Handbücher/Manuals

art of automation

VIPA
Gesellschaft für Visualisierung
und Prozessautomatisierung mbH

Ohmstraße 4
D-91074 Herzogenaurach
Tel.: +49-9132-744-0
Fax: +49-9132-744-144
Internet: www.vipa.de
E-Mail: Info@vipa.de

Handbuch

VIPA System 200V

CP

Best.-Nr.: VIPA HB97D_CP
Rev. 06/29

Die Angaben in diesem Handbuch erfolgen ohne Gewähr. Änderungen des Inhalts können jederzeit ohne Vorankündigung erfolgen.

Ohmstraße 4, D-91074 Herzogenaurach,
Tel.: +49 (91 32) 744 -0
Fax.: +49 (91 32) 744-144
EMail: info@vipa.de
http://www.vipa.de

Hotline: +49 (91 32) 744-114

Haftungsausschluss

Der Inhalt dieses Handbuchs wurde auf Übereinstimmung mit der beschriebenen Hard- und Software geprüft.

Dennoch können Abweichungen nicht ausgeschlossen werden. Die Angaben in diesem Handbuch werden regelmäßig überprüft und erforderliche Korrekturen sind in den nachfolgenden Auflagen enthalten.

Für Verbesserungsvorschläge sind wir dankbar.

Warenzeichen

VIPA, System 100V, System 200V, System 300V und System 500V

sind eingetragene Warenzeichen der VIPA Gesellschaft für Visualisierung und Prozessautomatisierung mbH.

SIMATIC, STEP und S7-300

sind eingetragenes Warenzeichen der Siemens AG.

Alle ansonsten im Text genannten Warenzeichen sind Warenzeichen der jeweiligen Inhaber und werden als geschützt anerkannt.

Über dieses Handbuch

Das Handbuch beschreibt die bei VIPA erhältlichen System 200V CP 240 Module. Hier finden Sie neben einer Produktübersicht eine detaillierte Beschreibung der einzelnen Module. Sie erhalten Informationen für den Anschluss und die Handhabe der CP 240 Module im System 200V. Am Ende eines Kapitels befinden sich die Technischen Daten der jeweiligen Module.

Überblick

Teil 1: Grundlagen

Im Rahmen dieser Einleitung erfolgt die Vorstellung des System 200V von VIPA als zentrales bzw. dezentrales Automatisierungssystem.

Weiter finden Sie hier auch allgemeine Angaben zum System 200V wie Maße, Montage und Betriebsbedingungen.

Teil 2: Montage und Aufbaurichtlinien

Alle Informationen, die für den Aufbau und die Verdrahtung einer Steuerung aus den Komponenten des Systems 200V erforderlich sind, finden Sie in diesem Kapitel.

Teil 3: Projektierung

In diesem Teil erhalten Sie Informationen über die grundsätzliche Vorgehensweise bei der Projektierung des CP 240. Neben der Einbindung der GSD und Baustein-Bibliothek in den Siemens SIMATIC Manager finden Sie hier alle Hantierungsbausteine beschrieben, die für den Einsatz des CP 240 erforderlich sind.

Teil 4: Kommunikationsprozessor CP 240 - RS232/RS485

In diesem Kapitel finden Sie Informationen über den Aufbau, die Anwendung sowie die Übertragungsprotokolle der Kommunikations-prozessoren CP 240 mit RS232- bzw. RS485-Schnittstelle.

Teil 5: Kommunikationsprozessor CP 240 - EnOcean

Der Einsatz und die Projektierung des CP 240 EnOcean zur Funkübertragung wird in diesem Teil näher erläutert.

Teil 6: Kommunikationsprozessor CP 240 - M-Bus

Inhalt dieses Teils ist die Beschreibung des CP 240 M-Bus. M-Bus (Metering Bus) ist ein genormter Feldbus zur Verbrauchsdatenerfassung von Energie- und Verbrauchszählern wie Wärme-, Wasser-, Strom- und Gaszähler.

Inhaltsverzeichnis

Benutzerhinweise

Zielsetzung und Inhalt

Dieses Handbuch beschreibt Module, die im System 200V eingesetzt werden können. Beschrieben werden Aufbau, Projektierung und Technische Daten.

Zielgruppe

Das Handbuch ist geschrieben für Anwender mit Grundkenntnissen in der Automatisierungstechnik.

Aufbau des Handbuchs

Das Handbuch ist in Kapitel gegliedert. Jedes Kapitel beschreibt eine abgeschlossene Thematik.

Orientierung im Dokument

Als Orientierungshilfe stehen im Handbuch zur Verfügung:

- Gesamt-Inhaltsverzeichnis am Anfang des Handbuchs
- Übersicht der beschriebenen Themen am Anfang jedes Kapitels
- Stichwortverzeichnis (Index) am Ende des Handbuchs

Verfügbarkeit

Das Handbuch ist verfügbar in:

- gedruckter Form auf Papier
- in elektronischer Form als PDF-Datei (Adobe Acrobat Reader)

Piktogramme Signalwörter

Besonders wichtige Textteile sind mit folgenden Piktogrammen und Signalworten ausgezeichnet:

Gefahr!
Unmittelbar drohende oder mögliche Gefahr.
Personenschäden sind möglich.

Achtung!
Bei Nichtbefolgen sind Sachschäden möglich.

Hinweis!
Zusätzliche Informationen und nützliche Tips

Sicherheitshinweise

Bestimmungs-gemäße Verwendung

Das System 200V ist konstruiert und gefertigt für:

- alle VIPA System 200V-Komponenten
- Kommunikation und Prozesskontrolle
- allgemeine Steuerungs- und Automatisierungsaufgaben
- den industriellen Einsatz
- den Betrieb innerhalb der in den technischen Daten spezifizierten Umgebungsbedingungen
- den Einbau in einen Schaltschrank

Gefahr!

Das Gerät ist nicht zugelassen für den Einsatz

- in explosionsgefährdeten Umgebungen (EX-Zone)

Dokumentation

Handbuch zugänglich machen für alle Mitarbeiter in

- Projektierung
- Installation
- Inbetriebnahme
- Betrieb

Vor Inbetriebnahme und Betrieb der in diesem Handbuch beschriebenen Komponenten unbedingt beachten:

- Änderung am Automatisierungssystem nur im spannungslosen Zustand vornehmen!
- Anschluss und Änderung nur durch ausgebildetes Elektro-Fachpersonal
- Nationale Vorschriften und Richtlinien im jeweiligen Verwenderland beachten und einhalten (Installation, Schutzmaßnahmen, EMV ...)

Entsorgung

Zur Entsorgung des Geräts nationale Vorschriften beachten!

Teil 1 Grundlagen

Überblick

Kernthema dieses Kapitels ist die Vorstellung des System 200V von VIPA. In einer Übersicht werden die Möglichkeiten zum Aufbau von zentralen und dezentralen Systemen aufgezeigt.

Auch finden Sie hier allgemeine Angaben zum System 200V wie Maße, Hinweise zur Montage und zu den Umgebungsbedingungen.

Nachfolgend sind beschrieben:

- Vorstellung des System 200V
- Allgemeine Beschreibung, wie Maße, Montage, Betriebssicherheit und Umgebungsbedingungen

Inhalt

Sicherheitshinweise für den Benutzer

Handhabung elektrostatisch gefährdeter Baugruppen

VIPA-Module und Baugruppen sind mit hochintegrierten Bauelementen in MOS-Technik bestückt. Diese Bauelemente sind hoch empfindlich gegenüber Überspannungen, die z.B. bei elektrostatischer Entladung entstehen.

Zur Kennzeichnung dieser gefährdeten Komponenten wird nachfolgendes Symbol verwendet:

Das Symbol befindet sich auf Modulen, Baugruppen, Baugruppenträgern oder auf Verpackungen und weist so auf elektrostatisch gefährdete Komponenten hin.

Elektrostatisch gefährdete Baugruppen können durch Energien und Spannungen zerstört werden, die weit unterhalb der Wahrnehmungsgrenze des Menschen liegen. Hantiert eine Person, die nicht elektrisch entladen ist, mit elektrostatisch gefährdeten Baugruppen, können diese Spannungen auftreten und zur Beschädigung von Bauelementen führen und so die Funktionsweise der Baugruppen beeinträchtigen oder die Baugruppe unbrauchbar machen. Auf diese Weise beschädigte Baugruppen werden in den wenigsten Fällen sofort als fehlerhaft erkannt. Der Fehler kann sich erst nach längerem Betrieb einstellen.

Durch statische Entladung beschädigte Bauelemente können bei Temperaturänderungen, Erschütterungen oder Lastwechseln zeitweilige Fehler zeigen.

Nur durch konsequente Anwendung von Schutzeinrichtungen und verantwortungsbewusste Beachtung der Handhabungsregeln lassen sich Funktionsstörungen und Ausfälle an elektrostatisch gefährdeten Baugruppen wirksam vermeiden.

Versenden von Baugruppen

Verwenden Sie für den Versand immer die Originalverpackung.

Messen und Ändern von elektrostatisch gefährdeten Baugruppen

Bei Messungen an elektrostatisch gefährdeten Baugruppen sind folgende Dinge zu beachten:

- Potenzialfreie Messgeräte sind kurzzeitig zu entladen.
- Verwendete Messgeräte sind zu erden.

Bei Änderungen an elektrostatisch gefährdeten Baugruppen ist darauf zu achten, dass ein geerdeter Lötkolben verwendet wird.

Achtung!

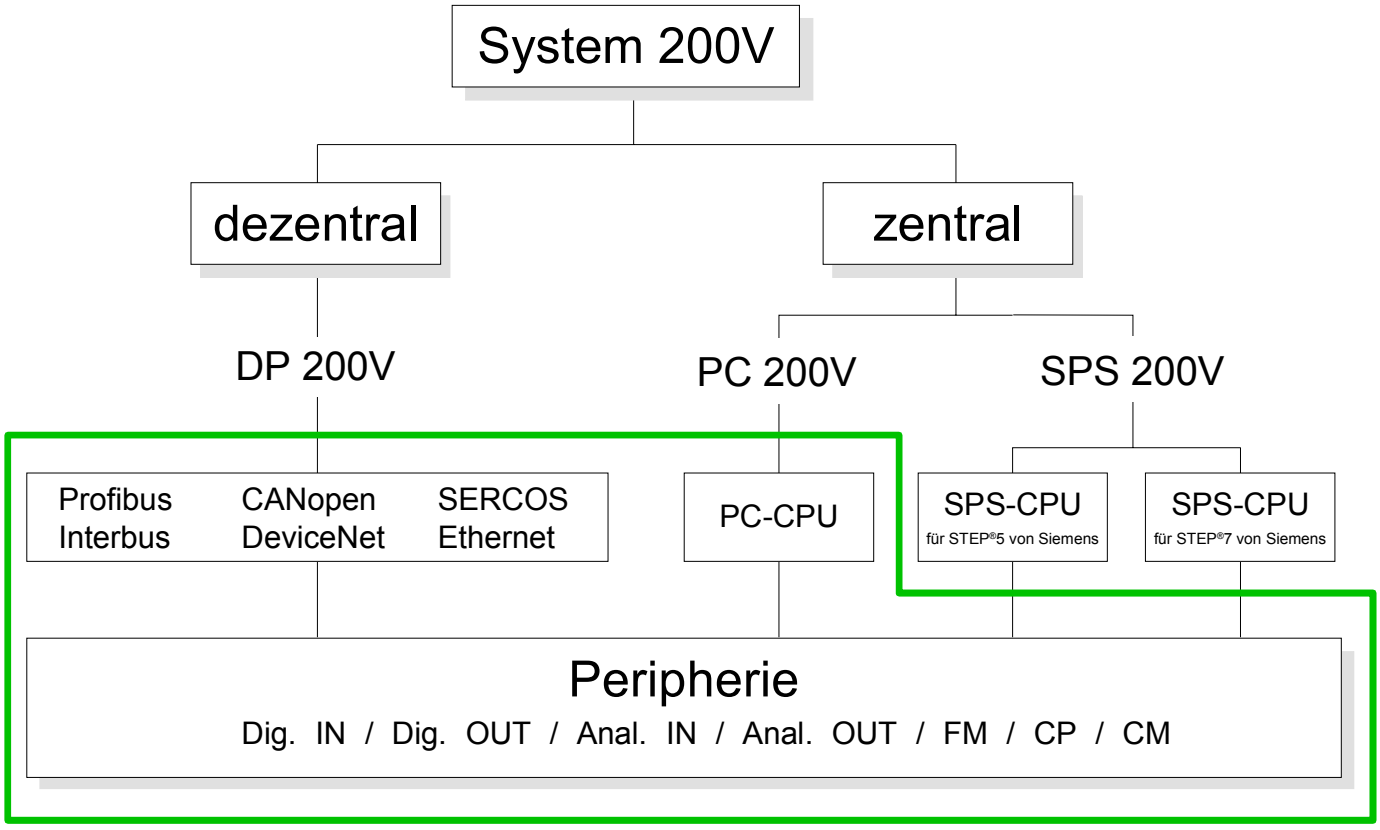
Bei Arbeiten mit und an elektrostatisch gefährdeten Baugruppen ist auf ausreichende Erdung des Menschen und der Arbeitsmittel zu achten.

Übersicht

Das System 200V

Das System 200V ist ein modulares, zentral wie dezentral einsetzbares Automatisierungssystem für Anwendungen im unteren und mittleren Leistungsbereich. Die einzelnen Module werden direkt auf eine 35mm-Normprofilschiene montiert und über Busverbinder, die vorher in die Profilschiene eingelegt werden, gekoppelt.

Die nachfolgende Abbildung soll Ihnen den Leistungsumfang des System 200V verdeutlichen:



Komponenten

Zentrales System

Im System 200V stehen verschiedene SPS-CPUs zur Verfügung. Programmiert wird in STEP®5 oder STEP®7 von Siemens.

CPUs mit integrierter Ethernetanschaltung oder mit zusätzlichen seriellen Schnittstellen garantieren eine komfortable Integration der SPS in ein Netzwerk oder den Anschluss von zusätzlichen Endgeräten.

Das Anwenderprogramm wird im Flash oder einem zusätzlich steckbaren Speichermodul gespeichert.

Bedienen/Beobachten, Steuerungsaufgaben oder andere Dateiverarbeitungsaufgaben können mit der PC-basierenden CPU 288 realisiert werden.

Programmiert wird in C++ oder Pascal.

Die PC 288-CPU ermöglicht einen aktiven Zugriff auf den Rückwandbus und ist so mit allen Peripherie- und Funktionsmodulen des VIPA System 200V als zentrale Steuerung einsetzbar.

Mit einer Zeilenanschaltung ist ein Aufbau des System 200V in bis zu 4 Zeilen möglich.

Dezentrales System

Die SPS-CPUs oder die PC-CPU bilden, in Kombination mit einem Profibus DP-Master, die Basis für ein Profibus-DP-Netzwerk nach DIN 19245-3. Das DP-Netzwerk können Sie mit dem VIPA Projektiertool WinNCS bzw. mit dem SIMATIC Manager projektieren.

Die Anbindung an weitere Feldbusgeräte ermöglichen Slaves für Interbus, CANopen, DeviceNet, SERCOS und Ethernet.

Peripheriemodule

Von VIPA erhalten Sie eine Vielzahl an Peripheriemodulen, wie z.B. für digitale bzw. analoge Ein-/Ausgabe, Zählerfunktionen, Wegmessung, Positionierung und serielle Kommunikation.

Die Peripheriemodule können zentral und dezentral betrieben werden.

Einbindung über GSD-Datei

Die Funktionalität aller Systemkomponenten von VIPA sind in Form von verschiedenen GSD-Dateien verfügbar.

Da die Profibus-Schnittstelle auch softwareseitig standardisiert ist, können wir auf diesem Weg gewährleisten, dass über die Einbindung einer GSD-Datei die Funktionalität in Verbindung mit dem Siemens SIMATIC Manager jederzeit gegeben ist.

Für jede Systemfamilie erhalten Sie eine GSD-Datei. Aktuelle GSD-Dateien finden Sie unter ftp.vipa.de/support.

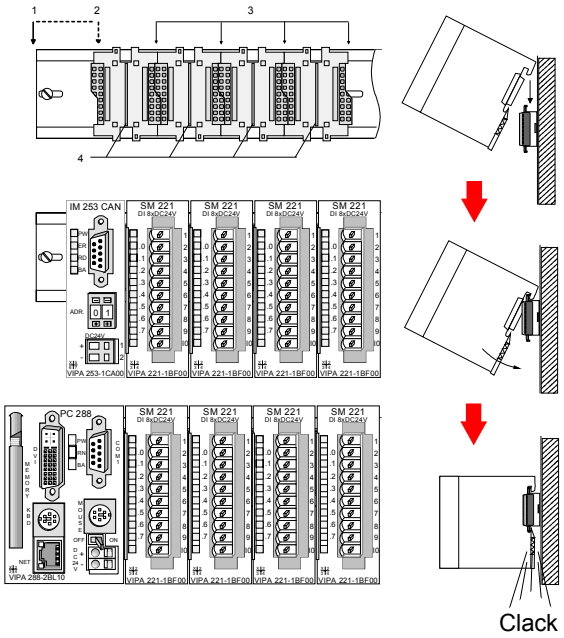
Allgemeine Beschreibung System 200V

Aufbau/Maße

- Normprofil-Hutschiene 35mm
- Peripherie-Module mit seitlich versenkbaren Beschriftungsstreifen
- Maße Grundgehäuse:
  1fach breit: (HxBxT) in mm: 76x25,4x74 in Zoll: 3x1x3
  2fach breit: (HxBxT) in mm: 76x50,8x74 in Zoll: 3x2x3

Montage

Bitte beachten Sie, dass Sie Kopfmodule, wie CPUs, PC und Koppler nur auf Steckplatz 2 bzw. 1 und 2 (wenn doppelt breit) stecken dürfen.



[1] Kopfmodul, wie PC, CPU, Buskoppler, wenn doppelt breit
[2] Kopfmodul, wenn einfach breit
[3] Peripheriemodule
[4] Führungsleisten

Hinweis

Sie können maximal 32 Module stecken, hierbei ist zu beachten, dass der **Summenstrom** von **3,5A** am Rückwandbus nicht überschritten wird!

Bitte montieren Sie Module mit hoher Stromaufnahme direkt neben das Kopfmodul.

Betriebssicherheit

- Anschluss über Federzugklemmen an Frontstecker, Aderquerschnitt 0,08...2,5mm^2 bzw. 1,5 mm^2 (18-fach Stecker)
- Vollisolierung der Verdrahtung bei Modulwechsel
- Potenzialtrennung aller Module zum Rückwandbus
- ESD/Burst gemäß IEC 61000-4-2 / IEC 61000-4-4 (bis Stufe 3)
- Schockfestigkeit gemäß IEC 60068-2-6 / IEC 60068-2-27 (1G/12G)

Umgebungs-bedingungen

- Betriebstemperatur: 0 ... +60°C
- Lagertemperatur: -25 ... +70°C
- Relative Feuchte: 5 ... 95% ohne Betauung
- Lüfterloser Betrieb

Teil 2 Montage und Aufbaurichtlinien

Überblick

In diesem Kapitel finden Sie alle Informationen, die für den Aufbau und die Verdrahtung einer Steuerung aus den Komponenten des Systems 200V erforderlich sind.

Nachfolgend sind beschrieben:

- Allgemeine Übersicht der Komponenten
- Schritte der Montage und Verdrahtung
- EMV-Richtlinien zum Aufbau eines System 200V

Inhalt

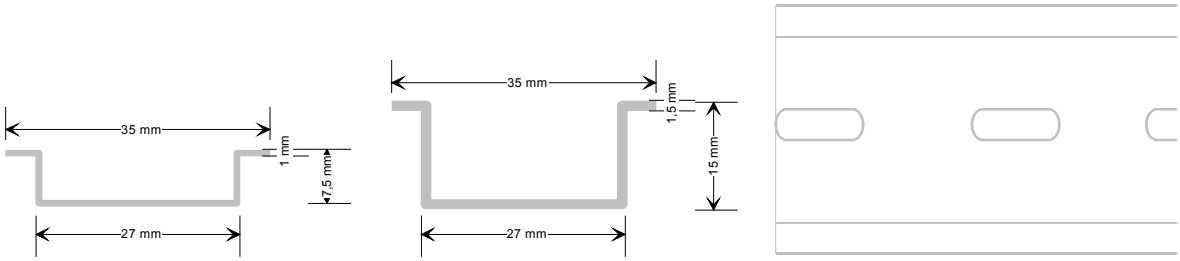
Übersicht

Allgemein

Die einzelnen Module werden direkt auf eine Tragschiene montiert und über Rückwandbusverbinder, die vorher in die Profilschiene eingelegt werden, gekoppelt.

Tragschienen

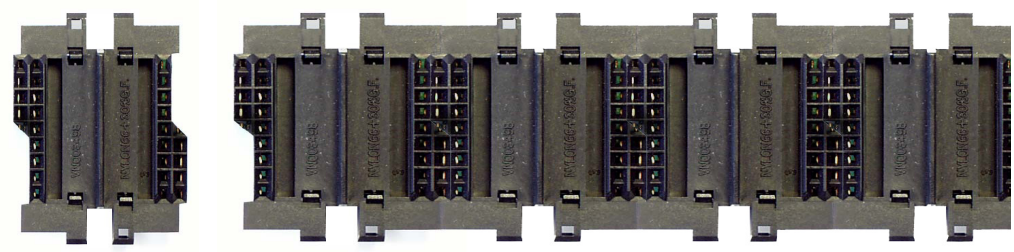
Für die Montage können Sie folgende 35mm-Normprofilschiene verwenden:



Busverbinder

Für die Kommunikation der Module untereinander wird beim System 200V ein Rückwandbusverbinder eingesetzt. Die Rückwandbusverbinder sind isoliert und bei VIPA in 1-, 2-, 4- oder 8facher Breite erhältlich.

Nachfolgend sehen Sie einen 1fach und einen 4fach Busverbinder:

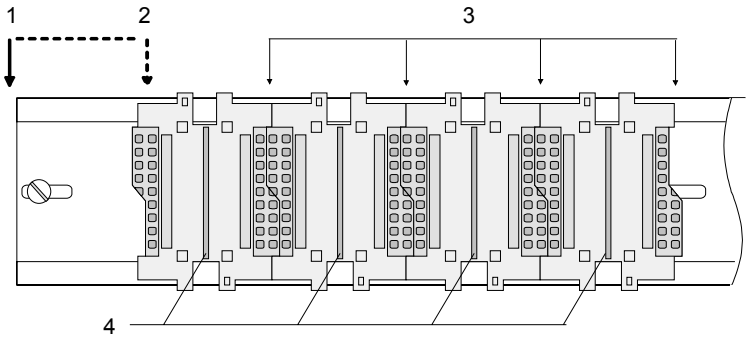


Der Busverbinder wird in die Tragschiene eingelegt, bis dieser sicher einrastet, so dass die Bus-Anschlüsse aus der Tragschiene herausschauen.

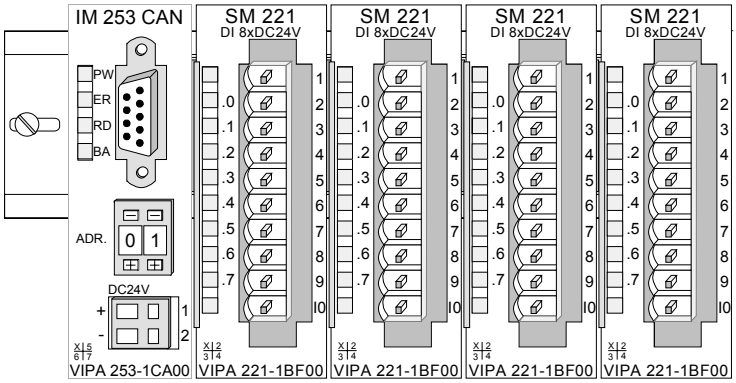
Montage auf Tragschiene

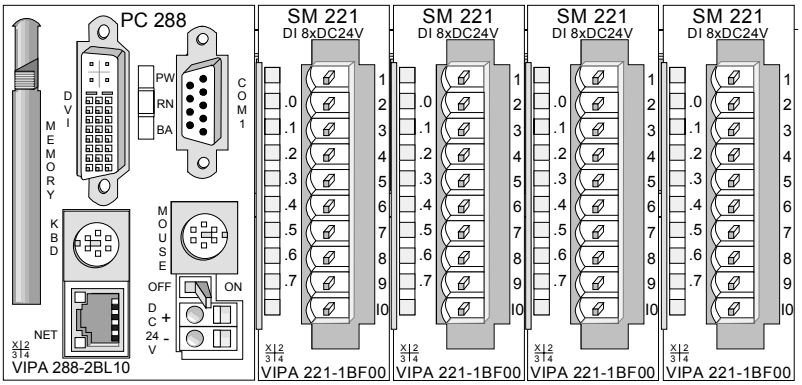
Die nachfolgende Skizze zeigt einen 4fach-Busverbinder in einer Tragschiene und die Steckplätze für die Module.

Die einzelnen Modulsteckplätze sind durch Führungsleisten abgegrenzt.



[1] Kopfmodul, wie PC, CPU, Bus-Koppler, wenn doppelt breit
[2] Kopfmodul (einfach breit)
[3] Peripheriemodule
[4] Führungsleisten





Hinweis

Sie können maximal 32 Module stecken. Hierbei ist zu beachten, dass der **Summenstrom** von **3,5A** am Rückwandbus nicht überschritten wird!

Montage unter Berücksichtigung der Stromaufnahme

- Verwenden Sie möglichst lange Busverbinder.
- Ordnen Sie Module mit hohem Stromverbrauch direkt rechts neben Ihrem Kopfmodul an. Unter ftp.vipa.de/manuals/system200v finden Sie alle Stromaufnahmen des System 200V in einer Liste zusammengefasst.

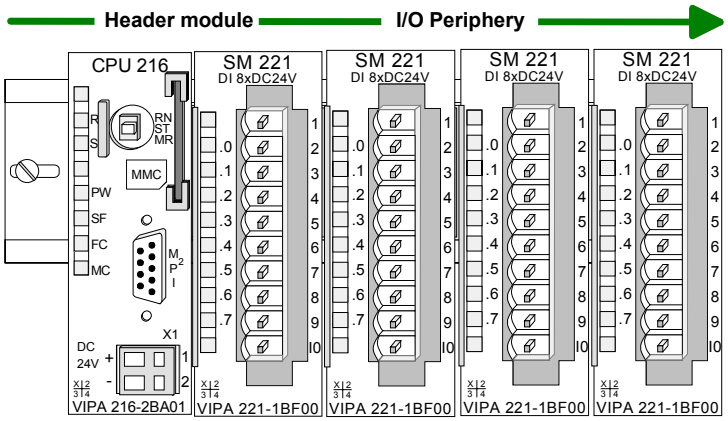
Aufbau waagrecht bzw. senkrecht

Sie haben die Möglichkeit das System 200V waagrecht oder senkrecht aufzubauen. Beachten Sie bitte die hierbei zulässigen Umgebungstemperaturen:

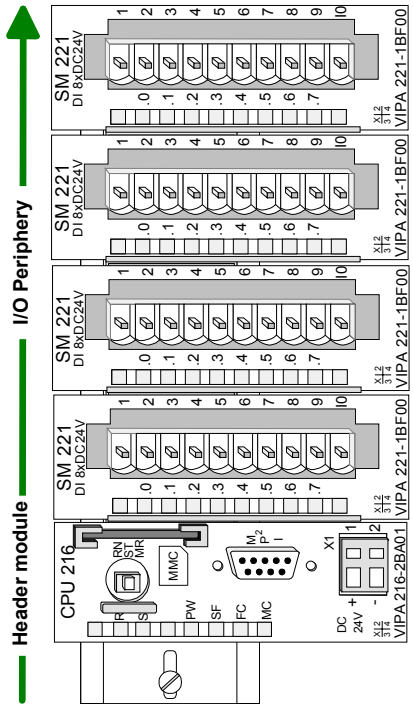
- waagrechter Aufbau: von 0 bis 60°
- senkrechter Aufbau: von 0 bis 40°

Der waagrechte Aufbau beginnt immer links mit einem Kopfmodul (CPU, Buskoppler, PC); rechts daneben sind die Peripherie-Module zu stecken.

Es dürfen maximal 32 Peripherie-Module gesteckt werden.



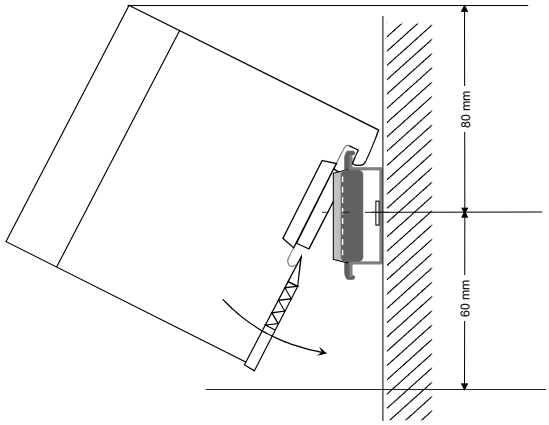
Der senkrechte Aufbau erfolgt gegen den Uhrzeigersinn um 90° gedreht.



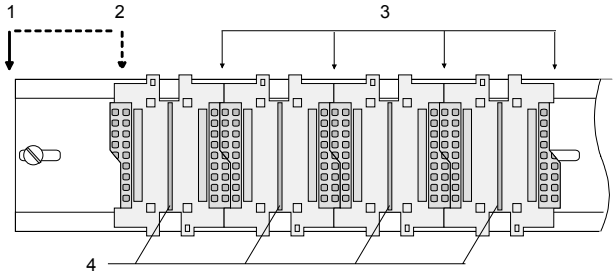
Montage

Bitte bei der Montage beachten!

- Schalten Sie die Stromversorgung aus, bevor Sie Module stecken bzw. abziehen!
- Bitte beachten Sie, dass Sie ab der Mitte der Busschiene nach oben einen Modul-Montageabstand von mindestens 80mm und nach unten von 60mm einhalten.



- Eine Zeile wird immer von links nach rechts aufgebaut und beginnt immer mit einem Kopfmodul (PC, CPU, Buskoppler).



[1] Kopfmodul, wie PC, CPU, Bus-Koppler, wenn doppelt breit
[2] Kopfmodul (einfach breit)
[3] Peripheriemodule
[4] Führungsleisten

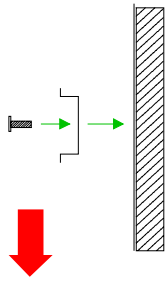
- Module müssen immer direkt nebeneinander gesteckt werden. Lücken zwischen den Modulen sind nicht zulässig, da ansonsten der Rückwandbus unterbrochen ist.
- Ein Modul ist erst dann gesteckt und elektrisch verbunden, wenn es hörbar einrastet.
- Steckplätze rechts nach dem letzten Modul dürfen frei bleiben.

Hinweis!

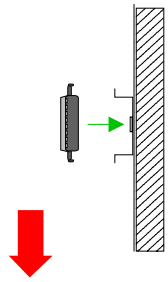
Am Rückwandbus dürfen sich maximal 32 Module befinden. Hierbei ist zu beachten, dass der **Summenstrom** von **3,5A** am Rückwandbus nicht überschritten wird!

Montage Vorgehensweise

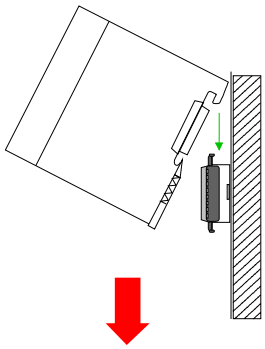
Die nachfolgende Abfolge stellt die Montageschritte in der Seitenansicht dar.



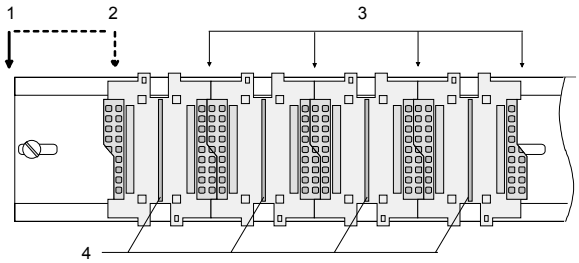
- Montieren Sie die Tragschiene! Bitte beachten Sie, dass Sie ab der Mitte der Busschiene nach oben einen Modul-Montageabstand von mindestens 80mm und nach unten von 60mm einhalten.



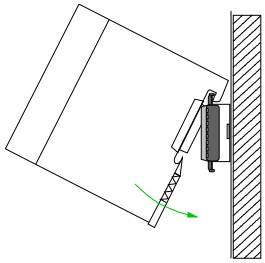
- Drücken Sie den Busverbinder in die Tragschiene, bis dieser sicher einrastet, so dass die Bus-Anschlüsse aus der Tragschiene herausschauen. Sie haben nun die Grundlage zur Montage Ihrer Module.



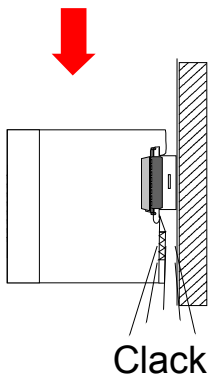
- Beginnen Sie ganz links mit dem Kopfmodul, wie CPU, PC oder Buskoppler und stecken Sie rechts daneben Ihre Peripherie-Module.



[1] Kopfmodul, wie PC, CPU, Bus-Koppler, wenn doppelt breit
[2] Kopfmodul (einfach breit)
[3] Peripheriemodule
[4] Führungsleisten



- Setzen Sie das zu steckende Modul von oben in einem Winkel von ca. 45Grad auf die Tragschiene und drehen Sie das Modul nach unten, bis es hörbar auf der Tragschiene einrastet. Nur bei eingerasteten Modulen ist eine Verbindung zum Rückwandbus sichergestellt.

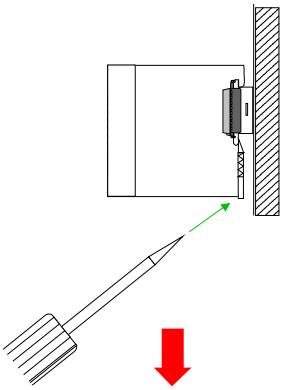


Achtung!

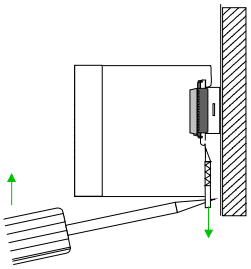
Module dürfen nur im spannungslosen Zustand gesteckt bzw. gezogen werden!

Demontage Vorgehensweise

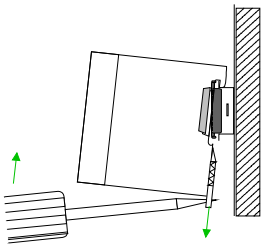
Die nachfolgende Abfolge stellt die Schritte zur Demontage in der Seitenansicht dar.



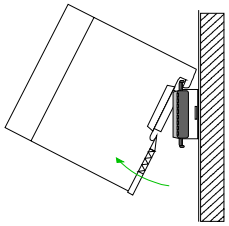
- Zur Demontage befindet sich am Gehäuseunterteil eine gefederter Demontageschlitz.
- Stecken Sie, wie gezeigt, einen Schraubendreher in den Demontage-schlitz.

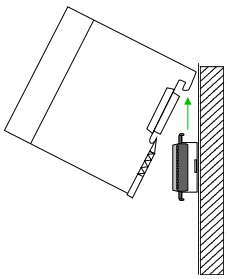


- Durch Druck des Schraubendrehers nach oben wird das Modul entriegelt.



- Ziehen Sie nun das Modul nach vorn und ziehen Sie das Modul mit einer Drehung nach oben ab.





Achtung!

Module dürfen nur im spannungslosen Zustand gesteckt bzw. gezogen werden!

Bitte beachten Sie, dass durch die Demontage von Modulen der Rückwandbus an der entsprechenden Stelle unterbrochen wird!

Verdrahtung

Übersicht

Die meisten Peripherie-Module besitzen einen 10poligen bzw. 18poligen Steckverbinder. Über diesen Steckverbinder werden Signal- und Versorgungsleitungen mit den Modulen verbunden.

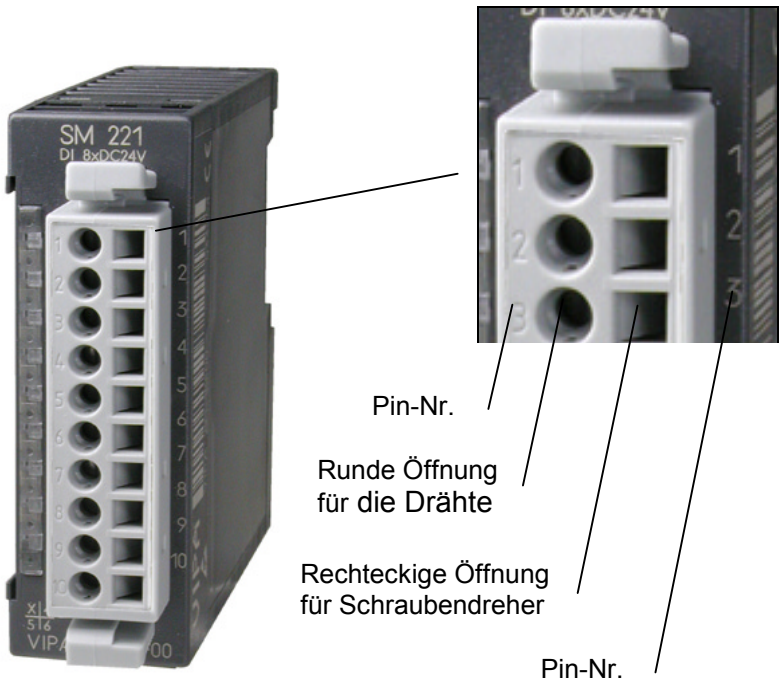
Bei der Verdrahtung werden Steckverbinder mit Federklemmtechnik eingesetzt.

Die Verdrahtung mit Federklemmtechnik ermöglicht einen schnellen und einfachen Anschluss Ihrer Signal- und Versorgungsleitungen.

Im Gegensatz zur Schraubverbindung, ist diese Verbindungsart erschütterungssicher. Die Steckerbelegung der Peripherie-Module finden Sie in der Beschreibung zu den Modulen.

Sie können Drähte mit einem Querschnitt von 0,08mm^2 bis 2,5mm^2 (bis 1,5mm^2 bei 18poligen) anschließen.

Folgende Abbildung zeigt ein Modul mit einem 10poligen Steckverbinder.

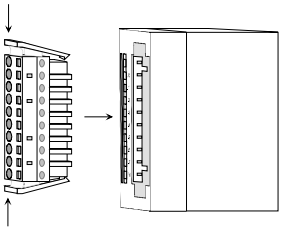


Hinweis!

Die Federklemme wird zerstört, wenn Sie den Schraubendreher in die Öffnung für die Leitungen stecken!

Drücken Sie den Schraubendreher nur in die rechteckigen Öffnungen des Steckverbinders!

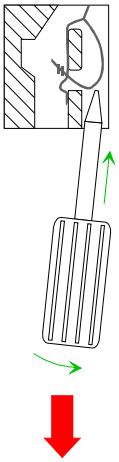
Verdrahtung
Vorgehensweise



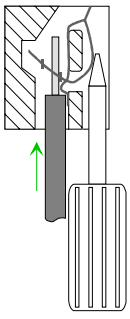
- Stecken Sie den Steckverbinder auf das Modul bis dieser hörbar einrastet. Drücken Sie hierzu während des Steckens, wie gezeigt, die beiden Verriegelungsklinken zusammen.

  Der Steckerverbinder ist nun in einer festen Position und kann leicht verdrahtet werden.

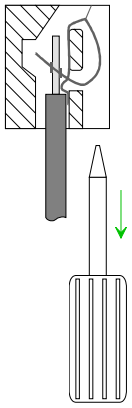
Die nachfolgende Abfolge stellt die Schritte der Verdrahtung in der Draufsicht dar.



- Zum Verdrahten stecken Sie, wie in der Abbildung gezeigt, einen passenden Schraubendreher leicht schräg in die rechteckige Öffnung.
- Zum Öffnen der Kontaktfeder müssen Sie den Schraubendreher in die entgegengesetzte Richtung drücken und halten.



- Führen Sie durch die runde Öffnung Ihren abisolierten Draht ein. Sie können Drähte mit einem Querschnitt von 0,08mm^2 bis 2,5mm^2 (bei 18poligen Steckverbindern bis 1,5mm^2) anschließen.



- Durch Entfernen des Schraubendrehers wird der Draht über einen Federkontakt sicher mit dem Steckverbinder verbunden.

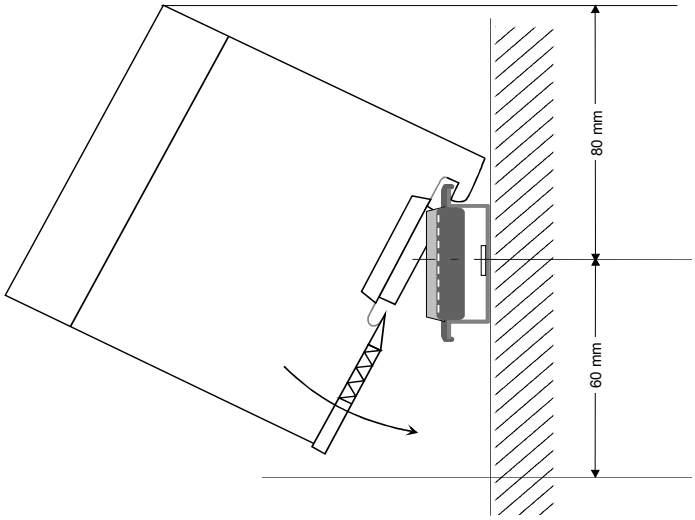
Verdrahten Sie zuerst die Versorgungsleitungen (Spannungsversorgung) und dann die Signalleitungen (Ein- und Ausgänge)!

Einbaumaße

Übersicht Hier finden Sie alle wichtigen Maße des System 200V.

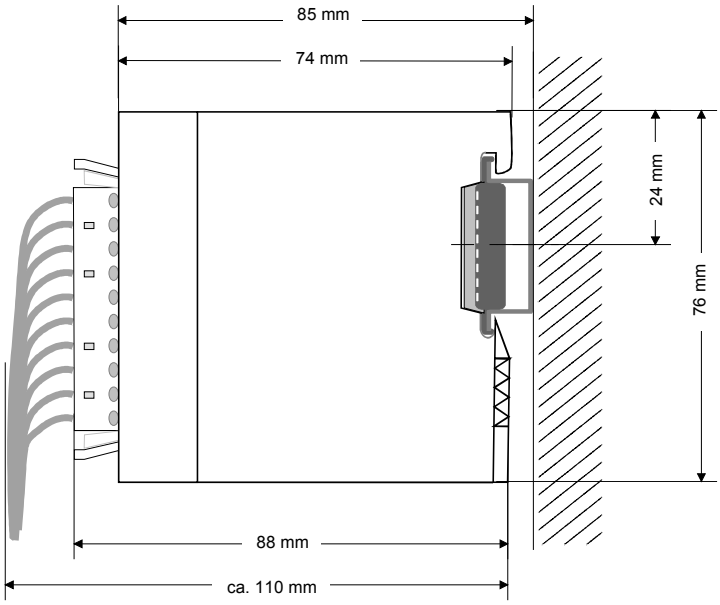
Maße Grundgehäuse
1fach breit (HxBxT) in mm: 76 x 25,4 x 74
2fach breit (HxBxT) in mm: 76 x 50,8 x 74

Montagemaße

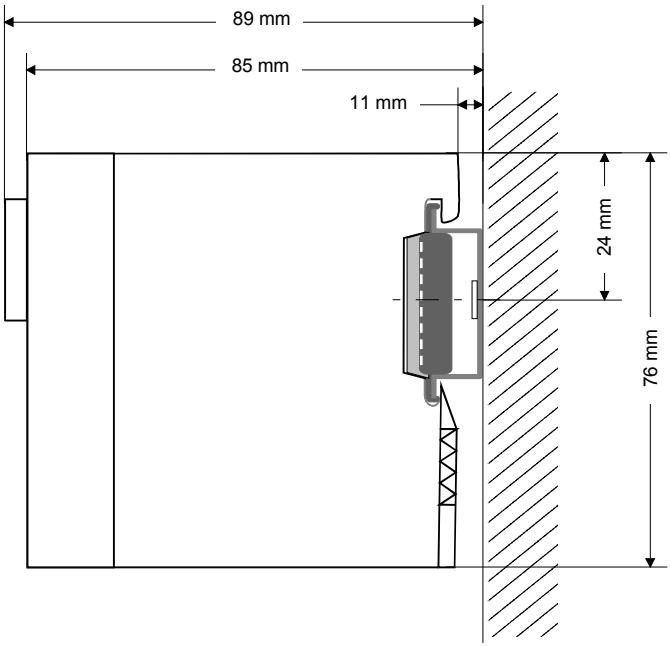


Maße montiert und verdrahtet

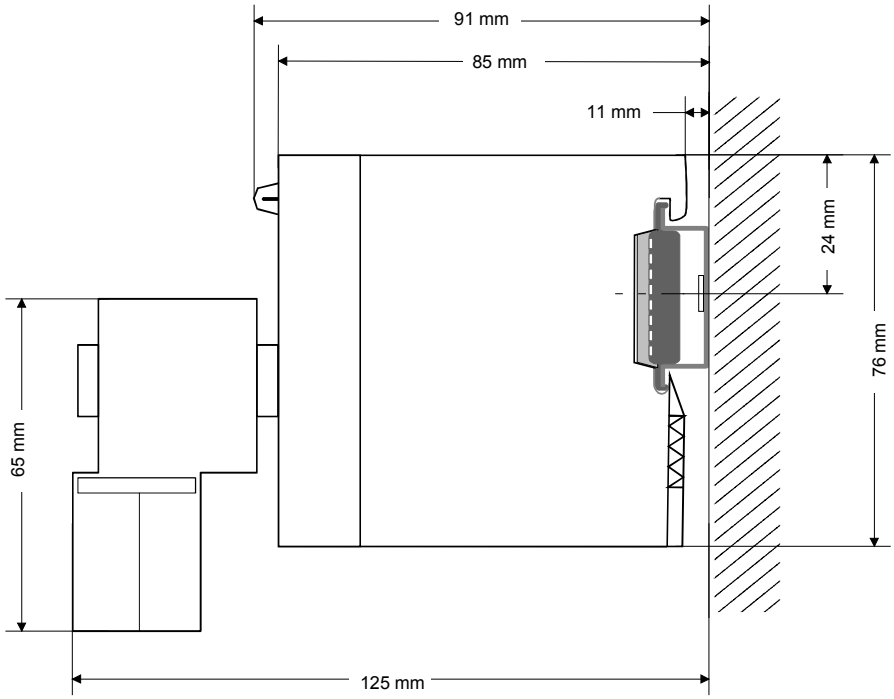
Ein- / Ausgabe-module



Funktionsmodule



CPUs (hier mit VIPA EasyConn)



Aufbaurichtlinien

Allgemeines

Die Aufbaurichtlinien enthalten Informationen über den störsicheren Aufbau des System 200V. Es wird beschrieben, wie Störungen in Ihre Steuerung gelangen können, wie die elektromagnetische Verträglichkeit (EMV) sicher gestellt werden kann und wie bei der Schirmung vorzugehen ist.

Was bedeutet EMV?

Unter Elektromagnetischer Verträglichkeit (EMV) versteht man die Fähigkeit eines elektrischen Gerätes, in einer vorgegebenen elektromagnetischen Umgebung fehlerfrei zu funktionieren ohne vom Umfeld beeinflusst zu werden bzw. das Umfeld in unzulässiger Weise zu beeinflussen.

Alle System 200V Komponenten sind für den Einsatz in rauen Industrieumgebungen entwickelt und erfüllen hohe Anforderungen an die EMV. Trotzdem sollten Sie vor der Installation der Komponenten eine EMV-Planung durchführen und mögliche Störquellen in die Betrachtung einbeziehen.

Mögliche Störeinwirkungen

Elektromagnetische Störungen können sich auf unterschiedlichen Pfaden in Ihre Steuerung einkoppeln:

- Felder
- E/A-Signalleitungen
- Bussystem
- Stromversorgung
- Schutzleitung

Je nach Ausbreitungsmedium (leitungsgebunden oder -ungebunden) und Entfernung zur Störquelle gelangen Störungen über unterschiedliche Kopplungsmechanismen in Ihre Steuerung.

Man unterscheidet:

- galvanische Kopplung
- kapazitive Kopplung
- induktive Kopplung
- Strahlungskopplung

Grundregeln zur Sicherstellung der EMV

Häufig genügt zur Sicherstellung der EMV das Einhalten einiger elementarer Regeln. Beachten Sie beim Aufbau der Steuerung deshalb die folgenden Grundregeln.

- Achten sie bei der Montage Ihrer Komponenten auf eine gut ausgeführte flächenhafte Massung der inaktiven Metallteile.
  - Stellen sie eine zentrale Verbindung zwischen der Masse und dem Erde/Schutzleitersystem her.
  - Verbinden Sie alle inaktiven Metallteile großflächig und impedanzarm.
  - Verwenden Sie nach Möglichkeit keine Aluminiumteile. Aluminium oxidiert leicht und ist für die Massung deshalb weniger gut geeignet.
- Achten Sie bei der Verdrahtung auf eine ordnungsgemäße Leitungsführung.
  - Teilen Sie die Verkabelung in Leitungsgruppen ein. (Starkstrom, Stromversorgungs-, Signal- und Datenleitungen).
  - Verlegen Sie Starkstromleitungen und Signal- bzw. Datenleitungen immer in getrennten Kanälen oder Bündeln.
  - Führen sie Signal- und Datenleitungen möglichst eng an Masseflächen (z.B. Tragholme, Metallschienen, Schrankbleche).
- Achten sie auf die einwandfreie Befestigung der Leitungsschirme.
  - Datenleitungen sind geschirmt zu verlegen.
  - Analogleitungen sind geschirmt zu verlegen. Bei der Übertragung von Signalen mit kleinen Amplituden kann das einseitige Auflegen des Schirms vorteilhaft sein.
  - Legen Sie die Leitungsschirme direkt nach dem Schrankeintritt großflächig auf eine Schirm-/Schutzleiterschiene auf und befestigen Sie die Schirme mit Kabelschellen.
  - Achten Sie darauf, dass die Schirm-/Schutzleiterschiene impedanzarm mit dem Schrank verbunden ist.
  - Verwenden Sie für geschirmte Datenleitungen metallische oder metallisierte Steckergehäuse.
- Setzen Sie in besonderen Anwendungsfällen spezielle EMV-Maßnahmen ein.
  - Beschalten Sie alle Induktivitäten mit Löschgliedern, die von System 200V Modulen angesteuert werden.
  - Benutzen Sie zur Beleuchtung von Schränken Glühlampen und vermeiden Sie Leuchtstofflampen.
- Schaffen Sie ein einheitliches Bezugspotential und erden Sie nach Möglichkeit alle elektrischen Betriebsmittel.
  - Achten Sie auf den gezielten Einsatz der Erdungsmaßnahmen. Das Erden der Steuerung dient als Schutz- und Funktionsmaßnahme.
  - Verbinden Sie Anlagenteile und Schränke mit dem System 200V sternförmig mit dem Erde/Schutzleitersystem. Sie vermeiden so die Bildung von Erdschleifen.
  - Verlegen Sie bei Potenzialdifferenzen zwischen Anlagenteilen und Schränken ausreichend dimensionierte Potenzialausgleichsleitungen.

Schirmung von Leitungen

Elektrische, magnetische oder elektromagnetische Störfelder werden durch eine Schirmung geschwächt; man spricht hier von einer Dämpfung.

Über die mit dem Gehäuse leitend verbundene Schirmschiene werden Störströme auf Kabelschirme zur Erde hin abgeleitet. Hierbei ist darauf zu achten, dass die Verbindung zum Schutzleiter impedanzarm ist, da sonst die Störströme selbst zur Störquelle werden.

Bei der Schirmung von Leitungen ist folgendes zu beachten:

- Verwenden Sie möglichst nur Leitungen mit Schirmgeflecht.
- Die Deckungsdichte des Schirmes sollte mehr als 80% betragen.
- In der Regel sollten Sie die Schirme von Leitungen immer beidseitig auflegen. Nur durch den beidseitigen Anschluss der Schirme erreichen Sie eine gute Störunterdrückung im höheren Frequenzbereich.

  Nur im Ausnahmefall kann der Schirm auch einseitig aufgelegt werden. Dann erreichen Sie jedoch nur eine Dämpfung der niedrigen Frequenzen. Eine einseitige Schirmanbindung kann günstiger sein, wenn:
  - die Verlegung einer Potenzialausgleichsleitung nicht durchgeführt werden kann.
  - Analogsignale (einige mV bzw. µA) übertragen werden.
  - Folienschirme (statische Schirme) verwendet werden.
- Benutzen Sie bei Datenleitungen für serielle Kopplungen immer metallische oder metallisierte Stecker. Befestigen Sie den Schirm der Datenleitung am Steckergehäuse. Schirm **nicht** auf den PIN 1 der Steckerleiste auflegen!
- Bei stationärem Betrieb ist es empfehlenswert, das geschirmte Kabel unterbrechungsfrei abzuisolieren und auf die Schirm-/Schutzleiterschiene aufzulegen.
- Benutzen Sie zur Befestigung der Schirmgeflechte Kabelschellen aus Metall. Die Schellen müssen den Schirm großflächig umschließen und guten Kontakt ausüben.
- Legen Sie den Schirm direkt nach Eintritt der Leitung in den Schrank auf eine Schirmschiene auf. Führen Sie den Schirm bis zum System 200V Modul weiter, legen Sie ihn dort jedoch **nicht** erneut auf!

Bitte bei der Montage beachten!

Bei Potenzialdifferenzen zwischen den Erdungspunkten kann über den beidseitig angeschlossenen Schirm ein Ausgleichsstrom fließen.

Abhilfe: Potenzialausgleichsleitung

Teil 3 Projektierung

Überblick

Diese Kapitel befasst sich allgemein mit der Projektierung und Programmierung eines CP 240. Detaillierte Informationen zur Projektierung eines speziellen CP 240 finden Sie als Beispielprojektierung in dem entsprechenden Kapitel.

Nach einem Schnelleinstieg erhalten Sie Informationen, wie Sie GSD-Dateien und Bausteinbibliotheken in Ihren Siemens SIMATIC Manager einbinden.

Mit einer Beschreibung der Standard-Hantierungsbausteine für die CP-Kommunikation endet das Kapitel.

Nachfolgend sind beschrieben:

- Schnelleinstieg
- Einbindung von GSD-Dateien und Hantierungsbausteinen im Siemens SIMATIC Manager
- Projektierung
- Standard Hantierungsbausteine

Inhalt

Schnelleinstieg

Übersicht

Die Adresszuordnung und die Parametrierung des CP 240 erfolgt im Siemens SIMATIC Manager in Form eines virtuellen Profibus-Systems. Hierzu ist die Einbindung der VIPA_21x.gsd (ab V. 1.67) erforderlich.

Für die Kommunikation zwischen Ihrer CPU und dem CP 240 sind Hantierungsbausteine in Form einer Bibliothek verfügbar, die Sie in Ihren Siemens SIMATIC Manager einbinden können.

Vorgehensweise

Vorbereitung

- Starten Sie den Siemens SIMATIC Manager mit einem neuen Projekt
- Binden Sie die VIPA_21x.gsd ein. Verwenden Sie hierbei eine GSD-Version ab V. 1.67.
- Binden Sie die Bausteinbibliothek ein, indem Sie die *FX000002_Vxxx.exe* ausführen und die Datei VIPA.ZIP dearchivieren.
- Öffnen Sie die Bibliothek und übertragen Sie FC0, FC1 und FC9 in Ihr Projekt

Hardware-Konfiguration

Für die Hardwarekonfiguration verfahren Sie auf die gleiche Weise wie im Handbuch HB97 - CPU beschrieben:

- Projektieren Sie ein Profibus-DP-Mastersystem mit der Siemens CPU 315-2DP (6ES7 315-2AF03 V1.2) und legen Sie ein Profibus-Subnetz an.
- Binden Sie an das Master-System aus dem Hardware-Katalog das Slave-System "VIPA_CPU21x" an. Sie finden das Slave-System im Hardware-Katalog unter
  Profibus-DP > Weitere Feldgeräte > I/O > VIPA_System_200V.
- Geben Sie dem Slave-System die Adresse 1. Hiermit identifiziert die VIPA CPU das System als zentrales Peripherie-System.
- Platzieren Sie in diesem Slave-System in der gesteckten Reihenfolge Ihre Module. Beginnen sie mit der CPU auf dem 1. Steckplatz.
- Binden Sie danach Ihre System 200V Module und an der entsprechenden Stelle Ihren CP 240 ein.
- Parametrieren Sie ggf. Ihren CP 240.

SPS-Programm

Für die Kommunikation ist ein SPS-Programm erforderlich. Hierzu können Sie folgende Hantierungsbausteine verwenden:

FC 0	SEND	Datenausgabe CPU an CP 240
FC 1	RECEIVE	Datenempfang vom CP 240
FC 8	STEUERBIT	Zugriff auf serielle Modemleitungen
FC 9	SYNCHRON_RESET	Synchronisation zwischen CPU und CP 240
FC 11	ASCII_FRAGMENT	Fragmentierter ASCII-Datenempfang

GSD und FCs einbinden

Projektierung über GSD

Adresszuordnung und die Parametrierung des CP 240 erfolgt im Siemens SIMATIC Manager in Form eines virtuellen Profibus-Systems. Da die Profibus-Schnittstelle softwareseitig standardisiert ist, können wir auf diesem Weg gewährleisten, dass über die Einbindung einer GSD-Datei die Funktionalität in Verbindung mit dem SIMATIC Manager von Siemens jederzeit gegeben ist. Ihr Projekt übertragen Sie über MPI in die CPU.

GSD einbinden

Folgende Schritte sind zur Installation der GSD erforderlich:

- Kopieren Sie die mitgelieferte VIPA GSD-Datei **VIPA_21X.GSD** (V. 1.31 oder höher) in Ihr GSD-Verzeichnis ... \siemens\step7\s7data\gsd
- Starten Sie den Hardware-Konfigurator von Siemens
- Schließen Sie alle Projekte
- Gehen Sie auf **Extras** > *Neue GSD-Datei installieren*
- Geben Sie hier **VIPA_21X.gsd** an

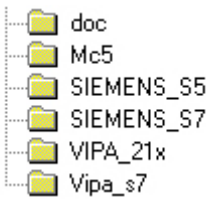
Die Module des System 200V von VIPA sind jetzt im Hardwarekatalog integriert und können projektiert werden.

VIPA-Bibliothek einbinden

Die VIPA spezifischen FCs sind in Form einer Bibliotheken verfügbar. Sie finden diese unter anderem auf der CD "SW-ToolDemo" bzw. unter ftp.vipa.de. Die Bibliotheken liegen als selbstentpackende exe-Datei vor.

Sobald Sie VIPA spezifische SFCs verwenden möchten, sind diese in Ihr Projekt nach folgender Vorgehensweise zu importieren:

- Zum Entpacken der Datei FX000002_Vxxx.exe ausführen:
  Durch einen Doppelklick auf die Datei FX000002_Vxxx.exe starten Sie das integrierte Entpackprogramm. Geben Sie ein Zielverzeichnis an, in das die Dateien zu entpacken sind und wählen Sie [Extrahieren] an.
- Bibliothek "dearchivieren":
  Zur Dearchivierung Ihrer FC-Bibliothek starten Sie den Siemens SIMATIC Manager. Über **Datei** > *Dearchivieren* öffnen Sie ein Dialogfenster zur Auswahl des Archivs. Sie finden die FC-Bibliothek in der Verzeichnis-Struktur unter VIPA_S7. Die Datei lautet **VIPA.ZIP**.



Wählen Sie VIPA.ZIP an und klicken Sie auf [Öffnen].

- Geben Sie ein Zielverzeichnis an, in dem die Bausteine abzulegen sind. Mit [OK] startet der Entpackvorgang.
- Bibliothek öffnen und FCs in Projekt übertragen:
  Öffnen Sie die Bibliothek nach dem Entpackvorgang. Öffnen Sie Ihr Projekt und kopieren Sie die erforderlichen FCs aus der Bibliothek in das Verzeichnis "Bibliothek" Ihres Projekts. Nun haben Sie in Ihrem Anwenderprogramm Zugriff auf die VIPA spezifischen Bausteine.

Projektierung

Allgemein

Die Adresszuordnung und die Parametrierung der direkt gesteckten System 200V Module erfolgt im SIMATIC Manager von Siemens in Form eines virtuellen Profibus-Systems. Ihr Projekt übertragen Sie seriell über die MPI-Schnittstelle oder über MMC in Ihre CPU.

Voraussetzung

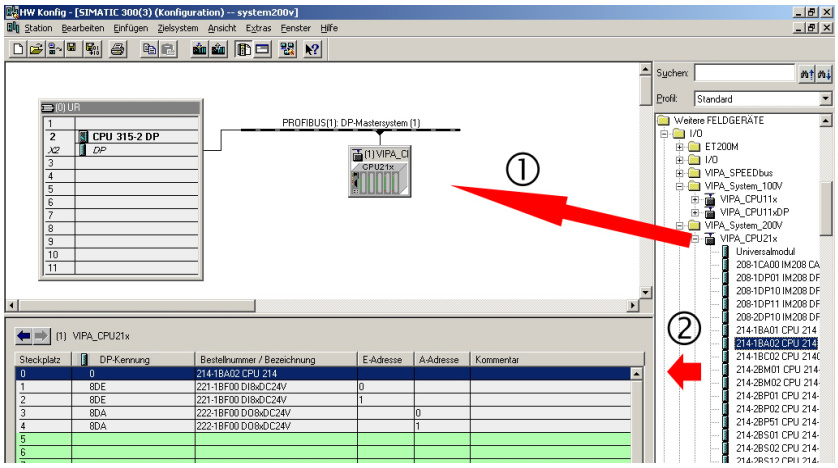
Für die Projektierung der CPU werden fundierte Kenntnisse im Umgang mit dem SIMATIC Manager und dem Hardware-Konfigurator von Siemens vorausgesetzt!

Folgende Voraussetzungen müssen für die Projektierung erfüllt sein:

- SIMATIC Manager von Siemens auf PC bzw. PG installiert
- GSD-Dateien in Hardware-Konfigurator von Siemens eingebunden
- Projekt kann in CPU übertragen werden (serielle z.B. "Green Cable" oder MMC)

Hardware-Konfiguration

- Starten Sie den Hardware-Konfigurator von Siemens mit einem neuen Projekt und fügen Sie aus dem Hardware-Katalog eine Profilschiene ein.
- Platzieren Sie auf dem ersten möglichen Steckplatz die CPU 315-2DP (6ES7 315-2AF03 V1.2) von Siemens.
- Sofern Ihre CPU 21x einen Profibus-DP-Master integriert hat, können Sie diesen jetzt mit Profibus vernetzen und Ihre DP-Slaves anbinden.
- Erzeugen Sie ein Profibus-Subnetz (falls noch nicht vorhanden)
- Hängen Sie an das Subnetz das System "VIPA_CPU21x". Sie finden dies im Hardware-Katalog unter *PROFIBUS DP* > *Weitere Feldgeräte* > *IO* > VIPA_System_200V. Geben Sie diesem Slave die **Profibus-Adresse 1**.
- Platzieren Sie in Ihrem Konfigurator **immer auf dem 1. Steckplatz** die CPU 21x, die Sie einsetzen, indem Sie diese dem Hardware-Katalog entnehmen.
- Binden Sie danach Ihre System 200V Module in der gesteckten Reihenfolge und an der entsprechenden Stelle Ihren CP 240 ein.
- Parametrieren Sie ggf. Ihren CP 240.
- Sichern Sie Ihr Projekt.



SPS-Programm

Für die nachfolgend gezeigte Kommunikation zwischen CPU und CP 240 kommen folgende Hantierungsbausteine zum Einsatz:

FC 0	SEND	Datenausgabe CPU an CP 240
FC 1	RECEIVE	Datenempfang vom CP 240
FC 9	SYNCHRON_RESET	Synchronisation zwischen CPU und CP 240

Die Hantierungsbausteine sind als Bibliothek verfügbar und können, wie weiter oben gezeigt, im Siemens SIMATIC Manager eingebunden werden.

Eine nähere Beschreibung der Hantierungsbausteine finden Sie auf den Folgeseiten.

Ihr SPS-Programm sollte nach folgender Struktur aufgebaut sein:

```
OB1:

      CALL  FC     9             //Synchron aufrufen
       ADR       :=0             //1. DW im SEND/EMPF_DB
       TIMER_NR  :=T2            //Wartezeit Synchron
       ANL       :=M3.0          //Anlauf erfolgt
       NULL      :=M3.1          //Zwischenmerker
       RESET     :=M3.2          //Baugruppenreset ausführen
       STEUERB_S :=MB2           //Steuerbits Sende_FC
       STEUERB_R :=MB1           //Steuerbits Receive_FC
      U     M      3.0           //solange Anlauf keine
                                 //SEND/RECEIVE Bearbeitung
      BEB

      CALL  FC     1             //Receive Daten
       ADR          :=0          //1. DW im SEND/EMPF_DB
       _DB          :=DB11       //Empfang_DB Telegramm
       ABD          :=W#16#14    //1. DW Empfangspuffer (DW20)
       ANZ          :=MW10       //Anzahl empfangener Daten
       EMFR         :=M1.0       //Empfang fertig
       PAFE         :=MB12       //Fehlerbyte
       GEEM         :=MW100      //Interne Daten
       ANZ_INT      :=MW102      //Interne Daten
       empf_laeuft  :=M1.1       //Interne Daten
       letzter_block:=M1.2       //Interne Daten
       fehl_empf    :=M1.3       //Interne Daten
      U     M      1.0           //Empfang fertig
      R     M      1.0           //loesche Empfang fertig
      CALL  FC     0             //Sende Daten
       ADR          :=0          //1. DW im SEND/EMPF_DB
       _DB          :=DB10       //Sende_DB Telegramm
       ABD          :=W#16#14    //1. DW Sendepuffer (DW20)
       ANZ          :=MW14       //Anzahl zu sendender Daten
       FRG          :=M2.0       //Senden fertig angeben
       PAFE         :=MB16       //Fehlerbyte
       GESE         :=MW104      //Interne Daten
       ANZ_INT      :=MW106      //Interne Daten
       ende_kom     :=M2.1       //Interne Daten
       letzter_block:=M2.2       //Interne Daten
       senden_laeuft:=M2.3       //Interne Daten
       fehler_kom   :=M2.4       //Interne Daten


OB100:

      UN    M      3.0
      S     M      3.0           //Anlauf der CPU erfolgt
```

Projekt übertragen

Die Datenübertragung erfolgt über MPI. Sollte Ihr Programmiergerät keine MPI-Schnittstelle besitzen, können Sie für eine serielle Punkt-zu-Punkt-Übertragung von Ihrem PC an MPI das "Green Cable" von VIPA verwenden.

Das "Green Cable" hat die Best.-Nr. VIPA 950-0KB00 und darf nur bei den VIPA CPUs mit MP^{2}I-Schnittstelle eingesetzt werden.

Bitte beachten Sie hierzu die Hinweise zum Green Cable in den Grundlagen!

- Verbinden Sie Ihr PG mit der CPU.
- Mit **Zielsystem** > *Laden in Baugruppe* in Ihrem Projektiertool übertragen Sie Ihr Projekt in die CPU.
- Stecken Sie eine MMC und übertragen Sie mit **Zielsystem** > *RAM nach ROM kopieren* Ihr Anwenderprogramm auf die MMC.
- Während des Schreibvorgangs blinkt die "MC"-LED auf der CPU. Systembedingt wird zu früh ein erfolgter Schreibvorgang gemeldet. Der Schreibvorgang ist erst beendet, wenn die LED erlischt.

Was ist das Green Cable ?

Das Green Cable ist ein grünes Verbindungskabel, das ausschließlich zum Einsatz an VIPA System-Komponenten konfektioniert ist.

Mit dem Green Cable können Sie:

- Projekte Punkt-zu-Punkt seriell übertragen
- Firmware-Updates der CPUs und Feldbus-Master durchführen

Wichtige Hinweise zum Einsatz des Green Cable

Bei Nichtbeachtung der nachfolgenden Hinweise können Schäden an den System-Komponenten entstehen.

Für Schäden, die aufgrund der Nichtbeachtung dieser Hinweise und bei unsachgemäßem Einsatz entstehen, übernimmt die VIPA keinerlei Haftung!

Hinweis zum Einsatzbereich

Das Green Cable darf ausschließlich direkt an den hierfür vorgesehenen Buchsen der VIPA-Komponenten betrieben werden (Zwischenstecker sind nicht zulässig). Beispielsweise ist vor dem Stecken des Green Cable ein gestecktes MPI-Kabel zu entfernen.

Zurzeit unterstützen folgende Komponenten das Green Cable:

VIPA CPUs mit MP^{2}I-Buchse sowie die Feldbus-Master von VIPA.

Hinweis zur Verlängerung

Die Verlängerung des Green Cable mit einem weiteren Green Cable bzw. die Kombination mit weiteren MPI-Kabeln ist nicht zulässig und führt zur Beschädigung der angeschlossenen Komponenten!

Das Green Cable darf nur mit einem 1:1 Kabel (alle 9 Pin 1:1 verbunden) verlängert werden.

Standardhantierungsbausteine für CPU 21x

SEND (FC 0)

Dieser FC dient zur Datenausgabe von der CPU an den CP 240. Hierbei legen Sie über die Bezeichner _DB, ADB und ANZ den Sendebereich fest.

Über das Bit FRG wird der Sendeanstoß gesetzt und die Daten werden gesendet. Nach dem Übertragen der Daten setzt der Hantierungsbaustein das Bit FRG wieder zurück.

Declaration	Name	Type	Comment
in	ADR	INT	Logical Address
in	_DB	BLOCK_DB	DB No. of DB containing data to send
in	ABD	WORD	No. of 1st data word to send
in	ANZ	WORD	No of bytes to send
in_out	FRG	BOOL	Start bit of the function
in_out	GESE	WORD	internal use
in_out	ANZ_INT	WORD	internal use
in_out	ENDE_KOMM	BOOL	internal use
in_out	LETZTER_BLOCK	BOOL	internal use
in_out	SENDEN_LAEUFT	BOOL	Status of function
in_out	FEHLER_KOM	BOOL	internal use
out	PAFE	BYTE	Return Code (00=OK)

ADR Peripherieadresse unter der der CP 240 anzusprechen ist. Über die Hardware-Konfiguration bestimmen Sie die Peripherieadresse.

_DB Nummer des Datenbausteins, der die zu sendenden Daten beinhaltet.

ABD Wortvariable, die die Nummer des Datenworts enthält, ab dem die auszugebenden Zeichen abgelegt sind.

ANZ Anzahl der Bytes, die zu übertragen sind.

FRG
Sendefreigabe Bei FRG = "1" werden die über _DB, ADB und ANZ definieren Daten einmalig an den über ADR adressierten CP übertragen. Nach der Übertragung wird FRG wieder zurückgesetzt. Ist beim Aufruf FRG = "0", wird der Baustein sofort wieder verlassen!

PAFE Alle Bits dieses Merker-Bytes sind bei richtiger Funktion "0". Bei Fehlfunktion wird ein Fehlercode eingetragen. Die Fehlerangabe ist selbstquittierend, d.h. nach Beseitigung der Fehlerursache wird das Byte wieder "0" gesetzt. Folgende Fehler sind möglich:

1 = Datenbaustein nicht vorhanden

2 = Datenbaustein zu kurz

3 = Datenbausteinnummer nicht im gültigen Bereich

GESE, ANZ_INT
ENDE_KOM
LETZTER_BLOCK
SENDEN_LAEUFT
FEHLER_KOM Diese Parameter werden intern verwendet. Sie dienen dem Informationsaustausch zwischen den Hantierungsbausteinen. Für den Einsatz des SYNCHRON_RESET (FC9) sind die Steuerbits ENDE_KOM, LETZTER _BLOCK, SENDEN_LAEUFT und FEHLER_KOM immer in einem Merker-Byte abzulegen.

RECEIVE (FC 1)

Dieser FC dient zum Datenempfang vom CP 240. Hierbei legen Sie über die Bezeichner _DB und ADB den Empfangsbereich fest.

Ist der Ausgang EMFR gesetzt, so ist ein neues Telegramm komplett eingelesen worden. Die Länge des eingelesenen Telegramms wird in ANZ abgelegt. Nach der Auswertung des Telegramms ist dieses Bit vom Anwender zurückzusetzen, da ansonsten kein weiteres Telegramm in der CPU übernommen werden kann.

Declaration	Name	Type	Comment
in	ADR	INT	Logical Address
in	_DB	BLOCK_DB	DB No. of DB containing received data
in	ABD	WORD	No. of 1st data word received
out	ANZ	WORD	No of bytes received
out	EMFR	BOOL	1=data received, reset by user
in_out	GEEM	WORD	internal use
in_out	ANZ_INT	WORD	internal use
in_out	EMPF_LAEUFT	BOOL	Status of function
in_out	LETZTER_BLOCK	BOOL	internal use
in_out	FEHLER_EMPF	BOOL	internal use
out	PAFE	BYTE	Return Code (00=OK)

ADR Peripherieadresse unter der der CP 240 anzusprechen ist. Über die Hardware-Konfiguration bestimmen Sie die Peripherieadresse.

_DB Nummer des Datenbaustein, der die empfangenen Daten beinhaltet.

ABD Wortvariable, die die Nummer des Datenworts enthält, ab dem die empfangenen Zeichen abgelegt sind.

ANZ Wort-Variable, die die Anzahl der Bytes enthält, die empfangen wurden.

EMFR Durch Setzen des EMFR zeigt der Hantierungsbaustein an, dass Daten empfangen wurden. Erst durch Rücksetzen von EMFR im Anwenderprogramm können weitere Daten empfangen werden.

PAFE Alle Bits dieses Merker-Bytes sind bei richtiger Funktion "0". Bei Fehlfunktion wird ein Fehlercode eingetragen. Die Fehlerangabe ist selbstquittierend, d.h. nach Beseitigung der Fehlerursache wird das Byte wieder "0" gesetzt. Folgende Fehler sind möglich:

1 = Datenbaustein nicht vorhanden

2 = Datenbaustein zu kurz

3 = Datenbausteinnummer nicht im gültigen Bereich

GEEM, ANZ_INT
LETZTER_BLOCK
EMPF_LAEUFT
FEHLER_EMPF

Diese Parameter werden intern verwendet. Sie dienen dem Informationsaustausch zwischen den Hantierungsbausteinen. Für den Einsatz des SYNCHRON_RESET (FC9) sind die Steuerbits LETZTER_BLOCK, EMPF_LAEUFT und FEHLER_EMPF immer in einem Merker-Byte abzulegen.

STEUERBIT (FC 8) Mit diesem Baustein haben Sie folgenden Zugriff auf die seriellen Modemleitungen:

Lesen: DTR, RTS, DSR, RI, CTS, CD

Schreiben: DTR, RTS

Declaration	Name	Type	Comment
in	ADR	INT	Logical Address
in	RTS	BOOL	New state RTS
in	DTR	BOOL	New state DTR
in	MASKE_RTS	BOOL	0: do nothing 1: set state RTS
in	MASKE_DTR	BOOL	0: do nothing 1: set state DTR
out	STATUS	BYTE	Status flags
out	DELTA_STATUS	BYTE	Status flags of change between 2 accesses
in_out	START	BOOL	Start bit of the function
in_out	AUFTRAG_LAEU	BOOL	Status of function
out	RET_VAL	WORD	Return Code (00=OK)

Hinweis!

Dieser Baustein darf nicht aufgerufen werden, solange ein Sendeauftrag läuft, ansonsten kann dies zu Datenverlust führen.

ADR Peripherieadresse unter welcher der CP 240 anzusprechen ist. Über die Hardware-Konfiguration bestimmen Sie die Peripherieadresse.

RTS, DTR Mit diesem Parameter geben Sie den Status für RTS bzw. DTR vor, den Sie über MASK_RTS bzw. MASK_DTR aktivieren können.

MASK_RTS, MASK_DTR Hier wird mit 1 der Status des entsprechenden Parameters übernommen, sobald Sie START auf 1 setzen.

STATUS, DELTA_STATUS STATUS liefert den aktuellen Status der Modem-Leitungen zurück. DELTA_STATUS liefert den Status der Modem-Leitungen zurück, die sich seit dem letzten Zugriff geändert haben.

Die Bytes haben folgenden Aufbau:

Bit-Nr.	7	6	5	4	3	2	1	0
STATUS	x	x	RTS	DTR	CD	RI	DSR	CTS
DELTA_STATUS	x	x	x	x	CD	RI	DSR	CTS

START Durch Setzen von START wird der über die Maske aktivierte Status übernommen.

AUFTRAG_LAEU Solange die Funktion abgearbeitet wird, bleibt dieses Bit gesetzt.

RET_VAL Dieser Parameter liefert zur Zeit immer 00h zurück und dient zukünftigen Fehlermeldungen.

SYNCHRON_ RESET
Synchronisation und Rücksetzen (FC 9)

Der Baustein ist im zyklischen Programmteil aufzurufen. Mit dieser Funktion wird die Anlaufkennung des CP 240 quittiert, und so die Synchronisation zwischen CPU und CP hergestellt. Weiterhin kann bei einer Kommunikationsunterbrechung der CP rückgesetzt werden und so ein synchroner Anlauf erfolgen.

Hinweis!
Eine Kommunikation mit SEND- und RECEIVE-Bausteinen ist nur möglich, wenn zuvor im Anlauf-OB der Parameter ANL des SYNCHRON-Bausteins gesetzt wurde.

Declaration	Name	Type	Comment
in	ADR	INT	Logical Address
in	TIMER_NR	WORD	No of timer for idle time
in_out	ANL	BOOL	restart progressed
in_out	NULL	BOOL	internal use
in_out	RESET	BOOL	1 = Reset the CP
in_out	STEUERB_S	BYTE	internal use
in_out	STEUERB_R	BYTE	internal use

ADR Peripherieadresse unter der der CP 240 anzusprechen ist. Über die Hardware-Konfiguration bestimmen Sie die Peripherieadresse.

TIMER_NR Nummer des Timers für die Wartezeit.

ANL Mit ANL = 1 wird dem Hantierungsbaustein mitgeteilt, dass an der CPU STOP/START bzw. NETZ-AUS/NETZ-EIN erfolgt ist und nun eine Synchronisation erfolgen muss. Nach der Synchronisation wird ANL automatisch zurückgesetzt.

NULL Parameter wird intern verwendet.

RESET Mit RESET = 1 können Sie den CP aus Ihrem Anwenderprogramm zurücksetzen.

STEUERB_S Hier ist das Merkerbyte anzugeben, in dem die Steuerbits ENDE_KOM, LETZTER_BLOCK, SENDEN_LAEUFT und FEHLER_KOM für den SEND-FC abgelegt sind.

STEUERB_R Hier ist das Merkerbyte anzugeben, in dem die Steuerbits LETZTER_BLOCK, EMPF_LAEUFT und FEHLER_EMPF für den RECEIVE-FC abgelegt sind.

ASCII_FRAGMENT (FC 11)

Dieser FC dient zum fragmentierten ASCII-Datenempfang. Hiermit haben Sie die Möglichkeit große Telegramme in 12Byte-Blöcken direkt nach dem Erhalt an die CPU weiterzureichen. Hierbei wartet der CP nicht, bis das komplette Telegramm empfangen wurde. Der Einsatz des FC 11 setzt voraus, dass Sie beim Empfänger "ASCII-fragmentiert" parametriert haben.

Im FC 11 legen Sie über die Bezeichner _DB und ADB den Empfangsbereich fest. Ist der Ausgang EMFR gesetzt, so ist ein neues Telegramm komplett eingelesen worden. Die Länge des eingelesenen Telegramms wird in ANZ abgelegt. Nach der Auswertung des Telegramms ist dieses Bit vom Anwender zurückzusetzen, da ansonsten kein weiteres Telegramm in der CPU übernommen werden kann.

Declaration	Name	Type	Comment
in	ADR	INT	Logical Address
in	_DB	BLOCK_DB	DB No. of DB containing received data
in	ABD	WORD	No. of 1st data word received
out	ANZ	WORD	No of bytes received
in_out	EMFR	BOOL	1=data received, reset by user
in_out	GEEM	WORD	internal use
in_out	ANZ_INT	WORD	internal use
in_out	EMPF_LAEUFT	BOOL	internal use
in_out	LETZTER_BLOCK	BOOL	internal use
in_out	FEHLER_EMPF	BOOL	internal use
out	PAFE	BYTE	Return Code (00=OK)

ADR Peripherieadresse unter der der CP 240 anzusprechen ist. Über die Hardware-Konfiguration bestimmen Sie die Peripherieadresse.

_DB Nummer des Datenbaustein, der die empfangenen Daten beinhaltet.

ABD Wortvariable, die die Nummer des Datenworts enthält, ab dem die empfangenen Zeichen abgelegt sind.

ANZ Wort-Variable, die die Anzahl der Bytes enthält, die empfangen wurden.

EMFR Durch Setzen des EMFR zeigt der Hantierungsbaustein an, dass Daten empfangen wurden. Erst durch Rücksetzen von EMFR im Anwenderprogramm können weitere Daten empfangen werden.

PAFE Alle Bits dieses Merker-Bytes sind bei richtiger Funktion "0". Bei Fehlfunktion wird ein Fehlercode eingetragen. Die Fehlerangabe ist selbstquittierend, d.h. nach Beseitigung der Fehlerursache wird das Byte wieder "0" gesetzt. Folgende Fehler sind möglich:

1 = Datenbaustein nicht vorhanden

2 = Datenbaustein zu kurz

3 = Datenbausteinnummer nicht im gültigen Bereich

GEEM, ANZ_INT LETZTER_BLOCK EMPF_LAEUFT FEHLER_EMPF Diese Parameter werden intern verwendet. Sie dienen dem Informationsaustausch zwischen den Hantierungsbausteinen. Für den Einsatz des SYNCHRON_REST (FC9) sind die Steuerbits LETZTER_BLOCK, EMPF_LAEUFT und FEHLER_EMPF immer in einem Merker-Byte abzulegen.

Teil 4 CP 240 - RS232/RS485

Überblick

In diesem Kapitel finden Sie Informationen über den Aufbau und die Anschlussbelegung des Kommunikationsprozessors CP 240 mit RS232- bzw. RS485-Schnittstelle. Den Kommunikationsprozessor CP 240 erhalten Sie von VIPA mit verschiedenen Übertragungsprotokollen, auf deren Einsatz hier näher eingegangen wird.

Nachfolgend sind beschrieben:

- Systemübersicht mit Schnelleinstieg
- Aufbau und Beschreibung der Komponenten
- Grundlagen und Einsatz von ASCII, STX/ETX, 3964(R), RK512 und Modbus
- Technische Daten

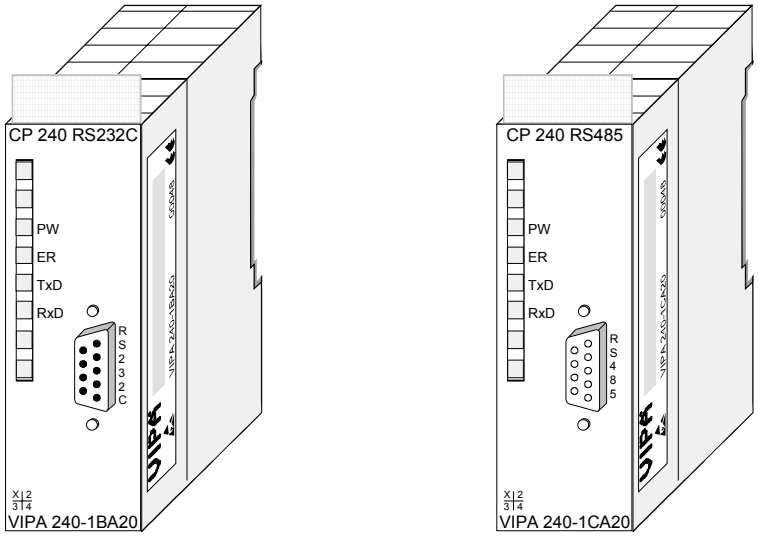
Inhalt

Systemübersicht

Eigenschaften

- RS232-Schnittstelle (nur VIPA 240-1BA20)
- RS485-Schnittstelle (nur VIPA 240-1CA20)
- Unterstützt werden die Protokolle ASCII, STX/ETX, 3964(R), RK512 und Modbus
- Parametrierung über 16Byte Parameterdaten
- Bis zu 250 Telegramme innerhalb der 1024Byte großen Empfangs- bzw. Sendepuffer
- Serielle Schnittstelle potenzialgetrennt zum Rückwandbus
- Spannungsversorgung über Rückwandbus

CP 240 RS232
CP 240 RS485



Bestelldaten

Typ	Bestellnummer	Beschreibung
CP 240	VIPA 240-1BA20	CP 240 mit RS232-Schnittstelle Protokolle: ASCII, STX/ETX, 3964(R), RK512, Modbus
CP 240	VIPA 240-1CA20	CP 240 mit RS485-Schnittstelle Protokolle: ASCII, STX/ETX, 3964(R), RK512, Modbus

Schnelleinstieg

Übersicht

Die Kommunikationsprozessoren CP 240 ermöglichen die serielle Prozessankopplung zu verschiedenen Ziel- oder Quellsystemen. Je nach Modul haben Sie eine RS232- oder eine RS485-Schnittstelle.

Die CP 240-Module werden über den Rückwandbus mit Betriebsspannung versorgt.

Zur internen Kommunikation sind VIPA FCs zu verwenden. Hier werden Daten mit einer maximalen Blockgröße von 12Byte übertragen.

Für die Projektierung des CP 240 in Verbindung mit einer CPU 21x im Siemens SIMATIC Manager, ist die Einbindung der GSD VIPA_21x.gsd erforderlich. Damit der CP 240 mit der CPU kommunizieren kann, ist für das System immer eine Hardware-Konfiguration durchzuführen.

Eine allgemeine Beschreibung zur Projektierung des CP 240 finden Sie im Teil "Projektierung".

Parameter

Zur Parametrierung können dem CP 16Byte Parameterdaten übergeben werden, die je nach gewähltem Protokoll entsprechend belegt sind.

Die Parametrierung erfolgt über die Hardware-Konfiguration im Siemens SIMATIC Manager durch Einbindung eines protokollspezifischen CP 240.

Protokolle

Nach der GSD-Einbindung sind CP 240 mit folgenden Protokollen verfügbar:

- ASCII
- STX/ETX
- 3964(R) und RK512
- Modbus (Master, Slave)

Kommunikation

Die serielle Kommunikation erfolgt unter Einsatz von Hantierungsbausteinen im SPS-Anwenderprogramm. Diese Hantierungsbausteine können Sie von ftp.vipa.de downloaden oder als Bestandteil der CD VIPA "ToolDemo" beziehen.

Je nach Protokoll kommen folgende Hantierungsbausteine zum Einsatz:

ASCII	STX 3964	RK512	Modbus	FC	Name
x	x		x	FC0	SEND_ASCII_STX_3964
x	x		x	FC1	RECEIVE_ASCII_3964
		x		FC2	FETCH_RK512
		x		FC3	SEND_RK512
		x		FC4	S/R_ALL_RK512
x	x	x		FC9	SYNCHRON_RESET
x				FC11	ASCII_FRAGMENT

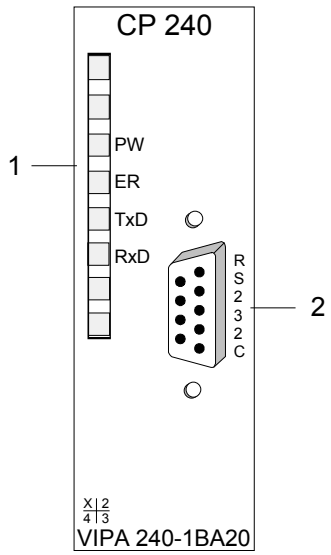
Hinweis!

Eine Kommunikation mit SEND- und RECEIVE-Bausteinen ist nur möglich, wenn zuvor im Anlauf-OB der Paramater ANL des SYNCHRON-Bausteins gesetzt wurde.

Aufbau

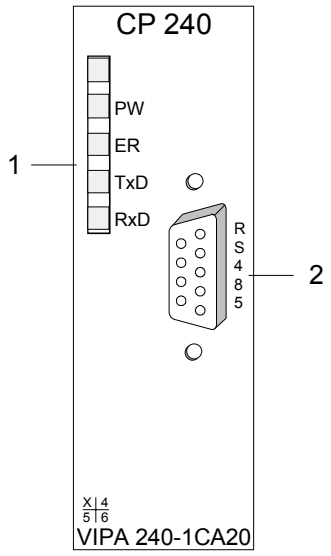
Frontansicht

CP 240 RS232
240-1BA20



[1] LED Statusanzeigen
[2] 9poliger serieller SubD-Stecker für RS232-Kommunikation

CP 240 RS485
240-1CA20



[1] LED Statusanzeigen
[2] 9polige serielle SubD-Buchse für RS485-Kommunikation

Komponenten

Spannungs-versorgung

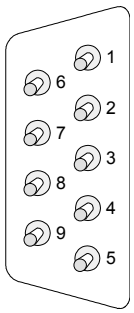
Der Kommunikationsprozessor bezieht seine Versorgungsspannung über den Rückwandbus.

LEDs

Der Kommunikationsprozessor besitzt 4 LEDs, die der Betriebszustand-Anzeige dienen. Die Bedeutung und die jeweiligen Farben dieser LEDs finden Sie in der nachfolgenden Tabelle.

Bez.	Farbe	Bedeutung
PW	Grün	Signalisiert eine anliegende Betriebsspannung
ER	Rot	Bei Modbus: Signalisiert internen Fehler. ansonsten: Signalisiert einen Fehler durch: Leitungsunterbrechung, Überlauf, Paritätsfehler oder Zeichenrahmenfehler. Automatisches Rücksetzen der Fehler-LED nach 4s. Bei aktivierter Diagnose erfolgt dann die Fehleranzeige über die Diagnosebytes.
TxD	Grün	Daten senden (transmit data)
RxD	Grün	Daten empfangen (receive data)

RS232-Schnittstelle

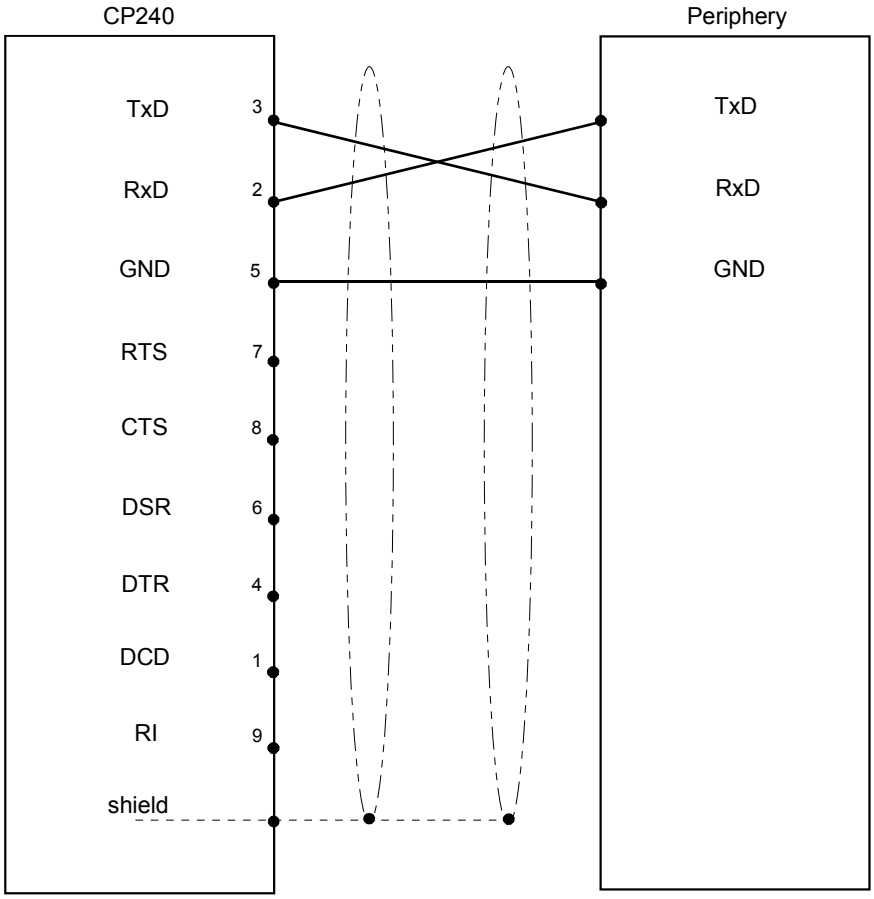


9poliger SubD-Stecker

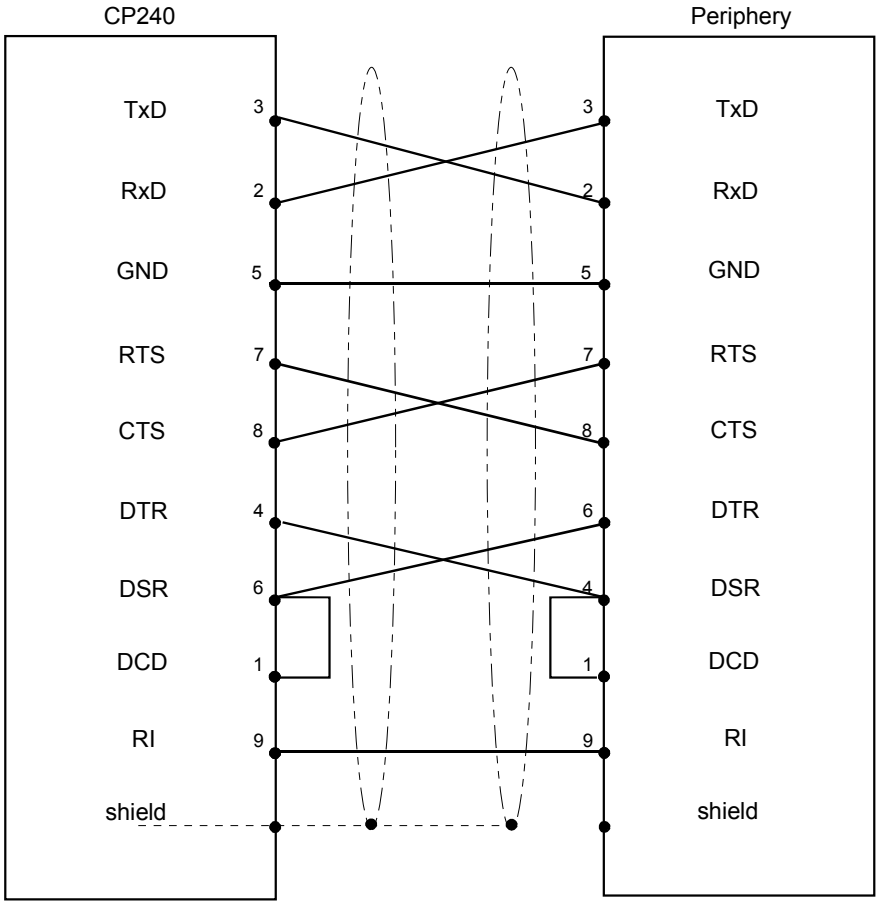
Pin	Bezeichnung		Signalbeschreibung
1	DCD	Data Carrier Detect	Daten können empfangen werden
2	RxD	Receive Data	Empfangsdaten von Modem an CP 240
3	TxD	Transmit Data	Sendedaten von CP 240 an Modem
4	DTR	Data Terminal Ready	CP 240 ist betriebsbereit
5	GND	Signal Ground	GND Nullbezugspunkt
6	DSR	Data Set Ready	Modem signalisiert, dass es betriebsbereit ist
7	RTS	Request to send	CP 240 zeigt an, dass er betriebsbereit ist
8	CTS	Clear to send	Modem zeigt an, dass CP 240 senden darf
9	RI	Ring indicator	Klingelzeichen

- Logische Zustände als Spannungspegel
- Punkt-zu-Punkt-Kopplung mit serieller Vollduplex-Übertragung
- Datenübertragung bis 15m Entfernung
- Datenübertragungsrate bis 115kBaud

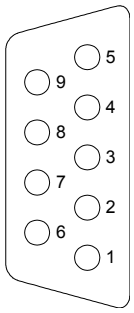
RS232-Verkabelung ohne Hardware-Handshake



RS232-Verkabelung mit Hardware-Handshake



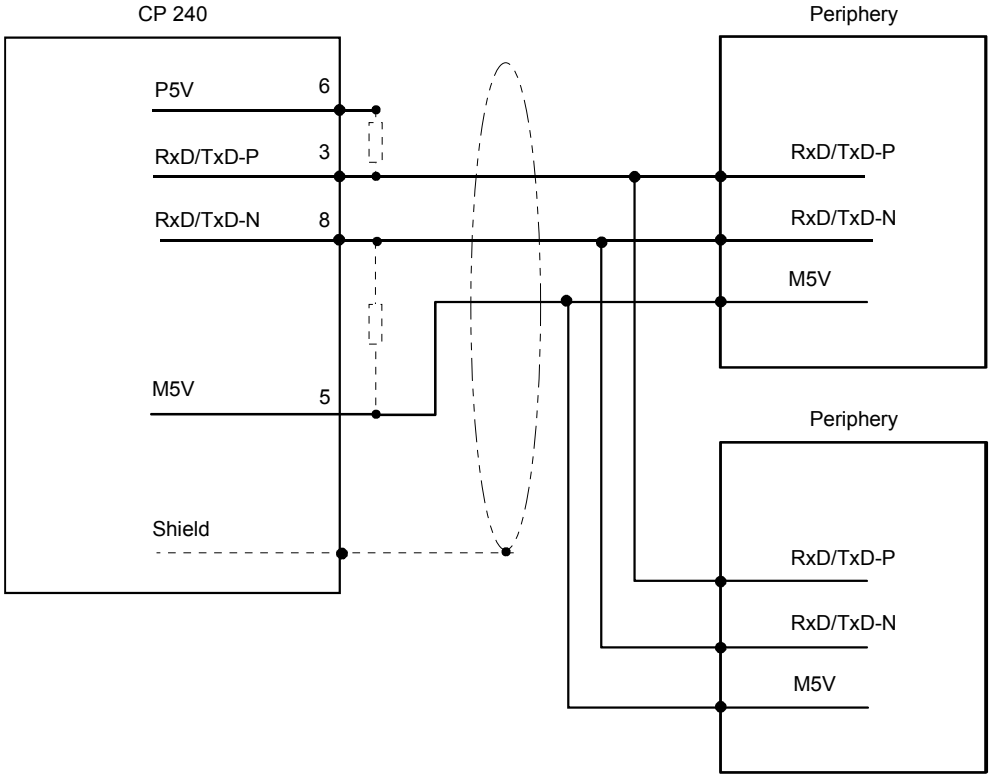
RS485-Schnittstelle



Pin	RS485
1	n.c.
2	n.c.
3	RxT/TxD-P (Leitung B)
4	RTS
5	M5V
6	P5V
7	n.c.
8	RxT/TxD-N (Leitung A)
9	n.c.

- Logische Zustände als Spannungsdifferenz zwischen zwei verdrillten Adern
- Serielle Busverbindung in 2-Draht-Technik im Halbduplex-Verfahren
- Multidrop-Verbindung
- Hohe Störfestigkeit
- Anschaltung von bis zu 32 Teilnehmern
- Datenübertragung bis 500m
- Datenübertragungsrate bis 115kBaud

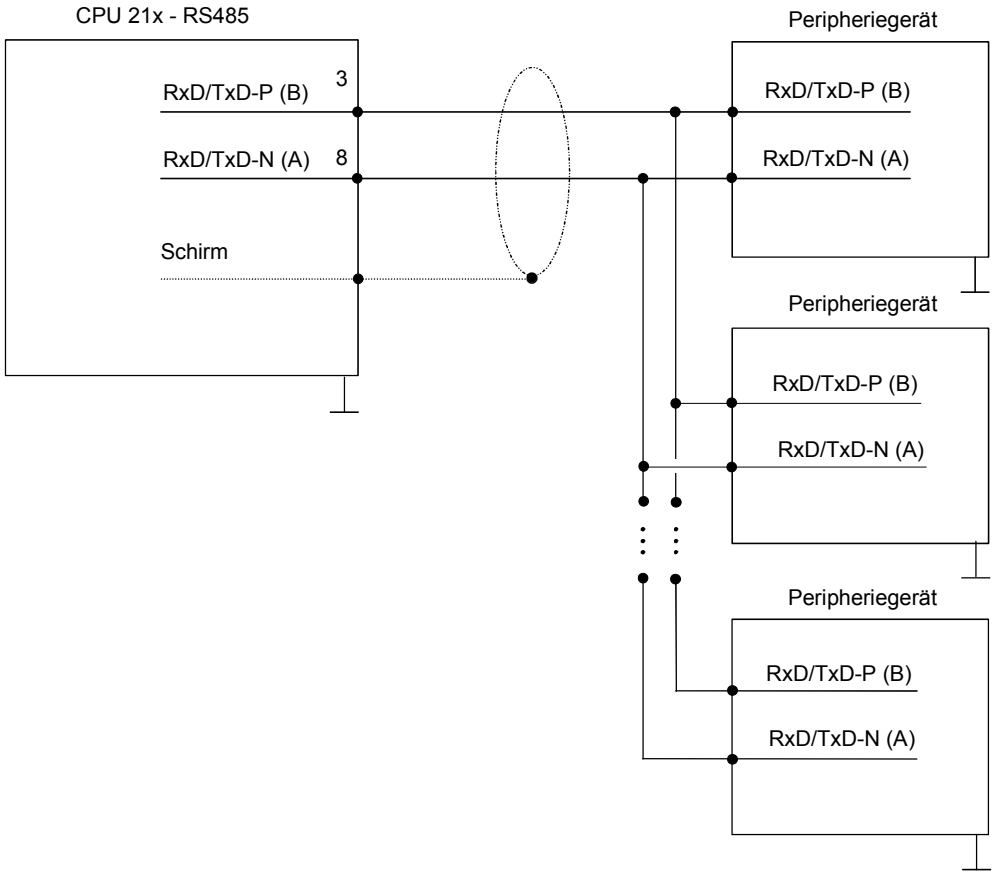
RS485-Verkabelung



Definierte Ruhepegel über Widerstände

Bei potenzialgetrennten Schnittstellen haben Sie auf Pin 6 isolierte 5V (P5V) und an Pin 5 die zugehörige Masse (M5V). Mit dieser isolierten Spannung können Sie über Widerstände zu den Signalleitungen definierte Ruhepegel vergeben und für einen reflexionsarmen Abschluss sorgen.

Verkabelung mittels Profibus-Kabel



ASCII / STX/ETX / 3964(R) / RK512 - Grundlagen

ASCII

Die Datenkommunikation via ASCII ist eine einfache Form des Datenaustauschs und kann mit einer Multicast/Broadcast-Funktion verglichen werden.

Die logische Trennung der Telegramme wird über 2 Zeitfenster gesteuert. Der Sender muss sein Telegramm innerhalb der "Zeichenverzugszeit" (ZVZ), die im Empfänger vorgegeben wird, schicken.

Solange "Zeit nach Auftrag" (ZNA) nicht abgelaufen ist, wird kein neuer Sendeauftrag angenommen.

Mit diesen beiden Zeitangaben kann eine einfach serielle SPS-SPS-Kommunikation aufgebaut werden.

Das Bit FRG (Sende-Freigabe) wird erst dann zurückgesetzt, wenn die Daten gesendet wurden und die ZNA abgelaufen ist.

ASCII-fragmentiert

Unter ASCII wird ein Telegramm erst dann an die CPU weitergereicht, wenn dieses vollständig empfangen wurde. Mit ASCII-fragmentiert haben Sie die Möglichkeit durch Einsatz des Receive-Bausteins FC11 (ASCII_FRAGMENT) große Telegramme in Blöcken direkt nach dem Erhalt an die CPU weiterzureichen. Hierbei beträgt die Blocklänge 12Byte. Bei ASCII-fragmentiert wartet der CP nicht bis das komplette Telegramm empfangen wurde.

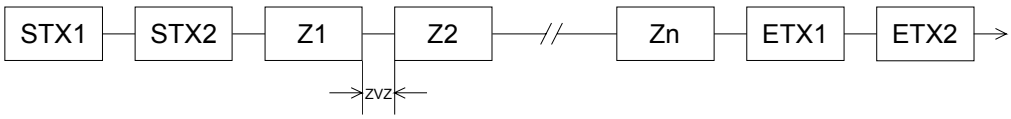
STX/ETX

STX/ETX ist ein einfaches Protokoll mit Header und Trailer. STX/ETX wird zur Übertragung von ASCII-Zeichen (20h…7Fh) eingesetzt. Dies erfolgt ohne Blockprüfung (BCC). Sollen Daten von der Peripherie eingelesen werden, muss als Startzeichen STX (Start of Text) vorhanden sein, anschließend folgen die zu übertragenden Zeichen. Als Schlusszeichen muss ETX (End of Text) vorliegen.

Die Nutzdaten, d.h. alle Zeichen zwischen STX und ETX, werden nach Empfang des Schlusszeichens ETX an die CPU übergeben.

Beim Senden der Daten von der CPU an ein Peripheriegerät werden die Nutzdaten an den CP 240 übergeben und von dort, mit STX als Startzeichen und ETX als Schlusszeichen, an den Kommunikationspartner übertragen.

Telegrammaufbau:



Sie können bis zu 2 Anfangs- und Endezeichen frei definieren. Auch hier kann eine ZNA für den Sender vorgegeben werden.

3964(R)

3964(R) steuert die Datenübertragung bei einer Punkt-zu-Punkt-Kopplung zwischen dem CP 240 und einem Kommunikationspartner. Hier werden bei der Datenübertragung den Nutzdaten Steuerzeichen hinzugefügt. Durch diese Steuerzeichen kann der Kommunikationspartner kontrollieren, ob die Daten vollständig und fehlerfrei bei ihm angekommen sind.

Folgende Steuerzeichen werden ausgewertet:

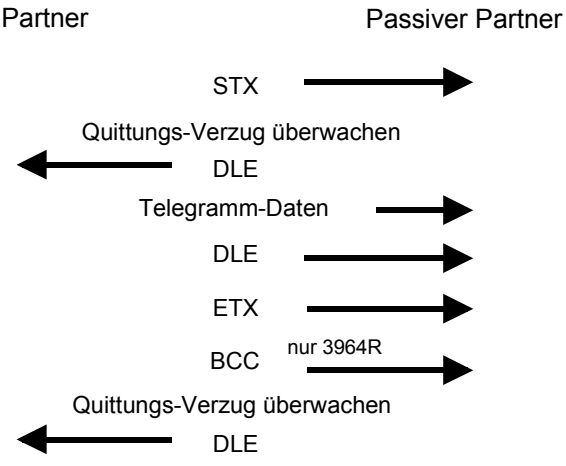
- STX Start of Text
- DLE Data Link Escape
- ETX End of Text
- BCC Block Check Character (nur bei 3964R)
- NAK Negative Acknowledge

Hinweis!

Wird ein DLE als Informationszeichen übertragen, so wird dieses zur Unterscheidung vom Steuerzeichen DLE beim Verbindungsauf- und -abbau auf der Sendeleitung doppelt gesendet (DLE-Verdoppelung). Der Empfänger macht die DLE-Verdoppelung wieder rückgängig.

Unter 3964(R) muss dem Kommunikationspartner eine niedrigere Priorität zugeordnet sein. Wenn beide Kommunikationspartner gleichzeitig einen Sendeauftrag erteilen, dann stellt der Partner mit niedriger Priorität seinen Sendeauftrag zurück.

Ablauf

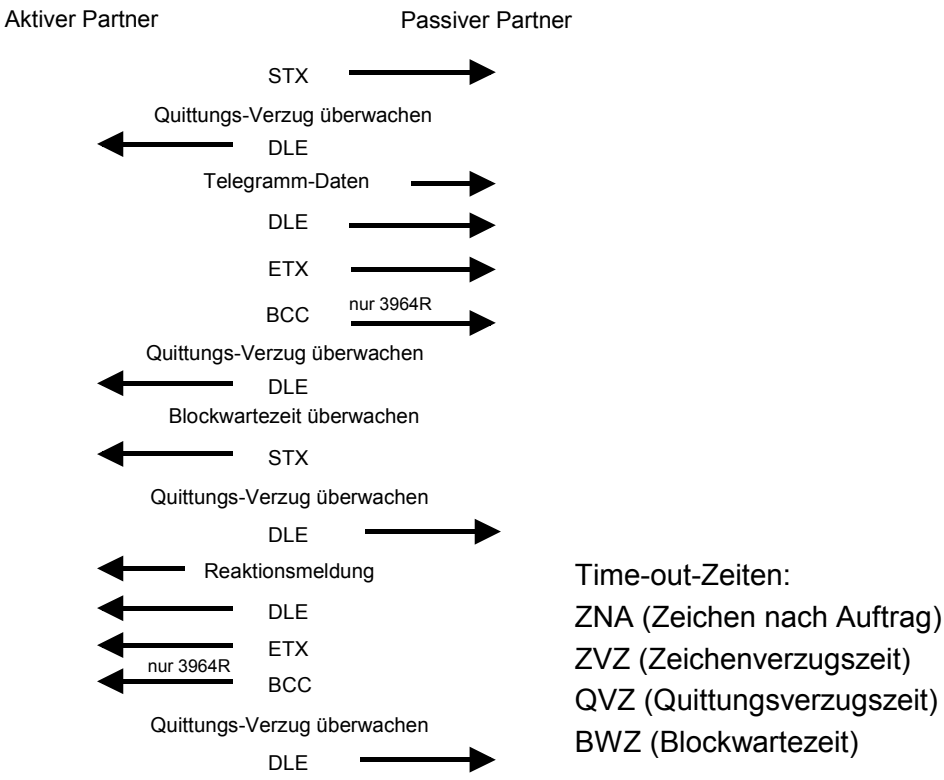


Sie können pro Telegramm maximal 250Byte übertragen.

3964(R) mit RK512

Das RK512 ist ein erweitertes 3964(R). Es wird lediglich vor der Übertragung der Nutzdaten ein Telegrammkopf gesendet. Der Telegrammkopf enthält für den Kommunikationspartner Informationen über Größe, Art und Länge der Nutzdaten.

Ablauf



Time-out-Zeiten

QVZ wird überwacht zwischen STX und DLE sowie zwischen BCC und DLE. ZVZ wird während des gesamten Telegramm-Empfangs überwacht.

Bei Verstreichen der QVZ nach STX wird erneut STX gesendet, nach 5 Versuchen wird ein NAK gesendet und der Verbindungsaufbau abgebrochen. Dasselbe geschieht, wenn nach einem STX ein NAK oder ein beliebiges Zeichen empfangen wird.

Bei Verstreichen der QVZ nach dem Telegramm (nach BCC-Byte) oder bei Empfang eines Zeichens ungleich DLE werden der Verbindungsaufbau und das Telegramm wiederholt. Auch hier werden 5 Versuche unternommen, danach ein NAK gesendet und die Übertragung abgebrochen.

Die Blockwartezeit (BWZ) ist die maximale Zeitdauer zwischen der Bestätigung eines Anforderungstelegrams (DLE) und STX des Reaktionstelegramms. Bei Überschreiten der BWZ wird mehrere Male (über DBL parametrierbar) versucht das Anforderungstelegramm zu senden. Sind diese Versuche erfolglos, wird die Übertragung abgebrochen.

Passivbetrieb

Wenn der Treiber auf den Verbindungsaufbau wartet und ein Zeichen ungleich STX empfängt, sendet er NAK. Bei Empfang eines Zeichens NAK sendet der Treiber keine Antwort.

Wird beim Empfang die ZVZ überschritten, wird ein NAK gesendet und auf erneuten Verbindungsaufbau gewartet.

Wenn der Treiber beim Empfang des STX noch nicht bereit ist, sendet er ein NAK.

Block-Check-Character (BCC-Byte)

Zur weiteren Datensicherung wird bei 3964R am Ende des Telegramms ein **B**lock-**C**heck-**C**harakter angehängt. Das BCC-Byte wird durch eine XOR-Verknüpfung über die Daten des gesamten Telegramms einschließlich DLE/ETX gebildet.

Beim Empfang eines BCC-Bytes, das vom selbst Ermittelten abweicht, wird anstatt des DLEs ein NAK gesendet.

Initialisierungs-konflikt

Versuchen beide Partner gleichzeitig innerhalb der QVZ einen Verbindungsaufbau, so sendet der Partner mit der niedrigeren Priorität das DLE und geht auf Empfang.

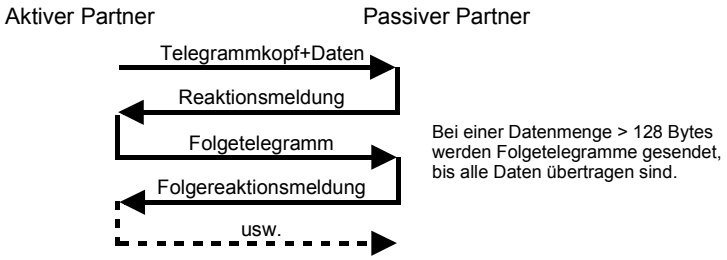
Data Link Escape (DLE-Zeichen)

Das DLE-Zeichen in einem Telegramm wird vom Treiber verdoppelt, d.h. es wird DLE/DLE gesendet. Beim Empfang werden doppelte DLEs als ein DLE im Puffer abgelegt. Als Ende des Telegramms gilt immer die Kombination DLE/ETX/BCC (nur bei 3964R).

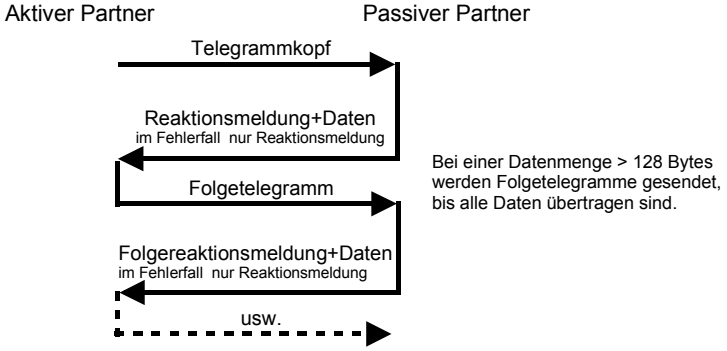
Die Steuercodes : 02h = STX
03h = ETX
10h = DLE
15h = NAK

Logischer Telegrammablauf

SEND (Senden von Daten)



FETCH (Holen von Daten)



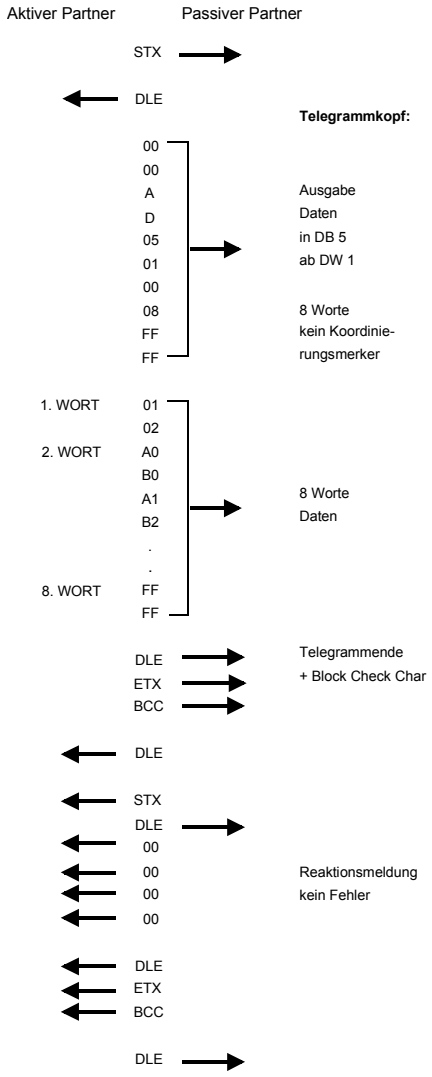
In beiden Fällen wird maximal 5s auf die Reaktionsmeldung gewartet, und danach der Empfang abgebrochen.

Inhalt der Telegramme

Jedes Telegramm besitzt einen Kopf. Je nach der Vorgeschichte des Telegrammverkehrs enthält dieser alle erforderlichen Informationen.

Aufbau Ausgabe-Telegramm

Beispiel Ausgabetelegramm



Normales Telegramm

Byte		
0	00	Kennung für
1	00	Telegramm
2	A	Ausgabebefehl
3	X	Art der Daten
4	xx	Parameter 1
5	xx	Ziel
6	yy	Parameter 2
7	yy	Anzahl
8	zz	Parameter 3
9	zz	Koordinierungsmerker
10	aa	
-	bb	Daten
N	xy	

mit N = 10 ... 127

Reaktionstelegramm

Byte		
0	00	Kennung für
1	00	Reaktions-
2	00	meldung
3	xx	Fehlercode

Bei Datenmengen >128Byte werden Folgetelegramme gesendet.

Aufbau Folgetelegramme

Folgetelegramm

Byte		
0	FF	Kennung für
1	00	Folgetelegramm
2	A	Ausgabebefehl
3	X	Art der Daten
4	aa	
-	bb	Daten
N	xy	

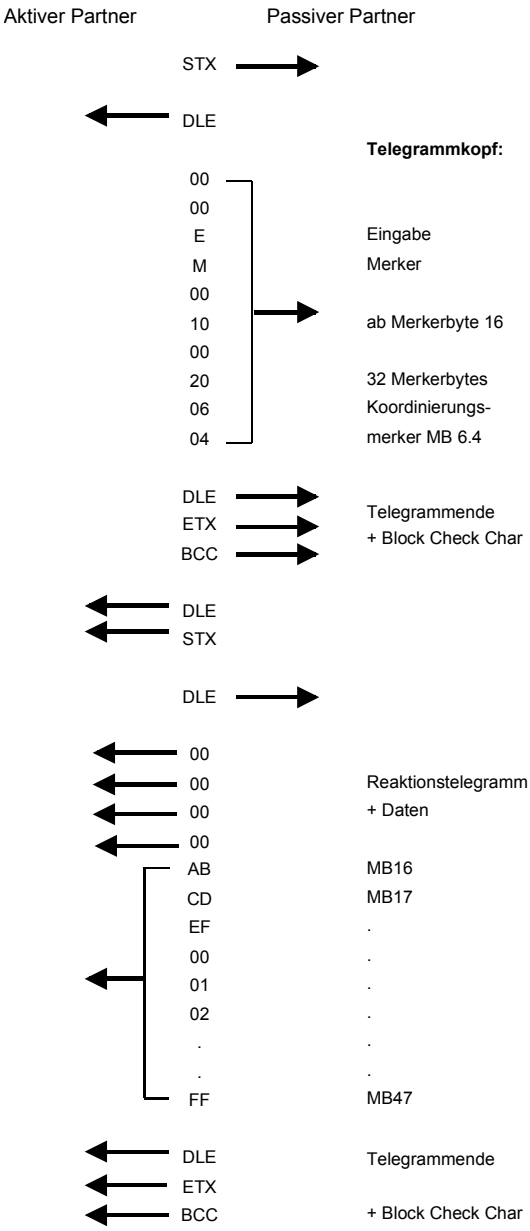
mit N = 4 ... 127

Folge-Reaktionstelegramm

Byte		
0	FF	Kennung für
1	00	Folge-Reaktions-
2	00	telegramm
3	xx	Fehlercode

Aufbau Eingabe-Telegramm

Beispiel Eingabetelegramm



Normales Telegramm

Byte		
0	00	Kennung für
1	00	Telegramm
2	E	Eingabebefehl
3	X	Art der Daten
4	xx	Parameter 1
5	xx	Ziel
6	yy	Parameter 2
7	yy	Anzahl
8	zz	Parameter 3
9	zz	Koordinierungsmerker

Reaktionstelegramm

Byte		
0	00	Kennung für
1	00	Reaktions-
2	00	meldung
3	xx	Fehlercode
4	aa	
-	bb	Daten
N	xy	

mit N = 4 ... 127

Bei Datenmengen >128Byte werden Folgetelegramme gesendet.

Aufbau Folgetelegramme

Folgetelegramm

Byte		
0	FF	Kennung für
1	00	Folgetelegramm
2	E	Eingabebefehl
3	X	Art der Daten

Folge-Reaktionstelegramm

Byte		
0	FF	Kennung für
1	00	Folge-Reaktions-
2	00	telegramm
3	xx	Fehlercode
4	aa	
-	bb	Daten
N	xy	

mit N = 4 ... 127

Koordinierungs-merker

Der Koordinierungsmerker wird im Aktiv-Betrieb im Partner-AG bei Empfang eines Telegramms gesetzt. Dies geschieht sowohl bei Eingabe- als auch bei Ausgabe-Befehlen. Ist der Koordinierungsmerker gesetzt und wird ein Telegramm mit diesem Merker empfangen, so werden die Daten nicht übernommen (bzw. übergeben), sondern es wird eine Fehler-Reaktionsmeldung gesendet (Fehlercode 32h). In diesem Fall muss der Koordinierungsmerker vom Anwender im Partner-AG zurückgesetzt werden.

ASCII / STX/ETX / 3964(R) / RK512 - Kommunikationsprinzip

Kommunikation über Hantierungs-bausteine

Die serielle Kommunikation erfolgt unter Einsatz von Hantierungs-bausteinen. Diese Hantierungsbausteine können Sie von ftp.vipa.de downloaden oder als Bestandteil der CD VIPA "ToolDemo" beziehen.

Je nach Protokoll kommen folgende Hantierungsbausteine zum Einsatz:

ASCII	STX 3964	RK512	Modbus	FC	Name
x	x		x	FC0	SEND_ASCII_STX_3964
x	x		x	FC1	RECEIVE_ASCII_3964
		x		FC2	FETCH_RK512
		x		FC3	SEND_RK512
		x		FC4	S/R_ALL_RK512
x	x	x		FC9	SYNCHRON_RESET
x				FC11	ASCII_FRAGMENT

Hinweis!

Eine Kommunikation mit SEND- und RECEIVE-Bausteinen ist nur möglich, wenn zuvor im Anlauf-OB der Paramater ANL des SYNCHRON-Bausteins gesetzt wurde.

Daten senden und empfangen

Daten, die von der CPU über den Rückwandbus in den entsprechenden Datenkanal geschrieben werden, werden vom Kommunikationsprozessor in den entsprechenden Sendepuffer (1024Byte) geschrieben und von dort über die Schnittstelle ausgegeben.

Empfängt der Kommunikationsprozessor Daten über die Schnittstelle, werden die Daten in einem Ringpuffer abgelegt (1024Byte). Die empfangenen Daten können über den Datenkanal von der CPU gelesen werden.

Kommunikation über Rück-wandbus

Der Austausch von empfangenen Telegrammen über den Rückwandbus erfolgt asynchron. Ist ein komplettes Telegramm über die serielle Schnittstelle eingetroffen (Ablauf der ZVZ), so wird dies in einem 1024Byte großen Ringpuffer abgelegt. Aus der Länge des noch freien Ringpuffers ergibt sich die max. Länge eines Telegramms. Je nach Parametrierung können bis zu 250 Telegramme gepuffert werden, wobei deren Gesamtlänge 1024 nicht überschreiten darf.

Ist der Puffer voll, werden neu ankommende Telegramme verworfen. Ein komplettes Telegramm wird in je 12Byte große Blöcke unterteilt und an den Rückwandbus übergeben. Das Zusammensetzen der Datenblöcke hat in der CPU zu erfolgen.

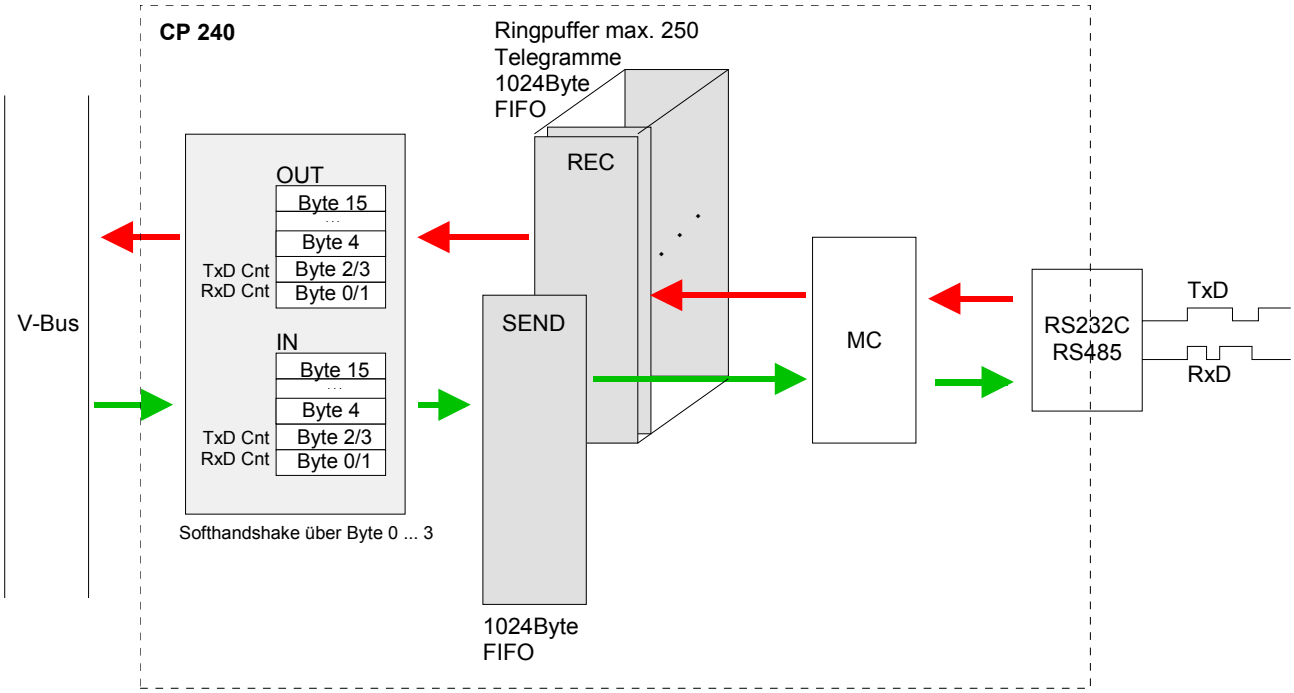
Kommunikation unter ASCII-fragmentiert

Unter ASCII-fragmentiert werden ankommende Daten eines Telegramms sofort in Blöcken an die CPU weitergereicht. Hierbei beträgt die Blocklänge mindestens 12Byte. Bei ASCII-fragmentiert wartet der CP nicht bis das komplette Telegramm empfangen wurde.

Aufgaben der CPU

Ein zu sendendes Telegramm ist in der CPU in 12Byte große Blöcke zu unterteilen und über den Rückwandbus an den CP 240 zu übergeben. Im CP 240 werden diese Blöcke im Sendepuffer zusammengesetzt und bei Vollständigkeit des Telegramms über die serielle Schnittstelle gesendet. Da der Datenaustausch über den Rückwandbus asynchron abläuft, wird ein "Software Handshake" zwischen dem CP 240 und der CPU eingesetzt. Die Register für den Datentransfer vom CP 240 sind 16Byte breit. Für den Handshake sind die Bytes 0 bis 3 (Wort 0 und 2) reserviert.

Folgende Abbildung soll dies veranschaulichen:



Software-handshake

Für den Einsatz des CP 240 in Verbindung mit einer System 200V CPU sind bei VIPA Hantierungsbausteine erhältlich, die den Softwarehandshake komfortabel übernehmen.

Bei Einsatz des CP 240 ohne Hantierungsbausteine soll hier die Funktionsweise anhand eines Beispiels für das Senden und Empfangen von Daten erläutert werden.

Beispiel Daten senden

Es soll z.B. ein Telegramm mit der Länge von 30Byte gesendet werden. So werden von der CPU die ersten 12Byte Nutzdaten des Telegramms in die Bytes 4 bis 15 und in Byte 2/3 die Länge des Telegramms (also "30") geschrieben. Der CP 240 empfängt die Daten über den Rückwandbus und kopiert die 12Byte Nutzdaten in den Sendepuffer. Zur Quittierung des Empfangs schreibt der CP 240 in Byte 2/3 den Wert "30" (Länge des Telegramms) zurück.

Beim Empfang der "30", kann die CPU weitere 12Byte Nutzdaten in Byte 4 bis 15 und die Restlänge des Telegramms ("18" Byte) in Byte 2/3 an den CP 240 senden. Dieser speichert wieder die Nutzdaten im Sendepuffer und gibt die Restlänge des Telegramms ("18") auf Byte 2/3 an die CPU zurück.

Beim Empfang der "18", kann die CPU die restlichen 6Byte Nutzdaten in den Byte 4 bis 9 und die Restlänge des Telegramms (also "6") in Byte 2/3 an den CP 240 senden. Dieser speichert die Nutzdaten im Sendepuffer ab und schreibt den Wert "6" auf Byte 2/3 an die CPU zurück.

Beim Empfang der "6" auf Byte 2/3 sendet die CPU eine "0" auf Byte 2/3. Der CP 240 stößt daraufhin das Senden des Telegramms über die serielle Schnittstelle an und schreibt, wenn alle Daten übertragen sind, eine "0" auf Byte 2/3 zurück.

Beim Empfang der "0" kann die CPU ein neues Telegramm an den CP 240 senden.

Beispiel Daten empfangen

Die Schnittstelle des CP 240 hat z.B. ein Telegramm mit 18Byte Länge über die serielle Schnittstelle empfangen. Der CP 240 schreibt die ersten 12Byte Nutzdaten in die Bytes 4 bis 15 des Empfangspuffers und in Byte 0/1 die Länge des Telegramms (also "18"). Die Daten werden über den Rückwandbus an die CPU übertragen. Die CPU speichert die 12Byte Nutzdaten und sendet den Wert "18" auf Byte 0/1 an den CP 240 zurück.

Beim Empfang der "18", schreibt der CP 240 die restlichen 6Byte Nutzdaten in die Byte 4 bis 9 des Empfangspuffers und in Byte 0/1 die Länge ("6") der übergebenen Nutzdaten. Die CPU speichert die Nutzdaten und gibt an den CP 240 in Byte 0/1 den Wert "6" zurück.

Beim Empfang der "6" sendet der CP 240 den Wert "0" auf Byte 0/1, für Telegramm komplett, an die CPU zurück. Die CPU sendet eine "0" auf Byte 0/1 an den CP 240 zurück.

Mit dem Empfang der "0" kann der CP 240 ein neues Telegramm an die CPU senden.

ASCII / STX/ETX / 3964(R) / RK512 - Parametrierung

Allgemein

Sie können dem CP 240 zur Parametrierung 16Byte Parameterdaten übergeben. Der Aufbau der Parameterdaten richtet sich nach dem gewählten Protokoll.

Bei der Hardware-Konfiguration ist immer der dem Protokoll entsprechende CP 240 zu verwenden.

Nachfolgend finden Sie eine Auflistung der Parameterbytes mit ihren Default-Werten.

Aufbau Parameterbytes bei ASCII

Byte	Funktion	Wertebereich	Defaultparameter
0	Baudrate	00h: Default (9600Baud) 01h: 150Baud 02h: 300Baud 03h: 600Baud 04h: 1200Baud 05h: 1800Baud 06h: 2400Baud 07h: 4800Baud 08h: 7200Baud 09h: 9600Baud 0Ah: 14400Baud 0Bh: 19200Baud 0Ch: 38400Baud 0Dh: 57600Baud 0Fh: 76800Baud 0Eh: 115200Baud	00h: 9600Baud
1	Protokoll	01h: ASCII 11h: ASCII-fragmentiert	01h: (ASCII)
2	Bit 1/0 Datenbits	00b: 5 Datenbits 01b: 6 Datenbits 10b: 7 Datenbits 11b: 8 Datenbits	11b: 8 Datenbits
	Bit 3/2 Parity	00b: none 01b: odd 10b: even 11b: even	00b: none
	Bit 5/4 Stopbits	01b: 1 10b: 1,5 11b: 2	01b: 1 Stopbit
	Bit 7/6 Flusskontrolle	00b: keine 01b: Hardware 10b: XON/XOFF	00b: keine
3	reserviert	0	0
4	ZNA (*20ms)	0..255	0
5	ZVZ (*20ms)	0..255	10
6	Anz.Receivebuffer	1..250	1
7...15	reserviert		

Aufbau Parameterbytes bei STX/ETX

Byte	Funktion	Wertebereich	Defaultparameter
0	Baudrate	00h: Default (9600Baud) 01h: 150Baud 02h: 300Baud 03h: 600Baud 04h: 1200Baud 05h: 1800Baud 06h: 2400Baud 07h: 4800Baud 08h: 7200Baud 09h: 9600Baud 0Ah: 14400Baud 0Bh: 19200Baud 0Ch: 38400Baud 0Dh: 57600Baud 0Fh: 76800Baud 0Eh: 115200Baud	00h: 9600Baud
1	Protokoll	02h: STX/ETX	02h: (STX/ETX)
2	Bit 1/0 Datenbits	00b: 5 Datenbits 01b: 6 Datenbits 10b: 7 Datenbits 11b: 8 Datenbits	11b: 8 Datenbits
	Bit 3/2 Parity	00b: none 01b: odd 10b: even 11b: even	00b: none
	Bit 5/4 Stopbits	01b: 1 10b: 1,5 11b: 2	01b: 1 Stopbit
	Bit 7/6 Flusskontrolle	00b: keine 01b: Hardware 10b: XON/XOFF	00b: keine
3	reserviert	0	0
4	ZNA (*20ms)	0..255	0
5	TMO (*20ms)	0..255	10
6	Anzahl Startkennungen	0..2	01
7	Startkennung 1	0..255	02
8	Startkennung 2	0..255	0
9	Anzahl Endekennungen	0..2	01
10	Endekennung 1	0..255	03
11	Endekennung 2	0..255	0
12	reserviert		
13	reserviert		
14	reserviert		
15	reserviert		

Aufbau Parameterbytes bei 3964(R) / 3964(R) mit RK512

Byte	Funktion	Wertebereich	Defaultparameter
0	Baudrate	00h: Default (9600Baud) 01h: 150Baud 02h: 300Baud 03h: 600Baud 04h: 1200Baud 05h: 1800Baud 06h: 2400Baud 07h: 4800Baud 08h: 7200Baud 09h: 9600Baud 0Ah: 14400Baud 0Bh: 19200Baud 0Ch: 38400Baud 0Dh: 57600Baud 0Fh: 76800Baud 0Eh: 115200Baud	00h: 9600Baud
1	Protokoll	03h: 3964 04h: 3964R 05h: 3964 + RK512 06h: 3964R + RK512	03h: 3964
2	Bit 1/0 Datenbits	00b: 5 Datenbits 01b: 6 Datenbits 10b: 7 Datenbits 11b: 8 Datenbits	11b: 8 Datenbits
	Bit 3/2 Parity	00b: none 01b: odd 10b: even 11b: even	00b: none
	Bit 5/4 Stopbits	01b: 1 10b: 1,5 11b: 2	01b: 1 Stopbit
	Bit 7/6	reserviert	-
3	reserviert	0	0
4	ZNA (*20ms)	0..255	0
5	ZVZ (*20ms)	0..255	10
6	QVZ (*20ms)	0..255	25
7	BWZ (*20ms)	0..255	100
8	STX Wiederholungen	0..255	5
9	DBL	0..255	6
10	Priorität	0: low 1: high	0: low
11	reserviert		
12	reserviert		
13	reserviert		
14	reserviert		
15	reserviert		

Parameter-beschreibung

Baudrate

Geschwindigkeit der Datenübertragung in bit/s (Baud).
Sie haben folgende Einstellmöglichkeiten:

00h:	Default (9600Baud)
01h:	150Baud
02h:	300Baud
03h:	600Baud
04h:	1200Baud
05h:	1800Baud
06h:	2400Baud
07h:	4800Baud
08h:	7200Baud
09h:	9600Baud
0Ah:	14400Baud
0Bh:	19200Baud
0Ch:	38400Baud
0Dh:	57600Baud
0Fh:	76800Baud
0Eh:	115200Baud

Default: 0 (9600Baud)

Protokoll

Das Protokoll, das verwendet werden soll. Diese Einstellung beeinflusst den weiteren Aufbau der Parameterdaten.
Sie haben folgende Einstellmöglichkeiten:

01h:	ASCII
02h:	STX/ETX
03h:	3964
04h:	3964R
05h:	3964 und RK512
06h:	3964R und RK512
11h:	ASCII-fragmentiert

Übertragungs-parameter-Byte

Für jeden Zeichenrahmen stehen je 3 Datenformate zur Verfügung. Die Datenformate unterscheiden sich durch Anzahl der Datenbits, mit oder ohne Paritätsbit und Anzahl der Stopbits.

Das Übertragungsparameter-Byte hat folgenden Aufbau:

Byte	Funktion	Wertebereich	Defaultparameter
2	Bit 1/0 Datenbits	00b: 5 Datenbits 01b: 6 Datenbits 10b: 7 Datenbits 11b: 8 Datenbits	11b: 8 Datenbits
	Bit 3/2 Parity	00b: none 01b: odd 10b: even 11b: even	00b: none
	Bit 5/4 Stopbits	01b: 1 10b: 1,5 11b: 2	01b: 1 Stopbit
	Bit 7/6 Flusskontrolle	00b: keine 01b: Hardware 10b: XON/XOFF	00b: keine

Datenbits

Anzahl der Datenbits, auf die ein Zeichen abgebildet wird.

Parity

Die Parität ist je nach Wert gerade oder ungerade. Zur Paritätskontrolle werden die Informationsbits um das Paritätsbit erweitert, das durch seinen Wert ("0" oder "1") den Wert aller Bits auf einen vereinbarten Zustand ergänzt. Ist keine Parität vereinbart, wird das Paritätsbit auf "1" gesetzt, aber nicht ausgewertet.

Stopbits

Die Stopbits werden jedem zu übertragenden Zeichen nachgesetzt und kennzeichnen das Ende eines Zeichens.

Flusskontrolle
(bei ASCII und STX/ETX)

Mechanismus, der den Datentransfer synchronisiert, wenn der Sender schneller Daten schickt als der Empfänger verarbeiten kann. Die Flusskontrolle kann hardware- oder softwaremäßig (XON/XOFF) erfolgen. Bei der Hardware-Flusskontrolle werden die Leitungen RTS und CTS verwendet, die dann entsprechend zu verdrahten sind.

Die Software-Flusskontrolle verwendet zur Steuerung die Steuerzeichen XON=11h und XOFF=13h. Bitte beachten Sie, dass dann Ihre Daten diese zwei Steuerzeichen nicht beinhalten dürfen.

Default: 13h (Datenbits: 8, Parität: keine, Stopbit: 1, Flusskontrolle: keine)

Zeit nach Auftrag (ZNA)

Wartezeit, die eingehalten wird, bis der nächste Sendeauftrag ausgeführt wird. Die ZNA wird in 20ms-Einheiten angegeben.

Bereich: 0 ... 255 *Default: 0*

Zeichenverzugszeit (ZVZ)
(bei ASCII, 3964(R) und RK512)

Die Zeichenverzugszeit definiert den maximal zulässigen zeitlichen Abstand zwischen zwei empfangenen Zeichen innerhalb eines Telegramms. Die ZVZ wird in 20ms-Einheiten angegeben.

Bei ZVZ=0 berechnet sich der CP anhand der Baudrate die ZVZ selbst (ca. doppelte Zeichenzeit).

Bereich: 0 ... 255 *Default: 10*

Anzahl Receivebuffer
(nur bei ASCII)

Legt die Anzahl der Empfangspuffer fest. Solange nur 1 Empfangspuffer verwendet wird und dieser belegt ist, können keine weiteren Daten empfangen werden. Durch Aneinanderreihung von bis zu 250 Empfangspuffern können die empfangenen Daten in einen noch freien Empfangspuffer umgeleitet werden.

Bereich: 1 ... 250 *Default: 1*

Time-out (TMO)
(nur bei STX/ETX)

Mit TMO definieren Sie den maximal zulässigen zeitlichen Abstand zwischen zwei Telegrammen. TMO wird in 20ms-Einheiten angegeben.

Bereich: 0 ... 255 *Default: 10*

Anzahl Startkennungen
(nur bei STX/ETX)

Hier können Sie 1 oder 2 Startkennungen einstellen. Ist "1" als Anzahl der Startkennungen eingestellt, wird der Inhalt des 2. Startkennzeichens (Byte 8) ignoriert.

Bereich: 0 ... 2 *Default: 1*

Startkennung 1 und 2 (STX)
(nur bei STX/ETX)

ASCII-Wert des Startzeichens, das einem Telegramm vorausgeschickt wird und den Start einer Übertragung kennzeichnet. Sie können 1 oder 2 Startzeichen verwenden. Bei Einsatz von 2 Startzeichen müssen Sie unter "Anzahl Startkennungen" eine 2 eintragen.

Startkennung 1, 2: *Bereich: 0 ... 255* *Default: 2 (Kennung 1)*
0 (Kennung 2)

Anzahl Endekennungen
(nur bei STX/ETX)

Hier können Sie 1 oder 2 Endekennungen einstellen. Ist "1" als Anzahl der Endekennungen eingestellt, wird der Inhalt des 2. Endekennzeichens (Byte 11) ignoriert.

Bereich: 0 ... 2 *Default: 1*

Endekennung 1 und 2 (ETX)
(nur bei STX/ETX)

ASCII-Wert des Endezeichens, das nach einem Telegramm folgt und das Ende einer Übertragung kennzeichnet. Sie können 1 oder 2 Endezeichen verwenden. Bei Einsatz von 2 Endezeichen müssen Sie unter "Anzahl Endekennungen" eine 2 eintragen.

Endekennung 1, 2: *Bereich: 0 ... 255* *Default: 3 (Kennung 1)*
0 (Kennung 2)

Quittungs-verzugszeit (QVZ)
(bei 3964(R), RK512)

Die Quittungsverzugszeit definiert den maximal zulässigen zeitlichen Abstand bis zur Quittung des Partners bei Verbindungsauf- und -abbau. Die QVZ wird in 20ms-Einheiten angegeben.

Bereich: 0 ... 255 *Default: 25*

Blockwartezeit (BWZ)
(bei 3964(R), RK512)

Die Blockwartezeit (BWZ) ist die maximale Zeitdauer zwischen der Bestätigung eines Anforderungstelegrams (DLE) und STX des Reaktions-telegramms. Die BWZ wird in 20ms-Einheiten angegeben.

Bereich: 0 ... 255 *Default: 100*

STX-Wiederholungen
(bei 3964(R), RK512)

Maximale Anzahl der Versuche des CP 240 eine Verbindung aufzubauen.

Bereich: 0 ... 255 *Default: 3*

Wiederholung Datenblöcke bei BWZ-Über-schreitung (DBL)
(bei 3964(R), RK512)

Bei Überschreiten der Blockwartezeit (BWZ) können Sie über den Parameter DBL die maximale Anzahl Wiederholungen für das Anforderungstelegramm vorgeben. Sind diese Versuche erfolglos, wird die Übertragung abgebrochen.

Bereich: 0 ... 255 *Default: 6*

Priorität
(bei 3964(R), RK512)

Ein Kommunikationspartner hat hohe Priorität, wenn sein Sendeversuch Vorrang gegenüber dem Sendewunsch des Partners hat. Bei niedriger Priorität muss dieser hinter dem Sendewunsch des Partners zurückstehen.

Bei den Protokollen 3964(R) und RK512 müssen die Prioritäten beider Partner unterschiedlich sein.

Sie haben folgende Einstellmöglichkeiten:

0: low

1: high *Default: 0 (low)*

Modbus - Grundlagen

Übersicht

Das Protokoll Modbus ist ein Kommunikationsprotokoll, das eine hierarchische Struktur mit einem Master und mehreren Slaves festlegt.

Physikalisch arbeitet Modbus über eine serielle Halbduplex-Verbindung als Punkt-zu-Punkt- unter RS232 oder als Mehrpunkt-Verbindung unter RS485.

Master-Slave-Kommunikation

Es treten keine Buskonflikte auf, da der Master immer nur mit einem Slave kommunizieren kann. Nach einer Anforderung vom Master wartet dieser solange auf die Antwort des Slaves, bis eine einstellbare Wartezeit abgelaufen ist. Während des Wartens ist eine Kommunikation mit einem anderen Slave nicht möglich.

Telegramm-Aufbau

Die Anforderungs-Telegramme, die ein Master sendet und die Antwort-Telegramme eines Slaves haben den gleichen Aufbau:

Start-zeichen	Slave-Adresse	Funktions-Code	Daten	Fluss-kontrolle	Ende-zeichen

Broadcast mit Slave-Adresse = 0

Eine Anforderung kann an einen bestimmten Slave gerichtet sein oder als Broadcast-Nachricht an alle Slaves gehen. Zur Kennzeichnung einer Broadcast-Nachricht wird die Slave-Adresse 0 eingetragen.

Nur Schreibaufträge dürfen als Broadcast gesendet werden.

ASCII-, RTU-Modus

Bei Modbus gibt es zwei unterschiedliche Übertragungsmodi

- ASCII-Modus: Jedes Byte wird im 2 Zeichen ASCII-Code übertragen. Die Daten werden durch Anfang- und Ende-Zeichen gekennzeichnet. Dies macht die Übertragung transparent aber auch langsam.
- RTU-Modus: Jedes Byte wird als ein Zeichen übertragen. Hierdurch haben Sie einen höheren Datendurchsatz als im ASCII-Modus. Anstelle von Anfang- und Ende-Zeichen wird eine Zeitüberwachung eingesetzt.

Die Modus-Wahl erfolgt bei der Parametrierung.

Modbus auf dem CP 240 von VIPA

Der CP 240 Modbus unterstützt verschiedene Betriebsarten, die nachfolgend beschrieben sind:

Modbus Master

Im *Modbus Master* Betrieb steuern Sie die Kommunikation über Ihr SPS-Anwenderprogramm. Hierzu sind die SEND- und RECEIVE-Hantierungsbausteine erforderlich. Sie haben hier die Möglichkeit unter Verwendung einer Blockung bis zu 250Byte Nutzdaten zu übertragen.

Modbus Slave Short

Im *Modbus Slave Short* Betrieb belegt der CP 240 je 16Byte für Ein- und Ausgabe-Daten an beliebiger Stelle in der CPU. Über die Adress-Parameter können Sie bei der Hardware-Konfiguration diesen Bereich definieren. Ein SPS-Programm für die Datenbereitstellung ist auf Slave-Seite nicht erforderlich. Diese Betriebsart eignet sich besonders zur schnellen Datenübertragung kleiner Datenmengen über Modbus

Modbus Slave Long

Für Daten, deren Länge 16Byte überschreiten, sollten Sie die Modbus Slave Long Betriebsart verwenden. Hier wird bei Datenempfang vom Master mit RECEIVE der Bereich an die CPU übergeben, innerhalb dessen eine Änderung stattgefunden hat. Der Datentransfer erfolgt nach folgendem Prinzip:

Der max. 1024Byte große Empfangsbereich wird in 128 8Byte-Blöcke aufgeteilt. Bei Datenänderung durch den Master werden nur die Blöcke an die CPU übergeben, in denen geändert wurde. In einem Baustein-Zyklus des RECEIVE-Bausteins können maximal 16 zusammenhängende 8Byte-Blöcke am Rückwandbus übergeben werden. Liegen die 8Byte-Blöcke nicht zusammen, ist für jeden geänderten 8Byte-Block ein Baustein-Zyklus erforderlich. Der Empfangs-DB des RECEIVE-Bausteins ist immer als ein Vielfaches von 8 anzugeben.

Mit einem SEND-Aufruf wird ein gewünschter Datenbereich in den CP übertragen, der vom Master gelesen werden kann. Schreibende Master-Zugriffe dürfen nicht außerhalb des Empfangs-Bereichs liegen!

Bitte beachten Sie, dass Modbus Slave Long ab der Baustein-Bibliothek FX000002_V120 oder höher unterstützt wird.

Hinweis!

Erst nachdem alle Daten im CP 240 vorliegen, sendet der CP 240 ein Antworttelegramm an den Master.

Inbetriebnahme

Nach Einschalten der Spannungsversorgung leuchten am Modbus-Modul die LEDs ER, TxD und RxD. Das Modul signalisiert hiermit, dass es noch keine gültigen Parameter von der CPU erhalten hat. Sobald Sie die CPU in RUN schalten, werden die Modbus-Parameter an das Modul übertragen. Bei gültigen Parametern erlöschen die LEDs ER, TxD und RxD. Das Modbus-Modul ist nun bereit für die Kommunikation.

Bei Einsatz im Master-Modus können Sie nun entsprechende Schreib-/Lesebefehle in Ihrem Anwenderprogramm ausführen lassen.

Sollte die ER-LED nicht erlöschen, liegt ein interner Fehler vor. Bei einem vorübergehenden Fehler können Sie diesen durch einen STOP-RUN-Übergang der CPU rücksetzen.

Modbus - Parametrierung

Aufbau Parameter bei Modbus

Byte	Funktion	Wertebereich	Defaultparameter
0	Baudrate	0h: 9600Baud 6h: 2400Baud 7h: 4800Baud 9h: 9600Baud Ah: 14400Baud Bh: 19200Baud Ch: 38400Baud	0h: 9600Baud
1	Protokoll	0Ah: Modbus Master ASCII short 0Bh: Modbus Master RTU short 0Ch: Modbus Slave ASCII short 0Dh: Modbus Slave RTU short 1Ch: Modbus Slave ASCII long 1Dh: Modbus Slave RTU long	Bh: Modbus-Master RTU
2	Bit 1/0 Datenbits	00b: 5 Datenbits 01b: 6 Datenbits 10b: 7 Datenbits 11b: 8 Datenbits	11b: 8 Datenbits
	Bit 3/2 Parity	00b: none 01b: odd 11b: even	00b: none
	Bit 5/4 Stopbits	01b: 1 10b: 1,5 11b: 2	01b: 1 Stopbit
	Bit 7/6	reserviert	-
3	reserviert	0	0
4	Adresse	1...255	1
5	Debug	0: Debug aus 1: Debug ein	0
6...7	Wartezeit	0: automat. Berechnung 1 ... 60000: Zeit in ms	0
8	reserviert		
9	reserviert		
10	reserviert		
11	reserviert		
12	reserviert		
13	reserviert		
14	reserviert		
15	reserviert		

Hinweis zu den Defaultparametern!

Sofern keine Parametrierung vorhanden ist und der CP 240 über Auto-Adressierung eingebunden werden soll, besitzt der CP folgende Default-Parameter:

Baudrate: 9600Baud, Protokoll: ASCII, Datenbits: 8, **Parity: even**, Stopbits: 1, Flusskontrolle: keine, ZNA: 0, ZVZ: 200ms, Receivebuffer: 1

Parameter-beschreibung

Baudrate

Geschwindigkeit der Datenübertragung in bit/s (Baud).
Sie haben folgende Einstellmöglichkeiten:

00h:	Default (9600Baud)
06h:	2400Baud
07h:	4800Baud
09h:	9600Baud
0Ah:	14400Baud
0Bh:	19200Baud
0Ch:	38400Baud

Default: 0 (9600Baud)

Protokoll

Das Protokoll, das verwendet werden soll. Diese Einstellung beeinflusst den weiteren Aufbau der Parameterdaten.

0Ah:	Modbus Master mit ASCII
0Bh:	Modbus Master mit RTU
0Ch:	Modbus Slave Short mit ASCII
0Dh:	Modbus Slave Short mit RTU
1Ch:	Modbus Slave Long mit ASCII
1Dh:	Modbus Slave Long mit RTU

Übertragungs-parameter-Byte

Für jeden Zeichenrahmen stehen je 3 Datenformate zur Verfügung. Die Datenformate unterscheiden sich durch Anzahl der Datenbits, mit oder ohne Paritätsbit und Anzahl der Stopbits.
Das Übertragungsparameter-Byte hat folgenden Aufbau:

Byte	Funktion	Wertebereich	Defaultparameter
2	Bit 1/0 Datenbits	00b: 5 Datenbits 01b: 6 Datenbits 10b: 7 Datenbits 11b: 8 Datenbits	11b: 8 Datenbits
	Bit 3/2 Parity	00b: none 01b: odd 11b: even	00b: none
	Bit 5/4 Stopbits	01b: 1 10b: 1,5 11b: 2	01b: 1 Stopbit
	Bit 7/6	reserviert	-

Datenbits — Anzahl der Datenbits, auf die ein Zeichen abgebildet wird.

Parity — Die Parität ist je nach Wert gerade oder ungerade. Zur Paritätskontrolle werden die Informationsbits um das Paritätsbit erweitert, das durch seinen Wert ("0" oder "1") den Wert aller Bits auf einen vereinbarten Zustand ergänzt. Ist keine Parität vereinbart, wird das Paritätsbit auf "1" gesetzt, aber nicht ausgewertet.

Stopbits — Die Stopbits werden jedem zu übertragenden Zeichen nachgesetzt und kennzeichnen das Ende eines Zeichens.

Default: 13h (Datenbits: 8, Parität: keine, Stopbit: 1)

Adresse — Stellen Sie hier im Slave-Modus die Modbus-Slave-Adresse ein.
Bereich: 1 ... 255 *Default: 1*

Debug — Dieser Modus ist für interne Tests. Diese Funktion sollte immer deaktiviert sein.
Bereich: 0, 1 *Default: 0*

Wartezeit — Hier ist im Master-Modus eine Wartezeit in ms vorzugeben. Mit "0" wird die Wartezeit protokollabhängig nach folgender Formel automatisch ermittelt:

Modbus ASCII: $50ms + \frac{2926000ms}{Baudrate} \cdot s$ mit Baudrate in Bit/s

Modbus RTU: $50ms + \frac{5190000ms}{Baudrate} \cdot s$ mit Baudrate in Bit/s

Im Slave-Modus wird dieser Parameter ignoriert.

Modbus - Einsatz

Übersicht

Sie können den CP 240 Modbus sowohl im Master- als auch im Slave-Modus betreiben. In beiden Modi belegt das Modul für Ein- und Ausgangs-Daten je 16Byte an beliebiger Stelle in der CPU.

Für den Einsatz unter Modbus ist immer eine Hardware-Konfiguration durchzuführen.

Voraussetzung für den Betrieb

Folgende Komponenten sind zum Einsatz der System 200V Modbus-Module erforderlich:

- Je 1 System 200V bestehend aus CPU 21x und CP 240
- Siemens SIMATIC Manager
- Programmierkabel für MPI-Kopplung (z.B. Green Cable von VIPA)
- GSD-Datei **VIPA_21x.gsd** (V1.67 oder höher)
- VIPA Hantierungsbausteine Fx000002_V120.zip oder höher
- Serielle Verbindung zwischen beiden CP 240

Parametrierung

Für den CP 240 ist immer eine Hardware-Konfiguration durchzuführen. Hierfür ist die Einbindung der VIPA_21x.gsd im Hardware-Katalog erforderlich. Die Parametrierung erfolgt nach folgender Vorgehensweise:

- Starten Sie den Siemens SIMATIC Manager
- Installieren Sie die GSD-Datei **VIPA_21x.gsd** im Hardware-Katalog.
- Legen Sie im Hardware-Konfigurator mit der CPU 315-2DP (6ES7 315-2AF03 V1.2) ein virtuelles Profibussystem an.
- Binden Sie an dieses System das Slave-System "VIPA_CPU21x" an und geben Sie diesem die Profibus-Adresse 1.
- Projektieren Sie beginnend mit der CPU 21x Ihr System 200V. Verwenden Sie einen mit "Modbus" bezeichneten CP.
- Parametrieren Sie den CP 240 nach Ihren Vorgaben. Der CP 240 belegt in der CPU je 16Byte für Ein- und Ausgabe.
- Übertragen Sie Ihr Projekt in die SPS.

SPS-Programm

Mit Ausnahme bei "Modbus Slave Short" ist immer für die Kommunikation zusätzlich ein SPS-Programm erforderlich. Hierbei erfolgt die Kommunikation über Hantierungsbausteine, die Sie in Form der VIPA-Library **Fx000002_V120.zip** oder höher im Siemens SIMATIC Manager einbinden können. Die Library ist auf der VIPA-CD "ToolDemo" verfügbar. Sie finden diese auch unter ftp.vipa.de.

Hinweis!

Näheres zur Installation der GSD-Datei und der Library finden Sie im Teil "Projektierung".

Kommunikationsmöglichkeiten

Nachfolgend sollen die Kommunikations-Möglichkeiten zwischen Modbus Master und Modbus Slave an folgenden Kombinationsmöglichkeiten gezeigt werden:

- CP 240 Modbus Master ↔ CP 240 Modbus Slave Short
- CP 240 Modbus Master ↔ CP 240 Modbus Long

Master ↔ Slave Short

Modbus Master

Die Kommunikation im Master-Modus erfolgt über Datenbausteine unter Einsatz der CP 240 SEND-RECEIVE-Hantierungsbausteine. Hier können unter Einsatz einer Blockung bis zu 250Byte Nutzdaten übertragen werden.

Modbus Slave Short

Im *Modbus Slave Short* Modus ist die Anzahl der Nutzdaten für Ein- und Ausgabe auf 16Byte begrenzt. Hierbei ist für den Einsatz auf Slave-Seite lediglich eine Hardware-Konfiguration durchzuführen.

Vorgehensweise

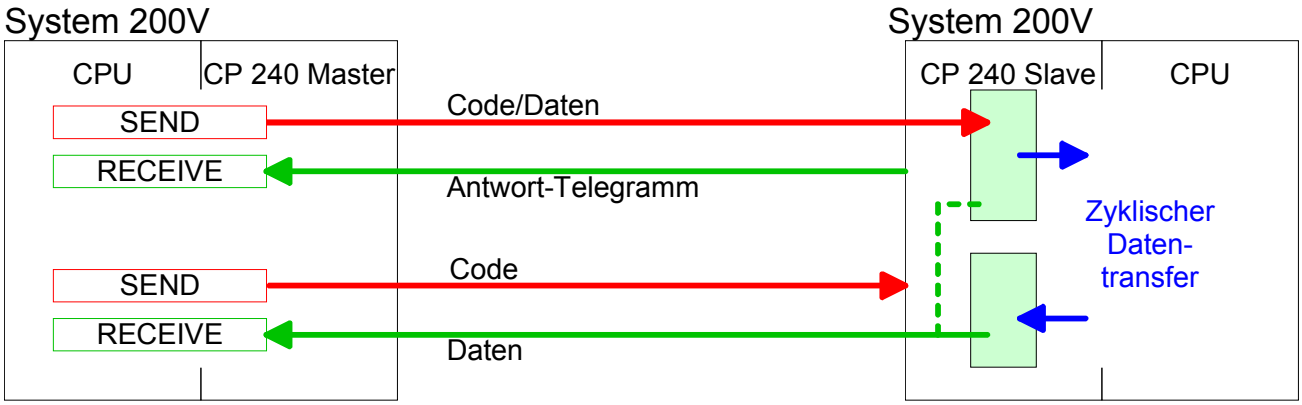
- Bauen Sie für Master- und Slave-Seite je ein System 200V bestehend aus jeweils einer CPU 21x und einem CP 240 auf und verbinden Sie beide Systeme über die serielle Schnittstelle.
- Projektieren Sie die Master-Seite.

  Die Parametrierung des CP 240 als Modbus-Master erfolgt über die Hardware-Konfiguration. Zusätzlich ist für die Kommunikation ein SPS-Anwenderprogramm erforderlich, das nach folgender Struktur aufgebaut sein sollte:

  OB 1: Aufruf des FC0 (SEND) mit Fehlerauswertung. Hierbei ist das Telegramm gemäß den Modbus-Vorgaben im Sendebaustein abzulegen.

  Aufruf des FC1 (RECEIVE) mit Fehlerauswertung. Gemäß den Modbus-Vorgaben werden die Daten im Empfangsbaustein abgelegt.
- Projektieren Sie die Slave-Seite

  Die Parametrierung des CP 240 erfolgt über die Hardware-Konfiguration. Geben Sie hier für Ein- und Ausgabe-Bereich die Startadresse an, ab welcher die fixe Anzahl von 16Byte für Ein- und Ausgabe an beliebiger Stelle in der CPU abliegen.



Master ↔ Slave Long

Modbus Master

Die Kommunikation im Master-Modus erfolgt über Datenbausteine unter Einsatz der CP 240 SEND-RECEIVE-Hantierungsbausteine. Hier können unter Einsatz einer Blockung bis zu 250Byte Nutzdaten übertragen werden.

Modbus Slave Long

Im *Modbus Slave Long* Modus wird nur ein geänderter Datenbereich beginnend bei 0 mit RECEIVE an die CPU übertragen. Fordert der Master Daten an, so ist dafür Sorge zu tragen, dass sich die relevanten Daten im CP befinden. Mit einem SEND-Aufruf wird ein gewünschter Datenbereich von 0 beginnend in den CP übertragen.

Vorgehensweise

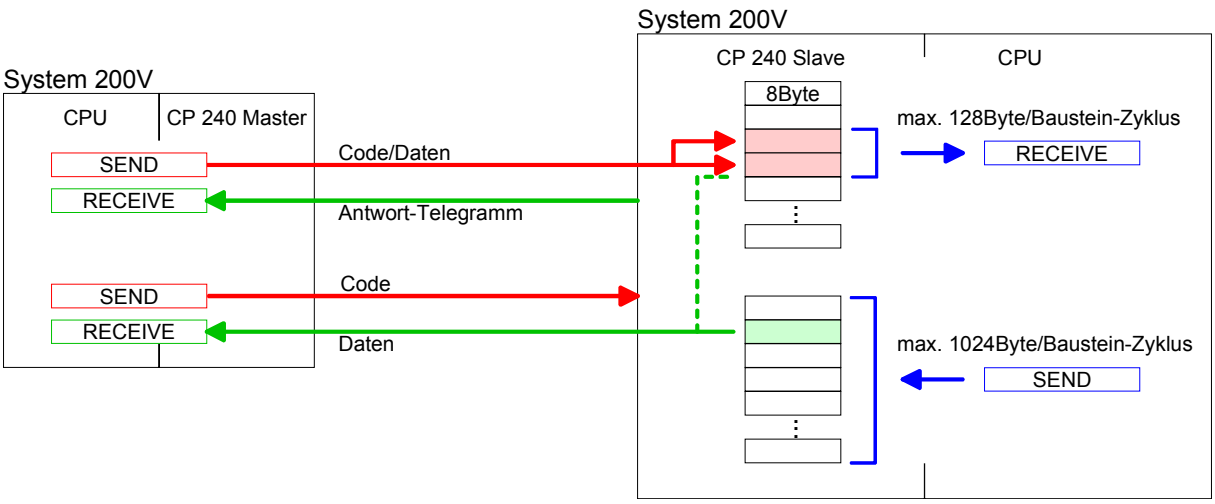
- Bauen Sie für Master- und Slave-Seite je ein System 200V bestehend aus jeweils einer CPU 21x und einem CP 240 auf und verbinden Sie beide Systeme über die serielle Schnittstelle.
- Projektieren Sie die Master-Seite.

  Die Projektierung auf der Master-Seite erfolgt auf die gleiche Weise, wie im Beispiel weiter oben beschrieben.
- Projektieren Sie die Slave-Seite

  Die Parametrierung des CP 240 als Modbus-Slave erfolgt über die Hardware-Konfiguration. Zusätzlich ist für die Kommunikation ein SPS-Anwenderprogramm erforderlich, das nach folgender Struktur aufgebaut sein sollte:

  OB 1: Aufruf des FC0 (SEND) mit Fehlerauswertung. Hierbei wird ein Bereich von 0 beginnend im CP 240 abgelegt, auf den vom Master über Modbus zugegriffen werden kann.

  Mit dem FC1 (RECEIVE) mit Fehlerauswertung können Sie einen Datenbereich in die CPU übertragen. Der max. 1024Byte große Empfangsbereich wird in 128 8Byte-Blöcke aufgeteilt. Bei Datenänderung durch den Master werden nur die Blöcke an die CPU übergeben, in denen geändert wurde. In einem Baustein-Zyklus des RECEIVE-Bausteins können maximal 16 zusammenhängende 8Byte-Blöcke am Rückwandbus übergeben werden. Liegen die 8Byte-Blöcke nicht zusammen, ist für jeden geänderten 8Byte-Block ein Baustein-Zyklus erforderlich. Der Empfangs-DB ist bei Aufruf des RECEIVE-Bausteins immer als ein Vielfaches von 8 anzugeben. Schreibende Master-Zugriffe dürfen nicht außerhalb des Empfangs-Bereichs liegen!



Zugriff auf mehrere Slaves

Bei Einsatz mehrerer Slaves können unter RS485 keine Buskonflikte auftreten, da der Master immer nur mit einem Slave kommunizieren kann. Der Master schickt an den über die Adresse spezifizierten Slave ein Kommandotelegramm und wartet eine gewisse Zeit, in der der Slave sein Antworttelegramm senden kann. Während des Wartens ist eine Kommunikation mit einem anderen Slave nicht möglich.

Zur Kommunikation mit mehreren Slaves ist für jeden Slave ein SEND-Datenbaustein für das Kommandotelegramm und ein RECEIVE-Datenbaustein für das Antworttelegramm erforderlich.

Eine Applikation mit mehreren Slaves würde aus entsprechend vielen Datenbausteinen mit Kommandos bestehen.

Diese werden der Reihe nach abgearbeitet:

1. Slave: Sende Kommandotelegramm an Slave-Adresse 1.Slave
   Empfange Antworttelegramm von Slave-Adresse 1.Slave
   Werte Antworttelegramm aus

2. Slave: Sende Kommandotelegramm an Slave-Adresse 2.Slave
   Empfange Antworttelegramm von Slave-Adresse 2.Slave
   Werte Antworttelegramm aus

...

Eine Anforderung kann an einen bestimmten Slave gerichtet sein oder als Broadcast-Nachricht an alle Slaves gehen. Zur Kennzeichnung einer Broadcast-Nachricht wird die Slave-Adresse 0 eingetragen.

Nur Schreibaufträge dürfen als Broadcast gesendet werden.

Hinweis!

Nach einem Broadcast wartet der Master nicht auf ein Antworttelegramm.

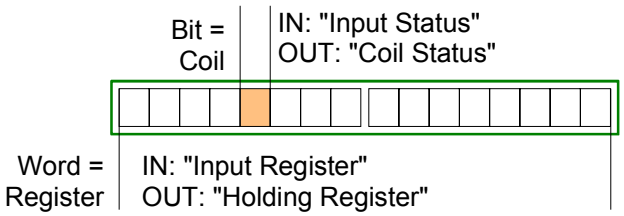
Master-Ausgabe-Bereich beschreiben

Durch "Ver-Oderung" des FC 0 Parameters *ANZ* mit 4000h werden zu sendende Slave-Daten nicht im Master-Eingabe- sondern im Master-Ausgabe-Bereich abgelegt. Da der Master unter Einsatz von Funktionscodes diesen Bereich lesen kann, können Sie diese Funktionalität beispielsweise zur direkten Fehlerübermittlung an den Master verwenden.

Modbus - Funktionscodes

Namens-konventionen

Für Modbus gibt es Namenskonventionen, die hier kurz aufgeführt sind:



- Modbus unterscheidet zwischen Bit- und Wortzugriff;
  Bits = "Coils" und Worte = "Register".
- Bit-Eingänge werden als "Input-Status" bezeichnet und Bit-Ausgänge als "Coil-Status".
- Wort-Eingänge werden als "Input-Register" und Wort-Ausgänge als "Holding-Register" bezeichnet.

Bereichs-definitionen

Üblicherweise erfolgt unter Modbus der Zugriff mittels der Bereiche 0x, 1x, 3x und 4x.

Mit 0x und 1x haben Sie Zugriff auf *digitale* Bit-Bereiche und mit 3x und 4x auf *analoge* Wort-Bereiche.

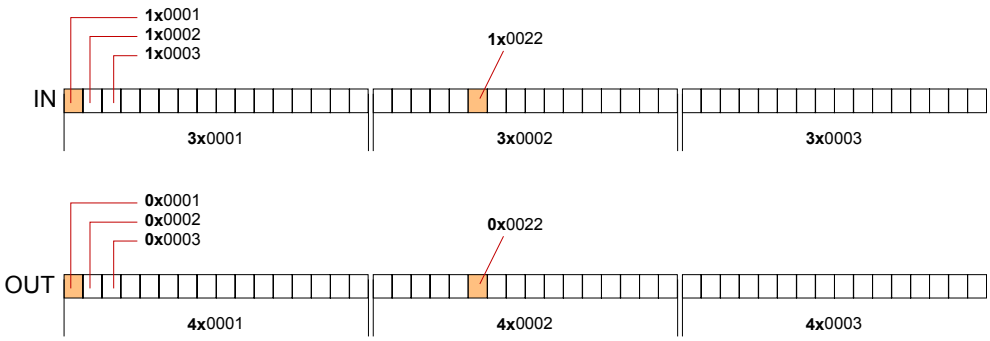
Da aber beim CP 240 von VIPA keine Unterscheidung zwischen Digital- und Analogdaten stattfindet, gilt folgende Zuordnung:

0x: Bit-Bereich für Ausgabe-Daten des Masters
Zugriff über Funktions-Code 01h, 05h, 0Fh

1x: Bit-Bereich für Eingabe-Daten des Masters
Zugriff über Funktions-Code 02h

3x: Wort-Bereich für Eingabe-Daten des Masters
Zugriff über Funktions-Code 04h

4x: Wort-Bereich für Ausgabe-Daten des Masters
Zugriff über Funktions-Code 03h, 06h, 10h



Eine Beschreibung der Funktions-Codes finden Sie auf den Folgeseiten.

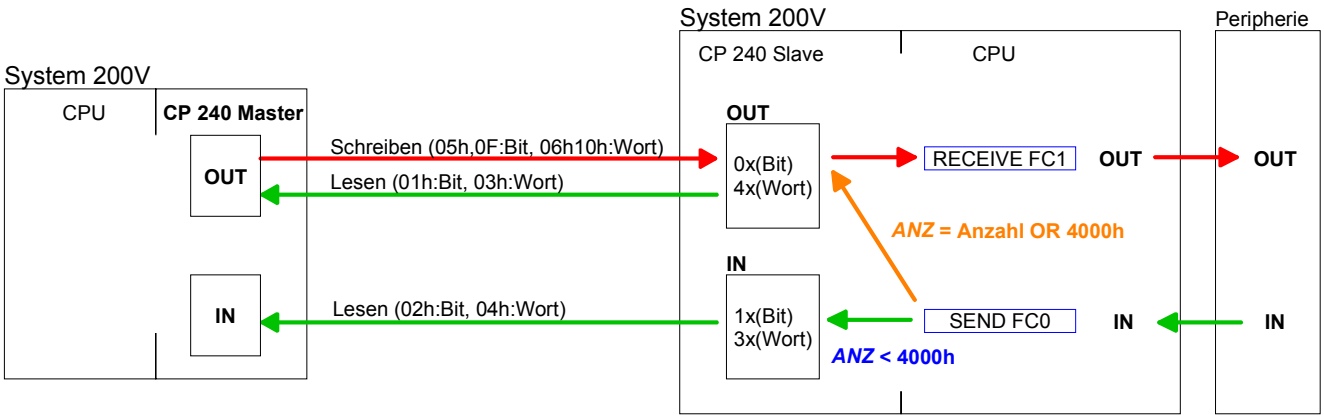
Übersicht

Mit folgenden Funktionscodes können Sie von einem Modbus-Master auf einen Slave zugreifen. Die Beschreibung erfolgt immer aus Sicht des Masters:

Code	Befehl	Beschreibung
01h	Read n Bits	n Bit lesen von Master-Ausgabe-Bereich 0x
02h	Read n Bits	n Bit lesen von Master-Eingabe-Bereich 1x
03h	Read n Words	n Worte lesen von Master-Ausgabe-Bereich 4x
04h	Read n Words	n Worte lesen von Master-Eingabe-Bereich 3x
05h	Write 1 Bit	1 Bit schreiben in Master-Ausgabe-Bereich 0x
06h	Write 1 Word	1 Wort schreiben in Master-Ausgabe-Bereich 4x
0Fh	Write n Bits	n Bit schreiben in Master-Ausgabe-Bereich 0x
10h	Write n Words	n Worte schreiben in Master-Ausgabe-Bereich 4x

Sichtweise für "Eingabe"- und "Ausgabe"-Daten

Die Beschreibung der Funktionscodes erfolgt immer aus Sicht des Masters. Hierbei werden Daten, die der Master an den Slave schickt, bis zu ihrem Ziel als "Ausgabe"-Daten (OUT) und umgekehrt Daten, die der Master vom Slave empfängt als "Eingabe"-Daten (IN) bezeichnet.



Antwort des Slaves

Liefert der Slave einen Fehler zurück, wird der Funktionscode mit 80h "verodert" zurückgesendet.

Ist kein Fehler aufgetreten, wird der Funktionscode zurückgeliefert.

Slave-Antwort:	Funktionscode OR 80h	→ Fehler
	Funktionscode	→ OK

Byte-Reihenfolge im Wort

Für die Byte-Reihenfolge im Wort gilt immer:

1 Wort

High-Byte	Low-Byte

Prüfsumme CRC, RTU, LRC

Die aufgezeigten Prüfsummen CRC bei RTU- und LRC bei ASCII-Modus werden automatisch an jedes Telegramm angehängt. Sie werden nicht im Datenbaustein angezeigt.

Read n Bits
01h, 02h

Code 01h: n Bit lesen von Master-Ausgabe-Bereich 0x
Code 02h: n Bit lesen von Master-Eingabe-Bereich 1x

Kommandotelegramm

Slave-Adresse	Funktions-Code	Adresse 1. Bit	Anzahl der Bits	Prüfsumme CRC/LRC
1Byte	1Byte	1Wort	1Wort	1Wort

Antworttelegramm

Slave-Adresse	Funktions-Code	Anzahl der gelesenen Bytes	Daten 1. Byte	Daten 2. Byte	...	Prüfsumme CRC/LRC
1Byte	1Byte	1Byte	1Byte	1Byte max. 250Byte		1Wort

Read n Words
03h, 04h

03h: n Worte lesen von Master-Ausgabe-Bereich 4x
04h: n Worte lesen von Master-Eingabe-Bereich 3x

Kommandotelegramm

Slave-Adresse	Funktions-Code	Adresse Bit	Anzahl der Worte	Prüfsumme CRC/LRC
1Byte	1Byte	1Wort	1Wort	1Wort

Antworttelegramm

Slave-Adresse	Funktions-Code	Anzahl der gelesenen Bytes	Daten 1. Wort	Daten 2. Wort	...	Prüfsumme CRC/LRC
1Byte	1Byte	1Byte	1Wort	1Wort max. 125Worte		1Wort

Write 1 Bit
05h

Code 05h: 1 Bit schreiben in Master-Ausgabe-Bereich 0x
Eine Zustandsänderung erfolgt unter "Zustand Bit" mit folgenden Werten:

"Zustand Bit" = 0000h → Bit = 0
"Zustand Bit" = FF00h → Bit = 1

Kommandotelegramm

Slave-Adresse	Funktions-Code	Adresse Bit	Zustand Bit	Prüfsumme CRC/LRC
1Byte	1Byte	1Wort	1Wort	1Wort

Antworttelegramm

Slave-Adresse	Funktions-Code	Adresse Bit	Zustand Bit	Prüfsumme CRC/LRC
1Byte	1Byte	1Wort	1Wort	1Wort

Write 1 Word 06h

Code 06h: 1 Wort schreiben in Master-Ausgabe-Bereich 4x

Kommandotelegramm

Slave-Adresse	Funktions-Code	Adresse Wort	Wert Wort	Prüfsumme CRC/LRC
1Byte	1Byte	1Wort	1Wort	1Wort

Antworttelegramm

Slave-Adresse	Funktions-Code	Adresse Wort	Wert Wort	Prüfsumme CRC/LRC
1Byte	1Byte	1Wort	1Wort	1Wort

Write n Bits 0Fh

Code 0Fh: n Bit schreiben in Master-Ausgabe-Bereich 0x

Bitte beachten Sie, dass die Anzahl der Bits zusätzlich in Byte anzugeben sind.

Kommandotelegramm

Slave-Adresse	Funktions-Code	Adresse 1. Bit	Anzahl der Bits	Anzahl der Bytes	Daten 1. Byte	Daten 2. Byte	...	Prüfsumme CRC/LRC
1 Byte	1 Byte	1 Wort	1 Wort	1 Byte	1 Byte	1 Byte	1 Byte	1 Wort
					max. 250 Byte			

Antworttelegramm

Slave-Adresse	Funktions-Code	Adresse 1. Bit	Anzahl der Bits	Prüfsumme CRC/LRC
1 Byte	1 Byte	1 Wort	1 Wort	1 Wort

Write n Words 10h

Code 10h: n Worte schreiben in Master-Ausgabe-Bereich

Kommandotelegramm

Slave-Adresse	Funktions-Code	Adresse 1. Wort	Anzahl der Worte	Anzahl der Bytes	Daten 1. Wort	Daten 2. Wort	...	Prüfsumme CRC/LRC
1 Byte	1 Byte	1 Wort	1 Wort	1 Byte	1 Wort	1 Wort	1 Wort	1 Wort
					max. 125 Worte			

Antworttelegramm

Slave-Adresse	Funktions-Code	Adresse 1. Wort	Anzahl der Worte	Prüfsumme CRC/LRC
1 Byte	1 Byte	1 Wort	1 Wort	1 Wort

Modbus - Fehlermeldungen

Übersicht

Bei der Kommunikation unter Modbus gibt es folgende 2 Fehlerarten:

- Master bekommt keine gültigen Daten
- Slave antwortet mit einer Fehlermeldung

Master bekommt keine gültigen Daten

Antwortet der Slave nicht innerhalb der vorgegebenen Wartezeit oder ist ein Telegramm fehlerbehaftet, trägt der Master im Empfangs-Baustein eine Fehlermeldung in Klartext ein.

Folgende Fehlermeldungen sind möglich:

ERROR01 NO DATA	*Error no data* Innerhalb der Wartezeit wurde kein Telegramm empfangen.
ERROR02 D LOST	*Error data lost* Es liegen keine Daten vor, da entweder der Empfangspuffer voll ist oder ein Fehler im Empfangsteil aufgetreten ist.
ERROR03 F OVERF	*Error frame overflow* Das Telegrammende wurde nicht erkannt und die maximale Telegrammlänge überschritten.
ERROR04 F INCOM	*Error frame incomplete* Es wurde nur ein Teiltelegramm empfangen.
ERROR05 F FAULT	*Error frame fault* Die Checksumme innerhalb eines Telegramms ist nicht korrekt.
ERROR06 F START	*Error frame start* Das Startzeichen ist falsch. Dieser Fehler kann ausschließlich unter Modbus-ASCII auftreten.

Slave antwortet mit einer Fehlermeldung

Liefert der Slave einen Fehler zurück, so wird der Funktionscode wie nachfolgend gezeigt mit 80h "verodert" zurückgesendet:

DB11.DBD 0	**DW#16#05900000**	**Antworttelegramm**
	mit 05 →	Slave-Adresse 05h
	90 →	Funktionscode 90h (Fehlermeldung, da 10h OR 80h = 90h)
	0000 →	Die Restdaten sind irrelevant, da Fehler rückgemeldet wurde.

Modbus - Beispiel

Übersicht

In dem nachfolgenden Beispiel wird eine Kommunikation zwischen einem Master und einem Slave über Modbus aufgebaut. Weiter soll das Beispiel zeigen, wie Sie unter Einsatz der Hantierungsbausteine auf einfache Weise die Kontrolle über die Kommunikationsvorgänge haben.

Bei Bedarf können Sie das Beispielprojekt von VIPA beziehen.

Voraussetzung

Folgende Komponenten sind für das Beispiel erforderlich:

- 2 System 200V bestehend aus CPU 21x mit CP 240
- Programmierkabel für MPI-Kopplung (z.B. Green Cable von VIPA)
- Serielles Verbindungskabel zur Verbindung beider CP 240

Vorgehensweise

- Bauen Sie ein Modbus-System bestehend aus Master-, Slave-System und verbinden Sie beide Systeme seriell.
- Projektieren Sie die Master-Seite! Öffnen Sie hierzu das Beispielprojekt in Ihrem Projektiertool. Passen Sie die Übertragungsparameter entsprechend an.
  Stellen Sie unter *Protokoll* "Modbus Master RTU" ein.
  Editieren Sie den OB1 und gleichen Sie ggf. die Modul-Adressen mit den Adressen der Parametrierung ab.
  Übertragen Sie Ihr Projekt in die CPU 21x der Master-Seite.
- Projektieren Sie die Slave Seite. Öffnen Sie hierzu das Beispielprojekt in Ihrem Projektiertool. Passen Sie in der Hardware-Konfiguration die CP 240 Parameter entsprechend an. Stellen Sie unter *Protokoll* "Modbus Slave RTU Short" ein. Geben Sie unter *Adresse* eine Slave-Adresse an. Für die Kommunikation unter Modbus ist auf Slave-Seite kein zusätzliches SPS-Programm erforderlich.

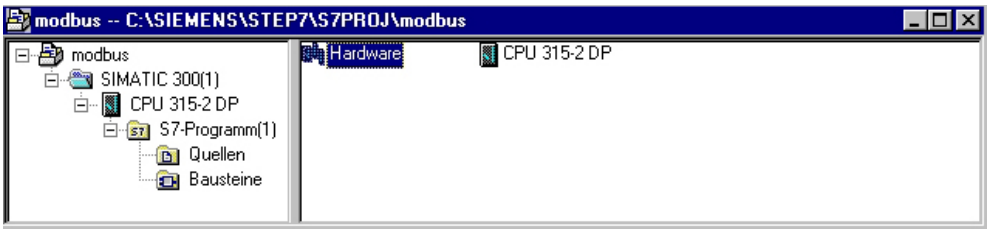
Projekt dearchivieren

Zum Dearchivieren in Ihr Konfigurationstool gehen Sie nach folgenden Schritten vor:

- Starten Sie den Siemens SIMATIC Manager.
- Zum Entpacken der Datei Modbus.zip gehen Sie auf **Datei** > *dearchivieren*.
- Wählen sie die Beispieldatei Modbus.zip aus und geben Sie als Zielverzeichnis "s7proj" an.
- Öffnen Sie nach dem Entpacken das Projekt.

Projekt-Struktur

Das Projekt hat folgende Struktur:



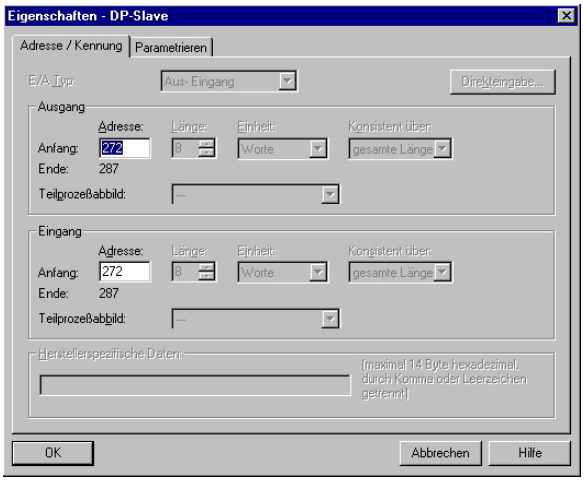
Master-Projektierung

Das Beispiel beinhaltet schon das SPS-Programm und die Parameter für den Modbus-Master. Sie müssen lediglich die Modbus-Parameter anpassen.

Parametrierung

Starten Sie hierzu den Hardware-Konfigurator und wählen Sie das Modul 240-1CA20 an. Durch Doppelklick gelangen Sie in die Parametrierung:

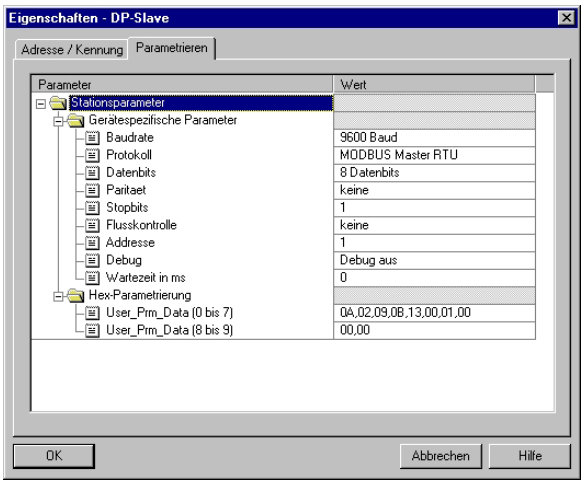
Dialog für Adress-Eingabe



Hier können Sie angeben, ab welcher Adresse die 16Byte für Ein- und Ausgabe in der CPU abliegen.

Bitte beachten Sie, dass Sie die Adressen, die Sie hier ändern, auch in Ihren SEND- und RECEIVE-Bausteinen ändern müssen.

Dialog für Modbus-Parameter



In diesem Teil der Parametrierung stellen Sie die Modbus-Parameter ein.

Folgende Parameter müssen bei allen Busteilnehmern gleich sein: Baudrate, Datenbits, Parität, Stopbits und Flusskontrolle.

Stellen Sie unter *Protokoll* "Modbus Master RTU" ein

Die Angabe einer Adresse ist nur auf der Slave-Seite erforderlich.

Bei der Master-Parametrierung wird die Adresse ignoriert.

SPS-Programm Die gewünschten Modbus-Befehle geben Sie über Ihr SPS-Programm vor. Im vorliegenden Beispiel wird im OB1 der Einsatz von SEND und RECEIVE gezeigt.

```
OB 1:


    CALL     FC       0                    //"SEND"
      ADR           :=256                  //Ausgangsadresse des Moduls
      _DB           :=DB10                 //In diesem Datenbaustein erstellen
                                           //Sie das zu sendende Telegramm
      ABD           :=W#16#0               //Ab diesem Byte-Offset beginnt
                                           //das Telegramm im _DB
      ANZ           :=MW12                 //Telegrammlaenge (zu sendende Laenge) in Byte
      PAFE          :=MB14                 //Fehlerbyte
      FRG           :=M1.0                 //Sendeanstoss (1=Anstoss, geht auf 0
                                           //wenn Senden abgeschlossen)
      GESE          :=MW16                 //intern erforderlich
      ANZ_INT       :=MW18                 //intern erforderlich
      ENDE_KOM      :=M2.0                 //intern erforderlich
      LETZTER_BLOCK:=M2.1                  //intern erforderlich
      SENDEN_LAEUFT:=M2.2                  //intern erforderlich
      FEHLER_KOM    :=M2.3                 //intern erforderlich

    CALL     FC       1                    //"RECEIVE"
      ADR           :=256                  //Eingangsadresse des Moduls
      _DB           :=DB11                 //In diesem Datenbaustein wird das
                                           //empfangene Telegramm abgelegt
      ABD           :=W#16#0               //Ab diesem Byte-Offset beginnt das Telegramm im _DB
      ANZ           :=MW22                 //Telegrammlaenge (empfangene Laenge) in Byte
      EMFR          :=M1.1                 //Empfangsstatus (1=Telegramm komplett empfangen)
      PAFE          :=MB34                 //Fehlerbyte
      GEEM          :=MW36                 //intern erforderlich
      ANZ_INT       :=MW38                 //intern erforderlich
      EMPF_LAEUFT   :=M3.0                 //intern erforderlich
      LETZTER_BLOCK:=M3.1                  //intern erforderlich
      FEHL_EMPF     :=M3.2                 //intern erforderlich

     U     M      1.1                      //solange Empfangsstatus=1 ist wird kein neues
                                           //Telegramm eingetragen
     R     M      1.1                      //daher muss der Empfangsstatus mit 0 quittiert werden
```

Passen Sie noch ggf. die Adressen, die der CP in der CPU belegt, an die Adressen in Ihrer Parametrierung an und übertragen Sie die Hardware-Konfiguration in Ihre CPU 21x des Master-Systems.

Hinweis!

Aufgrund der Übertragung der Daten in 8Byte-Blöcken, ist darauf zu achten, dass die Länge des Empfangsdatenbereichs ein Vielfaches von 8 ist. Auch sollten die schreibenden Master-Zugriffe nicht außerhalb des Empfangsbereichs liegen, da ansonsten der RECEIVE-Baustein einen Bereichsfehler meldet.

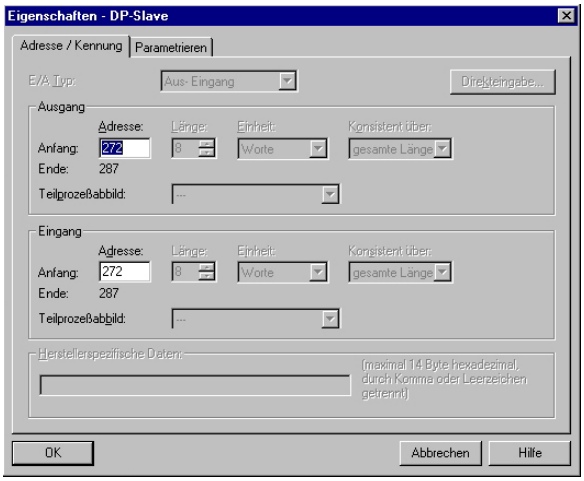
Slave-Projektierung

Für die Projektierung des Slave sind nur die Modbus-Parameter anzupassen. Ein SPS-Programm ist nicht erforderlich, da die Quell- und Zieldaten im Master-Telegramm mitgeliefert werden.

Parametrierung

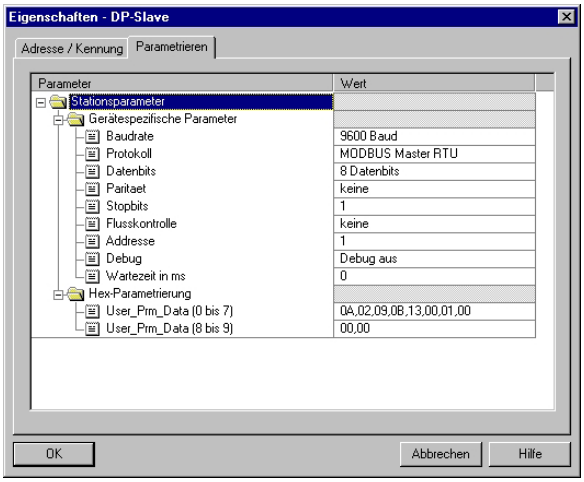
Zur Parametrierung des Slave-Moduls öffnen Sie das Beispielprojekt in Ihrem Hardware-Konfigurator. Wählen Sie das Modul 240-1CA20 an. Durch Doppelklick gelangen Sie in die Parametrierung.

Dialog für Adress-Eingabe



Hier können Sie angeben, ab welcher Adresse die 16Byte für Ein- und Ausgabe in der CPU abliegen.

Dialog für Modbus-Parameter



In diesem Teil der Parametrierung stellen Sie die Modbus-Parameter ein.

Folgende Parameter müssen bei allen Busteilnehmern gleich sein:

Baudrate, Datenbits, Parität, Stopbits und Flusskontrolle.

Geben Sie unter *Adresse* für den Slave eine gültige Modbus-Adresse an.

Übertragen Sie die Parametrierung in Ihre CPU des Slave-Systems.

Telegramme senden und empfangen

Öffnen Sie die Variablentabelle **Tabelle1** des Beispielprojekts und gehen Sie online.

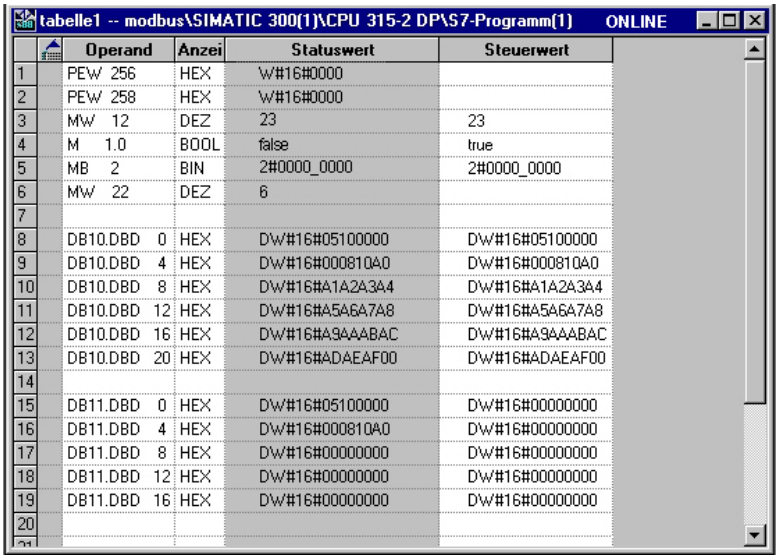


tabelle1 -- modbus\SIMATIC 300(1)\CPU 315-2 DP\S7-Programm(1) ONLINE

	Operand	Anzei	Statuswert	Steuerwert
1	PEW 256	HEX	W#16#0000	
2	PEW 258	HEX	W#16#0000	
3	MW 12	DEZ	23	23
4	M 1.0	BOOL	false	true
5	MB 2	BIN	2#0000_0000	2#0000_0000
6	MW 22	DEZ	6	
7				
8	DB10.DBD 0	HEX	DW#16#05100000	DW#16#05100000
9	DB10.DBD 4	HEX	DW#16#000810A0	DW#16#000810A0
10	DB10.DBD 8	HEX	DW#16#A1A2A3A4	DW#16#A1A2A3A4
11	DB10.DBD 12	HEX	DW#16#A5A6A7A8	DW#16#A5A6A7A8
12	DB10.DBD 16	HEX	DW#16#A9AAABAC	DW#16#A9AAABAC
13	DB10.DBD 20	HEX	DW#16#ADAEAF00	DW#16#ADAEAF00
14				
15	DB11.DBD 0	HEX	DW#16#05100000	DW#16#00000000
16	DB11.DBD 4	HEX	DW#16#000810A0	DW#16#00000000
17	DB11.DBD 8	HEX	DW#16#00000000	DW#16#00000000
18	DB11.DBD 12	HEX	DW#16#00000000	DW#16#00000000
19	DB11.DBD 16	HEX	DW#16#00000000	DW#16#00000000
20				

Sende-Baustein DB10

DB10.DBD 0	**DW#16#05100000** mit 05 → 10 → 0000 →	**Kommandotelegramm** Slave-Adresse 05h Funktionscode 10h (write n words) Offset 0000h
DB10.DBD 4	**DW#16#000810A0** mit 0008 → 10 → A0 →	**Kommandotelegramm + 1 Datenbyte** Wordcount 0008h Bytecount 10h Beginn 16 Byte Daten mit A0h
DB10.DBD 8	**DW#16#A1A2A3A4**	**Daten Byte 2 ... 5**
DB10.DBD 12	**DW#16#A5A6A7A8**	**Daten Byte 6 ... 9**
DB10.DBD 16	**DW#16#A9AAABAC**	**Daten Byte 10 ... 13**
DB10.DBD 20	**DW#16#ADAEAF00** mit ADAEAF → 00 →	**Daten Byte 14 ... 16 + 1 Byte nicht ben.** Daten Byte 14 ... 16 vom Modul nicht mehr belegt

Empfangs-Baustein DB11

DB11.DBD 0	**DW#16#05100000** mit 05 → 10 → 0000 →	**Antworttelegramm** Slave-Adresse 05h Funktionscode 10h (kein Fehler) Offset 0000h
DB11.DBD 4	**DW#16#000810A0** mit 0008 → 10 → A0 →	**Antworttelegramm + 1 Datenbyte** Wordcount 0008h Bytecount 10h Beginn 16 Byte Daten mit A0h (bei Schreibbefehl irrelevant)
DB11.DBD 8	**DW#16#00000000**	**Daten Byte 2 ... 5**
DB11.DBD 12	**DW#16#00000000**	**Daten Byte 6 ... 9**
DB11.DBD 16	**DW#16#00000000**	**Daten Byte 10 ... 13**
DB11.DBD 20	**DW#16#00000000** mit 000000 → 00 →	**Daten Byte 14 ... 16 + 1 Byte nicht ben.** Daten Byte 14 ... 16 vom Modul nicht mehr belegt

Empfangs-Baustein mit Fehlerrück-meldung

Slave antwortet nicht auf das Kommando des Masters

Antwortet der Slave nicht innerhalb der vorgegebenen Time-out-Zeit, trägt der Master im Empfangs-Baustein folgende Fehlermeldung ein:

ERROR01 NO DATA. In der Hex-Darstellung werden folgende Werte eingetragen:

DB11.DBD 0	**DW#16#4552524F**	**Antworttelegramm**
	mit 45 →	E
	52 →	R
	52 →	R
	4F →	O
DB11.DBD 4	**DW#16#52000120**	**Antworttelegramm**
	mit 52 →	R
	0001 →	0001h:1 (als Wort)
	20 →	" "
DB11.DBD 8	**DW#16#4E4F2044**	**Antworttelegramm**
	mit 4E →	N
	4F →	O
	20 →	" "
	44 →	D
DB11.DBD 12	**DW#16#41544100**	**Antworttelegramm**
	mit 41 →	A
	54 →	T
	41 →	A
	00 →	00h: (Nullterminierung)

.
.
.

Slave antwortet mit einer Fehlermeldung

Liefert der Slave einen Fehler zurück, so wird der Funktionscode mit 80h "verodert" zurückgesendet.

DB11.DBD	**DW#16#05900000**	**Antworttelegramm**
	mit 05 →	Slave-Adresse 05h
	90 →	Funktionscode 90h (Fehlermeldung, da 10h OR 80h = 90h)
	0000 →	Die Restdaten sind irrelevant, da Fehler rückgemeldet wurde.

Technische Daten

CP 240 RS232

Elektrische Daten	VIPA 240-1BA20
Anzahl der Kanäle	1
Spannungsversorgung	5V über Rückwandbus
Stromaufnahme	max. 150mA
Statusanzeige	über LED auf der Frontseite
Protokolle	ASCII, ASCII-fragmentiert, STX/ETX, 3964(R), 3964(R) mit RK512, Modbus (Master, Slave)
Anschlüsse / Schnittstellen	9pol. SubD-Stecker für RS232
RS232-Signale	TxD, RxD, RTS, CTS, DTR, DSR, RI, CD, GND
Potenzialtrennung	zum Rückwandbus
Übertragungsstrecke	max. 15m
Baudrate	max. 115200Baud
Programmierdaten	
Eingabedaten	16Byte
Ausgabedaten	16Byte
Parameterdaten	16Byte
Maße und Gewicht	
Abmessungen (BxHxT) in mm	25,4x76x78
Gewicht	80g

CP 240 RS485

Elektrische Daten	VIPA 240-1CA20
Anzahl der Kanäle	1
Spannungsversorgung	5V über Rückwandbus
Stromaufnahme	max. 150mA
Statusanzeige	über LED auf der Frontseite
Protokolle	ASCII, ASCII-fragmentiert, STEX/ETX, 3964(R), 3964(R) mit RK512, Modbus (Master, Slave)
Anschlüsse / Schnittstellen	9pol. SubD-Buchse für RS485
RS485-Signale	RxD/TxD-P, RxD/TxD-N, RTS, M5V, P5V
Potenzialtrennung	zum Rückwandbus
Übertragungsstrecke	max. 1200m
Baudrate	max. 115200Baud
Programmierdaten	
Eingabedaten	16Byte
Ausgabedaten	16Byte
Parameterdaten	16Byte
Maße und Gewicht	
Abmessungen (BxHxT) in mm	25,4x76x78
Gewicht	80g

Technische Daten
Protokolle

ASCII	
Telegrammlänge	max. 1024Byte
Baudrate	150, 300, 600, 1200, 1800, 2400, 4800, 7200, 9600, 14400, 19200, 38400, 57600, 76800, 115200
Zeichenverzugszeit ZVZ	0 ... 5,1s in 20ms-Schritten
Flusskontrolle	keine, Hardware, XON/XOFF
Anzahl pufferbarer Telegramme	max. 250
Endeerkennung eines Telegramms	nach Ablauf der Zeichenverzugszeit ZVZ
STX/ETX	
Telegrammlänge	max. 1024Byte
Baudrate	150, 300, 600, 1200, 1800, 2400, 4800, 7200, 9600, 14400, 19200, 38400, 57600, 76800, 115200
Zeichenverzugszeit TMO	0 ... 5,1s in 20ms-Schritten
Flusskontrolle	keine, Hardware, XON/XOFF
Anzahl pufferbarer Telegramme	max. 250
Endeerkennung eines Telegramms	nach Ablauf der Zeichenverzugszeit TMO
Anzahl Startzeichen	0 ... 2 (Zeichen parametrierbar)
Anzahl Endezeichen	0 ... 2 (Zeichen parametrierbar)
3964, 3964R	
Telegrammlänge	max. 1024Byte
Baudrate	150, 300, 600, 1200, 1800, 2400, 4800, 7200, 9600, 14400, 19200, 38400, 57600, 76800, 115200
Blockprüfzeichen	nur 3964R
Priorität	low/high
Zeichenverzugszeit ZVZ	0 ... 5,1s in 20ms-Schritten
Quittungsverzugszeit QVZ	0 ... 25,5s in 20ms-Schritten
Anzahl Aufbauversuche	0 ... 255
Anzahl Übertragungsversuche	1 ... 255
Rechnerkopplung RK512	
Telegrammlänge	max. 1024Byte
Baudrate	150, 300, 600, 1200, 1800, 2400, 4800, 7200, 9600, 14400, 19200, 38400, 57600, 76800, 115200
Zeichenverzugszeit ZVZ	0 ... 5,1s in 20ms-Schritten
Quittungsverzugszeit QVZ (Anwender)	0 ... 25,5s in 100ms-Schritten
Anzahl Aufbauversuche	0 ... 255
Anzahl Übertragungsversuche	1 ... 255
Modbus	
Telegrammlänge	max. 258Byte
Adressierbarer Bereich	je 1024Byte
Baudrate	2400, 4800, 7200, 9600, 14400, 19200, 38400
Modus	Master ASCII, Master RTU, Slave ASCII short, Slave RTU short, Slave ASCII long, Slave, RTU long
Adresse	1 ... 255
Wartezeit	automatisch, 1 ... 60000ms

Teil 5 CP 240 - EnOcean

Überblick

In diesem Kapitel finden Sie Informationen über den Aufbau und die Einbindung des Kommunikationsprozessors CP 240 mit EnOcean Transceiver-Modul.

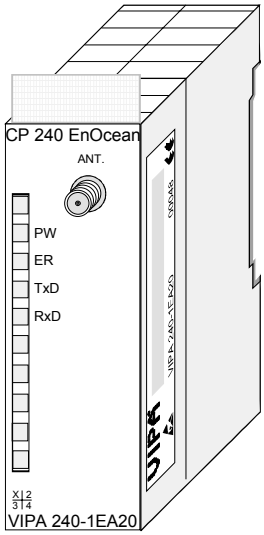
Nachfolgend sind beschrieben:

- Systemübersicht
- Aufbau
- Standardhantierungsbausteine
- Beispiel
- Übersicht der Telegramme
- Übernahme der ID im Ersatzfall
- Technische Daten

Inhalt

Systemübersicht

CP 240 EnOcean



Bestelldaten

Typ	Bestellnummer	Beschreibung
CP 240 EnOcean	VIPA 240-1EA20	CP mit EnOcean Funktransceiver Modul TCM 120
Portable Antenne	VIPA 240-0EA00	Portable Antenne mit SMA-Stecker
Magnetfuß Antenne	VIPA 240-0EA10	Magnetfußantenne mit 150cm Kabel und SMA-Stecker

Grundlagen

EnOcean

EnOcean ist ein batterieloses Funksystem, das im Jahre 2001 von der Firma EnOcean entwickelt wurde. Aufgrund der kurzen Signaldauer von 0,5ms und 10mW Sendeleistung hat die Funkübertragungstechnik einen Energiebedarf von 50µWs. Hierbei nutzt das System die Energie aus kleinsten Veränderungen von Druck oder Temperatur zur Stromversorgung der Sensoren.

Die Reichweite der Sensoren beträgt bis zu 300m im Freien. Jeder Sender erhält zudem bereits bei der Herstellung eine eindeutige 32Bit Adresse als ID. Die Module nutzen das international zugelassene SRD-Frequenzband bei 869 MHz.

Einsatzschwerpunkte von EnOcean sind Gebäudeautomation, industrielle Produktion und Automobiltechnik.

Eigenschaften

- Minimale Energieanforderungen
- Unterstützung mehrerer Sender in nächster Umgebung
- Telegrammdauer 0,5ms
- Übertragungsreichweite bis zu 300m
- Uni- und bidirektionale Kommunikation
- Einfache Erweiterbarkeit

Amplituden-modulation

Als Modulationsverfahren kommt bei EnOcean die inkohärente Amplituden-modulation (ASK) zum Einsatz. Ihre Fehlerwahrscheinlichkeit ist gegenüber der Frequenzmodulation bei gleichen Störsignalpegel in etwa gleichwertig. Die Digitale Amplitudenmodulation gestattet die Realisierung energie-sparender Sender, da hier nur die "1"-Bits übertragen werden.

Sicherheit durch Telegramm-wiederholung

Die Übertragung eines Datentelegramms dauert ca. 0,5ms. Zur Erhöhung der Datensicherheit wird jedes Telegramm innerhalb von 40ms zweimal wiederholt, wobei der zeitliche Abstand zwischen jeder Wiederholung zufällig gewählt wird.

Diese schnelle Mehrfachaussendung ermöglicht, dass viele benachbarte Sender parallel auf einer gemeinsamen Funkfrequenz mit niedriger Fehlerquote arbeiten können.

IDs zur Adressierung

EnOcean verwendet zur Adressierung IDs. Eine ID setzt sich zusammen aus einer *IDBase* und einem frei konfigurierbaren *Bitbereich*. Da die EnOcean-Module von VIPA mit einer unterschiedlichen IDBase ausgeliefert werden, empfiehlt es sich bei umfangreichen Projekten die *IDBase* aller Module zu notieren. Somit können Sie im Fehlerfall ein Modul ersetzen und die entsprechende *IDBase* übernehmen.

Näheres hierzu finden Sie unter "Modul ersetzen und IDBase übernehmen".

Schnelleinstieg

Übersicht

Der Kommunikationsprozessor CP 240 EnOcean ermöglicht die Prozessankopplung an verschiedene Ziel- oder Quell-Systeme auf Basis der drahtlosen EnOcean-Kommunikation.

Der CP 240 EnOcean wird über den Rückwandbus mit Spannung versorgt. Zur internen Kommunikation sind VIPA FCs zu verwenden. Für die Projektierung des CP 240 EnOcean in Verbindung mit einer CPU 21x im Siemens SIMATIC Manager, ist die Einbindung der GSD VIPA_21x.gsd erforderlich. Damit der CP 240 EnOcean mit der CPU kommunizieren kann, ist für das System immer eine Hardware-Konfiguration durchzuführen.

Eine allgemeine Beschreibung zur Projektierung des CP 240 finden Sie im Teil "Parametrierung".

Parameter

Durch Platzieren des CP 240 EnOcean in der Hardware-Konfiguration im "virtuellen" Profibus-System werden automatisch die erforderlichen Parameter angelegt. Der Parameterbereich hat folgenden Aufbau:

Byte	Funktion	Wertebereich	Defaultparameter
0	reserviert		
1	Protokoll	E0h: EnOcean	-
2...15	reserviert		

Hier ist lediglich im Byte 1 als Protokoll E0h für EnOcean anzugeben. Die restlichen Parameter sind reserviert und werden nicht ausgewertet.

Interne Kommunikation

Mit VIPA-FCs steuern Sie die Kommunikation zwischen CPU und CP 240. Hierbei steht für Sende- und Empfangsdaten je ein 2048Byte großer Puffer zur Verfügung, der maximal 150 Telegramme verwalten kann. In Verbindung mit einer CPU 21x kommen folgende Hantierungsbausteine zum Einsatz:

Name	FCs	Kurzbeschreibung
SEND	FC0	Sende-Baustein
RECEIVE	FC1	Receive-Baustein
SYNCHRON_RESET	FC9	Reset und Synchronisation des CP 240

11Byte Telegramm für EnOcean-Kommunikation

Verwenden Sie für die Kommunikation immer Telegramme mit einer Länge von 11Byte. Beim Senden werden im CP 240 EnOcean die 11Byte automatisch mit 2 Synchronisations-Bytes und einer Checksumme auf 14Byte ergänzt bzw. beim Empfang das 14Byte große Telegramm auf 11Byte beschnitten.

Installation

Aktuelle GSD-Datein und Bibliotheken finden Sie unter anderem auf der CD "SW-ToolDemo" bzw. unter ftp.vipa.de. Die Installation des CP 240 EnOcean erfolgt nach folgender Vorgehensweise:

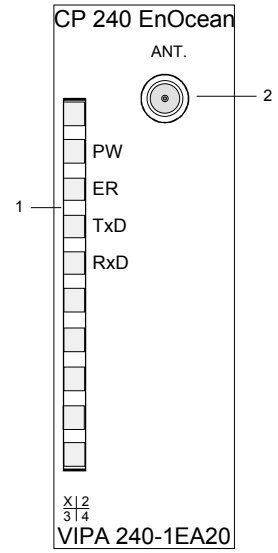
1. Installieren Sie die GSD-Datei **VIPA_21X.gsd** in Ihrem Projektiertool.
2. Installieren Sie die VIPA Library **Fx000002_Vxxx.zip** mit den Hantierungsbausteinen in Ihrem Projektiertool.
3. Projektieren Sie Ihr System 200V, parametrieren Sie den CP 240 EnOcean und programmieren Sie die Kommunikation.
4. Übertragen Sie Ihr Projekt in die CPU.

Aufbau

Eigenschaften

- Der Kommunikationsprozessor hat die Best.-Nr.: VIPA 240-1EA20
- 16Byte Parameterdaten
- Spannungsversorgung über Rückwandbus
- Das TCM 120 **T**rans**c**eiver-**M**odul arbeitet bei 868,3MHz.

Frontansicht 240-1EA20



[1] LED Statusanzeigen
[2] SMA-Antennenbuchse mit Außengewinde und Kelch

Komponenten

Spannungs-versorgung

Der Kommunikationsprozessor bezieht seine Versorgungsspannung über den Rückwandbus.

LEDs

Der Kommunikationsprozessor besitzt 4 LEDs zu Anzeige des Betriebszustands. Die Bedeutung und die jeweiligen Farben dieser LEDs finden Sie in der nachfolgenden Tabelle.

Bez.	Farbe	Bedeutung
PW	Grün	Signalisiert eine anliegende Betriebsspannung
ER	Rot	Signalisiert einen Fehler durch Pufferüberlauf
TxD	Grün	Daten senden (transmit data)
RxD	Grün	Daten empfangen (receive data)

Antennen

Im Lieferumfang ist keine Antenne enthalten. Sie können aber optional eine portable Anntenne oder eine Magnetfußantenne mit 150cm Kabel bestellen.

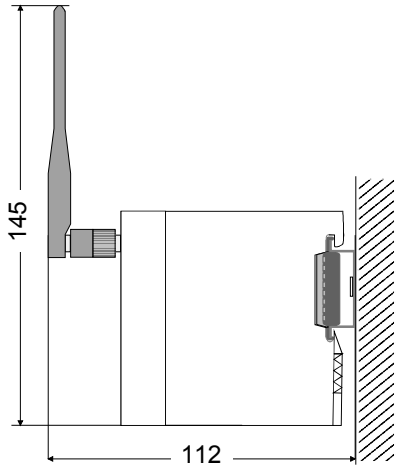
Beide Antennen sind mit einem SMA-Stecker ausgestattet. Der koaxial aufgebaute SMA-Stecker (**s**tright **m**edium **a**daptor) ist ein Miniatur-HF-Stecker mit Gewindeverschluss, der sich durch eine hohe HF-Dichtigkeit auszeichnet. In der Standardversion hat der Stecker eine Überwurfmutter mit Innengewinde und einem Stift.

Die SMA-Buchse, die sich am CP befindet, bildet mit dem Außengewinde und dem Kelch das Gegenstück für die Montage.

Portable Antenne

Bei der Portable Antenne handelt es sich um eine kurze Stabantenne, die über den SMA-Stecker ohne Kabel direkt am Modul montiert wird.

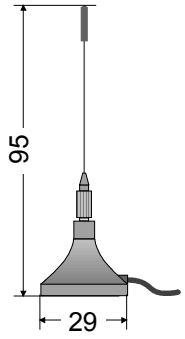
Die Antenne kann abgewinkelt und in alle Richtungen gedreht werden.



alle Maße in mm

Magnetfußantenne

Die Magnetfußantenne mit 150cm Kabel ist für den Einbau in Schaltschränke geeignet. Aufgrund des Magnetfußes können Sie die Antenne an allen Stahl-Flächen befestigen. Der Anschluss der Magnetfußantennen an den CP 240 EnOcean erfolgt über das 150cm lange Antennenkabel mit SMA-Stecker.



alle Maße in mm

Kommunikationsprinzip

Daten senden und empfangen

Zu sendende Daten werden von der CPU über den Rückwandbus in den entsprechenden Datenkanal geschrieben. Der Kommunikationsprozessor trägt diese in einem Ringpuffer (2048Byte) ein und gibt sie von dort über EnOcean aus.

Empfängt der Kommunikationsprozessor Daten über EnOcean, werden die Daten in einem Ringpuffer (2048Byte) abgelegt. Die empfangenen Daten können über den Datenkanal telegrammweise (11Byte) von der CPU gelesen werden.

Kommunikation über Rückwandbus

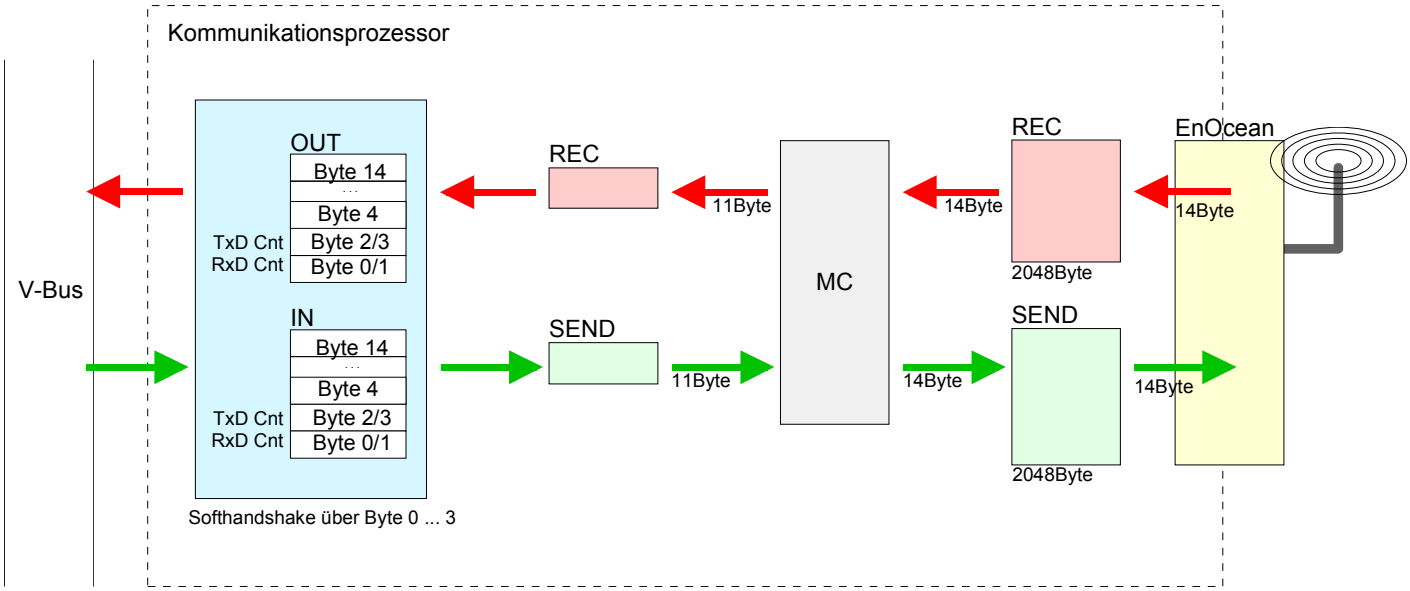
Der Austausch von empfangenen Telegrammen über den Rückwandbus erfolgt asynchron. Ist ein komplettes Telegramm über EnOcean eingetroffen, wird dies im Puffer abgelegt. Aus der Länge des Ringpuffers ergibt sich die maximale Anzahl der Telegramme. Ist der Puffer voll, werden neu ankommende Telegramme verworfen.

Aus den 14Byte großen Telegrammen werden telegrammweise 11Byte Nutzdaten über den Rückwandbus an die CPU übergeben. Die ersten beiden 2 Sync-Bytes und die Checksumme werden nicht weitergereicht.

Aufgaben der CPU

Ein zu sendendes Telegramm ist an den CP 240 zu übergeben. Dieser ergänzt das Telegramm mit den ersten beiden Sync-Bytes und der Checksumme und reicht das Telegramm an den Sendepuffer weiter. Im CP 240 werden diese Blöcke im Sendepuffer zusammengesetzt und bei Vollständigkeit des Telegramms über den EnOcean-Transceiver gesendet. Da der Datenaustausch über den Rückwandbus asynchron abläuft, wird ein "Software Handshake" zwischen dem CP 240 und der CPU eingesetzt. Die Register für den Datentransfer vom CP 240 sind 16Byte breit. Für den Handshake sind die Bytes 0 bis 3 (Wort 0 und 2) reserviert.

Folgende Abbildung soll dies veranschaulichen:



Software-handshake

Für den Einsatz des CP 240 in Verbindung mit einer System 200V CPU sind bei VIPA Hantierungsbausteine erhältlich, die den Softwarehandshake komfortabel übernehmen.

Bei Einsatz des CP 240 ohne Hantierungsbausteine soll hier die Funktionsweise anhand eines Beispiels für das Senden und Empfangen von Daten erläutert werden.

Beispiel Daten senden ohne Hantierungsbaustein

Ein EnOcean-Telegramm besitzt 11Byte Nutzdaten. Beim Senden werden von der CPU je Telegramm 11Byte Nutzdaten in die Bytes 4 bis 14 und in Byte 2/3 die Länge des Telegramms (also "11") geschrieben. Der CP 240 empfängt die Daten über den Rückwandbus. Zur Quittierung des Telegramms schreibt der CP 240 in Byte 2/3 den Wert "11" (Länge des Telegramms) an die CPU zurück.

Beim Empfang der "11" auf Byte 2/3 sendet die CPU eine "0" auf Byte 2/3. Daraufhin werden im CP 240 die Nutzdaten am Anfang mit 2 Sync-Bytes und am Ende mit der Checksum auf 14Byte ergänzt und im Sendepuffer abgelegt. Ist dies erfolgt, antwortet der CP mit einer "0" auf Byte 2/3. Beim Empfang der "0" kann die CPU ein neues Telegramm an den CP 240 senden.

Die im Sendepuffer abgelegten Telegramme werden sofort über EnOcean ausgegeben.

Beispiel Daten empfangen ohne Hantierungsbaustein

Jedes EnOcean-Telegramm hat eine Größe von 14Byte. Empfängt der CP 240 ein Telegramm, so wird dies im Empfangspuffer abgelegt. Von jedem Telegramm werden die 11Byte Nutzdaten in Byte 4 bis 14 und die Länge (also "11") in Byte 0/1 über den Rückwandbus an die CPU übergeben. Die ersten beiden Sync-Bytes und die Checksumme werden verworfen.

Die CPU speichert die Nutzdaten und antwortet mit dem Wert "11" auf Byte 0/1 Dies quittiert der CP mit einer "0" auf Byte 0/1 und meldet somit, dass der Transfer abgeschlossen ist. Sobald neue Daten übertragen werden können, antwortet die CPU mit "0"

Mit dem Empfang der "0" kann der CP 240 ein neues Telegramm an die CPU senden.

Beispiel zum Einsatz unter EnOcean

Übersicht

In dem nachfolgenden Beispiel wird eine EnOcean-Kommunikation (Senden und Empfangen) aufgebaut. Weiter soll das Beispiel zeigen, wie Sie unter Einsatz der Hantierungsbausteine auf einfache Weise die Kontrolle über die Kommunikationsvorgänge haben.

Bei Bedarf können Sie das Beispielprojekt von VIPA beziehen.

Voraussetzung

Folgende Komponenten sind für das Beispiel erforderlich:

1 System 200V bestehend aus CPU 21x und CP 240 EnOcean

1 Schalter mit EnOcean-Sender

Projektiertool SIMATIC Manager von Siemens mit Übertragungskabel

Vorgehensweise

Bauen Sie das System 200V auf.

Laden Sie das Beispielprojekt, passen Sie ggf. die Peripherieadresse an und übertragen Sie Ihr Projekt in die CPU.

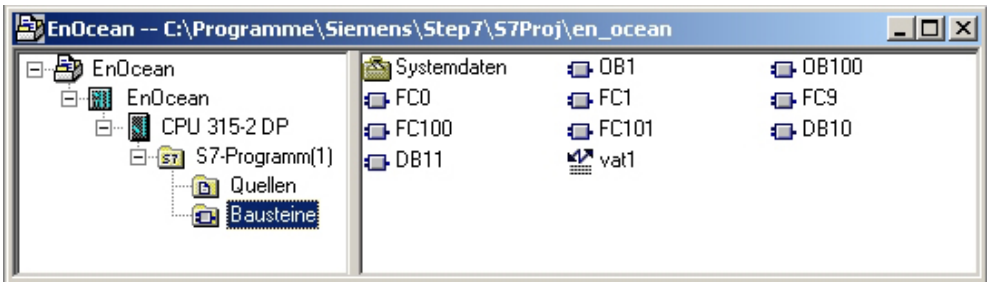
Projekt dearchivieren

Im Siemens SIMATIC Manager gehen Sie nach folgenden Schritten vor:

- Starten Sie den Siemens SIMATIC Manager.
- Zum Entpacken der Datei Enocean.zip gehen Sie auf **Datei** > *dearchivieren*.
- Wählen sie die Beispieldatei Enocean.zip aus und geben Sie als Zielverzeichnis "s7proj" an.
- Öffnen Sie das entpackte Projekt.

Projekt-Struktur

Das Projekt beinhaltet schon das SPS-Programm und die Hardware-Konfiguration und besitzt folgende Struktur:



Datenbausteine In diesem Beispiel werden folgende Datenbausteine verwendet:

DB10
Sendebaustein

Adr.	Name	Typ	Kommentar
0.0		STRUCT	
+0.0	Sendefach	STRUCT	
+0.0	RX_TX_Kennung	BYTE	0B=RX/6B=TX
+1.0	ORG	BYTE	
+2.0	Datenbyte3	BYTE	Datenbyte 3
+3.0	Datenbyte2	BYTE	Datenbyte 2
+4.0	Datenbyte1	BYTE	Datenbyte 1
+5.0	Datenbyte0	BYTE	Datenbyte 0
+6.0	IDbyte2_3	WORD	ID Byte 2 und 3
+8.0	IDbyte0_1	WORD	ID Byte 0 und 1
+10.0	Status	BYTE	Status
=12.0		END_STRUCT	
+12.0	Reserve	BYTE	
+13.0	SENDEN_LAEUFT	BOOL	Senden läuft noch
+13.1	LETZTER_BLOCK	BOOL	letzter Block wurde gesendet
+13.2	FEHL_KOM	BOOL	Fehler beim Senden aufgetreten
+13.3	ENDE_KOM	BOOL	Übertragung abgeschlossen
+14.0	PAFE	BYTE	Parametrierfehler-Byte des FC0
+15.0	Res00	BOOL	
+15.1	Res01	BOOL	
+15.2	Res02	BOOL	
+15.3	Res03	BOOL	
+15.4	Res04	BOOL	
+15.5	Res05	BOOL	
+15.6	Res06	BOOL	
+15.7	Senden_start	BOOL	Telegramm komplett gesendet
+16.0	GESE	WORD	schon gesendete Daten
+18.0	ANZ_INT	WORD	Anzahl gesendete Daten
+20.0	Reserve1	ARRAY[0..50]	
*1.0		BYTE	
=72.0		END_STRUCT	

DB11
Empfangsbaustein

Adr.	Name	Typ	Kommentar
0.0		STRUCT	
+0.0	Empfangsfach	STRUCT	
+0.0	RX_TX_Kennung	BYTE	0B=RX/6B=TX
+1.0	ORG	BYTE	
+2.0	Datenbyte3	BYTE	Datenbyte 3
+3.0	Datenbyte2	BYTE	Datenbyte 2
+4.0	Datenbyte1	BYTE	Datenbyte 1
+5.0	Datenbyte0	BYTE	Datenbyte 0
+6.0	IDbyte2_3	WORD	ID Byte 2 und 3
+8.0	IDbyte0_1	WORD	ID Byte 0 und 1
+10.0	Status	BYTE	Status
=12.0		END_STRUCT	
+12.0	Reserve	BYTE	
+13.0	EMP_LAEUFT	BOOL	Empfangen läuft noch
+13.1	LETZTER_BLOCK	BOOL	letzter Block wurde empfangen
+13.2	FEHL_EMPF	BOOL	Fehler beim Empfang aufgetreten
+14.0	PAFE	BYTE	Parametrierfehler-Byte des FC1
+15.0	Res00	BOOL	
+15.1	Res01	BOOL	
+15.2	Res02	BOOL	
+15.3	Res03	BOOL	
+15.4	Res04	BOOL	
+15.5	Res05	BOOL	
+15.6	Res06	BOOL	
+15.7	Empfang_fertig	BOOL	Telegramm komplett empfangen
+16.0	GEEM	WORD	schon empfangene Daten
+18.0	ANZ_INT	WORD	Anzahl empfangener Daten
+20.0	Reserve1	ARRAY[0..50]	
*1.0		BYTE	
=72.0		END_STRUCT	

SPS-Programm Das Beispiel beinhaltet schon das SPS-Programm und die Hardware-Konfiguration. Hierbei kommen folgende Bausteine zum Einsatz:

OB 1

```
CALL  FC 9                    //Neustart oder Reset
 ADR      :=256               //Adresse des Moduls
 TIMER_NR :=T2
 ANL      :=M3.0
 NULL     :=M3.1
 RESET    :=M3.2
 STEUERB_S:=MB4
 STEUERB_R:=MB6
U  M  3.0                     //Empfang kann erst nach Abarbeitung von FC 9
BEB                           //(SYNCHRON_RESET) gestartet werden
CALL FC 100                   //Aufruf des Empfangs-FC
```

OB 100

```
L  0
T  MB 1                       //Auftragsbit löschen
UN M  3.0                     //Neustart auslösen
S  M  3.0
```

FC 100

```
CALL  FC 1
 ADR  :=256                                    //Adresse des Moduls
 _DB  :="EMPFANG_en_ocean"                     //DB mit Empfangsdaten
 ABD  :=W#16#0                                 //1. DBB Empfangsdaten
 ANZ  :=W#16#B                                 //Empfangslänge immer 11
 EMFR :=M7.0                                   //alle Daten empfangen
 PAFE :="EMPFANG_en_ocean".Pafe                //Fehlerbyte
 GEEM :="EMPFANG_en_ocean".GEEM                //Empfangene Anzahl (intern)
 ANZ_INT:="EMPFANG_en_ocean".ANZ_INT           //Empfangslänge (intern)
 EMPF_LAEUFT:="EMPFANG_en_ocean".EMP_LAEUFT    //Datenempfang läuft (intern)
 LETZTER_BLOCK:="EMPFANG_en_ocean".LETZTER_BLOCK   //alle Daten empfangen
 FEHL_EMPF:="EMPFANG_en_ocean".FEHL_EMPF       //Fehler in der
                                               //Empfangsroutine
UN    M         7.0                           //kein Telegramm empfangen
BEB                                           //dann Ende
R     M         7.0                           //Empfangsbit löschen
L     "EMPFANG_en_ocean".Empfangsfach.IDbyte0_1//Schalterkennung
L     W#16#1C7A                               //Bitte hier Kennung Ihres
==I                                           //Schalters angeben.
SPB   e_a2                                    //Diese kann dem DB 11.DBW 8
BEA                                           //entnommen werden

e_a2: NOP   0
      L     5                                 //Kennung Schalter ein
      L     "EMPFANG_en_ocean".Empfangsfach.Datenbyte3    //Byte mit Kennung
      SRW   4                                 //Kennung im Low-Nippel
      ==I                                     //Prüfen ob Schalter gedrückt
      SPB   ein
      L     7                                 //Kennung Schalter aus
      ==I                                     //Prüfen ob Schalter gedrückt
      SPB   aus
      BEA

ein:  NOP   0
      S     A         0.0                     //Funktion ein
      BEA

aus:  NOP   0
      R     A         0.0                     //Funktion aus
      BEA
```

FC 101

```
L     B#16#6B                                          //Sendedaten vorbelegen
T     "SEND_en_ocean".Empfangsfach.RX_TX_Kennung       //Kennung senden
L     B#16#5                                           //ORG-Kennung
T     "SEND_en_ocean".Empfangsfach.ORG
L     B#16#2
T     "SEND_en_ocean".Empfangsfach.Datenbyte3
L     0
T     "SEND_en_ocean".Empfangsfach.Datenbyte2
T     "SEND_en_ocean".Empfangsfach.Datenbyte1
T     "SEND_en_ocean".Empfangsfach.Datenbyte0
T     "SEND_en_ocean".Empfangsfach.IDbyte2_3
L     W#16#3267                                        //Nur die letzten 7Bit
T     "SEND_en_ocean".Empfangsfach.IDbyte0_1           //sind für Adr. relevant
L     6                                                //und werden im CP 240
T     "SEND_en_ocean".Empfangsfach.Status              //mit der dort abgelegten
                                                       //IDBase verodert

CALL  FC 0
 ADR           :=256
 _DB           :="SEND_en_ocean"
 ABD           :=W#16#0                          //ab Datenbyte 0 senden
 ANZ           :=W#16#B                          //immer 11 Byte
 PAFE          :="SEND_en_ocean".Pafe
 FRG           :="SEND_en_ocean".Senden_start
 GESE          :="SEND_en_ocean".GEEM
 ANZ_INT       :="SEND_en_ocean".ANZ_INT
 ENDE_KOM      :="SEND_en_ocean".ENDE_KOM
 LETZTER_BLOCK:="SEND_en_ocean".LETZTER_BLOCK
 SENDEN_LAEUFT:="SEND_en_ocean".SENDEN_LAEUFT
 FEHLER_KOM    :="SEND_en_ocean".FEHL_KOM
```

Übersicht der EnOcean-Telegramme

Allgemeiner Aufbau

Die nachfolgende Tabelle zeigt den allgemeinen Aufbau eines EnOcean-Telegramms. Sende- und Empfangstelegramme besitzen die gleiche Struktur. Sie unterscheiden sich ausschließlich in der Kennung.

Bit 7 Bit 0

0xA5	Diese Bytes werden beim Senden automatisch generiert und beim Empfang ausgeblendet.
0x5A	
0x0B 0x6B	0x0B: Kennung für Empfangstelegramm 0x06: Kennung für Sendetelegramm
ORG	Siehe Tabelle *unterstützte ORG-Formate*
DataBytes3	Daten von einem Sensor bzw. an einen Aktor
DataBytes2	
DataBytes1	
DataBytes0	
IDBytes3*	ID des Transceiver-Moduls. Mit SET_IDBASE können Sie die ID bis zu 10 Mal ändern
IDBytes2*	
IDBytes1*	
IDBytes0*	
Status	Statusinformation des entsprechenden Sensors
Checksum	Wird beim Senden automatisch generiert und beim Empfang ausgeblendet.

*) Beim Senden wird die ID-Base im Telegramm durch die tatsächliche ID-Base des Moduls ersetzt.

Allgemein

Auf den Folgeseiten sind alle Telegramme aufgelistet, die vom CP 240 EnOcean unterstützt werden. Diese Beschreibung wurde mit freundlicher Genehmigung der Firma EnOcean in englischer Sprache direkt aus der Dokumentation übernommen.

Hinweis!

Bitte beachten Sie, dass im CP 240 bei empfangenen Telegrammen die ersten beiden Synchronisations-Bytes und die Checksumme nicht abgelegt werden. Beim Senden werden die 11Byte Nutzdaten automatisch mit diesen Bytes auf 14Byte ergänzt.

Description of ORG field

The TX_TELEGRAM and RX_TELEGRAM telegrams have the same structure. The only difference is that a TX_TELEGRAM is identified by “3” in H_SEQ instead of “0” for an RX_TELEGRAM.

ORG	Description	RRT / TRT Acronym
0x05	Telegram from a PTM switch module received (original or repeated message)	RPS
0x06	1 byte data telegram from a STM sensor module received (original or repeated message)	1BS
0x07	4 byte data telegram from a STM sensor module received (original or repeated message)	4BS
0x08	Telegram from a CTM module received (original or repeated message)	HRC
0x0A	6byte Modem Telegram (original or repeated)	6DT
0x0B	Modem Acknowledge Telegram	MDA

Serial command encoding for RPS, 1BS, 4BS, HRC

Bit 7 Bit 0

0xA5
0x5A
0x0B (RX_TELEGRAM) 0x6B(TX_TELEGRAM)
ORG
DataBytes3
DataBytes2
DataBytes1
DataBytes0
IDBytes3
IDBytes2
IDBytes1
IDBytes0
Status
ChkSum

DataBytes2= DataBytes1= DataBytes0= 0x00 for RPS,1BS, HRC

Serial command encoding for 6DT

Bit 7 Bit 0

0xA5
0x5A
0x0B (RX_TELEGRAM) 0x6B(TX_TELEGRAM)
0x0A
DataBytes5
DataBytes4
DataBytes3
DataBytes2
DataBytes1
DataBytes0
Address1
Address0
Status
ChkSum

Serial command encoding for MDA

Bit 7 Bit 0

0xA5
0x5A
0x0B (RX_TELEGRAM) 0x6B(TX_TELEGRAM)
0x0B
0xXX
0xXX
0xXX
0xXX
Address1
Address0
0xXX
0xXX
Status
ChkSum

Description of STATUS field

If ORG = 0x05 (Telegram from a PTM switch module)

7			0
Reserved	**T21**	**NU**	**RP_COUNTER**

Reserved (2 bit) Do not care

T21 (1 bit) T21=0 → PTM type 1, T21=1 → PTM type 2

Note: In transmission the TCM 120 always sets T21=1
→ it is only possible to transmit PTM type 2 telegrams!

NU (1 bit) NU=1 → N-message, NU=0 → U-message.

RP_COUNTER (4 bit) =0..15 Repeater level: 0 is original message

IMPORTANT NOTE

Within toggle switch applications using the RCM 120 or TCM 120 serial receiver mode in combination with the TCM 110 repeater module, please ensure that no serial command interpretation error may occur at the connected control unit. A toggle signal means that the same telegram (from e.g. PTM 100, PTM 200 or STM 100) is sent for switching something on and off. If e.g. the light is switched on by means of a RCM 120 receiving the I-button telegram from a PTM 100, the repeated telegram (delay <100ms) may switch off the light again. It is therefore mandatory to interpret the RP_COUNTER field as described in the RCM 120 User Manual. If a repeated telegram (RP_COUNTER>0) is received it has to be verified if the same telegram with a lower RP_COUNTER state has already been received in the previous 100 ms. In this case the repeated message has to be discarded.

PTM Type 1

PTM switch modules of Type 1 (e.g. PTM 100) do not support interpretation of operating more than one rocker at the same time:

N-message received → Only one pushbutton was pressed.

U-message received → No pushbutton was pressed when activating the energy generator, or more than one pushbutton was pressed.

PTM Type 2

PTM switch modules of Type 2 allow interpretation of operating two buttons simultaneously:

N-message received → Only one or two pushbuttons have been pressed.

U-message received → No pushbutton was pressed when activating the energy generator, or more than two pushbuttons have been pressed.

Note for telegrams from PTM 100 piezo transmitters:

Due to the mechanical hysteresis of the piezo energy bow, in most rocker switch device implementations, pressing the rocker sends an N-message and releasing the rocker sends a U-message!

If ORG = 0x06, 0x07, 0x08 or 0x0A:

7	0
Reserved	**RP_COUNTER**

Reserved (4 bit) Do not care

RP_COUNTER (4 bit) Repeater level: 0 original message
1 repeated message

Description of DATA_BYTE 3..0

If ORG = 0x05 and NU = 1 (N-message from a PTM switch module):

DATA_BYTE2..0 always = 0
DATA_BYTE3 as follows:

7					0
RID	**UD**	**PR**	**SRID**	**SUD**	**SA**

RID	(2 bit)	Rocker ID, from left (A) to right (D): 0, 1, 2 and 3 (decimal)
UD	(1 bit)	UD=1 → O-button, UD=0 → I-button
PR	(1 bit)	PR=1 → energy bow pressed PR=0 → energy bow released
SRID	(2 bit)	Second Rocker ID, from left to right: 0, 1, 2 and 3
SUD	(1 bit)	(Second) SUD=1 → O-button, SUD=0 → I-button
SA	(1 bit)	SA=1 → Second action (2 buttons pressed simultaneously), SA=0 → No second action

If ORG = 0x05 and NU = 0 (U-message from a PTM switch module):

DATA_BYTE2..0 always = 0
DATA_BYTE3 as follows:

7		0
BUTTONS	**PR**	**Reserved**

BUTTONS	(3 bit)	Number of simultaneously pressed buttons, as follows: PTM 100 / PTM200 0 = 0 Buttons / 0 = 0 Button 1 = 2 Buttons / 1 = not possible 2 = 3 Buttons / 2 = not possible 3 = 4 Buttons / 3 = 3 or 4 buttons 4 = 5 Buttons / 4 = not possible 5 = 6 Buttons / 5 = not possible 6 = 7 Buttons / 6 = not possible 7 = 8 Buttons / 7 = not possible
PR	(1 bit)	PR = 1 → energy bow pressed PR = 0 → energy bow released
Reserved	(4 bit)	for future use

If ORG = 0x06 (Telegram from a 1 Byte STM sensor):

DATA_BYTE2..0 always = 0
DATA_BYTE3 Sensor data byte.

If ORG = 0x07 (Telegram from a 4 Byte STM sensor):

DATA_BYTE3 Value of third sensor analog input
DATA_BYTE2 Value of second sensor analog input
DATA_BYTE1 Value of first sensor analog input
DATA_BYTE0 Sensor digital inputs as follows:

7				0
Reserved	**DI_3**	**DI_2**	**DI_1**	**DI_0**

If ORG = 0x08 (Telegram from a CTM module set into HRC operation):

DATA_BYTE2..0 always = 0
DATA_BYTE3 as follows:

7				0
RID	**UD**	**PR**	**SR**	**Reserved**

RID	(2 bit)	Rocker ID, from left (A) to right (D): 0, 1, 2 and 3
UD	(1 bit)	UD=1 → O-button, UD=0 → I-button
PR	(1 bit)	PR=1 → Button pushed, PR=0 → Button released
SR	(1 bit)	SR=1 → Store, SR=0 → Recall (see note)
Reserved	(3 bit)	for future use

Note: The SR bit is used only when the lower 3 bits from ID_BYTE0 = B'111' (scene switch), and RID ≠ 0 (indicates that the memory buttons M0-M6 are operated in the handheld remote control).

If ORG = 0x0A (Modem telegram):

Please note the different structure of modem telegrams with 6 data bytes and 2 address bytes for the ID of the receiving modem. See A.1.1.

Command Telegrams and Messages

INF_INIT

After a power-on, a hardware reset or a RESET command the TCM informs the user through several of these telegrams about the current status. The messages have the general syntax as shown. The information contained by the bytes marked as X should be decoded according to ASCII code.

Bit 7	*Bit 0*
0xA5	
0x5A	
0x8B	
0x89	
X	
X	
X	
X	
X	
X	
X	
X	
X	
ChkSum	

In total there are 15 telegrams:

0xA5 0x5A 0x8B 0x89	“ ”
0xA5 0x5A 0x8B 0x89	“EnOcean”
0xA5 0x5A 0x8B 0x89	“TCM120”
0xA5 0x5A 0x8B 0x89	“Version”
0xA5 0x5A 0x8B 0x89	Version number in ASCII
0xA5 0x5A 0x8B 0x89	“Bdrate”
0xA5 0x5A 0x8B 0x89	*“0x40” (9600 baud)*
0xA5 0x5A 0x8B 0x89	“Modem”
0xA5 0x5A 0x8B 0x89	“ON” or “OFF”
0xA5 0x5A 0x8B 0x89	“RxID”
0xA5 0x5A 0x8B 0x89	modem ID in ASCII
0xA5 0x5A 0x8B 0x89	“Mode”
0xA5 0x5A 0x8B 0x89	“Run”
0xA5 0x5A 0x8B 0x89	“PrgMem”
0xA5 0x5A 0x8B 0x89	“OK” or “CORRUPT”

OK

Standard message used to confirm that an action was performed correctly by the TCM.

Bit 7	*Bit 0*
0xA5	
0x5A	
0x8B	
0x58	
X	
X	
X	
X	
X	
X	
X	
X	
X	
ChkSum	

ERR

Standard error message response if after a TCT command the operation could not be carried out successfully by the TCM.

Bit 7	Bit 0
0xA5	
0x5A	
0x8B	
0x19	
X	
X	
X	
X	
X	
X	
X	
X	
X	
ChkSum	

RD_IDBASE

When this command is sent to the TCM, the base ID range number is retrieved though an INF_IDBASE telegram.

Bit 7	Bit 0
0xA5	
0x5A	
0xAB	
0x58	
X	
X	
X	
X	
X	
X	
X	
X	
X	
ChkSum	

SET_IDBASE

With this command the user can rewrite its ID range base number. The most significant ID byte is IDBaseByte3**.** The information of the 25 most significant bits is stored in EEPROM.

The allowed ID range is from 0xFF800000 to 0xFFFFFFFF.

Bit 7 Bit 0

0xA5
0x5A
0xAB
0x18
IDBaseByte3
IDBaseByte2
IDBaseByte1
IDBaseByte0
X
X
X
X
X
ChkSum

32 **0**

25 most significant bits	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID range base

This command can only be used a maximum number of 10 times. After successfully ID range reprogramming, the TCM answers with an OK telegram. If reprogramming was not successful, the TCM answers sending an ERR telegram if the maximum number of 10 times is exceeded or an ERR_IDRANGE telegram if the ID range base is not within the allowed range.

INF_IDBASE

This message informs the user about the ID range base number.

Bit 7 Bit 0

0xA5	
0x5A	
0x8B	
0x98	
IDBaseByte3	IDBaseByte3 is the most significant byte.
IDBaseByte2	
IDBaseByte1	
IDBaseByte0	
X	
X	
X	
X	
X	
ChkSum	

SET_RX_SENSITIVITY

This command is used to set the TCM radio sensitivity.

In LOW radio sensitivity, signals from remote transmitters are not detected by the TCM receiver. This feature is useful when only information from transmitters in the vicinity should be processed. An OK confirmation telegram is generated after TCM sensitivity has been changed.

Bit 7 Bit 0

0xA5
0x5A
0xAB
0x08
Sensitivity
X
X
X
X
X
X
X
X
ChkSum

Sensitivity=0x00 Low sensitivity
Sensitivity=0x01 High sensitivity

RD_RX_SENSITIVITY

This command is sent to the TCM to retrieve the current radio sensitivity mode (HIGH or LOW). This information is sent via a INF_RX_ SENSITIVITY command.

Bit 7 Bit 0

0xA5
0x5A
0xAB
0x48
X
X
X
X
X
X
X
X
X
ChkSum

INF_RX_SENSITIVITY

This message informs the user about the current TCM radio sensitivity.

Bit 7 Bit 0

0xA5
0x5A
0x8B
0x88
Sensitivity
X
X
X
X
X
X
X
X
ChkSum

Sensitivity= 0x00 Low sensitivity
Sensitivity= 0x01 High sensitivity

SLEEP

If the TCM receives the SLEEP command, it works in an energy-saving mode. The TCM will not wake up before a hardware reset is made or a WAKE telegram is sent via the serial interface.

Bit 7	Bit 0
0xA5	
0x5A	
0xAB	
0x09	
X	
X	
X	
X	
X	
X	
X	
X	
X	
ChkSum	

WAKE

If the TCM receives the WAKE command, it wakes up from sleep mode. In contrast to all other telegrams this telegram is only one byte long.

Bit 7	Bit 0
0xAA	

RESET

Performs a reset of the TCM micro controller. When the TCM is ready to operate again, it sends an ASCII message (INF_INIT) containing the current settings.

Bit 7	*Bit 0*
0xA5	
0x5A	
0xAB	
0x0A	
X	
X	
X	
X	
X	
X	
X	
X	
X	
ChkSum	

MODEM_ON

Activates TCM modem functionality and sets the modem ID. An OK confirmation telegram is generated. The modem ID is the ID at which the TCM receives messages of type 6DT. The modem ID and modem status (ON/OFF) is stored in EEPROM. The modem ID range is from 0x0001 to 0xFFFF. IF 0x0000 is provided as modem ID, the modem is activated with the ID previously stored in EEPROM.

Bit 7	Bit 0
0xA5	
0x5A	
0xAB	
0x28	
Modem ID (MSB)	
Modem ID (LSB)	
X	
X	
X	
X	
X	
X	
X	
ChkSum	

MODEM_OFF

Deactivates TCM modem functionality. When this command has been sent, an OK command should be received, confirming that the modem status is OFF. The modem ID is not erased.

Bit 7	Bit 0
0xA5	
0x5A	
0xAB	
0x2A	
X	
X	
X	
X	
X	
X	
X	
X	
X	
ChkSum	

RD_MODEM_ STATUS

This command requests the TCM to send information about its current modem current status. The requested information is reported to the user through an INF_MODEM_STATUS telegram.

Bit 7	Bit 0
0xA5	
0x5A	
0xAB	
0x68	
X	
X	
X	
X	
X	
X	
X	
X	
X	
ChkSum	

INF_MODEM_ STATUS

Informs the user about the TCM current modem status. The information provided is the following: Modem status (ON or OFF) and modem ID stored.

Modem state=0x01, modem ON
Modem state=0x00, modem OFF

Modem ID MSB= most significant modem ID byte.
Modem ID LSB=least significant modem ID byte.

Bit 7 Bit 0

0xA5
0x5A
0x8B
0xA8
Modem status
Modem ID MSB
Modem ID LSB
X
X
X
X
X
X
ChkSum

RD_SW_VER

This command requests the TCM to send its current software version number. This information is provided via an INF_SW_VER telegram by the TCM.

Bit 7 Bit 0

0xA5
0x5A
0xAB
0x4B
X
X
X
X
X
X
X
X
X
ChkSum

INF_SW_VER Informs the user about the current software version of the TCM.

Bit 7	Bit 0
0xA5	
0x5A	
0x8B	
0x8C	
TCM SW Version Pos.1	
TCM SW Version Pos.2	
TCM SW Version Pos.3	
TCM SW Version Pos.4	
X	
X	
X	
X	
X	
ChkSum	

Example: Version 1.0.1.16
TCM SW Version Pos.1 = 1
TCM SW Version Pos.2 = 0
TCM SW Version Pos.3 = 1
TCM SW Version Pos.4 =16

ERR_MODEM_NOTWANTEDACK When a 6DT modem telegram has been sent, the TCM waits for a modem acknowledge (MDA) telegram. This error message is generated if an MDA with the right modem ID is received after the timeout (100ms) or if there is more than one MDA received.

Bit 7	Bit 0
0xA5	
0x5A	
0x8B	
0x28	
X	
X	
X	
X	
X	
X	
X	
X	
X	
ChkSum	

ERR_MODEM_NOTACK When a 6DT modem telegram has been sent, the TCM waits for a modem acknowledge (MDA) telegram. This error message is generated if no acknowledge was received before the timeout (100ms).

Bit 7	Bit 0
0xA5	
0x5A	
0x8B	
0x29	
X	
X	
X	
X	
X	
X	
X	
X	
X	
ChkSum	

ERR_MODEM_ DUP_ID

When the TCM receives an original (not repeated) MDA telegram with the same modem ID as its own, it sends this message through the serial port and informs that at least 2 TCMs have the same modem ID. This is not necessarily a problem and may even be intended. On the other hand it may also indicate that there is another installation/building in the vicinity where the same modem ID is in use.

Bit 7	*Bit 0*
0xA5	
0x5A	
0x8B	
0x0C	
X	
X	
X	
X	
X	
X	
X	
X	
X	
ChkSum	

ERR_SYNTAX

This telegram is sent automatically through the serial port after the TCM has detected a syntax error in a TCT telegram.

Errors can occur in the H_SEQ, LENGTH, ORG or CHKSUM fields/bytes.

Bit 7	Bit 0
0xA5	
0x5A	
0x8B	
Field	
X	
X	
X	
X	
X	
X	
X	
X	
X	
ChkSum	

Field code:
H_SEQ=0x08 ORG=0x0B
LENGTH=0x09 CHKSUM=0x0A

ERR_TX_ IDRANGE

When a radio telegram intended to be sent has an ID number outside the ID range, this error message is generated. The radio telegram is not delivered.

Bit 7 Bit 0

0xA5
0x5A
0x8B
0x22
X
X
X
X
X
X
X
X
X
ChkSum

ERR_ IDRANGE

This message is generated when the user tries to change the ID range base using the SET_IDBASE command to a value outside the allowed range from 0xFF800000 to 0xFFFFFFFF.

Bit 7 Bit 0

0xA5
0x5A
0x8B
0x1A
X
X
X
X
X
X
X
X
X
ChkSum

Modul ersetzen und IDBase übernehmen

Übersicht

Da die IDBase jedes Moduls unterschiedlich ist, haben Sie die Möglichkeit im Ersatzfall bis zu 10 Mal die IDBase eines Moduls mit einem SET_IDBASE-Telegramm zu ändern. Somit entfällt das erneute Abstimmen der Aktoren auf das Ersatz-Modul. Nach erfolgreicher Übertragung der IDBase ist entweder die CPU neu zu starten oder Reset über FC 9 durchzuführen.

Bitte beachten Sie, dass nur die oberen 25 Bits als IDBase übernommen werden. Die restlichen 7 Bits können Sie über Ihr Anwenderprogramm zur Laufzeit angeben und hiermit mehrere Aktoren adressieren.

IDBase ermitteln

Mit RD_IDBASE können Sie die aktuelle IDBase Ihres Moduls abfragen.

RD_IDBASE

0xAB	Kennung für Sendetelegramm
0x58	ORG-Kennung für RD_IDBASE
X	Irrelevant
...	...
X	Irrelevant

INF_IDBASE

RD_IDBASE liefert die aktuelle IDBase des Moduls in Form eines INF_IDBASE-Telegramms zurück Das Telegramm hat folgenden Aufbau:

0x8B	Kennung für Empfangstelegramm
0x98	ORG-Kennung für INF_IDBASE
IDBaseByte3	Byte 3 aktuelle IDBase
IDBaseByte2	Byte 2 aktuelle IDBase
IDBaseBYte1	Byte 1 aktuelle IDBase
IDBaseByte0	Byte 0 aktuelle IDBase (Bit 6...0 irrelevant)
X	irrelevant
...	...
X	irrelevant

SET_IDBASE

Im Ersatzfall senden Sie ein SET_IDBASE-Telegramm nach folgender Struktur von Ihrer CPU an das Modul (Transceiver). Verwenden Sie als neue IDBase die Adresse des zu ersetzenden Moduls:

0xAB	Kennung für Sendetelegramm
0x18	ORG-Kennung für SET_IDBASE
IDBaseByte3	Byte 3 neue IDBase
IDBaseByte2	Byte 2 neue IDBase
IDBaseByte1	Byte 1 neue IDBase
IDBaseByte0	Byte 0 neue IDBase (Bit 6...0 irrelevant)
X	irrelevant
...	...
X	irrelevant

Mögliche Antwort-Telegramme

0x8B **0x58**	←	**OK** - IDBase wurde übernommen

Zur Übernahme der IDBase zur Laufzeit, ist Reset über FC 9 durchzuführen. Ansonsten steht ihnen nach einem CPU-Neustart die neue IDBase zur Verfügung.

Im Fehlerfall erhalten Sie eine dieser Meldungen. Hierbei bleibt die alte IDBase erhalten.

0x8B **0x19**	←	**ERR** Mehr als 10 IDBase-Wechsel sind nicht zulässig
0x8B **0x1A**	←	**ERR_IDRANGE** IDBase liegt außerhalb des zulässigen Bereichs (0xFF800000 ... 0xFFFFFFFF).

Überprüfen Sie Ihre ID-Angaben und senden Sie das Telegramm erneut.

Technische Daten

CP 240 EnOcean

Elektrische Daten	VIPA 240-1EA20
Anzahl der Kanäle	1
Spannungsversorgung	5V über Rückwandbus
Stromaufnahme	max. 120mA
Externe Spannungsversorgung	-
Statusanzeige (LEDs)	über LED auf der Frontseite
Funk Transceiver Modul	
Typ	EnOcean TCM120
Frequenz / Modulationsart / Sendeleistung	868,3MHz / ASK / max.10mW
Datenrate (Sender)/Kanalbreite (Empfänger)	120kBaud/280kHz
Reichweite	300m Freifeld
Antenne	externe 50Ω-Antenne montierbar
Art	portable oder mit 150cm-Kabel und Magnetfuß
Anschluss	SMA-Buchse
Bidirektionale serielle Schnittstelle	full-duplex, asynchron, 9,6MBaud
Programmierdaten	
Eingabedaten	16Byte
Ausgabedaten	16Byte
Parameterdaten	16Byte
Diagnosedaten	-
Maße und Gewicht	
Abmessungen (BxHxT) in mm	25,4x76x78
Gewicht	80g

Teil 6 CP 240 - M-Bus

Übersicht

In diesem Kapitel finden Sie Informationen über den Aufbau und die Einbindung des CP 240 M-Bus zur Kommunikation mit Energie- und Verbrauchszählern.

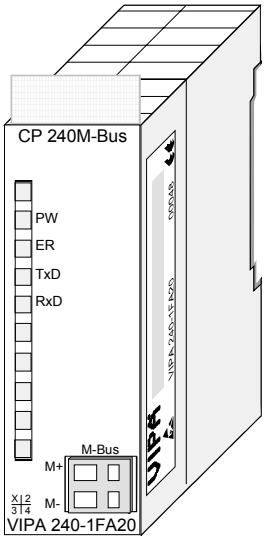
Nachfolgend sind beschrieben:

- Systemübersicht
- M-Bus-Grundlagen
- Aufbau
- Telegramme und Projektierung
- Technische Daten

Inhalt

Systemübersicht

CP 240 M-Bus



Bestelldaten

Typ	Bestellnummer	Beschreibung
CP 240 M-Bus	VPA 240-1FA20	CP mit M-Bus-Schnittstelle

Grundlagen

M-Bus

Das M-Bus-System (**M**etering **Bus**) ist ein europäisch genormter (DIN EN 1434-3) Zweidraht-Feldbus für die Verbrauchsdatenerfassung. Hierbei erfolgt die Datenübertragung seriell über eine verpolungssichere Zweidrahtleitung von Slave-Systemen (Messgeräte) zu einem Master-System.

Der M-Bus wurde von Prof. Dr. Horst Ziegler (Uni Paderborn) in Zusammenarbeit mit den Firmen Techem und Texas Instruments in Deutschland entwickelt.

Die größten Vorteile der M-Bus-Technik liegen in deren hoher Flexibilität. Durch Standardisierung lassen sich problemlos Geräte verschiedener Hersteller an einem Bus betreiben. Auch die Anbindung von Impulsgeberzählern ist über spezielle M-Bus-Adapter möglich. Bis zu 250 Zähler können maximal an einem Bus betrieben werden.

Eigenschaften

- Verpolungssicheres genormtes Bussystem nach DIN EN 1434-3
- Strom-, Gas-, Wasser- und Wärmezähler integrierbar
- Daten werden elektronisch ausgelesen
- Anschluss von bis zu 250 Zählern an einem Bus
- Zähler sind über eindeutige Adresse einzeln ansprechbar
- Fernablesung in dichten Ableseintervallen möglich
- Gewinnung statistischer Daten, als Basis zur Netzoptimierung
- Keine besonderen Anforderungen an Buskabel oder Verdrahtungstopologien
- Reichweite bis zu einigen km

Übertragungsprinzip

Die Datenübertragung vom Master zum Slave erfolgt über die Modulation der Versorgungsspannung. Hierbei ergeben 36V den Zustand "1" und 24V den Zustand "0".

Das Slave-System antwortet dem Master über die Modulation (Erhöhung) seines Stromverbrauchs. Hierbei ergeben 1,5mA den Zustand "1" und 11-20mA den Zustand "0".

Durch die Spannungsmodulation und die dadurch vorhandene M-Bus Spannung von 24V ist es möglich, die Endgeräte mit der erforderlichen Betriebsspannung zu versorgen.

Schnelleinstieg

Übersicht

Der Kommunikationsprozessor CP 240 M-Bus ermöglicht die Prozessankopplung an verschiedene Ziel- oder Quell-Systeme auf Basis der M-Bus-Kommunikation.

Der CP 240 M-Bus wird über den Rückwandbus mit Spannung versorgt. Zur internen Kommunikation sind VIPA FCs zu verwenden. Für die Projektierung des CP 240 M-Bus in Verbindung mit einer CPU 21x im Siemens SIMATIC Manager, ist die Einbindung der GSD VIPA_21x.gsd erforderlich. Damit der CP 240 M-Bus mit der CPU kommunizieren kann, ist für das System immer eine Hardware-Konfiguration durchzuführen.

Eine allgemeine Beschreibung zur Projektierung des CP 240 finden Sie im Teil "Projektierung".

Parameter

Durch Platzieren des CP 240 M-Bus in der Hardware-Konfiguration im "virtuellen" Profibus-System werden automatisch die erforderlichen Parameter angelegt. Der Parameterbereich hat folgenden Aufbau:

Byte	Funktion	Wertebereich	Defaultparameter
0	reserviert		
1	Protokoll	F0h: M-Bus	-
2...15	reserviert		

Hier ist lediglich im Byte 1 als Protokoll F0h für M-Bus anzugeben. Die restlichen Parameter sind reserviert und werden nicht ausgewertet.

Interne Kommunikation

Mit VIPA-FCs steuern Sie die Kommunikation zwischen CPU und CP 240 M-Bus. Hierbei steht für Sende- und Empfangsdaten je ein 2048Byte großer Puffer zur Verfügung. In Verbindung mit einer CPU 21x kommen folgende Hantierungsbausteine zum Einsatz:

Name	FCs	Kurzbeschreibung
SEND	FC0	Sende-Baustein
RECEIVE	FC1	Receive-Baustein
SYNCHRON_RESET	FC9	Reset und Synchronisation des CP 240

Telegrammaufbau

Für M-Bus-Telegramme, die von der CPU an den CP 240 geschickt werden, ist jedem Telegramm ein Byte voranzustellen, das die Baudrate beinhaltet. Mittels dieser Technik können Sie zur Laufzeit über unterschiedliche Baudraten mit verschiedenen Busteilnehmern kommunizieren.

Im Gegenzug erhalten Sie über dieses Byte vom CP 240 den Status des M-Bus-Datentransfers zurückgeliefert (0: OK - gültige Antwort, ≠0:Fehler).

Installation

Aktuelle GSD-Datein und Bibliotheken finden Sie unter anderem auf der CD "SW-ToolDemo" bzw. unter ftp.vipa.de. Die Installation des CP 240 M-Bus erfolgt nach folgender Vorgehensweise:

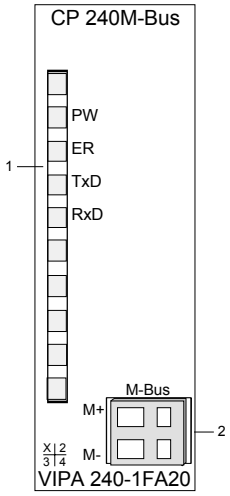
1. Installieren Sie die GSD-Datei **VIPA_21X.gsd** in Ihrem Projektiertool.
2. Installieren Sie die VIPA Library **Fx000002_Vxxx.zip** mit den Hantierungsbausteinen in Ihrem Projektiertool.
3. Projektieren Sie Ihr System 200V, parametrieren Sie den CP 240 M-Bus und programmieren Sie die Kommunikation.
4. Übertragen Sie Ihr Projekt in die CPU.

Aufbau

Eigenschaften

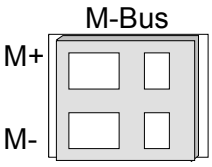
- Der Kommunikationsprozessor hat die Best.-Nr.: VIPA 240-1FA20
- Spannungsversorgung über Rückwandbus
- Bis zu 6 Slaves können angebunden werden.
- genormtes Bussystem nach DIN EN 1434-3

Frontansicht 240-1FA20



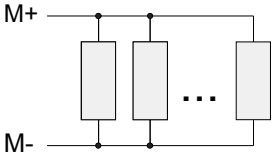
[1] LED Statusanzeigen
[2] M-Bus-Schnittstelle

M-Bus-Schnittstelle



Die Bezeichnung M+ und M- dienen zur Unterscheidung der M-Bus-Leitungen. Die Polung ist bei M-Bus-Installationen völlig unerheblich.

Da der CP seine Versorgungsspannung über den Rückwandbus bezieht und damit die angebundenen M-Bus-Module versorgt, können maximal 6 Slaves angeschlossen werden. Die Slaves sind parallel anzubinden.



LEDs

Der Kommunikationsprozessor besitzt 4 LEDs zur Anzeige des Betriebszustands. Die Bedeutung und die jeweiligen Farben dieser LEDs finden Sie in der nachfolgenden Tabelle.

Bez.	Farbe	Bedeutung
PW	Grün	Signalisiert eine anliegende Betriebsspannung
ER	Rot	Signalisiert Fehler verursacht z.B. durch Pufferüberlauf, falsche Baudrate oder fehlendes Stop-Bit.
TxD	Grün	Daten senden (transmit data)
RxD	Grün	Daten empfangen (receive data)

Kommunikationsprinzip

Daten senden und empfangen

Zu sendende Daten werden von der CPU über den Rückwandbus in den entsprechenden Datenkanal geschrieben.

Der Kommunikationsprozessor trägt diese in einem Ringpuffer (2048Byte) ein und gibt sie von dort über M-Bus aus.

Empfängt der Kommunikationsprozessor Daten über M-Bus, werden die Daten in einem Ringpuffer (2048Byte) abgelegt. Die empfangenen Daten können über den Datenkanal telegrammweise von der CPU gelesen werden.

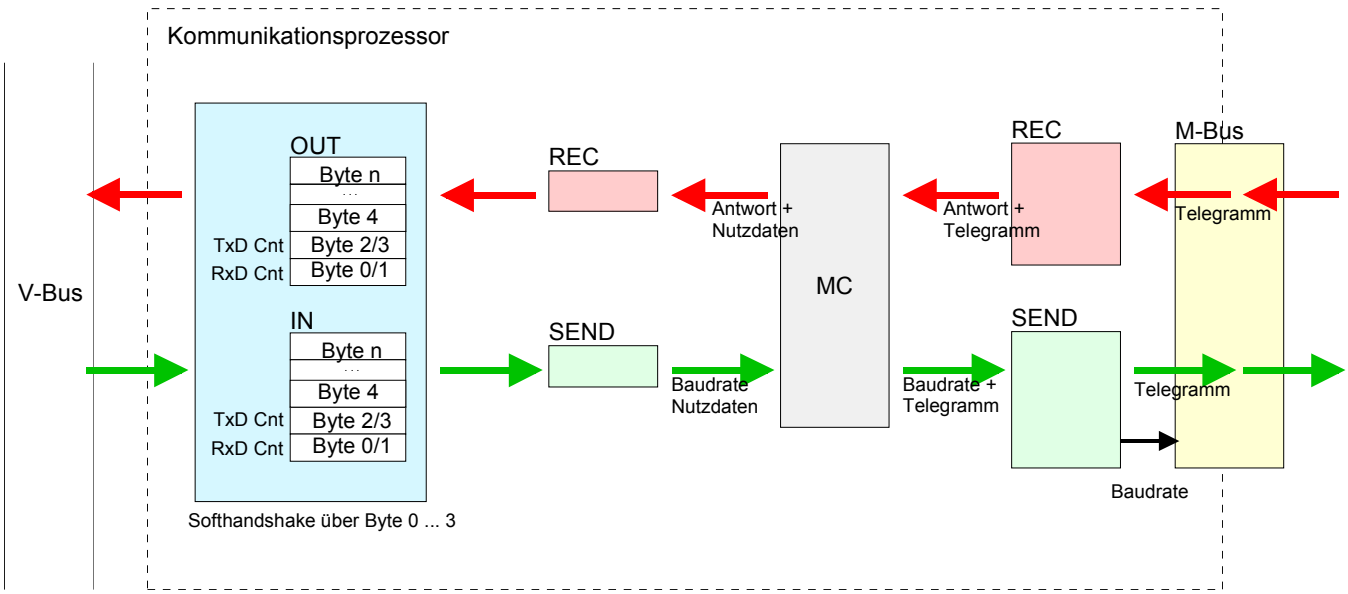
Kommunikation über Rück-wandbus

Der Austausch von empfangenen Telegrammen über den Rückwandbus erfolgt asynchron. Ist ein komplettes Telegramm über M-Bus eingetroffen, wird dies im Puffer abgelegt. Die Nutzdaten werden aus dem M-Bus-Telegramm herausgelöst und über den Rückwandbus an die CPU übergeben.

Aufgaben der CPU

Ein zu sendendes Telegramm ist an den CP 240 zu übergeben. Dieser erkennt aufgrund der Längenangabe den Telegrammtyp, ergänzt dieses mit den entsprechenden Telegramm-Bytes und reicht das Telegramm an den Sendepuffer weiter. Im CP 240 werden diese Blöcke im Sendepuffer zusammengesetzt und bei Vollständigkeit des Telegramms mit der vorgegebenen Baudrate über M-Bus gesendet. Da der Datenaustausch über den Rückwandbus asynchron abläuft, wird ein "Software Handshake" zwischen dem CP 240 und der CPU eingesetzt. Die Register für den Datentransfer vom CP 240 sind 16Byte breit. Für den Handshake sind die Bytes 0 bis 3 (Wort 0 und 2) reserviert.

Folgende Abbildung soll dies veranschaulichen:



Software-handshake

Für den Einsatz des CP 240 in Verbindung mit einer System 200V CPU sind bei VIPA Hantierungsbausteine erhältlich, die den Softwarehandshake komfortabel übernehmen.

Bei Einsatz des CP 240 ohne Hantierungsbausteine soll hier die Funktionsweise anhand eines Beispiels für das Senden und Empfangen von Daten erläutert werden.

Beispiel Daten senden

Es soll z.B. ein Telegramm mit der Länge von 30Byte gesendet werden. Bitte beachten Sie, dass der CP 240 das 1. Byte des Telegramms als Baudrate interpretiert. So werden von der CPU die ersten 12Byte Nutzdaten des Telegramms in die Bytes 4 bis 15 und in Byte 2/3 die Länge des Telegramms (also "30") geschrieben. Der CP 240 empfängt die Daten über den Rückwandbus und kopiert die 12Byte Nutzdaten in den Sendepuffer. Zur Quittierung des Empfangs schreibt der CP 240 in Byte 2/3 den Wert "30" (Länge des Telegramms) zurück.

Beim Empfang der "30", kann die CPU weitere 12Byte Nutzdaten in Byte 4 bis 15 und die Restlänge des Telegramms ("18" Byte) in Byte 2/3 an den CP 240 senden. Dieser speichert wieder die Nutzdaten im Sendepuffer und gibt die Restlänge des Telegramms ("18") auf Byte 2/3 an die CPU zurück.

Beim Empfang der "18", kann die CPU die restlichen 6Byte Nutzdaten in den Byte 4 bis 9 und die Restlänge des Telegramms (also "6") in Byte 2/3 an den CP 240 senden. Dieser speichert die Nutzdaten im Sendepuffer ab und schreibt den Wert "6" auf Byte 2/3 an die CPU zurück.

Beim Empfang der "6" auf Byte 2/3 sendet die CPU eine "0" auf Byte 2/3. Der CP 240 stößt daraufhin das Senden des Telegramms über M-Bus an und schreibt, wenn alle Daten übertragen sind, eine "0" auf Byte 2/3 zurück.

Beim Empfang der "0" kann die CPU ein neues Telegramm an den CP 240 senden.

Beispiel Daten empfangen

Der CP 240 hat z.B. ein Telegramm mit 18Byte Nutzdaten über M-Bus empfangen. Aus diesem Telegramm werden die ersten 11Byte Nutzdaten zusammen mit einem vorangestellten Antwort-Byte in die Bytes 4 bis 15 des Empfangspuffers und in Byte 0/1 die Länge des Telegramms (also "18") übernommen. Die Daten werden über den Rückwandbus an die CPU übertragen. Die CPU speichert die 12Byte und sendet den Wert "18" auf Byte 0/1 an den CP 240 zurück.

Beim Empfang der "18", schreibt der CP 240 die restlichen 7Byte Nutzdaten in Byte 4 bis 10 des Empfangspuffers und in Byte 0/1 die Länge ("7") der übergebenen Nutzdaten. Die CPU speichert die Nutzdaten und gibt an den CP 240 in Byte 0/1 den Wert "7" zurück.

Beim Empfang der "7" sendet der CP 240 den Wert "0" auf Byte 0/1, für Telegramm komplett, an die CPU zurück. Die CPU sendet eine "0" auf Byte 0/1 an den CP 240 zurück.

Mit dem Empfang der "0" kann der CP 240 ein neues Telegramm an die CPU senden.

Übersicht der M-Bus-Telegramme

Übersicht

Unter M-Bus werden folgende 4 Telegramme unterschieden:

- Einzelzeichen
  Das Einzelzeichen dient als Bestätigung korrekt empfangener Telegramme (Syntax und Checksumme sind korrekt)
- Kurzsatz
  Für ein Kurzsatz-Telegramm sind immer 3 Byte anzugeben. Für den M-Bus sind dies Telegramme vom CP 240 zu einem Slave, wie z.B.:
  - SND_NKE: Zähler initialisieren
  - REQ_UD2: Zählerdaten anfordern
- Steuersatz
  Für den Steuersatz sind immer 4 Byte vorzugeben. Hiermit können Sie M-Bus-Steuerbefehle senden wie beispielsweise:
  - Baudrate des Slaves setzen
  - Einen Reset im Slave durchführen
- Langsatz
  Der Langsatz beinhaltet die Anwenderdaten und hat somit eine variable Länge. Hierbei können die Anwenderdaten in beide Richtungen gesendet werden wie z.B.:
  - Anwenderdaten an den Slave senden
  - Slave über sekundäre Adresse selektieren
  - Uhrzeit und Datum eines Slaves setzen
  - Datenbereich zum Auslesen selektieren

Bitte beachten Sie, dass Sie jedem M-Bus Telegramm ein Byte voranstellen, das die zu verwendende Baudrate spezifiziert. Bei fehlerfreiem Empfang steht in Byte 0 des Empfangs-DB 00h.
Ein Wert <> 00h deutet auf einen Fehler hin.

Bei der Übermittlung eines M-Bus-Telegramms an den CP 240 wird der Telegrammtyp erkannt, die Daten werden automatisch in die entsprechende Telegrammstruktur eingebunden und über M-Bus ausgegeben. Mit dem Aufruf der VIPA-Hantierungsbausteine sind telegrammabhängig ausschließlich folgende Daten (grün markiert) zu übergeben. Hierbei ist die Summe dieser Bytes als Längenangabe zu verwenden.

Einzelzeichen	**Kurzsatz**	**Steuersatz**	**Langsatz**
Baudrate	Baudrate	Baudrate	Baudrate
E5h	10h Start	68h Start	68h Start
	C-Feld	L Field = 3	L Field
	A-Feld	L Field = 3	L Field
	Checksum	68h Start	68h Start
	Stop 16h	C-Feld	C-Feld
		A-Feld	A-Feld
		CI-Feld	CI-Feld
		Checksum	User Data (0...252Byte)
		Stop 16h	
			Checksum
			Stop 16h
Länge für Hantierungsbaustein: 2	Länge für Hantierungsbaustein: 3	Länge für Hantierungsbaustein: 4	Länge für Hantierungsbaustein: 5...n

Baudrate

Jedem M-Bus-Telegramm ist ein Byte voranzustellen, das die Baudrate spezifiziert. Hierbei werden folgende Baudraten unterstützt:

Hex-Wert	Baud
B8h	300
BBh	2400
BDh	9600

Befindet sich im 1. Byte keiner der oben genannten Werte, wird automatisch 2400Baud verwendet.

C-Feld

Über das C-Feld wird die Funktion eines Telegramms definiert. Es ermöglicht auch, auf Verbindungsebene die Aufruf- und Antwortrichtung zu unterscheiden.

Abhängig von der Richtung hat das C-Feld folgenden Aufbau:

Senden	0	1	FCB	FCV	F3	F2	F1	F0
Empfangen	0	0	ACD	DFC	F3	F2	F1	F0

Funktionen

Unter M-Bus sind folgende Funktionen definiert:

Name	**C-Feld binär**	**C-Feld hex**	**Telegramm**	**Beschreibung**
SND_NKE	0100 0000	40	Kurzsatz	Dies bewirkt eine Initialisierung der Slaves (Endgeräte) und entspricht einem Löschen des FCB-Bits und einer Quittung durch das Einzelzeichen E5h.
SND_UD	01*F*1 0011	53/73	Lang-/ Steuersatz	Hiermit können Anwenderdaten an Slaves gesendet werden.
REQ_UD2	01*F*1 1011	5B/7B	Kurzsatz	Diese Funktion fordert einen Slave auf, mit Daten der Klasse 2 (z.B. Zählerstände) zu antworten. Besitzt der Slave solche Daten nicht, antwortet dieser mit einem Einzelzeichen. Anderenfalls schickt dieser ein RSP_UD. Bei einer fehlerhaften Übertragung bleibt eine Antwort aus.
REQ_UD1	01*F*1 1010	5A/7A	Kurzsatz	Hiermit können Sie einen Slave auffordern, mit Daten der Klasse 1 (z.B. Alarmprotokolle) zu antworten. Besitzt der Slave solche Daten nicht, antwortet dieser mit einem Einzelzeichen. Anderenfalls schickt dieser ein RSP_UD. Bei einer fehlerhaften Übertragung bleibt eine Antwort aus.
RSP_UD	00*AD* 1000	08/18/28/38	Lang-/ Steuersatz	Datenübertragung nach Anfrage (Antwort des Slaves)

F: FCB-Bit, A: ACD-Bit, D: DFC-Bit

FCB-Bit

Das FCB Bit alterniert bei erfolgreicher Kommunikation. Ein gleichbleibendes FCB fordert das Endgerät auf, nochmals das zuletzt gesendete Telegramm zu wiederholen. Das Ausbleiben einer Antwort des Slaves wird nach 330 Bitzeiten zuzüglich 50ms angenommen. Der Master geht zunächst davon aus, dass ein Fehler in der Verbindungsschicht aufgetreten ist. Der Slave wiederholt die Übertragung des gleichen Telegramms bis zu zweimal. Liegt die Antwort des Slaves bis dahin immer noch nicht vor, so wird eine Pause von 33 Bitzeiten auf dem Bus eingelegt.

Auf die gleiche Weise wird verfahren, wenn der Master eine fehlerhafte Antwort des Slaves empfängt.

Baudrate	33x	330x
300	110	1100
2400	13,8	137,5
9600	3,4	34,4

Bitzeiten in ms

A-Feld

Für die Adressierung der Slaves stehen die Werte 1 bis 250 zur Verfügung. Unkonfigurierte (neue) Slaves besitzen die Primäradresse 0.

Die Adressen 254 und 255 sind als Broadcast-Adresse zu verwenden Unter 255 schickt der Master Informationen an alle Teilnehmer, erhält aber keine Rückantwort. Über 254 antwortet jeder Slave mit seiner Adresse. Bei mehr als einem Slave führt dies zu einer Kollision. Die Adresse 254 sollte ausschließlich für Testzwecke verwendet werden.

Die Adresse 253 zeigt eine Sekundäradressierung an. Die Adressen 251 und 252 sind für zukünftige Erweiterungen reserviert.

CI-Feld

Das CI-Feld definiert den Zweck des gesendeten Telegramms. Die Datenfelder werden immer mit dem niederwertigste Byte zuerst gesendet (LSB first).

Slave → Master

CI-Feld	Anwendung	Definiert in
70h	Senden eines Fehlerzustandes	Usergroup March ´94
71h	Senden eines Alarmzustandes	Usergroup March ´94
72h	Antwort mit variabler Datenstruktur	EN1434-3
73h	Antwort mit fester Datenstruktur	EN1434-3

Master → Slave

CI-Feld	Anwendung	Definiert in
51h	Daten senden	EN1434-3
52h	Slave selektieren	User group July ´93
50h	Reset auf Anwendungsebene	User group March ´94
54h	Slave synchronisieren	-
B8h	Baudrate 300 setzen	User group July ´93
B9h	Baudrate 600 setzen	User group July ´93
BAh	Baudrate 1200 setzen	User group July ´93
BBh	Baudrate 2400 setzen	User group July ´93
BCh	Baudrate 4800 setzen	User group July ´93
BDh	Baudrate 9600 setzen	User group July ´93
BEh	Baudrate 19200 setzen	-
BFh	Baudrate 38400 setzen	-
B1h	RAM Inhalt ausgeben	Techem suggestion
B2h	RAM Inhalt schreiben	Techem suggestion
B3h	Starte den Kalibrationstest-Modus	User roup July ´93
B4h	EEPROM lesen	Techem suggestion
B6h	Softwaretest starten	Techem suggestion
90h ... 97h	reserviert	

Checksum

Checksum dient dazu Übertragungs- und Synchronisationsfehler zu erkennen. Hierbei wird die Checksumme über folgende Datenbytes gebildet: C-Feld, A-Feld, CI-Feld (falls vorhanden) und User-Data (falls vorhanden).

Beispiele

In den nachfolgenden Beispielen soll gezeigt werden, wie beim Senden aus den vorgegebenen Daten ein Telegramm aufgebaut ist und wie ein empfangenes Telegramm im Datenbaustein abgelegt wird.

Daten senden (Kurzsatz)

Vorgabe über DB

BBh	Baudrate
7Bh	C-Feld
FEh	A-Feld

→

Telegramm über M-Bus

10h	Start
7Bh	C-Feld: REQ_UD2
FEh	A-Feld: PtP-Broadcast
79h	Checksum über C- und A-Feld
16h	Stop

ANZ für Hantierungsbaustein FC0: 3

Baudrate

Bitte beachten Sie, dass Sie jedem M-Bus Telegramm ein Byte voranstellen, das die zu verwendende Baudrate spezifiziert.

Daten empfangen

Telegramm über M-Bus

68h	68h Start
00h	L Field
03	L Field
68h	68h Start
08h	C-Feld: RSP_UD
02h	A-Feld: Adresse 02h
72h	CI-Feld: Antw. variabel
01h	Datenfeld
02h	
03h	
75h	Checksum über C, A,CI, Daten
16h	Stop

→

Ablage in DB:

00h	Antwort
08h	C-Feld: RSP_UD
02h	A-Feld: Adresse 02h
72h	CI-Feld: Antwort variable Länge
01h	Datenfeld
02h	
03h	

ANZ in Hantierungsbaustein FC1: 7

Antwort-Byte

Bei fehlerfreiem Empfang steht in Byte 0 des Empfangs-DB der Wert 00h. Ein Wert <> 00h deutet auf einen Fehler hin.

Hier haben die Bits folgende Belegung:

Bit 0: wird gesetzt wenn keine Rückantwort erhalten wurde

Bit 1: wird gesetzt bei Kurzschluss am Bus

Bit 2 ... 7: reserviert

Beispiel zum Einsatz unter M-Bus

Übersicht

In dem nachfolgenden Beispiel wird eine M-Bus-Kommunikation (Senden und Empfangen) aufgebaut. Weiter soll das Beispiel zeigen, wie Sie unter Einsatz der Hantierungsbausteine auf einfache Weise die Kontrolle über die Kommunikationsvorgänge haben.

Bei Bedarf können Sie das Beispielprojekt von VIPA beziehen.

Voraussetzung

Folgende Komponenten sind für das Beispiel erforderlich:

1 System 200V bestehend aus CPU 21x und CP 240 M-Bus

1 Erfassungsgerät mit M-Bus-Schnittstelle

Projektiertool SIMATIC Manager von Siemens mit Übertragungskabel

Vorgehensweise

Bauen Sie das System 200V auf.

Laden Sie das Beispielprojekt, passen Sie ggf. die Peripherieadresse an und übertragen Sie Ihr Projekt in die CPU.

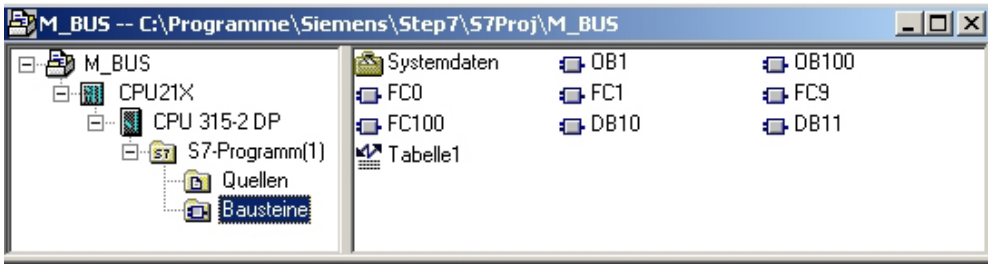
Projekt dearchivieren

Zum Dearchivieren im Siemens SIMATIC Manager gehen Sie nach folgenden Schritten vor:

- Starten Sie den Siemens SIMATIC Manager
- Zum Entpacken der Datei MBUS.zip gehen Sie auf **Datei** > *dearchivieren*.
- Wählen sie die Beispieldatei MBUS.zip aus und geben Sie als Zielverzeichnis "s7proj" an.
- Öffnen Sie das entpackte Projekt.

Projekt -Struktur

Das Projekt beinhaltet schon das SPS-Programm und die Hardware-Konfiguration und besitzt folgende Struktur:



Datenbausteine In diesem Beispiel werden folgende Datenbausteine verwendet:

DB10
Sendebaustein

Adr.	Name	Typ	Kommentar
0.0		STRUCT	
+0.0	Sendefach	STRUCT	
+0.0	Baudrate	BYTE	B8h=300, BBh=2400, BDh=9600
+1.0	OK / C-Feld	BYTE	E5h=OK / C-Feld
+2.0	A-Feld	BYTE	A-Feld
+3.0	CI-Feld	BYTE	CI-Feld
+4.0	User-Data Byte 0	BYTE	

...

+252.0	User-Data Byte 247	BYTE	Übertragung abgeschlossen
+253.0	Reserve	BYTE	
+254.0	Anzahl	WORD	Sendelänge
+256.0	gesendet	WORD	schon gesendete Daten
+258.0	Byte_Zaehler	WORD	Sendelänge (intern)
+260.0	Kom_Ende	BOOL	Telegramm komplett gesendet
+260.1	LB	BOOL	letzter Block wurde gesendet
+260.2	SL	BOOL	Senden läuft noch
+260.3	Fehl	BOOL	Fehler beim Senden aufgetreten
+260.4	Senden_Start	BOOL	Start-Bit
+261.0	PAFE	BYTE	Parametrierfehler
=262.0		END_STRUCT	

DB11
Empfangsbaustein

Adr.	Name	Typ	Kommentar
0.0		STRUCT	
+0.0	Data	ARRAY [0..100]	
*1.0		Byte	
+102.0	Anzahl	WORD	Anzahl empfangener Byte
+104.0	empfangen	WORD	schon empfangene Daten
+106.0	Byte_Zaehler	WORD	Anzahl empfangener Byte (intern)
+108.0	Empf_laeuft	BOOL	Empfang läuft
+108.1	LB	BOOL	letzter Block empfangen
+108.2	Fehl	BOOL	Fehler beim Empfang
+108.3	Reserve	BOOL	
+108.4	Empfang fertig	BOOL	Empfang fertig
+109.0	PAFE	BYTE	Parametrierfehler
=110.0		END_STRUCT	

SPS-Programm Das SPS-Programm hat folgenden Aufbau:

OB1

```
CALL       FC    9             //SYNCHRON_RESET
 ADR       :=256               //Baugruppenadresse
 TIMER_NR  :=T8                //Wartezeit auf CP
 ANL       :=M3.0              //Anlauf der CPU ist erfolgt
 NULL      :=M3.1              //Zwischenmerker
 RESET     :=M3.2              //Reset auf CP auslösen
 STEUERB_S:=DB10.DBB260        //Steuerbits für Send
 STEUERB_R:=DB11.DBB108        //Steuerbits für Receive

U          M     3.0           //solange Synchron aktiv
BEB                            //keine Bearbeitung des CP

CALL       FC    100           //M-Bus-Kommunikation
 Adr_CP    :=256               //Baugruppenadresse
 Baud      :=MB100             //Übergabe Baudrate
 C_Field   :=MB101             //Übergabe Wert C-Field
 A_Field   :=MB102             //Übergabe Wert A-Field
 CI_Field  :=MB103             //Übergabe Wert CI-Field
 Data      :=MB104             //Übergabe Telegrammlänge
 RET_VAL   :=MW106             //Rückgabewert
 Senden_Start:=M99.0           //Auftrag starten

U          M     99.0          //Auftrag läuft
BEB

L          MW    106           //Rückgabe von Sendefunktion
L          W#16#2000           //Fertig ohne Fehler
==I
SPB        copy

NOP        0                   //Fehlerauswertung
BEA

copy:  L          0            //Fertig ohne Fehler löschen
       T          MW    106
       L          MB    102    //Teilnehmeradresse
       L          20           //Basis-Nr. für die Daten-DBs
       +I
       T          MW    50     //Datenbausteinnummer zum Ablegen der Daten
       L          0            //1. zu kopierendes Byte
       T          MW    188    //Bytezähler vorbelegen

loo:   L          MW    188    //Bytezähler laden
       SLW        3            //x8 ist Byteadresse im DB
       T          MD    184    //Adresse sichern
       AUF        DB    11     //Empfangspuffer öffnen
       L          DBB   [MD 184]  //Wert aus Empfangspuffer
       AUF        DB    [MW 50]
       T          DBB   [MD 184]  //in Daten-DB ablegen

       L          MW    188
       +1                      //Bytenummer erhöhen
       T          MW    188
       L          DB11.DBW  102  //letztes zu kopierendes Byte
       <=I                     //noch nicht alle Byte kopiert
       SPB        loo          //dann weiter
```

OB 100

```
UN         M     3.0
S          M     3.0           //Anlauf-Kennung setzen
```

FC 100

Mit dieser Funktion wird eine Anfrage an einen M-Bus Teilnehmer gesendet und die Antwort entgegengenommen. Die Sendedaten sind vor dem Aufruf der Funktion in den DB10 ab Datenbyte 4 einzutragen.

```
        UN      #Senden_Start
        BEB
        U       DB11.DBX    108.7       //Warten auf Quittung
        SPB   REC
        NOP     0                       //Sendedaten in Sendepuffer eintragen
        L       #Baud                   //1.Sendebyte ist Baudrate
        T       DB10.DBB    0
        L       #C_Field                //2.Sendebyte ist C_Field
        T       DB10.DBB    1
        L       #A_Field                //3.Sendebyte ist A_Field
        T       DB10.DBB    2
        L       #CI_Field               //4.Sendebyte ist CI_Field
        T       DB10.DBB    3           //bei Long Frame müssen Daten ab User
        NOP   0                         //Data vor Aufruf des FC eingetragen
        SET                             //werden
        S       DB10.DBX    260.4       //Sendefreigabe setzen
        L       0
        L       #Data                   //Telegrammlänge bei Long Frame
        <>I
        +4
        T       DB10.DBW    254         //Telegrammlänge für Long Frame
        SPB     send
        L       0
        L       #CI_Field               //Kennung für Control Frame
        <>I
        L       4                       //Telegrammlänge für Control Frame
        SPB     sen1
        L       3                       //Telegrammlänge für Short Frame
sen1:   T       DB10.DBW    254         //Telegrammlänge
send:   CALL    FC    0                 //Baustein Send
         ADR    :=#Adr_CP               //Baugruppenadresse
         _DB    :=DB10                  //DB Sendepuffer
         ABD    :=W#16#0                //1. zu sendendes Datenbyte
         ANZ    :=DB10.DBW254           //Anzahl Sendedaten
         PAFE   :=DB10.DBB261           //Fehlerbyte
         FRG    :=DB10.DBX260.4         //Sendefreigabe
         GESE   :=DB10.DBW256           //Interne Variable
         ANZ_INT  :=DB10.DBW258         //Interne Variable
         ENDE_KOM  :=DB10.DBX260.0      //Interne Variable
         LETZTER_BLOCK:=DB10.DBX260.1   //Interne Variable
         SENDEN_LAEUFT:=DB10.DBX260.2   //Interne Variable
         FEHLER_KOM   :=DB10.DBX260.3   //Interne Variable
        U       DB10.DBX    260.4       //Senden läuft noch
        BEB                             //dann Ende
        S       DB11.DBX    108.7       //dann warten auf Quittung
REC:    NOP     0
        CALL    FC    1
         ADR    :=#Adr_CP               //Baugruppenadresse
         _DB    :=DB11                  //DB Empfangspuffer
         ABD    :=W#16#0                //1. Datenbyte Empfangspuffer
         ANZ    :=DB11.DBW102           //Anzahl empfangener Byte
         EMFR   :=DB11.DBX108.4         //Telegramm komplett empfangen
         PAFE   :=DB11.DBB109           //Fehlerbyte
         GEEM   :=DB11.DBW104           //Interne Variable
         ANZ_INT  :=DB11.DBW106         //Interne Variable
         EMPF_LAEUFT    :=DB11.DBX108.0 //Interne Variable
         LETZTER_BLOCK  :=DB11.DBX108.1 //Interne Variable
         FEHL_EMPF      :=DB11.DBX108.2 //Interne Variable
        UN      DB11.DBX    108.4       //neuer Wert noch nicht empfangen
        BEB
        R       DB11.DBX  108.4
        R       DB11.DBX  108.7
        R       #Senden_Start
        L       DB11.DBW  102
        L       1                       //wurde nur 1Byte empfangen ->Fehler
        ==I
        SPB     Fehl
        L       W#16#2000               //Nach Empfang einer Antwort, Startbit
Ende:   T       #RET_VAL                //löschen und Kennung an RET_VAL zurück
        BEA
Fehl:   L       DB11.DBB    0           //Empfangenes Byte
        L       1                       //Keine Antwort vom M-Bus-Slave
        ==I
        L       W#16#8001               //Fehlerkennung für keine Antwort
        SPB     Ende
        L       W#16#80FF               //Undefinierte Antwort vom CP
        SPA     Ende
```

Technische Daten

CP 240 mit M-Bus-Schnittstelle

Elektrische Daten	VIPA 240-1FA20
Anzahl der Kanäle	1
Anzahl anbindbarer Slaves	6
Spannungsversorgung	5V über Rückwandbus
Stromaufnahme	max. 300mA
Externe Spannungsversorgung	-
Statusanzeige	über LED auf der Frontseite
Busspannung Ruhepegel	30V
Max. Bus-Ruhestrom	9mA
Starre Bitschwelle	11mA
Kurzschlussfestigkeit	permanent
Abschaltschwelle bei Überstrom	50mA
Mindestabschaltzeit	50ms
Innenwiderstand	<100Ω
Galvanische Trennung zum M-Bus	ja
Anschlüsse / Schnittstellen	2pol. Buchse für M-Bus
Übertragungsrate	300, 2400, 9600Baud
Programmierdaten	
Eingabedaten	16Byte
Ausgabedaten	16Byte
Parameterdaten	16Byte
Diagnosedaten	-
Maße und Gewicht	
Abmessungen (BxHxT) in mm	25,4x76x78
Gewicht	80g

Anhang

A Index

T

Z

M.Stich