Jetter automation


# User Manual

## JC-970MC - Controller with Integrated Industrial PC

608 837 51

We automate your success.

| | |
|---|---|
| Item number: | 608 837 51 |
| Language of the original document: | German |
| Document revision: | 1.12.2 |
| Date of issue: | February 2018 |

**Address**

How to contact us:

Jetter AG
Graeterstrasse 2
71642 Ludwigsburg, Germany
Germany

| | |
|---|---|
| Phone - Switchboard: | +49 7141 2550-0 |
| Phone - Sales: | +49 7141 2550-433 |
| Phone - Technical Hotline: | +49 7141 2550-444 |
| Fax - Sales: | +49 7141 2550-484 |
| E-mail - Sales: | sales@jetter.de |
| E-mail - Technical Hotline: | hotline@jetter.de |


**Reference to products**

This manual is an integral part of the JC-970MC control system:

Type: _____

Serial #: _____

Year of manufacture: _____

Order #: _____

CE

To be entered by the customer:

Inventory #: _____

Place of operation: _____

**Significance of this user manual**

This document is an integral part of the JC-970MC:

- Keep this document in a way that it is always at hand until the JC-970MC will be disposed of.
- Pass this document on if the JC-970MC is sold or loaned/leased out.

In any case you encounter difficulties to clearly understand the contents of this document, please contact Jetter AG.

We would appreciate any kind of suggestions and contributions on your part. Please send your suggestions and comments to the e-mail address info@jetter.de. Your feedback will help us produce manuals that are more user-friendly, as well as address your wishes and requirements.

In this document you will find useful information on the following topics:

- Shipping
- Mechanical installation
- Electrical installation
- Programming
- Operation
- Maintenance
- Repairs

Therefore, you must carefully read, understand and observe this document, and especially the safety instructions.

In the case of missing or inadequate knowledge of this document Jetter AG shall be exempted from any liability. Therefore, the operating company is recommended to obtain the persons' confirmation that they have read and understood this manual in writing.

# Table of contents

# 1 Safety instructions

**Introduction**

This chapter contains safety instructions, information on residual dangers and measures for increasing electromagnetic immunity.

**Content**

## Basic safety instructions

**Introduction**

This device complies with the valid safety regulations and standards. Jetter AG attaches great importance to the safety of the users.

Of course, the user should adhere to the following regulations:

- Relevant accident prevention regulations
- Accepted safety rules
- EC guidelines and other country-specific regulations

**Intended use**

Usage according to the intended conditions of use includes operation in accordance with these operating instructions.

The controller JC-970MC is used to control machinery, such as conveyors, production machines, and handling machines.

Operate the JC-970MC only within the limits and conditions set forth in the technical specifications. The operating voltage of the JC-970MC is classified as SELV (Safety Extra Low Voltage). Therefore, the JC-970MC is not subject to the EU Low Voltage Directive.

**Usage other than intended**

This device must not be used in technical systems which to a high degree have to be fail-safe.

The JC-970MC is no safety-related part as per Machinery Directive 2006/42/EC. This device is not qualified for safety-relevant applications and must, therefore, NOT be used to protect persons.

If you intend to operate the device at ambient conditions not being in conformity with the permitted operating conditions, please contact Jetter AG beforehand.

**Personnel qualification**

Depending on the life cycle of the product, the persons involved must possess specific qualifications. These qualifications are required to ensure proper handling of the device in the corresponding life cycle.

| Product life cycle | Minimum qualification |
|---|---|
| **Transport/storage:** | Trained and instructed personnel with knowledge in handling electrostatically sensitive components |
| **Mounting/installation:** | Specialized personnel with training in electrical engineering, such as industrial electronics technician. |
| **Commissioning/ programming:** | Trained and instructed experts with profound knowledge of, and experience with, electrical/drive engineering, such as electronics engineer for automation technology. |
| **Operation:** | Trained, instructed and assigned personnel with knowledge in operating electronic devices. |
| **Decommissioning/ disposal:** | Specialized personnel with training in electrical engineering, such as industrial electronics technician. |

**Modifications and alterations to the device**

**For safety reasons, no modifications and changes to the device and its functions are permitted.**

Any modifications to the device not expressly authorized by Jetter AG will result in a loss of any liability claims to Jetter AG.

| | |
|---|---|
| | **The original parts are specifically designed for the device. Parts and equipment from other manufacturers have not been tested by Jetter AG and are, therefore, not released by Jetter AG.**<br>The installation of such parts may impair the safety and the proper functioning of the device.<br>Any liability on the part of Jetter AG for any damages resulting from the use of non-original parts and equipment is excluded. |
| **Shipping** | The device contains electrostatically sensitive components which can be damaged if not handled properly.<br>To exclude damages to the device during transport it must be shipped in its original packaging or in packaging protecting against electrostatic discharge.<br>▪ Use an appropriate outer packaging to protect the device against impact or shock.<br>▪ In case of damaged packaging inspect the device for any visible damage. Inform your freight forwarder and Jetter AG. |
| **Storing** | When storing the device observe the environmental conditions given in the technical specification. |
| **Repairing and maintaining** | The operator is not allowed to repair the device. The device does not contain any parts that could be repaired by the operator.<br>If the device needs repairing, please send it to Jetter AG. |
| **Disposal of obsolete equipment** | When disposing of devices, the local environmental regulations must be complied with. |

## Instructions on EMC

**Noise immunity of a system**

The noise immunity of a system is determined by the weakest component of the system. For this reason, correct wiring and shielding of cables is of paramount importance.

**Measures**

Measures for increasing EMC in electric plants:

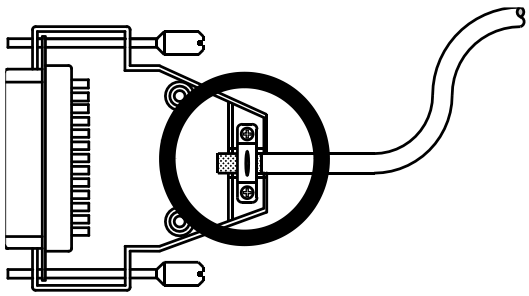
- Follow the instructions given in Application Note 016 *EMC-Compatible Installation of the Electric Cabinet*.

**The following instructions are excerpts from Application Note 016:**

- **Physically separate** signal and power lines. Jetter AG recommend spacing greater than 20 cm. Cables and lines should cross each other at an angle of 90°.
- The following line cables must be shielded:
  Analog lines, data lines, motor cables coming from inverter drives (servo output stage, frequency converter), lines between components and interference suppressor filter, if the suppressor filter has not been placed at the component directly.
- Shield cables **at both ends**.
- Unshielded wire ends of shielded cables should be as short as possible.
- Draw the **entire shield** behind the isolation, and then clamp it under an earthed strain relief with the **greatest possible surface area**.

**When male connectors are used:**

- Draw the shield, in its entire perimeter, under the shielding clamp of the metallized connector housing (impedance shielding), respectively of the EMC gland bushing, its greatest possible surface area being clamped under a strain relief.
- Only use metallized connectors, e.g. Sub-D with metallized housing. Make sure that the strain relief is directly connected with the housing here as well.



**Downloading Application Note 016**

You can download Application Note 016 from the Jetter AG **Homepage http://www.jetter.de**. You will find Application Note 016 *EMC-compatible installation of electric cabinets* at *Downloads - Application Notes*.

**Notice!**

If you want to route USB ports to the outside of an electric cabinet, e.g. using a Weidmüller USB adaptor IE-FCM-USB-AB (1222550000), please use USB cables by Lindy (e.g. item number 41612, length 2 m, www.lindy.de).“

**Notice!**

As connection cable between DisplayPort and display, we recommend DisplayPort cables from Lindy (e.g. item number 41533, length 3 m, www.lindy.de).

# 2 Product description and configuration options

**Introduction**

This chapter covers the design of the device, as well as how the order reference is made up including all options.

**Content**

# Product description – JC-970MC

**The controller JC-970MC**

The JetControl 970MC is a PC-based high-end controller of excellent performance for control tasks.

It is logically divided into a controller (hereinafter referred to as JetControl or realtime partition) and an industrial PC with Windows® operating system. The industrial PC performs tasks that do not have to be in real-time, such as visualization tasks, storage of data, e.g. recipes, or database applications.

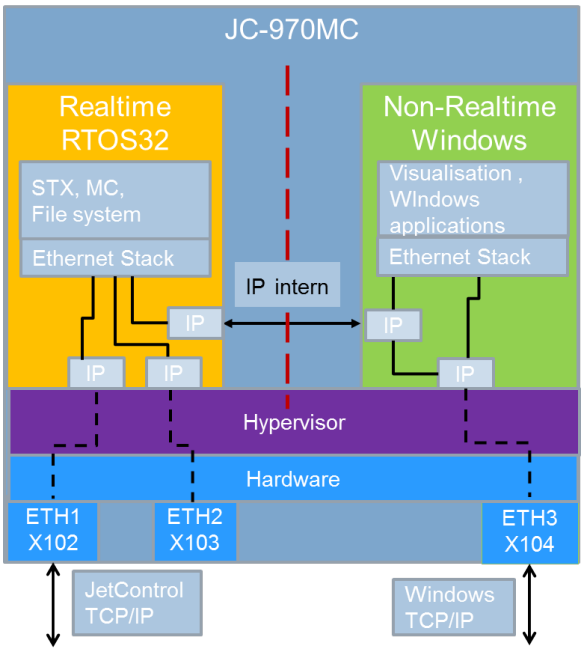
The controller, on the other hand, takes on hard real-time tasks, such as motion control.

The JC-970MC offers a motion control feature which lets the user program axis groups and complex path controls. Special attention was given to straightforward and efficient realization of complex path controls.

The controller can be integrated into a network via its Ethernet port.

The compact case of the controller JC-970MC has a fanless design and is equipped with a heat sink. The controller has been designed for being installed in a control cabinet.

Windows® PC operating systems do not have the real-time capability required to control industrial plants. Therefore, the JC-970MC uses a real-time operating system (RTOS) in parallel with Windows OS. This design allows deterministic processing of real-time tasks. Resources are allocated between these two operating systems by the virtualization software Hypervisor. The advantage of the virtualization approach is that RTOS is kept operational even when the Windows OS crashes. Furthermore, Windows can be shut down separately, while the controller continues to run. The strict separation of these two operating systems ensures a high level of stability. There is a direct connection between Windows and RTOS via an internal IP interface. This internal interface has got the IP address 192.168.2.3.



The two Ethernet ports X102 and X104 allow for access/remote access to the controller. Thus, remote maintenance of the Windows IPCs (via X104) and/or of the controller itself (via X102) is possible.

Controller and Windows IPC have got separate mass storages. For the IPC an SSD is used, for the controller a flash disk. This lets you easily update the controller via USB flash drive using the AutoCopy feature.

The register interface allows for direct data base access.

The JC-970MC can be equipped with two JI-PCIE-xxx cards. These cards must already be included in the order as they cannot be retrofitted.

Remote peripheral modules can be connected via Jetter Ethernet system bus.

Control and visualization are combined in one device. This reduces the number of components in the control cabinet

## Product features

The product features are listed below:


- Hypervisor technology is used to separate the controller's OS from Windows.
- 250 point-to-point (PTP) axes via Ethernet
- 64 MC axes via Ethernet
- 60 point-to-point (PTP) axes via JX2 system bus (equipped with JX6-SB-I)
- 2 Ethernet ports (1 x Windows, 1 x RTOS)
- 4 USB ports (2 x Windows, 2 x RTOS)
- Powerful programming language JetSym STX
- Non-volatile registers (NVRAM): 120,000
- SDRAM: 64 MB as shared data and program memory
- Flash disk: 32 MB (RTOS)
- SSD 60 GB (Windows)
- Real-time clock
- Integrated Web server/e-mail feature
- Modbus/TCP

## Engineering notes

JC-970MC supports the axis combinations listed below:

| Max. number of PCIe riser cards JI-PCIE-Exx | Max. number of JX6-SB-I submodules | Max. number of PTP axes per JX6-SB-I | Total number of PTP axes connected to JX6-SB-I | Max. number of MC axes via Ethernet (X102) | Max. number of PTP axes via Ethernet (X102) |
|---|---|---|---|---|---|
| 2 | 4 | 15 | 60 | 64 | 250 |

**Notice!**

JX6-SB-I: JX6-SB-I submodules do not support MC axes. Point-to-point axes only (PtP)

**Scope of delivery**

The following items are included in the scope of delivery:

| Quantity | Description |
|---|---|
| 1 | Controller JC-970MC |
| 1 | Power Subcon connector, 3-pin PSC 1.5/3-F |
| 1 | Microsoft® Software license agreement |
| 1 | Installation manual |

## Order reference/options

**Order reference**

The order reference consists of the name of the JC-970MC and the desired options. Each of the additional options given below supplements the JC-970MC. The order reference only reflects existing options.

| | | |
|---|---|---|
| JC-970MC | - | Exx |

| Element | Description |
|---|---|
| JC-970MC | Controller |
| E01 | Two PCIe slots equipped with the following submodules:<br>• JX6-SB<br>• JX6-SBI<br>• JX6-IO16<br>• JX6-SV |

| Item no. | Order reference | |
|---|---|---|
| 10000741 | JC-970MC | 64 MC axes; no JI-PCIE-Exx expansion |

**Ordering additional options**

The JC-970MC cannot be retrofitted with JI-PCIE-Exx cards by the customer. Retrofitting can be done by Jetter AG after sending in the device.

**Integrated Web server and e-mail feature**

If the controller JC-970MC is equipped with integrated Web server and e-mail feature, it is able to perform the following functions:

- **HTTP Server**: This feature lets the user download the homepages into the controller via FTP. These homepages can be accessed with any standard Internet browser.
- **SMTP client**: This feature lets the controller send e-mails.

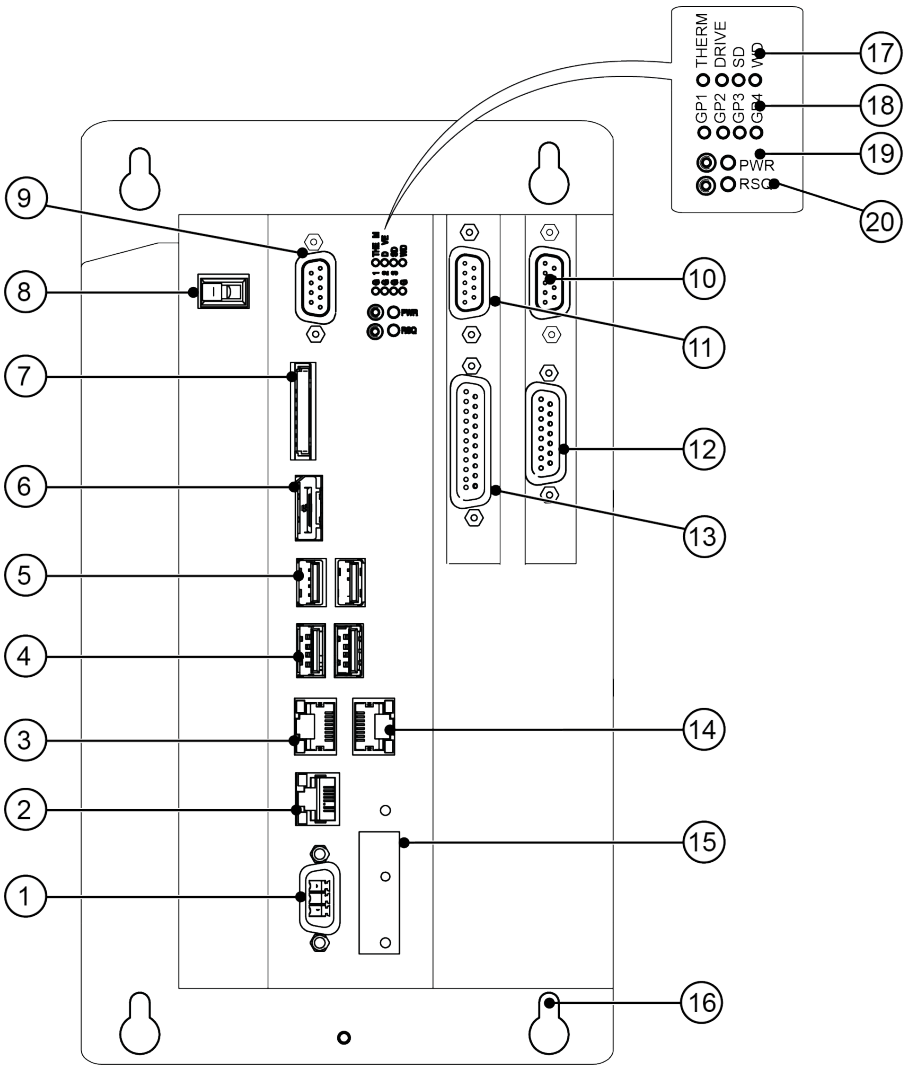
**Modbus/TCP**

The JC-970MC supports the **Modbus/TCP** protocol. The controller can act as both server (class 0, 1, 2) and client (class 0).

# Parts and interfaces

**Parts and interfaces**

The controller JC-970MC features the following parts and interfaces:




The illustration shows variant -E01.

| Number | Element | Description |
|---|---|---|
| **1** | X101 | Power supply |
| **2** | X102 (ETH) | Ethernet port for connecting programming PCs, servo amplifiers JM-200-ETH, JX3-BN-ETH, Modbus-TCP and user-programmable TCP/IP devices to the controller. |
| **3** | X103 (ETH) | Ethernet port.<br>Not used at the moment. Earmarked for real-time Ethernet, e.g. EtherCAT® |

| Number | Element | Description |
|---|---|---|
| **14** | X104 (ETH) | Ethernet interface of the Windows operating system (= IPC side. Not accessible from the controller side!)<br>An external programming PC can be connected here.<br>To access the controller, the specific internal IP address 192.168.2.3 of the JetControl must be used. |
| **4** | X105 | Bootable USB port (USB 3.0/2.0),<br>e.g. for data backup.<br>This port is only available from the IPC (Windows) side! |
| | X106 | USB port (USB 3.0/2.0)<br>For connecting a mouse, keyboard, etc.<br>This port is accessible only from the IPC side! |
| **5** | X107/108 | USB port (USB 2.0)<br>For connecting a USB flash drive (USB 2.0) for AutoCopy purposes.<br>This port is accessible only from the control system (RTOS)! |
| **6** | X109 | Display port<br>For connecting a digital monitor. |
| **7** | SDCARD | Card reader slot for SD memory cards<br>The card reader supports SD/SDHC/SDXC SD memory cards. The "SD" LED indicates card reader activity.<br>**Note:**<br>To prevent data loss, the SD card must not be removed while the "SD" LED is lit.<br>This port is accessible only from Windows OS! |
| **8** | S11<br>Mode selector | The mode selector has 3 positions:<br>RUN Once the JC-970MC is energized, it boots and launches the application program.<br>STOP Once the JC-970MC is energized, it boots, but does not launch the application program.<br>LOAD The controller boots up, but does not launch the STX application program. Instead, it switches to AutoCopy mode. |
| **9** | X110 | Serial interface RS-232<br>This interface is available only to Windows! |
| **17** | Status LEDs<br>THERM/DRIVE/SD/WD | System status indicators<br>THERM Temperature<br>DRIVE Hard disk activity<br>SD Card reader activity<br>WD Watchdog timeout |

| Number | Element | Description |
|---|---|---|
| **18** | General LEDs<br>GP1 … GP4 | Operating status indicators<br>- Controller section |
| **19** | PWR LED | Indicates that the device is energized<br>- Normal operation - green, lit<br>- Standy mode - green, flashing |
| | PWR button | The PWR button works as an interrupter which lets you shut down and restart the device.<br>See warning on page 59. |
| **20** | RSQ LED | Flashing red if the backup flash has been selected for booting. |
| | RSQ button | In the case of a corrupted BIOS flash, the RSQ button lets you force the controller to boot into a defined and safe state to allow for trouble-shooting. |
| **10** | X92 | Option (-E01):<br>SB - JX2 system bus interface for connecting non-intelligent JX2 and CANopen® modules. |
| **11** | X91 | Option (-E01):<br>SB-I - JX2 system bus interface for connecting intelligent or non-intelligent JX2 and CANopen® modules. |
| **12** | X82 | Option (-E01):<br>SV - Counter module for connecting an incremental or absolute (SSI) rotary encoder. |
| **13** | X81 | Option (-E01):<br>IO16CB - IO module for connecting digital inputs and outputs |
| **15** | X205 | Not assigned |
| **16** | Mounting holes | Machined keyholes (4x) for fixing the device in the electric cabinet. |

**Optional Interfaces**

At the moment, JC-970MC is available in the following configurations:

- JC-970MC-E01 equipped with the submodules listed below:

| Number | Element | Description |
|---|---|---|
| **13** | X81 | IO16CB - IO module for connecting digital inputs and outputs |
| **12** | X82 | SV - Counter module for connecting an incremental or absolute (SSI) rotary encoder. |
| **11** | X91 | SBI - Interface module for connecting Jetter's JX2 system bus<br>(SBI = System Bus Intelligent) |
| **10** | X92 | SB - Interface module for connecting Jetter's JX2 system bus<br>(SB = System Bus) |

## Digital JX6-I/O submodule JX6-IO16CB of the JC-970MC-E01

**Introduction**

The digital input/output module is equipped with eight galvanically isolated 24 V inputs, and eight galvanically isolated 24 V outputs with a maximum current of 0.5 A each.

**Inputs**

The inputs are type 3 inputs and their voltage range is DC -3 V ... DC +32 V. An input voltage of -3 V ... 5 V corresponds to logic 0 and an input voltage of +12 V ... +32 V corresponds to logic 1.

**Notice!**

Possible damage to the device!
If a line with reverse polarity is connected, the digital input can be destroyed.

The input resistance is 4.7 kΩ. All inputs are electrically isolated through optocouplers.

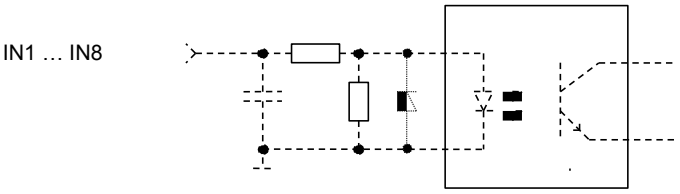
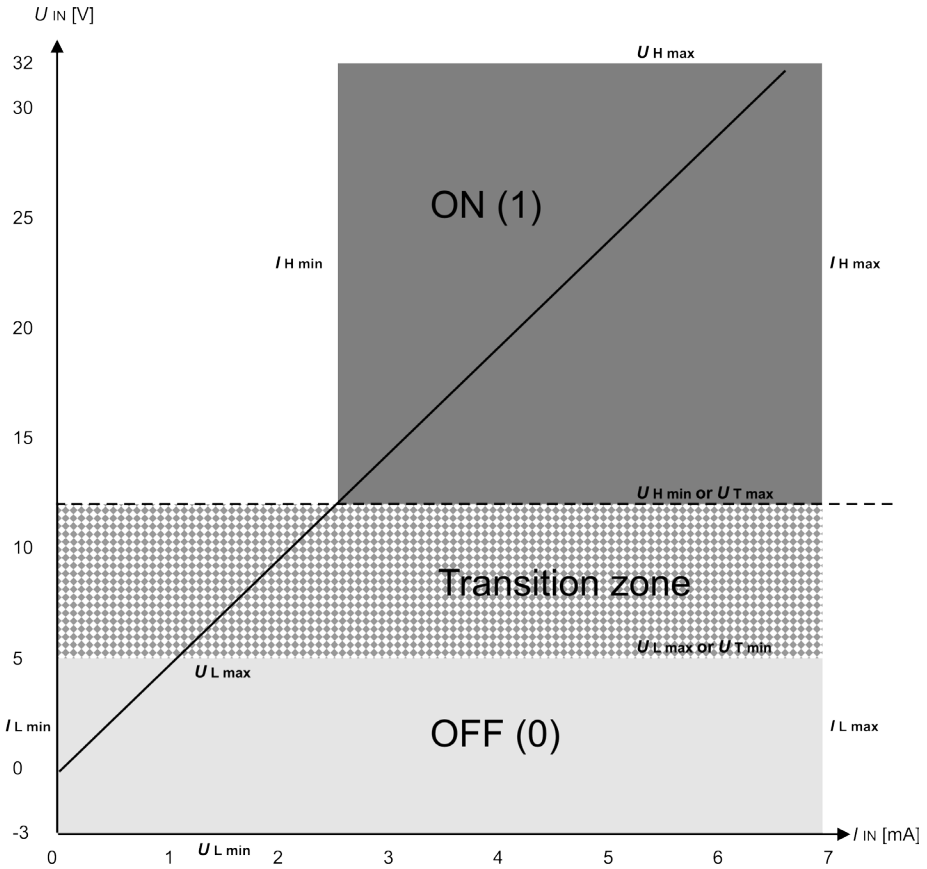



Fig.: JX6-IO16CB - input circuit

The delay time of the transition from 0 to 1 or from 1 to 0 depends on the adjustable filter and can be set between 20 µs and 16 ms. The signal evaluation is carried out statically in the application program.




Legend
UIN - Input voltage IIN - Input current
UHmax, and UHmin - Voltage thresholds for the ON (1) condition
IHmax, and IHmin - Current thresholds for the ON (1) condition
UTmax, and UTmin - Voltage thresholds for the transition between ON OFF
ULmax, and ULmin - Voltage thresholds for the OFF (0) condition
ILmax, and ILmin - Current thresholds for the ON (0) condition

Fig.: Digital inputs - Current/voltage diagram

Length of connecting cables: Assuming a standard control cable with a capacitance of 300 nF/km and an input resistance of 4.7 kΩ and the shortest pulse duration of 20 µs, this results in a maximum cable length of 4.7 m. A pulse duration of 2 ms results in a cable length of 470 m.

**Outputs**

All outputs are protected against short circuit, undervoltage and thermal overload. The permissible supply voltage range is DC 18 V ... 32 V. The propagation delay of the outputs is < 200 µs. All outputs are electrically isolated through optocouplers.

The outputs are able to drive ohmic, capacitive and inductive loads. Without an additional discharge circuit, the maximum demagnetizing power under inductive load is limited to 200 mJ. The demagnetization voltage is approx. 45 V (max. 55 V) below the supply voltage of DC 24 V.

In the event of an error (overload, short circuit), the drivers generate an error signal. This can be filtered with a digital filter.

The module can also be configured so that all outputs switch off when an error condition occurs (output register = 0).

If the supply voltage fails, all outputs are 0. The outputs are non-latching. If the controller starts in RUN mode, the outputs are switched according to the application program. If the controller starts in STOP mode, all outputs are 0.

## Technical specifications JX6-IO16CB

### General Data

| Parameter | Description |
|---|---|
| Supply voltage (device internal) | +5 V DC +/- 5 % |
| Current consumption | approx. 0.25 A |
| Ambient temperature (in operation) | 0 … 55 °C |
| Forced ventilation | No |
| Air humidity | 0 … 95 % (non-condensing) |

### Inputs DC 24 V

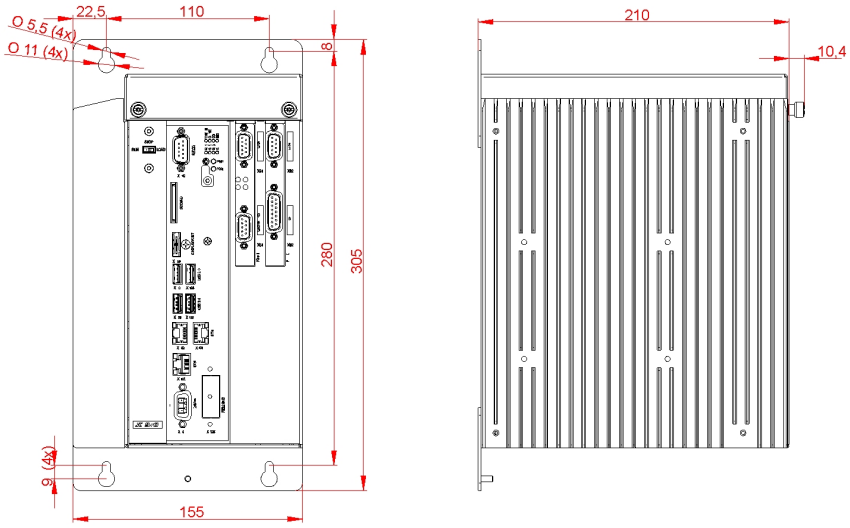
| Parameter | Description |
|---|---|
| Permissible voltage range | DC -3 … +32 V |
| Voltage range for logic 0 | DC -3 … +5 V |
| Voltage range for logic 1 | DC +12 … +32 V |
| Filtering | Adjustable |
| Input resistance | 4.7 kΩ |
| Insulation voltage | 500 V |

### Outputs DC 24 V

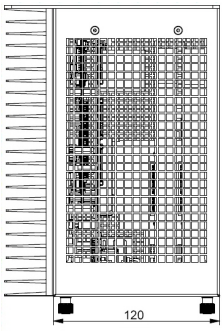
| Parameter | Description |
|---|---|
| Supply voltage | Min. DC +18 V DC, max. +32 V |
| Current consumption (all outputs disabled) | < 0.05 A |
| Voltage drop (output current 0.5 A) | 0.4 V (0.6 V max.) |
| Leakage current | Max. 300 µA |
| Short-circuit current | Typically 0.7 A (1.4 A max.) |
| Switching delay | < 200 µs |
| Switch-off delay (error condition for all outputs off) | Adjustable |
| Permissible capacitive load | <10 µF |
| Permissible inductive load | <1 H |
| Diversity factor (40 °C) | 1 |
| Diversity factor (55 °C) | 0.7 |

# Physical dimensions

## Mounting dimensions




Front view/view from right




Top view

All dimensions in mm.

**Minimum clearances**

To ensure sufficient convective air flow, the clearances mentioned below must be observed in mounting the JC-970MC in the control cabinet.

- Minimum clearance, above: 100 mm
- Minimum clearance, below: 100 mm
- Minimum clearance, right-hand side: 50 mm

**Mounting orientation**

The orientation of the JC-970MC is vertical.

# List of documentation

**Introduction**

Various documents and software tools support you in engineering, installing and programming the JC-970MC. You can download both the documents and software tools from our **homepage** at **http://www.jetter.de**.

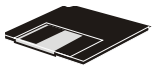
**Engineering**

The following documents and files support you in engineering the controller:

| | |
|---|---|
|  | **Industrial automation catalog** |
| | ■ Product description |
| | ■ Technical specifications |
|  | **Manual on the controller JC-970MC** |
| | ■ The document at hand |

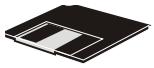
**Engineering a JX2 station on the JX2 system bus**

The following document and the following software tool support you in engineering a JX2 station on the JX2 system bus (JC-9xx):

| | |
|---|---|
|  | **JX6-SB/JX2 system bus - User information** |
| | ■ System bus topology |
| | ■ JX2 system bus specification |
|  | **System bus configurator** |
| | ■ Excel file supporting you in engineering the JX2 system bus: |
| | JX2 System Bus Configurator |

**Engineering a JX3 station on the JX3 system bus**

The following document and the following software tool support you in engineering a JX3 station on the JX3 system bus (JX3-BN-ETH):

| | |
|---|---|
|  | **User manual on the bus node JX3-BN-ETH** |
| | ■ Engineering a JX3 station |
| | ■ Product descriptions on JX3 modules |
|  | **System bus configurator** |
| | ■ Excel file supporting you in engineering the JX3 system bus: |
| | JX3 System Bus Configurator |

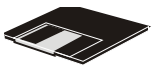
**Electrical installation**

The following documents support you in installing the controller:

| | |
|---|---|
|  | **Installation manual** |
| | It is included in the boxed JC-970MC and contains the following information: |
| | ■ Installing the controller |
| | ■ Terminal assignments |
| | ■ Specification of terminals |
| | ■ Diagnostics by means of LEDs |
|  | **User manual on the JC-970MC** |
| | ■ The document at hand |

**Programming**

The following documents and files support you in programming the controller:

| | |
|---|---|
|  | **User manual on the JC-970MC** |
| | ■ The document at hand |
|  | **JX6-SB - User information** |
| | ■ Configuring the JX2-system bus |
| | ■ Module numbering system |
| | ■ Diagnostics of modules on the JX2 system bus |
|  | **Application-oriented manuals** |
| | ■ Jetter Ethernet system bus |
| | ■ File system |
| | ■ User-programmable PRIM interface |
| | ■ System registers |
| | |
|  | **Programming tool** |
| | ■ JetSym |

# 3 Mechanical and electrical installation

**Purpose of this chapter**

This chapter supports you in mechanical and electrical installation of the JC-970MC and describes the following topics:

- Mechanical installation
- Wiring the JC-970MC
- Description of indicators
- Description of control elements

**Content**

# 3.1 Fastening and removing the JC-970MC

**Introduction**

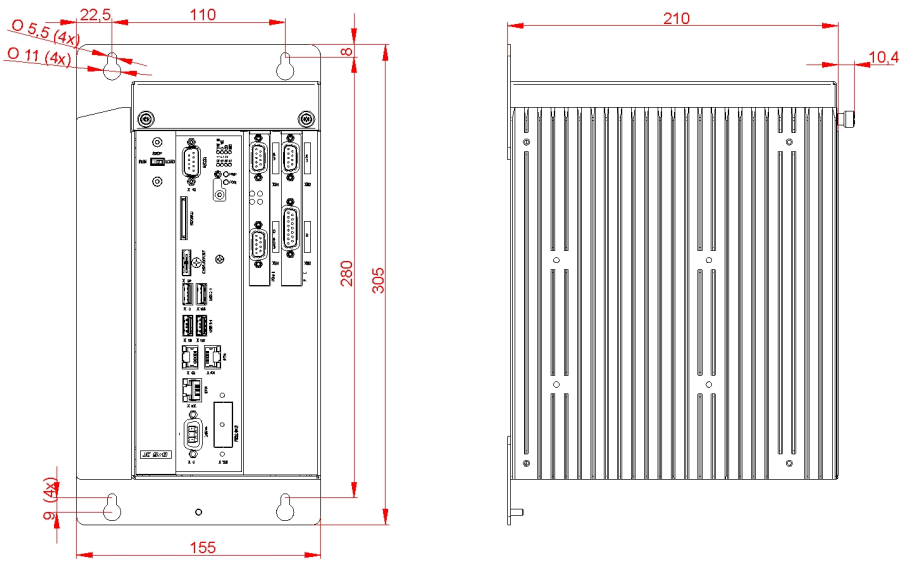
This chapter describes how to install and remove the controller JC-970MC.

## Fastening the controller JC-970MC

**Fastening**

To fasten the controller JC-970MC, proceed as follows:

| Step | Action |
|---|---|
| **1** | Hold the JC-970MC in vertical direction to the panel of the control cabinet, as is shown in the illustration below. |
| **2** | Make sure that the clearance above and below the controller is 100 mm respectively. For sufficient air circulation, a clearance of at least 50 mm must be kept on the side of the heat sink. |
| **3** | Mark on the panel four positions for the fastening screw threads.<br>The dimensions can be taken from the illustration below. |
| **4** | Drill the holes and tap the threads. |
| **5** | Screw the fastening bolts into the thread by half of their length. |
| **6** | By means of the keyholes in the rear panel, hang up the controller by the fastening screws. Then screw them down tightly. |




All dimensions in mm.

## Removing the controller JC-970MC

**Removal**

To remove the controller JC-970MC, proceed as follows:

| Step | Action |
|---|---|
| **1** | Remove power from the controller JC-970MC. |
| **2** | Disconnect all cables. |
| **3** | Unscrew the four fastening screw by half of their length. |
| **4** | Remove the controller JC-970MC from the fastening screws. |

# 3.2 Connections

**Content**

# Power supply terminal X101

**Terminal X101**

The power supply unit connects to terminal X101.

**Pin assignment**


| Pin | Description |
|---|---|
| + | Supply voltage |
| ⏚ | PE conductor |
| - | Reference potential |

**Technical specifications**

| Parameter | Description |
|---|---|
| Rated voltage | DC 24 V |
| Permissible voltage range | -15 % ... +20 % |
| Input current | 6 A max.<br>typical load 1.2 A (without USB) |
| Power consumption | 144 W max. |
| Protection against polarity reversal | Yes |

**Notice!**

Possible damage to the device!
If the power supply of the controller is outside the specification (voltage, frequency), this can lead to its destruction.

**Connector for power supply terminal X101**

The scope of delivery includes a 3-pin connector.

**Terminal**

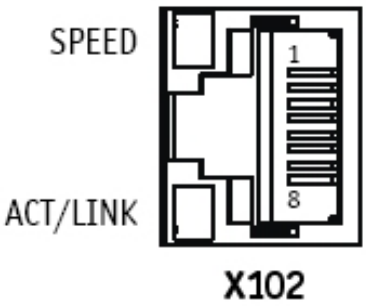
| Parameter | Description |
|---|---|
| Technology | Screw terminal |
| Screwdriver | SZS 0.6 x 3.5 |
| AWG | 16 ... 28 |
| Single conductor | 0.14 $mm^2$ ... 1.5 $mm^2$ |
| Flexible conductor | 0.14 $mm^2$ ... 1.5 $mm^2$ |
| With wire end ferrule | 0.14 $mm^2$ ... 1.5 $mm^2$ |
| Wire end ferrule with sleeve | 0.14 $mm^2$ ... 1.5 $mm^2$ |

## Ethernet interface ETH - Port X102

**Connection options**

Port X102 lets you connect, for example, a programming PC to the controller (RTOS side)

**Port X102**




| LED | Description |
|---|---|
| **ACT/LINK** | Is lit if X102 is connected to an Ethernet network. |
| **SPEED** | Indicates the connection speed:<br>Off = 10 Mbit/s<br>Green = 100 Mbit/s<br>Amber = 1000 Mbit/s |

**Technical specifications**

| Parameter | Description |
|---|---|
| Connector type | RJ45 Ethernet port |
| Bit rate | 10 MBit/s, 100 MBit/s (Cat 5e)<br>1,000 MBit/s (Cat 6) |
| Auto-crossover | Yes |

**Cables for ports X102 ... X103**

For connecting devices to ports X102, X103 and X104 you can order the following cables:

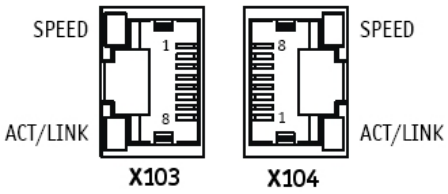
| Item no. | Item |
|---|---|
| 60537500 | Patch cable 1:1, 1 m gray Hirose, Cat 5e, shielded |
| 60854512 | Patch cable 1:1, 2 m gray Hirose, Cat 5e, shielded |
| 60854514 | Patch cable 1:1, 5 m gray Hirose, Cat 5e, shielded |
| 60854515 | Patch cable 1:1, 10 m gray Hirose, Cat 5e, shielded |

## Ethernet interface ETH - Port X103

**Connection options**

Port X103 is not used at the moment.

**Port X103**

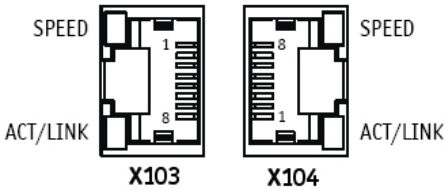



## Ethernet interface ETH - Port X104

**Connection options**

Port X104 is for connecting the integrated Windows IPC to an Ethernet network.

**Port X104**




| LED | Description |
|---|---|
| **ACT/LINK** | Is lit if X104 is connected to an Ethernet network. |
| **SPEED** | Indicates the connection speed:<br>Off = 10 Mbit/s<br>Green = 100 Mbit/s<br>Amber = 1000 Mbit/s |

**Technical specifications**

| Parameter | Description |
|---|---|
| Connector type | RJ45 Ethernet port |
| Bit rate | 10 MBit/s, 100 MBit/s (Cat 5e)<br>1,000 MBit/s (Cat 6) |
| Auto-crossover | Yes |

## USB jacks - Port X105 and X106

**Connection options**

Ports X105 and X106 (USB 3.0) connect to USB devices (USB 3.0/2.0), such as mouse, keyboard, etc.

Ports X105 and X106 can be accessed only from the Windows operating system of the IPC.

**Ports X105 and X106**


| Number | Description |
|---|---|
| **X105** | USB port 1 (USB 3.0) |
| **X106** | USB port 2 (USB 3.0) |

**Technical specifications**

| Parameter | Description |
|---|---|
| Maximum output current | 0.9 A |
| USB type | Type A (host) |
| Specification | USB 3.0 |
| Maximum permissible cable length | 3 m |

# Jacks

## Connection options

Ports X107 and X108 (USB 2.0) connect to USB flash drives. Access to the USB flash drive is possible only from the controller (RTOS).

The Autocopy function, see chapter 7 “AutoCopy - Automatic copying of controller data” on page 179 lets you automatically copy controller data from a USB flash drive to the controller or vice versa. To this end, sub-directory USB1 of the file system is used. For this, the USB flash drive must be connected to port X108.

**Notice!**

During operation, the assignment of USB1/USB2 to X107/X108 is not fix. The jack into which the first USB flash drive is plugged is automatically designated USB1. If a second USB flash drive is connected, USB2 is assigned to the corresponding jack. If the controller is rebooted with two inserted USB flash drives, X108 is automatically designated USB1.

For the AutoCopy function, the USB flash drive must be connected to the USB interface USB1. With jack X108 it is guaranteed in any case that USB1 is assigned to it when the controller is rebooted with inserted USB sticks.

➢ Therefore, plug the USB flash drive always into jack X108 for the AutoCopy function.

## Ports X107 and X108


| Number | Description |
|---|---|
| **X107** | USB port 2 (USB 2.0) |
| **X108** | USB port 1 (USB 2.0) |

## Technical specifications

| Parameter | Description |
|---|---|
| Maximum output current | 0.5 A |
| USB type | Type A (host) |
| Specification | USB 2.0 |
| Maximum permissible cable length | 5 m |

**Notice!**
If you want to route USB ports to the outside of an electric cabinet, e.g. using a Weidmüller USB adaptor IE-FCM-USB-AB (1222550000), please use USB cables by Lindy (e.g. item number 41612, length 2 m, www.lindy.de).“

## DisplayPort interface X109

**Connection options**

Port X109 connects to an external (digital) display.

**Port X109**


**Note on EMC**

If there is image interference (stripes in the image) on the connected display, the display is exposed to increased EMC noise. We recommend the use of high-quality cables and ferrite cores, which can also be retrofitted into the display cable. Ferrite cores for retrofitting are available under our item number 60240700.

**Notice!**
As connection cable between DisplayPort and display, we recommend DisplayPort cables from Lindy (e.g. item number 41533, length 3 m, www.lindy.de).

## SD card interface - Slot SDCARD

**Connection options**

Port X107 is a standardized slot for SD cards of the type SD/SDHC/SDXC.
The **SD** LED indicates card reader activity.
The card reader supports hot-plugging of SD cards and system booting.
SD cards can be accessed only from Windows OS.

**SDCARD**


**Notice!**

Loss of data!
To prevent data loss, do not remove the SD card while the "SD" LED is lit.

## Serial interface RS-232 - Port X110

**Connection options**

The serial interface is available only to Windows OS.

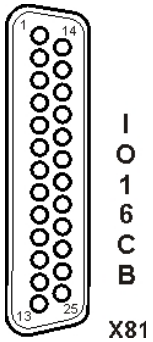
**Port X110**

## Interface for digital inputs/outputs - Port X81 (variant -E01)

**Connection options**

Port X81 is the 25-pin female Sub-D connector of the I/O module **JX6-IO16CB**. Port X81 connects to digital inputs or outputs.

**Port X81**




**Contact assignment**

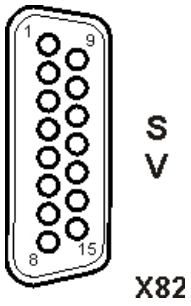
For information on pin assignment refer to chapter 3.4, starting from page 48.

## Interface for rotary encoders - Port X82 (variant -E01)

**Connection options**

The 15-pin Sub-D connector X82 **SV** connects to an incremental or absolute (SSI) rotary encoder.

**Port X82**




**Contact assignment**

For information on pin assignment refer to chapter 3.4, starting from page 48.

# 3.3 JX2 system bus interfaces (variant -E01)

**Introduction**

This chapter describes the JX2 system bus interfaces of the JC-970MC, variant -E01. Connectivity between system bus and controller is provided via PCIe bus on the controller.

**Possible number of JX2 system busses**

A riser card lets you plug two JX6-SB(-I) cards (variant E01) into the PCIe bus on the controller JC-970MC. That is, up to 2 JX2 system busses can be connected to the controller.

Submodules JX6-SB and JX6-SB-I take on converting the internal PCIe bus to the JX2 system bus.

**Supported Modules**

The JX2 system bus on controller JC-970MC supports the following type of modules:

- JX2-I/O modules
- JX2 slave modules
- Servo amplifiers JetMove 1xx, JetMove 2xx, and JetMove 6xx
- IP67 modules LioN-S, and LJX7-CSL
- Third-party CANopen® modules, e.g. valve terminals

For more information about the peripheral devices, refer to the manuals of the respective devices and the application-oriented manuals.

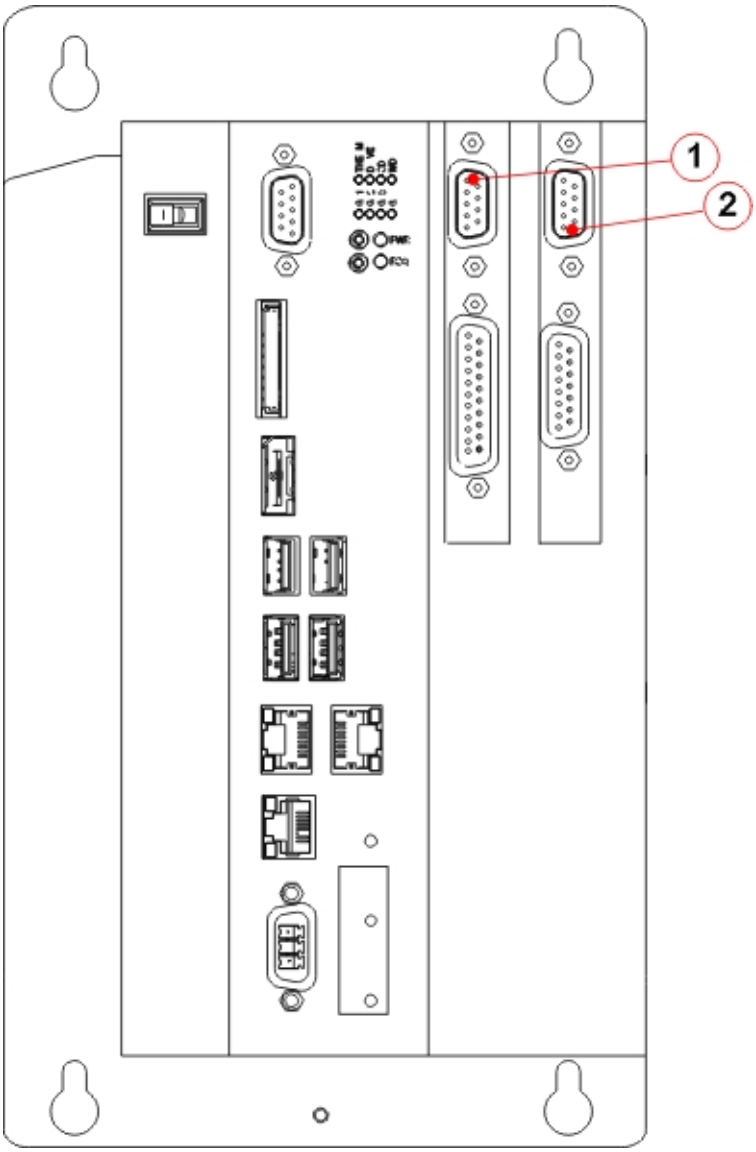
**Expandability**

Up to 31 JX2-I/O modules, 10 Smart-I/O modules, third-party CANopen® modules, as well as 16 JX2 slave modules or JetMoves can be connected to each of the JX2 system busses.

**Content**

## JX2 system bus interfaces X91/X92 for variant -E01

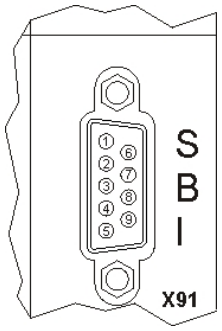
**Location of the JX2 system bus interfaces X91/X92**




| Number | Element | Description |
|---|---|---|
| **1** | X91 | JX2 system bus interface SBI |
| **2** | X92 | JX2 system bus interface SB |

# JX2 system bus - Pin assignment

**Sub-D connector X91 - Pin assignment**




| Pin | Signal | Description |
|---|---|---|
| **1** | CMODE0 | Commissioning |
| **2** | CL | Data signal |
| **3** | GND | Reference potential |
| **4** | CMODE1 | Commissioning |
| **5** | Unused | |
| **6** | Unused | |
| **7** | CH | Data signal |
| **8** | Unused | |
| **9** | Unused | |

**Technical specifications**

| Parameter | Description |
|---|---|
| Type of connector | Female Sub-D connector |
| Number of pins | 9 |
| Electrical isolation | None |
| Baud rates | 1,000/500/250/125 kBaud |

**Suitable cables**

For connecting modules to the JX2 system bus you can order the following cables separately:

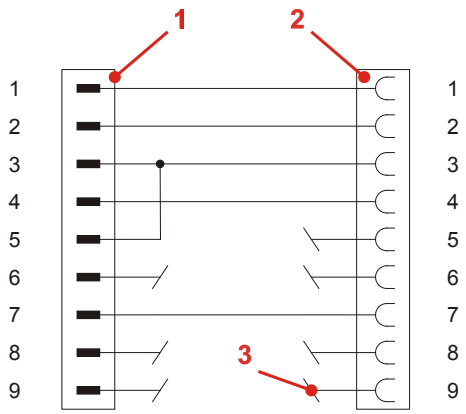
| Item no. | Item |
|---|---|
| 10309001 | Cable assy # 530 0.2 m |
| 10309002 | Cable assy # 530 0.5 m |
| 10309003 | Cable assy # 530 1.0 m |
| 10309004 | Cable assy # 530 1.5 m |
| 10309006 | Cable assy # 530 2.0 m |
| 10309016 | Cable assy # 530 2.5 m |
| 10309015 | Cable assy # 530 3.0 m |
| 10309007 | Cable assy # 530 4.0 m |
| 10309008 | Cable assy # 530 5.0 m |

# JX2 system bus cable - Specification

### JX2 system bus cable - Specification

| Parameter | Description |
|---|---|
| Wire cross-section | 1,000 kBaud: 0.25 ... 0.34 $mm^2$<br>500 kBaud: 0.34 ... 0.50 $mm^2$<br>250 kBaud: 0.34 ... 0.60 $mm^2$<br>125 kBaud: 0.50 ... 0.60 $mm^2$ |
| Cable capacitance | 60 pF/m max. |
| Resistivity | 1,000 kBaud: max. 70 Ω/km<br>500 kBaud: max. 60 Ω/km<br>250 kBaud: max. 60 Ω/km<br>125 kBaud: max. 60 Ω/km |
| Number of cores | 5 |
| Shielding | Complete shielding, no paired shielding |
| Twisting | CAN_L and CAN_H cables are twisted pairwise |

### Connection diagram




| Number | Part | Description |
|---|---|---|
| **1** | Male sub-D connector, 9-pin | For connection to BUS OUT |
| **2** | Female Sub-D connector, 9-pin | For connection to BUS-IN |
| **3** | Not connected | Do not connect these pins |

**Sub-D connector (male)**

Pinout of the 9-pin male Sub-D connector at the JX2 system bus cable:

| Pin | Signal name | Description |
|---|---|---|
| **1** | CMODE0 | Commissioning |
| **2** | CAN-L | Data signal |
| **3** | GND | Reference potential |
| **4** | CMODE1 | Commissioning |
| **5** | TERM | Short-circuited with pin 3 |
| **7** | CAN-H | Data signal |

**Female Sub-D connector**

Pinout of the 9-pin female Sub-D connector to the JX2 system bus cable:

| Pin | Signal name | Description |
|---|---|---|
| **1** | CMODE0 | Commissioning |
| **2** | CAN-L | Data signal |
| **3** | GND | Reference potential |
| **4** | CMODE1 | Commissioning |
| **7** | CAN-H | Data signal |

# JX2 system bus - Line lengths and baud rates

**Line lengths**

The maximum cable length depends on the baud rate used and the number of expansion modules connected to the bus.

| Baud rate | Cable length | Stub length | Total stub length |
|---|---|---|---|
| 1,000 kBaud | 25 m max. | 0.3 m max. | 3 m |
| 500 kBaud | 100 m max. | 1.0 m max. | 39 m |
| 250 kBaud | 200 m max. | 3.0 m max. | 78 m |
| 125 kBaud | 200 m max. | - | - |

**Rules for calculating the stub length**

When engineering the line length, follow the rules listed below:

- Each non-intelligent JX2-I/O module connected to the system bus reduces the maximum line length by 1.0 m
- Each connected intelligent JX2-I/O slave module reduces the maximum line length by 1.0 m
- Each JetMove reduces the maximum line length by 1.0 m
- Each connected IP67-I/O module reduces the maximum line length by 1.0 m

**Baud rates**

The baud rate setting depends on the number of modules connected to the JX2 system bus:

| JX2-I/O modules<br>JX2 slave modules<br>JetMove | JX-SIO<br>CANopen® modules | 1,000 kBaud | 500 kBaud | 250 kBaud | 125 kBaud |
|---|---|---|---|---|---|
| x | | x | x | x | x |
| | x | x | x | x | x |
| x | x | x | | | x |

# 3.4 Local JX6-I/O submodules on the PCIe bus (variant -E01)

**Introduction**

This chapter describes JX6-I/O submodules on the controller's internal PCIe bus. As they are located in the controller, they are called local submodules.

**Possible number of local JX6-I/O submodules**

Up to 2 riser cards (JI-PCIE-Exx) can be plugged into the controller's PCIe bus. Each riser card can accommodate up to 2 local JX6-I/O submodules. Thus, the controller can be equipped with a maximum of 4 JX6 submodules.

The riser card takes on converting the controller-internal PCIe bus to the local JX6-I/O submodules.

**Supported Modules**

JI-PCIE-Exx riser cards support the following JX6-I/O submodules:

- JX6-SV1: Counter module for connecting an incremental or absolute (SSI) rotary encoder.
- JX6-IO16CB: Digital inputs/outputs 24 V
- JX6-SB/JX6-SB-I: System bus module (see chapter *JX2 system bus interface on the PCIe bus*)

**Allowed combinations**

Each JI-PCIE-Exx riser card can accommodate up to 2 JX6-I/O submodules. However, only certain combinations are allowed due to mechanical restrictions.

**Content**

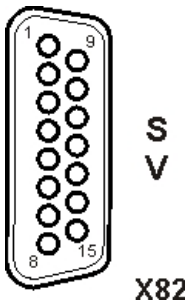
# Pinout of the JX6-SV1

**JX6-SV1 - Functions**

The female Sub-D connector of the I/O submodule JX6-SV1 interfaces with the following I/O functions:

- 1 incremental encoder input 5 V differential or
- 1 incremental encoder input 24 V or
- 1 absolute encoder input SSI

**Sub-D connector - Pinout**




| Pin | Signal Incremental encoder 5 V differential incremental encoder | Signal Incremental encoder 24 V | Signal Absolute encoder SSI |
|---|---|---|---|
| **1** | GND | GND | GND |
| **2** | K0+ | K0 | Unassigned |
| **3** | K0- | Unassigned | Unassigned |
| **4** | K1+ | K1 | Data+ |
| **5** | K1- | Unassigned | Data- |
| **6** | K2+ | K2 | Unassigned |
| **7** | K2- | Unassigned | Unassigned |
| **8** | Unassigned | Unassigned | Clock- |
| **9** | Unassigned | Unassigned | Clock+ |
| **10** | DC 5 V (50 mA) | Unassigned | Unassigned |

**Technical specifications**

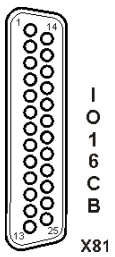
| Parameter | Description |
|---|---|
| Type of connector | Female Sub-D connector |
| Number of pins | 15 |
| Electrical isolation | Only at relay contact |

# Pinout of the JX6-IO16CB

**JX6-IO16CB - Interfaces**

The Sub-D connector of the I/O module JX6-IO16CB is the interface to digital inputs and outputs. For information on programming the module refer to chapter 6.3 Programming local JX6 submodules, page 124.

**Sub-D connector - Pinout**




| Pin | Signal | Description |
|---|---|---|
| **1** | Output 8 | |
| **2** | Output 7 | |
| **3** | DC 24 V | Power supply of the outputs |
| **4** | Output 6 | |
| **5** | Output 5 | |
| **6** | GND_A | Reference potential of the outputs |
| **7** | DC 24 V | Power supply of the outputs |
| **8** | Output 4 | |
| **9** | Output 3 | |
| **10** | GND_A | Reference potential of the outputs |
| **11** | DC 24 V | Power supply of the outputs |
| **12** | Output 2 | |
| **13** | Output 1 | |
| **14** | Input 8 | |
| **15** | GND_E | Reference potential of the inputs |
| **16** | Input 7 | |
| **17** | Input 6 | |
| **18** | Input 5 | |
| **19** | DC 24 V | Power supply of the outputs |
| **20** | Input 4 | |
| **21** | Input 3 | |
| **22** | Input 2 | |
| **23** | Input # 1 | |
| **24** | GND_A | Reference potential of the outputs |
| **25** | Not determined | |

**Technical specifications**

| Parameter | Description |
|---|---|
| Type of connector | Female Sub-D connector |
| Number of pins | 25 |
| Electrical isolation | Yes |
| Rated voltage | DC 24 V |
| Number of inputs | 8 |
| Number of outputs | 8 |
| Peak current per output | 0.5 A |

# 3.5 LEDs indicating various states

**LEDs of the JC-970MC**

The JC-970MC features the following LEDs:

- 4 LEDs for indicating operating system conditions and errors
- 4 LEDs for indicating drive activities, overtemperature, watchdog timeout of the controller
- 2 LEDs per port for indicating conditions of the Ethernet ports

**Content**

# LEDs GP

**LEDs GP**

LEDs **GP1 … GP4** (**G**eneral **P**urpose LEDs) are for indicating operating system conditions and errors.


| LED | Color | Description |
|---|---|---|
| **GP1** | Green | OS is running (RUN) |
| **GP2** | Amber | Special conditions (D1) |
| **GP3** | Red | Error (ERR) |
| **GP4** | | Not assigned |

**Normal operating condition**

In normal operating condition and with the application program running, the LEDs **GP1 ... GP4** indicate the following:

| GP3 | GP2 | GP1 | State |
|---|---|---|---|
| ○ OFF | ○ OFF | ● ON | Normal operating condition<br>■ Application program is running<br>■ No error |

**Status indications of LEDs GP1 ... GP4**

The table below lists the possible states of LEDs **GP1 ... GP4**:

| LED | State | Description |
|---|---|---|
| **GP1 (RUN)** | OFF | No power supply or failure |
| | 1Hz | Application program is not being executed |
| | ON | Application program is being executed |
| **GP2 (D1)** | OFF | Normal operating condition |
| | ON | Special conditions |
| **GP3 (ERR)** | OFF | No error |
| | ON | Error; refer to error register |

# GP1 ... GP4 LED indications during boot process

**Normal boot process**

If the following requirements are met, the controller goes through its normal boot process:

- Mode selector S11 is in *RUN* position.
- There is a valid OS.
- There must be a valid application program.

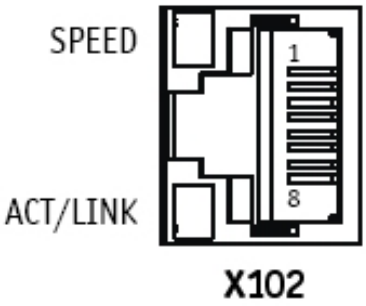
Then, LEDs **GP1 … GP3** indicate the following:

| Step | Description | | | |
|---|---|---|---|---|
| 1 | **GP3** | **GP2** | **GP1** | **State** |
| | ON | OFF | OFF | Reset |
| 2 | **GP3** | **GP2** | **GP1** | **State** |
| | OFF | ON | OFF | Initializing the runtime environment of the application program and rea-ltime communication |
| 3 | **GP3** | **GP2** | **GP1** | **State** |
| | OFF | ON | OFF | Motion control start |
| 4 | **GP3** | **GP2** | **GP1** | **State** |
| | OFF | ON | ON | Initializing additional functions (Web, Modbus/TCP etc.) |
| 5 | **GP3** | **GP2** | **GP1** | **State** |
| | OFF | OFF | ON | Normal condition; the application program is being executed |

## Status LEDs - Ethernet interface

**Status LEDs - Ethernet interface**

The status LEDs of the Ethernet port are located in the immediate vicinity of the RJ45 jack.




| LED | LED color | Description |
|---|---|---|
| **SPEED** | Off<br>Green<br>Yellow | Network connection speed<br>- 10 MBit/s<br>- 100 MBit/s<br>- 1,000 MBit/s |
| **ACT/LINK** | Yellow | ACTIVE - Flashing<br>LINK - Lit |

## Status LEDs

**Status LEDs**

The controller JC-970MC indicates conditions and drive activities via its status LEDs. The status LEDs are located in the upper part of the JC-970MC front panel.


**Indications of status LEDs**

The status LEDs indicate the following:

| LED | Function | Color | Description |
|---|---|---|---|
| THERM | Overtempera-ture warning | Red, flashing | System is shutting down due to overtemperature |
| DRIVE | Hard disk activ-ity | Green | HDD activity |
| SD | SD memory card | Green | SD card reader activity |
| WD | Watchdog | Red, flashing | Watchdog timeout |

# 3.6 Control elements

**Control elements of a JC-970MC**

The JC-970MC features the following control elements:

- PWR button for switching the controller ON or OFF
- RSQ button for booting the controller from backup flash
- Mode selector S11 with the positions RUN, STOP, and LOAD

**Content**

# PWR button

**PWR button**


**Function of the pushbutton**

The PWR button works as an ON/OFF switch.

As soon as voltage is applied to terminal X101, the controller boots and the status LED **PWR** is lit green. Depending on the position of the mode selector, the STX application program is loaded or not, or the AutoCopy function is launched.

If you press the PWR shortly, nothing happens, as the Hypervisor prevents the system from shutting down.

If you press the PWR button for more than 4 seconds, a forced shutdown is triggered. As this does not trigger a powerfail interrupt, the application registers will **not** be backed up. That's the reason why we recommend to interrupt the power supply instead. This will trigger the powerfail interrupt and the non-volatile register range (register no 1,000,000 … 1,199,999) will be backed up.

**NOTICE**

Loss of data!

A forced shutdown by means of PWR button may result in loss of data or other undesirable effects! All changes made to non-volatiles registers and flags since the controller was energized last will be lost.

- ➢ Back up your data before you shut down the system.
- ➢ We recommend the following workflow:
  - First, shut down Windows.
  - Then, de-energize the device.

**NOTICE**

Damage to the device!

Even when the controller JC-970MC is turned off by pressing the PWR button, the system is still energized. To de-energize the system, remove the power supply from terminal X101.

## RSQ button

**RSQ button**


**Function of the pushbutton**

Pressing the RSQ button causes the system to boot from the backup flash.

The LED RSQ is flashing red if the backup flash has been selected for booting. The backup flash contains a cloned BIOS (UEFI) version. In the event the system does not properly start-up or gets hung-up and restarting (cold booting) does not help, the RSQ function lets you boot from the backup flash and restart the system. To this end, press the RSQ button for more than 5 seconds. Whether the system is running or not, it will start-up using the backup flash for booting.

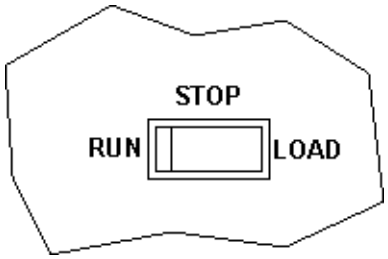
To revert to the default boot flash, the system must be cold started. To this end, press the PWR button or remove the power supply voltage and then reapply it.

**Notice!**

The rescue function is not intended for use within an application. It is intended to force the controller in the case of a corrupted BIOS flash to boot into a defined and safe state to allow for troubleshooting.

# Mode selector S11

**Mode selector**




| Position | Description |
|---|---|
| **RUN** | Once the controller has booted up, it launches the STX application program. |
| **STOP** | The controller boots up, but does not launch the STX application program. |
| **LOAD** | The controller boots up, but does not launch the STX application program. Instead, it switches to AutoCopy mode. |

**Functions of the mode selector**

JC-970MC checks the position of mode selector S11 and proceeds as follows:

| Step | Description | |
|---|---|---|
| **1** | Power supply of the controller is at terminal X101. | |
| **2** | The boot loader of the controller checks the position of selector S11. | |
| | **If ...** | **... then ...** |
| | ... mode selector S11 = *RUN* or *STOP*, ... | ... the OS is launched;<br>--> proceed with step 3. |
| | ... mode selector S11 = *LOAD*, … | ... the controller starts the AutoCopy function. |
| **3** | The controller checks the position of selector S11. | |
| | **If ...** | **... then ...** |
| | ... mode selector S11 = *RUN*, … | ... the application program is launched. |
| | ... mode selector S11 = *STOP*, … | ... the application program does not start. |
| **4** | **If ...** | **... then ...** |
| | ... the position of mode selector S11 is changed once the controller has been turned on, ... | ... this has no effect on the functioning of the controller. |

# 3.7 Ethernet interface X102 - IP configuration

**Introduction**

This chapter describes how to configure Ethernet port X102 which is attributed to the controller itself (RTOS). Ethernet port X104 which is attributed to Windows must be configured in the Windows settings.

You can configure the following parameters for X102:

- IP address of the controller
- Subnet mask
- IP address of default gateway
- Static route of the IP address
- Static route of the subnet mask
- Static route of the gateway
- IP address of DNS server
- Controller name
- IP port number for the JetSym debugger
- Basic port number for communication via JetIP

**Required skills of the network configurator**

IP networks:

- IP addressing (IP address, port number, subnet masks etc.)
- FTP (connection setup, data transmission, etc.)

**Jetter Ethernet system bus**

The Jetter Ethernet system bus (JetIPScan, JetSync, Publish/Subscribe, etc.) and thus synchronous transfer of data packets is supported by port ETH (X102) only.

**Content**

# Factory settings

**Introduction**

Before the JC-970MC is shipped, various parameters are set to a certain value. These parameters can be changed by the user.

**Factory settings**

| Parameter | Value |
|---|---|
| ETH (X102): IP address | 192.168.1.1 |
| ETH (X102): Subnet mask | 255.255.255.0 |
| ETH (X102): IP address of default gateway | 0.0.0.0 |
| IP address of DNS server | 0.0.0.0 |
| Controller name | JetControl970MC |
| IP port number for debugger | 52000 |
| IP port number for JetIP | 50000 |
| User's password *admin* | admin |
| User's password *system* | system |

# Determining the IP address of the JC-970MC

**Introduction**

The program JetIPScan lets you retrieve IP address, subnet mask and the IP address of the Default Gateway of the Ethernet port ETH (X102).

**Downloading JetIPScan**

You can download JetIPScan from the Jetter AG **homepage** at **http://www.jetter.de**. The location of **jetipscan_1-08-01.zip** is Downloads/Software/Classics/Additional software/jetipscan.

**Contents of the jetipscan_1-08-01.zip**

The zip file **jetipscan_1-08-01.zip** contains the following files:

- The program **JetIPScanV1.08_01.exe**
- The help **jetipscan_01_help_en.png**
- A batch file **read_IP_via_JETIPSCAN.bat** to determine the IP address
- A batch file **write_IP_via_JETIPSCAN_10_150.bat** to set IP address 192.168.10.150 for the controller

The batch files launch the program JetIPScan.

The files are unzipped to the folder **jetipscan_1-08-01**.

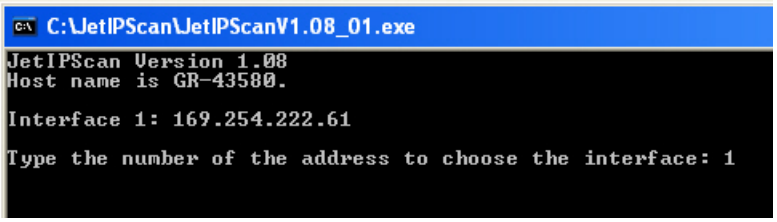
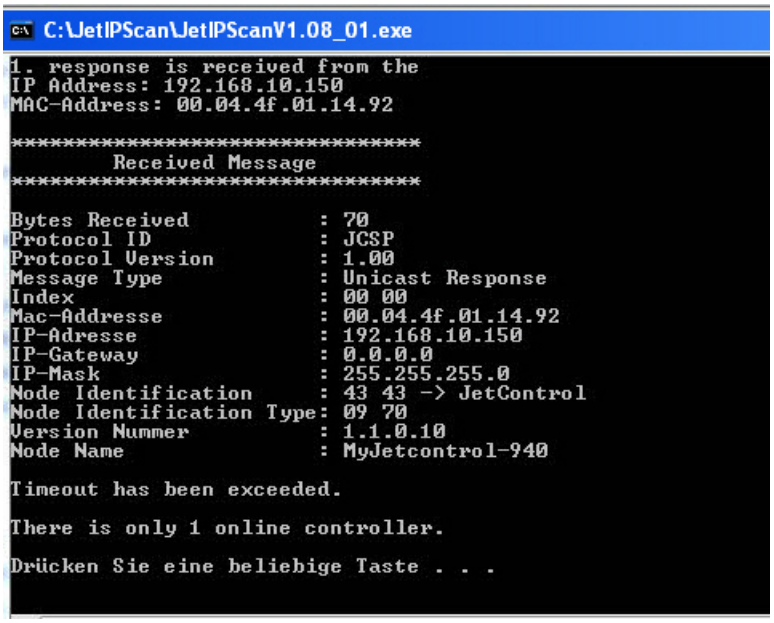
**Prerequisites**

First, an Ethernet connection between the PC and the controller JC-970MC must be established.

Then launch the JetIPScan program on your PC, for example via batch file **read_IP_via_JETIPSCAN.bat**.

**Determining the IP address**

To determine the IP address of the JC-970MC, proceed as follows:

| Step | Action |
|---|---|
| **1** | Launch the JetIPScan program on your PC. |
| ⇨ | **Result:** JetIPScan shows all IP addresses, which are presently active on your PC. |
| **2** | Select the interface (IP address) that is connected to the controller JC-970MC for which you want to determine the IP address. |
| |  |
| ⇨ | **Result:** Among others, JetIPScan shows the IP address of the controller (here MyJetControl-940). |
| |  |

# The configuration memory

**Introduction**

During the boot process, the controller reads the parameters for initializing the IP interface from the configuration memory. Registers 101200 ... 101299 let you access the data stored in the configuration memory.

**When does this take effect?**

The controller reads out data located in the configuration memory only during the boot process. If you make changes to the configuration memory, reboot the controller. Only this way, these changes will take effect.

**Default values**

Before the controller further processes data from the configuration memory, it checks them for plausibility. **If entries are invalid or absent, the controller uses the following default values:**

| Parameter | Default value |
|---|---|
| ETH (X102): IP address | 192.168.10.150 |
| ETH (X102): Subnet mask | 255.255.255.0 |
| ETH (X102): IP address of default gateway | 0.0.0.0 |
| IP address of DNS server | 0.0.0.0 |
| ETH (X102): Static route of the IP address | 0.0.0.0 |
| ETH (X102): Static route of the subnet mask | 0.0.0.0 |
| ETH (X102): Static route of the gateway | 0.0.0.0 |
| Controller name | JetControl970MC |
| Suffix type of the name | 0 |
| IP port number for debugger | 52000 |
| IP port number for JetIP | 50000 |

**Related topics**

- **Configuration file** *config.ini* (see page 68)
- **Configuration registers** (see page 72)

## Configuration file "config.ini"

**Introduction**

If the data for initializing the Ethernet interface ETH (X102) in the configuration memory are invalid, the controller JC-970MC uses the settings from the configuration file **config.ini**. The JC-970MC also uses **config.ini** for the AutoCopy function.

**Properties**

- You can access the config.ini through the file system of the controller.
- For an FTP connection, the user must have administrator or system rights.
- This file is located in the folder **System**.
- You cannot delete the file, but only overwrite it.
- When you format the flash disk, this file remains untouched.

**File structure**

The configuration file is a text file the entries of which are grouped into several sections. The controller replaces missing IP configuration parameters by their default values.

**Example: *config.ini***

This is an example of a configuration file **config.ini**:

```
;JC-970MC System Configuration
;Copyright (c) 2009 by Jetter AG, Ludwigsburg, Germany

[IP]
Address    = 192.168.50.1
SubnetMask = 255.255.255.0
DefGateway = 192.168.50.11
DNSServer  = 192.168.1.44

[HOSTNAME]
SuffixType = 0
Name       = JetControl970MC

[PORTS]
JetIPBase = 50000
JVMDebug  = 52000

[FILES]
AutoCopyIni = /USB1/autocopy.ini
```

**Section [IP]**

In section [IP] the required IP addresses and the subnet mask are specified.

| **Address** | |
|---|---|
| In the given example | 192.168.50.1 |
| Description | IP address of the controller |
| Valid values | ■ > 1.0.0.0<br>■ < 223.255.255.255 |
| Illegal values | ■ Network address<br>■ Broadcast address |
| In the event of an illegal value | The controller resets all four values to their fallback values. |
| **SubnetMask** | |
| In the given example | 255.255.255.0 |
| Description | Specifies the subnet mask |
| Valid values | ■ >= 128.0.0.0 |
| Illegal values | ■ 1 and 0 mixed |
| In the event of an illegal value | The controller resets all four values to their fallback values. |
| **DefGateWay** | |
| In the given example | 192.168.50.11 |
| Description | IP address of the gateway to other subnets;<br>The controller must be able to reach the subnet (Address/SubnetMask), otherwise it will set this parameter to 0.0.0.0. |
| Valid values | ■ >= 0.0.0.0 and<br>■ < 223.255.255.255 |
| Illegal values | ■ Network address<br>■ Broadcast address<br>■ A value (Address/SubnetMask) which cannot be reached by the controller.<br>■ *Address* value |
| In the event of an illegal value | The controller sets the value to 0.0.0.0 |
| **DNSServer** | |
| In the given example | 192.168.1.44 |
| Description | IP address of the server for the Domain Name System |
| Valid values | ■ >= 0.0.0.0 and<br>■ < 223.255.255.255 |
| In the event of an illegal value | The controller sets the value to 0.0.0.0 |

**Section [HOSTNAME]**

In section [HOSTNAME] the name of the controller is specified. If desired, the controller automatically generates an individual name. The JC-970MC presently does not use the host name.

| **SuffixType** | |
|---|---|
| In the given example | 0 |
| Description | The type of the automatically generated suffix that is attached to the controller name |
| Valid values | ■ 0: No attachment<br>■ 1: Low-order byte of the IP address in decimal notation<br>■ 2: Low-order byte of the IP address in hexadecimal notation |
| In the event of an illegal value | 0 |
| **Name** | |
| In the given example | JetControl970MC |
| Description | Specifies the controller name |
| Valid values | ■ First character: 'A' ... 'Z', 'a' ... 'z'<br>■ Next characters: 'A' ... 'Z', 'a' ... 'z', '0' ... '9', '-' |
| In the event of an illegal value | JetControl970MC |

**Section [PORTS]**

In section [PORTS] the IP port numbers of data and debug servers within the controller are specified. The IP port numbers must be consistent with, for example, the port numbers set in JetSym.

| **JetIPBase** | |
|---|---|
| In the given example | 50000 |
| Description | IP port number for OS updates and communication between controllers |
| Valid values | ■ 1024 ... 65535 |
| In the event of an illegal value | 50000 |
| **JVMDebug** | |
| In the given example | 52000 |
| Description | IP port number for Debugger/Setup in JetSym |
| Valid values | ■ 1024 ... 65535 |
| In the event of an illegal value | 52000 |

### Changing the IP configuration

| Step | Action |
|---|---|
| **1** | Create on your PC a configuration file named **config.ini** using a text editor and make the corresponding entries. |
| **2** | Open an FTP connection between the PC and the JC-970MC. |
| **3** | Log in as user with administrator or system rights.<br>Default login information:<br>User: *admin*; Password: *admin* |
| **4** | Browse to directory */System* JC-970MC. |
| **5** | Copy the configuration file **config.ini**, which has been created by you, to the controller. |
| **6** | Clear the FTP connection. |
| **7** | Reboot the controller.<br>**Result**:<br>If the data in the configuration memory is invalid, the configuration data of **config.ini** is used. |

### Related topics

- **Configuration memory** (see page 67)
- **Configuration registers** (see page 72)

# Configuration registers

**Introduction**

Configuration registers let you read out IP configuration parameters or make changes to them. A register area contains the data from the file **\System\config .ini.** Another range contains the parameters used for initializing the IP interface.

**Register numbers**

The basic register numbers of both ranges are dependent on the device. The register number is calculated by adding the number of the module register (MR) to the number of the basic register.

| Controller | Data range | Basic register number | Register numbers |
|---|---|---|---|
| JC-970MC | config.ini | 101100 | 101100 ... 101165 |
| | Parameters used | 101200 | 101200 ... 101265 |

**Configuration registers**

The following table lists the registers of both ranges, as well as their connection to the entries in the configuration file **\System\config.ini**:

| Register | Section in config.ini | Name in config.ini | Description |
|---|---|---|---|
| **MR 0** | IP | `Address` | IP address of the controller |
| **MR 1** | | `SubnetMask` | Specifies the subnet mask |
| **MR 2** | | `DefGateWay` | IP address of gateway to other subnets |
| **MR 3** | | `DNSServer` | IP address of the server for the Domain Name System |
| **MR 32** | HOSTNAME | `SuffixType` | The type of the automatically generated suffix that is attached to the controller name |
| **MR 33 ... MR 51** | | `Name` | Specifies the controller name |
| **MR 64** | PORTS | `JetIPBase` | IP port number for OS updates and communication between controllers |
| **MR 65** | | `JVMDebug` | IP port number for Debugger/Setup in JetSym |

**Notice!**

If you make manual changes to data in the configuration registers, also change the entries in **config.ini** to ensure that the configuration data matches

**Related topics**

- **Configuration memory** (see page 67)
- **Configuration file** *config.ini* (see page 68)

## Changing the IP address of the controller

**Introduction**

To be able to communicate with the controller (RTOS side) JC-970MC via Ethernet (X102), you must assign an unambiguous IP address to the controller.

**Configuration options**

There are several options to configure the IP address of the port ETH (X102):

- In the **config.ini** file
- During runtime via special registers (volatile)
- Via special registers (non-volatile)
- ia the program JetIPScan (non-volatile)

**Related topics**

- **Setting the IP address by means of file** *config.ini* (see page 68)
- **Setting the IP address during runtime** (see page 80)

# Making changes to the IP address of the JC-970MC using JetIPScan

**Introduction**

The program JetIPScan lets you make changes to IP address, subnet mask and the IP address of the Default Gateway of the JC-970MC. Yet, this only applies to the Ethernet port ETH (X102).

**Downloading JetIPScan**

You can download JetIPScan from the Jetter AG **homepage** at **http://www.jetter.de**. The location of **jetipscan_1-08-01.zip** is Downloads/Software/Classics/Additional software/jetipscan.

**Contents of the jetipscan_1-08-01.zip**

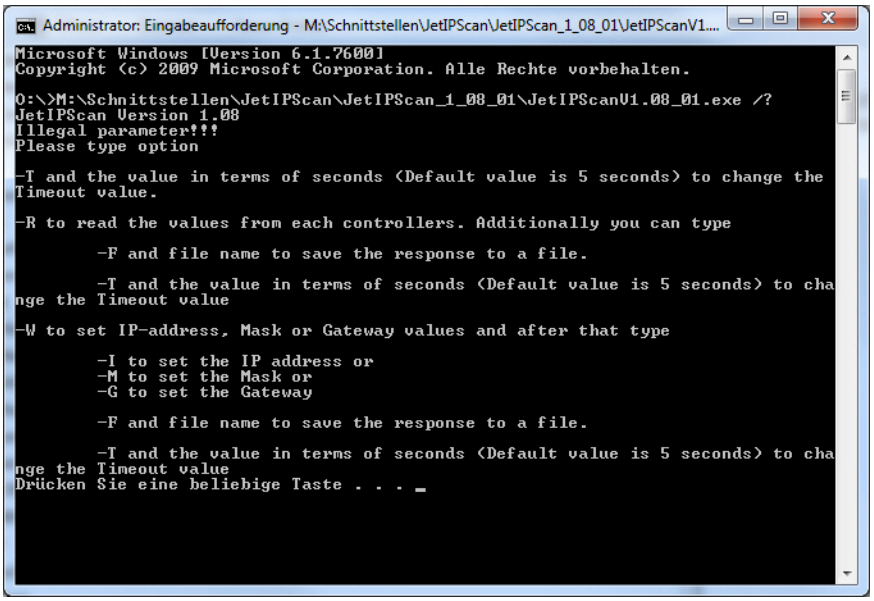
The zip file **jetipscan_1-08-01.zip** contains the following files:

- The program **JetIPScanV1.08_01.exe**
- The help **jetipscan_01_help_en.png**
- A batch file **read_IP_via_JETIPSCAN.bat** to determine the IP address
- A batch file **write_IP_via_JETIPSCAN_10_150.bat** to set IP address 192.168.10.150 for the controller

The batch files launch the program JetIPScan.

The files are unzipped to the folder **jetipscan_1-08-01**.

**Possible commands of the software JetIPScan**

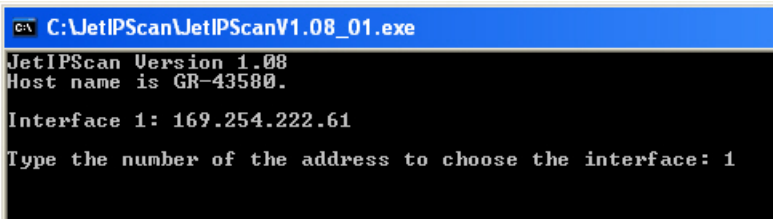



**Prerequisite**

First, an Ethernet connection between the PC and the controller JC-970MC must be established.

### Making changes to the IP address

To make changes to the IP address of the JC-970MC, proceed as follows:

| Step | Action |
|---|---|
| 1 | Write a batch file. The content of the batch file is **JetIPScanV1.08_01.exe -W -I 192.168.10.150**. |
| 2 | Execute the batch file. |
| ⇨ | **Result:** JetIPScan is launched and shows all IP addresses, which are presently active on your PC. |
| 3 | Select the interface (IP address) connected with the controller JC-970MC, of which you want to retrieve the IP address. |
| |  |
| 4 | JetIPScan shows all the devices found.<br>To make changes to the IP address of a device, select the respective device from the list. |
| ⇨ | **Result:** JetIPScan changes the IP address of the JC-970MC to the value 192.168.10.150. |

### Changing the subnet mask

To make changes to the subnet mask of the JC-970MC, proceed as follows:

| Step | Action |
|---|---|
| 1 | Write a batch file. The content of the batch file is, for example **JetIP-ScanV1.08_01.exe -W -M 250.255.248.0**. |
| 2 | Execute the batch file. |
| ⇨ | **Result:** JetIPScan is launched and shows all interfaces, which are presently active on your PC. |
| 3 | For all further steps, please refer to the instruction **Changing the IP address**. |

### Changing the Default Gateway

To make changes to the Default Gateway settings of the JC-970MC, proceed as follows:

| Step | Action |
|---|---|
| 1 | Write a batch file. The content of the batch file is, for example, **JetIP-ScanV1.08_01.exe -W -G 192.168.4.1**. |
| 2 | Execute the batch file. |
| ⇨ | **Result:** JetIPScan is launched and shows all interfaces, which are presently active on your PC. |
| 3 | For all further steps, please refer to the instruction **Changing the IP address**. |

## Setting the IP address via "config.ini" file

**Configuration file *config.ini***

You can set the IP address of the JC-970MC in the **config.ini** file.

```
[IP]
Address    = aaa.bbb.ccc.ddd
...
```

| Element | Description |
|---|---|
| `Address` | Enter the IP-address into this line. |
| `aaa` | First byte of IP address |
| `bbb` | Second byte of IP address |
| `ccc` | Third byte of IP address |
| `ddd` | Fourth byte of IP address |

**NOTICE**

If the data in the configuration memory are not o.k., the controller uses the IP address settings from the configuration file **config.ini**. Yet, this only applies to the Ethernet port ETH (X102).

**Transferring the configuration file**

| Step | Action |
|---|---|
| **1** | Establish an FTP connection to the JC-970MC. |
| **2** | Log in as user with administrator or system rights.<br>Default login information:<br>User: *admin*; Password: *admin*<br>User: *system*; Password: *system* |
| **3** | Open the folder **System**. |
| **4** | Copy the file **config.ini** to the folder **System**. |
| **5** | Clear the FTP connection. |
| **6** | Reboot the JC-970MC. |

# Setting the IP address via non-volatile registers

**Introduction**

The IP interface ETH (X102) is initialized by the settings in the configuration memory during the boot process.

The following non-volatile settings can also be changed via registers:

- IP address of the controller
- Subnet mask
- IP address of default gateway
- Static route of the IP address
- Static route of the subnet mask
- Static route of the gateway
- IP address of DNS server
- Host name and suffix type
- Port numbers for JetIP and JetSym debugger

**Registers - Overview**

| Register | Description |
|---|---|
| **101200** | ETH (X102): IP address |
| **101201** | ETH (X102): Subnet mask |
| **101202** | ETH (X102): IP address of default gateway |
| **101203** | IP address of DNS server |
| **101216** | ETH (X102): Static route of the IP address |
| **101217** | ETH (X102): Static route of the subnet mask |
| **101218** | ETH (X102): Static route of the gateway |
| **101232** | Host name suffix type |
| **101233 through 101251** | Host name |

| Register | Description |
|---|---|
| **101264** | Port number for JetIP |
| **101265** | Port number for STX debugger |
| **101299** | Saving the settings (0x77566152) |

**Setting IP addresses and subnet mask**

For changing of the IP addresses, the subnet mask, and the Default Gateway of the Ethernet port ETH (X102) via non-volatile registers, proceed as follows:

| Step | Action |
|---|---|
| **1** | Enter the IP address of Ethernet port ETH (X102) into register 101200. |
| **2** | Enter the subnet mask into register 101201. |
| **3** | Enter the IP address of the default gateway into register 101202. |
| **4** | To have the controller apply the values, you must enter a password.<br>Enter value 2002149714 (0x77566152) into register 101299. |
| **5** | Boot the controller JC-970MC. |

**Result:**

The settings are completed. Communication is possible again.

**Related topics**

- **Configuration memory** (see page 67)
- **Setting the IP address during runtime** (see page 80)
- **Setting up a static route** (see page 83)

## Setting the IP address automatically via USB flash drive

**Introduction**

To have the IP configuration of the controller automatically set by a USB flash drive, you can apply the function *Copying controller data automatically (Auto-Copy)*. For this, use the registers described in the chapter *Setting the IP address by non-volatile registers*.

**Prerequisites**

You must be familiar with the AutoCopy function (see chapter 7).

**AutoCopy command file**

The example below shows a command file of the AutoCopy function:

```
[OPTIONS]
CommandCount = 1
LogFile = /USB1/autocopy.log
LogAppend = 0

# set registers for IP configuration
[COMMAND_1]
Command = DaFileRead
DaFile = /USB1/ip-setup.da
```

**Data file *ip-setup.da***

This example shows a data file, by which the IP address, the subnet mask and the port number for the JetIP protocol are set in the configuration memory:

```
SD101
; Data File - Jetter AG
;
; Register 101200 ip address
RS    101200    -1062729066
; Register 101201; subnet mask
RS    101201    -512
; Register 101264 JetIP port number
RS    101264    51000
; Register 101299 write to configuration memory
RS    101299    2002149714
```

Note:

The register contents (e.g. IP address 192.168.10.150, subnet mask, etc.) are specified here as signed integer values (32 bits).
In the JetSym Setup window, you can have the IP address displayed in various notations (signed integer, hexadecimal, binary, ...).

**Related topics**

- **Configuration memory** (see page 67)
- **Automatic copying of controller data** (see page 179)

## Setting the IP address during runtime

**Introduction**

The IP interface is initialized by the settings in the configuration memory during the boot process.

The following non-volatile settings can also be changed via registers:

- IP address of the controller
- Subnet mask
- IP address of default gateway

**Important note**

The settings made during runtime do not overwrite the parameters in the configuration memory. At de-energizing the controller, your settings will be lost.

**Prerequisites**

- No communication may take place via the IP interface during setting. Otherwise, this would lead to a loss of data.
- The values entered must be valid. This can be ensured, e.g. by including a validity check in the application program.
  This is important because there is no check if you set the parameters during runtime of the controller.

**Registers - Overview**

| Register | Description |
|---|---|
| **104531** | ETH (X102): IP address |
| **104532** | ETH (X102): Subnet mask |
| **104533** | ETH (X102): IP address of default gateway |

**Setting IP addresses and subnet mask**

To make changes to the IP addresses and the subnet mask of Ethernet port ETH (X102), proceed as follows:

| Step | Action |
|---|---|
| **1** | Enter the value 0.0.0.0 into R 104533. |
| **2** | Enter the value 0.0.0.0 into R 104532. |
| **3** | Enter IP address of Ethernet port ETH (X102) into register 104531. |
| **4** | Enter the desired subnet mask into R 104532. |
| **5** | Enter the desired IP address of the default gateway into R 104533. |

**Related topics**

- **Configuration memory** (see page 67)
- **Setting the IP address via non-volatile registers** (see page 77)
- **Setting up a static route** (see page 83)

# E-mail feature - Using names for IP addresses

**Introduction**

In configuring the e-mail client, you can use names as IP addresses for target systems. The controller resolves the names into IP addresses. A configuration file or the Domain Name System is used to resolve a name into its corresponding IP address.

**Name resolution**

Names are resolved to IP addresses in the following way:

| Step | Description | |
|---|---|---|
| **1** | During the boot process the controller reads the IP address of the DNS server out of the configuration memory. | |
| **2** | During the boot process the controller reads the file **/etc/hosts**. Then, the controller creates a translation table with the names and IP addresses found in this file. | |
| **3** | After the boot process the controller detects a name instead of an IP address. | |
| **4** | Based on this translation table, the controller tries to resolve the name into a related IP address. | |
| | **If ...** | **... then ...** |
| | ... the controller could resolve the name, ... | ... proceed with step 6. |
| | ... the controller could not resolve the name, ... | ... proceed with step 5. |
| **5** | The controller tries to resolve the name into a related IP address by sending a request to the DNS server. | |
| | **If ...** | **... then ...** |
| | ... the controller could resolve the name, ... | ... it enters the name and IP address into the translation table;<br>--> proceed with step 6. |
| | ... the controller could not resolve the name, ... | ... the controller aborts the function, e.g. the system function for sending an e-mail, and issues an error message. |
| **6** | The controller uses the IP address resolved for further communication. | |

**Configuration file *hosts***

This file holds the static assignment between name and IP address. The controller reads this file once during boot-up.

File format: Text

Location: /etc

File name: hosts

**Example**

```
# Example host file for JC-9xx
192.168.33.209    jetter_mail
192.168.33.208    jetter_demo
192.168.1.1       JC970MC
192.168.1.2       JC970MC
```

**Domain Name System (DNS)**

If a name cannot be found in the file **/etc/hosts**, the controller tries to resolve the IP address by obtaining the corresponding IP address from a DNS server. During boot-up, the controller reads the IP address of the DNS server out of the configuration memory.

**Related topics**

- **Configuration memory** (see page 67)

## Setting a static route

**Introduction**

The operating system of the JC-970MC offers the possibility to configure simple routing of IP packets for Ethernet port ETH (X102). If networks have to be accessed which are not in the directly accessible range of the JC-970MC, the user must set up static routes to these networks. Static routes especially have to be established, if the default gateway cannot reach the destination network.

**Saving a route to a configuration memory**

For Ethernet port ETH (X102), entries can be made in the configuration memory. Non-volatile registers 101216 through 101218 let you make permanent changes to these entries. All entries saved to the configuration memory are activated when the controller is rebooted.

**Saving a route at runtime**

Another register set (R 104550 through 104555) lets you - during runtime - make further entries into the routing table or delete existing routes. This register set can hold up to five entries.

**R 104550**

**Status**

This register lets you read out the status when adding or deleting routes during runtime.

| Register properties | | |
|---|---|---|
| Reading values | 0 | No error |
| | -1 | Routing table is full |
| | -2 | Entry not found |
| | -3 | Port is not active |
| | -4 | TCP/IP stack not initialized |

**R 104551**

**Command**

This command lets you add or delete a route.

| Register properties | | |
|---|---|---|
| Writing values | 1 | Add route |
| | 2 | Delete route |

**R 104552**

**Port number**

This register lets you define the Ethernet port.

| Register properties | | |
|---|---|---|
| Writing values | 1 | ETH 1 (X102) |

**R 104553**

**IP address of the static route**

Enter the IP address of the route into this register.

| Register properties | |
|---|---|
| Writing values | 0.0.0.0 ... 255.255.255.255 |

**R 104554**

**Subnet mask of the static route**

Enter the number of the subnet mask of the route into this register.

| Register properties | |
|---|---|
| Writing values | 0.0.0.0 ... 255.255.255.255 |

**R 104555**

**Gateway of the static route**

Enter the Gateway IP address of the route into this register.

| Register properties | |
|---|---|
| Writing values | 0.0.0.0 ... 255.255.255.255 |

**Setting a static route**

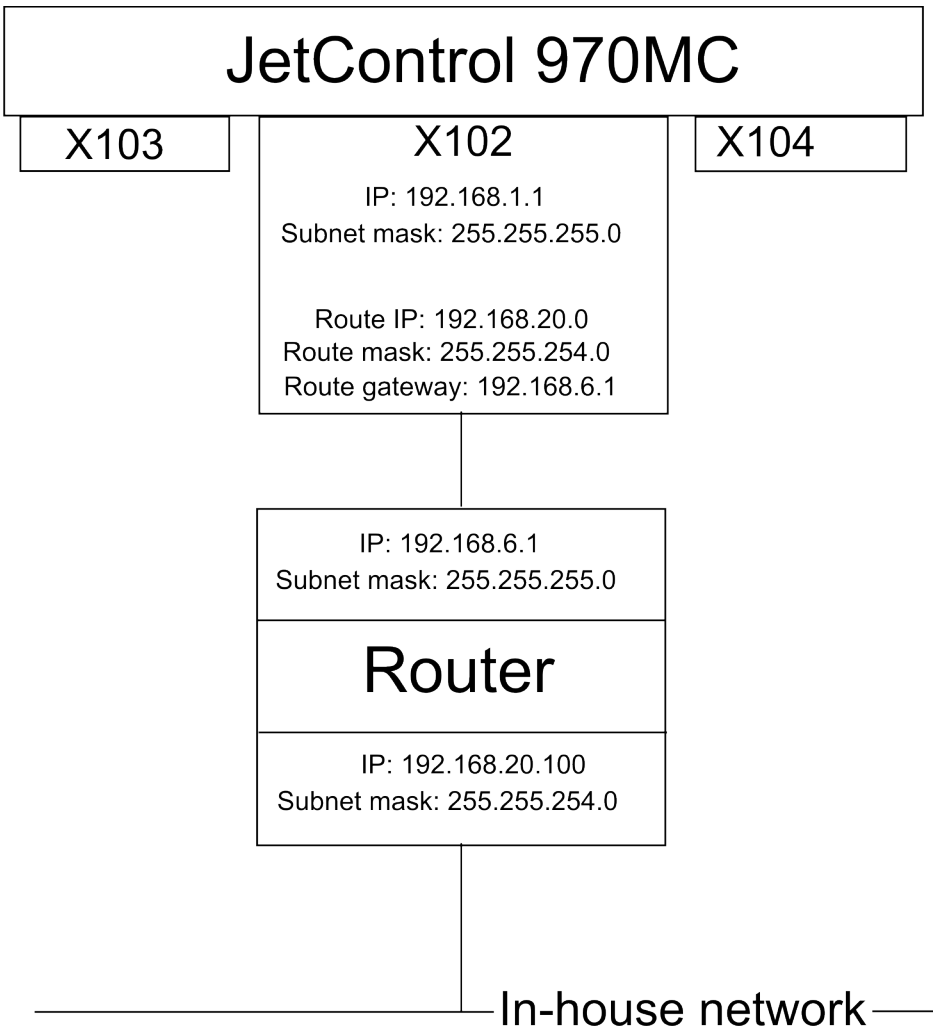
To either add the route to the routing table or delete it from there during runtime, proceed as follows:

| Step | Action |
|---|---|
| **1** | Enter the port number into register 104552. |
| **2** | Enter the IP address into register 104553. |
| **3** | Enter the subnet mask into register 104554. |
| **4** | Enter the IP address of the gateway into register 104555. |
| **5** | Enter the command into register 104551. |

**Result:** If register 104550 has got value 0, the command has been executed successfully. The route settings are active.

**Example**

The JC-940MC is connected with your in-house network, for example, via its Ethernet port X102 and a router.




The JC-940MC transmits frames to addresses within the subnets 192.168.20.xxx and 192.168.21.xxx.
The route set in the JC-940MC connects these with your in-house network via router.

**Related topics**

- **Setting the IP address using non-volatile registers** (see page 77)
- **Setting the IP address at runtime** (see page 80)

# 4 Initial commissioning

**Purpose of this chapter**

The first part of this chapter provides a compact description on how to commission the JC-970MC and covers the following functions:

- Creation and execution of a program which increments a variable.

The second part of this chapter covers the initial commissioning of the bus node JX3-BN-ETH involving the following steps:

- Configuring the hardware of a JX3 station and installing it
- Configuring the software in JetSym
- Creating and executing a program to read out a module register of a peripheral module connected to the JX3-BN-ETH.
- Addressing the bus node JX3-BN-ETH
- Describing the Jetter Ethernet system bus by the following means:
  - Hardware Manager
  - Publisher/Subscriber mechanisms

**Prerequisites**

Before the controller JC-970MC can be put into operation for the first time, the following requirements must be met:

- The controller must be connected to a PC directly via Ethernet or via switch or a hub.
- The programming software JetSym 5.1.3 or higher must be installed on the PC.
- Mode selector S11 must be in *STOP* position.
- The controller must be connected with the JX3-BN-ETH bus node.

**Content**

## Preparations for initial commissioning of the controller

**Energizing the controller**

Various diagnostic and self-test routines run when the controller is energized. The Windows part of the IPC performs its integrated Power-On Self Test (POST). In addition, the standard diagnostic options of Windows are also available, such as memory and CPU utilization. The controller section performs the following two tests:

- File system check of the internal USB flash drive
  (error register 200008, bit 0: Flash disk error)
- Check if the remanent registers were saved correctly at the last shutdown
  (error register 200008, bit 4: Error: Remanent registers)

To energize and boot the controller, proceed as follows:

| Step | Action |
|---|---|
| **1** | Set the mode selector S11 to STOP position.<br>This prevents the application program from being loaded. |
| **2** | Connect the 24 V DC supply voltage to terminal X101. |
| ⇨ | **Result:** The controller and the integrated industrial PC are booting up. |
| | The green LED GP1 (RUN) is flashing at a frequency of 1 Hz. |
| **3** | Configure the controller as described below. |

If you want to start the controller in normal operation, set switch S11 to **RUN**. With this setting, the application program is loaded and executed.

**Ethernet connection with the controller**

The default IP address (factory setting) of the JC-970MC is 192.168.1.1. This lets you, for example, access the controller from an external programming PC via Ethernet port X102. Configure the Ethernet interface of your PC so that it is able to communicate with the controller via this IP address.

**Internal IP address of the controller**

You can also access the controller internally from the Windows section of the JC-970MC. For this purpose, the IP address of the controller is 192.168.2.3.

**Behavior after power-up**

If the mode selector is in position *STOP* when the controller is powered-up, the application program will not be launched.

**LEDs**

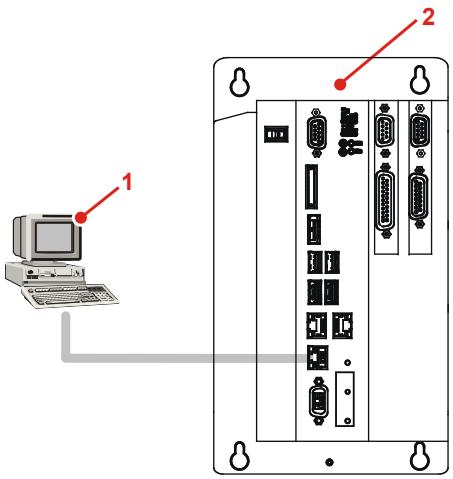
If commissioning has completed without errors, the LEDs GP are lit as follows:

| LED | Indication | Description |
|---|---|---|
| **GP1 (RUN)** |  | Logic voltage supply is OK; application program has been stopped. |
| **GP2 (D1)** |  | No error |
| **GP3 (ERR)** |  | No error |

# Initial commissioning of a JC-970MC

## Configuration

The initial commissioning is based on the following configuration:




| Number | Part | Description |
|---|---|---|
| 1 | PC | Programming system |
| 2 | JC-970MC | Controller |

## Creating a program

To create and check the program, proceed as follows:

| Step | Action |
|---|---|
| 1 | Launch the programming tool JetSym. |
| 2 | Create a new project. |
| 3 | Set the controller model (JC-970MC). |
| 4 | Set the IP address of the controller. |
| 5 | Open the program editor. |
| 6 | Enter the program specifications. |
| 7 | Upload the project to the controller. |
| 8 | Open a setup pane. |
| 9 | Enter the variable name (Count). |
| 10 | Activate the setup. |

**Software versions**

The sample program requires the software versions listed below:

- JetSym: Version 5.1.3 or higher
- Controller JC-970MC: OS version 1.07 or higher

You can also find the latest sample programs in the JetSym online help.

**JetSym STX program**

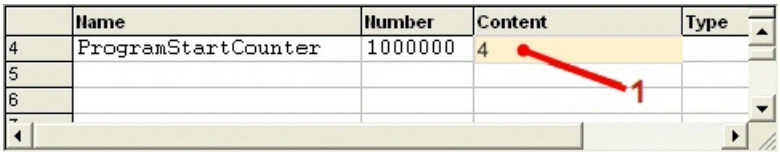
The following program increments the content of a variable by one every 2 seconds:

```
Var
    Count:    Int;
End_Var;

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

**Setup pane**

The JetSym setup pane shows the content of the variable:




| Number | Description | Function |
|---|---|---|
| 1 | Present content of the variable | The content of the variable is incremented by one every 2 seconds. |

# 5 Retrieving information on the controller

**Purpose of this chapter**

This chapter describes how to retrieve the following information on the controller JC-970MC:

- Hardware revision
- Electronic Data Sheet (EDS) information. Numerous production-relevant data are stored in the EDS.
- OS version of the controller and its software components

**Prerequisites**

To be able to identify the JC-970MC, the following prerequisites must be fulfilled:

- The controller must be connected to a PC.
- The programming tool JetSym 4.2 or higher must be installed on the PC.

**Information for hotline requests**

If you wish to contact the hotline of Jetter AG in case of a problem, please have the following information on the JC-970MC ready:

- Serial number
- OS version number of the controller
- Hardware revision

**Content**

# 5.1 Identification by means of the nameplate

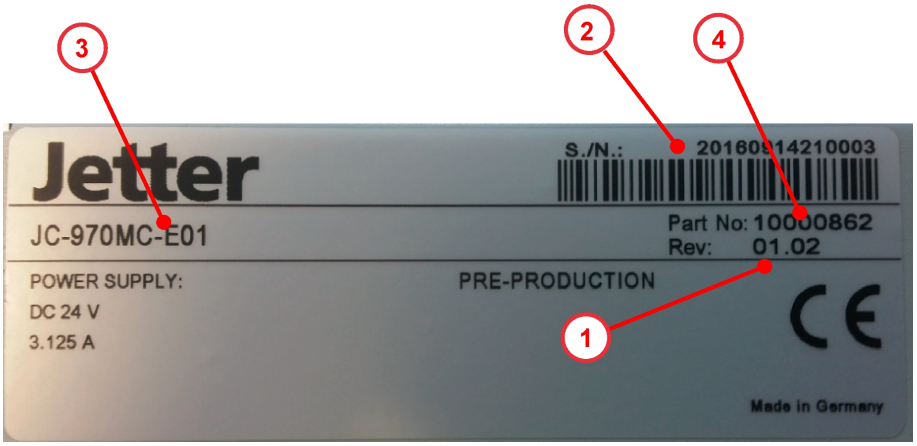
**Introduction**

The nameplate is attached to the housing of the JC-970MC and contains details, such as hardware revision number and serial number. If you wish to contact the hotline of Jetter AG in case of a problem, please have this information ready.

## Nameplate

**Nameplate**

The nameplate attached to the right side of the JC-970MC contains the following information:




| Number | Description |
|---|---|
| **1** | Hardware revision |
| **2** | Serial number |
| **3** | Controller name |
| **4** | Item number |

# 5.2 Electronic Data Sheet EDS

**Introduction**

Each JC-970MC features an Electronic Data Sheet (EDS). Numerous production-relevant data are permanently stored in the EDS. Files in the file system of the JC-970MC or special registers let you retrieve EDS data.

## EDS File "eds.ini"

**Introduction**

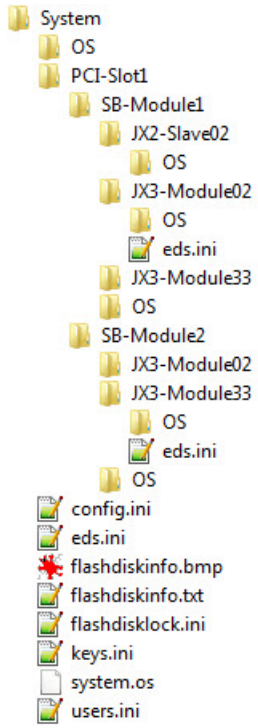
EDS data can be retrieved from the file **eds.ini**.

**Properties**

- You can access this file through the file system of the controller.
- For an FTP connection, the user needs administrator rights (user *admin*) or system rights (user *system*).
- The EDS file of the controller is located in the folder */System*.
- This file is read-only.
- The EDS file will even survive formatting of the disk drive.

**Path to EDS files**

The illustration below shows an example of the folder */System* holding the EDS files of the controller:




**File structure**

The EDS file is a text file the entries of which are grouped into several sections.

**Example - Controller**

Below is an example of an EDS file belonging to a JetControl 970MC:

```
;Jetter AG Electronic Data Sheet

[IDENTIFICATION]
Version = 0
Code = 2464
Name = JC-970MC
PcbRev = 02
PcbOpt = 00

[PRODUCTION]
Version = 0
SerNum = 10080703010015
Day = 14
Month = 08
Year = 2016
TestNum = 1
TestRev = 01.10.03.50

[FEATURES]
Version = 0
Axes = 64
NumberOfPCISlots = 02
STX = 1
NVRegs = 120000
```

**Section [IDENTIFICATION]**

General hardware configuration can be seen from section [IDENTIFICATION].

| Name | Example | Description |
|---|---|---|
| Version | 0 | Version of this section |
| Code | 2464 | Module code of JC-970MC |
| Name | JC-970MC | Corresponds to the information on the name-plate |
| PcbRev | 02 | Hardware revision |
| PcbOpt | 00 | Hardware option |

**Section [PRODUCTION]**

Serial number and production date can be seen from section [PRODUCTION].

| Name | Example | Description |
|---|---|---|
| Version | 0 | Version of this section |
| SerNum | 20160814210003 | Corresponds to the information on the name-plate |
| Day | 14 | Production date: Day |
| Month | 08 | Production date: Month |
| Year | 2016 | Production date: Year |
| TestNum | 1 | Internal usage |
| TestRev | 01.10.03.50 | Internal usage |

**Section [FEATURES]**

Special properties of the controller can be seen in section [FEATURES]. The OS of the controller will ignore properties of missing entries in the file.

| Name | Example | Description |
|---|---|---|
| `Version` | 0 | Version of this section |
| `Axes` | 64 | Number of MC-axes |
| `Number-ofPCIslots` | 02 | Number of PCIe slots |
| `STX` | 1 | Runtime environment for the application program is available |
| `NVRegs` | 120000 | Number of non-volatile application registers |

# EDS registers

**Introduction**

EDS registers let you retrieve entries from the Electronic Data Sheet (EDS).

**Register numbers**

The basic register number is dependent on the controller. The register number is calculated by adding the number of the module register (MR) to the number of the basic register.

| Controller | Basic register number | Register numbers |
|---|---|---|
| JC-970MC | 100000 | 100600 ... 100817 |

**EDS registers of a controller**

The following table lists the EDS registers of a controller, as well as their connection to the entries in the EDS file **/System/eds.ini**.

| Register | Section in eds.ini | Name in eds.ini | Description |
|---|---|---|---|
| **MR 600** | IDENTIFICATION | `Version` | Version of this section |
| **MR 601** | | `Code` | Module code |
| **MR 602 ... MR 612** | | `Name` | Module name or controller name |
| **MR 613** | | `PcbRev` | Hardware revision |
| **MR 614** | | `PcbOpt` | Hardware option |
| **MR 700** | PRODUCTION | `Version` | Version of this section |
| **MR 701 ... MR 707** | | `SerNum` | Serial number |
| **MR 708** | | `Day` | Production date: Day |
| **MR 709** | | `Month` | Production date: Month |
| **MR 710** | | `Year` | Production date: Year |
| **MR 711** | | `TestNum` | Internal usage |
| **MR 712** | | `TestRev` | Internal usage |
| **MR 800** | FEATURES | `Version` | Version of this section |
| **MR 804** | | `Switch` | Mode selector |
| **MR 805** | | `STX` | Runtime environment for the application program |
| **MR 806** | | `NVRegs` | Number of non-volatile registers |
| **MR 810** | | `Motion-Control` | MC software |
| **MR 811** | | `Axes` | Number of MC-axes |
| **MR 812** | | `Web` | Web server and e-mail client |
| **MR 813** | | `ModbusTCP` | Modbus/TCP client and server |
| **MR 817** | | `RTC` | Real-time clock |

# 5.3 Version registers

**Introduction**

The operating system provides several registers which can be used to read out revision numbers of the hardware or OS and its components. If you wish to contact the hotline of Jetter AG in case of a problem, please have this information ready.

## Hardware revisions

**Introduction**

The controller JC-970MC features special registers, the content of which lets you identify the hardware.

**Registers - Overview**

The registers listed below let you retrieve the hardware revisions:

| Register | Description |
|---|---|
| **200170** | Controller type |

**Version numbers in JetSym setup**

The following screenshot shows a JetSym setup pane displaying the version registers:

| | Name | Number | Content | Type | Comment |
|---|---|---|---|---|---|
| 1 | | | | | |
| 2 | JCtype | 200170 | 970 | | |
| 3 | | | | | |
| 4 | | | | | |

## Software versions

**Introduction**

The JC-970MC features software with unique version numbers which can be read out from special registers.

**Format of software version numbers**

The software version numbers of the JC-970MC are four-figure values.

| 1 | . | 2 | . | 3 | . | 4 |
|---|---|---|---|---|---|---|

| Element | Description |
|---|---|
| **1** | Major or main version number |
| **2** | Minor or secondary version number |
| **3** | Branch or intermediate version number |
| **4** | Build version number |

**Released version**

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

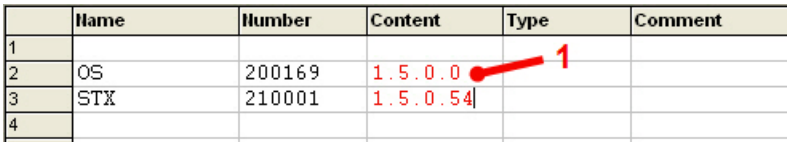
**Registers - Overview**

The registers listed below let you read out the software versions:

| Register | Description |
|---|---|
| **200169** | Operating system version |
| **210001** | Version of the runtime unit for the STX application program |

**Version numbers in JetSym setup**

The following screenshot shows a JetSym setup pane displaying version registers. To have the version number displayed in the setup pane of JetSym, select the format **IP address**.




| Number | Description | Function |
|---|---|---|
| **1** | V. 1.05.0.00 | OS version number of the controller<br>JetSym displays this information in the title bar of each setup pane. |

# 6 Programming

**Introduction**

JetControl 970MC is an embedded control system with a real-time multitasking operating system. The user can freely distribute the control program to up to 100 tasks which are processed by the controller virtually parallel to each other.

The programming language used is JetSym-STX which is based on IEC-61131-3 (ST). Although JetSym STX does not fully comply with IEC-61131-3 (ST), its scope of functions goes beyond this standard.

**Purpose of this chapter**

This chapter supports you in programming the following functions:

- Determining the register numbers of connected modules
- Determining the I/O numbers of connected modules
- Programming additional functions
- exchange data via various system buses
- exchange data via user-programmable interfaces

**Prerequisites**

To program the controller JC-970MC, the following requirements must be met:

- The controller must be connected to a PC.
- The JetSym programming software must be installed on the PC.

**Content**

# Abbreviations, module register properties and formats

**Abbreviations**

The following abbreviations are used in this manual:

| Abbreviation | Description |
|---|---|
| R 100 | Register 100 |
| MR 150 | Module register 150 |

**Module register properties**

Each module register is characterized by certain properties. For many module registers most properties are identical. For example, their value after reset is zero. In the following description, module register properties are mentioned only if a property deviates from the default properties listed below.

| Module register property | Default property for most module registers |
|---|---|
| Type of access | Read/write |
| Value after reset | 0 or undefined (e.g. revision/version number) |
| Takes effect | Immediately |
| Write access | Always |
| Data type | Integer |

**Numerical formats**

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

| Notation | Numerical format |
|---|---|
| 100 | Decimal |
| 0x100 | Hexadecimal |
| 0b100 | Binary |

**JetSym sample pograms**

The notation for sample programs used in this document is listed in the table below:

| Notation | Purpose |
|---|---|
| `Var, When, Task` | Keyword |
| `BitClear();` | Commands |
| `100 0x100 0b100` | Constant numerical values |
| `// This is a comment` | Comment |
| `// ...` | Further program processing |

## System run times, response times, remote peripheral devices

**System run times**

For more information on system run times refer to our application-oriented manual “System Registers” which is available for downloaded from our homepage.

**Response times**

You can read out program response times from registers 210007 (minimum program cycle time), 210008 (maximum program cycle time), and 210009 (current program cycle time).

**Peripherals not directly installed**

For peripheral devices not installed directly in the controller, such as JX3 modules on a JX3-BN-ETH, you can define how they must behave in the event of communication loss. For more information refer to the application-oriented manual “Jetter Ethernet System Bus” and the online help which comes with JetSym.

If a communication error occurs with a connected JetMove servo amplifier, it is detected by the Motion Control or the STX program via timeout.

## Self-test and configuration check

**Existing tests**

The programming language JetSym STX offers numerous possibilities to check the configuration and test the system, e.g. memory test, memory protection, illegal op code, division by 0, consistency of axis and IO configuration on the Ethernet system bus (NetConsistency), Try..Catch, etc. For more information on these topics refer to the JetSym online help.

# 6.1 Memory types - Overview

**Introduction**

The JC-970MC features several types of program and data memories. There is, for example, 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.

The memory is located directly in the CPU or in separate memory or I/O modules.

This chapter gives an overview of the available memory.

**Content**

## Operating system memory

| | |
|---|---|
| **Introduction** | The OS is stored to a non-volatile flash memory in the CPU. Therefore, the OS can be executed immediately after the device is powered up. |
| **Properties** | ▪ Internal flash memory for storing OS data<br>▪ Internal volatile RAM for storing OS data |
| **Memory access** | ▪ The user is not allowed to directly access the OS memory.<br>▪ Changes to the OS can only be made by means of an OS update. |
| **Related topics** | ▪ **Updating the operating system of the controller** (see page 186) |

## File system memory

| | |
|---|---|
| **Introduction** | The file system memory is for storing data and program files. |
| **Properties** | ▪ Internal flash disk<br>▪ Non-volatile<br>▪ Slow access: Milliseconds up to seconds<br>▪ Limited number of write/erase cycles: approx. 1 million<br>▪ Size of internal flash disk: 32 MB |
| **Memory access** | ▪ By controller OS (RTOS)<br>▪ By JetSym<br>▪ Via FTP connection<br>▪ By the e-mail client<br>▪ By browser (via HTTP server)<br>▪ By means of file commands from within the application program |

## Application program memory

**Introduction**

By default, the application program is uploaded by JetSym to the controller as a compiled file with the extension .es9 where it is stored in a mass memory (flash). If the "Autorun" option is activated in the JetSym Hardware Manager, the program is loaded into the main memory (SDRAM) after the download and then executed.

**Properties**

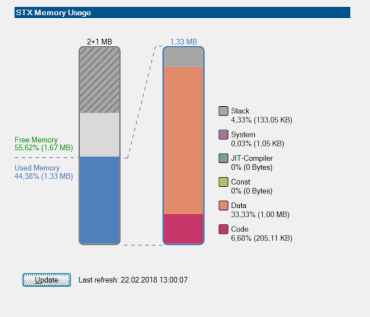
- Stored as file within the file system
- Default directory */app*
- Files may also be stored to other directories
- Size is limited by flash memory

**Memory access**

- By controller OS (RTOS)
- By JetSym
- Via FTP connection
- By means of file commands from within the application program

**Memory usage**

- After program start, the programming tool JetSym lets you determine the memory utilization by the application program in the SDRAM. Further information can be found in the JetSym online help under the search term "Memory usage".




## Memory for volatile application program variables

**Introduction**

Volatile variables are used to store data which may be discarded when the JC-970MC is de-energized.

**Properties**

- They are global variables which are not assigned to permanent addresses (not %VL or %RL)
- Local variables
- Variables are stored in a compact way.
- Variables are initialized with value 0 when they are created.

**Memory access**

- By JetSym
- From the application program

# Memory for non-volatile application program variables

**Introduction**

Non-volatile registers let you store data (variables) which must be saved when the JC-970MC is de-energized. These variables can be declared as %RL or %VL.

**Notice!**

Data can get lost!
During runtime, the operating system copies the area of non-volatile application registers into the volatile SDRAM memory because data can there be processed faster by the STX program.
If the voltage supply to the device is disconnected, a PowerFail interrupt is thrown. This interrupt causes all register variables to be copied from volatile to non-volatile memory.
An internal power supply provides the energy required for this process.
Pressing the PWR button will **not** cause the PowerFail interrupt to be thrown. Thus, all data stored in the SDRAM memory will be discarded. That's the reason why we recommend not to use the PWR button, but to interrupt the power supply instead.

**Properties of %VL variables**

- Global variables with dedicated addresses (%VL)
- Register variables always occupy 4 bytes.
- Register variables are not initialized by the operating system.
- Number of register variables: 120,000
- Register numbers: 1000000 ... 1119999

**Memory access**

- By JetSym
- By the e-mail client
- By browser (via HTTP server)
- From HMIs
- From the application program
- From other controllers

**JetSym STX program**

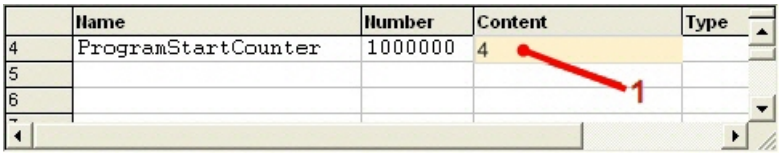
The following sample program increments the content of a register variable every time the application program is started. This way, the number of program starts is counted.

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

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

**Setup pane**

The JetSym setup pane displays the content of the register variable.




| Number | Description | Function |
|---|---|---|
| 1 | Present content of the register variable | The content of the register variable is incremented by one every time the program is started. |

**Properties of %RL variables**

- Global variables which are assigned to dedicated registers (%RL)
- Variables are stored in a compact way
- Size: 480,000 bytes
- Register numbers: 1000000 ... 1119999

**Memory access**

- By JetSym
- From the application program

**JetSym STX program**

The following program increments the content of four non-volatile variables every second.

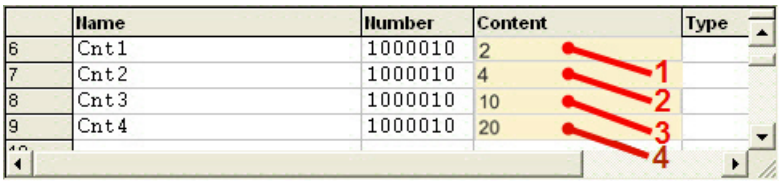
The working range of the counters is between 0 and 255 (variable type: byte). For these four variables the four bytes of register 1000010 are used.

```
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;
```

**Setup pane**

The JetSym setup pane displays the content of the variable. As the type of the four counters is byte, this will result in counter overflow after a relatively short time:



| Number | Description | Function |
|---|---|---|
| **1** | Current content of the variable Cnt1 | The content of the variable is incremented by one every second. |
| **2** | Current content of the variable Cnt2 | The content of the variable is incremented by two every second. |
| **3** | Current content of the variable Cnt3 | The content of the variable is incremented by five every second. |
| **4** | Current content of the variable Cnt4 | The content of the variable is incremented by ten every second. |

## Special registers

**Introduction**

Special registers let you control OS functions and retrieve status information.

**Properties**

- Global variables with dedicated addresses (%VL)
- When the operating system is launched, special registers are initialized using their default values.
- Register numbers: 100000 ... 999999
- Network registers: 1nnnxxxxxx (nnn = GNN)

**Memory access**

- By JetSym
- By the e-mail client
- By browser (via HTTP server)
- From HMIs
- From the application program
- From other controllers

## Inputs and outputs

**Introduction**

Inputs and outputs are 1-bit variables. This means they can have the value TRUE or FALSE. Inputs and outputs are addressed in immediate read/write cycles.

**Properties of virtual inputs/outputs**

- They are global variables assigned to permanent addresses (%IX, %QX)
- Used for RemoteScan via Modbus/TCP
- Quantity: 16,000
- I/O numbers: 20001 ... 36000

**Properties of digital inputs/outputs**

- They are global variables assigned to permanent addresses (%IX, %QX)
- Located on modules connected to the JX2 system bus
- I/O numbers on the JX2 system bus: 201100201 ... 201203216
- I/O numbers of remote devices connected to a JX3-BN-ETH: 1nnn010201 ... 1nnn011716 (nnn = GNN)

**Memory access**

- By JetSym
- By the e-mail client
- By browser (via HTTP server)
- From HMIs
- From the application program

**JetSym STX program**

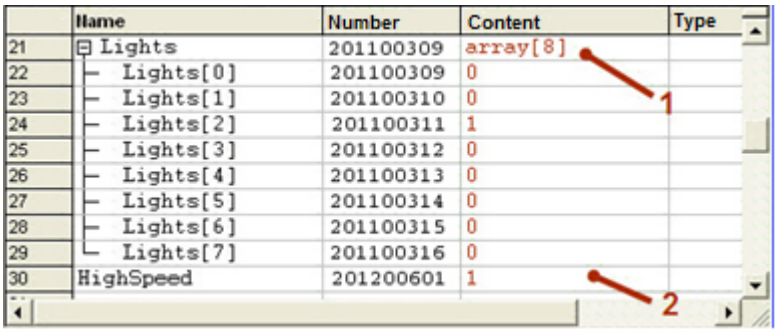
In the following program, moving lights are triggered on a digital output module connected to the first JX2 system bus. If a digital input is set at the second JX2 system bus, the corresponding output is activated for 100 ms.

```
Var
    Lights:    Array[8] Of Bool At %QX 201100309;
    HighSpeed:    Bool At %IX 201200601;
End_Var;

Task RunningLight Autorun
    Var
        Idx:    Int;
    End_Var;
    Loop
        For Idx := 0 To 7 Do
            Lights[Idx] := True;
            If HighSpeed Then
                Delay(T#100ms);
            Else
                Delay(T#300ms);
            End_If;
            Lights[Idx] := False;
        End_For;
    End_Loop;
End_Task;
```

**Setup pane**

The JetSym setup pane displays the state of inputs and outputs:




| Number | Description | Function |
|---|---|---|
| **1** | Present state of outputs | The outputs are set and reset one after another. |
| **2** | Present state of the output | When the input is set, the corresponding output is activated for 100 ms. |

# Flags

| | |
|---|---|
| **Introduction** | Flags are one-bit operands. This means they can have the value TRUE or FALSE. |
| **Properties of user flags**<br>▪ Non-volatile | ▪ Global variables which are assigned to permanent addresses (%MX)<br>▪ Quantity: 256<br>▪ Flag numbers: 0 ... 255 |
| **Properties of overlaid user flags**<br>▪ Overlaid by registers 1000000 ... 1000055 | ▪ Global variables which are assigned to permanent addresses (%MX)<br>▪ Non-volatile<br>▪ Quantity: 1,792<br>▪ Flag numbers: 256 ... 2047 |
| **Properties of special flags** | ▪ Global variables which are assigned to permanent addresses (%MX)<br>▪ When the operating system is launched, special flags are initialized using their default values.<br>▪ Quantity: 256<br>▪ Flag numbers: 2048 ... 2303 |
| **Memory access** | ▪ By JetSym<br>▪ By the e-mail client<br>▪ By browser (via HTTP server)<br>▪ From HMIs<br>▪ From the application program |

# 6.2 Numbering registers and I/Os

**Introduction**

Controllers and modules produced by Jetter AG offer a host of functions which can be accessed by the user via registers. A unique number is assigned to each register and each digital input or output.

**Purpose of register numbers**

Register numbers are applied in the following cases:

- Read/write access to a module register in the JetSym Setup pane.
- Declaration of a module register in a JetSym application program.
- Declaration of a module register as tag in JetViewSoft.

**Purpose of I/O numbers**

I/O numbers can be used in the following cases:

- Read access to a digital input in the JetSym Setup pane.
- Read/write access to a digital output in the JetSym Setup pane.
- Declaration of a digital input/output as variable in a JetSym application program.
- Declaration of a digital input/output as tag in JetViewSoft.

**Content**

# Registers and module registers

**Module registers - Definition**

Module registers let you read process, configuration and diagnostics data from the module, or write such data to it. The module register number within a module is unique.

**Access options**

There are several ways to access registers directly:

- In an application program
- In a JetSym setup pane
- In a visualization application

The register number within the system is unique.

**Definition - Global Node Number**

The Global Node Number (GNN) is an ID number to identify Jetter controllers (e.g. controllers, bus nodes) within an Ethernet network.

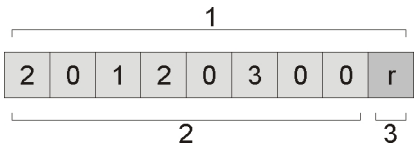
- The GNN within a network has to be unambiguous for each Jetter device.
- The JetSym Hardware Manager automatically assigns the GNN during configuration.
- The value range of the GNN within a project is 000 ... 199.
- The controller has always got GNN 000.

**Example: Module register**

Module register 9 lets you read out the OS revision of a JX2-IO16 module:

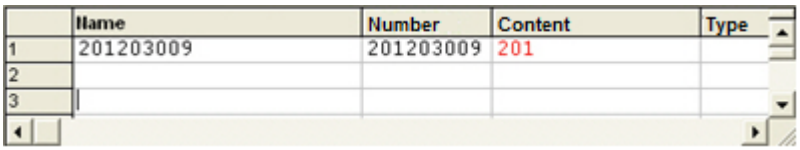
**Example: Register numbers on the JX2 system bus**

A JX2-IO16 module is connected as first module to the upper JX2 system bus module located in the PCIe slot 1 (riser card 1). On the JX2 system bus, the module number of this module is 2.



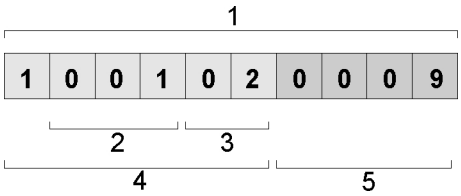

| Number | Element | Description |
|---|---|---|
| **1** | Register number | Can be used to directly access the register |
| **2** | Register prefix | 20120300: For the first JX2-I/O module connected to a JC-970MC. |
| **3** | Module register number | r = 9: OS version of the JX2-IO16 |

Register number 201203009 in the JetSym Setup pane shows OS version 2.01.



**Example: Registers on the Ethernet system bus**

A JX3-AI4 module is connected to a bus node JX3-BN-ETH. The module number of the JX3 module is 2. The bus node has got the ID (GNN) 001.



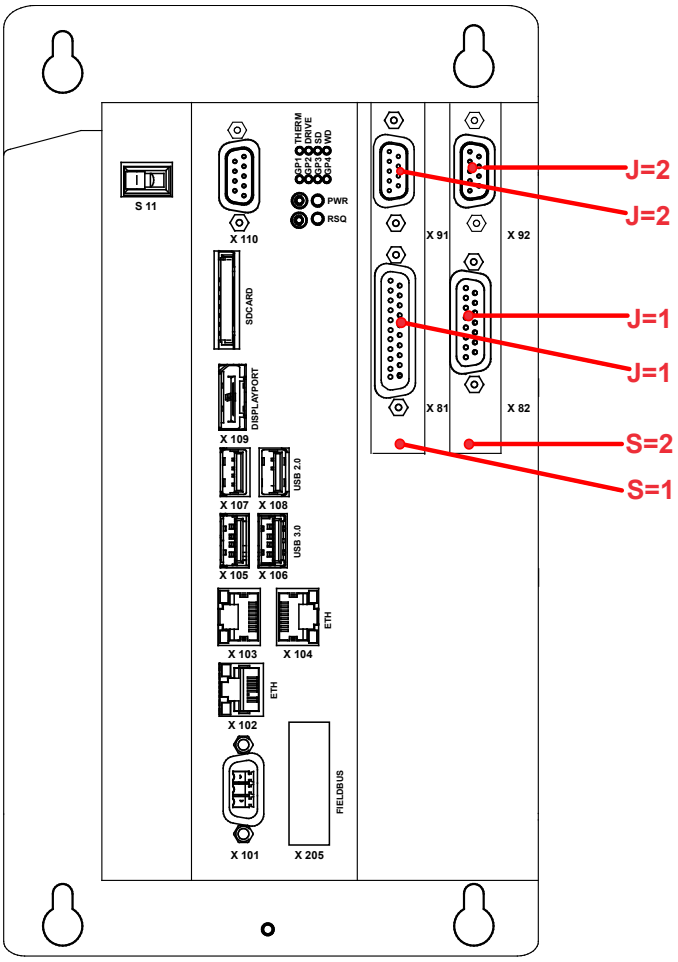

| Number | Element | Description |
|---|---|---|
| **1** | Register number | Can be used to directly access the register |
| **2** | Bus node ID, GNN | 001: ID of the first JX3-BN-ETH |
| **3** | Module number | 02: The first JX3-I/O module connected to the JX3-BN-ETH |
| **4** | Register prefix | 100102 |
| **5** | Module register number | 0009: OS version of the JX3-AI4 |

Register number 1001020009 in the JetSym Setup pane shows OS version 1.4.0.0 in direct representation.

| | Name | Number | Content | Type |
|---|---|---|---|---|
| 1 | 1001020009 | 100102000 | 1.4.0.0 | |
| 2 | | | | |

## Slot numbering

### Configuration




| Code | Description |
|---|---|
| S | Slot number of the riser card on the PCIe bus |
| J | Number of the JX6-I/O submodule located on the riser card |

### Numbering

The slot number breaks down as follows:

- The left riser card on the PCIe bus is assigned number 1.
- The right riser card on the PCIe bus is assigned number 2.
- The lower module on the riser card is assigned number 1.
- The upper module on the riser card is assigned number 2.

# Register and I/O numbers of modules located on a riser card

**Register numbers of modules located on the riser card**

The register number of modules located on a riser card breaks down as follows:

| 2 | 0 | S | J | 0 | 0 | z | z | z |
|---|---|---|---|---|---|---|---|---|

| Element | Purpose | Value range |
|---|---|---|
| **S** | Number of the riser card | 1 ... 2 |
| **J** | Number of the module located on the riser card | 1 ... 2 |
| **zzz** | Module register number | 100 ... 999 |

**I/O numbers of modules located on a riser card**

The I/O number of modules located on a riser card breaks down as follows:

| 2 | 0 | S | J | 0 | 0 | 1 | z | z |
|---|---|---|---|---|---|---|---|---|

| Element | Purpose | Value range |
|---|---|---|
| **S** | Number of the riser card | 1 ... 2 |
| **J** | Number of the module located on the riser card | 1 ... 2 |
| **zz** | Module-specific I/O number | 01 ... 08 |

# Register numbers of JX2 slave modules connected to the JX2 system bus

**Slave module numbers of JX2 slave modules**

To determine the slave module numbers of intelligent JX2 slave modules and JetMoves on the JX2 system bus of the JC-9xx, proceed as follows:

- Count the JX2 slave modules left-to-right, starting from 2.
- Do not count the power supply module JX2-PS1.
- Leave out non-intelligent JX2-I/O modules.

**Register numbers for JX2 slave modules**

Register numbers for JX2 slave modules on the JX2 system bus of the JC-9xx break down as follows:

| 2 | 0 | S | J | x | x | z | z | z |
|---|---|---|---|---|---|---|---|---|

| Element | Purpose | Value range |
|---|---|---|
| **S** | Number of the riser card | 1 ... 2 |
| **J** | Number of the JX6-I/O module (JX2 system bus) located on the riser card | 1 ... 2 |
| **xx** | Slave module number + 10 | 12 ... 27 |
| **zzz** | Module register number | 000 ... 999 |

# Register and I/O numbers of JX2- and JX3-I/O modules on the JX2 system bus

## I/O module numbers of JX2- and JX3-I/O modules

To determine the I/O module numbers of JX2- or JX3-I/O modules on the JX2 system bus of the JC-9xx, proceed as follows:

- Count the JX2- and JX3-I/O modules left-to-right, starting from 2.
- Leave out the intelligent JX2 slave modules and JetMoves.
- Leave out the power supply module JX2-PS1.
- Count the JX3-BN-CAN modules left-to-right, starting from 33.

**Notice!**

When JX3-I/O modules are used, set the timeout value in module register 20SJ02764 of the JX6-SB(-I) module to 8.

If JX3-DI/DO/DIO modules are used, their process data registers (MR 2-5) must not be used. For read or write access to registers overlaid by inputs and outputs use the registers listed in Quick Reference.

## Register numbers for JX2- and JX3-I/O modules

Register numbers for JX2- and JX3-I/O modules connected to the JX2 system bus of the JC-9xx consist of the following elements:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 2 | 0 | S | J | 0 | 3 | x | x | z |

| Element | Purpose | Value range |
|---|---|---|
| **S** | Number of the riser card | 1 ... 2 |
| **J** | Number of the JX6-I/O module (JX2 system bus) located on the riser card | 1 ... 2 |
| **xx** | JX2-I/O module number minus 2<br>Module number of the JX3-BN-CAN minus 2 | 00 ... 30<br>31 ... 61 |
| **z** | Module register number | 0 ... 9 |

## I/O numbers for JX2- and JX3-I/O modules

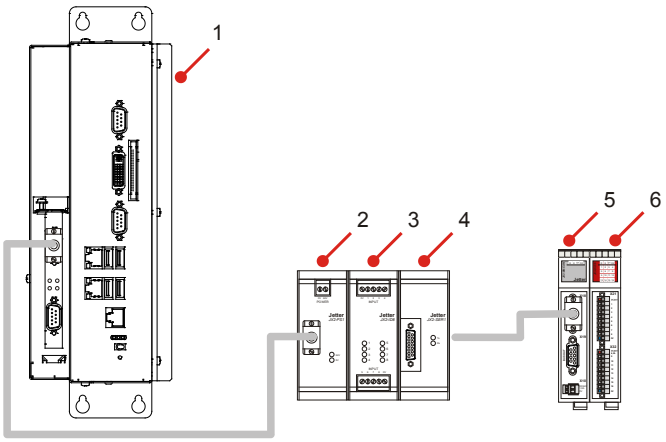
I/O numbers for JX2- and JX3-I/O modules connected to the JX2 system bus of the JC-9xx consist of the following elements:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 2 | 0 | S | J | 0 | x | x | z | z |

| Element | Purpose | Value range |
|---|---|---|
| **S** | Number of the riser card | 1 ... 2 |
| **J** | Number of the SB/SB-I module located on the riser card | 1 ... 2 |
| **xx** | Module-specific I/O number of the JX2/JX3 module | 02 ... 32 |
| **zz** | I/O number on the JX2/JX3 module | 1 ... 16 |

**Example**

Several JX2- and JX3-I/O modules are connected to a JC-9xxMC.




| Number | Module | I/O module number | Register | I/O |
|---|---|---|---|---|
| **1** | JC-9xxMC | 1 | Refer to documentation on JC-970MC. | |
| **2** | JX2-PS1 | - | - | - |
| **3** | JX2-ID8 | 2 | 20120300z | 2012002zz |
| **4** | JX2-SER1 | 3 | 20120301z | 2012003zz |
| **5** | JX3-BN-CAN | 33 | 20120331z | - |
| **6** | JX3-DIO16 | 4 | 20120302z | 2012004zz |

# Register and I/O numbers of CANopen® modules on the JX2 system bus

**I/O module numbers of CANopen® modules**

To determine the I/O module numbers of CANopen® modules on the JX2 system bus of the JC-9xx, proceed as follows:

- Determine the node ID. In most cases, the I/O module numbers correspond to the node ID of the CANopen® module.
- Exceptions: SMC EX120 and Lenze frequency inverter

**Register numbers for CANopen® modules**

Register numbers for CANopen® modules connected to the JX2 system bus of the JC-9xx consist of the following elements:

| 2 | 0 | S | J | 0 | x | x | z | z |
|---|---|---|---|---|---|---|---|---|

| Element | Purpose | Value range |
|---|---|---|
| **S** | Number of the riser card | 1 ... 2 |
| **J** | Number of the JX6-I/O module (JX2 system bus) located on the riser card | 1 ... 2 |
| **xx** | I/O module number | 70 ... 79 |
| **zz** | Module register number | 00 ... 99 |

**I/O numbers for CANopen® modules**

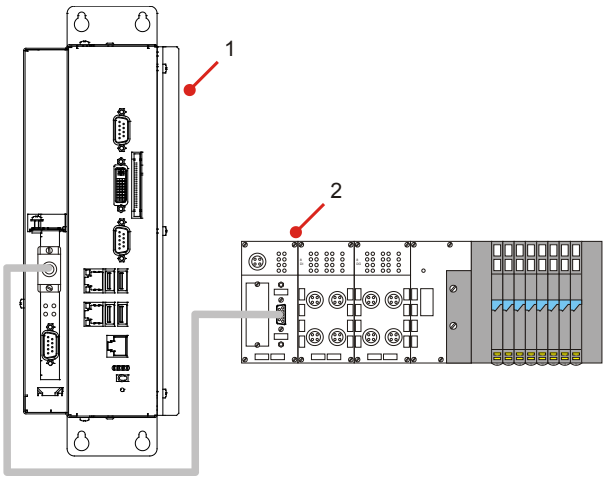
I/O numbers for CANopen® modules connected to the JX2 system bus of a JC-9xx consist of the following elements:

| 2 | 0 | S | J | 0 | x | x | z | z |
|---|---|---|---|---|---|---|---|---|

| Element | Purpose | Value range |
|---|---|---|
| **S** | Number of the riser card | 1 ... 2 |
| **J** | Number of the JX6-I/O module (JX2 system bus) located on the riser card | 1 ... 2 |
| **xx** | Module-specific I/O module number | 70 ... 79 |
| **zz** | Module-specific I/O number | 1 ... 64 |

**Example**

A CANopen® module is connected to a JC-9xxMC.




| Number | Module | I/O module number | Register | I/O |
|---|---|---|---|---|
| **1** | JC-9xxMC | 1 | Refer to documentation on JC-970MC. | |
| **2** | Festo CPX | 2 | 2012070zz | 2012070zz |

# Register and I/O numbers of JX3 modules connected to a JX3-BN-ETH

**Global Node Number of a JX3-BN-ETH on the Jetter Ethernet system bus**

JetSym Hardware Manager assigns a Global Node Number to the bus node JX3-BN-ETH on the Jetter Ethernet system bus.

**Register numbers for JX3 modules**

The register number for JX3 modules connected to an Ethernet bus node consists of the following elements:

| 1 | n | n | n | x | x | z | z | z | z |
|---|---|---|---|---|---|---|---|---|---|

| Element | Purpose | Value range |
|---|---|---|
| **nnn** | Global Node Number of the JX3-BN-ETH on the Ethernet system bus | 001 ... 199 |
| **xx** | Module number of the module within the JX3 station | 02 ... 17 |
| **zzzz** | Module register number | 0000 ... 9999 |

**I/O numbers for JX3 modules**

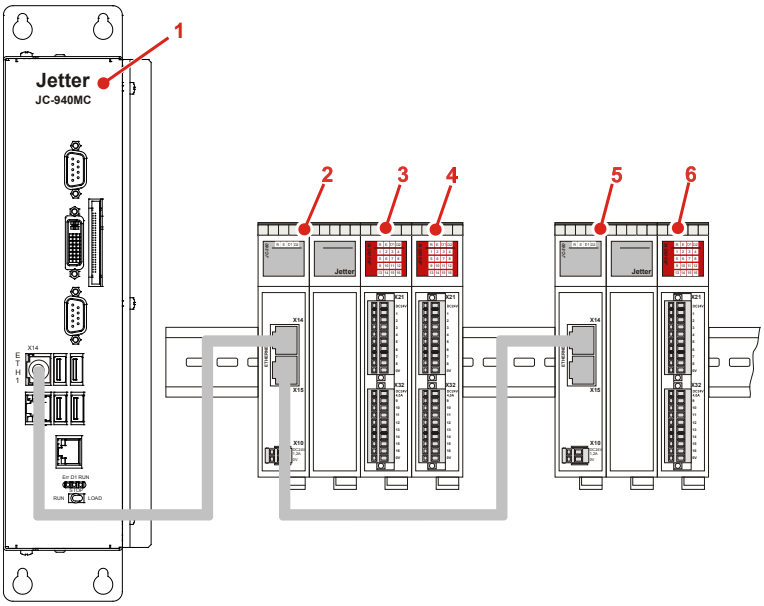
The I/O number for JX3 modules connected to an Ethernet bus node consists of the following elements:

| 1 | n | n | n | 0 | 1 | x | x | z | z |
|---|---|---|---|---|---|---|---|---|---|

| Element | Purpose | Value range |
|---|---|---|
| **nnn** | Global Node Number of the JX3-BN-ETH on the Ethernet system bus | 001 ... 199 |
| **xx** | Module number of the module within the JX3 station | 02 ... 17 |
| **zz** | Module-specific I/O number | 1 ... 16 |

**Example**

Two bus nodes JX3-BN-ETH are connected to a JC-940MC (as an example).




| Number | Module | Module number | GNN | Register | I/O |
|---|---|---|---|---|---|
| **1** | JC-9xxMC | 1 | 0 | Refer to documentation on JC-970MC. | |
| **2** | JX3-BN-ETH | - | 1 | Refer to documentation on JX3-BN-ETH | |
| **3** | JX3-DIO16 | 2 | | 100102zzzz | 10010102zz |
| **4** | JX3-DIO16 | 3 | - | 100103zzzz | 10010103zz |
| **5** | JX3-BN-ETH | - | 2 | Refer to documentation on JX3-BN-ETH | |
| **6** | JX3-DIO16 | 2 | - | 100202zzzz | 10020102zz |

## Real-time clock

**Introduction**

The controller is equipped with a realtime clock. This clock continues to run for some time even when the controller is de-energized. Then, the realtime clock is supplied with power from a GoldCap capacitor. Thus, the realtime clock is maintenance-free because there is no backup battery which needs to be replaced.

The realtime clock supports the Network Time Protocol (NTP) which is for synchronizing the time with a server on the Internet.

**Purpose**

Purpose of the realtime clock:

- Specifying the file date and time when creating a log file with timestamp

**Restrictions**

When using the real-time clock, the following restrictions apply:

- When the device is de-energized the power reserve is limited.
- The realtime clock has no automatic daylight savings time function.
- The realtime clock can only be set from Windows (IPC).
- When the clock has been set, wait 60 seconds before shutting down the device. Otherwise, any changes will be discarded.
- From the RTOS side the realtime clock is read-only.

## Technical specifications

**Technical data - Realtime clock**

| Parameter | Description |
|---|---|
| Power reserve<br>The controller must have been energized for at least 1 hour. | Minimum: 1 week<br>Typically: 3 weeks |
| Deviation | 1 minute per month max. |

**Factory settings**

At the end of the controller manufacturing process, the real-time clock is set to the actual date and time.

# 6.3 Programming local JX6 submodules

**Introduction**

This chapter describes how to program local JX6 submodules on the PCIe bus of the JC-970MC.

**Content**

# JX2- Systembus interface JX6-SB(- I)

**Introduction**

This chapter describes how to configure the JX2 system bus interface on the PCIe bus of the JC-970MC.

**Submodule JX6-SB(-I)**

The local submodules JX6-SB and JX6-SB-I take on converting the PCIe bus to the JX2 system bus. The operating mode is Master /Slave.

**Register**

The following registers let you configure the JX2 system bus:

- **Configuration and status registers of the JX6-SB(-I) submodule**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 2 | 0 | S | J | 0 | 0 | z | z | z |

| Element | Purpose | Value range |
|---|---|---|
| **S** | Number of the riser card | 1 ... 2 |
| **J** | Number of the JX6-I/O submodule located on the riser card | 1 ... 2 |
| **zzz** | Module register number | 100 .... 163 |

- **JX2 system bus registers**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 2 | 0 | S | J | 0 | 2 | z | z | z |

| Element | Purpose | Value range |
|---|---|---|
| **S** | Number of the riser card | 1 ... 2 |
| **J** | Number of the JX6-I/O submodule located on the riser card | 1 ... 2 |
| **zzz** | Module register number | 000 .... 999 |

**Further literature**

For more information on registers and engineering of the JX2 system bus, refer to the JX6-SB(-I) user information or application-oriented manual. These manuals are available for download on our homepage.

**Enhancements**

Since revision 2.11.2 of the JX6-SB user information the following enhancements have been made:

- The value range I/O module numbers on the JX2 system bus
  - old: 0, 2 ... 32, 70 ... 79
  - new: 0, 2 ... 64, 70 ... 79
- The value range of the number of connected JX2-I/O and JX-SIO modules (register 2013)
  - old: 0 ... 41
  - new: 0 ... 64

- The following products have been added to the module code:
  - JX3 modules
  - JX-TP20-R
  - Festo CP-FB modules
  - LioN-S modules
  - Third-party I/O modules
  - Third-party slave modules

# Digital I/O module JX6-IO16CB

**Introduction**

The digital input/output module is equipped with eight galvanically isolated 24 V inputs, and eight galvanically isolated 24 V outputs with a maximum current of 0.5 A each.

**I/O numbers for local JX6-I/O submodules**

I/O numbers for local JX6-I/O submodules built into the JC-970MC consist of the following elements:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 2 | 0 | S | J | 0 | 0 | 1 | z | z |

| Element | Purpose | Value range |
|---|---|---|
| **S** | Number of the riser card | 1 ... 2 |
| **J** | Number of the JX6-I/O submodule located on the riser card | 1 ... 2 |
| **zz** | Module-specific I/O number | 01 ... 08 |

**Register numbers**

In this chapter only the module register number is specified. To calculate the actually used register number, add the basic register number of the corresponding controller. The basic register number is made up of the riser card number and the number of the submodule on the riser card.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 2 | 0 | S | J | 0 | 0 | z | z | z |

| Element | Purpose | Value range |
|---|---|---|
| **S** | Number of the riser card | 1 ... 2 |
| **J** | Number of the JX6-I/O submodule located on the riser card | 1 ... 2 |
| **zzz** | Module register number | 100 ... 999 |

**Registers - Overview**

The local JX6-I/O module can be programmed using a set of module registers.

| Register | Description |
|---|---|
| **MR 100** | State of the digital inputs |
| **MR 101** | Access to the digital outputs |
| **MR 102** | Error state of the digital outputs |
| **MR 103** | Control register |
| **MR 104** | Filter interval of inputs 1 to 4 |
| **MR 105** | Filter interval of inputs 5 to 8 |

**MR 100**

## Digital inputs

MR 100 shows the input state. At a voltage greater than 12 V, the bit assigned to this input is set in this register (= 1). At lower voltages, the assigned bit is not set (= 0).

| **Meaning of the individual bits** | |
|---|---|
| **Bit 0** | **Input # 1** |
| **Bit 1** | **Input 2** |
| **...** | |
| **Bit 7** | **Input 8** |

| **Module register properties** | |
|---|---|
| Type of access | Read |
| Value after reset | State of the inputs |

**MR 101**

## Digital outputs

From MR 101, the output state can be read and the outputs can be energized and de-energized. When the bit is set, the assigned output is energized. When the bit is cleared, the assigned output is de-energized.

| **Meaning of the individual bits** | |
|---|---|
| **Bit 0** | **Output 1** |
| **Bit 1** | **Output 2** |
| **...** | |
| **Bit 7** | **Output 8** |

**MR 102**

## Error state of the digital outputs

The error state of the digital outputs can be read from bit 7 of MR 102. The other bits of this register remain cleared (= 0). Even if the error is cleared, bit 7 remains set. Bit 7 can be cleared by writing any value to this register.

| **Meaning of the individual bits** | |
|---|---|
| **Bit 7** | **Collective error message of the digital outputs** |
| | 0 = No error |
| | 1 = Error in at least one output |

| **Module register properties** | |
|---|---|
| Values | 0, 128 |

| Value after reset | 128 |
|---|---|

The following errors are recognized:

- Short-circuit of the output
- Overloading of the output

**MR 103**

## Control register

The control register lets you configure the JX6-I/O module.

| **Meaning of the individual bits** | | |
|---|---|---|
| **Bit 0** | **Bit 7 in output register MR 101** | |
| | 0 = | The error state of the digital outputs is stored to bit 7 of MR 101. |
| | 1 = | The state of output 8 is stored to bit 7 of MR 101 when reading. |
| **Bit 1** | **Behavior at errors of digital outputs** | |
| | 0 = | In case of an error only the faulty output is switched off. |
| | 1 = | In case of an error all outputs are switched off |
| **Bit 3, 2** | **Filter interval for the error signal of the outputs** | |
| | 00 = | 1.5 ... 2.0 ms |
| | 01 = | 0.4 ... 0.5 ms |
| | 10 = | approx. 0.1 ms |
| | 11 = | approx. 0.02 ms |

| **Module register properties** | |
|---|---|
| Value after reset | 1 |

**MR 104**

## Filter interval of inputs 1 to 4

This register lets you set the filter interval for inputs 1 to 4. Two bits of this register have been assigned to each input.

| Meaning of the individual bits | | |
|---|---|---|
| **Bit 1, 0** | **Filter interval of input 1** | |
| | 00 = | 1.5 ... 2.0 ms |
| | 01 = | 0.4 ... 0.5 ms |
| | 10 = | approx. 0.1 ms |
| | 11 = | approx. 0.02 ms |
| **Bit 3, 2** | **Filter interval of input 2** | |
| | 00 = | 1.5 ... 2.0 ms |
| | 01 = | 0.4 ... 0.5 ms |
| | 10 = | approx. 0.1 ms |
| | 11 = | approx. 0.02 ms |
| **Bit 5, 4** | **Filter interval of input 3** | |
| | 00 = | 1.5 ... 2.0 ms |
| | 01 = | 0.4 ... 0.5 ms |
| | 10 = | approx. 0.1 ms |
| | 11 = | approx. 0.02 ms |
| **Bit 7, 6** | **Filter interval of input 4** | |
| | 00 = | 1.5 ... 2.0 ms |
| | 01 = | 0.4 ... 0.5 ms |
| | 10 = | approx. 0.1 ms |
| | 11 = | approx. 0.02 ms |

**MR 105**

## Filter interval of inputs 5 to 8

This register lets you set the filter interval for inputs 5 to 8. Two bits of this register have been assigned to each input.

| Meaning of the individual bits | | |
|---|---|---|
| **Bit 1, 0** | **Filter interval of input 5** | |
| | 00 = | 1.5 ... 2.0 ms |
| | 01 = | 0.4 ... 0.5 ms |
| | 10 = | approx. 0.1 ms |
| | 11 = | approx. 0.02 ms |
| **Bit 3, 2** | **Filter interval of input 6** | |
| | 00 = | 1.5 ... 2.0 ms |
| | 01 = | 0.4 ... 0.5 ms |
| | 10 = | approx. 0.1 ms |
| | 11 = | approx. 0.02 ms |
| **Bit 5, 4** | **Filter interval of input 7** | |
| | 00 = | 1.5 ... 2.0 ms |
| | 01 = | 0.4 ... 0.5 ms |
| | 10 = | approx. 0.1 ms |
| | 11 = | approx. 0.02 ms |
| **Bit 7, 6** | **Filter interval of input 8** | |
| | 00 = | 1.5 ... 2.0 ms |
| | 01 = | 0.4 ... 0.5 ms |
| | 10 = | approx. 0.1 ms |
| | 11 = | approx. 0.02 ms |

**Related topics**

- **Slot numbering of the riser card and of the JX6-I/O submodule** (see page 114)

# Combi module JX6-SV1

**Introduction**

This submodule features a potential-free relay contact and a counter input for incremental encoders or SSI absolute encoders.

**Register numbers**

In this chapter only the module register number is specified. To calculate the actually used register number, add the basic register number of the corresponding controller. The basic register number is made up of the riser card number and the number of the submodule on the riser card.

| 2 | 0 | S | J | 0 | 0 | z | z | z |
|---|---|---|---|---|---|---|---|---|

| Element | Purpose | Value range |
|---|---|---|
| **S** | Number of the riser card | 1 ... 2 |
| **J** | Number of the JX6-I/O submodule located on the riser card | 1 ... 2 |
| **zzz** | Module register number | 100 ... 999 |

**Registers - Overview**

The JX6-SV submodule can be programmed using a set of module registers. The function of module registers 103 to 110 depends on whether an incremental encoder or an absolute encoder has been configured:

| Register | Description |
|---|---|
| **MR 100** | Module ID |
| **MR 101** | Hardware configuration |
| **MR 102** | Reserve |
| **MR 103** | Strobe value/encoder value received |
| **MR 104** | Preset/pulse generator |
| **MR 105** | Count value |
| **MR 106** | Reference/offset |
| **MR 107** | Control register |
| **MR 108** | Status register |
| **MR 110** | Sampling timer |

**MR 100** **Module ID**

This register shows you, whether a JX6-SV1 has been connected.

| Module register properties | |
|---|---|
| Type of access | Read |
| Value after reset | 1 (JX6-SV1 plugged in) |

**MR 101** **Hardware configuration**

This register is for setting the module hardware.
Always write 0 to bits that have not been documented.

| Meaning of the individual bits | | |
|---|---|---|
| **Bit 0** | **Relay** | |
| | 0 = | disabled; pin 14 not connected with pin 15 |
| | 1 = | enabled; pin 14 is connected with pin 15 |
| **Bit 2** | **CHD signal on the counter chip** | |
| | 0 = | Counter zeroing/reference has been disabled |
| | 1 = | Counter zeroing/reference has been enabled |
| **Bit 5** | **Encoder type** | |
| | 0 = | Incremental encoder |
| | 1 = | SSI absolute encoder |

**MR 103**

## Incremental encoder: Strobe register

| Module register properties | |
|---|---|
| Values | 24 bits |
| Write | Writing the actual count value to this register |
| Read | Reading the count value written to the register last |

## Absolute encoder: Encoder value

This register holds the position value output by the SSI encoder.

Only if bit 0 is set in the status register (MR 108), there are valid values to be read.

| Module register properties | |
|---|---|
| Type of access | Read |
| Values | 32 bits |

**MR 104**

## Incremental encoder: Preset

When writing a value into register MR 105, this value is taken over as new count value.

| Module register properties | |
|---|---|
| Values | 24 bits |

## Absolute encoder: Clock generator

This register is for configuring the connected SSI encoder.

| Meaning of the individual bits | |
|---|---|
| **Bit 9 ... 0** | **Frequency of the SSI pulse signal**<br>= Module clock/(SSI clock * 2) - 1<br>Example:<br>Module clock = 8 MHz; SSI clock = 500 kHz<br>= 8,000,000 / (500,000 * 2) - 1 = 7 |
| **Bit 15 ... 10** | **Data width of the SSI encoder in bits**<br>= Pulse number * 2 + 1 |

Example:
Number of increments = 24
= 24 * 2 + 1 = 49

| Module register properties | |
|---|---|
| Values | 0 ... 65,535 |
| Example | 49 * 1,024 + 7 = 50,183 |

**MR 105**

## Incremental encoder: Count value

This register lets you access the present count value of the incremental encoder.

| Module register properties | |
|---|---|
| Values | 24 bits |
| Read | Present count value |
| Write | The value of MR 104 is taken over into this register. |

**MR 106**

## Reference/offset

This register can be used both as reference or offset register, dependent on the settings in the control register

| Module register properties | |
|---|---|
| Values | 24 bits |

**MR 107**

## Control register

This register lets you set the operating modes and input filters.

| Meaning of the individual bits | | |
|---|---|---|
| **Bit 0** | **Strobe** | |
| | 0 = | Strobe not allowed |
| | 1 = | Strobe allowed |
| **Bit 1** | **Reference strobe** | |
| | 0 = | Reference strobe not allowed |
| | 1 = | Reference strobe allowed (with /CHA * CHB * CHD) |
| **Bit 2** | **Comparison with reference value** | |
| | 0 = | Comparison is not being executed |
| | 1 = | The count value is compared with the reference value In case of equal values, bit 2 is set in MR 008- |
| **Bit 3** | **Counter zeroing** | |
| | 0 = | No counter zeroing |
| | 1 = | Once-only counter zeroing using CHC * CHD |
| **Bit 4** | **Repeated counter zeroing** | |
| | 0 = | No counter zeroing |
| | 1 = | Counter zeroing at each CHC * CHD |
| **Bit 5** | **Counting inputs** | |
| | 0 = | Counter inputs are active |
| | 1 = | Deactivating the counting inputs |
| **Bit 6** | **Set by reference** | |
| | 0 = | Set-by reference deactivated |
| | 1 = | The counter is loaded with the reference value with CHC * CHD |
| **Bit 9, 8** | **Pulse edge selection** | |
| | 00 = | Pulse-direction mode |
| | 01 = | Single evaluation |
| | 10 = | Dual evaluation |
| | 11 = | Four-fold evaluation |
| **Bit 10** | **Addition of offset** | |
| | 0 = | Addition disabled |
| | 1 = | Addition of an offset to a count and strobe value |
| **Bit 11** | **File format using an SSI encoder** | |
| | 0 = | Binary |
| | 1 = | Gray-format |
| **Bit 12** | **Parity using an SSI encoder** | |
| | 0 = | No parity bit |

| | | |
|---|---|---|
| | 1 = | Parity check |
| **Bit 13** | **Polarity of the parity bit** | |
| | 0 = | Even parity |
| | 1 = | Odd parity |
| **Bit 23 ... 14** | **Input filters** | |
| | Minimum signal duration until a signal is detected: $= (T_{sig} / T_{sys}) - 1$ | |
| | $T_{sig}$ | Minimum signal duration |
| | $T_{sys}$ | Period of a system cycle (125 ns at 8 MHz) |

| **Module register properties** | |
|---|---|
| Values | 24 bits |
| Value after reset | 0x300 |

**MR 108**

## Status register

After read access to this register, all bits, except for bit 3, are cleared.

| **Meaning of the individual bits** | | |
|---|---|---|
| **Bit 0** | **Strobe** | |
| | Incremental encoder: | |
| | 0 = | Strobe value not yet detected |
| | 1 = | Strobing value detected |
| | Absolute encoder: | |
| | 0 = | Invalid positioning value |
| | 1 = | Valid positioning value |
| **Bit 1** | **Reference strobe** | |
| | 1 = | Reference strobe detected |
| **Bit 2** | **Comparison with reference value** | |
| | 1 = | Count value and reference value agree with each other |
| **Bit 3** | **Counter zeroing** | |
| | 1 = | Counter zeroing has been executed |
| **Bit 4** | **Strobing value has been overwritten** | |
| | Incremental encoder: | |
| | 1 = | The value in the strobe register has been overwritten |
| | Absolute encoder: | |
| | 1 = | The position value has been overwritten |

| | | |
|---|---|---|
| **Bit 5** | **Lost reference value** | |
| | 1 = | A new reference value got lost, as the previous value has not been read out yet. |
| **Bit 6** | **Z track error** | |
| | 1 = | Z track errors or maximum travel speed exceeded or illegal Z track change |
| **Bit 7** | **Parity error** | |
| | 1 = | A parity error has occurred |

**MR 110**

## Sampling timer

This register value determines the intervals of reading the position value from the encoder. When the register content is "0", sampling will not be carried out. If you write a value other than "0" to the register, a sampling cycle is carried out when the set interval has elapsed.

| **Module register properties** | |
|---|---|
| Values | 0 ... 65,535 |
| Calculation | = Module frequency/(sampling frequency * 2) - 1 |
| Example | Module frequency = 8 MHz<br>Sampling frequency = 1 kHz<br>= 8,000,000 / (1,000 * 2) - 1 = 3,999 |

**Related topics**

- **Slot numbering of the riser card and of the JX6-I/O submodule** (see page 114)

## Sample program for local JX6-I/O submodules

**Task**

Cyclically read the position value of an absolute encoder with an SSI interface. Simultaneously physically establish moving lights

**Solution**

For reading the position value, use module JX6-SV1. Physically establish the moving lights by using module JX6-IO16CB.

**Configuration used in the sample program**

This example is based on the following configuration:

- The riser card is inserted in PCI slot 1 (S = 1)
- The JX6-IO16CB module is inserted in slot 1 of the riser card (J = 1).
- The JX6-SV1 module is inserted in slot 2 of the module board (J = 2).

**Programming**

- In the task *Main*, the modules JX6-SV1 and JX6-IO16CB are initialized, and the cyclic task is started.
- Depending on a digital input the interval is set, at which each output of the moving lights is active.
- In the *While* loop, the eight outputs for the moving lights are activated and deactivated successively.
- The cyclic task *Encode* stops the sampling timer and reads the position value output by the encoder last.
- Then, the timer is started with an interval of 0.8 ms. After this, the JX6-SV1 transmits another position value from the SSI encoder so that at the next startup of the cyclic task, the result is ready for being fetched.
- Finally, the position value is masked according to the set format.

**Software versions**

The sample program requires the software versions listed below:

- JetSym: Version 5.1.3 or higher
- Controller JC-970MC: OS version 1.07 or higher

You can also find the latest sample programs in the JetSym online help.

**JetSym STX program**

```
Type
    DIG_I8O8:
    Struct
        Inputs              : Int;
        Outputs             : Int;
        OutStatus           : Int;
        Control             : Int;
        FilterI1to4         : Int;
        FilterI5to8         : Int;
    End_Struct;

    JX6_SV1:
    Struct
        ModuleIdentification : Int;
        Config              : Int;
```

```
        DAC                 : Int;
        CntValue            : Int;
        CntBitClock         : Int;
        CntDummy            : Int;
        CntRef              : Int;
        CntControl          : Int;
        CntStatus           : Int;
        IntMask             : Int;
        Timer               : Int;
    End_Struct;
End_Type;

Const
    // Gray/bin conversion; 2MHz filter
    SSI_CONTROL         = 0x4800;
    // Delay 800 µs
    SSI_TIMER_1MS       = 3199;
End_Const;

Var
    DigitalIO           : DIG_I8O8    At %VL 201100100;
    Encoder             : JX6_SV1     At %VL 201200100;

    SSIdataMask         : Int;
    SSIsignBit          : Int;
    SSIsignExt          : Int;

    SSItimerValue       : Int;
    Position            : Int;

    OutputTime          : Int;
End_Var;


Function SetUpSSI(DataLen, Clock_kHz: Int): Int;
    Var
        i, j : Int;
    End_Var;

    // Configuration as an SSI encoder
    Encoder.Config := 32;

    i := 0xffffffff;
    j := 32 - DataLen;
    i >>= j;
    SSIdataMask := i;
    SSIsignExt := SSIdataMask Wxor 0xffffffff;
    SSIsignBit := DataLen - 1;

    // Data length of the SSI encoder in bits
```

```
    i := DataLen * 2 + 1;
    i <<= 10;

    // Frequency of the SSI pulse signal
    j := 4000 / Clock_kHz - 1;
    Encoder.CntBitClock := i Wor j;

    Encoder.CntControl := SSI_CONTROL;

    SSItimerValue := SSI_TIMER_1MS;

    // Start the timer
    Encoder.Timer := SSItimerValue;
End_Function;


Task Main Autorun
    Var
        OutMask:    Byte;
    End_Var;

    // 24-bit data length, 500 kHz
    SetUpSSI(24, 500);

    // Acknowledge the error state of the digital outputs
    DigitalIO.OutStatus := 0;

    TaskAllEnableCycle();

    Loop
        If BitSet(DigitalIO.Inputs, 7) Then
            OutputTime := T#200ms;
        Else
            OutputTime := T#100ms;
        End_If;

        OutMask := 1;
        While OutMask Do
            DigitalIO.Outputs := OutMask;
            Delay(OutputTime);
            OutMask <<= 1;
        End_While;
    End_Loop;
End_Task;


Task Encode    Cycle T#1ms
    // Stop the sampling timer of the SSI encoder
    Encoder.Timer := 0;
    // Read the position value from the SSI encoder
```

```
    Position := Encoder.CntValue;
    // Restart timer
    Encoder.Timer := SSItimerValue;

    // Mask the data bits
    Position := Position Wand SSIdataMask;
    // Is a sign-extension required?
    If BitSet(Position, SSIsignBit) Then
        // Sign extension to 32 bits
        Position := Position Wor SSIsignExt;
    End_If;
End_Task;
```

# 6.4 E-mail

**Introduction**

The user creates template files for e-mails. Into these templates, the controller JC-970MC can enter variables for sending, if required. The controller sends e-mails to an e-mail server which will then forward the message.

This chapter describes how to configure the e-mail feature in the JC-970MC, and how the controller both creates and sends e-mails.

**Activating the e-mail feature**

To activate the e-mail feature in the JC-970MC, the following requirement must be met:

- A valid e-mail configuration file exists while the controller is booting.

If this requirement has been met, the corresponding bit in the web status register is set and the e-mail feature is available.

**Required programmer's skills**

To be able to use the e-mail feature, the following skills are required:

- Since files are used to configure the e-mail feature, and e-mails as such are based on these files, the user must be familiar with the file system.
- The user must be familiar with IP networks.

**Content**

## 6.4.1 Configuring the E-mail feature

**Introduction**

This chapter gives a description on how to configure the e-mail feature so as to allow sending of e-mails from within the application program.

During the boot process, the JC-970MC reads out configuration data from the file **/EMAIL/email.ini**.

**Prerequisites**

For creating the configuration file, the following prerequisites must be fulfilled:

- The IP address of the e-mail server must be known.
- If the IP address of the e-mail server is not known, name resolution through a DNS server must be possible - refer to *Using names for IP addresses*.
- The log-on and authentication parameters at the e-mail server must be known.

To obtain this information contact your network administrator.

**Content**

## Structure of the configuration file

**Introduction**

The e-mail client is configured with the data from the **/EMAIL/email.ini** file. The JC-970MC reads the values during the boot process only.

**Structure of the configuration file**

This configuration file is a text file the entries of which are grouped into several sections.

- These sections are for entering values which are then used by the e-mail client.
- You can insert blank lines as required.
- The following characters precede a comment line: "!", "#" or ";".

**Enrypted sending of e-mails**

The JC-970MC supports the encrypted sending of e-mails.

To use this function, the **SMTP** section of the **email.ini** file must contain the SECUREEMAIL and STARTTLS entries. If these entries are missing, e-mails are sent unencrypted.

**Sections**

The configuration file contains up to three sections. Section [SMTP] is mandatory. The user does not need to create the other sections unless they are actually required.

| Section | Configuration values |
|---|---|
| [SMTP] | ■ IP address and port number of the SMTP server<br>■ Log-on parameters<br>■ With/without encryption |
| [POP3] | ■ IP address and port number of the POP3 server<br>■ Log-on parameters |
| [DEFAULT] | ■ Name of an e-mail template file containing default values |

## Section [SMTP]

**Introduction**

This section lets you specify the parameters for establishing a connection with the SMTP server.

**Example:**

```
[SMTP]
IP          = 192.168.40.1
PORT        = 25000
HELO        = JetControl_2
USER        = JetControl0815
PASSWORD    = MyPassWord
SECUREMAIL = 1
STARTTLS    = 0
```

**Authentication**

With this type of authentication, the JC-970MC performs a user logon to the SMTP server before sending an e-mail. During the logon process USER and PASSWORD must be entered. The JC-970MC supports the following authentication methods:

- LOGIN
- PLAIN
- CRAM-MD5

**Configuration values**

| **IP** | |
|---|---|
| In the given example | 192.168.40.1 |
| Description | IP address of the SMTP server;<br>can also be specified as name. |
| Valid values | ■ > 1.0.0.0<br>■ < 223.255.255.255 |
| Illegal values | ■ Network address<br>■ Broadcast address |
| In case of illegal value or missing entry | The e-mail feature is not available |

| **PORT** | |
|---|---|
| In the given example | 25,000 |
| Description | Port number of the SMTP server |
| Valid values | ■ > 0<br>■ < 65.536 |
| Illegal values | ■ > 65,335 |
| In case of missing entry | 25 |

| **HELO** | |
|---|---|
| In the given example | JetControl_2 |
| Description | Name for logging on at the e-mail server |
| Valid values | String of 63 characters max. |
| In case of missing entry | When sending the e-mail, the controller uses the entry contained in [FROM]. |

| **USER** | |
|---|---|
| In the given example | JetControl0815 |
| Description | Log-on name for SMTP authentication.<br>If this entry exists, the entry PASSWORD is required, too. |
| Valid values | String of 63 characters max. |
| In case of missing entry | SMTP authentication will not be carried out. |

| **PASSWORD** | |
|---|---|
| In the given example | MyPassWord |
| Description | Log-on password for SMTP authentication.<br>If this entry exists, the entry USER is required, too. |
| Valid values | String of 63 characters max. |
| In case of missing entry | SMTP authentication will not be carried out. |

| **SECUREMAIL** | |
|---|---|
| In the given example | 1 |
| Description | Encryption: If the value is 0 e-mails are sent unencrypted, if the value is 1 e-mails are sent encrypted. |
| Valid values | 1 or 0 |
| In case of missing entry | E-mails are sent unencrypted |

| **STARTTLS** | |
|---|---|
| In the given example | 0 |
| Description | Encryption of the communication setup<br>If STARTTLS is enabled (1), the connection to the server is established unencrypted. As soon as the connection is established, the e-mail is transmitted in encrypted form. If STARTTLS is disabled (0), establishing contact with the server is already encrypted. |
| Valid values | 1 or 0 |
| In case of missing entry | Connection setup to the server is already encrypted. |

# Section [POP3]

**Introduction**

This section lets you specify the parameters for establishing a connection with the POP3 server.

This section is required only in case the e-mail server requires authentication via POP3-before-SMTP.

**Example:**

```
[POP3]
IP       = 192.168.40.1
PORT     = 25100
USER     = JetControl4711
PASSWORD = Pop3PassWord
```

**Authentication**

This type of authentication requires the JC-970MC to log on at the POP3 server. During the logon process USER and PASSWORD must be entered. After that, the SMTP server allows e-mails to be sent for a given period of time (usually 10 to 30 minutes).

**Configuration values**

| **IP** | |
|---|---|
| In the given example | 192.168.40.1 |
| Description | IP address of POP3 server;<br>can also be specified as name. |
| Valid values | ■ > 1.0.0.0<br>■ < 223.255.255.255 |
| Illegal values | ■ Network address<br>■ Broadcast address |
| In case of illegal value or missing entry | POP3 log-on will not be carried out. |

| **PORT** | |
|---|---|
| In the given example | 25,100 |
| Description | Port number of POP3 server |
| Valid values | ■ > 0<br>■ < 65.536 |
| Illegal values | ■ > 65,335 |
| In case of missing entry | 110 |

| **USER** | |
|---|---|
| In the given example | JetControl4711 |
| Description | Log-on name for POP3 authentication.<br>If this entry exists, the entry PASSWORD is required, too. |
| Valid values | String of 63 characters max. |
| In case of missing entry | POP3 log-on will not be carried out. |

| **PASSWORD** | |
|---|---|
| In the given example | Pop3PassWord |
| Description | Log-on password for POP3 authentication.<br>If this entry exists, the entry USER is required, too. |
| Valid values | String of 63 characters max. |
| In case of missing entry | POP3 log-on will not be carried out. |

## Section [DEFAULT]

**Introduction**

This section lets you specify the name of an e-mail template file which contains default settings for e-mails. If a section is not available in a given e-mail template, JC-970MC applies these default settings for sending an e-mail message.

**Example**

```
[DEFAULT]
MAILCFG = EmailDefaults.cfg
```

**Related topics**

- **Structure of the template file** (see page 154)

# Configuration file - Examples

**Introduction**

This section contains several examples of the e-mail configuration file **/EMAIL/email.ini**.

**Minimum configuration**

If no authentication is required and the default value is assigned to the IP port of the SMTP server, the configuration file must contain only the IP address of the SMTP server.

```
[SMTP]
IP       = 192.168.40.1
```

**Authentication through POP3 Log-on**

In case the e-mail server requires previous log-on through POP3 and an e-mail template containing default setting has been defined:

```
[SMTP]
IP       = 192.168.40.1

[POP3]
IP       = 192.168.40.1
USER     = JetControl4711
PASSWORD = Pop3PassWord

[DEFAULT]
MAILCFG = EmailDefaults.cfg
```

**Authentification through SMTP**

In case the e-mail server requires an encrypted authentication:

```
[SMTP]
IP       = 192.168.40.1
USER     = JetControl0815
PASSWORD = MyPassWord
```

## 6.4.2 Creating e-mails

**Introduction**

This chapter describes how to create an e-mail. Then, the application program sends these e-mails.

For each e-mail the user has to create an e-mail template file.

**Content**

# Name of the e-mail template file

**Introduction**

This naming convention must only be kept to, if system function 110 is applied, which should not be used any further, though.

The STX feature `EMailSend()` lets you select any file name and directory, as long as the limitations owed to the file system are kept to.

The name of an e-mail template file consists of a fixed part of the name and a variable part. The variable part of the name allows the application program to choose various e-mails for sending.

**File name**

`email_#.cfg`

| Part of the name | Description |
|---|---|
| email_ | Name prefix which always remains fixed |
| # | Number of the e-mail; value between 0 and 255 |
| .cfg | Fixed file extension |

**Storage location**

E-mail template files must be stored to the same directory on the internal flash disk as is the configuration file.

`/EMAIL`

**Examples**

```
email_0.cfg
email_37.cfg
email_255.cfg
```

## Structure of the e-mail template file

**Introduction**

An e-mail template file is a text file which is divided into sections. For sending the e-mail, the JC-970MC compiles the information contained in these sections.

**E-mail template file**

- Sections [FROM] and [TO] are mandatory. This information may be specified either in the e-mail to be sent or in the e-mail template file containing the default settings.
- All parameters in these sections can be tagged with realtime controller values (refer to *Inserting realtime controller values*).

[FROM]
Sender

[TO]
Addressee

[CC]
Additional addressee(s)

[SUBJECT]
Subject

[ATTACHMENT]
Complete path and file name

[MESSAGE]
E-mail message text

**Sections**

| **[FROM]** | |
|---|---|
| Description | E-mail sender |
| Comments | Please check with your IT administrator which information must be entered here. |
| Length | 63 characters |
| Example | `[FROM]`<br>`JetControl@jetter.de` |

| **[TO]** | |
|---|---|
| Description | E-mail addressee |
| Comments | Several addressees are separated by the semicolon “;”. |
| Length | 255 characters |
| Example | `[TO]`<br>`service@mydomain.com` |

| **[CC]** | |
|---|---|
| Description | Additional e-mail addressee(s) |
| Comments | Several addressees are separated by the semicolon “;”. |
| Length | 255 characters |
| Example | `[CC]`<br>`service@mydomain.com;hotline@mydomain.com` |

| **[SUBJECT]** | |
|---|---|
| Description | Subject |
| Length | 255 characters |
| Example | `[SUBJECT]`<br>`Fatal Error` |

| **[ATTACHMENT]** | |
|---|---|
| Description | Complete name of the file to be attached |
| Comments | The attached file must be a text file. |
| Length | 511 characters |
| Example | `[ATTACHMENT]`<br>`/logfiles/error_report.log` |

| **[MESSAGE]** | |
|---|---|
| Description | E-mail message text |
| Comments | Text only message |
| Length | 65,535 characters |
| Example | `[MESSAGE]`<br>`Have a nice day !`<br>`JetControl.` |

# Inserting real-time controller values

**Introduction**

Actual real-time controller values are integrated into parameter entries within the sections via tag functions. This way, the contents respectively states of registers, text registers, inputs, outputs and flags can be displayed.

**Tag delimiters**

All tags start and end with defined strings (delimiters). Between these tag delimiters, the variables are defined.

| Delimiter | String |
|---|---|
| Tag start | <JC:DTAG |
| Tag end | /> |

**Variable definition**

The variable definition in a tag contains attributes which are used to set, for example, how the value of a variable is to be displayed:

| **name** | |
|---|---|
| Description | Variable name |
| Comments | Code letter followed by the variable number |
| Example | `name="R1000023"` |

| **type** | |
|---|---|
| Description | Variable type of notation |
| Example | `type="REAL"` |

| **format** | |
|---|---|
| Description | Representation format |
| Comments | Refer to format definition |
| Example | `format="+0####.###"` |

| **factor** | |
|---|---|
| Description | Factor by which the real-time controller value is multiplied |
| Comments | Multiplication is executed prior to adding the offset |
| Example | `factor="1.5"` |

| **offset** | |
|---|---|
| Description | Value which is added to the real-time controller value |
| Comments | Multiplication by the factor is executed prior to adding the value to the real-time controller value |
| Example | `offset="1000"` |

**Format definition**

You can define the representation of variables by means of their attribute.

- The number of digits/characters used for representing a variable can be defined by the character "#".
- Prefix "0" sets the output of leading zeros. This applies to the register types INT, INTX and REAL.
- Prefix "+" sets the output of a sign. This applies to the register types INT and REAL.
- Prefixing a blank sets the output of a blank. This applies to the register types INT and REAL.

**Registers/text registers**

The variable name begins with a capital "R" followed by the register number.
The following types are possible:

| Type | Notation |
|---|---|
| INT | Integer, decimal |
| INTX | Integer, hexadecimal |
| INTB | Integer, binary |
| BOOL | Register content = 0 --> Display: 0<br>Register content != 0 --> Display: 1 |
| REAL | Floating point, decimal |
| STRING | Text register |

Standard type: INT

**Example:**

```
<JC:DTAG name="R1000250" type="REAL" format="+0####.###" fac-
tor="3.25" offset="500" />
```

**Result:**

This instruction causes the contents of register 1000250 to be multiplied by 3.25. Then, 500 is added to the product. The result appears in the Web browser with sign and at least five integer positions before the decimal point. Leading zeros are added as appropriate. Furthermore, three decimal positions are added.

**Flags**

The variable name begins with a capital "F" followed by the flag number.
The following types are possible:

| Type | Notation |
|---|---|
| BOOL | Flag = 0 --> Display: 0<br>Flag = 1 --> Display: 1 |
| STRING | Flag = 0 --> Display: FALSE<br>Flag = 1 --> Display: TRUE |

Standard type: BOOL

**Example:**

```
<JC:DTAG name="F100" type="STRING" format="#" />
```

**Result:**
The state of flag 100 is displayed as string "T" or "F".

**Inputs**

The variable name begins with a capital "I" followed by the input number.
The following types are possible:

| Type | Notation |
|---|---|
| BOOL | Input = 0 --> Display: 0<br>Input = 1 --> Display: 1 |
| STRING | Input = 0 --> Display: OFF<br>Input = 1 --> Display: ON |

Standard type: BOOL

**Example:**

```
<JC:DTAG name="I201200308" type="STRING" />
```

**Result:**
The state of input 201200308 is displayed as "ON" or "OFF".

**Outputs**

The variable name begins with a capital "O" followed by the output number.
The following types are possible:

| Type | Notation |
|---|---|
| BOOL | Output = 0 --> Display: 0<br>Output = 1 --> Display: 1 |
| STRING | Output = 0 --> Display: OFF<br>Output = 1 --> Display: ON |

Standard type: BOOL

**Example:**

```
<JC:DTAG name="O201100308" />
```

**Result:**
The state of output 201100308 is inserted as "1" or "0".

**Access via pointer register**

Access via pointer register is realized by inserting the capital letter "P" in front of the variable name. In each case the value of the variable is displayed whose number corresponds to the content of the register specified in the variable name.

**Examples:**

```
<JC:DTAG name="PR1000300" />
```

Result: The content of the register is displayed whose number is contained in register 1000300.

```
<JC:DTAG name="PF1000300" />
```

Result: The state of the flag is displayed whose number is contained in register 1000300.

```
<JC:DTAG name="PI1000300" />
```

Result: The state of the input is displayed whose number is contained in register 1000300.

```
<JC:DTAG name="PO1000300" />
```

Result: The state of the output is displayed whose number is contained in register 1000300.

**Access via pointer register and offset**

To specify the number of the variable to be displayed, it is also possible to add a constant value or another register content to the pointer register value

**Examples:**

```
<JC:DTAG name="PR1000300 + 100" />
```

**Result:** The content of the register is displayed whose number results from the addition of the content of register 1000300 and value 100.

```
<JC:DTAG name="PR1000300 + R1000100" />
```

**Result:** The content of the register is displayed whose number results from the addition of the content of register 1000300 and the content of register 1000100.

```
<JC:DTAG name="PF1000300 + 100" />
```

**Result:** The state of the flag is displayed whose number results from the addition of the content of register 1000300 and value 100.

```
<JC:DTAG name="PF1000300 + R1000100" />
```

**Result:** The state of the flag is displayed whose number results from the addition of the content of register 1000300 and the content of register 1000100.

```
<JC:DTAG name="PI1000300 + 100" />
```

**Result:** The state of the input is displayed whose number results from the addition of the content of register 1000300 and the value 100.

```
<JC:DTAG name="PI1000300 + R1000100" />
```

**Result:** The state of the input is displayed whose number results from the addition of the content of register 1000300 and the content of register 1000100.

```
<JC:DTAG name="PO1000300 + 100" />
```

**Result:** The state of the output is displayed whose number results from the addition of the content of register 1000300 and the value 100.

```
<JC:DTAG name="PO1000300 + R1000100" />
```

**Result:** The state of the output is displayed whose number results from the addition of the content of register 1000300 and the content of register 1000100.

## 6.4.3 Sending an e-mail

**Introduction**

This chapter gives a description on how to send previously created e-mails from within the application program.

When sending an e-mail from the application program, the JC-970MC creates the e-mail based on the e-mail template file and inserts variable values if required.

**Processing within the application program**

Sending an e-mail may take considerable time. Therefore, other tasks of the application program are processed while an e-mail is being sent. Only one e-mail function call is possible at a time. While an e-mail of a task is being sent, all other tasks which invoke the e-mail function are therefore blocked until this operation is completed.

**System function 110**

As of JetSym 5.0, system function 110 is outdated. Instead, apply JetSym STX function `EMailSend()`.

**JetSym STX function EMailSend()**

The JetSym STX function EMailSend() is described in detail in the JetSym online help.

Declaration of functions:

```
Function EmailSend(Const Ref FileName: String): Int;
```

## 6.4.4 Registers

**Introduction**

This chapter gives a description of those registers from which you can query the status of e-mail processing.

**Content**

# Registers - Overview

**Introduction**

The JC-970MC provides several registers for querying the processing status of the e-mail function.

**Registers - Overview**

| Register | Description |
|---|---|
| **202930** | Web status |
| **292932** | IP address of SMTP server |
| **292933** | IP address of POP3 server |
| **292934** | Port number of SMTP server |
| **292935** | Port number of POP3 server |
| **292937** | Status of e-mail processing |
| **292938** | ID of the task that is just sending an e-mail |

## Registers - Description

**R 202930** **Web status**

The Web status register displays all available functions in bit-coded mode.

| Meaning of the individual bits | |
|---|---|
| **Bit 0** | **FTP server**<br>1 = available |
| **Bit 1** | **HTTP server**<br>1 = available |
| **Bit 2** | **E-mail**<br>1 = available<br>Bit 2 is set only if the configuration file **/EMAIL/email.ini** is available. |
| **Bit 3** | **Data file function**<br>1 = available |
| **Bit 4** | **Modbus/TCP**<br>1 = existing |
| **Bit 5** | **Modbus/TCP**<br>1 = available |
| **Bit 7** | **FTP client**<br>1 = available |

| Module register properties | |
|---|---|
| Type of access | Read |
| Value after reset | Depending on options purchased |

**R 292932** **IP address of the SMTP server;**

This register lets you read the IP address of the SMTP server in the same format as specified in the file **/EMAIL/email.ini**.

| Module register properties | |
|---|---|
| Type of access | Read |
| Value after reset | Depending on configuration |
| Takes effect | Once R 202930.2 = 1 |

**R 292933**

## IP address of POP3 server

This register lets you read the IP address of the POP3 server in the same format as specified in the file **/EMAIL/email.ini**.

| **Module register properties** | |
|---|---|
| Type of access | Read |
| Value after reset | Depending on configuration |
| Takes effect | Once R 202930.2 = 1 |

**R 292934**

## Port number of the SMTP server

This register lets you read the port number of the SMTP server in the same format as specified in the file **/EMAIL/email.ini**.

| **Module register properties** | |
|---|---|
| Type of access | Read |
| Value after reset | Depending on configuration |
| Takes effect | Once R 202930.2 = 1 |

**R 292935**

## Port number of POP3 server

This register lets you read the port number of the POP3 server in the same format as specified in the file **/EMAIL/email.ini**.

| **Module register properties** | |
|---|---|
| Type of access | Read |
| Value after reset | Depending on configuration |
| Takes effect | Once R 202930.2 = 1 |

**R 292937**

## Status of e-mail processing

This register lets you track the e-mail status.

| Module register properties | | |
|---|---|---|
| Values | 0 | No e-mail is being sent |
| | 1 | Parameters are being handed over to the e-mail client of the JC-970MC. |
| | 2 | E-mail is being compiled and connection with the server is being established. |
| | 3 | E-mail has been sent to the server. |
| Type of access | Read | |

**R 292938**

## Task ID (e-mail)

Shows the ID of the task that is just sending an e-mail

| Module register properties | | |
|---|---|---|
| Values | 0 ... 99 | Task ID |
| | 255 | There is no task sending an e-mail. |
| Value after reset | 255 | |
| Type of access | Read | |

# 6.5 Sorting data

| | |
|---|---|
| **Introduction** | This chapter describes system function 50. This system function is used to trigger the sorting algorithm provided by the operating system. |
| **Application** | For sorting data in controller registers by their value.<br><br>The sort algorithm is provided by the operating system of the controller. The data to be sorted are indirectly addressed through a descriptor using parameter 1. |
| **System function 50** | As of JetSym 5.0, system function 50 is outdated. Instead, apply JetSym STX function `QSort()`. |
| **JetSym STX function QSort()** | The JetSym STX function QSort() is described in detail in the JetSym online help.<br>Declaration of functions:<br>`Function QSort(DataPtr: Int, ElementCnt: Int, ElementSize: Int, SortOffset: Int, SortType: STXBASETYPE, SortMode: QSORTMODE): Int;` |

# 6.6 Modbus/TCP

**Introduction**

This chapter describes the functions of the Modbus/TCP server and client integrated into the JC-970MC.

**Required programmer's skills**

To be able to use the functions described in this chapter, the user must be familiar with the following:

- Modbus/TCP and supported commands
- IP networks

**Content**

## 6.6.1 Modbus/TCP server

**Introduction**

After successful start of the Modbus/TCP server, an external client can access registers, flags, inputs and outputs.

This chapter describes the addressing process and the commands supported by the Modbus/TCP server.

**Number of possible connections**

Four connections may be opened at the same time.

**Restriction**

Modbus/TCP only supports transmission of registers with a width of 16 bits. From this follows, that only the 16 least significant bits are transmitted when 32-bit registers are sent.

When assigning incoming register values to the internal 32-bit registers no sign extension will be carried out.

**Content**

## Addressing

**Introduction**

The addresses which have been received via Modbus/TCP can be modified locally in the Modbus-TCP server. For this purpose, three registers have been provided. The basic addresses for accessing registers, inputs and outputs are entered into these registers. Then, the address contained in the Modbus/TCP frame specifies the address with reference to the basic address.

**R 272702**

**Register offset**

The basic address for accessing registers via Modbus/TCP is entered into R 272702.

| Register properties | |
|---|---|
| Value after reset | 1000000 |

**R 272704**

**Input offset**

The basic address for accessing inputs via Modbus/TCP is entered into R 272704.

| Register properties | |
|---|---|
| Value after reset | 100000000 |

**R 272705**

**Output offset**

The basic address for accessing outputs via Modbus/TCP is entered into R 272705.

| Register properties | |
|---|---|
| Value after reset | 100000000 |

**Example 1**

The Modbus/TCP server on the JetControl receives from a Modbus/TCP client the command **read multiple registers** starting from register number 100. The number of registers to be read is 5. Register 272702 *Register Offset* contains the value 1000000.

Hence, registers 1000100 through 1000104 will be read.

**Example 2**

The Modbus/TCP server on the JetControl receives from a Modbus/TCP client the command **read input discretes** specifying input number 210 and the instruction to read this input. Register 272704 *Input Offset* contains the value 100000000.

| | |
|---|---|
| | Hence, input 100000210 of a peripheral module, such as JX3-DI16, will be read. |
| **Example 3** | The Modbus/TCP server on the JetControl receives from a Modbus/TCP client the command **write coils** specifying output number 205 and the instruction to set this output. Register 272705 *Output Offset* contains value 100000000.<br>Hence, output 100000205 of a peripheral module JX3-DO16 will be set. |

## Supported commands - Class 0

| | |
|---|---|
| **fc 3** | **read multiple registers** |
| | Reading register sets<br><br>The starting register number within the JC-970MC is calculated as follows: Register number specified in the command plus the content of R 272702 *Register Offset*. |
| **fc 16** | **write multiple registers** |
| | Writing register sets<br><br>The starting register number within JC-970MC is calculated as follows: Register number specified in the command plus the content of R 272702 *Register Offset*. |

## Supported commands - Class 1

| | |
|---|---|
| **fc 1** | **read coils** |
| | Reading outputs<br><br>The output number within the JC-970MC is calculated as follows: Output number specified in the command plus the content of register 272705 *Output Offset*. |
| **fc 2** | **read input discretes** |
| | Reading inputs<br><br>The input number within the JC-970MC is calculated as follows: Input number specified in the command plus the content of register 272704 *Input Offset*. |
| **fc 4** | **read input registers** |
| | Reading inputs blockwise in 16-bit words.<br><br>The starting register number within the JC-970MC is calculated as follows: Register number specified in the command plus the content of R 272702 *Register Offset*. |
| **fc 5** | **write coil** |
| | Enabling/disabling an individual output<br><br>The output number within the JC-970MC is calculated as follows: Output number specified in the command plus the content of register 272705 *Output Offset*. |
| **fc 6** | **write single register** |
| | Entering values into the 16 least significant bits of a register<br><br>The starting register number within the JC-970MC is calculated as follows: Register number specified in the command plus the content of R 272702 *Register Offset*. |

## Supported commands - Class 2

| | |
|---|---|
| **fc 15** | **force multiple coils** |
| | Enabling/disabling several outputs<br><br>The output number within the JC-970MC is calculated as follows: Output number specified in the command plus the content of register 272705 *Output Offset*. |
| **fc 23** | **read/write registers** |
| | Reading/writing registers simultaneously<br><br>The starting register number within the JC-970MC is calculated as follows: Register number specified in the command plus the content of R 272702 *Register Offset*. |

## 6.6.2 Modbus/TCP client

**Introduction**

The Modbus/TCP client included in JC-970MC supports only Class 0 Conformance.

This class uses commands for reading and writing multiple registers. Up to 125 registers with a width of 16 bits can be transmitted in one frame.

As protocol ID "0" is used. Sent and received frames are associated by means of the transaction ID.

This chapter describes how to carry out noncyclical or cyclical transmission to a Modbus/TCP server using system functions.

**Number of possible connections**

Connections to eleven different Modbus/TCP servers may be opened at the same time.

**Noncyclical data transmission**

System functions 65 and 67 *reading registers*, as well as 66 and 68 *writing registers* are used to establish a acyclical transmission channel to a Modbus/TCP server.

These system functions establish a connection to the specified Modbus/TCP server, transmit the desired data and clear down the connection.

If RemoteScan has already established a connection for cyclical data transmission, this connection will be used. Setting-up and clearing-down the connection is, therefore, not required.

**Cyclical data transmission**

Cyclical data transmission is made through the configurable function `RemoteScan`. The inputs and outputs 20001 through 36000 that are combined in the 16-bit registers 278000 through 278999 are cyclically transmitted from and to the Modbus/TCP servers.

To each Modbus/TCP server (IP address and port) a connection is established. The connection is established independent of how many communication units are configured on this server.

If several communication units are configured on one Modbus/TCP server, accesses are serialized since servers often do not support **command pipelining**. If several servers have been configured, communication is carried out in parallel.

**Combined inputs and outputs**

| Register | Inputs and outputs |
|---|---|
| 278000 | 20001 ... 20016 |
| 278001 | 20017 ... 20032 |
| 278002 | 20033 ... 20048 |
| ... | ... |
| 278999 | 35985 ... 36000 |

These registers and the inputs and outputs mapped to them are merely storage cells within the RAM. There is no direct mapping to the hardware. Therefore, it is not defined whether inputs or outputs are mapped to a register. Assignment is made not until configuration in the communication units takes place.

**Unit ID**

The instruction header of a Modbus/TCP frame contains a *Unit ID*. The Unit ID

is not evaluated by Modbus/TCP devices, as they can be addressed without ambiguity by their IP address. Therefore, in the case of system functions 65, 66 and 80 always value "1" is sent.

Converters from Modbus/TCP to Modbus RTU use the *Unit ID* for addressing the Modbus RTU servers. Therefore, the corresponding system functions for reading and writing registers (system functions 67 and 68), as well as for initializing RemoteScan (system function 85) have been provided. These system functions can be used to set the Unit ID.

**Restriction**

Modbus/TCP only supports transmission of registers with a width of 16 bits. From this follows, that only the 16 least significant bits are transmitted when 32-bit registers are sent.

When assigning incoming register values to the internal 32-bit registers no sign extension will be carried out.

**Outdated system functions**

As of JetSym 5.0, the system functions are outdated. Instead, apply the corresponding JetSym STX functions.

**JetSym STX functions**

This is a comparison between the system functions and the corresponding JetSym STX functions.

| System functions | Corresponding JetSym STX function |
|---|---|
| 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; |

## 6.6.3 Modbus/TCP client with STX variables

**Introduction**

The Modbus/TCP client included in JC-970MC supports only Class 0 conformance.

This class uses commands for reading and writing multiple registers. One frame transmits up to 125 registers of 16 bits width.

As protocol ID, "0" is used. Assignment of transmitted and received frames is carried out using the transaction ID.

This chapter describes how to carry out noncyclical or cyclical transmission to a Modbus/TCP server using STX functions.

**Number of possible connections**

Connections to eleven different Modbus/TCP servers may be opened at the same time.

**Noncyclical data transmission**

Functions `ModbusReadReg()` and `ModbusWriteReg()` are used to establish a noncyclical transmission channel to a Modbus/TCP server.

These functions copy data between registers of a Modbus/TCP server and STX variables. They establish a connection to the specified Modbus/TCP server, transmit the desired data and clear down the connection again.

If RemoteScan has already established a connection for cyclical data transmission, this connection will be used. Setting-up and clearing-down the connection is, therefore, not required.

**Cyclical data transmission**

Cyclical data transmission is made through the configurable function `RemoteScanConfig()`. The data are cyclically transmitted from and to the Modbus/TCP servers by means of STX variables.

To each Modbus/TCP server (IP address and port) a connection is established. The connection is established independent of how many communication units are configured on this server.

If several communication units are configured on one Modbus/TCP server, accesses are serialized since servers often do not support **command pipelining**. If several servers have been configured, communication is carried out in parallel.

**Unit ID**

Converters from Modbus/TCP to Modbus RTU use the *Unit ID* for addressing the Modbus RTU servers. For this reason, the Unit ID can be set.

**JetSym STX functions**

The JetSym STX functions are described in detail in the JetSym online help.

| System functions | Corresponding JetSym STX function |
|---|---|
| 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; |

# 7 AutoCopy - Automatic copying of controller data

**Introduction**

This chapter describes the AutoCopy function which allows to copy data within the controller and/or between controller, the connected expansion modules and a controller within the network. 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 automatically processed by the controller during the boot process.

A detailed description of the AutoCopy function with all commands and examples can be found in the application-oriented manual „File System“ at www.jetter.de => Downloads=> Product independent Documentation => Application-oriented Manuals

**Features**

AutoCopy performs the following functions:

- Storing registers and flags to a file
- Restoring registers and flags from a file
- Creating directories
- Deleting directories
- Copying files
- Deleting files

**Areas of application**

Basically, AutoCopy is used in the following scenarios:

- Where remote maintenance is not possible
- Where there is no PC on site
- Where the operator is not able or should not be allowed to make modifications to the plant

The AutoCopy function lets you:

- Modify the application program
- Modify the application data
- Modify the controller configuration
- Update the operating system (of the controller and modules on the system bus)

**Prerequisites**

For automatic copying of controller data, the following prerequisites must be fulfilled:

- The programmer must be familiar with the file system.
- In the configuration file **config.ini** of the controller, the path and file name of the command file in the section *[FILES]* must be entered in **AutoCopyIni**. For access to the configuration file, at least system rights are required.
- The command file and other required files are available.

**config.ini - Example**

This is an example of a configuration file **config.ini** with an entry *AutoCopyIni*:

```
;JC-970MC System Configuration
;Copyright (c) 2009 by Jetter AG, Ludwigsburg, Germany

[IP]
Address     = 192.168.10.209
SubnetMask  = 255.255.255.0
DefGateway  =   0.  0.  0.  0
DNSServer   = 192.168. 10.244

[HOSTNAME]
SuffixType = 0
Name        = JetControl970MC

[PORTS]
JetIPBase   = 50000
JVMDebug    = 52000

[FILES]
AutoCopyIni = /USB1/autocopy.ini
```

**Naming convention**

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

# 7.1 AutoCopy - How it works

**Introduction**

This chapter describes how to launch and execute the AutoCopy function.

**Content**

## Launching the AutoCopy function

**Introduction**

The AutoCopy function can only be launched when the controller is booting (i.e. after startup).

**Prerequisites**

You have created the command file and stored it to the respective directory.

| | Value | Remarks |
|---|---|---|
| File name | autocopy.ini | All lower case letters |
| Directory | /USB1/ | Root directory on the USB flash drive |

**Launching the AutoCopy feature**

To launch the AutoCopy function, proceed as follows:

| Step | Action |
|---|---|
| **1** | Disconnect the controller from the power supply. |
| **2** | Set the mode selector S11 to *LOAD* position. |
| **3** | Switch the controller on. |
| **4** | Wait for the amber LED **GP2 (D1)** to be lit and for the green LED **GP1 (RUN)** to flash slowly at approximately 1 Hz. |
| ⇨ | **Result:** The controller executes the AutoCopy function. |
| **5** | Wait for the amber LED **GP2 (D1)** to be lit and for the green LED **GP1 (RUN)** to flash fast at approximately 4 Hz. |
| ⇨ | **Result:** The AutoCopy process is completed. |

# Executing AutoCopy commands

**Introduction**

During the boot process in AutoCopy mode the controller executes the commands contained in the command file.

**Restrictions**

In AutoCopy mode the following restrictions of controller functions apply:

- The controller does not execute the application program.
- There is no communication with the controller.
- When the AutoCopy function is completed the controller must be restarted.

**AutoCopy function - Steps**

The AutoCopy process involves the following steps:

| Step | Description |
|---|---|
| **1** | The controller opens the file **/USB1/autocopy.ini** that is specified by the entry **AutoCopyIni** in the configuration file **/System/config.ini**. |
| **2** | The controller reads the values from section *[OPTIONS]*. |
| **3** | The controller reads the command and its parameters from section *[COMMAND_1]*, processes it and writes the results, if any, into the log file. |
| **4 ... n** | The controller processes the other commands in ascending order up to the number given in section *[OPTIONS]*. |
| **n+1** | The controller calculates the statistic values for all command results and writes them into the log file. |

**LEDs of the JC-970MC in AutoCopy mode**

During the boot process of the controller, the OS status LEDs indicate the following:

| Step | Description | | | |
|---|---|---|---|---|
| **1** | **GP3** | **GP2** | **GP1** | **State** |
| | ON | OFF | OFF | Reset |
| **2** | **GP3** | **GP2** | **GP1** | **State** |
| | OFF | ON | OFF | Initialization |
| **3** | **GP3** | **GP2** | **GP1** | **State** |
| | OFF | ON | 1Hz | The AutoCopy function is executed. |
| **4a** | **GP3** | **GP2** | **GP1** | **State** |
| | OFF | ON | 4Hz | AutoCopy function is completed; no errors occurred |
| **4b** | **GP3** | **GP2** | **GP1** | **State** |
| | ON | ON | 4Hz | AutoCopy function is completed; errors occurred |

## Terminating AutoCopy mode

**Introduction**

Only a restart of the controller terminates the Autocopy mode.

**Prerequisite**

Processing the AutoCopy command is completed.

**Terminating AutoCopy mode**

To exit the AutoCopy mode, proceed as follows:

| Step | Action |
|---|---|
| **1** | Disconnect the controller from the power supply. |
| **2** | Set the mode selector to *RUN* or *STOP* position. |
| **3** | Switch the controller on. |

**Result:** The controller relaunches.

# 8 Operating system update

**Introduction**

Jetter AG are continuously striving to enhance the operating systems for their controllers and peripheral modules. Enhancing means adding new features, upgrading existing functions and fixing bugs.

This chapter describes how to perform an operating system update for a system equipped with a JC-970MC.

**Downloading an operating system**

You can download operating systems from the Jetter AG **homepage-** http://www.jetter.de. OS files are available for download in the downloads area of the respective product.

**JC-9xx system - Devices**

Operating systems can be updated on the following devices that are part of a system with the JC-970MC controller:

- Controller JC-970MC
- JX2 slave modules on a JX2 system bus
- Bus nodes on a JX2 system bus
- Bus nodes on a JX3 system bus
- Analog modules on a JX3 system bus
- Digital inputs and outputs on a JX3 system bus

**Content**

# 8.1 Updating the operating system of the controller

**Introduction**

This chapter describes how to update the OS of the JC-970MC. There are several options to transfer the OS file to the controller:

- From within the programming tool JetSym
- Via FTP connection
- From the application program

**Content**

# OS update by means of JetSym

**Introduction**

The programming tool JetSym offers an easy way to transfer an OS file to a controller.

**Prerequisites**

- A new OS file for the JC-970MC must be available.
- A UDP/IP and a TCP/IP connection between programming tool and JC-970MC must be possible.
  The number of the IP port must be set in the configuration memory as IP basic port number for JetIP communication.
- The operating system of the JC-970MC must be running.
  **Recommendation:**
  For this reason make sure that during this time the controller is not disconnected from the power supply. Make sure it remains energized.

**Updating the OS**

To update the OS, proceed as follows:

| Step | Action |
|---|---|
| 1 | Select in the JetSym menu **Build** menu item **Update OS...**. Alternative: In the **Advanced Configuration** dialog of the Hardware Manager, click on the button **Update OS**.<br>**Result:** The file selection dialog opens. |
| 2 | Select the new OS file here.<br>**Result:** In JetSym, a confirmation dialog opens. |
| 3 | Launch the OS upload by clicking the button **Yes**. |
| 4 | Wait until the update process is completed. |
| 5 | To activate the transferred OS, re-boot the controller. |

## Updating the operating system by means of FTP

**Introduction**

Using an FTP client an OS file can be transferred to the JC-970MC.

**Prerequisites**

- A new OS file for the JC-970MC must be available.
- An FTP connection to the controller must be possible.
- The login parameters for a user with administrator or system rights must be at hand.
- The operating system of the JC-970MC must be running. Therefore, make sure it remains energized.

**Updating the OS**

To update the OS, proceed as follows:

| Step | Action |
|---|---|
| **1** | Open an FTP connection to the JC-970MC. |
| **2** | Log in with administrator or system rights |
| **3** | Navigate to the directory */System/OS*. |
| **4** | Transfer the OS file. |
| **5** | Wait until the transfer process is completed. |
| **6** | Clear the FTP connection. |
| **7** | To activate the transferred OS, re-boot the controller. |

## Automatic OS update from USB flash drive

**Reference:**

An automatic OS update of the controller from the USB flash drive can be carried out using the AutoCopy function. For a detailed description, turn to chapter *AutoCopy* (see page 179).

## Updating the OS from within the application program

**Introduction**

The file functions included in the STX language let you carry out a program-controlled OS update of a JC-970MC from within an OS file.

**Prerequisites**

- An OS file must be available in the file system of the JC-970MC.
- The operating system of the JC-970MC and the application program must be running.
  **Note:**
  Make sure it remains energized.

**Updating the OS**

To start an OS update out of the application program, proceed as follows:

| Step | Action |
|---|---|
| 1 | Copy the OS file to a file of any name and of the extension ***.os** in the directory */System/OS*. |
| 2 | To activate the updated OS, for example by writing to the system command register, re-boot the controller. |

**JetSym STX program**

```
Var
    SourceName:             String;
    DestinationName:        String;
    UpdateIt:               Bool;
End_Var;

//*****************************************************
// Name:      OSupdate
// 1. Activate 'Tracing' in JetSym
// 2. Set the name of the source file in 'SourceName'
// 3. Set the flag 'UpdateIt'
//*****************************************************

Task OSupdate Autorun
    Var
        ResCopy:    Int;
    End_Var;

    DestinationName := '/System/OS/OperatingSystem.os';
    Loop
        UpdateIt := False;
        When UpdateIt Continue;
        ResCopy := FileCopy(SourceName,
                            DestinationName);
        Trace('Result : ' + IntToStr(ResCopy) + '$n');
    End_Loop;
End_Task;
```

# 8.2 Updating the operating system of a module

**Introduction**

This chapter describes how an OS update of a module connected to the system bus of the JC-970MC is carried out. There are several options to transfer the OS file to the module:

- From within the programming tool JetSym
- Via FTP connection
- From a USB flash drive
- From the application program

**Content**

## OS update by means of JetSym

**Introduction**

The programming tool JetSym offers an easy way to transfer an OS file to a module connected to the system bus of the controller.

**Prerequisites**

- A new OS file for the JC-970MC must be available.
- A UDP/IP and a TCP/IP connection between programming tool and IP port of the controller is possible (timeout 4,000 ms min.).
  The number of the controller in the configuration memory is set as IP basic port number for JetIP communication.
- The operating system is running.
- The system bus has been put into operation by the controller without any errors.
- The boot loader or the operating system of the module is running.
- The controller must remain energized during the OS update process.

**Updating the OS**

To update the OS, proceed as follows:

| Step | Action | |
|---|---|---|
| 1 | In JetSym, click on the menu **Build** and select item **Update OS...**, or click in the **Advanced Configuration** window of the Hardware Manager on **OS Update**.<br>**Result:** The file selection dialog opens. | |
| 2 | Select the new OS file here.<br>**Result:** In JetSym, a confirmation dialog opens. | |
| 3 | Confirm by clicking **Yes.**<br>**Result:** JetSym opens an input box for entering the interface type and module number. | |
| 4 | Enter the number of the riser card (1), the number of the submodule (1 or 2) and the module number (2 ... 63).<br>Launch the OS upload by clicking the button **Update**. | |
| 5 | Wait until the update process is completed. | |
| 6 | **If ...** | **... then ...** |
| | ... you wish to update other modules, ... | ... proceed with step 1. |
| | ... do not you wish to update other modules, ... | ... reboot the controller to launch the new operating system. |

## Updating the operating system by means of FTP

**Introduction**

Using an FTP client an OS file can be transferred to a module connected to the system bus of the controller.

**Prerequisites**

- An new OS file for the module must be available.
- An FTP connection to the controller must be possible.
- The login parameters for a user with administrator or system rights must be at hand.
- The operating system is running.
- The system bus has been put into operation by the controller without any errors.
- The boot loader or the operating system of the module is running.
- The controller must remain energized during the OS update process.

**Updating the OS**

To update the OS, proceed as follows:

| Step | Action | |
|---|---|---|
| **1** | Establish an FTP connection to the controller | |
| **2** | Log in with administrator or system rights | |
| **3** | Navigate to the OS directory of the module<br>Example: */System/PCI-Slot1/SB-Module1/JX2-Slave02/OS* or */System/PCI-Slot1/SB-Module2/JX3-Module05/OS* | |
| **4** | Transfer the OS file into this directory. | |
| **5** | Wait until the update process is completed. | |
| **6** | **If ...** | **... then ...** |
| | ... you wish to update other modules, ... | ... proceed with step 3. |
| | ... do not you wish to update other modules, ... | ... clear the FTP connection. Then reboot the controller to launch the new operating system. |

## Automatic OS update from USB flash drive

**Reference:** An automatic OS update of a JX2 or JX3 module from the USB flash drive can be carried out using the AutoCopy function. For a detailed description, turn to chapter *AutoCopy* (see page 179).

# Updating the OS from within the application program

**Introduction**

The file functions included in the STX language allow transferring an OS file to a module on the JX2 or JX3 system bus of the controller.

**Prerequisites**

- The OS file for the module is available in the file system of the controller.
- The operating system of the controller and the application program are running.
- The system bus has been put into operation by the controller without any errors.
- The boot loader or the operating system of the module is running.
- The controller must remain energized during the OS update process.

**Updating the OS**

To update the OS of a module, proceed as follows:

| Step | Action | |
|---|---|---|
| **1** | Open the OS file in read-only mode. | |
| **2** | Open a file with any name and the extension ***.os** in the OS directory of the module in write mode.<br>Example: */System/PCI-Slot1/SB-Module1/JX2-Slave02/OS/system.os* or */System/PCI-Slot1/SB-Module2JX3-Module05/OS/Any.os* | |
| **3** | Read the data out of the OS file. | |
| **4** | Write these data to the target file for the module. | |
| **5** | Close both files. | |
| **6** | **If ...** | **... then ...** |
| | ... you wish to update other modules, ... | ... proceed with step 1. |
| | ... do not you wish to update other modules, ... | ... reboot the controller to launch the new operating system. |

**Software versions**

The sample program requires the software versions listed below:

- JetSym: Version 5.1.3 or higher
- Controller JC-970MC: OS version 1.07 or higher

You can also find the latest sample programs in the JetSym online help.

**Sample program**

```
Var
    SourceName:             String;
    DestinationName:        String;
    UpdateIt:               Bool;
End_Var;
```

```
//*****************************************************
// Name:        FileCopy
//! \param[in]  SrcName        name of source file
//! \param[in]  DstName        name of destination file
//! \return     >= 0           size of source file
//! \return     < 0            error
//! \brief      copies a file
//*****************************************************
Function FileCopy(Ref SrcName: String,
                  Ref DstName: String):Int;
    Var
        SrcFile, DstFile:    File;
        FileBuffer:          Array[1000] Of Byte;
        Result:              Int;
        ReadSize:            Int;
        WriteSize:           Int;
        FileSize:            Int;
    End_Var;

    Result := 0;
    FileSize := 0;
    // open source file for reading
    If FileOpen(SrcFile, SrcName, 'r') Then
        // open destination file for writing
        If FileOpen(DstFile, DstName, 'w') Then
            // read first block of data
            ReadSize := FileRead(SrcFile,
                                 FileBuffer,
                                 SizeOf(FileBuffer));
            While ReadSize <> 0 Do
                // write read data to destination file
                WriteSize := FileWrite(DstFile,
                                       FileBuffer,
                                       ReadSize);
                If WriteSize <> ReadSize Then
                    // write error
                    Result := -3;
                    Exit;
                End_If;
                Inc(FileSize, WriteSize);
                // read next block of data
                ReadSize := FileRead(SrcFile,
                                     FileBuffer,
                                     SizeOf(FileBuffer));
            End_While;
            // close both files
            FileClose(SrcFile);
            FileClose(DstFile);
```

```
            Else
                // can't open destination file
                FileClose(SrcFile);
                Result := -2;
            End_If;
        Else
            // can't open source file
            Result := -1;
        End_If;
        If Result < 0 Then
            FileCopy := Result;
        Else
            FileCopy := FileSize;
        End_If;
    End_Function;

    //*****************************************************
    // 1. Enable Tracing in JetSym
    // 2. Put source file name into 'SourceName'
    // 3. Put destination file name into 'DestinationName'
    // 4. Set flag 'UpdateIt'
    //*****************************************************
    Task OSupdate Autorun
        Var
            ResCopy:    Int;
        End_Var;

        Loop
            UpdateIt := False;
            When UpdateIt Continue;
            ResCopy := FileCopy(SourceName,
                                DestinationName);
            Trace('Result : ' + IntToStr(ResCopy) + '$n');
        End_Loop;
    End_Task;
```

## 8.3 Updating the Windows® partition

| | |
|---|---|
| **Off-the-shelf Windows system** | The Windows section of the JC-975MC works like a normal Windows installation. |
| **Windows updates** | You can use the standard Windows on-board tools to perform updates. |
| **Backup/Restore** | You can backup and restore the Windows system partition (C:) and data partition (D:) using Windows on-board tools. |

# 9 Motion control

**Programming**

Please refer to the JetSym help for a description on functions and programming of the Motion Control.

The Motion Control is programmed in the Motion API interface in the programming language STX.

# 10 Quick reference - JC-970MC

## Corresponding OS version

This quick reference summarizes in brief the registers and flags of the JC-970MC with OS version 1.07.4.08

## Internal IP address of the controller section

IP address 192.168.2.3 lets you access the controller section from the Windows IPC.

## General overview - Registers

| | |
|---|---|
| 100000 ... 100999 | Electronic Data Sheet (EDS) |
| 101000 … 101999 | Configuration |
| 102000 … 102999 | Real-time clock |
| 104000 ... 104999 | Ethernet |
| 108000 ... 108999 | CPU |
| 200000 ... 209999 | General system registers |
| 210000 ... 219999 | Application program |
| 230000 ... 239999 | Networking via JetIP |
| 240000 ... 249999 | JetSync |
| 250000 ... 259999 | Ethernet system bus |
| 260000 ... 269999 | RemoteScan |
| 270000 ... 279999 | Modbus/TCP |
| 290000 ... 299999 | E-mail |
| 310000 ... 319999 | File system/data files |
| 320000 ... 329999 | FTP client |
| 350000 ... 359999 | User-programmable IP interface |
| 380000 ... 389999 | Error history |
| 390000 ... 399999 | I/O networking |
| 470000 ... 470999 | NetConsistency: Basic drivers |
| 471000 ... 471999 | NetConsistency: First instance |
| 1000000 ... 1119999 | Application registers (remanent; Int/Float) |
| 20SJ00000 ... 20SJ19999 | JX2 modules; JetMove 2xx<br>JX6 submodules |
| 50000000 ... 59999999 | Motion control |
| | Networking via Jetter Ethernet system bus<br>GNN: nnn = 000 ... 199: |
| 1nnn020000 ...<br>1nnn179999 | JX3 module registers |
| 1nnn202000 ...<br>1nnn227999 | JX2 module registers |
| 1nnn810000 ...<br>1nnn819999 | JetMove registers |
| 1nnn980000 ...<br>1nnn980199 | Indirect access via local R 236xxx |
| 1nnn990000 ...<br>1nnn999999 | Indirect access with variable destination window |

## I/Os - General overview

| | |
|---|---|
| 20001 ... 36000 | Virtual I/Os for RemoteScan |
| 20SJ0xx01 ... 20SJ0xx16 | JX2 modules (xx: 02 ... 32);<br>JX3 modules via JX3-BN-CAN<br>(xx: 02 ... 17) |
| 1nnn01xx01 ...<br>1nnn01xx16 | JX3 modules via JX3-BN-ETH<br>GNN: nnn = 000 ... 199:<br>xx: 02 ... 17 |

## Electronic Data Sheet (EDS)

| | |
|---|---|
| **[Identification]** | |
| 100600 | Internal version number |
| 100601 | Module ID |
| 100602 ... | Module name (register string) |
| 100612 | |
| 100613 | PCB revision |
| 100614 | PCB options |
| **[Production]** | |
| 100700 | Internal version number |
| 100701 ... | Serial number (register string) |
| 100707 | |
| 100708 | Day |
| 100709 | Month |
| 100710 | Year |
| 100711 | TestNum |
| 100712 | TestRev |
| **[Features]** | **JC-9xx** |
| 100800 | Internal version number |
| 100804 | Switch |
| 100805 | STX |
| 100806 | Non-volatile registers |
| 100810 | Motion Control |
| 100812 | HTTP/e-mail |
| 100813 | Modbus/TCP |
| 100817 | RTC |

## Configuration

| | |
|---|---|
| | **From the /System/config.ini file** |
| 101100 | IP address |
| 101101 | Subnet mask |
| 101102 | Default gateway |
| 101103 | DNS server |
| 101132 | HOSTNAME suffix type |
| 101133 ... | HOSTNAME (register string) |
| 101151 | |
| 101164 | JetIP port number |
| 101165 | Port number for STX debugger |
| | **Used by the system** |
| 101200 | ETH (X102): IP address |
| 101201 | ETH (X102): Subnet mask |
| 101202 | ETH (X102): Default gateway |
| 101203 | DNS server |
| 101216 | ETH (X102): Static route of the IP address |
| 101217 | ETH (X102): Static route of the subnet mask |
| 101218 | ETH (X102): Static route of the gateway |
| 101232 | HOSTNAME suffix type |
| 101233 … | HOSTNAME (register string) |
| 101251 | |
| 101264 | JetIP port number |
| 101265 | Port number for STX debugger |
| 101299 | Write trigger (0x77566152) |

## Real-time clock

| | |
|---|---|
| | **Direct access** |
| 102910 | Milliseconds |
| 102911 | Seconds |
| 102912 | Minutes |
| 102913 | Hours |
| 102914 | Day of the week (0 = Sunday) |
| 102915 | Day |
| 102916 | Month |
| 102917 | Year |
| | **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 |

## Ethernet

**Link status**

104002 Link status ETH1

| | |
|---|---|
| 0 | No link |
| 10 | 10 MBit/s, half-duplex |
| 20 | 10 MBit/s, full duplex |
| 100 | 100 MBit/s, half-duplex |
| 200 | 100 MBit/s, full duplex |
| 1000 | 1000 MBit/s, half-duplex |
| 2000 | 1000 MBit/s, full duplex |

**IP**

| | |
|---|---|
| 104531 | ETH (X102): Current IP address (rw) |
| 104532 | ETH (X102): Current subnet mask (rw) |
| 104533 | ETH (X102): Current default gateway (rw) |

**Routing table**

104550 Status

| | |
|---|---|
| 0 | No error |
| -1 | Routing table is full |
| -2 | Entry not found |
| -3 | Port is not active |
| -4 | TCP/IP stack not initialized |

104551 Command

| | |
|---|---|
| 1 | Add route |
| 0 | Delete route |

104552 Port number

| | |
|---|---|
| 1 | ETH (X102) |

| | |
|---|---|
| 104553 | IP address |
| 104554 | Subnet mask |
| 104555 | Gateway |

## CPU

108002 All LED ON/OFF (bit-coded)
Bit 0: LED **GP1 (RUN)**
Bit 1: LED **GP3 (ERR)**
Bit 2: LED **GP2 (D1)**

108003 LED **GP1 (RUN)**
0 = OFF
1 = Flashing slowly
2 = Flashing fast
3 = ON

108004 LED **GP3 (ERR)**
0 = OFF
3 = ON

108005 LED **GP2 (D1)**
0 = OFF
3 = ON

108015 Mode selector
1 = LOAD
2 = RUN
3 = STOP

## General system registers

200000 OS version (major * 100 + minor)
200001 Application program is running (bit 0 = 1)
200008 Error register (identical with 210004)

| | |
|---|---|
| Bit 0: | Flash disk error |
| Bit 2: | Error on JX2 system bus |
| Bit 3: | Error on Ethernet system bus |
| Bit 4: | Error on application register |
| Bit 7: | Error in expanded error register |
| 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 17: | Memory protection violated |
| Bit 24: | Timeout - Cycle time |
| Bit 25: | Timeout - Task lock |
| Bit 31: | Unknown error |

200009 Enhanced error register (bit-coded)

| | |
|---|---|
| Bit 3: | Error in ModConfig.da |
| Bit 10: | A bus node (publish/subscribe client) has reported an error to the controller |
| Bit 12: | Error message by JetIPScan |
| Bit 16: | Error message by NetConsistency |
| Bit 22: | Warning: Logger has been activated |

200010 Enhanced error register (bit-coded)
Bit 1: Error in the MC object

200051 JetIPScan error number register
200061 Error number register of NetConsistency

200169 OS version (IP format)
200170 Controller type (970)

201000 Runtime register in milliseconds (rw)
201001 Runtime register in seconds (rw)
201002 Runtime register in R 201003 Units (rw)
201003 * 10 ms units for R 201002 (rw)
201004 Runtime register in milliseconds (ro)
201005 Runtime registers in microseconds (ro)

202930 Web status (bit-coded)

| | |
|---|---|
| Bit 0 = 1: | FTP server available |
| Bit 1 = 1: | HTTP server available |
| Bit 2 = 1: | E-mail available |
| Bit 3 = 1: | Data file function available |
| Bit 4 = 1: | Modbus/TCP exists |
| Bit 5 = 1: | Modbus/TCP available |

202936 Control register - File system
0xc4697a4b: Formatting the flash disk

| | |
|---|---|
| 202960 | Password for system command register (0x424f6f74) |
| 202961 | System command register |
| | 102 Controller restart (reboot) |
| | 103 Application register test |
| | 104 Resetting the configuration register |
| | 122 Waiting for communication - OFF |
| | 123 Waiting for communication - ON |
| | 160 Task switch on I/O access - OFF |
| | 161 Task switch on I/O access - ON |
| | 301 Save the flash disk drive |
| | 310 Load the configuration data |
| | 311 Load ModConfig.da |
| | 312 Load configuration for Ethernet system bus |
| | 313 Stop Ethernet system bus |
| | 330 JetIPScan client - OFF |
| | 331 JetIPScan client - ON |
| 202962 | System status register (bit-coded) |
| | Bit 0 = 1: Task switch on I/O access |
| | Bit 1 = 1: Without waiting for communication |
| | Bit 2 = 1: JetIPScan client is ON |
| 202970 | Password for start delay (0x424f6f74) |
| 202971 | Delay time in 100 ms |
| 202980 | Error history: Status |
| 202981 | Error history: Command |
| 202984 | Error history: Number of entries |
| 202985 | Error history: Index |
| 202986 | Error history: Entry |
| 202987 | Error history: Stop error code |
| 202988 | Error history: Stop error code - Threshold |
| 203000 | Interface monitoring: JetIP |
| 203005 | Interface monitoring: Debug server |
| 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

| | |
|---|---|
| 210000 | Application program is running (bit 0 = 1) |
| 210001 | JetVM version |
| 210004 | Error register (bit-coded) |
| | Bit 0: Flash disk error |
| | Bit 2: Error on JX2 system bus |
| | Bit 3: Error on JetEthernetBus |
| | Bit 4: Error on application register |
| | Bit 7: Error in expanded error register |
| | 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 17: Memory protection violated |
| | 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 the presently ongoing runtime |
| 210051 | ID of the presently ongoing runtime |
| 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<br>Apply the STX function `TaskGetInfo()` as described in the JetSym online help. |
| 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<br>-1: Monitoring disabled |
| 210610 | Timeout (bit-coded)<br>Bit 0 -> Timer 0, etc. |
| | **TCP auto-close for the STX debug server** |
| 212000 | Number of open connections |
| 212001 | Mode |
| 212002 | Time |

## Networking via JetIP

| | |
|---|---|
| | **TCP auto-close for the JetIP/TCP server** |
| 230000 | Number of open connections |
| 230001 | Mode |
| 230002 | Time |
| | **Further registers for networking via JetIP** |
| 232708 | Timeout in milliseconds |
| 232709 | Response time in milliseconds |
| 232710 | Amount of network errors |

| | |
|---|---|
| 232711 | Error code of last access<br>0 = No error |
| | 1 = Timeout |
| | 3 = Error message from the remote station |
| | 5 = Invalid network address |
| | 6 = Invalid amount of registers |
| | 7 = Invalid interface number |
| 232717 | Maximum number of retries |
| 232718 | Number of retries |

**Network registers**

| | |
|---|---|
| 235000 ... 235399 | IP addresses |
| 235400 ... 235799 | Port numbers |
| 236000 ... 236399 | Indirect register numbers |
| | GNN: nnn = 000 ... 199: |
| 1nnn020000 ... 1nnn179999 | JX3 module registers |
| 1nnn202000 ... 1nnn227999 | JX2 module registers |
| 1nnn810000 ... 1nnn819999 | JetMove registers |
| 1nnn980000 ... 1nnn980199 | Indirect access via local register 236xxx |
| 1nnn990000 ... 1nnn999999 | Indirect access with variable destination window |

## Ethernet system bus

**Subscriber**

| | |
|---|---|
| 250000 | Status (bit-coded)<br>Bit 0 = 1: No CRC |
| | Bit 1 = 1: Error in connection with a subscription |
| | Bit 7 = 1: Subscriber is running |
| 250001 | Command<br>102: Restart |
| | 105: STOPP |
| | 110: Acknowledge error |
| 250002 | Subscription ID of the last error |
| 250003 | Number of subscriptions |
| 250004 | CRC of configuration file |
| 250010 | Selection via command |
| 250011 | Selection via ID |

**Subscription**

| | |
|---|---|
| 250020 | Status |
| 250021 | Mode |
| 250022 | Number of elements |
| 250023 | Multicast group |
| 250024 | Hash |
| 250025 | Current sequence number |
| 250026 | Size (bytes) |
| 250027 | Timeout |
| 250028 | Number of received publications |
| 250029 | Number of timeout errors |
| 250030 | Number of sequence number errors |
| 250100 ... 250999 | 9 more subscriber register blocks |

**Address of the bus node (or controller) exceeding the timeout time**

| | |
|---|---|
| 254001 | GNN |
| 254002 | IP address |
| 254003 | Port number |

**Publisher**

| | |
|---|---|
| 255000 | Status (bit-coded)<br>Bit 0 = 1: No CRC |
| | Bit 1 = 1: Error in connection with a publication |
| | Bit 7 = 1: Subscriber is running |
| 255001 | Command<br>102: Restart |
| | 105: STOPP |
| | 110: Acknowledge error |
| 255002 | Publication ID of the last error |
| 255003 | Number of publications |
| 255004 | CRC of configuration file |
| 255010 | Selection via command |
| 255011 | Selection via ID |

**Publication**

| | |
|---|---|
| 255020 | Status |
| 255021 | Mode |
| 255022 | Number of elements |
| 255023 | Multicast group |
| 255024 | Hash |
| 255025 | Current sequence number |
| 255026 | Size (bytes) |
| 255027 | Cycle time |
| 255028 | Number of publications sent |
| 255029 | Number of retries |
| 255030 | Number of transmit errors |
| 255100 ... 255999 | 9 more publisher register blocks |

## RemoteScan

| | |
|---|---|
| 262965 | Protocol type |
| 262966 | Amount of configuration blocks |
| 262967 | Status |

## Modbus/TCP

| | |
|---|---|
| 272702 | Register offset |
| 272704 | Input offset |
| 272705 | Output offset |
| 278000 ... 278999 | 16-bit I/O registers overlaid by virtual I/Os 20001 ... 36000 |

## E-mail

| | |
|---|---|
| 292932 | IP address of SMTP server |
| 292933 | IP address of POP3 server |
| 292934 | Port number of SMTP server |
| 292935 | Port number of POP3 server |
| 292937 | Status of e-mail processing |
| 292938 | Task ID - E-mail |

## File system/data file function

| | |
|---|---|
| 312977 | Status of file operation |
| 312978 | Task ID |

## FTP client

| | |
|---|---|
| 320000 | Number of open connections |
| 320001 | Command |
| 320002 | Timeout |
| 320003 | Server port |
| 320004 | Selection via number |
| 320005 | Selection via handle |
| 320006 | Server socket: IP address |
| 320007 | Server socket: Port |
| 320008 | Client socket: IP address |
| 320009 | Client socket: Port |
| 320100 | Access status |
| 320101 | Task ID |

## User-programmable IP interface

**Reading out the connection list**

| | |
|---|---|
| 350000 | Last result (-1 = no connection selected) |
| 350001 | 1 = Client; 2 = Server |
| 350002 | 1 = UDP; 2 = TCP |
| 350003 | IP address |
| 350004 | Port number |
| 350005 | Connection state |
| 350006 | Number of sent bytes |
| 350007 | Number of received bytes |

## Error history

| | |
|---|---|
| 380000 | Status<br>Bit 0 = 1: Recording<br>Bit 1 = 1: Stop if buffer is full<br>Bit 2 = 1: Stop on error code<br>Bit 3 = 1: Non-volatile memory |
| 380001 | Command<br>1: Clear error log<br>2: Start error log<br>3: Stop error log<br>4: Stop if error buffer is full<br>5: Circular buffer<br>6: Stop on error code ON<br>7: Stop on error code OFF<br>10: Non-volatile memory<br>11: Dynamic memory |
| 380002 | Buffer length |
| 380003 | Maximum buffer length |
| 380004 | Number of error entries |
| 380005 | Index to error list |
| 380006 | Error entry |
| 380007 | Error stop code |
| 380008 | Number of codes until stop |

## I/O networking

**Status register**

| | |
|---|---|
| 390000 + node * 10 | Error register |
| 390001 + node * 10 | Enhanced error register 1 |
| 390002 + node * 10 | Enhanced error register 2 |
| 390003 + node * 10 | JetSync status |
| 390004 + node * 10 | Subscriber status |
| 390005 + node * 10 | Subscription ID of the last error |

**Address of a bus node (not of a controller) having reported an error**

| | |
|---|---|
| 394001 | GNN |
| 394002 | IP address |
| 394003 | Port number |

**Control register**

| | |
|---|---|
| 395000 + node * 10 | Command |

## NetConsistency function

Applies to JC-340, JC-350, JC-970MC, and JC-945MC only.

**Basic drivers**

| | |
|---|---|
| 470000 ... 470008 | Cookie |
| 470009 | Version |
| 470010 | Status<br>Bit 0 = 1: Error<br>Bit 1 = 1: Alarms<br>Bit 2 = 1: Basic driver initialized |
| 470011 | Command<br>0: There are no commands |
| 470020 | Maximum possible number of instances |
| 470021 | Number of instances ready for operation |
| 470030 | Max. number of error messages for the logger |
| 470031 | Number of error messages transmitted to the logger |
| 470032 | Max. number of warnings for the logger |
| 470033 | Number of warnings forwarded to the logger |
| 470034 | Max. possible number of error history entries |
| 470035 | Number of entries in the error history |
| 470040 | Error numbers |
| 470041 | Time of the error in ms |
| 470042 | Instance, at which the error occurred |
| 470043 | Number of error parameters |
| 470044 ... 470048 | Error parameters 1 through 5 |
| 470049 | Number of characters of the error message |
| 470050 ... 470157 | Text of the error message |

**First instance**

| | |
|---|---|
| 471010 | Status<br>Bit 0 = 1: Error<br>Bit 1 = 1: Alarms<br>Bit 2 = 1: An instance has been initialized<br>Bit 3 = 1: Execution in process |
| 471011 | Command<br>0: There are no commands |

## Application registers

| | |
|---|---|
| 1000000 ... 1119999 | 32-bit integer or floating point number (non-volatile) |

## Basic register for initializing the PCIe bus

| | |
|---|---|
| 20SJ00000 | Global status register<br>Bit 30: Initialization completed<br>Bit 31: Error when initializing |
| 20SJ00001 | Command register<br>1: Initializing the bus |
| 20SJ00002 | Number of detected riser cards |
| 20SJ00005 | Bus initialization status<br>0: Initialization is running<br>1: Initialization OK<br>-1: Error when initializing |
| 20SJ00006 | Number of JX2 modules |
| 20SJ00007 | Types of the JX6-I/O submodules<br>73: JX6-SB and JX6-SB-I<br>5: JX6-SV1<br>16: JX6-IO16CB |
| 20SJ00010 | Timeout while waiting for resetting the command |
| 20SJ00011 | Timeout while waiting for semaphore |
| 20SJ00012 | Timeout in the interpreter for total access |

## Access to controller modules (JX6-SB(I))

Flag 2105 = 1 reports an error

| | |
|---|---|
| 20SJ00050 | Access error - controller module |

| | |
|---|---|
| | Slot number - 1 |
| 20SJ00051 | Access error - controller module<br>Axis number - 1 |
| 20SJ00052 | Access error - controller module<br>Register number |

## Digital I/O module JX6-IO16CB

| | |
|---|---|
| 20SJ00100 | State of the digital inputs |
| 20SJ00101 | State of digital outputs |
| 20SJ00102 | Error state of the digital outputs |
| 20SJ00103 | Control register |
| 20SJ00104 | Filter time of the inputs 1 through 4 |
| 20SJ00105 | Filter time of the inputs 5 through 8 |

## Combi module JX6-SV1

| | |
|---|---|
| 20SJ00100 | Module ID |
| 20SJ00101 | Hardware configuration |
| 20SJ00102 | Analog output |
| 20SJ00103 | Strobe value/encoder value received |
| 20SJ00104 | Preset/pulse generator |
| 20SJ00105 | Count value |
| 20SJ00106 | Reference/offset |
| 20SJ00107 | Control register |
| 20SJ00108 | Status register |
| 20SJ00110 | Sampling timer |

## Submodule JX6-SB(-I)

| | |
|---|---|
| | **Operating mode "Master-Slave"** |
| 20SJ00100 | Status register |
| 20SJ00101 | Command register |
| 20SJ00109 | Firmware version number |
| 20SJ00201 | Register index of the JX-SIO |
| 20SJ00202 | Register date of the JX-SIO |

## JX2 system bus registers

| | |
|---|---|
| 20SJ02000 | Version of JX2 system bus driver (IP) |
| 20SJ02008 | Error (bit-coded)<br>Bit 3: I/O or CANopen® module timeout<br>Bit 4: JX2 slave module timeout<br>Bit 9: Error of I/O module periphery<br>Bit 13: Error during JX2 system bus initialization<br>Bit 14: Timeout of system registers |
| 20SJ02011 | I/O module number at timeout |
| 20SJ02012 | JX2 slave module number at timeout |
| 20SJ02013 | Amount of connected I/O modules |
| 20SJ02014 | Amount of connected JX2 slave modules |
| 20SJ02015 | Index to module array |
| 20SJ02016 | Module array |
| 20SJ02023 | Dummy I/O module |
| 20SJ02024 | JX2 slave dummy modules |
| 20SJ02028 | Monitoring interval for I/O modules [10 ms] |
| 20SJ02029 | Baud rate of JX2 system bus |
| 20SJ02032 | ON delay |
| 20SJ02039 | I/O module where a peripheral fault has occurred (bit-coded) |
| 20SJ02070 | Number of CANopen® modules |
| 20SJ02071 | Actual I/O sum of modules on the JX2 system bus |
| 20SJ02072 | Version of JX2 system bus driver (IP) |
| 20SJ02073 | Timeout for register access to CANopen® modules |
| 20SJ02074 | CANopen® sync intervall [ms] |
| 20SJ02077 | Enabling JX2 system bus special functions<br>Bit 2: CAN-Prim<br>Bit 3: CAN-Prim only |
| 20SJ02080 | CANopen® module index for JX2 system bus application registers |
| 20SJ02085 | SysBus application regs: Register number (65-89) |
| 20SJ02086 | SysBus application regs: Object number |
| 20SJ02087 | SysBus application regs: Sub-index |
| 20SJ02088 | SysBus application regs: Length |
| 20SJ02638 | Special flag, overlaid<br>Bit 0: Flag 2048<br>Timeout during an access to a JX2-I/O module<br>Bit 1: Flag 2049<br>Timeout during an access to a JX2 slave module<br>Bit 2: Flag 2050<br>Timeout during register access to a JX2-I/O module |
| 20SJ02639 | Special flag, overlaid<br>Bit 1: Flag 2065<br>Activated error signalization for output driver errors<br>Bit 3: Fatal system bus error |
| 20SJ02651 | Special flag, overlaid<br>Bit 14: Flag 2270<br>Timeout during an access to an inactive JX-SIO module |
| 20SJ02652 | Special flag, overlaid<br>Bit 0: Flag 2272<br>Access to an unknown JX-SIO register<br>Bit 1: Flag 2273<br>Access to a JX-SIO register that is not supported<br>Bit 2: Flag 2274<br>Timeout when monitoring a JX-SIO module<br>Bit 3: Flag 2275<br>JX-SIO module is not ready for operation<br>Bit 4: Flag 2276<br>Overflow during a read access to a 32-bit register<br>Bit 5: Flag 2277<br>Selecting the upper 8 bits for 32-bit write access to JX-SIO-registers |
| 20SJ02760 | Max. number of I/O update retries |
| 20SJ02761 | Index of array of I/O retry counters |
| 20SJ02762 | Array of I/O retry counters |
| 20SJ02763 | Timeout for I/O update of I/O modules [ms] |
| 20SJ02764 | Timeout for register access to I/O modules [ms] |
| 20SJ02765 | Timeout for register access to JX2 slave modules [ms] |
| 20SJ02995 | Bootloader version of JX2 system bus interface |
| 20SJ03xx0 ...<br>20SJ03xx9 | Registers on I/O modules<br>xx: I/O module number - 2 (00 ... 22) |
| 20SJ04000<br>...<br>20SJ04367 | Inputs/outputs mapped to registers<br>(see below) |
| 20SJ05x00<br>...<br>20SJ06x99 | I/O registers: CANopen®/JX-SIO<br>x: I/O module number - 70 (0...9) |
| 20SJ07x00<br>...<br>20SJ07x99 | Configuration registers: CANopen®/JX-SIO<br>x: I/O module number - 70 (0...9) |
| 20SJxx100 ...<br>20SJxx999 | JX2 slave registers<br>xx: JX2 slave number + 10 |

## CAN-Prim registers

| | |
|---|---|
| 20SJ10500 | Status register<br>Bit 1 = 1: CAN message received<br>Bit 2 = 0: 11-bit CAN ID<br>Bit 2 = 1: 29-bit CAN ID |
| 20SJ10501 | Command register<br>7 = clear FIFO |

| | |
|---|---|
| | 8 = Set CAN ID to 11 bits |
| | 9 = Set CAN ID to 29 bits |
| | 10 = Check boxes for received messages |
| 20SJ10503 | FIFO buffer occupancy |
| 20SJ10504 | FIFO data |
| 20SJ10506 | Global receive mask |
| 20SJ10507 | Global receive ID |
| 20SJ10530 + box * 20 | Status register of the box |
| 20SJ10531 + box * 20 | Configuration register of the box |
| 20SJ10532 + box * 20 | CAN-ID |
| 20SJ10533 + box * 20 | Number of data bytes |
| 20SJ10534 ... 20SJ10541 + box * 20 | Data bytes |
| 20SJ10542 + box * 20 | CAN-ID mask |
| 20SJ10543 + box * 20 | Command register of the box |
| 20SJ10544 + box * 20 | Received CAN-ID |

## Inputs/outputs

| | |
|---|---|
| 20001 ... 36000 | Virtual I/Os for RemoteScan |
| 20SJ00101 ... 20SJ00108 | JX6-IO16CB |
| 20SJ0xx01 ... | JX2 modules (xx: 02 ... 32); |
| 20SJ0xx16 | JX3 modules via JX3-BN-CAN (xx: 02 ... 17) |
| 1nnn01xx01 ... | JX3 modules via JX3-BN-ETH |
| 1nnn01xx16 | GNN: nnn = 000 ... 199: xx: 02 ... 17 |

## 32 combined inputs

**JX2 system bus: + 20SJ00000**
**Network: + 1nnn910000**

| | | | | |
|---|---|---|---|---|
| 4000 | 101..108 | 109..116 | 201..208 | 209..216 |
| 4001 | 109..116 | 201..208 | 209..216 | 301..308 |
| 4002 | 201..208 | 209..216 | 301..308 | 309..316 |
| 4003 | 209..216 | 301..308 | 309..316 | 401..408 |
| 4004 | 301..308 | 309..316 | 401..408 | 409..416 |
| 4005 | 309..316 | 401..408 | 409..416 | 501..508 |
| 4006 | 401..408 | 409..416 | 501..508 | 509..516 |
| 4007 | 409..416 | 501..508 | 509..516 | 601..608 |
| 4008 | 501..508 | 509..516 | 601..608 | 609..616 |
| 4009 | 509..516 | 601..608 | 609..616 | 701..708 |
| 4010 | 601..608 | 609..616 | 701..708 | 709..716 |
| 4011 | 609..616 | 701..708 | 709..716 | 801..808 |
| 4012 | 701..708 | 709..716 | 801..808 | 809..816 |
| 4013 | 709..716 | 801..808 | 809..816 | 901..908 |
| 4014 | 801..808 | 809..816 | 901..908 | 909..916 |
| 4015 | 809..816 | 901..908 | 909..916 | 1001..1008 |
| 4016 | 901..908 | 909..916 | 1001..1008 | 1009..1016 |
| 4017 | 909..916 | 1001..1008 | 1009..1016 | 1101..1108 |
| 4018 | 1001..1008 | 1009..1016 | 1101..1108 | 1109..1116 |
| 4019 | 1009..1016 | 1101..1108 | 1109..1116 | 1201..1208 |
| 4020 | 1101..1108 | 1109..1116 | 1201..1208 | 1209..1216 |
| 4021 | 1109..1116 | 1201..1208 | 1209..1216 | 1301..1308 |
| 4022 | 1201..1208 | 1209..1216 | 1301..1308 | 1309..1316 |
| 4023 | 1209..1216 | 1301..1308 | 1309..1316 | 1401..1408 |
| 4024 | 1301..1308 | 1309..1316 | 1401..1408 | 1409..1416 |
| 4025 | 1309..1316 | 1401..1408 | 1409..1416 | 1501..1508 |
| 4026 | 1401..1408 | 1409..1416 | 1501..1508 | 1509..1516 |
| 4027 | 1409..1416 | 1501..1508 | 1509..1516 | 1601..1608 |
| 4028 | 1501..1508 | 1509..1516 | 1601..1608 | 1609..1616 |
| 4029 | 1509..1516 | 1601..1608 | 1609..1616 | 1701..1708 |
| 4030 | 1601..1608 | 1609..1616 | 1701..1708 | 1709..1716 |
| 4031 | 1609..1616 | 1701..1708 | 1709..1716 | 1801..1808 |
| 4032 | 1701..1708 | 1709..1716 | 1801..1808 | 1809..1816 |
| 4033 | 1709..1716 | 1801..1808 | 1809..1816 | 1901..1908 |
| 4034 | 1801..1808 | 1809..1816 | 1901..1908 | 1909..1916 |
| 4035 | 1809..1816 | 1901..1908 | 1909..1916 | 2001..2008 |
| 4036 | 1901..1908 | 1909..1916 | 2001..2008 | 2009..2016 |
| 4037 | 1909..1916 | 2001..2008 | 2009..2016 | 2101..2108 |
| 4038 | 2001..2008 | 2009..2016 | 2101..2108 | 2109..2116 |
| 4039 | 2009..2016 | 2101..2108 | 2109..2116 | 2201..2208 |
| 4040 | 2101..2108 | 2109..2116 | 2201..2208 | 2209..2216 |
| 4041 | 2109..2116 | 2201..2208 | 2209..2216 | 2301..2308 |
| 4042 | 2201..2208 | 2209..2216 | 2301..2308 | 2309..2316 |
| 4043 | 2209..2216 | 2301..2308 | 2309..2316 | 2401..2408 |
| 4044 | 2301..2308 | 2309..2316 | 2401..2408 | 2409..2416 |

## 16 combined inputs

**JX2 system bus: + 20SJ00000**
**Network: + 1nnn910000**

| | | |
|---|---|---|
| 4060 | 101..108 | 109..116 |
| 4061 | 109..116 | 201..208 |
| 4062 | 201..208 | 209..216 |
| 4063 | 209..216 | 301..308 |
| 4064 | 301..308 | 309..316 |
| 4065 | 309..316 | 401..408 |
| 4066 | 401..408 | 409..416 |
| 4067 | 409..416 | 501..508 |
| 4068 | 501..508 | 509..516 |
| 4069 | 509..516 | 601..608 |
| 4070 | 601..608 | 609..616 |
| 4071 | 609..616 | 701..708 |
| 4072 | 701..708 | 709..716 |
| 4073 | 709..716 | 801..808 |
| 4074 | 801..808 | 809..816 |
| 4075 | 809..816 | 901..908 |
| 4076 | 901..908 | 909..916 |
| 4077 | 909..916 | 1001..1008 |
| 4078 | 1001..1008 | 1009..1016 |
| 4079 | 1009..1016 | 1101..1108 |
| 4080 | 1101..1108 | 1109..1116 |
| 4081 | 1109..1116 | 1201..1208 |
| 4082 | 1201..1208 | 1209..1216 |
| 4083 | 1209..1216 | 1301..1308 |
| 4084 | 1301..1308 | 1309..1316 |
| 4085 | 1309..1316 | 1401..1408 |
| 4086 | 1401..1408 | 1409..1416 |
| 4087 | 1409..1416 | 1501..1508 |
| 4088 | 1501..1508 | 1509..1516 |
| 4089 | 1509..1516 | 1601..1608 |
| 4090 | 1601..1608 | 1609..1616 |
| 4091 | 1609..1616 | 1701..1708 |
| 4092 | 1701..1708 | 1709..1716 |
| 4093 | 1709..1716 | 1801..1808 |
| 4094 | 1801..1808 | 1809..1816 |
| 4095 | 1809..1816 | 1901..1908 |
| 4096 | 1901..1908 | 1909..1916 |
| 4097 | 1909..1916 | 2001..2008 |
| 4098 | 2001..2008 | 2009..2016 |
| 4099 | 2009..2016 | 2101..2108 |
| 4100 | 2101..2108 | 2109..2116 |
| 4101 | 2109..2116 | 2201..2208 |
| 4102 | 2201..2208 | 2209..2216 |
| 4103 | 2209..2216 | 2301..2308 |
| 4104 | 2301..2308 | 2309..2316 |
| 4105 | 2309..2316 | 2401..2408 |
| 4106 | 2401..2408 | 2409..2416 |

## 8 combined inputs

**JX2 system bus: + 20SJ00000**
**Network: + 1nnn910000**

| | |
|---|---|
| 4120 | 101..108 |
| 4121 | 109..116 |
| 4122 | 201..208 |
| 4123 | 209..216 |
| 4124 | 301..308 |
| 4125 | 309..316 |
| 4126 | 401..408 |
| 4127 | 409..416 |
| 4128 | 501..508 |
| 4129 | 509..516 |
| 4130 | 601..608 |
| 4131 | 609..616 |
| 4132 | 701..708 |

| | |
|---|---|
| 4133 | 709..716 |
| 4134 | 801..808 |
| 4135 | 809..816 |
| 4136 | 901..908 |
| 4137 | 909..916 |
| 4138 | 1001..1008 |
| 4139 | 1009..1016 |
| 4140 | 1101..1108 |
| 4141 | 1109..1116 |
| 4142 | 1201..1208 |
| 4143 | 1209..1216 |
| 4144 | 1301..1308 |
| 4145 | 1309..1316 |
| 4146 | 1401..1408 |
| 4147 | 1409..1416 |
| 4148 | 1501..1508 |
| 4149 | 1509..1516 |
| 4150 | 1601..1608 |
| 4151 | 1609..1616 |
| 4152 | 1701..1708 |
| 4153 | 1709..1716 |
| 4154 | 1801..1808 |
| 4155 | 1809..1816 |
| 4156 | 1901..1908 |
| 4157 | 1909..1916 |
| 4158 | 2001..2008 |
| 4159 | 2009..2016 |
| 4160 | 2101..2108 |
| 4161 | 2109..2116 |
| 4162 | 2201..2208 |
| 4163 | 2209..2216 |
| 4164 | 2301..2308 |
| 4165 | 2309..2316 |
| 4166 | 2401..2408 |
| 4167 | 2409..2416 |

## 32 combined outputs

**JX2 system bus: + 20SJ00000**
**Network: + 1nnn910000**

| | | | | |
|---|---|---|---|---|
| 4200 | 101..108 | 109..116 | 201..208 | 209..216 |
| 4201 | 109..116 | 201..208 | 209..216 | 301..308 |
| 4202 | 201..208 | 209..216 | 301..308 | 309..316 |
| 4203 | 209..216 | 301..308 | 309..316 | 401..408 |
| 4204 | 301..308 | 309..316 | 401..408 | 409..416 |
| 4205 | 309..316 | 401..408 | 409..416 | 501..508 |
| 4206 | 401..408 | 409..416 | 501..508 | 509..516 |
| 4207 | 409..416 | 501..508 | 509..516 | 601..608 |
| 4208 | 501..508 | 509..516 | 601..608 | 609..616 |
| 4209 | 509..516 | 601..608 | 609..616 | 701..708 |
| 4210 | 601..608 | 609..616 | 701..708 | 709..716 |
| 4211 | 609..616 | 701..708 | 709..716 | 801..808 |
| 4212 | 701..708 | 709..716 | 801..808 | 809..816 |
| 4213 | 709..716 | 801..808 | 809..816 | 901..908 |
| 4214 | 801..808 | 809..816 | 901..908 | 909..916 |
| 4215 | 809..816 | 901..908 | 909..916 | 1001..1008 |
| 4216 | 901..908 | 909..916 | 1001..1008 | 1009..1016 |
| 4217 | 909..916 | 1001..1008 | 1009..1016 | 1101..1108 |
| 4218 | 1001..1008 | 1009..1016 | 1101..1108 | 1109..1116 |
| 4219 | 1009..1016 | 1101..1108 | 1109..1116 | 1201..1208 |
| 4220 | 1101..1108 | 1109..1116 | 1201..1208 | 1209..1216 |
| 4221 | 1109..1116 | 1201..1208 | 1209..1216 | 1301..1308 |
| 4222 | 1201..1208 | 1209..1216 | 1301..1308 | 1309..1316 |
| 4223 | 1209..1216 | 1301..1308 | 1309..1316 | 1401..1408 |
| 4224 | 1301..1308 | 1309..1316 | 1401..1408 | 1409..1416 |
| 4225 | 1309..1316 | 1401..1408 | 1409..1416 | 1501..1508 |
| 4226 | 1401..1408 | 1409..1416 | 1501..1508 | 1509..1516 |
| 4227 | 1409..1416 | 1501..1508 | 1509..1516 | 1601..1608 |
| 4228 | 1501..1508 | 1509..1516 | 1601..1608 | 1609..1616 |
| 4229 | 1509..1516 | 1601..1608 | 1609..1616 | 1701..1708 |
| 4230 | 1601..1608 | 1609..1616 | 1701..1708 | 1709..1716 |
| 4231 | 1609..1616 | 1701..1708 | 1709..1716 | 1801..1808 |
| 4232 | 1701..1708 | 1709..1716 | 1801..1808 | 1809..1816 |
| 4233 | 1709..1716 | 1801..1808 | 1809..1816 | 1901..1908 |
| 4234 | 1801..1808 | 1809..1816 | 1901..1908 | 1909..1916 |
| 4235 | 1809..1816 | 1901..1908 | 1909..1916 | 2001..2008 |
| 4236 | 1901..1908 | 1909..1916 | 2001..2008 | 2009..2016 |
| 4237 | 1909..1916 | 2001..2008 | 2009..2016 | 2101..2108 |
| 4238 | 2001..2008 | 2009..2016 | 2101..2108 | 2109..2116 |
| 4239 | 2009..2016 | 2101..2108 | 2109..2116 | 2201..2208 |
| 4240 | 2101..2108 | 2109..2116 | 2201..2208 | 2209..2216 |
| 4241 | 2109..2116 | 2201..2208 | 2209..2216 | 2301..2308 |
| 4242 | 2201..2208 | 2209..2216 | 2301..2308 | 2309..2316 |
| 4243 | 2209..2216 | 2301..2308 | 2309..2316 | 2401..2408 |
| 4244 | 2301..2308 | 2309..2316 | 2401..2408 | 2409..2416 |

## 16 combined outputs

**JX2 system bus: + 20SJ00000**
**Network: + 1nnn910000**

| | | |
|---|---|---|
| 4260 | 101..108 | 109..116 |
| 4261 | 109..116 | 201..208 |
| 4262 | 201..208 | 209..216 |
| 4263 | 209..216 | 301..308 |
| 4264 | 301..308 | 309..316 |
| 4265 | 309..316 | 401..408 |
| 4266 | 401..408 | 409..416 |
| 4267 | 409..416 | 501..508 |
| 4268 | 501..508 | 509..516 |
| 4269 | 509..516 | 601..608 |
| 4270 | 601..608 | 609..616 |
| 4271 | 609..616 | 701..708 |
| 4272 | 701..708 | 709..716 |
| 4273 | 709..716 | 801..808 |
| 4274 | 801..808 | 809..816 |
| 4275 | 809..816 | 901..908 |
| 4276 | 901..908 | 909..916 |
| 4277 | 909..916 | 1001..1008 |
| 4278 | 1001..1008 | 1009..1016 |
| 4279 | 1009..1016 | 1101..1108 |
| 4280 | 1101..1108 | 1109..1116 |
| 4281 | 1109..1116 | 1201..1208 |
| 4282 | 1201..1208 | 1209..1216 |
| 4283 | 1209..1216 | 1301..1308 |
| 4284 | 1301..1308 | 1309..1316 |
| 4285 | 1309..1316 | 1401..1408 |
| 4286 | 1401..1408 | 1409..1416 |
| 4287 | 1409..1416 | 1501..1508 |
| 4288 | 1501..1508 | 1509..1516 |
| 4289 | 1509..1516 | 1601..1608 |
| 4290 | 1601..1608 | 1609..1616 |
| 4291 | 1609..1616 | 1701..1708 |
| 4292 | 1701..1708 | 1709..1716 |
| 4293 | 1709..1716 | 1801..1808 |
| 4294 | 1801..1808 | 1809..1816 |
| 4295 | 1809..1816 | 1901..1908 |
| 4296 | 1901..1908 | 1909..1916 |
| 4297 | 1909..1916 | 2001..2008 |
| 4298 | 2001..2008 | 2009..2016 |
| 4299 | 2009..2016 | 2101..2108 |
| 4300 | 2101..2108 | 2109..2116 |
| 4301 | 2109..2116 | 2201..2208 |
| 4302 | 2201..2208 | 2209..2216 |
| 4303 | 2209..2216 | 2301..2308 |
| 4304 | 2301..2308 | 2309..2316 |
| 4305 | 2309..2316 | 2401..2408 |
| 4306 | 2401..2408 | 2409..2416 |

## 8 combined outputs

**JX2 system bus: + 20SJ00000**
**Network: + 1nnn910000**

| | |
|---|---|
| 4320 | 101..108 |
| 4321 | 109..116 |
| 4322 | 201..208 |
| 4323 | 209..216 |
| 4324 | 301..308 |
| 4325 | 309..316 |
| 4326 | 401..408 |
| 4327 | 409..416 |
| 4328 | 501..508 |
| 4329 | 509..516 |
| 4330 | 601..608 |
| 4331 | 609..616 |
| 4332 | 701..708 |
| 4333 | 709..716 |
| 4334 | 801..808 |
| 4335 | 809..816 |
| 4336 | 901..908 |

| | |
|---|---|
| 4337 | 909..916 |
| 4338 | 1001..1008 |
| 4339 | 1009..1016 |
| 4340 | 1101..1108 |
| 4341 | 1109..1116 |
| 4342 | 1201..1208 |
| 4343 | 1209..1216 |
| 4344 | 1301..1308 |
| 4345 | 1309..1316 |
| 4346 | 1401..1408 |
| 4347 | 1409..1416 |
| 4348 | 1501..1508 |
| 4349 | 1509..1516 |
| 4350 | 1601..1608 |
| 4351 | 1609..1616 |
| 4352 | 1701..1708 |
| 4353 | 1709..1716 |
| 4354 | 1801..1808 |
| 4355 | 1809..1816 |
| 4356 | 1901..1908 |
| 4357 | 1909..1916 |
| 4358 | 2001..2008 |
| 4359 | 2009..2016 |
| 4360 | 2101..2108 |
| 4361 | 2109..2116 |
| 4362 | 2201..2208 |
| 4363 | 2209..2216 |
| 4364 | 2301..2308 |
| 4365 | 2309..2316 |
| 4366 | 2401..2408 |
| 4367 | 2409..2416 |

### Special flags for networks

| | |
|---|---|
| 2075 | Error in networking via JetIP |

### Special flags - Publish/subscribe

| | |
|---|---|
| 2080 | Enable for publishing an error |
| 2081 | Error collection of the subscriber |

### Special flags - Interface monitoring

| | |
|---|---|
| 2088 | OS flag - JetIP |
| 2089 | User flag - JetIP |
| 2098 | OS flag - Debug server |
| 2099 | User flag - Debug server |

### 32 combined flags

| | |
|---|---|
| 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

| | |
|---|---|
| 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

| | |
|---|---|
| 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

| | |
|---|---|
| 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 |

### Overlaid application registers/flags

| | |
|---|---|
| 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 |

## System functions

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

| | |
|---|---|
| 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 |
| 60 | CRC generation for Modbus RTU |
| 61 | CRC check for Modbus RTU |
| 65/67 | Reading register block via Modbus/TCP |
| 66/68 | Writing register block via Modbus/TCP |
| 80/85 | Initializing RemoteScan |
| 81 | Starting RemoteScan |
| 82 | Stopping RemoteScan |
| 90 | Writing a data file |
| 91 | Appending a data file |
| 92 | Reading a data file |
| 96 | Deleting a data file |
| 110 | E-mail feature |
| 150 | Configuring NetCopyList |
| 151 | Deleting NetCopyList |
| 152 | Sending NetCopyList |

## JetSym STX functions

| System functions | 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; |
| 60 | Function ModbusCRCgen(FramePtr: Int, Length: int): Int; |
| 61 | Function ModbusCRCcheck(FramePtr: Int, Length: int): Int; |
| 65/67 | Function ModbusReadReg(Const Ref MbParam: MODBUS_PARAM): Int; |
| 66/68 | Function ModbusWriteReg(Const Ref MbParam: MODBUS_PARAM): Int; |
| 80/85 | Function RemoteScanConfig(Protocol: RSCAN_PROTOCOL, Elements: Int, Const Ref Configuration: RSCAN_DSCR): Int; |
| 81 | Function RemoteScanStart(Protocol: Int): Int; |
| 82 | Function RemoteScanStop(Protocol: Int): Int; |
| 90/91 | Function FileDAWrite(Const Ref FileName: String, Const Ref Mode: String, VarType: DAWRITE_TYPE, First: Int, Last: int): Int; |
| 92 | Function FileDARead(Const Ref FileName: String): Int; |
| 110 | Function EmailSend(Const Ref FileName: String): Int; |
| 150 | Function NetCopyListConfig(IPAddr: Int, IPPort: Int, Const Ref List: TNetCopyLinstL): Int; |
| 151 | Function NetCopyListSend(Handle: Int): Int; |
| 152 | Function NetCopyListDelete(Handle: Int): Int; |

# Appendix

**Introduction**

The appendix contains electrical and mechanical data, as well as operating data.

**Content**

# A: Technical specifications

**Introduction**

This chapter contains information on electrical and mechanical data, as well as on operating data of the JC-970MC.

**Content**

# Technical specifications

## Power supply

| Parameter | Description |
|---|---|
| Rated voltage | 24 V DC |
| Permissible voltage range | -15 % ... +20 % |
| Input current | 6 A max., typically 1.2 A (without USB) |
| Power consumption | 144 W max. |
| Voltage dips | Duration of voltage dips <= 10 ms<br>Interval between two voltage dips >= 1 s<br>Severity level PS2 |

## Memory configurations

| Parameter | Description |
|---|---|
| Non-volatile memory (registers) | 120,000 registers |
| Program/data memory | 16 MB/16 MB |

## Optional expansions

| Parameter | Description |
|---|---|
| 2 pcs. PCIexpress slots for JI-PCIE-Exx add-on cards<br>(terminals X81, X82, X91, X92) | At the moment, the following add-on cards are available:<br>JI-PCIE-E01:<br>for connecting a maximum of 31 JX2 or JX3 modules (JX3-BN-CAN is required), and a maximum of 15 JetMove-100/200 servo drives via JX2 system bus<br><br>JI-PCIE-E02:<br>for connecting a maximum of 62 JX2 or JX3 modules (JX3-BN-CAN is required), and a maximum of 30 JetMove-100/200 servo drives via two JX2 system busses<br><br>JI-PCIE-E03<br>for connecting a maximum of 31 JX2 or JX3 modules (JX3-BN-CAN is required), and a maximum of 15 JetMove-100/200 servo drives via JX2 system bus, as well as 16 local digital I/Os for quick processing of machine signals<br><br>JI-PCI-E04<br>for connecting a maximum of 31 JX2 or JX3 modules (JX3-BN-CAN is required), and a maximum of 15 JetMove-100/200 servo drives via JX2 system bus, as well as 1 incremental or SSI encoder to the interface on the card |

| Parameter | Description |
|---|---|
| Remote modules via Ethernet System Bus (X102) | JX3-BN-ETH bus nodes for JX3-I/O modules<br>JM-200 servo drives equipped with Ethernet interface (option -ETH or -JC-310) |

**Features**

| Parameter | Description |
|---|---|
| Real-time clock | Yes |
| Web server | Yes |
| E-mail client | Yes |
| Modbus TCP | Yes |

**Real-time clock**

| Parameter | Description |
|---|---|
| Power reserve | Minimum: 1 week<br>Typically: 3 weeks |
| Deviation | 1 minute per month max. |

# Operating parameters - Environment and mechanics

**Environmental conditions**

| Parameter | Value | Standard |
|---|---|---|
| Operating temperature range | 0 °C ... +50 °C | |
| Storage temperature range | -20 ... +70 °C | |
| Air humidity | 93 % at 40 °C<br>non-condensing | IEC 60068-2-78 |
| Corrosion immunity/ chemical resistance | No special protection against corrosion. Ambient air must be free from higher concentrations of acids, alcaline solutions, corrosive agents, salts, metal vapors, or other corrosive or electroconductive contaminants | |
| Maximum operating altitude | 2,000 m above sea level | DIN EN 61131-2 |

**Mechanical parameters**

| Parameter | Value | Standard |
|---|---|---|
| Vibration resistance | 5 … 57.6 Hz,<br>0.075 mm amplitude<br>57.6 … 150 Hz, 1 g | IEC 60068-2-6:2008-10 |
| Shock resistance | 15 g, 11 ms, half sine wave | IEC 60068-2-27:2010-2 |
| Degree of protection | IP20 | DIN EN 60529 |
| Mounting orientation | Vertical | |
| Weight | 4.9 kg | |
| Dimensions (HxWxD), mm | 305 x 155 x 220.4 | |

# CE directives, standards and certifications

## CE directives

| CE directive | |
|---|---|
| Electrical safety | General Product Safety Directive 2001/95/EC |
| Electromagnetic Compatibility (EMC) | EMC Directive 2004/108/EC |
| CE marking | CE Directive 93/68/EEC |
| RoHS II Directive | 2011/65/EU |

## Electrical safety

| Electrical safety | Harmonized Standards |
|---|---|
| Europe | Information technology equipment - Safety - Part 1: General requirements<br>EN 60950-1:2006+A11:2009+A12:2011+A1:2010+ AC:2011 |

## EMC - Immunity to interference

| EMC | Harmonized Standards |
|---|---|
| Europe | Industrial, scientific and medical equipment - Radio-frequency disturbance characteristics limits and methods of measurement EN 55011, class B (conducted emissions),<br>EN 55022, class B<br>EN 61000-6-4 |
| | Information technology equipment - Immunity characteristics - Limits and methods of measurement EN 55024, EN 61000-6-2,<br>EN 61000-4-3 |
| | Industrial-process control systems - Part 2: Guidance for inspection and routine testing EN 61131-2 |

# B: Additional information

**Additional information**

For more information, please refer to the JetSym Online Help.

Please also refer to the following application-oriented manuals:

- *File system*
  - FTP server has been implemented in JC-970MC.
  - This manual describes how to store and load application programs.
- *System registers*
  - Realtime clock, runtime register and interface monitoring have been implemented in JC-970MC.
- *User-programmable PRIM interfaces*
  - Contains information about the freely programmable Ethernet interface
- *Jetter Ethernet system bus*
  - Describes the architecture of TCP/UDP-IP communication between Jetter devices.
- *JX3 system*
  - Describes engineering and connection of the JX3-IO system to a Jetter controller.

# C: Index

## Q



## R



## S



## T



## U



## V



## W

Jetter automation

Jetter AG
Graeterstrasse 2
71642 Ludwigsburg | Germany

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

We automate your success.