Jetter automation



Betriebsanleitung

JVM-104 - Bediengerät

60879281

We automate your success.

Variante: Jetter
Ausführung: O03
Artikelnummer: 60879281
Version 4.01.3
Dezember 2019 / Printed in Germany

Dieses Dokument hat die Jetter AG mit der gebotenen Sorgfalt und basierend auf dem ihr bekannten Stand der Technik erstellt.

Bei Änderungen, Weiterentwicklungen oder Erweiterungen bereits zur Verfügung gestellter Produkte wird ein überarbeitetes Dokument nur beigefügt, sofern dies gesetzlich vorgeschrieben oder von der Jetter AG für sinnvoll erachtet wird. Die Jetter AG übernimmt keine Haftung und Verantwortung für inhaltliche oder formale Fehler, fehlende Aktualisierungen sowie daraus eventuell entstehende Schäden oder Nachteile.

Die im Dokument aufgeführten Logos, Bezeichnungen und Produktnamen sind geschützte Marken der Jetter AG, der mit ihr verbundenen Unternehmen oder anderer Inhaber und dürfen nicht ohne Einwilligung des jeweiligen Inhabers verwendet werden.

Adresse

So erreichen Sie uns:

Jetter AG
Gräterstraße 2
D-71642 Ludwigsburg
Germany

Telefon - Zentrale:	+49 7141 2550-0
Telefon - Vertrieb:	+49 7141 2550-433
Telefon - Technische Hotline:	+49 7141 2550-444
Telefax - Vertrieb:	+49 7141 2550-484
E-Mail - Vertrieb:	sales@jetter.de
E-Mail - Technische Hotline:	hotline@jetter.de

Bedeutung der Betriebsanleitung

Das Dokument ist Bestandteil des Geräts JVM-104:

- Bewahren Sie das Dokument immer, also bis zur Entsorgung des Geräts JVM-104, griffbereit auf.
- Geben Sie das Dokument bei Verkauf, Veräußerung oder Verleih des Geräts JVM-104 weiter.

Wenn Sie Inhalte aus dem Dokument nicht eindeutig verstehen, wenden Sie sich an Ihren Ansprechpartner der Jetter AG.

Die Jetter AG ist dankbar für jede Art von Anregung und Kritik von Ihrer Seite. Sie bittet Sie, die Anregung und Kritik der Jetter AG unter der E-Mail-Adresse info@jetter.de mitzuteilen. Die Mitteilung hilft der Abteilung Dokumentation, die Dokumente noch anwenderfreundlicher zu gestalten und auf Ihre Wünsche und Erfordernisse einzugehen.

Für folgende Fälle enthält das Dokument wichtige Informationen:

- Gerät transportieren
- Gerät montieren
- Gerät installieren
- Gerät programmieren
- Gerät bedienen
- Gerät warten
- Gerät reparieren

Deshalb müssen Sie das Dokument und besonders die Sicherheitshinweise sorgfältig lesen, verstehen und beachten.

Fehlende oder unzureichende Kenntnisse des Dokuments führen zum Verlust jeglicher Haftungsansprüche gegen die Firma Jetter AG. Dem Betreiber empfiehlt die Jetter AG dringend, sich die Einweisung der Personen schriftlich bestätigen zu lassen.

Inhaltsverzeichnis

1 Sicherheitshinweise 9

Grundlegende Sicherheitshinweise ... 10

2 Produktbeschreibung und Geräteaufbau 13

Produktbeschreibung ... 14
Teile und Schnittstellen ... 16
Bestellbezeichnung ... 19
Mechanische Abmessungen ... 20

3 Identifikation des JVM-104 21

3.1 Identifikation über das Typenschild ... 22
Typenschild ... 23
3.2 Versionsregister ... 24
Softwareversionen ... 25
Informationsabfrage über die Gerätewebseite ... 26

4 Montage und Installation des JVM-104 29

4.1 Schnittstellen ... 30
Beispiel einer Verdrahtung ... 31
Anschluss der Spannungsversorgung ... 32
Anschluss der Multifunktionseingänge ... 35
Anschluss der Multifunktionsausgänge ... 38
CAN-Schnittstellen ... 41
Ethernet-Schnittstelle ... 43
USB-Schnittstelle ... 44
4.2 Montage des JVM-104 ... 45
Montage ... 46
4.3 IP-Konfiguration ... 51
Auslieferungszustand ... 52
Der Konfigurationsspeicher ... 53
Die Konfigurationsregister ... 54
IP-Adresse zur Laufzeit einstellen ... 55
Einstellungen über die Gerätewebseite tätigen ... 56

5 Erstinbetriebnahme 59

5.1 Vorbereitung und Einstieg in die Programmierung mit JetSym STX ... 60
Vorbereitungen zur Erstinbetriebnahme ... 61
Programmierung in der Sprache JetSym STX ... 62
5.2 Konfiguration bei der Plattform ER-STX-CE ... 63
Visualisierungsprojekt in JetViewSoft anlegen und konfigurieren ... 64
Visualisierungsprojekt in JetSym anlegen und konfigurieren ... 69
5.3 Programmierung bei der Plattform ER-STX-CE ... 77
Werteingabe über das Digipot ... 78
Visualisierungsobjekt ändern über die Visualisierungskommandos ... 82

6 CANopen®-STX-API 85

STX-Funktion CanOpenInit() ... 87
STX-Funktion CanOpenSetCommand() ... 89
STX-Funktion CanOpenUploadSDO() ... 91
STX-Funktion CanOpenDownloadSDO() ... 96
STX-Funktion CanOpenAddPDORx() ... 101
STX-Funktion CanOpenAddPDOTx() ... 108
Heartbeat-Überwachung ... 114
CANopen®-Objektverzeichnis des JVM-104 ... 118

7 SAE J1939-STX-API 123

Aufbau einer J1939-Nachricht ... 124
STX-Funktion SAEJ1939Init() ... 126
STX-Funktion SAEJ1939SetSA() ... 127
STX-Funktion SAEJ1939GetSA() ... 128
STX-Funktion SAEJ1939AddRx() ... 129
STX-Funktion SAEJ1939AddTx() ... 132
STX-Funktion SAEJ1939RequestPGN() ... 135
STX-Funktion SAEJ1939GetDM1() ... 138
STX-Funktion SAEJ1939GetDM2() ... 141
STX-Funktion SAEJ1939SetSPNConversion() ... 144
STX-Funktion SAEJ1939GetSPNConversion() ... 145

8 Dateisystem 147

8.1 Verzeichnisse ... 148
Verzeichnisse ... 149
8.2 Eigenschaften ... 152
Eigenschaften der Flash-Disk ... 153
Eigenschaften des USB-Sticks ... 154

9 FTP-Server 155

Anmeldung ... 156
Beispiel: Windows FTP-Client ... 157

10 Programmierung 159

Abkürzungen, Modulregistereigenschaften und Formatierungen ... 160
10.1 Speicherübersicht ... 161
Speicher des Betriebssystems ... 162
Speicher des Dateisystems ... 163
Speicher des Anwendungsprogramms ... 164
Speicher für flüchtige Variablen des Anwendungsprogramms ... 165
Speicher für nichtflüchtige Register des Anwendungsprogramms ... 166
Speicher für nichtflüchtige Variablen des Anwendungsprogramms ... 167
Spezialregister ... 169
Merker ... 170
10.2 Bedienelemente und Zündung ... 172
Eingabetasten ... 173
Digipot ... 175
Zündung und Ausschaltverzögerung ... 177

10.3 Multifunktionseingänge/-ausgänge ... 179
Multifunktionseingänge, Status und Kommandos ... 180
Multifunktionseingänge, analoge Funktionen ... 182
Multifunktionseingänge, digitale Funktionen ... 185
Multifunktionsausgänge, Status und Kommandos ... 188
Multifunktionsausgänge, analoge Funktionen ... 190
Multifunktionsausgänge, digitale Funktionen ... 192
Multifunktionsausgänge PA3 und PA4 als H-Brücke ... 194
10.4 Laufzeitregister ... 196
Beschreibung der Laufzeitregister ... 197
Beispielprogramm Laufzeitregister ... 199
10.5 Überwachung der Schnittstellenaktivität ... 201
Funktionsweise ... 202
Programmierung ... 204
10.6 Freiprogrammierbare IP-Schnittstelle ... 206
10.6.1 Programmierung ... 208
Initialisieren der freiprogrammierbaren IP-Schnittstelle ... 209
Verbindung öffnen ... 210
Daten senden ... 214
Daten empfangen ... 216
Verbindung schließen ... 219
10.6.2 Register ... 220
Registernummern ... 221
Registerbeschreibung ... 222
10.6.3 Programmbeispiele ... 225
Server ... 226
Client ... 230

11 Automatisches Kopieren von Steuerungsdaten 235

11.1 Funktionsweise ... 236
Autostartfunktion laden ... 237
Autostartfunktion ausführen ... 238
Betriebsart Autostart beenden ... 239
11.2 autostart.bat - Aufbau ... 240
Verfügbare Kommandos ... 241
Beispiel für eine Kommandodatei ... 242
11.3 Datendateien ... 243
Dateiformat ... 244
Datendateien automatisch laden ... 245

12 Betriebssystemupdate 247

12.1 Betriebssystemupdate bei einem Bediengerät ... 248
Betriebssystemupdate mit JetSym ... 249
Betriebssystemupdate über FTP ... 250
Betriebssystemupdate über USB ... 251
Betriebssystemupdate über die Gerätewebseite ... 254
Betriebssystemupdate über \App ... 256

13 Anwendungsprogramm 257

Standardablage des Anwendungsprogramms ... 258
Anwendungsprogramm laden ... 259

14 Kurzreferenz JVM-104 261

15 Kurzreferenz JVM-104 Multifunktionsein-/-ausgänge 267

Anhang 269

A: Schnittstellen ... 270
Gesamtübersicht der Anschlussbelegung ... 271
B: Technische Daten ... 275
Technische Daten ... 276
Technische Daten der Multifunktionsein-/-ausgänge ... 278
Mechanische Abmessungen ... 281
Betriebsparameter Umwelt und Mechanik ... 282
Betriebsparameter EMV ... 283
C: Index ... 284

1 Sicherheitshinweise

Einleitung

Dieses Kapitel enthält die grundlegenden Sicherheitshinweise. Wenn erforderlich, warnt das Kapitel auch vor Restgefahren.

Inhalt

Thema	Seite
Grundlegende Sicherheitshinweise	10

Grundlegende Sicherheitshinweise

Einleitung

Das Gerät erfüllt die geltenden Sicherheitsbestimmungen und Normen. Auf die Sicherheit der Anwender legt die Jetter AG besonderen Wert.

Für den Anwender gelten zusätzlich die folgenden Vorschriften:

- Einschlägige Unfallverhütungsvorschriften
- Allgemein anerkannte sicherheitstechnische Regeln
- EG-Richtlinien oder sonstige länderspezifische Bestimmungen

Bestimmungsgemäße Verwendung

Die bestimmungsgemäße Verwendung beinhaltet das Vorgehen nach dieser Betriebsanleitung.

Das Gerät ist zum Einbau in Nutzfahrzeugen und mobilen Arbeitsmaschinen bestimmt. Das Gerät JVM-104 ist ein Bediengerät mit integrierter Steuerung zum Datenaustausch mit Peripheriegeräten.

Das Bediengerät JVM-104 erfüllt die Anforderungen der KFZ-Richtlinie für elektrische/elektronische Unterbaugruppen.

Betreiben Sie das Bediengerät JVM-104 nur innerhalb der angegebenen Grenzen der technischen Daten. Das Bediengerät JVM-104 fällt aufgrund der niedrigen Betriebsspannung unter die Kategorie SELV (Safety Extra Low Voltage). Das Bediengerät JVM-104 fällt also nicht unter die EG-Niederspannungsrichtlinie.

Nicht bestimmungsgemäße Verwendung

Verwenden Sie das Gerät nicht in technischen Systemen, für die eine hohe Ausfallsicherheit vorgeschrieben ist, wie z. B. bei Seilbahnen und Flugzeugen.

Das Gerät JVM-104 ist kein Sicherheitsbauteil nach der Maschinenrichtlinie 2006/42/EG. Deshalb ist der Einsatz des Geräts für sicherheitsrelevante Aufgaben im Sinne des Personenschutzes ungeeignet und unzulässig.

Wenn Sie beabsichtigen, das Gerät bei Umgebungsbedingungen zu betreiben, die von den zulässigen Betriebsbedingungen abweichen, setzen Sie sich mit der Jetter AG vorher in Verbindung.

Personalqualifikation

Je nach Produktlebenszyklus ergeben sich andere Anforderungen an das Personal. Um einen sicheren Umgang mit dem Gerät in den jeweiligen Produktlebensphasen zu gewährleisten, müssen die Anforderungen erfüllt sein.

Produktlebensphase	Mindestanforderung an das Personal
Transport/Lagerung:	Geschultes und eingewiesenes Personal mit Kenntnissen vom richtigen Umgang mit elektrostatisch gefährdeten Bauelementen.
Montage/Installation:	Geschultes Fachpersonal mit elektrotechnischer Ausbildung im Bereich Fahrzeugtechnik z. B. KFZ-Mechatroniker/in.
Inbetriebnahme/ Programmierung:	Geschultes und eingewiesenes Fachpersonal mit weitreichenden Kenntnissen und Erfahrung in den Bereichen Fahrzeugtechnik / Automatisierung z. B. Fahrzeugtechniker/in für Arbeitsmaschinen.
Betrieb:	Geschultes, eingewiesenes und beauftragtes Personal mit Kenntnissen vom richtigen Umgang mit elektronischen Geräten für Arbeitsmaschinen.

Produktlebensphase	Mindestanforderung an das Personal
Außerbetriebnahme/ Entsorgung:	Geschultes Fachpersonal mit elektrotechnischer Ausbildung im Bereich Fahrzeugtechnik z. B. KFZ-Mechatroniker/in.

Umbauten und Veränderungen am Gerät

Aus Sicherheitsgründen sind keine Umbauten und Veränderungen am Gerät und dessen Funktion gestattet.

Nicht ausdrücklich durch die Jetter AG genehmigte Umbauten am Gerät führen zum Verlust jeglicher Haftungsansprüche gegen die Firma Jetter AG.

Die Originalteile sind speziell für das Gerät konzipiert. Teile und Ausstattungen anderer Hersteller sind von der Jetter AG nicht geprüft und deshalb auch nicht freigegeben.

Ihr An- und Einbau kann die Sicherheit und einwandfreie Funktion des Geräts beeinträchtigen.

Für Schäden, die durch die Verwendung von nicht originalen Teilen und Ausstattungen entstehen, ist jegliche Haftung durch die Firma Jetter AG ausgeschlossen.

Transport

Das Gerät JVM-104 enthält elektrostatisch gefährdete Bauelemente, die durch unsachgemäße Behandlung beschädigt werden können.

Der Transport des Geräts JVM-104, besonders auf dem Postweg, muss in Originalverpackung und geeigneter elektrostatischer Schutzverpackung erfolgen.

- Schützen Sie das Gerät JVM-104 durch geeignete Umverpackung vor äußeren Schlag- und Stoßeinwirkungen.
- Prüfen Sie bei beschädigter Verpackung das Gerät auf sichtbare Schäden. Informieren Sie den Transporteur und die Jetter AG.

Einlagerung

Beachten Sie bei der Einlagerung des Geräts JVM-104 die klimatischen Bedingungen aus den technischen Daten.

Reparatur und Wartung

Reparaturen an dem Gerät dürfen nicht vom Betreiber selbst durchgeführt werden. Das Gerät enthält keine vom Betreiber reparierbaren Teile.

Schicken Sie das Gerät zur Reparatur an die Firma Jetter AG ein.

Entsorgung

Für die Entsorgung des Geräts gelten für den Standort der Betreiberfirma die Umweltrichtlinien des jeweiligen Landes.

2 Produktbeschreibung und Geräteaufbau

Einleitung

Dieses Kapitel beschreibt den Geräteaufbau und den Aufbau der Bestellbezeichnung mit ihren Optionen.

Inhalt

Thema	Seite
Produktbeschreibung	14
Teile und Schnittstellen	16
Bestellbezeichnung	19
Mechanische Abmessungen	20

Produktbeschreibung

Das Bediengerät JVM-104

Das JetView der mobilen Serie 104 ist ein vollgrafisches kompaktes Bediengerät. Das Bediengerät JVM-104 ist aufgrund seiner kompakten Bauform und der integrierten Steuerung vielseitig einsetzbar. Das JVM-104 ist speziell für den rauen Einsatz in Nutzfahrzeugen und mobilen Arbeitsmaschinen entwickelt.

Das Bediengerät lässt sich bei allen Lichtverhältnissen optimal bedienen. Dafür sorgen die beleuchteten Tasten sowie ein Lichtsensor, der die Helligkeit des Displays automatisch an das Umgebungslicht anpasst.

Produkteigenschaften

In folgender Liste sind die Produkteigenschaften dargestellt:



- Display: 3,5"-TFT, 350 cd/m^2
- Auflösung: QVGA (320 x 240 Pixel)
- Touchscreen
- 4 Funktionstasten (beleuchtet)
- 1 Digipot mit Druckknopf
- Hintergrundbeleuchtung einstellbar
- Nachtbeleuchtung einstellbar
- Lautsprecher
  Lautstärke: 83 dB in 10 cm Abstand bei Resonanzfrequenz 2.670 Hz
  Frequenz und Lautstärke sind einstellbar.
- Leistungsfähige Programmiersprache JetSym STX
- Schnelle ARM11-CPU
- Nicht flüchtige Register: 30.000
- RAM-Speicher: 128 MByte
- Flash-Speicher: 512 MByte
- 1 Ethernet-Schnittstelle
- 2 CAN-2.0B-Schnittstelle
- 2 Multifunktionseingänge, konfigurierbar
  - Digitaler Eingang
  - Spannungs- und Strommessung
  - Frequenzmessung, Zählerfunktion
- 2 Multifunktionseingänge/-ausgänge, konfigurierbar
  - Digitaler Active-Low-Eingang
  - Digitaler Ausgang bis 2,5 A
  - PWM-Ausgang
  - H-Brücke
- 1 USB-2.0-Schnittstelle

Zubehör

Als Zubehör können Sie ein Montage-Kit bestellen. Es besteht aus einem Haltebügel, Dichtungsring und passenden Schrauben und Muttern.

Artikel-Nr.	Anzahl	Beschreibung
60880138	1	Montage-Kit

Lieferumfang

Zum Lieferumfang des Bediengeräts JVM-104 gehören folgende Artikel:

Artikel-Nr.	Anzahl	Beschreibung
10001132	1	Bediengerät JVM-104
60879283	1	Installationsanleitung

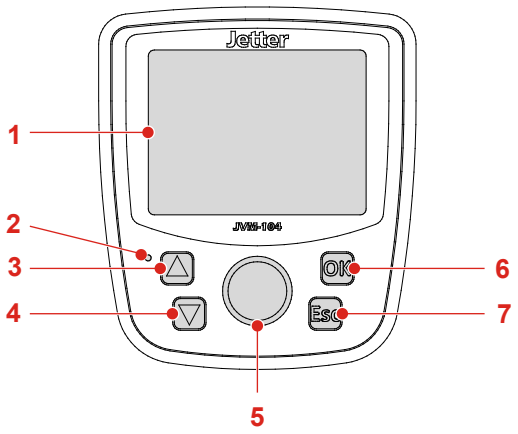
Teile und Schnittstellen

Einleitung

Dieses Kapitel beschreibt die Teile und Schnittstellen des JVM-104.

Vorderseite des JVM-104

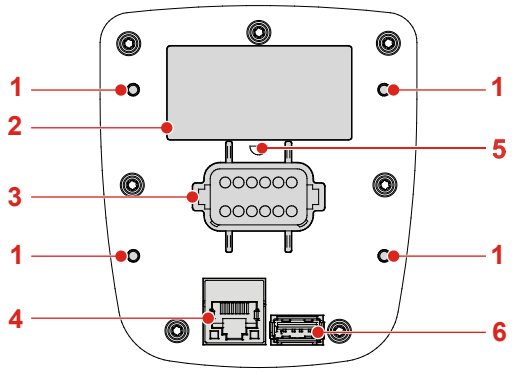
Das Gerät JVM-104 hat ein Touchscreen mit einer aktiven Fläche von 3,5". Die Abbildung zeigt die Vorderseite des Geräts mit allen Bedienelementen.



Nummer	Teil	Beschreibung
1	TFT-Display	Aktive Fläche, Touchscreen
2	Helligkeitssensor	Erfasst die Umgebungshelligkeit
3	Eingabetaste UP	Taste mit Hintergrundbeleuchtung
4	Eingabetaste DOWN	Taste mit Hintergrundbeleuchtung
5	Digipot	Drehknopf mit Taster
6	Eingabetaste OK	Taste mit Hintergrundbeleuchtung
7	Eingabetaste ESC	Taste mit Hintergrundbeleuchtung

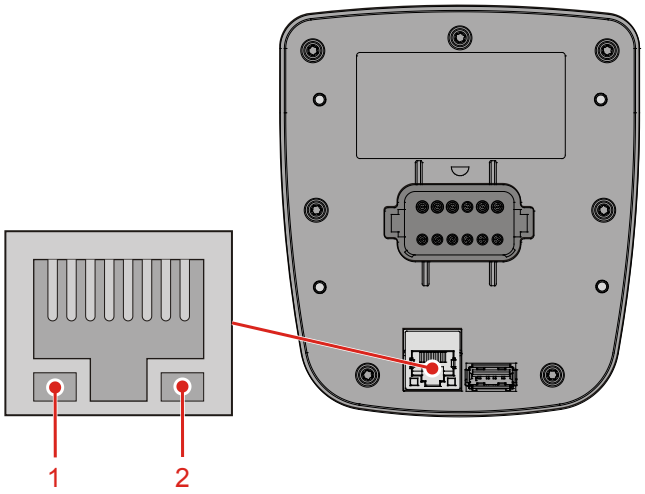
Rückseite des JVM-104

Die Abbildung zeigt die Rückseite des Geräts mit allen Anschlüssen und dem Typenschild.



Nummer	Teil	Beschreibung
1	Bohrung	Zur Befestigung des Bediengeräts, max. 12 mm tief
2	Typenschild	
3	Deutsch-Stecker	Fassung für 12-polige-Deutsch-Buchse
4	Ethernet-Buchse	Übertragungsrate 10 MBit/s, 100 MBit/s
5	DAE	Druckausgleichelement
6	USB-Buchse	USB 2.0 mit Host

LEDs auf der Unterseite Die Abbildung zeigt die Status-LEDs der RJ45-Buchse:



Nummer	Teil	Beschreibung
1	LED gelb	Blinkt bei Aktivität
2	LED grün	Leuchtet bei hergestellter Verbindung

Bestellbezeichnung

Bestellbezeichnung

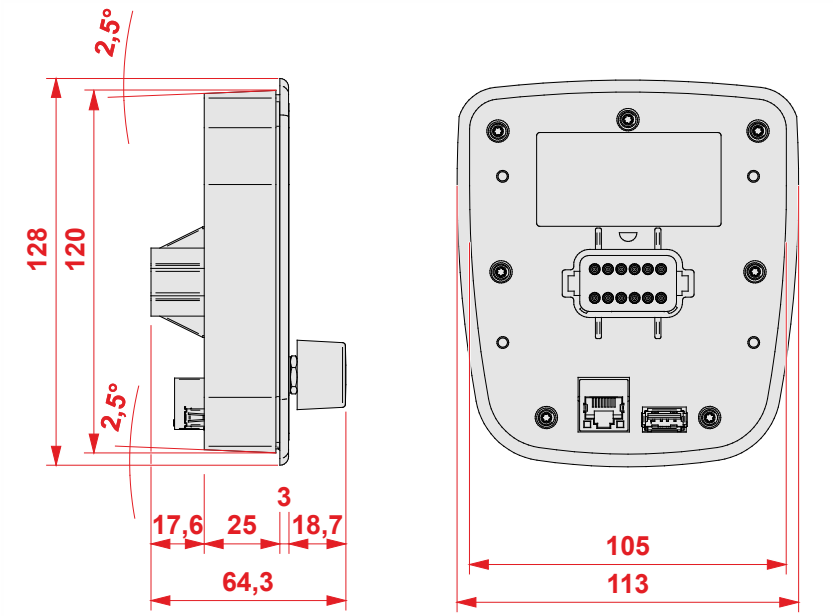
Das Bediengerät JVM-104 ist unter folgender Artikelnummer bei der Jetter AG bestellbar.

Artikel-Nr.	Bestellbezeichnung
10001132	JVM-104-K00-O03

Mechanische Abmessungen

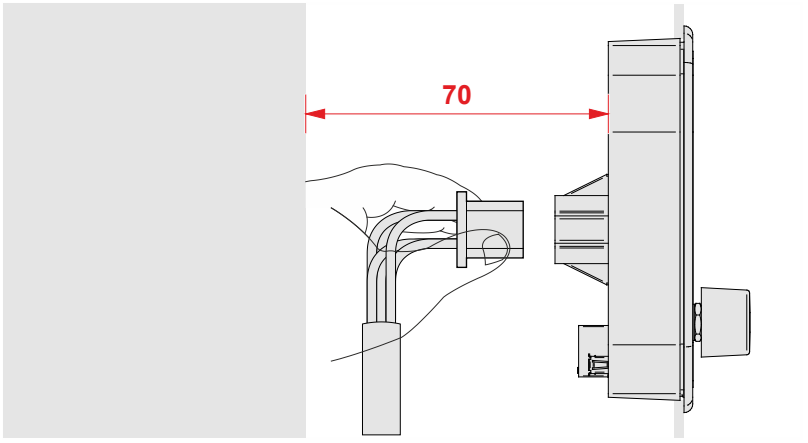
Mechanische Abmessungen

Die Abbildung zeigt die Abmessungen des JVM-104 in Millimeter.



Platzbedarf für den Einbau und Service

Die Abbildung zeigt den Platzbedarf für das Bediengerät JVM-104. Das Maß ist in Millimeter angegeben.



Halten Sie den Raum um das Gehäuse für den Servicefall frei.

- Die Stecker müssen sich jederzeit abziehen lassen.

3 Identifikation des JVM-104

Zweck des Kapitels

Dieses Kapitel unterstützt die Identifikation des JVM-104 in folgenden Punkten:

- Bestimmung der Hardwarerevision.
- Auslesen des elektronischen Typenschilds EDS. Im EDS sind zahlreiche fertigungsrelevante Daten abgelegt.
- Bestimmung der Softwareversionen.

Voraussetzungen

Zur Identifikation des Bediengeräts JVM-104 müssen folgende Voraussetzungen erfüllt sein:

- Das Bediengerät ist mit einem PC verbunden.
- Auf dem PC ist die Programmiersoftware JetSym ab der Version 5.1.2 installiert.

Infos für die Hotline

Wenn Sie sich wegen eines Problems an die Hotline der Jetter AG wenden müssen, halten Sie folgende Informationen des Bediengeräts JVM-104 bereit:

- Seriennummer
- Betriebssystemversion des Bediengeräts
- Hardwarerevision

Inhalt

Thema	Seite
Identifikation über das Typenschild	22
Versionsregister	24

3.1 Identifikation über das Typenschild

Einleitung

Jedes Bediengerät JVM-104 identifiziert sich über das Typenschild am Gehäuse. Wenn Sie sich wegen eines Problems an die Hotline der Jetter AG wenden müssen, dann benötigen Sie die Daten der Hardwarerevision und die Seriennummer.

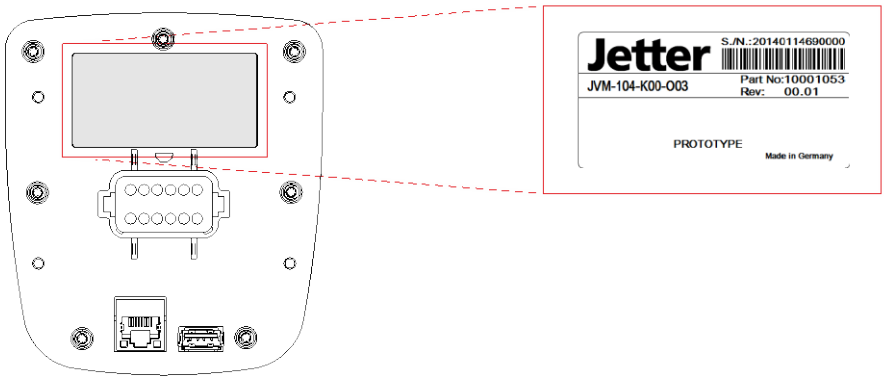
Inhalt

Thema	Seite
Typenschild	23

Typenschild

Position des Typenschilds

Die folgende Abbildung zeigt die Position des Typenschilds auf der Rückseite des JVM-104.



Typenschild

Das Typenschild eines Bediengeräts JVM-104 beinhaltet folgende Informationen:



Nummer	Beschreibung
1	Produktname
2	Seriennummer
3	Artikelnummer
4	Hardwarerevision

3.2 Versionsregister

Einleitung

Das Betriebssystem des Geräts JVM-104 besitzt einige Register, über die Sie die Versionsnummern des Betriebssystems und seiner Komponenten auslesen können. Wenn Sie sich wegen eines technischen Problems an die Hotline der Jetter AG wenden, dann benötigen Sie diese Angaben.

Inhalt

Thema	Seite
Softwareversionen	25
Informationsabfrage über die Gerätewebseite	26

Softwareversionen

Einleitung

Das Gerät JVM-104 beinhaltet Software mit eindeutigen Versionsnummern, die über Spezialregister lesbar sind.

Format von Softwareversionsnummern

Die Softwareversionsnummern des JVM-104 sind durch vier Zahlen dargestellt.

1	.	2	.	3	.	4

Element	Beschreibung
1	Major- oder Hauptversionsnummer
2	Minor- oder Nebenversionsnummer
3	Branch- oder Zwischenversionsnummer
4	Build-Versionsnummer

Freigegebene Version

Bei einer freigegebenen Version haben die Branch- und die Build-Versionsnummer den Wert 0.

Registerübersicht

Aus folgenden Registern können Sie die Softwareversionen auslesen:

Register	Beschreibung
200000	Version des Betriebssystems
210001	Version der STX-Interpreters für das STX-Anwendungsprogramm (JetVM-Version)

Versionsnummern im JetSym-Setup

Die folgende Abbildung zeigt ein Setup-Fenster in JetSym, das die Versionsregister darstellt. Wählen Sie zur Anzeige einer Versionsnummer im JetSym-Setup das Format **IP-Adresse** aus.

	Name	Nummer	Inhalt
1	OS	200000	328
2	JetVM_Version	210001	1.5.0.67
3			

Informationsabfrage über die Gerätewebseite

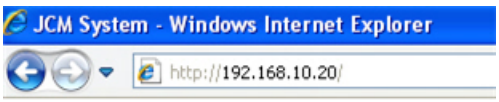
Einleitung

Auf der Gerätewebseite im Internet-Explorer sehen Sie alle Eigenschaften und Versionsnummern Ihres Geräts. Die Konfiguration der Ethernet-Schnittstelle und CANopen®-Schnittstelle ist auch möglich.

Aufruf der Gerätewebseite

Wenn Sie im Internet-Explorer die **aktuelle** IP-Adresse des Geräts eingeben, wird auf dem PC die Gerätewebseite angezeigt.

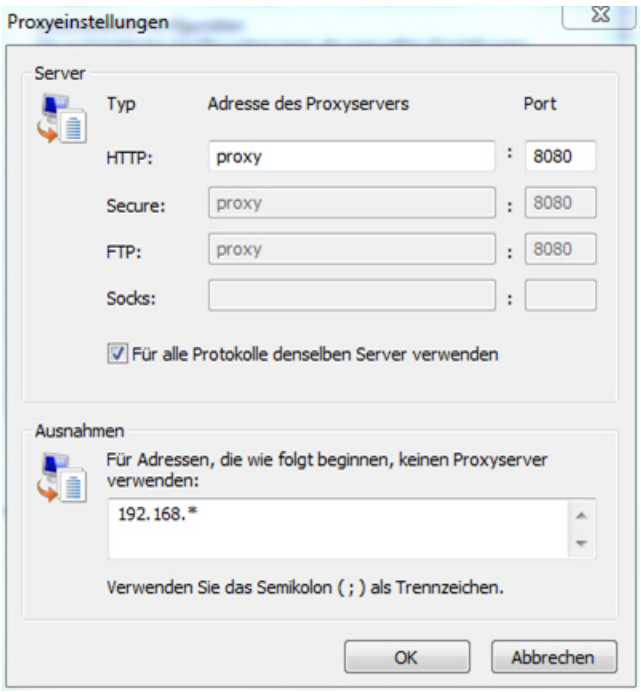
Beispiel:



Voraussetzungen

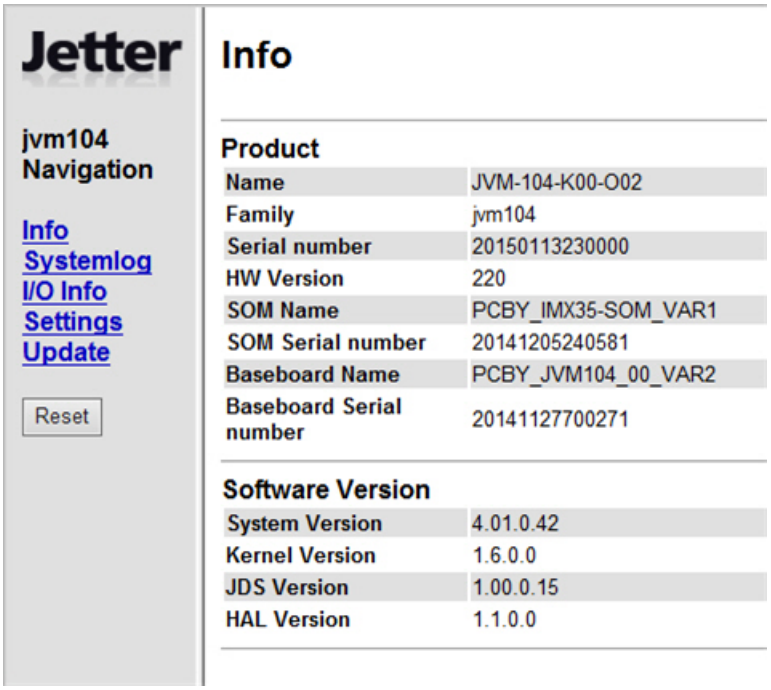
Um im Internet Explorer die IP-Adresse bei den Ausnahmen einzugeben, gehen Sie wie folgt vor:

Schritt	Vorgehen
1	Öffnen Sie über das Menü **Extras** das Fenster **Internetoptionen**.
2	Klicken Sie auf die Schaltfläche **LAN-Einstellungen** auf dem Reiter **Verbindungen**.
3	Klicken Sie auf die Schaltfläche **Erweitert** unter **Proxyserver**.
⇨	Das Fenster **Proxyeinstellungen** hat sich geöffnet.
4	Geben Sie im Fenster **Proxyeinstellungen** bei den **Ausnahmen** die IP-Adresse **192.168.*** ein, siehe die folgende Abbildung.



Info-Seite

Auf der Seite **Info** bekommen Sie die folgenden Informationen angezeigt.



4 Montage und Installation des JVM-104

Zweck des Kapitels

Dieses Kapitel unterstützt die Installation des Bediengeräts JVM-104 im Fahrzeug in folgenden Punkten:

- Planung der Verdrahtung des JVM-104
- Montage
- Konfiguration der IP-Schnittstelle des JVM-104

Inhalt

Thema	Seite
Schnittstellen	30
Montage des JVM-104	45
IP-Konfiguration	51

4.1 Schnittstellen

Einleitung

Das Bediengerät JVM-104 hat folgende Schnittstellen:

- Deutsch-Stecker
- Ethernet-Schnittstelle
- USB-Schnittstelle

Deutsch-Stecker

Der Deutsch-Stecker hat folgende Funktion:

- Spannungsversorgung des JVM-104
- Spannungsversorgung für höhere Lastströme
- Multifunktionseingänge
- Multifunktionsausgänge
- CANopen®-Busschnittstellen: CAN 1 und CAN 2
- Erkennung der Zündung

Ethernet-Schnittstelle

Die RJ45-Buchse hat folgende Funktion:

- Ethernet-Schnittstelle zu einem PC
- Ethernet-Schnittstelle zu weiteren Geräten, wie einer Steuerung

USB-Schnittstelle

Die USB-Schnittstelle hat folgende Funktion:

- Update des Systems über USB-Stick
- Schnittstelle für USB-Sticks für weiteren Speicherplatz
- Schnittstelle für USB-Geräte wie Maus oder Tastatur

Inhalt

Thema	Seite
Beispiel einer Verdrahtung	31
Anschluss der Spannungsversorgung	32
Anschluss der Multifunktionseingänge	35
Anschluss der Multifunktionsausgänge	38
CAN-Schnittstellen	41
Ethernet-Schnittstelle	43
USB-Schnittstelle	44

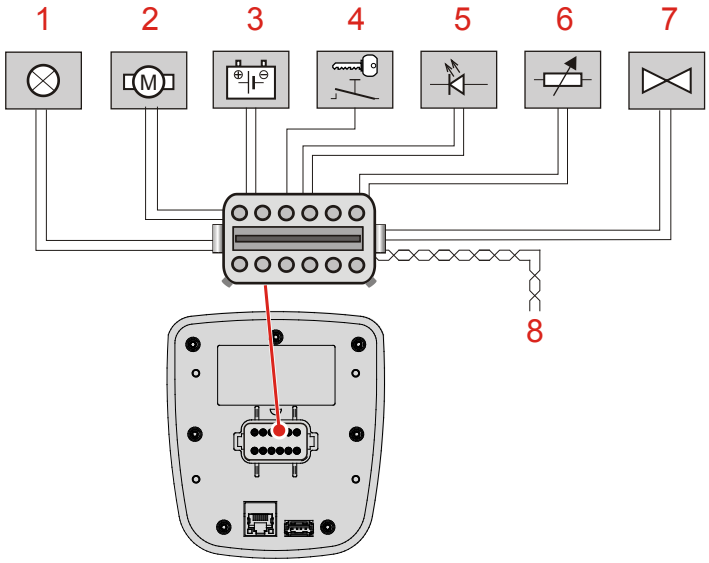
Beispiel einer Verdrahtung

Einleitung

Folgend ist anhand eines Beispiels das Anschließen eines JVM-104 beschrieben.

Beispiel

Die Abbildung zeigt ein Beispiel für eine Verdrahtung.



Nummer	Beschreibung
1	Ausgang, z. B. Lampe
2	Ausgang zur Ansteuerung eines Motors
3	Spannungsversorgung (Batterie)
4	Zündschloss
5	Ausgang zur Ansteuerung der Display-LEDs
6	Eingang, z. B. zum analogen Auslesen eines Füllstandssensors
7	Ausgang, z. B. zur Ansteuerung eines Ventils
8	CANopen®-Bus

Anschluss der Spannungsversorgung

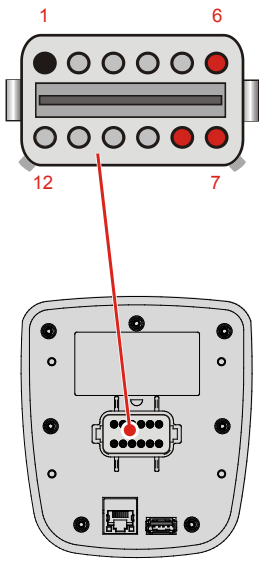
Verwendung des Steckers

Dieser Stecker wird auch für folgende Peripherie verwendet:

- Spannungsversorgung
- Multifunktionseingänge
- Multifunktionsausgänge
- CAN-Kommunikation
- Erkennung Zündung

Anschlusspins der Spannungsversorgung

Die Abbildung zeigt die Pinbelegung des Anschlusssteckers für die Spannungsversorgung und den Anschluss der Zündung (Blick auf die Kabelseite):



Die Pinbelegung ist wie folgt:

Pin	Beschreibung	Klemmenbezeichnung in KFZ
1	Bezugspotenzial GND	KL 31
6	Versorgungsspannung UB für die Logik des Geräts Spannung: DC12 V oder DC 24 V Stromaufnahme: Maximal 2 A	KL 30
7	Versorgungsspannung UB_PA für Multifunktionsausgänge des Geräts Spannung: DC12 V oder DC 24 V Stromaufnahme: Maximal 10 A	KL 30
8	Zündung (+)	KL 15

Technische Daten - Spannungsversorgung UB

Parameter	Beschreibung
Nennspannung	DC 12 V oder DC 24 V
Zulässiger Spannungsbereich UB	DC 8 V ... DC 32 V, gemäß ISO 7637
Zulässiger Spannungsbereich Zündung	DC 5 V ... DC 32 V
Maximaler Strom	2 A
Load-Dump Schutz	Bis DC 70 V
Typische Stromaufnahme Logik (UB)	170 mA bei DC 12 V 90 mA bei DC 24 V
Leistungsaufnahme	Ca. 2 W
Integrierte Schutzfunktionen	Verpolschutz, Überlast, Kurzzeitige Spannungspulse

Technische Daten - Spannungsversorgung UB_PA

Parameter	Beschreibung
Nennspannung	DC 12 V oder DC 24 V
Zulässiger Spannungsbereich UB_PA	DC 8 V ... DC 32 V
Zulässiger Spannungsbereich Zündung	DC 5 V ... DC 32 V
Maximaler Strom	10 A
Load-Dump Schutz	Bis DC 70 V
Integrierte Schutzfunktionen	Verpolschutz

Wichtiger Hinweis zur Versorgungsspannung

Pin 7 des Deutsch-Steckers ist für höhere Ströme ausgelegt. Legen Sie den Leitungsquerschnitt des Kabels für Pin 7 entsprechend aus.

Hinweis zur Zündung

Zum Starten des JVM-104 muss Pin 8 (Zündung +) mit Pin 6 verbunden sein. Das Steuersignal Zündung (+) wird mit der Schlüsselstellung *Zündung EIN* verbunden.

Hinweis zur Stromaufnahme

Die Stromaufnahme beim Einschalten des JVM-104 ist kurzzeitig höher. Um einen zuverlässigen Start des JVM-104 zu gewährleisten, stellen Sie mindestens das 3-fache des benötigten typischen Stroms bereit.

Gegenstücke

Kompatible Gegenstücke des 12-poligen Deutsch-Steckers sind die folgenden Komponenten:



Hersteller	Deutsch
Herstellerartikelnummer Gehäuse	DT06-12S
Herstellerartikelnummer Arretierungskeil	W12S
Herstellerartikelnummer Crimpkontakt (Buchse)	0-462-201-16141
Anschließbarer Adernquerschnitt	1,0 ... 1,5 mm^2 (AWG 18 ... 16)

Anschluss der Multifunktionseingänge

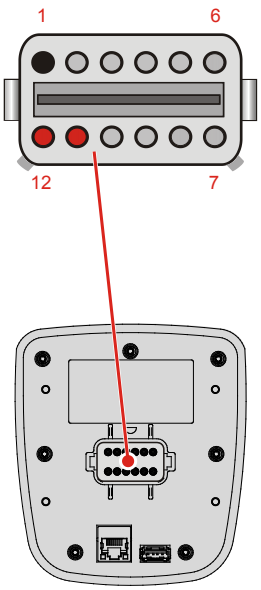
Funktionen der Multifunktionseingänge

Sie haben folgende Funktionen zur Verfügung:

- Analog
  - Spannungsmessung
  - Strommessung
- Digital
  - Digitaler Eingang
  - Frequenzmessung
  - Zählerfunktion

Anschlusspins der Multifunktionseingänge

Die Abbildung zeigt die Pinbelegung des Anschlusssteckers für die Multifunktionseingänge (Blick auf die Kabelseite). Der Pin 1 des Bezugspotenzials ist auch farblich gekennzeichnet.



Die Pinbelegung ist wie folgt:

Pin	Beschreibung
1	Bezugspotenzial GND
11	Multifunktionseingang MFQE1
12	Multifunktionseingang MFQE2

Analoge Funktionen

Sie haben folgende Messmöglichkeiten der analogen Eingänge:

- Spannungsmessung
- Strommessung

Sie schalten im Anwendungsprogramm die Multifunktionseingänge MFQE1 und MFQE2 zwischen Spannungs- und Strommessung um.

Technische Daten - Spannungsmessung

Parameter	Beschreibung
Messbereich	DC 0 V ... DC 15 V
Eingangswiderstand	50 kΩ
Auflösung	12 Bit
Potenzialtrennung	Keine

Technische Daten - Strommessung

Parameter	Beschreibung
Messbereich	DC 0 mA ... DC 20 mA
Messwiderstand	120 Ω
Auflösung	12 Bit
Überstromabschaltung	Ja
Potenzialtrennung	Keine

Digitale Funktionen

Sie haben folgende digitale Funktionen zur Verfügung:

- Digitaler Active-High-Eingang
- Frequenzmessung
- Zählerfunktion

Technische Daten - Digitale Eingänge MFQE

Parameter	Beschreibung
Art der Eingänge	Active-High-Eingang
Schaltschwelle EIN	Minimal 2,0 V
Schaltschwelle AUS	Maximal 1,0 V
Eingangswiderstand	50 kΩ
Maximale Eingangsfrequenz	1 Hz
NAMUR-Sensorauswertung	Über Software aktivierbar
Potenzialtrennung	Keine

Technische Daten - Frequenzmessung

Parameter	Beschreibung
Messbereich	0,1 Hz ... 10 kHz
Genauigkeit	99,9 %
Eingangswiderstand	20 kΩ
Typischer Eingangsstrom	Max. 1 mA
Maximale Eingangsfrequenz	Max. 10 kHz
Auflösung der Periodendauermessung	1 µs
Einstellbare Torzeiten für Torzeitmessung	1 ms ... 10 s
Duty-Cycle-Frequenzeingang	Mindestens 35 µs
Schaltschwelle EIN	Min. 4,7 V
Schaltschwelle AUS	Max. 2,5 V

Technische Daten - Zählfunktion

Parameter	Beschreibung
Messbereich	0,1 Hz ... 10 kHz
Auflösung	0,1 %
Zählbereich	32 Bit
Schaltschwelle EIN	Min. 4,7 V
Schaltschwelle AUS	Max. 2,5 V

Gegenstücke

Kompatible Gegenstücke des 12-poligen Deutsch-Steckers sind die folgenden Komponenten:



Hersteller	Deutsch
Herstellerartikelnummer Gehäuse	DT06-12S
Herstellerartikelnummer Arretierungskeil	W12S
Herstellerartikelnummer Crimpkontakt (Buchse)	0-462-201-16141
Anschließbarer Adernquerschnitt	1,0 ... 1,5 mm^2 (AWG 18 ... 16)

Anschluss der Multifunktionsausgänge

Funktionen der Multifunktionsausgänge

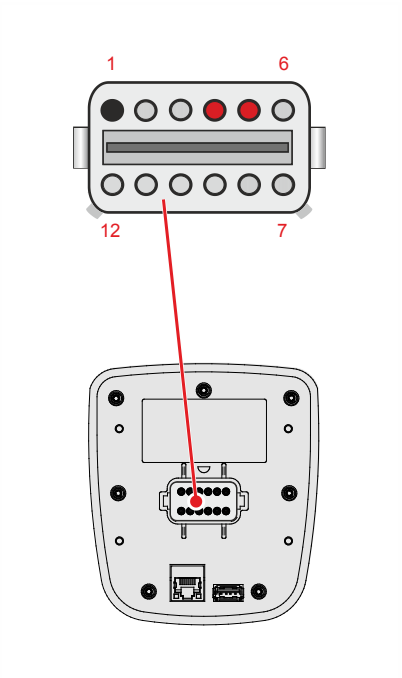
Sie haben folgende Funktionen zur Verfügung:

- Digitaler Ausgang
- Digitaler Eingang
- PWM-Ausgang
- H-Brücke

Sie schalten im Anwendungsprogramm die Multifunktionsausgänge PA3 bis PA4 zwischen den jeweiligen Funktionen um.

Anschlusspins der Multifunktionsausgänge

Die Abbildung zeigt die Pinbelegung des Anschlusssteckers für die Multifunktionsausgänge und die CAN-Schnittstelle (Blick auf die Kabelseite). Der Pin 1 des Bezugspotenzials ist auch farblich gekennzeichnet.



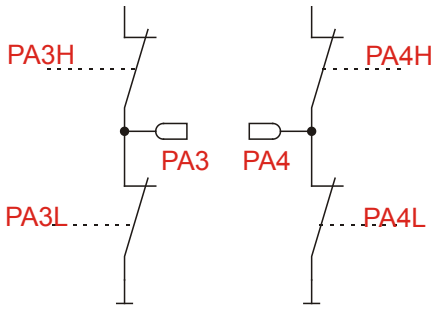
Die Pinbelegung ist wie folgt:

Pin	Beschreibung
1	Bezugspotenzial GND
4	Multifunktionsausgang PA3
5	Multifunktionsausgang PA4

Pinbelegung für die H-Brücke

Die H-Brücke ist auf die folgenden Ausgänge herausgeführt:

- Multifunktionsausgang PA3
- Multifunktionsausgang PA4



Bei der Programmierung schalten Sie die H-Brücke in die gewünschte Richtung:

- PA3H und PA4L für den Linkslauf
- PA4H und PA3L für den Rechtslauf

Technische Daten - Digitaler Eingang PAx

Parameter	Beschreibung
Art des Eingangs	Active-High-Eingang
Nennspannung	UB_PA - 10 %
Schaltschwelle EIN	Maximal DC 1,6 V, > 3,5 mA
Schaltschwelle AUS	Minimal DC 4,0 V, < 0,5 mA
Interner Pull-up-Widerstand	NAMUR: 1 kΩ, 8,2 V
Potenzialtrennung	Keine

Technische Daten - Digitaler Ausgang

Parameter	Beschreibung
Maximaler Ausgangsstrom pro Kanal	2,5 A
Nennspannung	Versorgungsspannung UB_PA
Diagnosemöglichkeit	Überstrom, Kurzschluss, Kabelbruch
Auswertung Überstrom	Einstellbare Stromabschaltung
Maximale Latenzzeit: Umschalten zwischen Aus- und Eingang	500 ms
Maximaler Summenstrom	Der Summenstrom aller Ausgänge darf 8 A nicht überschreiten, sonst Abschaltung nach 500 ms. Wenn der Summenstrom 10,5 A überschreitet, schalten die Ausgänge sofort ab.

Technische Daten - PWM-Ausgang

Parameter	Beschreibung
PWM-Frequenz	100 Hz ... 1 kHz
Auflösung Tastverhältnis	0,1 %
Genauigkeit Tastverhältnis	+/- 42 ns
Maximale Nennspannung	Versorgungsspannung UB_PA
Schaltfrequenz	100 Hz ... 1kHz
Schutzbeschaltung	Überspannung, Überstrom
Diagnosemöglichkeit	Überstrom, Kabelbruch
Maximaler Summenstrom	Der Summenstrom aller Ausgänge darf 8 A nicht überschreiten, sonst Abschaltung nach 500 ms. Wenn der Summenstrom 10,5 A überschreitet, schalten die Ausgänge sofort ab.

Technische Daten - H-Brücke

Parameter	Beschreibung
Maximale Nennspannung	Batteriespannung, max. DC 32 V
Maximaler Ausgangsstrom	2,5 A
Spitzenstrom	~ 5 A, max. 500 ms
Genauigkeit der Strommessung	+/- 20 %
Schaltfrequenz (1/Periodendauer)	100 Hz ... 1 kHz
Frequenzgenauigkeit	+/- 42 ns
Auflösung Tastverhältnis	0,1 %
Diagnosemöglichkeit	Überstrom

Gegenstücke

Kompatible Gegenstücke des 12-poligen Deutsch-Steckers sind die folgenden Komponenten:



Hersteller	Deutsch
Herstellerartikelnummer Gehäuse	DT06-12S
Herstellerartikelnummer Arretierungskeil	W12S
Herstellerartikelnummer Crimpkontakt (Buchse)	0-462-201-16141
Anschließbarer Adernquerschnitt	1,0 ... 1,5 mm^2 (AWG 18 ... 16)

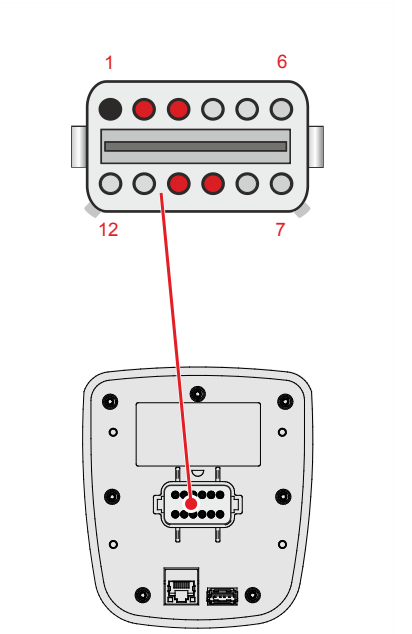
Verwandte Themen

- **Multifunktionsausgänge PA3 und PA4 als H-Brücke** (siehe Seite 194)

CAN-Schnittstellen

Anschlusspins des CANopen®-Busses

Die Abbildung zeigt die Pinbelegung des Anschlusssteckers für den CANopen®-Bus (Blick auf die Kabelseite). Der Pin 1 des Bezugspotenzials ist auch farblich gekennzeichnet.



Die Pinbelegung ist wie folgt:

Pin	Beschreibung
1	Bezugspotenzial GND
2	CAN_2_L
3	CAN_2_H
9	CAN_1_L
10	CAN_1_H

Technische Daten - CAN-Schnittstelle

Parameter	Beschreibung
Baudrate	250 kBaud ... 1 MBaud
CAN 1: Busabschlusswiderstand	Intern nicht bestückt
CAN 2: Busabschlusswiderstand	Intern mit 120 Ω bestückt
Erforderlich bei CAN 1: Externer Busabschluss	120 Ω
Kabelspezifikation	Verdrillte Adern, ungeschirmt

Busabschlusswiderstand

Das JVM-104 hat keinen integrierten Busabschlusswiderstand.

Drillung

Die Leitungen CAN_L und CAN_H müssen miteinander verdrillt sein.

Spezifikation CAN-Bus-Kabel

Parameter	Beschreibung
Querschnitt	1000 kBaud: 0,25 ... 0,34 mm^2 500 kBaud: 0,34 ... 0,50 mm^2 250 kBaud: 0,34 ... 0,60 mm^2
Kapazität des Kabels	Max. 60 pF/m
Spezifischer Widerstand	1000 kBaud: Max. 70 Ω/km 500 kBaud: Max. 60 Ω/km 250 kBaud: Max. 60 Ω/km
Adernzahl	2
Drillung	Adernpaar CAN_L und CAN_H verdrillt

Leitungslängen

Die maximal zulässige Leitungslänge ist abhängig von der verwendeten Baudrate und der Anzahl der angeschlossenen CANopen®-Geräte.

Baudrate	Leitungslänge	Stichleitungslänge	Gesamtstich-leitungslänge
1000 kBaud	max. 25 m	max. 0,3 m	1,5 m
500 kBaud	max. 100 m	max. 5 m	30 m
250 kBaud	max. 250 m	max. 10 m	60 m

Gegenstücke

Kompatible Gegenstücke des 12-poligen Deutsch-Steckers sind die folgenden Komponenten:

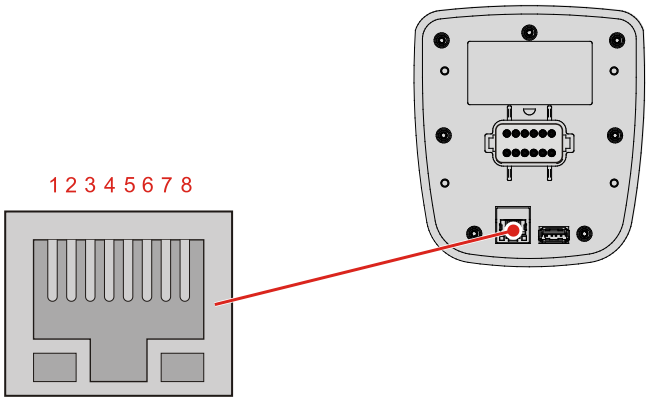


Hersteller	Deutsch
Herstellerartikelnummer Gehäuse	DT06-12S
Herstellerartikelnummer Arretierungskeil	W12S
Herstellerartikelnummer Crimpkontakt (Buchse)	0-462-201-16141
Anschließbarer Adernquerschnitt	1,0 ... 1,5 mm^2 (AWG 18 ... 16)

Ethernet-Schnittstelle

Pinbelegung Ethernet-Schnittstelle

Die Abbildung zeigt die Pinbelegung der Anschlussbuchse für das Ethernet-Kabel:



Technische Daten - Ethernet

Parameter	Beschreibung
Klemmenart	RJ45-Buchse
Anzahl Ports	1
Übertragungsrate	10 MBit/s, 100 MBit/s
Auto-Crossover	Ja

Kabel für die Ethernet-Schnittstelle

Zum Anschluss von Geräten an die Ethernet-Schnittstelle können Sie folgende Kabel separat bei der Jetter AG bestellen:

Artikel-Nr.	Artikel
60537500	Patchkabel 1:1, 1 m grau Hirose, Cat 5e, geschirmt
60854512	Patchkabel 1:1, 2 m grau Hirose, Cat 5e, geschirmt
60854514	Patchkabel 1:1, 5 m grau Hirose, Cat 5e, geschirmt
60854515	Patchkabel 1:1, 10 m grau Hirose, Cat 5e, geschirmt

USB-Schnittstelle

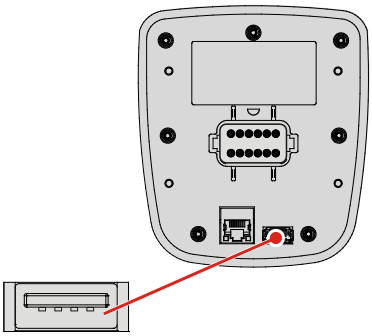
Funktionen der USB-Schnittstelle

Durch die USB-Schnittstelle haben Sie folgende Funktionen zur Verfügung:

- Eingabegeräte wie Maus oder Tastatur anschließen
- Speichern und Laden von Daten auf einem USB-Stick
- Betriebssystem aktualisieren

USB-Schnittstelle

Die Abbildung zeigt die Anschlussbuchse für USB:



Technische Daten - USB-Stick

Parameter	Beschreibung
Versorgung	5 V, max. 500 mA
USB-Anschluss	USB Host-Interface, USB 2.0
Kurzschlussfestigkeit	Ja, Kurzschlussstrom kurzzeitig: ~ 0,8 A

4.2 Montage des JVM-104

Einleitung

Dieses Kapitel beschreibt die Montage des JVM-104.

Inhalt

Thema	Seite
Montage	46

Montage

Einleitung

Dieses Kapitel beschreibt die Montage des Bediengeräts JVM-104.

Einbauort wählen

Wählen Sie einen geeigneten Einbauort.
Wenn der Einbauort folgende Bedingungen erfüllt, ist er geeignet:

- Die Montagefläche muss eben sein.
- Die Montagefläche darf maximal 5 mm dick sein.
- Der Einbauort muss Luftzirkulation erlauben.
- Der Einbauort muss für den Servicefall zugänglich sein.
- Der Einbauort muss ausreichend groß sein.

Ungeeignete Einbauorte meiden

Meiden Sie ungeeignete Einbauorte.
Folgende Einbauorte sind für die Montage des Bediengeräts ungeeignet:

Ungeeigneter Einbauort	Grund
Einbauort im Freien	Das Bediengerät darf nicht Regen oder einem Wasserstrahl ausgesetzt sein. Reinigen Sie das Bediengerät nicht mit einem Dampfstrahler oder ähnlichem.
Unbelüfteter Einbauort	Das Bediengerät kann bei Wärmestau überhitzen.
Einbauort in der Nähe hitze-empfindlicher Materialien	Die Materialien können sich durch die Wärmeentwicklung des Bediengeräts verformen.
Unebene Montagefläche mit Sicken und Erhebungen	Die Montagefläche kann sich beim Festschrauben des Bediengeräts verformen. Die Befestigung ist instabil und unsicher.

Ergonomische Grundsätze beachten

Beachten Sie ergonomische Grundsätze.
Wählen Sie einen benutzerfreundlichen Einbauort:

- Die Bedienelemente müssen leicht erreichbar sein.
- Das Bediengerät muss leicht ablesbar sein.

Vermeiden Sie ergonomisch ungünstige Einbauorte:

- Extreme Winkel, die das Ablesen des Bediengeräts erschweren
- Ungünstige Lichtverhältnisse mit Spiegelung und Blendwirkung
- Verdeckte Einbauorte, die dem Benutzer schwer zugänglich sind

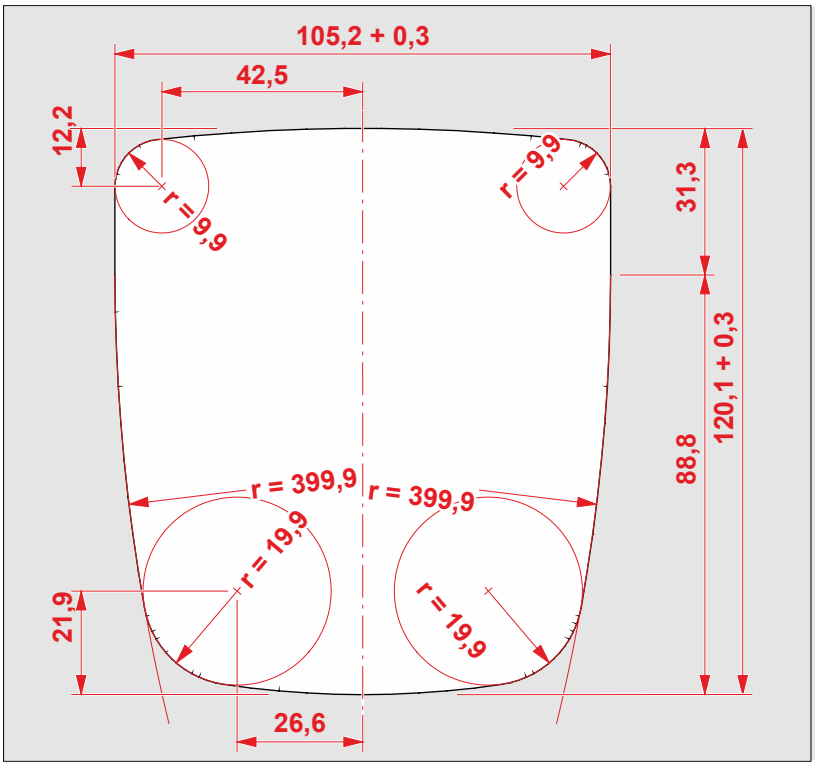
Zubehör

Als Zubehör können Sie ein Montage-Kit bestellen. Es besteht aus einem Haltebügel, Dichtungsring und passenden Schrauben und Muttern.

Artikel-Nr.	Anzahl	Beschreibung
60880138	1	Montage-Kit

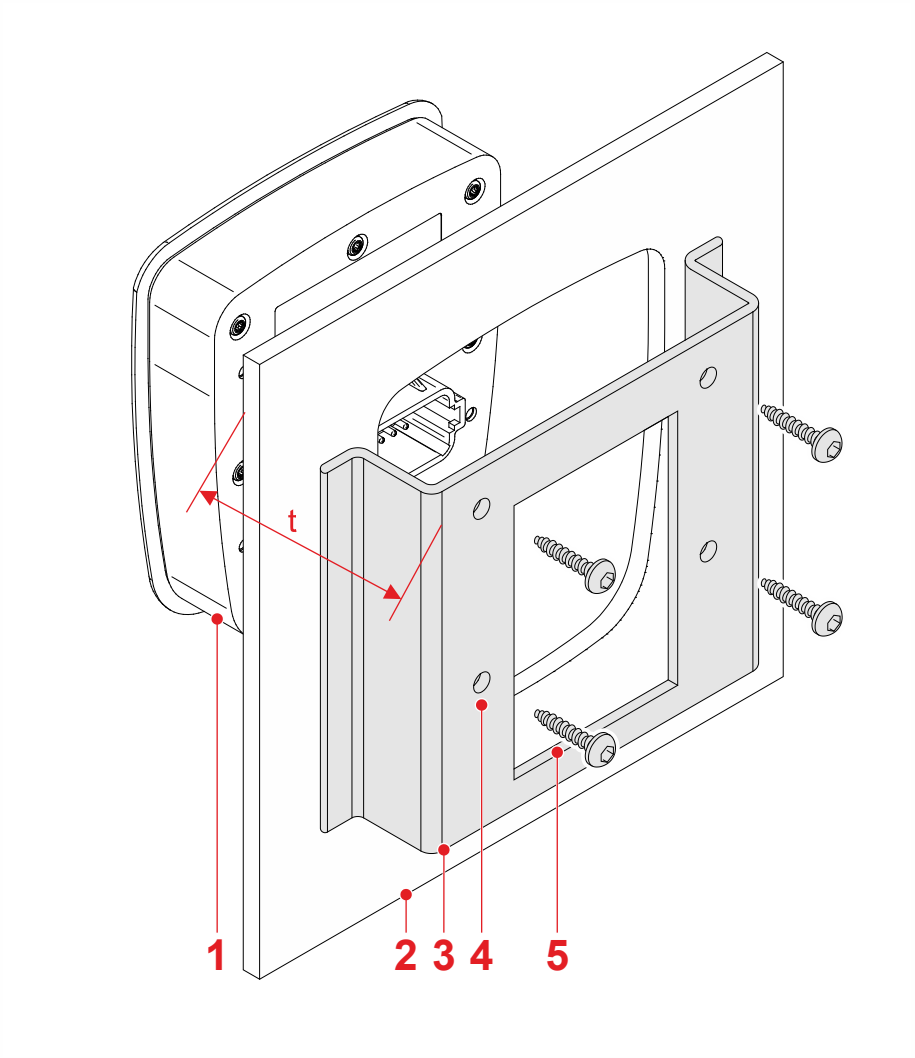
Montage vorbereiten

Stellen Sie eine passende Öffnung in der Montageplatte her.
Die Abbildung zeigt die Form der Öffnung und die Maße in Millimeter.



Bediengerät montieren

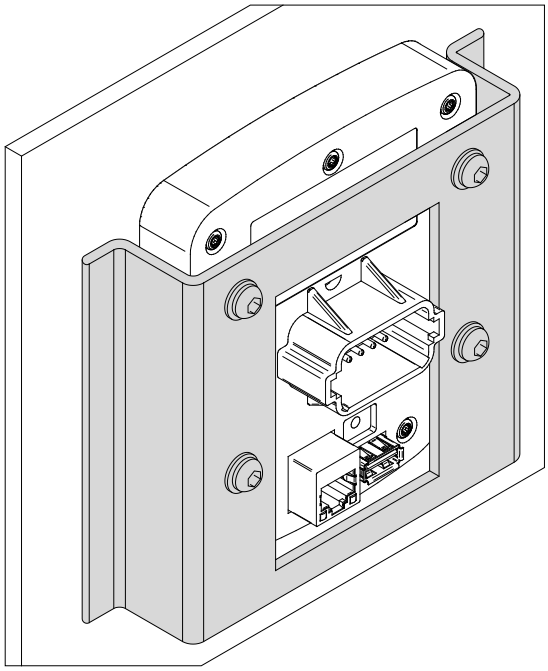
Die Abbildung zeigt die Montage.



Nummer	Beschreibung
1	JVM-104
2	Montageplatte mit Öffnung für das Bediengerät
3	Winkel zur Befestigung mit Öffnung für die Stecker
4	4 x Bohrung zur Befestigung des JVM-104
5	4 x selbstfurchende Schraube Schraubenmaß: 4 x 9 + t Anzugsmoment: 1,6 Nm ± 10 % Maximale Schraubtiefe: 12 mm

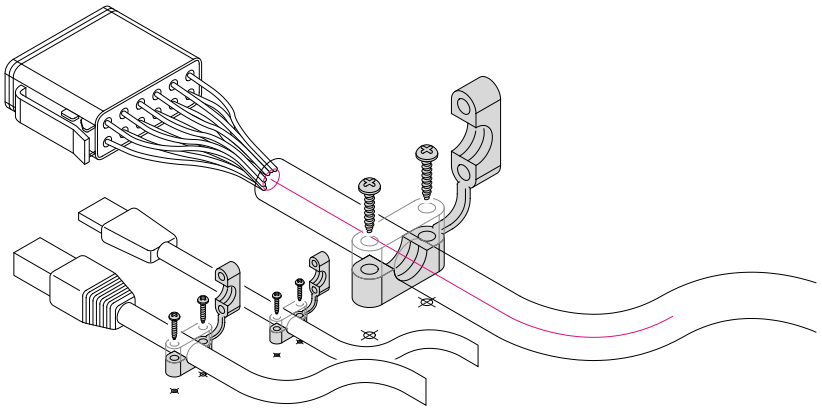
Schritt	Vorgehen
1	Stecken Sie das Bediengerät von vorn in die Öffnung der Montageplatte.
2	Halten Sie den Befestigungswinkel von hinten an die Montageplatte. Dabei müssen die Stecker durch die Öffnung des Befestigungswinkels schauen.
3	Schrauben Sie das Bediengerät zusammen mit dem Befestigungswinkel auf die Montageplatte. Das Anzugsmoment sollte 1,6 Nm ± 10 % betragen.

Die Abbildung zeigt das montierte Bediengerät JVM-104.



Zugentlastung montieren

Montieren Sie Zugentlastungen für die Anschlusskabel.
Achten Sie auf ausreichenden Abstand zu den Steckern.
Die Stecker müssen sich im Servicefall ungehindert abziehen lassen.



4.3 IP-Konfiguration

Einleitung

Dieses Kapitel beschreibt die IP-Konfiguration bei dem Bediengerät JVM-104. Folgende Parameter sind einstellbar:

- IP-Adresse des Bediengeräts
- Subnetzmaske
- IP-Adresse des Default Gateway
- IP-Adresse des DNS-Servers
- Name des Bediengeräts
- IP-Portnummer für den Debugger von JetSym
- IP-Basisportnummer für die JetIP-Kommunikation

Voraussetzungen an den Projektierer

Voraussetzung für die IP-Konfiguration des Bediengeräts JVM-104 sind Kenntnisse über IP-Netzwerke wie:

- IP-Adressierung (IP-Adresse, Port-Nummer, Subnetzmaske, usw.)

Inhalt

Thema	Seite
Auslieferungszustand	52
Der Konfigurationsspeicher	53
Die Konfigurationsregister	54
IP-Adresse zur Laufzeit einstellen	55
Einstellungen über die Gerätewebseite tätigen	56

Auslieferungszustand

Einleitung

Vor der Auslieferung des Bediengeräts JVM-104 sind verschiedene Parameter auf einen bestimmten Wert eingestellt.

Diese Parameter sind durch den Anwender änderbar.

Auslieferungszustand

Parameter	Wert
IP-Adresse des Bediengeräts	192.168.10.15
Subnetzmaske	255.255.255.0
IP-Adresse des Default Gateway	0.0.0.0
IP-Adresse des DNS-Servers	0.0.0.0
Name des Plattform	JVM-C02
IP-Portnummer für den Debugger	52000
IP-Portnummer für JetIP	50000
Passwort des Benutzers *admin*	admin

Der Konfigurationsspeicher

Einleitung

Die Parameter für die Initialisierung der IP-Schnittstelle liest das Bediengerät während der Boot-Phase aus dem Konfigurationsspeicher. Auf die Daten des Konfigurationsspeichers greift der Anwender auf folgende Arten zu:

- Über das Web-Interface lassen sich die Daten auslesen und ändern.
- Über Register lassen sich die Daten auslesen und ändern.

Wirksamkeit

Das Bediengerät liest die Daten im Konfigurationsspeicher nur während der Boot-Phase aus. Änderungen im Konfigurationsspeicher erfordern, dass Sie das Bediengerät anschließend neu booten. Nur so werden dann diese Änderungen auch wirksam.

Wenn Sie die IP-Adresse zur Laufzeit ändern, ist die Änderung sofort danach wirksam. Die Einstellung zur Laufzeit ist nicht-remanent.

Standardwerte

Bevor das Bediengerät Daten aus dem Konfigurationsspeicher weiter verarbeitet, prüft das Bediengerät sie auf Plausibilität. Bei ungültigen oder nicht vorhandenen Einträgen nimmt das Bediengerät folgende Standardwerte:

Parameter	Standardwert
IP-Adresse der Steuerung	192.168.10.15
Subnetzmaske	255.255.255.0
IP-Adresse des Default Gateway	0.0.0.0
IP-Adresse des DNS-Servers	0.0.0.0
Name der Plattform	JVM-C02
Suffixtyp des Namens	0
IP-Portnummer für den Debugger	52000
IP-Portnummer für JetIP	50000

Verwandte Themen

- **Konfigurationsregister** (siehe Seite 54)

Die Konfigurationsregister

Einleitung

Die Parameter der IP-Konfiguration sind über die Konfigurationsregister lesbar und schreibbar. Ein Bereich von Registern enthält die Daten aus dem Konfigurationsspeicher. Ein weiterer Bereich enthält die zur Initialisierung der IP-Schnittstelle verwendeten Parameter.

Registernummern

Die Registernummer ergibt sich aus der Addition der Modulregisternummer (MR) zur Basisregisternummer.

Bediengerät	Datenbereich	Zugriff	Basisregister-nummer	Register-nummern
JVM-104	Konfigurations-speicher	lesend/ schreibend	101100	101100 ... 101165
	Aktuell verwende-te Parameter	Nur lesend	101200	101200 ... 101265

Konfigurationsregister

Die folgende Tabelle zeigt eine Übersicht über die Register der beiden Bereiche und ihre Funktion:

Register	Beschreibung
MR 0	IP-Adresse des Geräts
MR 1	Legt die Subnetzmaske fest
MR 2	IP-Adresse des Gateways in andere Subnetze
MR 3	IP-Adresse des Domain-Name-Servers
MR 32	Typ des automatisch generierten Anhangs an den Namen des Geräts
MR 33 bis MR 51	Legt den Namen des Geräts fest
MR 64	IP-Port für das Betriebssystemupdate und die Kommunikation zwischen den Geräten
MR 65	IP-Port für Debugger/Setup in JetSym

Verwandte Themen

- **Der Konfigurationsspeicher** (siehe Seite 53)

IP-Adresse zur Laufzeit einstellen

Einleitung

Die IP-Schnittstelle wird während der Boot-Phase über die Einstellungen im Konfigurationsspeicher initialisiert.

Folgende Einstellungen sind auch nicht-remanent über Register änderbar:

- IP-Adresse des Bediengeräts
- Subnetzmaske
- IP-Adresse des Default Gateway

Wichtiger Hinweis

Die Einstellungen zur Laufzeit verändern nicht die Parameter im Konfigurationsspeicher. Wenn Sie die Steuerung ausschalten, sind die Einstellungen wieder verloren.

Registerübersicht

Register	Beschreibung
104531	IP-Adresse des Bediengeräts
104532	Subnetzmaske
104533	IP-Adresse des Default Gateway

IP-Adressen und die Subnetzmaske einstellen

Zur Änderung der IP-Adressen und der Subnetzmaske gehen Sie wie folgt vor:

Schritt	Vorgehen
1	Beschreiben Sie 104533 mit dem Wert 0.0.0.0.
2	Beschreiben Sie 104532 mit dem Wert 0.0.0.0.
3	Beschreiben Sie 104531 mit der gewünschten IP-Adresse des JVM-104.
4	Beschreiben Sie 104532 mit der gewünschten Subnetzmaske.
5	Beschreiben Sie 104533 mit der gewünschten IP-Adresse des Default Gateways.

Ergebnis:

Die Einstellung ist beendet. Die Kommunikation ist wieder möglich.

Verwandte Themen

- **Der Konfigurationsspeicher** (siehe Seite 53)

Einstellungen über die Gerätewebseite tätigen

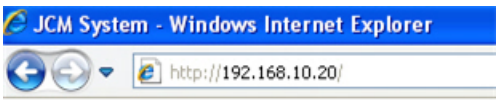
Einleitung

Auf der Gerätewebseite im Internet-Explorer sehen Sie alle Eigenschaften und Versionsnummern Ihres Geräts. Die Konfiguration der Ethernet-Schnittstelle und CANopen®-Schnittstelle ist auch möglich.

Aufruf der Gerätewebseite

Wenn Sie im Internet-Explorer die **aktuelle** IP-Adresse des Geräts eingeben, wird auf dem PC die Gerätewebseite angezeigt.

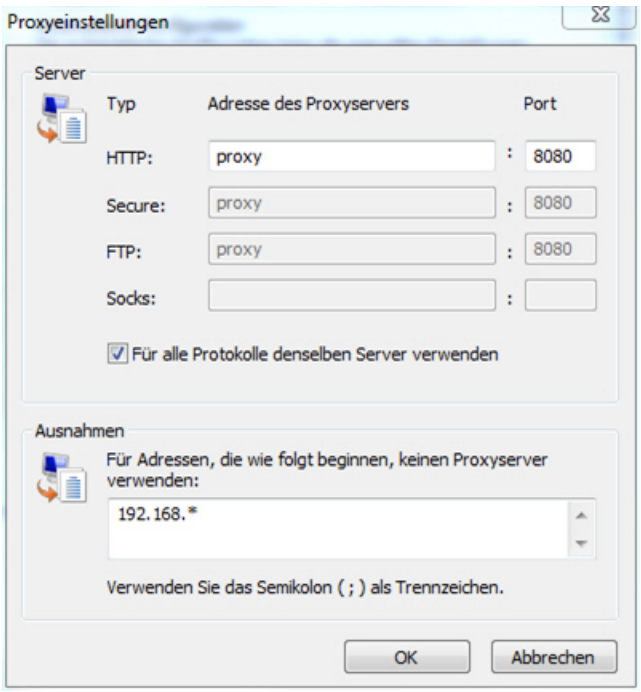
Beispiel:



Voraussetzungen

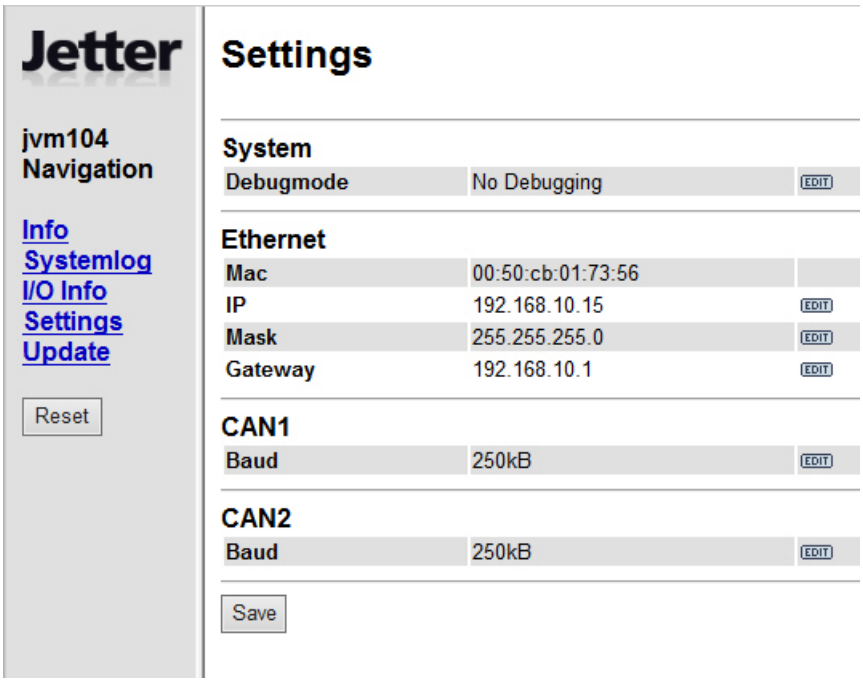
Um im Internet Explorer die IP-Adresse bei den Ausnahmen einzugeben, gehen Sie wie folgt vor:

Schritt	Vorgehen
1	Öffnen Sie über das Menü **Extras** das Fenster **Internetoptionen**.
2	Klicken Sie auf die Schaltfläche **LAN-Einstellungen** auf dem Reiter **Verbindungen**.
3	Klicken Sie auf die Schaltfläche **Erweitert** unter **Proxyserver**.
⇨	Das Fenster **Proxyeinstellungen** hat sich geöffnet.
4	Geben Sie im Fenster **Proxyeinstellungen** bei den **Ausnahmen** die IP-Adresse **192.168.*** ein, siehe die folgende Abbildung.



Settings-Seite

Auf der Seite **Settings** konfigurieren Sie die Ethernet-Schnittstelle und die CANopen®-Schnittstelle.



Die folgende Schritttabelle zeigt exemplarisch die Einstellung der IP-Adresse:

Schritt	Vorgehen
1	Klicken Sie auf die Schaltfläche **Edit** in der Zeile **IP**.
⇨	Es öffnet sich das Eingabefeld für die IP-Adresse. IP 192.168.10.20
2	Geben Sie in das editierbare Feld die neue IP-Adresse ein.
3	Bestätigen Sie die Eingabe mit einem Klick auf die Schaltfläche . Um Ihre Eingabe rückgängig zu machen, klicken Sie auf die Schaltfläche .
4	Klicken Sie auf die Schaltfläche **Save**.
⇨	Alle Eingaben sind jetzt gespeichert.
5	Schalten Sie das Gerät aus und dann wieder ein.

Ergebnis: Das Gerät ist jetzt über die neue IP-Adresse erreichbar.

5 Erstinbetriebnahme

Zweck des Kapitels

Dieses Kapitel beschreibt die Erstinbetriebnahme des Geräts JVM-104 anhand der folgenden Schritte:

- Erstinbetriebnahme in JetViewSoft
- Erstinbetriebnahme in JetSym

JetViewSoft ist ein Scada-System und JetSym ein Programmiertool, beides von der Jetter AG entwickelt.
Für weitere Details nutzen Sie die Online-Hilfe von JetSym und JetViewSoft.

Mindestanforderungen

Die Anleitung zur Erstinbetriebnahme bezieht sich auf JetSym ab der Version 5.1.2 und JetViewSoft ab der Version 4.0.2.

Inhalt

Thema	Seite
Vorbereitung und Einstieg in die Programmierung mit JetSym STX	60
Konfiguration bei der Plattform ER-STX-CE	63
Programmierung bei der Plattform ER-STX-CE	77

5.1 Vorbereitung und Einstieg in die Programmierung mit JetSym STX

Einleitung

Dieses Kapitel stellt die Vorbereitungen zur Erstinbetriebnahme vor. Es gibt auch einen ersten Einblick in die Programmiersprache JetSym STX.

Inhalt

Thema	Seite
Vorbereitungen zur Erstinbetriebnahme	61
Programmierung in der Sprache JetSym STX	62

Vorbereitungen zur Erstinbetriebnahme

Ethernet-Verbindung zur Steuerung

Das Bediengerät JVM-104 hat im Auslieferungszustand die IP-Adresse 192.168.10.15. Konfigurieren Sie die Ethernet-Schnittstelle Ihres PCs, dass er mit dem JVM-104 über diese IP-Adresse kommuniziert.

Voraussetzung für das Einschalten

Nur wenn an der Zündung (+) die Versorgungsspannung +UB anliegt, schaltet das JVM-104 ein.

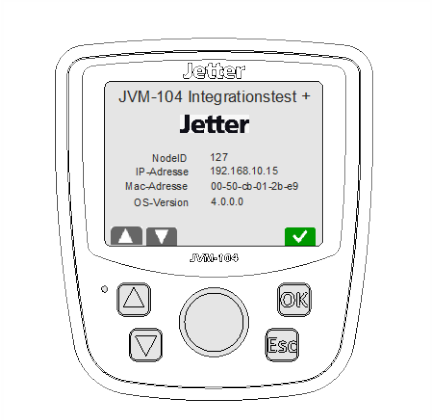
Start ohne Anwendungsprogramm

Wenn Sie beim Einschalten die Tasten ▼ und **OK** gleichzeitig drücken, startet das Anwendungsprogramm nicht.

Wenn das Gerät nach dem Booten nicht mehr reagiert, wie es während der Softwareentwicklungsphase passieren kann, ermöglicht dieser Zustand den Zugriff über FTP und über JetSym.

Darstellung bei Auslieferung

Das Anwendungsprogramm, das bei der Auslieferung nach dem Einschalten auf dem JVM-104 startet, stellt die folgende Eingabemaske auf dem Display des JVM-104 dar.



Die dargestellte Node-ID ist die im JVM-104 eingestellte Adresse des Busses CANopen® 1. Diese Adresse ist über die Tasten ▲ und ▼ einstellbar.

Über die Taste ▲ wird die Adresse in Einerschritten erhöht.

Über die Taste ▼ wird die Adresse in Einerschritten verringert.

Zusätzlich wird noch die IP-Adresse, MAC-Adresse und OS-Version dargestellt.

Programmierung in der Sprache JetSym STX

Einleitung

Mit JetViewSoft erstellen Sie Visualisierungsapplikationen für folgende Zielplattformen:

- PC-Systeme
- Bediengeräte für den Industriebereich
- Bediengeräte für den Mobilbereich

Sie greifen mit JetSym STX auf Visualisierungsobjekte zu und steuern die Darstellung auf dem Bediengerät. Sie programmieren das Bediengerät mit der Sprache JetSym STX wie eine Steuerung. Die kompilierten Programme können Sie dann ohne eine externe Steuerung im Bediengerät abarbeiten lassen. Der STX-Interpreter und die JVER (JetView Embedded Runtime) als grafische Laufzeitumgebung ermöglichen das. Beide sind Betriebssystembestandteile des Bediengeräts.

JetSym-STX-Programm

Das untenstehende Programm führt nichts Sinnvolleres aus, als eine interne Variable in einer Schleife bis zum Wert 20 zu verdoppeln. Das Beispiel zeigt exemplarisch, wie Sie JetSym STX nutzen können.

```
Task MiniBeispiel AutoRun
Var
    i, j : Int;
End_Var;
    j := 1;
    // j wird in einer Schleife bis zu dem Wert 1024 durchlaufen
    For i := 1 To 10 Do
      j := j * 2;
    End_For;
End_Task;
```

Speicherort des Programms

Wenn Sie das kompilierte Programm auf das Bediengerät laden, legt JetSym im Verzeichnis *\App* einen Ordner mit dem Projektnamen an. In diesen Ordner speichert JetSym das Anwendungsprogramm. Der Dateiname des Anwendungsprogramms besteht aus dem Projektnamen und der Dateierweiterung **.es3**. Pfad- und Dateinamen werden dabei immer in Kleinbuchstaben konvertiert.

Die Datei **start.ini** definiert, welches Anwendungsprogramm geladen werden soll, und wird automatisch beim Programmdownload erstellt.

5.2 Konfiguration bei der Plattform ER-STX-CE

Einleitung

Dieses Kapitel beschreibt, wie Sie ein Visualisierungsprojekt der Plattform ER-STX-CE in JetViewSoft und JetSym anlegen und konfigurieren.

Inhalt

Thema	Seite
Visualisierungsprojekt in JetViewSoft anlegen und konfigurieren	64
Visualisierungsprojekt in JetSym anlegen und konfigurieren	69

Visualisierungsprojekt in JetViewSoft anlegen und konfigurieren

Einleitung

Mit JetViewSoft werden die Visualisierungsdateien für das Gerät JVM-104 erstellt und auf das Gerät übertragen. An dieser Stelle ist Folgendes beschrieben:

- Ein Projekt in JetViewSoft anlegen
- Die Projekteinstellungen festlegen
- Visualisierungsdateien erstellen und auf das Gerät übertragen

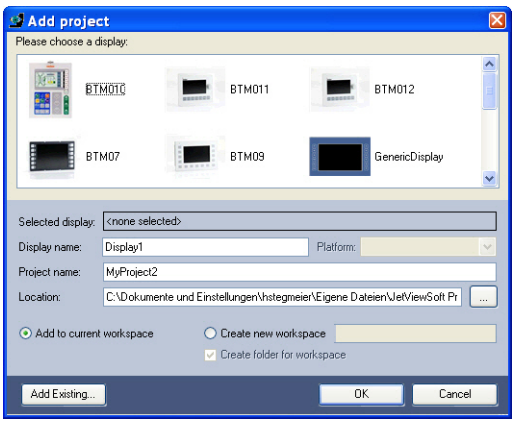
Voraussetzungen

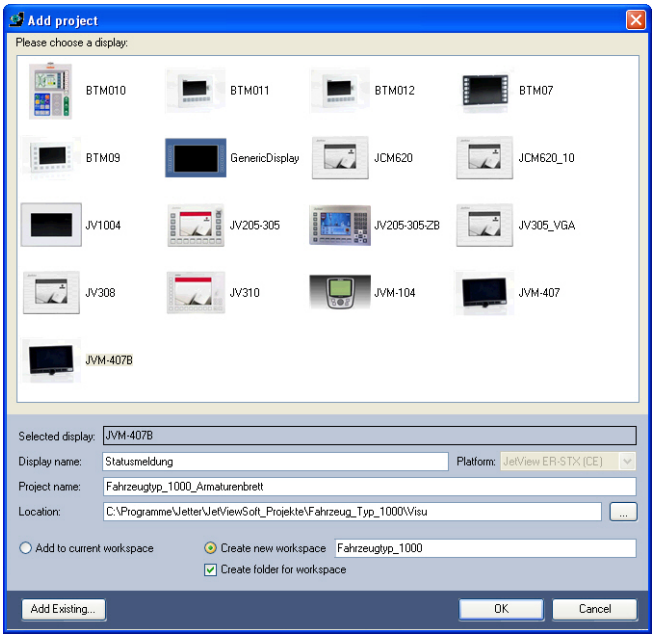
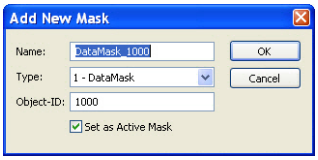
Erfüllen Sie folgende Voraussetzungen:

- Sie haben JetViewSoft auf dem verwendeten PC installiert.
- Sie haben JetViewSoft lizenziert (siehe Online-Hilfe in JetViewSoft).
- Eine aktive Ethernet-Verbindung zwischen dem PC und Ihrem Gerät ist eingestellt.

Projekt anlegen

So legen Sie in JetViewSoft ein neues Projekt für das Bediengerät an:

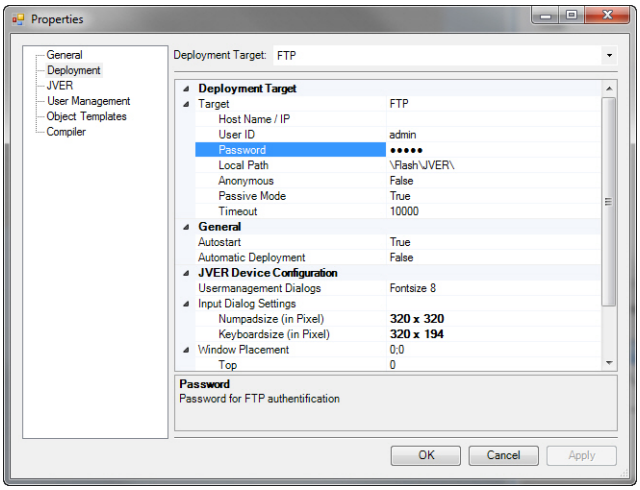
Schritt	Vorgehen
1	Starten Sie JetViewSoft
2	Öffnen Sie das Menü **File**. Wählen Sie den Menüpunkt **New Project** aus. **Ergebnis:** Der folgende Dialog öffnet sich: 
3	Wählen Sie bei **Selected display** das passende Gerät aus, indem Sie einmal auf das Symbol des Geräts klicken.
4	Wählen Sie bei **Display name** einen programminternen Namen für das Gerät aus. Sie können in einem Projekt mehrere Displays anlegen.
5	Wenn Sie eine Auswahlmöglichkeit haben, wählen Sie **Platform JetView ER-STX(CE)** aus.
6	Geben Sie bei **Project name** den Namen des Projekts ein.

Schritt	Vorgehen
7	Passen Sie, wenn erforderlich, bei **Location** den Pfad des Projekts an. Für eine bessere Übersichtlichkeit sollte der Pfad mit *\Visu* enden.
8	Geben Sie bei **Workspace** den Namen des Workspace ein.
⇨	Die folgende Abbildung zeigt einen möglich ausgefüllten Dialog:  **Wichtiger Hinweis:** Die Namen dürfen keine Leerzeichen enthalten, da sich sonst die erstellten Visualisierungsdateien nur schwer wieder löschen lassen.
9	Bestätigen Sie die Einstellungen mit **OK**. **Ergebnis:** Der Dialog schließt sich und der Dialog **Add New Mask** öffnet sich. 
10	Geben Sie bei **Name** den Namen der ersten DataMask an. Die anderen Einstellungen können Sie beibehalten. Diese Maske wird automatisch beim Start des Geräts die aktive Maske.
11	Bestätigen Sie mit **OK**.

Ergebnis: Ein Projekt ist angelegt.

Deployment konfigurieren

Um die mit JetViewSoft erstellten Dateien auf das Gerät zu übertragen, müssen Sie noch die nötigen Deployment-Einstellungen treffen.

Schritt	Vorgehen
1	Öffnen Sie das Menü **Project**. Wählen Sie den Menüpunkt **Properties** aus. **Ergebnis**: Der gleichnamige Dialog öffnet sich.
2	Öffnen Sie das Fenster **Deployment** über die Navigation auf der linken Seite des Dialogs. 
3	Wählen Sie bei **Deployment Target** den Menüpunkt **FTP** aus, indem Sie mit der Maus auf **Target** klicken.
4	Klappen Sie die **Target**-Einstellungen über das +-Zeichen daneben auf oder mit einem Doppelklick direkt auf **Target**.
5	Geben Sie bei **Host Name/IP** die IP-Adresse des Bediengeräts ein. Default-IP-Adresse eines JVM-104 ist **192.168.10.15**.
6	Geben Sie bei **User ID** den Namen des Users ein. Default-User ist *admin*.
7	Geben Sie bei **Password** das Passwort ein. Default-Passwort ist *admin*.
8	Geben Sie bei **Local Path** *\App\projektname* ein. Dabei ist *projektname* ein Platzhalter für den aktuellen Namen des Projekts in Kleinbuchstaben.
9	Bestätigen Sie die Einstellungen mit **OK**.

Ergebnis: Jetzt sind die Deployment-Einstellungen getroffen und Sie können die Dateien auf das Gerät übertragen.

Dateien für das Gerät

Um ein JetViewSoft-Projekt zu erzeugen und die Daten auf das Gerät zu übertragen, gehen Sie wie folgt vor:

Schritt	Vorgehen
1	Gestalten Sie die Maske mit den zur Verfügung stehenden Objekten (Rechtecke, Ellipsen usw.). Sie sehen die Objekte nach der Übertragung auf dem Bediengerät.
2	Öffnen Sie das Menü **File**. Wählen Sie den Menüpunkt **Save all** aus.
3	Drücken Sie die Taste **[F7]** für einen Projekt-Build. **Ergebnis:** JetViewSoft erstellt Projektdateien, sofern kein Fehler auftritt.
4	Öffnen Sie das Menü **Build**. Wählen Sie den Menüpunkt **Deploy** aus. Eine andere Möglichkeit ist, die Tastenkombination **[CTRL] + [F5]** einzugeben. **Ergebnis:** JetViewSoft überträgt die Dateien auf das Gerät.
5	Damit das Gerät die Dateien einliest, starten Sie das Gerät neu.

Ergebnis: Jetzt sehen Sie die JetViewSoft-Dateien des Projekts auf dem Gerät im Verzeichnis *\App\projektname*. Das Gerät zeigt jetzt die Startmaske an.

Keine Visualisierungsapplikation auf dem Gerät

Wenn keine Visualisierungsapplikation auf dem Gerät ist, zeigt das Display Folgendes an:



Der Ordner **Data** enthält keine Dateien. Somit ist keine Visualisierungsapplikation und keine JVER (JetView Embedded Runtime) auf dem Gerät. Eine Kommunikation zu JetSym ist bei angehaltener Runtime (Desktophintergrund ist sichtbar) nicht möglich.

Abhilfe: Übertragen Sie mit JetViewSoft eine Visualisierungsapplikation auf das Gerät.

IOP-Datei als Visualisierungsapplikation auf dem Gerät

Bei der Auslieferung ist es möglich, dass als Visualisierungsapplikation eine Datei mit der Dateiendung **.iop** im Ordner **Data** gespeichert ist.

Bei der Applikation, bei der die Node-ID des CAN-Busses eingestellt wird, ist das dann auch der Fall.

In diesem Fall: Das Gerät stellt Ihre erstellte Visualisierungsapplikation nicht dar.

Abhilfe:

Schritt	Vorgehen	
1	**Wenn ...**	**... dann ...**
	... die Datei **\App\visual.iop** oder **\Data\visual.iop** vorhanden ist,	... löschen Sie die Datei oder benennen sie diese um.
2	**Wenn ...**	**... dann ...**
	... die Datei **\App\JetViewERS.cfg** vorhanden ist,	... löschen Sie die Datei oder benennen sie diese um.
⇨	Die Visualisierungsapplikation entwickelt auf der Plattform ER-STX-CE wird dargestellt.	

Verwandte Themen

- **Erstinbetriebnahme in JetSym** (siehe Seite 69)

Visualisierungsprojekt in JetSym anlegen und konfigurieren

Einleitung

Sie erstellen mit JetSym das STX-Programm für die Visualisierung des Bediengeräts JVM-104. An dieser Stelle ist Folgendes beschrieben:

- Projekt in JetSym STX anlegen
- Hardware der Steuerung konfigurieren
- Einbinden der Visualisierungsbibliothek JVER-STX
- Kompilierfähiges Programm erstellen und auf das Bediengerät übertragen

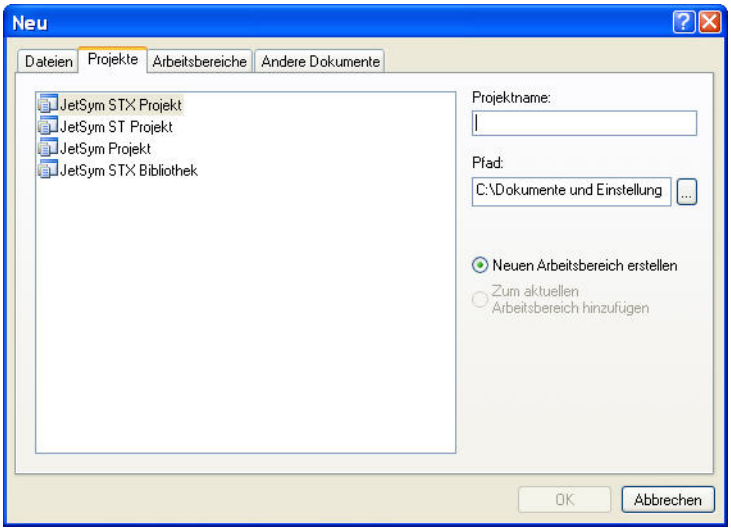
Voraussetzungen

Folgende Voraussetzungen müssen gegeben sein:

- JetSym ist auf dem verwendeten PC installiert.
- JetSym wurde lizenziert (siehe Online-Hilfe in JetSym).
- Sie haben eine Steuerung an das Netzwerk des PCs angeschlossen.
- Eine aktive Ethernet-Verbindung zwischen PC, Bediengerät und Steuerung ist eingerichtet.
- Die Erstinbetriebnahme in JetViewSoft wurde durchgeführt.

Projekt anlegen

So legen Sie in JetSym ein neues Projekt für die Programmierung an:

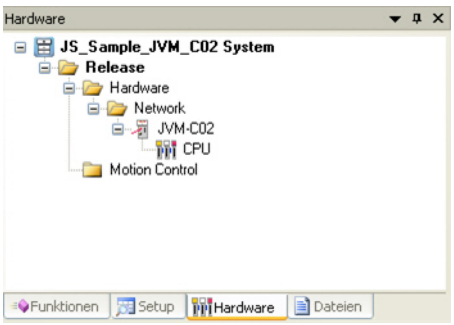
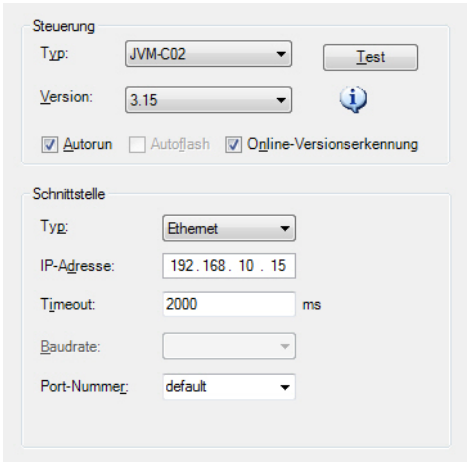
Schritt	Vorgehen
1	Starten Sie JetSym.
2	Öffnen Sie das Menü **Datei**. Wählen Sie den Menüpunkt **Neu** aus. **Ergebnis:** Der Dialog **Neu** öffnet sich. 
3	Wählen Sie als Projekttyp **JetSym STX Projekt** aus.
4	Geben Sie den Projektnamen ein.

Schritt	Vorgehen
5	Wählen Sie den Pfad aus. Es ist empfehlenswert, die Projektdateien innerhalb des JetViewSoft-Projekts im Ordner **STX** abzuspeichern. **Beispiel eines Pfads:** *C:\Programme\Jetter\JetViewSoft_Projekte\Fahrzeugtyp_1000\Visu\Fahrzeugtyp_1000\Fahrzeugtyp_1000_Armaturenbrett\STX* **Vorteil:** Die JetSym-Projektdaten sind in demselben Ordner abgespeichert wie die von JetViewSoft erzeugte Datei **VisualInterface.stxp**.
6	Bestätigen Sie die Einstellungen mit **OK**.

Ergebnis: Ein Projekt ist angelegt.

Hardware konfigurieren

Um eine Verbindung zwischen JetSym und dem Bediengerät aufzubauen, müssen Sie die Hardware konfigurieren.

Schritt	Vorgehen
1	Wechseln Sie in die Ansicht **Hardware** über die gleichnamige Registerkarte. 
2	Klappen Sie den **Hardwarebaum** komplett auf.
3	Wenn Sie als Hardware JVM-C02 oder Parameter zur Schnittstelle einstellen wollen, dann doppelklicken Sie auf **CPU**. **Ergebnis:** Das Fenster **Konfiguration** öffnet sich. 
4	Wählen Sie bei **Steuerung/Typ** JVM-C02 aus.

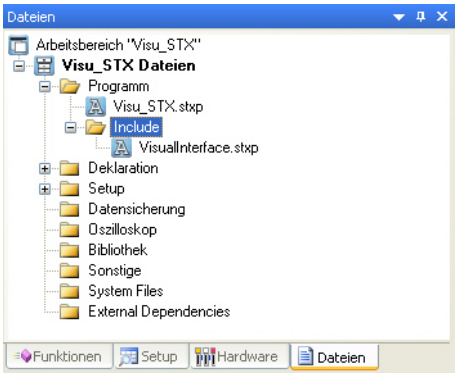
Schritt	Vorgehen
5	Wählen Sie bei **Schnittstelle/Typ** Ethernet aus.
6	Geben Sie bei **Schnittstelle/IP-Adresse** die IP-Adresse des Geräts ein.
7	Testen Sie die Verbindung bei laufender JVER, in dem Sie auf die Schaltfläche **Test** klicken. Wenn keine Verbindung vorhanden ist, prüfen Sie die IP-Adresse und die Ethernet-Verbindung zur JVM-104.
8	Speichern Sie die Einstellungen mit der Tastenkombination **[Strg] + [S]** ab.

Ergebnis: Die Hardwareeinstellungen in JetSym sind konfiguriert.

VisualInterface.stxp - In das Projekt einbinden

Damit die Bezeichnung der Objekte und Masken der Visualisierungsapplikation bei der Programmierung zur Verfügung stehen, müssen Sie die Datei **Visualinterface.stxp** folgendermaßen einbinden:

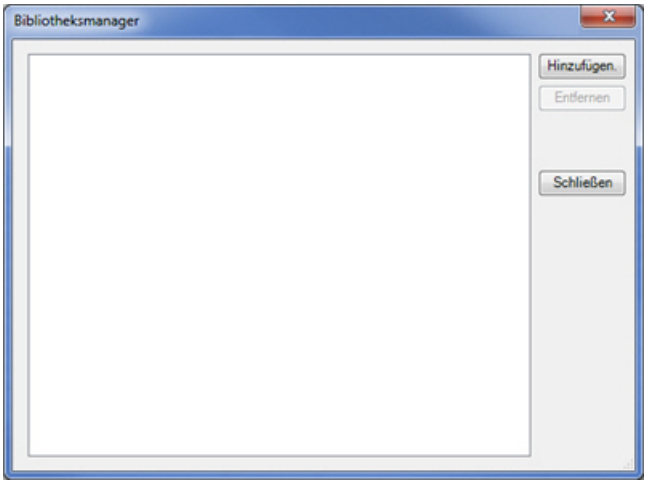
Schritt	Vorgehen
1	Wechseln Sie in die Ansicht **Dateien**. 
2	Klappen Sie den Ordner **Programm** auf.
3	Markieren Sie den Ordner **Include** und öffnen Sie das Kontextmenü (rechte Maustaste).
4	Wählen Sie im Kontextmenü den Eintrag **Dateien dem Verzeichnis hinzufügen** aus. **Ergebnis:** Ein Explorer-Fenster zur Auswahl einer Datei öffnet sich.
5	Navigieren Sie zu dem Ordner **STX** des JetViewSoft-Projekts. Der Ordner ist zu finden unter *[Projektpfad]/Name des JVS-Projekts/STX* zu finden.
6	Wählen Sie die Datei **VisualInterface.stxp** aus.

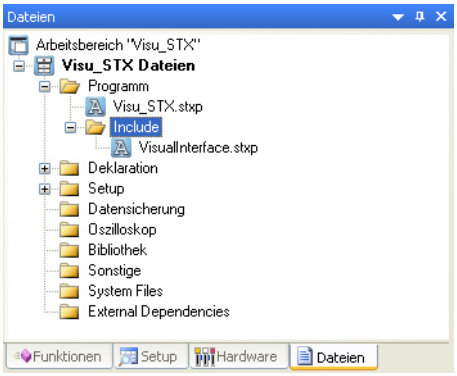
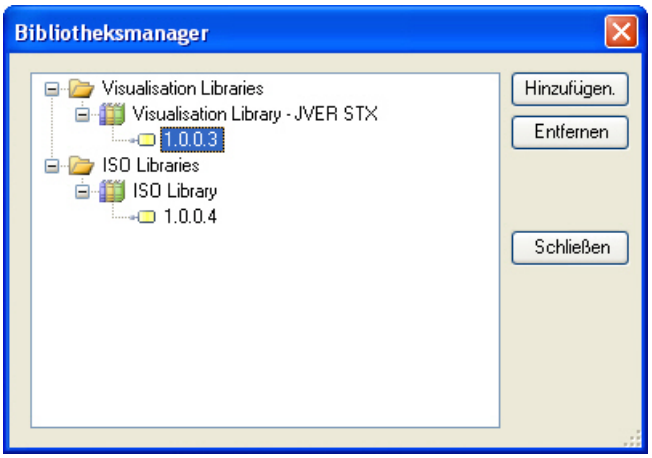
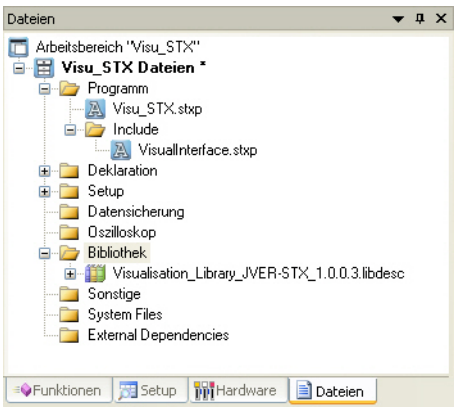
Schritt	Vorgehen
7	Klicken Sie auf die Schaltfläche **Öffnen**. 

Ergebnis: Die Datei **VisualInterface.stxp** ist in das JetSym-Projekt eingebunden.

Bibliothek einbinden

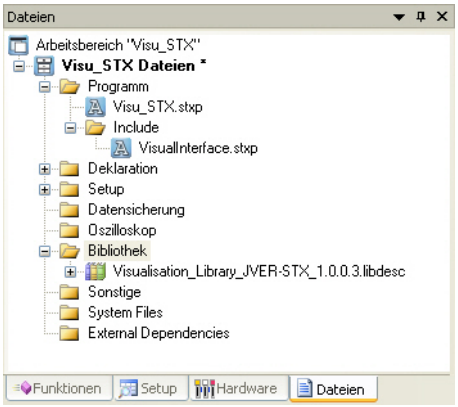
Damit die Bibliothek mit den Visualisierungsfunktionen in JetSym zur Verfügung steht, müssen Sie die Bibliothek wie folgt einbinden:

Schritt	Vorgehen
1	Öffnen Sie das Menü **Extras**. Wählen Sie den Menüpunkt **Bibliotheksmanager** aus.
⇨	Der gleichnamige Dialog öffnet sich. 
2	Klicken Sie auf die Schaltfläche **Hinzufügen**. **Ergebnis:** Ein Explorer-Fenster im Ordner **Lib** der JetSym-Installation.
3	Wählen Sie die Datei **Visualisation_Library_1.0.0.3.libpackage** oder eine entsprechende Version dieser Bibliothek aus.
4	Klicken Sie auf die Schaltfläche **Öffnen**. **Ergebnis:** Die Bibliothek ist im Bibliotheksmanager. Sie können jetzt die Bibliothek ins JetSym-Projekt einbinden.

Schritt	Vorgehen
5	Wechseln Sie in die Ansicht **Dateien**. 
6	Wählen Sie den Ordner **Bibliothek** aus. Öffnen Sie das Kontextmenü mit der rechten Maustaste.
7	Wählen Sie den Menüeintrag **Bibliotheken hinzufügen...** aus. **Ergebnis:** JetSym öffnet den Bibliotheksmanager.
8	Markieren Sie die Visualisierungsbibliothek und klicken Sie auf die Schaltfläche **Auswählen**. 
⇨	

Kompilierfähiges Programm erstellen

So erstellen und kompilieren Sie ein lauffähiges Programm:

Schritt	Vorgehen
1	Wechseln Sie in die Ansicht **Dateien**. 
2	Doppelklicken Sie auf die Programmdatei. Die Programmdatei hat den gleichen Namen wie das Projekt plus die Endung **stxp**. **Ergebnis:** Die Programmdatei im JetSym-Editor öffnet sich.

3	Geben Sie folgenden Programmcode ein.
	Beachten Sie dies bei der Include-Anweisung.
	`#Include "VisualInterface.stxp";`
	
	`Function OnKeyDown(KeyCode:long, Flags:long)`
	`End_Function;`
	`Function OnKeyUp(KeyCode:long, Flags:long)`
	`End_Function;`
	
	`Task Main Autorun`
	`End_Task;`

4	Klicken Sie auf die Taste **[F7]** für einen Build-Vorgang des Projekts. **Ergebnis:** Die Visualisierungsfunktionen und die VisualInterface-Header-Datei stehen bei der Programmierung zur Verfügung.

Ergebnis:

Sie können das Programm jetzt erweitern. Im **IntelliSense** (**Strg- + Space-Taste**) stehen die Visualisierungsfunktionen und die Informationen aus der VisualInterface-Header-Datei zur Verfügung. Sie übertragen das Programm durch die Tastenkombination **[Strg] + [F5]** auf das Bediengerät. Das Programm hat momentan noch keine Funktion.

Funktionen `OnKeyDown` und `OnKeyUp`

Die Funktionen `OnKeyDown` und `OnKeyUp` können bei Taste drücken oder bei Taste loslassen z. B. Visualisierungskommandos aufrufen. Abhängig davon, welche Taste Sie drücken, übermittelt das Betriebssystem einen bestimmten Keycode an die Funktionen.

Die Tasten **[UP]**, **[DOWN]**, **[OK]** und **[ESC]** z. B. haben die folgenden Keycodes:

Taste	Keycode	Konstante
▲	0x26	KEY_UP
▼	0x28	KEY_DOWN
OK	0x0D	KEY_RETURN
ESC	0x1B	KEY_ESCAPE

Die Konstanten von allen möglichen Tasten sind in der Datei **VISU_Defines.stxp** als Keycode definiert. Sie können also im Programm die Konstanten angeben.

Ein mögliches STX-Programm dazu:

```
#include "VisualInterface.stxp"

Function OnKeyDown(KeyCode:LONG, Flags:Long)
Case KeyCode Of
KEY_UP: VisuCmdAttribute(Ellipse_4000,
ELLIPSE_ATTR_FILLATTRIBUTE, FillAttribute_26000);
Break;
KEY_DOWN: VisuCmdAttribute(Ellipse_4000,
ELLIPSE_ATTR_FILLATTRIBUTE, FillAttribute_26001);
Break;
KEY_RETURN: VisuCmdAttribute(Ellipse_4000,
ELLIPSE_ATTR_FILLATTRIBUTE, FillAttribute_26002);
Break;
KEY_ESCAPE: VisuCmdAttribute(Ellipse_4000,
ELLIPSE_ATTR_FILLATTRIBUTE, FillAttribute_26003);
Break;
End_Case;
End_Function;
```

Empfehlenswertes

Empfehlenswert ist es, wenn Sie auch für **Ellipse_4000** und **FillAttribute_26000** aussagekräftigere Objektnamen verwenden. Damit Sie die Objekte wieder finden und ordentlich zuordnen können, ist diese Vorgehensweise sehr ratsam. Statt *FillAttribute_26000* können Sie z. B. *FillAttribute_Weiss* schreiben.

Verwenden Sie in den Objektnamen keine Leer- oder Sonderzeichen (ä, ö, ü, ß, -, ...).

Sie vergeben den Objektnamen in JetViewSoft im Properties-Fenster des jeweiligen Objekts. JetViewSoft trägt diesen Objektnamen und die Objekt-ID in die Datei **VisualInterface.stxp** ein und Sie können den Objektnamen und die Objekt-ID im Programm verwenden.

Verwandte Themen

- **Erstinbetriebnahme in JetViewSoft** (siehe Seite 64)

5.3 Programmierung bei der Plattform ER-STX-CE

Einleitung

Dieses Kapitel ist gegliedert in die folgenden zwei Bereiche:

- Werteingabe über das Digipot am Bediengerät
- Visualisierungsobjekte ändern über die Visualisierungskommandos (VisuCommands) vom Anwendungsprogramm aus

Voraussetzung

Diese Beschreibung trifft für die Plattform JetView ER-STX-CE/PC zu.

Ergänzende Literatur

Ergänzende Literatur finden Sie in den Online-Hilfen der Programme JetSym und JetViewSoft.

Inhalt

Thema	Seite
Werteingabe über das Digipot	78
Visualisierungsobjekt ändern über die Visualisierungskommandos	82

Werteingabe über das Digipot

Einleitung

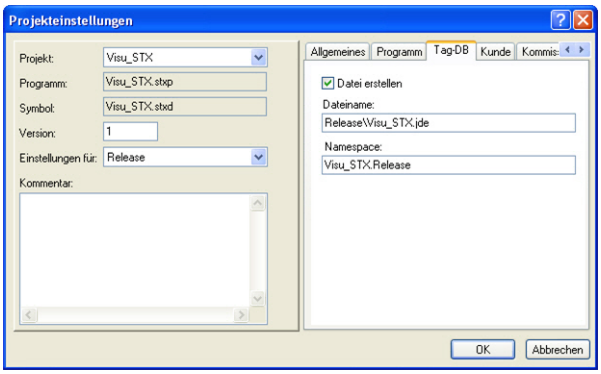
Mit nur wenig Programmcode ist es möglich, mit Hilfe des Digipots am Bediengerät Zahlenwerte einzugeben. Diese Zahlen stehen dann in einem JetSym-STX-Programm zyklisch zur Verfügung.

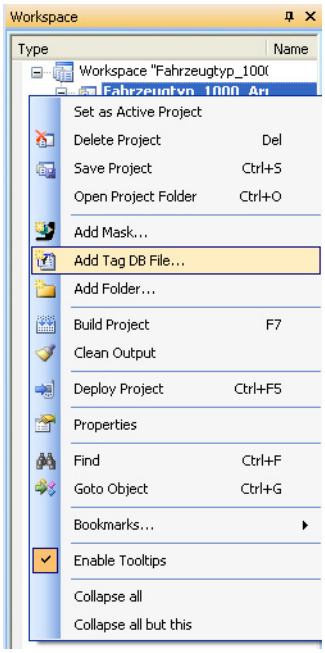
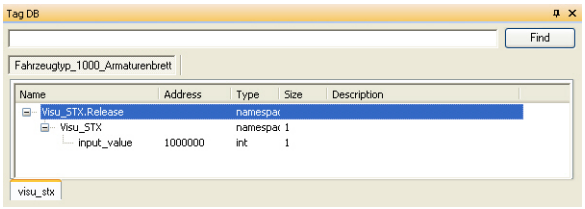
Werteingabe über das Digipot

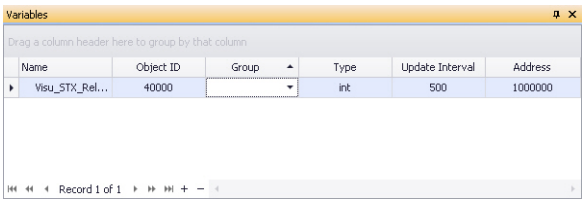
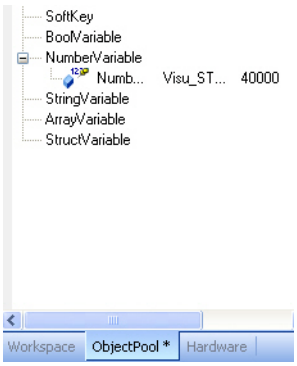
Um einen Wert über das Digipot einzustellen, sind die folgenden Schritte erforderlich:

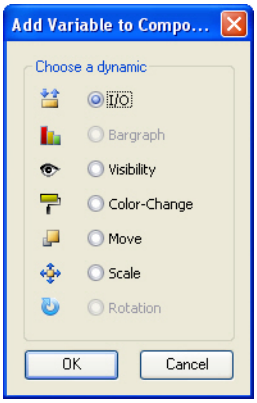
Schritt	Vorgehen

1	Deklarieren Sie die Variablen im JetSym-STX-Programm mit [export jde].
	
	`Task Visu_STX Autorun`
	`Var`
	`input_value: Int At %VL 1000000 [export jde];`
	`End_Var;`
	`End_Task;`

2	Aktivieren Sie in JetSym im Dialog **Projekt - Einstellungen ...** auf der Registerkarte **TagDB** die Checkbox **Datei erstellen**. 
3	Öffnen Sie das Menü **Build**. Wählen Sie den Menüpunkt **Build** aus. Oder drücken Sie die Taste **[F7]**.
⇨	Beim Build-Vorgang erstellt JetSym dann eine TagDB-Datei mit der Endung .jde. Diese Datei enthält die Deklaration der Variablen. **Speicherort:** JetSym erstellt einen Ordner namens **Release.** Hier speichert JetSym die Datei.

Schritt	Vorgehen
4	Binden Sie die TagDB-Datei in Ihr JetViewSoft-Visualisierungsprojekt ein. Rufen Sie dazu im Workspace das Kontextmenü auf und klicken Sie auf **Add TagDBFile...** . 
⇨	JetViewSoft stellt die in der TagDB-Datei deklarierten Variablen im Fenster TagDB dar. 
5	Ziehen Sie per Drag&Drop eine Variable in das Fenster **Variables**.

Schritt	Vorgehen
⇨	JetViewSoft legt eine Variable vom Typ *Number Variable* als Objekt an. Ansicht des Fensters **Variables**:  Ansicht der Registerkarte **ObjectPool**:  Wenn die TagDB-Datei Steuerungsinformationen enthält, legt JetViewSoft automatisch eine Steuerung an und verknüpft die Variablen.
6	Ziehen Sie per Drag&Drop die Variable aus dem Fenster **Variables** oder **ObjectPool** über das Visualisierungsobjekt **Edit Numeric**. Wenn das Pluszeichen erscheint, lassen Sie die Maustaste los. **Ergebnis:** Die Verbindung zwischen dem Visualisierungsobjekt und der Variablen ist hergestellt.

Schritt	Vorgehen
7	Wählen Sie die passende Dynamik aus, z. B. die I/O-Dynamik.  **Ergebnis:** JetViewSoft ergänzt in den Properties des Objekts die Variable und die Dynamik.
8	Führen Sie in JetSym und JetViewSoft einen **Build** und **Deploy** durch. **Ergebnis:** Die Applikation steht im Bediengerät zur Verfügung.
9	Starten Sie das Bediengerät neu.

Ergebnis:

Das Bediengerät zeigt die Startmaske an. Sie können über den Digipot einen Wert im Objekt **Edit Numeric** einstellen.

Wenn Sie im Setup-Fenster von JetSym das R 1000000 eingeben, stellt JetSym den eingestellten Wert dar.

Visualisierungsobjekt ändern über die Visualisierungskommandos

Einleitung

Die Visualisierungskommandos sind Funktionen aus der Visualisierungsbibliothek von JetSym. Diese Funktionen können im JetSym-STX-Programm aufgerufen werden. Mit den Visualisierungskommandos manipulieren Sie Visualisierungsobjekte direkt aus dem JetSym-STX-Programm. Im Folgenden wird gezeigt, wie Sie z. B. die Füllfarbe einer Ellipse über das Visualisierungskommando ändern.

Bestandteile der Visualisierungsbibliothek

Alle verfügbaren Kommandos sind in der Datei **VISU_Functions.stxp** deklariert. Vordefinierte Datentypen, wie Farbe als RGB-Wert, Attribute und Tastencodes, sind in der Datei **VISU_Defines.stxp** deklariert. Beide Dateien sind Bestandteil der Visualisierungsbibliothek.

Voraussetzung

Damit der Compiler das folgende Programm ohne Fehler kompiliert, müssen Sie die folgenden Beispiele um den folgenden Programmcode ergänzen:

```
#Include "VisualInterface.stxp";

Function OnKeyDown (KeyCode:long, Flags:long)
End_Function;
Function OnKeyUp (KeyCode:long, Flags:long)
End_Function;
```

Aufgabe 1

Vom Anwendungsprogramm gesteuert soll das Attribut Füllfarbe einer Ellipse geändert werden. Die Füllfarbe soll nach 5 s von rot nach blau und nach weiteren 5 s von blau wieder nach rot wechseln.

Lösung der Aufgabe 1

Das Anwendungsprogramm ruft zyklisch die Funktion **VisuCmdAttribute()** auf.

JetSym-STX-Programm der Aufgabe 1

```
Var
    Flag:    Bool At %MX 1;
End_Var;

Task Visu_STX Autorun
Flag := FALSE;
Loop
    If Flag = FALSE Then
        Flag := TRUE;
    ELSE
        Flag := FALSE;
    End_If;

Case Flag Of
    TRUE:  VisuCmdAttribute (Ellipse_4000,
ELLIPSE_ATTR_FILLATTRIBUTE, FillAttribute_Blue);
    Break;
    FALSE: VisuCmdAttribute (Ellipse_4000,
ELLIPSE_ATTR_FILLATTRIBUTE, FillAttribute_Red);
    Break;
```

```
End_Case;

Delay(T#5s);
End_Loop;
End_Task;
```

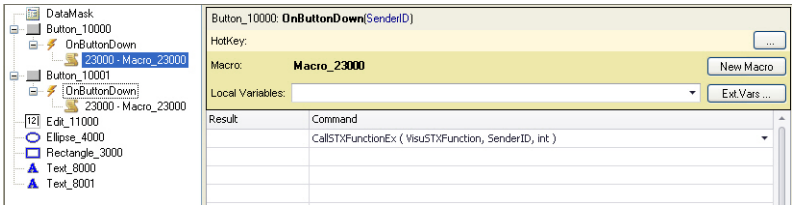
Aufgabe 2

Beim Betätigen der einen Schaltfläche (Button_10000) soll das Attribut Füllfarbe eines Rechteckobjekts rot sein.

Beim Betätigen der zweiten Schaltfläche (Button_10001) soll die Füllfarbe desselben Rechteckobjekts blau sein. Sie betätigen die Schaltfläche über das Digipot.

Lösung der Aufgabe 2

Weisen Sie in JetViewSoft beiden Schaltflächen das Event **OnButtonDown** zu. Wenn das Event eintritt, wird die Makrofunktion **CallSTXFunctionEx()** aufgerufen.



Im Anwendungsprogramm ist eine Funktion deklariert, die wiederum ein Visualisierungskommando ausführt. In diesem Fall ist es die Funktion **VisuSTXFunction()**.

Welche Schaltfläche gerade betätigt ist, wird über die SenderID übermittelt und in einem Case-Befehl abgefragt.

JetSym-STX-Programm der Aufgabe 2

```
Function VisuSTXFunction (SenderID : long)

Case SenderID Of
    Button_10000: VisuCmdAttribute (Rectangle_3000;
RECTANGLE_ATTR_FILLATTRIBUTE, FillAttribute_Red);
    Break;
    Button_10001: VisuCmdAttribute (Rectangle_3000;
RECTANGLE_ATTR_FILLATTRIBUTE, FillAttribute_Blue);
    Break;
End_Case;
End_Function;

Task Visu_STX Autorun

End_Task;
```

Hinweis

Sobald Sie in einem Makroobjekt die Makrofunktion **CallSTXFunctionEx()** ausgewählt haben, ist diese Funktion mit ihrem Namen in der Datei **VisualInterface.stxp** als **forward** deklariert.

6 CANopen®-STX-API

Einleitung

Dieses Kapitel beschreibt die STX-Funktionen der CANopen®-STX-API.

Der CANopen®-Standard

CANopen® ist ein offener Standard für die Vernetzung und Kommunikation z. B. im Kraftfahrzeugbereich.

Das CANopen®-Protokoll wird von der CiA e.V. (CAN in Automation) weiterentwickelt und arbeitet auf dem Physical Layer mit CAN-Highspeed nach ISO 11898.

Dokumente

Die CANopen®-Spezifikationen können von der Homepage des **CiA e.V. http://www.can-cia.org** bezogen werden. Die wichtigsten Spezifikationsdokumente sind dabei:

- CiA DS 301 - Dieses Dokument ist auch als Kommunikationsprofil bekannt und beschreibt die grundlegenden Dienste und Protokolle, die unter CANopen® verwendet werden.
- CiA DS 302 - Framework für programmierbare Geräte (CANopen®-Manager, SDO-Manager)
- CiA DR 303 - Informationen zu Kabeln und Steckverbindern
- CiA DS 4xx - Diese Dokumente beschreiben das Verhalten vieler Geräteklassen über sogenannte Geräteprofile.

Anwendung

Diese STX-Funktionen werden in der Kommunikation zwischen dem Gerät JVM-104 und anderen CANopen®-Teilnehmern angewendet.

Hinweis: Perspektive beachten!

Dieses Kapitel ist aus Sicht der übergeordneten Steuerung beschrieben, während im Dokument CiA DS 301 aus Gerätesicht dokumentiert ist.

Deshalb benötigen Sie z. B. zum Aufrufen der Funktion `CanOpenAddPDORx()` ein PDO_RX-Makro.

Begriffe und Abkürzungen

Folgende Begriffe und Abkürzungen kommen in diesem Kapitel vor:

Begriff	Beschreibung
Node-ID	Knotennummer des Teilnehmers: Über diese ID wird das Gerät angesprochen.
NMT	Network management - Netzwerkmanagement
ro	Read Only - Nur Lesezugriff
rw	Read/Write - Lese und Schreibzugriff

Inhalt

Thema	Seite
STX-Funktion CanOpenInit()	87
STX-Funktion CanOpenSetCommand()	89
STX-Funktion CanOpenUploadSDO()	91
STX-Funktion CanOpenDownloadSDO()	96
STX-Funktion CanOpenAddPDORx()	101
STX-Funktion CanOpenAddPDOTx()	108
Heartbeat-Überwachung	114
CANopen®-Objektverzeichnis des JVM-104	118

STX-Funktion CanOpenInit()

Einleitung

Mit dem Aufruf der Funktion `CanOpenInit()` wird einer der vorhandenen CAN-Busse initialisiert. Die JVM-104 sendet dann automatisch jede Sekunde die Heartbeat-Nachricht mit dem folgenden Kommunikationsobjekt-Identifier (COB-ID): Node-ID + 0x700.

Funktionsdeklaration

```
Function CanOpenInit(
    CANNo:Int,
    NodeID:Int,
    const ref SWVersion:String,
) :Int;
```

Funktionsparameter

Die Funktion `CanOpenInit()` hat die folgenden Parameter.

Parameter	Beschreibung	Wert
CANNo	CAN-Nummer der Busleitung	0 ... CANMAX
NodeID	Eigene Node-ID	1 ... 127
SWVersion	Referenz auf die eigene Softwareversion Diese Softwareversion wird unter dem Index 0x100A in das Objektverzeichnis eingetragen.	String bis zu 255 Zeichen

Rückgabewert

Die Funktion übergibt die folgenden Rückgabewerte an das übergeordnete Programm.

Rückgabewert	
0	ok
-1	Fehler bei der Parameterüberprüfung
-3	Die Initialisierung hat nicht funktioniert
-4	Der JX2-Systembus-Treiber ist aktiviert

CANNo-Parameter

Dieser Parameter gibt die Nummer der CAN-Schnittstelle an. Die erste Schnittstelle wird mit CANNo = 0 ausgewählt. Die Anzahl der CAN-Schnittstellen ist geräteabhängig. Die mögliche Anzahl der CAN-Schnittstellen CANMAX ist in den *technischen Daten* und der *Kurzreferenz* der jeweiligen Betriebsanleitung erwähnt.

Verwenden der Funktion

Initialisieren des CAN-Busses 0. Die JVM-104 hat die Node-ID 20 (0x14).

```
Result := CanOpenInit(0, 20, 'Version: 01.00.0.00');
```

Funktionsweise

Während der Initialisierung durchläuft die JVM-104 folgende Prozessstufen:

Stufe	Beschreibung
1	Zuerst wird die Bootup-Meldung als Heartbeat-Nachricht gesendet.
2	Sobald die JVM-104 in den Zustand **Pre-Operational** geht, sendet es die Heartbeat-Nachricht **Pre-Operational**.

Zugriff auf das Objekt-verzeichnis

Wenn das Gerät JVM-104 sich im Zustand **Pre-Operational** befindet, dann können Sie über SDO auf das Objektverzeichnis zugreifen.

NMT-Nachrichten

Nach der Initialisierung können NMT-Nachrichten gesendet und empfangen werden. Der eigene Heartbeat-Zustand kann mit der Funktion `CanOpenSetCommand()` geändert werden.

Verwandte Themen

- **STX-Funktion CanOpenSetCommand** (siehe Seite 89)

STX-Funktion CanOpenSetCommand()

Einleitung

Mit dem Aufruf der Funktion `CanOpenSetCommand()` kann der eigene Heartbeat-Zustand und der Heartbeat-Zustand aller anderen Geräte (NMT-Slaves) am CAN-Bus geändert werden.

Funktionsdeklaration

```
Function CanOpenSetCommand(
    CANNo:Int,
    iType:Int,
    Value:Int,
) :Int;
```

Funktionsparameter

Die Funktion `CanOpenSetCommand()` hat die folgenden Parameter.

Parameter	Beschreibung	Wert
CANNo	CAN-Nummer der Busleitung	0 ... CANMAX
iType	Auswahl des Kommandos	Siehe nächste Tabelle.

iType	Beschreibung: Value
CAN_CMD_HEARTBEAT	Nur der eigene Heartbeat-Zustand wird geändert. Auswahl der Heartbeat-Zustände: CAN_HEARTBEAT_STOPPED (0x04) CAN_HEARTBEAT_OPERATIONAL (0x05) CAN_HEARTBEAT_PREOPERATIONAL (0x7F)
CAN_CMD_NMT	Bei allen Geräten oder bei einem speziellen Gerät am CAN-Bus wird der Heartbeat-Zustand geändert. Auswahl der Heartbeat-Zustände (NMT-Master): CAN_NMT_OPERATIONAL (0x01) oder CAN_NMT_START (0x01) CAN_NMT_STOP (0x02) CAN_NMT_PREOPERATIONAL (0x80) CAN_NMT_RESET (0x81) CAN_NMT_RESETCOMMUNICATION (0x82)
CAN_CMD_TIME_CONSUMER	Dieses Kommando macht das Gerät empfangsbereit für die Synchronisierung der Uhrzeit über den CAN-Bus (CAN-ID 0x100). Siehe Dokument der CiA.e.V DS301 V402 Seite 59. *Auswahl der Synchronisierung*: CAN_TIME_CONSUMER_DISABLE = 0 (Abschalten der Synchronisierung) CAN_TIME_CONSUMER_ENABLE = 1 (Einschalten der Synchronisierung)
CAN_CMD_TIME_PRODUCER	Die Uhrzeit wird auf dem CAN-Bus veröffentlicht. Struktur siehe Dokument der CiA.e.V DS301 CAN-ID 0x100: CAN_TIME_PRODUCER_SEND = 1 (Sendet bei Aufruf einmalig TIME_OF_DAY)

Hinweis

Die Auswahl des Kommandos CAN_CMD_NMT erfolgt über die Makrofunktion `CAN_CMD_NMT_Value(NodeID, CAN_CMD_NMT)`.

Für den Parameter Node-ID sind Werte von 0 bis 127 zulässig. 1 bis 127 ist die Node-ID für ein bestimmtes Gerät. Soll das Kommando an alle Geräte am CAN-Bus gesendet werden, wird der Parameter `CAN_CMD_NMT_ALLNODES(0)` verwendet.

CANNo-Parameter

Dieser Parameter gibt die Nummer der CAN-Schnittstelle an. Die erste Schnittstelle wird mit CANNo = 0 ausgewählt. Die Anzahl der CAN-Schnittstellen ist geräteabhängig. Die mögliche Anzahl der CAN-Schnittstellen CANMAX ist in den *technischen Daten* und der *Kurzreferenz* der jeweiligen Betriebsanleitung erwähnt.

Rückgabewert

Die Funktion übergibt die folgenden Rückgabewerte an das übergeordnete Programm.

Rückgabewert	
0	ok
-1	Fehler bei der Parameterüberprüfung Kommando nicht bekannt

Verwenden der Funktion (Beispiel 1)

Der eigene Heartbeat-Zustand soll auf **Operational** gesetzt werden.

```
Result := CanOpenSetCommand(0, CAN_CMD_HEARTBEAT,
CAN_HEARTBEAT_OPERATIONAL);
```

Verwenden der Funktion (Beispiel 2)

Der eigene Heartbeat-Zustand und der Zustand von allen anderen Geräten am CAN-Bus soll auf **Operational** gesetzt werden.

```
Result := CanOpenSetCommand(0, CAN_CMD_NMT,
CAN_CMD_NMT_Value(CAN_CMD_NMT_ALLNODES, CAN_NMT_OPERATIONAL));
```

Verwenden der Funktion (Beispiel 3)

Der Heartbeat-Zustand von dem Gerät mit der Node-ID 60 (0x3C) soll auf **Operational** gesetzt werden.

```
Result := CanOpenSetCommand(0, CAN_CMD_NMT, CAN_CMD_NMT_Value(60,
CAN_NMT_OPERATIONAL));
```

Verwenden der Funktion (Beispiel 4)

Die Synchronisierung der Uhrzeit über den CAN-Bus (CAN-ID 0x100) soll eingeschaltet werden.

```
Result := CanOpenSetCommand(0, CAN_CMD_TIME_CONSUMER,
CAN_TIME_CONSUMER_ENABLE);
```

Verwenden der Funktion (Beispiel 5)

Die Uhrzeit soll auf dem CAN-Bus veröffentlicht werden.

```
Result := CanOpenSetCommand(0, CAN_CMD_TIME_PRODUCER,
CAN_TIME_PRODUCER_SEND);
```

STX-Funktion CanOpenUploadSDO()

Einleitung

Mit dem Aufruf der Funktion `CanOpenUploadSDO()` wird gezielt auf ein bestimmtes Objekt im Objektverzeichnis des Nachrichtenempfängers zugegriffen und der Wert des Objekts ausgelesen.

Der Datenaustausch erfolgt entsprechend dem SDO-Upload-Protokoll. Als Transfertyp wird **segmented** (mehr als 4 Datenbytes) und **expedited** (bis 4 Datenbytes) unterstützt.

Funktionsdeklaration

```
Function CanOpenUploadSDO(
    CANNo:Int,              // Nummer der Busleitung
    NodeID:Int,             // Geräte-ID
    wIndex:Word,
    SubIndex:Byte,
    DataType:Int,           // Typ der zu empfangenden Daten
    // Datengröße der globalen Variablen DataAddr
    DataLength:Int,
    // Globale Variable, in der der empfangene Wert steht
    const ref DataAddr,
    ref Busy: Int,          // Zustand der SDO-Übertragung
) :Int;
```

Funktionsparameter

Die Funktion `CanOpenUploadSDO()` hat die folgenden Parameter.

Parameter	Beschreibung	Wert
CANNo	CAN-Nummer der Busleitung	0 ... CANMAX
NodeID	Node-ID des Nachrichten-empfängers	1 ... 127
wIndex	Index-Nummer des Objekts	0 ... 0xFFFF
SubIndex	Subindex-Nummer des Objekts	0 ... 255
DataType	Typ der zu empfangenden Daten	2 ... 27
DataLength	Datengröße der globalen Variablen DataAddr	
DataAddr	Globale Variable, in die der empfangene Wert eingetragen werden soll	
Busy	Zustand der SDO-Übertragung	

Rückgabewert

Die Funktion übergibt die folgenden Rückgabewerte an das übergeordnete Programm.

Rückgabewert	
0	Ok
-1	Fehler bei der Parameterprüfung
-2	Gerät im Stoppzustand
-3	DataType ist größer als DataLength
-4	Nicht genug Speicher vorhanden

CANNo-Parameter

Dieser Parameter gibt die Nummer der CAN-Schnittstelle an. Die erste Schnittstelle wird mit CANNo = 0 ausgewählt. Die Anzahl der CAN-Schnittstellen ist geräteabhängig. Die mögliche Anzahl der CAN-Schnittstellen CANMAX ist in den *technischen Daten* und der *Kurzreferenz* der jeweiligen Betriebsanleitung erwähnt.

DataType-Parameter

Folgende Datentypen können empfangen werden.

Byte-Typen	CANopen®-Format	Jetter-Format
1	CANOPEN_INTEGER8 CANOPEN_UNSIGNED8	Byte
2	CANOPEN_INTEGER16 CANOPEN_UNSIGNED16	Word
3	CANOPEN_INTEGER24 CANOPEN_UNSIGNED24	-
4	CANOPEN_INTEGER32 CANOPEN_UNSIGNED32 CANOPEN_REAL	Int
5	CANOPEN_INTEGER40 CANOPEN_UNSIGNED40	-
6	CANOPEN_INTEGER48 CANOPEN_UNSIGNED48 CANOPEN_TIME_OF_DAY CANOPEN_TIME_DIFFERENCE	-
7	CANOPEN_INTEGER56 CANOPEN_UNSIGNED46	-
8	CANOPEN_INTEGER64 CANOPEN_UNSIGNED64 CANOPEN_REAL64	-
n	CANOPEN_VISIBLE_STRING CANOPEN_OCTET_STRING CANOPEN_UNICODE_STRING CANOPEN_DOMAIN	String

Busy-Parameter

Nach erfolgreichem Aufruf der Funktion wird der Parameter **Busy** auf SDOACCESS_INUSE gesetzt. Bei fehlerhafter Übertragung wird **Busy** auf SDOACCESS_ERROR gesetzt. Bei erfolgreicher Übertragung liefert die Funktion die Anzahl der übertragenen Bytes zurück.

Busy - Fehlercodes

Bei fehlerhafter Übertragung liefert **Busy** einen Fehlercode zurück. Die folgenden Fehlercodes gibt es:

SDOACCESS_STILLUSED

Ein anderer Task kommuniziert mit derselben Node-ID.

SDOACCESS_TIMEOUT

Es ist ein Timeout erfolgt, weil das Gerät mit der angegebenen Node-ID keine Antwort gibt.

Antwortet das Gerät nicht innerhalb 1 Sekunde wird der Timeout gesetzt.

SDOACCESS_ILLCMD

Die Antwort auf die Anfrage ist ungültig.

SDOACCESS_ABORT

Ein Abbruch des Geräts mit der Node-ID ist erfolgt.

SDOACCESS_SYSERROR

Allgemeiner interner Fehler

Makrodefinitionen

Folgende Makros wurden im Zusammenhang mit dieser Funktion definiert:

SDOACCESS_FINISHED (busy)

Dieses Makro prüft, ob die Kommunikation beendet wurde.

SDOACCESS_ERROR (busy)

Dieses Makro prüft, ob ein Fehler aufgetreten ist.

Verwenden der Funktion

```
Result := CanOpenUploadSDO(
    0,                              // CANNo
    66,                             // NodeID
    0x100A,                         // wIndex
    0,                              // SubIndex
    CANOPEN_STRING,                 // DataType
    sizeof(var_Versionstring),      // DataLength
    var_Versionstring,              // DataAddr
    busy);                          // Busy
```

JetSym-STX-Programm

Im folgenden Beispiel wird die Softwareversion des Herstellers aus dem CANopen®-Objektverzeichnis vom Gerät mit der adressierten Node-ID ausgelesen.

```
Const
    CANNo = 0;              // Nummer der Busleitung
    NodeID_Node_0 = 10;     // Geräte-ID von Node 1
    NodeID_Node_1 = 66;     // Geräte-ID von Node 2
End_Const;

Var
    busy: Int;
    Versionstring: String;
    Objektindex: Word;
    Subindex: Byte;
    Result: Int;
End_Var;


Task Example_UploadSDO autorun

Var
    SW_Version: String;
End_Var;

SW_Version := 'v4.3.0.2004';

// Initialisierung CAN 0
CanOpenInit(CANNo,              // Nummer der Busleitung
            NodeID_Node_0,      // Geräte-ID
            SW_Version);        // Softwareversion des Herstellers

// Alle Geräte am CAN-Bus sind im Status PREOPERATIONAL

// Softwareversion des Herstellers per SDO anfordern
Objektindex := 0x100A;
Subindex := 0;
```

```
Result:= CanOpenUploadSDO(CANNo,         // Nummer der Busleitung
                 NodeID_Node_1,          // Geräte-ID
                 Objektindex,            // wIndex
                 Subindex,               // SubIndex
                 CANOPEN_STRING,         // DataType
                 sizeof(Versionstring), // DataLength
                 Versionstring,          // DataAddr
                 busy);                  // Busy

// Prüfen ob Befehl erfolgreich ausgeführt wurde.
If (Result == 0) Then

    // Warten bis die Kommunikation beendet ist.
    When SDOACCESS_FINISHED(busy) Continue;

    // Prüfen, ob ein Fehler aufgetreten ist.
    If (SDOACCESS_ERROR(busy)) Then
        // Fehlerbehandlung
    End_If;
End_If;

//      ...
//      ...
//      ...

End_Task;
```

STX-Funktion CanOpenDownloadSDO()

Einleitung

Mit dem Aufruf der Funktion `CanOpenDownloadSDO()` wird gezielt auf ein bestimmtes Objekt im Objektverzeichnis des Nachrichtenempfängers zugegriffen und der Wert des Objekts beschrieben. Der Datenaustausch erfolgt entsprechend dem SDO-Downloadprotokoll. Als Transfertyp wird **segmented** oder **block** (mehr als 4 Datenbytes) und **expedited** (bis 4 Datenbytes) unterstützt.

Funktionsdeklaration

```
Function CanOpenDownloadSDO(
    CANNo:Int,              // Nummer der Busleitung
    NodeID:Int,             // Geräte-ID
    wIndex:Word,
    SubIndex:Byte,
    DataType:Int,           // Typ der zusendenden Daten
    // Datengröße der globalen Variablen DataAddr
    DataLength:Int,
    // Globale Variable, in der der zu sendende Wert steht
    const ref DataAddr,
    ref Busy: Int,          // Zustand der SDO-Übertragung
) :Int;
```

Funktionsparameter

Die Funktion `CanOpenDownloadSDO()` hat die folgenden Parameter.

Parameter	Beschreibung	Wert
CANNo	CAN-Nummer der Busleitung	0 ... CANMAX
NodeID	Node-ID des Nachrichtenempfängers	1 ... 127
wIndex	Index-Nummer des Objekts	0 ... 0xFFFF
SubIndex	Subindex-Nummer des Objekts	0 ... 255
DataType	Typ der zu sendenden Daten	2 ... 27
DataLength	Datengröße der globalen Variablen DataAddr	
DataAddr	Globale Variable, in die der zu sendende Wert eingetragen werden soll	
Busy	Zustand der SDO-Übertragung	

Rückgabewert

Die Funktion übergibt die folgenden Rückgabewerte an das übergeordnete Programm.

Rückgabewert	
0	Ok
-1	Fehler bei der Parameterüberprüfung
-2	Gerät im Zustand **Stopp** (eigener Heartbeat-Zustand)
-3	DataType ist größer als DataLength
-4	Nicht genug Speicher vorhanden

CANNo-Parameter

Dieser Parameter gibt die Nummer der CAN-Schnittstelle an. Die erste Schnittstelle wird mit CANNo = 0 ausgewählt. Die Anzahl der CAN-Schnittstellen ist geräteabhängig. Die mögliche Anzahl der CAN-Schnittstellen CANMAX ist in den *technischen Daten* und der *Kurzreferenz* der jeweiligen Betriebsanleitung erwähnt.

DataType-Parameter

Folgende Datentypen können empfangen werden.

Byte-Typen	CANopen®-Format	Jetter-Format
1	CANOPEN_INTEGER8 CANOPEN_UNSIGNED8	Byte
2	CANOPEN_INTEGER16 CANOPEN_UNSIGNED16	Word
3	CANOPEN_INTEGER24 CANOPEN_UNSIGNED24	-
4	CANOPEN_INTEGER32 CANOPEN_UNSIGNED32 CANOPEN_REAL	Int
5	CANOPEN_INTEGER40 CANOPEN_UNSIGNED40	-
6	CANOPEN_INTEGER48 CANOPEN_UNSIGNED48 CANOPEN_TIME_OF_DAY CANOPEN_TIME_DIFFERENCE	-
7	CANOPEN_INTEGER56 CANOPEN_UNSIGNED46	-
8	CANOPEN_INTEGER64 CANOPEN_UNSIGNED64 CANOPEN_REAL64	-
n	CANOPEN_VISIBLE_STRING CANOPEN_OCTET_STRING CANOPEN_UNICODE_STRING CANOPEN_DOMAIN	String

Busy-Parameter

Nach erfolgreichem Aufruf der Funktion wird der Parameter **Busy** auf SDOACCESS_INUSE gesetzt. Bei fehlerhafter Übertragung wird **Busy** auf SDOACCESS_ERROR gesetzt. Bei erfolgreicher Übertragung liefert die Funktion die Anzahl der übertragenen Bytes zurück.

Busy-Fehlercodes

Bei fehlerhafter Übertragung liefert Busy einen Fehlercode zurück. Die folgenden Fehlercodes gibt es:

SDOACCESS_STILLUSED

Ein anderer Task kommuniziert mit derselben Node-ID.

SDOACCESS_TIMEOUT

Es ist ein Timeout erfolgt, weil das Gerät mit der Node-ID keine Antwort gibt.

Antwortet die angegebene Node-ID nicht innerhalb 1 Sekunde, wird der Timeout gesetzt.

SDOACCESS_ILLCMD

Die Antwort auf die Anfrage ist ungültig.

SDOACCESS_ABORT

Ein Abbruch des Geräts mit der Node-ID ist erfolgt.

SDOACCESS_BLKSIZEINV

Kommunikationsfehler bei Block Download

SDOACCESS_SYSERROR

Allgemeiner interner Fehler

Makrodefinitionen

Folgende Makros wurden im Zusammenhang mit dieser Funktion definiert:

SDOACCESS_FINISHED (busy)

Dieses Makro prüft, ob die Kommunikation beendet wurde.

SDOACCESS_ERROR (busy)

Dieses Makro prüft, ob ein Fehler aufgetreten ist.

Verwenden der Funktion

```
Result := CanOpenDownloadSDO(
    0,                              // CANNo
    68,                             // NodeID
    0x1017,                         // wIndex
    0,                              // SubIndex
    CANOPEN_WORD,                   // DataType
    sizeof(var_Heartbeat_time),     // DataLength
    var_Heartbeat_time,             // DataAddr
    busy);                          // Busy
```

JetSym-STX-Programm

Im folgenden Beispiel wird die Heartbeat-Zeit in das CANopen®-Objektverzeichnis des Geräts mit der adressierten Node-ID eingetragen.

```
Const
    CANNo = 0;              // Nummer der Busleitung
    NodeID_Node_0 = 10;     // Node-ID Node 1
    NodeID_Node_1 = 68;     // Node-ID Node 2
End_Const;

Var
    busy: Int;
    Heartbeat_time: Int;
    Objektindex: Word;
    Subindex: Byte;
    Result: Int;
End_Var;


Task Example_DownloadSDO autorun

Var
    SW_Version: String;
End_Var;

SW_Version := 'v4.3.0.2004';

// Initialisierung CAN 0
CanOpenInit(CANNo,            // Nummer der Busleitung
            NodeID_Node_0,    // Geräte-ID
            SW_Version);      // Softwareversion des Herstellers

// Das Gerät mit der Geräte-ID NodeID_Node_1 am CAN-Bus in den Status
// PREOPERATIONAL setzen.
CanOpenSetCommand(CANNo, CAN_CMD_NMT_Value(NodeID_Node_1,
CAN_CMD_NMT), CAN_NMT_PREOPERATIONAL);

// Heartbeat-Zeit des adressierten Geräts per SDO ändern.
Objektindex := 0x1017;
Subindex := 0;
Result:= CanOpenDownloadSDO(CANNo,  // Nummer der Busleitung
                  NodeID_Node_1,          // Geräte-ID
                  Objektindex,            // wIndex
                  Subindex,               // SubIndex
                  CANOPEN_WORD,           // DataType
                  sizeof(Heartbeat_time), // DataLength
                  Heartbeat_time,         // DataAddr
                  busy);                  // Busy

// Prüfen ob Befehl erfolgreich ausgeführt wurde.
If (Result == 0) Then
```

```
        // Warten bis die Kommunikation beendet ist.
        When SDOACCESS_FINISHED(busy) Continue;

        // Prüfen, ob ein Fehler aufgetreten ist.
        If (SDOACCESS_ERROR(busy)) Then
        // Fehlerbehandlung
        End_If;
    End_If;

    // Alle Geräte am CAN-Bus wieder in den Status OPERATIONAL setzen.
    CanOpenSetCommand(CANNo, CAN_CMD_NMT_Value(CAN_CMD_NMT_ALLNODES,
    CAN_CMD_NMT), CAN_NMT_OPERATIONAL);

    //      ...
    //      ...
    //      ...

    End_Task;
```

STX-Funktion CanOpenAddPDORx()

Einleitung

Mit dem Aufruf der Funktion `CanOpenAddPDORx()` können Prozessdaten, die andere CANopen®-Geräte senden, zum Empfang eingetragen werden.

Wenn ein CANopen®-Gerät Prozessdaten sendet, nur dann werden diese Prozessdaten empfangen.

Hinweise

- Wenn die CANopen®-Geräte am Bus sich im Zustand **Operational** befinden, erst dann wird das PDO-Telegramm übertragen.
- Die kleinste Zeiteinheit der Event-Time ist 1 ms.
- Die kleinste Zeiteinheit der Inhibit-Time ist 1 ms.

Funktionsdeklaration

```
Function CanOpenAddPDORx(
    CANNo:Int,              // Nummer der Busleitung
    CANID:Int,              // CAN-Identifier
    // Startposition der zu empfangenden Daten
    BytePos:Int,
    DataType:Int,           // Datentyp der zu empfangenden Daten
    // Datengröße der globalen Variablen VarAddr
    DataLength:Int,
    // Globale Variable, in der der empfangene Wert steht
    const ref VarAddr,
    // Zykluszeit, in der ein Telegramm empfangen werden soll
    // Event-Zeit
    EventTime: Int,
    // Mindesabstand zwischen zwei empfangenen Telegrammen
    // Inhibit-Zeit
    InhibitTime: Int,
    Paramset: Int,          // Bitkodierter Parameter
) :Int;
```

Funktionsparameter

Die Funktion `CanOpenAddPDORx()` hat die folgenden Parameter.

Parameter	Beschreibung	Wert
CANNo	CAN-Nummer der Busleitung	0 ... CANMAX
CANID	CAN-Identifier 11 Bit CAN-Identifier 29 Bit	0 ... 0x7FF 0 ... 0x1FFFFFFF
BytePos	Startposition der zu empfangenden Daten	0 ... 7
DataType	Datentyp der zu empfangenden Daten	2 ... 13, 15 ... 27
DataLength	Datengröße der globalen Variablen VarAddr	
VarAddr	Globale Variable, in die der empfangene Wert eingetragen wird	
EventTime	Zeitlicher Abstand zwischen zwei Telegrammen (> InhibitTime)	

Parameter	Beschreibung	Wert
InhibitTime	Mindestabstand zwischen zwei empfangenen Telegrammen (< EventTime)	
Paramset	Bitkodierter Parameter	

Rückgabewert

Die Funktion übergibt die folgenden Rückgabewerte an das übergeordnete Programm.

Rückgabewert	
0	Ok
-1	Fehler bei der Parameterüberprüfung
-3	DataType ist größer als DataLength
-4	Nicht genug Speicher vorhanden

CANNo-Parameter

Dieser Parameter gibt die Nummer der CAN-Schnittstelle an. Die erste Schnittstelle wird mit CANNo = 0 ausgewählt. Die Anzahl der CAN-Schnittstellen ist geräteabhängig. Die mögliche Anzahl der CAN-Schnittstellen CANMAX ist in den *technischen Daten* und der *Kurzreferenz* der jeweiligen Betriebsanleitung erwähnt.

Hinweis: Perspektive beachten!

Dieser Abschnitt ist aus Sicht der übergeordneten Steuerung beschrieben, während die Beschreibung im Dokument CiA DS 301 von der Geräteseite ausgeht.

Deshalb benötigen Sie zum Aufrufen der Funktion `CanOpenAddPDORx()` von der Steuerung aus ein PDO_RX-Makro.

CANID-Parameter

Mit dem Parameter **CANID** wird der CAN-Identifier übergeben. Der CAN-Identifier wird mit einem Makro erstellt. Der CAN-Identifier ist abhängig von der Node-ID des anderen Kommunikationsteilnehmers und abhängig davon, ob es sich um eine PDO1-, PDO2-, PDO3- oder PDO4-Nachricht handelt.

Makrodefinitionen:

```
#Define CANOPEN_PDO1_RX (NodeID)    ((NodeID) + 0x180)
#Define CANOPEN_PDO2_RX (NodeID)    ((NodeID) + 0x280)
#Define CANOPEN_PDO3_RX (NodeID)    ((NodeID) + 0x380)
#Define CANOPEN_PDO4_RX (NodeID)    ((NodeID) + 0x480)

#Define CANOPEN_PDO1_TX (NodeID)    ((NodeID) + 0x200)
#Define CANOPEN_PDO2_TX (NodeID)    ((NodeID) + 0x300)
#Define CANOPEN_PDO3_TX (NodeID)    ((NodeID) + 0x400)
#Define CANOPEN_PDO4_TX (NodeID)    ((NodeID) + 0x500)
```

Beispiel für den Aufruf des Makros:

CANOPEN_PDO2_RX (64)

⇨ Der daraus resultierende CAN-Identifier ist: 2C0h = 40h + 280h

Default-CAN-Identifier-Verteilung

Bei CANopen® ist die folgende CAN-Identifier-Verteilung vordefiniert. Dabei wird die Knotennummer in den Identifier eingebettet.

Identifier 11-Bit (binär)	Identifier (dezimal)	Identifier (hexadezimal)	Funktion
000000000000	0	0	Netzwerkmanagement
000100000000	128	80h	Synchronisation
0001xxxxxxxx	129 - 255	81h - FFh	Emergency
0011xxxxxxxx	385 - 511	181h - 1FFh	PDO1 (tx)
0100xxxxxxxx	513 - 639	201h - 27Fh	PDO1 (rx)
0101xxxxxxxx	641 - 767	281h - 2FFh	PDO2 (tx)
0110xxxxxxxx	769 - 895	301h - 37Fh	PDO2 (rx)
0111xxxxxxxx	897 - 1023	381h - 3FFh	PDO3 (tx)
1000xxxxxxxx	1025 - 1151	401h -47Fh	PDO3 (rx)
1001xxxxxxxx	1153 - 1279	481h - 4FFh	PDO4 (tx)
1010xxxxxxxx	1281 - 1407	501h - 57Fh	PDO4 (rx)
1011xxxxxxxx	1409 - 1535	581h - 5FFh	SDO senden
1100xxxxxxxx	1537 - 1663	601h - 67Fh	SDO empfangen
1110xxxxxxxx	1793 - 1919	701h - 77Fh	NMT Error Control
xxxxxxxx = Knotennummer 1 - 127			

DataType-Parameter

Folgende Datentypen können empfangen werden.

Byte-Typen	CANopen®-Format	Jetter-Format
1	CANOPEN_INTEGER8 CANOPEN_UNSIGNED8	Byte
2	CANOPEN_INTEGER16 CANOPEN_UNSIGNED16	Word
3	CANOPEN_INTEGER24 CANOPEN_UNSIGNED24	-
4	CANOPEN_INTEGER32 CANOPEN_UNSIGNED32 CANOPEN_REAL	Int
5	CANOPEN_INTEGER40 CANOPEN_UNSIGNED40	-
6	CANOPEN_INTEGER48 CANOPEN_UNSIGNED48 CANOPEN_TIME_OF_DAY CANOPEN_TIME_DIFFERENCE	-
7	CANOPEN_INTEGER56 CANOPEN_UNSIGNED46	-
8	CANOPEN_INTEGER64 CANOPEN_UNSIGNED64 CANOPEN_REAL64	-
n	CANOPEN_VISIBLE_STRING CANOPEN_OCTET_STRING CANOPEN_UNICODE_STRING CANOPEN_DOMAIN	String

Paramset-Parameter

Folgende Parameter können an die Funktion übergeben werden. Mehrere Parameter können miteinander über die Oder-Funktion verknüpft werden.

CANOPEN_ASYNCPDORTRONLY

Empfange asynchrone PDOs durch das Senden eines RTR-Frames (nach jeder abgelaufener EventTime) an den Sender. Wenn auf die RTR-Frames nicht geantwortet wird, dann erhöht sich die Anfragezeit auf das Fünffache der EventTime.

CANOPEN_ASYNCPDO

Empfange asynchrone PDO.

CANOPEN_PDOINVALID

PDO wird nicht empfangen. Speicherplatz wird reserviert.

CANOPEN_NORTR

PDO kann nicht per RTR (Remote Request) angefordert werden.

Nur wenn ein CANOPEN_ASYNCPDORTROnly gesetzt ist, wird ein RTR gesendet.

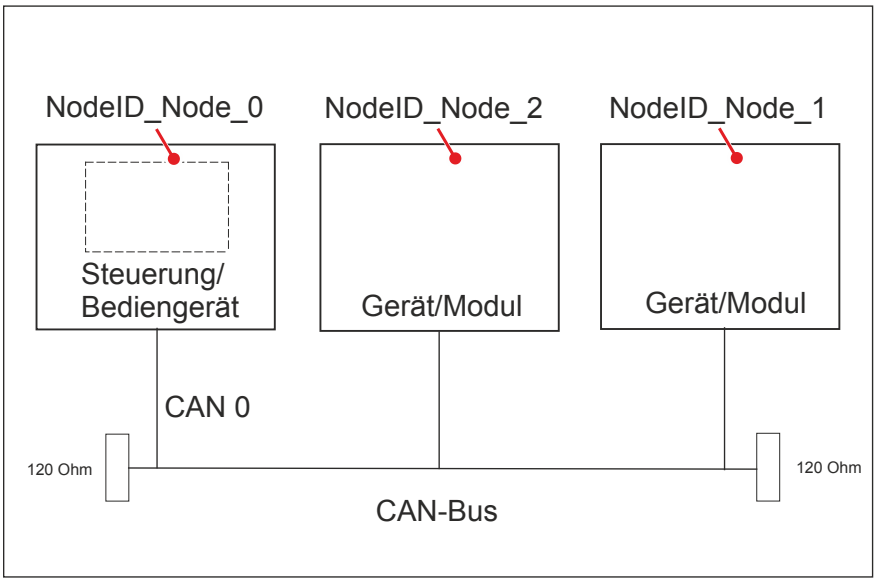
CANOPEN_29BIT
Verwende 29 Bit-Identifier
Default: 11 Bit-Identifier

Verwenden der Funktion

```
Result := CanOpenAddPDORx(
    0,                                      // CANNo
    662,                                    // CANID
    0,                                      // BytePos
    CANOPEN_DWORD,                          // DataType
    sizeof(var_Data_1_of_Node_1),           // DataLength
    var_Data_1_of_Node_1,                   // VarAddr
    1000,                                   // Event-Zeit
    10,                                     // Inhibit-Zeit
    CANOPEN_ASYNCPDO | CANOPEN_NORTR);      // Paramset
```

JetSym-STX-Programm

JVM-104 mit der Node-ID 10 möchte von zwei CANopen®-Geräten mit der Node-ID 64 und 102 ein PDO empfangen. Dazu wird die Funktion `CanOpenAddPDORx()` aufgerufen. Nach Ablauf des Programms empfängt das Gerät JVM-104 die zyklischen PDO-Telegramme.




```
Const
    CANNo = 0;              // Nummer der Busleitung
    NodeID_Node_0 = 10;     // Geräte-ID von Node 1
    NodeID_Node_1 = 64;     // Geräte-ID von Node 2
    NodeID_Node_2 = 102;    // Geräte-ID von Node 3
    Event_Time = 1000;      // Event-Zeit in ms
    Inhibit_Time = 10;      // Inhibit-Zeit in ms
End_Const;
```

```
Var
    Data_1_of_Node_1: Int;
    Data_2_of_Node_1: Int;
    Data_1_of_Node_2: Int;
End_Var;


Task main autorun

Var
    SW_Version: String;
End_Var;

SW_Version := 'v4.3.0.2004';

// Initialisierung CAN 0
CanOpenInit(CANNo,            // Nummer der Busleitung
            NodeID_Node_0,    // Geräte-ID
            SW_Version);      // Softwareversion des Herstellers

// Prozessdaten zum Empfang eintragen
CanOpenAddPDORx(
    CANNo,                              // Nummer der Busleitung
    CANOPEN_PDO2_RX(NodeID_Node_1),     // CANID
    0,                                  // BytePos
    CANOPEN_DWORD,                      // DataType
    sizeof(Data_1_of_Node_1),           // DataLength
    Data_1_of_Node_1,                   // VarAddr
    Event_Time,                         // Event-Zeit
    Inhibit_Time,                       // Inhibit-Zeit
    CANOPEN_ASYNCPDORTRONLY);           // Paramset

CanOpenAddPDORx(
    CANNo,                              // Nummer der Busleitung
    CANOPEN_PDO2_RX(NodeID_Node_1),     // CANID
    4,                                  // BytePos
    CANOPEN_DWORD,                      // DataType
    sizeof(Data_2_of_Node_1),           // DataLength
    Data_2_of_Node_1,                   // VarAddr
    Event_Time,                         // Event-Zeit
    Inhibit_Time,                       // Inhibit-Zeit
    CANOPEN_ASYNCPDORTRONLY);           // Paramset
```

```
CanOpenAddPDORx(
    CANNo,                                  // Nummer der Busleitung
    CANOPEN_PDO3_RX(NodeID_Node_2),         // CANID
    0,                                      // BytePos
    CANOPEN_BYTE,                           // DataType
    sizeof(Data_1_of_Node_2),               // DataLength
    Data_1_of_Node_2,                       // VarAddr
    Event_Time,                             // Event-Zeit
    Inhibit_Time,                           // Inhibit-Zeit
    CANOPEN_ASYNCPDO | CANOPEN_NORTR); // Paramset

// Alle Geräte am CAN-Bus sind im Status PREOPERATIONAL.

// Alle Geräte am CAN-Bus in den Status OPERATIONAL setzen.
CanOpenSetCommand(CANNo, CAN_CMD_NMT_Value(CAN_CMD_NMT_ALLNODES,
CAN_CMD_NMT), CAN_NMT_START);

// Ab jetzt werden PDO-Telegramme von der übergeordneten Steuerung
// angefordert oder empfangen.
//      ...
//      ...
//      ...

End_Task;
```

STX-Funktion CanOpenAddPDOTx()

Einleitung

Mit dem Aufruf der Funktion `CanOpenAddPDOTx()` können Prozessdaten auf den Bus gelegt werden.

Das muss aber nicht heißen, dass andere CANopen®-Geräte am Bus diese Prozessdaten auch lesen.

Hinweise

- Wenn die CANopen®-Geräte am Bus sich im Zustand **Operational** befinden, erst dann wird das PDO-Telegramm übertragen.
- Sobald sich die Prozessdaten ändern, wird sofort wieder ein PDO-Telegramm übertragen.
- Die kleinste Zeiteinheit der Event-Zeit ist 1 ms.
- Die kleinste Zeiteinheit der Inhibit-Zeit ist 1 ms.
- Alle nicht belegten Bytes eines Telegramms werden mit Null gesendet.

Funktionsdeklaration

```
Function CanOpenAddPDOTx(
    CANNo:Int,           // Nummer der Busleitung
    CANID:Int,           // CAN-Identifier
    BytePos:Int,         // Startposition der zu sendenden Daten
    DataType:Int,        // Datentyp der zu sendenden Daten
    // Datengröße der globalen Variablen VarAddr
    DataLength:Int,
    // Globale Variable, in der der zu sendende Wert steht
    const ref VarAddr,
    // Zykluszeit, in der ein Telegramm gesendet werden soll
    // Event-Zeit
    EventTime: Int,
    // Mindestabstand zwischen zwei zu sendenden Telegrammen
    // Inhibit-Zeit
    InhibitTime: Int,
    Paramset: Int,       // Bitkodierter Parameter
) :Int;
```

Funktionsparameter

Die Funktion `CanOpenAddPDOTx()` hat die folgenden Parameter.

Parameter	Beschreibung	Wert
CANNo	CAN-Nummer der Busleitung	0 ... CANMAX
CANID	CAN-Identifier 11 Bit CAN-Identifier 29 Bit	0 ... 0x7FF 0 ... 0x1FFFFFFF
BytePos	Startposition der zu sendenden Daten	0 ... 7
DataType	Datentyp der zu sendenden Daten	2 ... 13, 15 ... 27
DataLength	Datengröße der globalen Variablen VarAddr	
VarAddr	Globale Variable, in die der zu sendende Wert eingetragen wird	

Parameter	Beschreibung	Wert
EventTime	Zeitlicher Abstand zwischen zwei Telegrammen (> InhibitTime)	
InhibitTime	Mindestabstand zwischen zwei zu sendenden Telegrammen (< EventTime)	
Paramset	Bitkodierter Parameter	

Rückgabewert

Die Funktion übergibt die folgenden Rückgabewerte an das übergeordnete Programm.

Rückgabewert	
0	Ok
-1	Fehler bei der Parameterüberprüfung
-3	DataType ist größer als DataLength
-4	Nicht genug Speicher vorhanden

CANNo-Parameter

Dieser Parameter gibt die Nummer der CAN-Schnittstelle an. Die erste Schnittstelle wird mit CANNo = 0 ausgewählt. Die Anzahl der CAN-Schnittstellen ist geräteabhängig. Die mögliche Anzahl der CAN-Schnittstellen CANMAX ist in den *technischen Daten* und der *Kurzreferenz* der jeweiligen Betriebsanleitung erwähnt.

CANID-Parameter

Mit dem Parameter **CANID** wird der CAN-Identifier übergeben. Der CAN-Identifier wird mit einem Makro erstellt. Der CAN-Identifier ist abhängig von der Node-ID des anderen Kommunikationsteilnehmers und abhängig davon, ob es sich um eine PDO1-, PDO2-, PDO3- oder PDO4-Nachricht handelt.

Makrodefinitionen:

```
#Define CANOPEN_PDO1_RX (NodeID)    ((NodeID) + 0x180)
#Define CANOPEN_PDO2_RX (NodeID)    ((NodeID) + 0x280)
#Define CANOPEN_PDO3_RX (NodeID)    ((NodeID) + 0x380)
#Define CANOPEN_PDO4_RX (NodeID)    ((NodeID) + 0x480)

#Define CANOPEN_PDO1_TX (NodeID)    ((NodeID) + 0x200)
#Define CANOPEN_PDO2_TX (NodeID)    ((NodeID) + 0x300)
#Define CANOPEN_PDO3_TX (NodeID)    ((NodeID) + 0x400)
#Define CANOPEN_PDO4_TX (NodeID)    ((NodeID) + 0x500)
```

Beispiel für den Aufruf des Makros:

CANOPEN_PDO2_RX (64)

⇨ Der daraus resultierende CAN-Identifier ist: 2C0h = 40h + 280h

Default-CAN-Identifier-Verteilung

Bei CANopen® ist die folgende CAN-Identifier-Verteilung vordefiniert. Dabei wird die Knotennummer in den Identifier eingebettet.

Identifier 11-Bit (binär)	Identifier (dezimal)	Identifier (hexadezimal)	Funktion
000000000000	0	0	Netzwerkmanagement
000100000000	128	80h	Synchronisation
0001xxxxxxxx	129 - 255	81h - FFh	Emergency
0011xxxxxxxx	385 - 511	181h - 1FFh	PDO1 (tx)
0100xxxxxxxx	513 - 639	201h - 27Fh	PDO1 (rx)
0101xxxxxxxx	641 - 767	281h - 2FFh	PDO2 (tx)
0110xxxxxxxx	769 - 895	301h - 37Fh	PDO2 (rx)
0111xxxxxxxx	897 - 1023	381h - 3FFh	PDO3 (tx)
1000xxxxxxxx	1025 - 1151	401h -47Fh	PDO3 (rx)
1001xxxxxxxx	1153 - 1279	481h - 4FFh	PDO4 (tx)
1010xxxxxxxx	1281 - 1407	501h - 57Fh	PDO4 (rx)
1011xxxxxxxx	1409 - 1535	581h - 5FFh	SDO senden
1100xxxxxxxx	1537 - 1663	601h - 67Fh	SDO empfangen
1110xxxxxxxx	1793 - 1919	701h - 77Fh	NMT Error Control
xxxxxxxx = Knotennummer 1 - 127			

DataType-Parameter

Folgende Datentypen können empfangen werden.

Byte-Typen	CANopen®-Format	Jetter-Format
1	CANOPEN_INTEGER8 CANOPEN_UNSIGNED8	Byte
2	CANOPEN_INTEGER16 CANOPEN_UNSIGNED16	Word
3	CANOPEN_INTEGER24 CANOPEN_UNSIGNED24	-
4	CANOPEN_INTEGER32 CANOPEN_UNSIGNED32 CANOPEN_REAL	Int
5	CANOPEN_INTEGER40 CANOPEN_UNSIGNED40	-
6	CANOPEN_INTEGER48 CANOPEN_UNSIGNED48 CANOPEN_TIME_OF_DAY CANOPEN_TIME_DIFFERENCE	-
7	CANOPEN_INTEGER56 CANOPEN_UNSIGNED46	-
8	CANOPEN_INTEGER64 CANOPEN_UNSIGNED64 CANOPEN_REAL64	-

Byte-Typen	CANopen®-Format	Jetter-Format
n	CANOPEN_VISIBLE_STRING CANOPEN_OCTET_STRING CANOPEN_UNICODE_STRING CANOPEN_DOMAIN	String

Paramset-Parameter

Folgende Parameter können an die Funktion übergeben werden. Mehrere Parameter können miteinander über die Oder-Funktion verknüpft werden.

CANOPEN_ASYNCPDORTRONLY
Sende asynchrone PDOs durch das Empfangen eines RTR-Frames.
Diese Funktion wird derzeit noch nicht unterstützt.

CANOPEN_ASYNCPDO
Sende asynchrone PDO.

CANOPEN_PDOINVALID
PDO wird nicht gesendet. Der benötigte Speicherplatz wird reserviert.

CANOPEN_NORTR
PDO kann nicht per RTR (Remote Request) angefordert werden.

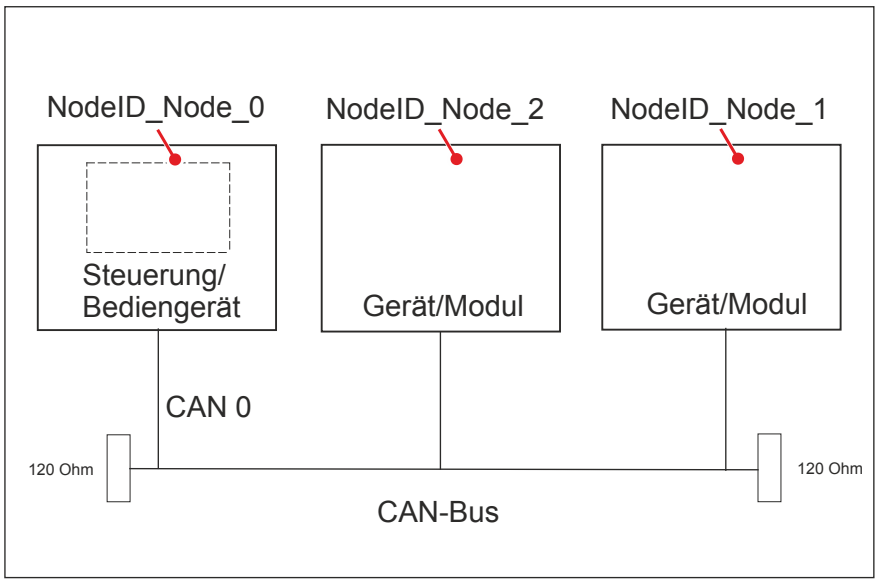
CANOPEN_29BIT
Verwende 29 Bit-Identifier
Default: 11 Bit-Identifier

Verwenden der Funktion

```
Result := CanOpenAddPDOTx(
    0,                                  // CANNo
    842,                                // CANID
    0,                                  // BytePos
    CANOPEN_DWORD,                      // DataType
    sizeof(var_Data_1_of_Node_3),       // DataLength
    var_Data_1_of_Node_3,               // VarAddr
    1000,                               // Event-Zeit
    100,                                // Inhibit-Zeit
    CANOPEN_ASYNCPDO | CANOPEN_NORTR);  // Paramset
```

JetSym-STX-Programm

JVM-104 sendet an zwei CANopen®-Geräte mit der Node-ID 74 und 112 Prozessdaten. Nach Ablauf des Programms und bei Änderungen sendet das Gerät JVM-104 zyklisch alle 3.000 ms (Event-Zeit) die PDO-Telegramme. Maximal wird alle 10 ms (Inhibit-Zeit) das PDO-Telegramm gesendet.




```
Const
    CANNo = 0;              // Nummer der Busleitung
    NodeID_Node_0 = 10;     // Geräte-ID von Node 1
    NodeID_Node_1 = 74;     // Geräte-ID von Node 2
    NodeID_Node_2 = 112;    // Geräte-ID von Node 3
    Event_Time = 3000;      // Event-Zeit in ms
    Inhibit_Time = 100;     // Inhibit-Zeit in ms
End_Const;

Var
    Data_1_of_Node_1: Int;
    Data_2_of_Node_1: Int;
    Data_1_of_Node_2: Byte;
End_Var;


Task main autorun

Var
    SW_Version: String;
End_Var;

SW_Version := 'v4.3.0.2004';

// Initialisierung CAN 0
CanOpenInit(CANNo,              // Nummer der Busleitung
            NodeID_Node_0,      // Geräte-ID
            SW_Version);        // Softwareversion des Herstellers
```

```
// Daten per PDO senden
CanOpenAddPDOTx(
    CANNo,                              // Nummer der Busleitung
    CANOPEN_PDO2_TX(NodeID_Node_1),     // CANID
    0,                                  // BytePos
    CANOPEN_DWORD,                      // DataType
    sizeof(Data_1_of_Node_1),           // DataLength
    Data_1_of_Node_1,                   // VarAddr
    Event_Time,                         // Event-Zeit
    Inhibit_Time,                       // Inhibit-Zeit
    CANOPEN_ASYNCPDORTRONLY);           // Paramset

CanOpenAddPDOTx(
    CANNo,                              // Nummer der Busleitung
    CANOPEN_PDO2_TX(NodeID_Node_1),     // CANID
    4,                                  // BytePos
    CANOPEN_DWORD,                      // DataType
    sizeof(Data_2_of_Node_1),           // DataLength
    Data_2_of_Node_1,                   // VarAddr
    Event_Time,                         // Event-Zeit
    Inhibit_Time,                       // Inhibit-Zeit
    CANOPEN_ASYNCPDORTRONLY);           // Paramset

CanOpenAddPDOTx(
    CANNo,                                  // Nummer der Busleitung
    CANOPEN_PDO3_TX(NodeID_Node_2),         // CANID
    0,                                      // BytePos
    CANOPEN_BYTE,                           // DataType
    sizeof(Data_1_of_Node_2),               // DataLength
    Data_1_of_Node_2,                       // VarAddr
    Event_Time,                             // Event-Zeit
    Inhibit_Time,                           // Inhibit-Zeit
    CANOPEN_ASYNCPDO | CANOPEN_NORTR);      // Paramset

// Alle Geräte am CAN-Bus sind im Status PREOPERATIONAL.

// Alle Geräte am CAN-Bus in den Status OPERATIONAL setzen.
CanOpenSetCommand(CANNo, CAN_CMD_NMT_Value(CAN_CMD_NMT_ALLNODES,
CAN_CMD_NMT), CAN_NMT_START);

// Ab jetzt werden PDO-Telegramme von den Geräten mit der NodeID 74
// und 112 übertragen.
//      ...

End_Task;
```

Heartbeat-Überwachung

Einleitung

Das Heartbeat-Protokoll dient zur Überwachung der Kommunikationspartner. Nach einer einstellbaren Zeit (Heartbeat consumer time), wird der Status auf **offline** gesetzt.

Im Anwendungsprogramm definieren Sie z. B.:

- Eine Information dem Benutzer anzeigen.
- Das Gerät neu starten.
- Die Prozessdaten ignorieren.

Voraussetzung

Die Funktion **Heartbeat-Überwachung** steht nur bei bestimmten Geräten zur Verfügung und ist abhängig von den OS-Versionen.

Gerät	OS-Version
FMC-01	Ab Version 1.18.1.00
JVM-C02	Ab Version 4.00.0.00
JCM-350	Ab Version 1.09.0.215
JCM-620	Ab JVER-Version 3.2.2.645 und JetVM-Version 3.04.0.00

Register der Heartbeat-Überwachung

Die Heartbeat-Überwachung belegt folgende Register.

Register	Registerbeschreibung	Datentyp	Attribute
40x001	Eigener Geräte-Heartbeat-Status; Wertebereich: 0 = Bootup 4 = Stopped 5 = Operational 127 = Preoperational 255 = Offline (Default-Wert)	Int	ro (read only)
40x100	Der Geräte-Heartbeat-Status aller überwachten Node-IDs hat sich geändert. Wertebereich: 0 = False 1 = True	Bool	rw (read and write)
40x101 ... 40x227	Geräte-Heartbeat-Status der Busteilnehmer Node-ID 1 ... 127; Wertebereich: 0 = Bootup 4 = Stopped 5 = Operational 127 = Preoperational 255 = Offline (Default-Wert)	Byte	ro
40x229 ... 40x355	Geräte-Heartbeat-Timeout der Busteilnehmer Node-ID 1 ... 127; Wertebereich: 0 ... 65535 [ms]	Word	rw

Das **x** in der Registernummer ist die Nummer von der verwendeten CAN-Busleitung: x = 0 ... CANMAX.

Starten der Heartbeat-Überwachung

Um die Heartbeat-Überwachung zu starten, gehen Sie wie folgt vor:

Schritt	Vorgehen
1	Schalten Sie die Heartbeat-Überwachung ein: Tragen Sie dafür den gewünschten Wert ins Timeout-Register ein. Der Wert muss zwischen 1 und 65535 [ms] liegen, z. B.: Für CAN 0 und Node-ID 1: Register 400229 auf den Wert 3000 [ms] setzen.
2	Legen Sie in Ihrem Anwendungsprogramm fest, wie das Gerät auf die einzelnen Werte aus dem Register (Geräte-Heartbeat-Status) reagieren soll. Wenn sich im Register 40x101 ... 40x227 der Status geändert hat, dann hat das Register 40x100 den Wert 1 (True).
3	Setzen Sie den Wert im Register 40x100 auf 0 (False) zurück. Dieser Schritt ist notwendig, damit nachfolgende Änderungen vom Register 40x101 ... 40x227 angezeigt werden.

Die Heartbeat-Überwachung startet mit dem Empfang des ersten Heartbeats (inclusive Bootup message). Der DLC (Data Length Code) von der Heartbeat-Nachricht muss 1 sein.

Beenden der Heartbeat-Überwachung

Um die Heartbeat-Überwachung zu beenden, gehen Sie wie folgt vor:

Schritt	Vorgehen
1	Schalten Sie die Heartbeat-Überwachung aus: Tragen Sie dafür ins Timeout-Register den Wert 0 [ms] ein.

Emergency-Nachricht

Wenn ein Heartbeat-Timeout erkannt wird, dann wird automatisch eine Emergency-Nachricht gesendet.

Wenn die nächste Heartbeat-Nachricht erfolgreich empfangen wird, dann wird die Emergency-Nachricht zurückgesetzt.

Beispiel:

Folgender Emergency-Nachricht wird ausgelöst:

Bezug	Wert
Error Code	0x8130
Error Register	0x81
Manufacturer Error	0x00,NodeID,0x00,0x00,0x00

Die Nachricht auf dem CAN-Bus sieht dann wie folgt aus:

- Eigene NodeID 5
- Überwachte NodeID 1
- ID: 0x85 DLC = 8 Data: 0x30 0x81 0x81 0x00 0x01 0x00 0x00 0x00

Emergency-Nachricht Rx

Die Deklaration der Emergency-Nachricht Rx ist wie folgt aufgebaut:

```
CanOpenAddEmergencyRx(
    CANNo:int,          // Nummer der Busleitung
    NodeID:int,         // Geräte-ID
    // Status, Anzahl gültiger Nachrichten
    ref stCanOpenEmergencyStat:CanOpenEmergencyStat,
    // Array mit den Emergency-Nachrichten
    ref CanOpenEmergencyMSG:CanOpenEmergencyArray,
    ):int
```

Beispiel:

Die einzelnen Programmzeilen müssen in den entsprechenden Task Ihres Anwendungsprogramms eingebunden werden. Das folgende Beispiel zeigt eine Emergency-Nachricht von einem Gerät mit der NodeID 21.

```
...
// Den CAN-Bus einmalig initialisieren.

...

// Globale Variablen definieren.
Var
    stCanOpenEmergencyMsg : ARRAY[5] of CanOpenEmergencyMsg;
    stCanOpenEmergencyStat : CanOpenEmergencyStat;
End_Var;

stCanOpenEmergencyStat.lBuffer := sizeof(stCanOpenEmergencyMsg);
iRet:= CanOpenAddEmergencyRx(0,                         // CANNo.
                             21,                        // NodeID
                             stCanOpenEmergencyStat,    // Status
                             stCanOpenEmergencyMsg);    // Array

...
```

Ergebnis der Programmzeilen:

Wenn im Register 400100 statt dem Wert 0 jetzt der Wert 1 (True) steht, dann hat das Gerät mit der NodeID 21 eine neue Emergency-Nachricht empfangen.

Setzen Sie diesen Wert immer wieder auf 0 (False), damit Sie den Empfang von weiteren Emergency-Nachrichten angezeigt bekommen.

Emergency-Nachricht Tx

Die Deklaration der Emergency-Nachricht Tx ist wie folgt aufgebaut:

```
CanOpenAddEmergencyTx(
    // Nummer der Busleitung
    CANNo:int,
    // Error Code siehe CiA DS 301 V4.02 Seite 60
    // oder CiA DS 4xx (Geräteprofil)
    ErrorCode:word,
    // Error Register (Object 0x1001)
    ErrorRegister:byte,
    // 5 Byte zur freien Verfügung
    ManufacturerArray:ByteArray5,
    // True = Fehler ist aufgetreten
    // False = Fehler ist nicht mehr vorhanden (Fehler quittiert)
    bSet:bool
    ):Int;
```

CANopen®-Objektverzeichnis des JVM-104

Unterstützte Objekte

Das Betriebssystem des Geräts JVM-104 unterstützt folgende Objekte:

Index (hex)	Objekt (Kürzel)	Objektname	Typ	Attribute
1000	VAR	Device Type	Unsigned32	ro (read only)
1001	VAR	Error Register	Unsigned8	ro
1002	VAR	Manufacturer Status	Unsigned32	ro
1003	ARRAY	Pre-defined Error Field	Unsigned32	ro
1008	VAR	Manufacturer Device Name	String	const
1009	VAR	Manufacturer Hardware Version	String	const
100A	VAR	Manufacturer Software Version	String	const
100B	VAR	Node-ID	Unsigned32	ro
1017	VAR	Producer Heartbeat Time	Unsigned16	rw (read & write)
1018	RECORD	Identity	Identity	ro
1200	RECORD	Server 1 - SDO-Parameter	SDO-Parameter	ro
1201	RECORD	Server 2 - SDO-Parameter	SDO-Parameter	rw
1203	RECORD	Server 3 - SDO-Parameter	SDO-Parameter	rw
1203	RECORD	Server 4 - SDO-Parameter	SDO-Parameter	rw

Device Type Object (Index 0x1000)

In der folgenden Tabelle ist die Struktur des **Device Type Object** dargestellt.

Index	Sub-Index	Default	Beschreibung
0x1000	0	0x0000012D	Geräteart (schreibgeschützt)

Error Register Object (Index 0x1001)

Die Bits in diesem Register werden über die STX-Funktion `CanOpenAddEmergencyTx()` gesetzt.

In der folgenden Tabelle ist die Struktur des **Error Register Object** dargestellt.

Index	Sub-Index	Default	Beschreibung
0x1001	0	0	Fehlerregister (schreibgeschützt)

Dieses Objekt übernimmt die Fehlerregisterfunktion von CANopen®.

Folgende Fehlermeldungen sind möglich:

- Bit 0 = nicht näher spezifizierter Fehler
- Bit 1 = Stromfehler aufgetreten
- Bit 2 = Spannungsfehler aufgetreten
- Bit 3 = Temperaturfehler aufgetreten
- Bit 4 = Kommunikationsfehler aufgetreten (overrun, error state)
- Bit 5 = spezifischer Geräteprofil-Fehler aufgetreten
- Bit 6 = Reserviert (Always 0)
- Bit 7 = Herstellerspezifischer Fehler aufgetreten

Pre-defined Error Field Object (Index 0x1003)

In der folgenden Tabelle ist die Struktur des **Pre-defined Error Field Object** dargestellt.

Index	Sub-Index	Default	Beschreibung
0x1003	0	0	Anzahl Fehler, die in das Standardfehlerfeld des Arrays eingetragen wurden
	1	0	Aktuellster Fehler 0 gibt an, dass kein Fehler vorliegt
	2 ... 254	-	Ältere Fehler

Dieses Objekt zeigt die Liste mit der Historie der vom JVM-104 erkannten Fehler. Die maximale Länge der Liste beträgt 254 Fehler. Bei einem Neustart wird der Inhalt der Liste gelöscht.

Aufbau des Standardfehlerfelds

2-Byte LSB: Fehlercode

2-Byte MSB: Ergänzende Informationen

Manufacturer Device Name Object (Index 0x1008)

In der folgenden Tabelle ist die Struktur des **Manufacturer Device Name Object** dargestellt.

Index	Sub-Index	Default	Beschreibung
0x1008	0	JVM-104	Name der Hardware

Manufacturer Hardware Version Object (Index 0x1009)

In der folgenden Tabelle ist die Struktur des **Manufacturer Hardware Version Object** dargestellt.

Index	Sub-Index	Default	Beschreibung
0x1009	0		OS-Version des Geräts

Manufacturer Software Version Object (Index 0x100A)

In der folgenden Tabelle ist die Struktur des **Manufacturer Software Version Object** dargestellt.

Index	Sub-Index	Default	Beschreibung
0x100A	0		Software-Version des Anwendungsprogramms, das auf dem JVM-104 läuft

Der Eintrag unter diesem Index erfolgt über den Parameter **SWVersion** der STX-Funktion `CanOpenInit()`.

Node-ID Object (Index 0x100B)

In der folgenden Tabelle ist die Struktur des **Node-ID Object** dargestellt.

Index	Sub-Index	Default	Beschreibung
0x100B	0		Eigene Node-ID

Producer Heartbeat Time Object (Index 0x1017)

In der folgenden Tabelle ist die Struktur des **Producer Heartbeat Time Object** dargestellt.

Index	Sub-Index	Default	Beschreibung
0x1017	0	1.000 [ms]	Heartbeat-Zeit

CANopen®-Register des Geräts JVM-104

In der folgenden Tabelle sind die Register des Geräts JVM-104 dargestellt, die in Bezug zum CANopen®-Objektverzeichnis stehen.

Der Buchstabe x in der Registernummer steht für die CAN-Busleitung im Bereich 0 ... CANMAX.

Register-nummer	Beschreibung	Wertebereich	Attribute	Datentyp
40x000	Eigene Node-ID	1 ... 127	rw (read & write)	Int
40x001	Eigener Heartbeat-Status	0 = Bootup 4 = Stopped 5 = Operational 127 = Pre-Operational 255 = Offline	ro (read only)	Int
40x002		siehe Objekt 0x1001	ro	Int
40x019			ro	Int (IP-Format)

Register-nummer	Beschreibung	Wertebereich	Attribute	Datentyp
40x020			rw	Int
40x021			rw	Int
40x022			rw	Int
40x023			rw	Int
40x030			rw	Int
40x100			rw	bool
40x400			rw	bool
40x101 ... 40x227	Node-ID 1 ... 127 Status	0 = Bootup 4 = Stopped 5 = Operational 127 = Pre-Operational 255 = Offline (Default)	ro	byte
40x229 ... 40x355	Node-ID 1 ... 127 Timeout	0 ... 65535 ms	rw	word

7 SAE J1939-STX-API

Einleitung

Dieses Kapitel beschreibt die STX-Funktionen der SAE J1939-STX-API.

Der SAE J1939-Standard

SAE J1939 ist ein offener Standard für die Vernetzung und Kommunikation im Nutzfahrzeugbereich. Schwerpunkt der Anwendung ist die Vernetzung von Antriebsstrang und Chassis. Das Protokoll J1939 stammt von der internationalen Society of Automotive Engineers (SAE) und arbeitet auf dem Physical Layer mit CAN-Highspeed nach ISO 11898.

Anwendung

Diese STX-Funktionen werden in der Kommunikation zwischen der Steuerung JVM-104 und anderen Steuergeräten im Fahrzeug angewendet. Es werden in der Regel Motordaten, z. B. Motordrehzahl, Geschwindigkeit oder Kühlwassertemperatur, ausgelesen und auf dem Display angezeigt.

Dokumente

Die wichtigsten SAE J1939-Spezifikationen sind:

- J1939-11 - Informationen zum Physical Layer
- J1939-21 - Informationen zum Data Link Layer
- J1939-71 - Informationen zum Application Layer Fahrzeuge
- J1939-73 - Informationen zum Application Layer Bereich Diagnose
- J1939-81 - Netzwerkmanagement

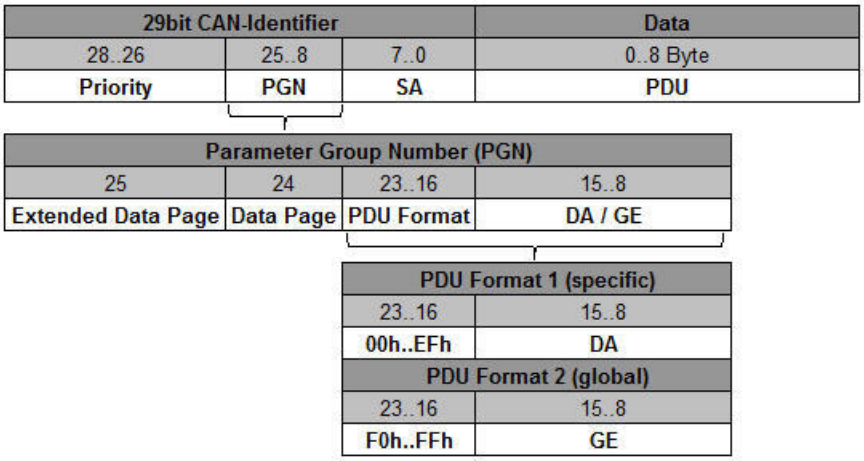
Inhalt

Thema	Seite
Aufbau einer J1939-Nachricht	124
STX-Funktion SAEJ1939Init()	126
STX-Funktion SAEJ1939SetSA()	127
STX-Funktion SAEJ1939GetSA()	128
STX-Funktion SAEJ1939AddRx()	129
STX-Funktion SAEJ1939AddTx()	132
STX-Funktion SAEJ1939RequestPGN()	135
STX-Funktion SAEJ1939GetDM1()	138
STX-Funktion SAEJ1939GetDM2()	141
STX-Funktion SAEJ1939SetSPNConversion()	144
STX-Funktion SAEJ1939GetSPNConversion()	145

Aufbau einer J1939-Nachricht

Aufbau einer J1939-Nachricht

Die folgende Abbildung zeigt den Aufbau einer J1939-Nachricht:



Abkürzung	Bezeichnung (Bedeutung)
DA	Destination Address (Zieladresse)
GE	Group Extensions (Gruppenerweiterungen)
PDU	Protocol Data Unit (Protokolldateneinheit)
PGN	Parameter Group Number (Parametergruppennummer)
SA	Source Address (Quelladresse)

Bedeutung der PGN - Parameter Group Number

Die PGN ist eine in der SAE J1939-Norm definierte Nummer, die mehrere SPNs zu einer sinnvollen Gruppe zusammenfügt. Die PGN ist Teil des CAN-Identifiers. Die 8-Byte-Daten (PDU) beinhalten die Werte der einzelnen SPN.

Es folgt ein Beispiel für die PGN 65262 (0xFEEE):

PGN 65262 Engine Temperature 1 - ET1

Teil der PGN	Wert	Bemerkung
Transmission Repetition Rate	1 s	
Data Length	8	
Extended Data Page	0	
Data Page	0	
PDU Format	254	
PDU Specific	238	PGN Supporting Information
Default Priority	6	
Parameter Group Number	65262	in Hex: 0xFEEE

Startposition	Länge	Parametername	SPN
1	1 Byte	Engine Coolant Temperature	110
2	1 Byte	Engine Fuel Temperature 1	174
3 - 4	2 Bytes	Engine Oil Temperature 1	175
5 - 6	2 Bytes	Engine Turbocharger Oil Temperature	176
7	1 Byte	Engine Intercooler Temperature	52
8	1 Byte	Engine Intercooler Thermostat Opening	1134

STX-Funktion SAEJ1939Init()

Einleitung

Mit dem Aufruf der Funktion `SAEJ1939Init()` wird einer der vorhandenen CAN-Busse (nicht CAN 0, weil reserviert für CANopen®) für das J1939-Protokoll initialisiert. Die JVM-104 hat ab dann die durch den Funktionsparameter mySA zugeteilte SA (Source Address). Es handelt sich dabei um die eigene Geräteadresse am Bus.

Funktionsdeklaration

```
Function SAEJ1939Init(
    CANNo:Int,
    mySA:Byte,
) :Int;
```

Funktionsparameter

Die Funktion `SAEJ1939Init()` hat die folgenden Parameter.

Parameter	Beschreibung	Wert
CANNo	CAN-Kanalnummer	1 ... CANMAX
mySA	Eigene Source-Adresse	0 ... 253

Rückgabewert

Die Funktion übergibt die folgenden Rückgabewerte an das übergeordnete Programm.

Rückgabewert	
0	ok
-1	Fehler bei der Parameterüberprüfung
-3	Nicht genug Speicher für SAE J1939

CANNo-Parameter

Dieser Parameter gibt die Nummer der SAEJ1939-Schnittstelle an. Die erste Schnittstelle wird mit CANNo = 1 ausgewählt. Die Anzahl der SAEJ1939-Schnittstellen ist geräteabhängig. Die mögliche Anzahl der SAEJ1939-Schnittstellen CANMAX ist in den *technischen Daten* und der *Kurzreferenz* der jeweiligen Betriebsanleitung erwähnt.

Verwenden der Funktion

Initialisieren des CAN-Busses 1. Das Gerät JVM-104 hat die SA 20 (0x14). Das Gerät JVM-104 kann jetzt Nachrichten mit der eingestellten SA (und nur diese Nachrichten) senden.

```
Result := SAEJ1939Init(1, 20);
```

Address Claiming

Address Claiming ist nicht implementiert.

STX-Funktion SAEJ1939SetSA()

Einleitung

Mit dem Aufruf der Funktion `SAEJ1939SetSA()` ändern Sie die eigene SA (Source Address) zur Laufzeit.

Funktionsdeklaration

```
Function SAEJ1939SetSA(
    CANNo:Int,
    mySA:Byte,
) :Int;
```

Funktionsparameter

Die Funktion `SAEJ1939SetSA()` hat die folgenden Parameter.

Parameter	Beschreibung	Wert
CANNo	CAN-Kanalnummer	1 ... CANMAX
mySA	Neue SA	0 ... 253

Rückgabewert

Die Funktion übergibt die folgenden Rückgabewerte an das übergeordnete Programm.

Rückgabewert	
0	ok
-1	Fehler bei der Parameterüberprüfung

CANNo-Parameter

Dieser Parameter gibt die Nummer der SAEJ1939-Schnittstelle an. Die erste Schnittstelle wird mit CANNo = 1 ausgewählt. Die Anzahl der SAEJ1939-Schnittstellen ist geräteabhängig. Die mögliche Anzahl der SAEJ1939-Schnittstellen CANMAX ist in den *technischen Daten* und der *Kurzreferenz* der jeweiligen Betriebsanleitung erwähnt.

Verwenden der Funktion

Die SA zur Laufzeit ändern.

Result := SAEJ1939SetSA(1, 20);

Wichtiger Hinweis

Nachrichten werden sofort mit der neuen SA gesendet oder empfangen.

STX-Funktion SAEJ1939GetSA()

Einleitung

Mit dem Aufruf der Funktion `SAEJ1939GetSA()` kann die eigene SA (Source Address) festgestellt werden.

Funktionsdeklaration

```
Function SAEJ1939GetSA(
    CANNo:Int,
    ref mySA:Byte,
) :Int;
```

Funktionsparameter

Die Funktion `SAEJ1939GetSA()` hat die folgenden Parameter.

Parameter	Beschreibung	Wert
CANNo	CAN-Kanalnummer	1 ... CANMAX
mySA	Aktuell eingestellte SA	0 ... 253

Rückgabewert

Die Funktion übergibt die folgenden Rückgabewerte an das übergeordnete Programm.

Rückgabewert	
0	ok
-1	Fehler bei der Parameterüberprüfung

CANNo-Parameter

Dieser Parameter gibt die Nummer der SAEJ1939-Schnittstelle an. Die erste Schnittstelle wird mit CANNo = 1 ausgewählt. Die Anzahl der SAEJ1939-Schnittstellen ist geräteabhängig. Die mögliche Anzahl der SAEJ1939-Schnittstellen CANMAX ist in den *technischen Daten* und der *Kurzreferenz* der jeweiligen Betriebsanleitung erwähnt.

Verwenden der Funktion

Es wird die aktuell eingestellte SA zurückgegeben.

```
Result := SAEJ1939SetSA(1, actual_SA);
```

STX-Funktion SAEJ1939AddRx()

Einleitung

Mit dem Aufruf der Funktion `SAEJ1939AddRx()` wird das JVM-104 aufgefordert, eine bestimmte Nachricht zu empfangen. Diese Nachricht wird von einem anderen Busteilnehmer gesendet. Die Adresse dieses Busteilnehmers wird als Parameter bySA dieser Funktion übergeben. Wenn die Nachricht nicht gesendet wird, bleibt der zuletzt empfangene Wert bestehen. Das zyklische Lesen erfolgt solange, bis die Funktion `SAEJ1939Init()` erneut aufgerufen wird.

Funktionsdeklaration

```
Function SAEJ1939AddRx(
    CANNo:Int,
    IPGN:Long,
    bySA:Byte,
    BytePos:Int,
    BitPos:Int,
    DataType:Int,
    DataLength:Int,
    const ref VarAddr,
    ref stJ1939:TJ1939Rx
    EventTime: Int,
    InhibitTime: Int,
) :Int;
```

Funktionsparameter

Die Funktion `SAEJ1939AddRx()` hat die folgenden Parameter.

Parameter	Beschreibung	Wert
CANNo	CAN-Kanalnummer	1 ... CANMAX
IPGN	PGN Parameter Group Number	0 ... 0x3FFFF
bySA	Source Address vom Sender der Nachricht	0 ... 253
BytePos	Startposition des Bytes der zu empfangenden Daten	1 ... n
BitPos	Startposition des Bits der zu empfangenden Daten	1 ... 8
DataType	Datentyp der zu empfangenden Daten	1 ... 3, 10 ... 16
DataLength	Datengröße der globalen Variablen VarAddr	
VarAddr	Globale Variable, in die der empfangene Wert eingetragen wird	
TJ1939Rx	Kontrollstruktur	
EventTime	Zeitlicher Abstand zw. zwei Telegrammen (> InhibitTime)	Default-Wert: 1.000 ms

Parameter	Beschreibung	Wert
InhibitTime	Mindestabstand zwischen zwei empfangenen Telegrammen (< EventTime)	Default-Wert: 100 ms

Rückgabewert

Die Funktion übergibt die folgenden Rückgabewerte an das übergeordnete Programm.

Rückgabewert	
0	ok
-1	Fehler bei der Parameterüberprüfung

CANNo-Parameter

Dieser Parameter gibt die Nummer der SAEJ1939-Schnittstelle an. Die erste Schnittstelle wird mit CANNo = 1 ausgewählt. Die Anzahl der SAEJ1939-Schnittstellen ist geräteabhängig. Die mögliche Anzahl der SAEJ1939-Schnittstellen CANMAX ist in den *technischen Daten* und der *Kurzreferenz* der jeweiligen Betriebsanleitung erwähnt.

DataType-Parameter

Die folgenden Datentypen sind möglich.

Byte-Typen	Bit-Typen	SAEJ1939
1	-	SAEJ1939_UNSIGNED8 SAEJ1939_BYTE
2	-	SAEJ1939_UNSIGNED16 SAEJ1939_WORD
4	-	SAEJ1939_UNSIGNED32 SAEJ1939_DWORD
n	-	SAEJ1939_STRING
-	1	SAEJ1939_1BIT
-	2	SAEJ1939_2BIT
-	3	SAEJ1939_3BIT
-	4	SAEJ1939_4BIT
-	5	SAEJ1939_5BIT
-	6	SAEJ1939_6BIT
-	7	SAEJ1939_7BIT

Kontrollstruktur TJ1939Rx

```
TJ1939Rx : Struct
// Status der empfangenen Nachricht
            byStatus      : Byte;
// Priorität der empfangenen Nachricht
            byPriority    : Byte;
           End_Struct;
```

Verwenden der Funktion

```
Result := SAEJ1939AddRx (
    1,
    0xFEEE,
    0x00,
    2
    0
    SAEJ1939_BYTE,
    sizeof(var_Fueltemp),
    var_Fueltemp,
    struct_TJ1939Rx_EngineTemperatureTbl,
    1500,
    120);
```

JetSym-STX-Programm

Das Gerät JVM-104 mit der eigenen SA von 20 möchte die aktuelle Treibstofftemperatur empfangen und anzeigen. Die Parameter **InhibitTime** und **EventTime** werden beim Funktionsaufruf nicht explizit aufgeführt. In diesem Fall werden die Default-Werte verwendet. Die Steuerung, die die Treibstofftemperatur erfasst, hat die SA von 0. In der Praxis ist die Adresse der Steuerung aus der Dokumentation des Motorherstellers zu erfahren.

Die Treibstofftemperatur hat die SPN 174 und ist Bestandteil (2. Byte) der PGN 65262 Motortemperatur 1.

```
#Include "SAEJ1939.stxp"

Var
    bySAEJ1939Channel  : Byte;
    own_Source_Address : Byte;

// PGN 65262 Engine Temperature 1
    Fueltemp  : Byte;
    EngineTemperatureTbl : TJ1939Rx;
End_Var;

Task main autorun

// Initialisierung CAN 1
bySAEJ1939Channel := 1;
own_Source_Address := 20;
SAEJ1939Init (bySAEJ1939Channel, own_Source_Address);

// Treibstofftemperatur empfangen
SAEJ1939AddRx (bySAEJ1939Channel, 65262, 0x00, 2, 1, SAEJ1939_BYTE,
sizeof(Fueltemp), Fueltemp, EngineTemperatureTbl);

End_Task;
```

Anleitung des Motorherstellers

Der Kunde erfährt die Daten (Priorität, PGN, SA und den Aufbau der Datenbytes) aus der Anleitung des Motorherstellers.

STX-Funktion SAEJ1939AddTx()

Einleitung

Mit dem Aufruf der Funktion `SAEJ1939AddTx()` wird das Gerät JVM-104 aufgefordert, eine bestimmte Nachricht zyklisch über den Bus zu senden.

Das zyklische Senden erfolgt solange, bis die Funktion `SAEJ1939Init()` erneut aufgerufen wird.

Die Daten werden nach Ablauf der Event-Time oder bei Änderung der abgegebenen Variablen und Ablauf der Inhibit-Time gesendet.

Funktionsdeklaration

```
Function SAEJ1939AddTx(
    CANNo:Int,
    IPGN:Long,
    BytePos:Int,
    BitPos:Int,
    dataType:Int,
    DataLength:Int,
    const ref VarAddr,
    ref stJ1939:TJ1939Tx
    EventTime: Int,
    InhibitTime: Int,
) :Int;
```

Funktionsparameter

Die Funktion `SAEJ1939AddTx()` hat die folgenden Parameter.

Parameter	Beschreibung	Wert
CANNo	CAN-Kanalnummer	1 ... CANMAX
IPGN	PGN Parameter Group Number	0 ... 0x3FFFF
BytePos	Startposition des Bytes der zu sendenden Daten	1 ... n
BitPos	Startposition des Bits der zu sendenden Daten	1 ... 8
dataType	Datentyp der zu sendenden Daten	1 ... 3, 10 ... 16
DataLength	Datengröße der globalen Variablen VarAddr	
VarAddr	Globale Variable, in die der sendende Wert eingetragen wird	
TJ1939Tx	Kontrollstruktur	
EventTime	Zeitlicher Abstand zw. zwei Telegrammen (> InhibitTime)	Default-Wert: 1.000 ms
InhibitTime	Mindestabstand zwischen zwei empfangenen Telegrammen (< EventTime)	Default-Wert: 100 ms

Rückgabewert

Die Funktion übergibt die folgenden Rückgabewerte an das übergeordnete Programm.

Rückgabewert	
0	ok
-1	Fehler bei der Parameterüberprüfung

CANNo-Parameter

Dieser Parameter gibt die Nummer der SAEJ1939-Schnittstelle an. Die erste Schnittstelle wird mit CANNo = 1 ausgewählt. Die Anzahl der SAEJ1939-Schnittstellen ist geräteabhängig. Die mögliche Anzahl der SAEJ1939-Schnittstellen CANMAX ist in den *technischen Daten* und der *Kurzreferenz* der jeweiligen Betriebsanleitung erwähnt.

DataType-Parameter

Die folgenden Datentypen sind möglich.

Byte-Typen	Bit-Typen	SAEJ1939
1	-	SAEJ1939_UNSIGNED8 SAEJ1939_BYTE
2	-	SAEJ1939_UNSIGNED16 SAEJ1939_WORD
4	-	SAEJ1939_UNSIGNED32 SAEJ1939_DWORD
n	-	SAEJ1939_STRING
-	1	SAEJ1939_1BIT
-	2	SAEJ1939_2BIT
-	3	SAEJ1939_3BIT
-	4	SAEJ1939_4BIT
-	5	SAEJ1939_5BIT
-	6	SAEJ1939_6BIT
-	7	SAEJ1939_7BIT

Kontrollstruktur TJ1939Tx

```
TJ1939Tx : Struct
// Status der gesendeten Nachricht
            byStatus      : Byte;
// Priorität der gesendeten Nachricht
            byPriority    : Byte;
          End_Struct;
```

Verwenden der Funktion

```
Result := SAEJ1939AddTx (
    1,
    0xFEEE,
    0x00,
    2
    0
    SAEJ1939_BYTE,
```

```
    sizeof(var_Fueltemp),
    var_Fueltemp,
    struct_TJ1939Tx_EngineTemperatureTbl,
    1500,
    120);
```

JetSym-STX-Programm

Festlegung einer neuen Priorität:

Der Prioritätswert 0 hat die höchste Priorität, der Prioritätswert 7 die niedrigste Priorität. Die Nachricht mit der Priorität 6 kann von einer Nachricht der Priorität 4 verdrängt werden (wenn die Nachrichten gleichzeitig gesendet werden). Die Parameter **InhibitTime** und **EventTime** werden beim Funktionsaufruf nicht explizit aufgeführt. In diesem Fall werden die Default-Werte verwendet.

```
#Include "SAEJ1939.stxp"

Var
    bySAEJ1939Channel  : Byte;
    own_Source_Address : Byte;

// PGN 65262 Engine Temperature 1
    Fueltemp  : Byte;
    EngineTemperatureTbl : TJ1939Tx;
End_Var;

Task main autorun

// Initialisierung CAN 1
bySAEJ1939Channel := 1;
own_Source_Address := 20;
SAEJ1939Init (bySAEJ1939Channel, own_Source_Address);

// PGN 65262 Engine Temperature
// Eine neue Priorität der festlegen
EngineTemperatureTbl.byPriority := 6;
SAEJ1939AddTx (bySAEJ1939Channel, 65262, 0x00, 2, 1, SAEJ1939_BYTE,
sizeof(Fueltemp), Fueltemp, EngineTemperatureTbl);

End_Task;
```

Anleitung des Motorherstellers

Der Kunde erfährt die Daten (Priorität, PGN, SA und den Aufbau der Datenbytes) aus der Anleitung des Motorherstellers.

STX-Funktion SAEJ1939RequestPGN()

Einleitung

Mit dem Aufruf der Funktion `SAEJ1939RequestPGN()` wird nach einer PGN eine Anfrage an die DA (Destination Address) gesendet.

Wenn ein gültiger Wert empfangen oder der Timeout von 1.250 ms abgelaufen ist, erst dann wird die Funktion beendet.

Um den Wert der angefragten Nachricht zu erhalten, muss diese mit der Funktion `SAEJ1939AddRx()` für den Empfang angemeldet sein.

Diese Funktion muss zyklisch immer wieder neu aufgerufen werden.

Funktionsdeklaration

```
Function SAEJ1939RequestPGN(
    CANNo:Int,
    byDA:Byte,
    ulPGN:Long,
    byPriority:Byte,
) :Int;
```

Funktionsparameter

Die Funktion `SAEJ1939RequestPGN()` hat die folgenden Parameter.

Parameter	Beschreibung	Wert
CANNo	CAN-Kanalnummer	1 ... CANMAX
byDA	Destination Address Adresse, von der die Nachricht angefordert wird	0 ... 253 Die eigene SA kann nicht verwendet werden.
ulPGN	PGN Parameter Group Number	0 ... 0x3FFFF
byPriority	Priorität	0 ... 7 Default-Wert: 6

Rückgabewert

Die Funktion übergibt die folgenden Rückgabewerte an das übergeordnete Programm.

Rückgabewert	
0	Nachricht wurde empfangen
-1	Timeout, weil keine Antwort erhalten

CANNo-Parameter

Dieser Parameter gibt die Nummer der SAEJ1939-Schnittstelle an. Die erste Schnittstelle wird mit CANNo = 1 ausgewählt. Die Anzahl der SAEJ1939-Schnittstellen ist geräteabhängig. Die mögliche Anzahl der SAEJ1939-Schnittstellen CANMAX ist in den *technischen Daten* und der *Kurzreferenz* der jeweiligen Betriebsanleitung erwähnt.

DataType-Parameter

Die folgenden Datentypen sind möglich.

Byte-Typen	Bit-Typen	SAEJ1939
1	-	SAEJ1939_UNSIGNED8 SAEJ1939_BYTE
2	-	SAEJ1939_UNSIGNED16 SAEJ1939_WORD
4	-	SAEJ1939_UNSIGNED32 SAEJ1939_DWORD
n	-	SAEJ1939_STRING
-	1	SAEJ1939_1BIT
-	2	SAEJ1939_2BIT
-	3	SAEJ1939_3BIT
-	4	SAEJ1939_4BIT
-	5	SAEJ1939_5BIT
-	6	SAEJ1939_6BIT
-	7	SAEJ1939_7BIT

Verwenden der Funktion

```
Result := SAEJ1939RequestPGN (
    1,
    0x00,
    0xFEE5,
    5);
```

JetSym-STX-Programm

JVM-104 mit der eigenen SA von 20 möchte die PGN 65253 *Engine Hours* von einer Motorsteuerung mit der SA 0 anfordern. Aus dieser PGN soll die SPN 247 *Engine Total Hours of Operation* ausgelesen werden. Deshalb ist es notwendig durch Aufruf der Funktion `SAEJ1939AddRx()` den Empfang der SPN 247 anzumelden.

Der Parameter **byPriority** wird beim Funktionsaufruf nicht explizit aufgeführt. In diesem Fall wird der Default-Wert verwendet.

```
#Include "SAEJ1939.stxp"

Var
    bySAEJ1939Channel  : Byte;
    own_Source_Address : Byte;

// PGN 65253 Engine Hours, Revolutions
    EngineTotalHours  : Int;
    EngineHoursTbl : TJ1939Rx;
End_Var;
```

```
Task main autorun

// Initialisierung CAN 1
bySAEJ1939Channel := 1;
own_Source_Address := 20;
SAEJ1939Init (bySAEJ1939Channel, own_Source_Address);

// Engine Hours, Revolutions  -- on Request
SAEJ1939AddRx (bySAEJ1939Channel, 65253, 0x00, 1, 0,
SAEJ1939_DWORD, sizeof(EngineTotalHours), EngineTotalHours,
EngineHoursTbl, 5000, 150);

// Wird benötigt für einen zyklischen Task
TaskAllEnableCycle ();
EnableEvents;

End_Task;


Task t_RequestPGN_5000 cycle 5000

Var
    Return_value : Int;
End_Var;

// Gesamte Betriebsstunden der Maschine anfordern
Return_value := SAEJ1939RequestPGN (bySAEJ1939Channel, 0x00,
65253);

If Return_value Then
      Trace ('PGN Request failed');
End_If;

End_Task;
```

STX-Funktion SAEJ1939GetDM1()

Einleitung

Mit dem Aufruf der Funktion `SAEJ1939GetDM1()` werden die aktuellen Diagnosefehlercodes angefordert (siehe SAE J1939-73 Nr. 5.7.1). Die entsprechende PGN-Nummer ist 65226. Diese Funktion muss zyklisch immer wieder neu aufgerufen werden.

Funktionsdeklaration

```
Function SAEJ1939GetDM1(
    CANNo:Int,
    bySA:Byte,
    ref stJ1939DM1stat:TJ1939DM1STAT
    ref stJ1939DM1msg:TJ1939DM1MSG
) :Int;
```

Funktionsparameter

Die Funktion `SAEJ1939GetDM1()` hat die folgenden Parameter.

Parameter	Beschreibung	Wert
CANNo	CAN-Kanalnummer	1 ... CANMAX
bySA	Source Address vom Sender der Nachricht	0 ... 253 Die eigene SA kann nicht verwendet werden
stJ1939DM1stat	lStatus lMsgCnt lBuffer	Lamp Status Anzahl der empfangenen Nachrichten Größe von Variable stJ1939DM1msg
stJ1939DM1msg	lSPN byOC byFMI	Fehlercode Fehlerzähler Fehlertyp

Rückgabewert

Die Funktion übergibt die folgenden Rückgabewerte an das übergeordnete Programm.

Rückgabewert	
0	ok
-1	Fehler bei der Parameterüberprüfung

CANNo-Parameter

Dieser Parameter gibt die Nummer der SAEJ1939-Schnittstelle an. Die erste Schnittstelle wird mit CANNo = 1 ausgewählt. Die Anzahl der SAEJ1939-Schnittstellen ist geräteabhängig. Die mögliche Anzahl der SAEJ1939-Schnittstellen CANMAX ist in den *technischen Daten* und der *Kurzreferenz* der jeweiligen Betriebsanleitung erwähnt.

stJ1939DM1stat.lStatus **Default**: 0xFF00

Typ	Byte	Bitgruppe	Beschreibung
Status	1	8 - 7	Malfunction Indicator Lamp Status
		6 - 5	Red Stop Lamp Status
		4 - 3	Amber Warning Lamp Status
		2 - 1	Protect Lamp Status
Flash	2	8 - 7	Flash Malfunction Indicator Lamp
		6 - 5	Flash Red Stop Lamp
		4 - 3	Flash Amber Warning Lamp
		2 - 1	Flash Protect Lamp

Typ	Byte	Bitgruppe Wert	Beschreibung
Status	1	00	Lamps off
		01	Lamps on
Flash	2	00	Slow Flash (1 Hz, 50 % duty cycle)
		01	Fast Flash (2 Hz or faster, 50 % duty cycle)
		10	Reserved
		11	Unavailable / Do not Flash

stJ1939DM1msg

Default-Wert:
ISPN = 0
byOC = 0
byFMI = 0
Bei älteren Controllern (Grandfathered Setting):
ISPN = 524287 (0x7FFFF)
byOC = 31 (0x1F)
byFMI = 127 (0x7F)

Verwenden der Funktion

```
Result := SAEJ1939GetDM1(
    1,
    0x00,
    stdm1stat_pow,
    stdm1msg_pow,);
```

JetSym-STX-Programm

JVM-104 fordert mit dem Aufruf der Funktion `SAEJ1939GetDM1()` die aktuellen Diagnosefehlercodes an (PGN 65226).

```
#Include "SAEJ1939.stxp"

Var
    bySAEJ1939Channel  : Byte;
    own_Source_Address : Byte;
    stdm1stat_pow : TJ1939DM1STAT;
    stdm1msg_pow : Array[10] of STJ1939DM1MSG;
    MyTimer : TTimer;
End_Var;

Task main autorun

// Initialisierung CAN 1
bySAEJ1939Channel := 1;
own_Source_Address := 20;
SAEJ1939Init (bySAEJ1939Channel, own_Source_Address);

TimerStart (MyTimer, T#2s);

Loop

When (TimerEnd (MyTimer)) Continue;

// Die Diagnose-Fehlercodes DM1 POW anfordern
stdm1stat_pow.lBuffer := sizeof (stdm1msg_pow);
SAEJ1939GetDM1 (bySAEJ1939Channel, 0x00, stdm1stat_pow,
stdm1msg_pow);

TimerStart (MyTimer, T#2s);

End_Loop;

End_Task;
```

STX-Funktion SAEJ1939GetDM2()

Einleitung

Mit dem Aufruf der Funktion `SAEJ1939GetDM2()` werden die Diagnosefehlercodes, die den Aktuellen vorausgingen, angefordert (siehe SAE J1939-73 Nr. 5.7.2). Die entsprechende PGN-Nummer ist 65227.

Funktionsdeklaration

```
Function SAEJ1939GetDM2(
    CANNo:Int,
    bySA:Byte,
    ref stJ1939DM2stat:TJ1939DM2STAT
    ref stJ1939DM2msg:TJ1939DM2MSG
) :Int;
```

Funktionsparameter

Die Funktion `SAEJ1939GetDM2()` hat die folgenden Parameter.

Parameter	Beschreibung	Wert
CANNo	CAN-Kanalnummer	1 ... CANMAX
bySA	Source Address vom Sender der Nachricht	0 ... 253 Die eigene SA kann nicht verwendet werden
stJ1939DM2stat	lStatus lMsgCnt lBuffer	Lamp-Status Anzahl der empfangenen Nachrichten Größe von Variable stJ1939DM2msg
stJ1939DM2msg	lSPN byOC byFMI	Fehlercode Fehlerzähler Fehlertyp

Rückgabewert

Die Funktion übergibt die folgenden Rückgabewerte an das übergeordnete Programm.

Rückgabewert	
0	ok
-1	Fehler bei der Parameterüberprüfung

CANNo-Parameter

Dieser Parameter gibt die Nummer der SAEJ1939-Schnittstelle an. Die erste Schnittstelle wird mit CANNo = 1 ausgewählt. Die Anzahl der SAEJ1939-Schnittstellen ist geräteabhängig. Die mögliche Anzahl der SAEJ1939-Schnittstellen CANMAX ist in den *technischen Daten* und der *Kurzreferenz* der jeweiligen Betriebsanleitung erwähnt.

stJ1939DM2stat.lStatus **Default**: 0xFF00

Typ	Byte	Bitgruppe	Beschreibung
Status	1	8 - 7	Malfunction Indicator Lamp Status
		6 - 5	Red Stop Lamp Status
		4 - 3	Amber Warning Lamp Status
		2 - 1	Protect Lamp Status
Flash	2	8 - 7	Flash Malfunction Indicator Lamp
		6 - 5	Flash Red Stop Lamp
		4 - 3	Flash Amber Warning Lamp
		2 - 1	Flash Protect Lamp

Typ	Byte	Bitgruppe Wert	Beschreibung
Status	1	00	Lamps off
		01	Lamps on
Flash	2	00	Slow Flash (1 Hz, 50 % duty cycle)
		01	Fast Flash (2 Hz or faster, 50 % duty cycle)
		10	Reserved
		11	Unavailable / Do not Flash

stJ1939DM2msg

Default-Wert:
ISPN = 0
byOC = 0
byFMI = 0
Bei älteren Controllern (Grandfathered Setting):
ISPN = 524287 (0x7FFFF)
byOC = 31 (0x1F)
byFMI = 127 (0x7F)

Verwenden der Funktion

```
Result := SAEJ1939GetDM2(
    1,
    0x00,
    stdm2stat_pow,
    stdm2msg_pow,);
```

JetSym-STX-Programm

JVM-104 fordert mit dem Aufruf der Funktion `SAEJ1939GetDM2()` die aktuellen Diagnosefehlercodes an (PGN 65227).

```
#Include "SAEJ1939.stxp"

Var
    bySAEJ1939Channel  : Byte;
    own_Source_Address : Byte;
    stdm2stat_pow : TJ1939DM2STAT;
    stdm2msg_pow : Array[10] of STJ1939DM2MSG;
End_Var;

Task main autorun
// Initialisierung CAN 1
bySAEJ1939Channel := 1;
own_Source_Address := 20;
SAEJ1939Init (bySAEJ1939Channel, own_Source_Address);

// Wird benötigt für einen zyklischen Task
TaskAllEnableCycle ();
EnableEvents;
End_Task;


Task t_RequestPGN_5000 cycle 5000

Var
    Return_value  : Int;
End_Var;

// Die Diagnosefehlercodes DM2 POW anfordern
stdm2stat_pow.lBuffer := sizeof (stdm2msg_pow);
Return_value := SAEJ1939GetDM2 (bySAEJ1939Channel, 0x00,
stdm2stat_pow, stdm2msg_pow);

If Return_value Then
    Trace ('DM2 Request failed');
End_If;

End_Task;
```

STX-Funktion SAEJ1939SetSPNConversion()

Einleitung

Mit dem Aufruf der Funktion `SAEJ1939SetSPNConversion()` wird die Anordnung der Bytes der Nachricht, die mit der Funktion `SAEJ1939GetDM1()` oder `SAEJ1939GetDM2()` angefordert wird, festgelegt. Anders ausgedrückt, es wird die Konvertierungsmethode festgelegt.

Funktionsdeklaration

```
Function SAEJ1939SetSPNConversion(
    CANNo:Int,
    bySA:Byte,
    iConversionMethod:Int,
) :Int;
```

Funktionsparameter

Die Funktion `SAEJ1939SetSPNConversion()` hat die folgenden Parameter.

Parameter	Beschreibung	Wert
CANNo	CAN-Kanalnummer	1 ... CANMAX
bySA	Source Address vom Sender der Nachricht	0 ... 253
iConversionMethod	Konvertierungsmethode	1 ... 4 4: Automatisch erkannt 2: Default

Rückgabewert

Die Funktion übergibt die folgenden Rückgabewerte an das übergeordnete Programm.

Rückgabewert	
0	ok
-1	Fehler bei der Parameterüberprüfung

CANNo-Parameter

Dieser Parameter gibt die Nummer der SAEJ1939-Schnittstelle an. Die erste Schnittstelle wird mit CANNo = 1 ausgewählt. Die Anzahl der SAEJ1939-Schnittstellen ist geräteabhängig. Die mögliche Anzahl der SAEJ1939-Schnittstellen CANMAX ist in den *technischen Daten* und der *Kurzreferenz* der jeweiligen Betriebsanleitung erwähnt.

Verwenden der Funktion

```
Result := SAEJ1939SetSPNConversion(
    1,
    0xAE,
    4);
```

STX-Funktion SAEJ1939GetSPNConversion()

Einleitung

Mit dem Aufruf der Funktion `SAEJ1939GetSPNConversion()` wird die aktuell eingestellte Konvertierungsmethode festgestellt.

Funktionsdeklaration

```
Function SAEJ1939SetSPNConversion(
    CANNo:Int,
    bySA:Byte,
    iConversionMethod:Int,
) :Int;
```

Funktionsparameter

Die Funktion `SAEJ1939GetSPNConversion()` hat die folgenden Parameter.

Parameter	Beschreibung	Wert
CANNo	CAN-Kanalnummer	1 ... CANMAX
bySA	Source Address vom Sender der Nachricht	0 ... 253
iConversionMethod	Konvertierungsmethode	1 ... 4 4: Automatisch erkannt 2: Default

Rückgabewert

Die Funktion übergibt die folgenden Rückgabewerte an das übergeordnete Programm.

Rückgabewert	
0	ok
-1	Fehler bei der Parameterüberprüfung

CANNo-Parameter

Dieser Parameter gibt die Nummer der SAEJ1939-Schnittstelle an. Die erste Schnittstelle wird mit CANNo = 1 ausgewählt. Die Anzahl der SAEJ1939-Schnittstellen ist geräteabhängig. Die mögliche Anzahl der SAEJ1939-Schnittstellen CANMAX ist in den *technischen Daten* und der *Kurzreferenz* der jeweiligen Betriebsanleitung erwähnt.

Verwenden der Funktion

```
Result := SAEJ1939GetSPNConversion(
    1,
    0xAE,
    actual_conversion_method);
```

8 Dateisystem

Einleitung

Dieses Kapitel beschreibt das Dateisystem des Geräts JVM-104. Das Dateisystem ermöglicht den Zugriff auf die Dateien der internen Flash-Disk und eines USB-Sticks.

Wenn zu lösende Probleme auftreten, ist es hilfreich, das Dateisystem zu verstehen.

Hinweis

Arbeiten Sie vorsichtig im Umgang mit dem Dateisystem, zumindest mit den Systemdateien. Das kann zu einem Gerät führen, das nicht mehr bootet.

Nicht alle Dateien sind lesbar, schreibbar oder löschbar. Dieses Verhalten ist normal. Einige dieser Dateien sind virtuelle Dateien, z. B. Firmware Images, oder geschützte Dateien, z. B. EDS-Dateien.

Aufteilung

Das Dateisystem unterscheidet zwischen folgenden Dateien:

- Vom Betriebssystem verwendete Systemverzeichnisse/-dateien
- Vom Anwender verwendete Dateien

Inhalt

Thema	Seite
Verzeichnisse	148
Eigenschaften	152

8.1 Verzeichnisse

Systemverzeichnisse

Systemverzeichnisse sind nicht löschbar. Sie sind auch nach dem Formatieren noch vorhanden.

Verzeichnis	Beschreibung
\System	■ Systemkonfiguration ■ Systeminformationen ■ Begrüßungsbild (Boot-Image) ■ Screenshot
\USB	■ Root-Verzeichnis des USB-Sticks
\App	■ Speicherbereich für die Applikation
\Data	■ Speicherbereich für die Daten
\Windows	■ Systemverzeichnis von Windows CE
\	■ RAM-Disk

Inhalt

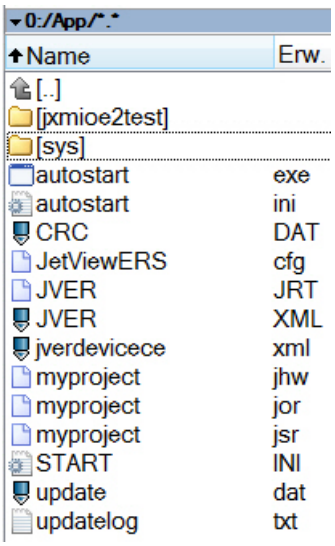
Thema **Seite**
Verzeichnisse 149

Verzeichnisse

Verzeichnis *\App*

\App

Dieses Verzeichnis enthält die Anwendung und die Visualisierung. In diesem Beispiel ist die STX-Anwendung in dem Ordner **jxmioe2test** gespeichert.



Die Daten basieren auf der neuen CE-Plattform. Da gibt es keine .iop-Datei. Stattdessen speichert JetViewSoft mehrere Visualisierungsdateien ab.

Hinweis:
Kopieren Sie alle Anwendungs- und Visualisierungsdateien in den Ordner **App** und nicht in den Ordner **Data**. Sonst verursacht das Verzögerungen beim Hochfahren, siehe Verzeichnis *\Data*.

start.ini

Der Inhalt dieser Textdatei legt fest, welche Anwendung gestartet wird.

\App\sys

Dieses Verzeichnis enthält die Interpreter der STX-Programmiersprache und der Visualisierung. **Hier nichts verändern!**

autostart.xxx

Diese Anwendung lässt ein Betriebssystemupdate zu. **Hier nichts verändern!**

updatelog.txt

Das ist eine Log-Datei, die während des Betriebssystemupdates geschrieben wird.

Verzeichnis *Data*

Data

Dieses Verzeichnis ist der große Speicher des Bediengeräts. Sie können einige Parameter- oder Konfigurationsdateien hier speichern.

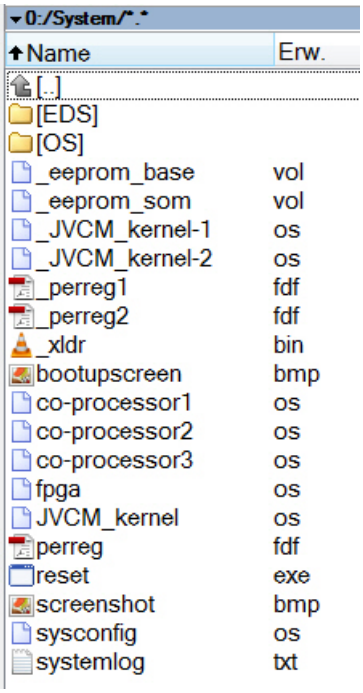
Wichtiger Hinweis!

Auf der Datenpartition können größere Datenmengen abgelegt werden. Um einen zügigen Systemstart sicherzustellen, wird diese Partition ggf. zeitversetzt nach dem Start der STX-Applikation eingebunden. Die Applikation kann nicht auf dieser Partition abgelegt werden.

Verzeichnis *System*

System

Dieses Verzeichnis enthält systemrelevante Dateien, wie den Kernel, die Co-Prozessor-Firmware, Konfigurationen, EDS usw.



bootupscreen.bmp

Das ist eine 16-Bit-Bmp-Datei (r5, g6, b5), die als Bild direkt nach dem Einschalten sichtbar ist.

Sie können eine eigene Bilddatei erstellen und diese Datei ersetzen.

co-processor1

Diese virtuelle Datei enthält die Firmware eines versteckten Co-Prozessors, der die meisten Interaktionen (Schaltflächen, Summer, Hintergrundlicht usw.) mit dem Anwender steuert.

reset.exe

Wenn Sie diese Datei löschen, führt das Bediengerät sofort einen Neustart aus. Diese Funktion können Sie z. B. in Batch-Dateien anwenden, die nach der Abarbeitung einen automatischen Neustart erfordern.

Verzeichnis *USB*

USB

Wenn Sie einen USB-Stick in das Bediengerät stecken, wird das Verzeichnis *\USB* dynamisch erstellt. Sonst ist dieses Verzeichnis nicht sichtbar.

Verzeichnis *Windows*

Windows

Dieses Verzeichnis enthält Dateien von Windows CE. **Hier nichts verändern!**

8.2 Eigenschaften

Einleitung

Dieses Kapitel beschreibt die Eigenschaften des Dateisystems. Unterschieden wird dabei zwischen der internen Flash-Disk und dem USB-Stick.

Allgemeine Eigenschaften

Für die interne Flash-Disk und den USB-Stick gelten folgende Eigenschaften:

- Maximal 8 Dateien gleichzeitig öffnen.
- Wenn das Bediengerät eine Datei erstellt, erhält die Datei Datum und Uhrzeit des Bediengeräts.
- Datum, Uhrzeit und Dateigröße sind nicht bei allen Systemdateien verfügbar.

Inhalt

Thema	Seite
Eigenschaften der Flash-Disk	153
Eigenschaften des USB-Sticks	154

Eigenschaften der Flash-Disk

Größe

Dem Anwender steht folgende Größe zur freien Verfügung:

Parameter	Wert
Größe der Flash-Disk	512 MByte
Davon Ordner **App**	64 MByte
Davon Ordner **Data**	368 MByte

Eigenschaften

Die interne Flash-Disk hat folgende weitere Eigenschaften:

- 7 Verzeichnisebenen und 1 Dateiebene sind zulässig.
- Eine Unterscheidung zwischen Groß- und Kleinschreibung gibt es nicht.
- Für die Zeichenlänge von Verzeichnis- und Dateinamen sind gesamt 63 Zeichen zulässig.
- Für Verzeichnis- und Dateinamen sind alle Zeichen außer "/" und ".." erlaubt.
- Die Ordner **App** und **Data** liegen auf der Flash-Disk.

Eigenschaften des USB-Sticks

Größe

Die Größe ist abhängig von dem verwendeten USB-Stick:

Parameter	Wert
Getestete Größe	256 MByte ... 8 GByte

Eigenschaften

Der USB-Stick hat folgende weitere Eigenschaften:

- Der USB-Stick muss FAT 16 oder FAT 32 kompatibel sein.
- Verzeichnis- und Dateinamen mit gesamt 260 Zeichen sind möglich.
- Groß- und Kleinschreibung wird unterschieden.
- In Verzeichnis- und Dateinamen sind nicht erlaubt: "/", "\", ":", "*", "?", """, "<", ">" und "|"
- Es gibt keine Benutzer-/Zugriffsverwaltung.

Die Jetter AG übernimmt keine Gewähr für die Funktion aller am Markt befindlichen USB-Sticks.

9 FTP-Server

Einleitung

Der FTP-Server erlaubt den Zugriff auf Verzeichnisse und Dateien mit einem FTP-Client. Die Dateien können auf folgenden Speichermedien abgespeichert sein:

- Interne Flash-Disk
- USB-Stick

Dieses Kapitel beschreibt den Vorgang der Anmeldung und die vom FTP-Server unterstützten Kommandos.

FTP-Clients

Außer dem Kommandozeilen-FTP-Client, wie er bei vielen PC-Betriebssystemen mitgeliefert wird, können Sie auch grafische FTP-Tools verwenden.

Anzahl möglicher Verbindungen

Der FTP-Server des JVM-104 verwaltet maximal vier gleichzeitig geöffnete FTP-Verbindungen.

Jeder weitere FTP-Client, der sich versucht mit dem FTP-Server zu verbinden, erhält auf seine Anfrage zum Aufbau einer Verbindung keine Antwort.

Unterstützte Kommandos

Der FTP-Server unterstützt die standardisierten Kommandos. Sie finden diese unter:

- WIKIPEDIA
- Google; nach FTP und Kommandos suchen
- Help-Menü des FTP-Servers; mit FTP-Server verbinden und als Kommando *help* oder *help binary* eingeben

Wenn Sie sich nicht um die Kommandos kümmern wollen, verwenden Sie ein FTP-Programm wie TotalCmd.

Kenntnisse des Programmierers

Dieses Kapitel setzt folgende Kenntnisse voraus:

- Kenntnisse über das Dateisystem der Steuerung sind erforderlich.
- Kenntnisse über IP-Netzwerke sind erforderlich.
- Kenntnisse über die Kommandos sind erforderlich.

Inhalt

Thema	Seite
Anmeldung	156
Beispiel: Windows FTP-Client	157

Anmeldung

Anmeldung

Um auf das Dateisystem über FTP zugreifen zu können, muss sich der FTP-Client bei der Aufnahme der Kommunikation wie folgt anmelden:

- Bei **Servername** geben Sie die IP-Adresse des Geräts ein.
- Bei **Benutzername** geben Sie den Benutzernamen ein, z. B. *admin*.
- Bei **Passwort** geben Sie das Passwort ein, z. B. *admin*.

Auslieferungszustand

Im Auslieferungszustand ist ein Benutzer (User) eingerichtet:

NAME=admin
PW=admin

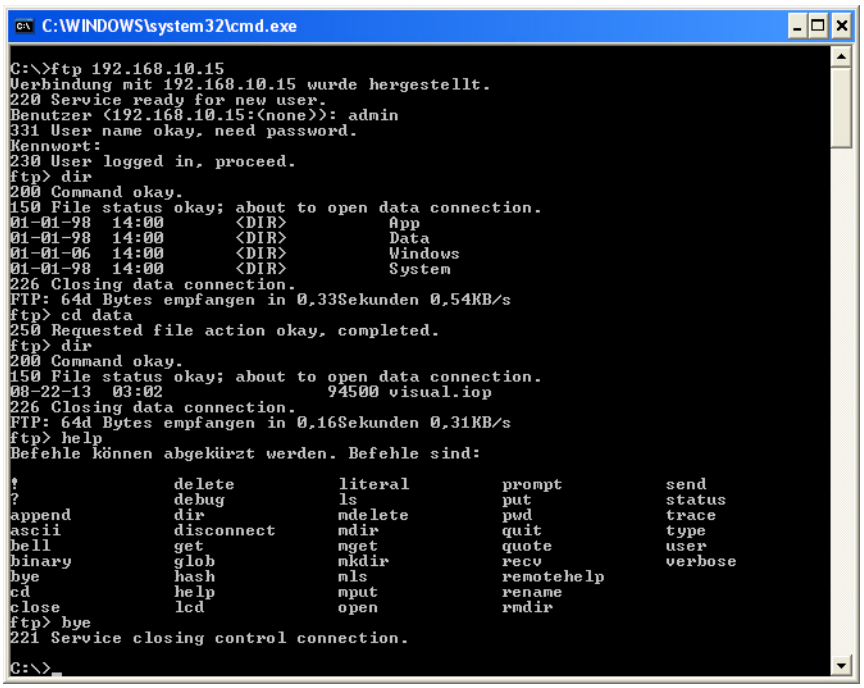
Beispiel: Windows FTP-Client

Aufgabe

Bearbeiten Sie mit dem FTP-Client von z. B. Windows XP die folgenden Aufgaben:

- Aufruf des FTP-Clients mit Verbindung öffnen und Eingabe der IP-Adresse
- Login mit Benutzer *admin* und Passwort *admin*
- Anzeigen des Inhalts des aktuellen Verzeichnisses mit *dir*
- Wechseln in das Verzeichnis *App* mit *cd app*
- Anzeigen des Inhalts des aktuellen Verzeichnisses mit *dir*
- Mögliche Kommandos anzeigen lassen mit *help*.
- Beenden der Sitzung und des FTP-Clients mit *bye*

Vorgehen



10 Programmierung

Zweck des Kapitels

Dieses Kapitel unterstützt die Programmierung des Bediengeräts JVM-104 in folgenden Punkten:

- Programmierung der Zusatzfunktionen

Voraussetzungen

Zur Programmierung des Bediengeräts JVM-104 müssen folgende Voraussetzungen erfüllt sein:

- Das Bediengerät ist mit einem PC verbunden.
- Auf dem PC ist die Programmiersoftware JetSym installiert.

Inhalt

Thema	Seite
Abkürzungen, Modulregistereigenschaften und Formatierungen	160
Speicherübersicht	161
Bedienelemente und Zündung	172
Multifunktionseingänge/-ausgänge	179
Laufzeitregister	196
Überwachung der Schnittstellenaktivität	201
Freiprogrammierbare IP-Schnittstelle	206

Abkürzungen, Modulregistereigenschaften und Formatierungen

Abkürzungen

In der folgenden Tabelle sind die in diesem Dokument benutzten Abkürzungen aufgelistet:

Abkürzung	Bedeutung
R 100	Register 100
MR 150	Modulregister 150

Modulregistereigenschaften

Jedes Modulregister ist durch bestimmte Eigenschaften gekennzeichnet. Die meisten Eigenschaften sind bei vielen Modulregistern identisch, z. B. dass der Wert nach einem Reset gleich Null ist. In der Beschreibung sind die Modulregistereigenschaften nur dann aufgeführt, wenn eine Eigenschaft von den folgenden Standardeigenschaften abweicht.

Modulregistereigenschaften	Standard für die meisten Modulregister
Zugriff	Lesen/schreiben
Wert nach einem Reset	0 oder undefiniert (z. B. die Versionsnummer)
Wird wirksam	Sofort
Schreibzugriff	Immer
Datentyp	Integer

Zahlenformate

In der folgenden Tabelle sind die in diesem Dokument benutzten Zahlenformate aufgelistet:

Darstellung	Zahlenformat
100	Dezimal
0x100	Hexadezimal
0b100	Binär

JetSym-Beispielprogramme

In der folgenden Tabelle ist die in diesem Dokument benutzte Darstellung für Beispielprogramme aufgelistet:

Darstellung	Bedeutung
`Var, When, Task`	Schlüsselwort
`BitClear();`	Befehle
`100 0x100 0b100`	Konstante Zahlenwerte
`// dies ist ein Kommentar`	Kommentar
`// ...`	Weitere Programmbearbeitung

10.1 Speicherübersicht

Einleitung

Das Gerät JVM-104 enthält verschiedene Arten von Programm- und Datenspeicher. Es gibt flüchtigen Speicher. Flüchtiger Speicher verliert beim Ausschalten seinen Inhalt. Nicht flüchtiger Speicher behält auch ohne Stromversorgung seinen Inhalt.

Dieses Kapitel gibt eine Übersicht über den zur Verfügung stehenden Speicher.

Inhalt

Thema	Seite
Speicher des Betriebssystems	162
Speicher des Dateisystems	163
Speicher des Anwendungsprogramms	164
Speicher für flüchtige Variablen des Anwendungsprogramms	165
Speicher für nichtflüchtige Register des Anwendungsprogramms	166
Speicher für nichtflüchtige Variablen des Anwendungsprogramms	167
Spezialregister	169
Merker	170

Speicher des Betriebssystems

Einleitung	Das Betriebssystem liegt im nichtflüchtigen Flash-Speicher der CPU. Das Betriebssystem ist nach dem Einschalten sofort ausführbar.
Eigenschaften	▪ Interner Flash-Speicher für das Betriebssystem ▪ Interner flüchtiger RAM-Speicher für die Daten des Betriebssystems
Speicherzugriff	▪ Auf den Speicherbereich des Betriebssystems kann der Anwender nicht direkt zugreifen. ▪ Das Betriebssystem wird über ein Update geändert.
Verwandte Themen	▪ **Update des Betriebssystems** (siehe Seite 248)

Speicher des Dateisystems

Einleitung	Im Speicher des Dateisystems werden Daten- und Programmdateien abgelegt.
Eigenschaften	▪ Nichtflüchtig ▪ Größe der internen Flash-Disk: 368 MByte ▪ Größe des USB-Sticks: 256 MByte bis 8 GByte
Speicherzugriff	▪ Durch das Betriebssystem ▪ Durch JetSym ▪ Über eine FTP-Verbindung ▪ Browser (über den HTTP-Server) ▪ Über Dateibefehle aus dem Anwendungsprogramm

Speicher des Anwendungsprogramms

Einleitung

Das Anwendungsprogramm (STX-Skript) wird standardmäßig von JetSym an das Bediengerät übertragen und dort gespeichert.

Eigenschaften

- Ablage als Datei im Dateisystem
- Standardverzeichnis *\App\Programmname*
- In anderen Verzeichnissen (auch USB-Stick) möglich

Speicherzugriff

- Durch das Betriebssystem
- Durch JetSym
- Über eine FTP-Verbindung
- Über Dateibefehle aus dem Anwendungsprogramm

Verwandte Themen

- **Anwendungsprogramm** (siehe Seite 257)

Speicher für flüchtige Variablen des Anwendungsprogramms

Einleitung

In flüchtigen Variablen werden Daten abgelegt, die nach dem Ausschalten der JVM-104 nicht erhalten bleiben müssen.

Eigenschaften

- Globale Variablen, die nicht fest Adressen zugeordnet sind (nicht %VL oder %RL)
- Lokale Variablen
- Variablen sind kompakt abgelegt
- Variablen sind beim Anlegen mit dem Wert 0 initialisiert

Speicherzugriff

- Durch JetSym
- Aus dem Anwendungsprogramm

JetSym-STX-Programm

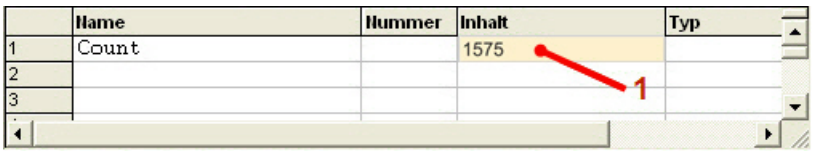
Das folgende Programm inkrementiert den Inhalt einer globalen Variable alle 2 s um eins.

```
Var
    Count:    Int;
End_Var;

Task Inkrement Autorun
    Loop
        Inc(Count);
        Delay(T#2s);
    End_Loop;
End_Task;
```

Anzeige im Setup

Das Setup-Fenster von JetSym zeigt den Inhalt der Variablen an.



Nummer	Beschreibung	Funktion
1	Aktueller Inhalt der Variablen	Der Inhalt der Variablen wird alle 2 s um eins inkrementiert.

Speicher für nichtflüchtige Register des Anwendungsprogramms

Einleitung

In nichtflüchtigen Registern werden Daten abgelegt, die nach dem Ausschalten des JVM-104 erhalten bleiben müssen.

Eigenschaften

- Globale Variablen, die fest Adressen zugeordnet sind (%VL)
- Registervariablen belegen immer 4 Byte
- Registervariablen werden vom Betriebssystem nicht initialisiert
- Anzahl Registervariablen: 30.000
- Registernummern: 1000000 ... 1029999

Speicherzugriff

- Durch JetSym
- Browser (über den HTTP-Server)
- Von Anzeige- und Bediengeräten
- Aus dem Anwendungsprogramm
- Von Steuerungen

JetSym-STX-Programm

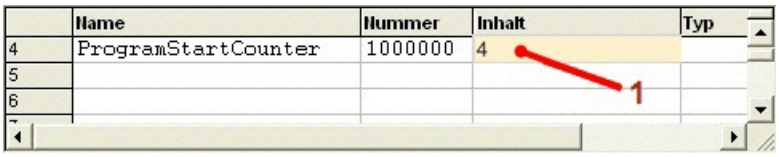
Das folgende Programm inkrementiert den Inhalt einer Registervariable bei jedem Anwendungsprogrammstart um eins. Auf diese Weise wird die Anzahl der Programmstarts gezählt.

```
Var
    ProgramStartCounter:    Int At %VL 1000000;
End_Var;

Task Work Autorun
    ProgramStartCounter := ProgramStartCounter + 1;
    Loop
        // ...
    End_Loop;
End_Task;
```

Anzeige im Setup

Das Setup-Fenster von JetSym zeigt den Inhalt der Registervariablen an.



Nummer	Beschreibung	Funktion
1	Aktueller Inhalt der Registervariablen	Der Inhalt der Registervariablen wird bei jedem Programmstart um eins erhöht.

Speicher für nichtflüchtige Variablen des Anwendungsprogramms

Einleitung

In nichtflüchtigen Variablen werden Daten abgelegt, die nach dem Ausschalten des JVM-104 erhalten bleiben müssen.

Eigenschaften

- Globale Variablen, die auf Register fest zugeordnet sind (%RL)
- Variablen werden kompakt abgelegt
- Größe: 120.000 Bytes
- Registernummern: 1000000 ... 1029999

Speicherzugriff

- Durch JetSym
- Von Anzeige- und Bediengeräten
- Aus dem Anwendungsprogramm

JetSym-STX-Programm

Das folgende Programm inkrementiert jede Sekunde den Inhalt von vier nichtflüchtigen Variablen.

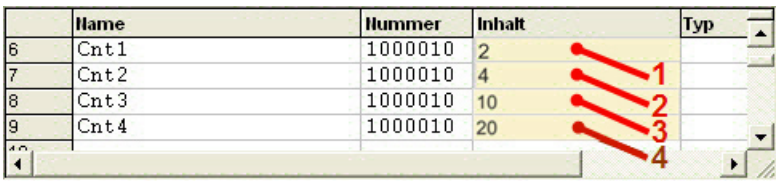
Die Zähler nehmen jeweils Werte zwischen 0 und 255 (Variablentyp Byte) an. Als Speicher für die vier Variablen werden die vier Bytes des Registers mit der Nummer 1000010 verwendet.

```
Var
    Cnt1, Cnt2, Cnt3, Cnt4:    Byte At %RL 1000010;
End_Var;

Task Count4 Autorun
    Loop
        Inc(Cnt1);
        Inc(Cnt2, 2);
        Inc(Cnt3, 5);
        Inc(Cnt4, 10);
        Delay(T#1s);
    End_Loop;
End_Task;
```

Anzeige im Setup

Das Setup-Fenster von JetSym zeigt den Inhalt der Variablen an. Da die vier Zähler vom Typ Byte sind, ergeben sich schon nach relativ kurzer Zeit Zahlenüberläufe.



Nummer	Beschreibung	Funktion
1	Aktueller Inhalt der Variablen Cnt1	Der Inhalt der Variablen wird jede Sekunde um eins inkrementiert.
2	Aktueller Inhalt der Variablen Cnt2	Der Inhalt der Variablen wird jede Sekunde um zwei inkrementiert.
3	Aktueller Inhalt der Variablen Cnt3	Der Inhalt der Variablen wird jede Sekunde um fünf inkrementiert.
4	Aktueller Inhalt der Variablen Cnt4	Der Inhalt der Variablen wird jede Sekunde um zehn inkrementiert.

Spezialregister

Einleitung

Über Spezialregister können Funktionen im Betriebssystem gesteuert und Statusinformationen abgerufen werden.

Eigenschaften

- Globale Variablen, die fest Adressen zugeordnet sind (%VL)
- Spezialregister werden beim Start des Betriebssystems mit Standardwerten vorbesetzt
- Registernummern: 100000 ... 999999

Speicherzugriff

- Durch JetSym
- Browser (über den HTTP-Server)
- Von Anzeige- und Bediengeräten
- Aus dem Anwendungsprogramm
- Von anderen Steuerungen

JetSym-STX-Programm

Das folgende Programm verwendet das Spezialregister für den Digipot-Wert. Es dimmt die Hintergrundbeleuchtung des JVM-104 über den Digipot. Dabei ist ein unteres und oberes Limit für den Digipot vorgegeben. Wenn Sie die Betätigungstaste drücken, stellt sich die volle Hintergrundbeleuchtung ein.

```
Var
    Digipot_Count    :  Int At %VL 363000;
    Digipot_Limit_min:  Int At %VL 363002;
    Digipot_Limit_max:  Int At %VL 363003;
    Digipot_Button   :  Int At %VL 363001;
    BackgroundLighting: Int At %VL 364000;
End_Var;

Task Main Autorun
    Digipot_Count := 0;
    Digipot_Limit_max := 17;
    Digipot_Limit_min := 0;


    Loop
      If Digipot_Button Then
                BackgroundLighting := 255;
           Else BackgroundLighting := Digipot_Count * 15;
      End_If
    End_Loop
End_Task;
```

Merker

Einleitung

Merker belegen ein Bit im Speicher und können den Wert TRUE oder FALSE annehmen.

Eigenschaften Anwendungsmerker

- Globale Variablen, die fest Adressen zugeordnet sind (%MX)
- Nicht flüchtig
- Anzahl: 256
- Merkernummern: 0 ... 255

Eigenschaften überlagerter Anwendungsmerker

- Globale Variablen, die fest Adressen zugeordnet sind (%MX)
- Nicht flüchtig
- Überlagert mit den Registern 1000000 ... 1000055
- Anzahl: 1.792
- Merkernummern: 256 ... 2047

Eigenschaften Spezialmerker

- Globale Variablen, die fest Adressen zugeordnet sind (%MX)
- Spezialmerker werden beim Start des Betriebssystems mit Standardwerten vorbesetzt
- Anzahl: 256
- Merkernummern: 2048 ... 2303

Speicherzugriff

- Durch JetSym
- Von Anzeige- und Bediengeräten
- Aus dem Anwendungsprogramm

JetSym-STX-Programm

Das folgende Programm setzt beim Drücken der Taste KEY_UP einen Merker und mit Drücken der Taste KEY_DOWN den Merker wieder zurück. Solange der Merker gesetzt ist, wird das Spezialregister 364000 (Hintergrundbeleuchtung) hochgezählt. Erst wenn der Merker wieder zurückgesetzt wird, dann wird das Hochzählen des Spezialregisters gestoppt.

```
Var
    Merker1:              Bool At %MX 1;
    Key_Up:               Bit  At %XL 361000.3;
    Key_Down:             Bit  At %XL 361000.2;
    Background_Light:     Int  At %VL 364000;
End_Var;

Task Main Autorun
    Merker1:= False;
    Loop
        If Key_Up Then
            Merker1 := True;
        ElseIf Key_Down Then
            Merker1 := False;
        End_IF;
```

```
            If Merker1 Then
                Inc(Background_Light);
                Delay(T#100ms);
            End_If;
        End_Loop;
    End_Task;
```

10.2 Bedienelemente und Zündung

Einleitung

Dieses Kapitel beschreibt die Programmierung der Bedienelemente und der Zündung und Ausschaltverzögerung des JVM-104.

Inhalt

Thema	Seite
Eingabetasten	173
Digipot	175
Zündung und Ausschaltverzögerung	177

Eingabetasten

Einleitung

Das Bediengerät JVM-104 verfügt über die vier Eingabetasten **[UP]**, **[DOWN]**, **[OK]** und **[ESC]**. Die Eingabetasten sind frei programmierbar.

Spezialregister

Im Register 361000 des JVM-104 ist ein bitkodiertes Abbild der Eingabetasten vorhanden. Sie können dieses Abbild zur Programmierung verwenden.

Folgende Register stehen für die Eingabetasten zur Verfügung:

Register	Beschreibung
361000	Bitkodiertes Abbild aller Eingabetasten
361000.0	**Eingabetaste [OK]** Bit 0 = 1: Taste **[OK]** ist gedrückt.
361000.1	**Eingabetaste [ESC]** Bit 1 = 1: Taste **[ESC]** ist gedrückt.
361000.2	**Eingabetaste [DOWN]** Bit 2 = 1: Taste **[DOWN]** ist gedrückt.
361000.3	**Eingabetaste [UP]** Bit 3 = 1: Taste **[UP]** ist gedrückt.

Softwareversionen

Das Beispielprogramm ist getestet mit den folgenden Softwareversionen:

- JetSym in der Version 5.2
- Bediengerät JVM-104 in der OS-Version 4.01

Sie finden weiterführende Informationen zum Thema STX-Programmierung in der Online-Hilfe von JetSym.

JetSym-STX-Programm

Beschreibung:

Das folgende Beispielprogramm fragt die Eingabetasten in einem Task permanent ab. Wenn Sie eine oder mehrere Tasten drücken, dann wird die Hintergrundbeleuchtung des Displays oder die Tastennachtbeleuchtung geändert.

```
Var
    btnKey_Ok:     Bit At %XL 361000.0;
    btnKey_Esc:    Bit At %XL 361000.1;
    btnKey_Dwn:    Bit At %XL 361000.2;
    btnKey_Up:     Bit At %XL 361000.3;

    dispBackLed :         Int At %VL 364000;
    dispButtonBackLed :   Int At %VL 364001;
End_Var;
```

```
Task Main Autorun
    Loop
        If btnKey_Up Then
            // Halbe Helligkeit der Hintergrundbeleuchtung
            dispBackLed := 127;
        End_If;
        If btnKey_Dwn Then
            // Volle Helligkeit der Hintergrundbeleuchtung
            dispBackLed := 255;
        End_If;
        If btnKey_Esc Then
            // Tastennachtbeleuchtung volle Helligkeit
            dispButtonBackLed := 255;
        End_If;
        If btnKey_Ok Then
            // Tastennachtbeleuchtung ausschalten
            dispButtonBackLed := 0;
        End_If;
        Delay(T#100ms);
    End_Loop;
End_Task;
```

Digipot

Einleitung

Bei dem JVM-104 ist ein Digipot mit einer Bestätigungstaste vorhanden, der eine komfortable Eingabemöglichkeit darstellt. An dieser Stelle sind die Spezialregister des Digipot beschrieben, sowie ein entsprechendes Beispielprogramm.

Register des Digipot

Es gibt folgende Spezialregister für den Digipot:

Register	Beschreibung
363000	Dieses Register enthält den aktuellen Zählwert. Durch Drehen des Digipot erhöht oder verringert sich der Zählwert. Dabei gilt: ■ **Digipot im Uhrzeigersinn drehen** = Register hochzählen ■ **Digipot gegen Uhrzeigersinn drehen** = Register abwärts zählen
363001	Bit 0: 0 = Bestätigungstaste nicht gedrückt Bit 0: 1 = Bestätigungstaste gedrückt
363002	Dieses Register gibt das untere Limit für den Zählwert des Digipot vor. Wenn Sie den Digipot gegen den Uhrzeigersinn weiterdrehen, bleibt das Register 363000 bei diesem Minimumwert stehen.
363003	Dieses Register gibt das obere Limit für den Zählwert des Digipot vor. Wenn Sie den Digipot im Uhrzeigersinn weiterdrehen, bleibt das Register 363000 bei diesem Maximumwert stehen.

Softwareversionen

Das Beispielprogramm ist getestet mit den folgenden Softwareversionen:

- JetSym in der Version 5.2
- Bediengerät JVM-104 in der OS-Version 4.01

Sie finden weiterführende Informationen zum Thema STX-Programmierung in der Online-Hilfe von JetSym.

JetSym-STX-Programm

Das folgende Beispielprogramm dimmt die Hintergrundbeleuchtung des JVM-104 über den Digipot. Dabei ist ein unteres und oberes Limit für den Digipot vorgegeben. Wenn Sie die Bestätigungstaste drücken, stellt sich die volle Hintergrundbeleuchtung ein.

```
Var
    Digipot_Count    :  Int At %VL 363000;
    Digipot_Limit_min:  Int At %VL 363002;
    Digipot_Limit_max:  Int At %VL 363003;
    Digipot_Button   :  Int At %VL 363001;
    BackgroundLighting: Int At %VL 364000;
End_Var;

Task Main Autorun
    Digipot_Count := 0;
    Digipot_Limit_max := 17;
    Digipot_Limit_min := 0;
```

```
    Loop
      If Digipot_Button Then
                BackgroundLighting := 255;
            Else BackgroundLighting := Digipot_Count * 15;
      End_If
    End_Loop
End_Task;
```

Zündung und Ausschaltverzögerung

Einleitung

Dieses Kapitel beschreibt die Zündung und die Funktion `Shutdown()`.

Spezialregister

Das Spezialregister 361100 des JVM-104 ist für die Abfrage der Zündung zuständig. Dabei gilt Folgendes:

Wenn ...	... dann ...
Bit 0 = 0,	... Zündung ist eingeschaltet und Spannung liegt an KL 15 Zündung (+) an.
Bit 0 = 1,	... Zündung ist aus. Keine Spannung liegt an KL 15 Zündung (+) an.

Default-Zündungsfunktion

Das Bediengerät hat im Zusammenhang mit der Zündung folgende Default-Einstellungen:

Wenn ...	... und ...	... dann ...
... das Bediengerät mit Spannung versorgt wird ...	... die Zündung aus ist,	... bootet das Bediengerät nicht.
... das Bediengerät mit Spannung versorgt wird ...	... die Zündung eingeschaltet ist,	... bootet das Bediengerät.
... das Bediengerät eingeschaltet ist ...	... die Zündung ausgeschaltet wird (nicht die Spannungsversorgung),	... dann bleibt das Bediengerät eingeschaltet.

Optionen Funktion Shutdown()

Abweichend von der Default-Zündungsfunktion bietet die Funktion `Shutdown()` folgende Optionen:

- Das Bediengerät kann definiert heruntergefahren werden.
- Ein Neustart des Bediengeräts kann durchgeführt werden.

Funktionsdeklaration

```
Function Shutdown(Reboot:Bool) :Bool;
```

Funktionsparameter

Die Funktion `Shutdown()` hat den folgenden Parameter.

Parameter	Beschreibung	Wert
Reboot	Neustart des Systems: Abschalten des Systems:	True False

Rückgabewert

Die Funktion übergibt die folgenden Rückgabewerte an das übergeordnete Programm.

Rückgabewert	
0	ok
-1	Zündung ist noch eingeschaltet

Hinweis

Wenn die Zündung noch eingeschaltet ist, schaltet das Gerät nicht ab. Ein Neustart wird allerdings immer durchgeführt. Der Neustart ist unabhängig von der Zündung.

Softwareversionen

Das Beispielprogramm ist getestet mit den folgenden Softwareversionen:

- JetSym in der Version 5.2
- Bediengerät JVM-104 in der OS-Version 4.01

Sie finden weiterführende Informationen zum Thema STX-Programmierung in der Online-Hilfe von JetSym.

JetSym-STX-Programm

Wenn Sie die Zündung des Fahrzeugs ausschalten, führt das Beispielprogramm die Funktion `Shutdown()` nach 3 s aus. Der Parameter **Reboot** der Funktion `Shutdown()` hat den Wert **false**. Das bedeutet, dass das Gerät abschaltet.

```
Var
    Ignition: Int At %VL 361100;
End_Var;

Task Ign Autorun
    Loop
     When Ignition Continue;
            Delay(3000);
            Shutdown(False);
    End_Loop;
End_Task;
```

10.3 Multifunktionseingänge/-ausgänge

Einleitung

Dieses Kapitel beschreibt die Programmierung der Multifunktionseingänge und Multifunktionsausgänge des JVM-104. Das Bediengerät hat Spezialregister für folgende Funktionen:

- Multifunktionseingänge programmieren
- Multifunktionsausgänge programmieren
- Lesen der Hard- und Softwareversion für I/O-Funktionen

Inhalt

Thema	Seite
Multifunktionseingänge, Status und Kommandos	180
Multifunktionseingänge, analoge Funktionen	182
Multifunktionseingänge, digitale Funktionen	185
Multifunktionsausgänge, Status und Kommandos	188
Multifunktionsausgänge, analoge Funktionen	190
Multifunktionsausgänge, digitale Funktionen	192
Multifunktionsausgänge PA3 und PA4 als H-Brücke	194

Multifunktionseingänge, Status und Kommandos

Einleitung

Die Multifunktionseingänge haben folgende Funktionen:

- Spannungsmessung
- Strommessung
- Digitaler Active-High-Eingang
- Frequenzmessung
- Zählmodus

Platzhalter x

Der Platzhalter x ist zur vereinfachten Darstellung. Die Tabelle zeigt, wie Sie die einzelnen Eingänge durch Platzhalter adressieren. Im Folgenden gilt:

Platzhalter	Wert	Beschreibung
x	1	Multifunktionseingang MFQE1
x	2	Multifunktionseingang MFQE2

Beispiel

x = 2 für Multifunktionseingang 2. Die Übertemperatur lesen Sie aus R 601020000.7.

Registerübersicht der Multifunktionseingänge

Die Multifunktionseingänge MFQE1 und MFQE2 haben jeweils die folgenden Register:

Register	Beschreibung
6010x0000	Status (bitkodiert)
6010x0001	Kommando (bitkodiert)
6010x0002	Analogwert in µA/mV
6010x0003	Frequenz- [Torzeit] / Perioden- [µs] / Zählerwert [Inkremente]
6010x0004	Digitaler Wert des Active-High-Eingang
6010x0005	Torzeit in ms bei der Frequenzmessung

R6010x0000

Statusregister des Multifunktionseingangs MFQEx

Das R 6010x0000 zeigt den Status des Multifunktionseingangs MFQEx an.

Bedeutung der Bits

Bit 0 **Überstrom**

1 = Im Modus Strommessung ausgelöst durch einen Strom > 2500 mA.

Bit 7	**Übertemperatur**
	1 = Eine Übertemperatur liegt vor.
Bit 14	**Multifunktionseingang MFQEx ist ...**
	0 = Multifunktionseingang MFQEx ist eingeschaltet
	1 = Multifunktionsausgang MFQEx ist ausgeschaltet

Registereigenschaften	
Zugriff	Lesen
Rücksetzen	Kommando Multifunktionseingang x einschalten

R 6010x0001

Kommandoregister Multifunktionseingang MFQEx, bitkodiert

Mit dem Kommandoregister lassen sich verschiedene Betriebsarten einstellen und den Eingang MFQEx konfigurieren.

Kommandos MFQEx, bitkodiert	
Bit 7 ... 4:	
0000 =	Frequenzmessung mit Torzeit
0011 =	Zählmodus
0100 =	Periodenzeitmessung, 10 s Timeout
0101 =	Periodenzeitmessung, 1 s Timeout
0110 =	Periodenzeitmessung, 100 ms Timeout
Bit 9:	
0 =	Spannungsmessung
1 =	Strommessung
Bit 12:	
0 =	Pull-up-Widerstand einschalten
1 =	Pull-up-Widerstand ausschalten

Multifunktionseingänge, analoge Funktionen

Einleitung

Das Bediengerät JVM-104 verfügt bei den Multifunktionseingängen über folgende analoge Funktionen:

- Spannungsmessung
- Strommessung

Platzhalter x

Der Platzhalter x ist zur vereinfachten Darstellung. Die Tabelle zeigt, wie Sie die einzelnen Eingänge durch Platzhalter adressieren. Im Folgenden gilt:

Platzhalter	Wert	Beschreibung
x	1	Multifunktionseingang MFQE1
x	2	Multifunktionseingang MFQE2

Beispiel

x = 2 für Multifunktionseingang 2. Die Übertemperatur lesen Sie aus R 601020000.7.

R 6010x0002

Analogwert Multifunktionseingang MFQEx

Der Registerwert ergibt sich aus der Messung des Analogwerts am Multifunktionseingang x.

Registereigenschaften		
Werte	Konfiguration Spannungsmessung	0 mV ... 15.000 mV
	Konfiguration Strommessung	0 µA ... 20.000 µA

Registereigenschaften	
Zugriff	Lesen

Spannungsmessung oder Strommessung aktivieren

Das Kommandoregister für den Multifunktionseingang MFQEx ist bitkodiert. Indem Sie mehrere Bits setzen, geben Sie mehrere Kommandos gleichzeitig.

Beispiel:

Kommando 0x1200 schaltet den Eingang ein, stellt den Modus auf Strommessung und deaktiviert den Pull-up-Widerstand des Eingangs.

Die Spannungs- oder Strommessung für den Multifunktionseingang MFQEx aktivieren Sie wie folgt:

Schritt	Vorgehen	
1	Schalten Sie den Eingang ein: Schreiben Sie den Wert R 6010x0001 := 0;	
2	Sie können einen Pull-up-Widerstand am Eingang einschalten.	
	Wenn ...	**... dann ...**
	... Sie den Pull-up-Widerstand einschalten wollen,	... löschen Sie Bit 12: BitClear(R 6010x0001, 12);
	... Sie den Pull-up-Widerstand ausschalten wollen,	... setzen Sie Bit 12: BitSet(R 6010x0001, 12);
3	Wählen Sie die Art der Messung:	
	Wenn ...	**... dann ...**
	... Sie eine Spannung messen wollen,	... löschen Sie Bit 9: BitClear(R 6010x0001, 9);
	... Sie Strom messen wollen,	... setzen Sie Bit 9: BitSet(R 6010x0001, 9);

Hinweis

Wenn Sie das Bediengerät einschalten, ist der Multifunktionseingang MFQEx eingeschaltet.

Softwareversionen

Das Beispielprogramm ist getestet mit den folgenden Softwareversionen:

- JetSym in der Version 5.2
- Bediengerät JVM-104 in der OS-Version 4.01

Sie finden weiterführende Informationen zum Thema STX-Programmierung in der Online-Hilfe von JetSym.

JetSym-STX-Programm

Das folgende Beispiel beschreibt, wie Sie den Multifunktionseingang MFQE1 für eine Spannungsmessung und den Multifunktionseingang MFQE2 für eine Strommessung nutzen.

```
Var
    // MFQE1: Eingang 1
    MFQE1_Command:     Int At %VL 601010001;
    MFQE1_Value:       Int At %VL 601010002;
    // MFQE2: Eingang 2
    MFQE2_Command:     Int At %VL 601020001;
    MFQE2_Value:       Int At %VL 601020002;
End_Var;

Task Main Autorun
// Eingang MFQE1 einschalten
MFQE1_Command := 0;
// Eingang MFQE1 in Messart Spannungsmessung
BitClear(MFQE1_Command, 9);
```

```
// Eingang MFQE2 einschalten
MFQE2_Command := 0;
// Eingang MFQE2 in Messart Strommessung
BitSet(MFQE2_Command, 9);

Loop
    Trace('Spannungswert an MFQE1 ' + IntToStr(MFQE1_Value) + '
mV.$n');
    Trace('Stromwert an MFQE2 ' + IntToStr(MFQE2_Value) + ' µA.$n');
    Delay(T#100ms);
End_Loop;
End_Task;
```

Multifunktionseingänge, digitale Funktionen

Einleitung

Das Bediengerät JVM-104 verfügt bei den Multifunktionseingängen über folgende digitale Funktionen:

- Digitaler Eingang
- Frequenzmessung
- Zählmodus

Platzhalter x

Der Platzhalter x ist zur vereinfachten Darstellung. Die Tabelle zeigt, wie Sie die einzelnen Eingänge durch Platzhalter adressieren. Im Folgenden gilt:

Platzhalter	Wert	Beschreibung
x	1	Multifunktionseingang MFQE1
x	2	Multifunktionseingang MFQE2

Beispiel

x = 2 für Multifunktionseingang 2. Die Übertemperatur lesen Sie aus R 601020000.7.

R 6010x0003

Frequenzwert Multifunktionseingang MFQEx

Je nach Konfiguration zeigt dieses Register die Frequenz, Periode oder einen Zählwert des Multifunktionseingangs x an.

Registereigenschaften		
Werte	Konfiguration Frequenzmessung	0,1 Hz ... 10 kHz
	Konfiguration Periodenzeitmessung	10 s ... 100 ms
	Konfiguration Inkremente zählen	32 Bit

Registereigenschaften	
Zugriff	Lesen

R 6010x0004

Digitalwert am Multifunktionseingang MFQEx

Der Registerwert ergibt sich aus der Messung des Digitalwerts am Multifunktionseingang x.

Der digitale Eingang ist ein Active-High-Eingang.

Registereigenschaften		
Werte	Eingang x < 1,0 V	Low-Pegel
	Eingang x > 2,0 V	High-Pegel

R 6010x0005

Torzeit für Frequenzmessung am Eingang MFQEx

Mit dem Registerwert lässt sich für die Frequenzmessung eine Torzeit einstellen.

Wenn Sie Frequenzen messen, stellen Sie eine Torzeit ungleich Null ein.

Registereigenschaften	
Werte	100 ms ... 10 s
Wert nach Reset	1.000 (1 s) 0: Abgeschaltet

Frequenz-/ Periodenzeitmessung oder Zählerfunktion aktivieren

Das Kommandoregister für den Multifunktionseingang MFQEx ist bitkodiert. Indem Sie mehrere Bits setzen, können Sie mehrere Kommandos gleichzeitig geben.

Die Frequenzmessung, Periodenzeitmessung oder Zählerfunktion für den Multifunktionseingang MFQEx aktivieren Sie folgendermaßen:

Schritt	**Vorgehen**	
1	Schalten Sie den Eingang ein: Schreiben Sie den Wert R 6010x0001 := 0;	
	Wenn ...	**... dann ...**
	... Sie eine Frequenz mit Torzeit messen wollen,	... schreiben Sie als Torzeit einen Wert ungleich Null in das R 6010x0005. Aktiveren Sie die Frequenzmessung, indem Sie den Wert 0x0040 in das R 6010x0001 schreiben.
	... Sie Inkremente zählen wollen,	... schreiben Sie den Wert 0x0030 in das R 601x0001.
	... Sie eine Periodenzeit messen wollen,	... schreiben Sie den Wert 0x0040 in R 601x0001.
⇨	Das Ergebnis steht in R 6010x0003.	

Softwareversionen

Das Beispielprogramm ist getestet mit den folgenden Softwareversionen:

- JetSym in der Version 5.2
- Bediengerät JVM-104 in der OS-Version 4.01

Sie finden weiterführende Informationen zum Thema STX-Programmierung in der Online-Hilfe von JetSym.

JetSym-STX-Programm

Das folgende Beispiel beschreibt, wie Sie den Multifunktionseingang MFQE1 für eine Messung von Inkrementen und MFQE2 für eine Frequenzmessung mit Torzeit nutzen.

```
Var
    // MFQE1: Eingang 1
    MFQE1_Command:    Int At %VL 601010001;
    MFQE1_Freq:       Int At %VL 601010003;

    // MFQE2: Eingang 2
    MFQE2_Command:    Int At %VL 601020001;
    MFQE2_Freq:       Int At %VL 601020003;
    MFQE2_GateTime:   Int At %VL 601020005;
End_Var;

Task Main Autorun
// Messung von Inkrementen an MFQE1 einschalten:
MFQE1_Command := 0x0030;
// Messung einer Frequenz an MFQE2
MFQE2_Command := 0x0040;
// Torzeit von 1 s an MFQE2 einstellen:
MFQE2_GateTime := 1000;

Loop
    // Werte ausgeben
    Trace('Inkremente: ' + InttoStr(MFQE1_Freq) + '$n');
    Trace('Frequenz in Hertz: : ' + InttoStr(MFQE2_Freq) + '$n');
    Delay(T#300ms);
End_Loop;
End_Task;
```

Multifunktionsausgänge, Status und Kommandos

Einleitung

Die Multifunktionsausgänge haben folgende Funktionen:

- Strommessung
- Digitaler Ausgang, einstellbar als High-Side oder Low-Side-Ausgang
- PWM-Ausgabe
- Betrieb als H-Brücke

Platzhalter x

Der Platzhalter x ist zur vereinfachten Darstellung. Die Tabelle zeigt, wie Sie die einzelnen Ausgänge durch Platzhalter adressieren. Im folgenden Kapitel gilt:

Platzhalter	Wert	Beschreibung
x	5	Multifunktionsausgang PA3
x	6	Multifunktionsausgang PA4

Beispiel:

x = 5 für Multifunktionsausgang PA3. Den digitalen Ausgang von PA3 schalten Sie ein über das R 601050001 mit dem Kommando 0x0008.

Registerübersicht der Multifunktionsausgänge

Die Multifunktionsausgänge PA3 und PA4 haben jeweils die folgenden Register:

Register	Beschreibung
6010x0000	Status (bitkodiert)
6010x0001	Kommando (bitkodiert)
6010x0002	PWM-Tastverhältnis in 0,1 %
6010x0005	Gemessener Strom des Ausgangs PAx in mA
6010x0006	PWM-Frequenz in Hz
6010x0007	Einstellbare Grenze für Überstrom in mA

R6010x0000

Statusregister der Multifunktionsausgänge PA3 und PA4

Im R 6010x0000 signalisiert das Bediengerät Statusmeldungen der Multifunktionsausgänge PA3 und PA4.

Bedeutung der Bits	
Bit 0	**Überstrom H-Side**
	1 = Überstromerkennung hat ausgelöst.

Bit 3	**Kurzschluss L-Side**
	1 = Überstromerkennung hat ausgelöst.
Bit 6	**Kabelbruch (open load)**
	1 = Ein Kabelbruch liegt vor.
Bit 7	**Übertemperatur**
	1 = Eine Übertemperatur liegt vor.
Bit 14	**Multifunktionsausgang ist ...**
	0 = Multifunktionsausgang PAx ist eingeschaltet
	1 = Multifunktionsausgang PAx ist ausgeschaltet
Bit 15	**Hardwareschutz hat ausgelöst**
	1 = Hardwareschutz hat durch Überstrom/Kurzschluss ausgelöst.

Registereigenschaften	
Zugriff	Lesen
Rücksetzen	Kommando Multifunktionsausgang x einschalten
Wert nach Reset	0x00000040 (falls nichts angeschlossen ist)

R 6010x0001

Kommandoregister Multifunktionsausgang PA3 und PA4, bitkodiert

Mit dem Kommandoregister lassen sich verschiedene Betriebsarten einstellen und den jeweiligen Ausgang PAx konfigurieren.

Kommandos PAx, bitkodiert	
Bit 1 ... 0	**Funktion des Ausgangs**
00 =	Ausgang PAx als digitaler Ausgang einschalten
01 =	Ausgang PAx als PWM-Ausgang einschalten
Bit 3 ... 2:	
00 =	Ausgang PAx als Eingang
01 =	Ausgang PAx als Low-Side-Ausgang
10 =	Ausgang PAx als High-Side-Ausgang
11 =	Ausgang PAx als Low-High-Side-Ausgang
Bit 15:	
0 =	Ausgang einschalten
1 =	Ausgang ausschalten

Multifunktionsausgänge, analoge Funktionen

Einleitung

Das Bediengerät JVM-104 verfügt bei den Multifunktionsausgängen über folgende analoge Funktionen:

- Strommessung
- Grenze des Überstroms einstellen

Platzhalter x

Der Platzhalter x ist zur vereinfachten Darstellung. Die Tabelle zeigt, wie Sie die einzelnen Ausgänge durch Platzhalter adressieren. Im folgenden Kapitel gilt:

Platzhalter	Wert	Beschreibung
x	5	Multifunktionsausgang PA3
x	6	Multifunktionsausgang PA4

Beispiel:
x = 5 für Multifunktionsausgang PA3. Den digitalen Ausgang von PA3 schalten Sie ein über das R 601050001 mit dem Kommando 0x0008.

6010x0005

Gemessener Strom PAx in mA

Das Register zeigt den gemessenen Strom in mA des Ausgangs PAx an. Die Strommessung gilt nur, wenn der PAx als Ausgang konfiguriert ist.

Registereigenschaften	
Werte	0 ... 2.300
Einheit	Milliampere

Registereigenschaften	
Zugriff	Lesen

6010x0007

Einstellbare Grenze des Überstroms für den Ausgang PAx

Mit dem Register stellen Sie eine Grenze für Überstrom ein. Wenn der Strom des Ausgang PAx diese Grenze überschreitet, schaltet der Ausgang PAx ab und das Bit 0 in R 6010x0000 ist gesetzt.

Registereigenschaften	
Werte	0 ... 2.500
Einheit	Milliampere
Wert nach Reset	0, das bedeutet einen maximalen Strom von 2,5 A

Digitalen Ausgang einschalten und Strom messen

Das Kommandoregister für den Multifunktionsausgang PAx ist bitkodiert. Indem Sie mehrere Bits setzen, geben Sie mehrere Kommandos gleichzeitig.

Beispiel:

Kommando 0x0004 schaltet den Ausgang als digitalen Low-Side-Ausgang ein.

Wie folgt messen Sie den Strom des Ausgangs und setzen die Grenze des erlaubten Überstroms:

Schritt	Vorgehen
1	Stellen Sie eine Grenze für den Überstrom ein: R 6010x0007 := 1.000;
2	Schalten Sie den Ausgang als digitalen Ausgang und z. B. als Low-Side-Ausgang ein: R 6010x0001 := 0x0004;
3	Lesen Sie aus R 6010x0005 den Strom in mA aus.

Softwareversionen

Das Beispielprogramm ist getestet mit den folgenden Softwareversionen:

- JetSym in der Version 5.2
- Bediengerät JVM-104 in der OS-Version 4.01

Sie finden weiterführende Informationen zum Thema STX-Programmierung in der Online-Hilfe von JetSym.

JetSym-STX-Programm

Das folgende Beispiel beschreibt, wie Sie zyklisch den Strom des digitalen Ausgangs PA3 in mA messen.

```
Var
    // PA3: Ausgang 3
    PA3_Command:            Int At %VL 601050001;
    PA3_Current_Value:      Int At %VL 601050005;
End_Var;

Task Main Autorun
// Ausgang PA3 als Low-Side einschalten
PA3_Command := 0x0004;

Loop
    Trace('Stromwert an PA1 ' + IntToStr(PA3_Current_Value) + '
µA.$n');
    Delay(T#100ms);
End_Loop;
End_Task;
```

Multifunktionsausgänge, digitale Funktionen

Einleitung

Das Bediengerät JVM-104 verfügt bei den Multifunktionsausgängen über folgende digitale Funktionen:

- Digitaler Eingang
- Digitaler Ausgang
- PWM-Ausgang

Platzhalter x

Der Platzhalter x ist zur vereinfachten Darstellung. Die Tabelle zeigt, wie Sie die einzelnen Ausgänge durch Platzhalter adressieren. Im folgenden Kapitel gilt:

Platzhalter	Wert	Beschreibung
x	5	Multifunktionsausgang PA3
x	6	Multifunktionsausgang PA4

Beispiel:

x = 5 für Multifunktionsausgang PA3. Den digitalen Ausgang von PA3 schalten Sie ein über das R 601050001 mit dem Kommando 0x0008.

6010x0002

PWM-Tastverhältnis für Ausgang PAx

Über dieses Register können Sie bei eingeschaltetem PWM-Ausgang den Wert des PWM-Tastverhältnisses einstellen.

Registereigenschaften	
Werte als Eingabe	0 ... 1000
Einheit	0,1 %

6010x0006

PWM-Frequenz in Hz für Ausgang PAx

Mit dem Register stellen Sie die PWM-Frequenz ein, die der Ausgang PAx im PWM-Modus ausgibt.

Registereigenschaften	
Werte	100 ... 1.000
Einheit	Hertz

Digitalen PWM-Ausgang einschalten

Das Kommandoregister für den Multifunktionsausgang PAx ist bitkodiert. Indem Sie mehrere Bits setzen, geben Sie mehrere Kommandos gleichzeitig.

Beispiel:

Kommando 0x0005 schaltet den PWM-Ausgang als digitalen Low-Side-Ausgang ein.

Wie folgt schalten Sie den PWM-Ausgang PAx ein. Zuvor stellen Sie das PWM-Tastverhältnis und die PWM-Frequenz ein:

Schritt	Vorgehen
1	Stellen Sie das PWM-Tastverhältnis auf 60 %: R 6010x0002 := 600;
2	Stellen Sie die PWM-Frequenz auf 300 Hz: R 6010x0006 := 300;
3	Schalten Sie den PWM-Ausgang PAx als z. B. Low-Side-Ausgang ein: R 6010x0001 := 0x0005;

Softwareversionen

Das Beispielprogramm ist getestet mit den folgenden Softwareversionen:

- JetSym in der Version 5.2
- Bediengerät JVM-104 in der OS-Version 4.01

Sie finden weiterführende Informationen zum Thema STX-Programmierung in der Online-Hilfe von JetSym.

JetSym-STX-Programm

Das folgende Beispiel beschreibt, wie Sie den Multifunktionsausgang PA4 als PWM-Ausgang einschalten.

```
Var
    // PA4: PWM-Ausgang 4
    PA4_Command:            Int At %VL 601060001;
    PA4_PWM_Duty_Cycle:     Int At %VL 601060002;
    PA4_PWM_Freq:           Int At %VL 601060006;
End_Var;

Task Main Autorun
// Ausgang PA4 als Low-Side und PWM-Ausgang einschalten
PA4_Command := 0x0005;

Loop
//
//
    Delay(T#100ms);
End_Loop;
End_Task;
```

Ausgang PAx als digitalen Eingang einschalten

Schreiben Sie das Kommando 0x0000 in das R 6010x0001. Der Ausgang PAx übernimmt jetzt die Funktion eines digitalen Eingangs. Das Bit 0 hat in diesem Fall keine Bedeutung.

Multifunktionsausgänge PA3 und PA4 als H-Brücke

Einleitung

Bei dem Bediengerät JVM-104 können Sie die Multifunktionsausgänge PA3 und PA4 als H-Brücke ansteuern.

Programmierung

Um die H-Brücke anzusteuern, müssen Sie folgende Schritte programmieren:

Rechtslauf:

Schritt	Vorgehen
1	Schalten Sie den Low-Side-Ausgang von PA3 ein. R 601050001 := 0x0005;
2	Stellen Sie das PWM-Tastverhältnis von 100 % an PA3 ein: R 601050002 := 1000;
3	Schalten Sie den High-Side-Ausgang von PA4 ein: R 601060001 := 0x0009;
4	Stellen Sie das gewünschte PWM-Tastverhältnis an PA4 ein, z. B. 50 %: R 601060002 := 500;
⇨	Die H-Brücke ist im Rechtslauf mit 50 % Tastverhältnis konfiguriert.

Linkslauf:

Schritt	Vorgehen
1	Schalten Sie den Low-Side-Ausgang von PA4 ein. R 601060001 := 0x0005;
2	Stellen Sie das PWM-Tastverhältnis von 100 % an PA4 ein: R 601060002 := 1000;
3	Schalten Sie den High-Side-Ausgang von PA3 ein: R 601050001 := 0x0009;
4	Stellen Sie das gewünschte PWM-Tastverhältnis an PA3 ein, z. B. 40 %: R 601050002 := 400;
⇨	Die H-Brücke ist im Linkslauf mit 40 % Tastverhältnis konfiguriert.

Softwareversionen

Das Beispielprogramm ist getestet mit den folgenden Softwareversionen:

- JetSym in der Version 5.2
- Bediengerät JVM-104 in der OS-Version 4.01

Sie finden weiterführende Informationen zum Thema STX-Programmierung in der Online-Hilfe von JetSym.

JetSym-STX-Programm:

```
Task H_Bruecke Autorun

Const
LS_ON = 0x0005;
HS_ON = 0x0009;
End_Const;

Var
    PA3_Command:           Int At %VL 601050001;
    PA3_PWM_Duty_Cycle:    Int At %VL 601050002;
    PA4_Command:           Int At %VL 601060001;
    PA4_PWM_Duty_Cycle:    Int At %VL 601060002;
End_Var;

// Rechtslauf
// Low-Side-Ausgang PA3 an:
PA3_Command := LS_ON;
// PA3 PWM-Tastverhältnis 100 %:
PA3_PWM_Duty_Cycle := 1000;
// PA4 High-Side-Ausgang an:
PA4_Command := HS_ON;
// Gewünschtes PWM-Tastverhältnis, z. B. 50 %
PA4_PWM_Duty_Cycle := 500;
// Jetzt läuft der Motor nach rechts

// Motor stoppen
PA4_PWM_Duty_Cycle := 0;
Delay(T#2s);

// Linkslauf
// Low-Side-Ausgang PA4 an:
PA4_Command := LS_ON;
// PA4 PWM-Tastverhältnis 100 %:
PA4_DutyCycle := 1000;
// PA3 High-Side-Ausgang an:
PA3_Command := HS_ON;
// Gewünschtes PWM-Tastverhältnis, z. B. 40 %
PA3_DutyCycle := 400;
// Jetzt läuft der Motor nach links
End_Task;
```

Verwandte Themen

- **Anschluss der H-Brücke** (siehe Seite 38)

10.4 Laufzeitregister

Einleitung

Das Gerät JVM-104 hat mehrere Register, die das Betriebssystem in festen Zeitabständen hochzählt.

Verwendung

Mit Hilfe dieser Register lassen sich auf einfache Weise im Anwendungsprogramm Zeitmessungen durchführen.

Inhalt

Thema	Seite
Beschreibung der Laufzeitregister	197
Beispielprogramm Laufzeitregister	199

Beschreibung der Laufzeitregister

Registerübersicht

Das Gerät hat die folgenden Laufzeitregister:

Register	Beschreibung
R 201000	Anwendungszeitbasis in Millisekunden
R 201001	Anwendungszeitbasis in Sekunden
R 201002	Anwendungszeitbasis in R 201003 * 10 ms
R 201003	Anwendungszeitbasis-Einheiten für R 201002
R 201004	Systemzeitbasis in Millisekunden
R 201005	Systemzeitbasis in Mikrosekunden

R 201000

Anwendungszeitbasis in Millisekunden

Dieses Register wird jede Millisekunde um eins inkrementiert.

Registereigenschaften	
Werte	-2.147.483.648 ... 2.147.483.647 (überlaufend)

R 201001

Anwendungszeitbasis in Sekunden

Dieses Register wird jede Sekunde um eins inkrementiert.

Registereigenschaften	
Werte	-2.147.483.648 ... 2.147.483.647 (überlaufend)

R 201002

Anwendungszeitbasis in Anwendungszeitbasis-Einheiten

Dieses Register wird alle [R 201003] * 10 ms um eins inkrementiert. Mit dem Reset-Wert 10 in Register 201003 wird dieses Register alle 100 ms inkrementiert.

Registereigenschaften	
Werte	-2.147.483.648 ... 2.147.483.647 (überlaufend)

R 201003 **Anwendungszeitbasis-Einheiten für R 201002**

Dieses Register enthält den Multiplikator für das Laufzeitregister R 201002.

Registereigenschaften	
Werte	1 ... 2.147.483.647 (* 10 ms)
Wert nach Reset	10 (--> 100 ms)
Wirksamkeit	Nach min. 10 ms

R 201004 **Systemzeitbasis in Millisekunden**

Dieses Register wird jede Millisekunde um eins inkrementiert.

Registereigenschaften	
Werte	-2.147.483.648 ... 2.147.483.647 (überlaufend)
Zugriff	Lesen

R 201005 **Systemzeitbasis in Mikrosekunden**

Dieses Register wird jede Mikrosekunde um eins inkrementiert.

Registereigenschaften	
Werte	-2.147.483.648 ... 2.147.483.647 (überlaufend)
Zugriff	Lesen

Beispielprogramm Laufzeitregister

Aufgabe

Messen Sie, wie lange das Abspeichern von Variablenwerten in einer Datei dauert.

Lösung

Bevor Sie die Werte abspeichern, setzen Sie das Register 201000 auf 0.

Wenn die Werte abgespeichert sind, lesen Sie aus diesem Register die benötigte Zeit in Millisekunden.

Softwareversionen

Das Beispielprogramm ist getestet mit den folgenden Softwareversionen:

- JetSym in der Version 5.2
- Bediengerät JVM-104 in der OS-Version 4.01

Sie finden weiterführende Informationen zum Thema STX-Programmierung in der Online-Hilfe von JetSym.

JetSym-STX-Programm

```
Var
    DataArray:    Array[2000] Of Int;
    File1:        File;
    WriteTime:    Int;
    WriteIt:      Bool;

    MilliSec:     Int At %VL 201000;
End_Var;

Task WriteToFile Autorun
    Loop
        // Startmerker zurücksetzen
        WriteIt := False;
        // Warten bis Anwender den Startmerker gesetzt hat
        When WriteIt Continue;

        // Datei im Schreibmodus öffnen
        // Wenn keine Datei vorhanden ist, dann wird eine neue Datei
        // erstellt
        If FileOpen(File1, 'Test.dat', fWrite) Then

            // Anwendungszeitbasis-Register auf Null setzen
            MilliSec := 0;

            // Datenbereich in die Datei schreiben
            FileWrite(File1, DataArray, SizeOf(DataArray));

            // Laufzeit erfassen
            WriteTime := MilliSec;
            FileClose(File1);

            // Laufzeit anzeigen
            Trace(StrFormat('Time : %d [ms]$n', WriteTime));
```

```
            Else
                // Fehlermeldung anzeigen
                Trace('Unable to open file!$n');
            End_If;
        End_Loop;
    End_Task;
```

10.5 Überwachung der Schnittstellenaktivität

Einleitung

Um die innerhalb des Anwendungsprogramms verwendeten Variablen von außen zugänglich zu machen, sind im Bediengerät mehrere Ethernet-Server integriert. Diese Server unterstützen unterschiedliche Protokolle auf verschiedenen Schnittstellen. Die Server benötigen keine Programmierung im Anwendungsprogramm, sondern bearbeiten Anfragen von externen Clients selbstständig.

Dieses Kapitel erläutert eine Möglichkeit, im Anwendungsprogramm festzustellen, ob über diese Schnittstellen mit den Servern kommuniziert wird.

Überwachte Schnittstellenaktivitäten

Die folgenden Schnittstellenaktivitäten sind überwachbar:

- JetIP-Server auf der Ethernet-Schnittstelle
- STX-Debug-Server auf der Ethernet-Schnittstelle

Verwendung

Die Überwachung der Schnittstellenaktivität findet z. B. bei folgenden Szenarien Verwendung:

- Bei Anlagen, die für den sicheren Betrieb eine Prozessvisualisierung benötigen. Wenn die Kommunikation ausfällt, sind dann diese Anlagen in einen sicheren Zustand überführbar.
- Wenn der Servicetechniker ein Bediengerät anschließt, werden vom Anwendungsprogramm automatisch erweiterte Statusinformationen ausgegeben.

Inhalt

Thema	Seite
Funktionsweise	202
Programmierung	204

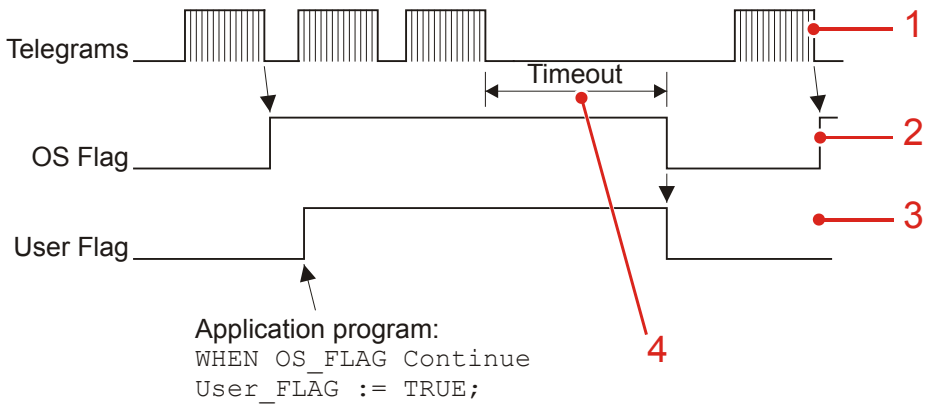
Funktionsweise

Einleitung

Mit Hilfe von zwei Spezialmerkern und einem Spezialregister pro Schnittstelle überwacht das Anwendungsprogramm die Aktivität eines Clients, der mit einem Server des Geräts JVM-104 kommuniziert.

Übersicht

Diese Abbildung zeigt den Zusammenhang zwischen der Aktivität der Schnittstelle, den beiden Spezialmerkern und dem Spezialregister:



Nummer	Element	Beschreibung
1	Telegrams	Client stellt Anfragen an den Server
2	OS-Flag	Betriebssystemmerker, den das Gerät JVM-104 setzt, wenn eine Anfrage empfangen wurde.
3	User-Flag	Anwendungsmerker, den Sie im Anwendungsprogramm setzen müssen, sobald das Gerät den Betriebssystemmerker setzt; hierüber stellen Sie fest, dass die Verbindung kurzzeitig unterbrochen war. Auch wenn das Gerät den Betriebssystemmerker sehr schnell wieder setzt.
4	Timeout	Zeit der Inaktivität, nach der das Betriebssystem beide Spezialmerker zurücksetzt; die Zeit ist in einem Spezialregister einstellbar.

Beschreibung

Die Überwachung der Schnittstellenaktivität ist wie folgend beschrieben:

Stufe	Beschreibung
1	Beschreiben Sie im Anwendungsprogramm das Timeout-Register mit dem gewünschten Wert. Der Überwachungsmodus ist somit auch eingeschaltet.
2	Nach dem Empfang des nächsten Telegramms wird von dem Gerät JVM-104 der entsprechende Betriebssystemmerker gesetzt.
3	Das Anwendungsprogramm setzt bei gesetztem Betriebssystemmerker auch den entsprechenden Anwendungsmerker.
4	Bei jedem weiteren Telegramm startet der Timeout neu.

Stufe	Beschreibung
5	Wenn das Gerät keine weiteren Telegramme mehr empfängt, dann werden nach Ablauf der Timeout-Zeit beide Spezialmerker von dem Gerät zurückgesetzt.
6	Das Anwendungsprogramm erkennt, dass das Gerät die Spezialmerker zurückgesetzt hat und leitet Maßnahmen ein.
7	Wenn das Gerät wieder Telegramme empfangen hat, dann wird der Betriebssystemmerker von dem Gerät wieder gesetzt; der Anwendungsmerker bleibt jedoch zurückgesetzt.

Programmierung

Register-/ Merkerübersicht

Zur Schnittstellenüberwachung verwendet das Gerät die folgenden Register und Merker:

Timeout-Register

Register	Schnittstelle	Verwendung
R 203000	JetIP (Ethernet)	▪ Visualisierung ▪ Vernetzung
R 203005	STX-Debug (Ethernet)	▪ JetSym über Ethernet

Spezialmerker

Merker	Schnittstelle	Verwendung
F 2088	JetIP (Ethernet)	Betriebssystemmerker
F 2089		Anwendungsmerker
F 2098	STX-Debug (Ethernet)	Betriebssystemmerker
F 2099		Anwendungsmerker

R 203000

Timeout bei JetIP (Ethernet)

Dieses Register enthält die Timeout-Zeit für den JetIP-Server (Ethernet) in Millisekunden.

Registereigenschaften	
Werte	0 ... 2.147.483.647 [ms]
Wert nach Reset	0 (Überwachung abgeschaltet)

R 203005

Timeout bei STX-Debug (Ethernet)

Dieses Register enthält die Timeout-Zeit für den STX-Debug-Server (Ethernet) in Millisekunden.

Registereigenschaften	
Werte	0 ... 2.147.483.647 [ms]
Wert nach Reset	0 (Überwachung abgeschaltet)

Einschalten der Überwachung

Um die Aktivitätsüberwachung für eine Schnittstelle einzuschalten, führen Sie folgende Schritte aus:

Schritt	Vorgehen
1	Beschreiben Sie das Timeout-Register dieser Schnittstelle mit dem gewünschten Wert.
2	Warten Sie, bis das Bediengerät den Betriebssystemmerker dieser Schnittstelle gesetzt hat.
3	Setzen Sie den entsprechenden Anwendungsmerker.

Erkennen eines Timeouts

Um den Timeout zu erkennen, führen Sie folgende Schritte aus:

<table>
<tr><th>Schritt</th><th colspan="2">Vorgehen</th></tr>
<tr><td>1</td><td colspan="2">Schalten Sie die Aktivitätsüberwachung ein (siehe oben).</td></tr>
<tr><td>2</td><td colspan="2">Warten Sie, bis das Bediengerät den Anwendungsmerker dieser Schnittstelle zurückgesetzt hat.
Ergebnis: Ein Timeout ist aufgetreten.</td></tr>
<tr><td rowspan="3">3</td><td colspan="2">Prüfen Sie den entsprechenden Betriebssystemmerker.</td></tr>
<tr><td>Wenn ...</td><td>... dann ...</td></tr>
<tr><td>... der Betriebssystemmerker gesetzt ist,</td><td>... war die Verbindung kurz unterbrochen.</td></tr>
<tr><td>... der Betriebssystemmerker zurückgesetzt ist,</td><td>... ist die Verbindung immer noch unterbrochen.</td></tr>
</table>

10.6 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 der JVM-104 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 der Steuerung 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

Thema	Seite
Programmierung	208
Register	220
Programmbeispiele	225

10.6.1 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 der JVM-104 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

Einschränkungen

Während das Gerät JVM-104 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

Thema	Seite
Initialisieren der freiprogrammierbaren IP-Schnittstelle	209
Verbindung öffnen	210
Daten senden	214
Daten empfangen	216
Verbindung schließen	219

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 JVM-104 arbeitet die Funktion in folgenden Stufen ab:

Stufe	Beschreibung
1	Das Gerät JVM-104 schließt alle geöffneten Verbindungen der freiprogrammierbaren IP-Schnittstelle.
2	Das Gerät JVM-104 initialisiert alle betriebssysteminternen Datenstrukturen der freiprogrammierbaren IP-Schnittstelle.

Verwandte Themen

- **Verbindung öffnen** (siehe Seite 210)
- **Verbindung schließen** (siehe Seite 219)
- **Daten senden** (siehe Seite 214)
- **Daten empfangen** (siehe Seite 216)

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 JVM-104 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 JVM-104 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 JVM-104 nur einen Kommunikationskanal, über den Daten an einen Netzwerkteilnehmer gesendet werden. Die Funktion läuft in folgenden Stufen ab:

Stufe	Beschreibung	
1	Das Gerät JVM-104 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 JVM-104 nur einen Kommunikationskanal über den Daten von einem Netzwerkteilnehmer empfangen werden. Die Funktion läuft in folgenden Stufen ab:

Stufe	Beschreibung	
1	Das Gerät JVM-104 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 219)
- **Daten senden** (siehe Seite 214)
- **Daten empfangen** (siehe Seite 216)
- **Initialisieren der freiprogrammierbaren IP-Schnittstelle** (siehe Seite 209)

Daten senden

Einleitung

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

Über die UDP/IP-Verbindung eines Servers können keine Daten gesendet, sondern nur empfangen 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 einem UDP/IP-Client

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

Funktionsweise bei einem UDP/IP-Client

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.

Verwandte Themen

- **Initialisieren der freiprogrammierbaren IP-Schnittstelle** (siehe Seite 209)
- **Verbindung öffnen** (siehe Seite 210)
- **Verbindung schließen** (siehe Seite 219)
- **Daten empfangen** (siehe Seite 216)

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.

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 Empfangs-datenblocks	
`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 JVM-104 arbeitet die Funktion in folgenden Stufen ab:

Stufe	Beschreibung	
1	Das Gerät JVM-104 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 JVM-104 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. Die Steuerung arbeitet die Funktion in folgenden Stufen ab:

Stufe	Beschreibung	
1	Das Gerät JVM-104 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 209)
- **Verbindung öffnen** (siehe Seite 210)
- **Verbindung schließen** (siehe Seite 219)
- **Daten senden** (siehe Seite 214)

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 210)
- **Daten senden** (siehe Seite 214)
- **Daten empfangen** (siehe Seite 216)
- **Initialisieren der freiprogrammierbaren IP-Schnittstelle** (siehe Seite 209)

10.6.2 Register

Einleitung

Dieses Kapitel beschreibt die Register der JVM-104, 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

Thema	Seite
Registernummern	221
Registerbeschreibung	222

Registernummern

Einleitung

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

Registernummern

Steuerung	Basisregisternummer	Registernummern
JC-340, JC-350, JC-360, JC-360MC	350000	350000 ... 350007
JC-940MC	350000	350000 ... 350007
JVM-407, JVM-407B	350000	350000 ... 350007
JVM-104	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 hinzu, 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

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. Aus diesem Register ist auslesbar, 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

10.6.3 Programmbeispiele

Einleitung

Dieses Kapitel enthält Programmbeispiele für die Implementierung eines Servers und eines dazu passenden Clients, die über TCP/IP kommunizieren.

Beide Programmbeispiele sind auf das Bediengerät JVM-104 anwendbar.

Da die Funktionalität der freiprogrammierbaren IP-Schnittstelle nicht plattformspezifisch implementiert ist, sind diese Beispiele ohne Änderung auch auf andere Konfigurationen übertragbar.

Praktischer Anwendungsfall

Der praktische Anwendungsfall ist, das eine Steuerung oder ein Bediengerät der Jetter AG mit einem Gerät eines Fremdherstellers über TCP/IP kommuniziert. Die freiprogrammierbare IP-Schnittstelle bietet die Möglichkeit, Protokollanpassungen auf Seiten eines Geräts der Jetter AG im Anwendungsprogramm durchzuführen.

Inhalt

Thema	Seite
Server	226
Client	230

Server

Aufgabe

Ein Server soll einen Datenblock mit einer festgelegten Anzahl von Zeichen empfangen und die empfangenen Daten wieder an den Client zurückschicken.

Lösung

Programmierung eines Servers für die freiprogrammierbare IP-Schnittstelle, der über TCP/IP kommuniziert.

Softwareversionen

Das Beispielprogramm ist getestet mit den folgenden Softwareversionen:

- JetSym in der Version 5.2
- Bediengerät JVM-104 in der OS-Version 4.01

Sie finden weiterführende Informationen zum Thema STX-Programmierung in der Online-Hilfe von JetSym.

JetSym-STX-Programm

```
Const
  TCP_PORT = 52100;
  MSG_LEN  = 4000;
End_Const;

Var
  // Handle als Rückgabewert der Funktion "Verbindung öffnen"
  ConnHandle      : Int;

  // Sendepuffer
  SendBuf         : Array[MSG_LEN] Of Char;
  // Empfangspuffer
  RecvBuf         : Array[MSG_LEN] Of Char;

  ResConnInit     : Int;
  ResConnCreate   : Int;
  ResConnReceive  : Int;
  ResConnSend     : Int;

  ConnTimeOut     : Int;
  RecvTimeOut     : Int;

  // Anzahl Empfangsfehler
  RecvErrors      : Int;
  // Anzahl Sendefehler
  SendErrors      : Int;
  // Zähler der korrekten Übermittlungen der Nachricht
  CommCnt         : Int;

  AmountToReceive : Int;

  // Dummy
  NotUsed         : Int;
End_Var;
```

```
Task TCPserver Autorun

  Var
    RecvTimer     : Timer;
    ReceiveCnt    : Int;
  End_Var;

  // Timeout einer Verbindung
  ConnTimeOut := T#5s;
  // Timeout während des Empfangs eines Datenpakets
  RecvTimeOut := T#5s;

  // Anzahl der zu empfangenden Daten
  AmountToReceive := MSG_LEN;

  // Alle Verbindungen schließen, Datenstrukturen initialisieren
  ResConnInit := ConnectionInitialize();

  If ResConnInit >= 0 Then

    Trace('Server running.$n');

    While (True) Do

      // Verbindungsversuch
      ResConnCreate := ConnectionCreate
                       ( CONNTYPE_SERVER,
                         IPTYPE_TCP,
                         0,
                         TCP_PORT,
                         ConnTimeOut );

      If ResConnCreate > 0 Then

        Trace('Connection established.');
        // Rückgabewert (Handle) speichern
        ConnHandle := ResConnCreate;

        // Wiederholen, so lange die Verbindung besteht
        Loop

// Timeout für die komplette Übertragung des Datenpakets einrichten
          TimerStart(RecvTimer, RecvTimeOut * 5);
          // Zähler für die empfangenen Daten initialisieren
          ReceiveCnt := 0;

  // Wiederholen, bis alle erwarteten Daten empfangen wurden
  // Andernfalls wiederholen, bis ein Timeout auftritt
          While ReceiveCnt < AmountToReceive
```

```
                And Not TimerEnd(RecvTimer) Do

              ResConnReceive := ConnectionReceiveData
                                ( ConnHandle,
                                  NotUsed,
                                  NotUsed,
                                  RecvBuf[ReceiveCnt],
                                  SizeOf(RecvBuf),
                                  RecvTimeOut );

              If ResConnReceive > 0 Then
                // Etwas wurde empfangen, Zähler inkrementieren
                ReceiveCnt := ReceiveCnt + ResConnReceive;
              Else
                // Fehler beim Empfangen
                ResConnReceive := -1;
                // Fehlerzähler inkrementieren
                Inc(RecvErrors);
                // Schleife verlassen
                Exit;
              End_If;
            End_While;

// Hier ist die Stelle, um die Server-Funktion zu implementieren.
// In diesem Beispiel senden wir einfach die empfangenen Daten
// zurück.
            If ReceiveCnt Then
              // Daten von Empfangspuffer in Sendepuffer kopieren
              MemCopy(SendBuf,RecvBuf,SizeOf(SendBuf));
              ResConnSend := ConnectionSendData
                             ( ConnHandle,
                               0,
                               0,
                               SendBuf,
                               ReceiveCnt );
              If ResConnSend < 0 Then
                // Fehlerzähler inkrementieren
                Inc(SendErrors);
              End_If;
            End_If;

            If ResConnSend >= 0 And ResConnReceive >= 0 Then
              // Kein Fehler --> Zähler OK inkrementieren
              Inc(CommCnt);
            Else
              // Schleife verlassen
              Exit;
            End_If;

          End_Loop;
```

```
        If ConnHandle > 0 Then
          // Verbindung schließen
          ConnectionDelete(ConnHandle);
          // Handle nicht länger gültig
          ConnHandle := 0;
          Trace('Connection closed.$n');
        End_If;

      End_If;

  // Kurz warten, bis wieder ein Verbindungsversuch gestartet wird.
      Delay(T#3s);

    End_While;

  Else
    Trace('ConnectionInitialize() failed, server stopped !$n');
  End_If;

End_Task;
```

Verwandte Themen

- **Client** (siehe Seite 230)

Client

Aufgabe

Ein Client soll einen Datenblock mit einer festgelegten Anzahl von Zeichen versenden und die vom Server zurückgeschickten Daten empfangen.

Lösung

Programmierung eines Clients für die freiprogrammierbare IP-Schnittstelle, der über TCP/IP kommuniziert.

Softwareversionen

Das Beispielprogramm ist getestet mit den folgenden Softwareversionen:

- JetSym in der Version 5.2
- Bediengerät JVM-104 in der OS-Version 4.01

Sie finden weiterführende Informationen zum Thema STX-Programmierung in der Online-Hilfe von JetSym.

JetSym-STX-Programm

```
Const
  TCP_ADDR = IP#192.168.10.210;
  TCP_PORT = 52100;
  MSG_LEN  = 4000;
End_Const;

Var
  // Handle als Rückgabewert der Funktion "Verbindung öffnen"
  ConnHandle      : Int;
  // Sendepuffer
  SendBuf         : Array[MSG_LEN] Of Char;
  // Empfangspuffer
  RecvBuf         : Array[MSG_LEN] Of Char;
  ResConnInit     : Int;
  ResConnCreate   : Int;
  ResConnReceive  : Int;
  ResConnSend     : Int;
  ConnTimeOut     : Int;
  RecvTimeOut     : Int;

  // Anzahl Empfangsfehler
  RecvErrors      : Int;
  // Anzahl Sendefehler
  SendErrors      : Int;
  // Zähler der korrekten Übermittlungen einer Nachricht
  CommCnt         : Int;
  AmountToReceive : Int;
  SendDelay       : Int;

  // Dummy
  NotUsed         : Int;
End_Var;
```

```
Task TCPclient Autorun

  Var
    RecvTimer      : Timer;
    ReceiveCnt     : Int;
  End_Var;

  // Timeout einer Verbindung
  ConnTimeOut := T#5s;
  // Timeout während des Empfangs eines Datenpakets
  RecvTimeOut := T#5s;

  // Anzahl der zu empfangenden Daten
  AmountToReceive := MSG_LEN;
  SendDelay       := T#500ms;

  // Alle Verbindungen schließen, Datenstrukturen initialisieren
  ResConnInit := ConnectionInitialize();

  If ResConnInit >= 0 Then

    Trace('Client running.$n');

    While (True) Do

      // Verbindungsversuch
      ResConnCreate  := ConnectionCreate
                        ( CONNTYPE_CLIENT,
                          IPTYPE_TCP,
                          TCP_ADDR,
                          TCP_PORT,
                          ConnTimeOut );

      If ResConnCreate > 0 Then

        Trace('Connection established.');
        // Rückgabewert (Handle) speichern
        ConnHandle := ResConnCreate;

        // Wiederholen, so lange die Verbindung besteht
        Loop

          ResConnSend := ConnectionSendData
                         ( ConnHandle,
                           0,
                           0,
                           SendBuf,
                           AmountToReceive );
          If ResConnSend < 0 Then
            // Fehlerzähler inkrementieren
            Inc(SendErrors);
```

```
        End_If;

// Timeout für die komplette Übertragung des Datenpakets
// einrichten
        TimerStart(RecvTimer, RecvTimeOut * 5);
        // Zähler für die empfangenen Daten initialisieren
        ReceiveCnt := 0;

// Wiederholen, bis alle erwarteten Daten empfangen wurden.
// Andernfalls wiederholen, bis ein Timeout auftritt.
        While ReceiveCnt < AmountToReceive
            And Not TimerEnd(RecvTimer) Do

          ResConnReceive := ConnectionReceiveData
                              ( ConnHandle,
                                NotUsed,
                                NotUsed,
                                RecvBuf[ReceiveCnt],
                                SizeOf(RecvBuf),
                                RecvTimeOut );

          If ResConnReceive > 0 Then
            // Etwas wurde empfangen, Zähler inkrementieren
            ReceiveCnt := ReceiveCnt + ResConnReceive;
          Else
            // Fehler beim Empfangen
            ResConnReceive := -1;
            // Fehlerzähler inkrementieren
            Inc(RecvErrors);
            // Schleife verlassen
            Exit;
          End_If;
        End_While;

        If ResConnSend >= 0 And ResConnReceive >= 0 Then
          // Kein Fehler --> Zähler OK inkrementieren
          Inc(CommCnt);
          Delay(SendDelay);
        Else
          // Schleife verlassen
          Exit;
        End_If;

      End_Loop;

      If ConnHandle > 0 Then
        // Verbindung schließen
        ConnectionDelete(ConnHandle);
        // Handle nicht länger gültig
        ConnHandle := 0;
        Trace('Connection closed.$n');
```

```
        End_If;

      End_If;

  // Kurz warten, bis wieder ein Verbindungsversuch gestartet wird.
      Delay(T#3s);

    End_While;

  Else
    Trace('ConnectionInitialize() failed, client stopped !$n');
  End_If;

End_Task;
```

Verwandte Themen

- **Server** (siehe Seite 226)

11 Automatisches Kopieren von Steuerungsdaten

Einleitung

Dieses Kapitel beschreibt die Autostartfunktion, die es ermöglicht, Dateien innerhalb des JVM-104 zu kopieren. Hierzu erstellen Sie eine Kommandodatei, die zusammen mit den Daten auf einem USB-Stick abgelegt ist. Die Kommandodatei wird beim Anstecken des USB-Sticks automatisch abgearbeitet.

Funktionen im lokalen Dateisystem

Folgende Funktionen führt die Autostartfunktion aus:

- Verzeichnisse anlegen
- Verzeichnisse löschen
- Dateien kopieren
- Dateien löschen

Anwendungsbereiche

Für die Autostartfunktion gibt es hauptsächlich folgende Anwendungsbereiche:

- Wo Fernwartung nicht möglich ist.
- Wo kein PC vor Ort vorhanden ist.
- Wo der Anlagenbediener nicht in der Lage ist oder nicht in die Lage versetzt werden soll, Änderungen an der Anlage vorzunehmen.

Folgendes kann mit der Autostartfunktion ausgeführt werden:

- Änderung des Anwendungsprogramms
- Änderung der Anwendungsdaten
- Änderung der Steuerungskonfiguration
- Betriebssystemupdate (JVM-104)

Voraussetzungen

Für das automatische Kopieren von Steuerungsdaten sind folgende Voraussetzungen erforderlich:

- Der Programmierer verfügt über Kenntnisse des Dateisystems.

Bezeichnung

Vollständiger Name bezeichnet in dieser Beschreibung den Namen der Datei oder des Verzeichnisses mit komplettem Pfad.

Inhalt

Thema	Seite
Funktionsweise	236
autostart.bat - Aufbau	240
Datendateien	243

11.1 Funktionsweise

Einleitung

Dieses Kapitel beschreibt das Laden und Ausführen der Autostartfunktion.

Inhalt

Thema	Seite
Autostartfunktion laden	237
Autostartfunktion ausführen	238
Betriebsart Autostart beenden	239

Autostartfunktion laden

Einleitung

Die Autostartfunktion sollte nur während der Boot-Phase (nach dem Einschalten) des JVM-104 ausgeführt werden.

Voraussetzungen

Sie haben die Kommandodatei erstellt und im entsprechenden Verzeichnis abgelegt.

	Wert	Bemerkung
Dateiname	autostart.bat	Alles Kleinbuchstaben
Verzeichnis USB	\USB\	Stammverzeichnis auf dem USB-Stick

Autostartfunktion laden

Um die Autostartfunktion zu laden, führen Sie folgende Schritte aus:

Schritt	Vorgehen
1	Schalten Sie das Gerät aus.
2	Stecken Sie den USB-Stick in die USB-Schnittstelle.
3	Schalten Sie das Gerät ein.

Ergebnis: Das Gerät bootet und lädt die Autostartdatei.

Autostartfunktion ausführen

Einleitung

Während der Boot-Phase in der Betriebsart Autostart führt das Gerät die Kommandos in der Kommandodatei aus.

Autostartfunktion ausführen

Das Betriebssystem des JVM-104 bearbeitet die Autostartfunktion in folgenden Stufen:

Stufe	Beschreibung
1	Das Gerät lädt die Datei **\USB\autostart.bat** vom USB-Stick.
2 ... n	Das Gerät bearbeitet die Kommandos in aufsteigender Reihenfolge.

Betriebsart Autostart beenden

Einleitung

Das letzte Kommando in der Datei autostart.bat beendet die Betriebsart Autostart.

Neustart

Wahlweise können Sie das Gerät zu einem Neustart veranlassen, indem Sie das folgende Kommando an die letzte Zeile Ihrer Kommandodatei setzen:
del \System\reset.exe

Betriebsart Autostart beenden

Um die Betriebsart Autostart zu beenden, führen Sie folgende Schritte aus:

Schritt	Vorgehen
1	Entfernen Sie den USB-Stick.

11.2 autostart.bat - Aufbau

Einleitung

Dieses Kapitel beschreibt den Aufbau der Datei **autostart.bat** und die verfügbaren Kommandos.

Aufbau der Datei

Die Kommandodatei ist eine Textdatei, deren Einträge in Kommando- und Kommentarzeilen unterteilbar sind:

- Leerzeilen können Sie beliebig einfügen.
- Leiten Sie Kommentarzeilen mit *REM* ein.

Inhalt

Thema	Seite
Verfügbare Kommandos	241
Beispiel für eine Kommandodatei	242

Verfügbare Kommandos

Einleitung	In diesen Sektionen geben Sie die Befehle an, die die Funktion Autostart ausführt.
Bearbeitung der Kommandos	Die Autostartfunktion bearbeitet die Kommandos in der Reihenfolge von oben nach unten. Darüber hinaus können Sie Sprünge in der Abarbeitungsreihenfolge einbringen. Ein Sprung ist hierbei über das Schlüsselwort `goto` und einem Sprungziel definiert. Ein Sprungziel ist durch einen vorausstehenden Doppelpunkt (:) definiert. ▪ Ein einfacher Sprung ist wie folgt definiert: `goto` end {übersprungener Bereich} :end ▪ Ein bedingter Sprung ist wie folgt definiert: `if` {Bedingung} `goto` end {übersprungener Bereich} :end
Verfügbare Kommandos	Die Kommandos entsprechen gängiger Windows®-Kommandozeilensyntax. Die Syntax der Kommandos ist hier beschrieben: http://msdn.microsoft.com/en-us/library/ee499728%28v=winembedded.60%29.aspx Eine Auflistung der Kommandos ist hier beschrieben: http://msdn.microsoft.com/en-us/library/ee505427%28v=winembedded.60%29.aspx

Beispiel für eine Kommandodatei

Aufgabe

Bei einem eingebauten JVM-104 wollen Sie die Funktion erweitern. Hierfür sind folgende Änderungen erforderlich:

- Betriebssystemupdate
- Neues Anwendungsprogramm
- Neue Werte für einige Register

Lösung

Sie kopieren die benötigten Dateien auf einen USB-Stick und erstellen eine Kommandodatei für die Autostartfunktion.

Inhalt des USB-Sticks

Der USB-Stick enthält folgende Dateien:

- Die Datei **autostart.bat**
- Das neue Betriebssystem
- Eine .da-Datei für die neuen Werte der Register
- Eine **start.ini** und eine .es3-Datei für das neue Anwendungsprogramm

Kommandodatei

```
@echo off
if not exist \app\autostart.exe goto end
if exist \app\update.ini goto end
if exist \app\usb_update_process_marker goto dm
echo \data\update\jvm_ce0_1.00.0.04.os >\app\update.ini
md \data\update
copy \%1\jvm_ce0_1.00.0.04.os \data\update\jvm_ce0_1.00.0.04.os
echo >\app\usb_update_process_marker
md \app\progtest
copy \%1\progtest\progtest.es3 \app\progtest\progtest.es3
copy \%1\register.da \app\register.da
del \System\reset.exe
goto end
:dm
del \app\usb_update_process_marker
:end
```

11.3 Datendateien

Einleitung

Dieses Kapitel beschreibt die Datendatei, von der das Gerät die Werte von Registern und Merkern einliest.

Inhalt

Thema	Seite
Dateiformat	244
Datendateien automatisch laden	245

Dateiformat

Format

Die Datendatei ist wie folgt aufgebaut:

- Reine Textdatei
- Jeder Eintrag muss in einer eigenen Zeile stehen
- Jede Zeile muss mit Wagenrücklauf / Zeilenvorschub (carriage return/ line feed) abgeschlossen sein
- Kommentarzeilen werden mit einem Semikolon (;) eingeleitet
- Als erster Eintrag in der Datei muss als Kennung *SD1001* stehen

Datenzeilen

Eine Datenzeile besteht aus den folgenden Teilen:

- Variablenkennung am Beginn der Zeile
- Durch Leerzeichen oder Tabulator getrennt folgt die Variablennummer
- Durch Leerzeichen oder Tabulator getrennt folgt der Variablenwert

Variablenkennung	Variablentyp
FS	Merker
RS	Ganzzahlregister
QS	Fließkommazahlregister

Beispiel

```
SD1001
; Data File - Jetter AG
;
; Register 1000000 ... 1000005
RS    1000000    12345
RS    1000001    2
RS    1000002    -1062729008
RS    1000003    502
RS    1000004    50
RS    1000005    3
QS    1009000    3.14
;
; Flag 10 ... 13
FS    10    0
FS    11    1
FS    12    1
FS    13    0
```

Datendateien automatisch laden

Mechanismus

Das JVM-104 verfügt über einen Mechanismus, der es erlaubt, eine Datendatei automatisch vor dem Start des Anwendungsprogramms zu laden.

Voraussetzung

Die Datendatei müssen Sie mit dem Namen **register.da** versehen und im Ordner **App** ablegen.

Ablauf

So lädt das Gerät Datendateien:

Stufe	Beschreibung
1	Wenn Sie die Datendatei **register.da** im Ordner **App** abgelegt haben, erkennt das Gerät beim nächsten Gerätestart die Datei an dieser Stelle.
2	Das Gerät lädt die Datei von dieser Stelle ein.
3	Das Gerät bildet deren Werte auf die Register ab.
4	Nach dem Laden löscht das Gerät die Datendatei.
5	Das Gerät lädt das Anwendungsprogramm.

12 Betriebssystemupdate

Einleitung

Die Betriebssysteme der Bediengeräte der Jetter AG werden laufend weiterentwickelt. Dabei kommen neue Funktionen hinzu, bestehende Funktionen werden erweitert und Fehler beseitigt.

Dieses Kapitel beschreibt die Durchführung eines Betriebssystemupdates.

Download eines Betriebssystems

Die Jetter AG stellt die Betriebssysteme auf ihrer **Homepage http://www.jetter.de** zur Verfügung. Sie finden die Betriebssystemdateien zum Download unter *Mobile Automation - Support - Downloads* und über den Link *Betriebssystemdownload* der *Quicklinks* auf den Seiten der jeweiligen Bediengeräte.

Inhalt

Thema	Seite
Betriebssystemupdate bei einem Bediengerät	248

12.1 Betriebssystemupdate bei einem Bediengerät

Einleitung

Dieses Kapitel beschreibt, wie das Betriebssystemupdate eines JVM-104 durchgeführt wird. Es gibt hierbei mehrere Möglichkeiten, die Betriebssystemdatei an das Gerät zu übertragen:

- Mit dem Programmiertool JetSym
- Über eine FTP-Verbindung
- Über die Gerätewebseite
- Über das Verzeichnis *\App*

Inhalt

Thema	Seite
Betriebssystemupdate mit JetSym	249
Betriebssystemupdate über FTP	250
Betriebssystemupdate über USB	251
Betriebssystemupdate über die Gerätewebseite	254
Betriebssystemupdate über \App	256

Betriebssystemupdate mit JetSym

Einleitung

Mit Hilfe des Programmiertools JetSym ist es auf komfortable Weise möglich, eine Betriebssystemdatei auf das Gerät JVM-104 zu übertragen.

Voraussetzungen

- Eine Betriebssystemdatei für das Gerät JVM-104 ist vorhanden.
- Das Gerät ist per Ethernet mit dem PC verbunden.
  Die Nummer des Geräts im Konfigurationsspeicher ist als IP-Basisportnummer für die JetIP-Kommunikation eingestellt.
- Während des Betriebssystemupdates das Gerät eingeschaltet lassen.

Betriebssystem updaten

Führen Sie zum Betriebssystemupdate folgende Schritte aus:

Schritt	Vorgehen
1	Wählen Sie in JetSym im Menü **Build** den Punkt **OS updaten...** aus. **Ergebnis:** Ein Dateiauswahlfenster öffnet sich.
2	Wählen Sie die gewünschte Betriebssystemdatei aus. **Ergebnis:** JetSym öffnet ein Bestätigungsfenster.
3	Starten Sie die Übertragung der Betriebssystemdatei durch Klicken auf die Schaltfläche **Ja**.
4	Warten Sie, bis das Update beendet ist.
5	Um das eingespielte Betriebssystem zu starten, booten Sie das Gerät neu.

Betriebssystemupdate über FTP

Einleitung

Mit Hilfe eines FTP-Clients ist es möglich, eine Betriebssystemdatei an das Gerät JVM-104 zu übertragen.

Voraussetzungen

- Eine Betriebssystemdatei für das Gerät JVM-104 ist vorhanden.
- Eine FTP-Verbindung zum Gerät ist aufgebaut.
- Das Betriebssystem des JVM-104 läuft.
- Während des Betriebssystemupdates das Gerät eingeschaltet lassen.

Betriebssystem updaten

Führen Sie zum Betriebssystemupdate folgende Schritte aus:

Schritt	Vorgehen
1	Öffnen Sie eine FTP-Verbindung zum JVM-104.
2	Loggen Sie sich mit Administratorrechten ein.
3	Wechseln Sie in das Verzeichnis *\App*.
4	Übertragen Sie die Betriebssystemdatei.
5	Warten Sie, bis die Übertragung abgeschlossen ist.
6	Erstellen Sie mit einem Texteditor eine Datei mit Namen update.ini.
7	Schreiben Sie in diese Datei den Pfad und Dateinamen der Betriebssystemdatei auf dem Gerät, z. B. **\app\jvm_ce0_1.00.0.07.os**.
8	Übertragen Sie diese Datei ebenfalls in das Verzeichnis *\App*.
9	Schließen Sie die FTP-Verbindung.
10	Um das eingespielte Betriebssystem zu starten, booten Sie das Gerät neu.
11	Das neue Betriebssystem wird installiert. Warten Sie mindestens eine Minute.
12	Starten Sie das Gerät neu.

Betriebssystemupdate über USB

Einleitung

Mit Hilfe eines USB-Sticks ist es möglich, eine Betriebssystemdatei an das Gerät JVM-104 zu übertragen.

Voraussetzungen

- Eine Betriebssystemdatei für das Gerät JVM-104 ist vorhanden.
- Das Betriebssystem des JVM-104 ab Version 4.01.0.44 läuft.
- Während des Betriebssystemupdates das Gerät eingeschaltet lassen.

Inhalt der Batch-Datei

Die Batch-Datei **autostart.bat** muss den folgenden Inhalt haben:

```
@echo off
if not exist \app\autostart.exe goto end
if exist \app\update.ini goto end
if exist \app\usb_update_process_marker goto dm
\app\sys\testregistry AUTOSTART AutoCopy == 0
if ERRORLEVEL 1 goto end
echo \data\update\jvm_ce0_1.00.0.04.os >\app\update.ini
if not exist \data\update md \data\update
@cmd /c start \App\sys\jvcm_si 102
copy \%1\jvm_ce0_1.00.0.04.os \data\update\jvm_ce0_1.00.0.04.os
echo >\app\usb_update_process_marker
del \System\reset.exe
goto end
:dm
del \app\usb_update_process_marker
:end
```

Bei Geräten **ohne** Bedientasten müssen Sie die folgenden Kommandos aus der Batch-Datei **autostart.bat** löschen:

```
\app\sys\testregistry AUTOSTART AutoCopy == 0
if ERRORLEVEL 1 goto end
```

Betriebssystem updaten

Führen Sie zum Betriebssystemupdate bei Geräten **mit** Bedientasten folgende Schritte aus:

Schritt	Vorgehen
1	Kopieren Sie die OS-Datei auf einen USB-Stick mit FAT-Formatierung.
2	Erstellen Sie eine Batch-Datei mit dem Namen **autostart.bat** auf dem USB-Stick. Inhalt der Batch-Datei siehe oben.
3	Ersetzen Sie alle Stellen mit **jvm_ce0_1.00.0.04** mit dem aktuellen Namen der OS-Datei.
4	Schalten Sie das Gerät JVM-104 aus.
5	Stecken Sie den USB-Stick in das JVM-104 ein.
6	Schalten Sie das Gerät JVM-104 ein.

Schritt	Vorgehen
7	Drücken Sie gleichzeitig die erste und dritte Taste, bevor das Anwendungsprogramm gestartet ist.
⇨	Das Gerät geht in den Update-Modus und zeigt als Dialog den folgenden Text an: *Please select update source:* *Press [F1] for USB update | Press [F3] for SD update*
8	Drücken Sie die entsprechende Taste.
⇨	Das Gerät zeigt als weiteren Dialog den folgenden Text an: *Preparing update* *Please do not remove the update device* *Update will be performed after restart*
⇨	Ca. 10 Sekunden später startet das Gerät selbstständig das Update.
⇨	Wenn das Update beendet ist, zeigt das Gerät den folgenden Text an: *Update completed* *Please wait*
⇨	Das Gerät bootet jetzt selbstständig neu und startet die Applikation.

Führen Sie zum Betriebssystemupdate bei Geräten **ohne** Bedientasten folgende Schritte aus:

Schritt	Vorgehen
1	Kopieren Sie die OS-Datei auf einen USB-Stick mit FAT-Formatierung.
2	Erstellen Sie eine Batch-Datei mit dem Namen **autostart.bat** auf dem USB-Stick. Inhalt der Batch-Datei siehe oben.
3	Ersetzen Sie alle Stellen mit **jvm_ce0_1.00.0.04** mit dem aktuellen Namen der OS-Datei.
4	Den USB-Stick in das JVM-104 einstecken. Das geht auch bei laufendem Betrieb.

Ergebnis: Jetzt erfolgt die Installation des neuen Betriebssystems.

Chronologische Reihenfolge des Updates

Folgend ist die chronologische Reihenfolge des Betriebssystemupdates beschrieben:

Stufe	Beschreibung
1	Überspielen der Dateien
2	Reset des Geräts
3	Installation der Dateien
4	Reset des Geräts
5	Die Applikation erscheint

Log-Datei zur Dokumentation	Der Updatevorgang ist in der Datei **\App\update.log** dokumentiert.

Betriebssystemupdate über die Gerätewebseite

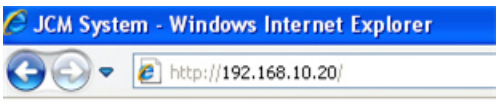
Einleitung

Auf der Gerätewebseite im Internet-Explorer sehen Sie alle Eigenschaften und Versionsnummern Ihres Geräts. Die Konfiguration der Ethernet-Schnittstelle und CANopen®-Schnittstelle ist auch möglich.

Aufruf der Gerätewebseite

Wenn Sie im Internet-Explorer die **aktuelle** IP-Adresse des Geräts eingeben, wird auf dem PC die Gerätewebseite angezeigt.

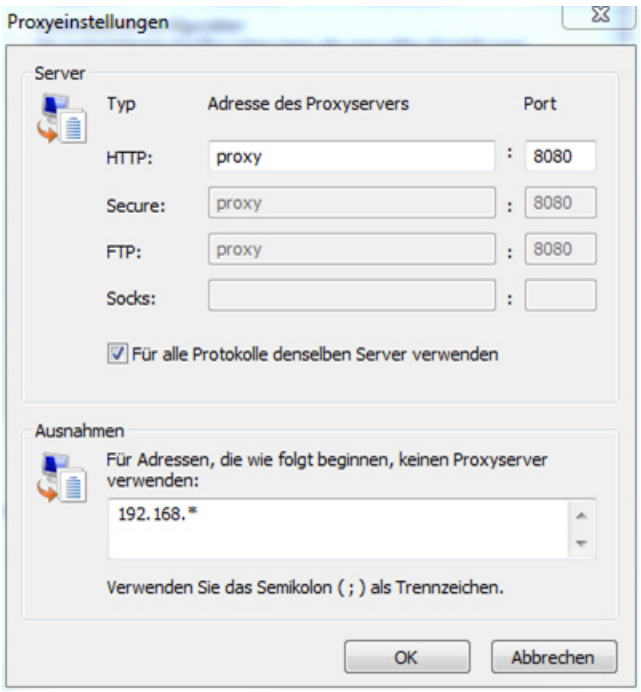
Beispiel:



Voraussetzungen

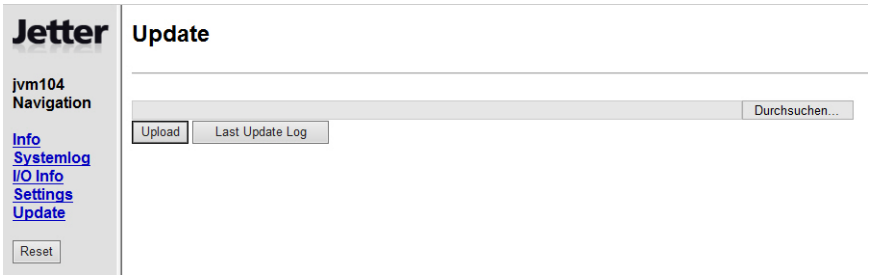
Um im Internet Explorer die IP-Adresse bei den Ausnahmen einzugeben, gehen Sie wie folgt vor:

Schritt	Vorgehen
1	Öffnen Sie über das Menü **Extras** das Fenster **Internetoptionen**.
2	Klicken Sie auf die Schaltfläche **LAN-Einstellungen** auf dem Reiter **Verbindungen**.
3	Klicken Sie auf die Schaltfläche **Erweitert** unter **Proxyserver**.
⇨	Das Fenster **Proxyeinstellungen** hat sich geöffnet.
4	Geben Sie im Fenster **Proxyeinstellungen** bei den **Ausnahmen** die IP-Adresse **192.168.*** ein, siehe die folgende Abbildung.



Update-Seite

Auf der Seite **Update** können Sie eine OS-Datei auswählen (*Durchsuchen*...) und auf das Gerät kopieren (*Upload*).



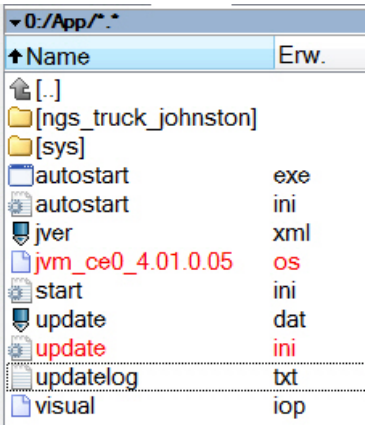
Wenn Sie auf die Schaltfläche **Last Update Log** klicken, wird Ihnen der Ablauf des letzten Updates angezeigt und Sie können eventuell aufgetretene Fehler nachvollziehen.

Betriebssystemupdate über \App

Vorgehensweise

Durch das Kopieren von Dateien in das Verzeichnis *\App* ist ein einfaches Betriebssystemupdate möglich. Führen Sie dazu die folgenden Schritte aus:

Schritt	Vorgehen
1	Schreiben Sie den Namen der Filecollection in die Datei **update.ini**. Sonst funktioniert das Update nicht.
2	Kopieren Sie die Filecollection oder OS und die Datei **update.ini** in das Verzeichnis *\App*.
3	Starten Sie das Bediengerät neu.
⇨	**Autostart.exe** findet das Update während dem Booten, installiert alle Dateien und führt einen Neustart aus. **Wichtig!** Unterbrechen Sie nicht diesen Vorgang.
4	Öffnen Sie die Datei **updatelog.txt** und schauen Sie, ob das Update erfolgreich war.



13 Anwendungsprogramm

Einleitung

Dieses Kapitel beschreibt, wie das Anwendungsprogramm in der JVM-104 abgelegt wird. Der Anwender legt fest, welches Programm ausgeführt werden soll.

Voraussetzungen an den Programmierer

Dieses Kapitel setzt Kenntnisse über das Erzeugen und Übertragen von Anwendungsprogrammen in JetSym und über das Dateisystem der JVM-104 voraus.

Inhalt

Thema	Seite
Standardablage des Anwendungsprogramms	258
Anwendungsprogramm laden	259

Standardablage des Anwendungsprogramms

Einleitung

Beim Download des Anwendungsprogramms von JetSym auf die Steuerung wird dieses Programm als Datei auf der internen Flash-Disk abgespeichert. Das Gerät trägt den Pfad- und Dateinamen in der Datei **\App\start.ini** ein.

Pfad- und Dateiname

Standardmäßig legt JetSym im Verzeichnis *\App* ein Unterverzeichnis mit dem Projektnamen an und speichert dort das Anwendungsprogramm mit der Dateierweiterung **.es3** ab. Pfad- und Dateinamen werden dabei immer in Kleinbuchstaben konvertiert.

\App\start.ini - Aufbau

Die Datei ist eine reine Textdatei mit einer Sektion mit zwei Einträgen:

Element	Beschreibung
[Startup]	Sektionsname
Project	Pfad zur Anwendungsprogrammdatei relativ zu *\App*
Program	Name der Anwendungsprogrammdatei

Beispiel:

```
[Startup]
Project = test_program
Program = test_program.es3
```

Ergebnis: Das Anwendungsprogramm wird aus der Datei **\App\test_program\test_program.es3** geladen.

Anwendungsprogramm laden

Einleitung

Beim Neustart des Anwendungsprogramms über JetSym oder nach dem Booten des JVM-104 wird das Anwendungsprogramm über das Dateisystem geladen und ausgeführt.

Der Ladevorgang

So wird das Anwendungsprogramm vom Betriebssystem des JVM-104 geladen.

Stufe	Beschreibung
1	Das Betriebssystem liest die Datei **\App\start.ini** von der internen Flash-Disk.
2	Das Betriebssystem wertet den Eintrag **Project** aus. Er enthält den Pfad zur Anwendungsprogrammdatei.
3	Das Betriebssystem wertet den Eintrag **Program** aus. Er enthält den Programmnamen.
4	Das Betriebssystem lädt das Anwendungsprogramm aus der Datei **<Project>\<Program>**.

14 Kurzreferenz JVM-104

Passende Betriebssystemversion

Diese Kurzreferenz beschreibt in stark zusammengefasster Form die Register und Merker des Bediengeräts JVM-104 in der OS-Version 4.01.

Default-Adresse am CANopen®-Bus

Die ausgelieferte Default-Adresse des JVM-104 ist:
Node-ID: 127 (0x7F)

Mögliche Anzahl an CANopen®-Schnittstellen

Mögliche Anzahl an CAN-Schnittstellen: 2
CANMAX: 1

Mögliche Anzahl an SAEJ1939-Schnittstellen

Mögliche Anzahl an CAN-Schnittstellen: 1
CANMAX: 1

Allgemeine Registerübersicht

100000 ... 100999	Electronic Data Sheet (EDS)
101000 ... 101999	Konfiguration
104000 ... 104999	Ethernet
106000 ... 106999	CAN
108000 ... 108999	CPU/Backplane
200000 ... 209999	Allg. Systemregister
210000 ... 219999	Anwendungsprogramm
230000 ... 239999	JetIP-Vernetzung
260000 ... 269999	RemoteScan
270000 ... 279999	Modbus/TCP
290000 ... 299999	E-Mail
310000 ... 319999	Dateisystem/Datendateien
350000 ... 359999	Freiprogrammierbare IP-Schnittstelle
360000 ... 369999	Display
1000000 ... 1029999	Anwendungsregister (remanent)

Allgemeine I/O-Übersicht

Eingabetasten

361000 ... 361007	Bitkodiertes Abbild der Eingabetasten

Allgemeine Merkerübersicht

0 ... 255	Anwendungsmerker (remanent)
256 ... 2047	Überlagert mit R 1000000 bis 1000055
2048 ... 2303	Spezialmerker

Electronic Data Sheet

100500	Schnittstelle (0 = CPU, 4 = Baseboard)

Identification

100600	Interne Versionsnummer
100601	Modulkennung
100602 ...	Modulname (Register-String)
100612	
100613	Platinenrevision
100614	Platinenoptionen

Production

100700	Interne Versionsnummer
100701 ...	Seriennummer (Register-String)
100707	
100708	Tag
100709	Monat
100710	Jahr
100711	TestNum.
100712	TestRev.

Features

100800	Interne Versionsnummer
100801	MAC-Adresse (Jetter)
100802	MAC-Adresse (Gerät)

Electronic Name Plate (Gesamtgerät)

Production

100900	Interne Versionsnummer
100901 ...	Seriennummer (Register-String)
100907	
100708	Tag
100709	Monat
100710	Jahr

Identification

100950	Interne Versionsnummer
100951	Modulkennung
100952 ...	Modulname (Register-String)
100962	
100965	Config-ID
100966	Vendor-ID
100967	Variant-ID
100968	Type-ID
100992	Navision-ID
100993	FBG-Version

Konfiguration

Aus Systemkonfiguration

101100	IP-Adresse (rw - remanent)
101101	Subnetzmaske (rw - remanent)
101102	Default Gateway (rw - remanent)

Vom System verwendet

101200	IP-Adresse
101201	Subnetzmaske
101202	Default Gateway

Ethernet

IP

104531	Aktuelle IP-Adresse (rw - temporär)
104532	Aktuelle Subnetzmaske (rw - temporär)
104533	Aktuelles Default Gateway (rw - temporär)

CAN

106000	Baudrate CAN 1
106001	Node-ID CAN 1
106100	Baudrate CAN 2
106101	Node-ID CAN 2

Flash-Speicher

107501	30: Lese aktuelle Flash-Speicher-Statistik 99: Lösche Flash-Speicher-Statistik
107510	Verfügbare Sektoren
107511	Verwendete Sektoren
107512	Blockierte Sektoren
107513	Freie Sektoren
107520	Größe Flash-Speicher in Byte
107521	Verwendeter Speicher in Byte
107522	Blockierter Speicher in Byte

107523	Freier Speicher in Byte

CPU-Hardware

108015	Backup-Spannung (z. B. der Uhr) 0 = Daten ungültig 1 = Versorgung ok Wenn die Versorgung wieder hergestellt ist, quittieren Sie das Register durch das Beschreiben mit 1.

Systeminformationen

108500 ... 108509	JetVM-DII-Version-String
108510 ... 108519	Versionsstring der Hostapplikation
108520 ... 108529	Dateiname der Hostapplikation
108530 ... 108539	OS-Version (String)
108570	CPU-Typ
108571	Anzahl der CPUs
108573	Physikalischer Arbeitsspeicher
108574	Freier physikalischer Arbeitsspeicher
108575	Auslastung des Arbeitsspeichers in Prozent
108581	Bildschirmbreite in Pixel
108582	Bildschirmhöhe in Pixel
108590	HID-Version

USB-Datenträger

109000	Bit 0 = 1: Datenträger vorhanden Bit 1 = 1: Datenträger bereit
109001	1 = Datenträger schreibgeschützt (nur gültig wenn R 109000 = 3)
109002	Größe in MByte

Allgemeine Systemregister

200000	OS-Version (Major * 100 + Minor)
200001	Anwendungsprogramm läuft (Bit 0 = 1)
200008	Fehlerregister (identisch zu 210004) Bit 8: Ungültiger Sprung Bit 9: Ungültiger Call Bit 10: Ungültiger Index Bit 11: Ungültiger Opcode Bit 12: Division durch Null Bit 13: Stack-Überlauf Bit 14: Stack-Unterlauf Bit 15: Stack ungültig Bit 16: Fehler beim Laden des Anwendungsprogramms Bit 24: Zykluszeitüberschreitung Bit 25: Tasklock Timeout Bit 31: Unbekannter Fehler
200168	Bootloader-Version (IP-Format)
200169	OS-Version (IP-Format)
201000	Laufzeitregister in Millisekunden (rw)
201001	Laufzeitregister in Sekunden (rw)
201002	Laufzeitregister in R 201003 Einheiten (rw)
201003	* 10 ms Einheiten für R 201002 (rw)
201004	Laufzeitregister in Millisekunden (ro)
201005	Laufzeitregister in Mikrosekunden (ro)
202930	Webstatus (bitkodiert) Bit 0 = 1: FTP-Server verfügbar Bit 1 = 1: HTTP-Server verfügbar Bit 2 = 1: E-Mail verfügbar Bit 3 = 1: Datendateifunktion verfügbar Bit 4 = 1: Modbus/TCP lizenziert Bit 5 = 1: Modbus/TCP verfügbar Bit 6 = 1: Ethernet/IP verfügbar
202960	Passwort für Systemkommandoregister (0x424f6f74)
202961	Systemkommandoregister
202980	Fehlerhistorie: Anzahl Einträge
202981	Fehlerhistorie: Index
202982	Fehlerhistorie: Eintrag
203000	Schnittstellenüberwachung: JetIP
203005	Schnittstellenüberwachung: Debug-Server
203100 ... 203107	32-Bit-Überlagerung Flag 0 ... 255
203108 ... 203123	16-Bit-Überlagerung Flag 0 ... 255
203124 ... 203131	32-Bit-Überlagerung Flag 2048 ... 2303
203132 ... 203147	16-Bit-Überlagerung Flag 2048 ... 2303
209700	Systemlogger: Globale Freigabe
209701 ... 209739	Freigabe Systemkomponenten

Anwendungsprogramm

210000	Anwendungsprogramm läuft (Bit 0 = 1)
210001	JetVM-Version
210004	Fehlerregister (bitkodiert) Bit 8: Ungültiger Sprung Bit 9: Ungültiger Call Bit 10: Ungültiger Index Bit 11: Ungültiger Opcode Bit 12: Division durch Null Bit 13: Stack-Überlauf Bit 14: Stack-Unterlauf Bit 15: Stack ungültig Bit 16: Fehler beim Laden des Anwendungsprogramms Bit 24: Zykluszeitüberschreitung Bit 25: Tasklock Timeout Bit 31: Unbekannter Fehler
210006	Höchste Task-Nummer
210007	Minimale Programmzykluszeit
210008	Maximale Programmzykluszeit
210009	Aktuelle Programmzykluszeit
210011	Aktuelle Task-Nummer
210050	Aktuelle Programmstelle innerhalb einer Ausführungseinheit
210051	ID der gerade bearbeiteten Ausführungseinheit
210056	Gewünschte Gesamtzykluszeit in µs
210057	Errechnete Gesamtzykluszeit in µs
210058	Maximale Zeitscheibe pro Task in µs
210060	Task-ID (für R 210061)
210061	Priorität für die Task [R 210060]

210063	Länge der Scheduler-Tabelle
210064	Index in Scheduler-Tabelle
210065	Task-ID in Scheduler-Tabelle
210070	Task-ID (für R 210071)
210071	Timer-Nummer (0 ... 31)
210072	Manuelles Auslösen eines Timer-Events (bitkodiert)
210073	Ende zyklischer Task (Task-ID)
210074	Kommando für zyklische Tasks
210075	Anzahl Timer
210076	Timer-Nummer (für R 210077)
210077	Timer-Wert in Millisekunden
210100 ... 210199	Task-Zustand
210400 ... 210499	Task-Programmadresse
210600	Task-ID eines zyklischen Tasks (für R 210601)
210601	Bearbeitungszeit eines zyklischen Tasks in Promille
210609	Tasklock Timeout in ms -1: Überwachung abgeschaltet
210610	Zeitüberschreitung (bitkodiert) Bit 0 -> Timer 0 usw.

JetIP-Vernetzung

230000	JetIP/TCP-Server: Anzahl offener Verbindungen
230001	JetIP/TCP-Server: Modus
230002	JetIP/TCP-Server: Zeit
232708	Timeout in Millisekunden
232709	Reaktionszeit in Millisekunden
232710	Anzahl Netzwerkfehler
232711	Fehlercode des letzten Zugriffs 0 = kein Fehler 1 = Timeout 3 = Fehlermeldung von Gegenstation 5 = ungültige Netzwerkadresse 6 = ungültige Anzahl Register 7 = ungültige Schnittstellennummer
232717	Maximale Anzahl Wiederholungen
232718	Anzahl Wiederholungen

Modbus/TCP

272702	Registeroffset
272704	Eingangsoffset
272705	Ausgangsoffset
278000 ... 278999	16-Bit-I/O-Register überlagert mit virtuellen I/O 20001 bis 36000

E-Mail

292932	IP-Adresse des SMTP-Servers
292933	IP-Adresse des POP3-Servers
292934	Port-Nummer des SMTP-Servers
292935	Port-Nummer des POP3-Servers
292937	Status der E-Mail-Bearbeitung
292938	E-Mail Task-ID

Dateisystem/Datendateifunktion

312977	Status der Dateioperation
312978	Task-ID

Freiprogrammierbare IP-Schnittstelle

Auslesen der Verbindungsliste

350000	Letztes Ergebnis (-1 = keine Verbindung ausgewählt)
350001	1 = Client; 2 = Server
350002	1 = UDP; 2 = TCP
350003	IP-Adresse
350004	Port-Nummer
350005	Zustand der Verbindung
350006	Anzahl gesendet Bytes
350007	Anzahl empfangener Bytes

Anwendungsregister

1000000 ... 1005999	32-Bit-Ganzzahl (remanent)

CAN-Prim-Register

200010500	Statusregister CAN-Prim
200010501	Kommandoregister CAN-Prim
200010502	Nummer der Nachrichtenbox
200010503	FIFO-Füllstand
200010504	FIFO-Daten
200010506	Globale Empfangsmaske
200010507	Globale Empfangs-ID
200010510	Boxstatusregister
200010511	Boxkonfigurationsregister
200010512	CAN-ID
200010513	Anzahl Datenbytes
200010514 ... 200010521	Datenbytes

Display

Eingabetasten

361000 ...	Bitkodiertes Abbild der Eingabetasten
361007	z. B. Bit 0: 1 = Taste 1 gedrückt
361000.0	KEY_OK
361000.1	KEY_ESC
361000.2	KEY_DOWN
361000.3	KEY_UP

Zündung (IGN)

361100	Bit 0: 0 = Zündung eingeschaltet 1 = Zündung ausgeschaltet

Digipot

363000	Aktueller Zählwert
363001	Bestätigungstaste Digipot
363002	Minimaler Zählwert
363003	Maximaler Zählwert

Display

364000	Hintergrundbeleuchtung
364001	Tastennachtbeleuchtung
364003	Helligkeitssensor

Visualisierung

365100	Sprachumschaltung nach ID

Netzwerkspezialmerker

2075	Fehler bei JetIP-Vernetzung

Spezialmerker Schnittstellenüberwachung

2088	OS-Flag JetIP
2089	User-Flag JetIP
2098	OS-Flag Debug-Server
2099	User-Flag Debug-Server

32 zusammengefasste Merker

203100	0 ... 31
203101	32 ... 63
203102	64 ... 95
203103	96 ... 127
203104	128 ... 159
203105	160 ... 191
203106	192 ... 223
203107	224 ... 255

16 zusammengefasste Merker

203108	0 ... 15
203109	16 ... 31
203110	32 ... 47
203111	48 ... 63
203112	64 ... 79
203113	80 ... 95
203114	96 ... 111
203115	112 ... 127
203116	128 ... 143
203117	144 ... 159
203118	160 ... 175
203119	176 ... 191
203120	192 ... 207
203121	208 ... 223
203122	224 ... 239
203123	240 ... 255

32 zusammengefasste Spezialmerker

203124	2048 ... 2079
203125	2080 ... 2111
203126	2112 ... 2143
203127	2144 ... 2175
203128	2176 ... 2207
203129	2208 ... 2239
203130	2240 ... 2271
203131	2272 ... 2303

16 zusammengefasste Spezialmerker

203132	2048 ... 2063
203133	2064 ... 2079
203134	2080 ... 2095
203135	2096 ... 2111
203136	2112 ... 2127
203137	2128 ... 2143
203138	2144 ... 2159
203139	2160 ... 2175
203140	2176 ... 2191
203141	2192 ... 2207
203142	2208 ... 2223
203143	2224 ... 2239
203144	2240 ... 2255
203145	2256 ... 2271
203146	2272 ... 2287
203147	2288 ... 2303

Anwendungsregister - Merker - Überlagerung

1000000	256 ... 287
1000001	288 ... 319
1000002	320 ... 351
1000003	352 ... 383
1000004	384 ... 415
1000005	416 ... 447
1000006	448 ... 479
1000007	480 ... 511
1000008	512 ... 543
1000009	544 ... 575
1000010	576 ... 607
1000011	608 ... 639
1000012	640 ... 671
1000013	672 ... 703
1000014	704 ... 735
1000015	736 ... 767
1000016	768 ... 799
1000017	800 ... 831
1000018	832 ... 863
1000019	864 ... 895
1000020	896 ... 927
1000021	928 ... 959
1000022	960 ... 991
1000023	992 ... 1023
1000024	1024 ... 1055
1000025	1056 ... 1087
1000026	1088 ... 1119
1000027	1120 ... 1151
1000028	1152 ... 1183
1000029	1184 ... 1215
1000030	1216 ... 1247
1000031	1248 ... 1279
1000032	1280 ... 1311
1000033	1312 ... 1343
1000034	1344 ... 1375
1000035	1376 ... 1407
1000036	1408 ... 1439
1000037	1440 ... 1471
1000038	1472 ... 1503
1000039	1504 ... 1535
1000040	1536 ... 1567
1000041	1568 ... 1599
1000042	1600 ... 1631
1000043	1632 ... 1663
1000044	1664 ... 1695
1000045	1696 ... 1727
1000046	1728 ... 1759
1000047	1760 ... 1791
1000048	1792 ... 1823
1000049	1824 ... 1855
1000050	1856 ... 1887
1000051	1888 ... 1919
1000052	1920 ... 1951
1000053	1952 ... 1983
1000054	1984 ... 2015
1000055	2016 ... 2047

Systemfunktion

Aus Kompatibilitätsgründen sind die Systemfunktionen hier gelistet. Nutzen Sie in JetSym STX anstelle der Systemfunktionen die entsprechenden JetSym-STX-Funktionen.

4	Konvertierung von BCD zu HEX
5	Konvertierung von HEX zu BCD
20	Quadratwurzel
21	Sinus
22	Cosinus
23	Tangens
24	Arcus Sinus
25	Arcus Cosinus
26	Arcus Tangens
27	Exponentialfunktion
28	Natürlicher Logarithmus
29	Absolutwert
30	Trennung von Vor- und Nachkommastellen
50	Registerwerte sortieren
60	CRC für Modbus RTU generieren
61	CRC für Modbus RTU prüfen
65/67	Registerblock über Modbus/TCP lesen
66/68	Registerblock über Modbus/TCP schreiben
80/85	RemoteScan initialisieren
81	RemoteScan starten
82	RemoteScan stoppen
90	Datendatei schreiben
91	Datendatei anfügen
92	Datendatei lesen
96	Datendatei löschen
110	E-Mail versenden
150	NetCopyList konfigurieren
151	NetCopyList löschen
152	NetCopyList senden

JetSym-STX-Funktionen

Systemfunktion	Entsprechende JetSym-STX-Funktion
4	Function Bcd2Hex(Bcd: Int): Int;
5	Function Hex2Bcd(Hex: Int): Int;
50	Function QSort(DataPtr: Int, ElementCnt: Int, ElementSize: Int, SortOffset: Int, SortType: STXBASETYPE, SortMode: QSORTMODE): Int;
60	Function ModbusCRCgen(FramePtr: Int, Length: Int): Int;
61	Function ModbusCRCcheck(FramePtr: Int, Length: Int): Int;
65/67	Function ModbusReadReg(Const Ref MbParam: MODBUS_PARAM): Int;
66/68	Function ModbusWriteReg(Const Ref MbParam: MODBUS_PARAM): Int;
80/85	Function RemoteScanConfig(Protocol: RSCAN_PROTOCOL, Elements: Int, Const Ref Configuration: RSCAN_DSCR): Int;
81	Function RemoteScanStart(Protocol: Int): Int;
82	Function RemoteScanStop(Protocol: Int): Int;
90/91	Function FileDAWrite(Const Ref FileName: String, Const Ref Mode: String, VarType: DAWRITE_TYPE, First: Int, Last: Int): Int;
92	Function FileDARead(Const Ref FileName: String): Int;
110	Function EmailSend(Const Ref FileName: String): Int;
150	Function NetCopyListConfig(IPAddr: Int, IPPort: Int, Const Ref List: TNetCopyLinstL): Int;
151	Function NetCopyListSend(Handle: Int): Int;
152	Function NetCopyListDelete(Handle: Int): Int;

15 Kurzreferenz JVM-104 Multifunktions-ein-/-ausgänge

Passende Betriebssystemversion

Diese Kurzreferenz beschreibt in stark zusammengefasster Form die Register der **Multifunktionsein-/-ausgänge** des JVM-104 ab der Betriebssystemversion 4.01.

Allgemeine Registerübersicht

Allgemeine Register der internen I/O-Platine

601000000	Identifikation Hardware
601000001	Hardwareversion (Anzeige im IP-Format)
601000003	Softwareversion (Anzeige im IP-Format)
601000013	Anzahl der CRC-Fehler ergänzen
601000014	Anzahl der Firmwarefehler seit dem letztem Neustart
601000015	CPU-Temperatur in 1 °C
601000016	Spannungsversorgung Logik in mV
601000020	Fehlercode des Geräts
601000021	Softwaremodul, aus dem der Fehler stammt

Multifunktionseingang MFQE1

601010000	Status MFQE1
601010001	Kommando MFQE1
601010002	Analogwert in µA/mV
601010003	Frequenz- [Torzeit] / Perioden- [µs] / Zählerwert [Inkremente]
601010004	Digitalwert am Eingang
601010005	Torzeit in ms bei der Frequenzmessung

Multifunktionseingang MFQE2

601020000	Status MFQE2
601020001	Kommando MFQE2
601020002	Analogwert in µA/mV
601020003	Frequenz- [Torzeit] / Perioden- [µs] / Zählerwert [Inkremente]
601020004	Digitalwert am Eingang
601020005	Torzeit in ms bei der Frequenzmessung

Multifunktionaler Ausgang PA3

601050000	Status PA3
601050001	Kommando PA3
601050002	PWM-Tastverhältnis in 0,1 %
601050004	Digitalwert am Ausgang
601050005	Gemessener Strom PA3 in mA
601050006	PWM-Frequenz in Hz
601050007	Einstellbare Grenze Überstrom in mA

Multifunktionsausgang PA4

601060000	Status PA4
601060001	Kommando PA4
601060002	PWM-Tastverhältnis in 0,1 %
601060004	Digitalwert am Ausgang
601060005	Gemessener Strom PA4 in mA
601060006	PWM-Frequenz in Hz
601060007	Einstellbare Grenze Überstrom in mA

Multifunktionseingang MFQEx

Register enthalten hier Platzhalter.
Es gilt:
x = 1 für Multifunktionseingang 1 (ab 601010000)
x = 2 für Multifunktionseingang 2 (ab 601020000)

6010x0000 **Status**

Bit 0 = 1:	Überstrom ausgelöst durch Strom > 2500 mA im Modus Strommessung
Bit 7 = 1:	Übertemperatur
Bit 14 = 0:	Multifunktionseingang MFQE1 ist eingeschaltet
Bit 14 = 1:	Multifunktionseingang MFQE1 ist ausgeschaltet

6010x0001 **Kommandos, bitkodiert**

Bit 7 ... 4:	
0000	Frequenzmessung mit Torzeit
0011	Zählmodus
0100	Periodenzeitmessung, 10 s Timeout
0101	Periondenzeitmessung, 1 s Timeout
0110	Periodenzeitmessung, 100 ms Timeout
Bit 9:	
0	Spannungsmessung
1	Strommessung
Bit 12:	
0	Pull-up-Widerstand ausschalten
1	Pull-up-Widerstand einschalten

6010x0002 Analogwert in mV/µA

6010x0003 Frequenz- [Torzeit] / Perioden- [µs] / Zählerwert [Inkremente]

6010x0004 Digitalwert am Eingang

6010x0005 Torzeit in ms bei der Frequenzmessung

Multifunktionsausgang PA3 und PA4

Register enthalten hier Platzhalter. Wenn zwei ähnliche Funktionen z. B. durch einen Offset verschoben sind, erleichtert das die Suche.
Es gilt:
x = 5 für Multifunktionsausgang 3 (ab 601050000)
x = 6 für Multifunktionsausgang 4 (ab 601060000)

6010x0000 **Status**

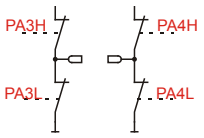
Bit 0 = 1:	Überstrom H-Side
Bit 3 = 1:	Überstrom L-Side
Bit 6 = 1:	Kabelbruch (open load)
Bit 7= 1:	Übertemperatur
Bit 14 = 0:	Multifunktionsausgang PAx ist eingeschaltet
Bit 14 = 1:	Multifunktionsausgang PAx ist ausgeschaltet
Bit 15 = 1:	Hardwareschutz hat ausgelöst

6010x0001 **Kommandos, bitkodiert**

Bit 1 ... 0:	
00	Ausgang als digitaler Ausgang x einschalten
01	Ausgang als PWM-Ausgang x einschalten
Bit 3 ... 2	
00	Ausgang PAx als Eingang
01	Ausgang PAx als Low-Side-Ausgang
10	Ausgang PAx als High-Side-Ausgang
11	Ausgang PAx als Low-High-Side-Ausgang
Bit 15:	
0	Ausgang PAx einschalten
1	Ausgang PAx ausschalten

6010x0002 PWM-Tastverhältnis in 0,1 %
6010x0004 Digitalwert am Ausgang PAx
6010x0005 Gemessener Strom PAx in mA
6010x0006 PWM-Frequenz in Hz
6010x0007 Einstellbare Grenze des Überstroms in mA

H-Brücke



Ausgänge der H-Brücke	
PA3H	Ausgang 1
PA3L	Ausgang 1
PA4H	Ausgang 2
PA4L	Ausgang 2

H-Brücke, Rechtslauf:
Schritte

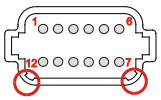
1	PA3: Low-Side-Ausgang einschalten: R 601050001 :=0x8005;
2	PA3: PWM-Tastverhältnis 100 % einstellen: R 601050002 := 1000;
3	PA4: High-Side-Ausgang einschalten: R 601060001 := 0x8009;
4	PA4: Tastverhältnis z. B. 50 % einstellen: R 601060002 := 500;

H-Brücke, Linkslauf:
Schritte

1	PA4: Low-Side-Ausgang einschalten: R 601060001 := 0x8005;
2	PA4: PWM-Tastverhältnis 100 % einstellen: R 601060002 := 1000;
3	PA3: High-Side-Ausgang einschalten: R 601050001 :=0x8009;
4	PA3: Tastverhältnis z. B. 50 % einstellen: R 601050002 := 500;

Belegung 12-polig DEUTSCH-Stecker DT06-12S

Sicht auf Geräterückseite



Pin	Beschreibung
1	Bezugspotenzial GND
2	CAN_2_L
3	CAN_2_H
4	Multifunktionsausgang PA3
5	Multifunktionsausgang PA4
6	Versorgungsspannung UB für die Logik des Geräts Spannung: DC 12 V oder DC 24 V Maximaler Strom: 2 A
7	Versorgungsspannung UB_PA für Multi-funktionsausgänge Spannung: DC 12 V oder DC 24 V Maximaler Strom: 10 A
8	Zündung
9	CAN_1_L
10	CAN_1_H
11	Multifunktionseingang MFQE1
12	Multifunktionseingang MFQE2

Anhang

Einleitung

Der Anhang enthält die elektrischen und mechanischen Daten sowie die Betriebsdaten.

Inhalt

Thema	Seite
Schnittstellen	270
Technische Daten	275
Index	284

A: Schnittstellen

Einleitung

Das Bediengerät JVM-104 hat folgende Schnittstellen:

- Deutsch-Stecker
- Ethernet-Schnittstelle
- USB-Schnittstelle

Deutsch-Stecker

Der Deutsch-Stecker hat folgende Funktion:

- Spannungsversorgung des JVM-104
- Spannungsversorgung für höhere Lastströme
- Multifunktionseingänge
- Multifunktionsausgänge
- CANopen®-Busschnittstellen: CAN 1 und CAN 2
- Erkennung der Zündung

Ethernet-Schnittstelle

Die RJ45-Buchse hat folgende Funktion:

- Ethernet-Schnittstelle zu einem PC
- Ethernet-Schnittstelle zu weiteren Geräten, wie einer Steuerung

USB-Schnittstelle

Die USB-Schnittstelle hat folgende Funktion:

- Update des Systems über USB-Stick
- Schnittstelle für USB-Sticks für weiteren Speicherplatz
- Schnittstelle für USB-Geräte wie Maus oder Tastatur

Inhalt

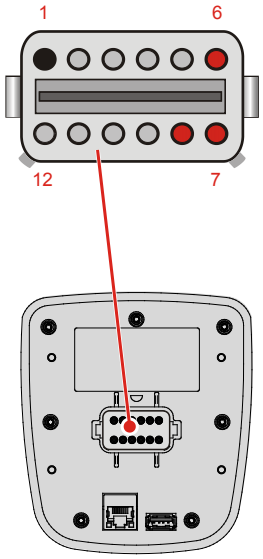
Thema	Seite
Gesamtübersicht der Anschlussbelegung	271

Gesamtübersicht der Anschlussbelegung

Spannungsversorgung

Folgend ist die Pinbelegung des Anschlusssteckers für die Spannungsversorgung beschrieben.

Die Abbildung zeigt die Pinbelegung des Anschlusssteckers für die Spannungsversorgung und den Anschluss der Zündung (Blick auf die Kabelseite):



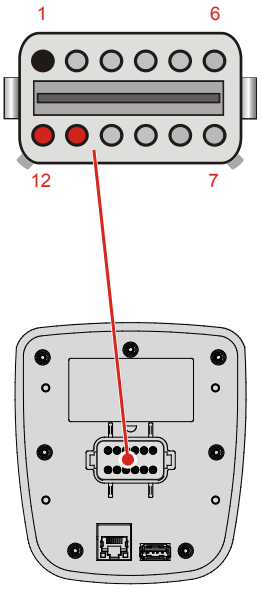
Die Pinbelegung ist wie folgt:

Pin	Beschreibung	Klemmenbezeichnung in KFZ
1	Bezugspotenzial GND	KL 31
6	Versorgungsspannung UB für die Logik des Geräts Spannung: DC12 V oder DC 24 V Stromaufnahme: Maximal 2 A	KL 30
7	Versorgungsspannung UB_PA für Multifunktionsausgänge des Geräts Spannung: DC12 V oder DC 24 V Stromaufnahme: Maximal 10 A	KL 30
8	Zündung (+)	KL 15

Multifunktionseingänge

Folgend ist die Pinbelegung des Anschlusssteckers für die Multifunktionseingänge beschrieben.

Die Abbildung zeigt die Pinbelegung des Anschlusssteckers für die Multifunktionseingänge (Blick auf die Kabelseite). Der Pin 1 des Bezugspotenzials ist auch farblich gekennzeichnet.



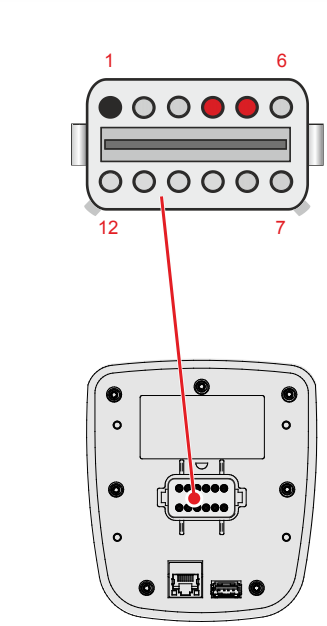
Die Pinbelegung ist wie folgt:

Pin	Beschreibung
1	Bezugspotenzial GND
11	Multifunktionseingang MFQE1
12	Multifunktionseingang MFQE2

Multifunktionsausgänge

Folgend ist die Pinbelegung des Anschlusssteckers für die Multifunktionsausgänge beschrieben.

Die Abbildung zeigt die Pinbelegung des Anschlusssteckers für die Multifunktionsausgänge und die CAN-Schnittstelle (Blick auf die Kabelseite). Der Pin 1 des Bezugspotenzials ist auch farblich gekennzeichnet.



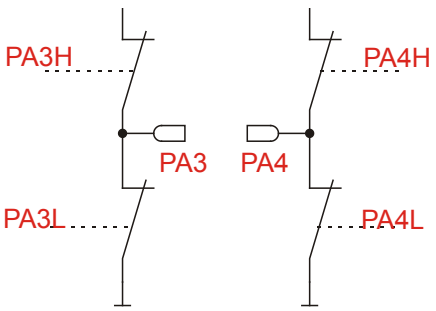
Die Pinbelegung ist wie folgt:

Pin	Beschreibung
1	Bezugspotenzial GND
4	Multifunktionsausgang PA3
5	Multifunktionsausgang PA4

Pinbelegung für die H-Brücke

Die H-Brücke ist auf die folgenden Ausgänge herausgeführt:

- Multifunktionsausgang PA3
- Multifunktionsausgang PA4

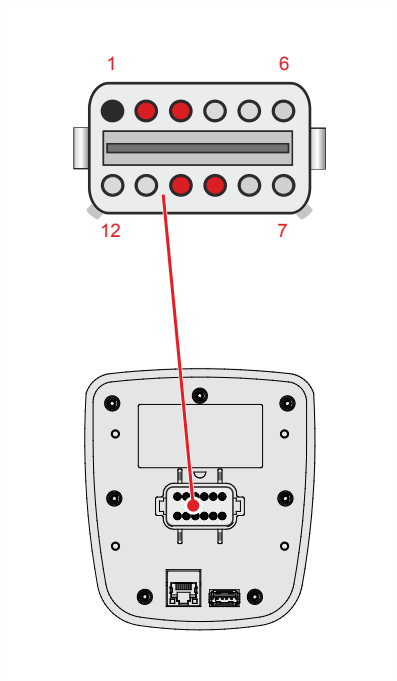


Bei der Programmierung schalten Sie PA3H und PA4L und PA4H und PA3L für Links- und Rechtslauf in die gewünschte Richtung.

CAN-Schnittstellen

Folgend ist die Pinbelegung des Anschlusssteckers für die CAN-Schnittstellen beschrieben.

Die Abbildung zeigt die Pinbelegung des Anschlusssteckers für den CANopen®-Bus (Blick auf die Kabelseite). Der Pin 1 des Bezugspotenzials ist auch farblich gekennzeichnet.



Die Pinbelegung ist wie folgt:

Pin	Beschreibung
1	Bezugspotenzial GND
2	CAN_2_L
3	CAN_2_H
9	CAN_1_L
10	CAN_1_H

B: Technische Daten

Einleitung

Dieses Kapitel im Anhang enthält die elektrischen und mechanischen Daten sowie die Betriebsdaten des JVM-104.

Inhalt

Thema	Seite
Technische Daten	276
Technische Daten der Multifunktionsein-/-ausgänge	278
Mechanische Abmessungen	281
Betriebsparameter Umwelt und Mechanik	282
Betriebsparameter EMV	283

Technische Daten

Technische Daten - Spannungsversorgung UB

Parameter	Beschreibung
Nennspannung	DC 12 V oder DC 24 V
Zulässiger Spannungsbereich UB	DC 8 V ... DC 32 V, gemäß ISO 7637
Zulässiger Spannungsbereich Zündung	DC 5 V ... DC 32 V
Maximaler Strom	2 A
Load-Dump Schutz	Bis DC 70 V
Typische Stromaufnahme Logik (UB)	170 mA bei DC 12 V 90 mA bei DC 24 V
Leistungsaufnahme	Ca. 2 W
Integrierte Schutzfunktionen	Verpolschutz, Überlast, Kurzzeitige Spannungspulse

Technische Daten - Display

Parameter	Beschreibung
Display	3,5"-TFT-LCD-Flachbildschirm
Lichtstärke	LED-Backlight (weiß) , typ. 350 cd/m^2
Auflösung Display	320 x 240 Pixel

Technische Daten - CAN-Schnittstelle

Parameter	Beschreibung
Baudrate	250 kBaud ... 1 MBaud
CAN 1: Busabschlusswiderstand	Intern nicht bestückt
CAN 2: Busabschlusswiderstand	Intern mit 120 Ω bestückt
Erforderlich bei CAN 1: Externer Busabschluss	120 Ω
Kabelspezifikation	Verdrillte Adern, ungeschirmt

Anzahl möglicher CANopen®-Schnittstellen

Parameter	Beschreibung
Anzahl möglicher CAN-Schnittstellen	2
CANMAX	1

Anzahl möglicher SAEJ1939-Schnittstellen

Parameter	Beschreibung
Anzahl möglicher CAN-Schnittstellen	1
CANMAX	1

Technische Daten - USB-Stick

Parameter	Beschreibung
Versorgung	5 V, max. 500 mA
USB-Anschluss	USB Host-Interface, USB 2.0
Kurzschlussfestigkeit	Ja, Kurzschlussstrom kurzzeitig: ~ 0,8 A

Technische Daten - Ethernet

Parameter	Beschreibung
Klemmenart	RJ45-Buchse
Anzahl Ports	1
Übertragungsrate	10 MBit/s, 100 MBit/s
Auto-Crossover	Ja

Speicherausbau

Parameter	Beschreibung
Anzahl remanente Register	30.000
Remanenter Variablenspeicher	120.000 Byte
Flash-Disk: Gesamter Speicher Ordner **App** Ordner **Data**	 512 MByte 64 MByte 368 MByte

Technische Daten der Multifunktionsein-/-ausgänge

Technische Daten - Spannungsversorgung UB_PA

Parameter	Beschreibung
Nennspannung	DC 12 V oder DC 24 V
Zulässiger Spannungsbereich UB_PA	DC 8 V ... DC 32 V
Zulässiger Spannungsbereich Zündung	DC 5 V ... DC 32 V
Maximaler Strom	10 A
Load-Dump Schutz	Bis DC 70 V
Integrierte Schutzfunktionen	Verpolschutz

Technische Daten - Digitale Eingänge MFQE

Parameter	Beschreibung
Art der Eingänge	Active-High-Eingang
Schaltschwelle EIN	Minimal 2,0 V
Schaltschwelle AUS	Maximal 1,0 V
Eingangswiderstand	50 kΩ
Maximale Eingangsfrequenz	1 Hz
NAMUR-Sensorauswertung	Über Software aktivierbar
Potenzialtrennung	Keine

Technische Daten - Spannungsmessung

Parameter	Beschreibung
Messbereich	DC 0 V ... DC 15 V
Eingangswiderstand	50 kΩ
Auflösung	12 Bit
Potenzialtrennung	Keine

Technische Daten - Strommessung

Parameter	Beschreibung
Messbereich	DC 0 mA ... DC 20 mA
Messwiderstand	120 Ω
Auflösung	12 Bit
Überstromabschaltung	Ja
Potenzialtrennung	Keine

Technische Daten - Frequenzmessung

Parameter	Beschreibung
Messbereich	0,1 Hz ... 10 kHz
Genauigkeit	99,9 %
Eingangswiderstand	20 kΩ
Typischer Eingangsstrom	Max. 1 mA
Maximale Eingangsfrequenz	Max. 10 kHz
Auflösung der Periodendauermessung	1 µs
Einstellbare Torzeiten für Torzeitmessung	1 ms ... 10 s
Duty-Cycle-Frequenzeingang	Mindestens 35 µs
Schaltschwelle EIN	Min. 4,7 V
Schaltschwelle AUS	Max. 2,5 V

Technische Daten - Zählfunktion

Parameter	Beschreibung
Messbereich	0,1 Hz ... 10 kHz
Auflösung	0,1 %
Zählbereich	32 Bit
Schaltschwelle EIN	Min. 4,7 V
Schaltschwelle AUS	Max. 2,5 V

Technische Daten - Digitaler Eingang PAx

Parameter	Beschreibung
Art des Eingangs	Active-High-Eingang
Nennspannung	UB_PA - 10 %
Schaltschwelle EIN	Maximal DC 1,6 V, > 3,5 mA
Schaltschwelle AUS	Minimal DC 4,0 V, < 0,5 mA
Interner Pull-up-Widerstand	NAMUR: 1 kΩ, 8,2 V
Potenzialtrennung	Keine

Technische Daten - Digitaler Ausgang

Parameter	Beschreibung
Maximaler Ausgangsstrom pro Kanal	2,5 A
Nennspannung	Versorgungsspannung UB_PA
Diagnosemöglichkeit	Überstrom, Kurzschluss, Kabelbruch
Auswertung Überstrom	Einstellbare Stromabschaltung
Maximale Latenzzeit: Umschalten zwischen Aus- und Eingang	500 ms
Maximaler Summenstrom	Der Summenstrom aller Ausgänge darf 8 A nicht überschreiten, sonst Abschaltung nach 500 ms. Wenn der Summenstrom 10,5 A überschreitet, schalten die Ausgänge sofort ab.

Technische Daten - PWM-Ausgang

Parameter	Beschreibung
PWM-Frequenz	100 Hz ... 1 kHz
Auflösung Tastverhältnis	0,1 %
Genauigkeit Tastverhältnis	+/- 42 ns
Maximale Nennspannung	Versorgungsspannung UB_PA
Schaltfrequenz	100 Hz ... 1kHz
Schutzbeschaltung	Überspannung, Überstrom
Diagnosemöglichkeit	Überstrom, Kabelbruch
Maximaler Summenstrom	Der Summenstrom aller Ausgänge darf 8 A nicht überschreiten, sonst Abschaltung nach 500 ms. Wenn der Summenstrom 10,5 A überschreitet, schalten die Ausgänge sofort ab.

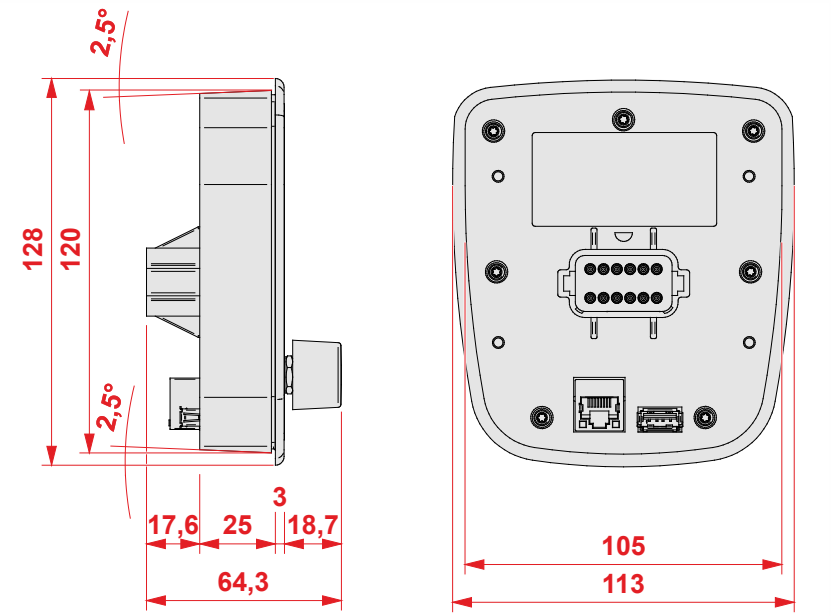
Technische Daten - H-Brücke

Parameter	Beschreibung
Maximale Nennspannung	Batteriespannung, max. DC 32 V
Maximaler Ausgangsstrom	2,5 A
Spitzenstrom	~ 5 A, max. 500 ms
Genauigkeit der Strommessung	+/- 20 %
Schaltfrequenz (1/Periodendauer)	100 Hz ... 1 kHz
Frequenzgenauigkeit	+/- 42 ns
Auflösung Tastverhältnis	0,1 %
Diagnosemöglichkeit	Überstrom

Mechanische Abmessungen

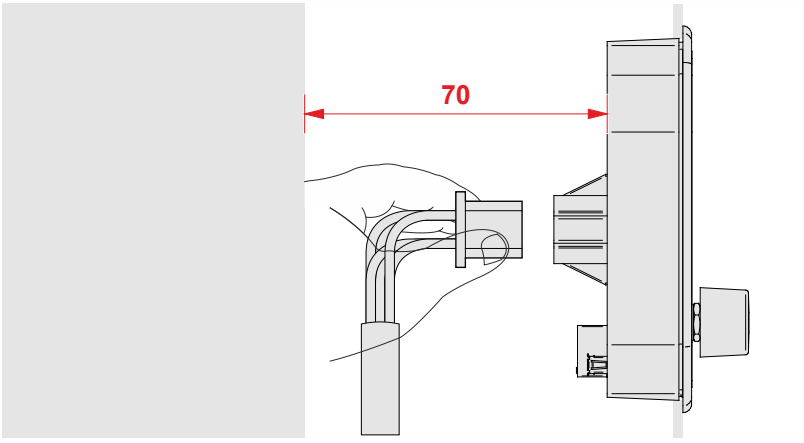
Mechanische Abmessungen

Die Abbildung zeigt die Abmessungen des JVM-104 in Millimeter.



Platzbedarf für den Einbau und Service

Die Abbildung zeigt den Platzbedarf für das Bediengerät JVM-104. Das Maß ist in Millimeter angegeben.



Halten Sie den Raum um das Gehäuse für den Servicefall frei.

- Die Stecker müssen sich jederzeit abziehen lassen.

Betriebsparameter Umwelt und Mechanik

Umwelt

Parameter	Wert	Norm
Betriebstemperaturbereich	-20 ... +60 °C	
Lagertemperaturbereich	-20 ... +70 °C	ISO 16750-4 DIN EN 60068-2-1 DIN EN 60068-2-2
Luftfeuchtigkeit	10 ... 95 % Das JVM-104 nicht mit Dampfstrahler oder ähnlichem reinigen.	DIN EN 61131-2
Klimatest	Feuchte Wärme	ISO 16750-4
Verschmutzungsgrad	2	DIN EN 61131-2
Einbauort	Das JVM-104 ist in der Fahrerkabine einzubauen.	

Mechanik

Parameter	Wert	Norm
Vibration	Breitbandrauschen, 10 Hz / 0,005 $(m/s^2)^2$/Hz 200 Hz / 0,02 $(m/s^2)^2$/Hz 300 Hz / 0,01 $(m/s^2)^2$/Hz 350 Hz / 0,002 $(m/s^2)^2$/Hz Dauer: 3x 24 h	Nach DIN EN 60068-2-64
Schockfestigkeit	Halbsinus, 30 g (300 m/s^2), 18 ms, 3 Schocks in alle 6 Richtungen	Nach DIN EN 60068-2-27
Schutzart	Frontseitig: IP65 Rückseitig: IP20	DIN EN 60529

Betriebsparameter EMV

Spannungsprüfung an UB und UB_PA

Die Spannungsprüfung nach DIN EN 16750-2 ist erfüllt.

EMV-Störaussendung

Parameter	Wert	Norm
Störaussendung nach e1	400 ... 1.000 MHz mit 63 dB (µV/m) konstant	DIN EN 55025
Störaussendung nach CE	0,15 ... 0,5 MHz, 66 ... 56 dB (µV) QP DC-Versorgung 0,5 ... 5 MHz, 56 dB (µV) QP 5 ... 30 MHz, 60 dB (µV) QP	DIN EN 55011-DC
	30 ... 230 MHz, 30/40 dB (µV/m) Gehäuse 230 ... 1.000 MHz, 37/47 dB (µV/m)	DIN EN 55011-HF

EMV-Störfestigkeit

Parameter	Wert	Norm
Störfestigkeit nach CE	10 V/m über 80 % des Frequenzbereichs	DIN EN 61000-4-3
	2/1 kV Daten 4/2 kV Power	DIN EN 61000-4-4
	± 1 kV Line/Ground ± 0,5 kV Line/Line	DIN EN 61000-4-5
	10 V von 0,15 ... 80 MHz, 80 % AM Sinus 1 kHz	DIN EN 61000-4-6
ESD	Luftentladung: Prüfscheitelspannung 8 kV Kontaktentladung: Prüfscheitelspannung 4 kV	DIN EN 61000-4-2

C: Index

A

Anwendungsprogramm
  laden • 259
  Standardablage • 258

B

Bestandteile • 16
Bestellbezeichnung • 19
Bestimmungsgemäße Verwendung • 10
Betriebsparameter
  EMV • 283
  Umwelt und Mechanik • 282
Betriebssystemupdate • 247

C

CANopen® • 85

D

Dateisystem • 147
  Eigenschaften • 152
  Verzeichnisse • 148

E

Eingabe über das Digipot • 78
Entsorgung • 10
Erstinbetriebnahme • 59

F

Freiprogrammierbare IP-Schnittstelle • 206
  IP-Schnittstelle programmieren • 208
  Registerübersicht • 221
FTP-Server • 155

I

IP-Adresse zur Laufzeit einstellen • 55

K

Kurzreferenz • 261
Kurzreferenz Multifunktionsein-/-ausgänge • 267

L

Laufzeitregister • 196

M

Mechanische Abmessungen • 20
Montage • 46

N

Nicht bestimmungsgemäße Verwendung • 10

P

Personalqualifikation • 10
Produktbeschreibung • 14
Programmierung
  Digipot • 175
  Eingabetasten • 173
  Multifunktionsein-/ausgänge • 179
  Zündung und Ausschaltverzögerung • 177

R

Reparatur • 10

S

SAE J1939 • 123
Schnittstellenaktivität überwachen • 201
Speicherarten • 161
Speicherübersicht • 161
Stecker • 30
  Beispiel einer Verdrahtung • 31
  CAN • 41
  Ethernet • 43
  Multifunktionsausgänge • 38
  Multifunktionseingänge • 35
  Spannungsversorgung • 32
  USB • 44

T

Technische Daten • 276
Transport • 10
Typenschild • 23

U

Umbauten • 10

V

Versionsregister • 24
Visualisierung ändern (Visualisierungskommando) • 82
Visualisierung anlegen
  in JetSym • 69
  in JetViewSoft • 64

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.