



# Betriebsanleitung

JVM-104 - Bediengerät

60879280

We automate your success.

Variante: Jetter  
Ausführung: O01  
Artikelnummer: 60879280  
Version 4.01.2  
Mai 2015 / 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 beigelegt, 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:	<a href="mailto:sales@jetter.de">sales@jetter.de</a>
E-Mail - Technische Hotline:	<a href="mailto:hotline@jetter.de">hotline@jetter.de</a>

---

### **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](mailto: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

<b>1</b>	<b>Sicherheitshinweise</b>	<b>9</b>
	Grundlegende Sicherheitshinweise .....	10
<b>2</b>	<b>Produktbeschreibung und Geräteaufbau</b>	<b>13</b>
	Produktbeschreibung .....	14
	Teile und Schnittstellen .....	15
	Bestellbezeichnung .....	17
	Mechanische Abmessungen .....	18
<b>3</b>	<b>Identifikation des JVM-104</b>	<b>19</b>
3.1	<b>Identifikation über das Typenschild</b> .....	<b>20</b>
	Typenschild .....	21
3.2	<b>Versionsregister</b> .....	<b>22</b>
	Softwareversionen .....	23
<b>4</b>	<b>Montage und Installation des JVM-104</b>	<b>25</b>
4.1	<b>Schnittstellen</b> .....	<b>26</b>
	Beispiel einer Verdrahtung .....	27
	Anschluss der Spannungsversorgung .....	28
	CAN-Schnittstelle .....	30
4.2	<b>Montage des JVM-104</b> .....	<b>32</b>
	Montage .....	33
<b>5</b>	<b>Erstinbetriebnahme</b>	<b>39</b>
5.1	<b>Vorbereitung und Einstieg in die Programmierung mit JetSym STX</b> .....	<b>40</b>
	Vorbereitungen zur Erstinbetriebnahme .....	41
	Programmierung in der Sprache JetSym STX .....	43
5.2	<b>Konfiguration bei der Plattform ER-STX-CE</b> .....	<b>44</b>
	Erstinbetriebnahme in JetViewSoft .....	45
	Visualisierungsprojekt in JetSym anlegen und konfigurieren .....	50
5.3	<b>Programmierung bei der Plattform ER-STX-CE</b> .....	<b>58</b>
	Werteingabe über das Digipot .....	59
	Visualisierungsobjekt ändern über die Visualisierungskommandos .....	63
<b>6</b>	<b>CANopen®-STX-API</b>	<b>65</b>
	STX-Funktion CanOpenInit() .....	67
	STX-Funktion CanOpenSetCommand() .....	69
	STX-Funktion CanOpenUploadSDO() .....	71
	STX-Funktion CanOpenDownloadSDO() .....	76
	STX-Funktion CanOpenAddPDORx() .....	81
	STX-Funktion CanOpenAddPDOTx() .....	88
	Heartbeat-Überwachung .....	94
	CANopen®-Objektverzeichnis des JVM-104 .....	98

<b>7</b>	<b>SAE J1939-STX-API</b>	<b>103</b>
	Aufbau einer J1939-Nachricht.....	104
	STX-Funktion SAEJ1939Init().....	106
	STX-Funktion SAEJ1939SetSA().....	107
	STX-Funktion SAEJ1939GetSA().....	108
	STX-Funktion SAEJ1939AddRx().....	109
	STX-Funktion SAEJ1939AddTx().....	112
	STX-Funktion SAEJ1939RequestPGN().....	115
	STX-Funktion SAEJ1939GetDM1().....	118
	STX-Funktion SAEJ1939GetDM2().....	121
	STX-Funktion SAEJ1939SetSPNConversion().....	124
	STX-Funktion SAEJ1939GetSPNConversion().....	125
<b>8</b>	<b>Dateisystem</b>	<b>127</b>
<b>8.1</b>	<b>Verzeichnisse</b> .....	<b>128</b>
	Verzeichnisse .....	129
<b>8.2</b>	<b>Eigenschaften</b> .....	<b>132</b>
	Eigenschaften der Flash-Disk .....	133
<b>9</b>	<b>Programmierung</b>	<b>135</b>
<b>9.1</b>	Abkürzungen, Modulregistereigenschaften und Formatierungen .....	136
	<b>Speicherübersicht</b> .....	<b>137</b>
	Speicher des Betriebssystems .....	138
	Speicher des Dateisystems .....	139
	Speicher des Anwendungsprogramms.....	140
	Speicher für flüchtige Variablen des Anwendungsprogramms.....	141
	Speicher für nichtflüchtige Register des Anwendungsprogramms.....	142
	Speicher für nichtflüchtige Variablen des Anwendungsprogramms .....	143
	Spezialregister .....	145
	Merker .....	146
<b>9.2</b>	<b>Bedienelemente und Zündung</b> .....	<b>148</b>
	Eingabetasten .....	149
	Digipot .....	151
	Zündung und Ausschaltverzögerung.....	153
<b>9.3</b>	<b>Laufzeitregister</b> .....	<b>155</b>
	Beschreibung der Laufzeitregister .....	156
	Beispielprogramm Laufzeitregister.....	158
<b>10</b>	<b>Betriebssystemupdate</b>	<b>161</b>
<b>10.1</b>	<b>Betriebssystemupdate bei einem Bediengerät</b> .....	<b>162</b>
	Betriebssystemupdate mit JetSym .....	163
	Betriebssystemupdate über \App .....	164
<b>11</b>	<b>Anwendungsprogramm</b>	<b>165</b>
	Standardablage des Anwendungsprogramms .....	166
	Anwendungsprogramm laden .....	167

---

<b>12</b>	<b>Kurzreferenz JVM-104</b>	<b>169</b>
<hr/>		
<b>Anhang</b>		<b>175</b>
<hr/>		
<b>A:</b>	<b>Schnittstellen</b> .....	<b>176</b>
	Gesamtübersicht der Anschlussbelegung .....	177
<b>B:</b>	<b>Technische Daten</b> .....	<b>179</b>
	Technische Daten.....	180
	Mechanische Abmessungen .....	182
	Betriebsparameter Umwelt und Mechanik .....	183
	Betriebsparameter EMV .....	184
<b>C:</b>	<b>Index</b> .....	<b>185</b>





---

# 1 Sicherheitshinweise

---

**Einleitung**

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

---

**Inhalt**

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

Produktlebensphase	Mindestanforderung an das Personal
<b>Außerbetriebnahme/ Entsorgung:</b>	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

<b>Thema</b>	<b>Seite</b>
Produktbeschreibung.....	14
Teile und Schnittstellen .....	15
Bestellbezeichnung .....	17
Mechanische Abmessungen.....	18

### 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<sup>2</sup>
- 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 CAN-2.0B-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
10001018	1	Bediengerät JVM-104
60879282	1	Installationsanleitung

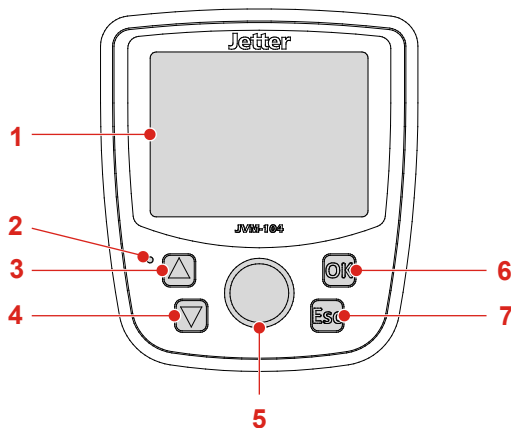
## Teile und Schnittstellen

### 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.



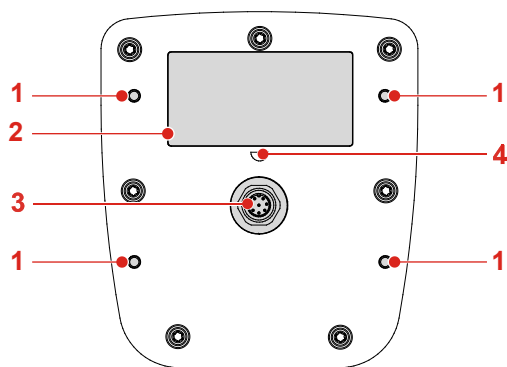
Nummer	Teil	Beschreibung
1	TFT-Display	Aktive Fläche, Touchscreen
2	Helligkeitssensor	Erfasst die Umgebungshelligkeit
3	Eingabetaste UP	Taste mit Hintergrundbeleuchtung
4	Eingabetaste DOWN	Taste mit Hintergrundbeleuchtung
5	Digipot	Drehknopf mit Taster
6	Eingabetaste OK	Taste mit Hintergrundbeleuchtung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

## 2 Produktbeschreibung und Geräteaufbau

---

### 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	M12-Stecker	
4	DAE	Druckausgleichselement

---



## Bestellbezeichnung

---

**Bestellbezeichnung**

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

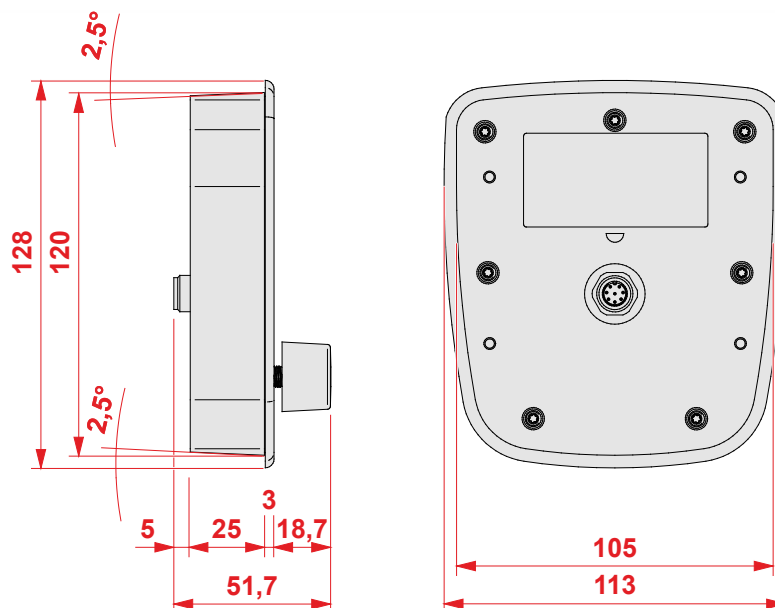
Artikel-Nr.	Bestellbezeichnung
10001018	JVM-104-K00-O01

---

### Mechanische Abmessungen

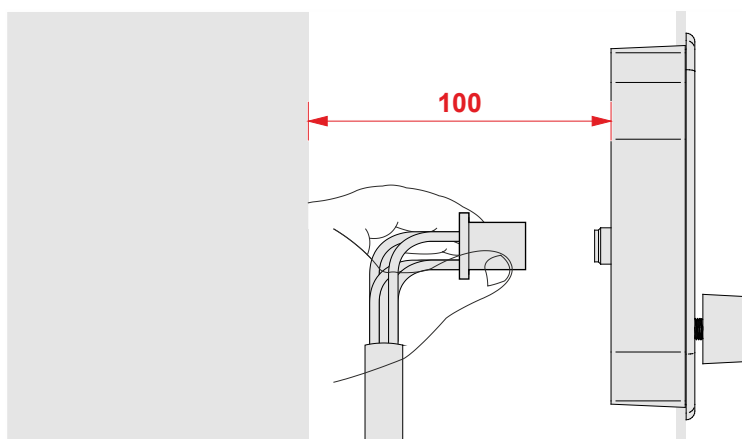
#### Mechanische Abmessungen

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.

- Der Stecker muss sich jederzeit abziehen lassen.

---

## 3 Identifikation des JVM-104

---

<b>Zweck des Kapitels</b>	Dieses Kapitel unterstützt die Identifikation des JVM-104 in folgenden Punkten: <ul style="list-style-type: none"><li>▪ Bestimmung der Hardwarerevision.</li><li>▪ Auslesen des elektronischen Typenschildes EDS. Im EDS sind zahlreiche fertigungsrelevante Daten abgelegt.</li><li>▪ Bestimmung der Softwareversionen.</li></ul>						
<b>Voraussetzungen</b>	Zur Identifikation des Bediengeräts JVM-104 müssen folgende Voraussetzungen erfüllt sein: <ul style="list-style-type: none"><li>▪ Das Bediengerät ist mit einem PC verbunden.</li><li>▪ Auf dem PC ist die Programmiersoftware JetSym ab der Version 5.1.2 installiert.</li></ul>						
<b>Infos für die Hotline</b>	Wenn Sie sich wegen eines Problems an die Hotline der Jetter AG wenden müssen, halten Sie folgende Informationen des Bediengeräts JVM-104 bereit: <ul style="list-style-type: none"><li>▪ Seriennummer</li><li>▪ Betriebssystemversion des Bediengeräts</li><li>▪ Hardwarerevision</li></ul>						
<b>Inhalt</b>	<table><thead><tr><th><b>Thema</b></th><th><b>Seite</b></th></tr></thead><tbody><tr><td>Identifikation über das Typenschild .....</td><td>20</td></tr><tr><td>Versionsregister .....</td><td>22</td></tr></tbody></table>	<b>Thema</b>	<b>Seite</b>	Identifikation über das Typenschild .....	20	Versionsregister .....	22
<b>Thema</b>	<b>Seite</b>						
Identifikation über das Typenschild .....	20						
Versionsregister .....	22						

## 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.

---

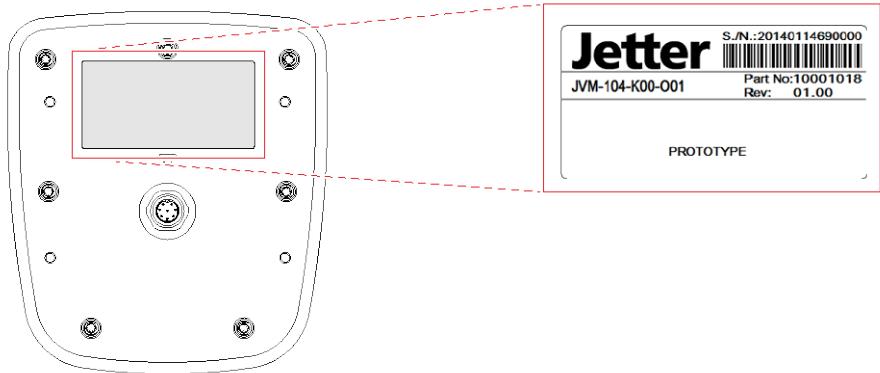
### Inhalt

Thema	Seite
Typenschild .....	21

## Typenschild

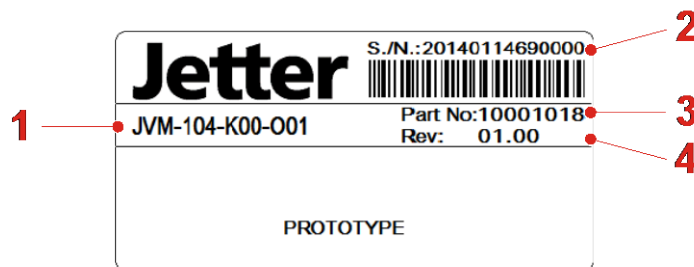
### Position des Typenschildes

Die folgende Abbildung zeigt die Position des Typenschildes 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 .....	23

## 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			





## 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**

<b>Thema</b>	<b>Seite</b>
Schnittstellen .....	26
Montage des JVM-104.....	32

## 4.1 Schnittstellen

---

### Einleitung

Das Bediengerät JVM-104 hat folgende Schnittstelle:

- M12-Stecker
- 

### M12-Stecker

Der M12-Stecker hat folgende Funktion:

- Spannungsversorgung des JVM-104
  - CANopen®-Busschnittstelle: CAN 1
  - Erkennung der Zündung
- 

### Inhalt

Thema	Seite
Beispiel einer Verdrahtung.....	27
Anschluss der Spannungsversorgung .....	28
CAN-Schnittstelle .....	30

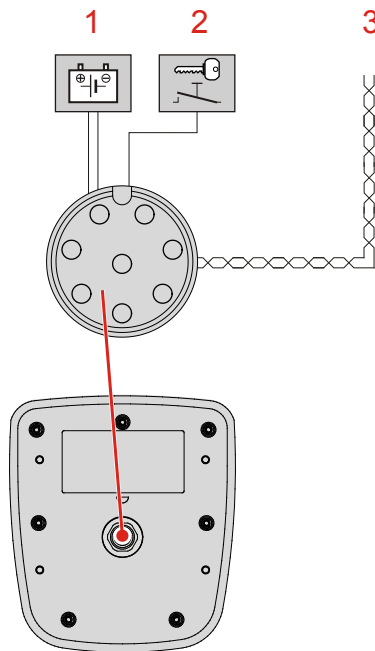
## Beispiel einer Verdrahtung

### Einleitung

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

### Beispiel

Die Abbildung zeigt ein Beispiel für eine Verdrahtung.



Nummer	Beschreibung
1	Spannungsversorgung (Batterie)
2	Zündschloss
3	CANopen®-Bus

### Anschluss der Spannungsversorgung

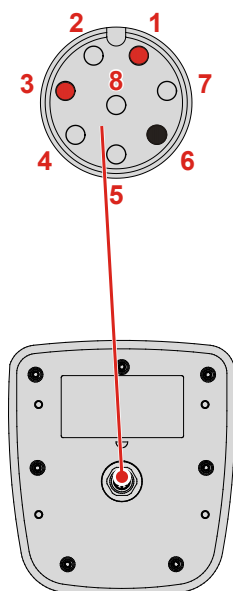
#### Verwendung des Steckers

Dieser Stecker wird auch für folgende Peripherie verwendet:

- Spannungsversorgung
- 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	Versorgungsspannung UB für die Logik des Geräts Spannung: DC12 V oder DC 24 V Stromaufnahme: Maximal 2 A	KL 30
3	Zündung (+)	KL 15
6	Bezugspotenzial GND	KL 31

**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

**Hinweis zur Zündung**


Zum Starten des JVM-104 muss Pin 3 (Zündung +) mit Pin 1 verbunden sein. Das Steuersignal Zündung (+) wird mit der Schlüsselstellung *Zündung EIN* verbunden.

**Hinweis zur Stromauf-  
nahme**

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.

**Gegenstück**

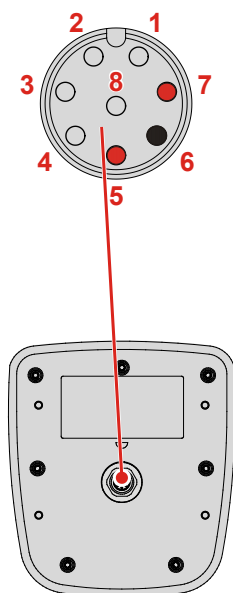
Kompatibles Gegenstück des 8-poligen M12-Steckers ist die folgende Buchse:

	Hersteller	Z. B. BELDEN Lumberg automation
	Herstellerartikelnummer	RKCN 8/9
	Anschließbarer Adernquerschnitt	0,5 mm <sup>2</sup> (AWG 20)

### CAN-Schnittstelle

#### Anschlusspins des CANopen®-Busses

Die Abbildung zeigt die Pinbelegung des Anschlusssteckers für den CANopen®-Bus (Blick auf die Kabelseite). Der Pin 6 des Bezugspotenzials ist auch farblich gekennzeichnet.



Die Pinbelegung ist wie folgt:

Pin	Beschreibung
5	CAN_L
6	Bezugspotenzial GND
7	CAN_H

#### Technische Daten - CAN-Schnittstelle

Parameter	Beschreibung
Baudrate	250 kBaud ... 1 MBaud
Busabschlusswiderstand	Keiner im Gerät verbaut
Externer Busabschluss	120 $\Omega$
Anschlusspezifikation	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 <sup>2</sup> 500 kBaud: 0,34 ... 0,50 mm <sup>2</sup> 250 kBaud: 0,34 ... 0,60 mm <sup>2</sup>
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
Aderzahl	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	Gesamtstichleitungslänge
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

**Gegenstück**

Kompatibles Gegenstück des 8-poligen M12-Steckers ist die folgende Buchse:

	Hersteller	Z. B. BELDEN Lumberg automation
	Herstellerartikelnummer	RKCN 8/9
	Anschließbarer Aderquerschnitt	0,5 mm <sup>2</sup> (AWG 20)

## 4.2 Montage des JVM-104

---

### Einleitung

Dieses Kapitel beschreibt die Montage des JVM-104.

---

### Inhalt

Thema	Seite
Montage .....	33



## 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 hitzeempfindlicher 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.

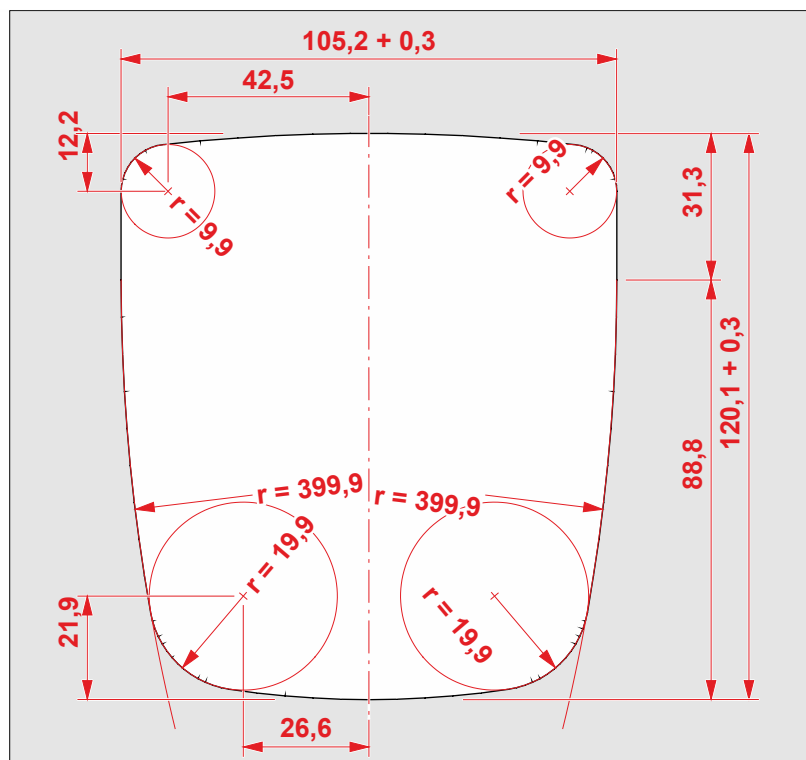
Artikel-Nr.	Anzahl	Beschreibung
60880138	1	Montage-Kit

## 4 Montage und Installation des JVM-104

---

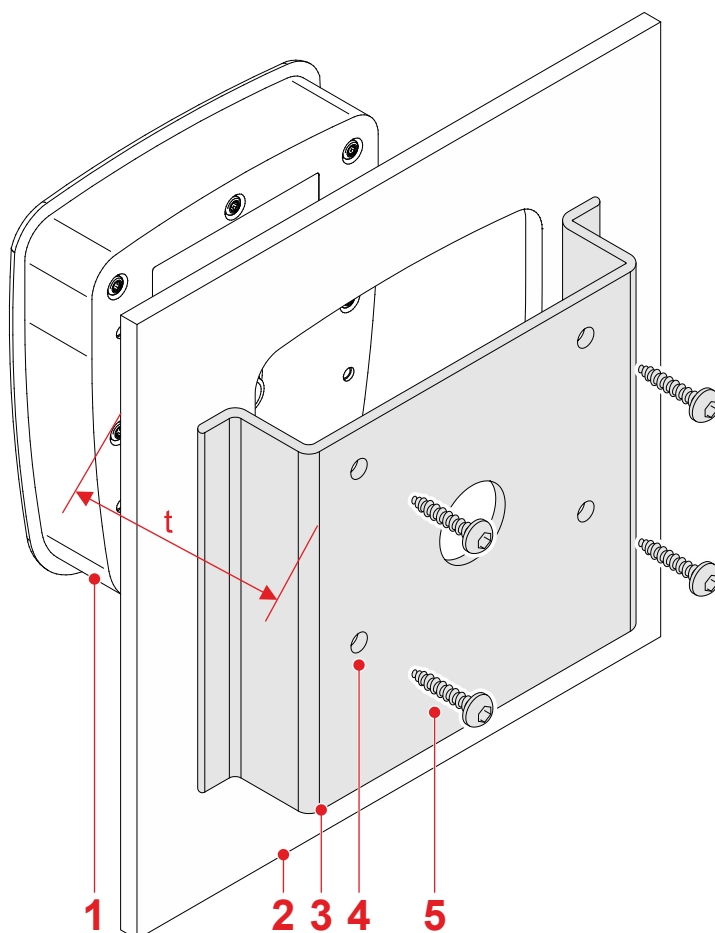
### Montage vorbereiten

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.



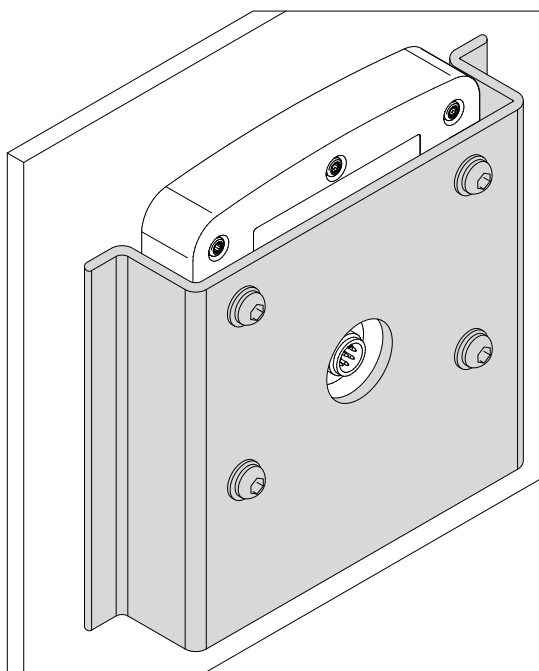
Nummer	Beschreibung
1	JVM-104
2	Montageplatte mit Öffnung für das Bediengerät
3	Winkel zur Befestigung mit Öffnung für die Stecker
4	4 x Bohrung zur Befestigung des JVM-104
5	4 x selbstfurchende Schraube Schraubenmaß: 4 x 9 + t Anzugsmoment: 1,6 Nm ± 10 % Maximale Schraubtiefe: 12 mm

## 4 Montage und Installation des JVM-104

---

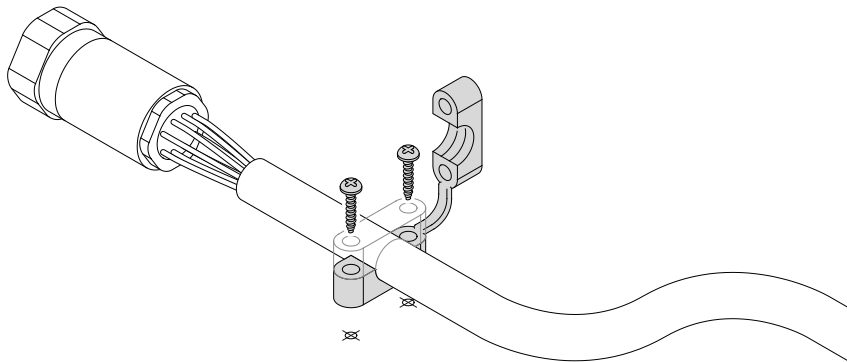
Schritt	Vorgehen
1	Stecken Sie das Bediengerät von vorn in die Öffnung der Montageplatte.
2	Halten Sie den Befestigungswinkel von hinten an die Montageplatte. Dabei müssen die Stecker durch die Öffnung des Befestigungswinkels schauen.
3	Schrauben Sie das Bediengerät zusammen mit dem Befestigungswinkel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





---

## 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 Programmierwerkzeug, 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**

<b>Thema</b>	<b>Seite</b>
Vorbereitung und Einstieg in die Programmierung mit JetSym STX.....	40
Konfiguration bei der Plattform ER-STX-CE .....	44
Programmierung bei der Plattform ER-STX-CE.....	58

## 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**

<b>Thema</b>	<b>Seite</b>
Vorbereitungen zur Erstinbetriebnahme .....	41
Programmierung in der Sprache JetSym STX.....	43



## Vorbereitungen zur Erstinbetriebnahme

### CAN-Verbindung herstellen

Zur Erstinbetriebnahme und Programmierung des Geräts JVM-104 sind folgende Vorbereitungen erforderlich:

- Verdrahtung der Spannungsversorgungen, der Zündung und der CAN-Schnittstelle
- Anschluss eines USB-CAN-Adapters zwischen Steuerung und PC
- Installation der entsprechenden Adaptertreibersoftware

Für eine Erstinbetriebnahme müssen Sie an die JVM-104 noch keine Peripherie angeschlossen haben.

### Default-Werte am CANopen®-Bus

Das Gerät JVM-104 besitzt folgende Default-Werte:

- CAN-Baudrate: 250 kBaud
- CANopen®-Node-ID: 0x7F

### Hinweis

Das Gerät JVM-104 besitzt keinen internen (zuschaltbaren) CAN-Bus-Abschlusswiderstand.

Achten Sie darauf, dass am Anfang und Ende des CAN-Busses jeweils ein Abschlusswiderstand mit je 120  $\Omega$  angeschlossen ist.

### Unterstützte USB-CAN-Adapter

Die Programmierumgebung JetSym unterstützt die folgenden USB-CAN-Adapter:

- **IXXAT Automation GmbH (<http://www.ixxat.de> <http://www.ixxat.de>):** Die Liste der aktuell unterstützten Hardware entnehmen Sie der Website von IXXAT Automation GmbH.  
Die unterstützten Treiberversionen sind: VCI-Version 3.3 und VCI-Version 2.18
- **PEAK-System Technik GmbH (<http://www.peak-system.com> <http://www.peak-system.com>):** Die Liste der aktuell unterstützten Hardware entnehmen Sie der Website von PEAK-System Technik GmbH.  
Die unterstützten Treiberversionen sind: Ab der Version 3.5.4.9547

### Installieren des USB-CAN-Adapters

#### Voraussetzungen:

Damit die Treibersoftware des USB-CAN-Adapters installiert werden kann, muss auf dem verwendeten PC **JetSym 4.3** oder höher installiert sein.

So installieren Sie den Adapter:

Schritt	Vorgehen
1	Schließen Sie den USB-CAN-Adapter an eine USB-Schnittstelle des verwendeten PCs an.
2	Wenn sich der Hardwareinstallationsassistent öffnet, beenden Sie diesen.
3	Installieren Sie den Treiber des USB-CAN-Adapters.
4	Je nach verwendetem USB-CAN-Adapter müssen Sie noch einen Treiber für JetSym installieren.

Schritt	Vorgehen	
	Wenn ...	... dann ...
	... Sie einen Adapter von PEAK-Systems verwenden,	... gehen Sie zu Schritt 5.
	... Sie keinen Adapter von PEAK-Systems verwenden,	... gehen Sie zu Schritt 7.
<b>5</b>	Wechseln Sie im Windows Explorer in den Ordner <b>PcanDrv</b> der JetSym-Installation. Standardmäßiger Pfad: <i>C:\Programme\Jetter\JetSym\Tools\PcanDrv</i>	
<b>6</b>	Führen Sie die Datei <b>PcanDrv.exe</b> aus. Folgen Sie den Installationsschritten.	
<b>7</b>	Stecken Sie den Sub-D-Stecker des Adapters in die Sub-D-Buchse der IN_CAN-Schnittstelle des JVM-104.	

**Ergebnis:** Wenn die Installation fehlerfrei funktioniert hat, ist die CANopen®-Verbindung zwischen dem PC und der Steuerung eingerichtet.

### Voraussetzung für das Einschalten

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

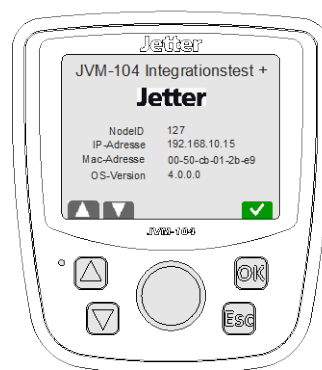
### Start ohne Anwendungsprogramm

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

<b>Thema</b>	<b>Seite</b>
Erstinbetriebnahme in JetViewSoft .....	45
Visualisierungsprojekt in JetSym anlegen und konfigurieren .....	50

## Erstinbetriebnahme in JetViewSoft

### 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 tätigen
- Visualisierungsdateien erstellen und auf das Gerät übertragen

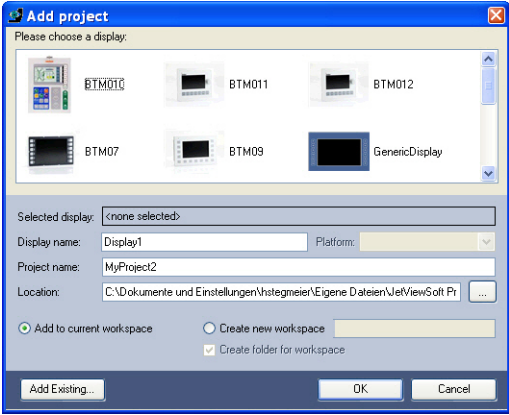
### Voraussetzungen

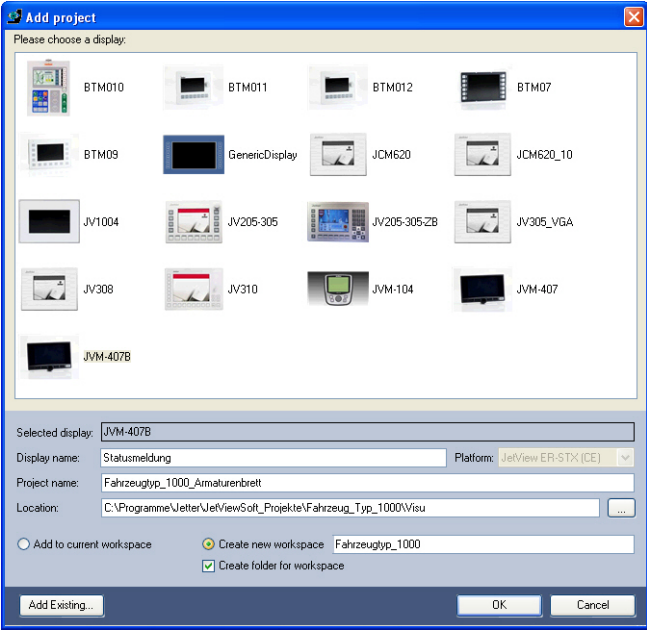
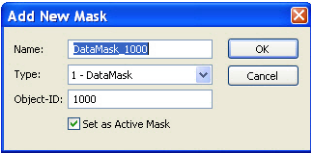
Erfüllen Sie folgende Voraussetzungen:

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

### Projekt anlegen

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

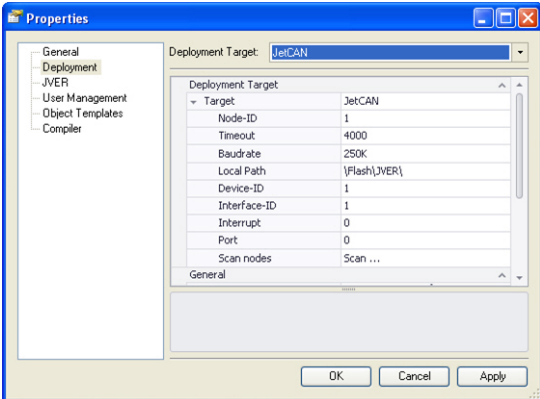
Schritt	Vorgehen
1	Starten Sie JetViewSoft
2	Öffnen Sie das Menü <b>File</b> . Wählen Sie den Menüpunkt <b>New Project</b> aus. <b>Ergebnis:</b> Der folgende Dialog öffnet sich:
	
3	Wählen Sie bei <b>Selected display</b> das passende Gerät aus, indem Sie einmal auf das Symbol des Geräts klicken.
4	Wählen Sie bei <b>Display name</b> einen programminternen Namen für das Gerät aus. Sie können in einem Projekt mehrere Displays anlegen.
5	Wenn Sie eine Auswahlmöglichkeit haben: Wählen Sie <b>Platform JetView ER-STX(CE)</b> aus.
6	Geben Sie bei <b>Project name</b> den Namen des Projekts ein.
7	Passen Sie, wenn erforderlich, bei <b>Location</b> den Pfad des Projekts an. Wenn der Pfad mit <i>Visu</i> endet, erhalten Sie eine bessere Übersicht.

Schritt	Vorgehen
<p><b>8</b></p> <p>⇒</p>	<p>Geben Sie bei <b>Workspace</b> den Namen des Workspace ein.</p> <p>Die folgende Abbildung zeigt einen möglich ausgefüllten Dialog:</p>  <p><b>Wichtiger Hinweis:</b> Die Namen dürfen keine Leerzeichen enthalten, da sich sonst die erstellten Visualisierungsdateien nur schwer wieder löschen lassen.</p>
<p><b>9</b></p>	<p>Bestätigen Sie die Einstellungen mit <b>OK</b>.</p> <p><b>Ergebnis:</b> Der Dialog schließt sich und der Dialog <b>Add New Mask</b> öffnet sich.</p> 
<p><b>10</b></p>	<p>Geben Sie bei <b>Name</b> den Namen der ersten DataMask an. Die anderen Einstellungen können Sie beibehalten. Diese Maske wird automatisch beim Start des Geräts die aktive Maske.</p>
<p><b>11</b></p>	<p>Bestätigen Sie mit <b>OK</b>.</p>

**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 <b>Project</b> . Wählen Sie den Menüpunkt <b>Properties</b> aus. <b>Ergebnis:</b> Der gleichnamige Dialog öffnet sich.
2	Öffnen Sie das Fenster <b>Deployment</b> über die Navigation auf der linken Seite des Dialogs. 
3	Wählen Sie bei <b>Deployment Target</b> den Menüpunkt <b>JetCAN</b> aus.
4	Klappen Sie die <b>Target</b> -Einstellungen über das +-Zeichen daneben auf oder mit einem Doppelklick direkt auf <b>Target</b> .
5	Geben Sie bei <b>Node-ID</b> die Node-ID des Bediengeräts ein. Default-Node-ID eines JVM-104 ist <b>0x7F</b> .
6	Geben Sie bei <b>Baudrate</b> die Baudrate ein. Default-Baudrate ist <b>250K</b> .
7	Geben Sie bei <b>Local Path</b> <code>\\App\projektname</code> ein. Dabei ist projektname ein Platzhalter für den aktuellen Namen des Projekts in Kleinbuchstaben.
8	Bestätigen Sie die Einstellungen mit <b>OK</b> .

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

## 5 Erstinbetriebnahme

---

### 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 <b>File</b> . Wählen Sie den Menüpunkt <b>Save all</b> aus.
3	Drücken Sie die Taste <b>[F7]</b> für einen Projekt-Build. <b>Ergebnis:</b> JetViewSoft erstellt Projektdateien, sofern kein Fehler auftritt.
4	Öffnen Sie das Menü <b>Build</b> . Wählen Sie den Menüpunkt <b>Deploy</b> aus. Eine andere Möglichkeit ist, die Tastenkombination <b>[CTRL] + [F5]</b> einzugeben. <b>Ergebnis:</b> 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 Dateierdung **.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	
	Wenn ...	... dann ...
1	Wenn ...	... dann ...
	... die Datei <b>\App\visual.iop</b> oder <b>\Data\visual.iop</b> vorhanden ist,	... löschen Sie die Datei oder benennen sie diese um.
2	Wenn ...	... dann ...
	... die Datei <b>\App\JetViewERS.cfg</b> 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 50)

## 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

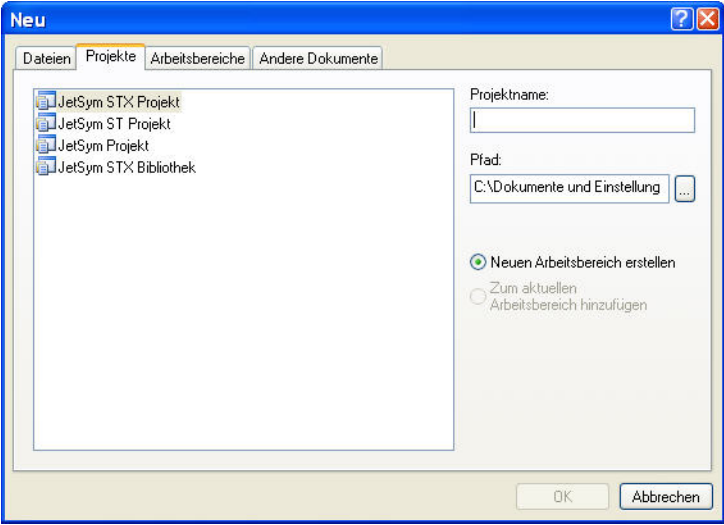
### Voraussetzungen

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 CAN-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:

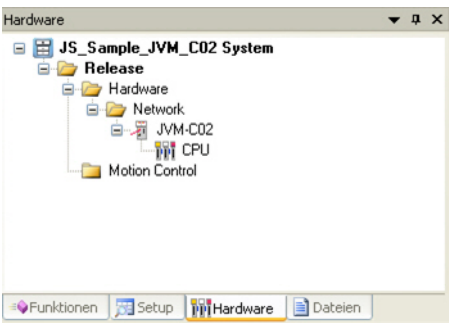
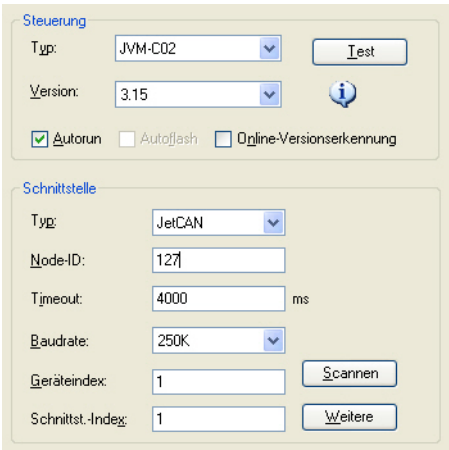
Schritt	Vorgehen
1	Starten Sie JetSym.
2	Öffnen Sie das Menü <b>Datei</b> . Wählen Sie den Menüpunkt <b>Neu</b> aus. <b>Ergebnis:</b> Der Dialog <b>Neu</b> öffnet sich.
	
3	Wählen Sie als Projekttyp <b>JetSym STX Projekt</b> aus.
4	Geben Sie den Projektnamen ein.

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

**Ergebnis:** Ein Projekt ist angelegt.

**Hardware konfigurieren**

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

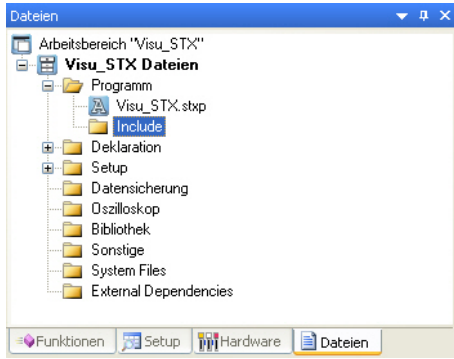
Schritt	Vorgehen
1	<p>Wechseln Sie in die Ansicht <b>Hardware</b> über die gleichnamige Registerkarte.</p> 
2	Klappen Sie den <b>Hardwarebaum</b> komplett auf.
3	<p>Wenn Sie als Hardware JVM-C02 oder Parameter zur Schnittstelle einstellen wollen, dann doppelklicken Sie auf <b>CPU</b>.</p> <p><b>Ergebnis: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

Schritt	Vorgehen
6	Geben Sie bei <b>Schnittstelle/Node-ID</b> die Node-ID des Geräts ein. Ist Ihnen die Node-ID nicht bekannt, können Sie über die Funktion <b>Hardware scannen</b> die Node-ID ermitteln.
7	Wählen Sie bei <b>Schnittstelle/Baudrate</b> 250K aus.
8	Testen Sie die Verbindung bei laufender JVER, in dem Sie die Schaltfläche <b>Test</b> drücken. Wenn keine Verbindung vorhanden ist, prüfen Sie die Node-ID und Baudrate sowie die CAN-Verbindung zur JVM-104.
9	Speichern Sie die Einstellungen mit der Tastenkombination <b>[Strg] + [S]</b> 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 <b>Dateien</b> . 
2	Klappen Sie den Ordner <b>Programm</b> auf.
3	Markieren Sie den Ordner <b>Include</b> und öffnen Sie das Kontextmenü (rechte Maustaste).
4	Wählen Sie im Kontextmenü den Eintrag <b>Dateien dem Verzeichnis hinzufügen</b> aus. <b>Ergebnis:</b> Ein Explorer-Fenster zur Auswahl einer Datei öffnet sich.
5	Navigieren Sie zu dem Ordner <b>STX</b> des JetViewSoft-Projekts. Der Ordner ist zu finden unter <i>[Projektpfad]/Name des JVS-Projekts/STX</i> zu finden.
6	Wählen Sie die Datei <b>VisualInterface.stxp</b> aus.

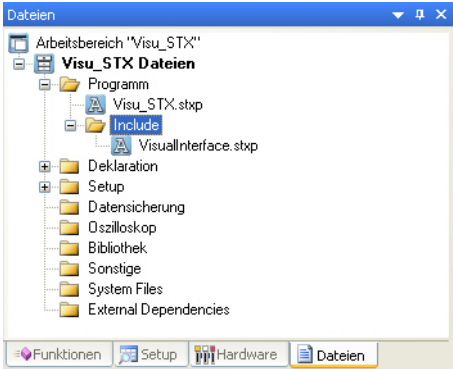
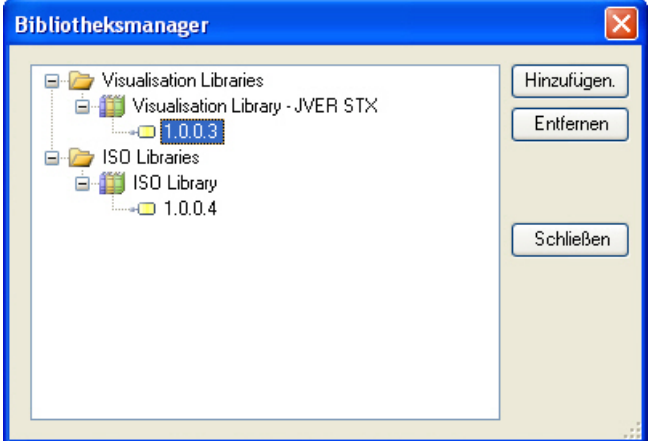
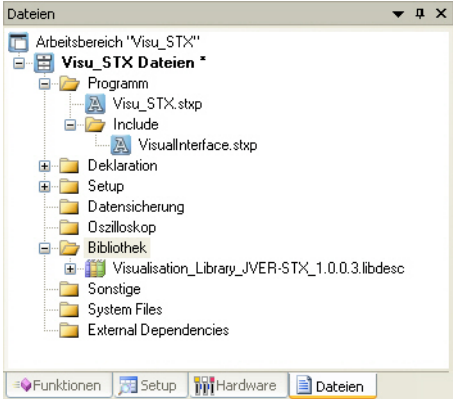
Schritt	Vorgehen
7	<p>Klicken Sie auf die Schaltfläche <b>Öffnen</b>.</p>

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

### Bibliothek einbinden

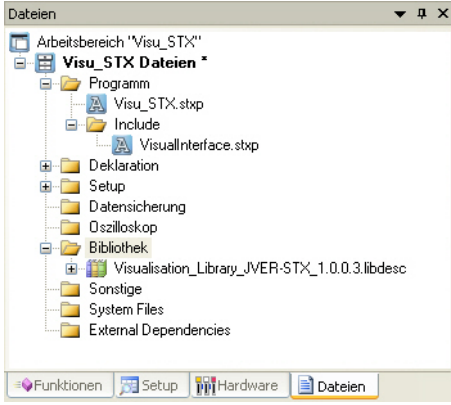
Damit die Bibliothek mit den Visualisierungsfunktionen in JetSym zur Verfügung steht, müssen Sie die Bibliothek wie folgt einbinden:

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

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

## Kompilierfähiges Programm erstellen

So erstellen und kompilieren Sie ein lauffähiges Programm:

Schritt	Vorgehen
1	<p>Wechseln Sie in die Ansicht <b>Dateien</b>.</p> 
2	<p>Doppelklicken Sie auf die Programmdatei. Die Programmdatei hat den gleichen Namen wie das Projekt plus die Endung <b>stxp</b>.</p> <p><b>Ergebnis:</b> Die Programmdatei im JetSym-Editor öffnet sich.</p>
3	<p>Geben Sie folgenden Programmcode ein. Beachten Sie dies bei der Include-Anweisung.</p> <pre>#Include "VisualInterface.stxp";  Function OnKeyDown(KeyCode:long, Flags:long) End_Function; Function OnKeyUp(KeyCode:long, Flags:long) End_Function;  Task Main Autorun End_Task;</pre>
4	<p>Klicken Sie auf die Taste <b>[F7]</b> für einen Build-Vorgang des Projekts.</p> <p><b>Ergebnis:</b> Die Visualisierungsfunktionen und die VisualInterface-Header-Datei stehen bei der Programmierung zur Verfügung.</p>

### 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_ESC: 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 45)
-

## 5.3 Programmierung bei der Plattform ER-STX-CE

---

<b>Einleitung</b>	Dieses Kapitel ist gegliedert in die folgenden zwei Bereiche: <ul style="list-style-type: none"><li>▪ Werteingabe über das Digipot am Bediengerät</li><li>▪ Visualisierungsobjekte ändern über die Visualisierungskommandos (VisuCommands) vom Anwendungsprogramm aus</li></ul>						
<b>Voraussetzung</b>	Diese Beschreibung trifft für die Plattform JetView ER-STX-CE/PC zu.						
<b>Ergänzende Literatur</b>	Ergänzende Literatur finden Sie in den Online-Hilfen der Programme JetSym und JetViewSoft.						
<b>Inhalt</b>	<table><thead><tr><th><b>Thema</b></th><th><b>Seite</b></th></tr></thead><tbody><tr><td>Werteingabe über das Digipot .....</td><td>59</td></tr><tr><td>Visualisierungsobjekt ändern über die Visualisierungskommandos.....</td><td>63</td></tr></tbody></table>	<b>Thema</b>	<b>Seite</b>	Werteingabe über das Digipot .....	59	Visualisierungsobjekt ändern über die Visualisierungskommandos.....	63
<b>Thema</b>	<b>Seite</b>						
Werteingabe über das Digipot .....	59						
Visualisierungsobjekt ändern über die Visualisierungskommandos.....	63						

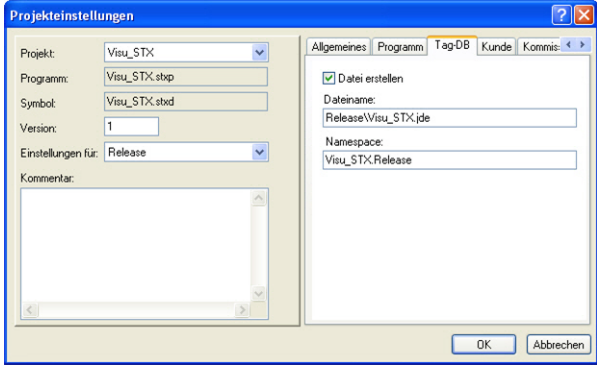
## Werteingabe über das DigiPot

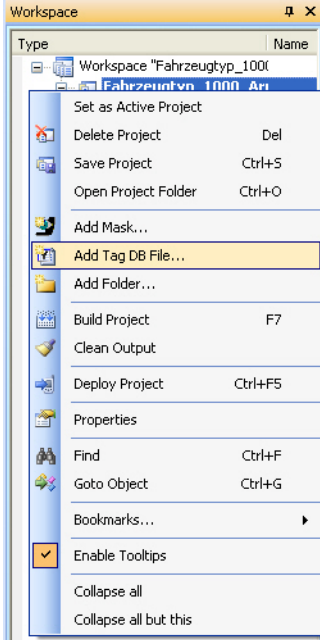
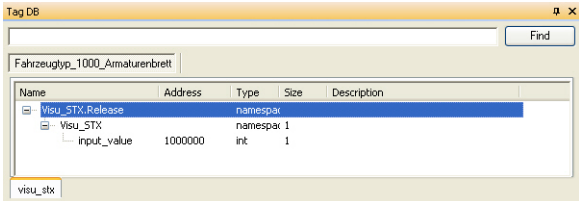
### Einleitung

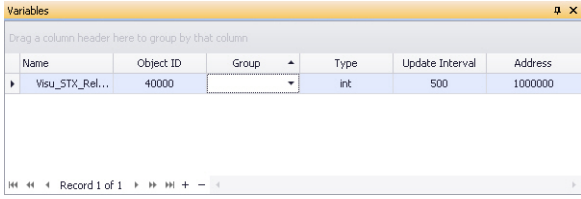
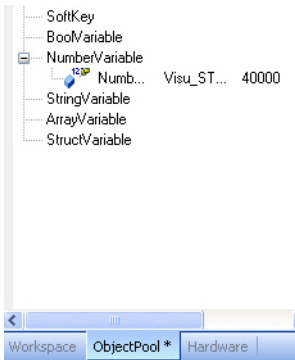
Mit nur wenig Programmcode ist es möglich, mit Hilfe des DigiPots am Bediengerät Zahlenwerte einzugeben. Diese Zahlen stehen dann in einem JetSym-STX-Programm zyklisch zur Verfügung.

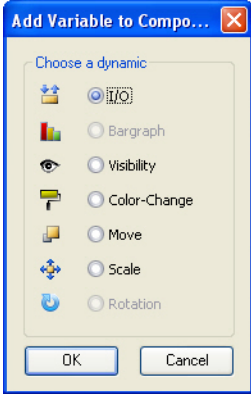
### Werteingabe über das DigiPot

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

Schritt	Vorgehen
1	<p>Deklarieren Sie die Variablen im JetSym-STX-Programm mit [export jde].</p> <pre>Task Visu_STX Autorun Var input_value: Int At %VL 1000000 [export jde]; End_Var; End_Task;</pre>
2	<p>Aktivieren Sie in JetSym im Dialog <b>Projekt - Einstellungen ...</b> auf der Registerkarte <b>TagDB</b> die Checkbox <b>Datei erstellen</b>.</p> 
3	<p>Öffnen Sie das Menü <b>Build</b>. Wählen Sie den Menüpunkt <b>Build</b> aus. Oder drücken Sie die Taste <b>[F7]</b>.</p>
⇒	<p>Beim Build-Vorgang erstellt JetSym dann eine TagDB-Datei mit der Endung <b>.jde</b>. Diese Datei enthält die Deklaration der Variablen. <b>Speicherort:</b> JetSym erstellt einen Ordner namens <b>Release</b>. Hier speichert JetSym die Datei.</p>

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

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

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

**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

---

<b>Einleitung</b>	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.
<b>Bestandteile der Visualisierungsbibliothek</b>	Alle verfügbaren Kommandos sind in der Datei <b>VISU_Functions.stxp</b> deklariert. Vordefinierte Datentypen, wie Farbe als RGB-Wert, Attribute und Tastencodes, sind in der Datei <b>VISU_Defines.stxp</b> deklariert. Beide Dateien sind Bestandteil der Visualisierungsbibliothek.
<b>Voraussetzung</b>	Damit der Compiler das folgende Programm ohne Fehler kompiliert, müssen Sie die folgenden Beispiele um den folgenden Programmcode ergänzen:  <pre>#Include "VisualInterface.stxp";  Function OnKeyDown (KeyCode:long, Flags:long) End_Function; Function OnKeyUp (KeyCode:long, Flags:long) End_Function;</pre>
<b>Aufgabe 1</b>	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.
<b>Lösung der Aufgabe 1</b>	Das Anwendungsprogramm ruft zyklisch die Funktion <b>VisuCmdAttribute()</b> auf.
<b>JetSym-STX-Programm der Aufgabe 1</b>	<pre>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;</pre>

## 5 Erstinbetriebnahme

---

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

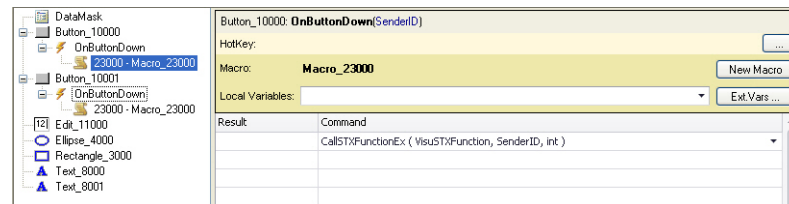
### Aufgabe 2

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

<b>Einleitung</b>	Dieses Kapitel beschreibt die STX-Funktionen der CANopen®-STX-API.
<b>Der CANopen®-Standard</b>	<p>CANopen® ist ein offener Standard für die Vernetzung und Kommunikation z. B. im Kraftfahrzeugbereich.</p> <p>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.</p>
<b>Dokumente</b>	<p>Die CANopen®-Spezifikationen können von der Homepage des <b>CiA e.V.</b> <a href="http://www.can-cia.org">http://www.can-cia.org</a> bezogen werden. Die wichtigsten Spezifikationsdokumente sind dabei:</p> <ul style="list-style-type: none"> <li>▪ CiA DS 301 - Dieses Dokument ist auch als Kommunikationsprofil bekannt und beschreibt die grundlegenden Dienste und Protokolle, die unter CANopen® verwendet werden.</li> <li>▪ CiA DS 302 - Framework für programmierbare Geräte (CANopen®-Manager, SDO-Manager)</li> <li>▪ CiA DR 303 - Informationen zu Kabeln und Steckverbindern</li> <li>▪ CiA DS 4xx - Diese Dokumente beschreiben das Verhalten vieler Geräteklassen über sogenannte Geräteprofile.</li> </ul>
<b>Anwendung</b>	Diese STX-Funktionen werden in der Kommunikation zwischen dem Gerät JVM-104 und anderen CANopen®-Teilnehmern angewendet.
<b>Hinweis: Perspektive beachten!</b>	<p>Dieses Kapitel ist aus Sicht der übergeordneten Steuerung beschrieben, während im Dokument CiA DS 301 aus Gerätesicht dokumentiert ist.</p> <p>Deshalb benötigen Sie z. B. zum Aufrufen der Funktion <code>CanOpenAddPDORx()</code> ein <code>PDO_RX</code>-Makro.</p>
<b>Begriffe und Abkürzungen</b>	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

<b>Thema</b>	<b>Seite</b>
STX-Funktion CanOpenInit() .....	67
STX-Funktion CanOpenSetCommand() .....	69
STX-Funktion CanOpenUploadSDO() .....	71
STX-Funktion CanOpenDownloadSDO() .....	76
STX-Funktion CanOpenAddPDORx() .....	81
STX-Funktion CanOpenAddPDOTx() .....	88
Heartbeat-Überwachung .....	94
CANopen®-Objektverzeichnis des JVM-104 .....	98

## 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-Identifizier (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 <b>Pre-Operational</b> geht, sendet es die Heartbeat-Nachricht <b>Pre-Operational</b> .

---

**Zugriff auf das Objektverzeichnis** 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 69)
-

## 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. <i>Auswahl der Synchronisierung:</i> 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)

<b>Hinweis</b>	<p>Die Auswahl des Kommandos CAN_CMD_NMT erfolgt über die Makrofunktion CAN_CMD_NMT_Value(NodeID, CAN_CMD_NMT).</p> <p>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.</p>						
<b>CANNo-Parameter</b>	<p>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 <i>technischen Daten</i> und der <i>Kurzreferenz</i> der jeweiligen Betriebsanleitung erwähnt.</p>						
<b>Rückgabewert</b>	<p>Die Funktion übergibt die folgenden Rückgabewerte an das übergeordnete Programm.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th colspan="2" style="text-align: left;">Rückgabewert</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td>ok</td> </tr> <tr> <td style="text-align: center;">-1</td> <td>Fehler bei der Parameterüberprüfung Kommando nicht bekannt</td> </tr> </tbody> </table>	Rückgabewert		0	ok	-1	Fehler bei der Parameterüberprüfung Kommando nicht bekannt
Rückgabewert							
0	ok						
-1	Fehler bei der Parameterüberprüfung Kommando nicht bekannt						
<b>Verwenden der Funktion (Beispiel 1)</b>	<p>Der eigene Heartbeat-Zustand soll auf <b>Operational</b> gesetzt werden.</p> <pre>Result := CanOpenSetCommand(0, CAN_CMD_HEARTBEAT, CAN_HEARTBEAT_OPERATIONAL);</pre>						
<b>Verwenden der Funktion (Beispiel 2)</b>	<p>Der eigene Heartbeat-Zustand und der Zustand von allen anderen Geräten am CAN-Bus soll auf <b>Operational</b> gesetzt werden.</p> <pre>Result := CanOpenSetCommand(0, CAN_CMD_NMT, CAN_CMD_NMT_Value(CAN_CMD_NMT_ALLNODES, CAN_NMT_OPERATIONAL));</pre>						
<b>Verwenden der Funktion (Beispiel 3)</b>	<p>Der Heartbeat-Zustand von dem Gerät mit der Node-ID 60 (0x3C) soll auf <b>Operational</b> gesetzt werden.</p> <pre>Result := CanOpenSetCommand(0, CAN_CMD_NMT, CAN_CMD_NMT_Value(60, CAN_NMT_OPERATIONAL));</pre>						
<b>Verwenden der Funktion (Beispiel 4)</b>	<p>Die Synchronisierung der Uhrzeit über den CAN-Bus (CAN-ID 0x100) soll eingeschaltet werden.</p> <pre>Result := CanOpenSetCommand(0, CAN_CMD_TIME_CONSUMER, CAN_TIME_CONSUMER_ENABLE);</pre>						
<b>Verwenden der Funktion (Beispiel 5)</b>	<p>Die Uhrzeit soll auf dem CAN-Bus veröffentlicht werden.</p> <pre>Result := CanOpenSetCommand(0, CAN_CMD_TIME_PRODUCER, CAN_TIME_PRODUCER_SEND);</pre>						

## 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 Nachrichtenempfängers	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 <b>Stopp</b> (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.

**Data Type-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
    CANNNo = 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(CANNNo,           // 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(CANNNo, 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(CANNNo, // 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-Identifizier
    // 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,
    // Mindestabstand 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-Identifizier 11 Bit CAN-Identifizier 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-Identifizier übergeben. Der CAN-Identifizier wird mit einem Makro erstellt. Der CAN-Identifizier 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-Identifizier ist: 2C0h = 40h + 280h

**Default-CAN-Identifizier-Verteilung**

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

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

**Data-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\_ASYNC\_PDORTRONLY**

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\_ASYNC\_PDO**

Empfange asynchrone PDO.

**CANOPEN\_PDOWINVALID**

PDO wird nicht empfangen. Speicherplatz wird reserviert.

**CANOPEN\_NORTR**

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

Nur wenn ein CANOPEN\_ASYNC\_PDORTRONLY gesetzt ist, wird ein RTR gesendet.

**CANOPEN\_29BIT**

Verwende 29 Bit-Identifizier

Default: 11 Bit-Identifizier

**Verwenden der Funktion**

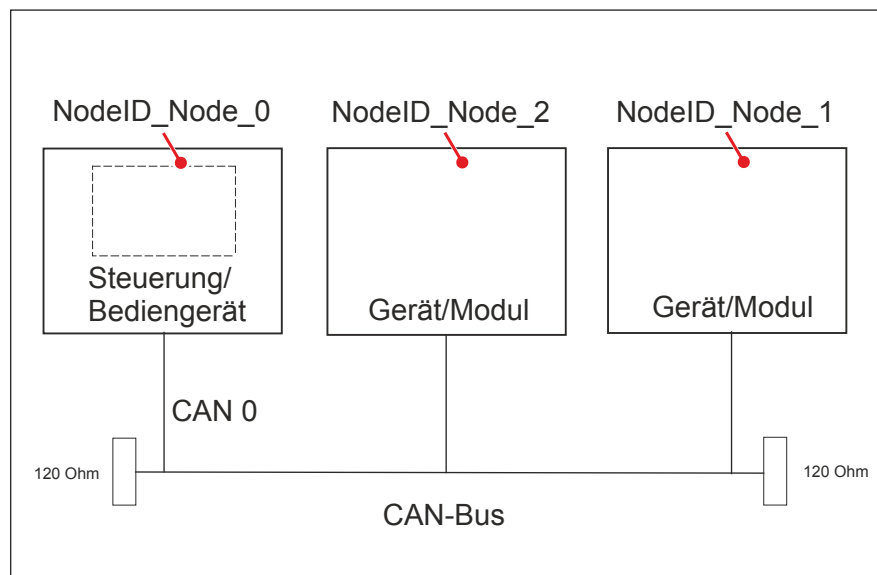
```

Result := CanOpenAddPDORx (
    0, // CANNNo
    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
    CANNNo = 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_ASYNC_PDORTRONLY); // 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_ASYNC_PDORTRONLY); // Paramset
```

```
CanOpenAddPDORx (  
    CANNNo,                                // 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(CANNNo, 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 (
    CANNNo: Int,           // Nummer der Busleitung
    CANID: Int,           // CAN-Identifizier
    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
CANNNo	CAN-Nummer der Busleitung	0 ... CANMAX
CANID	CAN-Identifizier 11 Bit CAN-Identifizier 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-Identifizierer übergeben. Der CAN-Identifizierer wird mit einem Makro erstellt. Der CAN-Identifizierer 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-Identifizierer ist: 2C0h = 40h + 280h

**Default-CAN-Identifizierer-Verteilung**

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

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

**Data Type-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\_ASYNC\_PDORTRONLY**

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

#### **CANOPEN\_ASYNC\_PDO**

Sende asynchrone PDO.

#### **CANOPEN\_PDINVALID**

PDO wird nicht gesendet. Der benötigte Speicherplatz wird reserviert.

#### **CANOPEN\_NORTR**

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

#### **CANOPEN\_29BIT**

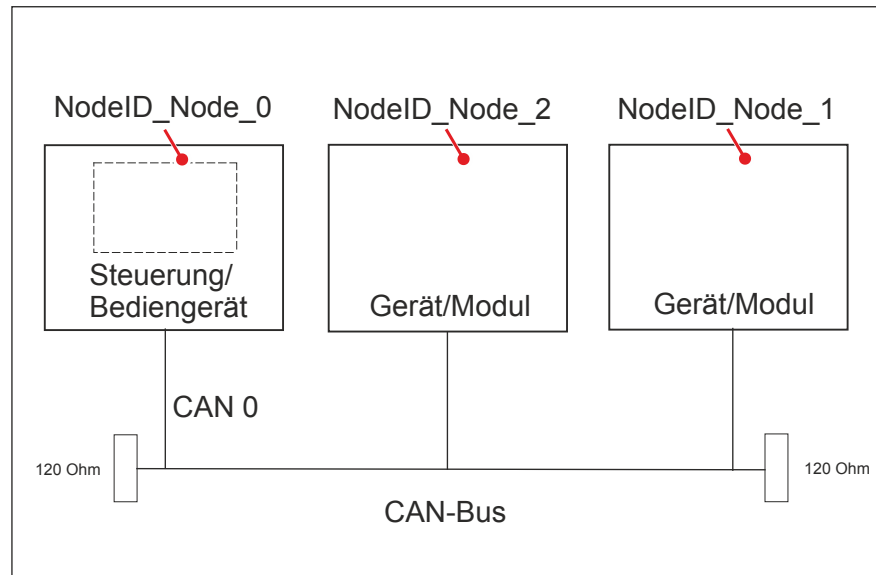
Verwende 29 Bit-Identifizier  
Default: 11 Bit-Identifizier

### 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_ASYNC_PDO | 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 (
    CANNNo, // 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 (
    CANNNo, // 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 (
    CANNNo, // 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(CANNNo, 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)

---

Registernummer	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

---



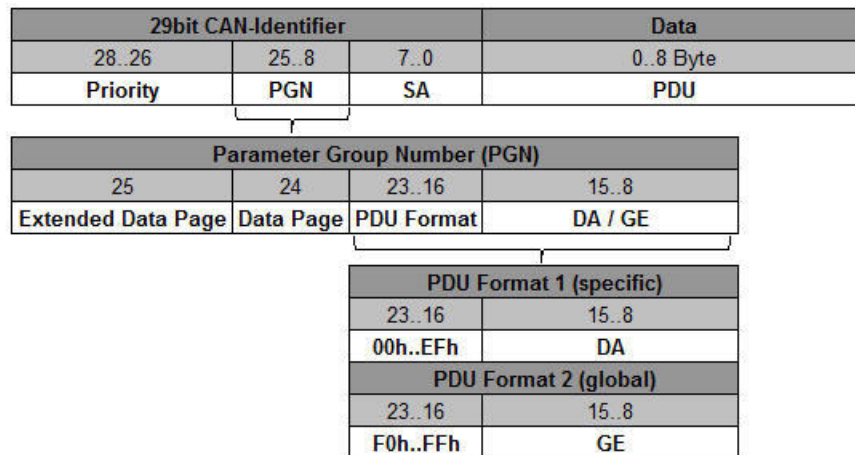
## 7 SAE J1939-STX-API

<b>Einleitung</b>	Dieses Kapitel beschreibt die STX-Funktionen der SAE J1939-STX-API.																								
<b>Der SAE J1939-Standard</b>	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.																								
<b>Anwendung</b>	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.																								
<b>Dokumente</b>	Die wichtigsten SAE J1939-Spezifikationen sind: <ul style="list-style-type: none"> <li>▪ J1939-11 - Informationen zum Physical Layer</li> <li>▪ J1939-21 - Informationen zum Data Link Layer</li> <li>▪ J1939-71 - Informationen zum Application Layer Fahrzeuge</li> <li>▪ J1939-73 - Informationen zum Application Layer Bereich Diagnose</li> <li>▪ J1939-81 - Netzwerkmanagement</li> </ul>																								
<b>Inhalt</b>	<table border="0" style="width: 100%;"> <thead> <tr> <th style="text-align: left;"><b>Thema</b></th> <th style="text-align: right;"><b>Seite</b></th> </tr> </thead> <tbody> <tr> <td>Aufbau einer J1939-Nachricht .....</td> <td style="text-align: right;">104</td> </tr> <tr> <td>STX-Funktion SAEJ1939Init() .....</td> <td style="text-align: right;">106</td> </tr> <tr> <td>STX-Funktion SAEJ1939SetSA() .....</td> <td style="text-align: right;">107</td> </tr> <tr> <td>STX-Funktion SAEJ1939GetSA().....</td> <td style="text-align: right;">108</td> </tr> <tr> <td>STX-Funktion SAEJ1939AddRx() .....</td> <td style="text-align: right;">109</td> </tr> <tr> <td>STX-Funktion SAEJ1939AddTx().....</td> <td style="text-align: right;">112</td> </tr> <tr> <td>STX-Funktion SAEJ1939RequestPGN().....</td> <td style="text-align: right;">115</td> </tr> <tr> <td>STX-Funktion SAEJ1939GetDM1().....</td> <td style="text-align: right;">118</td> </tr> <tr> <td>STX-Funktion SAEJ1939GetDM2().....</td> <td style="text-align: right;">121</td> </tr> <tr> <td>STX-Funktion SAEJ1939SetSPNConversion() .....</td> <td style="text-align: right;">124</td> </tr> <tr> <td>STX-Funktion SAEJ1939GetSPNConversion().....</td> <td style="text-align: right;">125</td> </tr> </tbody> </table>	<b>Thema</b>	<b>Seite</b>	Aufbau einer J1939-Nachricht .....	104	STX-Funktion SAEJ1939Init() .....	106	STX-Funktion SAEJ1939SetSA() .....	107	STX-Funktion SAEJ1939GetSA().....	108	STX-Funktion SAEJ1939AddRx() .....	109	STX-Funktion SAEJ1939AddTx().....	112	STX-Funktion SAEJ1939RequestPGN().....	115	STX-Funktion SAEJ1939GetDM1().....	118	STX-Funktion SAEJ1939GetDM2().....	121	STX-Funktion SAEJ1939SetSPNConversion() .....	124	STX-Funktion SAEJ1939GetSPNConversion().....	125
<b>Thema</b>	<b>Seite</b>																								
Aufbau einer J1939-Nachricht .....	104																								
STX-Funktion SAEJ1939Init() .....	106																								
STX-Funktion SAEJ1939SetSA() .....	107																								
STX-Funktion SAEJ1939GetSA().....	108																								
STX-Funktion SAEJ1939AddRx() .....	109																								
STX-Funktion SAEJ1939AddTx().....	112																								
STX-Funktion SAEJ1939RequestPGN().....	115																								
STX-Funktion SAEJ1939GetDM1().....	118																								
STX-Funktion SAEJ1939GetDM2().....	121																								
STX-Funktion SAEJ1939SetSPNConversion() .....	124																								
STX-Funktion SAEJ1939GetSPNConversion().....	125																								

## 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):

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



<b>Startposition</b>	<b>Länge</b>	<b>Parametername</b>	<b>SPN</b>
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 <code>VarAddr</code>	
VarAddr	Globale Variable, in die der empfangene Wert eingetragen wird	
TJ1939Rx	Kontrollstruktur	
EventTime	Zeitlicher Abstand zw. zwei Telegrammen (> <code>InhibitTime</code> )	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 (
    CANNNo: 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
CANNNo	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.

**Data Type-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 (
    CANNNo: Int,
    bySA: Byte,
    ref stJ1939DM1stat: TJ1939DM1STAT
    ref stJ1939DM1msg: TJ1939DM1MSG
) : Int;
```

### Funktionsparameter

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

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

### Rückgabewert

Die Funktion übergibt die folgenden Rückgabewerte an das übergeordnete Programm.

#### Rückgabewert

0	ok
-1	Fehler bei der Parameterüberprüfung

### CANNNo-Parameter

Dieser Parameter gibt die Nummer der SAEJ1939-Schnittstelle an. Die erste Schnittstelle wird mit `CANNNo = 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.IStatus**      **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,
    stdmlstat_pow,
    stdmlmsg_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;
    stdmlstat_pow : TJ1939DM1STAT;
    stdmlmsg_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
stdmlstat_pow.lBuffer := sizeof (stdmlmsg_pow);
SAEJ1939GetDM1 (bySAEJ1939Channel, 0x00, stdmlstat_pow,
stdmlmsg_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 (
    CANNNo: Int,
    bySA: Byte,
    ref stJ1939DM2stat: TJ1939DM2STAT
    ref stJ1939DM2msg: TJ1939DM2MSG
) : Int;
```

### Funktionsparameter

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

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

### Rückgabewert

Die Funktion übergibt die folgenden Rückgabewerte an das übergeordnete Programm.

#### Rückgabewert

0	ok
-1	Fehler bei der Parameterüberprüfung

### CANNNo-Parameter

Dieser Parameter gibt die Nummer der SAEJ1939-Schnittstelle an. Die erste Schnittstelle wird mit `CANNNo = 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.IStatus****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. 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 löscherbar. 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**

<b>Thema</b>	<b>Seite</b>
Verzeichnisse.....	128
Eigenschaften .....	132

## 8.1 Verzeichnisse

---

### Systemverzeichnisse

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

Verzeichnis	Beschreibung
\System	<ul style="list-style-type: none"><li>▪ Systemkonfiguration</li><li>▪ Systeminformationen</li><li>▪ Begrüßungsbild (Boot-Image)</li><li>▪ Screenshot</li></ul>
\App	<ul style="list-style-type: none"><li>▪ Speicherbereich für die Applikation</li></ul>
\Data	<ul style="list-style-type: none"><li>▪ Speicherbereich für die Daten</li></ul>
\Windows	<ul style="list-style-type: none"><li>▪ Systemverzeichnis von Windows CE</li></ul>
\	<ul style="list-style-type: none"><li>▪ RAM-Disk</li></ul>

### Inhalt

Thema	Seite
Verzeichnisse .....	129



## Verzeichnisse

### Verzeichnis \App

#### \App

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

Name	Erw.
[..]	
[jxmioe2test]	
[sys]	
autostart	exe
autostart	ini
CRC	DAT
JetViewERS	cfg
JVER	JRT
JVER	XML
jverdevicece	xml
myproject	jhw
myproject	jor
myproject	jsr
START	INI
update	dat
updatelog	txt

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.

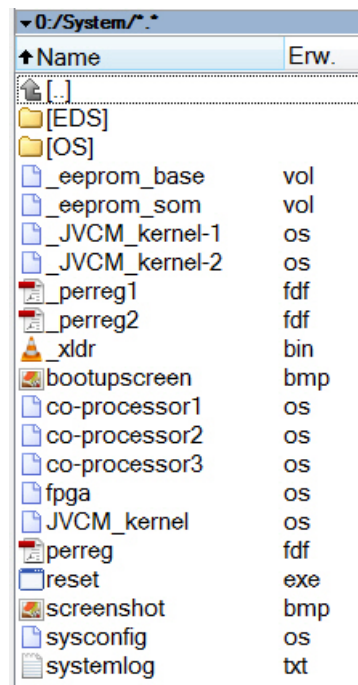
#### Wichtiger Hinweis!

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 *Windows******Windows***

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

---

## 8.2 Eigenschaften

---

### Einleitung

Dieses Kapitel beschreibt die Eigenschaften des Dateisystems auf der internen Flash-Disk.

### Allgemeine Eigenschaften

Für die interne Flash-Disk 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

<b>Thema</b>	<b>Seite</b>
Eigenschaften der Flash-Disk .....	133

## Eigenschaften der Flash-Disk

---

### Größe

Dem Anwender steht folgende Größe zur freien Verfügung:

Parameter	Wert
Größe der Flash-Disk	512 MByte
Davon Ordner <b>App</b>	64 MByte
Davon Ordner <b>Data</b>	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.
-



---

## 9 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**

<b>Thema</b>	<b>Seite</b>
Abkürzungen, Modulregistereigenschaften und Formatierungen .....	136
Speicherübersicht .....	137
Bedienelemente und Zündung .....	148
Laufzeitregister .....	155

## 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
<code>Var, When, Task</code>	Schlüsselwort
<code>BitClear();</code>	Befehle
<code>100 0x100 0b100</code>	Konstante Zahlenwerte
<code>// dies ist ein Kommentar</code>	Kommentar
<code>// ...</code>	Weitere Programmbearbeitung



---

## 9.1 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

<b>Thema</b>	<b>Seite</b>
Speicher des Betriebssystems .....	138
Speicher des Dateisystems .....	139
Speicher des Anwendungsprogramms .....	140
Speicher für flüchtige Variablen des Anwendungsprogramms .....	141
Speicher für nichtflüchtige Register des Anwendungsprogramms .....	142
Speicher für nichtflüchtige Variablen des Anwendungsprogramms .....	143
Spezialregister .....	145
Merker .....	146

### 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 162)
-

## Speicher des Dateisystems

---

<b>Einleitung</b>	Im Speicher des Dateisystems werden Daten- und Programmdateien abgelegt.
<b>Eigenschaften</b>	<ul style="list-style-type: none"><li>▪ Nichtflüchtig</li><li>▪ Größe der internen Flash-Disk: 368 MByte</li></ul>
<b>Speicherzugriff</b>	<ul style="list-style-type: none"><li>▪ Durch das Betriebssystem</li><li>▪ Durch JetSym</li><li>▪ Über Dateibefehle aus dem Anwendungsprogramm</li></ul>

---

### 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 möglich
- 

#### Speicherzugriff

- Durch das Betriebssystem
  - Durch JetSym
  - Über Dateibefehle aus dem Anwendungsprogramm
- 

#### Verwandte Themen

- **Anwendungsprogramm** (siehe Seite 165)
-

## 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.

	Name	Nummer	Inhalt	Typ
1	Count		1575	
2				
3				

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
  - 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.

	Name	Nummer	Inhalt	Typ
4	ProgramStartCounter	1000000	4	
5				
6				

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.

	Name	Nummer	Inhalt	Typ
6	Cnt1	1000010	2	
7	Cnt2	1000010	4	
8	Cnt3	1000010	10	
9	Cnt4	1000010	20	

Red arrows point from the 'Inhalt' column to the 'Typ' column, indicating the data type (Byte) for each variable.

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

---

<b>Einleitung</b>	Merker belegen ein Bit im Speicher und können den Wert TRUE oder FALSE annehmen.
<b>Eigenschaften Anwendungsmerker</b>	<ul style="list-style-type: none"><li>▪ Globale Variablen, die fest Adressen zugeordnet sind (%MX)</li><li>▪ Nicht flüchtig</li><li>▪ Anzahl: 256</li><li>▪ Merkernummern: 0 ... 255</li></ul>
<b>Eigenschaften überlagerter Anwendungsmerker</b>	<ul style="list-style-type: none"><li>▪ Globale Variablen, die fest Adressen zugeordnet sind (%MX)</li><li>▪ Nicht flüchtig</li><li>▪ Überlagert mit den Registern 1000000 ... 1000055</li><li>▪ Anzahl: 1.792</li><li>▪ Merkernummern: 256 ... 2047</li></ul>
<b>Eigenschaften Spezialmerker</b>	<ul style="list-style-type: none"><li>▪ Globale Variablen, die fest Adressen zugeordnet sind (%MX)</li><li>▪ Spezialmerker werden beim Start des Betriebssystems mit Standardwerten vorbesetzt</li><li>▪ Anzahl: 256</li><li>▪ Merkernummern: 2048 ... 2303</li></ul>
<b>Speicherzugriff</b>	<ul style="list-style-type: none"><li>▪ Durch JetSym</li><li>▪ Von Anzeige- und Bediengeräten</li><li>▪ Aus dem Anwendungsprogramm</li></ul>
<b>JetSym-STX-Programm</b>	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;
    End_Loop;
```

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

---

## 9.2 Bedienelemente und Zündung

---

### Einleitung

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

---

### Inhalt

<b>Thema</b>	<b>Seite</b>
Eingabetasten .....	149
Digipot .....	151
Zündung und Ausschaltverzögerung .....	153

## 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	<b>Eingabetaste [OK]</b> Bit 0 = 1: Taste <b>[OK]</b> ist gedrückt.
361000.1	<b>Eingabetaste [ESC]</b> Bit 1 = 1: Taste <b>[ESC]</b> ist gedrückt.
361000.2	<b>Eingabetaste [DOWN]</b> Bit 2 = 1: Taste <b>DOWN</b> ist gedrückt.
361000.3	<b>Eingabetaste [UP]</b> Bit 3 = 1: Taste <b>[UP]</b> 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: <ul style="list-style-type: none"> <li>▪ <b>Digipot im Uhrzeigersinn drehen</b> = Register hochzählen</li> <li>▪ <b>Digipot gegen Uhrzeigersinn drehen</b> = Register abwärts zählen</li> </ul>
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;
  
```

## 9 Programmierung

---

```
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;
```

---

---

## 9.3 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**

<b>Thema</b>	<b>Seite</b>
Beschreibung der Laufzeitregister .....	156
Beispielprogramm Laufzeitregister .....	158

## 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

---

<b>Aufgabe</b>	Messen Sie, wie lange das Abspeichern von Variablenwerten in einer Datei dauert.
<b>Lösung</b>	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.
<b>Softwareversionen</b>	Das Beispielprogramm ist getestet mit den folgenden Softwareversionen: <ul style="list-style-type: none"><li>▪ JetSym in der Version 5.2</li><li>▪ Bediengerät JVM-104 in der OS-Version 4.01</li></ul> 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
        // Startmarker zurücksetzen
        WriteIt := False;
        // Warten bis Anwender den Startmarker 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 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.....	162

## 10.1 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 Programmierwerkzeug JetSym
  - Über das Verzeichnis \App
- 

### Inhalt

<b>Thema</b>	<b>Seite</b>
Betriebssystemupdate mit JetSym .....	163
Betriebssystemupdate über \App.....	164

## 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 CAN mit dem PC verbunden.
- In JetSym ist eingestellt:
  - Gerätetyp
  - Schnittstellentyp
  - Node-ID
  - CAN-Baudrate
- Während des Betriebssystemupdates das Gerät eingeschaltet lassen.

### Betriebssystem updaten

Führen Sie zum Betriebssystemupdate folgende Schritte aus:

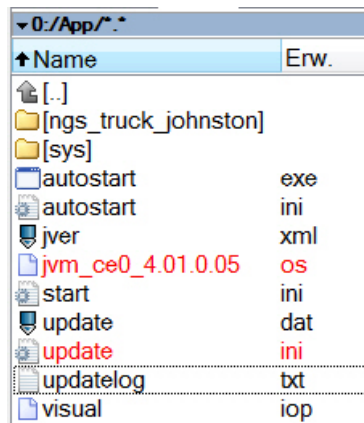
Schritt	Vorgehen
1	Wählen Sie in JetSym im Menü <b>Build</b> den Punkt <b>OS updaten...</b> aus. <b>Ergebnis:</b> Ein Dateiauswahlfenster öffnet sich.
2	Wählen Sie die gewünschte Betriebssystemdatei aus. <b>Ergebnis:</b> JetSym öffnet ein Bestätigungsfenster.
3	Starten Sie die Übertragung der Betriebssystemdatei durch Klicken auf die Schaltfläche <b>Ja</b> .
4	Warten Sie, bis das Update beendet ist.
5	Um das eingespielte Betriebssystem zu starten, booten Sie das Gerät neu.

## 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 <b>update.ini</b> . Sonst funktioniert das Update nicht.
2	Kopieren Sie die Filecollection oder OS und die Datei <b>update.ini</b> in das Verzeichnis \App.
3	Starten Sie das Bediengerät neu.
⇒	<b>Autostart.exe</b> findet das Update während dem Booten, installiert alle Dateien und führt einen Neustart aus. <b>Wichtig!</b> Unterbrechen Sie nicht diesen Vorgang.
4	Öffnen Sie die Datei <b>updatelog.txt</b> und schauen Sie, ob das Update erfolgreich war.



# 11 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**

<b>Thema</b>	<b>Seite</b>
Standardablage des Anwendungsprogramms.....	166
Anwendungsprogramm laden.....	167

## 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 <i>\App</i>
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 <b>\App\start.ini</b> von der internen Flash-Disk.
2	Das Betriebssystem wertet den Eintrag <b>Project</b> aus. Er enthält den Pfad zur Anwendungsprogrammdatei.
3	Das Betriebssystem wertet den Eintrag <b>Program</b> aus. Er enthält den Programmnamen.
4	Das Betriebssystem lädt das Anwendungsprogramm aus der Datei <b>&lt;Project&gt;\&lt;Program&gt;</b> .

---





# 12 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: 1  
**CANMAX: 0**

## Mögliche Anzahl an SAEJ1939-Schnittstellen

Mögliche Anzahl an CAN-Schnittstellen: 0  
**CANMAX: 0**

## 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

<b>Eingabetasten</b>	
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

## 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

## 12 Kurzreferenz JVM-104

### 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 ...	JetVM-DII-Version-String
108509	
108510 ...	Versionsstring der Hostapplikation
108519	
108520 ...	Dateiname der Hostapplikation
108529	
108530 ...	OS-Version (String)
108539	
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
203100 ...	32-Bit-Überlagerung Flag 0 ... 255
203107	
203108 ...	16-Bit-Überlagerung Flag 0 ... 255
203123	
203124 ...	32-Bit-Überlagerung Flag 2048 ... 2303
203131	
203132 ...	16-Bit-Überlagerung Flag 2048 ... 2303
203147	
209700	Systemlogger: Globale Freigabe
209701 ...	Freigabe Systemkomponenten
209739	

### 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 ...	Task-Zustand
210199	

210400 ...	Task-Programmadresse
210499	

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 ...	16-Bit-I/O-Register überlagert mit virtuellen I/O 20001 bis
278999	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 ...	32-Bit-Ganzzahl (remanent)
1005999	

### 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	Datenbytes
...	
200010521	

### 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

## 12 Kurzreferenz JVM-104

---

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

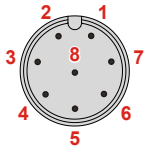
## JetSym-STX-Funktionen

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: STXBASETTYPE, 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: TNetCopyListL): Int;
151	Function NetCopyListSend(Handle: Int): Int;
152	Function NetCopyListDelete(Handle: Int): Int;

## 12 Kurzreferenz JVM-104

---

### Belegung 8-polig M12-Stecker



Pin	Funktion
1	Versorgungsspannung UB für die Logik des Geräts Spannung: DC 12 V oder DC 24 V Maximaler Strom: 2 A
2	Frei
3	Zündung (+)
4	Frei
5	CAN_L
6	Bezugspotential GND
7	CAN_H
8	Abschirmung

---

---

# Anhang

---

**Einleitung**

Der Anhang enthält die elektrischen und mechanischen Daten sowie die Betriebsdaten.

---

**Inhalt**

<b>Thema</b>	<b>Seite</b>
Schnittstellen .....	176
Technische Daten .....	179
Index .....	185

## A: Schnittstellen

---

### Einleitung

Das Bediengerät JVM-104 hat folgende Schnittstelle:

- M12-Stecker
- 

### M12-Stecker

Der M12-Stecker hat folgende Funktion:

- Spannungsversorgung des JVM-104
  - CANopen®-Busschnittstelle: CAN 1
  - Erkennung der Zündung
- 

### Inhalt

Thema	Seite
Gesamtübersicht der Anschlussbelegung .....	177

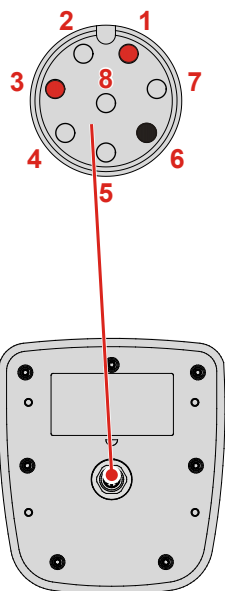


## Gesamtübersicht der Anschlussbelegung

### 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):



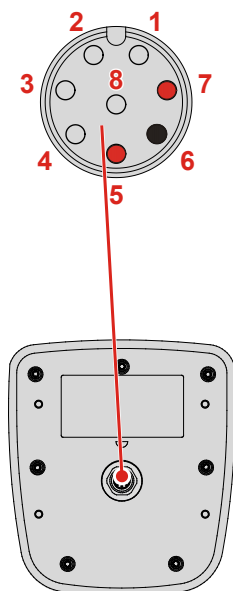
Die Pinbelegung ist wie folgt:

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

## CAN-Schnittstelle

Folgend ist die Pinbelegung des Anschlusssteckers für die CAN-Schnittstelle beschrieben.

Die Abbildung zeigt die Pinbelegung des Anschlusssteckers für den CANopen®-Bus (Blick auf die Kabelseite). Der Pin 6 des Bezugspotenzials ist auch farblich gekennzeichnet.



Die Pinbelegung ist wie folgt:

Pin	Beschreibung
5	CAN_L
6	Bezugspotenzial GND
7	CAN_H

---

---

## B: Technische Daten

---

### Einleitung

Dieses Kapitel im Anhang enthält die elektrischen und mechanischen Daten sowie die Betriebsdaten des JVM-104.

---

### Inhalt

<b>Thema</b>	<b>Seite</b>
Technische Daten .....	180
Mechanische Abmessungen.....	182
Betriebsparameter Umwelt und Mechanik .....	183
Betriebsparameter EMV .....	184

## 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, Kurzeitige Spannungspulse

### Technische Daten - Display

Parameter	Beschreibung
Display	3,5"-TFT-LCD-Flachbildschirm
Lichtstärke	LED-Backlight (weiß) , typ. 350 cd/m <sup>2</sup>
Auflösung Display	320 x 240 Pixel

### Technische Daten - CAN-Schnittstelle

Parameter	Beschreibung
Baudrate	250 kBaud ... 1 MBaud
Busabschlusswiderstand	Keiner im Gerät verbaut
Externer Busabschluss	120 Ω
Anschlusspezifikation	Verdrillte Adern, ungeschirmt

### Anzahl möglicher CANopen®- Schnittstellen

Parameter	Beschreibung
Anzahl möglicher CAN-Schnittstellen	1
CANMAX	0

### Anzahl möglicher SAEJ1939-Schnittstellen

Parameter	Beschreibung
Anzahl möglicher CAN-Schnittstellen	0
CANMAX	0

---

**Speicherausbau**

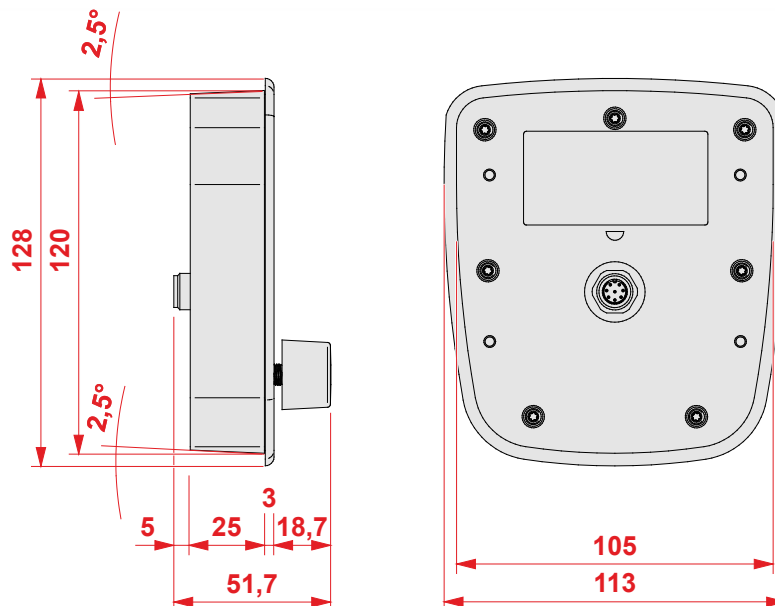
Parameter	Beschreibung
Anzahl remanente Register	30.000
Remanenter Variablenspeicher	120.000 Byte
Flash-Disk:	
Gesamter Speicher	512 MByte
Ordner <b>App</b>	64 MByte
Ordner <b>Data</b>	368 MByte

---

## Mechanische Abmessungen

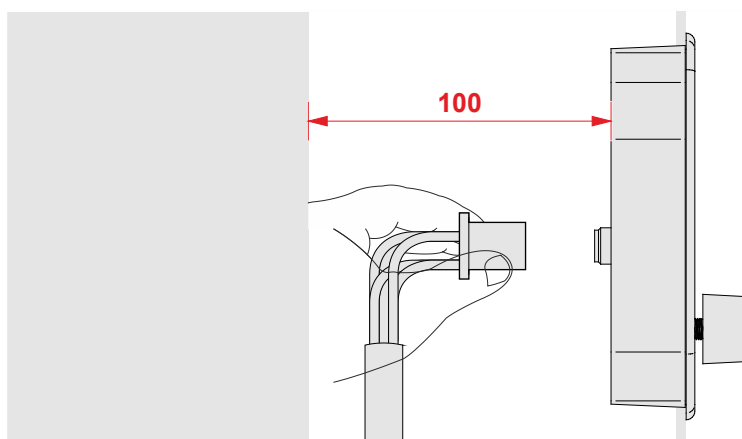
### Mechanische Abmessungen

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.

- Der Stecker muss sich jederzeit abziehen lassen.

## Betriebsparameter Umwelt und Mechanik

### Umwelt

Parameter	Wert	Norm
Betriebstemperaturbereich	-20 ... +60 °C	
Lagertemperaturbereich	-30 ... +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 <sup>2</sup> ) <sup>2</sup> /Hz 200 Hz / 0,02 (m/s <sup>2</sup> ) <sup>2</sup> /Hz 300 Hz / 0,01 (m/s <sup>2</sup> ) <sup>2</sup> /Hz 350 Hz / 0,002 (m/s <sup>2</sup> ) <sup>2</sup> /Hz Dauer: 3x 24 h	Nach DIN EN 60068-2-64
Schockfestigkeit	Halbsinus, 30 g (300 m/s <sup>2</sup> ), 18 ms, 3 Schocks in alle 6 Richtungen	Nach DIN EN 60068-2-27
Schutzart	Frontseitig: IP65 Rückseitig: IP65	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 ( $\mu\text{V}/\text{m}$ ) konstant	DIN EN 55025
Störaussendung nach CE	0,15 ... 0,5 MHz, 66 ... 56 dB ( $\mu\text{V}$ ) QP DC-Versorgung	DIN EN 55011-DC
	0,5 ... 5 MHz, 56 dB ( $\mu\text{V}$ ) QP 5 ... 30 MHz, 60 dB ( $\mu\text{V}$ ) QP 30 ... 230 MHz, 30/40 dB ( $\mu\text{V}/\text{m}$ ) Gehäuse 230 ... 1.000 MHz, 37/47 dB ( $\mu\text{V}/\text{m}$ )	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	DIN EN 61000-4-4
	4/2 kV Power	
	$\pm 1$ kV Line/Ground	DIN EN 61000-4-5
	$\pm 0,5$ kV Line/Line	
	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 • 167  
Standardablage • 166

### B

---

Bestandteile • 15  
Bestellbezeichnung • 17  
Bestimmungsgemäße Verwendung • 10  
Betriebsparameter  
EMV • 184  
Umwelt und Mechanik • 183  
Betriebssystemupdate • 161

### C

---

CANopen® • 65

### D

---

Dateisystem  
Eigenschaften • 132  
Verzeichnisse • 128

### E

---

Eingabe über das Digipot • 59  
Entsorgung • 10  
Erstinbetriebnahme • 39

### K

---

Kurzreferenz • 169

### L

---

Laufzeitregister • 155

### M

---

Mechanische Abmessungen • 18  
Montage • 33

### N

---

Nicht bestimmungsgemäße Verwendung • 10

### P

---

Personalqualifikation • 10  
Produktbeschreibung • 14  
Programmierung

Digipot • 151  
Eingabetasten • 149  
Zündung und Ausschaltverzögerung • 153

### R

---

Reparatur • 10

### S

---

SAE J1939 • 103  
Speicherarten • 137  
Speicherübersicht • 137  
Stecker • 26  
Beispiel einer Verdrahtung • 27  
CAN • 30  
Spannungsversorgung • 28

### T

---

Technische Daten • 180  
Transport • 10  
Typenschild • 21

### U

---

Umbauten • 10

### V

---

Versionsregister • 22  
Visualisierung ändern (Visualisierungskommando) • 63  
Visualisierung anlegen  
in JetSym • 50  
in JetViewSoft • 45

Jetter AG  
Gräterstraße 2  
71642 Ludwigsburg | Germany

Tel +49 7141 2550-0  
Fax +49 7141 2550-425  
[info@jetter.de](mailto:info@jetter.de)  
[www.jetter.de](http://www.jetter.de)

We automate your success.