Jetter
automation

User Manual

JC-310-JM - Controller module

60880725

We automate your success.

Item # 60880725

Revision 1.22.1

March 2016 / Printed in Germany

Address

How to contact us:

Jetter AG
Graeterstrasse 2
71642 Ludwigsburg
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

Assignment to product

This User Manual is an integral part of JC-310-JM:

Type:	
Serial #:	
Year of manufacture:	
Order #:	

To be entered by the customer:

Inventory #:	
Place of operation:	

Significance of this User Manual

This document is an integral part of the JC-310-JM:

- Keep this document in a way that it is always at hand until the JC-310-JM will be disposed of.
- Pass this document on if the JC-310-JM 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 suggestions and contributions on your part and would ask you to contact us at the following 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.

This document contains important information on the following topics:

- Transport
- Mounting
- Installation
- Programming
- Operation
- Maintenance
- Repair

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 informs the user of basic safety instructions. It also warns the user of residual dangers, if there are any.

Contents

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 conditions of use

Usage according to the intended conditions of use implies operation in accordance with this User Manual.

The controller module JC-310-JM is used to control machinery, such as conveyors, production machines, and handling machines.

Operate the JC-310-JM only within the limits and conditions set forth in the technical specifications. The operating voltage of the JC-310-JM is classified as SELV (Safety Extra Low Voltage). Therefore, the JC-310-JM is not subject to the EU Low Voltage Directive.

Usage other than intended

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

The JC-310-JM 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 different 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 electrostatic 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 module

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.

The installation of such parts may impair the safety and the proper functioning of the device.

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 JC-310-JM contains electrostatically sensitive components which can be damaged if not handled properly. The JC-310-JM is an optional part of the servo amplifier JM-2xx. Therefore, ship the JC-310-JM only if it is installed in a servo amplifier JM-2xx (especially if it is shipped by mail).

Thus, the following information on shipment refer the servo amplifier JM-2xx:

- Use an appropriate outer packaging to protect the JM-2xx against impact or shock.
- In case of damaged packaging inspect the JM-2xx for any visible damage. Inform your freight forwarder and Jetter AG.

Repair and maintenance

The operator is not allowed to repair the device. The device does not contain any parts that could be repaired by the operator.

If the device needs repairing, please send it to Jetter AG.

Replacement

Due to the hazardous DC link voltage the customer must not open the JM-2xx. The JC-310-JM must not be replaced or retrofitted to the JM-2xx.

Consult your contact person at Jetter AG.

Decommissioning and disposing of the device

When decommissioning and disposing of devices on the operating company's premises, the environmental regulations of the respective country must be complied with.

Since the JC-310-JM is an integral part of the servo amplifier JM-2xx, it must be disposed of along with it.

Residual dangers and protective measures

Residual dangers

Consider the residual dangers mentioned in this chapter when assessing the risks associated with your machine.

⚠	**DANGER**
⚠	**Hazard in explosive gas atmosphere!** This device can become a source of ignition in potentially explosive atmospheres. ➢ **Do not use this device in potentially explosive atmospheres.**

⚠	**DANGER**
⚠	**Hazard caused by high operating voltage!** **Risk of electric shock due to residual voltage in the capacitors of the servo amplifier!** The high operating voltage may cause muscle cramps, burns, unconsciousness, respiratory standstill, or death. Capacitors of the servo amplifier can store dangerous voltages for at least 7 minutes after the power has been switched off. ➢ Do not remove any cover plates and keep all control cabinet doors closed during operation. ➢ Wait at least 7 minutes after switching off, before you disconnect any terminal leads. ➢ Do not open the device. ➢ **While the device is in operation, do not touch the following terminals: Power supply, motor voltage and DC link voltage.**

2 Product description and design

Introduction

This chapter covers the design of the device and provides an overview of existing documents.

Contents

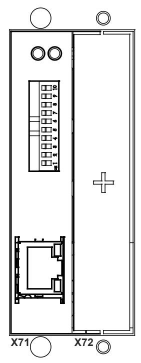
Product description of the JC-310-JM

The controller module JC-310-JM

The controller module JC-310-JM is integrated into a servo amplifier of the JM-2xx series and adds an Ethernet interface to it. This Ethernet interface lets you connect the servo amplifier to Jetter's Ethernet system bus and thus integrate it as an axis into a path group.

Product features

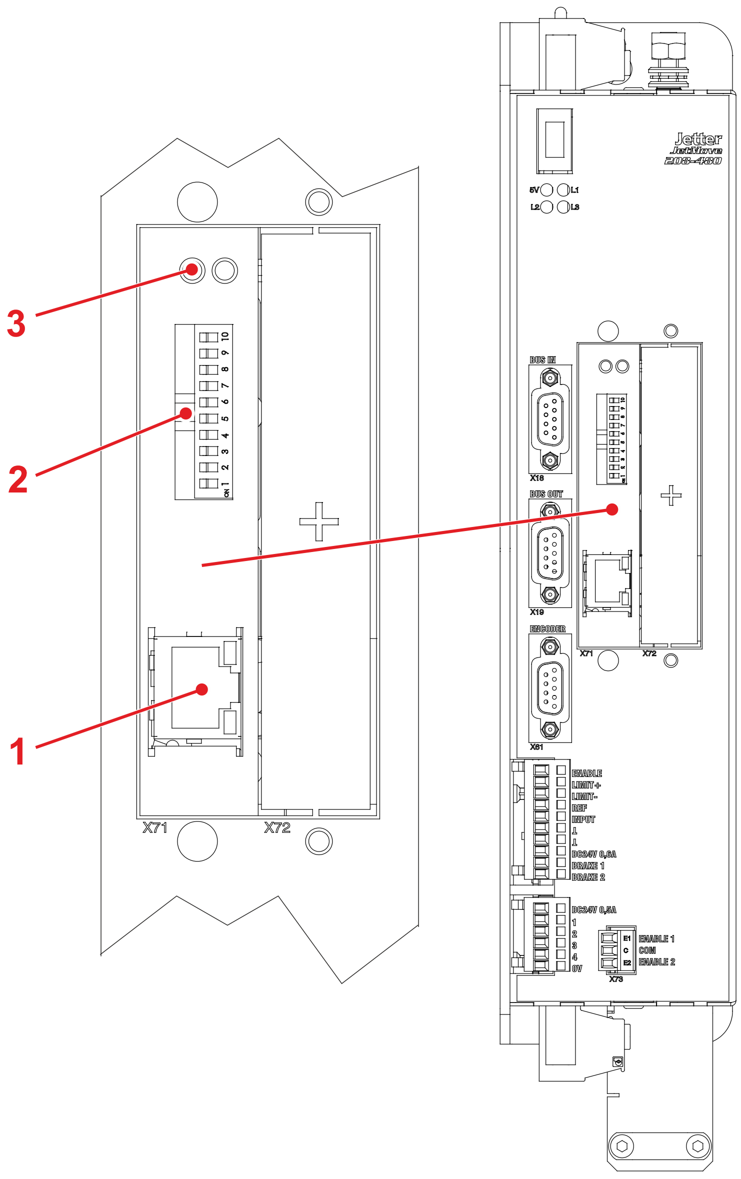
The features of this product are listed below:



- 1 Ethernet port
- Powerful programming language JetSym STX
- Volatile registers (%VL): 30,000
- Program/data memory: 2 MB
- Modbus/TCP

Parts and interfaces

Parts and interfaces The JC-310-JM features the following parts and interfaces:



Number	Part	Description
1	X71	Ethernet port
2	DIP switch	Sliders 1 through 8 IP address Sliders 9 and 10 RUN-STOP-LOAD
3	LED	Diagnostic and status LEDs

List of documentation

Introduction

Various documents and software tools support you in engineering and programming the JC-310-JM. You can download these documents and software tools from the Jetter AG **homepage http://www.jetter.de**.

Engineering

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

📄	**Industrial automation catalog**
	▪ Product description
	▪ Technical specifications
📄	**User manual on the JC-310-JM**
	▪ The document at hand
📄	**User manual on the servo amplifier JM-2xx**
	▪ Physical dimensions
	▪ Mechanical installation
	▪ Environmental and EMC conditions
	▪ Options
💾	**CAD data on the JC-310-JM**
	▪ dxf file with 2D illustrations
	▪ stp file with 3D illustrations

Programming

The following document and software tool support you at programming the JC-310-JM:

📄	**User manual on the JC-310-JM**
	▪ The document at hand
💾	**JetSym**
	▪ Programming tool

3 Identification

Purpose of this chapter

This chapter supports you in retrieving the following information about the JC-310-JM:

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

Prerequisites

To be able to identify the JC-310-JM the following prerequisites must be fulfilled:

- The JC-310-JM is connected to a PC.
- The programming tool JetSym 5.2 or higher is 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-310-JM ready:

- Serial number
- OS version
- Hardware revision

Contents

3.1 Electronic Data Sheet EDS

Introduction

Each JC-310-JM features an Electronic Data Sheet (EDS). Numerous production-relevant data are permanently stored in the EDS. The EDS data can be read out via files in the file system of the JC-310-JM or via special registers.

Contents

EDS file

Introduction

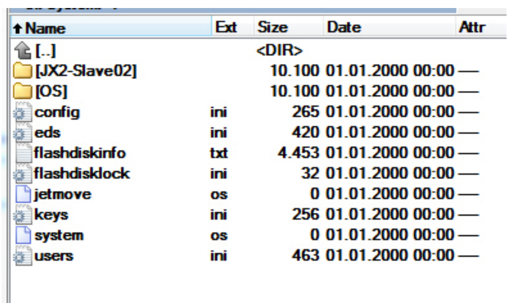
EDS data can be retrieved from the file **eds.ini**.

Properties

- You can access this file through the file system on the JC-310-JM.
- For an FTP connection, the user needs administrator rights (user *admin*) or system rights (user *system*).
- The EDS file of the JC-310-JM is located in the folder **System**.
- This file is read-only.
- The EDS file will even survive formatting of the flash disk drive.

Path to EDS files

The illustration below shows an example of the contents of the directory **System** holding the EDS files of the JC-310-JM:



↑Name	Ext	Size	Date	Attr
[..]		<DIR>		
[JX2-Slave02]		10.100	01.01.2000 00:00	—
[OS]		10.100	01.01.2000 00:00	—
config	ini	265	01.01.2000 00:00	—
eds	ini	420	01.01.2000 00:00	—
flashdiskinfo	txt	4.453	01.01.2000 00:00	—
flashdisklock	ini	32	01.01.2000 00:00	—
jetmove	os	0	01.01.2000 00:00	—
keys	ini	256	01.01.2000 00:00	—
system	os	0	01.01.2000 00:00	—
users	ini	463	01.01.2000 00:00	—

File structure

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

Example

This is an example of an EDS file belonging to a JC-310-JM:

```
;Jetter AG Electronic Data Sheet

[IDENTIFICATION]
Version = 2
Code = 54861
Name = JC-310-JM_01
PcbRev = 02
PcbOpt = 00
OSVersionMin = 0.0.0.0
BLVersionMin = 0.0.0.0

[PRODUCTION]
Version = 0
SerNum = 20101231884321
Day = 26
Month = 3
Year = 2015
TestNum = 5
TestRev = 01.02.03.04

[FEATURES]
Version = 1
MAC-Addr = 00:50:CB:01:C1:97
Switch = 1
STX = 1
CAN = 1
Axes = -1
Web = 1
ModbusTCP = 1
```

Section [IDENTIFICATION]

The general hardware configuration can be seen from section [IDENTIFICATION].

Name	Example	Description
Version	2	Version of this section
Code	54861	Module code of JC-310-JM
Name	JC-310-JM_01	Corresponds to the information on the nameplate
PcbRev	02	Hardware revision
PcbOpt	00	Hardware option
OSVersionMin	0.0.0.0	The JC-310-JM is available as of this OS version
BLVersionMin	0.0.0.0	The product is available as of this bootloader version

Section [PRODUCTION]

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

Name	Example	Description
Version	0	Version of this section
SerNum	20101231884321	Corresponds to the information on the nameplate
Day	26	Production date: Day
Month	03	Production date: Month
Year	2015	Production date: Year
TestNum	5	Internal usage
TestRev	01.02.03.04	Internal usage

Section [FEATURES]

In section [FEATURES] special properties of the JC-310-JM can be seen. The OS of the JC-310-JM will ignore properties of missing entries in the file.

Name	Example	Description
Version	1	Version of this section
MAC Addr	00:50:CB:01:C1:97	Ethernet MAC address
Switch	1	A switch (mode selector) RUN/STOP/LOAD is available
STX	1	Runtime environment for the application program is available
CAN	1	Bus interface for JX2 modules is available
Axes	-1	Number of supported JX2 axis modules
Web	1	Web server and e-mail client are available
ModbusTCP	1	Modbus/TCP client and server are available

Related topics

- **EDS registers** (see page 22)

EDS registers

Introduction

EDS registers let you retrieve entries in 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.

Device	Basic register number	Register numbers
JC-310-JM	100000	100500 ... 100817

Readable data

The following table lists the EDS registers of a JC-310-JM, as well as their connection to the entries in the EDS file **/System/eds.ini**. By means of this register array, you can have the EDS of the JC-310-JM displayed. To this end, select *CPU* in module register 500. The contents of the selected EDS are then displayed in the following registers.

Registers	Section in the EDS file	Name in the EDS file	Description
MR 500	-	-	Functional group: 0: CPU
MR 501	-	-	Module number (0) (if MR 500 > 0)
MR 600	IDENTIFICATION	Version	Version of this section
MR 601		Code	Module code
MR 602 through 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 through 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

Registers	Section in the EDS file	Name in the EDS file	Description
MR 800	FEATURES	Version	Version of this section
MR 801		MAC Addr	MAC address (manufacturer section)
MR 802		MAC Addr	MAC address (device section)
MR 804		Switch	Mode selector RUN/STOP/LOAD
MR 805		STX	Runtime environment for the application program
MR 808		CAN	Bus interface for JX2 modules
MR 810		MotionControl	MC software
MR 811		Axes	Number of supported JX2 axis modules
MR 812		Web	Web server and e-mail client
MR 813		ModbusTCP	Modbus/TCP client and server

Related topics

- **EDS file** (see page 19)

3.2 Version registers

Introduction

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

Contents

Hardware revisions

Introduction

The JC-310-JM features special registers, the content of which lets you identify the hardware.

Registers - Overview

The following registers let you read out the hardware revisions:

Register	Description
108021	Hardware revision - CPU board
200170	Controller type

Related topics

- **Software versions** (see page 26)

Software versions

Introduction

The JC-310-JM features software with unique version numbers which can be retrieved via special registers.

Format of software version numbers

The software version number of the JC-310-JM is a four-figure value.

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 following registers let you read out the software versions:

Registers	Description
200168	Boot loader version
200169	Operating system version
210001	Version of the execution unit for the STX application program

Related topics

- **Hardware revisions** (see page 25)

4 Mechanical and electrical installation

Purpose of this chapter

This chapter describes how to install the JC-310-JM and covers the following topics:

- Wiring the JC-310-JM
- Description of indicators
- Description of control elements
- IP configuration

Mechanical installation

The JC-310-JM is **always** integrated into a servo amplifier JM-2xx. For information on how to install the servo amplifier JM-2xx, refer to the corresponding manual.

You must not install or remove the JC-310-JM as hazardous operating and DC link voltages are present in the servo amplifier.

DANGER

Hazard caused by high operating voltage!

Risk of electric shock due to residual voltage in the capacitors of the servo amplifier!

The high operating voltage may cause muscle cramps, burns, unconsciousness, respiratory standstill, or death.

Capacitors of the servo amplifier can store dangerous voltages for at least 7 minutes after the power has been switched off.

- Do not remove any cover plates and keep all control cabinet doors closed during operation.
- Wait at least 7 minutes after switching off, before you disconnect any terminal leads.
- Do not open the device.
- **While the device is in operation, do not touch the following terminals: Power supply, motor voltage and DC link voltage.**

Contents

4.1 Ports and interfaces

Port X71

Port X71 is an Ethernet interface and lets you connect the following devices:

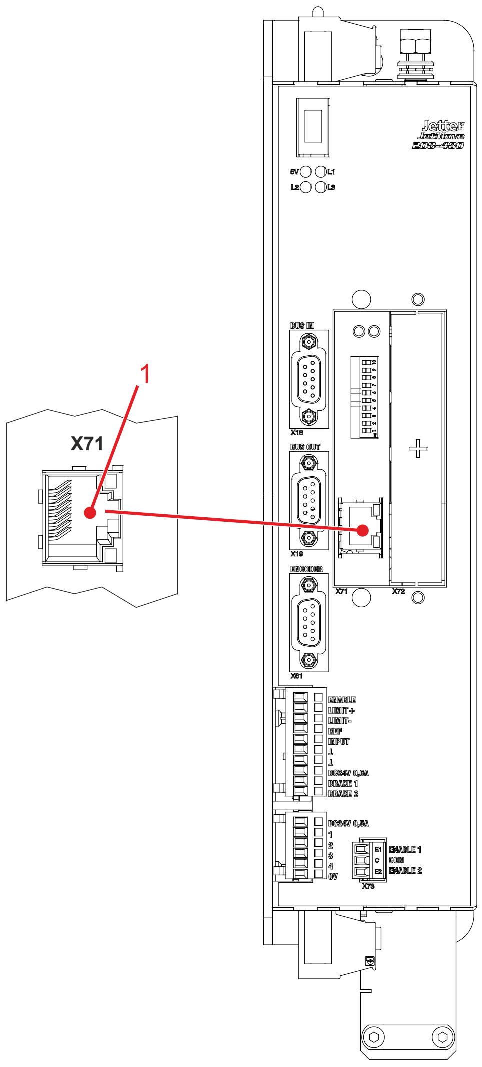
- Hub, switch or router
- PC
- HMI by Jetter AG
- JX3-BN-ETH or JX3-COM-xxxx
- Any device

Contents

Ethernet interface - Port X71

Pin assignment

Port X71 is internally connected to an Ethernet switch.



Number	Description
1	Connector X71 - Ethernet port

Technical specifications

Parameter	Description
Type of connector	RJ45 Ethernet jack
Number of ports	1 port
Bit rate	10 MBit/s, 100 MBit/s (Cat 5e)
Auto-crossover	Yes

Port X71 - Connecting cable

For connecting devices to port X71 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

4.2 DIP switches, indicators and LEDs

Control elements

The JC-310-JM has got a DIP switch. This DIP switch lets you make the following settings:

- Sliders 1 through 8 IP address
- Sliders 9 and 10 RUN-STOP-LOAD

Indicators

The JC-310-JM is equipped with the following LEDs:

- 2 LEDs indicating conditions and errors
- 2 LEDs indicating the condition of the Ethernet port

Contents

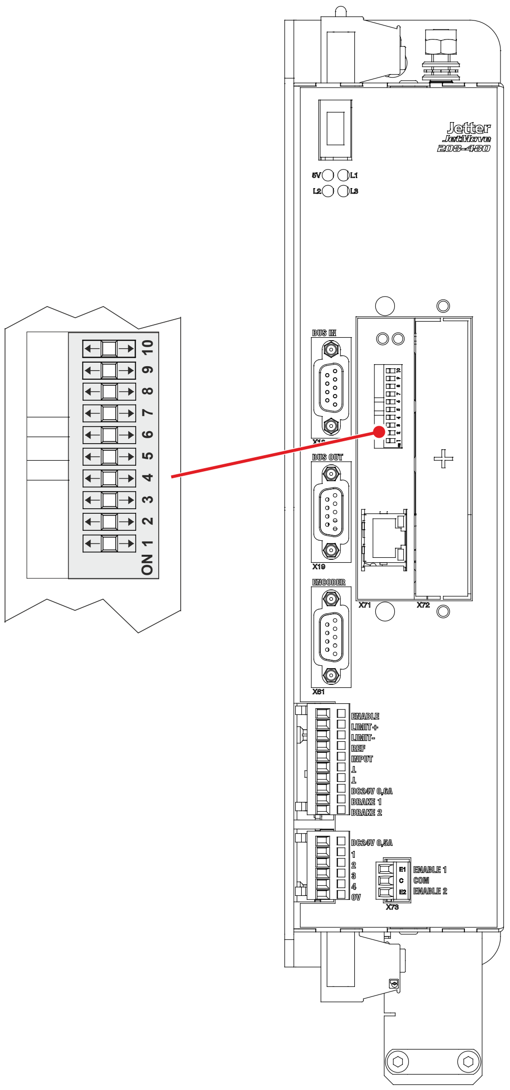
DIP switches - JC-310-JM

Loading settings

When the device is powered-up, the position of the DIP switch sliders is scanned and the corresponding settings are loaded. Once the boot process is completed, any changes to the DIP switches will not affect the actual configuration.

DIP switch - Slider positions

The illustration below shows the position of the DIP switch sliders.



Function of DIP switch sliders 1 through 8

DIP switch sliders 1 through 8 let you set the IP address. The slider positions let you select the following three ways of setting the IP address:

- Setting the default IP address 192.168.10.15
- Setting the IP address via **config.ini** file
- Setting the IP address via **config.ini** file and DIP switch

The links given in *Related Topics* refer you to the chapters where you find a detailed description of DIP switch sliders 1 through 8.

Function of DIP switch sliders 9 and 10

DIP switch sliders 9 and 10 let you configure how operating system and application program are launched.

The table below shows the possible functions and the corresponding positions of DIP switch sliders 9 and 10:

Position of DIP switch slider 9	Position of DIP switch slider 10	Function	Description
OFF	**ON**	LOAD	The boot process proceeds as far as to the boot loader.
ON	**ON**	STOP	The application program will not be launched.
ON	**OFF**	RUN	The application program will be launched.

R 108015

Set function

Register 108015 lets you read-out the function set by DIP switch sliders 9 and 10.

Module register properties	
Values	1 = LOAD
	2 = RUN
	3 = STOP
Type of access	Read

Related topics

- **Setting the default IP address 192.168.10.15** (see page 48)
- **Setting the IP address via configuration file** (see page 48)
- **Setting the IP address via configuration file and DIP switch** (see page 49)

LEDs of JC-310-JM

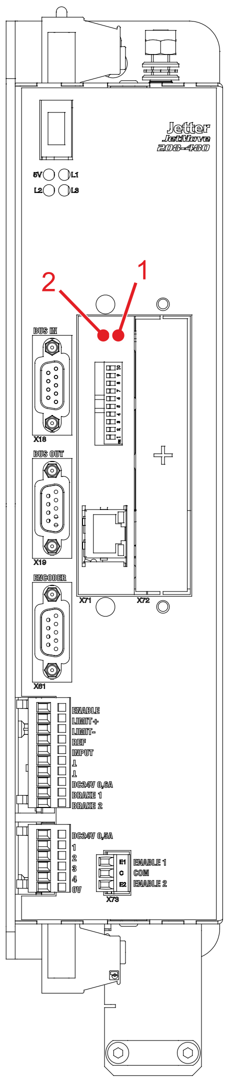
LEDs of JC-310-JM

The JC-310-JM indicates conditions and errors via its LEDs. The LEDs are split into 2 groups:

- Status messages of the operating system
- Status of the Ethernet port

OS status LEDs

The illustration below shows the OS status LEDs.



Number	OS LED	Color	Description
1	RUN	Green	OS is running
2	ERR	Red	Generic error

Normal operating condition

In normal operating condition, the OS status LEDs indicate the following:

RUN	ERR	State
🟢 ON	○ OFF	Normal operating condition ■ Operating system of JC-310-JM is running. Ethernet interface is working properly ■ Application program is running ■ No error

Indications of OS status LEDs

The following table shows possible state of the OS status LEDs RUN and ERR:

RUN	ERR	State
🟢 ON	○ OFF	No error, communication is active
🟢 1Hz	🔴 1Hz	If DIP switch slider 9 = OFF and 10 = ON (function LOAD) ■ Boot loader is running ■ OS will neither be checked or launched
🟢 1Hz	🔴 1Hz	If DIP switch slider 9 = ON (function RUN or STOP) ■ Boot loader is running ■ There is no valid OS
🟢 ON	🔴 ON	Errors For an error description refer to register 200008

LED indications during boot process

Normal boot process

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

- DIP switch slider 9 is *ON*, and slider 10 is *OFF*.
- There is a valid OS.
- There is a valid application program.
- The application program will be launched.

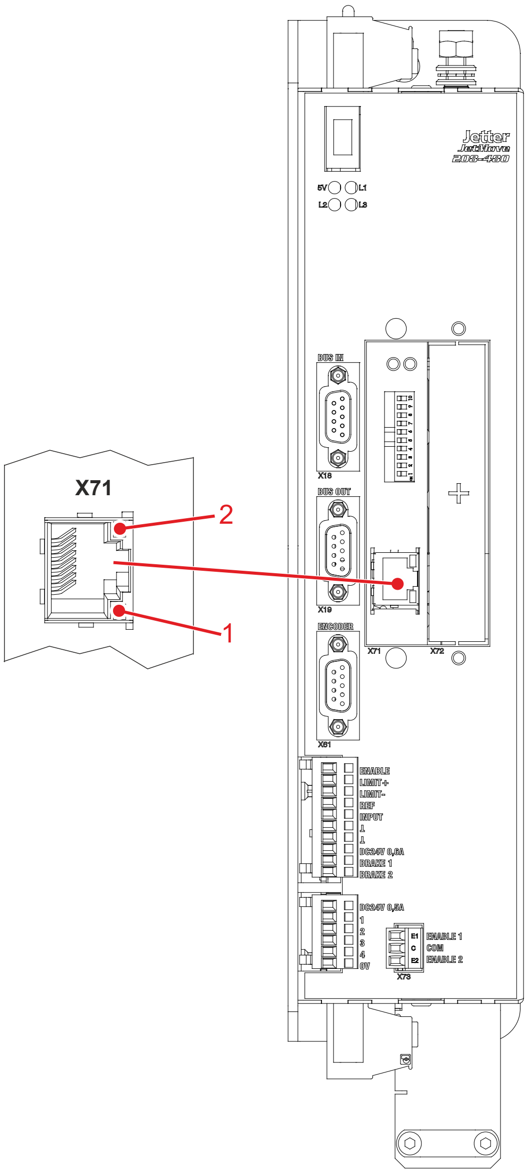
During the boot process, the OS status LEDs indicate the following:

Step	RUN	ERR	Description
1	✳ 1Hz	○ OFF	Boot loader is checking the validity of the OS
2	○ OFF	○ OFF	The OS is launched in the RAM
3	✳ 1Hz	● ON	The OS starts system initialization
4a	✳ 1Hz	○ OFF	If DIP switch slider 9 = *ON* and 10 = *ON* (function STOP): OS is okay; application program is not being executed.
4b	✳ 1Hz	● ON	If DIP switch slider 9 = *ON* and 10 = *OFF* (function RUN): OS is okay; but there is no valid application program to be executed.
4c	● ON	○ OFF	If DIP switch slider 9 = *ON* and 10 = *OFF* (function RUN): OS is okay; 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	Color	Description
X71-1	Green	LINK: Network connection has been established
X71-2	Amber	ACT: Data transmission

4.3 IP configuration

Introduction

This chapter covers the IP configuration for the JC-310-JM. The following parameters can be set:

- IP address of JC-310-JM
- Subnet mask
- IP address of default gateway
- IP address of DNS server
- Name of JC-310-JM
- IP port number for the JetSym debugger
- Basic port number for communication via JetIP

Required skills of the network configurator

To carry out the IP configuration of the JC-310-JM, knowledge of IP networks is required, such as

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

Contents

Factory settings

Introduction

Before the JC-310-JM is shipped, various parameters are set to a certain value.

These parameters can be modified by the user.

Factory settings

Parameter	Value
IP address of JC-310-JM	192.168.10.15
Subnet mask	255.255.255.0
IP address of default gateway	0.0.0.0
IP address of DNS server	0.0.0.0
Name of JC-310-JM	JC-310-JM
IP port number for debugger	52000
IP port number for JetIP	50000
DIP switch slider	All DIP switch sliders = OFF
Administrator password	admin
System password	system

The configuration memory

Introduction

During the boot process, the JC-310-JM retrieves the parameters for initializing the IP interface from the configuration memory. You can access the data stored in the configuration memory in the following ways:

- A file located in the system directory of the file system lets you read out and modify configuration data.
- Registers let you read out application data.

Enabling conditions

The JC-310-JM reads data located in the configuration memory only during the boot process. That is, when you have made changes to the configuration memory, you must reboot the JC-310-JM for these changes to become effective.

Default values

Before data from the configuration memory are used, the JC-310-JM checks them for plausibility. If entries are invalid or absent, the JC-310-JM uses the following default values:

Parameter	Default value
IP address of JC-310-JM	192.168.10.15
Subnet mask	255.255.255.0
IP address of default gateway	0.0.0.0
IP address of DNS server	0.0.0.0
Name of JC-310-JM	JC-310-JM
Suffix type of the name	0
IP port number for debugger	52000
IP port number for JetIP	50000

Related topics

- **Configuration file** (see page 41)
- **Configuration registers** (see page 45)

The configuration file

Introduction

The configuration file **config.ini** lets you access the configuration memory on the JC-310-JM.

Properties

- You can access this file through the file system on the JC-310-JM.
- 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.
- Formatting the Flash disk drive leaves the file unchanged.

File structure

The configuration file is a text file the entries of which are grouped into several sections. The JC-310-JM replaces missing IP configuration parameters by their default values.

Configuration file - Example

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

```
;JetControl System Configuration
;Copyright (c) 2008 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       = JC-310-JM

[PORTS]
JetIPBase = 50000
JVMDebug  = 52000
```

Section [IP]

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

Address	
In the given example	192.168.50.1
Function	IP address of JC-310-JM; The DIP switch can overwrite the least significant byte.
Allowed values	■ > 1.0.0.0 ■ < 223.255.255.255
Illegal values	■ Network address ■ Broadcast address
In the event of an illegal value	The JC-310-JM resets all four values to their defaults.
SubnetMask	
In the given example	255.255.255.0
Function	Subnet mask
Allowed values	■ >= 128.0.0.0
Illegal values	■ 1 and 0 mixed
In the event of an illegal value	The JC-310-JM resets all four values to their defaults.
DefGateWay	
In the given example	192.168.50.11
Function	IP address of the gateway to other subnets; The JC-310-JM must be able to reach the subnet (Address/SubnetMask), otherwise it will set this parameter to 0.0.0.0.
Allowed values	■ >= 0.0.0.0 and ■ < 223.255.255.255
Illegal values	■ Network address ■ Broadcast address ■ A value (Address/SubnetMask) which cannot be reached by the JC-310-JM. ■ The "Address" value
In the event of an illegal value	The JC-310-JM sets the value to 0.0.0.0
DNSServer	
In the given example	192.168.1.44
Function	IP address of the server for the Domain Name System
Allowed values	■ >= 0.0.0.0 and ■ 223,255,255,255
In the event of an illegal value	The JC-310-JM sets the value to 0.0.0.0

Section [HOSTNAME]

In section [HOSTNAME] the name of the JC-310-JM is specified. If desired, the JC-310-JM automatically generates an individual name. The JC-310-JM presently does not use the host name.

SuffixType	
In the given example	0
Function	The type of the automatically generated suffix is attached to the name of JC-310-JM
Allowed values	■ 0: No attachment ■ 1: Low-order byte of the IP address in decimal notation ■ 2: Low-order byte of the IP address in hexadecimal notation
In the event of an illegal value	0

Name	
In the given example	JC-310-JM
Function	Specifies the name of the JC-310-JM
Allowed values	■ First character: 'A' ... 'Z', 'a' ... 'z' ■ Next characters: 'A' ... 'Z', 'a' ... 'z', '0' ... '9', '-'
In the event of an illegal value	JC-310-JM

Section [PORTS]

In section [PORTS] the IP port numbers of data and debug servers within the JC-310-JM are specified. The IP port numbers must be consistent with, for example, the port numbers set in JetSym.

JetIPBase	
In the given example	50000
Function	IP port for OS update and communication between devices
Allowed values	■ 1024 ... 65535
In the event of an illegal value	50000

JVMDebug	
In the given example	52000
Function	IP port for debugger/setup in JetSym
Allowed 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 JC-310-JM.
3	Log in as user with administrator or system rights. Default login information: User: admin; Password: admin
4	Browse to directory */System* of the JC-310-JM.
5	Copy the configuration file *config.ini* you created to the JC-310-JM.
6	Clear the FTP connection.
7	Reboot JC-310-JM.

Result: The new configuration is active.

Related topics

- **The configuration memory** (see page 40)
- **Configuration registers** (see page 45)

Configuration registers

Introduction

The configuration registers let you read and modify IP configuration parameters. A range of registers holds the data contained in the configuration memory. 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.

Device	Data range	Basic register number	Register numbers
JC-310-JM	Configuration memory	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 JC-310-JM
MR 1		`SubnetMask`	Subnet mask
MR 2		`DefGateWay`	IP address of the gateway to other subnets
MR 3		`DNSServer`	IP address of the server for the Domain Name System
MR 32	HOSTNAME	`SuffixType`	Type of the automatically generated suffix that is attached to the name of the JC-310-JM
MR 33 through 51		`Name`	Specifies the name of the JC-310-JM
MR 64	PORTS	`JetIPBase`	IP port number for OS update and communication between controllers
MR 65		`JVMDebug`	IP port number for debugger/setup in JetSym

Related topics

- **The configuration memory** (see page 40)
- **The configuration file** (see page 41)

Changing the IP address of JC-310-JM

Introduction

To be able to communicate with the JC-310-JM via Ethernet, you must set an unambiguous IP address on the JC-310-JM.

Configuration options

You can configure the IP address in the following ways:

- Using the default IP address
- Configuration via file **config.ini**
- Configuration via file **config.ini** and DIP switch
- Configuration during runtime via special registers

Changing the IP address

Step	Action
1	Remove power from the JC-310-JM.
2	Make the corresponding DIP switch settings.
⇨	Following restart, the JC-310-JM can be reached at the new IP address.

Related topics

- **Setting the default IP address 192.168.10.15** (see page 47)
- **Setting the IP address via configuration file** (see page 48)
- **Setting the IP address via configuration file and DIP switch** (see page 49)
- **Setting the IP address during runtime** (see page 51)

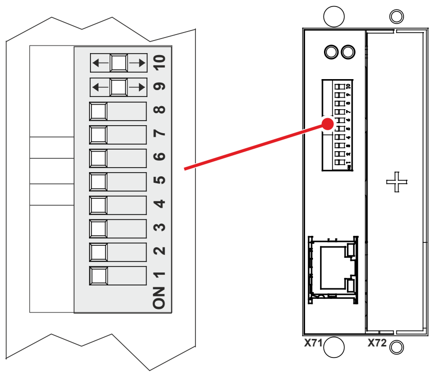
Setting the default IP Address 192.168.10.15

Default IP address

The default IP address of the JC-310-JM is 192.168.10.15. You may change the IP address of the JC-310-JM to its default IP address at any time.

DIP switch slider settings

Move sliders 1 through 8 on the DIP switch to OFF to set the default IP address 192.168.10.15:



Setting the IP address via configuration file

Setting the IP address

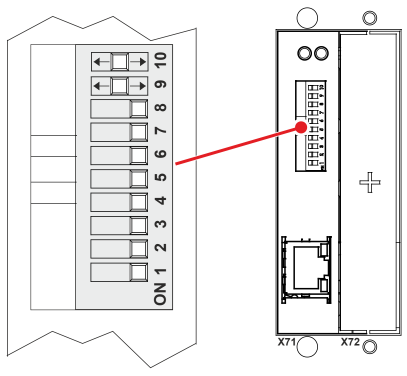
You can set the IP address of the JC-310-JM in the **config.ini** file.

```
[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

DIP switch slider settings

The following DIP switch settings cause the JC-310-JM to read out the IP address from the file **config.ini**:



Transferring the configuration file

Step	Action
1	Establish an FTP connection to the JC-310-JM.
2	Log in as user with administrator or system rights. Default login information: User: *admin*; Password: *admin* 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	Restart the JC-310-JM.

Setting the IP address via configuration file and DIP switch

Introduction

The JC-310-JM lets you set its IP address in the **config.ini** file or through the DIP switch sliders. To this end, set the three upper bytes of the IP address in the**config.ini** file, and the lower byte using the DIP switch sliders.

Configuration file - Setting the IP address

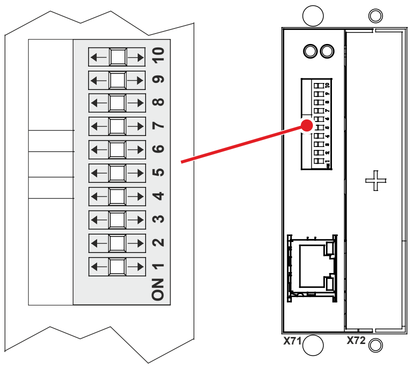
Set the upper three bytes of the IP address of the JC-310-JM in the configuration file **config.ini**.

```
[IP]
Address    = aaa.bbb.ccc.1
```

Element	Description
Address	Line for entering the upper three bytes of the IP address
aaa	First byte of IP address
bbb	Second byte of IP address
ccc	Third byte of IP address
1	Dummy entry - must be 1

DIP switch slider settings

The following DIP switch settings cause the JC-310-JM to read out the IP address from the file **config.ini** and the DIP switch sliders:



DIP switch slider								IP address
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	
OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	Default IP address
ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	aaa.bbb.ccc.1
OFF	**ON**	OFF	OFF	OFF	OFF	OFF	OFF	aaa.bbb.ccc.2
ON	**ON**	OFF	OFF	OFF	OFF	OFF	OFF	aaa.bbb.ccc.3
...								
OFF	**ON**	**ON**	**ON**	**ON**	**ON**	**ON**	**ON**	aaa.bbb.ccc.254
ON	**ON**	**ON**	**ON**	**ON**	**ON**	**ON**	**ON**	From config.ini

Transferring the configuration file

Step	Action
1	Establish an FTP connection to the JC-310-JM.
2	Log in as user with administrator or system rights. Default login information: User: *admin*; Password: *admin* 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	Restart the JC-310-JM.

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 JC-310-JM
- 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 JC-310-JM, your settings will be lost.

Exception:

If you save the currently used parameters to the configuration file **config.ini** by entering password 0x77566152 into R 101299, these settings are then permanently stored.

Prerequisites

- While settings are being made, no communication via IP interface is allowed. Otherwise, this would lead to a loss of data.
- The entered values must be valid.
  This is important because there is no check if you set the parameters during runtime of the JC-310-JM.

Registers - Overview

Register	Description
104531	IP address of JC-310-JM
104532	Subnet mask
104533	IP address of default gateway
101299	If you enter password 0x77566152 into this register, the currently used parameters of the IP interface are permanently saved to the **config.ini** file. This registers returns the following values: 1: During the saving process 0: After a successful saving process -1: In case of an error during the saving process

Setting IP addresses and subnet mask

To set the IP address and the subnet mask, 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 the desired IP address into R 104531.
4	Enter the desired subnet mask into R 104532.
5	Enter the desired IP address of the default gateway into R 104533.

Result:
The settings are completed. Communication is possible again.

Related topics

- **The configuration memory** (see page 40)

IP address in the GNN operating mode

Introduction

In GNN mode (Global Network Number mode), the JC-310-JM functions as a network node within a greater controller network. It derives its IP address at booting from the configuration file **config.ini**.

Next, the main controller, which is the NetConsistency master, commands the network node to log in with the GNN from the NetConsistency master. After accepting the network node, the main controller compares the set IP configuration with the actual IP configuration of the network node. If a difference results, the main controller makes the corresponding changes in the set IP configuration in the network node.

How does the JC-310-JM enter into GNN mode?

During the boot process, the JC-310-JM retrieves the DIP switch slider positions and sets the IP address accordingly. If a NetConsistency master retrieves the slider positions, they are interpreted as GNN. Then, the master is able to assign a new IP address.

The module itself does not feature a function for switching between IP or GNN.

Configuration file - Setting the IP address

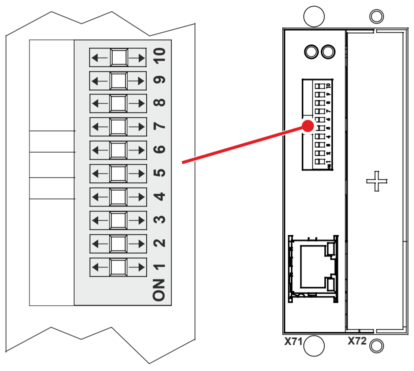
The configuration file **config.ini** lets you set the IP address.

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

Element	Description
`Address`	Line for entering the IP address
`aaa`	First byte of IP address
`bbb`	Second byte of IP address
`ccc`	Third byte of IP address
`ddd`	Fourth byte of IP address

DIP switch slider settings

The following DIP switch settings cause the JC-310-JM to retrieve the IP address from the file **config.ini** and the GNN out of the lower eight positions of the DIP switch sliders:



Valid values for the GNN: 1 ... 199.

The DIP switches are binary-coded.

DIP switch slider								GNN
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	
OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	Invalid
ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	1
OFF	**ON**	OFF	OFF	OFF	OFF	OFF	OFF	2
ON	**ON**	OFF	OFF	OFF	OFF	OFF	OFF	3
...								
ON	**ON**	**ON**	OFF	OFF	OFF	**ON**	**ON**	199
...								Invalid
ON	**ON**	**ON**	**ON**	**ON**	**ON**	**ON**	**ON**	Invalid

Using names for IP addresses

Introduction

The JC-310-JM lets you use names as IP addresses for target systems. The JC-310-JM 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 into IP addresses in the following way:

Step	Description	
1	During the boot process the JC-310-JM reads the IP address of the DNS server from the configuration memory.	
2	During the boot process, the JC-310-JM reads the file **/etc/hosts**. Then, the JC-310-JM creates a translation table with the names and IP addresses found in this file.	
3	After the boot process, the JC-310-JM detects a name instead of an IP address.	
4	Based on this translation table, the JC-310-JM tries to resolve the name into a related IP address.	
	If ...	**... then ...**
	... the JC-310-JM has revolved the name, ...	... proceed with step 6.
	... the JC-310-JM failed to resolve the name, ...	... proceed with step 5.
5	The JC-310-JM tries to resolve the name into a related IP address by sending a request to the DNS server.	
	If ...	**... then ...**
	... the JC-310-JM has revolved the name, ...	it enters the name and IP address into the translation table; --> proceed with step 6.
	... the JC-310-JM failed to resolve the name, ...	... the JC-310-JM aborts this process and issues an error message.
6	The JC-310-JM uses the IP address resolved for further communication.	

Configuration file

The configuration file **hosts** holds the static assignment between names and IP addresses. The JC-310-JM reads this file once during boot-up.

File format:	Text
Location:	/etc
File name:	hosts

Example

```
# Example - Hosts file for JC-3xx
192.168.33.209    jetter_mail
192.168.33.208    jetter_demo
192.168.1.1       JC340
192.168.1.2       JC350
```

Domain Name System (DNS)

If a name cannot be found in the file **/etc/hosts**, the JC-310-JM tries to resolve the IP address by obtaining the corresponding IP address from a DNS server. During boot-up, the JC-310-JM retrieves the IP address of the DNS server out of the configuration memory.

Related topics

- **The configuration memory** (see page 40)

5 File system

Introduction

This chapter describes the file system of the JC-310-JM. The file system lets you access files located on the internal flash disk.

File categories

The files of the file system are categorized as follows:

- System directories or system files used by the operating system
- Files accessible to the user

System directories

The system directories cannot be deleted. System directories even survive formatting.

Directory	Description
/System	▪ System configuration ▪ System information

Contents

5.1 Properties

Introduction

This chapter covers the properties of the file system.

General properties

The following conventions apply to the internal flash disk:

- 8 files max. to be opened simultaneously
- Separate directory names by a slash "/", not by a backslash "\".
- Files are marked with a time stamp. This time stamp indicates the time that has elapsed since the JC-310-JM has been started.

Contents

Flash disk - Properties

Capacity

The following disk space is available to the user:

Parameter	Value
Flash disk capacity	4 MB

Properties

The internal flash disk drive has got the following properties:

- Up to 7 directory levels and 1 file level are allowed.
- Differentiation between upper and lower case.
- Directory and file names with a length of up to 63 characters are possible.
- All characters except "/" and ".." are permitted for directory and file names
- User/access administration for a maximum number of 31 locks and 33 users.

5.2 User administration

Introduction

The file system for the internal flash disk lets you define authorization for access (locks) to directories, and set up users.

For each user, you can set individual access rights (keys).

Users are not allowed to access directories and files for which they do not have the required key. In case of an FTP/IP connection, these directories and files are not displayed.

Prerequisites

Administrator rights are required for user administration.

Properties

The properties of user administration are as follows:

Property	Max. value
Number of users	33
Number of predefined users	2
Length of a user name	31 alphanumeric characters
Password length	31 alphanumeric characters
Number of keys for read access	31
Number of keys for write access	31
Number of predefined keys	2

Files

You can make settings for user administration in three files located in the directory **System**:

File	Description
flashdisklock.ini	Assignment of locks to directories
keys.ini	Assignment of names to locks/keys
users.ini	Administration of users

These files are always existing. They cannot be deleted, but only modified or overwritten.

Restrictions

Please take the following restrictions into account:

- User administration can only be applied to the internal flash memory.
- If user administration has been assigned to a file, its contents are readable at once. These settings become active only after a reboot.

Contents

Administration of users

Introduction

The configuration file **/System/users.ini** lets you manage the user administration for the file system.

Prerequisites

If you want to use names for the keys, you must make them known to the device beforehand. Therefore, set up the names first as described in *Setting up names for keys/locks* (see page 67).

Administration of users

To manage user administration, proceed as follows:

Step	Action
1	Establish an FTP connection to the device. Log on as administrator.
2	Open the file **/System/users.ini**.
3	Enter the required information.
4	Save the changed file to the device.
5	Reboot the device.

Result: The changed user administration settings are now enabled.

Structure of the configuration file

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

- For each user a separate section is to be created.
- In these sections values can be set which are then used by the file system.
- You can insert blank lines as required.
- The following characters precede a comment line: "!", "#" or ";".

Sections

The sections are named *[USER1]* through *[USER33]*. Here, the user name and the related password, as well as read and write permissions are specified.

Example:

```
[USER4]
NAME=TestUser3
PW=testpass
READKEYS=5,openLock2,10,11
WRITEKEYS=openLock2,10,11
SYSKEYS=
```

NAME	
In the given example	TestUser3
Description	User's login name
Allowed values	A maximum of 31 alphanumeric characters
In case of illegal value or missing entry	User administration settings are not made
PW	
In the given example	testpass
Description	User's login password
Allowed values	A maximum of 31 alphanumeric characters
In case of missing entry	The user is allowed to log in without password
READKEYS	
In the given example	5,openLock2,10,11
Description	Keys for read access (read keys)
Allowed values	1 ... 31 (or corresponding names)
In case of missing entry	No read keys are assigned to the user
WRITEKEYS	
In the given example	openLock2,10,11
Description	Keys for write access (write keys)
Allowed values	1 ... 31 (or corresponding names)
In case of missing entry	No write keys are assigned to the user
SYSKEYS	
Description	No function assigned; reserved for future extensions

As-delivered condition/Predefined users and keys

Introduction

Two predefined users with set rights are included in the file system. It is not possible to delete these two users. The user administration lets you only change the password for these two users.

Factory settings

In delivered condition the content of the configuration file included in the controller is as follows.

```
[USER1]
NAME=admin
PW=admin
READKEYS=1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,2
2,23,24,25,26,27,28,29,30,31
WRITEKEYS=1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,
22,23,24,25,26,27,28,29,30,31
SYSKEYS=

[USER33]
NAME=system
PW=system
READKEYS=2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,
23,24,25,26,27,28,29,30,31
WRITEKEYS=2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22
,23,24,25,26,27,28,29,30,31
SYSKEYS=
```

Predefined users

All keys are available to this user *admin*, and he/she is, therefore, able to read all directories and files and to write to them.

All keys except for key **1** are available to user *system*, too.

Predefined keys

Two out of the 31 keys have a predefined function.

Lock/key	Function
1	■ Ethernet configuration ■ User administration
2	■ Operating system update of the CPU ■ Operating system update of JX2 and JX3 modules

Assigning locks

Introduction

In the configuration file **/System/flashdisklock.ini** you assign locks to directories located on the flash disk. Only users with the corresponding key are allowed to read or write (delete) files and subdirectories located in these directories.

Prerequisites

If you want to use names for the locks, you must make them known to the device beforehand. Therefore, set up the names first *Setting up names for keys/locks* (see page 67).

Installing the lock

To assign a lock to a directory, proceed as follows:

Step	Action
1	Establish an FTP connection to the device; when doing so, log in with administrator rights.
2	Open the file **/System/flashdisklock.ini**.
3	Adjust the file entries.
4	Save the changed file to the device.
5	Reboot the device.

Result: A lock is assigned to this directory.

Structure of the configuration file

This configuration file is a text file containing one section.

- In this section values can be set which are then used by the file system.
- Specify each directory with its lock number in an individual line.
- You can insert blank lines as required.
- The following characters precede a comment line: "!", "#" or ";".

Section

The section is named *[LOCKS]*. Here, locks are assigned to directories in accordance with the following rule:

Directory=Lock

Example:

```
[LOCKS]
test1=0
test1/sub1=2
test1/sub2=5
test2=userlock2
```

Lock numbers

Use the following lock numbers:

- The valid lock numbers are 0 ... 31.
- Lock number 0: No lock is assigned to this directory. You can access this directory without any special permissions.
- You can use numbers or previously defined names.

Assigning names to keys/locks

Introduction

Keys/locks are consecutively numbered from 1 through 31. To provide ease of handling, a name can be assigned to each key/lock combination. These names are assigned in the configuration file **/System/keys.ini**.

Configuring names

To assign names to keys/locks, proceed as follows:

Step	Action
1	Establish an FTP connection to the device; when doing so, log in with administrator rights.
2	Open the file **/System/keys.ini**.
3	Adjust the file entries.
4	Save the changed file to the device.
5	Reboot the device.

Result:

The names are available now. The names are now available and can be used when assigning locks and managing user accounts.

Structure of the configuration file

This configuration file is a text file containing one section.

- In this section values can be set which are then used by the file system.
- Each key is specified with its name in an individual line.
- You can insert blank lines as required.
- The following characters precede a comment line: "!", "#" or ";".

Section

The section is named *[KEYS]*. Here, names are assigned to keys/locks in accordance with the following rule:

KEYxx=Name

xx: Number of the key (01 ... 31)

Example:

```
[KEYS]
KEY01=Admin
KEY02=System
KEY03=
KEY04=
KEY05=service
...
KEY31=
```

Names for Locks/Keys

For names the following definitions are true:

- A maximum of 15 alphanumeric characters
- Lock and key have the same name.

5.3 Reviewing the flash disk capacity used

Introduction

You can view the application scope of the internal flash disk.
Details on the allocation of the application scope are given in this chapter.

Contents

Flash disk capacity used

README

You can view the application data area of the internal flash disk.

You can see the capacity used of the application data area from the file **/System/flashdiskinfo.txt**.

Example

In this example, the fictive capacity used of a flash disk in a JetControl 340 (4 MB) is shown:

```
Name  : flash disk
Date  : 25.11.2008
Time  : 15:04
Tracks: 64

Track   0:  sectors: 128  (used:  81 / blocked:  47 / free:   0)
Track   1:  sectors: 128  (used: 128 / blocked:   0 / free:   0)
Track   2:  sectors: 128  (used: 128 / blocked:   0 / free:   0)
Track   3:  sectors: 128  (used: 128 / blocked:   0 / free:   0)
Track   4:  sectors: 128  (used: 128 / blocked:   0 / free:   0)
Track   5:  sectors: 128  (used: 128 / blocked:   0 / free:   0)
Track   6:  sectors: 128  (used: 128 / blocked:   0 / free:   0)
Track   7:  sectors: 128  (used: 128 / blocked:   0 / free:   0)
Track   8:  sectors: 128  (used: 128 / blocked:   0 / free:   0)
Track   9:  sectors: 128  (used: 128 / blocked:   0 / free:   0)
Track  10:  sectors: 128  (used: 128 / blocked:   0 / free:   0)
Track  11:  sectors: 128  (used: 128 / blocked:   0 / free:   0)
Track  12:  sectors: 128  (used: 128 / blocked:   0 / free:   0)
Track  13:  sectors: 128  (used: 128 / blocked:   0 / free:   0)
Track  14:  sectors: 128  (used: 128 / blocked:   0 / free:   0)
Track  15:  sectors: 128  (used: 128 / blocked:   0 / free:   0)
Track  16:  sectors: 128  (used: 128 / blocked:   0 / free:   0)
Track  17:  sectors: 128  (used: 128 / blocked:   0 / free:   0)
Track  18:  sectors: 128  (used: 128 / blocked:   0 / free:   0)
Track  19:  sectors: 128  (used: 128 / blocked:   0 / free:   0)
Track  20:  sectors: 128  (used:  64 / blocked:  64 / free:   0)
Track  21:  sectors: 128  (used:  85 / blocked:  43 / free:   0)
Track  22:  sectors: 128  (used: 128 / blocked:   0 / free:   0)
Track  23:  sectors: 128  (used: 128 / blocked:   0 / free:   0)
Track  24:  sectors: 128  (used: 128 / blocked:   0 / free:   0)
Track  25:  sectors: 128  (used: 128 / blocked:   0 / free:   0)
Track  26:  sectors: 128  (used: 128 / blocked:   0 / free:   0)
Track  27:  sectors: 128  (used: 128 / blocked:   0 / free:   0)
Track  28:  sectors: 128  (used: 128 / blocked:   0 / free:   0)
Track  29:  sectors: 128  (used: 128 / blocked:   0 / free:   0)
Track  30:  sectors: 128  (used: 128 / blocked:   0 / free:   0)
Track  31:  sectors: 128  (used: 128 / blocked:   0 / free:   0)
Track  32:  sectors: 128  (used: 128 / blocked:   0 / free:   0)
Track  33:  sectors: 128  (used: 105 / blocked:   0 / free:  23)
Track  34:  sectors: 128  (used:   0 / blocked:   0 / free: 128)
Track  35:  sectors: 128  (used:   0 / blocked:   0 / free: 128)
Track  36:  sectors: 128  (used:   0 / blocked:   0 / free: 128)
Track  37:  sectors: 128  (used:   0 / blocked:   0 / free: 128)
Track  38:  sectors: 128  (used:   0 / blocked:   0 / free: 128)
Track  39:  sectors: 128  (used:   0 / blocked:   0 / free: 128)
Track  40:  sectors: 128  (used:   0 / blocked:   0 / free: 128)
Track  41:  sectors: 128  (used:   0 / blocked:   0 / free: 128)
Track  42:  sectors: 128  (used:   0 / blocked:   0 / free: 128)
Track  43:  sectors: 128  (used:   0 / blocked:   0 / free: 128)
Track  44:  sectors: 128  (used:   0 / blocked:   0 / free: 128)
Track  45:  sectors: 128  (used:   0 / blocked:   0 / free: 128)
Track  46:  sectors: 128  (used:   0 / blocked:   0 / free: 128)
Track  47:  sectors: 128  (used:   0 / blocked:   0 / free: 128)
Track  48:  sectors: 128  (used:   0 / blocked:   0 / free: 128)
Track  49:  sectors: 128  (used:   0 / blocked:   0 / free: 128)
Track  50:  sectors: 128  (used:   0 / blocked:   0 / free: 128)
Track  51:  sectors: 128  (used:   0 / blocked:   0 / free: 128)
Track  52:  sectors: 128  (used:   0 / blocked:   0 / free: 128)
Track  53:  sectors: 128  (used:   0 / blocked:   0 / free: 128)
Track  54:  sectors: 128  (used:   0 / blocked:   0 / free: 128)
Track  55:  sectors: 128  (used:   0 / blocked:   0 / free: 128)
Track  56:  sectors: 128  (used:   0 / blocked:   0 / free: 128)
Track  57:  sectors: 128  (used:   0 / blocked:   0 / free: 128)
Track  58:  sectors: 128  (used:   0 / blocked:   0 / free: 128)
Track  59:  sectors: 128  (used:   0 / blocked:   0 / free: 128)
Track  60:  sectors: 128  (used:   0 / blocked:   0 / free: 128)
Track  61:  sectors: 128  (used:   0 / blocked:   0 / free: 128)
Track  62:  sectors: 128  (used:   0 / blocked:   0 / free: 128)
Track  63:  sectors: 128  (used:   0 / blocked:   0 / free: 128)

Total:  sectors: 8192  (used: 4175 / blocked: 154 / free: 3863)

Used   :  2120900 byte
Blocked:    78232 byte
Free   :  1962404 byte
Total  :  4161536 byte
```

Elements of info file

Tracks and sectors represent the administration units of the flash disk. The info file comprises the following elements:

Element	Description
`Name`	Dedicated name of the flash disk
`Date/Time`	Point in time when the flash disk was formatted last
`Tracks`	Total number of tracks
`Track xx: sectors: 128`	Assignment of sectors of a track
`Total: sectors:`	Overall statistical data of the sectors
`Used`	Total number of used bytes
`Blocked`	Total number of blocked bytes
`Free`	Total number of available bytes
`Total`	Total size of the flash disk

States of the sectors

The smallest administrative unit of the flash disk, i.e. the sector, may assume the following states:

State	Description
`Used`	The sector is occupied by data.
`Blocked`	The sector is no longer occupied, but can not yet be used due to administrative reasons.
`Free`	The sector is not occupied and can be used.

5.4 Operating system update and application program

Introduction

An OS update for a controller, an HMI or an I/O module, as well as access to the application program can be carried out via file system. For a detailed description on this topic refer to the following chapter:

- *Operating system update* (see page 249)
- *Application program* (see page 253)

5.5 Formatting and checking

Introduction

This chapter describes formatting the internal flash disk.

The internal flash disk needs not be checked using a separate function, since it provides maximum safety of its administrative structures by design.

Operating principle

When the device boots up, the OS system checks the content of the control register. The control register is part of the file system.

Depending on the value contained in this register the following functions are carried out:

- Formatting the flash disk

Register number

The register number of the control register is dependent on the device:

Device	Register number
JC-310-JM	202936

Contents

Formatting the flash disk

Introduction

In the following cases, reformatting the flash disk is required:

- When you upload an OS version that has got another flash disk format
- When information for flash disk administration has been destroyed

Consequences

- All files and directories located in the user area will be deleted!
- Formatting will not affect system files and directories.

Formatting the flash disk

To have the device format the internal flash disk, proceed as follows:

Step	Action
1	Switch the device ON.
2	Enter value -999720373 (0xc4697a4b) into the control register 202936 of the file system.
3	Switch the device OFF.
4	Switch the device ON.

Result: During the boot process of the JC-310-JM, the flash disk is formatted and the control register 202936 is set to **0**.

6 FTP server

Introduction

The FTP server lets you handle directories and files using an FTP client. The files can be stored to the following storage media:

- Internal flash disk

This chapter covers the login process and describes the commands supported by the FTP server.

FTP clients

The user has the option of using a command line FTP client, which comes with many PC operating systems, or graphic FTP tools.

Number of possible connections

The FTP server on the JC-310-JM is able to manage up to four FTP connections simultaneously.

Any additional FTP client, which tries to connect with the FTP server, will get no response to its request for establishing a connection.

Supported commands

The FTP server supports standardized commands. For more information refer to:

- FTP server help menu: Connect with FTP server and enter the command *help* or *help binary*.
- Internet: Search for FTP and commands

If you do not wish to care about commands, we advise you to use an FTP program, such as TotalCmd.

Required programmer's skills

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

- File system
- IP networks
- FTP commands

Contents

Logon

Logon

To be able to access the file system via FTP, the FTP client must log on when the connection is established.

- As **Server Name** enter the IP address of the device.
- As **User Name** enter your user name, e.g. *admin*.
- As **Password** enter your password, e.g. *admin*.

Factory settings

The factory settings include two user accounts:

```
[USER1]
NAME=admin
PW=admin

[USER33]
NAME=system
PW=system
```

Administration of users

The user administration of the file system lets you change the passwords and add new users.

Related topics

- **User administration** (see page 60)

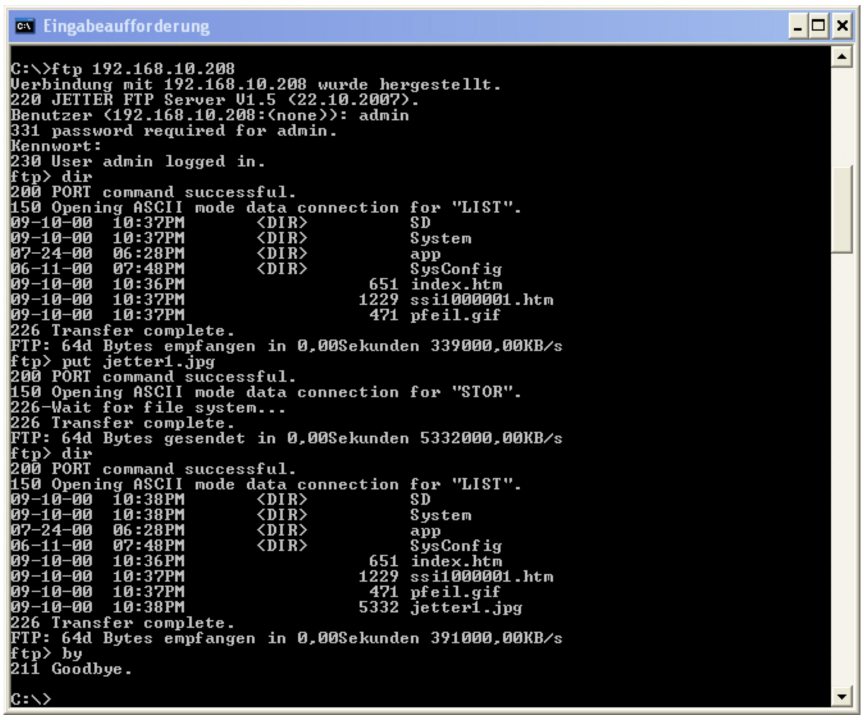
Example: Windows FTP client

Task

Carry out the following tasks using an FTP client, for example, the one which comes with Windows:

- Launch the FTP client by opening a connection and entering the IP address.
- Log in as user *admin* with password *admin*
- Displaying the content of the current directory using *dir*
- Transferring the file **jetter1.jpg** to the JetControl using the command *put*
- Displaying the content of the current directory using *dir*
- Terminating the session and the FTP client using *bye*

Action



```
C:\>ftp 192.168.10.208
Verbindung mit 192.168.10.208 wurde hergestellt.
220 JETTER FTP Server V1.5 (22.10.2007).
Benutzer (192.168.10.208:(none)): admin
331 password required for admin.
Kennwort:
230 User admin logged in.
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for "LIST".
09-10-00  10:37PM       <DIR>          SD
09-10-00  10:37PM       <DIR>          System
07-24-00  06:28PM       <DIR>          app
06-11-00  07:48PM       <DIR>          SysConfig
09-10-00  10:36PM                  651 index.htm
09-10-00  10:37PM                 1229 ssi1000001.htm
09-10-00  10:37PM                  471 pfeil.gif
226 Transfer complete.
FTP: 64d Bytes empfangen in 0,00Sekunden 339000,00KB/s
ftp> put jetter1.jpg
200 PORT command successful.
150 Opening ASCII mode data connection for "STOR".
226-Wait for file system...
226 Transfer complete.
FTP: 64d Bytes gesendet in 0,00Sekunden 5332000,00KB/s
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for "LIST".
09-10-00  10:38PM       <DIR>          SD
09-10-00  10:38PM       <DIR>          System
07-24-00  06:28PM       <DIR>          app
06-11-00  07:48PM       <DIR>          SysConfig
09-10-00  10:36PM                  651 index.htm
09-10-00  10:37PM                 1229 ssi1000001.htm
09-10-00  10:37PM                  471 pfeil.gif
09-10-00  10:38PM                 5332 jetter1.jpg
226 Transfer complete.
FTP: 64d Bytes empfangen in 0,00Sekunden 391000,00KB/s
ftp> by
211 Goodbye.

C:\>
```

7 FTP client

The FTP client

The FTP client allows access from within the application program to files and directories of a remote network device. To this end, the FTP client communicates with the FTP server of this network device.

Functions

The following functions are possible:

- Creating directories in the remote file system.
- Deleting directories in the remote file system.
- Copying files from the local file system into the remote file system.
- Copying files from the remote file system into the local file system.

Requirements

- To be able to use the FTP client feature basic knowledge of FTP connections and file systems is required.
- The IP address of the FTP server must be known.
- If the IP address of the FTP server is not known, name resolution through a DNS server must be possible.
- User name and password for logging on at the FTP server must be known.
- For programming this feature JetSym version 4.3 or higher is required.

Processing within the application program

- The controller completes only one FTP access at a time.
- The corresponding task in the application program stops at the command until the access is completed.
- During this time other tasks in the application program are processed.
- While an FTP access of a task is being processed, all other tasks which invoke an FTP command are blocked until the FTP access is completed.

Contents

7.1 Programming

Introduction

The FTP client allows to access files and directories on a network device from within the application program. For this purpose, function calls are used. These function calls are included in the programming language of the controller. To program this feature, proceed as follows:

Step	Action
1	Initialize the FTP client
2	Open the connections to the FTP servers
3	Transfer data
4	Terminate the connections

Restrictions

While the controller is processing one of the functions of the FTP client, tasks supporting the FTP client should not be stopped through `TaskBreak` or restarted through `TaskRestart`. Otherwise the controller fails to complete this function which will block new function calls by the FTP client.

Contents

Initializing the FTP client

Introduction

At each application program start, the FTP client must be initialized at least once.

Function declaration

```
Function FtpInitialize():Int;
```

Return value

The following return value is possible:

Return value	
0	Always

How to apply this function

The function is used and its return value assigned to a variable for further utilization in the following way:

```
Result := FtpInitialize();
```

Operating principle

The controller processes this function in the following steps:

Step	Description
1	The controller closes all open connections of the FTP client.
2	The controller initializes all OS-internal data structures of the FTP client.

Establishing a connection to the FTP server

Introduction

Before data can be sent or received, a connection to the FTP server must be established first. When establishing the connection, the client logs in to the FTP server by a user name and a password (login).

Function declaration

```
Function FtpConnect(Const Ref ServerAddr: String,
                    Const Ref UserName: String,
                    Const Ref PassWord: String):Handle;
```

Function parameters

Description of the function parameters:

Parameter	Value	Remarks
ServerAddr	IP address or name	Name resolution by DNS server
UserName	User name for logon	Login
Password	Password for logon	Login

Return value

If the return value was positive, the connection could be established and login was successful. If the return value was **0**, an error occurred and the connection could not be established.

Return value	
> 0	A positive return value must be stored in a variable. It must be made available as a handle at activating the following functions: ■ `Terminating a connection` ■ `Reading a file from the FTP server` ■ `Writing a file to the FTP server` ■ `Deleting a file from the FTP server` ■ `Changing a directory on the FTP server` ■ `Creating a directory on the FTP server` ■ `Deleting a directory from the FTP server` ■ `Determining the current directory on the FTP server`
0	Error when establishing a connection or logging in to the FTP server

Operating principle

The task stops at the program line until the connection is established or the timeout set for the FTP client has elapsed.

This function is processed in the following steps:

Step	Description	
1	The controller tries to establish a TCP/IP connection to the FTP server.	
2	**If ...**	**... then ...**
	... the network client has accepted the connection, ...	... proceed with step 3.
	... the connection could not be established and the timeout has not elapsed yet, ...	... proceed with step 1.
	... an error has occurred or the timeout has elapsed, ...	... the function is terminated and value **0** is returned.
3	The controller logs on to the FTP server with its user name *Administrator* and password *AdminPassword*.	
4	**If ...**	**... then ...**
	... the FTP server has accepted the connection, ...	... the function is terminated and a positive value is returned as handle for further access to this connection.
	... the FTP server has not accepted the connection, for example because of an invalid user name or wrong password, ...	... the function is terminated and value **0** is returned.

Related topics

- **Terminating a connection** (see page 86)

Terminating a connection

Introduction

Clear all connections which are no longer required as this will reduce PLC load for managing connections.

Function declaration

```
Function FtpDisconnect(FtpConnection:Handle):Int;
```

Function parameters

Description of the function parameters:

Parameter	Value	Remarks
FtpConnection	Handle	Value returned by the function `FtpConnect()`

Return value

The following return values are possible:

Return value	
0	Connection terminated and deleted
-1	Invalid handle
-2, -3	Communication error, there is, for example, no response from FTP server

Related topics

- **Establishing a connection to the FTP server** (see page 84)

Reading a file

Introduction

This function lets you read the content of a file from a remote network node and copy it to the local file system of the controller.

Function declaration

```
Function FtpFileRead(FtpConnection:Handle,
                     Const Ref ServerFile: String,
                     Const Ref ClientFile: String):Int;
```

Function parameters

Description of the function parameters:

Parameter	Value	Remarks
FtpConnection	Handle	Value returned by the function `FtpConnect()`
ServerFile	File name	Name of the file in the file system of the FTP server, which the controller is to read
ClientFile	File name	File name, as which the controller is to save the file read in the local file system

Return value

If the returned value is negative, an error has occurred. If the returned value is **0**, the controller was able to read the file and store it locally.

Return value	
0	No error
-1	Invalid handle
-2, -6	Error when storing the file locally
-3, -5, -7, -8	Communication error, there is, for example, no response from FTP server
-4	Error message from FTP server, for example, file does not exist

Operating principle

In the following cases, the task is not processed further after issuing the function call:

- The controller must first read the file, e. g. **ServerTestFile.txt** and save it to the local file system as, e. g., **LocalTestFile.txt**.
- An error has occurred.

This function is processed in the following steps:

Step	Description	
1	The controller sends a command to the FTP server that the content of the file **ServerTestFile.txt** must be transmitted.	
2	The controller receives the contents of the file **ServerTestFile.txt**.	
3	The controller writes the contents to the file **LocalTestFile.txt**.	
4	**If ...**	**... then ...**
	... no errors have occurred, ...	... the file has been copied successfully, the function is terminated and value **0** is returned.
	... errors have occurred, ...	the function is terminated and a negative value is returned.

File names

- The function parameter for the local file may contain the path to this file, e.g. `"/Data/TestFiles/LocalTestFile.txt"`.
- If the file system of the remote network node supports this, the function parameter for the file located on the FTP server can also contain the path to this file. Otherwise, the directory must be set beforehand using the command `FtpDirChange()`.
- The file system of a JC-310-JM supports both options.

Related topics

- **Writing to a file** (see page 89)

Writing to a file

Introduction

This function lets you copy the content of a file belonging a local file system to a file belonging to the file system of a remote network node.

Function declaration

```
Function FtpFileWrite(FtpConnection:Handle,
                      Const Ref ServerFile: String,
                      Const Ref ClientFile: String):Int;
```

Function parameters

Description of the function parameters:

Parameter	Value	Remarks
FtpConnection	Handle	Value returned by the function `FtpConnect()`
ServerFile	File name	File name as which the FTP server is to save the written file
ClientFile	File name	Name of the file in the local file system, the content of which the controller is to send to the FTP server

Return value

If the returned value is negative, an error has occurred. If the returned value is **0**, the controller was able to read the file and store it to the file system of the remote network node.

Return value	
0	No error
-1	Invalid handle
-2	Error when reading the local file, e.g. file does not exist
-3, -5, -8	Communication error, there is, for example, no response from FTP server
-4, -7	Error message from the FTP server, e.g. file cannot be created

Operating principle

In the following cases, the task is not processed further after issuing the function call:

- The controller must first read the file, e. g. **LocalTestFile.txt** and save it to the file system of the remote network node as, e. g., **ServerTestFile.txt**.
- An error has occurred.

This function is processed in the following steps:

Step	Description	
1	The controller sends a command to the FTP server that the content of the file **ServerTestFile.txt** must be saved.	
2	The controller sends the contents of the file **LocalTestFile.txt**.	
3	The FTP server writes the contents to the file **ServerTestFile.txt**.	
4	**If ...**	**... then ...**
	... no errors have occurred, ...	... the file has been copied successfully, the function is terminated and value **0** is returned.
	... errors have occurred, ...	the function is terminated and a negative value is returned.

File names

- The function parameter for the local file may contain the path to this file, e.g. `"/Data/TestFiles/LocalTestFile.txt"`.
- If the file system of the remote network node supports this, the function parameter for the file located on the FTP server can also contain the path to this file. Otherwise, the directory must be set beforehand using the command `FtpDirChange()`.
- The file system of a JC-310-JM supports both options.

Related topics

- **Reading a file** (see page 87)

Deleting a file

Introduction

This function lets you remove a file from the file system of a remote network node.

Function declaration

```
Function FtpFileRemove(FtpConnection:Handle,
                                Const Ref ServerFile: String):Int;
```

Function parameters

Description of the function parameters:

Parameter	Value	Remarks
FtpConnection	Handle	Value returned by the function `FtpConnect()`
ServerFile	File name	Name of the file to be removed.

Return value

If the returned value is negative, an error has occurred. If the returned value is **0**, the file could not be removed from the file system of the remote network node.

Return value	
0	No error
-1	Invalid handle
-2	Communication error, there is, for example, no response from FTP server
-3	Error message from FTP server, for example, file does not exist

Operating principle

In the following cases, the task is not processed further after issuing the function call:

- The FTP server must first delete the file **ServerTestFile.txt**. Please note: This file name serves as an example only.
- An error has occurred.

This function is processed in the following steps:

Step	Description	
1	The controller sends a command to the FTP server that the file **ServerTestFile.txt** must be deleted.	
2	The FTP server deletes the file **ServerTestFile.txt**.	
3	**If ...**	**... then ...**
	... no errors have occurred, ...	... the file has been deleted successfully, the function is terminated and value **0** is returned.
	... errors have occurred, ...	the function is terminated and a negative value is returned.

File names

- The function parameter for the local file may contain the path to this file, e.g. `"/Data/TestFiles/LocalTestFile.txt"`.
- If the file system of the remote network node supports this, the function parameter for the file located on the FTP server can also contain the path to this file. Otherwise, the directory must be set beforehand using the command `FtpDirChange()`.
- The file system of a JC-310-JM supports both options.

Changing directories

Introduction

This function lets you remove the current directory from the file system of a remote network node.

Function declaration

```
Function FtpDirChange(FtpConnection:Handle,
                      Const Ref ServerDir: String):Int;
```

Function parameters

Description of the function parameters:

Parameter	Value	Remarks
FtpConnection	Handle	Value returned by the function `FtpConnect()`
ServerDir	Directory name	Name of the directory into which the user wants to change

Return value

If the returned value is negative, an error has occurred. If the returned value is **0**, the system has managed to change directories.

Return value	
0	No error
-1	Invalid handle
-2	Communication error, there is, for example, no response from FTP server
-3	Error message from the FTP server, e.g. directory does not exist

Operating principle

In the following cases, the task is not processed further after issuing the function call:

- The FTP server must first change directories.
- An error has occurred.

This function is processed in the following steps:

Step	Description	
1	The controller sends a command to the FTP server that it has to change to a subdirectory.	
2	The FTP server changes directories.	
3	**If ...**	**... then ...**
	... no errors have occurred, ...	... the new directory is set, the function is terminated and value **0** has been returned.
	... errors have occurred, ...	the function is terminated and a negative value is returned.

Directory names

- If the file system of the remote network node supports this, the function parameter for the directory located on the FTP server can also contain the complete path including several subdirectories leading to this directory.
- If this feature is not supported, the user must navigate from one directory level to the next until the desired directory is reached. This is done using the command `FtpDirChange()`.
- The file system of a JC-310-JM supports both options.

Related topics

- **Determining the current directory** (see page 99)

Creating a directory

Introduction

This function lets you create a new directory from the file system of a remote network node.

Function declaration

```
Function FtpDirCreate(FtpConnection:Handle,
                      Const Ref ServerDir: String):Int;
```

Function parameters

Description of the function parameters:

Parameter	Value	Remarks
FtpConnection	Handle	Value returned by the function `FtpConnect()`
ServerDir	Directory name	Name of the directory to be created

Return value

If the returned value is negative, an error has occurred. If the returned value is **0**, the directory could successfully be created in the file system of the remote network node.

Return value	
0	No error
-1	Invalid handle
-2	Communication error, there is, for example, no response from FTP server
-3	Error message from FTP server, e.g. directory already exists

Operating principle

In the following cases, the task is not processed further after issuing the function call:

- The FTP server must first create a subdirectory.
- An error has occurred.

This function is processed in the following steps:

Step	Description	
1	The controller sends a command to the FTP server that it has to create a subdirectory.	
2	The FTP server creates the directory.	
3	**If ...**	**... then ...**
	... no errors have occurred, ...	... the new directory has been created, the function is terminated and value **0** is returned.
	... errors have occurred, ...	... the function is terminated and a negative value is returned.

Directory names

- If the file system of the remote network node supports this, the function parameter for the directory located on the FTP server can also contain the complete path including several subdirectories leading to this directory.
- If this feature is not supported, the user must navigate from one directory level to the next until the desired directory is reached. This is done using the command `FtpDirChange()`.
- The file system of a JC-310-JM supports both options.

Restrictions regarding the file system of a JetControl

If you specify a directory with the corresponding path as function parameter, all directories up to the directory you want to create must exist. Recursive creation of several directories is not supported.

Example:

```
Result := FtpDirCreate(FtpHandle,
                       '/DataFiles/TextFiles/Release');
```

To be able to create the folder **Release** in the directory tree */DataFiles/TextFiles* the directories must already exist.

Related topics

- **Deleting directories** (see page 97)

Deleting directories

Introduction

This function lets you remove a directory from the file system of a remote network node.

Function declaration

```
Function FtpDirRemove(FtpConnection:Handle,
                      Const Ref ServerDir: String):Int;
```

Function parameters

Description of the function parameters:

Parameter	Value	Remarks
FtpConnection	Handle	Value returned by the function `FtpConnect()`
ServerDir	Directory name	Name of the directory to be deleted

Return value

If the returned value is negative, an error has occurred. If the returned value is **0**, the directory could successfully be removed from the file system of the remote network node.

Return value	
0	No error
-1	Invalid handle
-2	Communication error, there is, for example, no response from FTP server
-3	Error message from the FTP server, e.g. directory does not exist

Operating principle

In the following cases, the task is not processed further after issuing the function call:

- The FTP server must first remove the subdirectory.
- An error has occurred.

This function is processed in the following steps:

Step	Description	
1	The controller sends a command to the FTP server that it has to remove the subdirectory.	
2	The FTP server removes the subdirectory.	
3	**If ...**	**... then ...**
	... no errors have occurred, ...	... the directory is removed, the function is terminated and value **0** is returned.
	... errors have occurred, ...	the function is terminated and a negative value is returned.

Directory names

- If the file system of the remote network node supports this, the function parameter for the directory located on the FTP server can also contain the complete path including several subdirectories leading to this directory.
- If this feature is not supported, the user must navigate from one directory level to the next until the desired directory is reached. This is done using the command `FtpDirChange()`.
- The file system of a JC-310-JM supports both options.

Related topics

- **Creating a directory** (see page 95)

Determining the current directory

Introduction

This function lets you determine the current directory in the file system of a remote network node.

Function declaration

```
Function FtpDirPrint(FtpConnection:Handle,
                     Ref str: String):Int;
```

Function parameters

Description of the function parameters:

Parameter	Value	Remarks
FtpConnection	Handle	Value returned by the function `FtpConnect()`
str	String address	Current directory with path specification

Return value

If the returned value is negative, an error has occurred. If the returned value is **0**, the current directory could successfully be determined in the file system of the remote network node.

Return value	
0	No error
-1	Invalid handle
-3	Communication error, there is, for example, no response from FTP server
-4	Error message sent by the FTP server
-5	Invalid response from server

Operating principle

In the following cases, the task is not processed further after issuing the function call:

- The FTP server must first determine the actual directory.
- An error has occurred.

This function is processed in the following steps:

Step	Description	
1	The controller sends a command to the FTP server that it has to determine the current directory.	
2	The FTP server transmits the actual directory with path specification.	
3	**If ...**	**... then ...**
	... no errors have occurred, ...	... the variable contains the complete path of the current directory, the function is terminated and value **0** is returned.
	... errors have occurred, ...	the function is terminated and a negative value is returned.

Related topics

- **Changing directories** (see page 93)

7.2 Registers

Introduction

This chapter describes the registers on the controller which contain status information of the FTP client. These registers can be used for debugging or diagnostic purposes. However, they can't be used for other functions such as establishing or terminating a connection.

Contents

Register numbers

Introduction

Data of one connection each are displayed within the registers of a coherent register block. Two other registers show the status of the command being executed by the application program. The basic register number of these registers is dependent on the controller.

Register numbers

Device	Basic register number	Register numbers
JC-310-JM	320000	320000 ... 320101

Determining the register number

In this chapter, only the last three figures of a register number are specified. e.g. MR 002. To determine the complete register number, add to this module register number the basic register number of the corresponding device, for example 320000.

Registers - Overview

FTP client module registers - Overview

Register	Description
MR 000	Number of open connections
MR 002	Timeout in seconds
MR 003	Port number of the FTP server
MR 004	Index in the connection table
MR 005	Connection handle
MR 006	IP address of the FTP server
MR 007	Port number of the FTP server
MR 008	IP address of FTP client
MR 009	Port number of FTP client
MR 100	Processing status on part of FTP client
MR 101	Task ID

Registers - Description

Introduction

Established connections are managed by the operating system of the controller in a list. Module registers MR 004 or 005 are used to copy connection data into registers MR 006 through MR 009.

MR 000

Number of open connections

The value in this register shows how many connections are currently open.

Module register properties		
Reading values	0 ... 2,147,483,647	Number of connections

MR 002

Timeout

To this register, write the timeout of the FTP client at accessing the FTP server.

Module register properties		
Values	0 ... 2,147,483,647	in seconds
Value after reset	20	

MR 003

Port number of the FTP server

The value in this register shows the IP port number of the FTP server.

Module register properties	
Values	0 ... 65,535
Value after reset	21

MR 004

Index in the connection table

The index of the connection table is entered into this register. If a connection has been established for a given index, the connection handle can be seen in module register MR 005 and connection data in module registers MR 006 through MR 009.

Module register properties	
Values	0 ... [MR 000] - 1
Value after reset	-1

MR 005

Connection handle

This register is for entering the connection handle. If a connection has been established for a given index, the connection handle can be seen in module register MR 004 and connection data in module registers MR 006 through MR 009.

Module register properties	
Values	0 ... 2,147,483,647

MR 006

IP address of the FTP server

The value in this register shows the IP address of the FTP server.

Module register properties	
Access	Read
Takes effect	if MR 004 >= 0

MR 007

Port number of the FTP server

The value in this register shows the port number of the FTP server.

Module register properties	
Access	Read
Takes effect	if MR 004 >= 0

MR 008

IP address of FTP client

The value in this register shows the IP address of the FTP client.

Module register properties	
Access	Read
Takes effect	if MR 004 >= 0

MR 009

Port number of FTP client

The value in this register shows the port number of the FTP client.

Module register properties	
Access	Read
Takes effect	if MR 004 >= 0

MR 100

Processing status on part of FTP client

This register lets you track the processing status on part of FTP client.

Module register properties		
Values	0	No access at the moment
	1	Parameters are being handed over to the FTP client of the controller
	2	The FTP client communicates with the FTP server.
	3	Access completed
Access	Read	

MR 101

Task ID

This register shows the ID of the task which is processing an FTP client function at that moment.

Module register properties		
Values	0 ... 99	Task ID
	255	None of the tasks is carrying out an FTP function.
Value after reset	255	
Access	Read	

8 HTTP server

Introduction

A standard browser is sufficient for accessing the HTTP server.

The browser is for reading and displaying files which have been downloaded to the controller via FTP.

Here, it may be necessary to enter the user name and password to have access to certain pages (depending on the file system configuration).

Default file names

The default file names are **index.htm** and **index.html**.

Supported file types

The following file types are supported:

- *.htm, *.html, *.shtml
- *.txt, *.ini
- *.gif, *.tif, *.tiff, *.bmp, *.wbmp
- *.jpg, *.jpe, *.jpeg, *.png
- *.xml
- *.js, *.jar, *.java, *.class, *.cab
- *.ocx
- *.pdf, *.zip, *.doc, *.rtf
- *.css
- *.wml, *.wmlc, *.wmls, *.wmlsc
- *.ico, *.svg

Required programmer's skills

To be able to use the functions described in this chapter, the following skills are required:

- File system
- IP networks

Contents

8.1 Server Side Includes

Introduction

The HTTP server features *Server Side Includes* (SSI). This function is for showing present real-time controller values on an HTML page.

Rules

You must specify a **Name Space** tag at the beginning of the HTML page that is to contain the real-time controller values.

This **Name Space** is for defining the namespace used in the HTML page.

In the body section of the HTML page the **Data** tags are specified.

Updating real-time controller values

When the HTML page is uploaded to the browser, the HTTP server once replaces the **Data** tags by actual real-time controller values.

To refresh the controller values, the HTML page must be reloaded over and over again.

The user triggers reloading by entering the controller address and the name of the required page, e.g. *http://192.168.10.209/Homepage/SSI/ssiTimeAndDate.htm*.

Contents

First entry in the HTML file

Configuration

The **Name Space** must be the first entry in the HTML file. It has got the following structure:

```
<NS:DTAG xmlns:NS=http://jetter.de/ssi/jetcontrol/
```

with **NS** representing the namespace. The namespace is a character string with a maximum length of 63 characters.

The namespace introduced here will be re-used for the subsequent **Data** tags. The remaining parts of the line are preassigned and have to be specified in exactly the same way.

In the following examples, the namespace applied is **JW**.

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	<JW: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	
Function	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 (standard type)	Integer, decimal
INTX	Integer, hexadecimal
INTB	Integer, binary
BOOL	Register content = 0 --> Display: 0 Register content != 0 --> Display: 1
REAL	Floating point, decimal
STRING	Text register

Example:

```
JW:DTAG name="R1000250" type="REAL" format="+0####.###"
factor="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 (standard type)	Flag = 0 --> Display: 0
	Flag = 1 --> Display: 1

STRING	Flag = 0 --> Display: FALSE
	Flag = 1 --> Display: TRUE

Example:

```
<JW: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 (standard type)	Input = 0 --> Display: 0
	Input = 1 --> Display: 1

STRING	Input = 0 --> Display: OFF
	Input = 1 --> Display: ON

Example:

```
<JW:DTAG name="I100000308" type="STRING" />
```

Result:
The state of input 100000308 is displayed as string "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 (standard type)	Output = 0 --> Display: 0
	Output = 1 --> Display: 1

STRING	Output = 0 --> Display: OFF
	Output = 1 --> Display: ON

Example:

```
<JW:DTAG name="O100000308" />
```

Result:
The state of output 100000308 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:

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

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

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

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

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

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

```
<JW: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:

```
<JW: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.

```
<JW: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.

```
<JW: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.

```
<JW: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.

```
<JW: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.

```
<JW: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.

```
<JW: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.

```
<JW: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.

Example of an HTML page

Task

Insert current real-time controller values into an HTML page.

It should then be possible to display the HTML page in a browser using the *Server Side Includes* feature.

Action

```
<JC:DTAG xmlns:JC="http://jetter.de/ssi/jetcontrol" />
<html>

<head>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1252">
<meta name="GENERATOR" content="Microsoft FrontPage 4.0">
<meta name="ProgID" content="FrontPage.Editor.Document">
<title>Index</title>
</head>

<body>
Hello World,&nbsp;
<p>Actual controller values can be inserted into an html page like
this:&nbsp;</p>
<p>Register 201000 = <JC:DTAG name="R201000" type = INT
format="+####" />,
or Hex notation: 0x<JC:DTAG name="R201000" type="INTX"
format="0###" />,
or rather this way: <JC:DTAG name="R201000" type="BOOL" />, if only
Boolean is queried.
Binary notation is also an option: <JC:DTAG name="R201000"
type="INTB" format=######## />b.&nbsp;</p>
<p>Strings could also be defined "<JC:DTAG name="R201000"
type="STRING" />".&nbsp;</p>
<p>A real number looks as follows: <JC:DTAG name="R1001500"
type="REAL" />
or this way: <JC:DTAG name="R1001500" type="REAL" factor="1.3"
format="###.##" />.&nbsp;</p>
<p>The value of a flag is represented as follows: <JC:DTAG name="F10"
/>
or this way: <JC:DTAG name="F10" type="STRING" />.&nbsp;</p>
<p>With inputs and outputs, it is done the same way: <JC:DTAG
name="I100000205" type="BOOL" />
or <JC:DTAG name="I100000205" type="STRING" />.&nbsp;</p>
<p>R201000 = <JC:DTAG name="R201000" type="INT"
format="+0##########" />&nbsp;</p>
<p>Regards&nbsp;</p>
<p>Yours JetControl</p>
</body>

</html>
```

Storage location

Now store the HTML page to the file system of the controller.

9 Programming

Purpose of this chapter

This chapter is for supporting you when programming a JC-310-JM controller in the following fields of activity:

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

Prerequisites

To be able to program the JC-310-JM the following prerequisites have to be fulfilled:

- The controller is connected to a PC.
- The JetSym programming software has been installed on the PC.

Contents

Abbreviations, module register properties and formats

Abbreviations

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

Abbreviation	Description
R 100	Register 100
MR 150	Module register 150

Module register properties

Each module register is characterized by certain properties. Most properties are identical for many module registers - the value after reset is always zero, for example. In the following description, module register properties are mentioned only if a property deviates from the following default properties.

Module register properties	Default property for most module registers
Access	Read/write
Value after reset	0 or undefined (e.g. release 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 programs

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

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

9.1 Speicherübersicht

Introduction

The JC-310-JM features a data memory. There is, for example, 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.

Contents

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 ▪ Internal volatile RAM for storing OS data
Memory access	▪ The user is not allowed to directly access the OS memory. ▪ To modify the OS, it must be updated.
Related topics	▪ **OS update** (see page 249)

File system memory

Introduction

The file system memory is for storing data and program files.

Properties

- Internal flash disk
- Non-volatile
- Slow access: Milliseconds up to seconds
- Limited number of write/erase cycles: approx. 1 million
- Size of internal flash disk: 4 MB

Memory access

- By operating system
- By JetSym
- Via FTP connection
- By the e-mail client
- By browser (via HTTP server)
- By means of file commands from within the application program

Application program memory

Introduction

By default, the application program is uploaded from JetSym to the controller and is stored to it.

Properties

- Stored as file within the file system
- Default directory */app*
- Files may also be stored to other directories (or on SD card)

Memory access

- By operating system
- By JetSym
- Via FTP connection
- By means of file commands from within the application program

Related topics

- **Application program** (see page 253)

Memory for volatile application program variables

Introduction

Volatile variables are used to store data which need not be maintained when the JC-310-JM is de-energized.

Properties

- 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

Special registers

Introduction	Special registers let you control OS functions and retrieve status information.
Properties	▪ Global variables which are assigned to permanent addresses (%VL) ▪ When the operating system is launched, special registers are initialized using default values. ▪ Register numbers: 100000 ... 999999
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 either have the value TRUE or FALSE.

Properties of virtual inputs/outputs

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

Properties of digital inputs/outputs

- Global variables assigned to permanent addresses (%IX, %QX)
- I/O numbers of remote nodes connected to a JX3-BN-ETH or another JC-3xx controller:
  1nnn010201 ... 1nnn011716 (nnn = GNN)

Memory access

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

Flags

Introduction

Flags are one-bit operands. This means they can either have the value TRUE or FALSE.

Properties of user flags

- Global variables which are assigned to permanent addresses (%MX)
- Volatile
- Amount: 256
- Flag numbers: 0 ... 255

Properties of overlaid user flags

- Global variables which are assigned to permanent addresses (%MX)
- Volatile
- Overlaid by registers 1000000 ... 1000055
- Amount: 1,792
- Flag numbers: 256 ... 2047

Properties of special flags

- Global variables which are assigned to permanent addresses (%MX)
- When the operating system is launched, special flags are initialized using their default values.
- Amount: 256
- Flag numbers: 2048 ... 2303

Memory access

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

9.2 Register and I/O numbers with a JC-3xx

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.

Applying a register number

Register numbers are applied in the following cases:

- You want to read or write to a module register in the JetSym setup.
- You want to declare a module register a variable in the JetSym application program.
- You want to declare a module register a tag in JetViewSoft

Applying an I/O number

I/O numbers are applied in the following cases:

- You want to read from a digital input in the JetSym setup.
- You want to read or write to a digital output in the JetSym setup.
- You want to declare a digital input or output a variable in the JetSym application program.
- You want to declare a digital input or output a tag in JetViewSoft.

Contents

Registers and module registers

Module registers - Definition

Module registers are the data interface of a JX3 module. Module registers let you read process, configuration and diagnostics data from the JX3 module, or write such data to it.

- The module register number within a module is unique.
- This unique register number lets you access a specific module register within the system.

Registers - Definition

There are several ways to access registers directly:

- From an application program
- From the JetSym setup pane
- From 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.

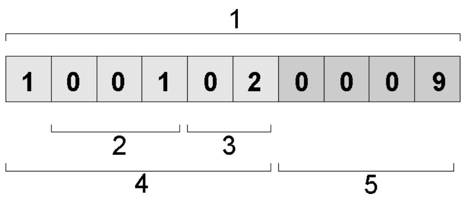
- The GNN within a network has to be unambiguous for each Jetter device.
- The JetSym Hardware Manager automatically assigns the GNN during configuration.
- The value range of the GNN within a project is 000 ... 199.
- The controller has always got GNN 000.

Example - Module registers

Module register 9 lets you access the OS version of a JX3-AI4 module.

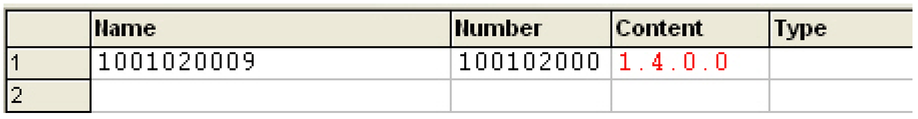
Example: Registers on the Ethernet system bus

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	Supports direct access
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

In the setup pane of JetSym you can directly read out the operating system version 1.4.0.0 from register number 1001020009.



	Name	Number	Content	Type
1	1001020009	100102000	1.4.0.0	
2				

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

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

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 the Ethernet bus node consists of the following elements:

1	n	n	n	x	x	z	z	z	z

Element	Description	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

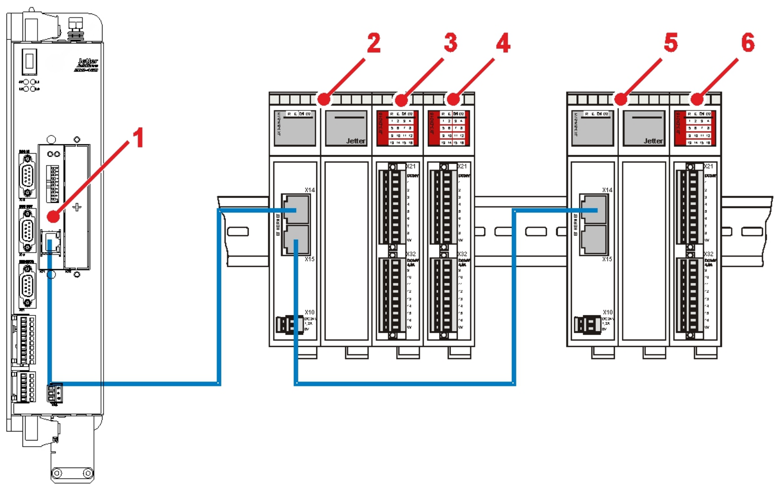
The I/O number for JX3 modules connected to an Ethernet bus node consists of the following elements:

1	n	n	n	0	1	x	x	z	z

Element	Description	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-310-JM.



Number	Module	Module number	GNN	Register	I/O
1	JC-310-JM	1	-	Refer to JC-3xx manual	
2	JX3-BN-ETH	-	1	Refer to JX3-BN-ETH manual	
3	JX3-DIO16	2	-	100102zzzz	10010102zz
4	JX3-DIO16	3	-	100103zzzz	10010103zz
5	JX3-BN-ETH	-	2	Refer to JX3-BN-ETH manual	
6	JX3-DIO16	2	-	100202zzzz	10020102zz

Registers and I/O numbers of JX3 nodules from the JX3-BN-ETH perspective

Application example

Explicit data transfer using `NetCopy()`.

Module numbers in a JX3 station

To determine module numbers in a JX3 station, proceed as follows:

- Count the JX3-I/O modules left-to-right, starting with 1.
- Leave out the power supply module JX3-PS1.

Register numbers for JX3 modules

From the perspective of the Ethernet bus node, the register number consists of the following elements:

1	0	0	x	x	z	z	z	z

Element	Description	Value range
xx	Number of the module within the JX3 station	02 ... 17
zzzz	Module register number	0000 ... 9999

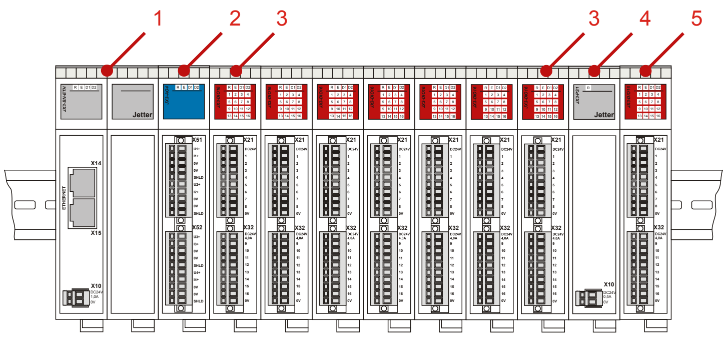
I/O numbers for JX3-modules

From the perspective of the Ethernet bus node, the I/O number consists of the following elements:

1	0	0	0	0	x	x	z	z

Element	Description	Value range
xx	Number of the module within the JX3 station	02 ... 17
zz	Module-specific I/O number	1 ... 16

Example of a JX3 station at a JX3-BN-ETH



Number	Module	Module number	Register	I/O
1	JX3-BN-ETH	1	Refer to JX3-BN-ETH manual	
2	JX3-AO4	2	10002zzzz	1000002zz
3	JX3-DIO16	3 ff.	10003zzzz	1000003zz
4	JX3-PS1	-	-	-
5	JX3-DIO16	10	10010zzzz	1000010zz

9.3 Jetter Ethernet system bus

Introduction

The Jetter Ethernet system bus has been designed for data exchange between the following devices via standard Ethernet.

- Programming device
- Controllers
- Bus node
- Communication modules

Data interchange

The Jetter Ethernet system bus makes a difference between the cyclic and acyclic data interchange between communication participants. Both kinds of data interchange can be executed simultaneously within a network.

Data exchange	Properties
Cyclic	▪ **Architecture:** Publish/subscribe ▪ **Nodes:** Controllers, bus nodes and communication modules ▪ **Access:** Automatically by OS ▪ **Access time:** Fast, deterministic ▪ **Data:** Registers, inputs/outputs ▪ **Configuration:** Hardware Manager in JetSym ▪ **Reach:** Subnet
Acyclic	▪ **Architecture:** Client/server ▪ **Client:** PC and controllers ▪ **Server:** PC, controllers, bus nodes and communication modules ▪ **Data:** E.g. registers, inputs/outputs, STX variables, application program ▪ **Access:** PC or application program ▪ **Access time:** Depending on the reaction time of the server ▪ **Configuration:** Only when using network registers ▪ **Reach:** International

Minimum requirements

The device is operated in a system consisting of various components by Jetter AG. In order to ensure proper interaction of these components, the operating system used and the programming tool JetSym must have at least the release numbers listed below.

Component	As of version
JC-310-JM	V. 1.22.0.00
JC-340	V. 1.22.0.00
JC-350	V. 1.22.0.00
JC-360	V. 1.22.0.00
JC-360MC	V. 1.22.0.00
JC-365	V. 1.26.0.00
JC-365MC	V. 1.26.0.00
JC-940MC	V. 1.06.0.20
JC-945MC	V. 1.01.0.00
JX3-BN-ETH	V. 1.18.0.02
JX3-COM-EIPA	V. 1.01.0.00
JX3-COM-PND	V. 1.03.0.06
JM-200-ETH	V. 1.22.0.00
JetSym	V. 5.1.2

Contents

The Global Node Number

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.

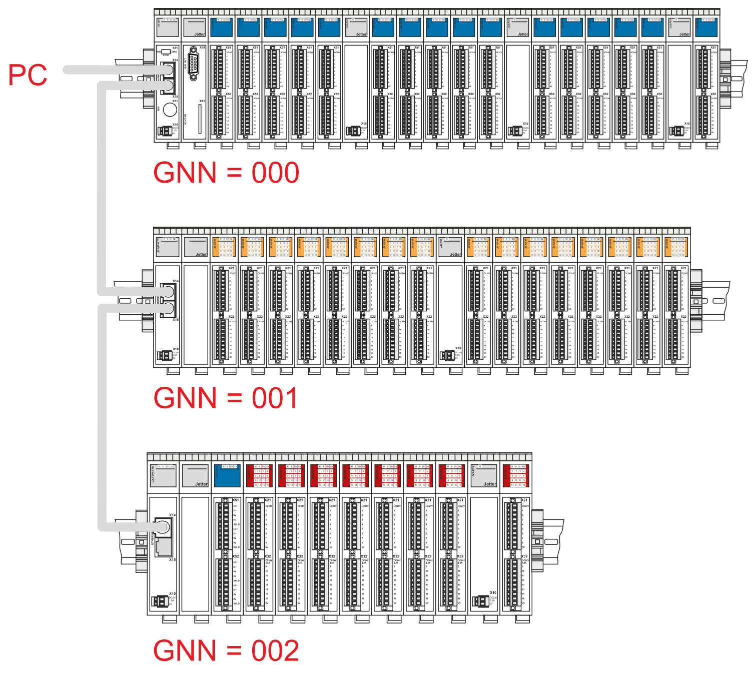
Using the GNN

The Global Node Number is used in the following applications:

- Register number for network registers
- Identification of publications and subscriptions at cyclic data interchange
- Identification of nodes at automatic network configuration (NetConsistency)

Networking example

The following illustration shows networking of a possible JX3 system with a JC-3xx and two JX3-BN-ETH.



9.3.1 Acyclic data interchange

Introduction

This chapter covers acyclic data interchange on the Jetter Ethernet system bus.

Properties

Acyclic data interchange on the Jetter Ethernet system bus can be characterized as follows:

Property	Description
Architecture	Client/server ▪ Data interchange is initiated by the client. ▪ The server gives a response to the request made by the client. ▪ Usage of unicast frames ▪ Network access is made once.
Client	▪ JetSym: Programming and debugging of application programs ▪ JetViewSoft: Setting up a visualization application ▪ Controllers: Data interchange out of the application program (NetCopy..., NetBit..., network register)
Server	▪ PC: E.g. for database applications ▪ Controllers, bus nodes and communication modules: E.g. variable or debugging server
Data	▪ PC: Registers, inputs/outputs, STX variables, application program ▪ Controllers, bus nodes and communication modules: Registers, STX variables
Access time	▪ It depends on the data transfer time and on the server's processing time
Configuration	▪ Network registers: Easy configuration in the application program ▪ Else, both client and server are completely configured by the operating system.
Reach	▪ Using TCP/IP and UDP/IP frames allow for data interchange exceeding the limits of one's own subnet.

Client

Below, programming the client in the controllers is described. In doing so, the following topics are dealt with:

- Transferring variable/register sets (command group NetCopy())
- Setting and clearing register bits (command group NetBit())
- Transmitting individual register values (network registers)

Examples of the application

- Event-triggered data interchange
- Parameterization
- Configuration

Used protocol

The client of the controller uses the JetIP protocol based on UDP/IP for data transfer.

Server

The server functions do not require any programming or configuration by the user.

Protocols

Acyclic data interchange on the Jetter system bus can be established by the following protocols:

- XCOM protocol by Jetter AG
- JetIP protocol by Jetter AG
- UDP/IP
- TCP/IP
- IPv4

Contents

Command group NetCopy()

Introduction

The NetCopy command is a versatile tool for data interchange between Jetter products via Ethernet.

The NetCopy command lets you copy the following data:

- Register values
- Values of register blocks
- Variable values
- Values of variable blocks

Advantages of NetCopy

Advantages of NetCopy commands as compared with the use of network registers:

- Within the command, you can directly specify any valid IP address.
- Within the command, you can directly specify any valid IP port.
- The entire register address range of a remote node can be directly addressed.
- By means of one command, a large register set or, in case NetCopyList is applied, a large number of registers can be copied.
- The result of the copying process can be evaluated directly.

Access via NetCopy

NetCopy functions with the following nodes:

- Controllers
- Bus node
- Communication modules
- PC

To access other nodes, use the NetCopy command as follows:

If ...	... then ...
... you wish to copy data from the controller to another node,	... use the following commands: ■ NetCopyRegToReg ■ NetCopyVarToReg ■ NetCopyList
... you wish to copy data from another node to the controller,	... use the following commands: ■ NetCopyRegFromReg ■ NetCopyVarFromReg ■ NetCopyList

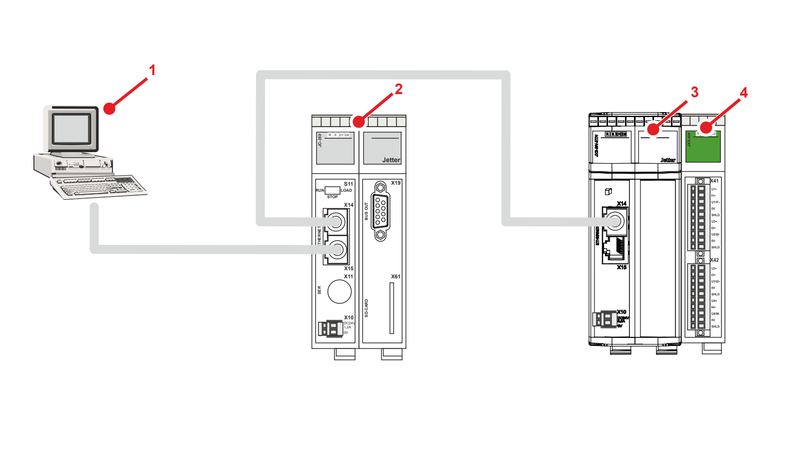
Parameters of the NetCopy commands

For detailed information on the parameters, refer to the JetSym help.

NetCopy - Example featuring a bus node

As you can see in the following illustration, a controller JC-3xx is connected to a PC. The bus node JX3-BN-ETH of IP address 192.168.10.2 is connected to a peripheral module JX3-AI4.

This example describes how to access the module registers of the peripheral module JX3-AI4 in acyclic mode.



Number	Part	Description
1	PC	PC with JetSym
2	JC-3xx	Controller
3	JX3-BN-ETH	Bus node
4	JX3-AI4	Peripheral module with analog inputs

Task

When an event occurs, user scaling of analog input 1 is to be changed.

Solution

The NetCopy command causes values from application program variables to which the user scaling parameters have been stored to be copied to the corresponding registers of the JX3-AI4.

The register number of the peripheral module is seen from the perspective of the JX3-BN-ETH:

1	0	0	x	x	z	z	z	z

with

- xx = 02: First module on the JX3-BN-ETH
- zzzz = 1124 through 1127: Parameter registers of the JX3-AI4 user scaling

```
// Copy values from the local array to the JX3-AI4
nResult := NetCopyVarToReg(IP#192.168.10.2, anParam,
                           100021124, 16, 3, 1);
```

Command group NetBit()

Introduction

The NetBit command is an all-purpose tool to set or clear register bits of Jetter products. The Jetter products are interconnected via an Ethernet network.

Advantages of NetBit

NetBit commands let you both set and clear bits in one go.

Simulating NetBit commands by means of NetCopy commands:

- A NetCopy command lets you copy the register value from the remote node to the local controller.
- A NetCopy command lets you change the state of the bits on the local controller as desired.
- Another NetCopy command lets you copy the register value to the remote node again.

For this, several commands are required. Thus, a register value may be changed during this action by an application program running on the remote controller. The second NetCopy command will then overwrite this value again. There is an undefined data condition, which is prevented by the NetBit functions.

Further advantages of NetBit commands as compared with the use of network registers:

- Within the command, you can directly specify any valid IP address.
- Within the command, you can directly specify any valid IP port.
- The entire register address range of a remote node can be directly addressed.
- The result of executing this command can be evaluated directly.

Access via NetBit

NetBit functions with the following nodes:

- Controllers
- Bus node
- Communication modules

To access other nodes, use the command NetBit as follows:

If ...	... then ...
you wish to set register bits for another node,	use the command ▪ NetBitSetReg
you wish to clear register bits of another node,	use the command ▪ NetBitClearReg

Parameters of the NetBit commands

For detailed information on the parameters, refer to the JetSym help.

Network registers

Introduction

The network registers let you access in transparent mode registers of remote nodes.

Advantages

Advantages of network registers as compared with NetCopy commands:

- Network registers are used just like any other registers in the application program.
- If programs or parts of programs are used for local and distributed applications, a program description is not needed.

Restrictions

The following restrictions apply to network registers as compared with NetCopy commands:

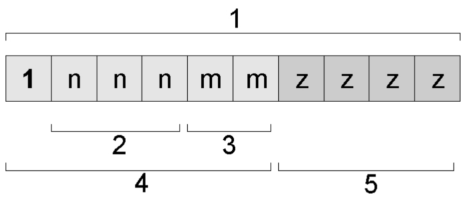
- IP address and IP port of the remote node must be set separately.
- Only part of the register address range of the remote nodes can be accessed directly.
- The outcome of the network access (diagnostics) cannot be logged directly.

Properties

If you access network registers of cyclic data interchange, the controller does not carry out acyclic network register access. The controller accesses the locally stored cyclic data.

Addressing scheme

The addressing scheme for network registers is as follows:



No.	Element	Description
1	Register number	Supports direct access
2	First part of register prefix: Bus node ID, GNN	nnn = 001 ... 199: ID of the network node, referred to as Global Node Number
3	Second part of register prefix: Number of the function module	mm = 02 ... 17: Number of the JX3 module of a remote node mm = 98: Indirect addressing of the register of a remote node mm = 99: Addressing the variable destination window of a remote node
4	Part 1 + 2: Register prefix	1nnnmm: The prefix is preceded by a leading ONE.
5	Module register number	zzzz = 0000 ... 9999

IP address and IP port

Before using a network register, the IP addresses and IP ports of the remote network nodes must be written to two tables in the local register array.

If ...	... then ...
... you carry out network configuration in the JetSym Hardware Manager,	... these tables are generated automatically, see file **ModConfig.da** below.
... you do not carry out network configuration in the Hardware Manager,	... you must generate the tables in your application program.

Content indexing of the tables is done via GNN of the node in the first part of the register prefix (2).

Register	Value range	Properties
235000 + GNN	235000 ... 235199	Register table for **IP addresses**
235400 + GNN	235400 ... 235599	Register table for **IP ports**

Note on the contents of the table:

- GNN = Global Node Number in the range 000 ... 199

File *ModConfig.da*

When you download the configuration files, the Hardware Manager transfers the file **ModConfig.da** to the controller.

The OS of the controller loads this file when the controller is energized or when the corresponding command is automatically issued by the Hardware Manager after download.

The file **ModConfig.da** lists registers with their corresponding values. The OS enters the corresponding values into these registers.

This file also holds the IP addresses (register 235000 + GNN) and port numbers (register 235400 + GNN) of the nodes on the network.

It is no longer required to enter values into registers via application program.

Registers located on JX3 modules

Introduction

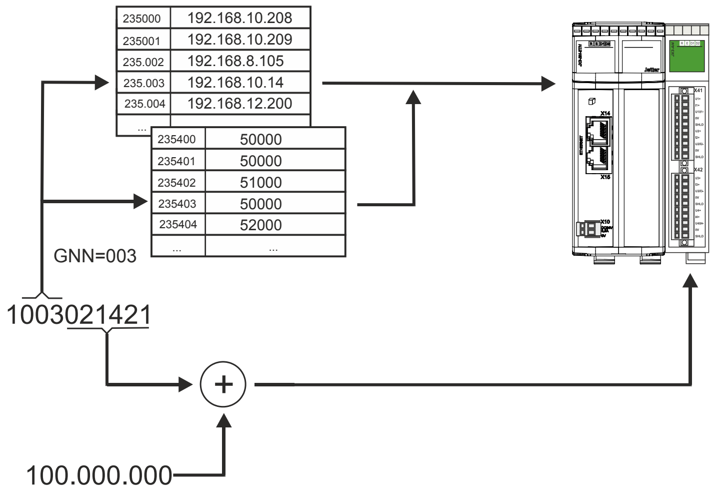
The controller handles access via network registers to module registers of JX3 modules of a remote node (second part of the register prefix mm = 02 ... 17) in a specific way:

If ...	... then ...
... the network register has been configured for cyclic data interchange,	... the controller accesses the locally stored register value.
... the network register has not been configured for cyclic data interchange,	... the controller executes acyclic network access.

Acyclic network access

For acyclic register access to a remote JX3 module, the controller uses the first part of the register prefix as an index to the tables containing the IP addresses and IP ports. The values read out of these tables are used by the controller to address the bus node in the network. For further information on registers 235000 ff, please refer to **Network registers - IP address and IP port** (see page 141).

The controller adds the register offset for the JX3 system bus of a remote node (100,000,000) to the second part of the register prefix and the module register number (no. 3 and 5 in the **addressing scheme** (see page 141)). The controller uses the resulting number to address the register.



Action

If you want to access the JX3 module register of a remote network node using register addresses as of 1 billion, proceed as follows:

Step	Action
1	Enter the **IP address** of the remote network node into register **235.000 + GNN**. Value range of the GNN: 1 ... 199
2	Enter the **port number** into register **235400 + GNN**. Value range of GNN: 1 ... 199
⇨	Now you can access the value via register **1nnnmmzzzz.** Value range of GNN = nnn: 001 ... 199 Value range mm: 02 ... 17 Value range zzzz: 0000 ... 9999

This lets you directly access all JX3 module registers of the remote network node.

Example

Via network, a controller and a bus node JX3-BN-ETH are connected.
A JX3-AI4 module is connected to the bus node.

Configuration of the bus node	Value
GNN	3
IP address	192.168.10.14
IP port	50000

Task:

The trailing indicator of the analog channel 4 peak value is to be read.

Solution:

You create a JetSym STX program by taking the following steps:

- Register 235003 is loaded with the IP address of the bus node.
- Register 235403 is loaded with the IP port of the bus node.
- The value of network register 1003021421 is assigned to a local variable.

Indirect addressing of remote modules

Introduction

Indirect addressing of network registers lets you access registers of a remote network node. First enter the number of the remote node register into a table of register numbers in the local controller. Content indexing of this table is carried out via the three low-order figures of the network register number.

Registers - Overview

Overview of the registers allowing indirect addressing of remote nodes:

Register	Value range	Properties
236000 + zzz	236000 ... 236199	Register table for the **register numbers**
1nnn980zzz	1nnn980000 ... 1nnn980199	Register array for the **Content**

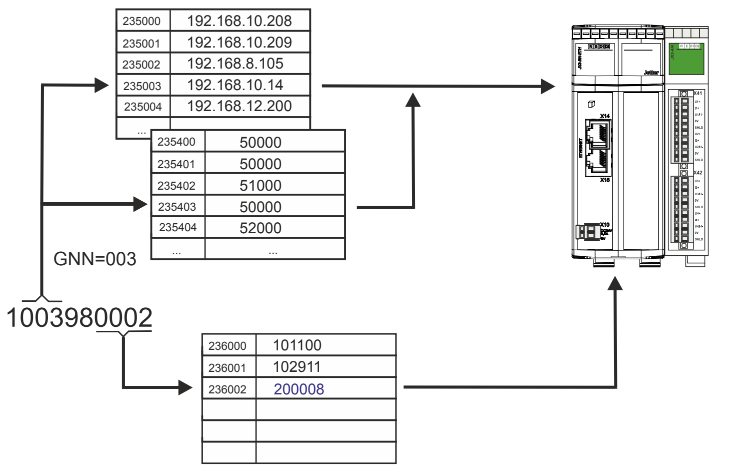
Note on the contents of the table:

- nnn = GNN in the range 000 ... 199
- zzz in the range 000 ... 199

Indirect network register access

For indirect access to a remote node via network register, the controller uses the first part of the register prefix as an index to the tables containing the IP addresses and IP ports. The values read out of these tables are used by the controller to address the bus node in the network. For further information on registers 235000 ff, please refer to **Network registers - IP address and IP port** (see page 141).

The module register number (no. 5 in the **addressing scheme** (see page 141)) is used by the controller as an index to a table of register numbers. The register number read out of this table is used by the controller to address the register in the bus node.



Action

If you want to access the register of a remote network node using register addresses as of 1 billion, proceed as follows:

Step	Action
1	Enter the **IP address** of the remote network node into register **235000 + GNN**. Value range of the GNN: 0 ... 199
2	Enter the **port number** into register **235400 + GNN**. Value range of the GNN: 0 ... 199
3	Enter the required **register number** of the remote network node into register **236000 + zzz**.
⇨	Now you can access the value via register **1nnn980zzz.** Value range of the GNN: nnn = 000 ... 199 Value range zzz: 000 ... 199

This configuration lets you indirectly access - via 200 controller registers - all module registers of the remote network node.

Example

Via network, a controller and a bus node JX3-BN-ETH are connected.

Configuration of the bus node	Value
GNN	3
IP address	192.168.10.14
IP port	50000

Task:
The global error register of the JX3-BN-ETH is to be read every second.

Solution:

- Register 235003 is loaded with the IP address of the bus node.
- Register 235403 is loaded with the IP port of the bus node.
- Register 236028 is loaded with the error register number 200008.

Addressing with variable destination window

Introduction

Indirect addressing also allows for a variable destination window. You shift the register array of 10,000 registers of the remote network nodes by an offset by entering a value into R 272702 of the remote network nodes.

Registers - Overview

Overview of the registers allowing indirect addressing with variable destination window:

Register	Value range	Properties
1nnn99zzzz	1nnn990000 ... 1nnn999999	**Register content** of a remote network node; The register is in the variable destination window which consists of 10,000 registers.
272702 (of the remote node)	0 ... 2,147,483,647	Variable destination window: The destination window is a register array of a remote network node. This destination window is shifted by this **offset**.

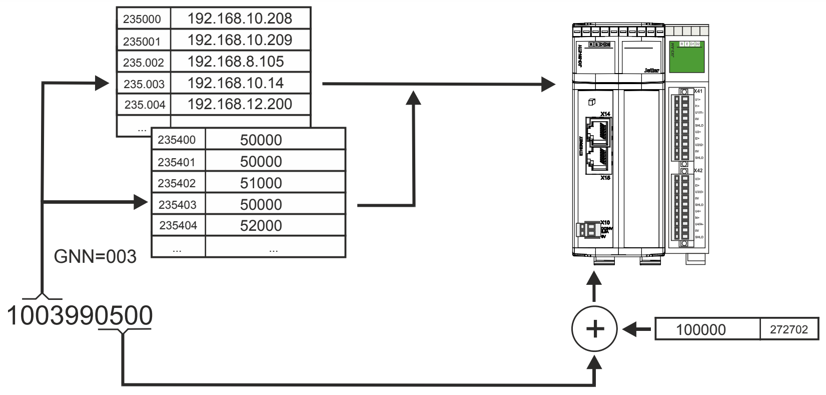
Note on the contents of the table:

- nnn = GNN in the range 000 ... 199
- zzzz in the range 0 ... 9,999

Network register access with variable destination window

For access via network register with variable destination window to a remote node, the controller uses the first part of the register prefix as an index to the tables containing the IP addresses and IP ports. The values read out of these tables are used by the controller to address the bus node in the network. For further information on registers 235000 ff, please refer to **Network registers - IP address and IP port** (see page 141).

The module register number (no. 5 in the **addressing scheme** (see page 141)) is used by the controller to address the register in the bus node. A register number is transmitted to the remote network node by the controller. The remote network node adds the content of register 272702 to this register number and uses the result as register number.



Steps to take for addressing with destination window

To use register addresses starting from 1 billion with variable destination window (offset), proceed as follows:

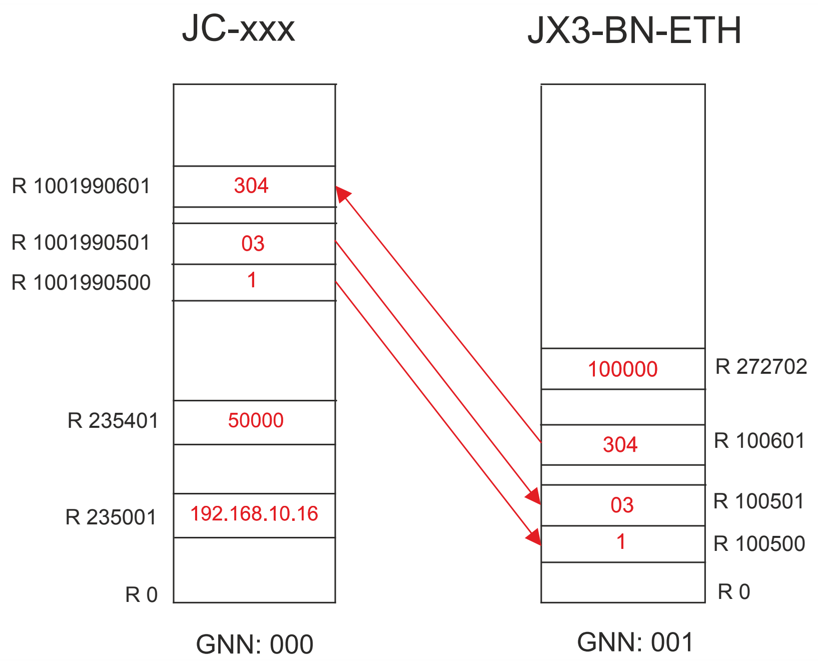
Step	Action
1	Enter the **IP address** of the remote network node into register **235000 + GNN**. Value range of the GNN: 0 ... 199
2	Enter the **port number** into register **235400 + GNN**. Value range of the GNN: 0 ... 199
3	Set the base address of the **destination window:** Enter a value into R 272702 of the remote network node.
⇨	Now, registers **1nnn990000 ... 1nnn999999** let you access the value.

Example

A JetControl is to read a register value from a JX3-BN-ETH. Control system and bus node are interconnected via the Jetter Ethernet system bus.

There are JX3 modules connected to the JX3-BN-ETH, such as a JX3-AO4 of module number 03.

By entering value 100000 into R 272702 of the JX3-BN-ETH, you get read access to the EDS of the connected JX3 modules. In this example, the module code of the JX3-AO4 is to be read. For further information on how to read an EDS, please refer to **EDS registers** (see page 22).



Reading is carried out in three steps:

Step	Action
1	Enter value 1 for a JX3 module into R 1001990500.
2	Enter module number 03 into R 1001990501.
3	Read module code 304 for JX3-AO4 from R 1001990601.

Register description - Acyclic data interchange

Introduction

In acyclic data interchange, data transmission from a controller to remote network nodes is carried out via JetIP protocol. The client in the controller is supplied with registers for configuration and error diagnostics.

Registers/flags - Overview

Register	Description
232708	Timeout in milliseconds
232709	Response time in milliseconds
232710	Amount of network errors
232711	Error code of last access
232717	Maximum number of retries
232718	Present number of retries

Flags	Description
2075	Network error

R 232708

Timeout

To R 232708, write the timeout (in milliseconds) for acyclic access via network.

Module register properties	
Values	1 ... 65,535 [ms]
Value after reset	250 [ms]

R 232709

Response time

R 232709 displays the total response time of latest acyclic access via network in milliseconds. The total response time includes the time for data transmission and the processing times in the controller and in the remote network node.

Module register properties	
Values	0 ... 65,535 [ms]
Type of access	Read

R 232710

Amount of network errors

R 232710 shows the total number of network errors.

Module register properties	
Values	-2,147,483,648 ... 2,147,483,647 (overflowing)

R 232711

Error code

R 232711 shows the error code of the latest network access.

Module register properties		
Values	0	No errors
	1	Timeout
	3	Error message from remote node
	5	Invalid network address
	6	Invalid amount of registers
	7	Invalid interface number

R 232717

Maximum number of retries

R 232717 lets you set the maximum possible number of network access retries. If a network access could not be made without errors, the controller will repeat the access at the most as often as it has been set in this register. If the network access could still not be made without errors, the controller will cancel the access and create an error message.

Module register properties	
Values	0 ... 255

R 232718

Present number of retries

R 232710 shows the total number of network access retries.

Module register properties	
Values	-2,147,483,648 ... 2,147,483,647 (overflowing)

M 2075

Network error

If a network error occurs, the operating system sets flag 2075. In order to detect further errors this way, you must manually reset the flag.

Flag properties		
Values	0	No network errors since last reset
	1	A network error has occurred

9.3.2 Cyclic data interchange

Introduction

This chapter covers cyclic data interchange via Jetter Ethernet system bus.

Properties

Properties of cyclic data interchange via Jetter Ethernet system bus:

Property	Description
Architecture	Publish/subscribe ■ The publishers send the data. ■ The subscribers receive the data. ■ Usage of multicast frames
Publisher	■ Each publisher sends one or several publications. ■ Data of a publication are consistently transferred in a frame. ■ The cycle time can be set for each publication.
Subscriber	■ The subscriber receives one or several publications and assigns them to the corresponding subscriptions. ■ The subscriber validates the received data.
Data	■ Registers ■ Inputs ■ Outputs
Access time	■ Very short, as the network nodes access the locally stored interchanged data.
Configuration	■ In the JetSym Hardware Manager
Reach	■ Restricted to own subnet

Examples of the application

- Cyclic, deterministic interchange of process data
- Cyclic, deterministic interchange of status information details

The JetSym Hardware Manager generates the configurations for cyclic data interchange using the status information details and the process data of the connected peripheral modules.

Restrictions

For cyclic data interchange, do not use any configuration registers or special registers. Access to these registers can take longer or trigger further action, which may lead to unwanted results.

Multicast in other networks

Please note that the Jetter Ethernet system bus operates with multicasts (multipoint connections). If you couple the Jetter Ethernet system bus with your local network, you have to filter out unwanted multicasts by a router.

As an alternative, the function JetSync blockage can be used, too.

Technical specifications

Technical specifications of cyclic data interchange via Jetter Ethernet system bus:

- Usage of multicast frames
- Reserved multicast groups: 255
- Multicast groups available to the user: 0 ... 254
- IP addresses for multicasts: 239.192.0.0 + multicast group
- MAC address for multicasts: 01:00:5E:40:00:00 + multicast group
- Maximum size of user data in a publication/subscription: 256 byte

Contents

Publish/subscribe

Introduction

Publish/subscribe is used as communication architecture for cyclic data interchange in the Jetter Ethernet system bus. The JetSym Hardware Manager generates the configurations for cyclic data interchange and transfers them to the controller. Based on this configuration, the configuration automatically carries out cyclic data interchange.

Basic data interchange

Basic data interchange via publish/subscribe is executed by the publishers and subscribers in the operating system of the Jetter devices at the Jetter Ethernet system bus.

Publisher

- The publishers publish data of the network node, on which they are being processed.
- A data record is published by the publisher. Therefore it is called **publication**.
- A publisher can manage several publications.

Subscriber

- The subscribers which are interested in these data receive the publications and transfer the contents to the data of the network node on which they are processed.
- A data record is received by the subscriber. Therefore it is called **subscription**.
- A subscriber can manage several subscriptions.
- To receive a publication, there must exist a corresponding subscription.
- One publication can be received by subscriptions on various network nodes simultaneously, as the publications are published via multicast frames.

JetSym

When a combination of a controller and one or several network nodes is configured in JetSym, the Hardware Manager generates the configuration files for the publishers and for the subscribers. The Hardware Manager generates one-to-one relationships between the publications and the subscriptions.

Features of publish/subscribe

If, in Hardware Manager, you add network nodes with the modules connected to them, Hardware Manager will automatically generate the process data belonging to these modules as publish/subscribe variables.

Features of publish/subscribe

Parameter	Value	Description
Number of network nodes	000 ... 199	200 network nodes max.: They are entered into the Hardware Manager by their name and as GNN
Maximum amount of process variables per publication/subscription	64	64 process variables max: This corresponds to 256 bytes of process data

Parameter	Value	Description
Cycle time	1 ... 2,147,483,647 ms	Default: 2 ms

Network nodes are the controller, the communication modules and the bus nodes.

For details on characteristic features of publish/subscribe, please turn to chapter **Hardware Manager** (see page 165).

Configuring and executing publish/subscribe

Publish/subscribe is configured in the JetSym Hardware Manager. Publish/subscribe is executed by the operating system of the respective network node:

- Publishers and subscribers are configured by means of configuration files in the file system of the network nodes.
- The configuration file for the publisher is **/SysConfig/JetSync/Publisher.pub**.
- The configuration file for the subscriber is **/SysConfig/JetSync/Subscriber.sub**.
- Automatic restart of the publishers and subscribers takes place in a controller at each restart of the application program.
- In the other network nodes, automatic restart of the publishers and subscribers takes place during the booting phase.
- For applying publishers and subscribers in a controller, an application program must be executed with at least one task running.

For transferring the configuration, the Hardware-Manager takes the following steps:

Step	Action
1	Stop all publishers and subscribers.
2	Transfer the configuration files to all network nodes.
3	Restart all publishers and subscribers.

Related topics

- **Hardware Manager** (see page 165)

Publish/subscribe - Registers

Introduction

If you transmit cyclic data by publish/subscribe, there are several module registers available for administration, configuration and error detection. You have got read and partial write access to these module registers.

Register overview

Module registers	Description
210004, 200008, 200009	General error registers
250000 ... 250004	Registers for administration of all subscriptions
250x10 ... 250x11	Registers for administration of one subscription
250x20 ... 250x30	Registers for configuring one subscription
254001 ... 254003	Registers for error detection
255000 ... 255004	Registers for administration of all publications
255x10 ... 255x11	Registers for administration of one publication
255x20 ... 255x30	Registers for configuring one publication
Flag 2080	Enable for publishing an error
Flag 2081	Error collection of the subscriber

x = 0 ... 9

Availability

Administration and configuration registers are available as follows:

- For subscriptions and publications, 10 arrays for administration and configuration registers are available.
- The register arrays differ by the hundred's place of the respective register number.
- The placeholder x indicates the number of the register array. Value range of x: 0 ... 9
- External clients use register array x = 1, such as JetSym with visualization application and PCOMX protocol.
- STX functions use register array x = 0.
- In order to gain faster access to individual publish/subscribe administration registers, several register arrays are at your disposal: There are individual publish/subscribe IDs to be called in each register array.

Registers for administration of all subscriptions

There are several registers available which go with all subscriptions.

Register	Name	Description
250000	Status	Status register
250001	Command	Command register
250002	ID in case of error	Displays the ID of the subscription, in which an error has occurred.
250003	Amount	Total amount of subscriptions
250004	CRC	16-bit CRC (**C**yclic **R**edundancy **C**ode) of the subscriber configuration file

Subscriber status

Status registers of all subscriptions

From MR 250000, you can read the collective status of all subscriptions. In case of an error, you first read out the ID of the subscription, in which an error has occurred.

Meaning of the individual bits

Bit 0	**Error in CRC computing of the configuration file**	
	0 =	No error has occurred.
	1 =	For CRC computing, the configuration file does not exist. For this reason, CRC computing has not taken place.
Bit 1	**Error in connection with a subscription**	
	1 =	An error has occurred in a subscription. At the moment, this is only a timeout error.
Bit 7	**Subscription is functioning.**	
	0 =	If a subscription fails, bit 7 is reset.
	1 =	The subscriptions are functioning.

Module register properties

Type of access	Read

Subscriber command

Command registers of all subscriptions

Via MR 250001, you transmit commands to all subscriptions.

Commands	
102	**Reboot all subscribers**
105	**Stop all subscribers**
110	**Acknowledge error**

Selecting a subscription

The following registers let you select a subscription as follows:

- The index is for selecting subscriptions.
    - If the subscription exists, R 250x11 shows its ID.
    - If the subscription does not exist, R 250x11 shows value **-1**.
- In this case, enter the ID of the subscription into R 250x11.
    - If the subscription exists, the content of R 250x11 is kept.
    - If the subscription does not exist, R 250x11 shows value **-1**.

Register	Name	Description
250x10	Index	Index of the subscriptions: 0: Selects the first subscription 1: Selects the next subscription 2: etc.
250x11	ID	The ID of the subscription is entered

Configuring a subscription

The following registers show the configuration of a subscription, which you have selected via R 250x10 and R 250x11.

Register	Name	Description
250x20	Status	Bit 0: Publication received Bit 1: Timeout
250x21	Mode	0: Cyclic 1: Upon request
250x22	Number of variables	As configured
250x23	Group address	As configured
250x24	Hash	Internal usage
250x25	Sequence number	Internal usage
250x26	Data size	Internal usage
250x27	Timeout in ms	Bus cycle * 3

Register	Name	Description
250x28	Number of received publications	-
250x29	Amount of timeouts	-
250x30	Amount of missing sequence numbers	The subscriber of a publication computes the difference between present and last received sequence number. If the value of the difference is greater than one, certain publications have not been received.

Registers for error detection

If a subscription has not received any process data from the assigned publication before timeout, the subscription will generate an error. Further, the operating system writes the address of the bus node into registers 254001 to 254003, with which communication has been terminated.

This helps you to search for the error exactly in this bus node using NetCopy commands.

Register	Name	Description
254001	GNN	Global Node Number
254002	IP address	
254003	Port number	

Registers for administration of all publications

There are several registers available which go with all publications.

Register	Name	Description
255000	Status	Status register
255001	Command	Command register
255002	ID in case of error	Displays the ID of the publication, in which an error has occurred.
255003	Amount	Amount of all publications
255004	CRC	16-bit CRC (Cyclic Redundancy Code) of the publication configuration file

Publisher status

Status registers of all publications

From MR 255000, you can read the collective status of all publications. In case of an error, you first read out the ID of the publication, in which an error has occurred.

Meaning of the individual bits

Bit 0 **Error in CRC computing of the configuration file**

0 =	No error has occurred.
1 =	For CRC computing, the configuration file does not exist. For this reason, CRC computing has not taken place.

Bit 1	**Error in connection with a publication**	
	1 =	An error has occurred in a publication.
Bit 7	**Publication is functioning**	
	0 =	If a publication fails, bit 7 is reset.
	1 =	The publications are functioning.

Module register properties	
Type of access	Read

Publisher command

Command registers of all publications

Via MR 255001, you transmit commands to all publications.

Commands	
102	**Reboot all publishers**
105	**Stop all publishers**
110	**Acknowledge error**

Selecting a publication

The following registers let you select a publication:

- The index is for selecting publications.
  - If the publication exists, R 255x11 shows its ID.
  - If the publication does not exist, R 255x11 shows value **-1**.
- In this case, enter the ID of the publication into R 255x11.
  - If the publication exists, the content of R 255x11 is kept.
  - If the publication does not exist, R 255x11 shows value **-1**.

Register	Name	Description
255x10	Index	Index of the publications: 0: Selects the first publication 1: Selects the next publication 2: etc.
255x11	ID	The ID of the publication is entered

Configuring a publication

The following registers show the configuration of a publication, which you have selected via R 255x10 and R 255x11.

Register	Name	Description
255x20	Status	Bit 0: Publication transmitted
255x21	Mode	0: Cyclic 1: Upon request

Register	Name	Description
255x22	Number of variables	As configured
255x23	Group address	As configured
255x24	Hash	Internal usage
255x25	Sequence number	Internal usage
255x26	Data size	Internal usage
255x27	Timeout in ms	Bus cycle
255x28	Number of publications sent	-
255x29	Number of retries	-
255x30	Number of transmit errors	-

Network registers, network inputs and outputs

Introduction

The network registers, network inputs and outputs let you access in transparent mode, at cyclic data interchange, registers, inputs and outputs of remote nodes. The controller accesses the local image of the cyclic data.

Prerequisites

These are the prerequisites for using the registers, inputs and outputs at cyclic data interchange:

- Via publish/subscribe, the data are interchanged in cyclic mode.

Properties

Network registers, network inputs and outputs are not used in cyclic data interchange:

- If network registers of non-cyclic data interchange are accessed, the controller generates acyclic network register access.
- If network inputs and outputs of non-cyclic data interchange are accessed, the controller does not generate acyclic network register access. There are no data being transmitted via network.

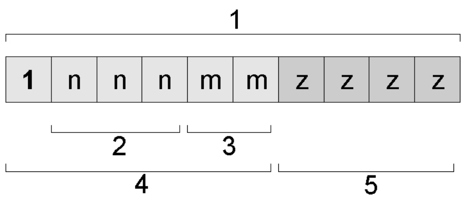
Advantages of network registers, network inputs and outputs

Advantages of network registers, network inputs and outputs in cyclical data interchange as compared with acyclic data interchange:

- The operating system cyclically interchanges data of the registers, inputs and outputs with other network nodes.
- This results in network load optimization.
- This is a very quick access, as, at the instance of use, only the local images of the data have to be accessed.

Register addressing scheme

The addressing scheme for network registers is as follows:



No.	Element	Description
1	Register number	Supports direct access
2	First part of register prefix: Bus node ID, GNN	nnn = 001 ... 199: ID of the network node, referred to as Global Node Number
3	Second part of register prefix: Number of the function module	mm = 02 ... 17: Number of the JX3 module of a remote node mm = 91: Registers of the combined digital inputs and outputs of a remote node
4	Part 1 + 2: Register prefix	1nnnmm: The prefix is preceded by a leading ONE.

No.	Element	Description
5	Module register number	zzzz = 0000 ... 9999

Network registers for accessing JX3 modules

Characteristic feature of the register number for access to remote JX3 modules: The value of the second part of the register prefix is the number of the module at the JX3 system bus (02 ... 17).

In cyclic data interchange, access to the process data of the remote JX3 modules is made via network registers.

For further information on configuration of data interchange and generated variables for access to JX3 modules, please turn to chapter **Hardware Manager** (see page 165).

Register overview - Inputs and outputs

The register number, in which the digital inputs and outputs of the remote nodes have been combined, is characterized by the value being 91 in the second part of the register prefix.

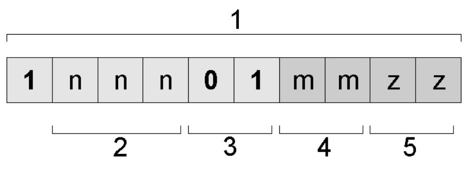
Overview

Registers	Description
1nnn914000 ... 1nnn914030	32 combined inputs
1nnn914060 ... 1nnn914092	16 combined inputs
1nnn914120 ... 1nnn914153	8 combined inputs
1nnn914200 ... 1nnn914230	32 combined outputs
1nnn914260 ... 1nnn914292	16 combined outputs
1nnn914320 ... 1nnn914353	8 combined outputs

Where nnn = GNN: 000 ... 199

Addressing scheme - Inputs and outputs

The addressing scheme for the digital network inputs and outputs at cyclic data interchange is as follows:



No.	Element	Description
1	I/O number	Supports direct access
2	Bus node ID, GNN	nnn = 001 ... 199: ID of the network node, referred to as Global Node Number.
3	Designation:**01**: I/O 01 as a fixed number	01: 01 indicates that a JX3 module is to be addressed.
4	Module number	mm = 02 ... 17: Number of the JX3 module of a remote node
5	Module-specific I/O number	zz = 01 ... 16: Specifies which input/output on the module is to be addressed

Example

Via network, a controller and a bus node JX3-BN-ETH are connected. A JX3-DO16 is connected to the bus node. The JX3-DO16 has got I/O module number 3.

Task:

The outputs of the JX3-DO16 are to be activated or deactivated as follows:

Step	Description
1	All outputs with odd numbers are active for half a second, while all outputs with even numbers are deactivated.
2	All outputs with even numbers are active for half a second, while all outputs with odd numbers are deactivated.
3	There is a moving light from output 1 to output 16; each corresponding output is activated for 200 ms.
4	Proceed with step 1.

Solution:

Configure the network group in the JetSym Hardware Manager and write an application program. Download both to the network nodes.

Related topics

- **Hardware Manager** (see page 165)

9.3.3 Hardware Manager

Introduction

The Hardware Manager lets you easily configure the peripheral devices. If possible, always use the Hardware Manager that is part of JetSym. Making configurations by hand is complicated and prone to errors

Detailed information

For detailed information on hardware configuration using Hardware Manager, refer to the JetSym help.

Contents

Hardware Manager

Hardware Manager

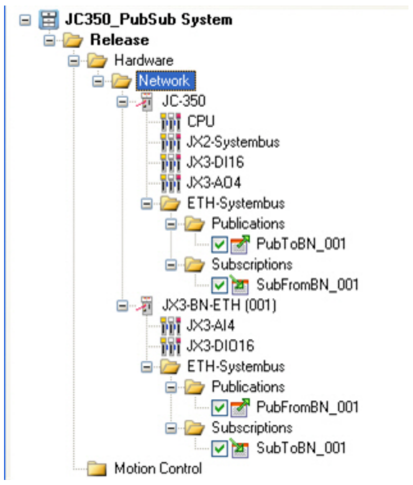
The Hardware Manager manages all connected hardware components.

The Hardware Manager assists you in the following aspects:

- Engineering and configuring control systems and bus nodes
- Engineering modules and axes at the JX2 system bus and configuring axes at the JX2 system bus
- Engineering JX3 modules at a JX3-BN-ETH and a JC-3xx
- Engineering and configuring Ethernet axes
- Engineering an axis group (path group and technology group)
- Configuring a path group
- Configuring technology group

Launching the Hardware Manager

For launching the Hardware Manager, klick, in JetSym, the tab **Hardware**. As an alternative, launch the Hardware Manager via keys **[Alt] + [5]**.



```
JC350_PubSub System
  Release
    Hardware
      Network
        JC-350
          CPU
          JX2-Systembus
          JX3-DI16
          JX3-AO4
          ETH-Systembus
            Publications
              PubToBN_001
            Subscriptions
              SubFromBN_001
        JX3-BN-ETH (001)
          JX3-AI4
          JX3-DIO16
          ETH-Systembus
            Publications
              PubFromBN_001
            Subscriptions
              SubToBN_001
    Motion Control
```

Related topics

- **Ethernet system bus** (see page 133)

9.3.4 Error handling at the Jetter Ethernet system bus

Introduction

This chapter covers error handling at the Jetter Ethernet system bus.

Contents

Acyclic data interchange - Error logging

Introduction

The programmer uses the following information for error logging:

- Return values of the commands
- JetIP networking registers and flags

NetCopy() and NetBit()

For error logging, use the return values of the respective command. You will find them in the JetSym online help.

Jetter AG recommends not to execute error logging via the registers and flags of the JetIP network.

Network registers

Error logging for network registers and flags of the JetIP networking:

Registers/flags	Description
Flag 2075	Errors at acyclic data interchange
Register 232710	Amount of errors at acyclic data interchange
Register 232711	Error code of the latest acyclic data interchange

Error message during CRC computing

Detecting the error

Both publisher and subscriber carry out a CRC of their configuration files. The calculated value can be read from registers 255004 and 250004. If there is no configuration file, they report an error.

Root cause of the error

This error may be caused by the following root cause:

- CRC computing failed, because there is no configuration file.

Response of the device to this error

The operating system of the device responds to the error by taking the following steps:

Step	Description
1	The operating system sets bit 0 in the status register of the publisher (R 255000) or of the subscriber (R 250000).

Fixing the root cause

Deploying a configuration file

Acknowledging the error

After deploying a configuration file, restart both publisher and subscriber.

Error message on part of a subscription

Detecting the error

If a subscriber has not received any process data from the assigned publisher before timeout, the subscriber will generate an error. The subscriber for the subscription of which the error has been generated, can run either on a controller or on a network node. The remote network node is a JX3-BN-ETH, for example.

Root cause of the error

The error may be caused as follows:

- Communication with the network client providing the process data is terminated.

Response of the device to this error

The operating system of the device responds to the error by taking the following steps:

Step	Description	
1	Sets bit 1 in R 250000.	
2	Writes the subscription ID to R 250002.	
3	Sets flag 2081.	
4	Writes value 11103 and the ID to the error buffers. The error buffer can be accessed via registers 380000 ff. (error history).	
5	Writes the GNN of the network node communication with which has been terminated to R 254001.	
6	Writes the IP address of the network node communication with which has been terminated to R 254002.	
7	Writes the port number of the network node communication with which has been terminated to R 254003.	
8	**If ...**	**... then ...**
	... flag 2080 is set,	... bit 3 is set in R 210004 and R 200008. The red status LED of the controller is lit.

Fixing the root cause

By means of NetCopy commands, you can precisely locate the error and fix the root cause. This works, because GNN, IP address and port number of the other network node are known.

Acknowledging the error

To acknowledge the error, write command 110 to register 250001.

Controller evaluates errors reported by a remote network node

Access to the status registers

The controller has got read access to the contents of the following status registers of all network nodes at the Jetter Ethernet system bus.

The contents are accessed via registers 39nnn0 through 39nnn5. (GNN: nnn = 001 ... 199).

Registers	JX3-BN-ETH, JX3-COM-EIPA	Controller
Error register	200008	39nnn0
Enhanced error register 1	200009	39nnn1
Enhanced error register 2	200010	39nnn2
JetSync status	240010	39nnn3
Subscriber status	250000	39nnn4
Subscription ID	250002	39nnn5

The operating system writes the ID of the subscription for which last an error has been reported to register 250002.

Locating faults

If the value of register 39nnn0 is unequal zero, an error has occurred. A network node has reported this error to the controller via its status registers.

In consequence, the operating system of the controller reacts by taking the following steps:

Step	Description		
1	The operating system sets bit 10 in R 200009.		
2	**If ...**	**... or ...**	**... then ...**
	... Bit x = 1 of R 200009,	Bit x = 1 of R 200010,	... the operating system sets bit 7 of R 200008.
3	The operating system enters the GNN of the network node having last reported an error to the controller into R 394001.		
4	The operating system enters the IP address of the network node having last reported an error to the controller into R 394002.		
5	The operating system enters the port number of the network node having last reported an error to the controller into R 394003.		

Fixing the root cause

By means of NetCopy commands, you can precisely locate the error and fix the root cause. This works, because GNN, IP address and port number of the other network node are known.

Make sure the contents of registers 39nnn0 through 39nnn5 are read by the application program. Further registers having got a value unequal zero indicate that further network nodes have reported an error. Make sure you also clear these errors.

9.3.5 Administrating the connections of the JetIP/TCP and STX debug server

Introduction

This document covers the connection management enhancements of the JetIP/TCP server and of the STX debug server in a JetControl PLC.

If, for example, the Ethernet cable was unplugged or cut, the node was not able to clear the connection. The connection remained active.

The enhanced connection management allows for the server to clear connections according to criteria that can be set by the user.

Number of connections

The number of simultaneously established connections for the TCP server in a JetControl is limited to the following value:

Server	Connections
JetIP/TCP server	4
STX debug server	20

Contents

Automatic termination of connections

Introduction

If the maximum number of simultaneously established connections has been reached, any further connections cannot be established. If further connect requests are made, the user can set the response by the JetIP/TCP server and of the STX Debug server. There are the following possibilities:

- Reject new connection.
- Terminate one existing connection and establish the new one.
- Terminate all existing connections and establish the new one.

Default setting

By default, the server terminates the connection with the longest time of inactivity.

No automatic termination of connections

If the server is not to terminate any of the existing connections, proceed as follows:

Step	Action
1	Enter value 0 into MR 1.

Terminating the connection with the longest time of inactivity

If the server is to terminate the connection that has been inactive the longest time, proceed as follows:

Step	Action
1	Enter value -1 into MR 2.
2	Enter value 1 into MR 1.

Terminating the connection when the set minimum time has expired

If the server is to terminate a connection after a set minimum time of inactivity, proceed as follows:

Step	Action
1	Enter the minimum time [ms] into MR 2.
2	Enter value 1 into MR 1.

If the set minimum value has not been exceeded yet, the server rejects the new connection.

Terminating any connection

If the server is to terminate any of the existing connections, proceed as follows:

Step	Action
1	Enter value 2 into MR -1.
2	Enter value 1 into MR 2.

Terminating all connections which exceed the minimum time of inactivity

If the server is to terminate all existing connections which have exceeded the minimum time of inactivity proceed as follows:

Step	Action
1	Enter the minimum time [ms] into MR 2.
2	Enter value 1 into MR 2.

Register

Register numbers

The register numbers to be used are calculated by adding and the controller-dependent basic register number and the module register number.

Controller/server	Basic register number	Register numbers
JC-310-JM: JetIP/TCP	230000	230000 ... 230002
JC-310-JM: STX-Debug	212000	212000 ... 212002

MR 0

Number of connections

The number of currently established connections can be read from module register 0.

Module register properties	
Values	0 ... 4 (JetIP/TCP server)
	0 ... 20 (STX debug server)

MR 1

Mode

If the maximum number of connections is active, and if the server is to establish a new connection, module registers 1 and 2 determine the behavior.

Module register properties	
Values	0 ... 2
Value after reset	1

MR 2

Minimum inactivity time

If the maximum number of connections is active, and if the server is to establish a new connection, module registers 1 and 2 determine the behavior.

Module register properties	
Values	-1 ... 2,147,483,647 [ms]
Value after reset	-1

9.3.6 Executing an ARP request

Use case

Several controllers are interconnected via the Jetter Ethernet system bus. This is the case now. Controller B is exchanged. In this case, the IP address remains the same, but the Ethernet address (MAC address) changes. This way, data interchange between controller A and the new controller B is not possible.

To enable data interchange between the two controllers again, controller A would have to be relaunched.

To prevent a relaunch of controller A, an ARP request must be executed on controller A.

Phases of an ARP request

The controller A inquires from the Jetter Ethernet system bus, which node has got which specific IP address. Controller B reports that it has got this IP address. MAC address and IP address of controller B are aligned with each other. Now, controller A is informed of the MAC address which controller B has got. From now on, data interchange is possible again.

Contents

Executing an ARP request

ARP request

When you enter the IP address of a network node into the corresponding register, the controller triggers an ARP request. This request is used for resolution of an IP address into an Ethernet address (MAC address).

R 104250

Executing an ARP request

Register properties	
Values	Valid IP address

9.4 General system registers

Introduction

In this chapter, the system command registers and the system commands will be explained in detail.

Contents

Description of system command registers

Registers - Overview

The following registers are described in this manual:

Register	Description
R 202960	System password register
R 202961	System command register
R 202962	System status register

R 202960

System password register

Enter system password 1112502132 (0x424F6F74) into this register. Then enter the required command value into the system command register. Now, the controller sets the value of this register to 0.

Register properties	
Value	1112502132 (0x424F6F74)

R 202961

System command register

Enter the system commands into this register. Then the controller executes the command. Then, it sets the value of this register to 0.

Commands	
102	**Restart the controller**
122	**Deactivate - Wait for communication**
123	**Activate - Wait for communication**
160	**Deactivate - Task switch on I/O access**
161	**Activate - Task switch on I/O access**

Commands	
170	**Deactivate - Resume task time slice** In OS version 1.22, this feature has not been implemented yet.
171	**Activate - Resume task time slice** In OS version 1.22, this feature has not been implemented yet.
310	**Load configuration files**
311	**Load module configuration**
312	**Load process data configuration for Ethernet system bus**

Register properties	
Access	System password register contains the correct password.

R 202962

System status register

The system status register lets you evaluate the system conditions.

Meaning of the individual bits		
Bit 0	**Task switch on I/O access**	
	0 =	No task switching in the application program on I/O access
	1 =	Task switching is carried out in the application program on I/O access
Bit 1	**Wait for communication**	
	0 =	The controller waits for communication requests for a short time
	1 =	The controller does not wait for communication requests
Bit 2	**JetIPScan client**	
	0 =	JetIPScan client is not active
	1 =	JetIPScan client is active
Bit 3	**Resume task time slice**	
	0 =	When a normal application task has been interrupted by a cyclic task or the Ethernet system bus publisher, the next application task is to be processed.
	1 =	When a normal application task has been interrupted by a cyclic task or the Ethernet system bus publisher, the interrupted application task is to be processed.

Register properties	
Access	Read

Description of system commands

System command 102

Restart the controller

Effect:
The controller is restarting. The effect is the same as when you switch the power supply off and on again.

Purpose:
Use this command, for example, if you have made changes to system registers or system files which become active only when the controller is rebooted.

System command 122

Deactivate - Wait for communication

Effect:
Not before there are definite requests, the controller will communicate with external communication partners.

Advantage:
The controller executes the application program faster.

Disadvantage:
On average, external communication partners have to wait longer for a response from the controller.

System command 123

Activate - Wait for communication

Effect:
The controller checks for communication requests from external partners at regular intervals of 1 to 2 ms.

Advantage:
External communication partners get a faster reply from the controller.

Disadvantage:
Application program processing takes slightly longer.

System command 160

Deactivate - Task switch on I/O access

Effect:
While the controller is accessing modules on the JX2 or JX3 system bus, it processes no other tasks of the application program.

Advantage:
The controller executes I/O accesses as fast as possible.

Disadvantage:

As certain I/O accesses are significantly slower than access to internal variables, response time of other tasks may increase.

System command 161

Activate - Task switch on I/O access

Effect:
While the controller is accessing modules on the JX2 or JX3 system bus, it processes the other tasks of the application program.

Advantage:
The execution time of certain I/O accesses which may be relatively long does not affect the response time of other tasks.

Disadvantage:
The run time of the other tasks influences the execution time of several I/O accesses.

System command 170

Deactivate - Resume task time slice

Note: In OS version 1.22.0.00, this feature has not been implemented yet.

Effect:
When a normal application task has been interrupted by a cyclic task or the Ethernet system bus publisher, the next application task is to be processed. The remaining time of the time slice of the interrupted task lapses for one cycle.

Advantage:
The total cycle time for processing all tasks is not influenced so much by the cyclic events.

Disadvantage:
This way, the interrupted task is assigned less runtime.

System command 171

Activate - Resume task time slice

Note: In OS version 1.22.0.00, this feature has not been implemented yet.

Effect:
When a normal application task has been interrupted by a cyclic task or the Ethernet system bus publisher, the interrupted application task is to be processed. This way, the interrupted task is processed for the remaining time of its time slice.

Advantage:
The interrupted task is assigned its total runtime.

Disadvantage:

The total cycle time for processing all tasks is influenced by the cyclic events to a greater extend.

System command 310

Load configuration files

Effect:

The controller loads the module configuration file (ModConfig.da) and the configuration files for process data communication on the Jetter Ethernet system bus (Publisher.pub, Subscriber.sub) from the file system. This corresponds to a combination of commands 311 and 312.

Purpose:

Once the transfer of these files into the controller's file system is completed, system command 310 enables their contents.

System command 311

Load module configuration

Effect:

The controller loads the module configuration file (ModConfig.da) from the file system.

Purpose:

Once the transfer of this file into the controller's file system is completed, system command 311 enables its contents.

System command 312

Load process data configuration for Ethernet system bus

Effect:

The controller loads the configuration files for process data communication on the Jetter Ethernet system bus (Publisher.pub, Subscriber.sub) from the file system.

Purpose:

Once the transfer of these files into the controller's file system is completed, system command 312 enables their contents.

9.5 Runtime registers

Introduction

The JC-310-JM provides several registers which are incremented by the operating system at regular intervals.

Application

These registers can be used to easily carry out time measurements in the application program.

Contents

Description of the runtime registers

Register overview

The device is equipped with the following runtime registers:

Register	Description
R 201000	Application time base in milliseconds
R 201001	Application time base in seconds
R 201002	Application time base in R 201003 * 10 ms
R 201003	Application time base units for R 201002
R 201004	System time base in milliseconds
R 201005	System time base in microseconds

R 201000

Application time base in milliseconds

Every millisecond this register is incremented by one.

Register properties	
Values	-2,147,483,648 ... 2,147,483,647 (overflowing)

R 201001

Application time base in seconds

Every second this register is incremented by one.

Register properties	
Values	-2,147,483,648 ... 2,147,483,647 (overflowing)

R 201002

Application time base in application time base units

Every [R 201003] * 10 ms this register value is incremented by one. Using the reset value 10 in register 201003, this register is incremented every 100 ms.

Register properties	
Values	-2,147,483,648 ... 2,147,483,647 (overflowing)

R 201003

Application time base units for R 201002

This register contains the multiplier for runtime register R 201002.

Register properties	
Values	1 ... 2,147,483,647 (* 10 ms)
Value after reset	10 (--> 100 ms)
Enabling conditions	After at least 10 ms

R 201004

System time base in milliseconds

Every millisecond this register value is incremented by one.

Register properties	
Values	-2,147,483,648 ... 2,147,483,647 (overflowing)
Type of access	Read

R 201005

System time base in microseconds

Every microsecond this register value is incremented by one.

Register properties	
Values	-2,147,483,648 ... 2,147,483,647 (overflowing)
Type of access	Read

9.6 Monitoring interface activities

Introduction

Several servers for variables have been integrated into the controller to make variables used within the controller accessible from outside. These servers support several protocols on different interfaces. The servers do not require any programming in the application program, but process requests from external clients on their own.

This chapter explains one possibility for detecting from within the application program whether communication with the servers takes place through these interfaces.

Monitored interface activities

The following interface activities can be monitored:

- JetIP server via Ethernet interface
- STX debug server via Ethernet interface

Description

The monitoring function for interface activities can be used, amongst others, for the following scenarios:

- Plants requiring process visualization to ensure safe operation. They can be transferred into a save state if communication fails.
- When the service technician connects an HMI, the application program automatically displays additional status information.

Contents

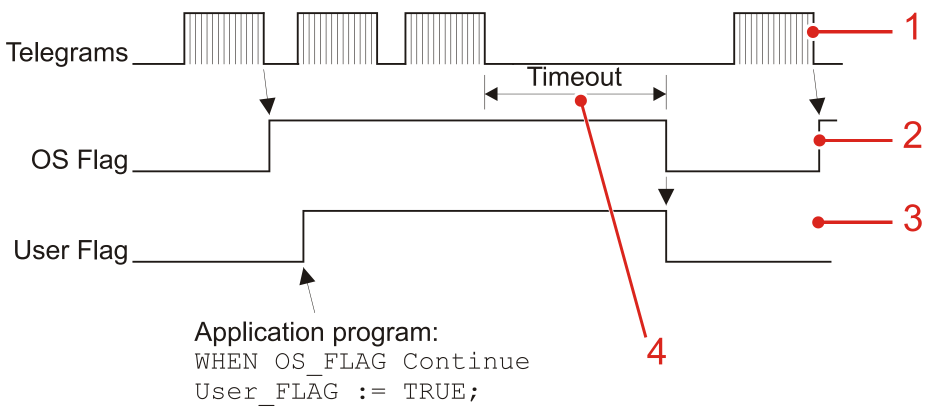
Operating principle

Introduction

The application program monitors the activity of a client communicating with a server of the device JC-310-JM by means of two special flags and one special register per interface.

Overview

The illustration below shows the interdependence between interface activity and the two special flags, as well as the special register:



Number	Element	Description
1	Telegrams	The client places requests to the server.
2	OS flag	OS flag set by the device JC-310-JM after receiving a request
3	User flag	You must set the user flag in the application program once the device has set the OS flag.This indicates that the connection has temporarily been disrupted even if the device resets the OS flag very quickly.
4	Timeout	Time of inactivity after which the OS resets both special flags. This time can be set in a special register.

Description

Interface activities are monitored as follows:

Step	Description
1	Enter the desired value into the timeout register of the application program. This way, the monitoring mode is activated as well.
2	When the controller receives the next telegram, the device JC-310-JM sets the corresponding OS flag.
3	If the OS flag has been set, the application program also sets the respective user flag.
4	Each new telegram causes the timeout to restart.
5	If telegrams cease to arrive, both special flags are reset by the controller upon expiry of the timeout interval.

Step	Description
6	The application program detects that the device has reset the special flags and therefore takes appropriate action.
7	When further telegrams start arriving, the device sets the corresponding OS flag. The user flag, however, remains reset.

Programming

Registers/flags - Overview

For interface monitoring, the device provides the following registers and flags:

Timeout registers

Register	Interface	Description
R 203000	JetIP (Ethernet)	■ Visualization applications ■ Controller networking
R 203005	STX debugging (Ethernet)	■ JetSym via Ethernet

Special flags

Flags	Interface	Description
F 2088	JetIP (Ethernet)	OS flag
F 2089		User flag
F 2098	STX debugging (Ethernet)	OS flag
F 2099		User flag

R 203000

Timeout in the case of JetIP (Ethernet)

This register contains the timeout for the JetIP server (Ethernet) in milliseconds.

Register properties	
Values	0 ... 2,147,483,647 [ms]
Value after reset	0 (monitoring disabled)

R 203005

Timeout in the case of STX debugging (Ethernet)

This register contains the timeout for the STX debug server (Ethernet) in milliseconds.

Register properties	
Values	0 ... 2,147,483,647 [ms]
Value after reset	0 (monitoring disabled)

Enabling the monitoring function

To enable monitoring of interface activities, proceed as follows:

Step	Action
1	Enter the desired value into the timeout register of this interface.
2	Wait until the controller has set the OS flag of this interface.
3	Set the corresponding user flag.

Detecting a timeout

To detect a timeout, proceed as follows:

Step	Action	
1	Enable monitoring of interface activities (see above).	
2	Wait until the controller has reset the user flag of this interface. **Result**: A timeout has occurred.	
3	Check the corresponding OS flag.	
	If ...	**... then ...**
	... the OS flag is set,	... the connection was temporarily disrupted.
	... the OS flag is reset,	... the connection is still disrupted.

9.7 E-mail

Introduction

The user creates template files for e-mails. Into these templates, the controller JC-310-JM 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 gives a description on how to configure the e-mail feature in the JC-310-JM, and on how to create and send e-mails.

Activating the E-mail feature

To activate the e-mail feature in the JC-310-JM, the following requirement must be met:

- A valid e-mail configuration file **/EMail/email.ini** exists while the controller is booting.

If both requirements have been met, the corresponding bit in the web status register is set.

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.
- IP networks

Contents

9.7.1 Configuring the E-mail feature

Introduction

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

During the boot process, the JC-310-JM 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 is 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 are known.

To obtain this information contact your network administrator.

Contents

Structure of the configuration file

Introduction

The configuration of the e-mail client in the controller is based on the contents of the file **/EMAIL/email.ini**. The JC-310-JM 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 ";".

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 ▪ Log-on parameters
[POP3]	▪ IP address and port number of the POP3 server ▪ 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
```

Authentication

This type of authentication requires the JC-310-JM to log on at the SMTP server before sending an e-mail. During the logon process USER and PASSWORD must be entered. The JC-310-JM supports the following authentication procedures:

- LOG-ON
- PLAIN
- CRAM-MD5

Configuration values

IP	
In the given example	192.168.40.1
Description	IP address of the SMTP server; can also be specified as name.
Allowed values	■ > 1.0.0.0 ■ < 223.255.255.255
Illegal values	■ Network address ■ 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
Allowed values	■ > 0 ■ < 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
Allowed 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. If this entry exists, the entry PASSWORD is required, too.
Allowed 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. If this entry exists, the entry USER is required, too.
Allowed values	String of 63 characters max.
In case of missing entry	SMTP authentication will not be carried out.

Section [POP3]

Introduction

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

Only in case the e-mail server requires authentication via POP3-before-SMTP, this section is required.

Example:

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

Authentication

This type of authentication requires the JC-310-JM 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; can also be specified as name.
Allowed values	■ > 1.0.0.0 ■ < 223.255.255.255
Illegal values	■ Network address ■ 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
Allowed values	■ > 0 ■ < 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. If this entry exists, the entry PASSWORD is required, too.
Allowed 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. If this entry exists, the entry USER is required, too.
Allowed values	String of 63 characters max.
In case of missing entry	POP3 log-on will not be carried out.

Section [DEFAULT]

Introduction

In this section, specify the name of an e-mail template file which contains default settings for e-mails. If the respective section is not available in the respective e-mail template, the JC-310-JM applies these default settings for sending an e-mail message.

Example

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

Related topics

- **Structure of template file** (see page 204)

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
```

9.7.2 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.

Contents

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-310-JM 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* (see page 110)).

[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]`
	`JetControl@jetter.de`

[TO]	
Description	E-mail addressee
Comments	Several addressees are separated by the semicolon “;”.
Length	255 characters

Example	`[TO]`
	`service@mydomain.com`

[CC]	
Description	Additional e-mail addressee(s)
Comments	Several addressees are separated by the semicolon “;”.
Length	255 characters

Example	`[CC]`
	`service@mydomain.com;hotline@mydomain.com`

[SUBJECT]	
Description	Subject
Length	255 characters

Example	`[SUBJECT]`
	`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]`
	`/logfiles/error_report.log`

[MESSAGE]	
Description	E-mail message text
Comments	Text only message
Length	65,535 characters

Example	`[MESSAGE]`
	`Have a nice day !`
	`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	<JW: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	
Function	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 (standard type)	Integer, decimal
INTX	Integer, hexadecimal
INTB	Integer, binary
BOOL	Register content = 0 --> Display: 0 Register content != 0 --> Display: 1
REAL	Floating point, decimal
STRING	Text register

Example:

```
JW:DTAG name="R1000250" type="REAL" format="+0####.###"
factor="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 (standard type)	Flag = 0 --> Display: 0
	Flag = 1 --> Display: 1

STRING	Flag = 0 --> Display: FALSE
	Flag = 1 --> Display: TRUE

Example:

```
<JW: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 (standard type)	Input = 0 --> Display: 0
	Input = 1 --> Display: 1

STRING	Input = 0 --> Display: OFF
	Input = 1 --> Display: ON

Example:

```
<JW:DTAG name="I100000308" type="STRING" />
```

Result:
The state of input 100000308 is displayed as string "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 (standard type)	Output = 0 --> Display: 0
	Output = 1 --> Display: 1

STRING	Output = 0 --> Display: OFF
	Output = 1 --> Display: ON

Example:

```
<JW:DTAG name="O100000308" />
```

Result:
The state of output 100000308 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:

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

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

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

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

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

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

```
<JW: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:

```
<JW: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.

```
<JW: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.

```
<JW: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.

```
<JW: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.

```
<JW: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.

```
<JW: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.

```
<JW: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.

```
<JW: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.

9.7.3 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 device JC-310-JM 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 a function call via e-mail is not possible. 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() has been described in detail in the online help of JetSym.

Declaration of functions:

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

9.7.4 Registers

Introduction

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

Contents

Overview of registers

Introduction

The device JC-310-JM makes the registers available from which you can query the status of e-mail processing.

Register overview

Register	Description
202930	Web status
292932	IP address of the SMTP server
292933	IP address of the POP3 server
292934	Port number of the 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**	
	1 =	available
Bit 1	**HTTP server**	
	1 =	available
Bit 2	**E-mail**	
	1 =	available
	Bit 2 is only set, if configuration file **/EMAIL/email.ini** exists.	
Bit 3	**Data file function**	
	1 =	available
Bit 4	**Modbus/TCP**	
	1 =	existing
Bit 5	**Modbus/TCP**	
	1 =	available
Bit 7	**FTP client**	
	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-310-JM.
	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)

The ID of the task that is just sending an e-mail can be seen from this register.

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	

9.8 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.

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() has been described in detail in the online help of JetSym.

Declaration of functions:

```
Function QSort(DataPtr: Int, ElementCnt: Int, ElementSize: Int,
SortOffset: Int, SortType: STXBASETYPE, SortMode: QSORTMODE): Int;
```

9.9 Modbus/TCP

Introduction

This chapter describes the functions of the Modbus/TCP server and client integrated into JC-310-JM.

Enabling the Modbus/TCP feature

In case of JC-340, you must order the controller plus option -M. In any other controller of this series, the feature Modbus/TCP remains activated.

If this requirement has been met, bits 4 and 5 in the Web status register 202930 are set.

Required programmer's skills

To be able to use the functions described in this chapter, the following skills are required:

- Modbus/TCP and the supported commands
- IP networks

Contents

9.9.1 Modbus/TCP server

Introduction

If a valid license is available, if the Modbus/TCP feature has been enabled and the Modbus/TCP server has been launched, an external client can access registers, flags, inputs and outputs.

This chapter covers the addressing process and describes 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.

Contents

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 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.

Hence, output 100000205 of a peripheral module JX3-DO16 will be set.

Supported commands - Class 0

fc 3	**read multiple registers** Reading register sets The starting register number within JC-310-JM 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 The starting register number within JC-310-JM 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

The output number within the JC-310-JM 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

The input number within JC-310-JM 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.

The starting register number within JC-310-JM 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

The output number within the JC-310-JM 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

The starting register number within JC-310-JM 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 The output number within the JC-310-JM 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 The starting register number within the JC-310-JM is calculated as follows: Register number specified in the command plus the content of R 272702 *Register Offset*.

9.9.2 Modbus/TCP client

Introduction

The Modbus/TCP client included in JC-310-JM supports only Class 0 Conformance.

In this class, commands for reading and writing multiple registers are used. Up to 125 registers with a width of 16 bits can be transmitted in one frame.

As protocol ID "0" is used. Assignment of sent 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 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 noncyclical 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.

Only one connection is established to each Modbus/TCP server (IP address and port) irrespective of the number of communication units which have been 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 telegram 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 function	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;

9.9.3 Modbus/TCP client with STX variables

Introduction

The Modbus/TCP client included in JC-310-JM supports only Class 0 Conformance.

In this class, commands for reading and writing multiple registers are used. 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.

Only one connection is established to each Modbus/TCP server (IP address and port) irrespective of the number of communication units which have been 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 have been described in detail in the online help of JetSym.

System function	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;

9.10 User-programmable IP interface

The user-programmable IP interface

The user-programmable IP interface allows to send or receive any data via Ethernet interface on the JC-310-JM using TCP/IP or UDP/IP. When using this feature, data processing is completely carried out by the application program.

Applications

The user-programmable IP interface allows the programmer to carry out data exchange via Ethernet connections which do not use standard protocols, such as FTP, HTTP, JetIP or Modbus/TCP. The following applications are possible:

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

Required programmer's skills

To be able to program user-programmable IP interfaces the following knowledge of data exchange via IP networks is required:

- IP addressing (e.g. IP address, port number, subnet mask)
- TCP (e.g. connection establishment/termination, data stream, data backup)
- UDP (e.g. datagram)

Restrictions

For communication via user-programmable IP interface, the programmer must not use any ports which are already used by the operating system of the controller. Therefore, do not use the following ports:

Protocol	Port number	Default value	User
TCP	Depending on the FTP client	20	FTP server (data)
TCP	21		FTP server (controller)
TCP	23		System logger
TCP	80		HTTP server
TCP	From the file /EMAIL/email.ini	25, 110	E-mail client
TCP	502		Modbus/TCP server
TCP, UDP	1024 - 2047		Various
TCP, UDP	IP configuration	50000, 50001	JetIP
TCP	IP configuration	52000	Debug server

9 Programming

Contents

9.10.1 Programming

Introduction

The user-programmable IP interface is used to carry out data exchange between application program and network client via TCP/IP or UDP/IP connections. For this purpose, function calls are used. These function calls are included in the programming language of the JC-310-JM. To program this feature proceed as follows:

Step	Action
1	Initializing the user-programmable IP interface
2	Open connections
3	Transfer data
4	Terminate the connections

Technical data

Technical data of the user-programmable IP interface:

Feature	Description
Number of connections	20
Maximum data size	4,000 byte

Restrictions

While the controller JC-310-JM is processing one of the functions of the user-programmable IP interface, tasks having called the functions should not be stopped through `TaskBreak` or restarted through `TaskRestart`.

Failure to do so could result in the following errors:

- Connections do not open
- Data loss during sending or receiving
- Connections remain open unintentionally
- Connections are closed unintentionally

Contents

Initializing the user-programmable IP interface

Introduction

This function must be initialized each time the application program is launched.

Function declaration

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

Return value

The following return value is possible:

Return value	
0	Always

How to use this function

The function is used and its return value assigned to a variable for further utilization in the following way:

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

Operating principle

The device JC-310-JM processes this function in the following steps:

Step	Description
1	The device JC-310-JM closes all open connections of the user-programmable IP interface.
2	The device JC-310-JM initializes all OS-internal data structures of the user-programmable IP interface.

Related topics

- **Establishing a connection** (see page 233)
- **Terminating a connection** (see page 242)
- **Sending data** (see page 237)
- **Receiving data** (see page 239)

Establishing a connection

Introduction

Before data can be sent or received, a connection has to be established. Here, the following criteria have to be discerned:

- Which transaction log (TCP or UDP) has to be used?
- Is it a client or a server that has to be installed?

Function declaration

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

Function parameters

Description of the function parameters:

Parameter	Value	Comment

`ClientServerType`	Client = 1 = CONNTYPE_CLIENT
	Server = 2 = CONNTYPE_SERVER	

`IPType`	UDP/IP = 1 = IPTYPE_UDP TCP/IP = 2 = IPTYPE_TCP	
`IPAddr`	Valid IP address	Required only for TCP/IP client
`IPPort`	Valid IP port number	Will be ignored for UDP/IP client
`Timeout`	0 ... 1,073,741,824 [ms]	0 = infinitely

Return value

If the return value was positive, the connection could be established. If the returned value was negative, an error occurred and the connection could not be established.

Return value	
> 0	A positive return value must be stored in a variable. It must be made available as a handle at activating the functions `Send data`, `Receive data`, and `Terminate connection`.
-1	Error during connection set-up
-2	Internal error
-3	Invalid parameter
-8	Timeout

Using this function with a TCP/IP client

If a client is to establish a TCP/IP connection to a server, you can invoke the function and assign the return value of a variable for further evaluation as follows:

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

Functioning principle with a TCP/IP client

The task stops at the program line until the connection is established or the specified timeout has elapsed. This function is processed in the following steps:

Step	Description	
1	The device JC-310-JM tries to establish a TCP/IP connection via port 46000 to the network client with IP address 192.168.75.123.	
2	**If ...**	**... then ...**
	the network client has accepted the connection,	the function is terminated and a positive value is returned as handle for further access to the connection.
	the connection could not be established and the timeout of 10 seconds has not elapsed yet,	step 1 is carried out.
	an error has occurred or the timeout has elapsed,	the function is terminated and a negative value is returned.

Using this function with a TCP/IP server

If a server is to establish a TCP/IP connection to a client, you can invoke the function and assign the return value of a variable for further evaluation as follows:

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

Functioning principle with a TCP/IP server

The task stops at the program line until the connection is established or the specified timeout has elapsed. This function is processed in the following steps:

Step	Description	
1	The device JC-310-JM sets up TCP/IP port 46000 for receiving connection requests.	
2	**If ...**	**... then ...**
	the network client has established a connection,	no further connection requests to this port are accepted, the function is terminated and a positive value is returned as handle for further access to the connection.
	the connection could not be established and the timeout of 100 seconds has not elapsed yet,	the system waits for a connection to be established.
	an error has occurred or the timeout has elapsed,	the function is terminated and a negative value is returned.

Using this function with a UDP/IP client

If a client is to establish a UDP/IP connection to a server, you can invoke the function and assign the return value of a variable for further evaluation as follows:

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

Functioning principle with a UDP/IP client

UDP is a connectionless communication mode. For this reason, the device JC-310-JM opens only one communication channel for sending data to a network client. This function is processed in the following steps:

Step	Description	
1	The device JC-310-JM sets up a UDP/IP communication channel for sending data.	
2	**If ...**	**... then ...**
	no error has occurred,	the function is terminated and a positive value is returned as handle for further access to the connection.
	an error has occurred,	the function is terminated and a negative value is returned.

Using this function with a UDP/IP server

If a server is to establish a UDP/IP connection to a server, you can invoke the function and assign the return value of a variable for further evaluation as follows:

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

Functioning principle with a UDP/IP server

UDP is a connectionless communication mode. For this reason, the device JC-310-JM opens only one communication channel for receiving data from a network client. This function is processed in the following steps:

Step	Description	
1	The device JC-310-JM sets up a UDP/IP communication channel at port 46000 for receiving data.	
2	**If ...**	**... then ...**
	no error has occurred,	the function is terminated and a positive value is returned as handle for further access to the connection.
	an error has occurred,	the function is terminated and a negative value is returned.

Related topics

- **Terminating a connection** (see page 242)
- **Sending data** (see page 237)
- **Receiving data** (see page 239)
- **Initializing the user-programmable IP interface** (see page 232)

Sending data

Introduction

Data can be sent via a previously established TCP/IP connection or via a UDP/IP connection of a client.

Via UDP/IP connection of a server data cannot be sent, but only received.

Function declaration

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

Function parameters

Description of the function parameters:

Parameter	Value	Comment
`IPConnection`	Handle	Return value of the function `ConnectionCreate()`
`IPAddr`	Valid IP address	Required only for UDP/IP client
`IPPort`	Valid IP port number	Required only for UDP/IP client
`SendData`	Address of the data block to be sent	
	1 ... 4,000	Data block length in bytes

Return value

The following return values are possible:

Return value	
0	Data have been sent successfully.
-1	Error when sending, e.g. connection interrupted.
-3	Invalid handle, e.g. sending via a UDP/IP server.

Using this function with a TCP/IP connection

If data are to be sent via a TCP/IP connection, you can invoke the function and assign the return value of a variable for further evaluation as follows:

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

Functioning principle with a TCP/IP connection

When using TCP/IP, data are sent via a previously opened connection. Therefore, specification of the IP address and IP port number is not required anymore and can be ignored in the function.

In the following cases, the task is not processed further after issuing the function call:

- The data have been sent and their reception confirmed.
- An error has occurred.

Using this function with a UDP/IP client

If, with a client, data are to be sent via a UDP/IP connection, you can invoke the function and assign the return value of a variable for further evaluation as follows:

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

Functioning principle with a UDP/IP client

With UDP/IP there is no connection between two given network clients. Therefore, with each function call data can be sent to another client or another port. The task will pause at the function call until the data are sent.

You will not get any acknowledgment of the remote network client having received the data.

Related topics

- **Initializing the user-programmable IP interface** (see page 232)
- **Establishing a connection** (see page 233)
- **Terminating a connection** (see page 242)
- **Receiving data** (see page 239)

Receiving data

Introduction

Data can be sent via a previously established TCP/IP connection or via a UDP/IP connection of a server.

Via UDP/IP connection of a client data cannot be received, but only sent.

Function declaration

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

Function parameters

Description of the function parameters:

Parameter	Value	Comment
`IPConnection`	Handle	Return value of the function `ConnectionCreate()`
`IPAddr`	Address of a variable for saving the IP address of the sender	Required only for UDP/IP server
`IPPort`	Address of a variable for saving the IP port number of the sender	Required only for UDP/IP server
`ReceiveData`	Address of the data block to be received	
`DataLen`	1 ... 4,000	Maximum data block length in bytes
`Timeout`	0 ... 1,073,741,824 [ms]	0 = infinitely

Return value

The following return values are possible:

Return value	
> 0	Number of received data bytes
-1	Error when receiving data, e.g. connection interrupted.
-3	Invalid handle, e.g. receiving data via a UDP/IP client.
-8	Timeout

Using this function with a TCP/IP connection

If data are to be received via a TCP/IP connection, you can invoke the function and assign the return value of a variable for further evaluation as follows:

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

Functioning principle with a TCP/IP connection

When using TCP/IP, data are sent via a previously opened connection. Therefore, specification of the IP address and IP port number is not required any more and can be ignored in the function.

In the following cases, the task is not processed further after issuing the function call:

- The data have been received
- An error has occurred

In case of a TCP/IP connection, data are transmitted as data stream.

The device JC-310-JM processes this function in the following steps:

Step	Description	
1	The device JC-310-JM waits until data have been received, but no longer than the specified timeout.	
2	**If ...**	**... then ...**
	the timeout has elapsed or the connection has been terminated,	the function is exited and an error message is issued.
	data have been received,	they are copied to the receiving buffer given along with the data (but not exceeding the amount given along with the data). Then, the function continues with stage 3.
3	**If ...**	**... then ...**
	more data have been received than could have been copied into the receiving buffer,	these are buffered by the JC-310-JM to be fetched by further function calls.
4	The function is exited and the number of data, which have been copied into the receiving buffer, is returned.	

Using this function with a UDP/IP server

If, with a server, data are to be received via a UDP/IP connection, you can invoke the function and assign the return value of a variable for further evaluation as follows:

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

Functioning principle with a UDP/IP server

In the following situations, the task is not processed further after issuing the function call:

- All data have been received.
- An error has occurred.

In case of a UDP/IP connection, data are transmitted as datagram.
The controller processes this function in the following steps:

Step	Description	
1	The device JC-310-JM waits until all data of a datagram have been received, but no longer than the specified timeout.	
2	**If ...**	**... then ...**
	the timeout has elapsed or the connection has been terminated,	the function is exited and an error message is issued.
	data have been received,	they are copied to the receiving buffer given along with the data (but not exceeding the amount given along with the data). Then, the function continues with stage 3.
3	**If ...**	**... then ...**
	... more data have been received than could be copied into the receiving buffer - that is, if the sent datagram is too large,	... these data are discarded.
4	The sender's IP address and IP port number are transferred into the variables which are given along with the data.	
5	The function is exited and the number of data, which have been copied into the receiving buffer, is returned.	

Related topics

- **Initializing the user-programmable IP interface** (see page 232)
- **Establishing a connection** (see page 233)
- **Terminating a connection** (see page 242)
- **Sending data** (see page 237)

Terminating a connection

Introduction

Clear all connections which are no longer required as the number of concurrently opened connections is limited.

Function declaration

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

Function parameters

Description of the function parameters:

Parameter	Value	Comment
`IPConnection`	Handle	Return value of the function `ConnectionCreate()`

Return value

The following return values are possible:

Return value	
0	Connection terminated and deleted
-1	Invalid handle

How to use this function

This way, you can invoke the function and assign its return value to a variable for further utilization:

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

Related topics

- **Establishing a connection** (see page 233)
- **Sending data** (see page 237)
- **Receiving data** (see page 239)
- **Initializing the user-programmable IP interface** (see page 232)

9.10.2 Registers

Introduction

This chapter describes the registers of the JC-310-JM which contain the current connection list of the user-programmable IP interface. These registers can be used for debugging or diagnostic purposes. However, they can't be used for other functions such as establishing or terminating a connection.

Contents

Register numbers

Introduction

Data of one connection each are displayed within the registers of a coherent register block. The basic register number of this block is dependent on the controller.

Register numbers

Device	Basic register number	Register numbers
JC-310-JM	350000	350000 ... 350007

Determining the register number

In this chapter only the last figure of a register number is specified, for example MR 1. To determine the complete module register number, add to this figure the basic register number of the corresponding device, for example 350000.

Register overview

Register	Description
MR 0	Selecting a connection
MR 1	Type of connection
MR 2	Transport protocol
MR 3	IP address
MR 4	IP port number
MR 5	State
MR 6	Number of sent bytes
MR 7	Number of received bytes

Register description

Introduction

The operating system manages the established connections in a list. Module register MR 0 *Selection of a connection* is used to copy connection details into other registers of a register block.

MR 0

Selecting a connection

Connections are selected by writing values to this register. Read access to this register is used to display whether the following registers contain connection details.

Module register properties		
Reading values	0	Connection exists
	-1	Connection does not exist

Module register properties		
Writing values	0	Address the first connection in the list
	> 0	Address the next connection in the list
	< 0	Address the previous connection in the list

MR 1

Type of connection

The value in this register shows whether the connection is a client or a server connection.

Module register properties		
Values	1	Client
	2	Server

MR 2

Transport protocol

The value in this register shows whether TCP or UDP is used as transport protocol.

Module register properties		
Values	1	UDP
	2	TCP

MR 3

IP address

The value in this register shows the configured IP address.

Module register properties	
Values	0,0,0,0 ... 255,255,255,255

MR 4

IP port number

The value in this register shows the configured IP port number.

Module register properties	
Values	0 ... 65,535

MR 5

State

The value in this register shows status the connection is currently in.

Module register properties		
Values	0	Connection terminated
	1	Connection is being established
	2	Connection is established
	3	TCP/IP server: Waiting for connection request from client
	4	Internal usage

MR 6

Number of sent bytes

The value in this register shows the number of data bytes sent via the given connection. Since this is a signed 32-bit register and the sent bytes are added each time, the number range may be exceeded from the positive maximum value to the negative maximum value.

Module register properties	
Values	-2,147,483,648 ... 2,147,483,647

MR 7

Number of received bytes

The value in this register shows the number of data bytes received via the given connection. Since this is a signed 32-bit register and the received bytes are added each time, the number range may be exceeded from the positive maximum value to the negative maximum value.

Module register properties	
Values	-2,147,483,648 ... 2,147,483,647

10 Updating the operating system

Introduction

Jetter AG is continuously striving to enhance the operating systems of its controller module JC-310-JM. Enhancing means adding new features, upgrading existing functions and fixing bugs.

This chapter describes how to update the OS of a JC-310-JM.

Downloading an operating system

You can download operating systems from the Jetter AG **homepage http://www.jetter.de**. For OS download, please turn to the respective product information page.

Transferring the OS file

You have got several options to transfer the OS file to the controller module:

- From within the programming tool JetSym
- Via FTP connection
- From the application program

Contents

OS update by means of JetSym

Introduction

The programming tool JetSym offers an easy way to transfer an OS file to the JC-310-JM.

Prerequisites

- An OS file for the JC-310-JM must be available.
- A UDP/IP and a TCP/IP connection between programming tool and JC-310-JM is possible.
  The number of the IP port is set in the configuration memory as IP basic port number for JetIP communication.
- While booting, the module must wait for the OS update to load, or the OS must start up.
  Note:
  Do not switch off the module!

Updating the OS

To update the OS, proceed as follows:

Step	Action
1	Select in the JetSym menu **Build** the menu item **Update OS.** Alternative: In the **Advanced Configuration** dialog of the Hardware Manager, click on the button **Update OS**. **Result:** A file selection dialog opens.
2	Select the new OS file here. **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 newly installed OS, re-boot the module.

Operating system update via FTP

Introduction

Using an FTP client an OS file can be transferred to the JC-310-JM.

Prerequisites

- An OS file for the JC-310-JM must be available.
- An FTP connection to the JC-310-JM must be possible.
- The login parameters for a user with administrator or system rights are at hand.
- The operating system of the JC-310-JM must be running.
  Note:
  Do not switch off the module!

Updating the OS

To update the OS, proceed as follows:

Step	Action
1	Establish an FTP connection to the JC-310-JM.
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 newly installed OS, re-boot the module.

Operating system update from within the application program

Introduction

The file functions included in the STX language allow carrying out a program-controlled OS update of a JC-310-JM from within an OS file.

Prerequisites

- An OS file must be available in the file system of the JC-310-JM.
- The operating system of the JC-310-JM and the application program must be running.
  Note:
  Do not switch off the module!

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 transferred OS, for example by writing to the system command register, re-boot the JC-310-JM.

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;
```

11 Application program

Introduction

This chapter describes how to store the application program in JC-310-JM. The user determines the program that is to be executed.

Required programmer's skills

This chapter requires knowledge on how to create application programs in JetSym and how to transmit them via the file system of the JC-310-JM.

Contents

Application program - Default path

Introduction

When uploading the application program from JetSym to the JC-310-JM, this program is stored as a file to the internal flash disk. The device enters the path and file name into the file **/app/start.ini**.

Path and file name

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

/app/start.ini - Structure

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

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

Example:

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

The application program is loaded from the file **/app/test_program/test_program.es3**.

Loading an application program

Introduction

At reboot of the application program via JetSym or after booting the JC-310-JM, the application program is loaded and executed via the file system. To this end, DIP switch sliders 9 and 10 must be in position *RUN* (slider 9 in position *ON* and slider 10 in position *OFF*).

The loading process

The application program is loaded by the JC-310-JM's OS as follows:

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

12 Quick reference - JC-310-JM

Corresponding OS version

This quick reference summarizes in brief the registers and flags of the controller module JC-310-JM running OS version 1.22.0.00.

General overview - Registers

100000 ... 100999	Electronic Data Sheet (EDS)
101000 … 101999	Configuration
104000 ... 104999	Ethernet
107500 ... 107599	Flash disk
108000 ... 108999	CPU
200000 ... 209999	General system registers
210000 ... 219999	Application program
230000 ... 239999	Networking via JetIP
250000 ... 259999	Ethernet system bus
260000 ... 269999	RemoteScan
270000 ... 279999	Modbus/TCP
290000 ... 299999	E-mail
310000 ... 319999	File system/data files
320000 ... 324999	FTP client
350000 ... 359999	User-programmable IP interface
380000 ... 389999	Error history
390000 ... 399999	I/O networking
1000000 ... 1029999	Application registers (volatile; Int/Float)
200012000 ... 200012999	Local JetMove registers
	Networking via Jetter Ethernet system bus GNN: nnn = 000 ... 199:
1nnn020000 ... 1nnn179999	JX3 module registers
1nnn980000 ... 1nnn980199	Indirect access via local R 236xxx
1nnn990000 ... 1nnn999999	Indirect access with variable target window

I/Os - General overview

20001 ... 36000	Virtual I/Os for RemoteScan
1nnn010201 ... 1nnn011716	JX3 modules via JX3-BN-ETH GNN: nnn = 000 ... 199:

Flags - General overview

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

Electronic Data Sheet (EDS)

100500	Interface (0 = CPU)
100501	Module number (0)
	If <100500> = 0: The EDS of the JC-310-JM is displayed.
[Identification]	
100600	Internal version number
100601	Module ID
100602 ... 100612	Module name (register string)
100613	PCB revision
100614	PCB options
[Production]	
100700	Internal version number
100701 ... 100707	Serial number (register string)
100708	Day
100709	Month
100710	Year
100711	TestNum.
100712	TestRev.
[Features]	
100800	Internal version number
100801	MAC address (Jetter)
100802	MAC address (device)
100804	Switch
100805	STX
100808	CAN bus
100811	Intelligent slave modules
100812	HTTP/e-mail
100813	Modbus/TCP

Configuration

	From the file /system/config.ini
101100	IP address
101101	Subnet mask
101102	Default gateway
101103	DNS server
101132	HOSTNAME suffix type
101133 ... 101151	HOSTNAME (register string)
101164	Port number for JetIP
101165	Port number for STX debugger
	Used by the system
101200	IP address
101201	Subnet mask
101202	Default gateway
101203	DNS server
101232	HOSTNAME suffix type
101233 … 101251	HOSTNAME (register string)
101264	Port number for JetIP
101265	Port number for STX debugger
101299	Saving the settings (0x77566152)
101908	CRC of ModConfig.da

Ethernet

	Phy/Switch
104000	Options
104010 ... 104016	FEC statistics
104020	Status Bit 0: Status is valid Bit 1: Link Bit 2: Duplex Bit 3: Crossover Bit 4: 10 MBit/s Bit 5: 100 MBit/s
	Ethernet
104100 ... 104156	MIB counter
	ARP

104200	Transmitted requests
104201	Received requests
104202	Transmitted responses
104203	Received responses
104204	Dynamic entries
104205	Static entries
104206	Obsolete entries
104250	Executing an ARP request
	IP
104500	Transmitted packets
104501	Transmitted bytes
104502	Received packets
104503	Received bytes
104504	Invalid packets
104505	Discarded received packets
104506	Checksum error at reception
104507	Discarded transmitted packets
104508	Transmitted fragments
104509	Received fragments
104531	Current IP address (rw)
104532	Current subnet mask (rw)
104533	Current default gateway (rw)
	TCP
104800	Transmitted packets
104801	Transmitted bytes
104802	Received packets
104803	Received bytes
104804	Invalid packets
104805	Discarded received packets
104806	Checksum error
104807	Connections
104808	Disconnections
104809	Discarded connections
104810	Repeated transmitted packets
	UDP
104900	Transmitted packets
104901	Transmitted bytes
104902	Received packets
104903	Received bytes
104904	Invalid packets
104905	Discarded received packets
104906	Checksum error

Flash disk

107500	Status
107501	Command 30: Read statistics
	Sector statistics
107510	Total
107511	Used
107512	Blocked
107513	Unassigned
	Byte statistics
107520	Total
107521	Used
107522	Blocked
107523	Unassigned

CPU

108002	All LEDs on/off (bit-coded)
	Bit 0: LED **RUN**
	Bit 1: LED **ERR**
108003	LED **RUN**
	0 = OFF
	1 = Flashing slowly
	2 = Flashing fast
	3 = ON
108004	LED **ERR**
	0 = OFF
	1 = Flashing slowly
	2 = Flashing fast
	3 = ON
108010	DIP switch - All switches
108011	DIP switch - Address
108015	DIP switch 9 and 10
	1 = LOAD
	2 = RUN
	3 = STOP
108021	CPU board revision

General system registers

200000	OS version (major * 100 + minor)	
200001	Application program is running (bit 0 = 1)	
	0/2: Stop program	
	1: Start program	
	3: Continue program	
200008	Error register 1 (identical with 210004)	
	Bit 0:	Error on flash disk
	Bit 2:	Error on JX2 system bus
	Bit 3:	Error on Ethernet system bus
	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 1 (bit-coded) Bit 3: Error in ModConfig.da Bit 10: A bus node (publish/subscribe client) has reported an error	
200168	Bootloader version (IP format)	

200169	OS version (IP format)	
200170	Controller type (54861)	
200300	Currently available heap	
200301	Available heap at system launch	
200302	Available heap before application program	
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 has been licensed
	Bit 5 = 1:	Modbus/TCP available
	Bit 6 = :	Reserved
	Bit 7 = 1:	FTP client available
202936	Control register of the file system	
	0xc4697a4b	Formatting the flash disk
202960	Password for system command register (0x424f6f74)	
202961	System command register	
	102:	Controller restart (reboot)
	104:	Reset non-volatile parameters
	122:	Wait for communication - OFF
	123:	Wait for communication - ON
	160:	Task switch on I/O access - OFF
	161:	Task switch on I/O access - ON
	170:	Continue task time slice - OFF
	171:	Continue task time slice - ON
	310:	Load the configuration data
	311:	Load the module configuration
	312:	Load process data configuration for Ethernet system bus
202962	System status register	
	Bit 0 = 1:	Task switch on I/O access
	Bit 1 = 1:	Without waiting for communication
	Bit 3 = 1:	Continue task time slice - ON
203000	Interface monitoring: JetIP	
203005	Interface monitoring: STX 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 ... 209744	Enabling system components	

Application program

210000	Application program is running (bit 0 = 1)	
	0/2: Stop program	
	1: Start program	
	2: Continue program	
210001	JetVM version	
210004	Error register (bit-coded)	
	Bit 1:	Error on JX3 system bus
	Bit 2:	Error on JX2 system bus
	Bit 3:	Error on Ethernet system bus
	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 24:	Timeout - Cycle time
	Bit 25:	Timeout - Task lock
	Bit 31:	Unknown error
210006	Highest task number	
210007	Minimum program cycle time	
210008	Maximum program cycle time	
210009	Current program cycle time	
210011	Current task number	
210050	Current program position within an execution unit	
210051	ID of the execution unit being processed	
210056	Desired total cycle time in µs	
210057	Calculated total cycle time in µs	
210058	Maximum time slice per task in µs	
210060	Task ID (for R210061)	
210061	Priority for task [R210060]	
210063	Length of scheduler table	
210064	Index in scheduler table	
210065	Task ID in scheduler table	
210070	Task ID (for R210071)	
210071	Timer number (0 ... 31)	
210072	Manual triggering of a timer event (bit-coded)	
210073	End of cyclic task (task ID)	
210074	Command for cyclic tasks	
210075	Number of timers	
210076	Timer number (for R210077)	
210077	Timer value in milliseconds	
210100 ... 210199	Task state 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
	-1: Monitoring disabled
210610	Timeout (bit-coded,
	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
	Other 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
	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
1nnn980000 ... 1nnn980199	Indirect access via local register 236xxx
1nnn990000 ... 1nnn999999	Indirect access with variable target window

Ethernet system bus

	Subscriber
250000	Status (bit-coded) Bit 0 = 1: No CRC
	Bit 1 = 1: Error in connection with a subscription
	Bit 7 = 1: Subscriber is running
250001	Command 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	Amount 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) Bit 0 = 1: No CRC
	Bit 1 = 1: Error in connection with a publication
	Bit 7 = 1: Subscriber is running
255001	Command 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 the SMTP server
292933	IP address of the POP3 server
292934	Port number of the 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 Bit 0 = 1: Recording Bit 1 = 1: Stop if buffer is full Bit 2 = 1: Stop on error code Bit 3 = 1: Remanent memory
380001	Command 1: Clear error log 2: Start error log 3: Stop error log 4: Stop if error buffer is full 5: Circular buffer 6: Stop on error code ON 7: Stop on error code OFF 10: Remanent memory 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
380029	Group index to error list
380030 ... 380093	64 error entries

Special flags for networks

2075	Error in networking via JetIP
2080	Ethernet system bus error in R 200008
2081	Ethernet system bus error

Special flags - Interface monitoring

2088	OS flag - JetIP
2089	User flag - JetIP
2098	OS flag - Debug server
2099	User flag - Debug server

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

System function

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

4	Conversion from BCD to HEX
5	Conversion from 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

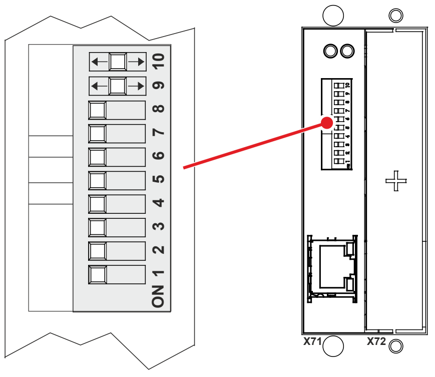
91	Appending a data file
92	Reading a data file
96	Deleting a data file
110	Sending an e-mail
150	Configuring NetCopyList
151	Deleting NetCopyList
152	Sending NetCopyList

JetSym STX functions

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

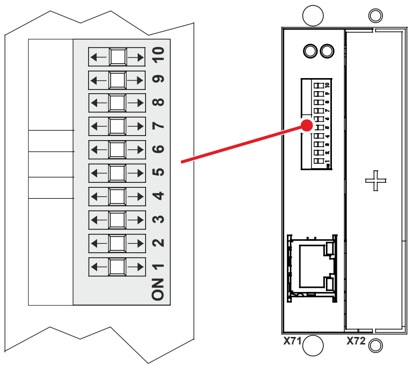
Setting the default IP address

To set the module to its default IP address 192.168.10.15, move the DIP switch sliders to the positions shown below:



Setting the IP address via config.ini file and DIP switch

The following DIP switch settings cause the JC-310-JM to read out the IP address from the file **config.ini** and the DIP switches:



Function of DIP switches 9 and 10

DIP switch sliders 9 and 10 let you configure how operating system and application program are launched.
The table below shows the possible functions and the corresponding positions of DIP switches 9 and 10:

Position of DIP switch 9	Position of DIP switch 10	Function
OFF	ON	LOAD
ON	ON	STOP
ON	OFF	RUN

Appendix

Introduction

This appendix contains electrical and mechanical data, as well as operating data.

Contents

A: Technical data

Introduction

This chapter contains information on electrical and mechanical data, as well as on operating data of the JC-310-JM.

Contents

Technical specifications

Electrical data - Power supply

The JC-310-JM is supplied with power by the servo amplifiers of the JM-200 series.

Technical data - Ethernet port

Parameter	Description
Type of connector	RJ45 Ethernet jack
Number of ports	1 port
Bit rate	10 MBit/s, 100 MBit/s (Cat 5e)
Auto-crossover	Yes

Memory configurations

Parameter	Description
Only JC-310-JM: Number of volatile registers	30.000
Only JC-310-JM: Volatile memory for variables	2 MB
Flash disk	4 MB

Notes on operating parameters

Notes on operating parameters	The JC-310-JM is installed in servo amplifiers of the JM-200 series. Therefore, the operating parameters listed in the appendix to the User Manual on the JM-200 series also apply to the JC-310-JM.

B: Index

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.