BUCHER automation

User Manual



60884486_02

JCM-501-E01

Mobile controller

This document has been compiled by Jetter AG with due diligence based on the state of the art as known to them. Any revisions and technical advancements of our products are not automatically made available in a revised document.
The Jetter AG shall not be liable for any errors either in form or content, or for any missing updates, as well as for any damage or detriment resulting from such failure.

Bucher Automation AG
Thomas-Alva-Edison-Ring 10
71672 Marbach/Neckar, Germany
T +49 7141 2550-0
info@bucherautomation.com

Technical support
T +49 7141 2550-444
support@bucherautomation.com

Sales
T +49 7141 2550-663
sales@bucherautomation.com

www.bucherautomation.com

Translation of the german original User Manual

Revision	4.12.2
Date of issue	2/25/2026

Table of contents

1 Introduction ... 6
1.1 Information on this document ... 6
1.2 Typographical conventions ... 6
2 Safety ... 7
2.1 General ... 7
2.2 Purpose ... 7
2.2.1 Intended use ... 7
2.2.2 Non-intended use ... 7
2.3 Warnings used in this document ... 7
3 Product description ... 9
3.1 Design ... 9
3.2 Product features ... 9
3.3 LED indicators ... 9
3.3.1 Diagnostic capability via LEDs ... 9
3.4 Nameplate ... 10
3.5 Scope of delivery ... 10
4 Technical specifications ... 11
4.1 Dimensions ... 11
4.2 Mechanical specifications ... 12
4.3 Electrical properties ... 12
4.3.1 Ports and interfaces ... 12
4.4 Environmental conditions ... 14
4.5 Acoustic signal generator ... 14
4.6 EMI values ... 14
5 Mechanical installation ... 16
5.1 Requirements for the installation location ... 16
5.2 Installing the control ... 17
6 Electrical connection ... 19
6.1 Pin assignment ... 19
6.1.1 Deutsch connector – voltage supply, CAN, USB, RS-232 ... 19
7 Identification and Configuration ... 21
7.1 Identification ... 21
7.1.1 EDS registers ... 21
7.1.2 Version registers ... 21

7.2 Operating system ... 22
7.2.1 Operating system update of the controller ... 22
7.3 File system ... 25
7.3.1 Features ... 25
8 Programming ... 27
8.1 Abbreviations, module register properties and formats ... 27
8.2 CANopen STX API ... 27
8.2.1 STX Functions ... 28
8.2.2 Heartbeat monitoring ... 29
8.2.3 CANopen object directory ... 29
8.3 SAE J1939 STX API ... 30
8.3.1 STX Functions ... 30
8.4 Serial interface RS-232 ... 31
8.4.1 Operating principle ... 31
8.4.2 Registers - Description ... 34
8.4.3 Programming ... 39
8.5 Real-time clock ... 41
8.6 Storage options - Overview ... 42
8.6.1 File system memory ... 42
8.6.2 Application program memory ... 42
8.6.3 Flag ... 42
8.6.4 Storing registers and variables ... 43
8.7 Automatic copying of controller data ... 43
8.7.1 Loading the Autostart function ... 44
8.7.2 Executing the AutoStart function ... 44
8.7.3 Terminating the AutoStart mode ... 45
8.7.4 autostart.bat - Structure ... 45
8.7.5 Example of a command file ... 45
8.7.6 Data files ... 46
8.8 Saving and loading an application program ... 47
9 Registers - Overview ... 49
10 Maintenance ... 56
10.1 Repairs ... 56
10.2 Storage and Shipment ... 56
10.3 Return and Disposal ... 56

11 Service 58
11.1 Technical support 58
12 Spare parts and accessories 59
12.1 Accessories 59

1 Introduction

1.1 Information on this document

This document forms an integral part of the product and must be read and understood prior to using the device. It contains important and safety-related information for the proper use of the product as intended.

Target Groups

This document is intended for specialists with appropriate qualifications.
Only competent and trained personnel are allowed to put this device into operation.
During the whole product life cycle, safe handling and operation of the device must be ensured. In the case of missing or inadequate technical knowledge or knowledge of this document any liability is excluded.

Availability of Information

Make sure this document is kept at the ready in the vicinity of the product throughout its service life.
For information on new revisions of this document, visit the download area on our website. This document is not subject to any updating service.

Start | www.bucherautomation.com

For further information refer to the following information products:

– Online help for the JetSym software
  Functions of software products with application examples
– Application-oriented manuals
  Product-independent documentation
– Version updates
  Information about changes to the software products and operating system of your device

1.2 Typographical conventions

This manual uses different typographical effects to support you in finding and classifying information. Below, there is an example of a step-by-step instruction:

✓ This symbol indicates requirements which have to be met before executing the following action.

► This sign or a numbering at the beginning of a paragraph marks an action instruction that must be executed by the user. Execute the instructions one after the other.

⇨ The target after a list of instructions indicates reactions to, or results of these actions.

ⓘ Info	**Further information and practical tips** In the info box you will find helpful information and practical tips about your product.

2 Safety

2.1 General

The product corresponds to the current state of science and technology when placed on the market.
In addition to this user manual, the laws, regulations and directives of the country of operation or the EU apply to the operation of the product. The operator is responsible for compliance with the relevant accident prevention regulations and generally recognized safety rules.

2.2 Purpose

2.2.1 Intended use

The device is intended for controlling applications in the field of commercial vehicles and mobile machines.

Operate the device only in accordance with the intended conditions of use and within the limits set forth in the technical specifications.
Intended use of the product includes its operation in accordance with this manual.

SELV/PELV

If this device is not used in a vehicle or mobile machine, a SELV or PELV power supply unit must be used to supply the device.

2.2.2 Non-intended use

Do not use the device in technical systems for which a high level of failure safety is specified.

Machinery Directive

This device is not a safety-related part as per Machinery Directive 2006/42/EC and must, therefore, not be used for safety-relevant applications.

2.3 Warnings used in this document

⚠ DANGER



High risk

Indicates an imminently hazardous situation which, if not avoided, will result in death or serious injury.

⚠ WARNING



Medium risk

Indicates a potentially hazardous situation which, if not avoided, could result in death or serious injury.

⚠ CAUTION



Low risk

Indicates a hazardous situation which, if not avoided, could result in minor or moderate injury.

NOTICE



Material damage

Indicates a situation which, if not avoided, could result in malfunctions or material damage.

3 Product description

The JCM-501-E01 controller forms the heart of compact, distributed architectures. With its numerous interfaces, as well as the proven, powerful 32-bit CPU, it can be used in system solutions of simple to medium complexity.

3.1 Design

3.2 Product features

- Deutsch connector DT06-12S
- Ports and interfaces: 2x CAN, 1x USB, 1x RS-232 (all in Deutsch connector)
- CPU: ARM11, 500 MHz, 128 MB RAM, 512 MB Flash
- Front/rear degree of protection: IP65/IP65
- Integrated real-time clock

3.3 LED indicators

The JCM-501-E01 has 2 LEDs which can be configured by the user.

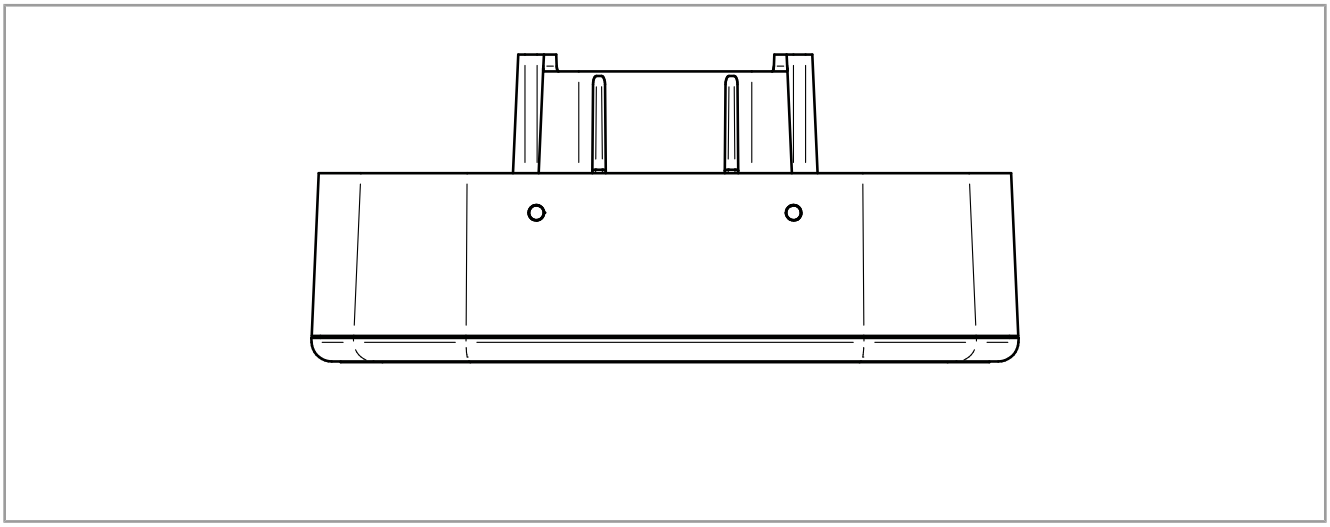


Fig. 1: LED indicators

1	LED D2
2	LED D1

3.3.1 Diagnostic capability via LEDs

Color and status of the LEDs are an excellent source of information for analyzing various states. In the JetSym programming tool, diagnostics can be performed in the Hardware Manager or via the Setup pane by entering the corresponding register number.

LED	Status	Color	Description
D2	OFF	---	This LED is controlled by register 108008. If bit 2 and bit 3 of register 108008 are 0 at the same time, the LED is off.
	ON	Red	Bit 3 is set in register 108008.
	ON	Green	Bit 2 is set in register 108008.
	ON	Orange	In register 108008, bits 2 and 3 are set simultaneously.
D1	OFF	---	This LED is controlled by register 108008. If bit 0 and bit 1 of register 108008 are 0 at the same time, the LED is off.
	ON	Red	Bit 1 is set in register 108008.
	ON	Green	Bit 0 is set in register 108008. The system is switched on (MR 108008=1).
	ON	Orange	In register 108008, bits 0 and 1 are set simultaneously.

3.4 Nameplate



Fig. 2: Nameplate

1	Logo
2	Serial number
3	Barcode
4	Article number and hardware revision
5	Model code number

3.5 Scope of delivery

Scope of delivery	Item number	Quantity
JCM-501-E01	10001818	1

4 Technical specifications

This chapter contains information on both electrical and mechanical data as well as on operating data of the JCM-501-E01 device.

4.1 Dimensions

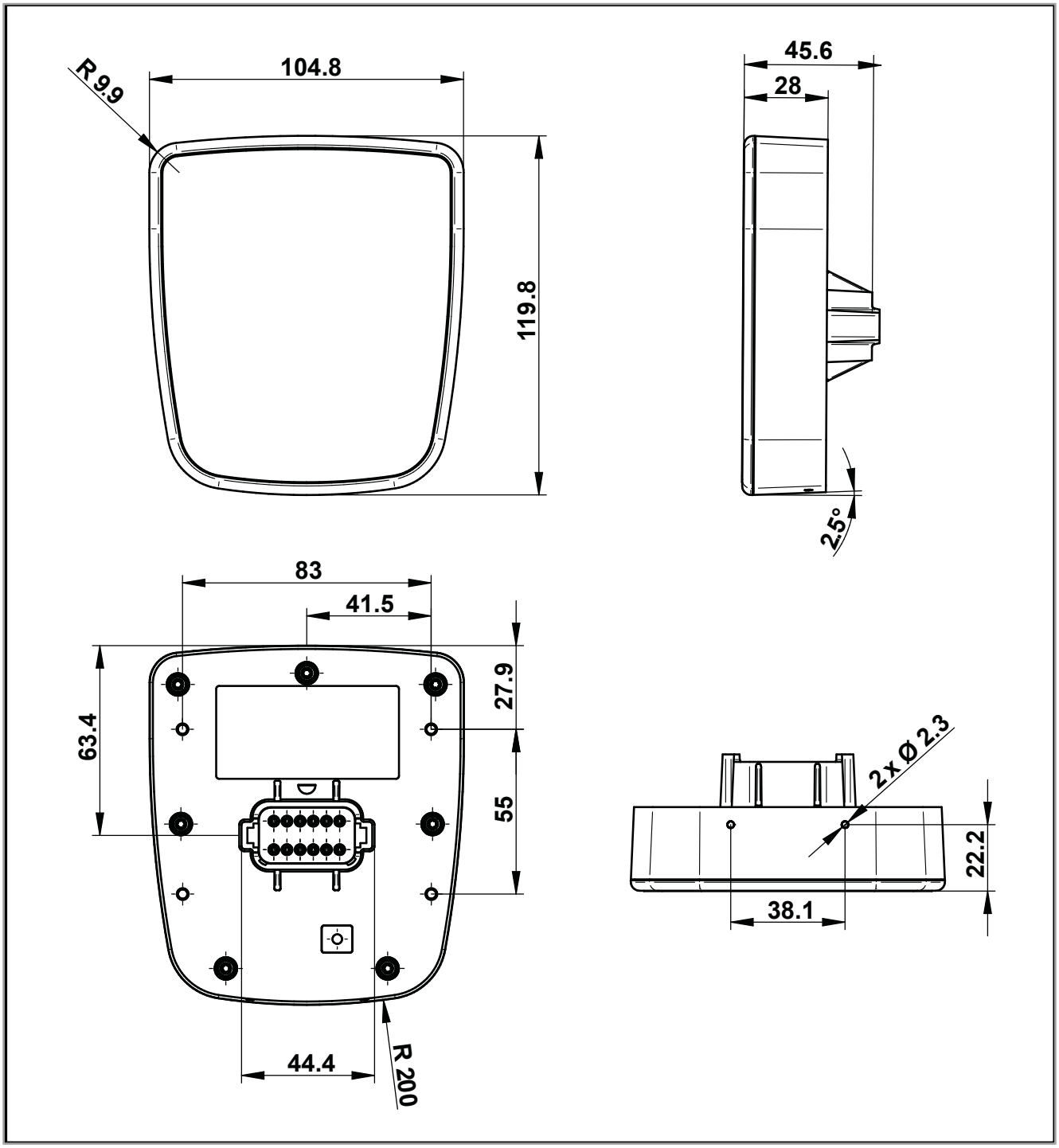


Fig. 3: Dimension in mm

4.2 Mechanical specifications

Parameter	Description	Standards
Weight	245 g	
Vibration		
Broadband noise	10 Hz / 0.005 $(m/s^2)^2/Hz$	DIN EN 60068-2-64
	200 Hz / 0.02 $(m/s^2)^2/Hz$	
	300 Hz / 0.01 $(m/s^2)^2/Hz$	
	350 Hz / 0.002 $(m/s^2)^2/Hz$	
Duration	3x 24 h	
Shock resistance		
Type of shock	Half-sine wave	DIN EN 60068-2-27
Intensity and duration	30 g (300 m/s^2) for 18 ms	
Number and direction	18 shocks in all 6 directions	
Degree of protection		
Front panel	IP65	DIN EN 60529
Rear panel	IP65	

Tab. 1: Mechanical specifications

4.3 Electrical properties

Power supply VBAT_ECU

Parameter	Description
Nominal voltage	DC 12 V or 24 V
Maximum current	2 A
Typical logic current consumption (VBAT_ECU)	170 mA at DC 12 V 90 mA at DC 24 V
Power consumption	Approx 2 W
Integrated protective functions	Reverse polarity protection, overvoltage, short voltage pulses
Permissible voltage range VBAT_ECU	DC 8 V ... 32 V

Tab. 2: Power supply VBAT_ECU

4.3.1 Ports and interfaces

CAN interfaces

Parameter	Description	
Quantity	2	CAN_1 and CAN_2
Baud rate	125 kBaud, 250 kBaud, 500 kBaud	

Parameter	Description	
Protocols	CANopen	
	SAE J1939	
Default node ID on the CANopen bus	127 (0x7F)	
Bus terminating resistor	CAN 1	Not built in
	CAN 2	Built in (120 Ω)
External bus termination	Required for CAN 1	120 Ω
Cable specification	Twisted pair conductors, unshielded	

Tab. 3: CAN interfaces

CAN bus cable

Parameter	Description
Wire cross-section	500 kBaud: 0.34 mm² … 0.50 mm² 250 kBaud: 0.34 mm² … 0.60 mm²
Cable capacitance	60 pF/m max.
Resistivity	500 kBaud: Max. 60 Ω/km 250 kBaud: Max. 60 Ω/km
Number of cores	2
Twisting	CAN_L and CAN_H cables are twisted pairwise

Tab. 4: Specification - CAN bus cable

The maximum permitted cable length depends on the baud rate being used and the number of CANopen devices being connected.

Baud rate	Max. cable length	Max. stub length	Total cable length
500 kBaud	100 m	5 m	30 m
250 kBaud	250 m	10 m	60 m

Tab. 5: Cable lengths

USB port

NOTICE



The USB port does not meet the immunity requirements relevant for KBA E1 approval.

- ► Use the USB port for service purposes only.
- ► Do not use the USB port while the unit is in operation.

Parameter	Description
Supply unit	5 V, max. 500 mA

Parameter	Description
Data rate	USB 2.0
High-speed type	480 MBaud max.
Protocol	USB host interface
Line length limitation	30 cm

Tab. 6: USB port

RS-232 interface

Parameter	Description	
Baud rates	Up to 115,200 Baud	8N1
Connection topology	1x Deutsch connector socket	

Tab. 7: RS-232 interface specification

4.4 Environmental conditions

Parameter	Description	Standards
Operating temperature	-20 °C ... +65 °C	ISO 16750-4
Climatic conditions	Humid heat	
Storage temperature	-20 °C ... +70 °C	ISO 16750-4 DIN EN 60068-2-1 DIN EN 60068-2-2
Relative humidity	10 % … 95 %	DIN EN 61131-2
Pollution degree	2	DIN EN 61131-2

Tab. 8: Environmental conditions

4.5 Acoustic signal generator

Parameter	Description	
Type	Loudspeaker	Adjustable frequency and volume.
Volume	83 dB	10 cm distance and resonance frequency 2,670 Hz

Tab. 9: Acoustic signal generator

4.6 EMI values

The JCM-501-E01 has E1 approval according to ECE R10 Rev. 5 and CE conformity according to ISO 14982.

NOTICE



The USB port does not meet the immunity requirements relevant for KBA E1 approval.

- ► Use the USB port for service purposes only.
- ► Do not use the USB port while the unit is in operation.

Pulses ISO 7637-2

Test pulse	Values	Function class
1	-450 V	C
2 a	+37 V	A
2b	+20 V	C
3 a	-150 V	A
3b	+150 V	A
4	Ua1: -12 V / 50 ms Ua2: -5 V / 500 ms	A (24 V systems)
	Ua1: -6 V / 15 ms Ua2: -2.5 V / 1,000 ms	C (12 V systems)

Tab. 10: Pulses ISO 7637-2

Irradiation ISO 11452

Parameter	Values	Function class
Irradiation	20 MHz ... 2 GHz 30 V/m	A

Tab. 11: Irradiation ISO 11452

ESD EN 61000-4-2

Parameter	Values	Function class
Contact discharge	±4 kV (to conductive surfaces)	A
Discharge through air	±8 kV (to insulating surfaces)	A

Tab. 12: ESD EN 61000-4-2

5 Mechanical installation

NOTICE



Damages to material or functional impairment due to welding

Welding on the chassis may damage the device material, or impair device functions.

- ► Before you start welding, disconnect all connections between the device and the electric system of the vehicle.
- ► Protect the device from flying sparks and welding beads (splatter).
- ► Do not touch the device with the welding electrode or earth clamp.

NOTICE



Dirt and moisture can affect the electrical connections

- ► Protect unused pins using blanking plugs.
- ► Protect all electrical connections with appropriate single wire seals.
- ► Clean the area around a connector prior to removing the mating connector.

5.1 Requirements for the installation location

The installation location must meet the following requirements:

– The installation location must allow air to circulate.
– The installation location must be of sufficient size.
– The device must be easily accessible to allow for service work.

Space required for installation and service

It should be possible to disconnect the connectors at any time.

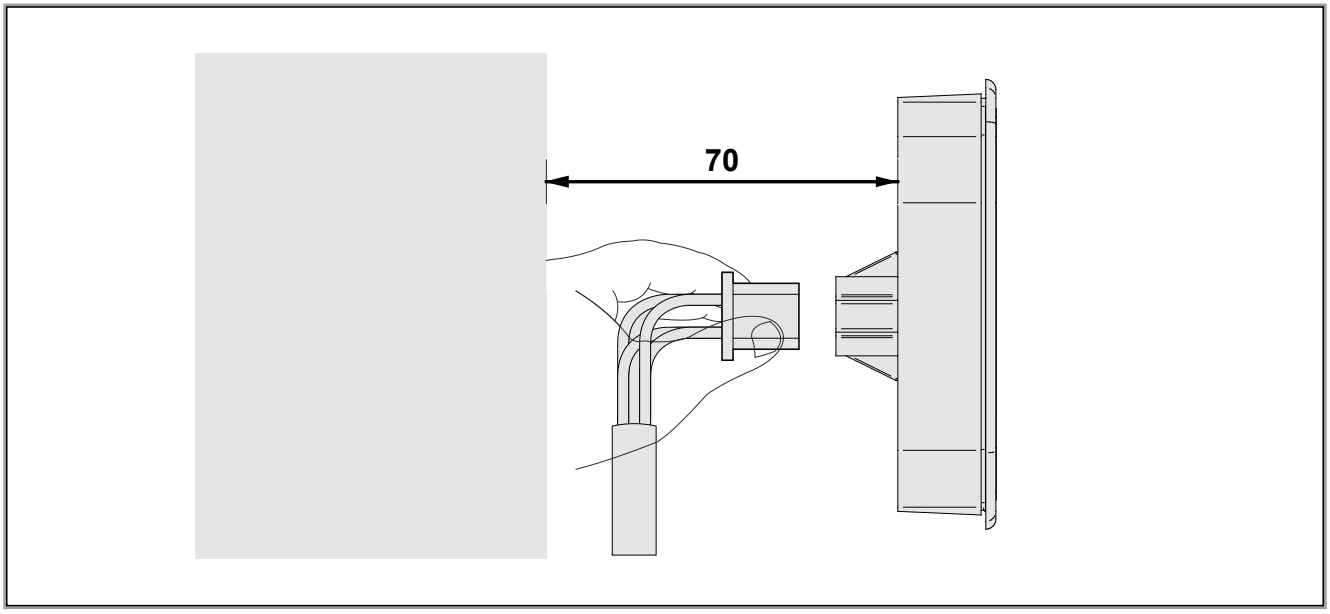


Fig. 4: Space requirements for installation work (in mm)

Avoiding unsuitable installation locations

The following installation locations are unsuitable for mounting the device:

Unsuitable installation location	Reason
Outdoor installation	The device must not be exposed to rain or a jet of water. Do not use a steam jet or other such devices to clean the device.
Installation location close to heat-sensitive materials	The materials could become warped or misshapen as a result of heat produced by the device.

Tab. 13: Unsuitable installation locations

5.2 Installing the control

Screw holes

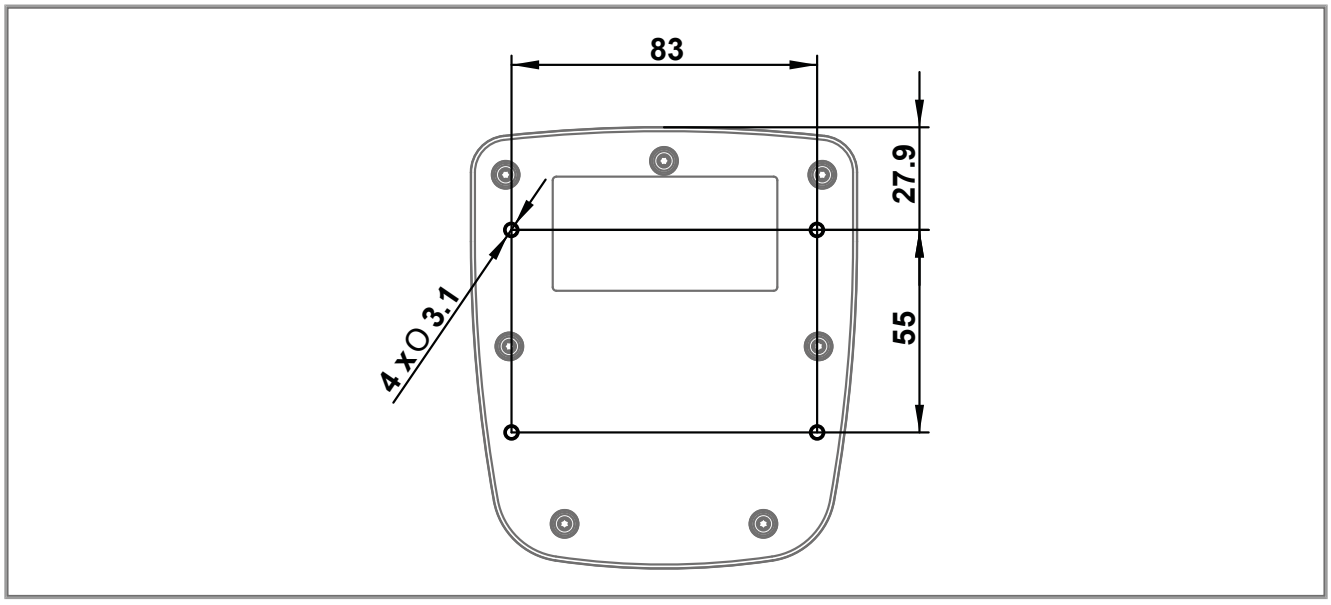


Fig. 5: Screw holes, dimensions in mm

Installing the strain relief

Install strain reliefs for the connecting cables.

– Ensure that there is sufficient clearance between the strain reliefs and the connectors.
– Connectors must not be obstructed, so that they can be removed in the event of service.

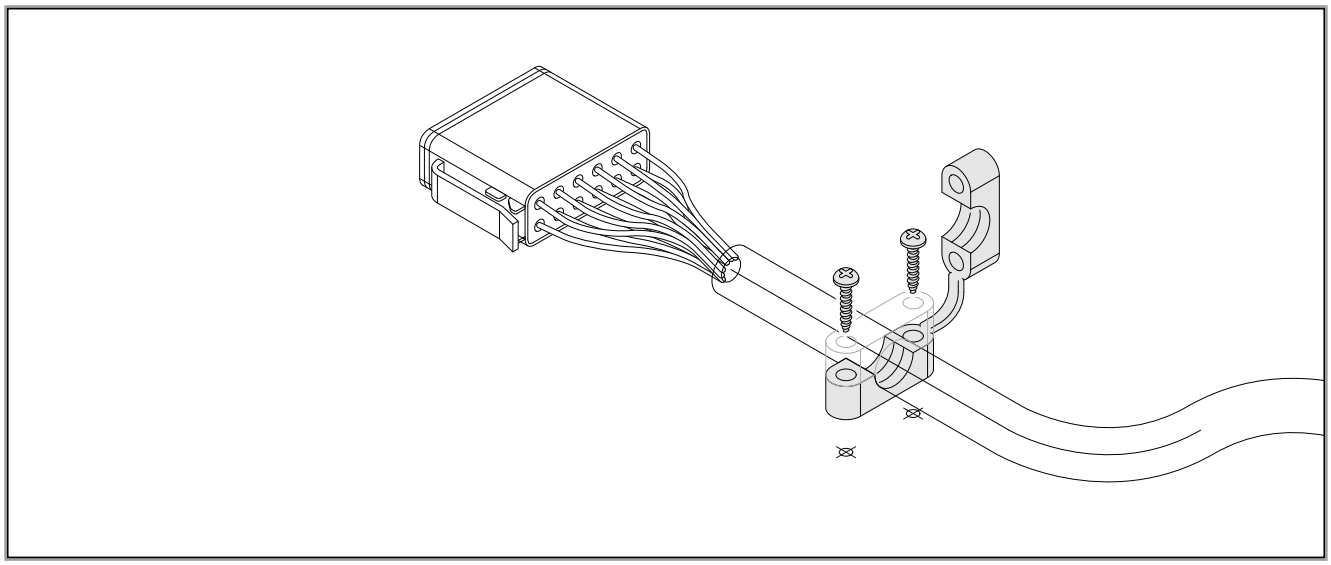


Fig. 6: Installing the strain relief

6 Electrical connection

NOTICE

Damages to material or functional impairment

Improper implementation of the wiring harness may cause mechanical stress.

- ► Protect the cables from bending, twisting or chafing.
- ► Install strain reliefs for the connecting cables.

NOTICE

Surges resulting from missing protection or fusing

Surges may cause malfunctions or damage to the product.

- ► Protect the voltage inputs from surges according to the requirements.
- ► Ensure that the device is handled in accordance with ESD regulations.

6.1 Pin assignment

6.1.1 Deutsch connector – voltage supply, CAN, USB, RS-232

Function

The Deutsch connector has the following functions:

– Power supply to the JCM-501-E01
– CAN communication
– USB
– RS-232

ⓘ Info

Current consumption

When the device is energized, the current consumption is temporarily higher. To ensure a reliable start-up of the device, provide at least 3 times the typical current required.

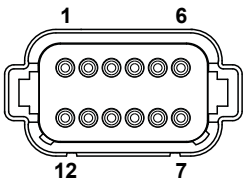


Fig. 7: Deutsch connector, 12 pins

12	CAN_2_H	1	GND
11	CAN_2_L	2	USB_+5V
10	CAN_1_H	3	USB_D+
9	CAN_1_L	4	USB_D-
8	RXD_RS232	5	GND
7	TXD_RS232	6	VBAT_ECU

Mating parts

Compatible mating parts for the 12-pin DEUTSCH connector are as follows:

Parameter	Description
Manufacturer	German
Manufacturer item number – housing	DT06-12S
Manufacturer item number – wedge lock	W12S
Manufacturer item number – Crimp contact (receptacle)	0-462-201-16141
Wire size range	1.0 mm² ... 1.5 mm² (AWG 18 ... 16)

7 Identification and Configuration

7.1 Identification

This chapter describes how to identify the JCM-501-E01 device:

– Determining the hardware revision
– Retrieving Electronic Data Sheet (EDS) information. The EDS holds numerous non-volatile production-relevant data.
– Determining the OS version of the device and its software components

7.1.1 EDS registers

Use the EDS registers to retrieve Electronic Data Sheet (EDS) information. These registers contain the exact same information as the EDS file. They are read-only (ro).

Register overview

You can find an overview of the EDS registers in the chapter Register overview.

7.1.2 Version registers

The operating system provides several registers which can be used to read out the hardware revision or OS version of the device and its components.
You will need this information when contacting the Jetter AG support hotline in case of a problem.

Operating system version

The device has special registers, the content of which are unique OS version numbers.

Format of software version numbers

The software version numbers of the JCM-501-E01 are represented by 4 numbers.

1 . 2 . 3 . 4

Fig. 8: Software version numbers

Digits	Description
1	Major or main version number
2	Minor or secondary version number
3	Branch or intermediate version number
4	Build version number

Tab. 14: Format of software version numbers

Released version

A released version can be recognized by both Branch and Build having got value 0.

Registers - Overview

The registers listed below let you retrieve the operating system versions:

Register	Description
200169	Operating system version
210001	JetVM version

Tab. 15: Registers - Overview

7.2 Operating system

We are continuously striving to enhance the operating systems of our products. Enhancing means adding new features, and upgrading existing functions. Current OS files are available for download from our website on the pages of the respective product.

ⓘ Info

Further information

More information on this subject is available on our website.

Start | www.bucherautomation.com

7.2.1 Operating system update of the controller

This chapter describes how to carry out an OS update of the JCM-501-E01 controller. There are several ways of transferring the OS file to the controller:

– From a USB flash drive
– Via JetEasyDownload

OS update from a USB flash drive

A USB flash drive lets you transfer an OS file to the JCM-501-E01.

Contents of the batch file

The batch file **autostart.bat** must have the following contents:

```
@echo off
if not exist \App\autostart.exe goto end
if exist \App\update.ini goto end
if exist \App\usb_update_process_marker goto dm
echo \Data\update\jcm_ce0.os >\App\update.ini
if not exist \Data\update md \Data\update
copy \%1\jcm_ce0.os \Data\update\jcm_ce0.os
echo.>\App\usb_update_process_marker
del \System\reset.exe
goto end
:dm
del \App\usb_update_process_marker
:end
```

Updating the OS

✓ An OS file for the JCM-501-E01 is available.

✓ The operating system of the JCM-501-E01 as of version 4.07.0.12 is running.

✓ The device remains powered on during the operating system update.

1. Copy the OS file to your USB flash drive. This drive must be FAT formatted.
2. Create a batch file on the USB flash drive and name it **autostart.bat**. Contents of the batch file: See above.
3. Replace all entries **jcm_ce0** with the actual name of the OS file.
4. Switch off the JCM-501-E01.
5. Insert the USB flash drive into the JCM-501-E01.
6. Switch on the JCM-501-E01.

   ⇨ About 10 seconds later the device automatically launches the update process.

⇨ The device reboots automatically and starts the application.

Update sequence

The operating system update runs in the following chronological order:

Step	Description
1	The files are transferred.
2	The device is reset.
3	The files are installed.
4	The device is reset.
5	The application appears.

Log file for documentation purposes

The update process is documented in the **\App\update.log** file.

OS update via JetEasyDownload

To update the operating system of a device, use a PEAK CAN dongle and the Jetter command line tool JetEasyDownload (version 1.00.0.15 or higher).

JetEasyDownload parameters

Calling JetEasyDownload requires the applicable specifications parameters.

Parameter	Description	Values	
-H<Num>	Hardware	0 =	PCAN_ISA1CH
		1 =	PCAN_ISA2CH
		2 =	PCAN_PCI_1CH
		3 =	PCAN_PCI_2CH
		4 =	PCAN_PCC_1CH
		5 =	PCAN_PCC_2CH
		6 =	PCAN_USB_1CH
		7 =	PCAN_USB_2CH
		8 =	PCAN_Dongle Pro
		9 =	PCAN_Dongle
		10 =	PCAN_NET Jetter
		11 =	PCAN_DEV default device
		20 =	IXXAT V2.18
		22 =	IXXAT V3
		100 =	CAN hardware detected first
-T<nodeID>	Target node ID	The Node ID is given as a decimal.	
-B<Num>	Baud rate **Mind the permissible baud rates of your device!**	0 =	10 kBd
		1 =	20 kBd
		2 =	50 kBd
		3 =	100 kBd
		4 =	125 kBd
		5 =	250 kBd
		6 =	500 kBd
		7 =	1 MBd
-S<Num>	SDO timeout	Default	300 ms
-L<name>	OS file name	e.g. JCM-501-E01_Vx.xx.x.xx.os	

Tab. 16: JetEasyDownload parameters

Performing the update

JetEasyDownload –H100 –T127 –B5 –S8000 –Ljcm_ce0_X.XX.X.XX.os

- ✓ JetEasyDownload and the PEAK CAN dongle are ready for use.
- ✓ A CAN connection is established between the PEAK CAN dongle and the JCM-501-E01.

1. Call up JetEasyDownload with the above parameters and a valid OS file.
   - ⇨ The device carries out a reset.
   - ⇨ The device starts in boot loader mode with a single heartbeat in init state (data = 0x00).
2. Wait for approx. 7 seconds while the device formats the flash memory.

⇨ The device initiates the download process.

⇨ The device starts automatically with the new firmware.

7.3 File system

The file system lets you access files located on the internal flash disk or an USB flash drive. Some files may be protected against read/write access or deletion. This is normal behavior. Some of these files are virtual files, such as firmware images, or protected files, such as EDS files.

NOTICE



Malfunctions caused by missing or damaged system files

Careless working with system files can result in malfunctions of the device.

► Do not delete or move any system files.

Division

The file system distinguishes between the following file types:

– System directories/files used by the operating system

– Files used by the user

System directories

System directories cannot be deleted. They are still available after formatting.

Directory	Description
\System	– System configuration – System information – Boot screen (**bootupscreen.bmp**) – Screenshot
\App	Directory for applications
\Data	Folder for storing data
\Windows	Windows CE system directory
\	RAM disk drive
\USB	Root directory of the USB flash drive

ⓘ Info

Further information

For in-depth information, see the *File System* application-oriented manual.

For application-oriented manuals open www.bucherautomation.com and go to *product page JCM-501-E01 > Product-independent documentation*.

7.3.1 Features

The following properties apply to the internal flash disk and USB flash drive:

– Open a maximum of 8 files at the same time.

– Separate directory names with a slash "/" rather than a backslash "\".

– Any file created by the controller contains the date and time of the device.
– Date, time and file size are not available for all system files.

Flash disk - Properties

Size

The following disk space is available to the user:

Parameter	Value
Flash disk capacity	512 MB
Of which folder **App**	64 MB
Of which folder **Data**	368 MB

Features

The internal flash disk drive has got the following properties:

– Up to 7 directory levels and 1 file level are allowed.
– No case sensitivity.
– Directory and file names with a length of up to 63 characters are possible.
– All characters except "/" and ".." are permitted for directory and file names.
– The location of the directories "App" and "Data" is on the flash disk drive.

USB flash drive - Properties

Size

The available capacity depends on the USB flash drive used. Tested capacity:

– 256 MB ... 8 GB

Features

The USB flash drive has got the following properties:

– The USB flash drive must be compatible with FAT 16, or FAT 32.
– No case sensitivity.
– Directory and file names of 260 characters' length max. can be used.
– The following characters are not permitted in directory and file names: "/", "\", ":", "*", "?", "<", ">" and "|"
– There is no user/access administration.

Guarantee

Jetter AG only guarantees for the proper functioning of USB flash drives optionally available from Jetter AG.

8 Programming

8.1 Abbreviations, module register properties and formats

Abbreviations

The abbreviations used in this document are listed in the table below:

Abbreviation	Description
R 100	Register 100
MR 150	Module register 150

Tab. 17: Abbreviations

Module register properties

Each module register is characterized by certain properties. Most properties are identical for many module registers. In the following description, module register properties are mentioned only if a property deviates from the default properties listed below.

Property	Default
Access	Read/write
Value after reset	0 or undefined (e.g. revision/version number)
Takes effect	Immediately
Write access	Always
Data type	Integer

Tab. 18: Module register properties

Numerical formats

The numerical formats used in this document are listed in the table below:

Notation	Format of numerical values
100	Decimal
0x100	Hexadecimal
0b100	Binary

Tab. 19: Numerical formats

8.2 CANopen STX API

The CANopen STX API is a platform allowing users to send and receive CANopen messages via STX functions.

The device uses the STX functions to communicate with other CANopen nodes.

The CANopen standard

CANopen is an open standard for networking and communication, used e.g. in automotive applications. The CANopen protocol is being further developed by the CiA e.V. (CAN in Automation). On the physical layer it uses CAN high-speed in accordance with ISO 11898.

Specifications

The CANopen specifications can be obtained from the **CiA e.V.** homepage http://www.can-cia.org.
The key specification documents are:

– CiA DS 301 – This document is also known as Communication Profile and describes the fundamental services and protocols used under CANopen.
– CiA DS 302 – Framework for programmable devices (CANopen Manager, SDO Manager)
– CiA DR 303 – Information on cables and connectors
– CiA DS 4xx - These documents describe the behavior of a number of device classes in, what are known as, device profiles.

ⓘ Info

Further information

For in-depth information, see the *CANopen STX API* application-oriented manual.

For application-oriented manuals open www.bucherautomation.com and go to *product page JCM-501-E01 > Product-independent documentation*.

8.2.1 STX Functions

Application

STX functions are used to enable communication between the JCM-501-E01 and other CANopen nodes.

The JCM-501-E01 supports the following STX functions:

Function	Description
`CanOpenInit()`	Calling up the `CanOpenInit()` function initializes one of the available CAN buses.
`CanOpenSetCommand()`	The function `CanOpenSetCommand()` lets you change the heartbeat status of the device itself and of all other devices (NMT slaves) on the CAN bus.
`CanOpenUploadSDO()`	The function `CanOpenUploadSDO()` lets you access a particular object in the Object Directory of the message recipient and read the value of the object. Data is exchanged in accordance with the SDO upload protocol.
`CanOpenDownloadSDO()`	The function `CanOpenDownloadSDO()` lets you access a particular object in the Object Directory of the message recipient and specify the value of the object. Data is exchanged in accordance with the SDO download protocol.
`CanOpenAddPDORx()`	Calling up the `CanOpenAddPDORx()` function allows for process data coming from other CANopen devices to be entered for reception.
`CanOpenAddPDOTx()`	By calling up the `CanOpenAddPDOTx()` function, process data can be placed on the bus.

Tab. 20: Supported STX functions

ⓘ Info

Further information

For in-depth information, see the *CANopen STX API* application-oriented manual.

For application-oriented manuals open www.bucherautomation.com and go to *product page JCM-501-E01 > Product-independent documentation*.

ⓘ Info

Further information

For more information on this subject, refer to the JetSym Online Help.

8.2.2 Heartbeat monitoring

The heartbeat protocol is for monitoring the activity of communication partners. If the inactivity exceeds the set interval (Heartbeat consumer time), the status is set to "**offline**".

The application program lets you define heartbeat functions, such as

– Displaying information to the user
– Rebooting the device
– Ignoring process data

ⓘ Info

Further information

For in-depth information, see the *CANopen STX API* application-oriented manual.

For application-oriented manuals open www.bucherautomation.com and go to *product page JCM-501-E01 > Product-independent documentation*.

8.2.3 CANopen object directory

The device's OS supports the following objects:

Index (hex)	Object (code)	Object name	Data type	Access
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	1008
1009	VAR	Manufacturer Hardware Version	String const	1009
100 A	VAR	Manufacturer Software Version	String const	100 A
100 B	VAR	Node ID	Unsigned32	RO
1017	VAR	Producer Heartbeat Time	Unsigned16	R/W (read & write)
1018	RECORD	Identity	Identity	RO
1200	RECORD	Server 1 - SDO parameter	SDO parameter	RO
1201	RECORD	Server 2 - SDO parameter	SDO parameter	R/W
1203	RECORD	Server 3 - SDO parameter	SDO parameter	R/W
1203	RECORD	Server 4 - SDO parameter	SDO parameter	R/W

ⓘ Info

Further information

For in-depth information, see the *CANopen STX API* application-oriented manual.

For application-oriented manuals open www.bucherautomation.com and go to *product page JCM-501-E01 > Product-independent documentation*.

8.3 SAE J1939 STX API

SAE J1939 is an open standard for networking and communication in the commercial vehicle sector. The focal point of the application is the networking of the power train and chassis. The J1939 protocol originates from the international **Society of Automotive Engineers (SAE)** and works on the physical layer with CAN high-speed according to ISO 11898.

Specifications

The SAE J1939 specifications can be obtained from the SAE homepage http://www.sae.org.

The key SAE J1939 specifications are:

– J1939-11 - Information on the physical layer
– J1939-21 - Information on the data link layer
– J1939-71 - Information on the application layer vehicles
– J1939-73 - Information on the application layer range analysis
– J1939-81 - Network management

ⓘ Info

Further information

For further information refer to the *SAE J1939 STX API* application-oriented manual in the download area of our homepage.

For application-oriented manuals open www.bucherautomation.com and go to *product page JCM-501-E01 > Product-independent documentation*.

8.3.1 STX Functions

Application

STX functions are used in the communication between the JCM-501-E01 and other devices in a vehicle.

The JCM-501-E01 supports the following STX functions:

Function	Description
`SAEJ1939Init()`	Calling up the funvtion `SAEJ1939Init()` initializes one of the existing CAN buses (not CAN 0, because it is reserved for CANopen) for the J1939 protocol.
`SAEJ1939SetSA()`	The function `SAEJ1939SetSA()` lets you change your own SA (Source Address) during runtime.
`SAEJ1939GetSA()`	The function `SAEJ1939GetSA()` lets you determine your own SA (Source Address).
`SAEJ1939AddRx()`	Calling up the function `SAEJ1939AddRx()` prompts the JCM-501-E01 to receive a specific message.

Function	Description
`SAEJ1939AddTx()`	Calling up the function `SAEJ1939AddTx()` prompts the JCM-501-E01 to cyclically send a specific message via the bus.
`SAEJ1939RequestPGN()`	Calling up the function `SAEJ1939RequestPGN()` sends a request to the DA (Destination Address) following a PGN.
`SAEJ1939GetDM1()`	Calling up the function `SAEJ1939GetDM1()` requests the current diagnostics error codes (also see SAE J1939-73 No. 5.7.1).
`SAEJ1939GetDM2()`	Calling up the function `SAEJ1939GetDM2()` requests the diagnostics error codes that preceded the current ones (also see SAEJ1939-73 No. 5.7.2).
`SAEJ1939 SetSPNConversion()`	Calling up the function `SAEJ1939SetSPNConversion()` defines the arrangement of bytes in the message which is requested by the `SAEJ1939GetDM1()` or `SAEJ1939GetDM2()` function.
`SAEJ1939 GetSPNConversion()`	Calling up the function `SAEJ1939GetSPNConversion()` determines the currently set conversion method.

ⓘ Info

Further information

For further information refer to the *SAE J1939 STX API* application-oriented manual in the download area of our homepage.

For application-oriented manuals open www.bucherautomation.com and go to *product page JCM-501-E01 > Product-independent documentation*.

8.4 Serial interface RS-232

The JCM-501-E01 has a user-programmable RS-232 interface.

8.4.1 Operating principle

The OS provides a receive buffer and a transmit buffer for the user-programmable serial interface. These buffers can be used to adjust the transfer rate between application program and serial interface.

Block diagram

The following illustration shows the block diagram of the user-programmable serial interface:

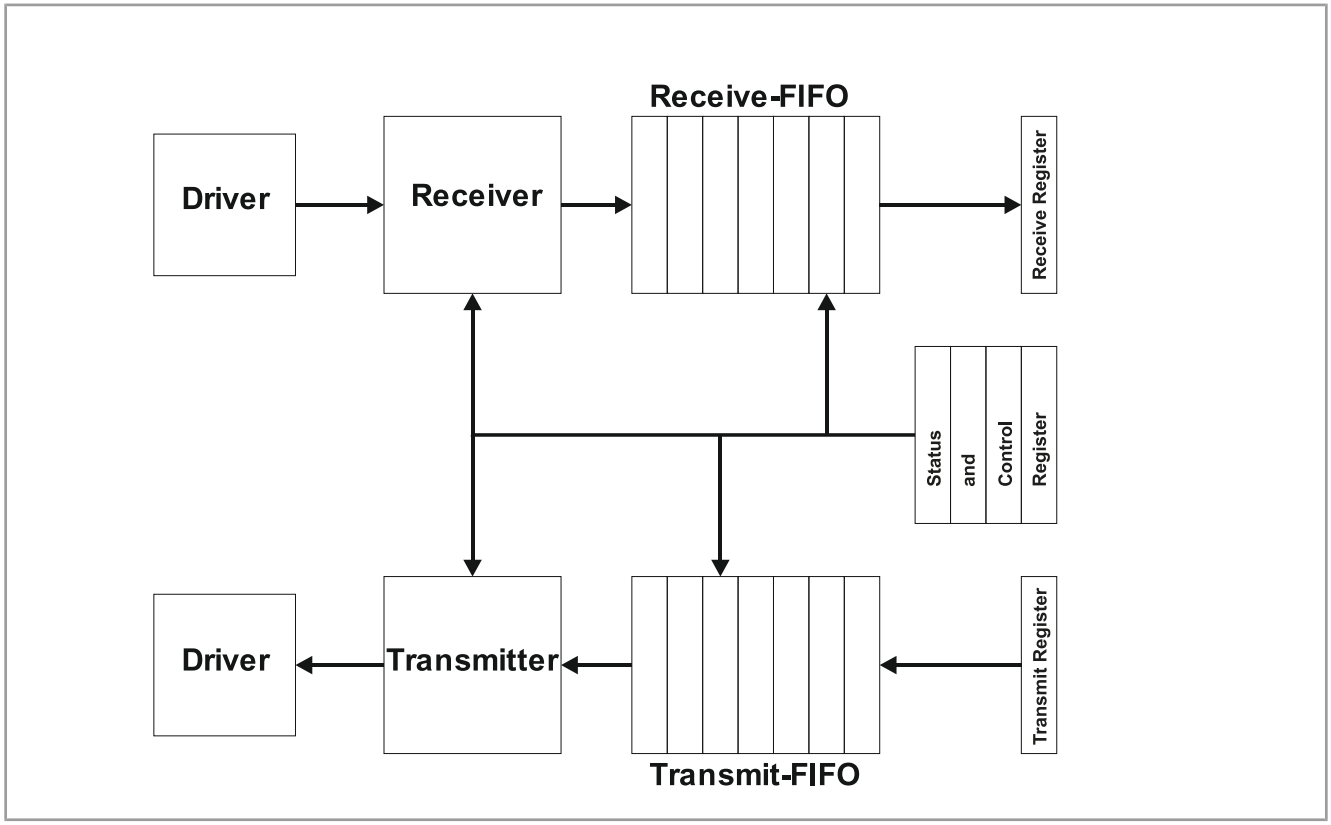


Fig. 9: Block Diagram - Serial Interface

Elements of the interface

The user-programmable serial interface consists of the following elements:

Number	Section	Function
1	Interface driver	Converts the signals of the interface into internal signal levels
2	Addressee	Performs serial/parallel conversion
3	Receive buffer	Buffer for received characters
4	Receive register	Read access to this register reads the received characters in the receive buffer (3).
5	Interface driver	Converts internal signal levels into interface signals.
6	Transmitter	Performs parallel/serial conversion
7	Sending buffer	Buffer for characters to be sent
8	Transmit register	Write access to this register causes the characters to be entered into the transmit buffer (7) and to be sent by the transmitter (6).
9	Status and control register	Query of filling levels and error states of buffers; setting of transmission parameters

Tab. 21: Elements of the serial interface

Receiving a character

Receiving a character is done in 4 steps:

Step	Description
1	The interface driver converts signals "on the line" into internal signal levels and forwards them to the receiver.
2	The receiver performs serial/parallel conversion of this character and checks the set communication parameters.
3	The receiver enters the character into the receive buffer if there is any place left. Otherwise, the character is discarded and buffer overflow is signaled.
4	Via receive register the character can be read out of the receive buffer.

Sending a character

A character is sent in 4 steps:

Step	Description
1	Via transmit register the character is entered into the transmit buffer if there is any place left. Otherwise the character is discarded.
2	Once the transmitter has sent a character, it reads the next character from the transmit buffer.
3	The transmitter performs parallel/serial conversion and sends this character to the interface driver using the set communication parameters.
4	The interface driver converts internal signal levels into the various interface standards

Error detection

When receiving characters, the following errors are detected by the controller and displayed in the register *Error state*:

Errors	Description	Effect
Framing error	The format of the received character does not match the set parameters.	The erroneous character(s) is (are) stored in the receive buffer and error bit *Framing error* is set. The error counter is incremented.
Parity error	The parity bit of the received character is not correct.	The erroneous character is stored in the receive buffer and error bit *Parity error* is set. The error counter is incremented.
Buffer overflow	A character is received, although the receive buffer is full.	The character is discarded and error bit *Overflow* is set. The error counter is incremented.

Troubleshooting

As error bits cannot be assigned to individual characters in the receive buffer, all characters should be removed from the receive buffer and discarded when an error bit is set.

Possible causes of error and troubleshooting:

Errors	Possible cause	Troubleshooting
Framing error	Jammed data transmission caused by EMC problems, defective cables or connectors	– Check the wiring and connectors. – Use shielded cables. – Do not lay cables near sources of interference.
	Incorrectly set communication parameters (baud rate, number of stop bits, etc.)	– Make sure the set communication parameters are consistent with the settings of the connected device.
Parity error	Jammed data transmission caused by EMC problems, defective cables or connectors	– Check the wiring and connectors. – Use shielded cables. – Do not lay cables near sources of interference.
	Incorrectly set parity	– Make sure the parity setting is consistent with the setting of the connected device.
Buffer overflow	The external device sends characters at too high a rate and the application program is not able to read them out of the receive buffer in due time.	– Program a software handshake. – Set a lower baud rate. – Make sure that characters are read out from the receive buffer faster. To achieve this the program code has to be optimized.

8.4.2 Registers - Description

This chapter describes the registers associated with the user-programmable serial interface. These registers are used for the following tasks:

– Parameterizing the interface
– Sending characters
– Receiving characters

Register numbers

The registers of the interface are combined into one register block. The basic register number of this block is dependent on the device.

Device	Basic register number	Register numbers
JCM-501-E01	103100	103100 … 103119

Tab. 22: Register numbers of the serial interface

Determining register numbers

In this chapter, only the last two figures of a register number are specified. e.g. MR 14. To calculate the complete register number, add the basic register number of the corresponding device, e.g. 103100, to the module register number.

ⓘ Info

Re-initializing the interface

When entering values into registers MR 1 through MR 6, the entire interface is re-initialized and the transmit and receive buffers are cleared.

MR 1

Protocol

This register lets you set the protocol which is supported by the OS of the controller. That is, this register is for defining how the interface is used.

Property	Description	
Values	1	System logger
	2	User-programmable interface
	3	PcomX
Value after reset	3	

Tab. 23: Protocol

MR 2

Baud rate

This register lets you set the baud rate.

Property	Description
Values	2,400 ... 115,200
Value after reset	9,600

Tab. 24: Baud rate

MR 3

Number of data bits per character

This register lets you set the number of data bits per character.

Property	Description
Values	5, 6, 7, 8
Value after reset	8

Tab. 25: Number of data bits per character

MR 4

Stop bits

This register lets you set the number of stop bits per character.

Property	Description	
Values	1	1 stop bit
	2	1.5 stop bits if MR 3 = 5 2 stop bits if MR 3 = 6, 7, 8
Value after reset	1	

Tab. 26: Stop bits

MR 5

Parity

This register lets you set the parity of a character.

Property	Description	
Values	0	None (no parity)
	1	Odd parity
	2	Even parity
	3	1 (mark)
	4	0 (space)
Value after reset	2	

Tab. 27: Parity

MR 7

Activating the serial interface

This register lets you activate the serial interface.

Property	Description	
Values	0	The serial interface is inactive, no data can be received/transmitted.
	1	The serial interface is active, data can be received/transmitted.
Value after reset	0	

Tab. 28: Activating the serial interface

MR 10

Sending buffer

The character that has to be sent must be entered into this register.

– If the sending buffer is able to accommodate the character, it is entered into this buffer. This character will be sent once all previously entered characters have been sent.
– Prior to sending characters from the application program, it must be checked whether the sending buffer is able to accommodate characters. This can be checked by reading out MR 11.
– The sending buffer functions according to the FIFO principle. The first character entered is sent first.

Property	Description	
Values	0 ... 31	5 bits per character
	0 ... 63	6 bits per character
	0 ... 127	7 bits per character
	0 ... 255	8 bits per character
Type of access	Read	Character written last
	Write access	Sending a character

Tab. 29: Sending buffer

MR 11

Sending buffer filling level

This register shows how many characters the sending buffer accommodates.

There is space for 32,768 characters max. within the buffer.

Property	Description
Values	0 ... 32,768

Tab. 30: Sending buffer filling level

MR 12

Receive buffer, 8 bits (without deleting the character on reading)

This register shows the "oldest" character stored in the receive buffer. On reading, this character will not be removed from the buffer.

Property	Description	
Values	0 ... 31	5 bits per character
	0 ... 63	6 bits per character
	0 ... 127	7 bits per character
	0 ... 255	8 bits per character
Type of access	Read	Oldest character in buffer
Takes effect	if MR 14 > 0	

Tab. 31: Receive buffer, 8 bits (without deleting the character on reading)

MR 13

Receive buffer, 8 bits (with deleting the character on reading)

This register shows the "oldest" character stored in the receive buffer. This character is removed from the buffer. Thus, the character received next can be read out during the next read access.

Property	Description	
Values	0 ... 31	5 bits per character
	0 ... 63	6 bits per character
	0 ... 127	7 bits per character
	0 ... 255	8 bits per character
Type of access	Read	Oldest character in buffer
Takes effect	if MR 14 > 0	

Tab. 32: Receive buffer, 8 bits (with deleting the character on reading)

MR 14

Receive buffer filling level

This register shows how many characters the receive buffer accommodates. Each read access to MR 13 decrements this register by 1.

Property	Description
Values	0 ... 32,768

Tab. 33: Receive buffer filling level

MR 15

Receive buffer, 16-bit, little endian

Read access to this register removes 2 characters from the receive buffer and returns them as 16-bit value.

Assignment:

Character	Bits in register
First	Bit 0 ... 7
Second	Bit 8 ... 15

Property	Description	
Values	0 ... 65,535	
Type of access	Read	Removes 2 characters from the buffer
Takes effect	if MR 14 > 1	

Tab. 34: Receive buffer, 16-bit, little endian

MR 16

Receive buffer; 16-bit; big endian

Read access to this register removes 2 characters from the receive buffer and returns them as 16-bit value.

Assignment:

Character	Bits in register
First	Bit 8 ... 15
Second	Bit 0 ... 7

Property	Description	
Values	0 ... 65,535	
Type of access	Read	Removes 2 characters from the buffer
Takes effect	if MR 14 > 1	

Tab. 35: Receive buffer; 16-bit; big endian

MR 17

Receive buffer, 32-bit, little endian

Read access to this register removes 4 characters from the receive buffer and returns them as 32-bit value.

Assignment:

Character	Bits in register
First	Bit 0 ... 7
Second	Bit 8 ... 15
Third	Bit 16 ... 23
Fourth	Bit 24 ... 31

Property	Description
Values	-2,147,483,648 ... 2,147,483,647
Type of access	Read Removes 4 characters from the buffer
Takes effect	if MR 14 > 3

Tab. 36: Receive buffer, **32-bit, little endian**

MR 18

Receive buffer; 32-bit; big endian

Read access to this register removes 4 characters from the receive buffer and returns them as 32-bit value.

Assignment:

Character	Bits in register
First	Bit 24 ... 31
Second	Bit 16 ... 23
Third	Bit 8 ... 15
Fourth	Bit 0 ... 7

Property	Description	
Values	-2,147,483,648 ... 2,147,483,647	
Type of access	Read	Removes 4 characters from the buffer
Takes effect	if MR 14 > 3	

Tab. 37: Receive buffer; **32-bit; big endian**

MR 19

Error counter

This register shows the number of detected errors.

Property	Description
Values	0 ... 2,147,483,647
Type of access	Read/write (clearing)

Tab. 38: Error counter

8.4.3 Programming

Activating the serial interface

Module register MR 7 lets you activate the user-programmable serial interface.

Configuring the interface

- ✓ The JCM-501-E01 and the device to communicate with have been wired according to the RS-232 interface standard.
- ► Enter value 1 into MR 7.

⇨ The serial interface is activated.

Configuring the serial interface

Module registers MR 1 … MR 6 are used to configure the user-programmable serial interface.

Configuring the interface

✓ The JCM-501-E01 and the device to communicate with have been wired according to the RS-232 interface standard..

1. Enter value 2 into MR 1.
2. Enter the desired communication parameters into MR 2 … MR 6.

⇨ The serial interface is set as a user-programmable interface. Transmit and receive buffer are cleared.

Sending characters

A character is sent by entering it into the register *Transmit buffer*.

Sending characters

✓ The user-programmable serial interface is configured.

1. Check the transmit buffer filling level, whether there is enough space in the transmit buffer.
2. If there is no space in the transmit buffer, wait, until there is enough space.
3. Enter the character to be sent into register *Transmit buffer*.

⇨ The character is written into the transmit buffer and will be sent from there.

Receiving characters

Characters are received by reading characters from register *Receive Buffer*.

Receiving characters

✓ The user-programmable serial interface is configured.

1. Check the filling level of the receive buffer to make sure that it contains at least 1 character.
2. Read the character from the register *Receive buffer*.

⇨ The character is taken from the receive buffer.

Sending Texts

An easy way to send texts via user-programmable serial interface is redirecting the instructions `DisplayText()` and `DisplayText2()` to **Device 9**.

Restrictions

When redirecting the instructions `DisplayText()` and `DipslayText2()` to the user-programmable serial interface the following restrictions apply:

– The cursor position will not be taken into account.

– The characters for "Delete Screen" and "Delete to End of Line" are of no special significance and will be output without any changes.

Sending Texts

✓ The user-programmable serial interface is configured.

✓ The user is familiar with the options of the instructions `DisplayText()` and `DisplayText2()` (refer to the online help which comes with JetSym).

1. Use the instruction `DisplayText()` or `DisplayText2()`.
2. Specify **Device 9**.

⇨ The task waits at this instruction until all characters have been entered into the transmit buffer.

Sending values

A simple way to send values via user-programmable serial interface is to redirect the `DisplayValue()` command to **Device 9**.

Restrictions

When redirecting instruction `DisplayValue()` to the user-programmable serial interface, the following restriction applies:

– The cursor position will not be taken into account.

Sending values

✓ The user-programmable serial interface is configured.

✓ The user is familiar with the options of the instruction `DisplayValue()` (refer to the online help which comes with JetSym).

1. Set the special registers for formatting the display, which are used in connection with the instruction `DisplayValue()`, to the desired values.
2. Use the instruction `DisplayValue()`.
3. Specify **Device 9**.

⇨ The task waits at this instruction until all characters have been entered into the transmit buffer.

Receiving values

Values are received by reading characters from registers MR 15 through MR 18 *Receive buffer registers*.

Receiving values

✓ The user-programmable serial interface is configured.

1. Check the filling level of the receive buffer to make sure that it contains at least 2 or 4 characters.
2. Read the values from *Receive buffer* registers MR 15 through MR 18.

⇨ The characters are read from the receive buffer.

8.5 Real-time clock

A real-time clock is used for time measurement. If there is a permanent connection between ECU_PWR of the controller and the vehicle battery, the real-time clock will continue to run for a certain period of time even when the controller is switched off. The real-time clock has no automatic daylight savings time function.

The JCM-501-E01 has a built-in battery with a service life of at least 10 years.

8.6 Storage options - Overview

The JCM-501-E01 features several types of program and data memory. There is volatile and non-volatile memory. Volatile memory loses its content at switching off. Non-volatile memory keeps its content even when the power supply is off. This type of memory is also referred to as remanent memory, and it is located directly in the CPU or in separate I/O modules.

This chapter gives an overview of the available memory.

8.6.1 File system memory

Data and program files are stored in the file system memory.

Features

– Non volatile
– Size of the internal flash disk: 368 MB

Access

– By the operating system
– By JetSym
– By means of file commands from within the application program

8.6.2 Application program memory

By default, the application program is uploaded from JetSym to the JCM-501-E01 and is stored there.

Properties

– Stored as file within the file system
– Default directory *app**program name*
– Files may also be stored to other directories (or USB flash drive)

Type of access

– By operating system
– By JetSym
– By means of file commands from within the application program

8.6.3 Flag

Flags are one-bit operands. This means they can be either TRUE or FALSE.

Properties of user flags

– Global variables with dedicated addresses (%MX)
– Non-volatile
– Quantity: 256
– Flag numbers: 0 ... 255

Properties of overlaid user flags

– Global variables with dedicated addresses (%MX)
– Non-volatile
– Overlaid by registers 1000000 ... 1000055
– Quantity: 1,792
– Flag numbers: 256 ... 2047

Properties of special flags

– Global variables with dedicated addresses (%MX)
– When the operating system is launched, special flags are initialized using their default values
– Quantity: 256
– Flag numbers: 2048 ... 2303

8.6.4 Storing registers and variables

Storing to a non-volatile (remanent) memory

Registers and variables of the application program can be globally stored to a non-volatile memory with dedicated registers.
The register variables %VL are always 4 bytes in size. They are not initialized by the operating system.
These %RL registers always occupy 480,000 bytes.
The number of %VL register variables is limited to 120,000.
The register array for %VL or %RL ranges from 1000000 to 1119999

Storing to volatile (non-remanent) memory

Variables of the application program can be stored to volatile memory.
Global variables that do not have a static assignment to addresses and are stored compactly.
Their register number starts with the value 0.

8.7 Automatic copying of controller data

This chapter describes the Autostart feature allowing for data to be copied within the JCM-501-E01. To this end, you can create a command file which is then stored along with the data to a USB flash drive. This command file is then automatically processed when the USB flash drive is inserted.

Functions within the local file system

The Autostart function executes the following functions:

– Creating directories
– Deleting directories
– Copying files
– Deleting files

Areas of application

The Autostart function is typically used in situations where

– remote maintenance is not possible;

– there is no PC on site;
– the operator is not able or should not be allowed to make modifications to the plant.

The following actions can be taken using the Autostart function:
– Modify the application program
– Modify the application data
– Modify the controller configuration
– Operating system update (JCM-501-E01)

Prerequisites

For automatic copying of controller data, the following prerequisites must be fulfilled:
– The programmer must be familiar with the file system.

Designation

In this description, *Full Name* means the name of the file or directory including its full path.

8.7.1 Loading the Autostart function

The AutoStart function should only be carried out during the boot phase (after power-up) of the JCM-501-E01.

Relevant directories

	Value	Remarks
File name	**autostart.bat**	All lower case letters
Directory - USB	**\USB**	Root directory on the USB flash drive

Tab. 39: Relevant directories

Loading the Autostart function

✓ You have created the command file and stored it to the respective directory.
1. Insert the USB flash drive into the USB port.
2. Switch the device ON.

⇨ The device boots and loads the **autostart.bat** file.

8.7.2 Executing the AutoStart function

During the boot phase in *autostart* mode, the device executes the commands in the command file.

Executing the Autostart function

The operating system of the JCM-501-E01 processes the autostart function in the following steps:

Step	Description
1	The file **\USB\autostart.bat** is loaded from the USB stick.

Step	Description
2 … n	The commands are processed in ascending order.

8.7.3 Terminating the AutoStart mode

The last command in the **autostart.bat** file terminates the *Autostart* mode.

Restart

As an option, you can reboot the device by placing the following command in the last line of your command file:

```
del \System\reset.exe
```

Exiting Autostart mode

- ► To exit the *Autostart* mode, remove the USB flash drive.

8.7.4 autostart.bat - Structure

The command file is a text file with command and comment lines:

– You can insert blank lines as required.

– Start each comment line with *REM*.

In these sections you can specify commands which are then executed by the Autostart function.

Processing commands

The Autostart function processes the commands in descending order. Furthermore, this function lets you insert jumps into the processing sequence. A jump is defined by the keyword `goto` and a destination. The destination must be preceded by a colon (:).

A **simple jump** is defined as follows:

```
goto end
{skipped block}
:end
```

A **conditional jump** is defined as follows:

```
if {condition} goto end
{skipped block}
:end
```

Available commands

The commands correspond to common Windows® command line syntax.

8.7.5 Example of a command file

Task

Add new functions to an installed JCM-501-E01. To this end, the following modifications are required:

– Operating system update
– New application program
– New values for some of the registers

Solution

You copy the required files to a USB flash drive and create a command file for the Autostart function.

USB flash drive - Content

The USB flash drive contains the following files:

– The **autostart.bat** file
– The new OS
– A *.da file containing the new register values
– A **start.ini** file and a *.es3 file containing the new application program

Command file

```
@echo off
if not exist \app\autostart.exe goto end
if exist \app\update.ini goto end
if exist \app\usb_update_process_marker goto dm
echo \data\update\jvm_ce0_1.00.0.04.os >\app\update.ini
md \data\update
copy \%1\jvm_ce0_1.00.0.04.os \data\update\jvm_ce0_1.00.0.04.os
echo >\app\usb_update_process_marker
md \app\progtest
copy \%1\progtest\progtest.es3 \app\progtest\progtest.es3
copy \%1\register.da \app\register.da
del \System\reset.exe
goto end
:dm
del \app\usb_update_process_marker
:end
```

8.7.6 Data files

File format

The data file consists of the following elements:

– Pure text file
– Each entry must be in a separate line of text
– Each line must be completed by carriage return/line feed
– Comment lines must be preceded by a semicolon (;)
– Each data file must start with the entry *SD1001*

Data lines

A data line consists of the following elements:

– ID of the variable at the beginning of the line
– Now follows the number of the variable separated by a blank or tab

– Then follows the value of the variable separated by a blank or tab

Variable ID	Variable type
FS	Flag
RS	Integer register
QA	Floating-point registers

Example

```
SD1001
; Data File - Jetter AG;
; Registers 1000000 ... 1000005
RS 1000000 12345
RS 1000001 2
RS 1000002 -1062729008
RS 1000003 502
RS 1000004 50
RS 1000005 3
QS 1009000 3.14
;
; Flags 10 ... 13
FS 10 0
FS 11 1
FS 12 1
FS 13 0
```

Automatically loading data file information

The JCM-501-E01 features a mechanism allowing a data file to be loaded automatically before the application program starts. You must name the data file **register.da** and store it in the **App** folder.

Process

The table below shows how the device retrieves data file information:

Step	Description
1	If the data file **register.da** has been stored to the directory **App**, the device will detect this file the next time the device is started.
2	The device retrieves the file.
3	The device maps the values to the registers.
4	Once the device has retrieved the file, it deletes the data file.
5	The device loads the application program.

8.8 Saving and loading an application program

The user determines the program that is to be executed. When uploading the application program to the controller, this program is stored as a file to the internal flash disk. The device enters the path and file name into the **\App\start.ini** file.

Path and file name

By default, JetSym creates in the */app* directory a subdirectory and assigns the project name to it. Then, JetSym stores the application program to this subdirectory assigning the extension ***.es3** to it. Path and file names are always converted into lower case letters.

\App\start.ini - Structure

This file is a text file with one section holding 2 entries:

Element	Description
[Startup]	Section name
Project	Path to the application program file. This path is relative to \ *app*.
Program	Name of the application program file

Example

```
[Startup]
Project = test_program
Program = test_program.es3
```

The application program is loaded from the file **\App\test_program\test_program.es3**.

Loading the application program

When the application program is restarted via JetSym or after booting the JCM-501-E01, the application program is loaded via the file system and executed.

The application program is loaded by the OS of the JCM-501-E01 as follows:

Step	Description
1	The operating system reads the file **\App\start.ini** from the internal flash disk.
2	The OS evaluates the **Project** entry. It contains the path leading to the application program file.
3	The OS evaluates the **Program** entry. This entry contains the program name.
4	The OS loads the application program from the file **<Project>/<Program>**.

9 Registers - Overview

This register overview gives a condensed summary of the registers and flags of the JCM-501-E01 device.

Default address on the CANopen bus

The pre-set default address on the CANopen bus is:

Node ID	127 (0x7F)

General overview - Registers

Register range	Description
100000 ... 100999	Electronic Data Sheet (EDS)
106000 ... 106999	CAN
108000 ... 108999	CPU/backplane
200000 ... 209999	General system registers
210000 ... 219999	Application program
310000 ... 319999	File system/data files
1000000 ... 1029999	Application registers (non-volatile)

Flags - General overview

Flag	Description
0 ... 255	Application flags (non-volatile)
256 ... 2047	Overlaid by registers R 1000000 through 1000055
2048 ... 2303	Special flags

Electronic Data Sheet (EDS)

EDS data can be retrieved for the CPU or the entire device.

Register range	Description
100500	**Interface**
0	CPU
1	Baseboard
100600 ... 100614	**Identification**
100600	Internal version number
100601	Module ID
100602 ... 100612	Module name (register string)
100613	PCB revision
100614	PCB options
100700 ... 100712	**Production**
100700	Internal version number
100701 ... 100707	Serial number (register string)
100708	Day
100709	Month
100710	Year
100711	TestNum.
100712	TestRev.
100800 ... 100802	**Features**
100800	Internal version number
100801	MAC address (Jetter)
100802	MAC address (JCM-501-E01)

Electronic nameplate (entire device)

Register range	Description
100708	Day
100709	Month
100710	Year
100900 … 100907	**Production**
100900	Internal version number

Register range	Description
100901 ... 100907	Serial number (register string)
100950 … 100993	**Identification**
100950	Internal version number
100951	Module ID
100952 ... 10 0962	Module name (register string)
100965	Config ID
100966	Vendor ID
100967	Variant ID
100968	Type ID
100992	Navision ID
100993	FBG version

Real-time clock

Registers	Description
102910 … 102917	**Direct access**
102910	Milliseconds
102911	Seconds
102912	Minutes
102913	hours
102914	Day of the week (0 = Sunday)
102915	Day
102916	Month
102917	Year
102920 … 102928	**Buffer access**
102920	Milliseconds
102921	Seconds
102922	Minutes
102923	hours
102924	Day of the week (0 = Sunday)
102925	Day
102926	Month
102927	Year
102928	Read/write trigger

CAN

Register	Description
106000	Baud rate CAN 1
106001	Node ID CAN 1
106100	Baud rate CAN 2
106101	Node ID CAN 2

Serial port

Register	Description
103001	Protocol
103002	Baud rate
103003	Number of data bits per character
103004	Number of stop bits
103005	Parity
103107	Activating the serial interface
0	The serial interface is inactive, no data can be received/transmitted.
1	The serial interface is active, data can be received/transmitted.
103010	Sending buffer
103011	Sending buffer filling level
103012	Receiving buffer (without immediate clearing)
103013	Receiving buffer (with immediate clearing)
103014	Receive buffer filling level
103015	Receive buffer, 16-bit, little endian
103016	Receive buffer; 16-bit; big endian
103017	Receive buffer, 32-bit, little endian
103018	Receive buffer; 32-bit; big endian
103019	Error counter

Flash memory

Register range	Description
107510 … 107513	**Sector statistics**
107510	Available sectors
107511	Used sectors
107512	Blocked sectors
107513	Free sectors
107520 … 107523	**Byte statistics**
107520	Size of the flash disk in bytes
107521	Used memory in bytes
107522	Blocked memory in bytes
107523	Free memory in bytes

LEDs

Register	Description
108008	**LED D1 … D2**
Bit 0	LED D1 green
Bit 1	LED D1 red
Bit 2	LED D2 green
Bit 3	LED D2 red

System information

Register	Description
108500 … 108509	JetVM DII version string
108510 … 108519	Version string of the host application
108520 … 108529	File name of the host application
108530 … 108539	OS version (string)
108570	CPU type
108571	Number of CPUs
108573	Physical RAM
108574	Free physical RAM
108575	Memory utilization (in %)
108581	Screen width (in pixels)
108582	Screen height (in pixels)
108590	HID version

USB flash drive

Register	Description
109000	**Status**
Bit 0 = 1	USB flash drive is plugged
Bit 1= 1	USB flash drive is ready
109001	**Write protection**
1	Data medium is write-protected
109002	Memory size in MByte
109003	Free memory in MB
109004	Memory available to users in MB
109005	Memory used in MB

General system registers

Register	Description
200000	OS version (major * 100 + minor)
200001	**Application program is running (bit 0 = 1)**
0/2	Stop program
1	Start program
3	Continue program
200008	**Error register 1 (identical with 210004)**
Bit 8	Illegal jump
Bit 9	Illegal call
Bit 10	Illegal index
Bit 11	Illegal opcode
Bit 12	Division by 0
Bit 13	Stack overflow
Bit 14	Stack underflow
Bit 15	Illegal stack
Bit 16	Error when loading the application program
Bit 24	Timeout - Cycle time
Bit 25	Timeout - Task lock

Register	Description
Bit 31	Unknown error
200168	Boot loader version (IP format)
200169	OS version (IP format)
201000	Runtime register in milliseconds (rw)
201001	Runtime register in seconds (rw)
201002	Runtime register in register 201003
201003	10 ms units for register 201002 (rw)
201004	Runtime register in milliseconds (ro)
201005	Runtime registers in microseconds (ro)
202960	Password for system command register (0x424f6f74)
202961	System command register
202980	Error history: Number of entries
202981	Error history: Index
202982	Error history: Entry
203100 ... 203107	32-bit overlaying - Flag 0 ... 255
203108 ... 203123	16-bit overlaying - Flag 0 ... 255
203124 ... 203131	32-bit overlaying - Flag 2048 ... 2303
203132 ... 203147	16-bit overlaying - Flag 2048 ... 2303
209700	System logger: Global enable
209701 ... 209739	Enabling system components

Application program

Register	Description
210000	**Application program is running (bit 0 = 1)**
210001	JetVM version
210004	**Error register (bit-coded)**
Bit 8	Illegal jump
Bit 9	Illegal call
Bit 10	Illegal index
Bit 11	Illegal opcode
Bit 12	Division by 0
Bit 13	Stack overflow
Bit 14	Stack underflow
Bit 15	Illegal stack
Bit 16	Error when loading the application program
Bit 24	Timeout - Cycle time
Bit 25	Timeout - Task lock
Bit 31	Unknown error
210006	Highest task number
210007	Minimum program cycle time
210008	Maximum program cycle time
210009	Current program cycle time
210011	Current task number
210050	Current program position within an execution unit
210051	ID of the execution unit being processed
210056	Desired total cycle time in µs
210057	Calculated total cycle time in µs
210058	Maximum time slice per task in µs
210060	Task ID (for R210061)
210061	Priority for task [R210060]
210063	Length of scheduler table
210064	Index in scheduler table
210065	Task ID in scheduler table
210070	Task ID (for R210071)
210071	Timer number (0 ... 31)
210072	Manual triggering of a timer event (bit-coded)
210073	End of cyclic task (task ID)
210074	Command for cyclic tasks
210075	Number of timers
210076	Timer number (for R210077)
210077	Timer value in milliseconds
210100 ... 210199	Task state Please use the STX function TaskGetInfo() as described in the JetSym online help.

Register	Description
210400 ... 210499	Task - Program address
210600	Task ID of a cyclical task (for R210601)
210601	Processing time of a cyclical task in per mil figure
210609	**Task lock timeout in ms**
-1	Monitoring disabled
210610	Timeout (bit-coded, bit 0 → timer 0 etc.)

File system/data file function

Register	Description
312977	Status of file operation
312978	Task ID

Application registers

Register	Description
1000000 ... 1005999	32-bit integer (remanent)

32 combined flags

Registers	Description
203100	0 ... 31
203101	32 ... 63
203102	64 ... 95
203103	96 ... 127
203104	128 ... 159
203105	160 ... 191
203106	192 ... 223
203107	224 ... 255

16 combined flags

Registers	Description
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 combined special flags

Registers	Description
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 combined special flags

Registers	Description
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

Registers	Description
203144	2240 ... 2255
203145	2256 ... 2271

Registers	Description
203146	2272 ... 2287
203147	2288 ... 2303

System Functions

For reasons of compatibility, the system functions are listed below.
In JetSym STX, use the corresponding JetSym STX functions instead of the system functions.

System function	Description
4	Converting BCD to HEX
5	Converting HEX to BCD
20	Square root
21	Sine
22	Cosine
23	Tangent
24	Arc sine
25	Arc cosine
26	Arc tangent
27	Exponential function
28	Natural logarithm
29	Absolute value
30	Separation of digits before and after the decimal point
50	Sorting register values
90	Writing a data file
91	Appending a data file
92	Reading a data file
96	Deleting a data file

System function	Corresponding JetSym STX function
4	Function Bcd2Hex(Bcd: int): Int;
5	Function Hex2Bcd(Hex: int): Int;
50	Function QSort(DataPtr: Int, ElementCnt: Int, ElementSize: Int, SortOffset: Int, SortType: STXBASETYPE, SortMode: QSORTMODE): Int;
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;

10 Maintenance

This product is maintenance-free. The operation of the product does not required inspection or maintenance efforts.

10.1 Repairs

Defective components can cause dangerous malfunctions and compromise safety.
Only the manufacturer is authorized to perform repair work.
It is forbidden to open the product.

Product modifications

Modifications and alterations to the product and its functions are not allowed. The manufacturer exempts from liability for any modifications made to the product.

The original parts are specifically designed for the product. Parts and equipment from other manufacturers must not be used.

The manufacturer exempts from liability for any damage resulting from the use of non-original parts and equipment is excluded.

10.2 Storage and Shipment

Storage

When storing the product, observe the environmental conditions given in chapter “Technical specifications”.

Shipment and packaging

The product contains electrostatically sensitive components which can be damaged if not handled properly. Damages to the product can impair its reliability.

To protect the product from impact or shock, it must be shipped in its original packaging, or in an appropriate protective ESD packaging.
If the packaging is damaged, inspect the device for any visible damage, and immediately inform your freight forwarder and Jetter AG of the damage caused during shipment. It is strictly forbidden to use a product that has been damaged or dropped.

10.3 Return and Disposal

The WEEE icon (crossed-out wheeled bin) says that end users must not dispose of waste electrical and electronic equipment together with household waste, but separately in an appropriate way. Applicable local environmental directives and regulations must be complied with.



Fig. 10: WEEE icon – crossed out wheeled bin

Disposal options

To do so, you may either resort to professional disposal service providers or return waste electronics to Jetter AG.

Find detailed information along with the required Return Delivery Form on our website.

Batteries

Prior to disposing of waste electronics, remove any batteries where this is possible in a safe and non-destructive way. Dispose of batteries properly.

Personal data

It is the responsibility of the end user to delete any personal data stored on waste electric and electronic equipment prior to disposal.

Packaging material

The product's packaging materials must not be disposed of together with household waste. Find detailed information on how to return packaging material to Jetter AG on our website.

11 Service

11.1 Technical support

In case of questions, suggestions, or issues, please contact our experts from Technical Support. You may reach out by phone or through the contact form on our homepage:

Technical Support | www.bucherautomation.com

Or email us:

support@bucherautomation.com

Please supply the following information when contacting Technical Support:

– Hardware revision and serial number
  The hardware revision and serial number is printed on the nameplate of the product.
– OS version
  To determine the operating system version, use the functions of the development environment.

12 Spare parts and accessories

NOTICE



Inadequate accessories might cause damage to the product

Parts and equipment from other manufacturers might impede the function of the device and cause damage to the product.

► Only use accessories recommended by Jetter AG.

12.1 Accessories

ⓘ Info

Ordering Accessories

The accessories are not part of the scope of delivery.

Suitable accessories can be obtained from Jetter AG.

Accessories	Item number
Connector set consisting of Deutsch housing, crimp contacts (female)	10001264
Connecting cable consisting of Deutsch housing, pre-assembled and tested with 1.5 m open stranded wire	60878741
Connecting cable consisting of Deutsch housing, assembled and tested with 0.5 m open wire, Sub-D for CAN and power plug & switch	60878737
Interconnecting programming cable consisting of Deutsch housing on both sides, assembled and tested, CAN1 brought out to Sub-D	60882076

List of figures

Fig. 1 LED indicators 9
Fig. 2 Nameplate 10
Fig. 3 Dimension in mm 11
Fig. 4 Space requirements for installation work (in mm) 17
Fig. 5 Screw holes, dimensions in mm 18
Fig. 6 Installing the strain relief 18
Fig. 7 Deutsch connector, 12 pins 19
Fig. 8 Software version numbers 21
Fig. 9 Block Diagram - Serial Interface 32
Fig. 10 WEEE icon – crossed out wheeled bin 56

List of tables

Tab. 1 Mechanical specifications 12
Tab. 2 Power supply VBAT_ECU 12
Tab. 3 CAN interfaces 12
Tab. 4 Specification - CAN bus cable 13
Tab. 5 Cable lengths 13
Tab. 6 USB port 13
Tab. 7 RS-232 interface specification 14
Tab. 8 Environmental conditions 14
Tab. 9 Acoustic signal generator 14
Tab. 10 Pulses ISO 7637-2 15
Tab. 11 Irradiation ISO 11452 15
Tab. 12 ESD EN 61000-4-2 15
Tab. 13 Unsuitable installation locations 17
Tab. 14 Format of software version numbers 21
Tab. 15 Registers - Overview 22
Tab. 16 JetEasyDownload parameters 24
Tab. 17 Abbreviations 27
Tab. 18 Module register properties 27
Tab. 19 Numerical formats 27
Tab. 20 Supported STX functions 28
Tab. 21 Elements of the serial interface 32
Tab. 22 Register numbers of the serial interface 34
Tab. 23 Protocol 35
Tab. 24 Baud rate 35
Tab. 25 Number of data bits per character 35
Tab. 26 Stop bits 35
Tab. 27 Parity 36
Tab. 28 Activating the serial interface 36
Tab. 29 Sending buffer 36
Tab. 30 Sending buffer filling level 37
Tab. 31 Receive buffer, 8 bits (without deleting the character on reading) 37
Tab. 32 Receive buffer, 8 bits (with deleting the character on reading) 37
Tab. 33 Receive buffer filling level 37
Tab. 34 Receive buffer, 16-bit, little endian 38
Tab. 35 Receive buffer; 16-bit; big endian 38
Tab. 36 Receive buffer, **32-bit, little endian** 39
Tab. 37 Receive buffer; **32-bit; big endian** 39

Tab. 38 Error counter 39
Tab. 39 Relevant directories 44

Bucher Automation AG
Thomas-Alva-Edison-Ring 10
71672 Marbach/Neckar, Germany
T +49 7141 2550-0
info@bucherautomation.com

www.bucherautomation.com