BUCHER automation

Application-oriented Manual



60881632_03

File system

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

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

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

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

www.bucherautomation.com

Translation of the german original User Manual

Revision	1.40
Date of issue	12/10/2025

Table of Contents

1 File system ... 6
1.1 Directories ... 6
1.1.1 Directories for controllers ... 7
1.1.2 Directories referring to HMIs ... 7
1.2 Properties ... 9
1.2.1 Flash disk - Properties ... 10
1.2.2 SD memory card - Properties ... 10
1.2.3 USB flash drive - Properties ... 11
1.3 User administration ... 11
1.3.1 Definition of terms ... 12
1.3.2 Administration of users ... 12
1.3.3 Factory settings/predefined users and keys ... 13
1.3.4 Assigning locks ... 14
1.3.5 Assigning names to keys/locks ... 16
1.3.6 Advanced user management for JetControl JC-4xx and JC-9xx ... 17
1.4 Reviewing the flash disk capacity used ... 20
1.4.1 Disk space utilization ... 20
1.5 Formatting and checking ... 22
1.5.1 Formatting the flash disk ... 23
1.5.2 Formatting the SD card ... 23
1.5.3 Formatting the USB flash drive ... 23
1.5.4 Checking the SD card ... 24
1.5.5 Checking the USB flash drive ... 24
2 E-mail feature ... 26
2.1 Configuring the E-mail feature ... 26
2.1.1 Structure of the configuration file ... 26
2.1.2 Section [SMTP] ... 27
2.1.3 Section [POP3] ... 29
2.1.4 Sektion [DEFAULT] ... 31
2.1.5 Configuration file - Examples ... 31
2.2 Create email ... 31
2.2.1 Name of the e-mail template file ... 32
2.2.2 Structure of the e-mail template file ... 32
2.2.3 Inserting real-time controller values ... 34
2.3 E-mail feature ... 34
2.3.1 STX-Funktion EMailSend() ... 34

2.4 Registers ... 35
2.4.1 Register overview ... 35
2.4.2 Register description ... 36
3 FTP server ... 39
3.1 Logging in ... 39
3.2 FTPS Server ... 40
3.2.1 User management ... 40
3.2.2 Log in as "admin" ... 40
4 HTTP server ... 41
4.1 Server Side Includes ... 41
4.1.1 First entry in the HTML file ... 42
4.1.2 Inserting real-time controller values ... 42
4.1.3 Example of an HTML page ... 46
5 FTP client ... 48
5.1 Programming ... 48
5.1.1 Initialize the FTP client ... 49
5.1.2 Establishing a connection to the FTP server ... 49
5.1.3 Terminating a connection ... 51
5.1.4 Reading a file ... 51
5.1.5 Writing to a file ... 52
5.1.6 Deleting a file ... 54
5.1.7 Changing directories ... 55
5.1.8 Creating a directory ... 56
5.1.9 Deleting directories ... 57
5.1.10 Determining the current directory ... 58
5.2 Registers ... 59
5.2.1 Register numbers ... 59
5.2.2 Register description ... 59
6 AutoCopy - Automatic copying of controller data ... 62
6.1 Operating principle ... 63
6.1.1 Launching the AutoCopy feature ... 63
6.1.2 Executing AutoCopy commands ... 64
6.1.3 Terminating AutoCopy mode ... 67
6.2 autocopy.ini - Structure ... 67
6.2.1 Section [OPTIONS] ... 67
6.2.2 Command sections ... 69
6.2.3 Example of a command file ... 77
6.3 Log file ... 79

6.3.1 File contents 79
6.4 Data files 80
6.4.1 File format 80
7 Save and load application program 82
7.1 Application program - Default path 82
7.2 Saving the application program to the SD memory card or the USB flash drive 82
7.3 Loading an application program 84

1 File system

This chapter describes the file system. The file system allows access to files on the internal flash disk, SD card, and USB stick.

When problems arise that need to be solved, it is helpful to understand the file system.

Note

Be careful when working with the file system, especially with system files. Corrupted system files can result in a device that no longer boots.

Not all files are readable, writable, or deletable. This behavior is normal. Some of these files are virtual files, such as firmware images, or protected files, such as EDS files.

Distribution

The file system distinguishes between the following files:

– System directories/files used by the operating system
– Files used by the user

1.1 Directories

System directories

System directories cannot be deleted. They are still present even after formatting.

Control systems

Separate directory names with a forward slash "/" and not with a backslash "\".

Directory	Description
/System	– system configuration – system information
/SD	– Root directory of the SD card
/USBx	– Root directory of the USB stick x

Tab. 1: system directories

Control devices

Separate directory names with a backslash “\” rather than a slash “/”.

Directory	Description
\System	– system configuration – system information – Welcome screen (boot image) – Screenshot
\SD	– Root directory of the SD card
\USB	– Root directory of the USB stick
\App	– Memory space for the application
\Data	– Storage area for the data

Directory	Description
\Windows	– Windows CE system directory
\	– RAM-Disk

Tab. 2: control devices

1.1.1 Directories for controllers

Directory /SD

/SD

When you insert an SD card into the SD card slot, the /SD directory is created dynamically. Otherwise, this directory is not visible.

Directory /System

/System

This directory contains system-relevant files such as the kernel, co-processor firmware, configurations, EDS, etc.

Directory /USBx

/USBx

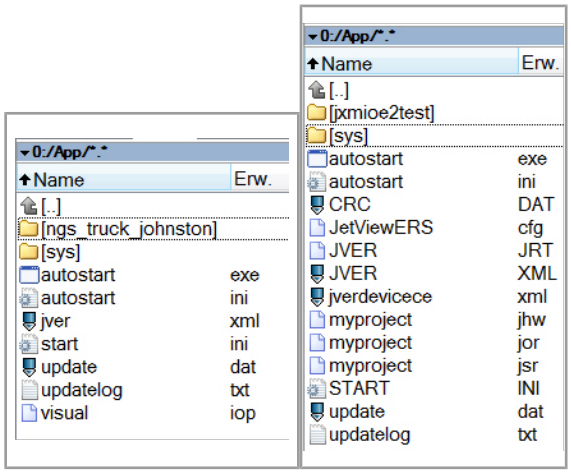
When you insert a USB stick into the USB port x, the directory /USBx is created dynamically. Otherwise, this directory is not visible.

1.1.2 Directories referring to HMIs

Directory \App

\App

This directory contains the application and visualization. In this example, the STX application is stored in the ngs_truck_johnston (left) or jxmioe2test (right) folders.



The data on the left is based on the older S platform. The visualization is stored in an .iop file, in this case the file visual.iop.

The data is based on the new CE platform. There is no .iop file. Instead, JetViewSoft saves multiple visualization files.

NOTICE



Copying to the wrong folders causes delays during startup

► Copy all application and visualization files to the **App** folder, **not** the **Data** folder. See the \Data directory.

start.ini

This text file defines which application will be started.

\App\sys

This directory holds the interpreter of the STX programming language and of the visualization software. Do not make any changes here!

autostart.exe

This application lets you update the operating system. Do not make any changes here!

Further, this file lets you start the device plus its visualization feature.

updatelog.txt

This is a log file which is created during an OS update.

Directory \Data

\Data

This directory holds the HMI's bulk data. The HMI lets you store parameter or configuration files to this directory.

NOTICE! Larger amounts of data can be stored on the data partition. To ensure a quick system start, this partition may be mounted with a delay after the STX application has been started. The application cannot be stored on this partition.

Directory \SD

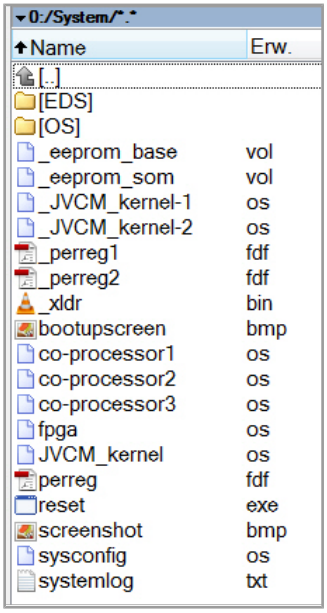
\SD

When you insert an SD card into the SD card slot on the HMI, the \SD directory is created dynamically. Otherwise, this directory is not visible.

Directory \System

\System

This directory contains system-relevant files such as the kernel, co-processor firmware, configurations, EDS, etc.



bootupscreen.bmp

This file is a 16-bit bmp file (r5, g6, b5) which is displayed while the device is booting.

You may create an image of your own and replace this file.

co-processor1

This virtual file holds the firmware of a hidden co-processor controlling most of the interactions with the user (buttons, buzzer, background lighting, etc.).

reset.exe

Deleting this file triggers the HMI to reboot immediately. You can use this function in batch files, for example, which, after complete processing, require automatic rebooting.

Directory \USB

\USB

When you insert a USB stick into the HMI, the \USB directory is created dynamically. Otherwise, this directory is not visible.

Directory \Windows

\Windows

This directory holds the Windows CE files. **Do not make any changes here!**

1.2 Properties

This chapter covers the properties of the file system. A distinction is made between controllers and HMI devices and then again between the internal flash disk, the SD card and the USB stick.

Controllers: General properties

The following properties apply to the internal flash disk and the SD card:

– 8 files max. to be opened simultaneously
– Separate directory names with a forward slash "/" and not with a backslash "\".
– When the controller creates a file, the file contains date and time assigned by the controller.
– Date, time, and file size are not available for all system files.

HMIs: General properties

The following properties apply to the internal flash disk drive, SD memory card, and USB flash drive:

– 8 files max. to be opened simultaneously
– Only apply lower case for directory and file names.
– When the device creates a file, it assigns its date and time.
– Date, time, and file size are not available for all system files.

1.2.1 Flash disk - Properties

The memory size is dependent on the respective device.

Controllers: Properties

The internal flash drive has the following properties:

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

HMIs: Properties

The internal flash drive has the following properties:

– Supports up to 7 directory levels and 1 file level.
– Upper- and lower-case are not distinguished.
– Directory and file names are permitted to have a total length of 63 characters.
– All characters except "\" and ".." are permitted for directory and file names.
– The location of the directories "App" and "Data" is on the flash disk drive.
– There is no user/access management.

1.2.2 SD memory card - Properties

The memory size is dependent on the respective device.

Controllers: Properties

The SD memory card has got the following properties:

– The SD memory card must be compatible with FAT 16.
– No case sensitivity.
– Directory and file name length must not exceed 63 characters.
– The following characters are not permitted in directory and file names: "/", "\", ":", "*", "?", "<", ">" and "|"
– The number of subdirectory levels depends on the formatting.
– There is no user/access management.

Bucher Automation AG shall be liable only for those SD cards marketed as an option.

HMIs: Properties

The SD memory card has got the following properties:

– The SD memory card must be compatible with FAT 16 or FAT 32.

To store data to the SD memory card, please mind:

– Directory and file names are permitted to have a total length of 260 characters.
– Upper- and lower-case are not distinguished.
– The following characters are not permitted in directory and file names: "/", "\", ":", "*", "?", "<", ">" and "|"

Bucher Automation AG shall be liable only for those SD cards marketed as an option.

1.2.3 USB flash drive - Properties

The memory size is dependent on the respective device.

Controllers: Properties

The USB flash drive has the following properties:

– The USB flash drive must be compatible with FAT 12, FAT 16, or FAT 32.
– No case sensitivity.
– Directory and file name length must not exceed 63 characters.
– The following characters are not allowed: "/", "\", ":", "*", "?", "<", ">" and "|"
– The number of subdirectory levels depends on the formatting.
– There is no user/access management.

HMIs: Properties

The USB flash drive has the following properties:

– The USB flash drive must be compatible with FAT12, FAT16, or FAT32.
– Directory and file names are permitted to have a total length of 260 characters.
– Upper- and lower-case are not distinguished.
– The following characters are not permitted in directory and file names: "/", "\", ":", "*", "?", "<", ">" and "|"
– There is no user/access management.

The Bucher Automation AG only provides a warranty for the USB sticks available as an option.

1.3 User administration

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

Access rights (keys) can be set for each standard user.

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

Requirements

Administrator rights are required for user administration.

Properties

The properties of user administration are as follows:

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

Tab. 3: Properties

Files

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

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

Tab. 4: Files

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

Restrictions

Please take the following restrictions into account:

– User administration can only be applied to the internal flash disk. It cannot be applied to SD cards and USB flash.

– If user administration has been assigned to a file, its contents are readable at once. The settings become active only after a reboot.

1.3.1 Definition of terms

Term	Description
Key	Keys for directories, files, or permissions
admin	The user named admin with admin rights
system	The user named system with system rights
XCOM	XCOM is a protocol for communication between JetSym and JetControl controllers from Bucher Automation.

Tab. 5: Terms

1.3.2 Administration of users

The configuration file /System/users.ini lets you manage the users of the file system.

Requirements

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

Administration of users

To manage user administration, proceed as follows:

1. Establish an FTP connection to the device. Log on as administrator.
2. Open the file "/System/users.ini".
3. Change the file entries.
4. Save the modified file back to the device.
5. Reboot the device.

⇨ The modified user administration settings are now enabled.

Structure of the configuration file

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

– For each user a separate section is to be created.

– In these sections values can be set which are then used by the file system.

– You can insert blank lines as required.

– The following characters precede a comment line: "!", "#" or ";".

Sections

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

Example

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

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

1.3.3 Factory settings/predefined users and keys

Two predefined users with set rights are included in the file system. It is not possible to delete these two users. In the user administration only the password can be changed for these two users.

Factory settings

The factory settings include the content of the configuration file in the controller as follows:

```
[USER1]
NAME=admin
PW=admin
READKEYS=1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,2
6,27,28,29,30,31
WRITEKEYS=1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,
26,27,28,29,30,31
SYSKEYS=PlcControl, ProgControl, ProgDebug, ProgProfile, ProgDownload,
OsDownload, VarRead, VarWrite, FileRead, FileWrite
[USER33]
NAME=system
PW=system
READKEYS=2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,
27,28,29,30,31
WRITEKEYS=2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26
,27,28,29,30,31
SYSKEYS=PlcControl, ProgControl, ProgDebug, ProgProfile, ProgDownload,
OsDownload, VarRead, VarWrite, FileRead, FileWrite
```

Predefined users

The user admin has all keys and can therefore access all directories and files for reading and writing.

The user system also has all keys except key 1.

Predefined keys

Two out of the 31 keys have a predefined function.

Lock/key	Function
1	– Ethernet configuration – User administration
2	– Operating system update of the CPU – Operating system update for peripheral modules

Tab. 6: Predefined keys

1.3.4 Assigning locks

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

Requirements

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

Assigning locks

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

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

⇨ A lock is assigned to this directory.

Structure of the configuration file

This configuration file is a text file containing one section.

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

Section

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

Directory=Lock

Example:

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

Lock numbers

The lock numbers have got the following properties:

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

1.3.5 Assigning names to keys/locks

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

Assigning the names

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

1. Establish an FTP connection to the device; when doing so, log in with administrator rights.
2. Open the file "/System/keys.ini".
3. Change the file entries.
4. Save the modified file back to the device.
5. Reboot the device.

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

Structure of the configuration file

This configuration file is a text file containing one section.

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

Section

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

KEYxx=Name

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

Example:

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

Names for locks and keys

For names the following definitions are true:

– A maximum of 15 alphanumeric characters
– Lock and key must have the same name

1.3.6 Advanced user management for JetControl JC-4xx and JC-9xx

For user management with increased security and rights, the previous user management system has been expanded.

This chapter essentially only documents the differences from the user management described above.

Starting with the following OS versions, there is extended user management for JC-4xx and JC-9xx.

JC-4xx: 1.15.0.08

JC-9xx: 1.23.0.14

Description of functionality

General

- You can manage the users of the file system in the **file /System/users.ini.**
  - Permissions for **XCOM** are integrated as additional keys.
  - Also includes the **"admin password" (ADMINPW)** to force the change of an individual password.
  - The **users.ini** file is visible to all users and can be modified, subject to user restrictions.
  - The changes will only be active when the control system is rebooted!

Rights of individual users

	User admin	User system	Standard User
Display	Views all entries for all users except for individual passwords.	Only sees his entry.	Only sees his entry.
Assignment of names and keys	Can assign names and keys.	No	No
Deletion/change of name	Cannot be changed or deleted.	Cannot be changed or deleted.	Can be changed or deleted by the admin.
Permissions	Cannot be changed or deleted.	Cannot be changed or deleted.	Can be changed or deleted by the admin.
Admin rights (Key 1)	Yes	No	No
Password change	Can set admin password for other users.	Can only change individual passwords.	Can only change individual passwords.

Tab. 7: Comparison of user rights

Special features Admin password

- If the administrator has entered an **admin password** for a standard user, this user must log in with this admin password.
  - His individual password will then no longer work.
  - When querying the permissions (WRITEKEYS, READKEYS, XCOMKEYS), 0 is returned until a new individual password has been entered. Until then, he does **not** have any special keys.
  - He must enter a new individual password; the previous one cannot be used again.
  - When you enter an individual password, the admin password is deleted.

Password requirements

A password must comply with all of the following conventions:

– Length 8-33 characters
– Lowercase letters a-z.
– Capital letters A-Z
– Numbers 0-9

If all 4 conditions are not met, the password will **not** be accepted.

XCOM permissions

– The name of the item for XCOM permissions is: XCOMKEYS
– XCOM permissions are entered by name only; numbers are not possible.
  – 31 permissions can be assigned.
  – The names are permanently stored in the OS.
  – The characters a-z, A-Z, and 0-9 are permitted.
  – Complies with the conventions of the other key names.

NOTICE! XCOMKEYS are not yet evaluated in current operating systems (up to 1.18.0.00 for JC-4xx and up to 1.26.0.00 for JC-9xx).

Examples for users.ini

The `users.ini` file is displayed differently depending on the logged-in user.

Log in as user "admin"

The user "admin" can see all registered users, except for their passwords (PW entry), and can make changes.

```
[USER1]
NAME=admin
ADMINPW=
PW=admin
READ-
KEYS=1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27
,28,29,30,31
WRITEKEYS=1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,
26,27,28,29,30,31
XCOMKEYS= PlcControl, ProgControl, ProgDebug, ProgProfile, ProgDownload, Os-
Download, VarRead, VarWrite, FileRead, FileWrite

[USER15]
NAME=Homer
ADMINPW=
READKEYS=15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31
WRITEKEYS=11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31
XCOMKEYS= ProgDownload, OsDownload, VarRead, FileRead

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

```
,27,28,29,30,31
XCOMKEYS= PlcControl, ProgControl, ProgDebug, ProgProfile, ProgDownload,
OsDownload, VarRead, VarWrite, FileRead, FileWrite
```

Log in as user "Homer"

```
[USER15]
NAME=Homer
ADMINPW=
PW=Simpson99
READKEYS=15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31
WRITEKEYS=11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31
XCOMKEYS= ProgDownload, OsDownload, VarRead, FileRead
```

Logging

Log file

In order to make access to user administration traceable, an entry is made in a log file for certain actions.

Path and file name	Format	Access rights
/Drive_C/UserLogIn.log	text file	admin

Tab. 8: Log file

The entries are appended to the existing file. It is a good idea to delete them from time to time so that they do not become too large.

Actions

The following actions are logged:

– Logins with unknown user names
– Logins with incorrect passwords
– Changing the ADMIN password
– Changing the user password
– Creating a new user
– Deleting an existing user

Example

Example of the contents of the log file. The message text is preceded by the date (month/day/year) and time of day.

```
09/17/2024 10:54:24    Set ADMIN password for user 'system'.
09/17/2024 10:55:54    Bad authorization of user 'system'
09/17/2024 10:58:59    Set password for user 'system'.
09/17/2024 11:12:40    Unknown user name 'Marge'.
09/17/2024 13:00:33    Add user 'Marge'.
09/17/2024 13:09:31    Remove user 'Marge'.
```

Lock login

To prevent a user password from being guessed, a user is locked out after entering the wrong password five times. The lock can only be lifted by rebooting the control system.

STX program download protection

To provide minimal protection against unauthorized downloading of an STX application program, proceed as follows:

1. Open the FTP connection to the controller as user "admin".
2. Download the file "/System/flashdisklock.ini" from the control panel and set the item "app=0" using a text editor.
3. Transfer the file to the controller.

   ⇨ When saving the modified file to the controller, all locks are deleted from the "/app" directory.
4. Download the STX application program in JetSym.
5. Download the file "/System/flashdisklock.ini" from the control panel and set the item "app=1" using a text editor.
6. Transfer the file to the controller.

   ⇨ When saving the modified file on the controller, the "admin" lock is applied to the "/app" directory.

⇨ Now only the user "admin" can access the directory "/app" and the program files it contains.

1.4 Reviewing the flash disk capacity used

The internal flash disk drive lets you retrieve disk information on the application data area.

Detailed information on the application data area are given in this chapter.

1.4.1 Disk space utilization

README

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

You can read the allocation of the area with the application data from the file /System/flashdiskinfo.txt.

Example

The example shows the possible allocation of the flash disk of a JetControl 340 (4 MB):

```
Name : flash disk
Date : 25.11.2008
Time : 15:04
Tracks: 64
Track 0: sectors: 128 (used: 81 / blocked: 47 / free: 0)
Track 1: sectors: 128 (used: 128 / blocked: 0 / free: 0)
Track 2: sectors: 128 (used: 128 / blocked: 0 / free: 0)
Track 3: sectors: 128 (used: 128 / blocked: 0 / free: 0)
Track 4: sectors: 128 (used: 128 / blocked: 0 / free: 0)
Track 5: sectors: 128 (used: 128 / blocked: 0 / free: 0)
Track 6: sectors: 128 (used: 128 / blocked: 0 / free: 0)
Track 7: sectors: 128 (used: 128 / blocked: 0 / free: 0)
Track 8: sectors: 128 (used: 128 / blocked: 0 / free: 0)
Track 9: sectors: 128 (used: 128 / blocked: 0 / free: 0)
Track 10: sectors: 128 (used: 128 / blocked: 0 / free: 0)
Track 11: sectors: 128 (used: 128 / blocked: 0 / free: 0)
Track 12: sectors: 128 (used: 128 / blocked: 0 / free: 0)
Track 13: sectors: 128 (used: 128 / blocked: 0 / free: 0)
Track 14: sectors: 128 (used: 128 / blocked: 0 / free: 0)
Track 15: sectors: 128 (used: 128 / blocked: 0 / free: 0)
```

```
Track 16: sectors: 128 (used: 128 / blocked: 0 / free: 0)
Track 17: sectors: 128 (used: 128 / blocked: 0 / free: 0)
Track 18: sectors: 128 (used: 128 / blocked: 0 / free: 0)
Track 19: sectors: 128 (used: 128 / blocked: 0 / free: 0)
Track 20: sectors: 128 (used: 64 / blocked: 64 / free: 0)
Track 21: sectors: 128 (used: 85 / blocked: 43 / free: 0)
Track 22: sectors: 128 (used: 128 / blocked: 0 / free: 0)
Track 23: sectors: 128 (used: 128 / blocked: 0 / free: 0)
Track 24: sectors: 128 (used: 128 / blocked: 0 / free: 0)
Track 25: sectors: 128 (used: 128 / blocked: 0 / free: 0)
Track 26: sectors: 128 (used: 128 / blocked: 0 / free: 0)
Track 27: sectors: 128 (used: 128 / blocked: 0 / free: 0)
Track 28: sectors: 128 (used: 128 / blocked: 0 / free: 0)
Track 29: sectors: 128 (used: 128 / blocked: 0 / free: 0)
Track 30: sectors: 128 (used: 128 / blocked: 0 / free: 0)
Track 31: sectors: 128 (used: 128 / blocked: 0 / free: 0)
Track 32: sectors: 128 (used: 128 / blocked: 0 / free: 0)
Track 33: sectors: 128 (used: 105 / blocked: 0 / free: 23)
Track 34: sectors: 128 (used: 0 / blocked: 0 / free: 128)
Track 35: sectors: 128 (used: 0 / blocked: 0 / free: 128)
Track 36: sectors: 128 (used: 0 / blocked: 0 / free: 128)
Track 37: sectors: 128 (used: 0 / blocked: 0 / free: 128)
Track 38: sectors: 128 (used: 0 / blocked: 0 / free: 128)
Track 39: sectors: 128 (used: 0 / blocked: 0 / free: 128)
Track 40: sectors: 128 (used: 0 / blocked: 0 / free: 128)
Track 41: sectors: 128 (used: 0 / blocked: 0 / free: 128)
Track 42: sectors: 128 (used: 0 / blocked: 0 / free: 128)
Track 43: sectors: 128 (used: 0 / blocked: 0 / free: 128)
Track 44: sectors: 128 (used: 0 / blocked: 0 / free: 128)
Track 45: sectors: 128 (used: 0 / blocked: 0 / free: 128)
Track 46: sectors: 128 (used: 0 / blocked: 0 / free: 128)
Track 47: sectors: 128 (used: 0 / blocked: 0 / free: 128)
Track 48: sectors: 128 (used: 0 / blocked: 0 / free: 128)
Track 49: sectors: 128 (used: 0 / blocked: 0 / free: 128)
Track 50: sectors: 128 (used: 0 / blocked: 0 / free: 128)
Track 51: sectors: 128 (used: 0 / blocked: 0 / free: 128)
Track 52: sectors: 128 (used: 0 / blocked: 0 / free: 128)
Track 53: sectors: 128 (used: 0 / blocked: 0 / free: 128)
Track 54: sectors: 128 (used: 0 / blocked: 0 / free: 128)
Track 55: sectors: 128 (used: 0 / blocked: 0 / free: 128)
Track 56: sectors: 128 (used: 0 / blocked: 0 / free: 128)
Track 57: sectors: 128 (used: 0 / blocked: 0 / free: 128)
Track 58: sectors: 128 (used: 0 / blocked: 0 / free: 128)
Track 59: sectors: 128 (used: 0 / blocked: 0 / free: 128)
Track 60: sectors: 128 (used: 0 / blocked: 0 / free: 128)
Track 61: sectors: 128 (used: 0 / blocked: 0 / free: 128)
Track 62: sectors: 128 (used: 0 / blocked: 0 / free: 128)
Track 63: sectors: 128 (used: 0 / blocked: 0 / free: 128)
```

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

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

Elements of the info file

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

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

Tab. 9: Elements of the info file

States of the sectors

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

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

Tab. 10: States of the sectors

1.5 Formatting and checking

This chapter covers the following topics:

– Formatting the flash disk
– Formatting the SD card
– Checking the SD card
– Formatting the USB flash drive
– Checking the USB flash drive

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

ⓘ Info These functions are not supported by all controllers. Check the user manual for your control system to see whether it supports the above functions.

Operating principle

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

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

– Formatting the flash disk
– Formatting the SD card
– Formatting the USB flash drive
– Checking the SD card
– Checking the USB flash drive

Register number

The number of the control register is 202936

1.5.1 Formatting the flash disk

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

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

Effect

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

Formatting the flash disk

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

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

⇨ During the boot process the flash disk is formatted and control register 202936 is set to 0.

1.5.2 Formatting the SD card

In the following cases, reformatting the SD card is required:

– When information for SD card administration has been destroyed

Effect

All files and directories on the SD card will be deleted.

Formatting the SD card

To have the device format the SD card, proceed as follows:

1. Switch the device ON.
2. Enter value -748362163 (0xd364e64d) into control register 202936 of the file system.
3. Switch the device OFF.
4. Switch the device ON.

⇨ During the boot process, the SD card is formatted and control register 202936 is set to 0.

1.5.3 Formatting the USB flash drive

Sometimes it might be necessary to reformat the USB flash drive. This might be the case when information for USB flash drive administration has been destroyed.

Effect

All files and directories on the USB flash drive will be deleted.

Formatting the USB flash drive

To format the USB flash drive proceed as follows:

1. Switch the device ON.
2. Enter value (0x8f3d5185) into the control register of the file system.
3. Switch the device OFF.
4. Switch the device ON.

⇨ During the boot process of the device the USB flash drive is formatted and the control register is set to 0.

1.5.4 Checking the SD card

In the following cases, checking the SD card for faults is required:

– If the device was turned off while accessing the SD card.

Effect

All files and directories on the SD card will be checked and errors, if any, will be fixed.

Following such a check, the administrative structures on the SD card will be in consistent condition.

Depending on the size of the SD card and the number of directories and files, the boot phase may take up to several minutes.

Checking the SD card

To have the device check the SD card, proceed as follows:

1. Switch the device ON.
2. Enter value 748371092 (0x2c9b3c94) into control register 202936 of the file system.
3. Switch the device OFF.
4. Switch the device ON.

⇨ While booting, the device checks the SD card. The value in the control register is retained so that the check is performed each time the device is restarted.

Restrictions

This function repairs the administrative structures on the SD card. You can therefore continue to use the SD card. However, it may happen that the device cannot restore in all cases data of a file, which, for example, has been written incompletely.

1.5.5 Checking the USB flash drive

Sometimes it might be necessary to check the USB flash drive for errors. This might be the case if the device was de-energized while accessing the USB flash drive.

Effect

– All files and directories on the USB flash drive will be checked and errors, if any, will be fixed.

  Following such a check, the administrative structures on the USB flash drive are in consistent condition.

– Depending on the USB flash drive capacity and the number of files and directories to be checked the boot process may take several minutes.

Check

To have the device check the USB flash drive for errors proceed as follows:

1. Switch the device ON.
2. Enter value (0x17dbd42a) into the control register of the file system.
3. Switch the device OFF.

4. Switch the device ON.

⇨ During the boot process of the device, the USB flash drive is being checked. The value in the control register remains unchanged so that the stick is checked whenever the device is rebooted.

Restrictions

This function only repairs the administrative structures on the USB flash drive so that it can be used further. However, it may happen that data of a file, which has been written incompletely, can't be restored.

2 E-mail feature

The user creates template files for emails, where the controller enters the corresponding variables when an email is being sent. This allows for automated transfer of reports or other production or maintenance information via email.

The controller sends emails to an email server that will then forward the message to the addressee.

This chapter describes how users can configure the email function and both create and send emails.

Activating the email feature

The following requirements must be met in order to activate the email function:

– The control system can be ordered with specific options. Depending on the option, the email function may or may not be enabled.

– A valid email configuration file /EMAIL/email.ini must be present during the boot phase of the controller.

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

Required programmer's skills

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

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

– Knowledge of IP networks is required.

2.1 Configuring the E-mail feature

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

The device reads the configuration data from the file /EMAIL/email.ini during the boot phase.

Requirements

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

– The IP address of the e-mail server is known.

– If the IP address of the e-mail server is not known, name resolution through a DNS server must be possible refer to Using Names for IP addresses.

– The log-on and authentication parameters at the e-mail server are known.

To obtain this information contact your network administrator.

2.1.1 Structure of the configuration file

The email client is configured using the data from the file /EMAIL/email.ini. The device reads the values during the boot process only.

Structure of the configuration file

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

– These sections are for entering values which are then used by the e-mail client.

– You can insert blank lines as required.

– The following characters precede a comment line: "!", "#" or ";".

Sections

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

Section	Configuration values
[SMTP]	– IP address and port number of the SMTP server – Log-on parameters
[POP3]	– IP address and port number of the POP3 server – Log-on parameters
[DEFAULT]	– Name of an e-mail template file containing default values

2.1.2 Section [SMTP]

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

Example:

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

Authentication

With this type of authentication, the device logs in to the SMTP server before sending the email. During the logon process USER and PASSWORD must be entered. The device supports the following authentication methods.

– LOGIN
– PLAIN
– CRAM-MD5

Configuration values

IP

Property	Description
In the given example	– 192.168.40.1 – IP address of the SMTP server; can also be specified as a name
Valid values	– > 1.0.0.0 – < 223.255.255.255
Invalid values	– Network address – Broadcast address
If the value is invalid or the entry does not exist	The e-mail feature is not available

PORT

Property	Description
In the given example	– 25 000 – Port number of SMTP server
Valid values	– > 0 – < 65.536
Invalid values	– > 65,335
In case of missing entry	– 25

HELO

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

USER

Property	Description
In the given example	– JetControl0815 – Log-on name for SMTP authentication; – If this entry exists, the entry PASSWORD is required, too.
Valid values	– String of 63 characters max.
In case of missing entry	– 25

PASSWORD

Property	Description
In the given example	– MyPassWord – Log-on password for SMTP authentication. – If this entry exists, the entry USER is required, too.
Valid values	– String of 63 characters max.
In case of missing entry	– No SMTP authentication

SECUREEMAIL

Property	Description
In the given example	– 1
Value = 1	– The SMTP server must support encryption or require encryption. – If STARTTLS = 0, the controller sends encrypted messages from the moment the connection is established.

Property	Description
	– If STARTTLS = 1, the connection is established unencrypted and encryption is then negotiated.
Value = 0	– The email is sent completely unencrypted. – A STARTTLS = 1 is not taken into account.
Valid values	0 or 1
In case of missing entry	Email is sent unencrypted.

STARTTLS

Property	Description
In the given example	– 0
Value = 1	– If the SMTP server requires STARTTLS, START-TLS must be set to 1. – The SMTP server first requires an unencrypted connection to be established via a plaintext port, e.g. 25 or 587. – After the connection has been successfully established, the server and client negotiate an encryption algorithm, e.g., TLS 1.0 or TLS 1.1. – The following message content is then sent in encrypted form via the same port.
Value = 0	– If the SMTP server does not require or support STARTTLS. – The controller has been sending encrypted messages since the connection was established.
Valid values	0 or 1
In case of missing entry	Email is sent unencrypted.

2.1.3 Section [POP3]

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

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

Example:

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

Authentication

This type of authentication requires the device to log on at the POP3 server. During the log-on process USER and PASSWORD must be entered. The SMTP server then enables the sending of emails for a certain period of time (usually 10 to 30 minutes).

Configuration values

IP

Property	Description
In the given example	– 192.168.40.1 – IP address of the POP3 server; can also be specified as a name.
Valid values	– > 1.0.0.0 – < 223.255.255.255
Invalid values	– Network address – Broadcast address
BIf the value is invalid or the entry does not exist	POP3 log-on will not be carried out.

PORT

Property	Description
In the given example	– 25,100 – Port number of POP3 server
Valid values	– > 0 – < 65.536
Invalid values	– > 65,335
In case of missing entry	110

USER

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

PASSWORD

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

2.1.4 Sektion [DEFAULT]

In this section, you specify the name of an email template file that contains default settings for sending emails. If the corresponding section is not included in the respective email template file, the device will use the definitions made here when sending emails.

Example

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

2.1.5 Configuration file - Examples

Here are some examples for the email configuration file /EMAIL/email.ini.

Minimum configuration

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

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

Authentication through POP3 Log-on

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

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

[DEFAULT]
MAILCFG    = EmailDefaults.cfg
```

Authentification through SMTP

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

```
[SMTP]
IP            = 192.168.40.1
USER            = JetControl0815
PASSWORD        = MyPassWord
SECUREEMAIL    = 1
STARTTLS        = 0
```

2.2 Create email

This chapter describes how to create an e-mail. These emails are then sent in the user program.

An email template file must be created for each email.

2.2.1 Name of the e-mail template file

Absolute file name

When using the STX function

```
EMailSend()
```

file names and directories (paths) can be chosen freely within the limits set by the file system.

2.2.2 Structure of the e-mail template file

An e-mail template file is a text file which is divided into sections. The device compiles the email from the information in these sections when sending it.

Structure

– Sections [FROM] and [TO] are mandatory. This information may be specified either in the e-mail to be sent or in the e-mail template file containing the default settings.

– All parameters in these sections can be tagged with tags containing real-time control values, see Inserting real-time control values.

E-mail template file

[FROM]

Sender

[TO]

Addressee

[CC]

Additional addressee(s)

[SUBJECT]

Subject

[ATTACHMENT]

Complete path and file name

[MESSAGE]

Email message text

Sections

[FROM]

Parameter	Description
Description	Email sender
Comment	Please check with your IT administrator which information must be entered here.
Length	63 characters
Example	`[FROM]` `JetControl@bucherautomation.com`

[TO]

Parameter	Description
Description	Email addressee
Comment	Several addressees are separated by the semicolon.

Parameter	Description
Length	255 characters
Example	[TO] service@mydomain.com

[CC]

Parameter	Description
Description	Additional e-mail addressee(s)
Comment	Several addressees are separated by the semicolon.
Length	255 characters
Example	[CC] service@mydomain.com;hotline@mydomain.com

[SUBJECT]

Parameter	Description
Description	Subject
Length	255 characters
Example	[SUBJECT] Fatal Error

[ATTACHMENT]

Parameter	Description
Description	Full path and name of a file to be appended, see example 1.
Comment	The attached file must be a text file. A string consisting of the path and file name can alternatively be read out as a real-time control value, see example 2. Only one attachment is possible.
Length	511 characters
Example 1	[ATTACHMENT] /logfiles/error_report.log
Example 2	[ATTACHMENT] <JW:DTAG name="R420" type="STRING"/>

[MESSAGE]

Parameter	Description
Description	Email message text
Comment	Text only message
Length	65,535 characters
Example	[MESSAGE] Have a nice day ! JetControl.

2.2.3 Inserting real-time controller values

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

2.3 E-mail feature

This chapter describes how to send previously created emails in the application program.

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

Processing within the application program

Sending an e-mail may take considerable time. Therefore, other tasks of the application program are processed while an e-mail is being sent.

Only one e-mail function call is possible at a time.

While an e-mail of a task is being sent, all other tasks which invoke the e-mail function are therefore blocked until this operation is completed. During this time, no TaskBreak or TaskRestart may be performed on this task, nor may the program be restarted with JetSym.

System function 110*

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

2.3.1 STX-Funktion EMailSend()

Function declaration

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

Parameter	Description
`FileName`	The file name of the email template file must be specified absolutely, including the path.

Tab. 11: Parameter FileName

Return value

Parameter	Description
0	No error
-1	Internal OS error
-2	FROM not defined
-3	TO not defined
-4	No connection to the e-mail server
-10	The e-mail feature is not available
-11/-12	Internal OS error

Tab. 12: Return value

JetSym STX program

If an error occurs in the machinery, the controller is to send an e-mail.

```
Var
     nResult: Int;
        bSendError: Bool;
End_Var;

Task tMain Autorun
        ...
        If bSendError Then
             nResult := EmailSend('/ErrorMails/MainError.email');
             bSendError := FALSE;
        End_If;

        ...
End_Task;
```

2.4 Registers

This chapter describes the registers that you can use to query the processing status of the email function.

2.4.1 Register overview

The device provides several registers that you can use to check the processing status of the email function.

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

Tab. 13: Register overview

2.4.2 Register description

R 202930

Web status

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

Registers	Description	
Access	Read	
Bit meaning		
Bit 0	**FTP server**	
	1 =	available
Bit 1	**HTTP server**	
	1 =	available
Bit 2	**Email**	
	1 =	available
	Bit 2 is set only if the /EMAIL/email.ini configuration file is available.	
Bit 3	**Data file function**	
	1 =	available
Bit 4	**Modbus/TCP**	
	1 =	existing
Bit 5	**Modbus/TCP**	
		available
Bit 7	**FTP client**	
		available
Value after reset	Depending on options purchased	

Tab. 14: Web status R 202930

292932

The IP address of the SMTP server can be found here, as specified in the file /EMAIL/email.ini.

Property	Description
Access	Read
Value after reset	Depending on configuration
Takes effect	Once R 202930.2 = 1

Tab. 15: IP address of the SMTP server R 292932

R 292933

The IP address of the POP3 server can be read here, as specified in the file /EMAIL/email.ini.

IP address of the POP3 server

Property	Description
Access	Read
Value after reset	Depending on configuration
Takes effect	Once R 202930.2 = 1

Tab. 16: IP address of the POP3 server R 292933

R 292934

IP address of the POP3 server

The port number of the SMTP server can be read here, as specified in the file /EMAIL/email.ini.

Property	Description
Access	Read
Value after reset	Depending on configuration
Takes effect	Once R 202930.2 = 1

Tab. 17: IP address of the POP3 server R 292934

R 292935

Port number of POP3 server

The port number of the POP3 server can be read here, as specified in the file /EMAIL/email.ini.

Property	Description
Access	Read
Value after reset	Depending on configuration
Takes effect	Once R 202930.2 = 1

Tab. 18: Port number of the POP3 server R 292935

R 292937

Status of e-mail processing

The execution of the email dispatch is tracked here.

Property	Description
Access	Read
Werte:	0 = No email will be sent.
	1 = Transfer of parameters to the device's email client.
	2 = The email is being compiled and the connection to the server is being established.
	3 = E-mail has been sent to server.

Tab. 19: Status of email processing R 292937

R 292938

Task ID (e-mail)

The ID of the task that is just sending an e-mail can be seen from this register.

Property	Description
Access	Read
Values:	0 … 99 = Task ID
	255 = No task, sending an email right now.
Value after reset	255

Tab. 20: Task-ID (E-Mail) R 292938

3 FTP server

The FTP server handles directories and files using an FTP client. The files can be stored on the following media:

– Internal flash drive
– SD memory card
– USB flash drive

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

FTP clients

In addition to the command line FTP client that comes with many PC operating systems, you can also use graphical FTP tools.

Maximum number of connections

The FTP server manages four simultaneously open FTP connections.

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

Supported commands

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

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

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

Required programmer's skills

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

– File system
– IP networks
– FTP commands

3.1 Logging in

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

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

Factory settings

The following users are set up in the delivery state:

[USER1]

NAME=admin

PW=admin

[USER33]

NAME=system

PW=system

Administration of users

You can change the password and add additional users via the file system's User administration [▶ 11].

3.2 FTPS Server

NOTICE! Only for JC-4xx and JC-9xx from OS versions JC-4xx: 1.14.0.00 and JC-9xx: 1.22.0.00 available.

In addition to the FTP server, an FTPS server (FTPS = FTP Secure) also runs on the controller, which can be used to establish an encrypted connection between the client and server.

– The FTPS server uses the same TCP/IP ports as the FTP server.
– The setting for whether to communicate with or without encryption is made in the FTP client.
– Encryption via TLS/SSL is supported.

3.2.1 User management

In order to achieve maximum compatibility with existing operating systems that do not yet support FTPS, all users are accessible via FTP **and** FTPS, with the exception of the "admin" user.

3.2.2 Log in as "admin"

Logging in with "admin" – and thus accessing the user management files in the "System" directory – is only possible via FTPS.

4 HTTP server

A standard browser is sufficient for accessing the HTTP server.

The browser allows you to read and display files that have been transfered to the controller via FTP.

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

Default file names

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

Supported file types

The following file types are supported:

– *.htm, *.html, *.shtml
– *.txt, *.ini
– *.gif, *.tif, *.tiff, *.bmp, *.wbmp
– *.jpg, *.jpe, *.jpeg, *.png
– *.xml
– *.js, *.jar, *.java, *.class, *.cab
– *.ocx
– *.pdf, *.zip, *.doc, *.rtf
– *.css
– *.wml, *.wmlc, *.wmls, *.wmlsc
– *.ico, *.svg
– *.map
– *.ttf, *.woff, *.woff2

Required programmer's skills

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

– File system
– IP networks

4.1 Server Side Includes

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

Rules

If you want to enter current real-time control values in the HTML page, you must specify the Name Space tag at the beginning of this HTML page.

The name space used is defined in the Tag Name Space.

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

Update real-time control values

When the HTML page is loaded into the browser, the HTTP server replaces the data tags once with current real-time control values.

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

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

4.1.1 First entry in the HTML file

Design

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

```
<NS:DTAG xmlns:NS=http://bucherautomation.com/ssi/jetcontrol/
```

With "NS" representing the "name space. The namespace is a character string with a maximum length of 63 characters.

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

In the following examples, the name space used is JC.

4.1.2 Inserting real-time controller values

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

Tag delimiters

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

Tag delimiters

Parameter	Description
Tag start	<JC:DTAG
Tag end	/>

Variable definition

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

name

Parameter	Description
Description	Variable name
Comment	Code letter followed by the variable number
Example	name="R1000023"

type

Parameter	Description
Description	Variable type of notation
Example	type="REAL"

format

Parameter	Description
Description	Representation format
Comment	Refer to format definition
Example	`format="+0####.###"`

factor

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

offset

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

Format definition

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

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

Registers/text registers

The variable name begins with a capital "R" followed by the register number.

The following types are possible:

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

Tab. 21: Registers/text registers

Example:

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

Result:

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

Flag

The variable name begins with a capital "F" followed by the flag number.

The following types are possible:

Type	Description
BOOL (Standard type)	Flag = 0 --> Display: 0 Flag = 1 --> Display: 1
STRING	Flag = 0 --> Display: FALSE Flag = 1 --> Display: TRUE

Example:

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

Result:

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

Inputs

The variable name begins with a capital "I" followed by the input number.

The following types are possible:

Type	Description
BOOL (Standard type)	Input = 0 --> Display: 0 Input = 1 --> Display: 1
STRING	Input = 0 --> Display: OFF Input = 1 --> Display: ON

Example:

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

Result:

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

Outputs

The variable name begins with a capital "O" followed by the output number.

The following types are possible:

Type	Description
BOOL (Standard type)	Output = 0 --> Display: 0 Output = 1 --> Display: 1
STRING	Output = 0 --> Display: OFF Output = 1 --> Display: ON

Example:

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

Result:

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

Access via pointer register

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

Examples:

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

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

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

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

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

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

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

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

Access via pointer register and offset

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

Examples:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4.1.3 Example of an HTML page

Task

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

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

Procedure

```
<JC:DTAG xmlns:JC="http://bucherautomation.com/ssi/jetcontrol" />
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<meta name="GENERATOR" content="Microsoft FrontPage 4.0">
<meta name="ProgID" content="FrontPage.Editor.Document">
<title>Index</title>
</head>
<body>
Hello World, 
<p>Actual controller values can be inserted into an html page like
this: </p>
<p>Register 201000 = <JC:DTAG name="R201000" type = INT format="+####" />,
or Hex: 0x<JC:DTAG name="R201000" type="INTX" format="0###" />,
or rather this way: <JC:DTAG name="R201000" type="BOOL" />, if only Boolean
is queried.
But binary is also possible: <JC:DTAG name="R201000" type="INTB"
format=######## />b. </p>
<p>Strings could also be defined "<JC:DTAG name="R201000" type="STRING" />
>". </p>
<p>A real number looks as follows: <JC:DTAG name="R1001500" type="REAL" />
or this way: <JC:DTAG name="R1001500" type="REAL" factor="1.3"
format="###.##" />. </p>
<p>The value of a flag is represented as follows: <JC:DTAG name="F10" />
or this way: <JC:DTAG name="F10" type="STRING" />. </p>
<p>For inputs/outputs, it is done the same way: <JC:DTAG name="I100000205"
type="BOOL" />
or <JC:DTAG name="I100000205" type="STRING" />. </p>
<p>R201000 = <JC:DTAG name="R201000" type="INT" format="+0##########" />
> </p>
```

```
<p>Regards </p>
<p>Yours JetControl</p>
</body>
</html>
```

Storage location

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

5 FTP client

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

Functions

The following functions are possible:

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

Requirements

To be able to use the FTP client feature basic knowledge of FTP connections and file systems is required.

– The IP address of the FTP server must be known.
– If the IP address of the FTP server is not known, name resolution through a DNS server must be possible.
– User name and password for logging on at the FTP server must be known.
– For programming this feature JetSym version 4.3 or higher is required.

Processing within the application program

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

5.1 Programming

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:

1. Initialize FTP client.
2. Open connections to FTP servers.
3. Transfer data.
4. Terminate connection.

Limitations

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

5.1.1 Initialize the FTP client

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

Function declaration

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

Return value

The following return value is possible:

0	Always

How to apply this function

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

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

Operating principle

The controller processes this function in the following steps:

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

5.1.2 Establishing a connection to the FTP server

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

Function declaration

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

Function parameters

Parameter	Description	Comment
ServerAddr	IP address or name	Name resolution by DNS server
UserName	User name for logon	Login
PassWord	Password for logon	Login

Tab. 22: Description of the functional parameters

Return value

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

Return value	Description
>0	A positive return value must be stored in a variable. It must be made available as a handle at activating the following functions: – Terminating a connection – Reading a file from the FTP server – Writing a file to the FTP server – Deleting a file from the FTP server – Change directory on the FTP server – Creating a directory on the FTP server – Deleting a directory from the FTP server – Determining the current directory on the FTP server
0	

Operating principle

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

This function is processed in the following steps:

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

5.1.3 Terminating a connection

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

Parameter	Description	Comment
FtpConnection	Handle	Return value of the FtpConnect() function

Tab. 23: Description of the functional parameters

Return value

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

5.1.4 Reading a file

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

Function declaration

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

Function parameters

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

Tab. 24: Description of the functional parameters

Return value

If the returned value is negative, an error has occurred. If the return value is 0, the controller has read the file and saved it locally

Return value	Description
0	No error
-1	Invalid handle
-2, -6	Error when storing the file locally
-3, -5, -7, -8	Communication error, there is, for example, no response from FTP server

Return value	Description
-4	Error message from FTP server, for example, file does not exist

Operating principle

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

– The controller must first read the file **ServerTestFile.txt**, for example, and save it in its local file system, for example, under the name **LocalTestFile.txt**.

– An error has occurred.

This function is processed in the following steps:

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

File names

– The function parameter for the local file may contain the path to this file, e.g. `'/Data/TestFiles/LocalTestFile.txt'`.

– If the file system of the remote network node supports this, the function parameter for the file located on the FTP server can also contain the path to this file. Otherwise, the directory must be set beforehand using the command `FtpDirChange()`.

– The file system supports both options.

5.1.5 Writing to a file

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

Function declaration

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

Function parameters

Parameter	Description	Comment
FtpConnection	Handle	Value returned by the function `FtpConnect ()`
ServerFile	File name	File name as which the FTP server is to save the written file

Parameter	Description	Comment
ClientFile	File name	Name of the file in the local file system, the content of which the controller is to send to the FTP server

Tab. 25: Description of the functional parameters

Return value

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

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

Operating principle

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

– For example, the controller must first read the **LocalTestFile.txt** file and write it to the remote network participant's file system, e.g., under the name **ServerTestFile.txt**.

– An error has occurred.

This function is processed in the following steps:

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

File names

– The function parameter for the local file may contain the path to this file, e.g. '/Data/TestFiles/LocalTestFile.txt'.

– If the file system of the remote network node supports this, the function parameter for the file located on the FTP server can also contain the path to this file. Otherwise, the directory must be set beforehand using the command FtpDirChange().

– The file system supports both options.

5.1.6 Deleting a file

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

Function declaration

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

Function parameters

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

Tab. 26: Description of the functional parameters

Return value

If the returned value is negative, an error has occurred. Bei einem Rückgabewert von 0 konnte die Datei im Dateisystem des fernen Netzwerkteilnehmers gelöscht werden.

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

Operating principle

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

– The FTP server must first delete the file **ServerTestFile.txt**. Please observe the following: The file name is only an example.

– An error has occurred.

This function is processed in the following steps:

Step	Description	
1	The controller sends the command to the FTP server to delete the file **ServerTestFile.txt**.	
2	The controller sends the contents of the file **LocalTestFile.txt**.	
3	Der FTP-Server löscht die Datei **ServerTestFile.txt**.	
4	**If ...**	**... then ...**
	... no errors have occurred,	... the file has been deleted successfully, the function is terminated and value **0** is returned.
	... errors have occurred,	... the function is terminated and a negative value is returned.

File names

– The function parameter for the local file may contain the path to this file, e.g. `"/Data/TestFiles/LocalTestFile.txt"`.

– If the file system of the remote network node supports this, the function parameter for the file located on the FTP server can also contain the path to this file. Otherwise, the directory must be set beforehand using the command `FtpDirChange()`.

– The file system supports both options.

5.1.7 Changing directories

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

Function declaration

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

Function parameters

Parameter	Description	Comment
FtpConnection	Handle	Value returned by the function `FtpConnect ()`
ServerFile	File name	Name of the directory into which the user wants to change

Tab. 27: Description of the functional parameters

Return value

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

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

Operating principle

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

– The FTP server must first change directories.

– An error has occurred.

This function is processed in the following steps:

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

Directory names

– If the file system of the remote network node supports this, the function parameter for the directory located on the FTP server can also contain the complete path including several subdirectories leading to this directory.

– Otherwise, you must first use the FtpDirChange() command to branch from one directory level to the next until you reach the desired directory.

– The file system supports both options.

5.1.8 Creating a directory

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

Function declaration

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

Function parameters

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

Tab. 28: Description of the functional parameters

Return value

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

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

Operating principle

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

– The FTP server must first create a subdirectory.

– An error has occurred.

This function is processed in the following steps:

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

Directory names

– If the file system of the remote network node supports this, the function parameter for the directory located on the FTP server can also contain the complete path including several subdirectories leading to this directory.

– If this feature is not supported, the user must navigate from one directory level to the next until the desired directory is reached. This is done using the command `FtpDirChange()`.

– The file system supports both options.

Restrictions regarding the file system of a JetControl

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

Example:

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

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

5.1.9 Deleting directories

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

Function declaration

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

Function parameters

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

Tab. 29: Description of the functional parameters

Return value

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

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

Operating principle

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

– The FTP server must first delete the subdirectory.

– An error has occurred.

This function is processed in the following steps:

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

Directory names

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

5.1.10 Determining the current directory

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

Function declaration

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

Function parameters

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

Tab. 30: Description of the functional parameters

Return value

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

Return value	Description
0	No error
-1	Invalid handle
-3	Communication error, there is, for example, no response from FTP server
-4	Fehlermeldung des FTP-Servers
-5	Invalid response from server

Operating principle

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

– The FTP server must first determine the currently set directory.
– An error has occurred.

This function is processed in the following steps:

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

5.2 Registers

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

5.2.1 Register numbers

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

Basic register number	Register numbers
320000	320000 ... 320101

Determining the register number

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

Register overview

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

Tab. 31: Overview of the module registers of the FTP client

5.2.2 Register description

Established connections are managed by the operating system of the controller in a list. The connection data for a connection is copied to registers MR 006 to MR 009 using module registers MR 004 or MR 005.

MR 000

Total of open connections

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

Module register properties

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

MR 002

Timeout

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

Module register properties

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

MR 003

Port number of FTP server

The value in this registry entry indicates the IP port number of the FTP server.

Module register properties

Values	0 ... 65,535
Value after reset	21

MR 004

Index in the connection table

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

Module register properties

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

MR 005

Connection handle

This register is for entering the connection handle. If a connection with this handle exists, you can then read the index of the connection table from module register MR 004 and the connection data from module registers MR 006 to MR 009.

Module register properties

Values	0 ... 2,147,483,647

MR 006

IP address of the FTP server

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

Module register properties

Access	Read
Takes effect	if MR 004 ≥ 0

MR 007

Port number of FTP server

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

Module register properties

Access	Read
Takes effect	if MR 004 ≥ 0

MR 008

IP address of FTP client

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

Module register properties

Access	Read
Takes effect	if MR 004 ≥ 0

MR 009

Port number of FTP client

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

Module register properties

Access	Read
Takes effect	if MR 004 ≥ 0

MR 100

Processing status on part of FTP client

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

Module register properties

Values	0	No access is performed
	1	Parameters are being handed over to the FTP client of the controller
	2	The FTP client communicates with the FTP server
	3	Access completed
Access	Read	

MR 101

Task ID

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

Module register properties

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

6 AutoCopy - Automatic copying of controller data

This chapter describes the AutoCopy function which allows to copy data within the controller and/or between the controller and an FTP server, the connected expansion modules and a controller within the network. To do this, create a command file that is stored together with the data, either on the SD card or the USB stick. This command file is automatically processed by the controller during the boot process.

Functions within the local file system

AutoCopy executes the following functions:

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

Functions within the file system of an FTP server

AutoCopy executes the following functions:

– Copying files from the FTP server
– Copying files to the FTP server
– Deleting files
– Changing directories
– Creating a directory
– Deleting directories

Areas of application

Basically, AutoCopy is used in the following scenarios:

– Remote maintenance is not possible.
– Where there is no PC on site.
– If the operator is not able or should not be allowed to make modifications to the plant.

The AutoCopy function lets you:

– Modify the application program
– Modify the application data
– Modify the controller configuration
– Operating system update (controller, modules on the system bus, network devices)

Requirements

For automatic copying of controller data, the following prerequisites must be fulfilled:

– The programmer must be familiar with the file system.
– The programmer must have basic knowledge in the area of FTP application.

config.ini – Example

This is an example of a configuration file **config.ini** with an entry *AutoCopyIni*.

```
[IP]
Address = 192.168. 10,209
SubnetMask = 255.255.255. 0
DefGateway = 0. 0. 0. 0
DNSServer = 192.168. 10,244

[HOSTNAME]
SuffixType = 0
Name = JetControl350

[PORTS]
JetIPBase = 50000
JVMDebug = 52000

[FILES]
AutoCopyIni = /SD/project_name/autocopy.ini
```

AutoCopyIni – Notes

– The AutoCopy function only makes sense if the data to be copied is stored on the SD card or USB stick. The root directory is then necessarily */SD/, /USB1/,* or */USB2/*, depending on the controller.
  – The assignment between the USB stick slot and the root directory is described in the USB interface section of the respective user manual.
  – If the controller only has one USB port, the root directory is */USB/*.
– The file **autocopy.ini** can be stored to any directory.
– Instead of **autocopy.ini**, you can name the file as desired.

Designation

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

6.1 Operating principle

This chapter describes how to launch and execute the AutoCopy function.

6.1.1 Launching the AutoCopy feature

The AutoCopy function can only be executed when the controller is booting (i.e. after startup).

Requirements

– You have created the command file and stored it to the respective directory.

If the entry AutoCopyIni is not available in the configuration file **config.ini** the name of the command file and of the directory is set as follows:

Parameter	Value	Description
File name	autocopy.ini	All lower case letters
Directory	/SD/	Root directory on the SD card
Directory	/USB/ or /USB1/	Root directory on the USB flash drive

– The file autocopy.ini can be stored to any directory.
– Instead of autocopy.ini, you can name the file as desired.

In this case, it is prerequisite that the configuration file **config.ini** contains the entry *config.ini*. This entry defines the directory and file name of the command file.

Launching the AutoCopy feature

To launch the AutoCopy function, proceed as follows:

1. Switch off the controller.
2. **If** you are using an **SD card**, insert the SD card completely into the SD card slot.
   If you are using a **USB flash drive**, insert the USB flash drive into the USB port on the controller.
3. Set the switch to the *LOAD* position.
4. Switch the controller on.
5. **If** JC-3xx, wait until the red LED **D1** is on and the green LED **R** and yellow LED **SD** are flashing slowly at approx. 1 Hz.

   If JC-4xx, wait until the red LED **D1** is on and the green LED **R** and yellow LED  are flashing slowly at approx. 1 Hz.
   If JC-94xMC, wait until the yellow LED **D1** is on and the green LED **RUN** flashes slowly at approx. 1 Hz.
   If JC-97xMC, wait until the yellow LED **GP2** is on and the green LED **GP1** flashes slowly at approx. 1 Hz.

   ⇨ The controller executes the AutoCopy function.
6. **If** JC-3xx, wait until the red LED **D1** and the green LED **R** flash slowly at approx. 1 Hz.
   If JC-4xx, wait until the red LED **D1** and the green LED **R** flash slowly at approx. 1 Hz.
   If JC-94xMC, wait until the yellow LED **D1** is on and the green LED **RUN** flashes rapidly at approx. 4 Hz.
   If JC-97xMC, wait until the yellow LED **GP2** is on and the green LED **GP1** flashes rapidly at approx. 4 Hz.

⇨ The AutoCopy process is completed.

6.1.2 Executing AutoCopy commands

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

Limitations

In AutoCopy mode the following restrictions of controller functions apply:

– The controller does not execute the application program.

– Communication with the controller is not possible.

– When the AutoCopy function is completed the controller must be restarted.

Executing AutoCopy commands

Step	Description
1	The controller opens the command file specified in the *AutoCopyIni* entry in the configuration file **/System/config.ini**.
2	The controller reads the values from section `[OPTIONS]`.
3	The controller reads the command and command parameters from the [COMMAND_1] section, processes them, and writes the result to the log file if necessary.
4 ... n	The controller processes the other commands in ascending order up to the number given in section `[OPTIONS]`.
n+1	The controller calculates the statistic values for all command results and writes them into the log file.

JC-3xx and JC-4xx LEDs in AutoCopy mode

Step	LEDs					State
	R	E	D1	D2	SD/ 	
1	☼ 4 Hz	☼ 4 Hz	☼ 4 Hz	☼ 4 Hz	○ OFF	Reset
2	☼ 1 Hz	○ OFF	○ OFF	● ON	○ OFF	The boot loader is running and is checking the OS. For controllers not having got a boot loader: The controller initializes the OS.
3	☼ 1 Hz	○ OFF	○ OFF	○ OFF	○ OFF	The OS reads the settings of the DIP switch on the backplane module and checks whether an Ethernet switch exists.
4	☼ 1 Hz	● ON	○ OFF	○ OFF	○ OFF	The OS initializes the realtime clock, the Ethernet port and the file system.
5	☼ 1 Hz	● ON	● ON	○ OFF	☼ / ○ OFF	The OS initializes the modules on the JX3 and JX2 system bus and the SD.
6	☼ 1 Hz	○ OFF	● ON	○ OFF	☼	The command file of the AutoCopy function is being processed.
7a	☼ 1 Hz	○ OFF	☼ 1 Hz	○ OFF	○ OFF	AutoCopy function is completed; no errors occurred.
7b	☼ 1 Hz	● ON	☼ 1 Hz	○ OFF	○ OFF	AutoCopy function is completed; errors occurred.

JC-94xMC LEDs in AutoCopy mode

Step	LEDs			State
	ERR	D1	RUN	
1	● ON	○ OFF	○ OFF	Reset
2	○ OFF	● ON	○ OFF	Initialization
3	○ OFF	● ON	☼ 1 Hz	The AutoCopy function is executed.
4 a	○ OFF	● ON	☼ 4 Hz	AutoCopy function is completed; no errors occurred.
4b	● ON	● ON	☼ 4 Hz	AutoCopy function is completed; errors occurred.

JC-97xMC LEDs in AutoCopy mode

Step	LEDs			State
	ERR	D1	RUN	
1	● ON	○ OFF	○ OFF	Reset
2	○ OFF	● ON	○ OFF	Initialization
3	○ OFF	● ON	☼ 1 Hz	The AutoCopy function is executed.
4 a	○ OFF	● ON	☼ 4 Hz	AutoCopy function is completed; no errors occurred.
4b	● ON	● ON	☼ 4 Hz	AutoCopy function is completed; errors occurred.

6.1.3 Terminating AutoCopy mode

Only a restart of the controller terminates the Autocopy mode.

Prerequisite

Processing the AutoCopy command is completed.

Terminating AutoCopy mode

To exit the AutoCopy mode, proceed as follows:

1. Switch off the controller.
2. The SD card or the USB flash drive can now be removed (not required).
3. Set the switch to RUN or STOP.
4. Switch the controller on.

⇨ The controller relaunches.

6.2 autocopy.ini - Structure

This chapter describes the structure of the autocopy.ini file 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 you can set values then used by the AutoCopy function.
– You can insert blank lines as required.
– Use the comment marks "!", "#" oder ";" to designate comment lines.

Sections

The command file has two section types:

– In the *[OPTIONS]* section, you can make default settings. This file is unique.
– In the *[COMMAND_#]* section, you can specify the commands that are to be executed. The number of command sections is limited to a value of 128.

6.2.1 Section [OPTIONS]

In the `[OPTIONS]` section, you can make default settings. This section exists only once, preferably at the beginning of the file.

Example

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

Elements of this section

The section consists of the following elements:

CommandCount

Property	Description
In the given example	14
Function	Number of command sections that follow
Valid values	≥ 0

Property	Description
Invalid values	< 0
BIf the value is invalid or the entry does not exist	0

LogFile

Property	Description
In the given example	/SD/autocopy.log
Function	Full name of the log file
Valid values	– All allowed file names – Directory exists
Invalid values	– Incorrect file name – Non-existent directory
If the value is invalid or the entry does not exist	The controller does not create a log file.

LogAppend

Property	Description
In the given example	1
Function	Defines whether a new log file is to be created or whether it is to be appended to an existing one.
Valid values	– 0 = Delete file which may exist and create a new one – 1 = append to existing file; if no file exists, the controller creates a new log file.
Invalid values	– < 0 – > 1
If the value is invalid or the entry does not exist	The controller re-creates the log file.

6.2.2 Command sections

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

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 names of the sections consist of the `COMMAND_` string followed by a value. The value is between one and the value of the `CommandCount` entry from 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 only contain one command. Thus, you have to create an individual section for each command.

Troubleshooting

When an error occurs while a command is being processed, the controller makes a corresponding entry in the log file. For each command, you can specify whether the controller should log the `Error` as an error or a `Warning` in the log file. Make this setting by the optional parameter `ErrorAsWarning`.

ErrorAsWarning	Entry into the 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"`.

– If the file system supports this, the function parameter for the file located on the FTP server can also contain the path to this file. Otherwise, the directory must be set beforehand using the command `FtpDirChange()`.

– The file system supports both options.

Available commands in the local file system

Command = DirCreate

Property	Description
Function	Creates a subdirectory
Parameter name	Path
Parameter value	Complete directory name
Valid values	– All valid directory names – Higher-level directories are available
Invalid values	– Invalid directory name – Non-existent higher-level directory – Name of an already existing directory
In the event of an invalid value	The controller does not generate the directory. It enters the error into the log file.

Example	`[COMMAND_1]`
	`Command = DirCreate`
	`Path = /sub1`
	`[COMMAND_2]`
	`Command = DirCreate`
	`Path = /sub1/sub2`

Command = DirRemove

Property	Description
Function	Removes a subdirectory
Parameter name	Path
Parameter value	Complete directory name
Valid values	– All valid directory names – The directory is empty
Invalid values	– Invalid directory name – Directory is not empty
In the event of an invalid value	The controller does not delete the directory. It enters the error into the log file.

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

Command = FileCopy

Property	Description
Function	Copying a file
Parameter name 1	Source
Parameter value 1	Full name of the source file
Parameter name 2	Destination
Parameter value 2	Full name of the destination file

Property	Description
Valid values	– All allowed file names – The destination directory does exist
Invalid values	– Incorrect file name – Non-existent source file – Non-existent destination directory
In the event of an invalid value	The controller does not copy the file. It enters the error 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

Property	Description
Function	Deleting a file
Parameter name	Path
Parameter value	Full name of the file
Valid values	All allowed file names
Invalid values	Incorrect file name
In the event of an invalid value	The controller does not delete the file. It enters the error into the log file.

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

Command = DaFileRead

Property	Description
Function	This command is for transferring register values and flag states from a data file to the controller
Parameter name	DaFile
Parameter value	Full name of the data file
Valid values	All allowed file names for data files
Invalid values	– Incorrect file name – Nonexistent data file
In the event of an invalid value	The data is not transferred to the control system. It enters the error into the log file.

Property	Description

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

Command = DaFileWrite

Property	Description
Function	This command is for storing register values and flag states to a data file
Parameter name 1	DaFile
Parameter value 1	Full name of the data file
Valid values	– All allowed file names for data files – The destination directory does exist
Invalid values	– Incorrect file name – Non-existent destination directory
In the event of an invalid value	The controller does not generate the data file. It enters the error 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
Valid values	– 0 = Delete the data file which may exist and create a new data file – 1 = Append the file to an existing one. If no file exists, the controller creates a new data file
Invalid values	– < 0 – > 1
In the event of an invalid value	A new data file will be created.
Parameter name 3	Type
Parameter value 3	Defines whether registers or flags are to be stored
Valid values	– Registers – Flag
Invalid values	Values other than *Register* or *Flag*
In the event of an invalid value	The controller does not generate the data file. It enters the error into the log file.
Parameter name 4	First
Parameter value 4	Number of the first register or flag
Valid values	All valid numbers from the memory area of the corresponding controller
Invalid values	Invalid numbers
In the event of an invalid value	The controller does not generate the data file. It enters the error into the log file.
Parameter name 5	Last

Property	Description
Parameter value 5	Number of the last register or flag
Valid values	All valid numbers from the memory area of the respective controller that are greater than or equal to the value at *First*
Invalid values	– Invalid numbers – Numbers less than *First*
In the event of an invalid value	The controller stores only one value (First).

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

Command = FtpConnect

Property	Description
Function	Establishing a connection to an FTP server
Parameter name 1	ServerAddr
Parameter value 1	IP address or name of FTP server
Valid values	– Reachable IP address – Name which can be resolved through DNS
Invalid values	– Unreachable IP address – Name which cannot be resolved

Property	Description
In the event of an invalid value	The controller does not generate the data file. It enters the error into the log file.
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 event of an invalid value	The controller does not establish the connection. It enters the error into the log file.

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

Limitation	Only one connection with an FTP server can be established at a time. The controller terminates the existing connection, before a connection to another FTP server is established.

Command = FtpFileRead

Property	Description
Function	Copying file from FTP server into the local file system
Parameter name 1	ServerFile
Parameter value 1	Full name of the source file in the FTP server
Parameter name 2	ClientFile
Parameter value 2	Full name of the destination file in the local file system
Valid values	– All allowed file names – The destination directory does exist
Invalid values	– Incorrect file name – Non-existent source file – Non-existent destination directory
In the event of an invalid value	The controller does not copy the file. It enters the error into the log file.

Example	`[COMMAND_8]`
	`Command = FtpFileRead`
	`ServerFile = /app/cantest/cantest.es3`
	`ClientFile = /SD/cantest3.es`

Command = FtpFileWrite

Property	Description
Function	Copying the file from the local file system into the file system of the FTP server
Parameter name 1	ServerFile
Parameter value 1	Full name of the destination file in the FTP server
Parameter name 2	ClientFile
Parameter value 2	Full name of the source file in the local file system
Valid values	– All allowed file names – The destination directory does exist
Invalid values	– Incorrect file name – Non-existent source file – Non-existent destination directory
In the event of an invalid value	The controller does not copy the file. It enters the error 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

Property	Description
Function	Deleting a file from the FTP server
Parameter name	ServerFile
Parameter value	Complete filename
Valid values	– All allowed file names
Invalid values	– Incorrect file name
In the event of an invalid value	The controller does not delete the file. It enters the error into the log file.

Example	`[COMMAND_9]`
	`Command = FtpFileRemove`
	`ServerFile = /sub1/Manual.pdf`

Command = FtpDirChange

Property	Description
Function	Changing the working directory in FTP server
Parameter name	ServerDir
Parameter value	Complete directory name
Valid values	– All valid directory names
Invalid values	– Invalid directory name

Property	Description
In the event of an invalid value	The controller does not switch the directory. It enters the error into the log file.

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

Command = FtpDirCreate

Property	Description
Function	Creating a subdirectory in the FTP server
Parameter name	ServerDir
Parameter value	Complete directory name
Valid values	– All valid directory names – Higher-level directories are available
Invalid values	– Invalid directory name – Non-existent higher-level directory – Name of an already existing directory
In the event of an invalid value	The controller does not create the directory. It records the error in the log file.

Example	`[COMMAND_6]`
	`Command = FtpDirCreate`
	`ServerDir = /Data/MyTestData`

Limitation	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 multiple directories is not supported.

Command = FtpDirRemove

Property	Description
Function	Clear the subdirectory in the FTP server
Parameter name	ServerDir
Parameter value	Complete directory name
Valid values	– All valid directory names – The directory is empty
Invalid values	– Invalid directory name – The directory is not empty
In the event of an invalid value	The controller does not delete the directory. It enters the error into the log file.

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

6.2.3 Example of a command file

Task

The JetControl 340 control system serves as an example here. It controls an existing plant via various JX3-I/O modules. In this plant, you want to enhance the functions.

To this end, the following modifications are required:

– 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

1. You copy the required files to an SD card and create a command file for the AutoCopy function.
2. Then you send this SD card along with a short instruction sheet to the plant operator. Once the update is completed, the operator is to return the SD card.

Configuration example

This example is based on the following configuration:

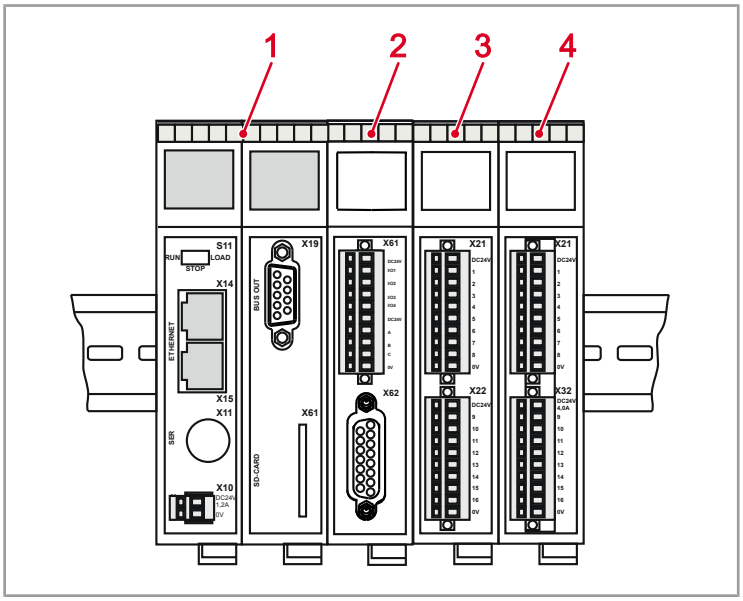


Fig. 1: Configuration example

1	JC-340 Controller
2	JX3-AO4 Analog Output Module Module Number: 2
3	JX3-DI16 Digital input module
4	JX3-DO16 Digital output modules

SD card contents

The following figure shows the directory structure and files on the SD card from the controller's perspective before the AutoCopy function is executed.

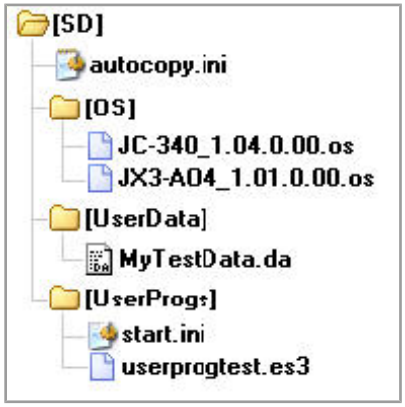


Fig. 2: Example of SD card contents

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
```

6.3 Log file

This chapter covers the structure and contents of the log file into which the device enters the outcome of the respective commands.

6.3.1 File contents

The log file is a plain text file. By making an entry into the command file, you define whether a log file is to be created or whether the device is to append the entries 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 executed 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, it contains short statistics on command processing.

In the above example an error occurs when trying to create the directory /app as this directory already exists. The device enters this error as a warning. When the device reads the DA file, an error also occurs. The device enters this error into the log file.

6.4 Data files

6.4.1 File format

Format

The data file consists of the following elements:

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

Data lines

A data line consists of the following elements:

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

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

Example

```
SD1001
; Data File – Bucher Automation 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
```

7 Save and load application program

This chapter covers the following topics:

– Where does JetSym store the application program in the device during download.

– From which location does the device's operating system load the application program during bootup.

The user can also specify which program is to be executed.

Required programmer's skills

This chapter requires knowledge on how to create application programs in JetSym and how to transmit them via the file system of the controller.

7.1 Application program - Default path

When you download the JetSym application program to the device, this program is saved as a file on the internal flash disk. The device enters the path and file name into the file **start.ini** which is in the folder **app**.

Path and file name

In the folder **app**, JetSym, by default, creates a subdirectory and assigns the project name to it. Then, JetSym stores the application program to this subdirectory assigning the extension ***.esX** to it. The path and file names are always converted into lower case letters. **X** is a hardware-dependent placeholder. e.g. **es4** for JetControl 400 and **es9** for JetControl 900.

start.ini – structure

The file is a plain text file with one section containing two entries.

Parameter	Description
[Startup]	Section name
Project	Path to the application program file relating to the folder **app**
Program	Name of the application program file

Example	`[Startup]`
	`Project = test_program`
	`Program = test_program.es9`

The application program is loaded from the file **test_program.es9** which is located in the folder **app** in subdirectory *test_program*.

7.2 Saving the application program to the SD memory card or the USB flash drive

When downloading the JetSym application program to the device, the default storage location is the internal flash disk.

– If you want the storage location to be the SD card or USB stick, you must configure the storage location.

– If you want the storage location to be saved in a different directory on the internal flash disk than the default, proceed in the same way.

Requirements

Only apply lower case for directory and file names.

Saving the application program to the SD memory card or the USB flash drive

If you want to save the application program on the SD card or USB stick, configure the device as follows:

1. Create an application program file by JetSym.
2. Create the desired directory on the SD card or the USB flash drive.
3. Store the application program file to the desired directory.
4. Enter the path to the application program file and the program name in the start.ini file in the app folder on the device's internal flash disk.

⇨ On re-boot, the device loads the application program from the SD card or USB flash drive.

start.ini - Structuree

The file is a plain text file with one section containing two entries.

Parameter	Description
[Startup]	Section name
Project	Path leading to the application program file
Program	Name of the application program file

Controller

Example - SD card:

```
[Startup]
Project = /sd/testprogram
Program = test1.esx
```

Example - USB flash drive:

```
[Startup]
Project = /usb/testprogram
Program = test1.esx
```

HMI

Example - SD card:

```
[Startup]
Project = /sd/testprogram
Program = test1.esx
```

Example - USB flash drive:

```
[Startup]
Project = /usb/testprogram
Program = test1.esx
```

The HMI JetView

Example - SD card:

```
[Startup]
Project = \..\..\..\Storage Card
Program = test1.esx
```

Example - USB flash drive:

```
[Startup]
Project = \..\..\..\USBMemory
Program = test1.esx
```

Result

The application program is loaded from the file **test1.esx** located in the subdirectory **testprogram** of folder *sd* on the SD card or in the folder *usb* on the USB flash drive.

7.3 Loading an application program

At reboot of the application program via JetSym or after booting the device, the application program is loaded and executed via the file system.

Loading the application program by the OS of the controller

For this, mode selector S11 must be in *RUN* position.

The application program is loaded by the controller's OS as follows:

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

Loading the application program by the OS of the HMI

The application program is loaded by the OS of the HMI as follows:

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

List of figures

Fig. 1 Configuration example 77
Fig. 2 Example of SD card contents 78

List of tables

Tab. 1 system directories ... 6
Tab. 2 control devices ... 6
Tab. 3 Properties ... 11
Tab. 4 Files ... 12
Tab. 5 Terms ... 12
Tab. 6 Predefined keys ... 14
Tab. 7 Comparison of user rights ... 17
Tab. 8 Log file ... 19
Tab. 9 Elements of the info file ... 22
Tab. 10 States of the sectors ... 22
Tab. 11 Parameter FileName ... 34
Tab. 12 Return value ... 34
Tab. 13 Register overview ... 35
Tab. 14 Web status R 202930 ... 36
Tab. 15 IP address of the SMTP server R 292932 ... 36
Tab. 16 IP address of the POP3 server R 292933 ... 37
Tab. 17 IP address of the POP3 server R 292934 ... 37
Tab. 18 Port number of the POP3 server R 292935 ... 37
Tab. 19 Status of email processing R 292937 ... 37
Tab. 20 Task-ID (E-Mail) R 292938 ... 38
Tab. 21 Registers/text registers ... 43
Tab. 22 Description of the functional parameters ... 49
Tab. 23 Description of the functional parameters ... 51
Tab. 24 Description of the functional parameters ... 51
Tab. 25 Description of the functional parameters ... 52
Tab. 26 Description of the functional parameters ... 54
Tab. 27 Description of the functional parameters ... 55
Tab. 28 Description of the functional parameters ... 56
Tab. 29 Description of the functional parameters ... 57
Tab. 30 Description of the functional parameters ... 58
Tab. 31 Overview of the module registers of the FTP client ... 59

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

www.bucherautomation.com