SIEMENS

Anwendungsbeispiel • 02/2016

Master-Slave-Kommunikation über UDP-Broadcast

SIMATIC S7-1200/S7-1500

https://support.industry.siemens.com/cs/ww/de/view/20983558

Gewährleistung und Haftung

Hinweis

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

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

Unsere Haftung, gleich aus welchem Rechtsgrund, für durch die Verwendung der in diesem Anwendungsbeispiel beschriebenen Beispiele, Hinweise, Programme, Projektierungs- und Leistungsdaten usw. verursachte Schäden ist ausgeschlossen, soweit nicht z. B. nach dem Produkthaftungsgesetz in Fällen des Vorsatzes, der groben 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.

Security-hinweise

Siemens bietet Produkte und Lösungen mit Industrial Security-Funktionen an, die den sicheren Betrieb von Anlagen, Lösungen, Maschinen, Geräten und/oder Netzwerken unterstützen. Sie sind wichtige Komponenten in einem ganzheitlichen Industrial Security-Konzept. Die Produkte und Lösungen von Siemens werden unter diesem Gesichtspunkt ständig weiterentwickelt. Siemens empfiehlt, sich unbedingt regelmäßig über Produkt-Updates zu informieren.

Für den sicheren Betrieb von Produkten und Lösungen von Siemens ist es erforderlich, geeignete Schutzmaßnahmen (z. B. Zellenschutzkonzept) zu ergreifen und jede Komponente in ein ganzheitliches Industrial Security-Konzept zu integrieren, das dem aktuellen Stand der Technik entspricht. Dabei sind auch eingesetzte Produkte von anderen Herstellern zu berücksichtigen. Weitergehende Informationen über Industrial Security finden Sie unter http://www.siemens.com/industrialsecurity.

Um stets über Produkt-Updates informiert zu sein, melden Sie sich für unseren produktspezifischen Newsletter an. Weitere Informationen hierzu finden Sie unter http://support.industry.siemens.com.

Inhaltsverzeichnis

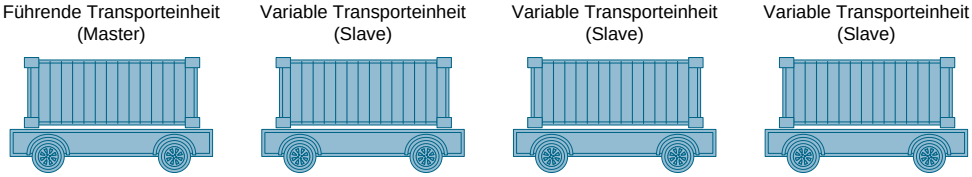
1 Aufgabe

Einleitung

In vielen Anwendungen besteht die Anforderung, mehreren Stationen schnell und gleichzeitig einen Datensatz schicken zu können.

Dies könnte zum Beispiel in einem Förder- und Transportsystem der Fall sein. So kann etwa bei frei fahrenden Transporteinheiten mit nur einem Telegramm eine quasi synchrone Reaktion initiiert werden. Ein Telegramm könnte die Information enthalten, alle Einheiten um 15° nach links zu lenken.

Abbildung 1-1



Nicht selten besteht dabei ein Zug aus variabel zusammengestellten Einheiten. Es soll also möglich sein, während der Laufzeit eine Einheit aus dem Zug auszukoppeln und eine andere Einheit einzukoppeln.

Anforderungen

- Übertragung von anwenderspezifischen Daten von einem Master an alle Slaves unter Nutzung eines Broadcasts
- Quittierung des Erhalts der Daten durch die Slaves an den Master
- Einheitliches Software-Paket für Master und Slaves
- Effiziente Übertragung der Daten
- Hinzufügen weiterer Stationen soll jederzeit möglich sein, ohne Änderungen an bereits vorhandenen Stationen durchführen zu müssen
- Jede Station kann während des laufenden Betriebs zur führenden Station werden (Master-Slave-Umschaltung)

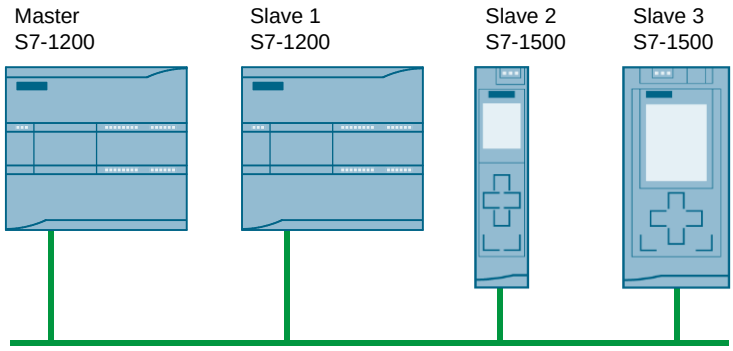
2 Lösung

2.1 Übersicht

Aufbau

Dieses Anwendungsbeispiel realisiert die Aufgabe mit SIMATIC Steuerungen S7-1200 und S7-1500, die über PROFINET miteinander verbunden sind:

Abbildung 2-1



Die Anzahl der teilnehmenden Steuerungen ist dabei beliebig.

Für die Kommunikation wird das UDP-Protokoll genutzt, da so die Verbindungen nicht statisch projektiert werden müssen, sondern programmgesteuert eingerichtet werden können (siehe auch Kapitel 3.3 „Verbindungen bei SIMATIC Steuerungen“). Dadurch müssen sich die teilnehmenden Stationen bei der Projektierung nicht gegenseitig bekannt sein und die Master-Slave-Umschaltung kann dynamisch erfolgen.

Vorteile

- Bereits vorhandene PROFINET-/Ethernet-Infrastrukturen können verwendet werden
- Keine zusätzliche Kommunikationsprozessoren notwendig, da die PROFINET-Ports der Steuerungen verwendet werden
- Flexible Topologien
- Einfache Erweiterbarkeit
- In STEP 7 beinhalteten Kommunikationsbausteinen (TUSEND bzw. TURCV) werden verwendet

Vorausgesetzte Kenntnisse

Folgende grundlegende Kenntnisse werden vorausgesetzt:

- Umgang mit SIMATIC Steuerungen
- Kenntnisse in STEP 7-Programmierung
- Grundlagen in Industrieller Kommunikation

2.2 Beschreibung der Kernfunktionalität

Einleitung

Über einen UDP-Broadcast versendet ein Master Daten an Slaves bzw. fordert Daten von Slaves an. Die Slaves quittieren den Erhalt des Broadcasts an den Master.

Dabei müssen die Teilnehmer bei der Projektierung nicht bekannt sein und können sich während des Betriebs ändern. Auch die Funktion des Masters kann während des Betriebs auf eine beliebige Steuerung übertragen werden.

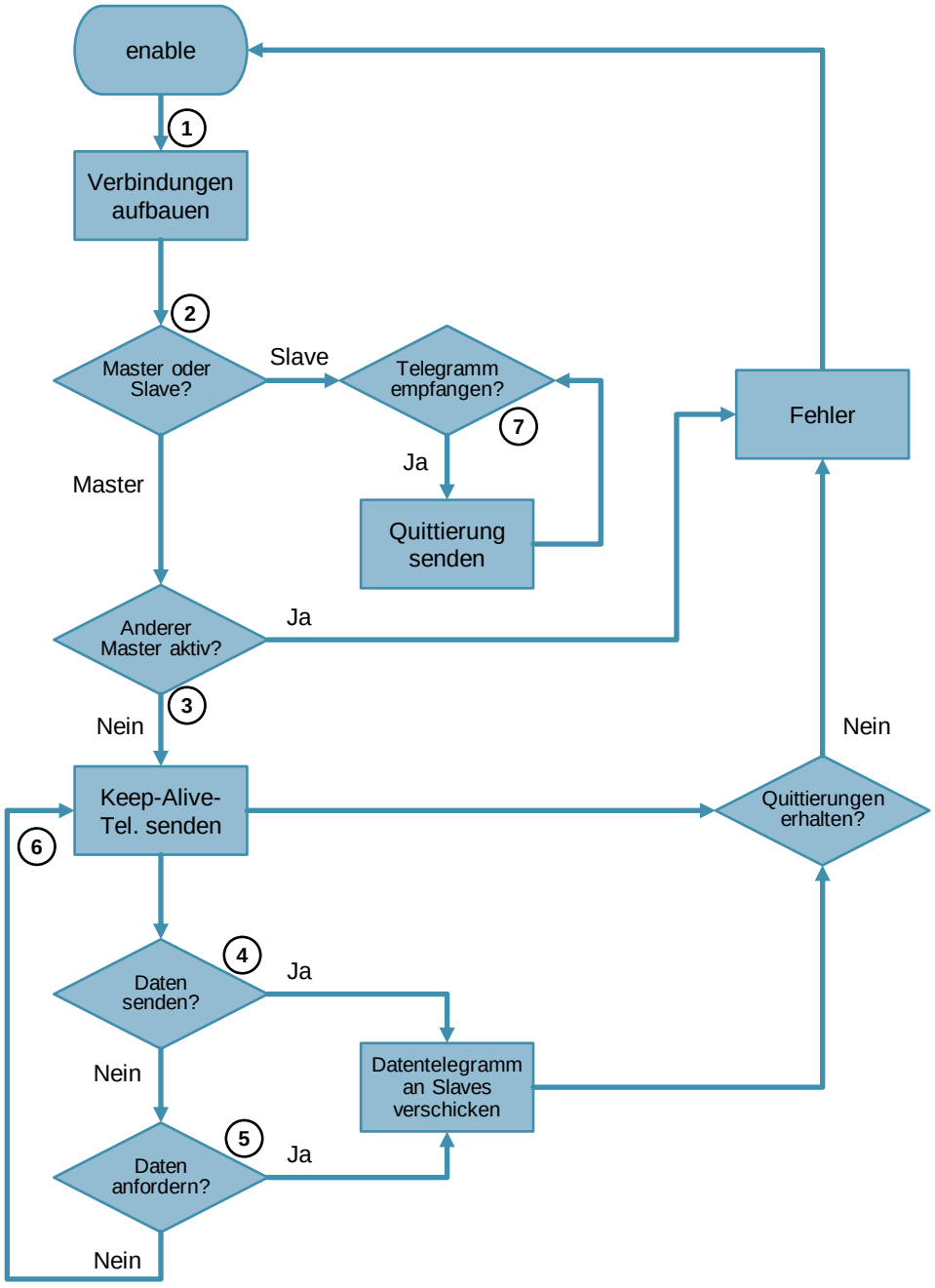
Der Master kann Daten an Slaves senden oder Daten der Slaves anfordern. Die Anwendung versendet dabei max. 236 Zeichen pro Sendeauftrag.

Stehen keine Sendeaufträge an, sendet der Master periodisch Abfragen, um sicherzustellen, dass alle Teilnehmer verfügbar sind. Diese Abfragen werden im Nachfolgenden als Keep-Alive-Telegramme bezeichnet.

Grafischer Programmablauf

Im Nachfolgenden wird der Ablauf des STEP 7-Programms erläutert.

Abbildung 2-2



Ablauf als Master

Tabelle 2-1

	Aktion	Hinweis
1.	Verbindungen werden aufgebaut	
2.	Wenn die Station die Funktion des Masters übernehmen soll, wird geprüft, ob ein anderer Teilnehmer bereits als Master aktiv ist.	Wird ein anderer Master im Netzwerk erkannt, wird ein Fehler ausgelöst. Dies hat keinen Einfluss auf den anderen Master
3.	Ist kein Master aktiv, wird ein Keep-Alive-Telegramm versandt.	
4.	Mit einem Send-Befehl werden Daten an die Slaves verschickt.	
5.	Mit einem Request-Befehl werden Daten von den Slaves angefordert.	
6.	Stehen weder Send- oder Request-Befehle an, werden periodisch Keep-Alive-Telegramme versandt.	

Hinweis Quittieren nicht alle bekannten Slaves rechtzeitig den Empfang eines Telegramms, wird ein Fehler ausgegeben. Der Sendebetrieb wird dadurch allerdings nicht unterbrochen.

Ablauf als Slave

Tabelle 2-2

	Aktion	Hinweis
1.	Verbindungen werden aufgebaut	
2.	Wenn die Station nicht die Funktion des Masters übernehmen soll, nimmt sie automatisch die Funktion eines Slaves an.	
7.	Wenn die Station ein Telegramm vom Master empfängt, wird dieses über ein Telegramm an den Master quittiert.	

2.3 Hard- und Software-Komponenten

2.3.1 Gültigkeit

Diese Applikation ist gültig für

- STEP 7 ab V13 SP1
- S7-1200 ab Firmware-Version 4.1, S7-1500

2.3.2 Verwendete Komponenten

Die Applikation wurde mit den nachfolgenden Komponenten erstellt:

Hardware-Komponenten

Tabelle 2-3

Komponente	Anz.	Artikelnummer	Hinweis
CPU 1214C	2	6ES7214-1AG40-0XB0	FW V4.1
CPU 1511	1	6ES7511-1AK00-0AB0	
CPU 1516	1	6ES7516-1AN00-0AB0	
SCALANCE XB008	1	6GK5008-0BA00-1AB2	
Stromversorgung 24 V	1	6EP1332-4BA00	70 W 120/230 V AC

Hinweis Mit den angegebenen Hardwarekomponenten wurde die Funktionalität getestet. Es können auch ähnliche, von obiger Liste abweichende Produkte verwendet werden.

Software-Komponenten

Tabelle 2-4

Komponente	Anz.	Artikelnummer	Hinweis
STEP 7 Professional	1	6ES7822-1AA03-0YA5	V13 SP1

Beispieldateien und Projekte

Die folgende Liste enthält alle Dateien und Projekte, die in diesem Beispiel verwendet werden.

Tabelle 2-5

Komponente	Hinweis
20983558_UDP_Broadcast_DOC_V21_de.pdf	Dieses Dokument
20983558_UDP_Broadcast_CODE_V21.zip	Das STEP 7-Projekt
20983558_UDP_Broadcast_LIB_V11.zip	Die für das Anwendungsbeispiel erstellte Bibliothek

3 Grundlagen

3.1 Grundbegriffe

Broadcast

Soll in einem Computernetzwerk ein Datenpaket von einem Punkt aus an alle Teilnehmer eines Netzwerks übertragen werden, spricht man von einem Broadcast (engl. „Rundruf“). Dabei müssen die Empfänger des Broadcast-Pakets nicht explizit angegeben werden.

Directed Broadcast

Das Ziel sind die Teilnehmer eines bestimmten Netzes. Die Adresse für einen directed broadcast in das Netz 192.168.0.0 mit der Netzmaske 255.255.255.0 lautet somit: 192.168.0.255.

Port

Ein Port ist der Teil einer Netzwerk-Adresse, der die Zuordnung von TCP- und UDP-Verbindungen und -Datenpaketen zu Server- und Client-Programmen durch Betriebssysteme bewirkt. Zu jeder Verbindung dieser beiden Protokolle gehören zwei Ports, je einer auf Seiten des Clients und des Servers.

Socket

IP-Adresse einer Station und Portnummer bilden zusammen einen sogenannten Socket, der als eindeutige Adresse des Anwenderprogramms im gesamten Netzwerk definiert ist.
Mit einem Socket kann also ein beliebiger Dienst eines Prozesses auf einer Station innerhalb eines Netzwerkes adressiert werden.

Unicast

Unicast bezeichnet die Übertragung von Nachrichten zwischen einem Sender und einem einzigen Empfänger.

3.2 UDP

Einleitung

Das UDP-Protokoll (User Datagram Protocol) wurde aus Gründen der Performance implementiert. Es bietet aufgrund des schlanken Telegrammkopfes vielfältige Nutzungsmöglichkeiten. Zudem wird das UDP-Protokoll von nahezu jeder Komponente im Industrial Ethernet unterstützt. Nachteilig wirkt sich die unquittierte Übertragung der Daten aus.

Umfang

Das UDP-Protokoll stellt aufgrund der Forderung, dass Daten schnell übertragen werden, lediglich grundlegende Funktionen zur Verfügung. Somit können mit minimalem Aufwand Daten zwischen kommunizierenden Parteien ausgetauscht werden. Auf Sicherungsmechanismen, wie sie bei TCP/IP vorhanden sind, wird dabei verzichtet.

Abbildung 3-1

IP-Adresse	UDP-Header	Datenbereich des UDP-Pakets

Außerdem ist das UDP-Protokoll verbindungslos und nicht stream-orientiert, d.h. Datenpakete werden am Stück gesendet.

Nachteile

Aufgrund der fehlenden Sicherheitsmechanismen entstehen Nachteile, die bei der Nutzung berücksichtigt und gegebenenfalls applikativ gelöst werden müssen:

- Verlorene Datenpakete werden nicht erneut gesendet.
- Datenpakete mit fehlerhafter Prüfsumme werden verworfen und nicht neu angefordert.
- Mehrfachzustellungen einzelner Pakete sind durch die Eigenschaften des IP-Protokolls als unterlagertes Protokoll möglich.
- Die Ankunftsreihenfolge der Daten beim Empfänger kann nicht vorhergesagt werden.

3.3 Verbindungen bei SIMATIC Steuerungen

Einleitung

Eine Verbindung definiert eine logische Zuordnung zweier Kommunikationspartner zur Ausführung von Kommunikationsdiensten. Eine Verbindung legt Folgendes fest:

- Beteiligte Kommunikationspartner
- Typ der Verbindung (z. B. S7-Verbindung)
- Spezielle Eigenschaften (z. B. ob eine Verbindung permanent aufgebaut bleibt oder ob sie im Anwenderprogramm dynamisch auf- und abgebaut wird und ob Betriebszustandsmeldungen gesendet werden sollen)
- Verbindungsweg

Projektierung von Verbindungen

Kommunikationsverbindungen können Sie projektieren oder unter bestimmten Voraussetzungen auch programmgesteuert einrichten.

Die Adressen des lokalen und des fernen Verbindungspartners werden durch die Vernetzung der Geräte vorgegeben und den Verbindungen bei der Projektierung zugeordnet.

Ausnahmen hiervon bilden folgende Verbindungen:

- Freie UDP-Verbindung: Bei der freien UDP-Verbindung wird die Adresse an der Kommunikationsschnittstelle im Anwenderprogramm angegeben.
- Programmierte Kommunikationsverbindung

Voll spezifizierte Verbindungen

Voll spezifizierte Verbindungen haben folgende Eigenschaften:

- Die Adressen und die Netzparameter des lokalen und des fernen Kommunikationspartners sind festgelegt.
- Die Kommunikationsverbindung ist nach dem Laden der Verbindungsparameter in das Gerät betriebsfähig.

Unspezifizierte (teilweise spezifizierte) Verbindungen

Unspezifizierte Verbindungen haben folgende Eigenschaften:

- Nur der lokale Kommunikationspartner ist festgelegt.

  Dies kann folgende Gründe haben:

  - Der Partner für den gewählten Verbindungstyp ist nicht vernetzt
  - Der Partner liegt außerhalb des Projekts
  - Der Partner soll wegen einer speziellen Handhabung nicht festgelegt werden
- Die Kommunikationsverbindung ist nach dem Laden der Verbindungsparameter in das Gerät bedingt betriebsfähig.

Anwendungsfälle:

- Die Hardware-Konfiguration ist noch unvollständig, so dass die vollständige Vernetzung der Geräte noch nicht möglich ist.
- Eine Empfangsbereitschaft für beliebige (unspezifizierte) Kommunikationspartner soll hergestellt werden.

Programmierte Kommunikationsverbindungen

Programmierte Kommunikationsverbindungen sind unspezifizierte bzw. teilweise spezifizierte Verbindungen.

In bestimmten Anwendungsbereichen ist es vorteilhaft, die Kommunikationsverbindungen nicht über die Projektierung statisch einzurichten. Alternativ besteht daher die Möglichkeit, bestimmte Verbindungstypen über eine spezifische Applikation programmgesteuert und damit bei Bedarf dynamisch einzurichten.

Anweisungen

Die Tabelle listet die STEP 7-Anweisungen, mit denen programmierte, offene Kommunikationsdienste (Open User Communication) mit UDP verwendet werden können.

Tabelle 3-1

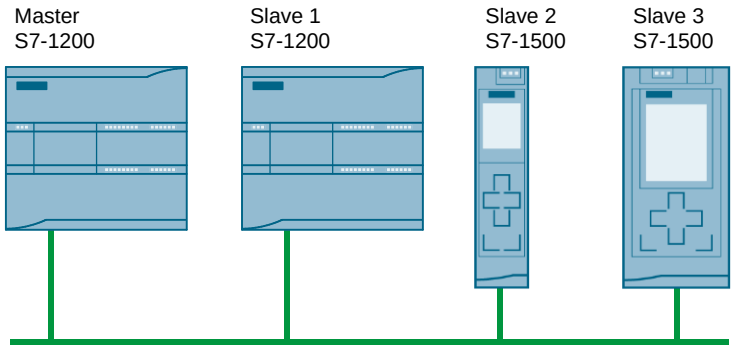
Bezeichnung	Kurzbeschreibung
TCON	Verbindungsaufbau
TDISCON	Verbindungsabbau
TUSEND	Senden von Daten; verbindungsloses Protokoll UDP gemäß RFC 768
TURCV	Empfangen von Daten; verbindungsloses Protokoll UDP gemäß RFC 768

4 Funktionsweise

4.1 Gesamtübersicht

Einleitung

Abbildung 4-1



Über PROFINET sind mehrere Steuerungen miteinander verbunden. Jede Steuerung kann temporär die Funktion des Masters übernehmen, wobei immer nur ein Master aktiv sein darf.

Der Master schickt über einen UDP-Broadcast Daten an die Slaves bzw. fordert Daten der Slaves an. Die Slaves quittieren den empfangenen Broadcast über ein Telegramm an den Master.

Die Daten der Slaves werden vom Master verwaltet und in einem Datenbaustein gespeichert.

Kommunikation

In diesem Anwendungsbeispiel werden programmierte Kommunikationsverbindungen verwendet. Dies hat den Vorteil, dass die jeweils anderen Steuerungen bei der Projektierung nicht bekannt sein müssen und beliebig weitere Steuerungen ergänzt werden können.

Für das Versenden und Empfangen von Telegrammen werden die in STEP 7 integrierten Kommunikationsbausteine TUSEND und TURCV verwendet.

Hinweis Dieses Anwendungsbeispiel realisiert einen directed broadcast. Die Kommunikation ist daher auf ein Subnetz beschränkt. Alle Teilnehmer müssen sich im selben Subnetz befinden.

Daten

Pro Sendeauftrag werden 236 Zeichen an Nutzdaten zwischen Master und Slaves übertragen. Die Anzahl der zu übertragenden Zeichen kann in dem Datentyp „LBC_typeUserData“ angepasst werden. Die Größe des Arrays wird von den jeweiligen Funktionsbausteinen bei der Initialisierung berechnet und die Parameter der TUSEND- bzw. TURCV-Bausteine dementsprechend gesetzt.

Mehr Informationen zum Aufbau der Telegramme finden Sie in den Kapiteln 4.3.2 „Versenden von Telegrammen“ und 4.4.2 „Quittieren von Telegrammen“.

Programmübersicht

Für dieses Anwendungsbeispiel wurde die Bausteinbibliothek „LBC“ erstellt. Bausteine, die Teil der Bibliothek sind, sind mit dem Präfix „LBC“ gekennzeichnet. Die Bibliothek ist nicht Know-How-geschützt und kann separat heruntergeladen werden (siehe \2\).

Abbildung 4-2

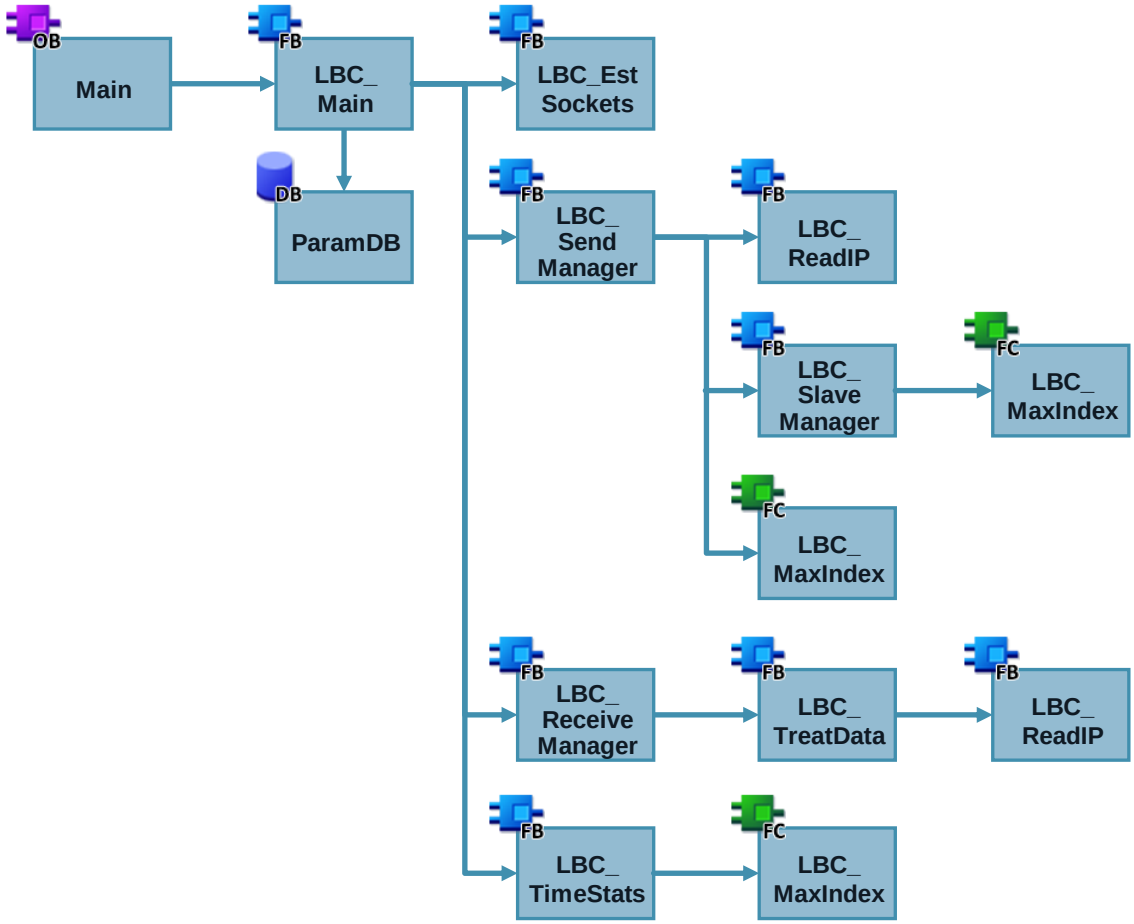


Tabelle 4-1

symbolischer Name	Funktion
LBC_Main	Anwenderschnittstelle, zentraler Baustein: • Auswertung der Anwenderparameter • Festlegen Master/Slave-Status • Aufruf der unterliegenden Funktionsbausteine • Koordination Keep-Alive-Telegramme, Send/Request-Anforderungen • Ausgabe der Anwenderparameter
LBC_EstSockets	Baut zwei Verbindungen zum Senden und Empfangen von Telegrammen auf. Die Verbindungen werden dann wie folgt genutzt: Verbindung 1: • Als Master: Versenden eines Broadcasts an alle

4.1 Gesamtübersicht

symbolischer Name	Funktion
	Teilnehmer im Subnetz • Als Slave: Empfangen eines Broadcast-Telegramms. Verbindung 2: • Als Master: Empfangen der Quittierungen der Slaves • Als Slave: Versenden der Quittierungen an den Master
LBC_SendManager	Realisiert das Senden von Telegrammen als Master: • Erstellen der Nutzdaten des Broadcast-Telegramms • Versenden des Broadcast-Telegramms • Aufrufen von LBC_SlaveManager
LBC_ReadIP	Liest die IP-Adresse der internen PROFINET-Schnittstelle der S7-CPU aus. Die eigene IP-Adresse wird jeweils in das Broadcast-Telegramm bzw. die Quittierung der Slaves geschrieben.
LBC_SlaveManager	Verwaltet das Slaves-Array: • Empfangen der Quittierungen der Slaves • Zuordnung der ankommenden Telegramme zu den Elementen des Slaves-Arrays
LBC_ReceiveManager	Realisiert die Slave-Funktionen und überprüft, ob ein weiterer Master im Netzwerk vorhanden ist: • Empfängt Broadcast-Telegramme des Masters • Ruft LBC_TreatData auf • Sendet Antwort/Quittierungs-Telegramm an den Master • Meldet, wenn ein Telegramm eines Masters angekommen ist
LBC_TreatData	Wertet die empfangenen Telegramme des Masters aus und erstellt die Quittierung.
LBC_TimeStats	Führt Zeitmessungen aus: • Als Master: Zeit zwischen dem Senden eines Telegramms und dem Empfang aller Quittierungen der aktiven Slaves • Als Slave: Zeit zwischen dem Empfang zweier Telegramme eines Masters
LBC_MaxIndex	Berechnet die Anzahl der Elemente eines Arrays.

4.2 Anwenderschnittstelle

Der Funktionsbaustein „LBC_Main“ ist der zentrale Baustein des Programms und dient als Anwenderschnittstelle. Über diesen Baustein werden die Parameter zur Bedienung des Programms übergeben und die weiteren Bausteine darin aufgerufen.

Schnittstelle

Abbildung 4-3

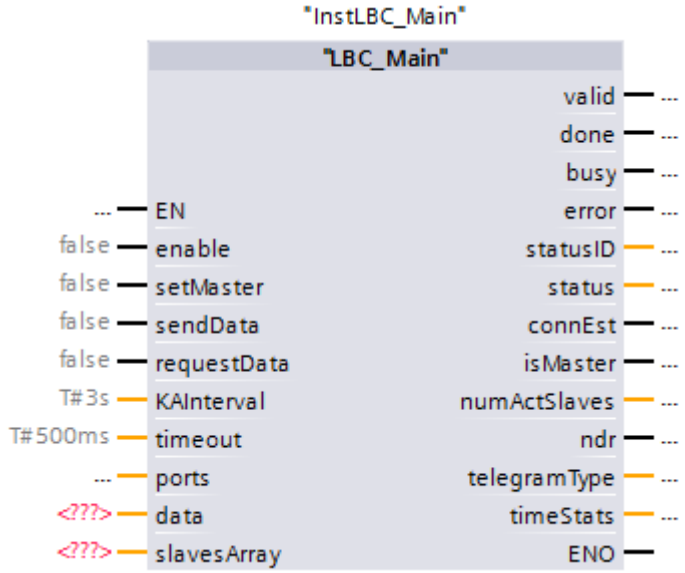


Tabelle 4-2

Symbol	Typ	Anmerkung
enable	Input Bool	Aktiviert das Programm. Solange #enable aktiv ist, verschickt der Master periodisch Keep-Alive Telegramme und die Slaves quittieren diese. Mit einer fallenden Flanke an #enable werden der Baustein und folgende Parameter zurückgesetzt: • Zeitmessungen • Zähler der empfangenen Telegramme • Inhalt des Slaves-Arrays (Slaves-Array wird mit dem nächsten Telegramm erneut initialisiert)
setMaster	Input Bool	Bei #setMaster = TRUE wird das Device zum Master, sofern kein anderer Master aktiv ist.
sendData	Input Bool	Bei einer positiven Flanke an #sendData werden Daten an die Slaves verschickt. Der Eingang ist im Slave-Modus irrelevant.
requestData	Input Bool	Bei einer positiven Flanke an #requestData werden Daten der Slaves angefordert. Der Eingang ist im Slave-Modus irrelevant.

Symbol	Typ	Anmerkung
KAInterval	Input Time	Gibt das Intervall an, in dem Keep-Alive-Telegramme an die Slaves versandt werden. Dieser Parameter muss bei allen Teilnehmern identisch sein, um zu garantieren, dass ein aktiver Master rechtzeitig erkannt wird, bevor eine Steuerung zum Master werden kann. Der Eingang ist im Slave-Modus irrelevant. Min.-Wert: 200 ms.
timeout	Input Time	Gibt an, nach welcher Zeit alle Quittierungen der Slaves eingetroffen sein müssen, bevor ein Fehler ausgegeben wird. Der Eingang ist im Slave-Modus irrelevant. Min.-Wert: 100 ms.
ports	Input “LBC_typePorts”	Gibt an, welche Ports für die Kommunikation zwischen Master und Slaves verwendet werden sollen.
valid	Output Bool	Zeigt an, dass die Parameter zulässig sind und das Programm ausgeführt wird.
done	Output Bool	Als Master: Broadcast-Telegramm wurde erfolgreich versandt. Das Bit ist nur einen Zyklus lang gesetzt.
busy	Output Bool	Das Programm ist beschäftigt.
error	Output Bool	Ein Fehler wurde erkannt. Das Bit ist aktiv, solange #enable gesetzt ist. Durch eine positive Flanke an #enable werden Fehler quittiert und das Programm erneut gestartet.
statusID	Output UInt	Gibt die Quelle der Statusmeldungen an. Der Ausgang ist aktiv, solange #enable gesetzt ist.
status	Output Word	Gibt es Statusmeldungen aus und spezifiziert bei #error den Fehler. Der Ausgang ist aktiv, solange #enable gesetzt ist.
connEst	Output Bool	Die Verbindungen sind aufgebaut.
isMaster	Output Bool	#isMaster = TRUE; das Gerät arbeitet als Master. #isMaster = FALSE; das Gerät arbeitet als Slave.
numActSlaves	Output Int	Gibt die Anzahl der aktiven Slaves an.
ndr	Output Bool	Neues Telegramm vom Master wurde empfangen. Das Bit ist nur einen Zyklus lang gesetzt. Der Ausgang ist im Master-Modus irrelevant.

Symbol	Typ	Anmerkung
telegramType	Output Int	Zeigt die Art des empfangenen Telegramms bei #ndr = TRUE an. 1: sendData 2: requestData 3: KA-Telegramm Der Ausgang ist im Master-Modus irrelevant.
timeStats	Output „LBC_typeTimeStats“	Gibt Zeitmessungen aus.
data	InOut „LBC_ typeUserData“	Master: zu sendende Daten. Slave: empfangene Daten.
slavesArray	InOut „LBC_typeSlavesArray“	Array mit Informationen und Nutzdaten der Slaves.

4.3 Funktionalität als Master

4.3.1 Master-Funktion übernehmen

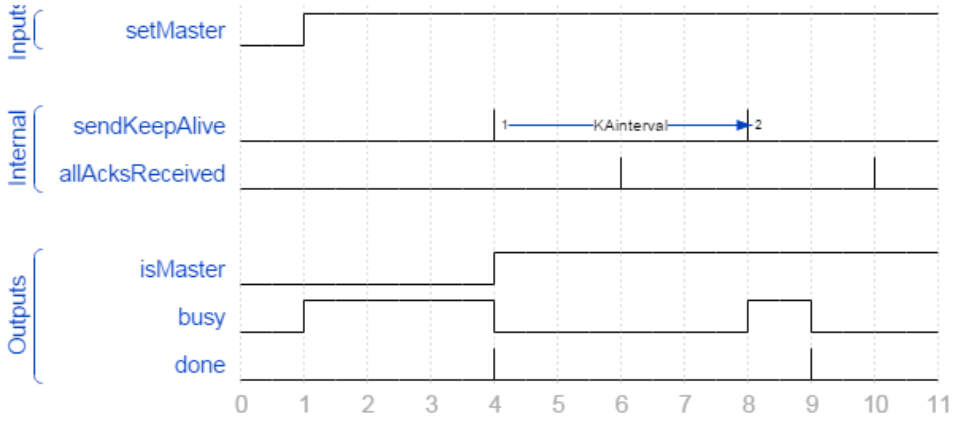
Jede beliebige Steuerung kann temporär zum Master werden. Allerdings kann stets nur ein Master aktiv sein.

Ist der Eingang #enable gesetzt, wird die Applikation gestartet und die Station baut die benötigten Verbindung auf. Durch das Setzen des Eingangs #setMaster wird geprüft, ob ein Master bereits im Netzwerk aktiv ist. Dabei wird die parametrierte Zeit #KAInterval abgewartet. Wird innerhalb dieser Zeit kein Telegramm von einem Master empfangen, ist kein Master aktiv und die Steuerung übernimmt die Funktion des Masters. Dass die Steuerung die Funktion des Masters übernommen hat, wird am Ausgang #isMaster angezeigt.

ACHTUNG	Der Parameter #KAInterval muss bei allen Teilnehmern identisch sein, um zu garantieren, dass ein aktiver Master rechtzeitig erkannt wird, bevor eine Steuerung zum Master werden kann.

Die nachfolgende Grafik zeigt die Übernahme der Master-Funktion und das Versenden von Keep-Alive-Telegrammen.

Abbildung 4-4



4.3.2 Versenden von Telegrammen

Allgemein

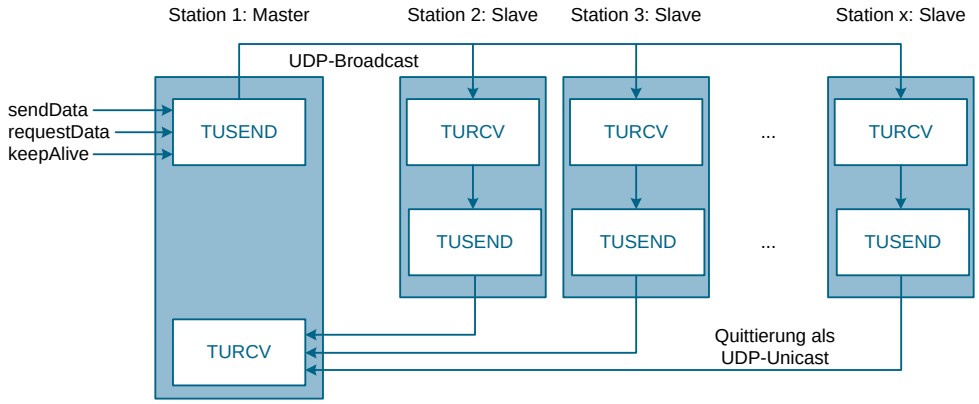
Über die Anwenderschnittstelle des Masters kann der Anwender folgende Telegramme verschicken:

- sendData: Daten an die Slaves schicken
- requestData: Daten der Slaves anfordern

Steht kein Sendeauftrag an, werden automatisch in parametrierten Zeitabständen Keep-Alive-Telegramme versandt, um den Status der Slaves abzufragen.

Es werden stets alle Telegramme vom Master durch die Slaves quittiert. Der Empfang der Quittierungen wird vom Master zeitlich überwacht.

Abbildung 4-5



Falls mehrere Sendeaufträge gleichzeitig aktiv sind, so priorisiert das Programm die Sendeaufträge folgendermaßen:

1. Send-Auftrag
2. Request-Auftrag
3. Keep-Alive-Telegramm

Sobald ein Telegramm erfolgreich verschickt wurde, ist der Baustein bereit ein weiteres Telegramm zu verschicken. Dies wird durch den Ausgang #done signalisiert, der jeweils einen Zyklus lang aktiv ist (siehe Abbildung 4-4). Der Empfang der Quittierungen wird dabei nicht abgewartet.

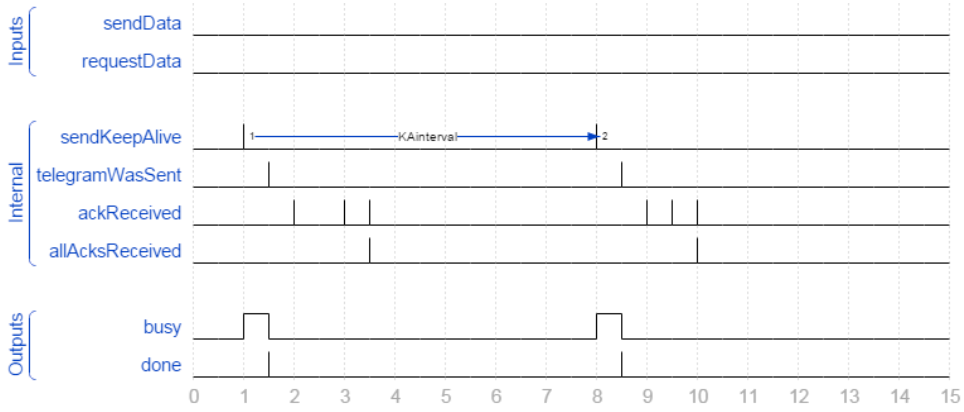
Keep-Alive

Keep-Alive-Telegramme werden periodisch vom Master versandt und deren Empfang durch die Slaves quittiert. Dadurch kann der Master auch während einer Sendepause sicherstellen, dass noch alle Slaves aktiv sind bzw. kurzfristig erkennen, wenn die Verbindung zu einem Slave unterbrochen wurde.

Das Intervall zum Versenden von Keep-Alive-Telegramm wird über den Eingang #KAInterval festgelegt.

Diese Telegramme werden nur dann versandt, wenn aktuell kein Send- oder Request-Telegramm verschickt werden soll.

Abbildung 4-6



Send

Mit einem Send-Telegramm werden Daten vom Master an die Slaves übertragen. Dies wird mit einer positiven Flanke an dem Eingang #sendData angestoßen.

Die zu übertragenden Nutzdaten werden als Datentyp „LBC_typeUserData“ an dem InOut-Parameter #data zur Verfügung gestellt. Dieser Datentyp besteht aus einem Array mit 236 Zeichen (Chars), die übertragen werden können.

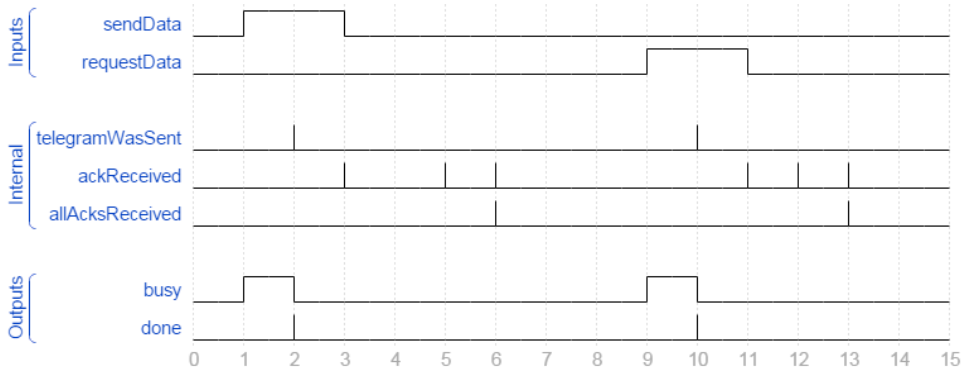
Zusätzlich zu den Nutzdaten werden folgende Daten in einem Telegramm übertragen:

- Quelladresse des Masters (damit die Slaves wissen, wohin sie die Quittierungen adressieren müssen)
- Zeitstempel
- Telegrammtyp (Keep-Alive, Send oder Request)

Abbildung 4-7

Quelladresse	Zeitstempel	Telegrammtyp	Nutzdaten

Abbildung 4-8



Request

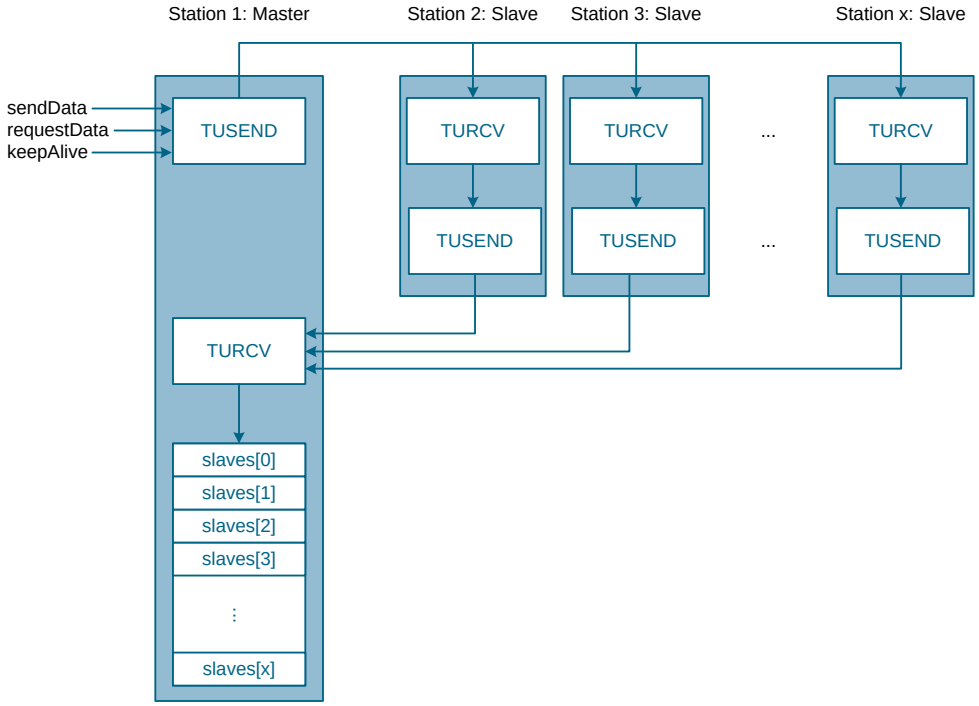
Mit einem Request-Telegramm fordert der Master Daten von den Slaves an. Die empfangenen Daten werden in der Slave-Verwaltung vom Master abgelegt.

Der Versand eines Request-Telegramms wird mit einer positiven Flanke an dem Eingang #requestData angestoßen (siehe Abbildung 4-8).

4.3.3 Slave-Verwaltung

Durch die empfangenen Quittierungen werden dem Master alle Slaves bekannt und in einem Array abgelegt. Dies geschieht in dem Funktionsbaustein „LBC_SlaveManager“.

Abbildung 4-9



Dieses Slaves-Array wird durch den Datentyp „LBC_SlavesArray“ definiert und enthält folgende Daten:

- IP-Adresse und Port des Slaves
- Status, ob Slave aktiv ist
- Status, ob Slave den Erhalt des letzten Telegramms quittiert hat
- Empfangene Daten des Slaves
- Anzahl empfangener Keep-Alive-Telegramme des Slaves
- Anzahl empfangener Request- und Send-Telegramme des Slaves

Da Speicher nicht während der Laufzeit der Steuerung allokiert werden kann, muss bei der Projektierung die max. Anzahl der zu verwalteten Slaves bekannt sein. Dies kann in dem Datentyp „LBC_typeSlavesArray“ angepasst werden. Im Auslieferungszustand des Anwendungsbeispiels ist eine max. Anzahl von 20 Slaves parametriert.

Abbildung 4-10

LBC_typeSlavesArray

	Name	Data type	Default value
1	slaves	Array[0..19] ...	
2	slaves[0]	"LBC_typeSlave"	
3	slaves[1]	"LBC_typeSlave"	
4	slaves[2]	"LBC_typeSlave"	
5	slaves[3]	"LBC_typeSlave"	
6	slaves[4]	"LBC_typeSlave"	
7	slaves[5]	"LBC_typeSlave"	
8	slaves[6]	"LBC_typeSlave"	
9	slaves[7]	"LBC_typeSlave"	
10	slaves[8]	"LBC_typeSlave"	
11	slaves[9]	"LBC_typeSlave"	
12	slaves[10]	"LBC_typeSlave"	
13	slaves[11]	"LBC_typeSlave"	
14	slaves[12]	"LBC_typeSlave"	
15	slaves[13]	"LBC_typeSlave"	
16	slaves[14]	"LBC_typeSlave"	
17	slaves[15]	"LBC_typeSlave"	
18	slaves[16]	"LBC_typeSlave"	
19	slaves[17]	"LBC_typeSlave"	
20	slaves[18]	"LBC_typeSlave"	
21	slaves[19]	"LBC_typeSlave"	

Initialisierung

Damit die Anzahl der Slaves während der Laufzeit änderbar ist, muss ein dynamischer Aufbau des Slaves-Arrays erfolgen. In einem Initialisierungsvorgang werden die Slaves über ein Keep-Alive-Telegramm angesprochen und das Slaves-Array wird in der Reihenfolge der eintreffenden Antworten der Slaves gefüllt.

Da die Antwortreihenfolge unterschiedlich ausfallen kann, kann das Slaves-Array pro Initialisierungsvorgang unterschiedlich aufgebaut sein.

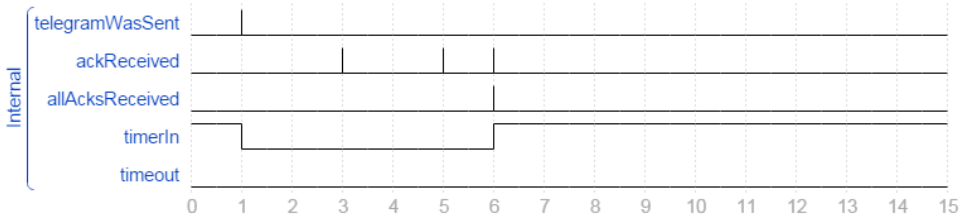
Nach der Initialisierung des Slaves-Arrays werden bei jedem Telegramm die Antworten der Slaves mit den bekannten IP-Adressen verglichen und dadurch dem jeweiligen Slave zugeordnet.

Timeout eines Slaves

Die Quittierungen der Slaves werden zeitlich überwacht. Mit dem erfolgreichen Versand eines Telegramms wird ein Timer mit der parametrierten Zeit #timeout gestartet. Treffen innerhalb dieser Zeit alle Quittierungen der aktiven Slaves ein, wird der Timer zurückgesetzt. Mit dem nächsten Versand eines Telegramms, wird der Timer erneut gestartet.

Die nachfolgende Grafik zeigt den Empfang aller Quittierungen bei drei aktiven Slaves innerhalb der parametrierten Zeit.

Abbildung 4-11

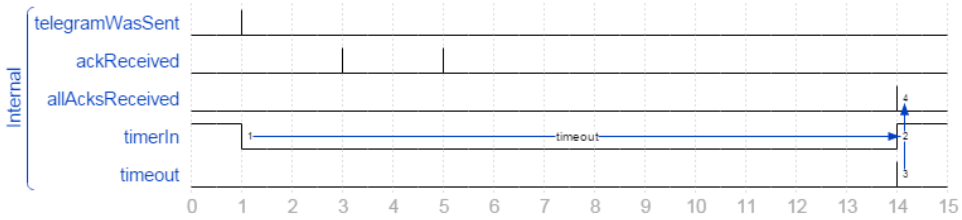


Antwortet ein Slave innerhalb der parametrierten Antwortzeit #timeout nicht, geschieht Folgendes:

- Eine Statusmeldung wird am Ausgang #status ausgeben (siehe Kapitel 7.3).
- Das Bit #isActive im Slaves-Array des jeweiligen Slaves wird zurückgesetzt.
- Die Anzahl der aktiven Slaves #numActSlaves wird um 1 verringert.
- Das Element im Slaves-Array bleibt erhalten.

Die nachfolgende Grafik zeigt den Empfang von nur zwei Quittierungen bei drei aktiven Slaves und den Ablauf der parametrierten Zeit #timeout.

Abbildung 4-12



Rückkehr eines Slaves

Antwortet ein inaktiver Slave wieder auf Telegramm, geschieht Folgendes:

- Eine Statusmeldung wird am Ausgang #status ausgeben (siehe Kapitel 7.3).
- Das Bit #isActive im Slaves-Array des jeweiligen Slaves wird wieder gesetzt.
- Die Anzahl der aktiven Slaves #numActSlaves wird um 1 erhöht.
- Das zuvor verwendete Element im Slaves-Array wird weiterverwendet.

Antwort eines unbekannten Slaves

Antwortet während des Betriebs ein bisher unbekannter Slave, geschieht Folgendes:

- Eine Statusmeldung wird am Ausgang #status ausgeben (siehe Kapitel 7.3).
- Die Informationen des Slaves werden in einem neuen Element des Slaves-Arrays gespeichert.
- Das Bit #isActive im Slaves-Array des jeweiligen Slaves wird gesetzt.
- Die Anzahl der aktiven Slaves #numActSlaves wird um 1 erhöht.

4.3.4 Zeitmessung

Als Master misst das Programm die Zeit zwischen dem Versenden eines Telegramms und dem Empfang aller Quittierungen der bekannten Slaves. Das Ergebnis der Messung wird an dem Ausgang #timeStats im Format „LBC_typeTimeStats“ ausgegeben.

Darin enthalten sind der aktuelle Messwert, die letzten 20 Messwerte und der gemittelte Messwert über die letzten 20 Messwerte.

Die Anzahl der zu speichernden Messwerte kann in dem Datentyp „LBC_typeTimeStats“ angepasst werden.

4.4 Funktionalität als Slave

Ist der Eingang #enable gesetzt, wird die Applikation gestartet und die Station baut die benötigten Verbindung auf. Wenn die Station kein Master sein soll und #setMaster daher nicht gesetzt ist, übernimmt die Station die Funktion eines Slaves und antwortet auf alle an dem parametrierten Port eintreffenden Telegramme.

4.4.1 Empfangen von Telegrammen

Eine Station im Slave-Modus ruft zyklisch den Funktionsbaustein „LBC_ReceiveManager“ auf, in dem Telegramme empfangen und quittiert werden. An dem Ausgang #ndr zeigt der Slave einen Zyklus lang an, dass ein neues Telegramm empfangen wurde. Der Ausgang #telegramType spezifiziert die Art des Telegramms.

Die empfangenen Nutzdaten werden an dem InOut-Parameter #data bereitgestellt.

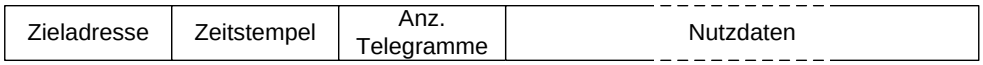
4.4.2 Quittieren von Telegrammen

Es werden stets alle Telegramme durch die Slaves quittiert. Die Größe der Quittierung ist stets dieselbe, unabhängig von der Art des Telegramms, das quittiert wird.

Die Quittierung besteht aus folgenden Daten:

- Zieladresse des Masters
- Zeitstempel
- Anzahl der empfangenen Telegramme
- Zu übertragende Nutzdaten (die Nutzdaten werden nur bei der Antwort auf Request-Telegramme aktualisiert)

Abbildung 4-13



4.4.3 Zeitmessung

Als Slave misst das Programm die Zeit zwischen zwei empfangenen Telegrammen vom Master. Das Ergebnis der Messung wird an dem Ausgang #timeStats im Format „LBC_typeTimeStats“ ausgegeben.

Darin enthalten sind der aktuelle Messwert, die letzten 20 Messwerte und der gemittelte Messwert über die letzten 20 Messwerte.

4.5 Leistungsmerkmale

4.5.1 Zykluszeit als Master

CPU 1214C

Tabelle 4-3

Typ	Zykluszeit in ms
kürzeste	2
durchschnittlich	3
längste	8

CPU 1516

Tabelle 4-4

Typ	Zykluszeit in ms
kürzeste	0
durchschnittlich	1
längste	3

4.5.2 Reaktionszeit der Slaves

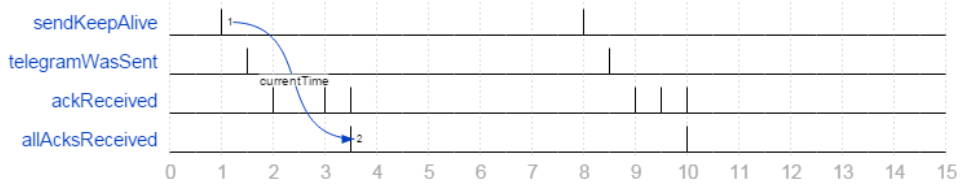
Gemessen wird die Zeit zwischen dem Senden von periodischen Keep-Alive-Telegrammen vom Master und dem Empfang aller Quittierungen durch die Slaves bei drei aktiven Slaves.

Tabelle 4-5

Typ	Reaktionszeit in ms
kürzeste	19
durchschnittlich	44
längste	84

Hinweis Der Mittelwert der Reaktionszeiten wurde aus 100 gemessenen Werten ermittelt, während alle 500 ms ein Keep-Alive-Telegramm versandt wurde.

Abbildung 4-14



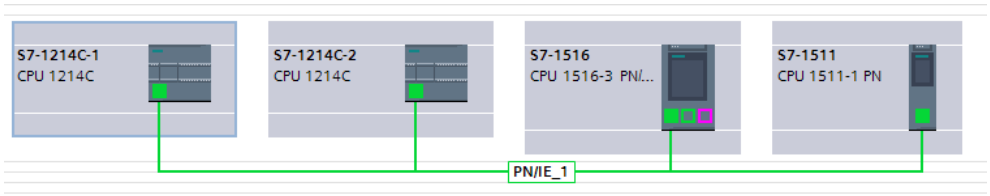
5 Konfiguration und Projektierung

Das mitgelieferte Projekt bedarf keiner weiteren Konfiguration. Sollten Sie das Anwendungsbeispiel mit anderen Komponenten nachbauen, werden in diesem Kapitel die wichtigsten Einstellungen gezeigt.

5.1 Konfiguration der Stationen

1. Öffnen Sie TIA Portal und legen Sie ein neues Projekt an.
2. Öffnen Sie „Geräte und Netze“ („Devices & network“) aus der Projektnavigation.
3. Ziehen Sie die verwendeten Steuerungen aus dem Hardwarekatalog in den Arbeitsbereich.
4. Alternativ können Sie auch für jede Steuerung ein eigenes Projekt erstellen. Achten Sie dabei darauf, dass jede Steuerung eine eindeutige IP-Adresse besitzt und die Steuerungen sich im selben Subnetz befinden.
5. Verbinden Sie die PROFINET-Schnittstellen der erstellten Steuerungen miteinander, sofern Ihr Projekt mehrere Steuerungen enthält.
   Dieser Schritt ist nicht zwingend. Allerdings werden dadurch den Steuerungen automatisch unterschiedliche IP-Adressen vergeben.

Abbildung 5-1



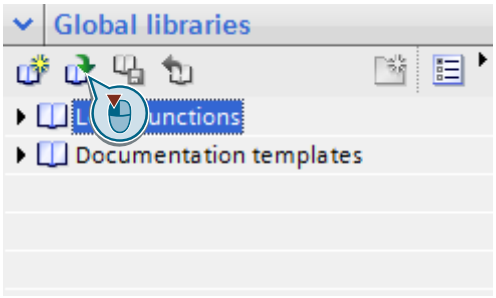
Hinweis Sollten nach der Projektierung noch weitere Steuerungen hinzukommen, muss die Projektierung der bereits vorhandenen Steuerungen nicht anpasst werden. Ebenso kann jede Steuerung in einem eigenen Projekt projektiert werden, ohne die restlichen Steuerungen zu kennen.

5.2 Verwenden der LBC-Bibliothek

Laden der Bibliothek

1. Laden Sie die LBC-Bibliothek herunter und entpacken Sie die Datei. Den Downloadlink finden Sie unter \2\.
2. Öffnen Sie in TIA Portal V13 im rechten Bildschirmrand den Bereich „Bibliotheken“ („Libraries“)
3. Klappen Sie das Register „Globale Bibliotheken“ („Global libraries“) aus.
4. Klicken Sie auf das Symbol „Globale Bibliothek öffnen“ („Open global library“) und wählen Sie die entpackte Bibliothek aus.

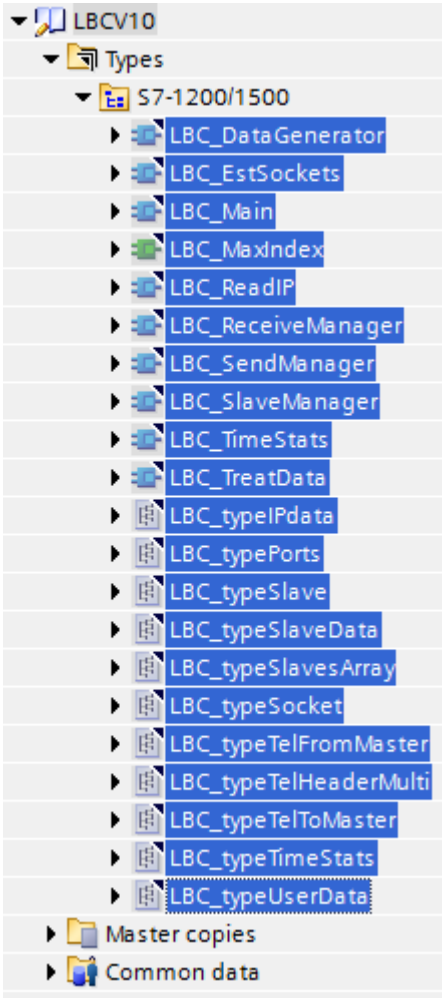
Abbildung 5-2



Instanziieren der Bibliotheksbausteine

1. Klappen Sie den Ordner einer Ihrer Steuerungen aus der Projektnavigation auf.
2. Klappen Sie die Bibliothek „LBC“ und den Ordner „Typen“ („Types“) auf und ziehen Sie den Inhalt des Ordners „S7-1200_S7-1500“ auf den Ordner „Programmbausteine“ („Program blocks“) der jeweiligen Steuerung. Die Programmbausteine und Datentypen der Bibliothek werden nun instanziiert.

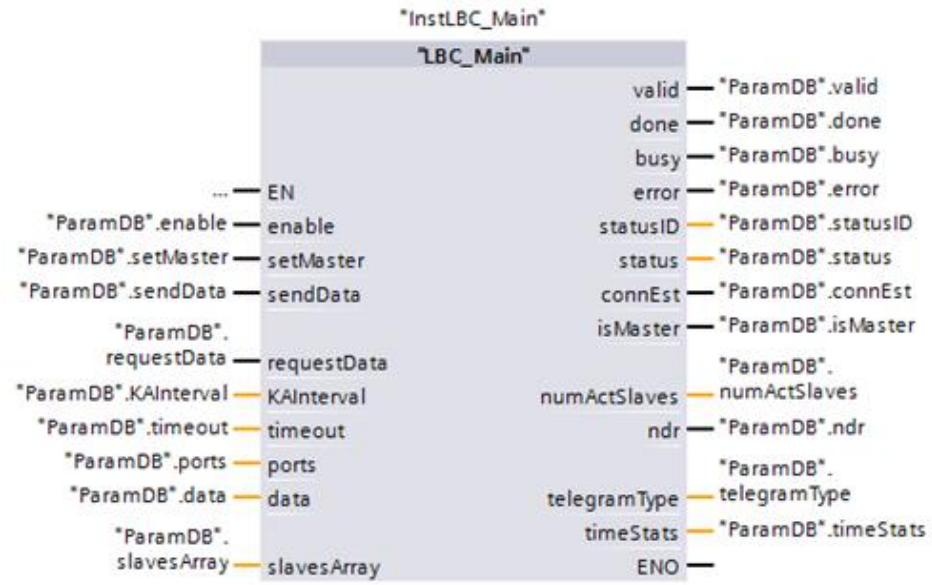
Abbildung 5-3



3. Öffnen Sie unter „Programmbausteine“ den Organisationsbaustein OB1.
4. Ziehen Sie unter „Programmbausteine“ den Funktionsbaustein „LBC_Main“ in ein freies Netzwerk und erstellen Sie einen Instanzdatenbaustein.
5. Belegen Sie die Formalparameter des Funktionsbausteins mit Aktualparametern.

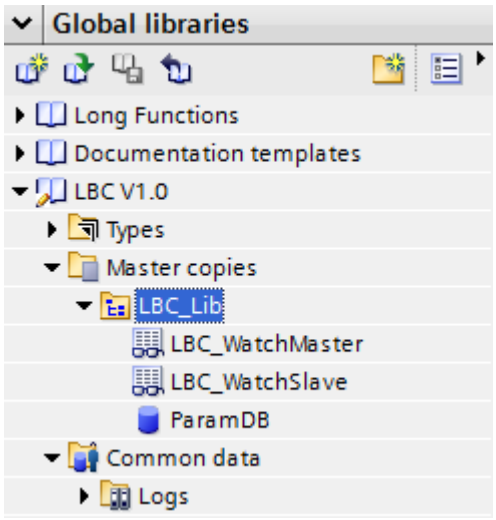
   Für Testzwecke können Sie die Aktualparameter des Datenbausteins „LBC_ParamDB“ aus dem Ordner „Kopiervorlagen“ („Master copies“) der Bibliothek verwenden. Ziehen Sie dazu diesen aus der Bibliothek auf den Ordner „Programmbausteine“.

Abbildung 5-4



6. Ziehen Sie die Beobachtungstabellen „LBC_WatchMaster“ und „LBC_WatchSlave“ aus dem Ordner „Kopiervorlagen“ („Master copies“) der Bibliothek auf den Ordner „Beobachtungstabelle“ („Watch and force tables“).

Abbildung 5-5



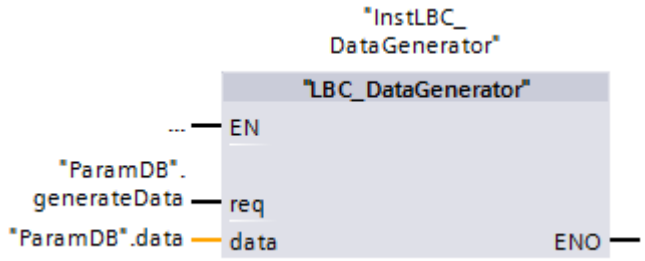
7. Wiederholen Sie die Schritte 5 bis 9 für die weiteren Steuerungen Ihres Projekts.

Testdaten generieren

Führen Sie die nachfolgenden Schritte durch, wenn die Applikation eigene Testdaten für Testzwecke generieren soll.

1. Öffnen Sie OB1 oder erstellen Sie einen anderen Organisationsbaustein in dessen Zyklus Testdaten generiert werden sollen.
2. Ziehen Sie den Funktionsbaustein „LBC_DataGenerator“ aus dem Ordner „LBC_Lib“ unter „Programmbausteine“ in ein leeres Netzwerk des Organisationsbausteins.
3. Belegen Sie die Formalparameter des Funktionsbausteins mit Aktualparametern des Datenbausteins „LBC_ParamDB“.

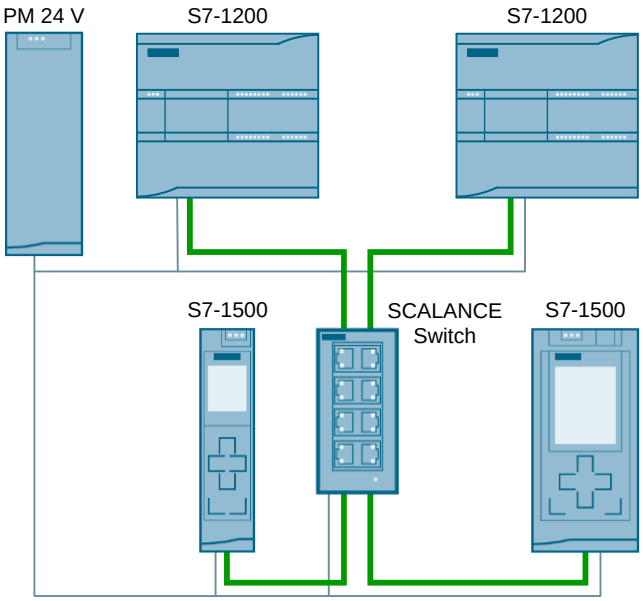
Abbildung 5-6



6 Installation und Inbetriebnahme

6.1 Installation der Hardware

Abbildung 6-1



1. Montieren Sie die Steuerungen, Spannungsversorgung und den Switch auf Hutschienen.
2. Schließen Sie die Steuerungen und den Switch an die 24 V DC Spannungsversorgung an.
3. Verbinden Sie die PROFINET-Ports der Steuerungen mit dem SCALANCE Switch.

6.2 Installation der Software

6.2.1 Vorbereitung

Laden Sie die Datei 20983558_MasterSlave_UDP-Broadcast_CODE_V21.zip runter. Den Downloadlink finden Sie unter \2\.

1. Speichern Sie die zip-Dateien in einem beliebigen Verzeichnis auf Ihrem Computer und entpacken Sie diese.
2. Stellen Sie die IP-Adresse des PG/PCs ein, sodass sich das PG/PC im selben Subnetz wie die CPUs befindet.
3. Verbinden Sie mit einem Ethernet-Kabel das/den PG/PC mit dem SCALANCE Switch.

Für dieses Anwendungsbeispiel wurden folgende IP-Adressen verwendet:

CPU 1214C (1)
IP-Adresse: 192.168.0.10
Subnetz-Maske: 255.255.255.0

CPU 1214C (2)
IP-Adresse: 192.168.0.11
Subnetz-Maske: 255.255.255.0

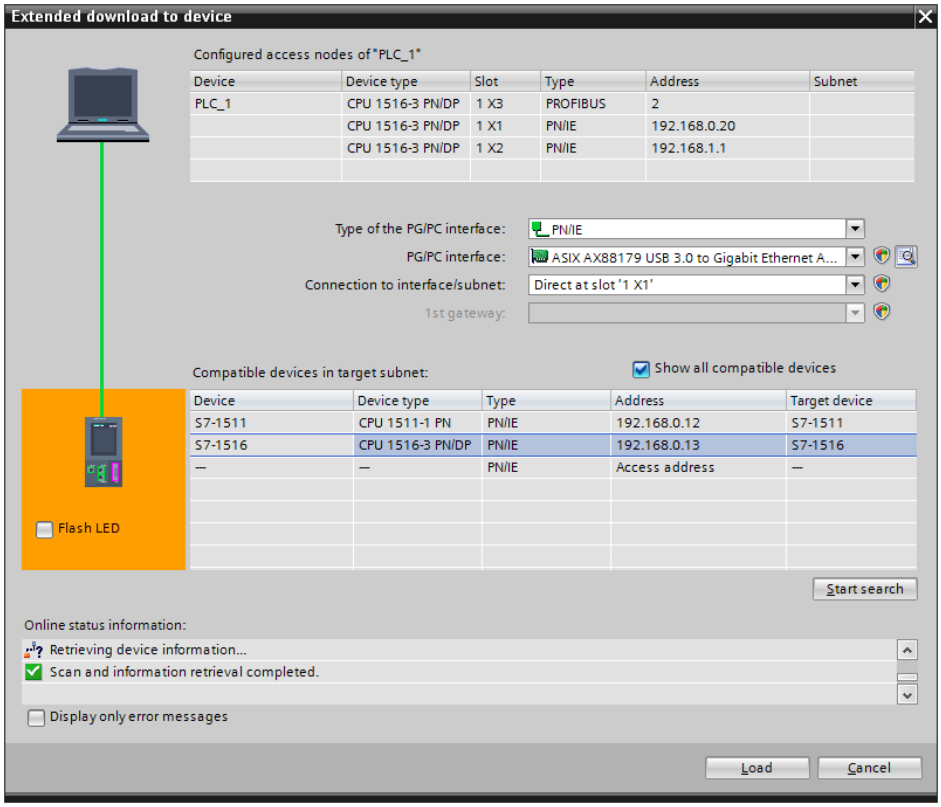
CPU 1511
IP-Adresse: 192.168.0.12
Subnetz-Maske: 255.255.255.0

CPU 1516
IP-Adresse: 192.168.0.13
Subnetz-Maske: 255.255.255.0

6.2.2 S7-Projekt in die CPU laden

1. Öffnen Sie TIA Portal.
2. Wechseln Sie in die Projektansicht.
3. Klicken Sie in der Menüleiste im TIA Portal auf „Projekt > Öffnen“ („Project > Open“).
4. Klicken Sie „Durchsuchen“ („Browse“) und öffnen Sie das jeweilige Projekt.
5. Stellen Sie die jeweilige CPU auf STOP.
6. Klicken Sie mit der rechten Maustaste im Projektbaum auf die jeweilige CPU und dann auf „Laden in Gerät > Hardware und Software (nur Änderungen)“ („Download to device > Hardware and Software (only changes)“).
7. Wählen Sie die jeweilige Schnittstelle aus und klicken Sie auf „Suche starten“ („Start search“).

Abbildung 6-2



8. Wählen Sie die CPU anhand der Adresse aus und klicken Sie anschließend auf „Laden“ („Load“).

Hinweis Die IP-Adresse und der Gerätename werden beim Laden des Projekts in die CPU automatisch zugewiesen.

9. Bestätigen Sie den Dialog indem Sie auf „Laden“ („Load“) klicken.
10. Klicken Sie auf „Fertig“ („Finish“), wenn der Ladevorgang abgeschlossen ist.

7 Bedienung der Applikation

Im Nachfolgenden wird die Bedienung der Applikation für Testzwecke über Beobachtungstabellen erläutert. Diese sind als Kopiervorlagen in der mitgelieferten Bibliothek bereits enthalten (siehe \2\).

Kopieren Sie dazu die Beobachtungstabellen „LBC_WatchMaster“ und „LBC_WatchSlave“ für jede teilnehmende Steuerung in den jeweiligen Projektordner (siehe Kapitel 5.2).

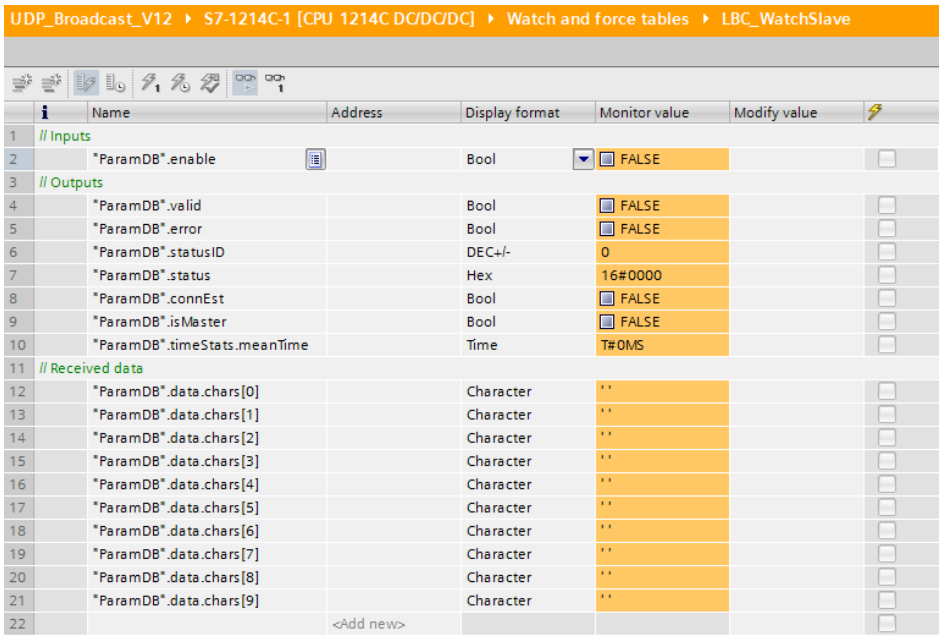
7.1 Szenario Station als Slave

In diesem Kapitel wird erläutert, wie die Slave-Funktion für eine Station aktiviert wird.

Tabelle 7-1

Nr.	Aktion	Anmerkung
1.	Öffnen Sie die Beobachtungstabelle „LBC_WatchSlave“ der jeweiligen Steuerung.	Die Beobachtungstabelle ist in den Kopiervorlagen der mitgelieferten Bibliothek enthalten (siehe Kapitel 5.2).
2.	Steuern Sie die Variable „LBC_ParamDB“.enable auf TRUE.	Die Applikation wird daraufhin gestartet und die Verbindungen aufgebaut. Sobald die Verbindungen aufgebaut sind, wird die Station Telegramme empfangen und diese quittieren. Die empfangenen Daten werden in dem Array „LBC_ParamDB“.data dargestellt. Die durchschnittliche Zeit zwischen dem Empfang zweier Telegramme vom Master werden in der Variable „LBC_ParamDB“.timeStats.meanTime angezeigt.

Abbildung 7-1



	Name	Address	Display format	Monitor value	Modify value
1	// Inputs				
2	"ParamDB".enable		Bool	FALSE	
3	// Outputs				
4	"ParamDB".valid		Bool	FALSE	
5	"ParamDB".error		Bool	FALSE	
6	"ParamDB".statusID		DEC+/-	0	
7	"ParamDB".status		Hex	16#0000	
8	"ParamDB".connEst		Bool	FALSE	
9	"ParamDB".isMaster		Bool	FALSE	
10	"ParamDB".timeStats.meanTime		Time	T#0MS	
11	// Received data				
12	"ParamDB".data.chars[0]		Character	''	
13	"ParamDB".data.chars[1]		Character	''	
14	"ParamDB".data.chars[2]		Character	''	
15	"ParamDB".data.chars[3]		Character	''	
16	"ParamDB".data.chars[4]		Character	''	
17	"ParamDB".data.chars[5]		Character	''	
18	"ParamDB".data.chars[6]		Character	''	
19	"ParamDB".data.chars[7]		Character	''	
20	"ParamDB".data.chars[8]		Character	''	
21	"ParamDB".data.chars[9]		Character	''	
22		<Add new>			

7.2 Szenario Station als Master

7.2.1 Master-Funktion übernehmen

In diesem Kapitel wird erläutert, wie die Master-Funktion für eine Station aktiviert wird.

Tabelle 7-2

Nr.	Aktion	Anmerkung
1.	Öffnen Sie die Beobachtungstabelle „LBC_WatchMaster“ der jeweiligen Steuerung.	Die Beobachtungstabelle ist in den Kopiervorlagen der mitgelieferten Bibliothek enthalten (siehe Kapitel 5.2).
2.	Steuern Sie die Variable „LBC_ParamDB“.enable auf TRUE.	Die Applikation wird daraufhin gestartet und die Verbindungen aufgebaut.
3.	Steuern Sie die Variable „LBC_ParamDB“.setMaster auf TRUE.	Die Station wartet nun die parametrierte Zeit #KAinterval ab, um zu prüfen, ob bereits ein Master aktiv ist. Wird innerhalb dieser Zeit kein Telegramm von einem Master empfangen, ist kein Master aktiv und die Steuerung übernimmt die Funktion des Masters und die Variable „LBC_ParamDB“.isMaster wird gesetzt. Ab jetzt werden periodisch Keep-Alive-Telegramme vom Master versandt. Mit dem Empfang der Quittierungen werden die #isActive-Variablen des Slaves-Arrays gesetzt und die Anzahl der aktiven Slaves unter „LBC_ParamDB“.numActSlaves angezeigt.

Nr.	Aktion	Anmerkung
		Die durchschnittliche Zeit zwischen dem Versand eines Telegramms und dem Empfang aller Quittierungen der aktiven Slaves wird in der Variable „LBC_ParamDB“.timeStats.meanTime angezeigt.

Abbildung 7-2

UDP_Broadcast_V12 ▸ S7-1214C-1 [CPU 1214C DC/DC/DC] ▸ Watch and force tables ▸ LBC_WatchMaster

	Name	Display format	Monitor value	Modify value
1	// Input parameters LBC_Manager			
2	"ParamDB".enable	Bool	FALSE	
3	"ParamDB".setMaster	Bool	FALSE	
4	"ParamDB".putData	Bool	FALSE	
5	"ParamDB".getData	Bool	FALSE	
6	"ParamDB".autoSend	Bool	FALSE	
7	"ParamDB".KAInterval	Time	T#3S	
8	"ParamDB".timeout	Time	T#500MS	
9	// Output parameters LBC_Manager			
10	"ParamDB".valid	Bool	FALSE	
11	"ParamDB".done	Bool	FALSE	
12	"ParamDB".busy	Bool	FALSE	
13	"ParamDB".error	Bool	FALSE	
14	"ParamDB".statusID	DEC+/-	0	
15	"ParamDB".status	Hex	16#0000	
16	"ParamDB".savedStatus	Hex	16#0000	
17	"ParamDB".connEst	Bool	FALSE	
18	"ParamDB".isMaster	Bool	FALSE	
19	"ParamDB".timeStats.meanTime	Time	T#0MS	
20	"InstLBC_Manager".statMode	DEC	0	
21	// Active slaves			
22	"ParamDB".numActSlaves	DEC+/-	0	
23	"ParamDB".slavesArray.slaves[0].isActive	Bool	FALSE	
24	"ParamDB".slavesArray.slaves[1].isActive	Bool	FALSE	
25	"ParamDB".slavesArray.slaves[2].isActive	Bool	FALSE	
26	"ParamDB".slavesArray.slaves[0].hasSentAck	Bool	FALSE	
27	"ParamDB".slavesArray.slaves[1].hasSentAck	Bool	FALSE	
28	"ParamDB".slavesArray.slaves[2].hasSentAck	Bool	FALSE	
29	// Data			
30	"ParamDB".generateData	Bool	FALSE	
31	"ParamDB".data.chars[0]	Character	' '	
32	"ParamDB".data.chars[1]	Character	' '	
33	"ParamDB".slavesArray.slaves[0].receivedData.data.chars[0]	Character	' '	
34	"ParamDB".slavesArray.slaves[0].receivedData.data.chars[1]	Character	' '	
35	"ParamDB".slavesArray.slaves[1].receivedData.data.chars[0]	Character	' '	
36	"ParamDB".slavesArray.slaves[1].receivedData.data.chars[1]	Character	' '	
37	"ParamDB".slavesArray.slaves[2].receivedData.data.chars[0]	Character	' '	
38	"ParamDB".slavesArray.slaves[2].receivedData.data.chars[1]	Character	' '	
39	// Slaves			
40	"ParamDB".slavesArray.slaves[0].address.IP.ADDR[4]	DEC	0	
41	"ParamDB".slavesArray.slaves[0].receivedData.receivedPutGet	DEC	0	
42	"ParamDB".slavesArray.slaves[0].receivedData.receivedKeepAlive	DEC	0	
43	"ParamDB".slavesArray.slaves[1].address.IP.ADDR[4]	DEC	0	
44	"ParamDB".slavesArray.slaves[1].receivedData.receivedPutGet	DEC	0	
45	"ParamDB".slavesArray.slaves[1].receivedData.receivedKeepAlive	DEC	0	
46	"ParamDB".slavesArray.slaves[2].address.IP.ADDR[4]	DEC	0	
47	"ParamDB".slavesArray.slaves[2].receivedData.receivedPutGet	DEC	0	
48	"ParamDB".slavesArray.slaves[2].receivedData.receivedKeepAlive	DEC	0	

7.2.2 Testdaten generieren

Steuern Sie die Variable „LBC_ParamDB“.generateData auf TRUE, um das Array „LBC_ParamDB“.data mit Zeichen von „a“ bis „z“ füllen zu lassen. Abhängig von der Zykluszeit des Organisationsbausteins, in dem „LBC_GenerateData“ aufgerufen wird, werden die Elemente zyklisch mit neuen Zeichen überschrieben.

7.2.3 Send-Telegramm versenden

Tabelle 7-3

Nr.	Aktion	Anmerkung
1.	Stellen Sie sicher, dass die jeweilige Station als Master aktiv ist und kein Fehler vorliegt.	„LBC_ParamDB“.isMaster und „LBC_ParamDB“.valid sind TRUE.
2.	Steuern Sie die Variablen „LBC_ParamDB“.data.chars[0] und data.chars[1] auf beliebige Zeichen oder lassen Sie diese automatisch befüllen (siehe 7.2.2).	
3.	Steuern Sie die Variable „LBC_ParamDB“.sendData auf TRUE.	Es wird nun ein Send-Telegramm mit den Daten aus dem Array #data verschickt. Sobald das Telegramm verschickt wurde, wird #done für einen Zyklus gesetzt. Der Empfang des Telegramms wird durch die Slaves quittiert und jeweils an der Variable #hasSentAck bis zum nächsten Sendeauftrag angezeigt.

7.2.4 Request-Telegramm versenden

Tabelle 7-4

Nr.	Aktion	Anmerkung
1.	Stellen Sie sicher, dass die jeweilige Station als Master aktiv ist und kein Fehler vorliegt.	„LBC_ParamDB“.isMaster und „LBC_ParamDB“.valid sind TRUE.
2.	Steuern Sie die Variable „LBC_ParamDB“.requestData auf TRUE.	Es wird nun ein Request-Telegramm verschickt und Daten von den Slaves angefordert. Sobald das Telegramm verschickt wurde, wird „LBC_ParamDB“.done für einen Zyklus gesetzt. Die Slaves antworten nun auf den Erhalt des Telegramms, indem sie die Daten aus ihrem Array #data verschicken. Wurden diese seit dem Versand des Send-Telegramms nicht verändert wurden, schicken die Slaves die empfangenen Daten zurück. Die empfangenen Daten der Slaves werden in den Elementen des Arrays „LBC_ParamDB“.slavesArray angezeigt.

7.3 Diagnose

Der Funktionsbaustein „LBC_Main“ gibt an dem Ausgang #status Statusmeldungen aus. Wird ein Fehler erkannt, wird dieser dadurch spezifiziert. Zusätzlich wird an dem Ausgang #statusID die Quelle des Fehlers spezifiziert.

Tabelle 7-5

Status-ID	Status	Bedeutung	Abhilfe
0	16#0000	Es liegen keine Meldungen vor.	
1	16#7001	Main: Kein Master im Netzwerk aktiv	Aktivieren Sie die Master-Funktion an einer Station.
1	16#8001	Main: Station soll Master sein, aber es ist bereits ein Master aktiv.	Überprüfen Sie die Konfiguration Ihrer Teilnehmer und stellen Sie sicher, dass nur ein Master im Netzwerk vorhanden ist.
1	16#8201	Main: Wert am Eingang #KAInterval ist kleiner als 200 ms.	Ein Keep-Alive-Interval größer als 200 ms parametrieren.
1	16#8202	Main: Wert am Eingang #timeout ist kleiner als 100 ms.	Eine Timeout-Zeit größer als 100 ms am parametrieren.
2	16#7400	SlavesManager: Keine Slaves erreichbar	
2	16#7401	SlavesManager: Neuer Slave erreichbar	
2	16#7402	SlavesManager: Timeout beim Warten auf Antwort mindestens eines Slaves.	Überprüfen Sie die folgenden Einstellungen, falls mehr Slaves erreichbar sein sollten: • Sind alle Slaves physikalisch mit dem Netzwerk verbunden? • Ist das Anwenderprogramm in allen Slaves geladen und befinden sich die Slaves im Zustand RUN? • Sind alle Slaves über das Netzwerk zu erreichen?
2	16#7403	SlavesManager: Ein inaktiver Slave ist wieder erreichbar.	
2	16#7404	SlavesManager: Das Slaves-Array ist gefüllt. Es werden keine weiteren Slaves verwaltet.	Ändern Sie die max. Anzahl der zu verwaltenden Slaves im Datentyp „LBC_typeSlavesArray“ und laden
3	16#xxxx	Fehler in „LBC_EstSockets“: Status von TCON oder TDISCON	Nutzen Sie die Online-Hilfe von STEP 7.
4	16#xxxx	Fehler in „LBC_ReceiveManager“: Status von TUSEND oder TURCV.	Nutzen Sie die Online-Hilfe von STEP 7.

Status-ID	Status	Bedeutung	Abhilfe
5	16#xxxx	Fehler in „LBC_SendManager“: Status von TUSEND.	Nutzen Sie die Online-Hilfe von STEP 7.
6	16#xxxx	Fehler in „LBC_SlaveManager“: Status von TURCV.	Nutzen Sie die Online-Hilfe von STEP 7.

8 Literaturhinweise

Tabelle 8-1

	Thema
\1\	Siemens Industry Online Support https://support.industry.siemens.com
\2\	Downloadseite des Beitrags https://support.industry.siemens.com/cs/ww/de/view/20983558

9 Historie

Tabelle 9-1

Version	Datum	Änderung
V1.0	05/2004	Erste Ausgabe
V2.0	12/2015	Neuerstellung des Anwendungsbeispiels
V2.1	02/2016	Begrenzung des Broadcasts auf das Subnetz des Masters