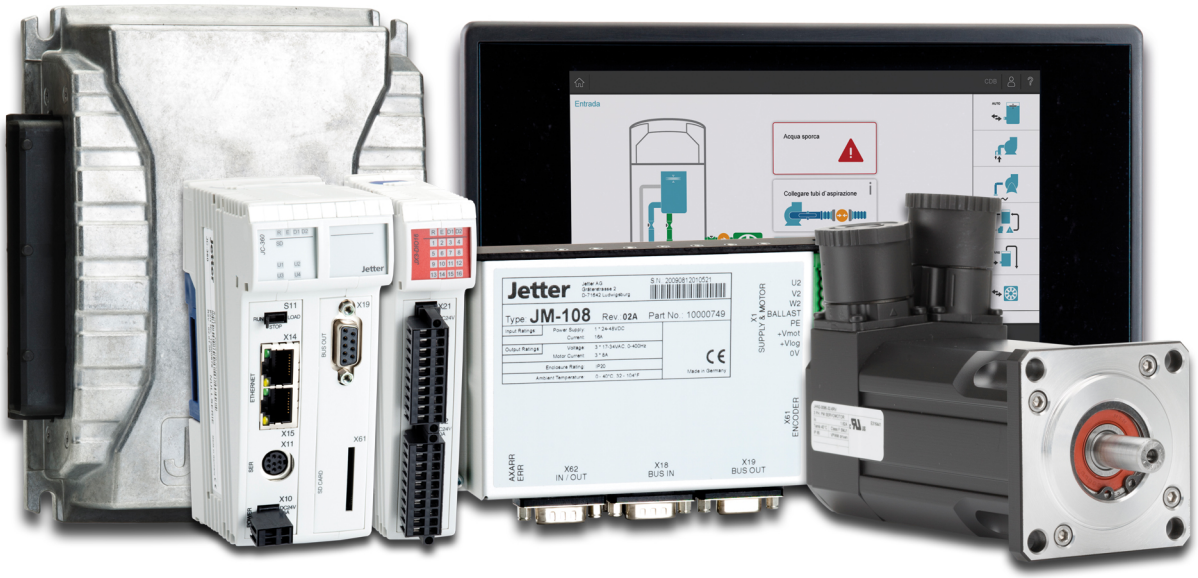
Jetter automation



Themenhandbuch

Freiprogrammierbare Prim-Schnittstellen

60881373

We automate your success.

Artikelnummer: 60881373
Version 1.10
April 2017 / Printed in Germany

Inhaltsverzeichnis

1 Freiprogrammierbare serielle Schnittstelle

Einleitung

Dieses Kapitel beschreibt, wie die serielle Schnittstelle der Steuerung im Anwendungsprogramm angesprochen wird, um Zeichen zu senden und zu empfangen.

Anwendungen

Die freiprogrammierbare serielle Schnittstelle ermöglicht den Anschluss von Geräten, die vom Betriebssystem der Steuerung nicht unterstützte Protokolle zur Kommunikation verwenden. Das sind z. B.:

- Waagen
- Scanner
- Anzeigen
- Frequenzumrichter
- Temperaturregler
- usw.

Voraussetzungen an den Programmierer

Dieses Kapitel wendet sich an Entwickler von Anwendungsprogrammen, die Erfahrung mit der Datenübertragung über asynchrone serielle Schnittstellen haben. Voraussetzungen sind z. B. folgende Kenntnisse:

- Verdrahtung von seriellen Schnittstellen
- Übertragungsparameter (Baudrate, Parität, usw.)
- Sende- und Empfangspuffer
- usw.

Inhalt

1.1 Anschluss

Einleitung

Dieses Kapitel beschreibt den Anschluss an eine asynchrone serielle Schnittstelle des Geräts.

Inhalt

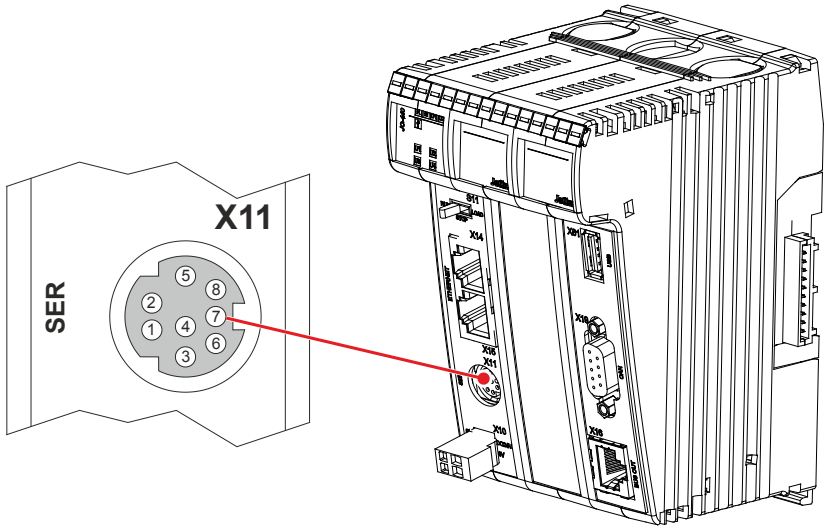
Serielle Schnittstelle Buchse X11

Schnittstellen der Buchse

An die Buchse X11 schließen Sie an:

- Einen PC
- Ein Bediengerät der Jetter AG
- Ein beliebiges Gerät mit RS-232/422/485-Schnittstelle

Belegung der Buchse



Pin	Signal	Beschreibung
1	RDA	RS-422; Empfangsdaten invertiert
2	GND	Bezugspotenzial
3	RDB	RS-422; Empfangsdaten nicht invertiert
4	RxD	RS-232; Empfangsdaten
5	SDB	RS-422; Sendedaten nicht invertiert RS-485; Sende-/Empfangsdaten nicht invertiert
6	DC24V	Versorgungsspannung Bediengerät
7	SDA	RS-422; Sendedaten invertiert RS-485; Sende-/Empfangsdaten invertiert
8	TxD	RS-232; Sendedaten

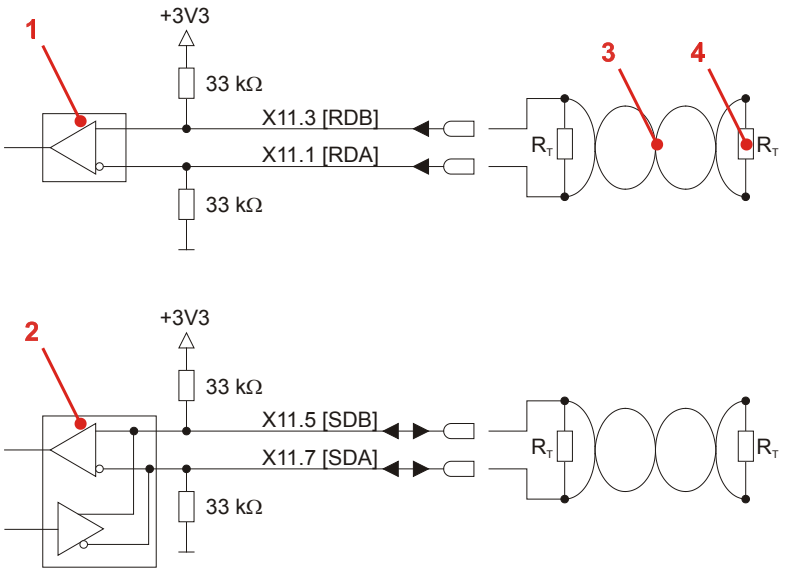
Einschränkungen

Obwohl verschiedene Hardwaretreiber bestückt sind, ist nur eine Schnittstelle vorhanden.

Das bedeutet:

Wenn z. B. über RS-422 kommuniziert wird, kann nicht gleichzeitig und unabhängig davon über RS-232 kommuniziert werden.

Prinzipschaltbild



Nummer	Teil	Funktion bei RS-422	Funktion bei RS-485
1	Receiver	Empfängt Daten	Unbenutzt
2	Receiver/Transmitter	Sendet Daten	Sendet und empfängt Daten
3	Serielle Leitung	Verdrillte Leitung der seriellen Schnittstelle	
4	R_T	Abschlusswiderstand	

Abschlusswiderstand

Verbinden Sie in den folgenden Fällen die beiden seriellen Leitungen mit einem Abschlusswiderstand:

- Bei großer Leitungslänge
- Bei hoher Baudrate

Wählen Sie einen Abschlusswiderstand, der dem Wellenwiderstand der verwendeten Leitung entspricht.

Technische Daten

Parameter	Beschreibung
Klemmenart	MiniDIN, geschirmt
Anzahl Pins	8
Potenzialtrennung	Keine
Anzahl Schnittstellen	1 serielle Schnittstelle
Schnittstellenstandards	RS-232/RS-422/RS-485-2
Baudraten	JC-4xx: 1.200 ... 115.200 Baud JC-3xx: 2.400 ... 115.200 Baud
Bits pro Zeichen	5, 6, 7, 8
Anzahl Stoppbits	1, 2
Parität	Gerade, ungerade, keine

Kabel für Buchse X11

Zum Anschluss von Geräten an die Buchse X11 können Sie folgende Kabel separat bestellen:

Artikel-Nr.	Artikel	Beschreibung
60867209	KAY_0576-0050	Vom JetControl zum Modem mit 9-poligem Sub-D, Länge 0,5 m
60868359	KABEL-KONF-NR.196 2.5M	Vom JetControl zum PC mit 9-poligem Sub-D, Länge 2,5 m
60860013	KABEL-KONF-NR.196 5M	Vom JetControl zum PC mit 9-poligem Sub-D, Länge 5 m
60868956	KABEL-KONF-NR.196 8M	Vom JetControl zum PC mit 9-poligem Sub-D, Länge 8 m
60860011	KABEL-KONF-NR.192 2.5M	Vom JetControl zum Bediengerät mit 15-poligem Sub-D, Länge 2,5 m
60860012	KABEL-KONF-NR.193 5M	Vom JetControl zum Bediengerät mit 15-poligem Sub-D, Länge 5 m
60872142	KABEL-KONF-NR.192 10M	Vom JetControl zum Bediengerät mit 15-poligem Sub-D, Länge 10 m
60872884	KABEL-KONF-NR.192 15M	Vom JetControl zum Bediengerät mit 15-poligem Sub-D, Länge 15 m
60864359	KAY_0386-0250	Vom JetControl zum LCD 60 mit 15-poligem Sub-D, Länge 2,5 m
60864360	KAY_0386-0500	Vom JetControl zum LCD 60 mit 15-poligem Sub-D, Länge 5 m
60864897	KAY_0533-0025	Vom JetControl zum LCD 52/54 mit 15-poligem Sub-D, Länge 0,25 m
60864257	KABEL-KONF-NR.197 5M	Vom JetControl zum JetView 200/300 mit 9-poligem Sub-D, Länge 5 m
60871930	KABEL-KONF-NR.197 12M	Vom JetControl zum JetView 200/300 mit 9-poligem Sub-D, Länge 12 m

1.2 Funktion der freiprogrammierbaren Schnittstelle

Einleitung

Dieses Kapitel beschreibt die Funktionsweise der freiprogrammierbaren seriellen Schnittstelle.

Einschränkungen

Bei der Verwendung als freiprogrammierbare serielle Schnittstelle gelten folgende Einschränkungen:

- Obwohl verschiedene Hardwaretreiber bestückt sind, ist nur eine Schnittstelle vorhanden.
  Das bedeutet: Wenn z. B. über RS-422 kommuniziert wird, kann nicht gleichzeitig und unabhängig davon über RS-232 kommuniziert werden.
- Das Betriebssystem führt das pcomX-Protokoll nicht mehr aus.
  Das bedeutet: Auf dieser Schnittstelle kann nicht mehr z. B. mit JetSym, JetViewSoft oder Bediengeräten über dieses Protokoll kommuniziert werden.

Inhalt

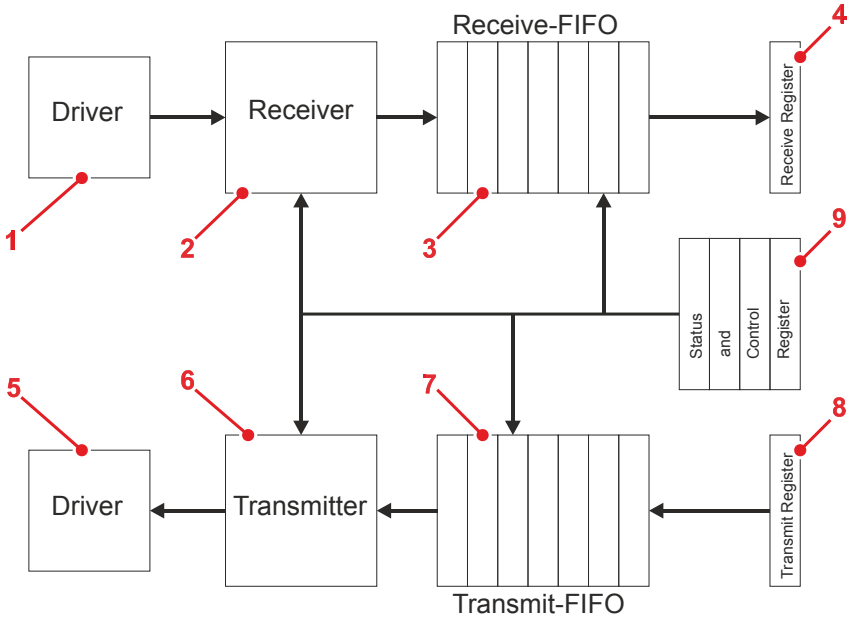
Funktionsweise

Einleitung

Das Betriebssystem des Geräts stellt für die freiprogrammierbare serielle Schnittstelle einen Empfangspuffer und einen Sendepuffer zur Verfügung. Die Puffer ermöglichen, die Übertragungsgeschwindigkeit zwischen dem Anwendungsprogramm und der seriellen Schnittstelle anzupassen.

Blockdiagramm

Die folgende Abbildung zeigt das Blockdiagramm der freiprogrammierbaren seriellen Schnittstelle:



Elemente der Schnittstelle

Die freiprogrammierbare serielle Schnittstelle besteht aus den folgenden Teilen:

Nummer	Teil	Funktion
1	Schnittstellentreiber	Wandelt die Signale der unterschiedlichen Schnittstellenstandards (RS-232, RS-422, RS-485) in interne Signalpegel um
2	Empfänger	Führt die Seriell-Parallel-Wandlung durch
3	Empfangspuffer	Zwischenpuffer für empfangene Zeichen
4	Empfangsregister	Durch einen Lesezugriff auf dieses Register werden die empfangenen Zeichen aus dem Empfangspuffer (3) gelesen
5	Schnittstellentreiber	Wandelt die internen Signalpegel in die unterschiedlichen Schnittstellenstandards (RS-232, RS-422, RS-485) um
6	Sender	Führt die Parallel-Seriell-Wandlung durch
7	Sendepuffer	Zwischenpuffer für die zu sendenden Zeichen

Nummer	Teil	Funktion
8	Senderegister	Durch einen Schreibzugriff auf dieses Register werden die zu sendenden Zeichen in den Sendepuffer (7) eingetragen und von dort vom Sender (6) gesendet
9	Status- und Steuerregister	Abfrage von Pufferfüllständen und Fehlerzuständen; Einstellung der Übertragungsparameter

Ein Zeichen empfangen

Der Empfang eines Zeichens geschieht, wie folgend beschrieben:

Stufe	Beschreibung
1	Der Schnittstellentreiber wandelt die Signale "auf der Leitung" in interne Signalpegel um und leitet sie weiter an den Empfänger.
2	Der Empfänger führt die Seriell-Parallel-Wandlung des Zeichens durch und prüft die eingestellten Übertragungsparameter.
3	Der Empfänger trägt das Zeichen in den Empfangspuffer ein, wenn in diesem noch Platz vorhanden ist. Sonst wird das Zeichen verworfen und ein Überlauffehler signalisiert.
4	Über das Empfangsregister kann das Zeichen aus dem Empfangspuffer gelesen werden.

Ein Zeichen senden

Das Senden eines Zeichens geschieht, wie folgend beschrieben:

Stufe	Beschreibung
1	Über das Senderegister wird ein Zeichen in den Sendepuffer eingetragen, wenn in diesem noch Platz vorhanden ist. Sonst wird das Zeichen verworfen.
2	Sobald der Sender ein Zeichen gesendet hat, liest er das nächste Zeichen aus dem Sendepuffer.
3	Der Sender führt die Parallel-Seriell-Wandlung durch und sendet das Zeichen mit den eingestellten Übertragungsparametern an den Schnittstellentreiber.
4	Der Schnittstellentreiber wandelt die internen Signalpegel in die unterschiedlichen Schnittstellenstandards um.

Fehlererkennung

Folgende Fehler beim Empfang eines Zeichens werden von der Steuerung erkannt und im Register *Fehlerstatus* angezeigt:

Fehler	Beschreibung	Auswirkung
Rahmenfehler	Das Format des empfangenen Zeichens stimmt nicht mit den eingestellten Parametern überein.	Die verfälschten Zeichen werden im Empfangspuffer gespeichert und das Fehlerbit *Framing error* gesetzt. Der Fehlerzähler wird erhöht.

Fehler	Beschreibung	Auswirkung
Paritätsfehler	Das Paritätsbit des empfangenen Zeichens stimmt nicht.	Das verfälschte Zeichen wird im Empfangspuffer gespeichert und das Fehlerbit *Parity error* gesetzt. Der Fehlerzähler wird erhöht.
Pufferüberlauf	Ein Zeichen wird empfangen, obwohl der Empfangspuffer voll ist.	Das Zeichen wird verworfen und das Fehlerbit *Overflow* gesetzt. Der Fehlerzähler wird erhöht.

Fehlerbehandlung

Da die Fehlerbits nicht einzelnen Zeichen im Empfangspuffer zugeordnet werden können, sollten bei einem gesetzten Fehlerbit alle Zeichen aus dem Empfangspuffer entnommen und verworfen werden.

Mögliche Fehlerursachen und ihre Behandlung:

Fehler	Mögliche Ursache	Fehlerbehandlung
Rahmenfehler	Störung der Datenübertragung durch EMV-Probleme, schadhafte Kabel oder Steckverbindungen	▪ Prüfen Sie die Verdrahtung und die Steckverbindungen. ▪ Verwenden Sie geschirmte Kabel. ▪ Legen Sie die Kabel nicht in die Nähe von Störquellen.
	Falsche Einstellung der Übertragungsparameter (Baudrate, Anzahl Stoppbits usw.)	▪ Stellen Sie die Übertragungsparameter passend zu den Einstellungen im externen Gerät ein.
Paritätsfehler	Störung der Datenübertragung durch EMV, schadhafte Kabel oder Steckverbindungen	▪ Prüfen Sie die Verdrahtung und die Steckverbindungen. ▪ Verwenden Sie geschirmte Kabel. ▪ Legen Sie die Kabel nicht in die Nähe von Störquellen.
	Falsche Einstellung der Parität	▪ Stellen Sie die Parität passend zu der Einstellung im externen Gerät ein.
Pufferüberlauf	Das externe Gerät sendet die Zeichen schneller, als sie vom Anwendungsprogramm aus dem Empfangspuffer gelesen werden.	▪ Programmieren Sie einen Software-Handshake. ▪ Stellen Sie eine niedrigere Baudrate ein. ▪ Lesen Sie, durch geeignete Programmierung, die Zeichen schneller aus dem Empfangspuffer.

1.3 Register

Einleitung

Dieses Kapitel beschreibt die Register der freiprogrammierbaren seriellen Schnittstelle. Über diese Register führen Sie folgende Funktionen aus:

- Schnittstelle parametrieren
- Zeichen senden
- Zeichen empfangen

Inhalt

Registernummern

Einleitung

Die Register jeweils einer Schnittstelle sind in einem Registerblock zusammengefasst. Die Basisregisternummer dieses Blocks ist steuerungsabhängig.

Registernummern

Basisregisternummer	Registernummern
103000	103000 ... 103019

Registernummern ermitteln

In diesem Kapitel sind jeweils nur die letzten zwei Ziffern der Registernummer angegeben, z. B. MR 14. Addieren Sie zu dieser Modulregisternummer die Basisregisternummer des jeweiligen Geräts, z. B. 103000, um die vollständige Registernummer zu ermitteln.

Registerübersicht

Register	Beschreibung
MR 0	Fehlerstatus
MR 1	Protokoll
MR 2	Baudrate
MR 3	Anzahl Datenbits pro Zeichen
MR 4	Anzahl Stoppbits
MR 5	Parität
MR 6	Schnittstellenstandard
MR 10	Sendepuffer
MR 11	Sendepufferfüllstand
MR 12	Empfangspuffer (ohne Entfernen der Zeichen)
MR 13	Empfangspuffer (mit Entfernen der Zeichen)
MR 14	Empfangspufferfüllstand
MR 15	Empfangspuffer, 16 Bit, little endian
MR 16	Empfangspuffer, 16 Bit, big endian
MR 17	Empfangspuffer, 32 Bit, little endian
MR 18	Empfangspuffer, 32 Bit, big endian
MR 19	Fehlerzähler

Registerbeschreibung

Einleitung

Wenn Sie die Steuerregister MR 1 bis MR 6 beschreiben, wird immer die gesamte Schnittstelle neu initialisiert und dabei der Sendepuffer und der Empfangspuffer gelöscht.

MR 0

Fehlerstatus

Dieses Register zeigt bitkodiert Fehler an, die beim Empfang eines Zeichens festgestellt worden sind.

Bedeutung der Bits		
Bit 12	**Pufferüberlauf**	
	1 =	Obwohl der Empfangspuffer voll ist, wurden ein oder mehrere Zeichen empfangen
Bit 13	**Paritätsfehler**	
	1 =	Das Paritätsbit des empfangenen Zeichens stimmt nicht
Bit 14	**Rahmenfehler**	
	1 =	Das Format des empfangenen Zeichens stimmt nicht mit den eingestellten Parametern überein

Modulregistereigenschaften	
Zugriff	Lesen/schreiben (löschen)

MR 1

Protokoll

In diesem Register stellen Sie ein, welches Protokoll vom Betriebssystem der Steuerung unterstützt wird. Das Register definiert also, wie die Schnittstelle verwendet wird.

Modulregistereigenschaften		
Werte	1	System-Logger
	2	Freiprogrammierbare Schnittstelle
	3	PcomX
Wert nach Reset	3	

MR 2

Baudrate

In diesem Register stellen Sie die Übertragungsgeschwindigkeit in Baud ein.

Modulregistereigenschaften	
Werte	JC-4xx: 1.200 ... 115.200 JC-3xx: 2.400 ... 115.200
Wert nach Reset	9.600

MR 3

Anzahl Datenbits pro Zeichen

In diesem Register stellen Sie die Anzahl der Datenbits eines Zeichens ein.

Modulregistereigenschaften	
Werte	5, 6, 7, 8
Wert nach Reset	8

MR 4

Stoppbits

In diesem Register stellen Sie die Anzahl der Stoppbits eines Zeichens ein.

Modulregistereigenschaften		
Werte	1	1 Stoppbit
	2	1,5 Stoppbits bei MR 3 = 5
		2 Stoppbits bei MR 3 = 6, 7, 8
Wert nach Reset	1	

MR 5

Parität

In diesem Register stellen Sie die Parität eines Zeichens ein.

Modulregistereigenschaften		
Werte	0	Keine (No)
	1	Ungerade (Odd)
	2	Gerade (Even)
	3	1 (Mark)
	4	0 (Space)
Wert nach Reset	2	

MR 6

Schnittstellenstandard

In diesem Register stellen Sie die Hardwareschnittstelle ein, über die die Zeichen empfangen und gesendet werden.

Modulregistereigenschaften		
Werte	0	RS-232
	1	RS-422
	2	Reserviert
	3	RS-485, 2-Draht
Wert nach Reset	1	

MR 10

Sendepuffer

In dieses Register wird ein zu sendendes Zeichen geschrieben.

- Wenn noch Platz im Sendepuffer ist, wird das Zeichen dort eingetragen. Gesendet wird das Zeichen, sobald alle zuvor eingetragenen Zeichen gesendet worden sind.
- Ob noch Platz im Sendepuffer ist, muss vor dem Senden im Anwendungsprogramm durch das Lesen von MR 11 geprüft werden.
- Der Sendepuffer arbeitet nach dem FIFO-Prinzip. Das erste eingetragene Zeichen wird als Erstes gesendet.

Modulregistereigenschaften		
Werte	0 ... 31	5 Bit pro Zeichen
	0 ... 63	6 Bit pro Zeichen
	0 ... 127	7 Bit pro Zeichen
	0 ... 255	8 Bit pro Zeichen
Zugriff	Lesen	Letztes geschriebenes Zeichen
	Schreiben	Senden eines Zeichens

MR 11

Sendepufferfüllstand

Dieses Register zeigt an, wie viele Zeichen im Sendepuffer enthalten sind.
Es passen max. 32.768 Zeichen in den Puffer.

Modulregistereigenschaften	
Werte	0 ... 32.768

MR 12

Empfangspuffer, 8 Bit (ohne Entfernen des Zeichens)

Dieses Register zeigt das älteste im Empfangspuffer gespeicherte Zeichen. Das Zeichen wird nicht aus dem Puffer entfernt.

Modulregistereigenschaften		
Werte	0 ... 31	5 Bit pro Zeichen
	0 ... 63	6 Bit pro Zeichen
	0 ... 127	7 Bit pro Zeichen
	0 ... 255	8 Bit pro Zeichen
Zugriff	Lesen	Ältestes Zeichen im Puffer
Wird wirksam	Wenn MR 14 > 0	

MR 13

Empfangspuffer, 8 Bit (mit Entfernen des Zeichens)

Dieses Register zeigt das älteste im Empfangspuffer gespeicherte Zeichen. Das Zeichen wird aus dem Puffer entfernt, so dass beim nächsten Lesen das nächste empfangene Zeichen ausgelesen werden kann.

Modulregistereigenschaften		
Werte	0 ... 31	5 Bit pro Zeichen
	0 ... 63	6 Bit pro Zeichen
	0 ... 127	7 Bit pro Zeichen
	0 ... 255	8 Bit pro Zeichen
Zugriff	Lesen	Ältestes Zeichen im Puffer
Wird wirksam	Wenn MR 14 > 0	

MR 14

Empfangspufferfüllstand

Dieses Register zeigt an, wie viele Zeichen im Empfangspuffer enthalten sind. Bei jedem Lesezugriff auf MR 13 wird dieses Register um 1 verringert.

Modulregistereigenschaften	
Werte	0 ... 32.768

MR 15 **Empfangspuffer, 16 Bit, little endian**

Ein Lesezugriff auf dieses Register entfernt 2 Zeichen aus dem Empfangspuffer und liefert sie als 16-Bit-Wert zurück.

Zuordnung:

Zeichen	Bits im Register
Erstes	Bit 0 ... 7
Zweites	Bit 8 ... 15

Modulregistereigenschaften		
Werte	0 ... 65.535	
Zugriff	Lesen	Entnimmt 2 Zeichen aus dem Puffer
Wird wirksam	Wenn MR 14 > 1	

MR 16 **Empfangspuffer, 16 Bit, big endian**

Ein Lesezugriff auf dieses Register entfernt 2 Zeichen aus dem Empfangspuffer und liefert sie als 16-Bit-Wert zurück.

Zuordnung:

Zeichen	Bits im Register
Erstes	Bit 8 ... 15
Zweites	Bit 0 ... 7

Modulregistereigenschaften		
Werte	0 ... 65.535	
Zugriff	Lesen	Entnimmt 2 Zeichen aus dem Puffer
Wird wirksam	Wenn MR 14 > 1	

MR 17

Empfangspuffer, 32 Bit, little endian

Ein Lesezugriff auf dieses Register entfernt 4 Zeichen aus dem Empfangspuffer und liefert sie als 32-Bit-Wert zurück.
Zuordnung:

Zeichen	Bits im Register
Erstes	Bit 0 ... 7
Zweites	Bit 8 ... 15
Drittes	Bit 16 ... 23
Viertes	Bit 24 ... 31

Modulregistereigenschaften		
Werte	-2.147.483.648 ... 2.147.483.647	
Zugriff	Lesen	Entnimmt 4 Zeichen aus dem Puffer
Wird wirksam	Wenn MR 14 > 3	

MR 18

Empfangspuffer, 32 Bit, big endian

Ein Lesezugriff auf dieses Register entfernt 4 Zeichen aus dem Empfangspuffer und liefert sie als 32-Bit-Wert zurück.
Zuordnung:

Zeichen	Bits im Register
Erstes	Bit 24 ... 31
Zweites	Bit 16 ... 23
Drittes	Bit 8 ... 15
Viertes	Bit 0 ... 7

Modulregistereigenschaften		
Werte	-2.147.483.648 ... 2.147.483.647	
Zugriff	Lesen	Entnimmt 4 Zeichen aus dem Puffer
Wird wirksam	Wenn MR 14 > 3	

MR 19 **Fehlerzähler**

Dieses Register zeigt die Anzahl der festgestellten Fehler an.

Modulregistereigenschaften	
Werte	0 ... 2.147.483.647
Zugriff	Lesen/schreiben (löschen)

1.4 Programmierung

Einleitung

Dieses Kapitel beschreibt, wie die serielle Schnittstelle der Steuerung für die Verwendung als freiprogrammierbare serielle Schnittstelle konfiguriert wird und wie Zeichen über sie gesendet und empfangen werden.

Inhalt

Schnittstelle konfigurieren

Einleitung

Die Konfiguration der freiprogrammierbaren seriellen Schnittstelle geschieht über die Modulregister MR 1 bis MR 6.

Voraussetzungen

Diese Anleitung setzt voraus, dass die Verdrahtung zwischen der Steuerung und dem Gerät, mit dem kommuniziert werden soll, dem gewählten Schnittstellenstandard gemäß geschehen ist.

Schnittstelle konfigurieren

So konfigurieren Sie die freiprogrammierbare serielle Schnittstelle:

Schritt	Vorgehen
1	Beschreiben Sie MR 1 mit dem Wert 2.
2	Beschreiben Sie MR 2 bis MR 6 mit den gewünschten Übertragungsparametern.

Ergebnis: Die serielle Schnittstelle ist als freiprogrammierbare Schnittstelle eingestellt. Der Sendepuffer und Empfangspuffer sind gelöscht.

Zeichen senden

Einleitung

Das Senden von Zeichen geschieht, indem Sie das Zeichen in das Register *Sendepuffer* schreiben.

Voraussetzungen

Diese Anleitung setzt voraus, dass die freiprogrammierbare serielle Schnittstelle konfiguriert ist.

Zeichen senden

So senden Sie Zeichen über die freiprogrammierbare serielle Schnittstelle:

Schritt	Vorgehen
1	Prüfen Sie den Sendepufferfüllstand, ob Platz im Sendepuffer ist.
2	Wenn kein Platz im Sendepuffer ist, dann warten Sie, bis Platz vorhanden ist.
3	Schreiben Sie das zu sendende Zeichen in das Register *Sendepuffer*.

Ergebnis: Das Zeichen wird in den Sendepuffer eingetragen und von dort gesendet.

Texte senden

Einleitung

Eine einfache Möglichkeit, Texte auf der freiprogrammierbaren seriellen Schnittstelle zu senden, bietet die Umleitung der Befehle `DisplayText()` und `DisplayText2()` auf **Device 9**.

Voraussetzungen

Für diese Anleitung gelten folgende Voraussetzungen:

- Die freiprogrammierbare serielle Schnittstelle ist konfiguriert.
- Die detaillierte Beschreibung der Befehle `DisplayText()` und `DisplayText2()` ist bekannt (siehe JetSym-Online-Hilfe).

Einschränkungen

Bei der Umleitung der Befehle `DisplayText()` und `DisplayText2()` auf die freiprogrammierbare serielle Schnittstelle gelten folgende Einschränkungen:

- Die Cursor-Position wird nicht ausgewertet.
- Die Zeichen für 'Anzeige löschen' und 'lösche bis Zeilenende' haben keine besondere Bedeutung, sondern werden unverändert ausgegeben.

Texte senden

So senden Sie Texte über die freiprogrammierbare serielle Schnittstelle:

Schritt	Vorgehen
1	Verwenden Sie den Befehl `DisplayText()` oder `DisplayText2()`.
2	Geben Sie hierbei **Device 9** an.

Ergebnis: Der Task wartet an dem Befehl, bis alle Zeichen in den Sendepuffer eingetragen werden konnten.

Werte senden

Einleitung

Eine einfache Möglichkeit, Werte auf der freiprogrammierbaren seriellen Schnittstelle zu senden, bietet die Umleitung des Befehls `DisplayValue()` auf **Device 9**.

Voraussetzungen

Für diese Anleitung gelten folgende Voraussetzungen:

- Die freiprogrammierbare serielle Schnittstelle ist konfiguriert.
- Die detaillierte Beschreibung des Befehls `DisplayValue()` ist bekannt (siehe JetSym-Online-Hilfe).

Einschränkungen

Bei der Umleitung des Befehls `DisplayValue()` auf die freiprogrammierbare serielle Schnittstelle gilt folgende Einschränkung:

- Die Cursor-Position wird nicht ausgewertet.

Senden von Werten

So senden Sie Werte über die freiprogrammierbare serielle Schnittstelle:

Schritt	Vorgehen
1	Stellen Sie die Spezialregister für die Formatierung der Anzeige beim Befehl `DisplayValue()` auf die gewünschten Werte ein.
2	Verwenden Sie den Befehl `DisplayValue()`.
3	Geben Sie hierbei **Device 9** an.

Ergebnis: Der Task wartet an dem Befehl, bis alle Zeichen in den Sendepuffer eingetragen werden konnten.

Zeichen empfangen

Einleitung

Das Empfangen von Zeichen geschieht, indem Sie Zeichen aus dem Register *Empfangspuffer* lesen.

Voraussetzungen

Diese Anleitung setzt voraus, dass die freiprogrammierbare serielle Schnittstelle konfiguriert ist.

Zeichen empfangen

So empfangen Sie Zeichen über die freiprogrammierbare serielle Schnittstelle:

Schritt	Vorgehen
1	Warten Sie bis mindestens 1 Zeichen im Empfangspuffer ist, indem Sie den Empfangspufferfüllstand prüfen.
2	Lesen Sie das Zeichen aus dem Register *Empfangspuffer*.

Ergebnis: Das Zeichen wird aus dem Empfangspuffer entnommen.

Werte empfangen

Einleitung

Das Empfangen von Werten geschieht, indem Sie Zeichen aus den Registern MR 15 bis MR 18 *Empfangspufferregister* lesen.

Voraussetzungen

Diese Anleitung setzt voraus, dass die freiprogrammierbare serielle Schnittstelle konfiguriert ist.

Werte empfangen

So empfangen Sie Werte über die freiprogrammierbare serielle Schnittstelle:

Schritt	Vorgehen
1	Warten Sie bis mindestens 2 oder 4 Zeichen im Empfangspuffer sind, indem Sie den Empfangspufferfüllstand prüfen.
2	Lesen Sie den Wert aus den Registern MR 15 bis MR 18 *Empfangspuffer*.

Ergebnis: Die Zeichen werden aus dem Empfangspuffer entnommen.

2 Freiprogrammierbare IP-Schnittstelle

Die freiprogrammierbare IP-Schnittstelle

Die freiprogrammierbare IP-Schnittstelle bietet die Möglichkeit, beliebige Daten mit TCP/IP oder UDP/IP über die Ethernet-Schnittstelle des Geräts zu versenden oder zu empfangen. Die Verarbeitung der Daten geschieht dabei ganz im Anwendungsprogramm.

Anwendungen

Die freiprogrammierbare IP-Schnittstelle erlaubt dem Programmierer Daten über Ethernet-Verbindungen auszutauschen, die sich nicht Standardprotokollen - wie z. B. FTP, HTTP, JetIP oder Modbus/TCP - bedienen. Folgende Anwendungen sind dabei möglich:

- Server
- Client
- TCP/IP
- UDP/IP

Voraussetzungen an den Programmierer

Die Funktionalität der freiprogrammierbaren IP-Schnittstelle setzt folgende Kenntnisse der Datenübertragung über IP-Netzwerke voraus:

- IP-Adressierung (z. B. IP-Adresse, Port-Nummer, Subnetzmaske)
- TCP (z. B. Verbindungsaufbau / -abbau, Datastream, Datensicherung)
- UDP (z. B. Datagram)

Einschränkungen

Für die Kommunikation über die freiprogrammierbare IP-Schnittstelle darf der Programmierer keine Ports verwenden, die schon das Betriebssystem verwendet. Verwenden Sie deshalb folgende Ports nicht:

Protokoll	Port-Nummer	Standardwert	Benutzer
TCP	Abhängig vom FTP-Client	20	FTP-Server (Daten)
TCP	21		FTP-Server (Steuerung)
TCP	23		System-Logger
TCP	80		HTTP-Server
TCP	Aus der Datei /EMAIL/email.ini	25, 110	E-Mail-Client
TCP	502		Modbus/TCP-Server
TCP, UDP	1024 - 2047		Diverse
TCP, UDP	IP-Konfiguration	50000, 50001	JetIP
TCP	IP-Konfiguration	52000	Debug-Server

Inhalt

2.1 Programmierung

Einleitung

Bei der freiprogrammierbaren IP-Schnittstelle werden die Daten über TCP/IP- oder UDP/IP-Verbindungen zwischen dem Anwendungsprogramm und einem Netzwerkteilnehmer ausgetauscht. Verwendung finden hierbei Aufrufe von Funktionen, die im Sprachumfang des Geräts enthalten sind. Führen Sie zur Programmierung folgende Schritte aus:

Schritt	Vorgehen
1	Freiprogrammierbare IP-Schnittstelle initialisieren
2	Verbindungen öffnen
3	Daten übertragen
4	Verbindungen schließen

Technische Daten

Die technischen Daten der freiprogrammierbaren IP-Schnittstelle:

Funktion	Beschreibung
Anzahl Verbindungen	20
Maximale Datengröße	4.000 Byte
Anzahl Empfangspuffer pro Verbindung	4

Einschränkungen

Während das Gerät gerade eine der Funktionen der freiprogrammierbaren IP-Schnittstelle bearbeitet, dürfen die Tasks, die die Funktionen aufgerufen haben, nicht durch `TaskBreak` angehalten oder durch `TaskRestart` neu gestartet werden.

Dies kann zu folgenden Fehlern führen:

- Verbindungen öffnen sich nicht
- Datenverlust beim Senden oder Empfangen
- Verbindungen bleiben ungewollt offen
- Verbindungen werden ungewollt geschlossen

Inhalt

Initialisieren der freiprogrammierbaren IP-Schnittstelle

Einleitung

Die Initialisierung muss mindestens einmal bei jedem Anwendungsprogrammstart ausgeführt werden.

Funktionsdeklaration

```
Function ConnectionInitialize():Int;
```

Rückgabewert

Folgender Rückgabewert ist möglich:

Rückgabewert	
0	Immer

Verwenden der Funktion

So wird die Funktion verwendet und der Rückgabewert einer Variablen zur weiteren Auswertung zugewiesen:

```
Result := ConnectionInitialize();
```

Funktionsweise

Das Gerät arbeitet die Funktion in folgenden Stufen ab:

Stufe	Beschreibung
1	Das Gerät schließt alle geöffneten Verbindungen der freiprogrammierbaren IP-Schnittstelle.
2	Das Gerät initialisiert alle betriebssysteminternen Datenstrukturen der freiprogrammierbaren IP-Schnittstelle.

Verwandte Themen

- **Verbindung öffnen** (siehe Seite 35)
- **Verbindung schließen** (siehe Seite 44)
- **Daten senden** (siehe Seite 39)
- **Daten empfangen** (siehe Seite 41)

Verbindung öffnen

Einleitung

Bevor Daten gesendet oder empfangen werden können, muss eine Verbindung geöffnet werden. Dabei gilt Folgendes zu unterscheiden:

- Welches Transportprotokoll (TCP oder UDP) ist zu verwenden?
- Ist ein Client oder ein Server einzurichten?

Funktionsdeklaration

```
Function ConnectionCreate(ClientServerType:Int,
                          IPType:Int,
                          IPAddr:Int,
                          IPPort:Int,
                          Timeout:Int):Int;
```

Funktionsparameter

Beschreibung der Funktionsparameter:

Parameter	Wert	Bemerkung
`ClientServerType`	Client = 1 = CONNTYPE_CLIENT Server = 2 = CONNTYPE_SERVER	
`IPType`	UDP/IP = 1 = IPTYPE_UDP TCP/IP = 2 = IPTYPE_TCP	
`IPAddr`	Gültige IP-Adresse	Nur bei TCP/IP-Client erforderlich
`IPPort`	Gültige IP-Port-Nummer	Wird bei UDP/IP-Client ignoriert
`Timeout`	0 ... 1.073.741.824 [ms]	0 = unendlich

Rückgabewert

Bei einem positiven Rückgabewert konnte die Verbindung geöffnet werden. Bei einem negativen Rückgabewert ist ein Fehler aufgetreten und die Verbindung konnte nicht geöffnet werden.

Rückgabewert	
> 0	Ein positiver Rückgabewert muss in einer Variablen gesichert werden. Er muss als Handle beim Aufruf der Funktionen für Daten senden, Daten empfangen und Verbindung schließen mitgegeben werden.
-1	Fehler beim Verbindungsaufbau
-2	Interner Fehler
-3	Ungültiger Parameter
-8	Zeitüberschreitung

Verwenden der Funktion bei einem TCP/IP-Client

Wenn ein Client eine TCP/IP-Verbindung zu einem Server aufbauen soll, können Sie die Funktion so aufrufen und den Rückgabewert einer Variablen zur weiteren Auswertung zuweisen:

```
Result := ConnectionCreate(CONNTYPE_CLIENT,
                           IPTYPE_TCP,
                           IP#192.168.75.123,
                           46000,
                           T#10s);
```

Funktionsweise bei einem TCP/IP-Client

Der Task bleibt bei der Programmzeile stehen, bis die Verbindung aufgebaut wurde oder der angegebene Timeout abgelaufen ist. Die Funktion läuft in folgenden Stufen ab:

Stufe	Beschreibung	
1	Das Gerät versucht eine TCP/IP-Verbindung zu Port 46000 zu dem Netzwerkteilnehmer mit der IP-Adresse 192.168.75.123 aufzubauen.	
2	**Wenn ...**	**... dann ...**
	... der Netzwerkteilnehmer die Verbindung akzeptiert hat,	... wird die Funktion beendet und ein positiver Wert als Handle zum weiteren Zugriff auf die Verbindung zurückgeliefert.
	... die Verbindung nicht aufgebaut werden konnte und die Timeout-Zeit von 10 Sekunden noch nicht abgelaufen ist,	... wird mit Stufe 1 fortgefahren.
	... ein Fehler aufgetreten ist oder der Timeout abgelaufen ist,	... wird die Funktion beendet und ein negativer Wert zurückgeliefert.

Verwenden der Funktion bei einem TCP/IP-Server

Wenn ein Server eine TCP/IP-Verbindung mit einem Client aufbauen soll, können Sie die Funktion so aufrufen und den Rückgabewert einer Variablen zur weiteren Auswertung zuweisen:

```
Result := ConnectionCreate(CONNTYPE_SERVER,
                           IPTYPE_TCP,
                           0,
                           46000,
                           T#100s);
```

Funktionsweise bei einem TCP/IP-Server

Der Task bleibt bei der Programmzeile stehen, bis die Verbindung aufgebaut wurde oder der angegebene Timeout abgelaufen ist. Die Funktion läuft in folgenden Stufen ab:

Stufe	Beschreibung	
1	Das Gerät richtet den TCP/IP-Port 46000 zum Empfang von Verbindungsanfragen ein.	
2	**Wenn ...**	**... dann ...**
	... der Client eines Netzwerkteilnehmers eine Verbindung aufgebaut hat,	... werden keine weiteren Verbindungsanfragen auf diesen Port akzeptiert, die Funktion beendet und ein positiver Wert als Handle zum weiteren Zugriff auf die Verbindung zurückgeliefert.
	... die Verbindung nicht aufgebaut werden konnte und die Timeout-Zeit von 100 Sekunden noch nicht abgelaufen ist,	... wird auf einen Verbindungsaufbau gewartet.
	... ein Fehler aufgetreten ist oder der Timeout abgelaufen ist,	... wird die Funktion beendet und ein negativer Wert zurückgeliefert.

Verwenden der Funktion bei einem UDP/IP-Client

Wenn ein Client eine UDP/IP-Verbindung aufbauen soll, können Sie die Funktion so aufrufen und den Rückgabewert einer Variablen zur weiteren Auswertung zuweisen:

```
Result := ConnectionCreate(CONNTYPE_CLIENT,
                           IPTYPE_UDP,
                           0,
                           0,
                           0);
```

Funktionsweise bei einem UDP/IP-Client

UDP ist eine verbindungslose Art der Kommunikation. Deshalb öffnet das Gerät nur einen Kommunikationskanal, über den Daten an einen Netzwerkteilnehmer gesendet werden. Die Funktion läuft in folgenden Stufen ab:

Stufe	Beschreibung	
1	Das Gerät richtet einen UDP/IP-Verbindungskanal zum Senden von Daten ein.	
2	**Wenn ...**	**... dann ...**
	... kein Fehler aufgetreten ist,	... wird die Funktion beendet und ein positiver Wert als Handle zum weiteren Zugriff auf die Verbindung zurückgeliefert.
	... ein Fehler aufgetreten ist,	... wird die Funktion beendet und ein negativer Wert zurückgeliefert.

Verwenden der Funktion bei einem UDP/IP-Server

Wenn ein Server eine UDP/IP-Verbindung aufbauen soll, können Sie die Funktion so aufrufen und den Rückgabewert einer Variablen zur weiteren Auswertung zuweisen:

```
Result := ConnectionCreate(CONNTYPE_SERVER,
                           IPTYPE_UDP,
                           0,
                           46000,
                           0);
```

Funktionsweise bei einem UDP/IP-Server

UDP ist eine verbindungslose Art der Kommunikation. Deshalb öffnet das Gerät nur einen Kommunikationskanal über den Daten von einem Netzwerkteilnehmer empfangen werden. Die Funktion läuft in folgenden Stufen ab:

Stufe	Beschreibung	
1	Das Gerät richtet einen UDP/IP-Verbindungskanal am Port 46000 zum Empfangen von Daten ein.	
2	**Wenn ...**	**... dann ...**
	... kein Fehler aufgetreten ist,	... wird die Funktion beendet und ein positiver Wert als Handle zum weiteren Zugriff auf die Verbindung zurückgeliefert.
	... ein Fehler aufgetreten ist,	... wird die Funktion beendet und ein negativer Wert zurückgeliefert.

Verwandte Themen

- **Verbindung schließen** (siehe Seite 44)
- **Daten senden** (siehe Seite 39)
- **Daten empfangen** (siehe Seite 41)
- **Initialisieren der freiprogrammierbaren IP-Schnittstelle** (siehe Seite 34)

Daten senden

Einleitung

Daten können über eine zuvor geöffnete Verbindung gesendet werden.

Funktionsdeklaration

```
Function ConnectionSendData(IPConnection:Int,
                            IPAddr:Int,
                            IPPort:Int,
                            Const Ref SendData,
                            DataLen:Int):Int;
```

Funktionsparameter

Beschreibung der Funktionsparameter:

Parameter	Wert	Bemerkung
`IPConnection`	Handle	Rückgabewert der Funktion `ConnectionCreate()`
`IPAddr`	Gültige IP-Adresse	Nur bei UDP/IP-Client erforderlich
`IPPort`	Gültige IP-Port-Nummer	Nur bei UDP/IP-Client erforderlich
`SendData`	Adresse des zu sendenden Datenblocks	
`DataLen`	1 ... 4.000	Länge des Datenblocks in Byte

Rückgabewert

Folgende Rückgabewerte sind möglich:

Rückgabewert	
0	Daten erfolgreich gesendet
-1	Fehler beim Senden, z. B. Verbindung abgebrochen
-3	Ungültiges Handle, z. B. senden über einen UDP/IP-Server

Verwenden der Funktion bei einer TCP/IP-Verbindung

Wenn Daten über eine TCP/IP-Verbindung gesendet werden sollen, können Sie die Funktion so aufrufen und den Rückgabewert einer Variablen zur weiteren Auswertung zuweisen:

```
Result := ConnectionSendData(hConnection,
                             0,
                             0,
                             SendBuffer,
                             SendLen);
```

Funktionsweise bei einer TCP/IP-Verbindung

Bei TCP/IP werden die Daten über eine Verbindung übertragen, die vorher geöffnet wurde. Deshalb ist die Angabe von IP-Adresse und IP-Port-Nummer nicht mehr erforderlich und werden in der Funktion ignoriert.

Der Task bleibt in folgenden Fällen bei diesem Funktionsaufruf stehen:

- Die Daten wurden gesendet und ihr Empfang bestätigt.
- Ein Fehler ist aufgetreten.

Verwenden der Funktion bei einer UDP/IP-Verbindung

Wenn Daten bei einem Client über eine UDP/IP-Verbindung gesendet werden sollen, können Sie die Funktion so aufrufen und den Rückgabewert einer Variablen zur weiteren Auswertung zuweisen:

```
Result := ConnectionSendData(hConnection,
                             IP#192.168.75.123,
                             46000,
                             SendBuffer,
                             SendLen);
```

Funktionsweise bei einer UDP/IP-Verbindung

Da bei UDP/IP keine Verbindung zwischen zwei Netzwerkteilnehmern besteht, können Daten bei jedem Funktionsaufruf an einen anderen Teilnehmer oder anderen Port gesendet werden. Der Task bleibt bei diesem Funktionsaufruf stehen, bis die Daten gesendet wurden.

Sie erhalten keine Bestätigung, dass der andere Netzwerkteilnehmer die Daten empfangen hat.

UDP/IP-Client und -Server

Über eine UDP/IP-Client-Verbindung kann nur gesendet werden. Der Sende-Port wird vom Betriebssystem festgelegt.

Über eine UDP/IP-Server-Verbindung kann gesendet und empfangen werden. Als Sende-Port wird der bei der Verbindungseröffnung angegebene Port verwendet.

Verwandte Themen

- **Initialisieren der freiprogrammierbaren IP-Schnittstelle** (siehe Seite 34)
- **Verbindung öffnen** (siehe Seite 35)
- **Verbindung schließen** (siehe Seite 44)
- **Daten empfangen** (siehe Seite 41)

Daten empfangen

Einleitung

Daten können über eine zuvor geöffnete TCP/IP-Verbindung oder über die UDP/IP-Verbindung eines Servers empfangen werden.

Über die UDP/IP-Verbindung eines Clients können keine Daten empfangen, sondern nur gesendet werden.

Einschränkungen

Wenn Datenpakete über das Netzwerk empfangen werden, müssen diese vom Anwendungsprogramm abgeholt werden. Vom Betriebssystem der Steuerung werden pro Verbindung maximal vier Pakete zwischengespeichert. Alle weiteren Pakete werden verworfen.

Funktionsdeklaration

```
Function ConnectionReceiveData(IPConnection:Int,
                               Ref IPAddr:Int,
                               Ref IPPort:Int,
                               Ref ReceiveData,
                               DataLen:Int,
                               Timeout:Int):Int;
```

Funktionsparameter

Beschreibung der Funktionsparameter:

Parameter	Wert	Bemerkung
`IPConnection`	Handle	Rückgabewert der Funktion `ConnectionCreate()`
`IPAddr`	Adresse einer Variablen, um die IP-Adresse des Senders zu speichern	Nur bei UDP/IP-Server erforderlich
`IPPort`	Adresse einer Variablen, um die IP-Port-Nummer des Senders zu speichern	Nur bei UDP/IP-Server erforderlich
`ReceiveData`	Adresse des Empfangsdatenblocks	
`DataLen`	1 ... 4.000	Maximale Länge des Datenblocks in Byte
`Timeout`	0 ... 1.073.741.824 [ms]	0 = unendlich

Rückgabewert

Folgende Rückgabewerte sind möglich:

Rückgabewert	
> 0	Anzahl empfangener Datenbyte
-1	Fehler beim Empfang, z. B. Verbindung abgebrochen
-3	Ungültiges Handle, z. B. empfangen über einen UDP/IP-Client
-8	Timeout

Verwenden der Funktion bei einer TCP/IP-Verbindung

Wenn Daten über eine TCP/IP-Verbindung empfangen werden sollen, können Sie die Funktion so aufrufen und den Rückgabewert einer Variablen zur weiteren Auswertung zuweisen:

```
Result := ConnectionReceiveData(hConnection,
                                Dummy,
                                Dummy,
                                ReceiveBuffer,
                                sizeof(ReceiveBuffer),
                                T#10s);
```

Funktionsweise bei einer TCP/IP-Verbindung

Bei TCP/IP werden die Daten über eine Verbindung übertragen, die vorher geöffnet wurde. Deshalb ist die Angabe von IP-Adresse und IP-Port-Nummer nicht mehr erforderlich und werden in der Funktion ignoriert.

Der Task bleibt in folgenden Fällen bei diesem Funktionsaufruf stehen:

- Die Daten wurden empfangen.
- Ein Fehler ist aufgetreten.

Die Daten werden bei einer TCP/IP-Verbindung als Datenstrom übertragen.

Das Gerät arbeitet die Funktion in folgenden Stufen ab:

Stufe	Beschreibung	
1	Das Gerät wartet, bis Daten empfangen wurden, längstens jedoch die mitgegebene Timeout-Zeit.	
2	**Wenn ...**	**... dann ...**
	... der Timeout abgelaufen oder die Verbindung geschlossen wurde,	... wird die Funktion mit einer Fehlermeldung verlassen.
	... Daten empfangen wurden,	... werden sie in den mitgegebenen Empfangspuffer kopiert (höchstens jedoch bis zur mitgegebenen Anzahl) und bei Stufe 3 fortgefahren.
3	**Wenn ...**	**... dann ...**
	... mehr Daten empfangen wurden als in den Empfangspuffer kopiert werden konnten,	... werden diese vom Gerät gepuffert und können durch weitere Funktionsaufrufe abgeholt werden.
4	Die Funktion wird verlassen und die Anzahl der in den Empfangspuffer kopierten Daten zurückgegeben.	

Verwenden der Funktion bei einem UDP/IP-Server

Wenn Daten bei einem Server über eine UDP/IP-Verbindung empfangen werden sollen, können Sie die Funktion so aufrufen und den Rückgabewert einer Variablen zur weiteren Auswertung zuweisen:

```
Result := ConnectionReceiveData(hConnection,
                                IPAddr,
                                IPPort,
                                ReceiveBuffer,
                                sizeof(ReceiveBuffer),
                                T#10s);
```

Funktionsweise bei einem UDP/IP-Server

Der Task bleibt in folgenden Fällen bei diesem Funktionsaufruf stehen:

- Alle Daten wurden empfangen.
- Ein Fehler ist aufgetreten.

Die Daten werden bei einer UDP/IP-Verbindung als Datagramme übertragen. Das Gerät arbeitet die Funktion in folgenden Stufen ab:

Stufe	Beschreibung	
1	Das Gerät wartet bis alle Daten eines Datagramms empfangen wurden, längstens jedoch die mitgegebene Timeout-Zeit.	
2	**Wenn ...**	**... dann ...**
	... der Timeout abgelaufen oder die Verbindung geschlossen wurde,	... wird die Funktion mit einer Fehlermeldung verlassen.
	... Daten empfangen wurden,	... werden sie in den mitgegebenen Empfangspuffer kopiert (höchstens jedoch bis zur mitgegebenen Anzahl) und bei Stufe 3 fortgefahren.
3	**Wenn ...**	**... dann ...**
	... mehr Daten empfangen wurden als in den Empfangspuffer kopiert werden konnten, also das gesendete Datagramm zu groß ist,	... werden diese verworfen.
4	IP-Adresse und IP-Port-Nummer des Senders werden in die mitgegebenen Variablen übertragen.	
5	Die Funktion wird verlassen und die Anzahl der in den Empfangspuffer kopierten Daten zurückgegeben.	

Verwandte Themen

- **Initialisieren der freiprogrammierbaren IP-Schnittstelle** (siehe Seite 34)
- **Verbindung öffnen** (siehe Seite 35)
- **Verbindung schließen** (siehe Seite 44)
- **Daten senden** (siehe Seite 39)

Verbindung schließen

Einleitung

Schließen Sie nicht mehr benötigte Verbindungen, da die Anzahl gleichzeitig geöffneter Verbindungen begrenzt ist.

Funktionsdeklaration

```
Function ConnectionDelete(IPConnection:Int):Int;
```

Funktionsparameter

Beschreibung der Funktionsparameter:

Parameter	Wert	Bemerkung
`IPConnection`	Handle	Rückgabewert der Funktion `ConnectionCreate()`

Rückgabewert

Folgende Rückgabewerte sind möglich:

Rückgabewert	
0	Verbindung geschlossen und gelöscht
-1	Ungültiges Handle

Verwenden der Funktion

So können Sie die Funktion aufrufen und den Rückgabewert einer Variablen zur weiteren Auswertung zuweisen:

```
Result := ConnectionDelete(hConnection);
```

Verwandte Themen

- **Verbindung öffnen** (siehe Seite 35)
- **Daten senden** (siehe Seite 39)
- **Daten empfangen** (siehe Seite 41)
- **Initialisieren der freiprogrammierbaren IP-Schnittstelle** (siehe Seite 34)

2.2 Register

Einleitung

Dieses Kapitel beschreibt die Register des Geräts, in denen die aktuelle Verbindungsliste der freiprogrammierbaren IP-Schnittstelle enthalten ist. Die Register können zu Debug- oder Diagnosezwecken verwendet werden. Weitere Funktionen, wie Verbindung öffnen oder schließen, können hierüber nicht ausgelöst werden.

Inhalt

Registernummern

Einleitung

Die Daten jeweils einer Verbindung erscheinen in den Registern eines zusammenhängenden Registerblocks. Die Basisregisternummer dieses Blocks ist steuerungsabhängig.

Registernummern

Basisregisternummer	Registernummern
350000	350000 ... 350007

Registernummer ermitteln

In diesem Kapitel ist jeweils nur die letzte Ziffer der Registernummer angegeben, z. B. MR 1. Addieren Sie zu dieser Ziffer die Basisregisternummer des jeweiligen Geräts, z. B. 350000, um die vollständige Registernummer zu ermitteln.

Registerübersicht

Register	Beschreibung
MR 0	Verbindungsauswahl
MR 1	Verbindungstyp
MR 2	Transportprotokoll
MR 3	IP-Adresse
MR 4	IP-Port-Nummer
MR 5	Zustand
MR 6	Anzahl gesendeter Bytes
MR 7	Anzahl empfangener Bytes
MR 8	Anzahl verworfener Bytes
MR 9	Anzahl verworfener Pakete

Registerbeschreibung

Einleitung

Das Betriebssystem verwaltet die geöffneten Verbindungen in einer Liste. Mit Hilfe des Modulregisters MR 0 *Verbindungsauswahl* werden die Verbindungsdaten einer Verbindung in die anderen Register des Registerblocks kopiert.

MR 0

Verbindungsauswahl

Die Anwahl der Verbindungen geschieht, indem Werte in dieses Register geschrieben werden. Dieses Register zeigt, ob die folgenden Register Verbindungsdaten enthalten.

Modulregistereigenschaften		
Werte lesen	0	Verbindung vorhanden
	-1	Verbindung nicht vorhanden

Modulregistereigenschaften		
Werte schreiben	0	Erste Verbindung in der Liste anwählen
	> 0	Nächste Verbindung in der Liste anwählen
	< 0	Vorherige Verbindung in der Liste anwählen

MR 1

Verbindungstyp

Der Wert in diesem Register zeigt an, ob es sich um eine Client- oder eine Serververbindung handelt.

Modulregistereigenschaften		
Werte	1	Client
	2	Server

MR 2

Transportprotokoll

Der Wert in diesem Register zeigt die Art des Transportprotokolls an, UDP oder TCP.

Modulregistereigenschaften		
Werte	1	UDP
	2	TCP

MR 3

IP-Adresse

Der Wert in diesem Register zeigt die konfigurierte IP-Adresse an.

Modulregistereigenschaften	
Werte	0.0.0.0 ... 255.255.255.255

MR 4

IP-Port-Nummer

Der Wert in diesem Register zeigt die konfigurierte IP-Port-Nummer an.

Modulregistereigenschaften	
Werte	0 ... 65.535

MR 5

Zustand

Der Wert in diesem Register zeigt an, in welchem Zustand sich die Verbindung befindet.

Modulregistereigenschaften		
Werte	0	Verbindung geschlossen
	1	Verbindung wird geöffnet
	2	Verbindung ist geöffnet
	3	TCP/IP-Server: Warten auf Verbindungsanfrage von Client
	4	Interne Verwendung

MR 6

Anzahl gesendeter Bytes

Der Wert in diesem Register zeigt die Anzahl der über diese Verbindung gesendeten Datenbytes an. Da es sich um ein vorzeichenbehaftetes 32-Bit-Register handelt und die gesendeten Bytes jeweils hinzuaddiert werden, können Zahlenüberläufe vom positiven zum negativen Maximalwert auftreten.

Modulregistereigenschaften	
Werte	-2.147.483.648 ... 2.147.483.647

MR 7 **Anzahl empfangener Bytes**

Der Wert in diesem Register zeigt die Anzahl der über diese Verbindung empfangenen Datenbytes an. Da es sich um ein vorzeichenbehaftetes 32-Bit-Register handelt und die empfangenen Bytes jeweils hinzuaddiert werden, können Zahlenüberläufe vom positiven zum negativen Maximalwert auftreten.

Modulregistereigenschaften	
Werte	-2.147.483.648 ... 2.147.483.647

MR 8 **Anzahl verworfener Bytes**

Der Wert in diesem Register zeigt die Anzahl der Datenbytes an, die nicht empfangen werden konnten, weil das Anwendungsprogramm die zwischengespeicherten Datenbytes nicht abgeholt hat.

Modulregistereigenschaften	
Werte	0 ... 2.147.483.647

MR 9 **Anzahl verworfener Pakete**

Der Wert in diesem Register zeigt die Anzahl der Datenpakete an, die nicht empfangen werden konnten, weil das Anwendungsprogramm die zwischengespeicherten Datenpakete nicht abgeholt hat.

Modulregistereigenschaften	
Werte	0 ... 2.147.483.647

3 Freiprogrammierbare CAN-Prim-Schnittstelle

Die CAN-Prim-Schnittstelle

Die freiprogrammierbare CAN-Prim-Schnittstelle bietet die Möglichkeit, beliebige CAN-Nachrichten zu senden und zu empfangen. Die CAN-Nachrichten werden vollständig im Anwendungsprogramm verarbeitet.

Vorteil von CAN-Prim

Das Feature ist nicht nur für CANopen®-Geräte. Hier hat der Kunde die Möglichkeit mit Fremdgeräten mit CAN-basiertem Protokoll zu kommunizieren.

Anwendungen

Folgende Anwendungen sind mit der freiprogrammierbaren CAN-Prim-Schnittstelle möglich:

- Ansteuerung von Geräten mit CAN-Schnittstelle mit proprietären Protokollen
- Ansteuerung von CANopen®-fähigen Geräten
- ...

Voraussetzungen an den Programmierer

Die Funktionalität der freiprogrammierbaren CAN-Prim-Schnittstelle setzt grundlegende Kenntnisse des Controller Area Networks CAN voraus. Dazu zählen:

- Aufbau einer CAN-Nachricht
- CANopen®-Dienste

Voraussetzungen an die Hardware

Als Hardware wird eine JetControl vorausgesetzt, die über eine CAN-Schnittstelle und/oder einen JX2-Systembus verfügt.

Registernummern bei JC-3xx

Die Registernummer bei JC-3xx besteht aus folgenden Elementen:

2	0	0	0	z	z	z	z	z

Element	Bedeutung	Wertebereich
zzzzz	Modulregisternummer	2029, 2077 10500 ...10599

Registernummern bei JC-9xx

Die Registernummer für das Submodul JX6-SB(-I) bei JC-9xx besteht aus folgenden Elementen:

2	0	S	J	z	z	z	z	z

Element	Bedeutung	Wertebereich
S	Nummer der Trägerplatine	1 ... 3
J	Nummer des Submoduls JX6-SB-I auf der Trägerplatine	1 ... 2
zzzzz	Modulregisternummer	2029, 2077 10500 ...10599

Inhalt

Einschränkungen der CAN-Prim-Schnittstelle

Einschränkung der anschließbaren Module

Bei Verwendung der freiprogrammierbaren CAN-Prim-Schnittstelle gelten die folgenden Einschränkungen:

- Wenn 29-Bit-CAN-Identifier benutzt werden, müssen die Seriennummern der nicht intelligenten JX2-I/O-Module mit 2 beginnen.

CAN-Nachrichten in der Boot-Phase

Zwischen dem Einschalten des Geräts und dem Start des Anwendungsprogramms (Bootphase des JX2-Systembusses) dürfen die angeschlossenen CAN-Module keine CAN-Nachrichten versenden.

Zeitverhalten

Der zeitliche Abstand zwischen zwei CAN-Nachrichten, die über die CAN-Schnittstelle empfangen wurden, sollte mindestens 10 ms betragen. Bei kürzerem Abstand kann das Gerät nicht alle CAN-Nachrichten für den CAN-Prim-Empfang verarbeiten.

Wenn mehrere CAN-Nachrichten mit derselben CAN-ID empfangen werden sollen, ist eine hohe Reaktions- und Verarbeitungsgeschwindigkeit des Anwendungsprogramms erforderlich. So vermeiden Sie Pufferüberläufe (Overrun-Bit). Anpassungen am Taskwechselverfahren und der Taskpriorisierung (TASKPRIORITY) garantieren nicht zwangsläufig, dass alle CAN-Nachrichten verarbeitet werden können.

Reservierte CAN-IDs

Beim gleichzeitigen Betrieb von Erweiterungsmodulen am JX2-Systembus und der CAN-Prim-Schnittstelle, sind bestimmte CAN-IDs reserviert.

Module am JX2-Systembus	Reservierte CAN-IDs
Bei allen Modulen	0x100, 0x701, 0x702, 0x703, 0x704, 0x705, 0x706, 0x707, 0x708, 0x709, 0x70A, 0x732, 0x733, 0x734, 0x735, 0x736, 0x737, 0x738, 0x739, 0x73A, 0x73B, 0x746, 0x747, 0x748, 0x749, 0x74A, 0x74B, 0x74C, 0x74D, 0x74E, 0x74F
JX2-I/O Module	0x180, 0x181, 0x182, 0x183, 0x184, 0x185, 0x186, 0x187, 0x188, 0x189, 0x18A, 0x18B, 0x18C, 0x18D, 0x18E, 0x18F, 0x190, 0x191, 0x192, 0x193, 0x194, 0x195, 0x196, 0x197, 0x198, 0x199, 0x19A, 0x19B, 0x19C, 0x19D, 0x19E, 0x19F, 0x1A0, 0x1A1, 0x1A2, 0x1A3, 0x1A4, 0x1A5, 0x1A6, 0x1A7, 0x1A8, 0x1A9, 0x1AA, 0x1AB, 0x1AC, 0x1AD, 0x1AE, 0x1AF, 0x1B0, 0x1B1, 0x1B2, 0x1B3, 0x1B4, 0x1B5, 0x1B6, 0x1B7, 0x1B8, 0x1B9, 0x1BA, 0x1BB, 0x1BC, 0x1BD, 0x1BE, 0x1BF, 0x380, 0x381, 0x382, 0x383, 0x384, 0x385, 0x386, 0x387, 0x388, 0x389, 0x38A, 0x38B, 0x38C, 0x38D, 0x38E, 0x38F, 0x390, 0x391, 0x392, 0x393, 0x394, 0x395, 0x396, 0x397, 0x398, 0x399, 0x39A, 0x39B, 0x39C, 0x39D, 0x39E, 0x39F, 0x3A0, 0x3A1, 0x3A2, 0x3A3, 0x3A4, 0x3A5, 0x3A6, 0x3A7, 0x3A8, 0x3A9, 0x3AA, 0x3AB, 0x3AC, 0x3AD, 0x3AE, 0x3AF, 0x3B0, 0x3B1, 0x3B2, 0x3B3, 0x3B4, 0x3B5, 0x3B6, 0x3B7, 0x3B8, 0x3B9, 0x3BA, 0x3BB, 0x3BE, 0x3BF

Module am JX2-Systembus	Reservierte CAN-IDs
JX2-Slave-Module	0x081, 0x082, 0x083, 0x084, 0x085, 0x086, 0x087, 0x088, 0x089, 0x08A, 0x08B, 0x08C, 0x08D, 0x08E, 0x08F, 0x090, 0x09F, 0x0A0, 0x0A1, 0x0A2, 0x0A3, 0x0A4, 0x0A5, 0x0A6, 0x0A7, 0x0A8, 0x0A9, 0x0AA, 0x0AB, 0x0AC, 0x0AD, 0x0AE, 0x0AF, 0x161, 0x162, 0x163, 0x164, 0x165, 0x166, 0x167, 0x168, 0x169, 0x16A, 0x16B, 0x16C, 0x16D, 0x16E, 0x16F, 0x1D1, 0x1D2, 0x1D3, 0x1D4, 0x1D5, 0x1D6, 0x1D7, 0x1D8, 0x1D9, 0x1DA, 0x1DB, 0x1DC,0x1DD, 0x1DE, 0x1DF
JX3-Module	0x180, 0x181, 0x182, 0x183, 0x184, 0x185, 0x186, 0x187, 0x188, 0x189, 0x18A, 0x18B, 0x18C, 0x18D, 0x18E, 0x18F, 0x190, 0x191, 0x192, 0x193, 0x194, 0x195, 0x196, 0x197, 0x198, 0x199, 0x19A, 0x19B, 0x19C, 0x19D, 0x19E, 0x19F, 0x1A0, 0x1A1, 0x1A2, 0x1A3, 0x1A4, 0x1A5, 0x1A6, 0x1A7, 0x1A8, 0x1A9, 0x1AA, 0x1AB, 0x1AC, 0x1AD, 0x1AE, 0x1AF, 0x1B0, 0x1B1, 0x1B2, 0x1B3, 0x1B4, 0x1B5, 0x1B6, 0x1B7, 0x1B8, 0x1B9, 0x1BA, 0x1BB, 0x1BC, 0x1BD, 0x1BE, 0x1BF, 0x320, 0x321, 0x322, 0x323, 0x324, 0x325, 0x326, 0x327, 0x328, 0x329, 0x32A, 0x32B, 0x32C, 0x32D, 0x32E, 0x32F, 0x330, 0x331, 0x332, 0x333, 0x334, 0x335, 0x336, 0x337, 0x338, 0x339, 0x33A, 0x33B, 0x33C, 0x33D, 0x33E, 0x380, 0x381, 0x382, 0x383, 0x384, 0x385, 0x386, 0x387, 0x388, 0x389, 0x38A, 0x38B, 0x38C, 0x38D, 0x38E, 0x38F, 0x390, 0x391, 0x392, 0x393, 0x394, 0x395, 0x396, 0x397, 0x398, 0x399, 0x39A, 0x39B, 0x39C, 0x39D, 0x39E, 0x39F, 0x3A0, 0x3A1, 0x3A2, 0x3A3, 0x3A4, 0x3A5, 0x3A6, 0x3A7, 0x3A8, 0x3A9, 0x3AA, 0x3AB, 0x3AC, 0x3AD, 0x3AE, 0x3AF, 0x3B0, 0x3B1, 0x3B2, 0x3B3, 0x3B4, 0x3B5, 0x3B6, 0x3B7, 0x3B8, 0x3B9, 0x3BA, 0x3BB, 0x3BE, 0x3BF, 0x3E0, 0x3E1, 0x3E2, 0x3E3, 0x3E4, 0x3E5, 0x3E6, 0x3E7, 0x3E8, 0x3E9, 0x3EA, 0x3EB, 0x3EC, 0x3ED, 0x3EE, 0x3EF, 0x3F0, 0x3F1, 0x3F2, 0x3F3, 0x3F4, 0x3F5, 0x3F6, 0x3F7, 0x3F8, 0x3F9, 0x3FA, 0x3FB, 0x3FC, 0x3FD, 0x3FE

Module am JX2-Systembus	Reservierte CAN-IDs
JX-SIO und CANopen®-Module	0x1C6, 0x1C7, 0x1C8, 0x1C9, 0x1CA, 0x1CB, 0x1CC, 0x1CD, 0x1CE, 0x1CF, 0x246, 0x247, 0x248, 0x249, 0x24A, 0x24B, 0x24C, 0x24D, 0x24E, 0x24F, 0x2C6, 0x2C7, 0x2C8, 0x2C9, 0x2CA, 0x2CB, 0x2CC, 0x2CD, 0x2CE, 0x2CF, 0x346, 0x347, 0x348, 0x349, 0x34A, 0x34B, 0x34C, 0x34D, 0x34E, 0x34F, 0x3C6, 0x3C7, 0x3C8, 0x3C9, 0x3CA, 0x3CB, 0x3CC, 0x3CD, 0x3CE, 0x3CF, 0x446, 0x447, 0x448, 0x449, 0x44A, 0x44B, 0x44C, 0x44D, 0x44E, 0x44F, 0x4C6, 0x4C7, 0x4C8, 0x4C9, 0x4CA, 0x4CB, 0x4CC, 0x4CD, 0x3CE, 0x4CF, 0x581, 0x582, 0x583, 0x584, 0x585, 0x586, 0x587, 0x588, 0x589, 0x58A, 0x5B2, 0x5B3, 0x5B4, 0x5B5, 0x5B6, 0x5B7, 0x5B8, 0x5B9, 0x5BA, 0x5BB, 0x5C6, 0x5C7, 0x5C8, 0x5C9, 0x5CA, 0x5CB, 0x5CC, 0x5CD, 0x5CE, 0x5CF, 0x601, 0x602, 0x603, 0x604, 0x605, 0x606, 0x607, 0x608, 0x609, 0x60A, 0x632, 0x633, 0x634, 0x635, 0x636, 0x637, 0x638, 0x639, 0x63A, 0x63B, 0x646, 0x647, 0x648, 0x649, 0x64A, 0x64B, 0x64C, 0x64D, 0x64E, 0x64F, 0x732, 0x733, 0x734, 0x735, 0x736, 0x737, 0x738, 0x739, 0x73A, 0x73B, 0x746, 0x747, 0x748, 0x749, 0x74A, 0x74B, 0x74C, 0x74D, 0x74E, 0x74F
Festo CP-FB Module	0x010, 0x110, 0x120, 0x130, 0x140, 0x150, 0x1E0, 0x1F0, 0x250, 0x260, 0x270, 0x350, 0x360, 0x370, 0x3B0
LioN-S Module	0x2E0, 0x2E1, 0x2E2, 0x2E3, 0x2E4, 0x2E5, 0x2E6, 0x2E7, 0x2E8, 0x2E9, 0x2EA, 0x2EB, 0x2EC, 0x2ED, 0x2EE, 0x2EF, 0x2F0, 0x2F1, 0x2F2, 0x2F3, 0x2F4, 0x2F5, 0x2F6, 0x2F7, 0x2F8, 0x2F9, 0x2FA, 0x2FB, 0x2FC, 0x2FD, 0x2FE, 0x360, 0x361, 0x362, 0x363, 0x364, 0x365, 0x366, 0x367, 0x368, 0x369, 0x36A, 0x36B, 0x36C, 0x36D, 0x36E, 0x36F, 0x370, 0x371, 0x372, 0x373, 0x374, 0x375, 0x376, 0x377, 0x378, 0x379, 0x37A, 0x37B, 0x37C, 0x37D, 0x37E, 0x581, 0x582, 0x583, 0x584, 0x585, 0x586, 0x587, 0x588, 0x589, 0x58A, 0x58B, 0x58C, 0x58D, 0x58E, 0x58F, 0x590, 0x591, 0x592, 0x593, 0x594, 0x595, 0x596, 0x597, 0x598, 0x599, 0x59A, 0x59B, 0x59C, 0x59D, 0x59E, 0x59F, 0x5A0, 0x601, 0x602, 0x603, 0x604, 0x605, 0x606, 0x607, 0x608, 0x609, 0x60A, 0x60B, 0x60C, 0x60D, 0x60E, 0x60F, 0x610, 0x611, 0x612, 0x613, 0x614, 0x615, 0x616, 0x617, 0x618, 0x619, 0x61A, 0x61B, 0x61C, 0x61D, 0x61E, 0x61F, 0x620, 0x701, 0x702, 0x703, 0x704, 0x705, 0x706, 0x707, 0x708, 0x709, 0x70A, 0x70B, 0x70C, 0x70D, 0x70E, 0x70F, 0x710, 0x711, 0x712, 0x713, 0x714, 0x715, 0x716, 0x717, 0x718, 0x719, 0x71A, 0x71B, 0x71C, 0x71D, 0x71E, 0x71F, 0x720

Module am JX2-Systembus	Reservierte CAN-IDs
BWU1821	0x281, 0x282, 0x283, 0x284, 0x285, 0x286, 0x287, 0x288, 0x289, 0x28A, 0x28B, 0x28C, 0x28D, 0x28E, 0x28F, 0x290, 0x291, 0x292, 0x293, 0x294, 0x295, 0x296, 0x297, 0x298, 0x299, 0x29A, 0x29B, 0x29C, 0x29D, 0x29E, 0x29F, 0x301, 0x302, 0x303, 0x304, 0x305, 0x306, 0x307, 0x308, 0x309, 0x30A, 0x30B, 0x30C, 0x30D, 0x30E, 0x30F, 0x310, 0x311, 0x312, 0x313, 0x314, 0x315, 0x316, 0x317, 0x318, 0x319, 0x31A, 0x31B, 0x31C, 0x31D, 0x31E, 0x31F, 0x481, 0x482, 0x483, 0x484, 0x485, 0x486, 0x487, 0x488, 0x489, 0x48A, 0x48B, 0x48C, 0x48D, 0x48E, 0x48F, 0x490, 0x491, 0x492, 0x493, 0x494, 0x495, 0x496, 0x497, 0x498, 0x499, 0x49A, 0x49B, 0x49C, 0x49D, 0x49E, 0x49F, 0x501, 0x502, 0x503, 0x504, 0x505, 0x506, 0x507, 0x508, 0x509, 0x50A, 0x50B, 0x50C, 0x50D, 0x50E, 0x50F, 0x510, 0x511, 0x512, 0x513, 0x514, 0x515, 0x516, 0x517, 0x518, 0x519, 0x51A, 0x51B, 0x51C, 0x51D, 0x51E, 0x51F, 0x5C6, 0x5C7, 0x5C8, 0x5C9, 0x5CA, 0x5CB, 0x5CC, 0x5CD, 0x5CE, 0x5CF, 0x646, 0x647, 0x648, 0x649, 0x64A, 0x64B, 0x64C, 0x64D, 0x64E, 0x64F, 0x746, 0x747, 0x748, 0x749, 0x74A, 0x74B, 0x74C, 0x74D, 0x74E, 0x74F
LJX7-CSL	0x481, 0x482, 0x483, 0x484, 0x485, 0x486, 0x487, 0x488, 0x489, 0x48A, 0x48B, 0x48C, 0x48D, 0x48E, 0x48F, 0x490, 0x491, 0x492, 0x493, 0x494, 0x495, 0x496, 0x497, 0x498, 0x499, 0x49A, 0x49B, 0x49C, 0x49D, 0x49E, 0x49F, 0x501, 0x502, 0x503, 0x504, 0x505, 0x506, 0x507, 0x508, 0x509, 0x50A, 0x50B, 0x50C, 0x50D, 0x50E, 0x50F, 0x510, 0x511, 0x512, 0x513, 0x514, 0x515, 0x516, 0x517, 0x518, 0x519, 0x51A, 0x51B, 0x51C, 0x51D, 0x51E, 0x51F, 0x581, 0x582, 0x583, 0x584, 0x585, 0x586, 0x587, 0x588, 0x589, 0x58A, 0x58B, 0x58C, 0x58D, 0x58E, 0x58F, 0x590, 0x591, 0x592, 0x593, 0x594, 0x595, 0x596, 0x597, 0x598, 0x599, 0x59A, 0x59B, 0x59C, 0x59D, 0x59E, 0x59F, 0x5A0, 0x601, 0x602, 0x603, 0x604, 0x605, 0x606, 0x607, 0x608, 0x609, 0x60A, 0x60B, 0x60C, 0x60D, 0x60E, 0x60F, 0x610, 0x611, 0x612, 0x613, 0x614, 0x615, 0x616, 0x617, 0x618, 0x619, 0x61A, 0x61B, 0x61C, 0x61D, 0x61E, 0x61F, 0x620, 0x701, 0x702, 0x703, 0x704, 0x705, 0x706, 0x707, 0x708, 0x709, 0x70A, 0x70B, 0x70C, 0x70D, 0x70E, 0x70F, 0x710, 0x711, 0x712, 0x713, 0x714, 0x715, 0x716, 0x717, 0x718, 0x719, 0x71A, 0x71B, 0x71C, 0x71D, 0x71E, 0x71F, 0x720

Funktion der CAN-Prim-Schnittstelle

Funktion

Bei der freiprogrammierbaren CAN-Prim-Schnittstelle funktioniert der Datenaustausch zwischen dem CAN-Bus und dem Anwendungsprogramm über Nachrichtenboxen. Jede Nachrichtenbox bietet Platz für eine komplette CAN-Nachricht.

Dem Anwender stehen 16 Nachrichtenboxen zur Verfügung. Jede Nachrichtenbox kann als Sende- oder Empfangsbox konfiguriert werden und verfügt über eine eigene CAN-ID.

Technische Daten

Funktion	Beschreibung
CAN-ID	11-Bit oder 29-Bit
Anzahl Nachrichtenboxen	16

Aktivierung der CAN-Prim-Schnittstelle

Die Aktivierung der CAN-Prim-Schnittstelle erfolgt über Bits im MR 2077 *JX2-Systembus-Sonderfunktionen Registerbeschreibung MR 2077*. (siehe Seite 59)

Interne Prozesse der CAN-Prim-Schnittstelle

Einleitung

Die CAN-Prim-Schnittstelle arbeitet die folgenden Aufgaben selbstständig ab.

- Senden von CAN-Nachrichten
- Empfangen von CAN-Nachrichten
- Filtern von CAN-Nachrichten beim Empfang

Interner Empfang von CAN-Nachrichten

Die CAN-Prim-Schnittstelle empfängt folgendermaßen eine neue CAN-Nachricht:

Stufe	Beschreibung	
1	Eine gültige CAN-Nachricht wurde vom CAN-Bus empfangen.	
2	Die CAN-ID stimmt mit der Empfangsmaske überein.	
3	Die CAN-ID stimmt mit der CAN-ID einer Nachrichtenbox, die auf Empfang konfiguriert wurde, überein.	
4	**Wenn in MR 10530 + Nachrichtenboxnummer*20 der Nachrichtenbox ...**	**... dann ...**
	... das Bit 1 *NEW-DAT* = 0 ist,	... wird das Bit 1 *NEW-DAT* = 1; weiter mit Stufe 5.
	... das Bit 1 *NEW-DAT* = 1 ist,	... wird das Bit 2 *OVERRUN* = 1; die Daten der CAN-Nachricht werden verworfen.
5	Das MR 10503 *Fifo-Füllstand* wird um Eins erhöht. Dieses Register zeigt, ob und wie viele neue CAN-Nachrichten empfangen wurden.	
6	Die Nachrichtenboxnummer wird in das MR 10504 *Fifo-Daten* eingetragen. Dieses Register zeigt, in welcher Nachrichtenbox eine neue CAN-Nachricht empfangen wurde.	
7	Im MR 10500 *Status CAN-Prim* wird das Bit 1 *NEW-DAT* = 1.	

Registerbeschreibung der CAN-Prim-Schnittstelle

Register zur Konfiguration des JX2-Systembusses

Die CAN-Prim-Schnittstelle wird im MR 2077 *JX2-Systembus-Sonderfunktionen* aktiviert.

Register	Beschreibung
MR 2029	Baudrate des JX2-Systembusses
MR 2077	JX2-Systembus-Sonderfunktionen

Register zur Konfiguration der CAN-Prim-Schnittstelle

Register	Beschreibung
MR 10500	Statusregister CAN-Prim
MR 10501	Kommandoregister CAN-Prim
MR 10503	Fifo-Füllstand - Anzahl empfangener Nachrichten
MR 10504	Fifo-Daten - Nachrichtenboxnummern mit neu empfangenen Nachrichten
MR 10506	Globale Empfangsmaske
MR 10507	Globale Empfangs-ID

CANopen®-STX-API ist nicht implementiert:

MR 2077

Freigabe JX2-Systembus-Sonderfunktionen

Der Wert in diesem Register beeinflusst das Verhalten bei der Initialisierung des JX2-Systembusses.

Bedeutung der Bits		
Bit 2	**CAN-Prim zusätzlich zum JX2-Systembus aktivieren**	
	1 =	CAN-Prim-Schnittstelle und der JX2-Systembus werden beim nächsten Start des JX2-Systembusses initialisiert. Das erfordert einen Neustart der Steuerung. JX2-Erweiterungsmodule können angeschlossen werden.
Bit 3	**Nur CAN-Prim aktivieren**	
	1 =	Nur die CAN-Prim-Schnittstelle wird beim nächsten Start des JX2-Systembusses initialisiert. Das erfordert einen Neustart der Steuerung. Alle Node-IDs sind **ohne** Einschränkung verwendbar. Die Steuerung nimmt keine JX2-Erweiterungsmodule am JX2-Systembus in Betrieb. Deshalb können **keine** JX2-Erweiterungsmodule angeschlossen werden.
Bit 4	**CAN-IDs 0x081 ... 9x09F für CAN-Prim**	
	1 =	Die CAN-Prim-Schnittstelle erlaubt die Kommunikation mit den CAN-IDs 0x081 ... 0x09F. Über diese CAN-IDs wird normalerweise der Master-Slave-Betrieb mit JX2-Slave-Modulen und MC-Achsen abgewickelt.

Modulregistereigenschaften	
Wert nach Reset	Remanent, Werkseinstellung: 0
Wird wirksam	Beim nächsten Start der Steuerung

CANopen®-STX-API ist implementiert:

MR 2077 **Freigabe JX2-Systembus-Sonderfunktionen**

Der Wert in diesem Register beeinflusst das Verhalten bei der Initialisierung des JX2-Systembusses (CAN 1).

Bedeutung der Bits		
Bit 3, Bit 2	**CAN-Prim zusätzlich zum JX2-Systembus aktivieren**	
	01 =	CAN-Prim-Schnittstelle und der JX2-Systembus werden beim nächsten Start des JX2-Systembusses initialisiert. Das erfordert einen Neustart der Steuerung. JX2-Erweiterungsmodule können angeschlossen werden.
Bit 3, Bit 2	**Nur CAN-Prim und CANopen®-STX-API aktivieren**	
	1X =	Beim nächsten Neustart wird der JX2-Systembus nicht initialisiert. Die CAN-Prim-Schnittstelle kann verwendet werden. Alle Node-IDs sind **ohne** Einschränkung verwendbar. Die Steuerung nimmt keine JX2-Erweiterungsmodule am JX2-Systembus in Betrieb. Deshalb können **keine** JX2-Erweiterungsmodule angeschlossen werden. Die Verwendung der CANopen®-STX-API ist möglich.
Bit 4	**CAN-IDs 0x081 ... 9x09F für CAN-Prim**	
	1 =	Die CAN-Prim-Schnittstelle erlaubt die Kommunikation mit den CAN-IDs 0x081 ... 0x09F. Über diese CAN-IDs wird normalerweise der Master-Slave-Betrieb mit JX2-Slave-Modulen und MC-Achsen abgewickelt.

Modulregistereigenschaften	
Wert nach Reset	Remanent, Werkseinstellung: 0
Wird wirksam	Beim nächsten Start der Steuerung

MR 10500 **Statusregister CAN-Prim**

Über das MR 10500 kann der Zustand der CAN-Prim-Schnittstelle ausgewertet werden.

Bedeutung der Bits		
Bit 1	**NEW-DAT**	
	1 =	Mindestens eine Nachrichtenbox hat eine neue CAN-Nachricht erhalten.
Bit 2	**CAN-ID-Länge**	
	0 =	Es werden CAN-IDs mit 11-Bit-Länge gesendet/empfangen
	1 =	Es werden CAN-IDs mit 29-Bit-Länge gesendet/empfangen

Modulregistereigenschaften	
Zugriff	Lesen
Wird wirksam	Wenn CAN-Prim-Schnittstelle aktiviert ist.

MR 10501

Kommandoregister CAN-Prim

Über das MR 10501 werden bestimmte Kommandos zur CAN-Prim-Schnittstelle übertragen.

Diese Kommandos gelten bei direktem und indirektem Zugriff auf die CAN-Nachrichtenbox.

Kommandos der CAN-Prim-Schnittstelle	
7	**Fifo löschen** Alle Einträge im Fifo werden gelöscht. Ergebnis: MR 10503 = 0
8	**Standard-ID-Länge auf 11-Bit einstellen** Die ID-Länge für alle CAN-Nachrichten wird auf 11-Bit eingestellt. Ergebnis: Bit 2 = 0 im MR 10500 MR 10506 := 0 MR 10507 := 0 MR 10542 + Nachrichtenboxnummer*20 := 0x7FF (in allen Nachrichtenboxen)
9	**Standard-ID-Länge auf 29-Bit einstellen** Die ID-Länge für alle CAN-Nachrichten wird auf 29-Bit eingestellt. Ergebnis: Bit 2 = 1 im MR 10500 MR 10506 := 0 MR 10507 := 0 MR 10542 + Nachrichenboxnummer*20 := 0x1FFFFFFF (in allen Nachrichtenboxen)
10	**Nachrichtenboxen auf den Empfang neuer Nachrichten prüfen** Die CAN-Prim-Schnittstelle prüft selbstständig den Empfang neuer CAN-Nachrichten. Mit dem Kommando 10 wird eine manuelle Prüfung auf anstehende Nachrichten erzwungen. Die Erteilung des Kommandos 10 ist inzwischen nicht mehr erforderlich.

Modulregistereigenschaften	
Wird wirksam	Wenn CAN-Prim-Schnittstelle aktiviert ist.

MR 10503

Fifo-Füllstand

Das MR 10503 zeigt, ob und wie viele weitere CAN-Nachrichten empfangen wurden.

Durch die Bildung der Differenz zwischen zwei Lesevorgängen erhalten Sie die Anzahl neuer Nachrichten.

Modulregistereigenschaften		
Werte	Anzahl empfangener Nachrichten:	0 ... 16
Zugriff	Lesen	
Wird wirksam	Wenn CAN-Prim-Schnittstelle aktiviert ist.	

MR 10504

Fifo-Daten

Das MR 10504 zeigt, in welcher Nachrichtenbox die letzte, neue CAN-Nachricht empfangen wurde. Beim Lesen des MR 10504 wird der gerade gelesene Wert aus dem Fifo entfernt. Der Wert des MR 10503 wird dabei um eins verringert.

Hinweis:

Jeder Lesezugriff, auch über einen aktiven JetSym-Setupbildschirm, auf dieses Register verringert die Anzahl empfangener CAN-Nachrichten.

Modulregistereigenschaften		
Werte	Keine Fifo-Daten vorhanden:	-1
	Nachrichtenboxnummer mit neuen Daten:	0 ... 15
Zugriff	Lesen entfernt Zeichen	
Wert nach Reset	-1	
Wird wirksam	Wenn CAN-Prim-Schnittstelle aktiviert ist.	

MR 10506

Globale Empfangsmaske

Die globale Empfangsmaske filtert die Bits der empfangenen CAN-IDs. Bei gesetztem Bit der globalen Empfangsmaske wird das empfangene Bit der CAN-ID mit der globalen Empfangs-ID, siehe MR 10507, verglichen.

Modulregistereigenschaften		
Werte	Bei 11-Bit CAN-IDs	0 ... 0x7FF
	Bei 29-Bit CAN-IDs	0 ... 0x1FFFFFFF
Bit = 0	Bit wird nicht mit MR 10507 verglichen	
Bit = 1	Bit wird mit MR 10507 verglichen	
Wird wirksam	Wenn CAN-Prim-Schnittstelle aktiviert ist.	

MR 10507

Globale Empfangs-ID

Über die globale Empfangs-ID und die MR 10506 *Globale Empfangsmaske* wird ein Bereich von CAN-IDs eingestellt, der an die CAN-Prim-Schnittstelle weitergeleitet wird.

Modulregistereigenschaften		
Werte	Bei 11-Bit CAN-IDs	0 ... 0x7FF
	Bei 29-Bit CAN-IDs	0 ... 0x1FFFFFFF
Wird wirksam	Wenn CAN-Prim-Schnittstelle aktiviert ist.	

Registerbeschreibung der CAN-Nachrichtenbox bei direktem Zugriff

Submodul JX6-SB(-I)

Der direkte Zugriff auf die Nachrichtenbox ist mit dem Submodul JX6-SB(-I) nicht möglich.

Direkter Zugriff

Verwenden Sie zur Programmierung stets die Register für den direkten Zugriff auf die Nachrichtenboxen. Jeder Nachrichtenbox sind 20 Register mit identischer Funktion zugeordnet. Die Register der einzelnen Nachrichtenboxen beginnen ab einer bestimmten Basis-Registernummer.

Nachrichtenboxnummer	Modulregisternummer
0	MR 10530
1	MR 10550
2	MR 10570
3	MR 10590
4	MR 10610
5	MR 10630
6	MR 10650
7	MR 10670
8	MR 10690
9	MR 10710
10	MR 10730
11	MR 10750
12	MR 10770
13	MR 10790
14	MR 10810
15	MR 10830

Register der Nachrichtenboxen der CAN-Prim-Schnittstelle

Jeder Nachrichtenbox sind 20 Register mit identischer Funktion zugeordnet. Die Register der einzelnen Nachrichtenboxen berechnen sich aus der Basis-Registernummer und der Nummer der Nachrichtenbox (0 ... 15).

Modulregister	Beschreibung
MR 10530 + Nachrichtenboxnummer*20	Box-Statusregister
MR 10531 + Nachrichtenboxnummer*20	Box-Konfigurationsregister
MR 10532 + Nachrichtenboxnummer*20	CAN-ID
MR 10533 + Nachrichtenboxnummer*20	Anzahl Datenbytes
MR 10534 + Nachrichtenboxnummer*20	Datenbyte 0

Modulregister	Beschreibung
MR 10535 + Nachrichtenboxnummer*20	Datenbyte 1
MR 10536 + Nachrichtenboxnummer*20	Datenbyte 2
MR 10537 + Nachrichtenboxnummer*20	Datenbyte 3
MR 10538 + Nachrichtenboxnummer*20	Datenbyte 4
MR 10539 + Nachrichtenboxnummer*20	Datenbyte 5
MR 10540 + Nachrichtenboxnummer*20	Datenbyte 6
MR 10541 + Nachrichtenboxnummer*20	Datenbyte 7
MR 10542 + Nachrichtenboxnummer*20	CAN-ID-Maske
MR 10543 + Nachrichtenboxnummer*20	Box-Kommandoregister
MR 10544 + Nachrichtenboxnummer*20	Empfangene CAN-ID
MR 10545 + Nachrichtenboxnummer*20	Nicht verwendet
MR 10546 + Nachrichtenboxnummer*20	Nicht verwendet
MR 10547 + Nachrichtenboxnummer*20	Nicht verwendet
MR 10548 + Nachrichtenboxnummer*20	Nicht verwendet
MR 10549 + Nachrichtenboxnummer*20	Nicht verwendet

MR 10530 + Nachrichtenboxnummer* 20

Box-Statusregister

Dieses Register beschreibt den Zustand der Nachrichtenbox.

Bedeutung der Bits		
Bit 0	**Valid**	
	1 =	Die Nachrichtenbox ist aktiviert
Bit 1	**NEW-DAT**	
	1 =	Die Nachrichtenbox hat eine CAN-Nachricht empfangen. Der Empfang weiterer CAN-Nachrichten ist blockiert.
Bit 2	**OVERRUN**	
	1 =	Es wurde eine neue CAN-Nachricht für diese Nachrichtenbox empfangen, als Bit 1 *NEW-DAT* = 1 war. Die neue Nachricht wird verworfen.

Bedeutung der Bits	
Bit 3	**Sendefehler** 1 = Beim Senden einer CAN-Nachricht aus dieser Nachrichtenbox ist ein Fehler aufgetreten.

Modulregistereigenschaften	
Zugriff	Lesen
Wird wirksam	Wenn CAN-Prim-Schnittstelle aktiviert ist.

MR 10543 + Nachrichtenboxnummer* 20

Box-Kommandoregister

Über das R 200010543 + Nachrichtenboxnummer*20 werden Kommandos zur Nachrichtenbox übertragen.

Kommandos der CAN-Prim-Schnittstelle	
1	**Nachrichtenbox aktivieren** Nachrichtenbox wird aktiviert. Bei der Aktivierung wird geprüft, ob die CAN-ID der Nachrichtenbox vom JX2-Systembus nicht reserviert ist. **Ergebnis, wenn die CAN-ID nicht reserviert ist:** Bit 0 = 1 im MR 10530 + Nachrichtenboxnummer*20
2	**Nachrichtenbox deaktivieren** Die Nachrichtenbox wird deaktiviert. **Ergebnis:** Bit 0 = 0 im MR 10530 + Nachrichtenboxnummer*20
3	**CAN-Nachricht senden** Es wird eine CAN-Nachricht gesendet.
4	**NEW-DAT-Bit löschen** Löscht das Bit 1 *NEW-DAT* im MR 10530 + Nachrichtenboxnummer*20. Die Nachrichtenbox kann wieder CAN-Nachrichten empfangen. **Ergebnis:** Bit 1 = 0 im MR 10530 + Nachrichtenboxnummer*20 Wenn bei allen Nachrichtenboxen das NEW-DAT-Bit 0 ist, dann wird Bit 1 = 0 im MR 10500.
5	**OVERRUN-Bit löschen** Löscht das Bit 2 *OVERRUN* im MR 10530 + Nachrichtenboxnummer*20 der Nachrichtenbox. **Ergebnis:** Bit 2 = 0 im MR 10530 + Nachrichtenboxnummer*20
6	**Sendefehlerbit löschen** Löscht das Bit 3 *Sendefehler* im MR 10530 + Nachrichtenboxnummer*20 der Nachrichtenbox. **Ergebnis:** Bit 3 = 0 im MR 10530 + Nachrichtenboxnummer*20

Modulregistereigenschaften	
Wird wirksam	Wenn CAN-Prim-Schnittstelle aktiviert ist.

MR 10531 + Nachrichtenboxnummer* 20

Box-Konfigurationsregister

Über das MR 10531 + Nachrichtenboxnummer*20 kann die Nachrichtenbox konfiguriert werden.

Konfigurationswerte	
0	**Empfangsbox** Konfiguriert die Nachrichtenbox als Empfangsbox
1	**Sendebox** Konfiguriert die Nachrichtenbox als Sendebox für Standard-Telegramme
2	**Sendebox RTR** Konfiguriert die Nachrichtenbox als Sendebox für RTR-Telegramme

Modulregistereigenschaften	
Wird wirksam	Wenn CAN-Prim-Schnittstelle aktiviert ist.

MR 10532 + Nachrichtenboxnummer* 20

CAN-ID

Bei einer Sendebox wird eine CAN-Nachricht mit dieser CAN-ID gesendet.

Bei einer Empfangsbox werden CAN-Nachrichten mit dieser CAN-ID - maskiert mit der CAN-ID-Maske - empfangen.

Modulregistereigenschaften		
Werte	Bei 11-Bit CAN-IDs	0 ... 0x7FF
	Bei 29-Bit CAN-IDs	0 ... 0x1FFFFFFF
Wird wirksam	Wenn CAN-Prim-Schnittstelle aktiviert ist und die Nachrichtenbox nicht aktiviert ist, d.h. Bit 0 = 0 im MR 10530 + Nachrichtenboxnummer*20.	

MR 10542 + Nachrichtenboxnummer* 20

CAN-ID-Maske

Sie können mit der CAN-ID-Maske konfigurieren, welche Bits einer empfangenen CAN-ID mit der konfigurierten CAN-ID der Nachrichtenbox verglichen wird.

Modulregistereigenschaften		
Werte	Bit = 0	Bit wird nicht mit CAN-ID verglichen
	Bit = 1	Bit wird mit CAN-ID verglichen
Wird wirksam	Wenn CAN-Prim-Schnittstelle aktiviert ist	

MR 10544 + Nachrichtenboxnummer* 20

Empfangene CAN-ID

Bei einer Empfangsbox werden die CAN-ID der empfangenen CAN-Nachrichten hier eingetragen.

Modulregistereigenschaften		
Zugriff	Lesen	
Werte	Bei 11-Bit CAN-IDs	0 ... 0x7FF
	Bei 29-Bit CAN-IDs	0 ... 0x1FFFFFFF
Wird wirksam	Wenn CAN-Prim-Schnittstelle aktiviert.	

MR 10533 + Nachrichtenboxnummer* 20

Anzahl Datenbytes

Bei einer Sendebox wird eine CAN-Nachricht mit dieser Anzahl Datenbytes gesendet.

Bei einer Empfangsbox wird die Anzahl empfangener Datenbytes der CAN-Nachricht eingetragen.

Modulregistereigenschaften		
Werte	Anzahl Datenbytes:	0 ... 8
Wird wirksam	Wenn CAN-Prim-Schnittstelle aktiviert ist.	

MR 10534 ... MR 10541 + Nachrichtenboxnummer* 20

Datenbytes 0 bis 7

Bei einer Sendebox wird eine CAN-Nachricht mit diesen Datenbytes gesendet.

Bei einer Empfangsbox werden die empfangenen Datenbytes der CAN-Nachricht eingetragen.

Modulregistereigenschaften		
Werte	Daten der Datenbytes:	0 ... 255
Wird wirksam	Wenn CAN-Prim-Schnittstelle aktiviert ist.	

Registerbeschreibung der CAN-Nachrichtenbox bei indirektem Zugriff

Indirekter Zugriff

Beim indirekten Zugriff auf die Nachrichtenboxen der CAN-Prim-Schnittstelle muss immer die Nachrichtenbox über MR 10502 *Nummer der Nachrichtenbox* ausgewählt werden.

Die Register für den indirekten Zugriff bleiben aus Kompatibilitätsgründen zu älteren Betriebssystemen bestehen. **Verwenden Sie zur Programmierung der CAN-Prim-Schnittstelle stets die Register für direkten Zugriff.**

MR 10501

Kommandoregister CAN-Prim

Über das MR 10501 werden bestimmte Kommandos zur CAN-Prim-Schnittstelle übertragen.

Kommandos der CAN-Prim-Schnittstelle	
1	**Nachrichtenbox aktivieren** Die angewählte Nachrichtenbox in MR 10502 wird aktiviert. Bei der Aktivierung wird geprüft, ob die CAN-ID der Nachrichtenbox vom Systembus nicht reserviert ist. **Ergebnis:** Bit 0 = 1 im MR 10510
2	**Nachrichtenbox deaktivieren** Die angewählte Nachrichtenbox in MR 10502 wird deaktiviert. **Ergebnis:** Bit 0 = 0 im MR 10510
3	**CAN-Nachricht senden** Es wird eine CAN-Nachricht mit den Daten der angewählten Nachrichtenbox gesendet.
4	**NEW-DAT-Bit löschen** Löscht das Bit 1 *NEW-DAT* im MR 10510. Die angewählte Nachrichtenbox kann wieder CAN-Nachrichten empfangen. **Ergebnis:** Bit 1 = 0 im MR 10510
5	**OVERRUN-Bit löschen** Löscht das Bit 2 *OVERRUN* im MR 10510 der angewählten Nachrichtenbox. **Ergebnis:** Bit 2 = 0 im MR 10510
6	**Sendefehlerbit löschen** Löscht das Bit 3 **Sendefehler** im MR 10510 der angewählten Nachrichtenbox. **Ergebnis:** Bit 3 = 0 im MR 10510

Kommandos der CAN-Prim-Schnittstelle	
7	**Fifo löschen** Alle Einträge im Fifo werden gelöscht. **Ergebnis:** MR 10503 = 0
8	**Standard-ID-Länge auf 11-Bit einstellen** Die ID-Länge für alle CAN-Nachrichten wird auf 11-Bit eingestellt. **Ergebnis:** Bit 2 = 0 im MR 10500 MR 10506 := 0 MR 10507 := 0
9	**Standard-ID-Länge auf 29-Bit einstellen** Die ID-Länge für alle CAN-Nachrichten wird auf 29-Bit eingestellt. **Ergebnis:** Bit 2 = 1 im MR 10500 MR 10506 := 0 MR 10507 := 0
10	**Nachrichtenboxen auf den Empfang neuer Nachrichten prüfen** Die CAN-Prim-Schnittstelle prüft selbständig den Empfang neuer CAN-Nachrichten. Mit dem Kommando 10 wird eine manuelle Prüfung auf anstehende Nachrichten erzwungen. Die Erteilung des Kommandos 10 ist inzwischen nicht mehr erforderlich.

Modulregistereigenschaften	
Wird wirksam	Wenn CAN-Prim-Schnittstelle aktiviert ist.

MR 10502

Nummer der Nachrichtenbox

Über das MR 10502 wird eine Nachrichtenbox ausgewählt. Die Daten der Nachrichtenbox sind dann über MR 10510 bis MR 10521 erreichbar.

Modulregistereigenschaften		
Werte	Nachrichtenboxnummer:	0 ... 15
Wird wirksam	Wenn CAN-Prim-Schnittstelle aktiviert ist.	

MR 10510

Box-Statusregister

Dieses Register beschreibt den Zustand der Nachrichtenbox.

Bedeutung der Bits		
Bit 0	**Valid**	
	1 =	Die Nachrichtenbox ist aktiviert.
Bit 1	**NEW-DAT**	
	1 =	Die Nachrichtenbox hat eine CAN-Nachricht empfangen. Der Empfang weiterer CAN-Nachrichten ist blockiert.
Bit 2	**OVERRUN**	
	1 =	Es wurde eine neue CAN-Nachricht für diese Nachrichtenbox empfangen, als Bit 1 *NEW-DAT* = 1 war. Die neue Nachricht wird verworfen.
Bit 3	**Sendefehler**	
	1 =	Beim Senden einer CAN-Nachricht aus dieser Nachrichtenbox ist ein Fehler aufgetreten.

Modulregistereigenschaften	
Zugriff	Lesen
Wird wirksam	Wenn CAN-Prim-Schnittstelle aktiviert ist.

MR 10511

Box-Konfigurationsregister

Über das MR 10511 kann die Nachrichtenbox konfiguriert werden.

Konfigurationswerte	
0	**Empfangsbox** Konfiguriert die Nachrichtenbox als Empfangsbox
1	**Sendebox** Konfiguriert die Nachrichtenbox als Sendebox für Standard-Telegramme
2	**Sendebox RTR** Konfiguriert die Nachrichtenbox als Sendebox für RTR-Telegramme

Modulregistereigenschaften	
Wird wirksam	Wenn CAN-Prim-Schnittstelle aktiviert ist.

MR 10512

CAN-ID

Bei einer Sendebox wird eine CAN-Nachricht mit dieser CAN-ID gesendet.

Bei einer Empfangsbox werden nur CAN-Nachrichten mit dieser CAN-ID empfangen.

Modulregistereigenschaften		
Werte	Bei 11-Bit CAN-IDs	0 ... 0x7FF
	Bei 29-Bit CAN-IDs	0 ... 0x1FFFFFFF
Wird wirksam	Wenn CAN-Prim-Schnittstelle aktiviert ist und die Nachrichtenbox nicht aktiviert ist, d.h. Bit 0 = 0 im MR 10510.	

MR 10513

Anzahl Datenbytes

Bei einer Sendebox wird eine CAN-Nachricht mit dieser Anzahl Datenbytes gesendet.

Bei einer Empfangsbox wird die Anzahl empfangener Datenbytes der CAN-Nachricht eingetragen.

Modulregistereigenschaften		
Werte	Anzahl Datenbytes:	0 ... 8
Wird wirksam	Wenn CAN-Prim-Schnittstelle aktiviert ist.	

MR 10514 ... MR 10521

Datenbytes 0 bis 7

Bei einer Sendebox wird eine CAN-Nachricht mit diesen Datenbytes gesendet.

Bei einer Empfangsbox werden die empfangenen Datenbytes der CAN-Nachricht eingetragen.

Modulregistereigenschaften		
Werte	Daten der Datenbytes:	0 ... 255
Wird wirksam	Wenn CAN-Prim-Schnittstelle aktiviert ist.	

Verwendung der CAN-Prim-Schnittstelle (direkter Zugriff)

Initialisierung

Führen Sie zur Initialisierung der CAN-Prim-Schnittstelle folgende Schritte aus:

Schritt	Vorgehen	
1	Setzen Sie Bit 2 = 1 in MR 2077 *JX2-Systembus-Sonderfunktionen*.	
2	Starten Sie den JX2-Systembus.	
3	Konfigurieren Sie die CAN-ID-Länge für alle Nachrichtenboxen.	
	Wenn die CAN-ID-Länge ...	**... dann ...**
	... 11 Bit ist,	... MR 10501 := 8;
	... 29 Bit ist,	... MR 10501 := 9;

Konfiguration einer Nachrichtenbox zum Senden

Führen Sie zur Konfiguration einer Nachrichtenbox zum Senden folgende Schritte aus:

Schritt	Vorgehen
1	Bestimmen Sie eine Nachrichtenbox. In dieser Anleitung wird die Nachrichtenbox 1 verwendet (MR 10550).
2	Konfigurieren Sie die Nachrichtenbox 1 als Sendebox: MR 10551 := 1;
3	Konfigurieren Sie die CAN-ID zum Senden: MR 10552 := CAN-ID;
4	Aktivieren Sie die Nachrichtenbox 1: MR 10563 := 1; **Ergebnis der erfolgreichen Konfiguration:** Bit 0 = 1 in MR 10550

Senden einer CAN-Nachricht

Führen Sie zum Senden einer CAN-Nachricht folgende Schritte aus:

Schritt	Vorgehen
1	Bestimmen Sie eine Nachrichtenbox. In dieser Anleitung wird die Nachrichtenbox 1 verwendet.
2	Tragen Sie die Anzahl der zu sendende Datenbytes ein: MR 10553 := Anzahl Bytes;
3	Schreiben Sie den Inhalt der zu sendenden Datenbytes: MR 10554 := Datenbyte 0; MR 10555 := Datenbyte 1; ... MR 10561 := Datenbyte 7;
4	Starten Sie die Übertragung der CAN-Nachricht: MR 10563 := 3; **Ergebnis des erfolgreichen Sendens:** Bit 3 = 0 in MR 10550

Konfiguration einer Nachrichtenbox zum Empfangen

Führen Sie zur Konfiguration einer Nachrichtenbox zum Empfangen folgende Schritte aus:

Schritt	Vorgehen
1	Bestimmen Sie eine Nachrichtenbox. In dieser Anleitung wird die Nachrichtenbox 0 verwendet (MR 10530).
2	Konfigurieren Sie die Nachrichtenbox 0 als Empfangsbox: MR 10531 := 0;
3	Konfigurieren Sie die CAN-ID zum Empfangen MR 10532 := CAN-ID;
4	Aktivieren Sie die Nachrichtenbox 1: MR 10543 := 1; **Ergebnis der erfolgreichen Konfiguration:** Bit 0 = 1 in MR 10530

Empfang einer CAN-Nachricht

Führen Sie zum Empfang einer CAN-Nachricht in Nachrichtenbox 0 folgende Schritte aus:

<table>
<tr><th>Schritt</th><th colspan="2">Vorgehen</th></tr>
<tr><td rowspan="3">1</td><td colspan="2">Prüfen Sie Bit 1 NEW-DAT in MR 10500.</td></tr>
<tr><td>Wenn ...</td><td>... dann ...</td></tr>
<tr><td>... Bit 1 NEW-DAT = 1 in MR 10500,</td><td>... wurde eine CAN-Nachricht empfangen.
Weiter mit Schritt 2.</td></tr>
<tr><td>2</td><td colspan="2">Lesen Sie die Nummer der Nachrichtenbox, die eine neue CAN-Nachricht empfangen hat.
Nachrichtenboxnummer := MR 10504;</td></tr>
<tr><td rowspan="3">3</td><td colspan="2">Prüfen Sie auf Überlauf der Nachrichtenbox.</td></tr>
<tr><td>Wenn ...</td><td>... dann ...</td></tr>
<tr><td>... Bit 2 OVERRUN = 1 in MR 10530,</td><td>... trat ein Überlauf auf.</td></tr>
<tr><td>4</td><td colspan="2">Lesen Sie die Anzahl der empfangenen Bytes.
Anzahl Bytes := MR 10533;</td></tr>
<tr><td>5</td><td colspan="2">Lesen Sie die empfangenen Bytes.
Datenbyte 0 := MR 10534;
Datenbyte 1 := MR 10535;
...
Datenbyte 7 := MR 10541;</td></tr>
<tr><td>6</td><td colspan="2">Quittieren Sie den Empfang.
MR 10543 := 4;
Ergebnis des erfolgreichen Empfangs:
Bit 1 = 0 in MR 10530</td></tr>
</table>

CAN-ID-Masken verwenden

Einleitung

Im Normalfall empfängt die CAN-Prim-Schnittstelle nur die CAN-Nachrichten, bei denen die CAN-ID auf dem Bus mit der konfigurierten CAN-ID der Nachrichtenbox genau übereinstimmt.

Mit einer Maske können Sie die CAN-IDs einer Nachrichtenbox, die empfangen werden sollen, erweitern. Jede Nachrichtenbox hat eine eigene CAN-ID und eine eigene CAN-ID-Maske.

Funktionsweise

Wenn ...	... dann ...
... Bit = 0 in MR 10542 + Nachrichtenboxnummer*20 ist,	... wird das Bit der empfangenen CAN-ID nicht ausgewertet.
... Bit = 1 in MR 10542 + Nachrichtenboxnummer*20 ist,	... muss das Bit der empfangenen CAN-ID mit der konfigurierten CAN-ID übereinstimmen.

RTR-Telegramme über die CAN-Prim-Schnittstelle

RTR-Telegramme

RTR-Telegramme (Remote Transmission Request) sind eine besondere Nachrichtenart bei CAN. Mit Hilfe eines RTR-Telegramms kann ein CAN-Teilnehmer A einen anderen CAN-Teilnehmer B zum Senden einer Nachricht auffordern. Mit einem RTR-Telegramm können keine Nutzdaten verschickt werden. Teilnehmer B wird aufgefordert, ein Telegramm mit derselben CAN-ID und seinen zugehörigen Daten zu schicken.

Submodul JX6-SB-I

Das Senden von RTR-Telegrammen ist mit dem Submodul JX6-SB(-I) nicht möglich.

Konfiguration zum Senden und Empfangen von RTR-Telegrammen

Schritt	Vorgehen
1	Bestimmen Sie eine beliebige Nachrichtenbox zum Senden und eine Nachrichtenbox zum Empfangen des RTR-Telegramms. In dieser Anleitung wird die Nachrichtenbox 0 zum Senden und die Nachrichtenbox 1 zum Empfang verwendet.
2	Konfigurieren Sie die Nachrichtenbox 0 zum Senden von RTR-Telegrammen: MR 10531 := 2;
3	Konfigurieren Sie die CAN-ID des RTR-Telegramms: MR 10532 := CAN-ID;
4	Aktivieren Sie die Nachrichtenbox 0: MR 10543 := 1; **Ergebnis:** Bit 0 = 1 in MR 10530
5	Konfigurieren Sie die Nachrichtenbox 1 zum Empfang der Antwort auf das RTR-Telegramm: MR 10551 := 0;
6	Konfigurieren Sie die CAN-ID des RTR-Telegramms: MR 10552 := CAN-ID;
7	Aktivieren Sie die Nachrichtenbox 1: MR 10563 := 1; **Ergebnis:** Bit 0 = 1 in MR 10550

Senden und Empfangen eines RTR-Telegramms

Schritt	Vorgehen	
1	Veranlassen Sie das Senden des RTR-Telegramms aus der Nachrichtenbox 0: MR 10543 := 3;	
2	Warten Sie auf den Empfang der Antwort auf das RTR-Telegramm in der Nachrichtenbox 1.	
	Wenn ...	**... dann ...**
	... Bit 1 *NEWDAT* = 1 in MR 10550 ist,	... hat die Steuerung die Antwort auf das RTR-Telegramm empfangen. Weiter mit Schritt 3.
3	Lesen Sie die Anzahl der empfangenen Bytes Anzahl Bytes := MR 10553;	
4	Lesen Sie die empfangenen Bytes Datenbyte 0 := MR 10554; Datenbyte 1 := MR 10555; ... Datenbyte 7 := MR 10561;	
5	Quittieren Sie den Empfang MR 10563 := 4;	
⇨	Die Nachrichtenbox ist wieder empfangsbereit.	

Jetter automation

Jetter AG
Gräterstraße 2
71642 Ludwigsburg | Germany

Tel +49 7141 2550-0
Fax +49 7141 2550-425
info@jetter.de
www.jetter.de

We automate your success.