JC-340

Version Update
from V. 1.08 to V. 1.09

Version Update

Jetter

Revision 1.01

January 2011 / Printed in Germany

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_1.09.0.0.os	OS file for JC-340 with version 1.09

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 1.09 JetSym 4.2 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:

Function	New	Fixed
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:

Function	New	Fixed
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	✓	

Function	New	Fixed
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		✓

V 1.08

The following table gives an overview of newly added features and fixed software bugs in OS version 1.08:

Function	New	Fixed
System Configuration:		
System rights for configuration file	✓	
JX2 System Bus: V1.11.0.00		
Timeout after CAN-PRIM message		✓
Registers of LJX7-CSL modules		✓
Write access to analog outputs of CANopen® modules		✓
State of digital inputs when the controller is powered on		✓
Digital outputs on JX-SIO or CANopen® modules		✓
Input/output 64 on JX-SIO or CANopen® modules		✓
User-programmable CAN Interface		✓
Application Program:		
NetCopyList functions	✓	
StrCopy()		✓
Crash in the case of "invalid" application program		✓
NetCopyVarFromReg()		✓
JX3 System Bus:		
Module registers for digital I/Os	✓	
Displays and HMIs:		
UserInput()		✓

V 1.09

The following table gives an overview of newly added features and fixed software bugs in OS version 1.09:

Function	New	Fixed
System:		
System command register	✓	
JX2 System Bus: V1.13.0.00		
Status change of inputs on JX2-ID8		✓
Status change of fast inputs		✓
Application Program:		
FTP client	✓	
Axis instructions		✓
Taskrestart in the case of Delay()		✓
Crash in the case of missing library		✓
Floating-point number registers in data files		✓
NetCopyVarToReg with floating-point number registers		✓
JX3 System Bus:		
Dummy modules	✓	
AutoCopy:		
FTP commands	✓	
Serial Interface:		
Initialization after booting		✓

2 New Features

Introduction

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

Contents

2.1 Various New Features and Modifications

Introduction

This chapter covers the new features and modifications

Contents

Dummy Modules on the JX3 System Bus

Dummy Modules

A dummy module is a module on the JX3 system bus that actually does not exist. When assigning I/O and slave module numbers, the controller JC-340 treats dummy modules as if they were existing modules.

Dummy modules allow the user to insert a nonexistent module between existing modules. So, dummy modules can be used as placeholders for future extensions or optional plant configurations.

Overview of Registers

The value contained in the register for dummy module configuration is remanent. Any changes become effective only after the controller JC-340 has been re-booted.

Register(s)	Description
R 100002023	I/O Dummy Module

Configuring Dummy Modules

Step	Action
1	Modify the dummy module configuration within the controller via R 100002023.
2	Switch the controller off.
3	Then, switch the controller on. **Result:** When initializing the JX3 system bus, the configured dummy modules were taken into account. The dummy modules can be read out via module array.

2.2 FTP Client

FTP Client

The FTP client integrated into the controller JC-340 allows a remote network device to access files and directories from within the application program. 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.

Prerequisites

- 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 PLC completes only one FTP access at a time.
- The corresponding task in the application program stops at the command until the access is completed.
- While 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.

Content

2.2.1 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	Establish connection(s) to FTP server(s)
3	Transfer data
4	Terminate the connection(s)

Restrictions

In the application program, tasks serving the FTP client should not be stopped through `TaskBreak` or restarted through `TaskRestart` while the controller is processing one of these functions. Otherwise it may happen that the controller fails to complete this function which will block new function calls by the FTP client.

Content

Initializing the FTP Client

Introduction

This function must be initialized each time the application program is launched.

Function Declaration

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

Result of the Function

This function will return the following value:

Result of the Function	
0	Always

Using this Function

This function can be used and its returned value be assigned to a variable for further utilization in the following way:

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

How it works

The controller processes this function in the following steps:

Step	Description
1	All connections established by the FTP client are terminated.
2	All OS-internal data structures of the FTP client are initialized.

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 using "UserName" and "Password".

Function Declaration

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

Function Parameters

Description of function parameters:

Parameter	Value	Comment
ServerAddr	IP address or name	Name resolution by DNS server
UserName	Login name	Login
Password	Login password	Login

Result of the Function

If the returned value is positive, the connection could have been established and login was successful. If the returned value is "0", an error occurred and the connection could not be established.

Result of the Function	
> 0	A positive result must be stored to a variable, since it has to be passed on as handle with functions for receiving and sending data via this connection, as well as for terminating this connection.
0	Errors when establishing a connection or logging in to the FTP server

Using this Function

This function can be used and its returned value be assigned to a variable for further utilization in the following way:

```
Task FtpClientTask Autorun
    Var
        FtpHandle: Handle;
    End_Var;

    ....
    FtpHandle := FtpConnect('192.168.123.45',
                            'Administrator',
                            'AdminPassword');
    ....
End_Task;
```

How it works

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:

<table>
<tr><th>Step</th><th>Description</th></tr>
<tr><td>1</td><td>The controller tries to establish a TCP/IP connection to the FTP server with IP address 192.168.123.45.</td></tr>
<tr><td>2</td><td>
<table>
<tr><th>If ...</th><th>... Then ...</th></tr>
<tr><td>the network client has accepted the connection</td><td>step 3 is carried out</td></tr>
<tr><td>the connection could not be established and the timeout has not elapsed, yet</td><td>step 1 is carried out</td></tr>
<tr><td>an error has occurred or the timeout has elapsed</td><td>the function is terminated and value "0" is returned</td></tr>
</table>
</td></tr>
<tr><td>3</td><td>The controller logs on to the FTP server with its user name "Administrator" and password "AdminPassword".</td></tr>
<tr><td>4</td><td>
<table>
<tr><th>If ...</th><th>... Then ...</th></tr>
<tr><td>the FTP server has accepted the connection</td><td>the function is terminated and a positive value is returned as handle for further access to this connection</td></tr>
<tr><td>the FTP server has not accepted the connection (e.g. invalid user name or wrong password)</td><td>the function is terminated and value "0" is returned</td></tr>
</table>
</td></tr>
</table>

Related Topics:

- **Terminating a Connection** on page 18

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 function parameters:

Parameter	Value	Comment
FtpConnection	Handle associated with the connection	Return value when establishing the connection

Result of the Function

This function will return one of the following values:

Result of the Function	
0	Connection terminated and deleted
-1	Invalid handle
-2, -3	Communication error (e.g. no response from FTP server)

Using this Function

This function can be invoked and its returned value be assigned to a variable for further utilization in the following way:

```
Task FtpClientTask Autorun
    Var
        FtpHandle: Handle;
        Result:    Int;
    End_Var;

    ....
    Result := FtpDisconnect(FtpHandle);
    ....
End_Task;
```

Related Topics:

- **Establishing a Connection to the FTP Server** on page 16

Reading a File from FTP Server

Introduction

This function is for reading a file from an FTP server and storing it to the local file system.

Function Declaration

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

Function Parameters

Description of function parameters:

Parameter	Value	Comment
FtpConnection	Handle associated with the connection	Return value when establishing the connection
ServerFile	File Name	Name of the file to be read from the FTP server.
ClientFile	File Name	Name as which the file read from the FTP server is to be saved

Result of the Function

If the returned value is negative, an error has occured. If the returned value is "0", the file could have been read from the FTP server and stored locally.

Result of the Function	
0	No error
-1	Invalid handle
-2 , -6	Error when storing the file locally
-3, -5, -7, -8	Communication error (e.g. no response from FTP server)
-4	Error message from FTP server (e.g. file does not exist)

Using this Function

This function can be used and its returned value be assigned to a variable for further utilization in the following way:

```
Task FtpClientTask Autorun
    Var
        FtpHandle: Handle;
        Result:    Int;
    End_Var;

    ....
    Result := FtpFileRead(FtpHandle,
                          'ServerTestFile.txt',
                          'LocalTestFile.txt');
    ....
End_Task;
```

How it works

The task waits at the program line until the file "ServerTestFile.txt" will be read from the FTP server and stored to the local file system as "LocalTestFile.txt" or an error will occur. This function is processed in the following steps:

Step	Description
1	The controller sends a command to the FTP server that file "ServerTestFile.txt" is to be read.
2	The controller receives the contents of this file and writes it into the file "LocalTestFile.txt".
3	<table><tr><th>If ...</th><th>... Then ...</th></tr><tr><td>no errors occurred</td><td>the file was successfully copied, the function is terminated and value "0" is returned</td></tr><tr><td>errors have occurred</td><td>the function is terminated and a negative value is returned</td></tr></table>

File Names

- The function parameter for the local file may contain the path to this file (e.g. `"/Data/TestFiles/LocalTestFile.txt"`).
- The function parameter for the file on the FTP server may contain the path to this file if this feature is supported by the file system. If this feature is not supported, the corresponding directory must be set using the command `FtpDirChange(...)`.
- The file system of a JetControl PLC supports both options.

Related Topics:

- **Writing a File to the FTP Server** on page 21

Writing a File to the FTP Server

Introduction

This function is for sending a file from the local file system to an FTP server.

Function Declaration

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

Function Parameters

Description of function parameters:

Parameter	Value	Comment
FtpConnection	Handle associated with the connection	Return value when establishing the connection
ServerFile	File Name	Name as which the file is to be saved to the FTP server
ClientFile	File Name	Name of the file to be read from the local file system.

Result of the Function

If the returned value is negative, an error has occured. If the returned Value is "0", the file could have been read and stored to the FTP server.

Result of the Function	
0	No error
-1	Invalid handle
-2	Error when reading the local file (e.g. file does not exist)
-3, -5, -8	Communication error (e.g. no response from FTP server)
-4, -7	Error message from FTP server (e.g. file cannot be created)

Using this Function

This function can be used and its returned value be assigned to a variable for further utilization in the following way:

```
Task FtpClientTask Autorun
    Var
        FtpHandle: Handle;
        Result:    Int;
    End_Var;

    ....
    Result := FtpFileWrite(FtpHandle,
                           'ServerTestFile.txt',
                           'LocalTestFile.txt');
    ....
End_Task;
```

How it works

The task waits at the program line until the file "LocalTestFile.txt" will be read from the local file system and stored to the FTP server as "ServerTestFile.txt" or an error will occur. This function is processed in the following steps:

Step	Description
1	The controller sends a command to the FTP server that file "ServerTestFile.txt" is to be stored.
2	The controller sends the contents of the file "LocalTestFile.txt".
3	<table><tr><th>If ...</th><th>... Then ...</th></tr><tr><td>no errors occurred</td><td>the file was successfully copied, the function is terminated and value "0" is returned</td></tr><tr><td>errors have occurred</td><td>the function is terminated and a negative value is returned</td></tr></table>

File Names

- The function parameter for the local file may contain the path to this file (e.g. `"/Data/TestFiles/LocalTestFile.txt"`).
- The function parameter for the file on the FTP server may contain the path to this file if this feature is supported by the file system. If this feature is not supported, the corresponding directory must be set using the command `FtpDirChange(...)`.
- The file system of a JetControl PLC supports both options.

Related Topics:

- **Reading a File from FTP Server** on page 19

Deleting a File from the FTP Server

Introduction

This function is for removing a file from the FTP server.

Function Declaration

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

Function Parameters

Description of function parameters:

Parameter	Value	Comment
FtpConnection	Handle associated with the connection	Return value when establishing the connection
ServerFile	File Name	Name of the file to be removed.

Result of the Function

If the returned value is negative, an error has occured. If the returned Value is "0", the file could not be removed from the FTP server.

Result of the Function	
0	No error
-1	Invalid handle
-2	Communication error (e.g. no response from FTP server)
-3	Error message from FTP server (e.g. file does not exist)

Using this Function

This function can be used and its returned value be assigned to a variable for further utilization in the following way:

```
Task FtpClientTask Autorun
    Var
        FtpHandle: Handle;
        Result:    Int;
    End_Var;

    ....
    Result := FtpFileRemove(FtpHandle,
                            'ServerTestFile.txt');
    ....
End_Task;
```

How it works

The task waits at the program line until the file "ServerTestFile.txt" will be deleted from the FTP server or an error will occur. This function is processed in the following steps:

Step	Description
1	The controller sends a command to the FTP server that file "ServerTestFile.txt" is to be deleted.
2	<table><tr><th>If ...</th><th>... Then ...</th></tr><tr><td>no errors occurred</td><td>the file was deleted, the function is terminated and value "0" is returned</td></tr><tr><td>errors have occurred</td><td>the function is terminated and a negative value is returned</td></tr></table>

File Names

- The function parameter for the local file may contain the path to this file (e.g. `"/Data/TestFiles/LocalTestFile.txt"`).
- The function parameter for the file on the FTP server may contain the path to this file if this feature is supported by the file system. If this feature is not supported, the corresponding directory must be set using the command `FtpDirChange(...)`.
- The file system of a JetControl PLC supports both options.

Changing a Directory on the FTP Server

Introduction

This function is for changing the current directory on the FTP server.

Function Declaration

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

Function Parameters

Description of function parameters:

Parameter	Value	Comment
FtpConnection	Handle associated with the connection	Return value when establishing the connection
ServerDir	Directory name	Name of the directory into which the user wants to change

Result of the Function

If the returned value is negative, an error has occured. If the returned Value is "0", the system failed to change into directory on the FTP server.

Result of the Function	
0	No error
-1	Invalid handle
-2	Communication error (e.g. no response from FTP server)
-3	Error message from FTP server (e.g. directory does not exist)

Using this Function

This function can be used and its result be assigned to a variable for further utilization in the following way:

```
Task FtpClientTask Autorun
    Var
        FtpHandle: Handle;
        Result:    Int;
    End_Var;

    ....
    Result := FtpDirChange(FtpHandle,
                           'DataFiles');
    ....
End_Task;
```

How it works

The task waits at the program line until change into subdirectory "DataFiles" on the FTP server will be completed or an error will occur. This function is processed in the following steps:

Step	Description
1	The controller sends a command to the FTP server to change into directory "DataFiles".
2	<table><tr><th>If ...</th><th>... Then ...</th></tr><tr><td>no errors occurred</td><td>the new directory is set, the function is terminated and value "0" is returned</td></tr><tr><td>errors have occurred</td><td>the function is terminated and a negative value is returned</td></tr></table>

Directory Names

- The function parameter for the file on the FTP server may contain the complete path to this directory - even across several directory levels - if this feature is supported by the file system.
- If this feature is not supported, the user must navigate from one directory to the next until the corresponding directory is reached. This is done using the command `FtpDirChange(...)`.
- The file system of a JetControl PLC supports both options.

Related Topics:

- **Determining the Current Directory on the FTP Server** on page 31

Creating a Directory on the FTP Server

Introduction

This function is for creating a new directory on the FTP server.

Function Declaration

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

Function Parameters

Description of function parameters:

Parameter	Value	Comment
FtpConnection	Handle associated with the connection	Return value when establishing the connection
ServerDir	Directory name	Name of the directory to be created

Result of the Function

If the result of this function is negative, an error has occured. If the returned value is "0", the system failed to create the directory on the FTP server.

Result of the Function	
0	No error
-1	Invalid handle
-2	Communication error (e.g. no response from FTP server)
-3	Error message from FTP server (e.g. directory already exists)

Using this Function

This function can be used and its result be assigned to a variable for further utilization in the following way:

```
Task FtpClientTask Autorun
    Var
        FtpHandle: Handle;
        Result:    Int;
    End_Var;

    ....
    Result := FtpDirCreate(FtpHandle,
                           'DataFiles');
    ....
End_Task;
```

How it works

The task waits at the program line until subdirectory "DataFiles" in the file system of the FTP server will be created or an error will occur. This function is processed in the following steps:

Step	Description
1	The controller sends a command to the FTP server that directory "DataFiles" is to be created.
2	<table><tr><th>If ...</th><th>... Then ...</th></tr><tr><td>no errors occurred</td><td>the new directory is created, the function is terminated and value "0" is returned</td></tr><tr><td>errors have occurred</td><td>the function is terminated and a negative value is returned</td></tr></table>

Directory Names

- The function parameter for the file on the FTP server may contain the complete path to this directory - even across several directory levels - if this feature is supported by the file system.
- If this feature is not supported, the user must navigate from one directory to the next until the corresponding directory is reached. This is done using the command `FtpDirChange(...)`.
- The file system of a JetControl PLC supports both options.

Restrictions Regarding the File System of a JetControl

If a directory with the corresponding path is specified as function parameter, all directories up to the directory to be created must exist. Recursive creation of several directories is not supported.

Beispiel:

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

To be able to create subdirectory "Release" subdirectory "TextFiles" in directory "DataFiles" must already exist.

Related Topics:

- **Removing a Directory on the FTP Server** on page 29

Removing a Directory on the FTP Server

Introduction

This function is for removing an existing directory on the FTP server.

Function Declaration

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

Function Parameters

Description of function parameters:

Parameter	Value	Comment
FtpConnection	Handle associated with the connection	Return value when establishing the connection
ServerDir	Directory name	Name of the directory to be removed

Result of the Function

If the result of this function is negative, an error has occured. If the returned value is "0", the system failed to remove the directory on the FTP server.

Result of the Function	
0	No error
-1	Invalid handle
-2	Communication error (e.g. no response from FTP server)
-3	Error message from FTP server (e.g. directory does not exist)

Using this Function

This function can be used and its result be assigned to a variable for further utilization in the following way:

```
Task FtpClientTask Autorun
    Var
        FtpHandle: Handle;
        Result:    Int;
    End_Var;

    ....
    Result := FtpDirRemove(FtpHandle,
                           'DataFiles');
    ....
End_Task;
```

How it works

The task waits at the program line until subdirectory "DataFiles" in the file system of the FTP server will be removed or an error will occur. This function is processed in the following steps:

Step	Description
1	The controller sends a command to the FTP server that directory "DataFiles" is to be removed.
2	<table><tr><th>If ...</th><th>... Then ...</th></tr><tr><td>no errors occurred</td><td>the directory is removed, the function is terminated and value "0" is returned</td></tr><tr><td>errors have occurred</td><td>the function is terminated and a negative value is returned</td></tr></table>

Directory Names

- The function parameter for the file on the FTP server may contain the complete path to this directory - even across several directory levels - if this feature is supported by the file system.
- If this feature is not supported, the user must navigate from one directory to the next until the corresponding directory is reached. This is done using the command `FtpDirChange(...)`.
- The file system of a JetControl PLC supports both options.

Related Topics:

- **Creating a Directory on the FTP Server** on page 27

Determining the Current Directory on the FTP Server

Introduction

This function is for determining the directory currently set on the FTP server.

Function Declaration

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

Function Parameters

Description of function parameters:

Parameter	Value	Comment
FtpConnection	Handle associated with the connection	Return value when establishing the connection
str	String address	This address specifies where the name of the current directory with its path is to be stored.

Result of the Function

If the returned value is negative, an error has occured. If the returned Value is "0", the system failed to determine the current directory on the FTP server.

Result of the Function	
0	No error
-1	Invalid handle
-3	Communication error (e.g. no response from FTP server)
-4	Error message from FTP server
-5	Invalid response from server

Using this Function

This function can be used and its return value be assigned to a variable for further utilization in the following way:

```
Task FtpClientTask Autorun
    Var
        FtpHandle: Handle;
        Result:    Int;
        Directory: String;
    End_Var;

    ....
    Result := FtpDirPrint(FtpHandle,
                          Directory);
    ....
End_Task;
```

How it works

The task waits at the program line until the current directory in the file system of the FTP server will be determined or an error will occur. This function is processed in the following steps:

Steps	Description
1	The controller sends a command to the FTP server that the current directory is to be determined.
2	<table><tr><th>If ...</th><th>... Then ...</th></tr><tr><td>no errors occurred</td><td>the variable contains the complete path of the current directory, the function is terminated and value "0" is returned</td></tr><tr><td>errors have occurred</td><td>the function is terminated and a negative value is returned</td></tr></table>

Related Topics:

- **Changing a Directory on the FTP Server** on page 25

2.2.2 Registers

Introduction

This chapter describes the registers on the controller from which status information on the FTP client can be read out. 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.

Content

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

Controller	Basic Register Number	Register Numbers
JC-24x		
JM-D203-JC24x		
JC-340, JC-350, JC-360, JC-940MC	320000	320000 ... 350101

Determining the Register Number

In this chapter, only the last 3 figures of a register number are specified. To calculate the actually used register number, the basic register number of the corresponding controller must be added.

Overview of Registers

FTP client module registers - Overview

Registers	Description
MR 000	Number of open connections
MR 002	Timeout in seconds
MR 003	Port number of FTP server
MR 004	Index in connection table
MR 005	Connection handle
MR 006	IP address of FTP server
MR 007	Port number of FTP server
MR 008	IP address of FTP client
MR 009	Port number of FTP client
MR 100	State of application task
MR 101	ID of application task

Register 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

This register is for setting the timeout for FTP client access to the FTP server.

Module Register Properties		
Values	0 ... 2,147,483,647	in seconds
Value following reset	20	

MR 003

Port number of FTP server

This register is for setting the IP port number of the FTP server.

Module Register Properties	
Values	0 ... 65,535
Value following reset	21

MR 004

Index in connection table

This register is for entering the index of the connection table. 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 009.

Module Register Properties	
Values	0 ... [MR 000] - 1
Value following reset	-1

MR 005

Connection handle

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

Module Register Properties	
Values	0 ... 2,147,483,647

MR 006

IP address of FTP server

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

Module Register Properties	
Access	Read access
Takes effect	if MR 004 >= 0

MR 007

Port number of FTP server

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

Module Register Properties	
Access	Read access
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 access
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 access
Takes effect	if MR 004 >= 0

MR 100

Processing status on part of FTP client

With the help of this register the user can 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	Communication with the FTP server is in progress
	3	Access completed
Access	Read access	

MR 101

Task ID

The ID of the task that is just carrying out an FTP client function can be seen from this register.

Module Register Properties		
Values	0 ... 99	Task ID
	255	None of the tasks is carrying out an FTP function
Value following reset	255	
Access	Read access	

3 Automatic Copying of Controller Data

Introduction

This chapter describes the AutoCopy function which allows to copy data within the controller and/or between controller and an FTP server. 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 Within the Local File System

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

Functions Within the File System of an FTP Server

The following functions can be performed:

- Copying files from the FTP server
- Copying files to the FTP server
- Deleting files
- Changing directories
- Creating directories
- Deleting directories

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, modules on the system bus, network devices)
- Duplication of a control system

Prerequisites

The following requirements must be met:

- the programmer must be familiar with the file system of the controller JC-340
- the programmer must have basic knowledge in the area of FTP application
- 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

3.1 Operating Principle

Introduction

This chapter describes how the AutoCopy funcion is started and how it is executed by the JC-340.

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 stages:

Stage	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
1	
2	
3	

R	E	D1	D2	SD	State
4Hz	4Hz	4Hz	4Hz	OFF	Reset

R	E	D1	D2	SD	State
1Hz	OFF	OFF	ON	OFF	The bootloader is running and is checking the OS

R	E	D1	D2	SD	State
1Hz	OFF	OFF	OFF	OFF	The OS reads the backplane DIP switch settings and checks if an Ethernet switch exists

<table>
<tr><th>Level</th><th>Description</th></tr>
<tr><td>4</td><td>
<table>
<tr><th>R</th><th>E</th><th>D1</th><th>D2</th><th>SD</th><th>State</th></tr>
<tr><td>1Hz</td><td>ON</td><td>OFF</td><td>OFF</td><td>OFF</td><td>The OS initializes the realtime clock and file system</td></tr>
</table>
</td></tr>
<tr><td>5</td><td>
<table>
<tr><th>R</th><th>E</th><th>D1</th><th>D2</th><th>SD</th><th>State</th></tr>
<tr><td>1Hz</td><td>ON</td><td>ON</td><td>OFF</td><td></td><td>The OS initializes the modules on the JX3 and JX2 system bus and the SD card</td></tr>
</table>
</td></tr>
<tr><td>6</td><td>
<table>
<tr><th>R</th><th>E</th><th>D1</th><th>D2</th><th>SD</th><th>State</th></tr>
<tr><td>1Hz</td><td>OFF</td><td>ON</td><td>OFF</td><td></td><td>The command file of the AutoCopy function is being processed</td></tr>
</table>
</td></tr>
<tr><td>7a</td><td>
<table>
<tr><th>R</th><th>E</th><th>D1</th><th>D2</th><th>SD</th><th>State</th></tr>
<tr><td>1Hz</td><td>OFF</td><td>1Hz</td><td>OFF</td><td>OFF</td><td>AutoCopy function is completed; no errors occurred</td></tr>
</table>
</td></tr>
<tr><td>7b</td><td>
<table>
<tr><th>R</th><th>E</th><th>D1</th><th>D2</th><th>SD</th><th>State</th></tr>
<tr><td>1Hz</td><td>ON</td><td>1Hz</td><td>OFF</td><td>OFF</td><td>AutoCopy function is completed; errors occurred</td></tr>
</table>
</td></tr>
</table>

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.

3.2 The File "autocopy.ini"

Introduction

This chapter covers the structure of the file "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 JC-340.

Example

```
[COMMAND_1]
Command = DirCreate
Path    = /Homepage
ErrorAsWarning = 1

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

[COMMAND_3]
Command        = FtpConnect
ServerAddr     = 192.168.123.45
UserName       = admin
Password       = admin
```

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.

Troubleshooting

When an error occurs while a command is being processed, the corresponding entry in the log file is made. For each command the user can set, whether the error is entered into the log file as `Error` or as `Warning`. This setting is made through the optional parameter `ErrorAsWarning`:

ErrorAsWarning	Entry in log file
Parameter does not exist	Error
ErrorAsWarning = 0	Error
ErrorAsWarning = 1	Warning

File Names

- The function parameter for the local file may contain the path to this file (e.g. `"/Data/TestFiles/LocalTestFile.txt"`).
- The function parameter for the file on the FTP server may contain the path to this file if this feature is supported by the file system. If this feature is not supported, the corresponding directory must be set using the command `FtpDirChange(...)`.
- The file system of a JetControl PLC supports both options.

Available Commands in the Local File System

The following commands are available for access to the local file system:

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 JC-340
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	▪ Registers ▪ 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 JC-340
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 JC-340 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
```

Available Commands for Access via FTP

The following commands are available for access via network using FTP:

Command = FtpConnect

Function	Establishing a connection to an FTP server
Parameter name 1	ServerAddr
Parameter value 1	IP address or name of FTP server
Allowed values	▪ IP address of the FTP server ▪ Name which can be resolved through DNS
Illegal values	▪ IP address other than tat of the FTP server ▪ Name which cannot be resolved
Parameter name 2	UserName
Parameter value 2	User name for logging on at the FTP server
Parameter name 3	Password
Parameter value 3	Password for logging on at the FTP server
In the case of a illegal values	Connection will not be established and the error message will be entered into the log file

Example

```
[COMMAND_1]
Command = FtpConnect
ServerAddr = 192.168.123.45
UserName   = admin
Password   = admin
```

Comment	Only one connection with an FTP server can be established at a time. If a connection to another FTP server is to be established, the JC-340 terminates the existing connection beforehand.

Command = FtpFileRead	
Function	Copying file from FTP server into the local file system
Parameter name 1	ServerFile
Parameter value 1	Complete name of the source file in the FTP server
Parameter name 2	ClientFile
Parameter value 2	Complete name of the destination file in the local file system
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_8]`
	`Command  = FtpFileRead`
	`ServerFile  = /app/cantest/cantest.es3`
	`ClientFile  = /SD/cantest3.es`

Command = FtpFileWrite	
Function	Copying file from the local file system into the file system of the FTP server
Parameter name 1	ServerFile
Parameter value 1	Complete name of the destination file in the FTP server
Parameter name 2	ClientFile
Parameter value 2	Complete name of the source file in the local file system
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_5]`
	`Command  = FtpFileWrite`
	`ServerFile  = /System/OS/op_system.os`
	`ClientFile  = /SD/OS/JC-340_1.09.0.00.os`

Command = FtpFileRemove	
Function	This command is for deleting a file in the FTP server
Parameter name	ServerFile
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_9]`
	`Command    = FtpFileRemove`
	`ServerFile = /sub1/Manual.pdf`

Command = FtpDirChange

Function	Changing the working directory in FTP server
Parameter name	ServerDir
Parameter value	Complete name of the directory
Allowed values	All valid directory names
Illegal values	Invalid directory names
In case of an illegal value	The directory will not be changed and the error message will be entered into the log file

Example	`[COMMAND_12]`
	`Command = FtpDirChange`
	`ServerDir   = /Data/MyTestData`

Commando = FtpDirCreate

Function	This command is for creating a subdirectory in FTP server
Parameter name	ServerDir
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_6]`
	`Command     = FtpDirCreate`
	`ServerDir   = /Data/MyTestData`

Restriction	If a directory with the corresponding path is specified as function parameter, all directories up to the directory to be created must exist. Recursive creation of several directories is not supported.

Commando = FtpDirRemove

Function	This command is for removing a subdirectory in FTP server
Parameter name	ServerDir
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 removed and the error message will be entered into the log file

Example	`[COMMAND_8]`
	`Command  = FtpDirRemove`
	`ServerDir  = /Data/MyTestData`

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:

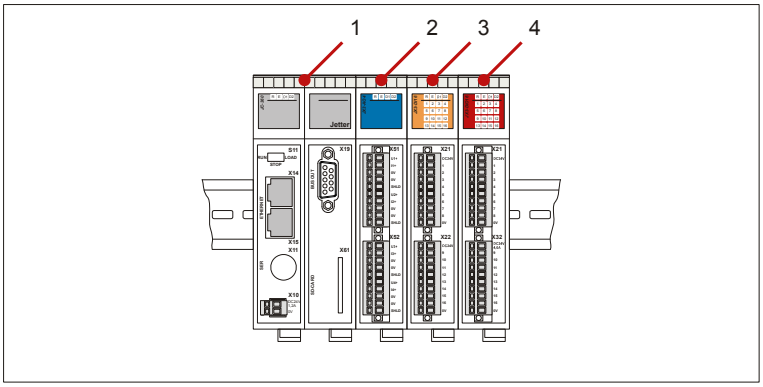
- Operating system update for the controller
- Operating system update for an analog output module
- New application program
- New values for some of the registers

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	Description	Function
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 from the controller's point of view before the AutoCopy function is executed:

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
ErrorAsWarning = 1

[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
```

3.3 Log File

Introduction

This chapter covers the structure and contents of the log file into which the results of each command are 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: Warning - 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: Error   - DaFileRead /SD/UserData/MyTestData.da

Command statistics:
  Total  : 7
  Ok     : 5
  Warning: 1
  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 message occurs (which will be entered as warning) when trying to create the directory "/app" as this directory already exists. When reading the DA file an error occurs, too. The corresponding error message is entered into the log file.

3.4 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
```

3.5 System Command Registers

Introduction

The system command register can be used to influence the performance of the JC-340 controller.

Procedure

To issue a system command proceed as follows:

Step	Action
1	Enter the correct value into the system password register.
2	Enter the required command value into the system command register.
3	Check the corresponding bits in the system status register, where applicable.
⇨	**Result:** The controller sets the system password register and the system command register to 0.

Contents

Description of System Command Registers

Overview of Registers

The following registers are used in this manual:

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

R 202960

System password register

To allow access to the system command register the system password 1112502132 (0x424F6F74) must be entered into this register. Once a value has been entered into the system command register, the controller sets the system password register to zero.

Register properties	
Value	1112502132 (0x424F6F74)

R 202961

System command register

System commands are entered into this register. Commands entered into this register are processed by the controller. Once the controller has executed the command, it sets the system command register to 0.

Commands	
102	**Restart controller**
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**

Register properties	
Access	System password register contains the correct password.

R 202962

System status register

The system status register can be used to evaluate 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 performed 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

Register properties	
Access	Read access

Description of System Commands

System Command 102

Restart controller

Effect:

The controller will restart. This command corresponds to power cycling the controller.

Application:

This command can be used, for example, if changes have been made to system registers or system files which become active only when the controller is rebooted.

System Command 122

Wait for communication OFF

Effect:

The controller starts communication with external communcation partners only when concrete requests have been received from them.

Advantage:

The application program can be processed faster.

Disadvantage:

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

System Command 123

Wait for communication ON

Effect:

The controller cyclically waits for 1 to 2 milliseconds to check for communication requests from external partners.

Advantage:

External communication partners get a faster reply from the controller.

Disadvantage:

Application program processing takes slightly longer.

System Command 160

Task switch on I/O access OFF

Effect:

While the controller accesses modules on the JX2 or JX3 system bus other tasks of the application program are not processed.

Advantage:

I/O accesses are executed 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

Task switch on I/O access ON

Effect:

While the controller accesses modules on the JX2 or JX3 system bus other tasks of the application program are processed.

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 execution time of certain I/O accesses is affected by the processing time of other tasks.

4 Fixed Software Bugs

Introduction

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

Contents

Axis Instructions Pos, ActualPos, AxArr

Effects of this Bug

Axis instructions `Pos, ActualPos and AxArr` used to program JX2 system bus modules JX2-SV1, JX2-SM2, and JX2-SM1D fail to function.

Affected Versions/Revisions

The following versions/revisions are affected by this bug:

OS version	JC-340/350 JC-360	< 1.09.0.00 < 1.09.0.00
Hardware revision	not applicable	
Configuration or operating mode	not applicable	

Remedy / Workaround

Use register access to program axis modules.

Bug Fix

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

OS version	JC-340/350 JC-360	1.09.0.00 1.09.0.00
Hardware revision	not applicable	
Configuration or operating mode	not applicable	

Axis Numbers

Axis numbers for JX2-Slave modules connected to the JX2 system bus of a JC-3xx consist of the following elements:

2	0	0	0	0	0	0	m	a

Element	Description	Value range
m	Number of slave module	2 ... 9
a	Number of the axis on the module	1 ... 9

Task restart fails to abort a delay instruction

Effects of this Bug

If a task residing at a `Delay()` gets reset by another task using the instruction `Task Restart`, the task so reset restarts only when the delay time has elapsed.

Affected Versions/Revisions

The following versions/revisions are affected by this bug:

OS version	JC-340/350 JC-360	< 1.09.0.00 < 1.09.0.00
Hardware revision	not applicable	
Configuration or operating mode	not applicable	

Remedy / Workaround

There is no remedy for affected versions/revisions.

Bug Fix

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

OS version	JC-340/350 JC-360	1.09.0.00 1.09.0.00
Hardware revision	not applicable	
Configuration or operating mode	not applicable	

Crash in the case of missing library

Effects of this Bug

If an application program has been created using a certain library and this library is missing in the controller on program start, then the controller crashes.

Affected Versions/Revisions

The following versions/revisions are affected by this bug:

OS version	JC-340/350 JC-360	< 1.09.0.00 < 1.09.0.00
Hardware revision	not applicable	
Configuration or operating mode	not applicable	

Remedy / Workaround

The workaround for this bug is as follows:

Step	Action
1	Switch the controller off.
2	Set the mode selector to "STOP" position.
3	Then, switch the controller on.
4	Upload the application program from JetSym to the controller.
5	Set the mode selector to "RUN" and restart the controller.

Bug Fix

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

OS version	JC-340/350 JC-360	1.09.0.00 1.09.0.00
Hardware revision	not applicable	
Configuration or operating mode	not applicable	

NetCopyVarToReg does not work properly in conjunction with floating point variables

Effects of this Bug

If a network client copies the content of floating point variables into float registers using the function `NetCopyVarToReg()`, wrong values are entered into the registers.

Affected Versions/Revisions

The following versions/revisions are affected by this bug:

OS version	JC-340/350 JC-360	< 1.09.0.00 < 1.09.0.00
Hardware revision	not applicable	
Configuration or operating mode	not applicable	

Remedy / Workaround

From within the application program of the network client which is to copy the values store the values to float registers and, then, use the function `NetCopyRegToReg()`.

Bug Fix

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

OS version	JC-340/350 JC-360	1.09.0.00 1.09.0.00
Hardware revision	not applicable	
Configuration or operating mode	not applicable	

The serial interface is not properly initialized after booting

Effects of this Bug

At the beginning of the application program the serial interface is to be initialized by entering the corresponding values into its registers. However, the controller fails to adopt the entered values. Thus, the interface does not work properly in the given application. This problem occurs only after booting the controller. When the application program is restarted (e.g. from JetSym) the interface is initialized correctly.

Affected Versions/Revisions

The following versions/revisions are affected by this bug:

OS version	JC-340/350 JC-360	< 1.09.0.00 < 1.09.0.00
Hardware revision	not applicable	
Configuration or operating mode	not applicable	

Remedy / Workaround

Insert a short delay (e.g. `Delay(T#10 ms)` into the application program before the first access to registers of the serial interface after program start.

Bug Fix

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

OS version	JC-340/350 JC-360	1.09.0.00 1.09.0.00
Hardware revision	not applicable	
Configuration or operating mode	not applicable	

Error when storing floating point registers to a data file

Effects of this Bug

When floating point registers are stored to a data file wrong values are entered into this file. Both the AutoCopy function and `System Function` 90 and 91 for entering values into a data file within the application program are affected by this bug.

Affected Versions/Revisions

The following versions/revisions are affected by this bug:

OS version	JC-340/350 AutoCopy System function JC-360 AutoCopy System function	 < 1.09.0.00 = 1.08.0.16 < 1.09.0.00 = 1.08.0.11
Hardware revision	not applicable	
Configuration or operating mode	not applicable	

Remedy / Workaround

There is no remedy for affected versions/revisions.

Bug Fix

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

OS version	JC-340/350 JC-360	1.09.0.00 1.09.0.00
Hardware revision	not applicable	
Configuration or operating mode	not applicable	

Changes in the input condition of the JX2-ID8 module are not recognized

Effects of this Bug

Reading a digital input of a JX2-ID8 module from within the application program sometimes fails to return the correct results. Both the IN instruction and access via registers of combined inputs are affected by this bug.

Affected Versions/Revisions

The following versions/revisions are affected by this bug:

OS version	JC-340/350 JC-360(MC)	< 1.09.0.00 < 1.09.0.00
Hardware revision	not applicable	
Configuration or operating mode	no CANopen® modules on the JX2 system bus	

Remedy / Workaround

The workaround for this bug is as follows:

Step	Action
1	In R 200002077 set bit 6 = TRUE. This setting prevents the controller JC-340 from searching for connected CANopen® modules.
2	Reboot the controller.

Bug Fix

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

OS version	JC-340/350 JC-360(MC)	1.09.0.00 1.09.0.00
Hardware revision	not applicable	
Configuration or operating mode	not applicable	

Rapid changes in input states of digital inputs on the JX2 system bus

Effects of this Bug

If the state of digital inputs of modules on the JX2 system bus is changing in rapid succession (1 ... 5 ms), an error message on the controller JC-340 may result.

This bug causes the following effects:

- The red error LED on the controller JC-340 is lit.
- In R 210004 bit 2 = 1 "JX2 system bus error" is set.
- Reading the JX2 system bus registers 200xxxxxx returns wrong values.

Affected Versions/Revisions

The following versions/revisions are affected by this bug:

OS version	JC-340	< 1.09.0.00
	JC-350	< 1.09.0.00
	JC-360	< 1.09.0.00
	JC-360MC	< 1.09.0.00
Hardware revision	not applicable	
Configuration or operating mode	not applicable	

Remedy / Workaround

The only workaround is to increase the interval between changes in the state of digital inputs on the JX2 system bus. However, this workaround is difficult to implement in practice.

Bug Fix

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

OS version	JC-340	1.09.0.00
	JC-350	1.09.0.00
	JC-360	1.09.0.00
	JC-360MC	1.09.0.00
Hardware revision	not applicable	
Configuration or operating mode	not applicable	