachfolgenden Tabelle werden die einzelnen Einträge erläutert.



	Address	Symbol	Display form	Status value	Modify value
1	// Start gateway				
2	M 1.0	"bStart"	BOOL	true	true
3	// Total amount of stations connected to the gateway				
4	MW 2	"nStations"	DEC	2	2
5	// Timestamps of the receiving pockets				
6	DB50.DBD 2	"ReceivedData1".Head.TimeStamp	TIME_OF_...	TOD#00:00:00.000	
7	DB51.DBD 2	"ReceivedData2".Head.TimeStamp	TIME_OF_...	TOD#23:12:33.943	
8	// Amount of entries in the ringbuffers, telegram types and timestamps in the first position of the first two ringbuffers				
9	DB60.DBW 20	"Ringbuffer_Dat1".ANZ_REC	HEX	W#16#0000	
10	DB61.DBW 20	"Ringbuffer_Dat2".ANZ_REC	HEX	W#16#0000	
11	DB60.DBD 30	"Ringbuffer_Dat1".R_BUF[0].Head.TimeStamp	TIME_OF_...	TOD#23:12:33.166	
12	DB61.DBD 30	"Ringbuffer_Dat2".R_BUF[0].Head.TimeStamp	TIME_OF_...	TOD#23:12:33.166	
13	// Timestamps of the sending pockets				
14	DB40.DBD 2	"SendData1".Head.TimeStamp	TIME_OF_...	TOD#23:12:33.943	
15	DB41.DBD 2	"SendData2".Head.TimeStamp	TIME_OF_...	TOD#23:12:33.943	
16	// Cycle Time after 100 measured values				
17	MD 4	"CycleAvg"	TIME	T#3ms	
18	MD 8	"CycleHi"	TIME	T#12ms	
19	MD 12	"CycleLo"	TIME	T#2ms	
20	// Status of the send units (BSEND-status)				
21	DB10.DBW 10	"BSENDAck_IDB1".STATUS	DEC	0	
22	DB11.DBW 10	"BSENDAck_IDB2".STATUS	DEC	0	
23	DB30.DBW 10	"BSEND_IDB1".STATUS	DEC	0	
24	DB31.DBW 10	"BSEND_IDB2".STATUS	DEC	0	

Bild 6-2

Tabelle 6-3

Bereich	Operand	Symbol	Erklärung
Einstellung für das Gateway	M1.0	bStart	Startet die Gateway-Applikation, falls TRUE.
	MW2	nStations	Gibt die Gesamtzahl der Stationen an, die das Gateway verwalten soll. **Achtung:** Dies entspricht gleichzeitig der Anzahl an projektierten Verbindungen und der größten projektierten Local ID in der Gateway-CPU!
Zeitstempel der Empfangsfächer	DB50.DBD2	ReceivedData1. Head.TimeStamp	Zeigt den zuletzt im Empfangsfach 1 empfangenen Zeitstempel eines Datentelegramms an.
	DB51.DBD2	ReceivedData2. Head.TimeStamp	Zeigt den zuletzt im Empfangsfach 2 empfangenen Zeitstempel eines Datentelegramms an.
Stati der Ringpuffer	DB60.DBW20	Ringbuffer_Dat1. ANZ_REC	Zeigt die Anzahl der Einträge im Ringpuffer der ersten Sendeeinheit an.
	DB61.DBW20	Ringbuffer_Dat2. ANZ_REC	Zeigt die Anzahl der Einträge im Ringpuffer der zweiten Sendeeinheit an.
	DB60.DBD30	Ringbuffer_Dat1. R_BUF[0]. Head.TimeStamp	Zeigt den Zeitstempel an, der im ersten Eintrag des Ringpuffers der ersten Sendeeinheit vorhanden ist. Da der Ringpuffer die Größe 10 hat, zeigt dieser Wert jeden 10. Zeitstempel im Ringpuffer der ersten Sendeeinheit an

Bereich	Operand	Symbol	Erklärung
	DB61.DBD30	Ringbuffer_Dat2. R_BUF[0]. Head.TimeStamp	Zeigt den Zeitstempel an, der im ersten Eintrag des Ringpuffers der zweiten Sendeeinheit vorhanden ist. Da der Ringpuffer die Größe 10 hat, zeigt dieser Wert jeden 10. Zeitstempel im Ringpuffer der zweiten Sendeeinheit an
Zeitstempel der Sendefächer	DB40.DBD2	SendData1. Head.TimeStamp	Zeigt den zuletzt im Sendefach 1 vorhandenen Zeitstempel eines Daten-/Quittungstelegramms an.
	DB41.DBD2	SendData2. Head.TimeStamp	Zeigt den zuletzt im Sendefach 2 vorhandenen Zeitstempel eines Daten-/Quittungstelegramms an.
Zykluszeiten	MD4	CycleAvg	Durchschnittliche Zykluszeit *
	MD8	CycleHi	Höchste Zykluszeit *
	MD12	CycleLo	Niedrigste Zykluszeit *
Anzeige der Stati für Daten- und Quittungstelegramme für die Station A und B	DB10.DBW10	BSENDAck_IDB1. STATUS	Sendestatus der Sendeeinheit 1 für den Fall, dass ein Quittungstelegramm gesendet wird.
	DB11.DBW10	BSENDAck_IDB2. STATUS	Sendestatus der Sendeeinheit 2 für den Fall, dass ein Quittungstelegramm gesendet wird.
	DB30.DBW10	BSEND_IDB1.STATUS	Sendestatus der Sendeeinheit 1 für den Fall, dass ein Datentelegramm gesendet wird.
	DB31.DBW10	BSEND_IDB2.STATUS	Sendestatus der Sendeeinheit 2 für den Fall, dass ein Datentelegramm gesendet wird.

(*) Gemessen bei 100 Werten

6.2 Steuerung der Datenübertragung

Starten der Applikation

Mit nachfolgenden Schritten können Sie den Prozess der Datenübertragung starten.

Tabelle 6-4

Schritt	Aktion	Hinweis / Erklärung
1	Öffnen Sie die Variablentabellen VAT_A, VAT_B und VAT_C.	Ordnen Sie die Fenster am besten so an, dass VAT_A links, VAT_C in der Mitte und VAT_B rechts ist
2	Klicken Sie auf das Brillensymbol  oder im Menü auf `Variable` -> `Beobachten`.	Sie können nun die Variablen online beobachten und steuern.
3	Kontrollieren Sie, ob in der Tabelle VAT_A die folgenden Bedingungen erfüllt ist: • nSourceID = 1: Dies entspricht der Local ID im Gateway zur Quellstation (s. NetPro); hier ist Station A die Quellstation • nDestID = 2: Dies entspricht der Local ID im Gateway zur Zielstation (s. NetPro); hier ist Station B die Zielstation	Wenn die Bedingungen nicht erfüllt sind, steuern Sie diese Variablen auf die angegebenen Werte. Markieren Sie die 3 Zeilen und drücken F9 oder betätigen Sie den Knopf  .

Schritt	Aktion	Hinweis / Erklärung
	• wTelegrammID = 1: Freier Parameter zur Unterscheidung von Telegrammen ➔ Mit diesen Einstellungen werden Telegramme von Station A zu Station B gesendet.	
4	Kontrollieren Sie, ob in der Tabelle VAT_B die folgenden Bedingungen erfüllt sind: • nSourceID = 2: Dies entspricht der Local ID im Gateway zur Quellstation (s. NetPro); hier ist Station B die Quellstation • nDestID = 1: Dies entspricht der Local ID im Gateway zur Zielstation (s. NetPro); hier ist Station A die Zielstation • wTelegrammID = 1: Freier Parameter zur Unterscheidung von Telegrammen ➔ Mit diesen Einstellungen werden Telegramme von Station B zu Station A gesendet.	Beachten Sie den Hinweis in Schritt 3.
5	Kontrollieren Sie, ob in der Tabelle VAT_C die folgenden Bedingung erfüllt ist: • nStations = 2	Beachten Sie den Hinweis in Schritt 3.
6	Kontrollieren Sie, ob in allen Tabellen die folgende Bedingung erfüllt ist: • bStart = TRUE	Beachten Sie den Hinweis in Schritt 3.
7	Folgender Schritt ist in den Variablentabellen VAT_A und VAT_B auszuführen: Wählen Sie, ob das Senden der Daten automatisch oder manuell erfolgen soll. Die Auswahl erfolgt über „bUseOneShot“: • bUseOneShot = FALSE: automatischer Betrieb • bUseOneShot = TRUE: manueller Betrieb Im automatischen Fall fahren Sie mit Schritt 8 fort. Im manuellen Fall fahren mit Schritt 9 fort.	Zum Steuern von Variablen beachten Sie bitte den Hinweis Schritt 3. **Hinweis:** Um Daten bidirektional senden zu können, sollte das Senden in mindestens einer Station automatisch erfolgen.
8	Automatisches Senden von Daten: Steuern Sie den Parameter „bRequestSend“ in der VAT A/B auf TRUE. **Hinweise:** • Dieser Parameter wird von der Applikation wieder zurückgesetzt. • Das automatische Senden wird gestoppt, wenn „bUseOneShot“=TRUE geschaltet wird. Die Stationen A und B senden nun Daten.	Da es im Einzelfall vorkommen kann, dass der Parameter nicht übernommen wird, sollten Sie den Vorgang wiederholen, bis die Station automatisch sendet. Dies können Sie an den sich verändernden Zeitstempeln erkennen.
9	Manuelles Senden von Daten: Steuern Sie den Parameter „bSendDataOnce“ auf TRUE. **Achtung:** Dieser Parameter wird von der Applikation wieder zurückgesetzt.	

Ergebnisse

Hinweis

Jedes Daten-Telegramm enthält einen Zeitstempel, der in der Quellstation gebildet wird. Dieser Zeitstempel wird während der gesamten Lebensdauer eines Telegramms nicht mehr verändert.

Sie sehen nun folgende Ergebnisse:

- Fall1:
  Automatisches Senden von Daten von Station B nach Station A; Station A sendet nicht:
  - VAT_B:
    - Das Sendefach wird schnellstmöglich (Telegrammlaufzeit!) gefüllt.
    - Das Quittungsfach zeigt den Zeitstempel des letzten gesendeten Telegramms
    - Die Telegrammlauf- und Zykluszeiten werden nach 100 gesendeten Telegrammen aktualisiert.
  - VAT_A:
    - Das Empfangsfach enthält den Zeitstempel des zuletzt empfangenen Datentelegramms
  - VAT_C:
    - Das Empfangsfach 2 enthält den Zeitstempel des zuletzt empfangenen Datentelegramms von Station B
    - Das Sendefach 1 enthält den Zeitstempel des zuletzt gesendeten Datentelegramms an Station A
    - Das Sendefach 2 enthält den Zeitstempel des zuletzt gesendeten Quittungstelegramms an Station B
    - Die Zykluszeiten werden in jedem 100-ten Zyklus aktualisiert
- Fall2:
  Manuelles Senden von Daten von Station B nach Station A; Station A sendet nicht (z.B. durch „bUseOneShot“=TRUE und „bSendDataOnce“=FALSE):
  - VAT_A:
    - Das Empfangsfach enthält den Zeitstempel des zuletzt empfangenen Datentelegramms
  - VAT_B:
    - Das Sendefach enthält den Zeitstempel, der bei Setzen von „bUseOneShot“ → TRUE aktuell war.
    - Das Quittungsfach zeigt zeitversetzt den Zeitstempel des gesendeten Telegramms
    - Die Telegrammlauf- und Zykluszeiten werden nur nach dem 100-ten gesendeten Telegramm aktualisiert.
  - VAT_C:

- Das Empfangsfach 2 enthält den Zeitstempel des empfangenen Datentelegramms von Station B
- Das Sendefach 1 enthält den Zeitstempel des gesendeten Datentelegramms an Station A
- Das Sendefach 2 enthält den Zeitstempel des gesendeten Quittungstelegramms an Station B
- Die Zykluszeiten werden in jedem 100-ten Zyklus aktualisiert

Diagnose

Zur Diagnose beobachten Sie bitte die oben erklärten Stati der BSEND- und BRCV-Bausteine und beachten Sie das Kap. 6.5.

6.3 Simulation von Übertragungsfehlern

Vorraussetzung

Es ist eine bidirektionale Datenübertragung eingestellt (s. Kap. 6.2).

Simulation

Mit nachfolgenden Schritten können Sie Übertragungsfehler simulieren.

Tabelle 6-5

Schritt	Aktion	Reaktionen des Systems
Fehler: Profibus-Kabel gezogen		

Schritt	Aktion	Reaktionen des Systems
1	Ziehen Sie das Profibuskabel vom CP der Station B.	• Das Gateway verschickt das letzte Datentelegramm von Station B nach Station A. → Beachten Sie den Zeitstempel des Empfangsfachs von Station A. • Der Sendebaustein in Station B liefert Fehler beim BSEND bzw. beim BRCV-Aufruf. → Es wird kein Telegramm generiert. • Station B erhält vom Gateway keinerlei Quittungstelegramme, da der BSEND-Aufruf des Gateways fehl schlägt. → Beachten Sie den Zeitstempel des Quittungfachs der Station B. • Aufgrund der Profibus-Spezifikation bemerkt das Gateway nach einer definierten Zeit, dass die Station B nicht mehr erreicht werden kann. ➔ - Das Gateway kann kein Datentelegramm mehr von Station A nach Station B schicken. - In den Quittungstelegrammen, die das Gateway zu Station A schickt, steht die Information, dass das Datentelegramm von Station A nach Station B nicht gesendet werden konnte. Der jeweilige Fehler und dessen Status ist dem Quittungstelegramm zu entnehmen. („AckDB.Head.wError“). **Achtung:** Alle Datentelegramme von A nach B und von B nach A werden verworfen. Es ist keine Wiederholroutine implementiert! Diese sollte in den Quell-/Zielstationen implementiert werden. (s. Kap. 8.4.1) **Hinweis:** Beachten Sie die jeweiligen Stati der einzelnen BSEND/BRCV-Bausteine.
2	Stecken Sie das Profibuskabel wieder an den CP der Station B.	Das Gateway bemerkt, dass Station B wieder erreicht werden kann. ➔ Der Datenverkehr wird wieder aufgenommen.
Fehler: Industrial Ethernet-Kabel gezogen		
3	Ziehen Sie das Industrial Ethernet-Kabel vom CP der Station A.	• Das Gateway verschickt das letzte Datentelegramm von Station A nach Station B. → Beachten Sie den Zeitstempel des Empfangfachs von Station B. • Der Sendebaustein in Station A liefert nach einiger Zeit den BSEND-Status 1. Der Sendebaustein in Station A ist blockiert, solange das IE-Kabel gezogen ist. → Beachten Sie den Zeitstempel des Sendefachs im automatischen Sendebetrieb: Der Zeitstempel ändert sich nicht mehr. **Achtung:** In der Station A ist keine Routine implementiert, die den BSEND-Versuch abbricht. Dies kann nachträglich implementiert werden. (s. Kap. 8.4.1)

Schritt	Aktion	Reaktionen des Systems
		• Station A erhält vom Gateway keinerlei Quittungstelegramme. → Beachten Sie den Zeitstempel des Quittungsfachs der Station A. • Aufgrund der IE-Spezifikation muss das Gateway nicht unbedingt bemerken, dass die Station A nicht mehr erreicht werden kann. ➔ - Das Gateway bemerkt einen Kommunikationsfehler beim Versuch, eine Quittung an Station A zu senden (DB10.DBW10 im Gateway: Status 1). - Das Gateway schickt kein Quittungstelegramm zu Station B, da das Datentelegramm von B nach A noch nicht gesendet werden konnte (DB30.DBW10 im Gateway: Status 25). Da der Sendebaustein in Station B auf das Quittungstelegramm vom Gateway wartet, ist der Sendebaustein blockiert. Station B verschickt also keinerlei Daten mehr. **Achtung:** Die Kommunikation ist blockiert. Es findet kein Datenverkehr mehr statt. Es ist keine TimeOutroutine implementiert! Diese sollte in den Quell-/Zielstationen implementiert werden. (s. Kap. 8.4.1) **Hinweis:** Beachten Sie die jeweiligen Stati der einzelnen BSEND/BRCV-Bausteine.
4	Stecken Sie das Industrial Ethernet Kabel wieder an den CP	Das Gateway bemerkt nach einer gewissen Zeit, dass Station A wieder erreicht werden kann. ➔ Der Datenverkehr wird wieder aufgenommen.
Fehlerhafte Parametrierung oder fehlerhafte Quell-/Zielinformationen		
5	Folgende Schritte beziehen sich immer auf VAT_B. Setzen Sie die Variablen in gegebener Reihenfolge auf folgende Werte: • `bUseOneShot` = TRUE • `bSendDataOnce` =TRUE	Station B sendet über das Gateway ein Datenpaket zu Station A. Die Zeitstempel im Sende- und Quittungsfach in Station B und im Empfangsfach in Station A sind nach kurzer Zeit identisch.
6	Steuern Sie die Variable nDestID auf 0.	
7	Schicken Sie ein Datenpaket mit `bSendDataOnce` = TRUE	Station B sendet über das Gateway ein Datenpaket. Dieses kommt in Station A jedoch nicht an (→ beachten Sie die Zeitstempel). Der Sendebaustein in Station B wartet nun auf das Quittungstelegramm, wonach das Gateway das Datenpaket erhalten hat, es aber nicht zuordnen konnte.
8	Setzen Sie `nDestID` = 1.	
9	Versuchen Sie, noch ein Datenpaket mit `bSendDataOnce`= TRUE zu versenden.	Der Zeitstempel im Sendefach ändert sich **nicht**. Der Sendebaustein ist blockiert.

Schritt	Aktion	Reaktionen des Systems
		Hinweis: Die Variable `bRequestSend` wird auf TRUE gesetzt. Diese Variable wird später wieder automatisch zurückgesetzt.
10	Setzen Sie `bReset` = TRUE.	`BRequestSend` wird auf FALSE gesetzt.
11	Setzen Sie `bReset` = FALSE	
12	Setzen Sie `bSendDataOnce` = TRUE	Es wird ein Datentelegramm von Station B zu Station A geschickt (→ beachten Sie die Zeitstempel).

Hinweis

Im beschriebenen Fall wurde das Funktionselement „Sendebaustein“ (FB25) zurückgesetzt.

Sollte jedoch der BSEND-Baustein im FB25 blockiert oder in einem inkonsistenten Zustand sein, so muss nach Schritt 10 die Variable „bSendDataOnce“ auf TRUE gesetzt werden, da der BSEND-Baustein nur dadurch eine positive Flanke erhält.

6.4 Messungen am System

Wie in Kap. 1 bereits kurz erläutert, werden die Telegrammlaufzeit in den Quell- und Zielstationen sowie die Zykluszeit gemessen.

Zur Messdurchführung siehe Kap. 7.1.3 bzw. 4.6.7.

Die Applikation lässt es über die Variablentabelle VAT_C zu, die Telegramm- und die Zykluszeit zu beeinflussen. Dabei wird zugleich die Auswirkung der Zykluszeit des Gateways auf die Telegrammlaufzeit deutlich.

Folgende Tabelle beschreibt die dafür notwendigen Schritte:

Tabelle 6-6

Schritt	Aktion	Hinweis / Erklärung
1	Stellen Sie in den Variablentabellen VAT_A und VAT_B das automatische Senden von Daten ein. (siehe Kap. 6.2)	Kontrollieren Sie, dass die Kommunikation bidirektional erfolgt.
2	Erhöhen Sie in VAT_C den Parameter `nStations` auf 8.	Der Parameter `nStations` beschreibt die Anzahl der zu verwaltenden Stationen. Da jeder Station eine Empfangs- und eine Sendeeinheit zugeordnet ist und diese Einheiten in einem Zyklus durchlaufen werden, erhöht sich dadurch auch die Zykluszeit. Die Auswirkung auf die Telegrammlaufzeit kann nach 100 Messungen beobachtet werden.

6.5 Diagnosemöglichkeiten

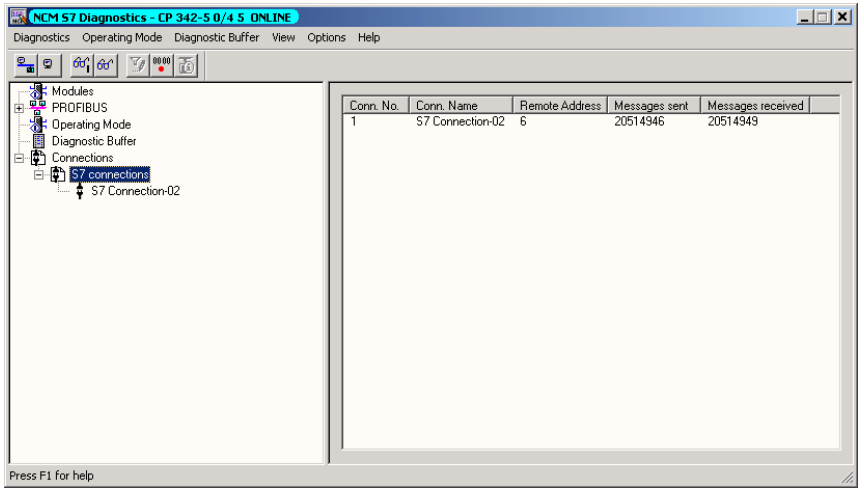
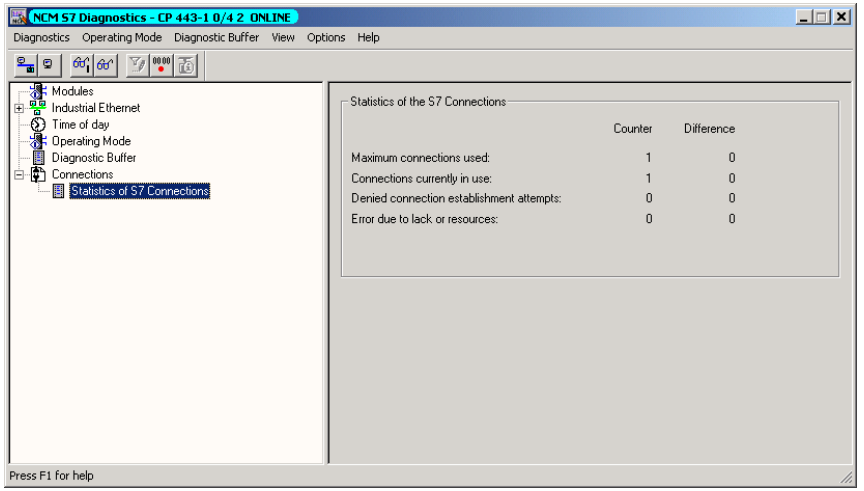
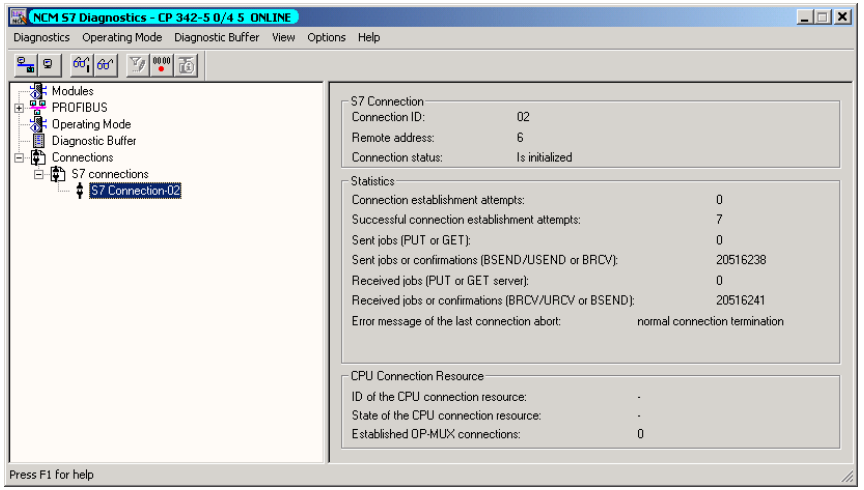
Übersicht

In folgendem Kapitel werden Diagnosemöglichkeiten für S7-Verbindungen durch die CPU und die hier genutzten CPs erläutert.
Dabei wird ausschließlich auf Diagnosemöglichkeiten in Step7 eingegangen.
Zur Diagnose von Kommunikationsfehlern in der Applikation sei auf Kap. 4.6.6 respektive 7.1.2 „Fehlerbehandlung“ verwiesen.

S7-Verbindungsdiagnose durch NCM S7

Das Projektierungspaket NCM S7 bietet für die Diagnose von S7 Verbindungen Möglichkeiten innerhalb der CP Diagnose an.

Tabelle 6-7: Diagnosemöglichkeiten über NCM S7

	S7 – 300	S7 – 400
Verbindungs-übersicht		
Verbindungs-spezifische Informationen		Nicht vorhanden

Je nach der verwendeten Systemfamilie und dem genutzten System stehen unterschiedlich komfortable Diagnosemöglichkeiten zur Verfügung.

In der S7 - 300 Familie besteht ein unterschiedliches Verhalten zwischen den folgenden beiden Verbindungstypen und deren Diagnosemöglichkeit über NCM:

Tabelle 6-8: CP Diagnosemöglichkeiten der S7 – 300 CP über NCM S7

Über CPU gehaltene unidirektional Verbindungen	Über CP gehaltene uni- und bidirektionale Verbindungen
Die NCM Diagnose bietet hier keine Möglichkeit, Ressourcen oder den Telegrammverkehr zu prüfen.	Über die NCM Diagnose sind Diagnosemöglichkeiten des CPs und seiner Verbindungen gegeben. Hier sind sowohl detaillierte Zahlen über die gesendeten und empfangenen Telegramme sowie Informationen über die laufenden Aufträge und Hintergrundinformationen über möglicherweise stattgefundene Verbindungsabbrüche auslesbar.

Hinweis

CPs der S7 – 300 besitzen erst ab dem CP 342-5DA02 bzw. CP 343-1EX11 und höher die Möglichkeit des S7-Verbindungsmultiplexens.

Innerhalb der S7 – 400 Familie ist keine verbindungsspezifische Diagnose möglich, da die Verbindung, entgegen der S7 300, von der CPU gehalten wird.

Die S7 400 NCM Diagnose für die CPs bietet einzig eine Übersicht der über den CP geführten S7 Verbindungen an.

Verbindungsdiagnose der S7 CPU

Die S7 CPU bietet in der Funktion „Baugruppenzustand“ unter der Lasche „Kommunikation“ eine Übersicht der für die Kommunikation projektierten und verwendeten Ressourcen.

Es besteht hier die Möglichkeit zu prüfen, ob die im Projekt projektierten Verbindungen auch aufgebaut wurden.

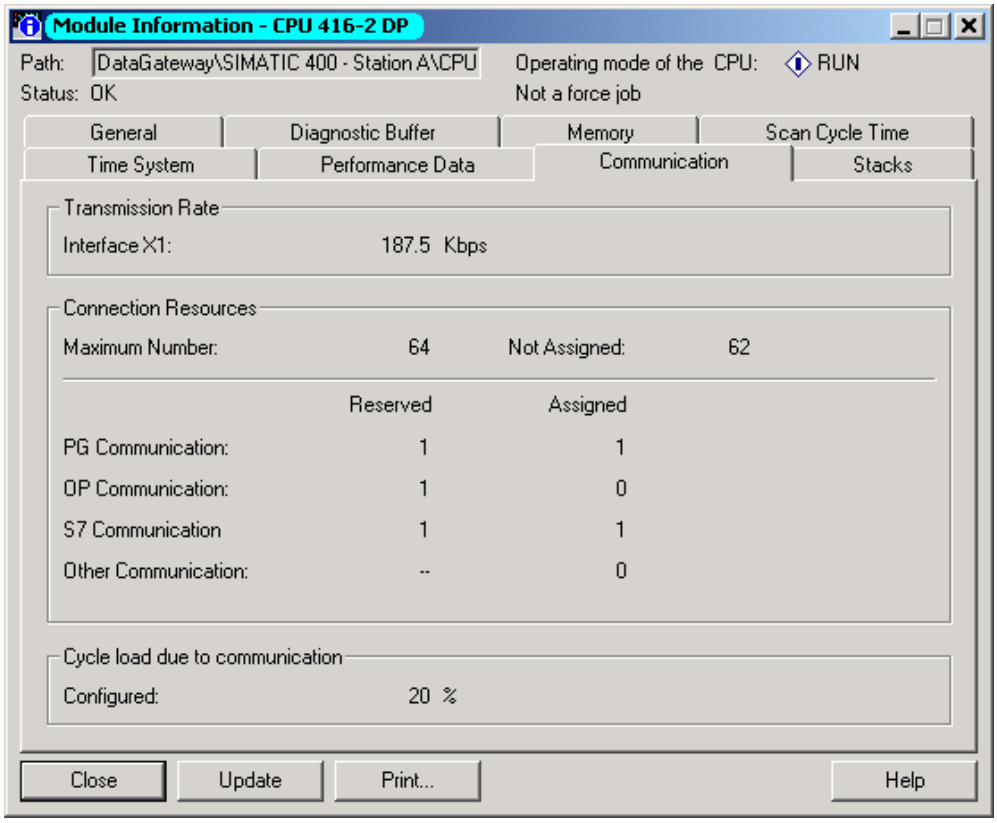


Bild 6-3 Screenshot CPU Moduldiagnose

Die hier verwendete Ansicht der S7 416 CPU ist vom Schema her für alle S7-CPUen identisch.

Im mittleren Bereich des Fensters ist die Übersicht der Verbindungsressourcen zu sehen. Es werden die maximal möglichen Verbindungen sowie die noch verfügbaren Verbindungsressourcen angezeigt. Weiterhin werden die auf die einzelnen Ressourcentypen verteilten projektierten und genutzten Verbindungen angezeigt.

Der einzige Unterschied zwischen einer S7 - 300 CPU und einer S7 – 400 CPU besteht in diesem Fall in der Anzeige der S7-Basis-Kommunikation anstelle der S7-Kommunikation.

Bild 6-4: Anzeige der S7 – 300; S7-Basiskommunikation

S7-Kommunikations-Verbindungen werden in der S7 - 300 als „Sonstige Kommunikation“ aufgeführt, da sie nicht direkt projektierbar sind.

Verbindungsdiagnose über NetPro in der Verbindungstabelle

Die letzte Möglichkeit der Verbindungsdiagnose ist innerhalb des von Step 7 angebotenen Projektierungstools NetPro möglich.

Durch Aktivieren der Verbindungsstatusfunktion (Drücken des Buttons  oder über „Zielstation!“ > „Verbindungsstatus aktivieren“) erhalten Sie innerhalb der Verbindungstabelle den aktuellen Verbindungsstatus der projektierten Verbindung.

Connection status	Local ID	Partner ID	Partner	Type
Set up	1	1	SIMATIC 400 - Station A / CPU 416-2 DP	S7 connection
Set up	2	1	SIMATIC 400 - Station B / CPU 416-2 DP	S7 connection

Bild 6-5 Screenshot der Verbindungsdiagnose aus NetPro.

Teil C : Programmbeschreibung

Ziel Teil C

Dieser Teil der Dokumentation soll

- dem Leser Details aus dem Code einiger Kernprogrammteile erläutern
- Hinweise liefern, wo Erweiterungen sinnvoll sind

Voraussetzung

Dies ist keine Einführung in die STEP7-Sprache AWL. Der Leser sollte die Grundlagen dieser Sprache beherrschen.

Hilfreich ist es, vor der Code-Beschreibung die Kapitel im Teil A1 und A2 zu lesen.

Inhalt Teil C

7 Erläuterungen zum STEP7-Programm

Einleitung

In diesem Kapitel werden zu den wichtigsten Funktionselementen die wichtigsten Kern-Codesequenzen dargestellt und erläutert. Im Einzelfall wird auf den gut kommentierten Code in der Applikation verwiesen.

Inhalt Kapitel Erläuterungen zum STEP7-Programm

Tabelle 7-1

Kap.-Nr.	Kapitelüberschrift	Hier erfahren Sie...
7.1	Funktionselemente in der Quell-/Zielstation	... welche Funktionselemente in der Quell-/Zielstation angewendet werden und wie diese funktionieren.
7.2	Funktionselemente im Gateway	... welche Funktionselemente im Gateway angewendet werden und wie diese funktionieren.

7.1 Funktionselemente in der Quell-/Zielstation

7.1.1 Pointergenerator

Das Funktionselement „Pointergenerator“ findet auch im Gateway Anwendung.

Aufgaben des Funktionselements

Um Datenbausteine dynamisch adressieren zu können, müssen Any-Pointer verwendet werden, die die Datenbausteinnummer, die Anfangsadresse und die Länge des zu adressierenden Datenbereichs spezifizieren.

Das Funktionselement „Pointergenerator“ erstellt einen ANY-Pointer auf einen Datenbaustein.

Mechanismus des Funktionselements

Um einen Any-Pointer auf einen DB erzeugen zu können, müssen folgende Parameter bekannt sein:

- DB-Nummer
- Länge des DBs
- Startadresse/Offset des Pointers innerhalb des DBs

Ist der Any-Pointer erstellt, so muss er noch auf den Ausgabepointer kopiert werden.

Da der Any-Pointer 10 Bytes lang ist, erfolgt das über eine Reihe von Lade-/Transferiere-Anweisungen.

Folgendes Vorgehen ist dazu notwendig:

Tabelle 7-2

Schritt	Aktion	Verweis/Code*
1	Öffne Datenbaustein und berechne die Länge des DBs	FC21, Netzwerk 1
2	Generiere einen Any-Pointer auf den Datenbaustein mit Startadresse 0	FC21, Netzwerk 1 und FC20, Netzwerk 1 FC20 : Pointer Generator generates an Data-Block ANY pointer to the OUT-Parameter 'ANY_Pointer' from IN-Parameters: DB_Number[WORD], Length[WORD] and Strartaddress[WORD] Network 1: Title: Simple Pointer Generator for Datablocks. L P##ANY_Pointer // load the pointer into memory LAR1 L B#16#10 // sets start parameter T B [AR1,P#0.0] L B#16#2 // sets the Datatype Byte T B [AR1,P#1.0] L #Length // sets the length of the field T W [AR1,P#2.0] L #DB_Number // sets the DB number T W [AR1,P#4.0] L #Startaddress // sets the Startaddress T #Memory L #Memory SLD 3 T #Memory L B#16#84 // sets the Type Datablock T LB 0 L #Memory T D [AR1,P#6.0] BEU
3	Kopiere Any-Pointer in den Output-Parameter	// Copy any pointer L P##anyOutPointer LAR1 L LW 0 T W [AR1,P#0.0] L LW 2 T W [AR1,P#2.0] L LW 4 T W [AR1,P#4.0] L LW 6 T W [AR1,P#6.0] L LW 8 T W [AR1,P#8.0] Der Output-Any-Pointer ist im Lokaldatenbereich bei Adresse 0 deklariert.

*) Die Angaben beziehen sich auf das Gateway

Aufgrund der Formatierung des Any-Pointers und der restriktiven Aufgabe dieses Bausteins, dass der Any-Pointer ausschliesslich auf einen DB verweisen soll, sind einige Informationen fest vorgegeben.

So kennzeichnet z.B. das 1. Byte des 10 Bytes großen Any-Pointers, auf welchen Datenbereich er zeigen soll.
Im Falle eines DB ist hier die 2 einzutragen.

Die anderen Parameter sind dementsprechend zu vergeben.

Hinweise

- Ein Any-Pointer kann nur bei einem FC (Function) als OUT-Parameter deklariert werden.
- Das Format und die jeweiligen Parameter eines Any-Pointers sind in der SIMATIC-Hilfe abgedruckt.

Parameter des Funktionselements

s. Kap. 4.6.3

7.1.2 Fehlerbehandlung

Das Funktionselement „Fehlerbehandlung“ findet auch im Gateway Anwendung.

Aufgaben des Funktionselements

Das Funktionselement Fehlerbehandlung hat folgende Aufgaben:

- Protokollierung eines auftretenden Fehlers bei BSEND/BRCV-Aufrufen mit dem dazugehörigen Status.
- Überprüfung, ob der vorgefundene Fehlerstatus bekannt ist und ob darauf reagiert werden kann.
- Ermittlung der projektierten Fehlerreaktion und Rückgabe derselbigen an die aufrufende Funktion.

Mechanismus der Fehlerauswertung

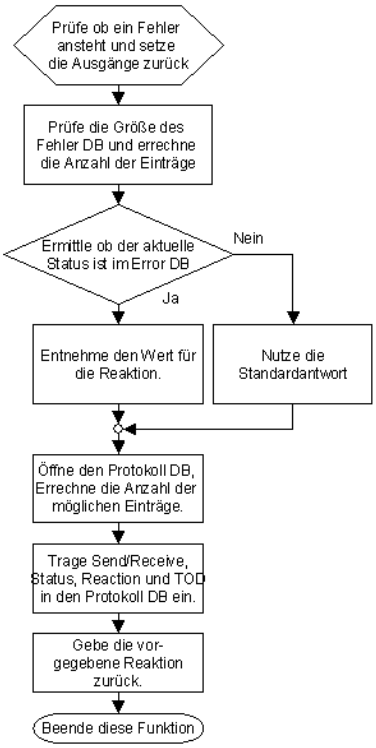
Die Fehlerauswertung an sich basiert auf einem Verzeichnis von Regeln. Diese Regeln werden in Form eines Datenbausteines (Reaktions-DB) abgelegt. In diesem Datenbaustein werden die „erwarteten“ Statuszustände der Funktionen zusammen mit den angestrebten Reaktionen abgelegt.

Tritt ein Fehler in einer der zu prüfenden Funktionen auf, wird anhand der Statusausgabe versucht, im Fehler-DB die verknüpfte Reaktion zu finden.

Wird eine Reaktion gefunden, so wird diese ausgegeben; wird keine Reaktion gefunden, so kann von der Applikation nicht mit einer vorgegebenen Reaktion reagiert werden.

Funktionsprinzip des Error_Control FB 33

Tabelle 7-3

Flussdiagramm	Beschreibung
	Im ersten Schritt wird geprüft, ob ein Fehler ansteht. Ist das der Fall, so wird der Baustein weiterbearbeitet. Weiterhin werden alle Ausgänge zurück gesetzt. Daraufhin wird der Fehler DB nach dem vorgefundenen Status durchsucht; wird der Status gefunden, so wird die im DB hinterlegte Information über die Reaktion entnommen. Ist der Status nicht im DB hinterlegt, wird ein Standardwert für die Reaktion zurückgeliefert. Für den aktuellen Eintrags-Index werden nun die Information über den Pfad, in dem der Fehler aufgetreten ist sowie der aufgetretene Status und die darauf gewählte Reaktion inkl. dem aktuellen Zeitstempel hinterlegt. Am Ende der Funktion wird die Reaktion (z.B. Restart, Stop) entsprechend der verfügbaren Ausgangsparameter ausgegeben.

Notwendige Datenstrukturen zur Fehlerauswertung

Der Reaktions-DB ist ein Datenbaustein, der aus einem Array von Variablen des Typs UDT 110 besteht. Dieser Datentyp ist wie folgt aufgebaut:

Address	Name	Type	Initial val	Comment
0.0		STRUCT		
+0.0	Status	WORD	W#16#0	
+2.0	Action	WORD	W#16#0	
=4.0		END_STRUCT		

Bild 7-1: UDT 110 „Table_Status_Reaction “

Ein im Reaktions-DB gebildetes Array diesen Datentyps enthält eine beliebige, von der Funktion automatisch erkannte Anzahl von Datensätzen. Diese werden sukzessive durchlaufen, um die Reaktion (Action) als Ergebnis auf den Status zu ermitteln.

Protokollierung der auftretenden Ereignisse

Um die auftretenden Ereignisse protokollieren zu können, sind folgende Informationen notwendig:

- Die Send_Receive Information, welche beschreibt, ob der erhaltene Fehler im Sende- bzw. im Empfangspfad der Funktion aufgetreten ist.
- Die Statusinformation, die bei Aufruf der Funktion am zu untersuchenden Baustein angezeigt wurde.
- Die Reaktion, die durch den Fehler DB definierte Reaktion auf den Status.
- Der Zeitstempel, der eine eindeutige Dokumentation über den Zeitpunkt des Fehlers ermöglicht.
- Der Ort, an dem der Fehler passiert ist, inkl. 2 beliebigen Integer-Parametern (hier: verwendete ID und R_ID)

Ein im Protokollbaustein gebildetes Array des Datentyps „Errorstruct“ enthält eine beliebige, von der Funktion automatisch erkannte, Anzahl von Datensätzen. Jeder neue Eintrag wird automatisch an den letzten Eintrag angefügt. Ist die maximale Anzahl von Einträgen erreicht, wird an den Beginn des Protokollbausteines zurück gesprungen.

Notwendige Datenstruktur des Protokollbausteins

Der Protokollbaustein ist ein Datenbaustein, der aus einem Array von Variablen des Typs UDT111 „Errorstruct“ besteht. Dieser Datentyp ist wie folgt aufgebaut:

UDT111 -- DataGateway\SIMATIC 400 - Station B\CPU 416-2 DP

Address	Name	Type	Initial val	Comment
0.0		STRUCT		
+0.0	Send_Receive	BOOL	FALSE	Send (false) / Receive (true)
+2.0	Status	WORD	W#16#0	
+4.0	Reaktion	WORD	W#16#0	1 = Stop, 2 = Restart, FFFF = Unknown
+6.0	Timestamp	TIME_OF_DAY	TOD#0:0:0.0	
+10.0	Location	"Errorlocationst:		
=18.0		END_STRUCT		

Bild 7-2: UDT 111 „Errorstruct“

Er bedient sich folgender Struktur (UDT 109), damit der Ort des Fehlers erfasst werden kann:

Address	Name	Type	Initial val	Comment
0.0		STRUCT		
+0.0	Blocktype	CHAR	' '	'C': FC, 'B': FB, 'O': OB
+2.0	Blocknumber	INT	0	
+4.0	ID1	INT	0	can be used for IDs. E.g. BSEND --> ID
+6.0	ID2	INT	0	can be used for IDs. E.g. BSEND --> R_ID
=8.0		END_STRUCT		

Bild 7-3 UDT 109 „Errorlocationstruct“

Parameter des Funktionselements

s. Kap. 4.6.6.

7.1.3 Statistik

Aufgaben des Funktionselements

Das Funktionselement „Statistik“ soll aus einer Anzahl von Messwerten den höchsten, niedrigsten und mittleren Wert (bzgl. 100 Messwerten) berechnen.

Die Messwerte dieser Applikation entsprechen der Telegrammlaufzeit und der Zykluszeit. Dabei werden immer die letzten 100 Werte berücksichtigt. D.h. nach 100 Werten wird der Mittelwertbildner im „Manager“-Baustein zurückgesetzt, um die nächsten 100 Messwerte zu erfassen.

Mechanismus des Funktionselements

Nachfolgend wird lediglich das Vorgehen zur Ermittlung der Telegrammlaufzeit erläutert. Für die Zykluszeit gilt entsprechendes:

Um die Telegrammlaufzeit ermitteln zu können, müssen zwei Zeitpunkte gemessen werden:

- Anfangszeit: Wann wird das Telegramm gesendet
- Endzeit: Wann ist das Telegramm quittiert

Die Telegrammlaufzeit ist sodann die Differenz aus Stop- und Startzeitpunkt.

Es ist also folgendes zu tun:

Tabelle 7-4

Schritt	Aktion	Bemerkung
1	Sobald ein Telegramm gesendet werden soll, merke die Zeit	Dies kann z.B. mit einem Merker oder einer statischen Variable realisiert werden
2	Sobald ein Telegramm quittiert wurde, merke die Zeit	Dies kann z.B. mit einem Merker oder einer statischen Variable realisiert werden
3	Bilde die Differenz der Zeiten	Da die Differenz der Zeiten im Format TOD in der Regel gering ist, muss keine separate Umrechnung durchgeführt werden
4	Übergebe den berechneten Wert dem Mittelwertbildner	Der Mittelwertbildner errechnet nun die gewünschten Werte
5	Konvertierung der gelieferten Werte	Der Mittelwertbildner rechnet mit dem Datentyp Real. Dieser Datentyp muss noch in ein Doppelwort umgewandelt werden, damit er in der Variablentabelle als Zeitunterschied angezeigt werden kann.

Parameter des Funktionselements

s. Kap. 4.6.7

7.1.4 Sendebaustein

Aufgaben des Funktionselements

Der Sendebaustein stellt die Anwenderschnittstelle zum Senden von Daten über das Gateway dar.

Er hat folgende Aufgaben:

- Ausgehend von Nutzdaten muss ein Telegramm erstellt werden, das vom Gateway weitergeleitet werden kann
- Der BSEND-Baustein muss korrekt versorgt werden
- Der Sendebaustein muss auf die Quittung vom Gateway warten und diese auswerten
- Solange kein Fehler auftritt und solange der Baustein sendet oder auf eine Quittung wartet, darf er kein weiteres Telegramm verschicken (Verriegelungsfunktion während eines Sendevorgangs)

Notwendige Datenstrukturen für den Sendebaustein

Um den Sendebaustein betreiben zu können, ist eine Reihe von Datenstrukturen anzulegen.

1. Datenstruktur mit Nutzdaten (UDT 101)
   Diese Datenstruktur kann beliebiges Aussehen haben.
   In der Applikation ist diese Struktur im UDT101 als eine Reihe von 200 Char-Datentypen hinterlegt.
2. Datenstruktur „IDData“ (UDT 103)
   Diese Datenstruktur ist Teil der Kopfdaten. Sie enthält die Quell- und Zielinformationen – dies entspricht der jeweiligen Local_ID aus Sicht des Gateways.
   Diese Struktur besteht in der aktuellen Applikation lediglich aus einem Integer; falls es jedoch nötig ist, eine Erweiterung zu implementieren, so ist dies durch diese Struktur ohne weiteres möglich (z.B. R_ID mit eintragen).
3. Datenstruktur „Telheader“ (UDT 100)
   Die Datenstruktur „Telheader“ entspricht nun den Kopfdaten:
   Diese Struktur enthält eine Reihe an intern genutzten Datentypen:
   - byType: Gibt an, ob es sich um Daten- oder ein Quittungstelegramm handelt.
     Dadurch, dass eine Byte verwendet wurde, sind weitere Telegrammtypen denkbar.
     In der aktuellen Applikation gilt folgende Codierung:
     1: Datentelegramm
     2: Quittungstelegramm
   - TimeStamp: Mit dem Zeitstempel kann der Zeitpunkt des Versendens eines Datentelegramms exakt nachvollzogen werden

- tSourceInfo und tDestInfo entsprechen den Quell- und Zielinformationen, sind also Typen von „IDData“.
- wTelegrammID ermöglicht das Versenden von unterschiedlichen benutzerdefinierten Telegrammtypen.
  Dadurch ist es möglich, mehrere verschiedene BSEND/BRCV-Pärchen zwischen **einer** Quell- und **einer** Zielstation zu betreiben.
  Dieses Datum wird von der aktuellen Applikation nicht ausgewertet.
- wError zeigt den Fehlercode bei BSEND-Aufrufen im Gateway an
- wStatus zeigt den Status des Gateways an.
  Dabei gilt folgende Codierung:
  0: Routing noch nicht gestartet oder Routing läuft
  1: Routing ist beendet

UDT100 -- DataGateway\SIMATIC 400 - Station B\CPU 416-2 DP

Address	Name	Type	Initial val	Comment
0.0		STRUCT		
+0.0	byType	BYTE	B#16#0	is it data or an acknowledgement
+2.0	TimeStamp	TIME_OF_DAY	TOD#0:0:0.0	Timestamp
+6.0	tSourceInfo	"IDData"		Source information
+8.0	tDestInfo	"IDData"		Destination information
+10.0	wTelegrammID	WORD	W#16#0	ID of telegram for sending different telegramms on one port
+12.0	wError	WORD	W#16#0	Error Code of used "BSEND" in Gateway
+14.0	wStatus	WORD	W#16#0	Status of routing
=16.0		END_STRUCT		

Bild 7-4 Kopfdaten eines Telegramms

4. Datenstruktur „SendStruct“ (UDT 102)
   “SendStruct“ ist nun nichts weiteres als ein aus Nutzdaten (UDT 101 „Datastruct“) und Kopfdaten (UDT 100 „Telheader“) zusammengesetzter Datentyp:

UDT102 -- DataGateway\SIMATIC 400 - Station B\CPU 416-2 DP

Address	Name	Type	Initial val	Comment
0.0		STRUCT		
+0.0	Head	"TelHeader"		
+16.0	Data	"DataStruct"		
=216.0		END_STRUCT		

Bild 7-5 Zusammengesetztes Telegramm aus Kopf- und Nutzdaten

Flussdiagramm des Sendebausteins

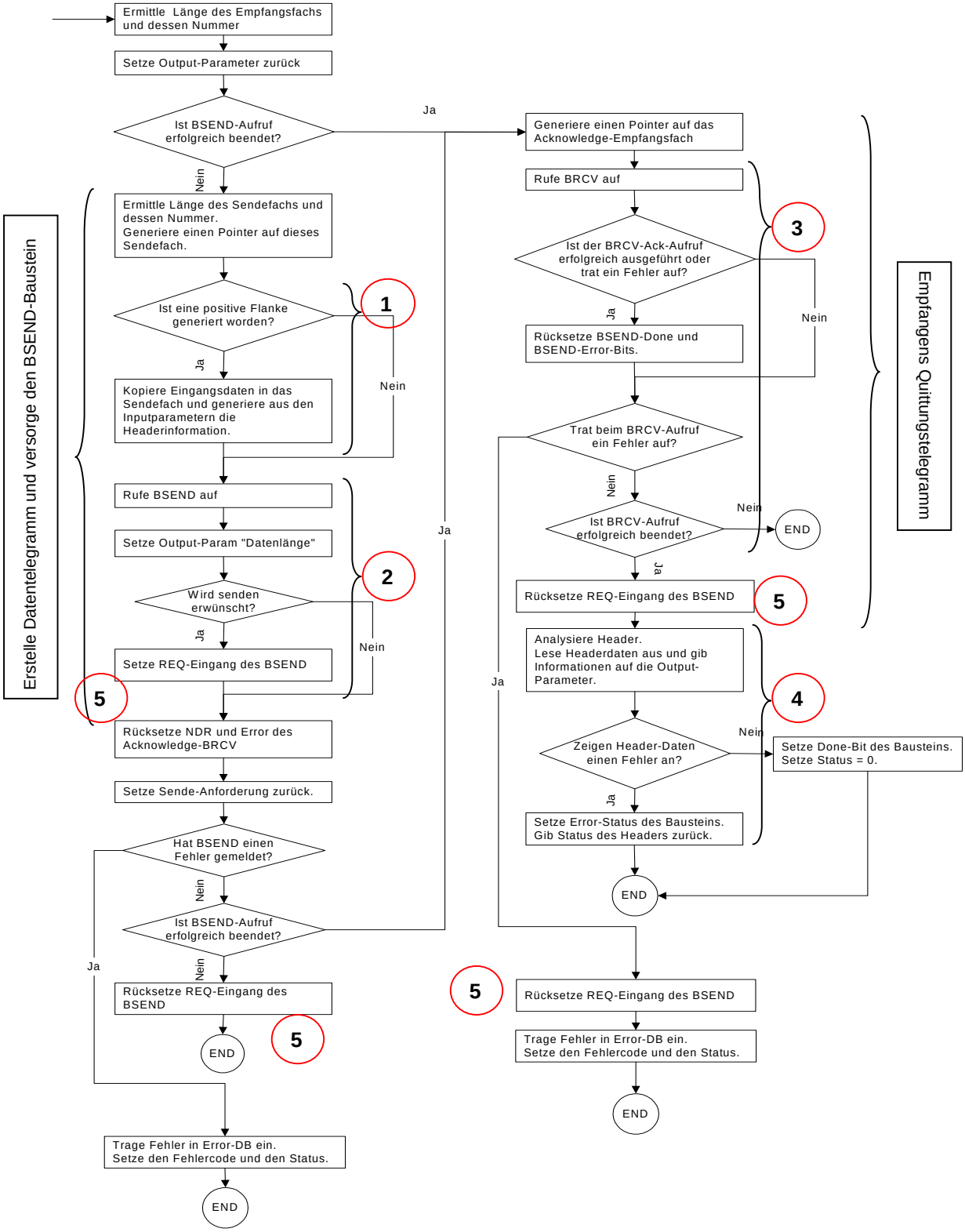


Bild 7-6 Flussdiagramm des Sendebausteins

Erläuterung zum Sendebaustein

1. Erstellung eines Datentelegramms:

   Die Erstellung eines Telegramms erfolgt nur nach einer positiven Flanke am REQ-Eingang des BSEND.
   Damit die Kopfdaten und damit der Zeitstempel genau dann erstellt werden, bevor der BSEND-Baustein die Daten verschickt, muss eine Flankenauswertung realisiert werden. Siehe dazu FB25 „SendBlock“, Netzwerk 3.

2. Korrekte Versorgung des BSEND-Bausteins:

   Die Variable bBSEND_REQ wird nur dann auf TRUE gesetzt, falls der Eingang bIOBSEND_REQ des Sendebausteins TRUE ist.
   Der BSEND-Baustein wird sodann folgendermaßen aufgerufen:

```
snd:  CALL  "BSEND" , DB12
       REQ   :=#bBSEND_REQ
       R     :=#bIOReset
       ID    :=W#16#1                    // We will send with ID=1, because the gateway is connected in this way
       R_ID  :=DW#16#1                   // R_ID is always "1", because we want to send data to the gateway
       DONE  :=#bBsend_Done
       ERROR :=#bBsend_Error
       STATUS:=#wBsend_Status
       SD_1  :=#anyTmpSndPocket
       LEN   :=#wBsendLen

      L     #wBsendLen
      T     #wOutBSendLen

// If Bsend is requested, then set the rising-edge flag "bBSEND_REQ".
      SET
      U     #bIOBsend_REQ
      S     #bBSEND_REQ
```

Bild 7-7

D.h. der BSEND-Baustein wird zunächst mit REQ:=FALSE aufgerufen. Sobald bIOBSEND_REQ TRUE wird, wird im **nächsten** Zyklus der BSEND-Baustein ebenfalls mit TRUE aufgerufen.
Somit startet der Sendevorgang.

Hinweis

Da das Datentelegramm mit einer positiven Flanke an bBSEND_REQ erstellt wird und der REQ-Eingang des BSEND-Bausteins mit der gleichen Variable verschaltet ist, erfolgen sowohl die Erstellung des Datentelegramms als auch der Anstoß des Sendevorgangs im gleichen Zyklus.

Die Beschaltung des BSEND-Baustein erfolgt nach folgendem Muster (nur die wichtigsten Parameter):

Tabelle 7-5

Parameter	Bedeutung	Beschaltung
R	Reset Bricht den aktuellen Sende-auftrag ab.	Wird hier mit dem Reset-Eingabeparameter verschaltet, um einen laufenden Auftrag abbrechen zu können (s. dazu Kap 6.3 und 8.4)
ID	Local ID Dieser Parameter entspricht der Local ID, für die das Gateway bei der Quell-/Zielstation projektiert ist	Das Gateway ist in dieser Applikation die einzige projektierte Verbindung, die die Quell-/Zielstation hat. Da der Verbindung die Local ID 1 zugewiesen wurde, ist hier fest eine 1 anzugeben
R_ID	R_ID Mit der R_ID wird das BSEND/BRCV-Pärchen identifiziert. Der R_ID-Parameter muss auf Sender- und Empfängerseite gleich sein.	Da das Gateway Datentelegramme mit R_ID=1 empfängt, ist hier zwingend eine 1 anzugeben. (s. dazu Kap. 4.5.1)

3. Warte auf Quittungstelegramm:

Sobald der BSEND-Baustein erfolgreich beendet wurde, wird der BRCV-Baustein zum Empfangen eines Quittungstelegramms aufgerufen.

Folgender Code-Auszug illustriert die Implementierung:

```
          SET
          AN    #bBsend_Done
          BEC

Network 6 : RcvAck
Receives the acknowledge

    rAck: NOP   0

          CALL  "GenPointerOnDB"
           nInDBNr      :=#nTmpAckDBNr
           anyOutPointer:=#anyTmpAckDB

          CALL  "BRCV" , DB13
           EN_R  :=TRUE
           ID    :=W#16#1                  // We will receive with ID=1, because the gateway is connected in this way
           R_ID  :=DW#16#3                 // R_ID is always "3", because we want to receive an acknowledge from the gateway
           NDR   :=#bNDRAck
           ERROR :=#bErrorAck
           STATUS:=#wStatusAck
           RD_1  :=#anyTmpAckDB
           LEN   :=#wLenAck

          CLR
          0     #bNDRAck
          0     #bErrorAck
          R     #bBsend_Done
          R     #bBSEND_REQ
```

Bild 7-8

Ist also der DONE-Ausgang des BSEND-Bausteins gesetzt (also ist der BSEND-Baustein erfolgreich beendet worden), so wird der BRCV-Baustein mit dem dazugehörigen Empfangsfach aufgerufen.

Sobald der BRCV-Baustein entweder einen Fehler oder das Ende des Datenempfangs signalisiert, wird der Sendebaustein intern wieder freigeschaltet (bBSEND_REQ wird wieder zurückgesetzt).

Hinweis

Beachten Sie, dass die Local ID zwar 1, die R_ID jedoch 3 sein muss, da das Gateway über R_ID:=3 das Quittungstelegramm sendet. (s. Kap. 4.5.1)

4. Die Auswertung der Quittungsdaten wird durch Umkopieren in die definierte Kopfstruktur realisiert (s. FB25 „SendBlock“, Netzwerk 7).

Weiterhin wird das Done-Bit des Sendebausteins gesetzt, der damit nun auch nach außen signalisiert, dass er neue Daten verschicken kann.

Hinweis

An diese Stelle könnte noch der Zeitstempel des Quittungstelegramms ausgewertet und mit dem Zeitstempel des gesendeten Datentelegramms verglichen werden (s. Kap. 8.4.1).

5. Verriegelung des Bausteins, solange kein Fehler auftritt und solange der Baustein sendet oder auf eine Quittung wartet

Am Flussdiagramm ist zu erkennen, dass die Variable bBSEND_REQ erst dann wieder zurückgesetzt wird, wenn entweder ein Fehler auftrat oder der gesamte Sendeauftrag inkl. Quittung beendet ist.

D.h. der BSEND-Baustein wird erst dann einen neuen Sendevorgang starten, wenn bBSEND_REQ auf FALSE und im nächsten Zyklus wieder auf TRUE gesetzt wird.

Dies kann aber nur eintreten, wenn ein Fehler mit BSEND/BRCV-Aufrufen auftrat oder der BRCV-Quittungsauftrag erfolgreich beendet wurde.

Hinweis

In Kap. 4.6.8 wird der Ablauf vereinfacht anhand eines Zustandsdiagramms aufgezeigt.

Parameter des Funktionselements

s. Kap. 4.6.8.

7.2 Funktionselemente im Gateway

7.2.1 FC1 „Manager“

Aufgaben des Funktionselements

Der FC1 „Manager“ hat folgende Aufgaben:

- Ausführen der Statistik-Funktionalität.

- Prüfen, ob ein Ringpuffer voll ist.
  Wenn ein Ringpuffer voll ist, so wird nicht mehr empfangen.
- Zyklischer Aufruf aller Empfangseinheiten, sofern kein Ringpuffer voll ist.
- Zyklischer Aufruf aller Sendeeinheiten.

Flussdiagramm des Funktionselements

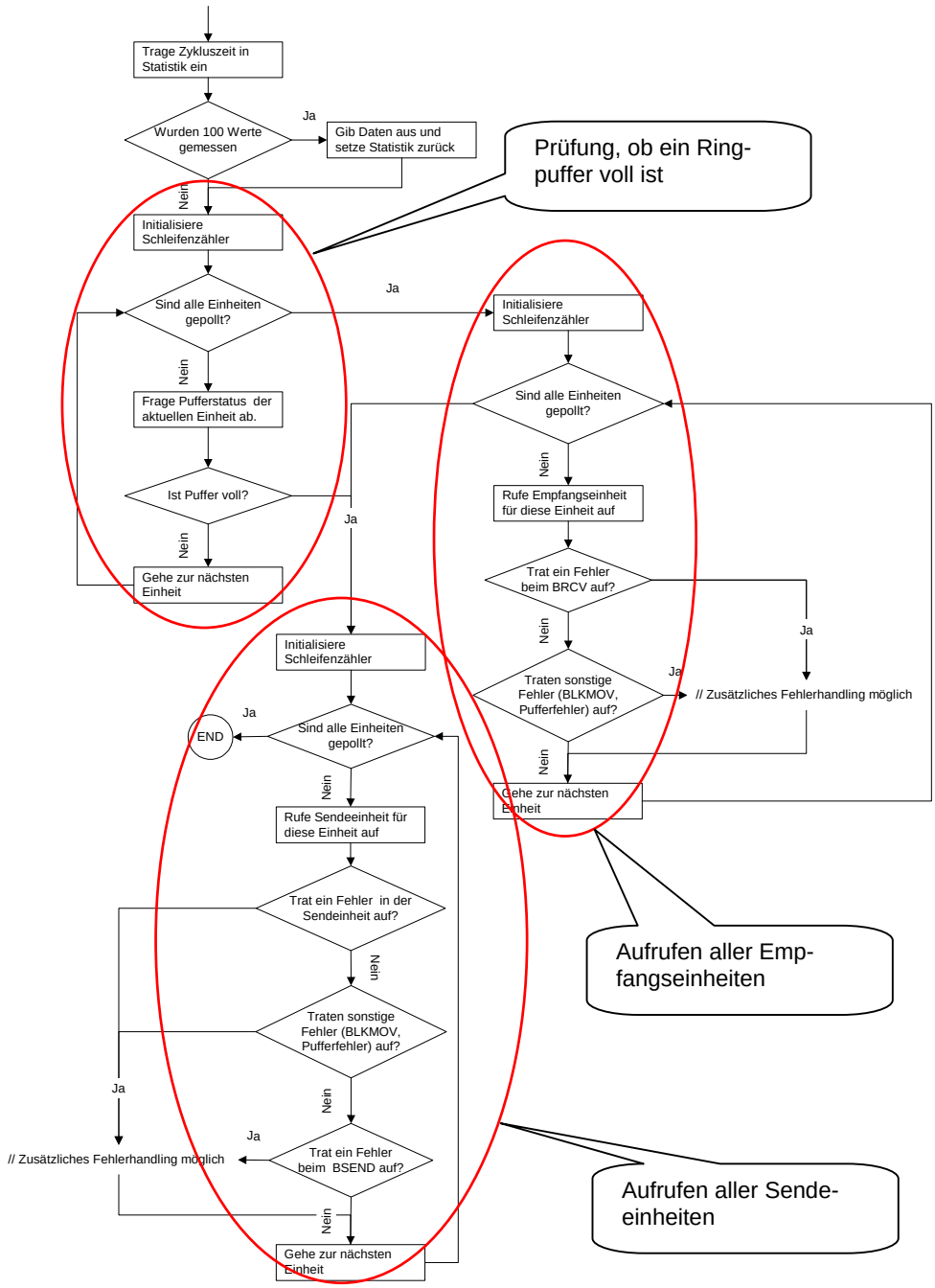


Bild 7-9 Flussdiagramm zum Funktionselement FC1 „Manager“

Erläuterung zum Funktionselement

Der FC1 „Manager“ besteht im wesentlichen aus 3 Schleifen:

- Prüfen, ob ein Ringpuffer voll ist
  Zeigt der Status eines einzigen Ringpuffers an, dass er voll ist, so wird mit den Sendeeinheiten fortgefahren. Dadurch wird verhindert, dass innerhalb des Gateways ein Datentelegramm „verloren“ gehen kann. Der Sendevorgang auf der Quellstation verlängert sich entsprechend.
- Aufrufen aller Empfangseinheiten und
- Aufrufen aller Sendeeinheiten
  Sowohl die Empfangs- als auch die Sendeeinheiten werden innerhalb einer separaten Schleife aufgerufen. Der Schleifenzähler entspricht dabei der Anzahl zu verwaltender Stationen.
  Dadurch werden in **jedem** Zyklus **alle** Kommunikationskanäle nach neuen Daten abgefragt und vorhandene Daten verschickt.
  Innerhalb einer Schleife führen Fehler, die die Empfangs- oder Sendeeinheit liefert (z.B. ein Fehler beim Schreiben in den Ringpuffer), zu einem Übergang zur nächsten Empfangs- bzw. Sendeeinheit. Ein Fehlerhandling ist nicht implementiert, da die Fehlerwahrscheinlichkeit gering ist (siehe dazu Kap. 8.4.2).
  Weiteres Fehlerhandling, wie z.B. bei BSEND/BRCV-Aufrufen, findet in den untergeordneten Bausteinen über das Funktionselement „Fehlerbehandlung„ (siehe Kap. 4.6.6 bzw. 7.1.2) statt.

Parameter des Funktionselements

s. Kap. 4.6.2

7.2.2 Ringpuffer

Aufgaben des Funktionselements

Nachfolgend wird nicht auf das gesamte Funktionselement „Ringpuffer“ eingegangen, sondern lediglich auf die eigentliche Funktionalität eines Ringpuffers, also das Speichern und Lesen von Daten, die in einer ringförmigen Struktur enthalten sind. Speichern und Lesen von Daten erfolgt dabei nach dem FIFO-Prinzip.

Diese Funktionalität ist im Baustein „FB2“ „Ringbuffer“ gekapselt.

Notwendige Datenstrukturen

Der Ringpuffer verwaltet folgende Informationen:

Address	Declara	Name	Type	Initial valu	Comment
0.0	in	BEREICH	ANY		Any Pointer (Source and Destination)
10.0	in	INSTANZ	BLOCK_DB		own instance DB of Function Block
12.0	in	FUNC	CHAR	'W'	determines the functionality ('0': o
14.0	out	STAT	BYTE	B#16#0	Status of the Ringbuffer
16.0	out	Entries	INT	0	Amount of entries stored in the ring
18.0	in_out	ACT	BOOL	FALSE	Activity Bit
20.0	stat	ANZ_REC	INT	0	Number of stored records in buffer
22.0	stat	WZ	INT	0	Number of next free buffer record
24.0	stat	RZ	INT	0	Number of next record to read
26.0	stat	STATUS	BYTE	B#16#0	Status of the last operation
28.0	stat	R_BUF	ARRAY[0..9]		

Bild 7-10

Dabei ist R_BUF der eigentliche Ringpuffer.
Der Ringpuffer hat also eine Größe von 10 Einträgen.

Der Schreib- bzw. Lesezeiger ist über WZ bzw. RZ realisiert.
Diese Zeiger sind keine SIMATIC-Zeiger, sondern logische Zeiger, wie folgender Codeauszug zeigt:

```
IF (FUNC = 'W' AND ACT = TRUE) THEN

  // If the buffer is full, set the status and do nothing
  IF (ANZ_REC = MAX_SIZE) THEN
      STATUS := S_FULL;
      STAT := STATUS;
      Entries := MAX_SIZE;
      RETURN;
  END_IF;

  // Calculate the length of the data structure (UDT102)
  VAR_LEN.VAR_LENDAT(Variable := R_BUF[0], Baustein := VAR_LENDAT);

  // Calculate the Offset to copy the data to
  OFFS := OFFS_STRCT + WZ * INT_TO_DINT(WORD_TO_INT(VAR_LENDAT.Summe));

  QUELLDAT := BEREICH;        // save the source pointer from the parameter

  // Modify the ANY Pointer to point to the proper ringbuffer location
  // length of UDT102
  INSTANZ.DW2 := VAR_LENDAT.Summe;
  // number of Instance DB
  INSTANZ.DW4 := INT_TO_WORD(GETDBNR (SRC := INSTANZ));
  INSTANZ.DD6 := DINT_TO_DWORD (8 * OFFS) OR 16#8400_0000;

  ZIELDAT := BEREICH;         // save the destination pointer

  // Copy data to Ringbuffer
  SFC_ERR := BLKMOV(SRCBLK := QUELLDAT,DSTBLK := ZIELDAT);

  IF SFC_ERR <> 0 THEN        // Error from "Block Move"
      STATUS:= S_BLKERR;
      STAT := S_BLKERR;
      ACT := FALSE;
      RETURN;
  END_IF;

  // Read-/Write Pointer Management
  ANZ_REC := ANZ_REC + 1;          // increment number of entries
  Entries := ANZ_REC;              // return number of entries
  STATUS := S_OK;                    // set Status = OK

  WZ := ( WZ + 1 ) MOD MAX_SIZE;       // increment Write location

  ACT := FALSE;                        // reset activation bit
  STAT := STATUS;
  RETURN;
END_IF;
```

Bild 7-11 Es wird ein Datenbereich in den Ringpuffer gespeichert

Hinweis

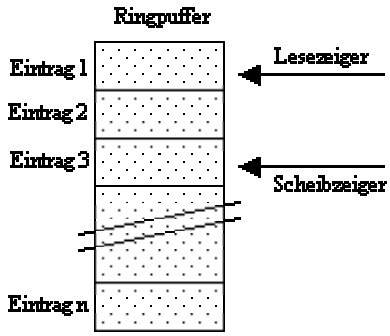
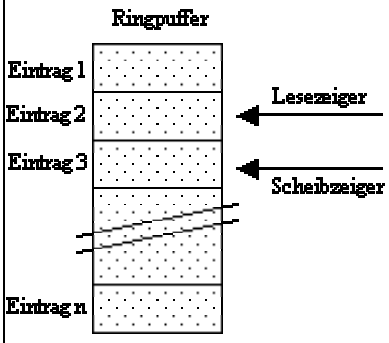
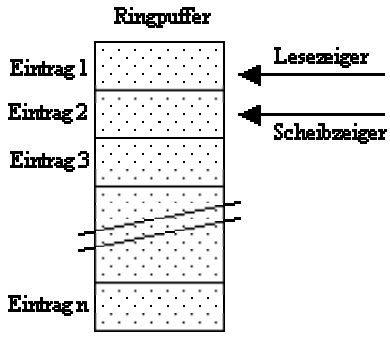
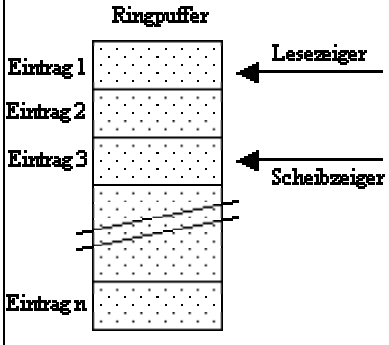
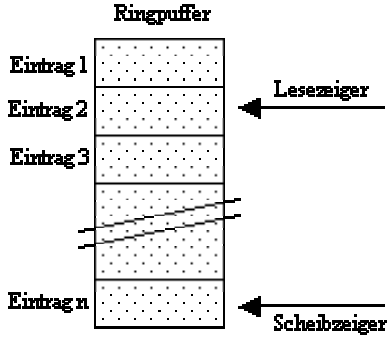
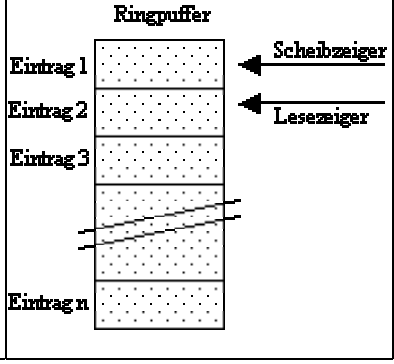
Da hier für die Parameterübergabe der Daten call-by-reference genutzt wird (durch Verwendung des Any-Pointers) ist der auszuführende Code komplexer.

Alternativ könnte die Parameterübergabe der Daten auch über call-by-value erfolgen. Dadurch würden sich jedoch der Speicherplatzbedarf und die Laufzeit erhöhen; vor allem, wenn die Größe der Nutzdaten mehrere kBytes beträgt.

Mechanismus des Funktionselements

Auf weitere Codebeispiele wird hier nun verzichtet. Statt dessen wird die prinzipielle Arbeitsweise des Ringpuffers gezeigt:

Tabelle 7-6

Operation	Vorher	Nachher	Erklärung
Lesen			Der Lesezeiger wird um eine Position inkrementiert.
Schreiben			Der Schreibzeiger wird um eine Position inkrementiert.
Schreiben (mit Übergang)			Der Schreibzeiger wird um eine Position inkrementiert. Da der Ringpuffer eine feste Größe hat, wird wieder von vorne beschrieben. Die Anzahl der Einträge entspricht nun: „Größe des Ringpuffers – 1“ = „n – 1“

Parameter des Bausteins FB2 „Ringpuffer“

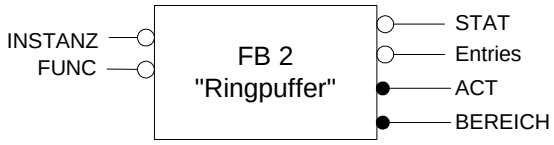


Bild 7-12

Tabelle 7-7

Name	Typ	Erklärung
BEREICH	Any-Pointer	Zeiger auf die Daten (in/out)
INSTANZ	DB	Instanz-DB des Ringpuffers
FUNC	Char	Auszuführende Operation (S: Status; O: Overwrite; W: Write; R: Read; C: Clear)
STAT	Word	Status des Ringpuffers (0: keine Meldung; 1: Overflow beim Schreiben; 2: Puffer voll; 3: Puffer leer; 4: Fehler bei BlkMov)
Entries	Int	Anzahl der Einträge im Ringpuffer
ACT	Bool	Der Ringpuffer wird nur mit ACT=TRUE aktiv

7.2.3 Empfangseinheit(en)

Aufgaben des Funktionselements

Eine Empfangseinheit soll mittels BRCV bzgl. einer fest verschalteten Verbindung ständig Telegramme empfangen.

Sobald ein Telegramm vollständig empfangen wurde, sollen die Daten in den Ringpuffer der jeweiligen Sendeeinheit eingetragen werden. Dadurch wird das Ziel-Routing realisiert (Ein Übersichtbild finden Sie in Kap. 4.5 oder 4.6.10).

Notwendige Datenstrukturen für das Funktionselement

Das Funktionselement benötigt die Datenstruktur des Telegramms, die auch das Funktionselement „Sendebaustein“ benötigt. (siehe dazu Kap. 7.1.4).

Die Empfangseinheit analysiert dabei nur die **Zielinformation** des Datentelegramms, welche sich in der Kopfstruktur eines jeden Telegramms befindet. Zur Struktur siehe Kap. 7.1.4 bei „Notwendige Datenstrukturen“

Mechanismus der Funktionseinheit

Alle Empfangseinheiten, also alle Kommunikationskanäle, werden in einem Zyklus vom FC1 „Manager“ über eine Schleife aufgerufen (s. Kap. 7.2.1). Dadurch werden in einem Zyklus alle ankommenden Daten bearbeitet und geroutet.

In jeder Empfangseinheit findet folgender Ablauf statt:

Tabelle 7-8

Schritt	Aktion	Bemerkung
1	Erstelle mit dem Pointergenerator einen Any-Pointer auf das Empfangsfach	Die Datenbausteinnummer des Empfangsfaches wird durch einen Eingabeparameter am Baustein angegeben.
2	Solange keine Daten empfangen sind, rufe BRCV auf	Die Empfangseinheit empfängt auf ihrem Kommunikationskanal (gegeben durch den Parameter ID) ständig Daten. Durch das NDR-Bit des BRCV-Bausteins wird der Empfang von Daten signalisiert. Im Fehlerfall wird der Fehler protokolliert.
3	Analysiere Datentelegramm	Die Empfangseinheit liest den Kopf des Datentelegramms aus, um das Ziel des Datentelegramms feststellen zu können.
4	Trage Datentelegramm in den Ringpuffer ein	Da es eine 1-zu-1-Verbindung zwischen Ringpuffer, Sendeeinheit und Zielstation gibt, wird hiermit der Routing-Mechanismus durchgeführt.

Folgende Codeauszüge zeigen die einzelnen Schritte:

- Schritt 1:
  Der Pointergenerator erstellt einen Any-Pointer auf den angegebenen Datenbaustein (FB4 „ReceiveUnit“, Netzwerk 1).
- Schritt 2:
  Der zur Empfangseinheit gehörende BRCV-Baustein wird über eine switch-case-Struktur ausgewählt. Welche Empfangseinheit nun empfangen soll, hat bereits der überlagerte Manager-Baustein FC1 „Manager“ entschieden (s. Kap. 7.2.1).

```
Network 2 : Receiving
Receiving the data with BRCV

    // start receiving

          L     #nInUnit
          JL    swit
          JU    dumm                        // the first unit starts with number '1', not with number '0'
          JU    one
          JU    two
          JU    thre
          JU    four
          JU    five
          JU    six
          JU    seve
          JU    eigh
    swit: JU    end
    dumm: NOP   0
          L     "GlobConstants".E_INVALIDARG
          T     #wOutStatus
          JU    end
    one:  NOP   0
          CALL  "BRCV300" , "BRCVData_IDB1"
           EN_R  :=TRUE
           ID    :=W#16#1                  // First unit -> ID=1
           R_ID  :=DW#16#1                 // R_ID is fixed on "1", because we want to receive a data telegram
           NDR   :=#bNDR
           ERROR :=#bOutError
           STATUS:=#wOutStatus
           RD_1  :=#anyTmpRcv
           LEN   :=#wLen
          JU    end
    two:  NOP   0
          CALL  "BRCV300" , "BRCVData_IDB2"
           EN_R  :=TRUE
           ID    :=W#16#2                  // Second unit -> ID=2
           R_ID  :=DW#16#1                 // R_ID is fixed on "1", because we want to receive a data telegram
           NDR   :=#bNDR
```

Bild 7-13

- Schritt 3:
  Die Analyse des Telegrammkopfes erfolgt über einen einfachen Blkmov.
- Schritt 4:
  Über den FB5 „Buffermanager“ wird das Datentelegramm in den richtigen Ringpuffer eingetragen.
  Achten Sie auf den Parameter `nInUnit` !
  Hierdurch wird das Routing realisiert.

```
// Insert the telegram into the buffer of the destination
      CALL  "BufferManager" , DB5
       cInMode          :='W'
       nInUnit          :=#tTmpRcvHeader.tDestInfo.nLocal_ID
       wOutBufferStatus :=#wOutBufStat
       bOutError        :=#bOutBufErrWriting
       nOutBufferEntries:=
       anyIOData        :=#anyTmpRcv

End:  BE
```

Hier findet das Routing statt:
Die Zielinformation im Telegrammkopf bestimmt, in welchen Ringpuffer das Telegramm geschrieben wird.

Bild 7-14

Parameter des Funktionselements

s. Kap. 4.6.10.

7.2.4 Sendeeinheit(en)

Aufgaben des Funktionselements

Die Sendeeinheit stellt eine 1-zu-1-Verbindung zur Zielstation dar.

Sie hat die Aufgabe, Daten aus ihrem Ringpuffer an ihre Zielstation weiterzuleiten und das Quittungstelegramm in den Ringpuffer der Sendeeinheit der Quellstation einzutragen (s. Kap. 4.6.11 oder Übersichtsbild in Kap. 4.5.1).

Notwendige Datenstrukturen für das Funktionselement

Das Funktionselement benötigt die gleichen Datenstrukturen wie die Empfangseinheit(en) (s. Kap.7.2.3).

Die Sendeeinheit analysiert dabei nur die **Quellinformation** des Datentelegramms.

Flussdiagramm der Sendeeinheit(en)

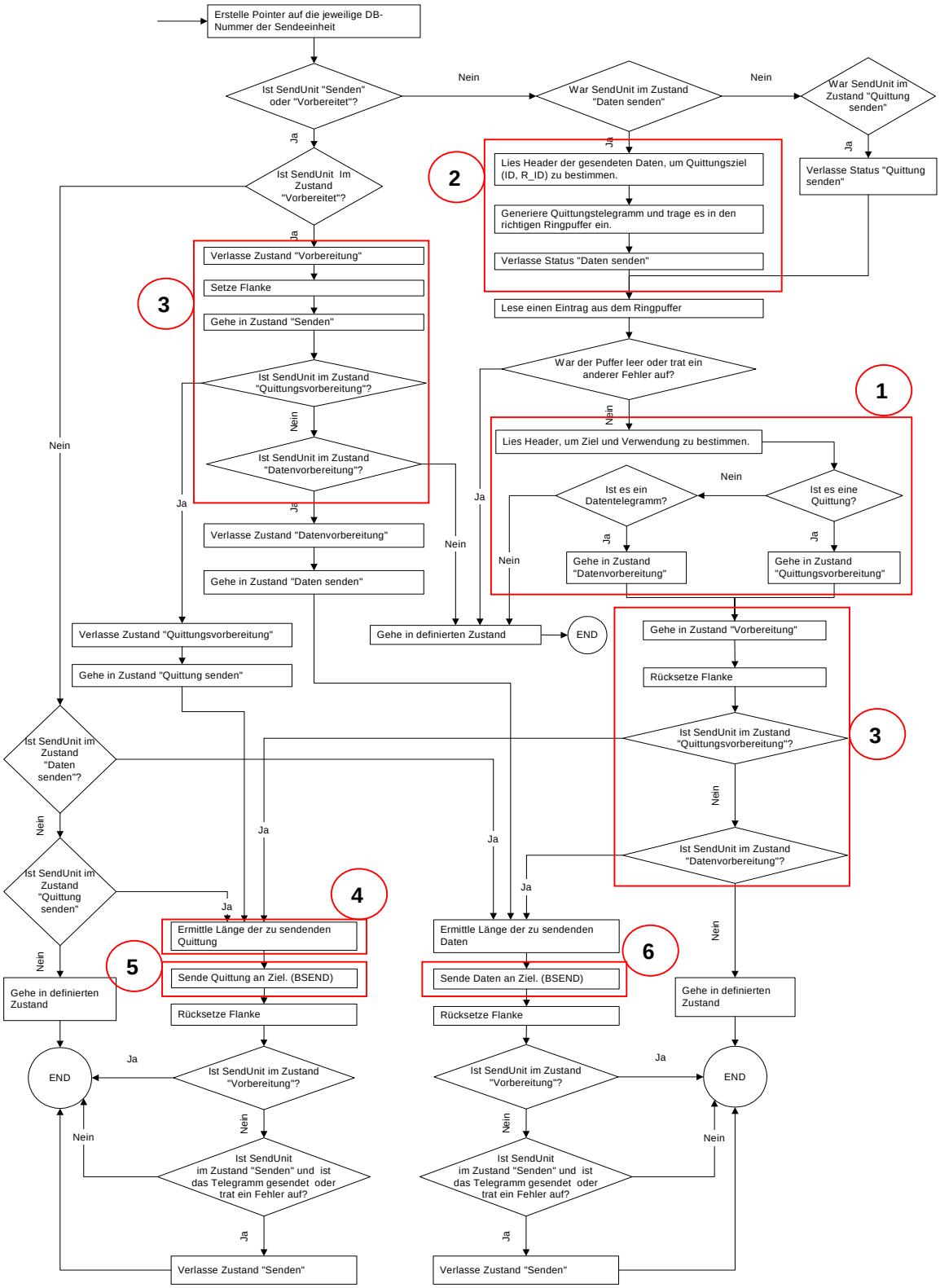


Bild 7-15 Flussdiagramm der Sendeeinheit im Gateway

Erläuterung zur Sendeeinheit(en)

Hinweis

In Kap. 4.6.11wird der Ablauf vereinfacht anhand eines Zustandsdiagramms aufgezeigt.

Da BSEND/BRCV-Bausteine feste ID und R_ID-Parameter benötigen und im Gateway 3 Vorgänge (Empfangen von Daten, Senden von Daten, Senden von Quittungen) statt finden, müssen auch 3 verschiedene R_ID-Parameter festgelegt werden. Diese sind:

- 1: es werden Daten empfangen
- 2: es werden Daten gesendet
- 3: es werden Quittungen gesendet

Diese R_ID-Parameter sind mit dem jeweiligen BSEND/BRCV-Baustein fest verschaltet (s. Kap. 4.4.4 und 4.5.1).

Es werden nun exemplarisch Codeausschnitte dargestellt, die Teilfunktionalitäten der Sendeeinheit realisieren:

1. Analyse des Kopfes:
   Um feststellen zu können, welcher BSEND-Aufruf (also einer mit R_ID=2 oder einer mit R_ID=3) zu wählen ist, muss bestimmt werden, ob es sich um ein Daten- oder ein Quittungstelegramm handelt.
   Dazu wird das Datum, das sich aktuell im Sendefach des BSEND-Bausteins befindet, in eine Kopfstruktur kopiert und der Telegrammtyp „byType“ untersucht.
   Je nach Telegrammtyp werden entsprechende Merker gesetzt. (FB7 „SndUnit“, Netzwerk 7)
2. Generierung eines Quittungstelegramms:
   Um ein Quittungstelegramm zu erstellen, wird der Kopf des zuletzt gesendeten Datentelegramms verändert.
   Es wird der Parameter „byType“ des Kopfes von 1 auf 2 gesetzt.
   Weiterhin wird der Status des letzten Daten-BSEND-Aufrufes in den Parameter “wStatus“ des Kopfes geschrieben. Dadurch wird es der Quellstation ermöglicht, aufgetretene Fehler beim BSEND-Aufruf auszulesen.
   Schließlich wird das so angepasste Telegramm über den Buffermanager in den Ringpuffer der Sendeeinheit eingetragen, die der Quellstation zugeordnet ist. Achten Sie dazu auf den Parameter `nInUnit` beim Aufruf des FB5 „Buffermanager“:

```
Network 4 : Handle acknowledgement

Handles the acknowledgement by generating a telegram and putting it into the
corresponding ringbuffer.

    // Read header of sent data.
    // The sent data is given by the send pocket.
    // Copy this data to the header struct.
          CALL  "BLKMOV"
           SRCBLK :=#anyTmpSndData
           RET_VAL:=#nOutBlkMovStat
           DSTBLK :="HeadDB".Head
          L     0
          L     #nOutBlkMovStat
          <>I
          JC    def

    // Generate acknowledge telegram:
    // Sign the telegram to be an acknowledgement
    // To do this, switch sender and destination.
          L     "GlobConstants".TELTYPE_ACK
          T     "HeadDB".Head.byType
    // Saving Error-Code
    // First: Reset error code
    // Second: Set error code, if necessary
          L     0
          T     "HeadDB".Head.wError
          SET
          A     #bOutBSENDError
          JCN   srou
          L     #wOutBSENDStat
          T     "HeadDB".Head.wError

    srou: L     "GlobConstants".ROUTING_FINISHED
          T     "HeadDB".Head.wStatus

          CALL  "BufferManager" , DB5
           cInMode          :='W'
           nInUnit          :="HeadDB".Head.tSourceInfo.nLocal_ID    // Destination is sender of original data
           wOutBufferStatus :=#wOutBufStatus
           bOutError        :=#bOutError
           nOutBufferEntries:=
           anyIOData        :="HeadDB".Head // Data is only the head

    // Change the state independant from errors.
          SET
          R     #bSndData

    // continue with next entry in ringbuffer
          JU    snd2
```

Quell-Routing: Das Telegramm wird an die Quelle gesendet.

Bild 7-16

3. Versorgung des BSEND-Bausteins mit einer steigenden Flanke:
   Um den BSEND-Baustein mit einer steigenden Flanke zu versorgen, muss er zunächst mit REQ:=0 und danach mit REQ:=1 aufgerufen werden. Die Flankenauswertung übernimmt der BSEND-Baustein.
   Um dieses Prozedere gewährleisten zu können, verfügt die Sendeeinheit über die Zustände „bIsPrepared“ und „bIsSending“. Je nach Signalzustand wird der BSEND-Baustein über die Variable „bEdge“ nun mit REQ:=0 („bIsPrepared:=TRUE und bIsSending:=FALSE“) oder REQ:=1 („bIsPrepared:=FALSE und bIsSending:=TRUE“) aufgerufen.
   Um weiterhin unterscheiden zu können, welche Art von BSEND-Baustein (Quittung oder Daten) gerade bearbeitet werden soll, werden die Variabeln „bPrepAck“ (Acknowledge) und „bPrepData“ benötigt, die bei der Kopfanalyse entsprechend gesetzt wurden:

```
Network 8 : Rising edge - "prepare state"
1. We have to prepare the BSEND and set REQ to 0 once.
2. The next cyclic call we change from "prepare" to sending by calling BSEND
with REQ=1. So we got a rising edge.

    pre0: NOP   0
          SET
          S     #bIsPrepared
          R     #bEdge

          A     #bPrepAck
          JC    ack1

          SET
          A     #bPrepData
          JC    dat1

    // This case should never occur, because of network7. To be sure -> quit
          SET
          S     #bOutError
          JU    end

    pre1: NOP   0
          SET
          R     #bIsPrepared
          S     #bEdge
          S     #bIsSending

          SET
          A     #bPrepAck
          R     #bPrepAck
          S     #bSndAck
          JC    ack1

          SET
          A     #bPrepData
          R     #bPrepData
          S     #bSndData
          JC    dat1

    // This case should never occur, because of network7. To be sure -> quit
          SET
          S     #bOutError
          JU    end
```

Bild 7-17

4. Anpassen eines Any-Pointers
   Das Sendefach wird durch einen Any-Pointer adressiert.
   Da die Länge der zu sendenden Daten je nach BSEND-Typ (S7-300 oder S7-400) von der Bereichslängenangabe des Any-Pointers bzw. der LEN-Angabe des BSEND-Parameters abhängt, müssen diese angepasst werden.

```
Network 9 : Length/Pointer adjustment
The Length and the any pointer need to be adjusted, if it is an
acknowledgement,
which is to send.
Otherwise only get the length of the send DB.

    // It is an acknowledge telegram
    ack1: NOP   0

    // Adjust length of send pocket
    // The send pointer has only header length
    // Load any-pointer
          L     LD     0
    // Zero the fourth tetrade
          AD    DW#16#FFFFFF00
    // Set length to "wInHeaderLen"
          L     #wInHeaderLen
          AD    DW#16#FFFF
          +D
          T     LD     0

    // set len parameter
          L     #wInHeaderLen
          T     #wLen

          JU    sAck
```

Bild 7-18

Hinweis

Das Format des Any-Pointers entnehmen Sie bitte der Step7-Online-Hilfe.

5. Senden von Daten:
   Der folgende Codeauszug zeigt den Aufruf von 2 BSEND-Bausteinen. Diese senden Daten (wg. R_ID:=2) zum jeweiligen Ziel (gegeben durch ID:=...).
   In diesem Beispiel wird das Ziel also entweder über Local ID := 1 oder Local ID := 2 erreicht. Die Auswahl, ob nun die Sendeeinheit 1 oder 2 aktiv sein soll, trifft der FC1 „Manager“ (dieser ruft alle Sendeeinheiten innerhalb einer Schleife auf; s. Kap. 4.6.2).
   An diesem Beispiel erkennt man auch die 1-zu-1-Zuordnung einer Sendeeinheit zur Quell/-Zielstation, denn Sendeeinheit 1 ruft den BSEND-Baustein mit ID := 1 auf, Sendeeinheit 2 ruft den BSEND-Baustein mit ID := 2 auf. Dies ist für insgesamt 8 IDs implementiert:

```
sDat: NOP   0

// start sending
      L     #nInUnit
      JL    swil
      JU    duml                        // the first unit starts with number '1', not with number '0'
      JU    onel
      JU    twol
      JU    thrl
      JU    foul
      JU    fivl
      JU    sixl
      JU    sevl
      JU    eigl
swil: JU    endl
duml: NOP   0
      SET
      S     #bOutError
      JU    endl
onel: NOP   0
      CALL  "BSEND300" , "BSEND_IDB1"
       REQ   :=#bEdge
       R     :=FALSE
       ID    :=W#16#1                  // SendUnit 1 => Channel 1
       R_ID  :=DW#16#2                 // Sending data always uses R_ID=2
       DONE  :=#bDone
       ERROR :=#bOutBSENDError
       STATUS:=#wOutBSENDStat
       SD_1  :=#anyTmpSndData
       LEN   :=#wLen

      JU    endl
twol: NOP   0

      CALL  "BSEND300" , "BSEND_IDB2"
       REQ   :=#bEdge
       R     :=FALSE
       ID    :=W#16#2                  // SendUnit 2 => Channel 2
       R_ID  :=DW#16#2                 // Sending data always uses R_ID=2
       DONE  :=#bDone
```

Bild 7-19

6. Senden von Quittungen:
   Das Senden von Quittungen erfolgt analog zum Senden von Daten (Pkt. 5), jedoch mit dem Unterschied, dass die R_ID := 3 ist.

Parameter des Funktionselements

s. Kap. 4.6.11

8 Veränderungen im STEP7-Programm

Einleitung

Nachfolgend ist aufgeführt, welche Schritte Sie unternehmen müssen, um die Applikation Ihren Anforderungen anzupassen.

Inhalt Kapitel Veränderungen im STEP7-Programm

Tabelle 8-1

Kap.-Nr.	Kapitelüberschrift	Hier erfahren Sie...
8.1	Änderung der Nutzdatenlänge	... wie Sie vorgehen sollten, um die Nutzdatenlänge Ihren Anforderungen anzupassen
8.2	Hinzufügen weiterer Quell-/Zielstationen	... wie Sie vorgehen sollten, um weitere Quell- und Zielstationen dem Routingmechanismus hinzuzufügen
8.3	Ändern der Gateway-CPU in eine S7-400	... wie Sie vorgehen sollten, um die Gateway-CPU der vorhandenen Applikation durch eine S7-400-Gateway-CPU auszutauschen.
8.4	Hinzufügen von zusätzlichem Fehlerhandling	... wie Sie vorgehen sollten, um weiteres Fehlerhandling zu implementieren.

Vorraussetzung

Kenntnisse der Teile A1 respektive A2 sind notwendig.

8.1 Änderung der Nutzdatenlänge

Hinweis

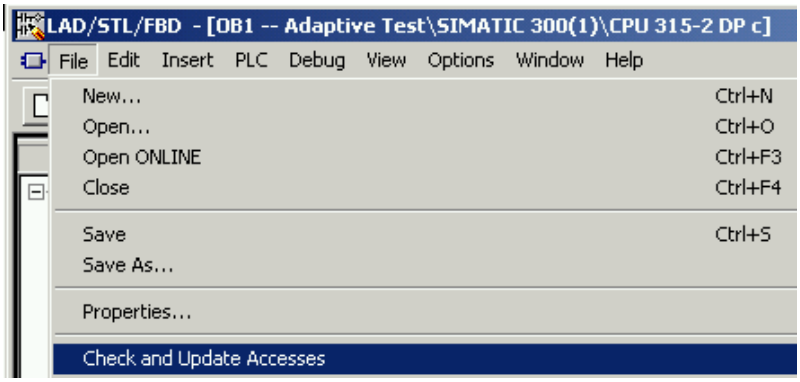
Beachten Sie bitte die Leistungseckdaten in Kap. 3.3.

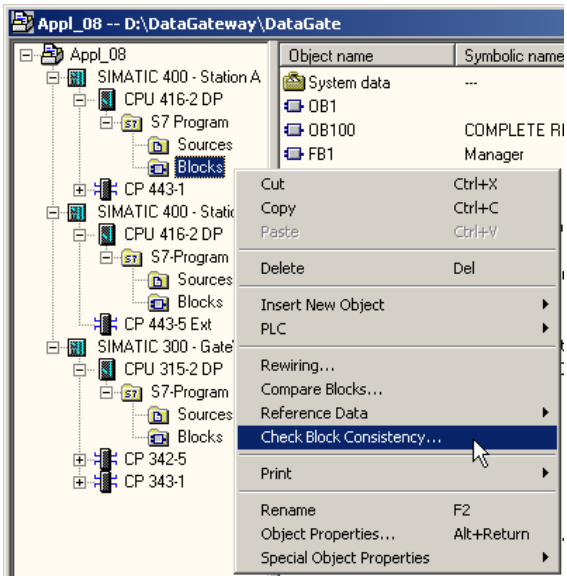
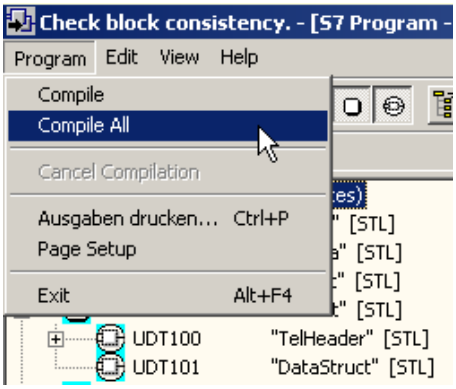
Um die Nutzdatenlänge ändern zu können, müssen Sie folgendermaßen vorgehen:

Quell-/Zielstation

Folgende Schritte müssen in jeder Quell-/Zielstation durchgeführt werden:

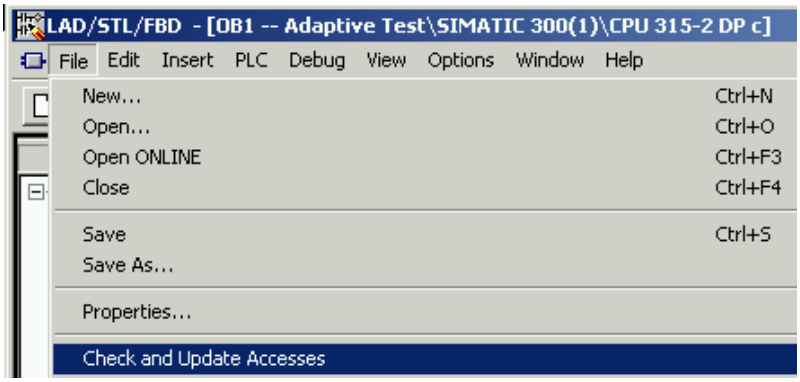
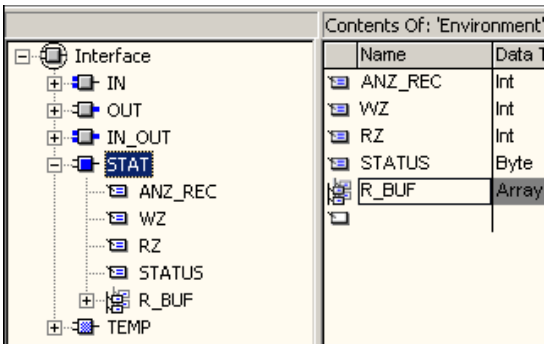
Tabelle 8-2

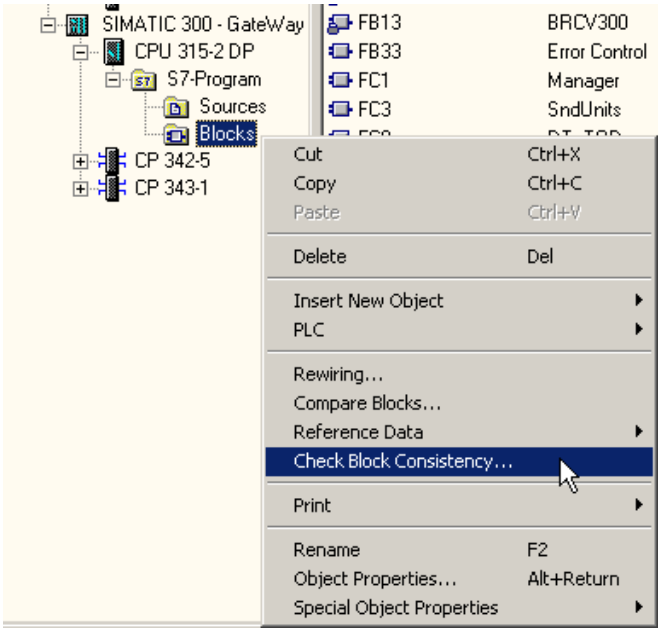
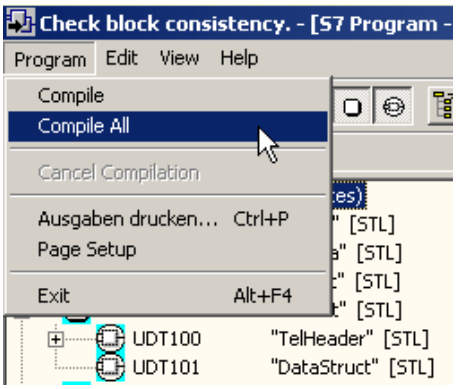
Schritt	Beschreibung
1	Tauschen Sie den UDT 101 „DataStruct“ durch ihren eigenen Datentyp aus
2	Öffnen Sie den UDT102 „SendStruct“ und betätigen die Funktion „Datei > Zugriffe prüfen und aktualisieren“ 
3	Löschen Sie den DB4 „UserData“.
4	Erstellen Sie den DB4 „UserData“ auf Grundlage des UDT101 „DataStruct“.

Schritt	Beschreibung
5	Lassen Sie die Bausteinkonsistenz prüfen. 
6	Übersetzen Sie alle Bausteine.  **Hinweis:** Sie können die Warnungen ignorieren.
7	Stoppen Sie die Quell-/Zielstation und laden Sie das Programm mit „Zielsystem → Laden“

Gateway

Tabelle 8-3

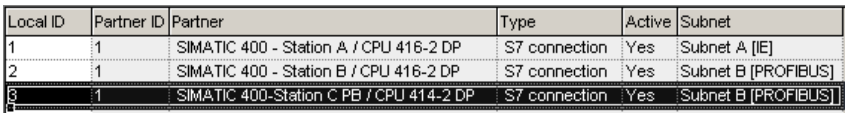
Schritt	Beschreibung
1	Tauschen Sie den UDT 101 „DataStruct“ durch ihren eigenen Datentyp aus
2	Öffnen Sie den UDT102 „TelStruct“ und betätigen die Funktion „Datei > Zugriffe prüfen und aktualisieren“ 
3	Falls Sie SCL V5.1 installiert haben, so fahren Sie mit Schritt 7 fort.
4	Öffnen Sie den Baustein FB2 „Ringbuffer“.
5	Löschen Sie die Variable R_BUF aus dem Datenbereich „STAT“. 
6	Legen Sie eine Variable R_BUF als Array[0..9] vom Typ UDT 102 an und speichern Sie den Baustein. Fahren Sie mit Schritt 9 fort.
7	Öffnen Sie die SCL-Quelle „Statistic_Streng_Trenner“.
8	Kompilieren Sie den Baustein.

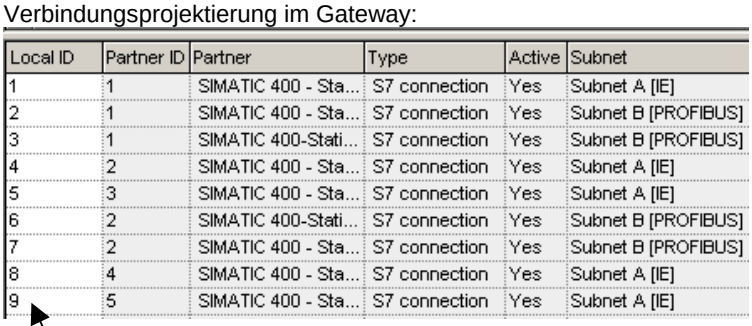
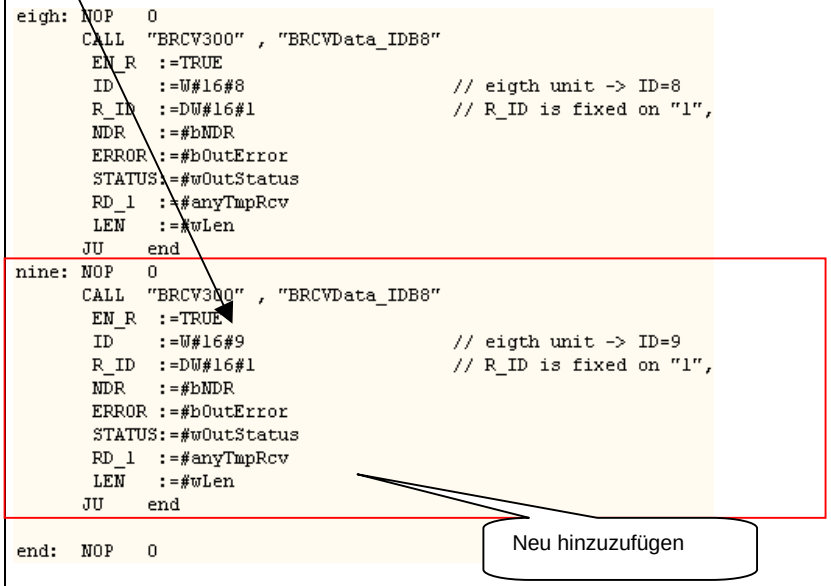
Schritt	Beschreibung
9	Lassen Sie die Bausteinkonsistenz prüfen. 
10	Übersetzen Sie alle Bausteine.  **Hinweis:** Sie können die Warnungen ignorieren.
11	Stoppen Sie das Gateway und laden Sie es mit „Zielstation → Laden“

8.2 Hinzufügen weiterer Quell-/Zielstationen

Um neue Quell/Zielstationen hinzuzufügen, müssen Sie folgendermaßen vorgehen (es wird weder auf die Bedienung des SIMATIC Managers noch auf den Hardwareaufbau eingegangen):

Tabelle 8-4

Schritt	Beschreibung
1	Stellen Sie alle CPUs auf STOP.
2	Fügen Sie über den SIMATIC Manager die neuen Stationen und deren Hardwarekonfiguration über HWKonfig ein.
3	Stellen Sie in den neuen Stationen die Uhrzeitsynchronisation gemäß Kap. 5.3.2 ein. Projektieren Sie die neuen S7-Verbindungen in NetPro (s. dazu Kap. 5.3.1) und übersetzen Sie alles neu. Achten Sie darauf, dass die Local IDs der S7-Verbindungen im Gateway lückenlos in aufsteigender Form sind. Als Beispiel wird die Verbindungsprojektierung mit einer neuen Station gezeigt: • Verbindungsprojektierung in der neuen Station:  • Verbindungsprojektierung im Gateway:  Das Gateway unterstützt in der derzeitigen Version max. 8 Quell-/Zielstationen.
4	Kopieren Sie die Bausteine im Bausteinordner der Station A in den Bausteinordner der neuen Quell-/Zielstationen. **ACHTUNG:** Kopieren Sie **nicht** die Systemdaten! Anderenfalls müssen Sie die Hardwarekonfiguration über NetPro noch einmal übersetzen.
5	Falls die neu hinzugekommenen Quell-/Zielstationen vom Typ S7-400 sind, so fahren Sie bitte mit Schritt 8 fort.
6	Löschen Sie aus dem Bausteinordner der neu hinzugekommenen Stationen die Bausteine „BSEND“ und „BRCV“ sowie deren Instanz-Datenbausteine.
7	Fügen Sie in den Bausteinordner der hinzugekommenen S7-300-Stationen die Bausteine „BSEND300“ und „BRCV300“ hinzu. **Hinweise:** • Die Bausteine „BSEND300“ und „BRCV300“ können Sie dem Bausteinordner des Gateways entnehmen. • Die S7-300-Stationen **müssen** die S7-Kommunikation mittels BSEND/BRCV unterstützen!
8	Öffnen Sie in den neuen Quell-/Zielstationen den FB25 „SendBlock“ und den FB1 „Manager“.
9	Passen Sie den Parameter ID des BSEND-Aufrufes an, indem Sie hier die Local ID von dieser CPU zum Gateway eintragen. Gehen Sie analog mit dem BRCV-Aufruf vor (im FB25 und im FB1).
10	Betätigen Sie die Funktion „Datei > Zugriffe prüfen und aktualisieren“. Lassen Sie dabei, falls nötig, die neuen Instanz-Datenbausteine generieren.

Schritt	Beschreibung
11	Wiederholen Sie Schritt 4 für alle neu hinzugefügten CPUs. Falls Sie nicht mehr als 8 Quell-/Zielstationen benutzen wollen, gehen Sie zu Schritt 21.
12	Öffnen Sie im Gateway den Baustein FB4 „ReceiveUnit“.
13	Fügen Sie im switch-case-Teil die noch fehlenden BRCV-Aufrufe ein. Achten Sie dabei auf die Parameter für „ID“ und „R_ID“. Die jeweilige ID entnehmen Sie NetPro. Dort sind durch Schritt 3 neue Local_IDs hinzugekommen. Diese dienen als ID-Parameter für den jeweiligen BRCV-Aufruf. Beispiel: Verbindungsprojektierung im Gateway:  Anpassung im FB4 „ReceiveUnit“ im Netzwerk2: 
14	Öffnen Sie im Gateway den Baustein FB3 „Bufferhandler“.
15	Fügen Sie im switch-case-Teil (Netzwerk 2)die noch fehlenden Aufrufe für den Ringpuffer ein.
16	Öffnen Sie im Gateway den Baustein FC3 „SndUnits“.
17	Fügen Sie im switch-case-Teil (Netzwerk 1) die noch fehlenden Aufrufe für den FB7 „SndUnit“ ein.
18	Öffnen Sie im Gateway den Baustein FB7 „SndUnit“.

Local ID	Partner ID	Partner	Type	Active	Subnet
1	1	SIMATIC 400 - Sta...	S7 connection	Yes	Subnet A [IE]
2	1	SIMATIC 400 - Sta...	S7 connection	Yes	Subnet B [PROFIBUS]
3	1	SIMATIC 400-Stati...	S7 connection	Yes	Subnet B [PROFIBUS]
4	2	SIMATIC 400 - Sta...	S7 connection	Yes	Subnet A [IE]
5	3	SIMATIC 400 - Sta...	S7 connection	Yes	Subnet A [IE]
6	2	SIMATIC 400-Stati...	S7 connection	Yes	Subnet B [PROFIBUS]
7	2	SIMATIC 400 - Sta...	S7 connection	Yes	Subnet B [PROFIBUS]
8	4	SIMATIC 400 - Sta...	S7 connection	Yes	Subnet A [IE]
9	5	SIMATIC 400 - Sta...	S7 connection	Yes	Subnet A [IE]

```
eigh: NOP   0
      CALL  "BRCV300" , "BRCVData_IDB8"
       EN_R  :=TRUE
       ID    :=W#16#8                    // eigth unit -> ID=8
       R_ID  :=DW#16#1                   // R_ID is fixed on "1",
       NDR   :=#bNDR
       ERROR :=#bOutError
       STATUS:=#wOutStatus
       RD_1  :=#anyTmpRcv
       LEN   :=#wLen
      JU    end
nine: NOP   0
      CALL  "BRCV300" , "BRCVData_IDB8"
       EN_R  :=TRUE
       ID    :=W#16#9                    // eigth unit -> ID=9
       R_ID  :=DW#16#1                   // R_ID is fixed on "1",
       NDR   :=#bNDR
       ERROR :=#bOutError
       STATUS:=#wOutStatus
       RD_1  :=#anyTmpRcv
       LEN   :=#wLen
      JU    end

end:  NOP   0
```

Neu hinzuzufügen

Schritt	Beschreibung
19	Fügen Sie im switch-caseTeil sowohl beim Senden von Daten als auch beim Senden von Quittungen die noch fehlenden BSEND-Aufrufe ein (Netzwerke 10 und 11). Achten Sie dabei auf die Parameter ID (diese können Sie wiederum der Verbindungsprojektierung in NetPro entnehmen) und R_ID (siehe auch Schritt 13).
20	Speichern und übersetzen Sie alle veränderten Bausteine.
21	Laden Sie alle Stationen.
22	Öffnen Sie die Variablentabelle VAT_C. Tragen Sie dort bei „nStations“ die neue Anzahl der zu verwaltenden Stationen ein.
23	Stellen Sie alle CPUs auf RUN oder RUN-P, beginnend mit der Gateway-CPU.
24	Öffnen Sie die Variablentabellen der Quell-/Zielstationen und belegen Sie dort die Parameter, wie es in Teil B in den Kapiteln 6.1 und 6.2 beschrieben ist. Hinweis: Um die symbolische Information zu erhalten, müssen Sie die Symboltabellen der neuen Quell-/Zielstationen aktualisieren.

8.3 Ändern der Gateway-CPU in eine S7-400

Tabelle 8-5

Schritt	Beschreibung
1	Stellen Sie alle CPUs auf STOP.
2	Erstellen Sie mit HWKonfig eine neue S7-400 Station inkl. aller Komponenten.
3	Kopieren Sie die Bausteine im Bausteinordner der Gateway-CPU in die neue S7-400-Station. **Achtung:** Kopieren Sie **nicht** die Systemdaten! Anderenfalls müssen Sie die Hardwarekonfiguration über NetPro noch einmal übersetzen.
4	Benennen Sie nun die S7-400-Station in „Gateway400“ um.
5	Löschen Sie die bisherige Gateway-Station.
6	Projektieren Sie die Verbindungen in NetPro wie in Kap. 6.2 beschrieben. Achten Sie darauf, dass im Gateway die IDs aufsteigend und lückenlos vergeben sind.
7	Übersetzen Sie die gesamte Hardwarekonfiguration.
8	Löschen Sie aus dem Bausteinordner des Gateway400 die Bausteine „BSEND300“ und „BRCV300“ sowie deren Instanz-Datenbausteine.
9	Fügen Sie dem Bausteinordner der Station Gateway400 die Bausteine „BSEND“ und „BRCV“ hinzu. **Hinweis:** Die Bausteine „BSEND“ und „BRCV“ können Sie dem Bausteinordner der Station A entnehmen.
10	Öffnen Sie im Gateway400 die Bausteine FB4 „ReceiveUnit“ und FB7 „SndUnit“.

Schritt	Beschreibung
11	Betätigen Sie die Funktion „Datei > Zugriffe prüfen und aktualisieren“. Lassen Sie dabei, falls nötig, die neuen Instanz-Datenbausteine generieren.
12	Laden Sie die Station Gatway400 über „Zielstation > Laden“
13	Belegen Sie die Variablentabelle VAT_C der Station Gateway400 wie es in Kap. 6.2 beschrieben ist.
14	Stellen Sie alle CPUs auf RUN.
15	Belegen Sie die Variablentabellen VAT_A und VAT_B der Stationen A und B wie es in Kap. 6.2 beschrieben ist.

8.4 Hinzufügen von zusätzlichem Fehlerhandling

8.4.1 Fehlerhandling in der Quell-/Zielstation

Auswertung des Zeitstempels des Quittungstelegramms

Tabelle 8-6

Nr	Beschreibung
1	Öffnen Sie in den Quell-/Zielstationen den Baustein FB1 „Manager“ oder den FB25 „SendBlock“.
2	Werten Sie den Zeitstempel des angekommenen Quittungstelegramms aus, indem Sie den zurückgelieferten Any-Pointer (oder das Quittungs-Empfangsfach) in eine Kopfstruktur umkopieren.
3	Unterscheidet sich der Zeitstempel des gesendeten Telegramms vom Zeitstempel des Quittungstelegramms, so ging entweder ein Telegramm verloren oder die Quittung wurde an eine andere Station gesendet.

Hinweis

Sollte dieser Fehler beobachtet werden, so kontrollieren Sie zunächst die Quell-/Zielinformationen.
Sind diese korrekt, so stellen Sie sicher, dass die Zuordnung der Sende- und Empfangseinheiten im Gateway sowie die Verbindungsprojektierung korrekt sind.

Ausführen der projektierten Fehlerreaktionen

Tabelle 8-7

Nr	Beschreibung
1	Öffnen Sie in den Quell-/Zielstationen den Baustein FB1 „Manager“ und FB25 „SendBlock“.
2	Werten Sie den Reaktionsparameter des FB33 „Error_Cont“ aus.
3	Fügen Sie dem FB25 „SendBlock“ Ausgabeparameter hinzu, die die Reaktionsparameter an den FB1 „Manager“ zurückliefern.

Nr	Beschreibung
4	Fügen Sie im FB1 „Manager“ den Code zum Ausführen der projektierten Reaktion hinzu. (z.B. Wiederholung des letzten Sendeauftrages)

Implementierung einer Zeitüberwachung

Wie bereits in Kap. 6.3 erwähnt, kann der Sendebaustein FB25 „Send-Block“ z.B. durch falsche Parametrierung der Quell-/Zielinformationen in einen blockierenden Zustand übergehen.

Um dies zu verhindern, sollte eine Zeitüberwachung implementiert werden, um den Sendebaustein über den Parameter „bIOReset“ wieder zurücksetzen und den Sendeauftrag wiederholen zu können.

Dabei können Sie folgendermaßen vorgehen:

Tabelle 8-8

Nr	Beschreibung
1	Öffnen Sie in den Quell-/Zielstationen den Baustein FB1 „Manager“.
2	Starten Sie vor dem Sendevorgang einen Timer
3	Sollte der Timer eine Grenzschwelle überschreiten, so setzen Sie den Sendebaustein FB25 „SendBlock“ mittels „bIOReset“ und „bIOBSEND_REQ“ wieder zurück.
4	Starten Sie nun den Sendeauftrag erneut, indem Sie „bIOReset“ zurücksetzen und eine positive Flanke auf „bIOBSEND_REQ“ legen.

Hinweis

Sollte die Zeitüberschreitung nachvollziehbar (also nicht sporadisch) auftreten, so kontrollieren Sie bitte die Kopfdaten – v.a. die Quell-/Zielinformation – und stellen Sie sicher, dass die Verbindungparametrierung und die Zuordnung der Sende- und Empfangseinheiten korrekt sind.

8.4.2 Fehlerhandling im Gateway

Im Gateway gibt der Großteil der implementierten Bausteine umfangreiche Fehlercodes aus, die lediglich ausgewertet werden müssen.

Folgende Beispiele zeigen ein mögliches Fehlerhandling.

Überprüfung des Programmablaufs

Tabelle 8-9

Nr	Beschreibung
1	Öffnen Sie im Gateway den FC1 „Manager“.

Nr	Beschreibung
2	Erstellen Sie eine Funktion, die einen entsprechenden Fehlercode in den Ringpuffer der Sendeeinheit der Quellstation einträgt. Dies können Sie z.B. durch Anpassen der Kopfstruktur durchführen. Markieren Sie dieses Telegramm als Quittung.
3	Fügen Sie nach jeder Fehlerabfrage innerhalb der Schleifen diese Funktion ein. Die Sendeeinheit wird sodann den im Ringpuffer vorhandenen Fehlercode als Quittung an die Quellstation zurückschicken, wo Sie dann auf den Fehler entsprechend reagieren können (dazu ist u.U. eine Anpassung des Reaktions-DBs im Element „Fehlerbehandlung“ nötig).

Überprüfung der Quell-/Zielinformationen

Tabelle 8-10

Nr	Beschreibung
1	Öffnen Sie im Gateway den Baustein FB4 „RcvUnit“.
2	Fügen Sie bei der Telegrammkopfanalyse folgendes hinzu: • Lesen Sie die Quell- **und** Zielinfomation aus • Führen Sie einen Range-Check durch; d.h. überprüfen Sie, ob das Gateway sowohl die Quell- als auch die Zielstation erreichen kann. Gehen Sie dazu folgendermaßen vor: - Erstellen Sie einen Datenbaustein, in dem Sie hinterlegen, welche Stationen bzw. IDs das Gateway unterstützt - Suchen Sie die ausgelesenen Quell- und Zielinformationen in diesem Datenbaustein • Ist der Range-Check negativ, so tragen Sie, falls die Quellinformation korrekt ist, über den FB5 „Buffermanager“ ein entsprechendes Telegramm in den Ringpuffer der Quellstation (markieren Sie dieses Telegramm vorher als Quittung!). Ist die Quellinformation ungültig, so können Sie die Quellinformation aus der Nummer der Empfangseinheit ableiten, falls es eine 1-zu-1-Zuordnung zwischen Empfangseinheit und Station gibt. **Hinweis:** In der vorliegenden Applikation besteht immer eine 1-zu-1-Zuordnung zwischen Empfangseinheit ←→ Quell-/Zielstation und Sendeeinheit ←→Quell-/Zielstation!
3	Die Sendeeinheit wird sodann den im Ringpuffer vorhandenen Fehlercode als Quittung an die Quellstation zurückschicken, wo Sie dann auf den Fehler entsprechend reagieren können (dazu ist u.U. eine Anpassung des Reaktions-DBs im Element „Fehlerbehandlung“ nötig).

9 Glossar

Any-Pointer

S7-Datentyp, der einen Operanden oder einen Zeiger auf eine beliebige Speicheradresse sowie zusätzlich auch den Datentyp des Operanden und einen Wiederholfaktor beinhaltet.

Baudrate

Unter der Baudrate versteht man die Geschwindigkeit, in der Daten zwischen Teilnehmern ausgetauscht werden. Diese Geschwindigkeit wird in Bit/s oder auch Baud gemessen.

Busprofil

Das Busprofil ist eine Sammlung von Parametern des PROFIBUS zur Festlegung von Systemeigenschaften. Hiermit wird zum Beispiel die Reaktionszeit der angesprochenen Partner genauso festgelegt wie die Anzahl der Wiederholversuche bis der angesprochene Partner als nicht erreichbar eingestuft wird.

CP

Communication Processor. Baugruppe für Kommunikationsaufgaben

Datenkonsistenz

Die Größe des Datenbereichs, der nicht gleichzeitig durch konkurrierende Prozesse verändert werden kann, wird als konsistenter Datenbereich bezeichnet. Ein Datenbereich, der größer als der konsistente Datenbereich ist, kann somit in seiner Gesamtheit verfälscht werden.

Deterministik

Die Deterministik beschreibt den Freiheitsgrad eines geschlossenen Systems. Es definiert im Fall des Profibus die Vorhersagbarkeit, wann ein bestimmter Teilnehmer wieder das Senderecht und damit die Möglichkeit der Übertragung seiner Daten hat.

Dienste

Angebotene Leistungen (Datenaustauschmöglichkeiten) eines Kommunikationsprotokolls.

FDL

Fieldbus Data Link, die Datensicherungsschicht (Schicht 2 des ISO/OSI-Referenzmodells) bei Profibus.

FIFO

Unter FIFO (FirstIn-FirstOut) versteht man eine Datenstruktur, aus der das zuerst geschriebene - und damit das „älteste“ - Datum ausgelesen wird.

ISO/OSI–Referenzmodell

Modell für Kommunikationsstandards, bestehend aus den 7 Schichten
1. Physikalische Schicht
2. Datenverbindungsschicht
3. Netzwerkschicht
4. Transportschicht
5. Sitzungsschicht
6. Darstellungsschicht
7. Anwendungsschicht

Protokoll

Ein Protokoll ist eine bit-genaue Vereinbarung zwischen Kommunikationspartnern, um einen bestimmten Kommunikationsdienst auszuführen. Das Protokoll definiert die inhaltliche Struktur des Datenverkehrs auf der physikalischen Leitung und legt z.B. die Betriebsart, die Vorgehensweise beim Verbindungsaufbau, die Datensicherung oder die Übertragungsgeschwindigkeit fest.

RFC 1006

Der RFC 1006 (RFC: „Request for comments“) spezifiziert ein Protokoll, welches auf TCP aufsetzt.
Der vollständige Titel des RFC 1006 lautet: ISO Transport Service on the top of the TCP.
Den RFC 1006 können Sie unter anderem einsehen unter:
http://www.rz.informatik.uni-muenchen.de/doku/rfc/rfc1006.html

Routing

Unter Routing wird im allgemeinen die Wegesuche von A nach B verstanden.
Im Automatisierungs- und IT-Umfeld spricht man vom Routing, wenn Datenpakete jedweder Art über Netzgrenzen hinweg von Station A nach Station B geleitet (geroutet) werden. Dabei ist nicht immer sicher gestellt, dass für die Daten der „kürzeste“ Weg gewählt wird.

Station

Eine Station wird durch eine PROFIBUS–Adresse am PROFIBUS identifiziert

10 Literaturangaben

Literaturangaben

Diese Liste ist keinesfalls vollständig und spiegelt nur eine Auswahl an geeigneter Literatur wieder.

Tabelle 10-1 Literaturliste

	Themengebiet	**Titel**
	Profibus Spezifikation	
\1\	EN 50170	http://www.profibus.com
	Industrial Ethernet	
\2\	Ethernet/IP	http://www.ad.siemens.de/industrial-ethernet/
	STEP7	
\3\	Handbuch „Systemsoftware für S7-300/400 System- und Standardfunktionen.	MLFB: 6ES7810-4CA06-8AR0
\4\	STEP 7 Online Hilfe	Teil der Step 7 Installation.
\5\		Automatisieren mit STEP7 in AWL und SCL Hans Berger, Publicis MCD Verlag ISBN 3-89578-113-4