JC-340
Version Update
from V 1.04 to V 1.05

Revision 1.01 September, 2009 / Printed in Germany

Revision 1.01

Jetter AG reserves the right to make alterations to its products in the interest of technical progress. These alterations need not be documented in every single case.

This Version Update and the information contained herein have been compiled with due diligence. However, Jetter AG assume no liability for printing or other errors or damages arising from such errors.

The brand names and product names used in this document are trademarks or registered trademarks of the respective title owner.

Table of Contents

1 Introduction

Introduction

This chapter shows the history of OS versions for the controller JC-340.

Operating System Update - Why?

An OS update allows you to:

- add new functions to your controller
- fix software bugs
- make sure your controller is working with a definite OS version, for example, if a definite OS version has been released for a certain customer

Contents

Operating System Update

OS File for Updating the Operating System

For updating the OS the following file is needed:

OS File	Description
JC-340_1.05.0.0.os	OS file for JC-340 with version 1.05

Downloading the OS File

Jetter AG make operating system files available for download from their **homepage at** http://www.jetter.de . OS files can be found in the support area or on the page of the JC-340 controller via quicklink.

Operating System Update by means of JetSym

To update your OS proceed as follows:

Step	Action
1	Download the OS file from www.jetter.de
2	Establish a connection between PC and controller
3	In JetSym: Select menu item "Build -> Update OS" or Click on the button "OS Update" in the CPU window of the hardware manager
4	Select the OS File
5	Initiate the OS update by clicking OK
6	**Result:** Following Power OFF / Power ON the new OS is launched.

Minimum Requirements

For programming a JC-340 with version 1.05 JetSym 4.1 or higher is required.

JC-340 Version Update - Overview

V 1.04

The following table gives an overview of newly added features and fixed software bugs in OS version 1.04:

Description	New	Bug
JX2 System Bus:		
Register overlaying for digital inputs/outputs	✓	
Support of JX-SIO modules and third-party CANopen devices	✓	
JX3 system bus:		
Register overlaying for digital inputs/outputs	✓	
System bus special registers for status and control	✓	
Operating System Update:		
Via FTP: On completion notification the OS has actually been stored.		✓
Updating a JX2 slave module while registers are being accessed blocks communication		✓
Application program:		
Task switch could fail to happen		✓
Error signal in case of invalid file "/app/start.ini"		✓
Display commands:		
Redirection to JX2-SER1 works only if JX2-PRN1 has been configured		✓

V 1.05

The following table gives an overview of newly added features and fixed software bugs in OS version 1.05:

Description	New	Bug
JX2 System Bus: V1.05.0.00		
AS interface gateway BWU1821 is supported	✓	
Frequency inverter 8200 vector is supported	✓	
JetMove 1xx is not detected during boot process		✓
Automatic baud rate recognition does not work reliably for some of the baud rates and configurations of IP67 modules.		✓
Repetition counter does not work when polling I/O modules		✓
AutoCopy function:		
Automatic copying of controller data	✓	
Application program:		
Pending cyclic tasks are started immediately after Taskunlock	✓	
For function pow(x,y) a floating point number can be entered as exponent	✓	

Cyclic tasks can be debugged	✓	
Length of project and program names > 39 characters		✓
Restart of an elapsed timer		✓
The value returned by DateTimeDecode() was always 1 day short of the actual day.		✓
DateTimeEncode and -IsValid might return the value TRUE irrespective of an invalid date		✓
User registers:		
The register type can be set up without having to start the application program	✓	
Displays and HMIs:		
A floating point value can be used as default for UserInput	✓	
The default value for UserInput is not displayed correctly		✓
It is not possible to enter LED register numbers		✓

2 New Features

Introduction

This chapter describes the features which have been added or enhanced in the new software release.

Contents

Product Description - Module BWU1821 by Bihl+Wiedemann

BWU1821

The module BWU1821 is a gateway between CANopen® and AS interface.

Designation	Description
BWU1821	AS interface CANopen® gateway

Technical Data

Number of BWU1821 modules connected to the JX2 system bus of JC-3xx.	1 max.
Digital inputs and outputs	248 max.
Analog I/Os	124 max.

Restrictions

When connecting a BWU1821 module to the JX2 system bus of the controller JC-340 the following restrictions have to be observed:

- For the BWU1821 module only I/O module number 70 or 71 is allowed.
- The BWU1821 module occupies the next 8 module numbers.

Minimum Requirements

The BWU1821 module can be connected to the JX2 system bus of the following controllers and modules by Jetter AG:

Controller / Module	Starting from software release
JC-340 /JC-350	V 1.05.0.00
JC-360	V 1.01.0.00
JC-24x	V 3.23
JM-D203-JC24x	V 1.12.0.00
JX6-SB(-I)	V 2.18

Product Description - Lenze 8200 vector

Lenze 8200 vector

The frequency inverter 8200 vector is a product by Lenze GmbH und Co. KG. The frequency inverter can directly be connected to the JX2 system bus via communications module or function module.

Designation	Description
2175 CANopen®/DeviceNet	Communications module for CANopen®
CANopen PT	Function module for CANopen®

Technical Data

Number of 8200 vector modules connected to JX2 system bus	10 max.

Minimum Requirements

The frequency converter 8200 vector can directly be connected to the JX2 system bus of the following controllers and modules by Jetter AG:

Controller / Module	Starting from software release
JC-340 /JC-350	V 1.05.0.00
JC-360	V 1.01.0.00
JC-24x	V 3.10
JM-D203-JC24x	V 1.10.0.00
JX6-SB(-I)	V 2.10

Cyclic tasks are started immediately after Taskunlock

Introduction

Basically, the following rules apply when processing tasks of an application program:

- Following the instruction `Tasklock` no task switch is carried out until the instruction `Taskunlock` appears.
- When the cycle time of a cyclic task has elapsed, the currently processed task is aborted and processing of the cyclic task starts. However, this happens only if the task being processed does not block the task switch by means of `Tasklock`.

Obsolete Function

When the cycle time of a cyclic task has elapsed while the current task has blocked task switching, the cyclic task will only be started on the next "normal" task switch.

New Function

When the cycle time of a cyclic task has elapsed while the current task has blocked task switching, the cyclic task will be started immediately once the current task has executed the instruction `Taskunlock`.

Reason for this change

The new function minimizes jitter of cyclic tasks.

Function pow(x,y) with floating-point exponent

Obsolete Function	For arithmetic function pow(x,y) only integers are allowed as exponent. If a floating point number is entered, it is converted into an integer before it is used.
New Function	For arithmetic function pow(x,y) integers and floating point numbers are allowed as exponent (y).

Cyclic tasks can be debugged

Obsolete Function	If in a cyclic task a breakpoint is set by JetSym debugger, the task stops at the breakpoint. This causes a timeout of the cyclic task which prevents it from being executed further.
New Function	If a cyclic task stops at a breakpoint, timeout monitoring for this task is disabled. If the task is re-started by debugger, processing of the task is continued during its next cycle and timeout monitoring is re-activated.
Restrictions	The following restrictions apply to debugging cyclic tasks: ▪ The set cycle time will not be complied with. ▪ Step-by-step execution of a task going beyond its end (instruction `TaskExit`) is not possible.

Setting up the register type without having to start the application program

Introduction	The default setting for remanent user register types is integer. Floating point number can be assigned by defining the register variable within the application program.
Obsolete Function	If the controller is booted with mode selector S11 in STOP position, type definitions for user-registers are not loaded. Thus, the type for all user-registers remains integer.
New Function	If the controller is booted with mode selector S11 in STOP position, type definitions for user-registers are loaded from the application program. Thus, the type of all user-registers corresponds to the settings in the application program.
Reason for this change	This function ensures that user-registers can be saved or restored with their correct type. When doing so, the application program is not required to run.

A floating point value can be used as default for UserInput()

Obsolete Function	The register specifying the default value for `UserInput()` on an HMI accepts only integers. It is not possible to use floating-point numbers as default value.
New Function	The register type specifying the default value for `UserInput()` on an HMI depends on the type of the value entered into this register last. This way, both integer and floating-point numbers are possible as default values.

3 Fixed Software Bugs

Introduction

This chapter describes the software bugs which have been fixed in the new operating system release.

Contents

Length of project and program name

Effects of this Bug

If a project and/or program name is longer than 39 characters the program will not be executed on the controller.

Affected Versions/Revisions

The following versions/revisions are affected by this bug:

OS version	JC-340/350 JC-360 JC-24x	< 1.05.0.00 < 1.01.0.00 < 10.05.0.00
Hardware revision	not applicable	
Configuration or operating mode	not applicable	

Remedy / Workaround

Do not use project or program names with more than 39 characters.

Bug Fix

Starting from the following versions/revisions this bug has been fixed:

OS version	JC-340/350 JC-360 JC-24x	1.05.0.00 1.01.0.00 10.05.0.00
Hardware revision	not applicable	
Configuration or operating mode	not applicable	

Thus, the allowed length of a project or program name is only limited by the file system of the controller.

Restart of an elapsed timer

Effects of this Bug

A timer is started using the function `TimerStart()`. Once the timer has elapsed, function `TimerEnd()` returns the value `TRUE` and function `TimerValueActual()` the value `0`. If the controller is not switched off for several days, function `TimerEnd()` returns the value `FALSE` and function `TimerValueActual()` the value `0`, even though the timer has not been restarted.

Affected Versions/Revisions

The following versions/revisions are affected by this bug:

OS version	JC-340/350	< 1.05.0.00
	JC-360	< 1.01.0.00
	JC-24x	< 10.05.0.00
Hardware revision	not applicable	
Configuration or operating mode	not applicable	

Remedy / Workaround

Restart the timer after 10 days at the latest.

Bug Fix

Starting from the following versions/revisions this bug has been fixed:

OS version	JC-340/350	1.05.0.00
	JC-360	1.01.0.00
	JC-24x	10.05.0.00
Hardware revision	not applicable	
Configuration or operating mode	not applicable	

The function "DateTimeDecode()" returned a value that was always 1 day short of the actual day.

Effects of this Bug

When decoding the contents of a `dtDateTime` variable into individual date and time components using the function `DateTimeDecode()` the return value is 1 day short of the actual day. This bug occurs only in a leap year.

Affected Versions/Revisions

The following versions/revisions are affected by this bug:

OS version	JC-340/350	< 1.05.0.00
	JC-360	< 1.01.0.00
	JC-24x	< 10.05.0.00
Hardware revision	not applicable	
Configuration or operating mode	not applicable	

Remedy / Workaround

If the actual time is needed only, read it directly from the registers of the realtime clock.

Bug Fix

Starting from the following versions/revisions this bug has been fixed:

OS version	JC-340/350	1.05.0.00
	JC-360	1.01.0.00
	JC-24x	10.05.0.00
Hardware revision	not applicable	
Configuration or operating mode	not applicable	

Incorrect return value of the functions DateTimeIsValid() and DateTimeEncode()

Effects of this Bug

The valid range of the function DateTimeXXXXXXX() ranges from 1/1/1901 0:0:0 a.m. to 12/31/2099 11:59:59 p.m. For the following dates the functions DateTimeIsValid() and DateTimeEncode() return the value TRUE, although the values are invalid:

- Day = 0
- Month = 0
- Year <100

Affected Versions/Revisions

The following versions/revisions are affected by this bug:

OS version	JC-340/350	< 1.05.0.00
	JC-360	< 1.01.0.00
	JC-24x	< 10.05.0.00
Hardware revision	not applicable	
Configuration or operating mode	not applicable	

Remedy / Workaround

Check your application program containing day, month and year specifications for the a.m. values.

Bug Fix

Starting from the following versions/revisions this bug has been fixed:

OS version	JC-340/350	1.05.0.00
	JC-360	1.01.0.00
	JC-24x	10.05.0.00
Hardware revision	not applicable	
Configuration or operating mode	not applicable	

Incorrect default value for UserInput() is displayed

Effects of this Bug

The default value for `UserInput()` on an HMI is displayed using the format settings for the `DisplayValue()` function. If the default value is to be displayed with decimal positions, it is displayed incorrectly.

Affected Versions/Revisions

The following versions/revisions are affected by this bug:

OS version	JC-340/350	< 1.05.0.00
	JC-360	< 1.01.0.00
	JC-24x	< 10.05.0.00
Hardware revision	not applicable	
Configuration or operating mode	not applicable	

Remedy / Workaround

There is no remedy/workaround for affected versions/revisions.

Bug Fix

Starting from the following versions/revisions this bug has been fixed:

OS version	JC-340/350	1.05.0.00
	JC-360	1.01.0.00
	JC-24x	10.05.0.00
Hardware revision	not applicable	
Configuration or operating mode	not applicable	

JetMove 1xx is not detected during boot process

Effects of this Bug

When the boot process is completed, a JetMove 1xx connected to the JX2 system bus is not entered into the module array. Each register access to this JetMove 1xx results in a timeout.

Affected Versions/Revisions

The following versions/revisions are affected by this bug:

OS version	JC-340/350 JC-360	< 1.05.0.00 < 1.01.0.00
Hardware revision	not applicable	
Configuration or operating mode	not applicable	

Remedy / Workaround

There is no remedy/workaround for affected versions/revisions.

Bug Fix

Starting from the following versions/revisions this bug has been fixed:

OS version	JC-340/350 JC-360	1.05.0.00 1.01.0.00
Hardware revision	not applicable	
Configuration or operating mode	not applicable	

Repetition counter does not work when polling I/O modules

Effects of this Bug

If a timeout occurs when polling I/O modules on the JX2 system bus, the value in register 200.002.762 is not incremented.

Affected Versions/Revisions

The following versions/revisions are affected by this bug:

OS version	JC-340/350 JC-360	< 1.05.0.00 < 1.01.0.00
Hardware revision	not applicable	
Configuration or operating mode	not applicable	

Remedy / Workaround

There is no remedy/workaround for affected versions/revisions.

Bug Fix

Starting from the following versions/revisions this bug has been fixed:

OS version	JC-340/350 JC-360	1.05.0.00 1.01.0.00
Hardware revision	not applicable	
Configuration or operating mode	not applicable	

Automatic baud rate detection on JX2 system bus

Effects of this Bug

When the boot process is completed, not all modules connected to the JX2 system bus are entered into the module array if modules with automatic baud rate detection are connected.

Modules with automatic baud rate detection

The following modules feature automatic baud rate detection:

- IP67-I/O module LioN-S
- IP67-I/O module LJX7-CSL

Affected Versions/Revisions

The following versions/revisions are affected by this bug:

OS version	JC-340/350 JC-360	< 1.05.0.00 < 1.01.0.00
Hardware revision	not applicable	
Configuration or operating mode	not applicable	

Remedy / Workaround

There is no remedy/workaround for affected versions/revisions.

Bug Fix

Starting from the following versions/revisions this bug has been fixed:

OS version	JC-340/350 JC-360	1.05.0.00 1.01.0.00
Hardware revision	not applicable	
Configuration or operating mode	not applicable	

LED registers for HMIs

Effects of this Bug

In multi-display mode of HMIs the states of LEDs located in the keys are acquired from register bits of the controller. Registers, into which the number of key registers is entered, are able to hold only values between 0 and 65,535. Therefore, the register range of JC-340 can not be reached.

Affected registers	222833 ... 222836

Affected Versions/Revisions

The following versions/revisions of the JC-340 are affected by this bug:

OS version	JC-340/350 JC-360	< 1.05.0.00 < 1.01.0.00
Hardware revision	not applicable	
Configuration or operating mode	Multi-Display Mode	

Remedy / Workaround

There is no remedy for affected versions/revisions.

Bug Fix

Starting from the following versions/revisions of the JC-340 this bug has been fixed:

OS version	JC-340/350 JC-360	1.05.0.00 1.01.0.00
Hardware revision	not applicable	
Configuration or operating mode	Multi-Display Mode	

4 Automatic Copying of Controller Data

Introduction

This chapter describes the AutoCopy function which allows to copy data between controller and SD card. To this end, a command file has to be created which is then stored to the SD card along with the data. This command file is automatically processed by the controller during the boot process.

Functions

The following functions can be performed:

- Storing registers and flags to a file
- Restoring registers and flags from a file
- Creating directories
- Deleting directories
- Copying files
- Deleting files

Areas of Application

This function can be used in systems where remote maintenance is not feasible, no PC is available or the operator is not able (or should not be allowed) to make modifications to the plant. This function includes the following:

- Modification to the application program
- Modification to user data
- Modification to the controller configuration
- Operating system update (controller and/or modules on the system bus)
- Duplication of a control system

Prerequisites

The following requirements must be met:

- the programmer is familiar with the file system of the controller JC-340
- the JC-340 features the SD card option

Designation

In this description *Complete Name* means the name of the file or directory including the complete path.

Contents

4.1 Operating Principle

Introduction

This chapter describes how the AutoCopy funcion is started and how it is executed by the controller.

Contents

Starting the AutoCopy Function

Introduction

The AutoCopy function can only be executed when the controller is booting.

Prerequisites

The command file has been created and stored to the SD card.

	Value	Comment
File Name	autocopy.ini	All lower case letters
Directory	/SD/	Root directory on the SD Card

Starting the AutoCopy Function

To start the AutoCopy function proceed as follows:

Step	Action
1	Switch the controller off.
2	Insert the SD card completely into the SD slot.
3	Set the mode selector to "LOAD" position
4	Switch the controller on.

Result: The controller is booting in AutoCopy mode.

Executing the AutoCopy Function

Introduction

During the boot process in AutoCopy mode the controller executes the commands contained in the command file.

Restrictions

In AutoCopy mode the following restrictions of controller functions apply:

- The application program is not executed
- No communication with the controller possible

Executing the AutoCopy Function

The OS of the controller processes the AutoCopy function in the following steps:

Step	Description
1	The controller loads the file "/SD/autocopy.ini" from the SD card.
2	The controller reads the values from section [OPTIONS]
3	The PLC reads the command and its parameters from the section [COMMAND_1], processes it and writes the results, if any, into the log file
4 .. n	The controller processes the other commands in ascending order up to the number given in section [OPTIONS]
n+1	The PLC calculates the statistic values for all command results and writes them into the log file.

LEDs of the Controller in AutoCopy Mode

During boot process the OS status LEDs indicate the following:

Level	Description					
	R	**E**	**D1**	**D2**	**SD**	**State**
1	🟢 4Hz	🔴 4Hz	🔴 4Hz	🔴 4Hz	○ OFF	Reset
2	🟢 1Hz	○ OFF	○ OFF	🔴 ON	○ OFF	The bootloader is running and is checking the OS
3	🟢 1Hz	○ OFF	○ OFF	○ OFF	○ OFF	The OS reads the backplane DIP switch settings and checks if an Ethernet switch exists
4	🟢 1Hz	🔴 ON	○ OFF	○ OFF	○ OFF	The OS initializes the realtime clock and file system

Level	Description					
5	**R**	**E**	**D1**	**D2**	**SD**	**State**
	🟢 1Hz	🔴 ON	🔴 ON	○ OFF	🟡	The OS initializes the modules on the JX3 and JX2 system bus and the SD card
6	**R**	**E**	**D1**	**D2**	**SD**	**State**
	🟢 1Hz	○ OFF	🔴 ON	○ OFF	🟡	The command file of the AutoCopy function is being processed
7a	**R**	**E**	**D1**	**D2**	**SD**	**State**
	🟢 1Hz	○ OFF	🔴 1Hz	○ OFF	○ OFF	AutoCopy function is completed; no errors occurred
7b	**R**	**E**	**D1**	**D2**	**SD**	**Zustand**
	🟢 1Hz	🔴 ON	🔴 1Hz	○ OFF	○ OFF	AutoCopy function is completed; errors occurred

Terminating AutoCopy Mode

Introduction

The AutoCopy mode can only be exited by booting the controller.

Terminating AutoCopy Mode

Once the AutoCopy function is completed, proceed as follows to exit the AutoCopy mode:

Step	Action
1	Switch the controller off.
2	The SD card can now be removed (not required)
3	Set the mode selector to "RUN" or "STOP" position
4	Switch the controller on.

Result: The controller is rebooting.

4.2 Creating the File "autocopy.ini"

Introduction

This chapter covers the structure of the file "/SD/autocopy.ini" and the available commands.

File Structure

This command file of the AutoCopy function is a text file the entries of which are grouped into several sections.

- In these sections values can be set which are then used by the AutoCopy function.
- Blank lines can be inserted as required
- The following characters precede a comment line: "!", "#" or ";"

Sections

The command file has two section types:

- In section [OPTIONS] the basic settings are made. It exists only once.
- In the sections [COMMAND_#] the commands to be executed are specified. The number of commands is limited to 128.

Contents

Section [OPTIONS]

Introduction

This section contains the basic settings of the AutoCopy function. It exists only once, preferably at the beginning of the file.

Example

```
[OPTIONS]
CommandCount = 14
LogFile      = /SD/autocopy.log
LogAppend    = 1
```

Elements of this Section

This section consists of the following elements:

CommandCount	
In the given example	14
Description	Number of command sections that follow
Allowed values	> = 0
Illegal values	< 0
In case of illegal value or missing entry	0

LogFile	
In the given example	/SD/autocopy.log
Description	Complete name of the log file
Allowed values	■ All allowed file names ■ Directory exists
Illegal values	■ Invalid file name ■ Nonexistent directory
In case of illegal value or missing entry	No log file will be created.

LogAppend	
In the given example	1
Description	Defines whether a new log file is to be created or it is to be appended to an existing one.
Allowed values	■ 0 = Delete file which may exist and create a new one. ■ 1 = Append file to existing one. If no file exists, a new log file is created.
Illegal values	■ < 0 ■ > 1
In case of illegal value or missing entry	A new log file will be created.

Command Sections

Introduction

In these sections commands can be specified which are then executed by the AutoCopy function of the controller.

Example

```
[COMMAND_1]
Command = DirCreate
Path    = /Homepage

[COMMAND_2]
Command     = FileCopy
Source      = /SD/Index.htm
Destination = /Homepage/index.htm
```

Section Names

The section names consist of the string `COMMAND_` followed by a number which indicates the number of the entry `CommandCount` given in section `[OPTIONS]`.

Processing Commands

The AutoCopy function processes the commands in order of their section names.

- Starting with the command under section `[COMMAND_1]`
- Ending with the command under the section with the value of entry `CommandCount` from section `[OPTIONS]`
- Each command section may hold only one command. That is, for each command a separate section has to be created.

Available Commands

The following commands are available:

Command = DirCreate	
Function	This command is for creating a subdirectory
Parameter name	Path
Parameter value	Complete name of the directory
Allowed values	■ All valid directory names ■ Existing higher-level directories
Illegal values	■ Invalid directory names ■ Nonexistent higher-level directory ■ Name of an already existing directory
In case of an illegal value	The directory will not be created and the error message will be entered into the log file
Example	

```
[COMMAND_1]
Command = DirCreate
Path    = /sub1

[COMMAND_2]
Command = DirCreate
Path    = /sub1/sub2
```

Command = DirRemove

Function	This command is for deleting a subdirectory
Parameter name	Path
Parameter value	Complete name of the directory
Allowed values	■ All valid directory names ■ An empty directory
Illegal values	■ Invalid directory names ■ Directory is not empty
In case of an illegal value	The directory will not be deleted and the error message will be entered into the log file

Example

```
[COMMAND_8]
Command = DirRemove
Path    = /sub1/sub2
```

Command = FileCopy

Function	This command is for copying a file
Parameter name 1	Source
Parameter value 1	Complete name of the source file
Parameter name 2	Destination
Parameter value 2	Complete name of the destination file
Allowed values	■ All allowed file names ■ The destination directory does exist
Illegal values	■ Invalid file name ■ Nonexistent source file ■ Nonexistent destination directory
In case of an illegal value	The file will not be copied and the error message will be entered into the log file

Example

```
[COMMAND_1]
Command     = FileCopy
Source      = /SD/OS/JC-340_1.04.0.03.os
Destination = /System/OS/op_system.os

[COMMAND_2]
Command     = FileCopy
Source      = /SD/Manual.pdf
Destination = /sub1/Manual.pdf
```

Command = FileRemove

Function	This command is for deleting a file.
Parameter name	Path
Parameter value	Complete name of the file
Allowed values	All allowed file names
Illegal values	Invalid file name
In case of an illegal value	The file will not be deleted and the error message will be entered into the log file

Example	`[COMMAND_5]`
	`Command = FileRemove`
	`Path    = /sub1/Manual.pdf`

Command = DaFileRead	
Function	This command is for transferring register values and flag states from a data file to the controller
Parameter name	DaFile
Parameter value	Complete name of the data file
Allowed values	All allowed file names for data files
Illegal values	■ Invalid file name ■ Nonexistent data file
In case of an illegal value	The date will not be transferred to the controller and the error message will be entered into the log file

Example	`[COMMAND_12]`
	`Command = DaFileRead`
	`DaFile      = /SD/Data/MyTestData.da`

Command = DaFileWrite	
Function	This command is for storing register values and flag states to a data file
Parameter name 1	DaFile
Parameter value 1	Complete name of the file
Allowed values	■ All allowed file names for data files ■ The destination directory does exist
Illegal values	■ Invalid file name ■ Nonexistent destination directory
In case of an illegal value	The file will not be created and the error message will be entered into the log file
Parameter name 2	Append
Parameter value 2	Defines whether a new data file is to be created or it is to be appended to an existing one.
Allowed values	■ 0 = Delete file which may exist and create a new one. ■ 1 = Append file to existing one. If no file exists, create a new data file.
Illegal values	■ < 0 ■ > 1
In case of an illegal value	A new data file will be created
Parameter name 3	Type
Parameter value 3	Defines whether registers or flags are to be stored.
Allowed values	■ Register(s) ■ Flag
Illegal values	Values other than "Register" or "Flag"
In case of an illegal value	The file will not be created and the error message will be entered into the log file

Parameter name 4	First
Parameter value 4	Number of the first register or flag
Allowed values	All valid numbers from the memory area of the corresponding controller
Illegal values	Invalid numbers
In case of an illegal value	The file will not be created and the error message will be entered into the log file
Parameter name 5	Last
Parameter value 5	Number of the last register or flag
Allowed values	All valid numbers from the memory area of the corresponding controller which are equal to or greater than the value for "First".
Illegal values	■ Invalid numbers ■ Numbers less than "First"
In case of an illegal value	Only one value (First) is stored

Example

```
[COMMAND_11]
Command = DaFileWrite
DaFile      = /SD/MyTestData2.da
Append      = 0
Type        = Register
First       = 1000000
Last        = 1000000

[COMMAND_12]
Command = DaFileWrite
DaFile      = /SD/MyTestData2.da
Append      = 1
Type        = Flag
First       = 10
Last        = 20

[COMMAND_13]
Command = DaFileWrite
DaFile      = /SD/MyTestData2.da
Append      = 1
Type        = Register
First       = 1000001
Last        = 1000999
```

Example of a Command File

Task

New functions are to be added to an existing plant which is controlled by a JetControl 340 equipped with several JX3 modules. To this end, the following modifications have to be made to the configuration:

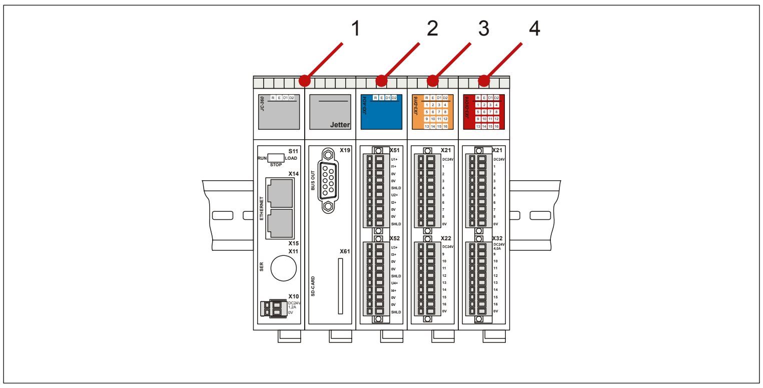
- Operating system update for the controller
- Operating system update for an analog output module
- New application program
- New values for some of the registers

Solution

The required files are copied to an SD card and a command file for the AutoCopy function is created. This SD card along with a short instruction sheet is sent to the plant operator. Once the update is completed, the operator returns the card.

Sample Configuration

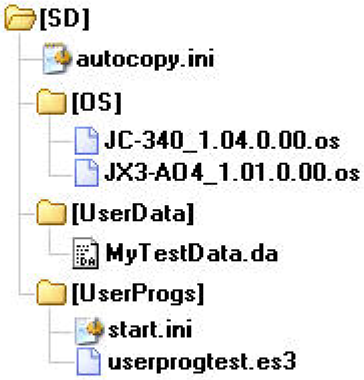
This example is based on the following configuration:



Number	Module	Description
1	JC-340	Controller
2	JX3-AO4	Analog output module I/O module number 2
3	JX3-DI16	Digital input module
4	JX3-DIO16	Digital output module

SD Card Contents

The following illustration shows the directory structure and the files on the SD card before the AutoCopy function is executed:



```
[SD]
  autocopy.ini
  [OS]
    JC-340_1.04.0.00.os
    JX3-AO4_1.01.0.00.os
  [UserData]
    MyTestData.da
  [UserProgs]
    start.ini
    userprogtest.es3
```

Following execution the log file "autocopy.log" has been added.

Command File

```
[OPTIONS]
CommandCount = 7
LogFile       = /SD/autocopy.log
LogAppend     = 0

# update operating system of controller
[COMMAND_1]
Command     = FileCopy
Source      = /SD/OS/JC-340_1.04.0.00.os
Destination = /System/OS/op_system.os

# update operating system of JX3-AO4 module
[COMMAND_2]
Command     = FileCopy
Source      = /SD/OS/JX3-AO4_1.01.0.00.os
Destination = /System/JX3-Module02/OS/system.os

# create user program directories
# probably already present - but to be sure ...
[COMMAND_3]
Command     = DirCreate
Path        = /app

[COMMAND_4]
Command     = DirCreate
Path        = /app/userprogtest

# copy user program start file
[COMMAND_5]
Command     = FileCopy
Source      = /SD/UserProgs/start.ini
```

```
Destination = /app/start.ini

# copy user program
[COMMAND_6]
Command     = FileCopy
Source      = /SD/UserProgs/userprogtest.es3
Destination = /app/userprogtest/userprogtest.es3

# set registers and flags
[COMMAND_7]
Command     = DaFileRead
DaFile      = /SD/UserData/MyTestData.da
```

4.3 Log File

Introduction

This chapter covers the structure and contents of the log file into which the results of each command is entered.

Contents

File Contents

Introduction

The log file is a text file. The command file defines whether a log file is to be created. And whether it is to be created from scratch or whether the entries are to be appended to an existing log file.

Example

```
JetControl AutoCopy log file 07.11.2008 09:14:09

 1: Ok    - FileCopy   /SD/OS/JC-340_1.04.0.00.os
                       /System/OS/op_system.os (345740 byte)
 2: Ok    - FileCopy   /SD/OS/JX3-AO4_1.01.0.00.os
                       /System/JX3-Module02/OS/system.os
                       (16832 byte)
 3: Error - DirCreate  /app
 4: Ok    - DirCreate  /app/userprogtest
 5: Ok    - FileCopy   /SD/UserProgs/start.ini
                       /app/start.ini (63 byte)
 6: Ok    - FileCopy   /SD/UserProgs/userprogtest.es3
                       /app/userprogtest/userprogtest.es3
                       (169 byte)
 7: Ok    - DaFileRead /SD/UserData/MyTestData.da
                       (entries: 6)

Command statistics:
  Total  : 7
  Ok     : 6
  Error  : 1
```

Description

When for each execution of the AutoCopy function a section is appended to an existing log file, the log file consists of three elements:

- The header contains date and time
- The following block contains information on the executed commands.
- Finally, short statistics on command processing is shown.

In the above example an error occurs when trying to create the directory "/app" as this directory already exists.

4.4 Data Files

Introduction

This chapter covers data files where register and flag values are stored.

Contents

File Format

Format

The file is structured as follows:

- Pure text file
- Each entry must be in a separate line of text
- Each line must be terminated by carriage return / line feed
- Comment lines must be preceded by ";"
- Each data file is to start with the entry "SD1001".

Data Lines

A data line consists of the following elements:

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

Variable ID	Variable type
FS	Flags
RS	Integer registers
QS	Floating-point registers

Example

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