SoMachine
FtpRemoteFileHandling Library Guide

06/2017

EIO0000002405.01

www.schneider-electric.com

Schneider Electric

The information provided in this documentation contains general descriptions and/or technical characteristics of the performance of the products contained herein. This documentation is not intended as a substitute for and is not to be used for determining suitability or reliability of these products for specific user applications. It is the duty of any such user or integrator to perform the appropriate and complete risk analysis, evaluation and testing of the products with respect to the relevant specific application or use thereof. Neither Schneider Electric nor any of its affiliates or subsidiaries shall be responsible or liable for misuse of the information contained herein. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us.

No part of this document may be reproduced in any form or by any means, electronic or mechanical, including photocopying, without express written permission of Schneider Electric.

All pertinent state, regional, and local safety regulations must be observed when installing and using this product. For reasons of safety and to help ensure compliance with documented system data, only the manufacturer should perform repairs to components.

When devices are used for applications with technical safety requirements, the relevant instructions must be followed.

Failure to use Schneider Electric software or approved software with our hardware products may result in injury, harm, or improper operating results.

Failure to observe this information can result in injury or equipment damage.

Table of Contents

Safety Information

Important Information

NOTICE

Read these instructions carefully, and look at the equipment to become familiar with the device before trying to install, operate, service, or maintain it. The following special messages may appear throughout this documentation or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

The addition of this symbol to a “Danger” or “Warning” safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.

This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

⚠ DANGER

DANGER indicates a hazardous situation which, if not avoided, **will result in** death or serious injury.

⚠ WARNING

WARNING indicates a hazardous situation which, if not avoided, **could result in** death or serious injury.

⚠ CAUTION

CAUTION indicates a hazardous situation which, if not avoided, **could result** in minor or moderate injury.

NOTICE

NOTICE is used to address practices not related to physical injury.

PLEASE NOTE

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

A qualified person is one who has skills and knowledge related to the construction and operation of electrical equipment and its installation, and has received safety training to recognize and avoid the hazards involved.

BEFORE YOU BEGIN

Do not use this product on machinery lacking effective point-of-operation guarding. Lack of effective point-of-operation guarding on a machine can result in serious injury to the operator of that machine.

⚠ WARNING

UNGUARDED EQUIPMENT

- Do not use this software and related automation equipment on equipment which does not have point-of-operation protection.
- Do not reach into machinery during operation.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

This automation equipment and related software is used to control a variety of industrial processes. The type or model of automation equipment suitable for each application will vary depending on factors such as the control function required, degree of protection required, production methods, unusual conditions, government regulations, etc. In some applications, more than one processor may be required, as when backup redundancy is needed.

Only you, the user, machine builder or system integrator can be aware of all the conditions and factors present during setup, operation, and maintenance of the machine and, therefore, can determine the automation equipment and the related safeties and interlocks which can be properly used. When selecting automation and control equipment and related software for a particular application, you should refer to the applicable local and national standards and regulations. The National Safety Council's Accident Prevention Manual (nationally recognized in the United States of America) also provides much useful information.

In some applications, such as packaging machinery, additional operator protection such as point-of-operation guarding must be provided. This is necessary if the operator's hands and other parts of the body are free to enter the pinch points or other hazardous areas and serious injury can occur. Software products alone cannot protect an operator from injury. For this reason the software cannot be substituted for or take the place of point-of-operation protection.

Ensure that appropriate safeties and mechanical/electrical interlocks related to point-of-operation protection have been installed and are operational before placing the equipment into service. All interlocks and safeties related to point-of-operation protection must be coordinated with the related automation equipment and software programming.

NOTE: Coordination of safeties and mechanical/electrical interlocks for point-of-operation protection is outside the scope of the Function Block Library, System User Guide, or other implementation referenced in this documentation.

START-UP AND TEST

Before using electrical control and automation equipment for regular operation after installation, the system should be given a start-up test by qualified personnel to verify correct operation of the equipment. It is important that arrangements for such a check be made and that enough time is allowed to perform complete and satisfactory testing.

⚠ WARNING

EQUIPMENT OPERATION HAZARD

- Verify that all installation and set up procedures have been completed.
- Before operational tests are performed, remove all blocks or other temporary holding means used for shipment from all component devices.
- Remove tools, meters, and debris from equipment.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Follow all start-up tests recommended in the equipment documentation. Store all equipment documentation for future references.

Software testing must be done in both simulated and real environments.

Verify that the completed system is free from all short circuits and temporary grounds that are not installed according to local regulations (according to the National Electrical Code in the U.S.A, for instance). If high-potential voltage testing is necessary, follow recommendations in equipment documentation to prevent accidental equipment damage.

Before energizing equipment:

- Remove tools, meters, and debris from equipment.
- Close the equipment enclosure door.
- Remove all temporary grounds from incoming power lines.
- Perform all start-up tests recommended by the manufacturer.

OPERATION AND ADJUSTMENTS

The following precautions are from the NEMA Standards Publication ICS 7.1-1995 (English version prevails):

- Regardless of the care exercised in the design and manufacture of equipment or in the selection and ratings of components, there are hazards that can be encountered if such equipment is improperly operated.
- It is sometimes possible to misadjust the equipment and thus produce unsatisfactory or unsafe operation. Always use the manufacturer's instructions as a guide for functional adjustments. Personnel who have access to these adjustments should be familiar with the equipment manufacturer's instructions and the machinery used with the electrical equipment.
- Only those operational adjustments actually required by the operator should be accessible to the operator. Access to other controls should be restricted to prevent unauthorized changes in operating characteristics.

About the Book

At a Glance

Document Scope

This document describes the library FtpRemoteFileHandling.

The library offers FTP (File Transfer Protocol) client functionalities to a controller to access and handle files remotely from and to an FTP server.

The FtpRemoteFileHandling library uses system functions and resources which are supported on specific controller platforms:

- Modicon M241 Logic Controller
- Modicon M251 Logic Controller
- Modicon M258 Logic Controller
- Modicon LMC078 Motion Controller
- Modicon LMC058 Motion Controller

Validity Note

This document has been updated for the release of SoMachine V4.3.

The technical characteristics of the devices described in this document also appear online. To access this information online:

Step	Action
1	Go to the Schneider Electric home page *www.schneider-electric.com*.
2	In the **Search** box type the reference of a product or the name of a product range. ● Do not include blank spaces in the reference or product range. ● To get information on grouping similar modules, use asterisks (*).
3	If you entered a reference, go to the **Product Datasheets** search results and click on the reference that interests you. If you entered the name of a product range, go to the **Product Ranges** search results and click on the product range that interests you.
4	If more than one reference appears in the **Products** search results, click on the reference that interests you.
5	Depending on the size of your screen, you may need to scroll down to see the data sheet.
6	To save or print a data sheet as a .pdf file, click **Download XXX product datasheet**.

The characteristics that are presented in this manual should be the same as those characteristics that appear online. In line with our policy of constant improvement, we may revise content over time to improve clarity and accuracy. If you see a difference between the manual and online information, use the online information as your reference.

Related Documents

Document title	Reference
SoMachine Functions and Libraries User Guide	*EIO0000000735 (ENG)*; *EIO0000000792 (FRE)*; *EIO0000000793 (GER)*; *EIO0000000795 (SPA)*; *EIO0000000794 (ITA)*; *EIO0000000796 (CHS)*
SoMachine Programming Guide	*EIO0000000067 (ENG)*; *EIO0000000069 (FRE)*; *EIO0000000068 (GER)*; *EIO0000000071 (SPA)*; *EIO0000000070 (ITA)*; *EIO0000000072 (CHS)*

You can download these technical publications and other technical information from our website at *http://www.schneider-electric.com/en/download*.

Part I
General Information

What Is in This Part?

This part contains the following chapters:

Chapter 1
Specific Safety Information

Overview

This section contains information regarding working with the FtpRemoteFileHandling library. Personnel working with the FtpRemoteFileHandling library must read and observe this information.

What Is in This Chapter?

This chapter contains the following topics:

Qualification of Personnel

Overview

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel.

No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

A qualified person is one who has skills and knowledge related to the construction and operation of electrical equipment and the installation, and has received safety-related training to recognize and avoid the hazards involved.

The qualified person must be able to detect possible hazards that may arise from parameterization, modifying parameter values and generally from mechanical, electrical, or electronic equipment. The qualified person must be familiar with the standards, provisions, and regulations for the prevention of industrial accidents, which they must observe when designing and implementing the system.

Proper Use

Overview

This product is a library to be used together with the control systems and servo amplifiers intended solely for the purposes as described in the present documentation as applied in the industrial sector.

Always observe the applicable safety-related instructions, the specified conditions, and the technical data.

Perform a risk evaluation concerning the specific use before using the product. Take protective measures according to the result.

Since the product is used as a part of an overall system, you must ensure the safety of the personnel by means of the concept of this overall system (for example, machine concept).

Any other use is not intended and may be hazardous. Electrical devices and equipment must only be installed, operated, maintained, and repaired by qualified personnel.

Product Related Information

Product Related Information

⚠ WARNING

LOSS OF CONTROL

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical control functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop and overtravel stop, power outage and restart.
- Separate or redundant control paths must be provided for critical control functions.
- System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.
- Observe all accident prevention regulations and local safety guidelines.[1]
- Each implementation of this equipment must be individually and thoroughly tested for proper operation before being placed into service.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

[1] For additional information, refer to NEMA ICS 1.1 (latest edition), "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control" and to NEMA ICS 7.1 (latest edition), "Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems" or their equivalent governing your particular location.

Before you attempt to provide a solution (machine or process) for a specific application using the POUs found in the library, you must consider, conduct and complete best practices. These practices include, but are not limited to, risk analysis, functional safety, component compatibility, testing and system validation as they relate to this library.

⚠ WARNING

IMPROPER USE OF POUS

- Perform a safety-related analysis for the application and the devices installed.
- Ensure that the POUs are compatible with the devices in the system and have no unintended effects on the proper functioning of the system.
- Use appropriate parameters, especially limit values, and observe machine wear and stop behavior.
- Verify that the sensors and actuators are compatible with the selected POUs.
- Thoroughly test all functions during verification and commissioning in all operation modes.
- Provide independent methods for critical control functions (emergency stop, conditions for limit values being exceeded, etc.) according to a safety-related analysis, respective rules, and regulations.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Only use software approved by Schneider Electric for use with this equipment.
- Update your application program every time you change the physical hardware configuration.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Update your application program as required, paying particular attention to I/O address adjustments, whenever you modify the hardware configuration.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Incomplete file transfers, such as data files, application files and/or firmware files, may have serious consequences for your machine or controller. If you remove power, or if there is a power outage or communication interruption during a file transfer, your machine may become inoperative, or your application may attempt to operate on a corrupted data file. If a an interruption occurs, reattempt the transfer. Be sure to include in your risk analysis the impact of corrupted data files.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION, DATA LOSS, OR FILE CORRUPTION

- Do not interrupt an ongoing data transfer.
- If the transfer is interrupted for any reason, re-initiate the transfer.
- Do not place your machine into service until the file transfer has completed successfully, unless you have accounted for corrupted files in your risk analysis and have taken appropriate steps to prevent any potentially serious consequences due to unsuccessful file transfers.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

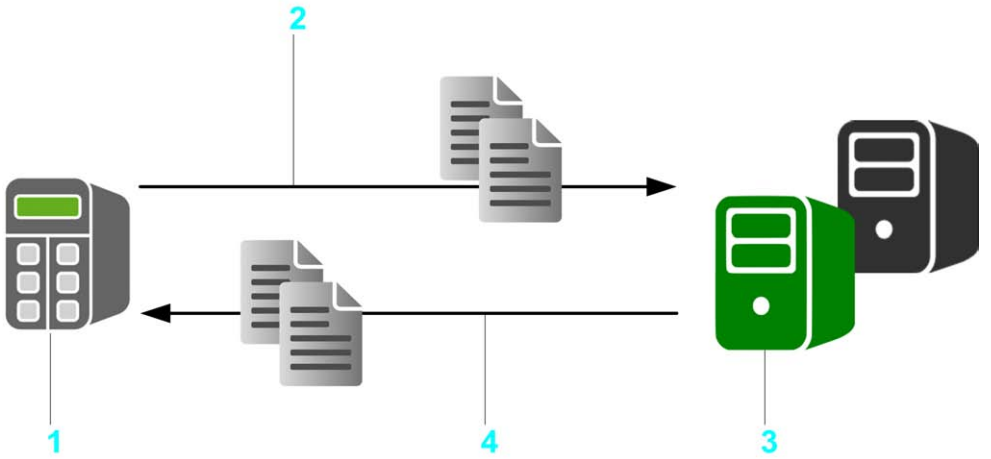
Chapter 2
Presentation of the Library

General Information

Introduction

The FtpRemoteFileHandling library provides the following FTP client functionalities for remote file handling:

- Reading files
- Writing files
- Deleting files
- Listing content of remote directories
- Adding directories
- Removing directories



1 Controller as FTP client
2 Upload file (`Store` command)
3 FTP server (on PC or controller)
4 Download file (`Retrieve` command)

The following logic controllers provide FTP client and FTP server functionalities:

- Modicon M241 Logic Controller
- Modicon M251 Logic Controller
- Modicon M258 Logic Controller

- Modicon LMC078 Motion Controller
- Modicon LMC058 Motion Controller

The following table indicates the characteristics of the library:

Characteristic	Value
Library title	FtpRemoteFileHandling
Company	Schneider Electric
Category	**Communication**
Component	**Internet protocol suite**
Default namespace	**SE_FTP**
Language model attribute	Qualified-access-only *(see SoMachine, Functions and Libraries User Guide)*
Forward compatible library	Yes (FCL *(see SoMachine, Functions and Libraries User Guide)*)

NOTE: For this library, qualified-access-only is set. This means that the POUs, data structures, enumerations, and constants have to be accessed using the namespace of the library. The default namespace of the library is **SE_FTP**.

General Considerations

Consider the following limitations for FTP data transfer:

- Only ASCII symbols are supported for file and directory names to be exchanged with the FTP server.
- Only IPv4 (Internet Protocol version 4) is supported.
- Only passive mode FTP is supported.
- Only one FTP connection is allowed at a time.
- Since the response time of the FTP server cannot be controlled, execute the function blocks in a low-priority, cyclic task. Adapt the watchdog function for this task to allow sufficient time for the connection. Alternatively, execute the function blocks in a **Freewheeling** task. For this type of task, no cycle time is defined.

The library described in this document internally uses the TcpUdpCommunication library.

The TcpUdpCommunication (Schneider Electric) and the CAA Net Base Services library (CAA Technical Workgroup) use the same system resources on the controller. The simultaneous use of both libraries in the same application may lead to disturbances during the operation of the controller.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Do not use the library TcpUdpCommunication (Schneider Electric) together with the library CAA Net Base Services (CAA Technical Workgroup) simultaneously in the same application.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Considerations Concerning Cyber Security

The FtpRemoteFileHandling library functions do not support secure connections using TLS (Transport Layer Security) or SSL (Secure Socket Layer). Therefore, communication must only be performed inside your industrial network, isolated from other networks inside your company, and protected from the Internet.

NOTE: Schneider Electric adheres to industry best practices in the development and implementation of control systems. This includes a "Defense-in-Depth" approach to secure an Industrial Control System. This approach places the controllers behind one or more firewalls to restrict access to authorized personnel and protocols only.

⚠ WARNING

UNAUTHENTICATED ACCESS AND SUBSEQUENT UNAUTHORIZED MACHINE OPERATION

- Evaluate whether your environment or your machines are connected to your critical infrastructure and, if so, take appropriate steps in terms of prevention, based on Defense-in-Depth, before connecting the automation system to any network.
- Limit the number of devices connected to a network to the minimum necessary.
- Isolate your industrial network from other networks inside your company.
- Protect any network against unintended access by using firewalls, VPN, or other, proven security measures.
- Monitor activities within your systems.
- Prevent subject devices from direct access or direct link by unauthorized parties or unauthenticated actions.
- Prepare a recovery plan including backup of your system and process information.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Overview of the POUs

Function block	Use
`FB_FTPClient` *(see page 43)*	Implements an FTP client, which connects to an FTP server for operations on files and directories.

Overview of the Structures in the Module-Specific Interface

Structure	Use
`ST_Credentials` *(see page 30)*	Contains the user-specific information for connecting to an external FTP server.
`ST_Content` *(see page 31)*	Contains the user information related to the selected directory.
`ST_Element` *(see page 33)*	Substructure of `ST_Content` that contains element-specific information.

Overview of the Enumerations

Enumeration	Use
`ET_FtpCommand` *(see page 26)*	Defines the commands which can be executed by the function block `FB_FTPClient` via `i_etCommand`.
`ET_Result` *(see page 27)*	Contains the possible values that indicate the result of operations executed by the function block.

Part II
Enumerations and Structures

What Is in This Part?

This part contains the following chapters:

Chapter 3
Enumerations

What Is in This Chapter?

This chapter contains the following topics:

ET_FtpCommand

Overview

Type:	Enumeration
Available as of:	V1.0.4.0

Description

The enumeration ET_FtpCommand defines the commands which can be executed by the FB_FtpClient function block *(see page 43)* via i_etCommand.

The command names are similar to the FTP commands defined in IETF RFC 959.

NOTE: Commands to handle a login are not included in the enumeration ET_FtpCommand. The user credentials (Username, Password) are used to establish a connection to the specified host automatically after the function block has been enabled. Port 21 is the default, monitored port for the FTP server. In order to modify these credentials, disable the function and re-enable it with the new information. For more information, refer to ST_Credentials *(see page 30)*.

Enumeration Elements

Name	Data type	Value	Description
NoCommand	UINT	0	Default value: initial state.
ChangeWorkingDirectory	UINT	1	Change the working directory.
Retrieve	UINT	2	Download file from the external FTP server to a specified local directory.
Store	UINT	3	Upload file to the external FTP server.
Rename	UINT	4	Rename a file or a directory on the external FTP server.
Delete	UINT	5	Delete a file on the external FTP server.
RemoveDirectory	UINT	6	Delete a directory on the external FTP server. **NOTE:** The directory on the external FTP server must be empty.
MakeDirectory	UINT	7	Create a new directory on the external FTP server.
List	UINT	8	List the content of the selected directory.

Used By

- FB_FtpClient

ET_Result

Overview

Type:	Enumeration
Available as of:	V1.0.4.0

Description

The enumeration ET_Result contains the possible values that indicate the result of operations executed by the function block.

Enumeration Elements

Name	Data type	Value	Description
If q_xError of a function block *(see page 43)* is FALSE, one of the following status messages is shown.			
Disabled	UDINT	0	The function block is disabled.
OperationSuccessful	UDINT	1	The last FTP command has been completed successfully.
NoCommandToExecute	UDINT	2	No command to execute.
ProcessingCommand	UDINT	3	The function block is executing an FTP command.
Initializing	UDINT	4	The function block is initializing.
Operational	UDINT	5	The function block is operational.
Disabling	UDINT	6	The function block is disabling.
If q_xError of a function block *(see page 43)* is TRUE, one of the following status messages is shown.			
UnexpectedProgramBehavior	UDINT	100	An internal error has been detected.
InvalidIpAddress	UDINT	101	The given IP address is not valid. Only IPv4 addresses are allowed.
UnsupportedFtpCommand	UDINT	102	The given FTP command is not supported. Use the enumeration ET_FtpCommand *(see page 26)*.
UnableToMakeDirectory	UDINT	103	Unable to create directory on the external FTP server. Verify whether the directory already exists.

Name	Data type	Value	Description
UnableToRemoveDirectory	UDINT	104	Unable to remove directory on the external FTP server. Verify whether the directory is empty since only empty directories can be deleted. Verify whether the directory has already been deleted.
UnableToChangeDirectory	UDINT	105	Unable to change working directory on the external FTP server. Verify whether the destination directory exists.
UnableToEstablishConnection	UDINT	106	Unable to establish an FTP connection. Verify whether the FTP server is reachable.
UnableToLogin	UDINT	107	Unable to log in to the external FTP server. Verify whether the credentials are correct.
UnableToSetTransferMode	UDINT	108	Unable to set the transfer mode for the file transfer (TYPE I). Verify whether the FTP server is capable of binary data transfer.
UnableToEnterPassiveMode	UDINT	109	Unable to enter passive mode (PASV). Verify whether the FTP server has been configured accordingly.
UnableToPrintWorkingDirectory	UDINT	110	Unable to display the working directory of the external FTP server.
UnableToInitiateFileTransfer	UDINT	111	Unable to perform the file transfer. Verify that i_sCommandArg1 and i_sCommandArg2 are not empty.
InconsistencyDuringFileTransfer	UDINT	112	An error has been detected during the file transfer. Verify whether the data has been corrupted.
UnableToListContent	UDINT	113	Unable to display the content of the directory on the external FTP server.
UnableToEstablishMutlipleConnections	UDINT	114	Unable to establish more FTP connections. Only one instance of FB_FtpClient is allowed to be active at a time.
UnableToDeleteFile	UDINT	115	Unable to delete file from external FTP server.
UnableToRenameElement	UDINT	116	Unable to rename element on external FTP server.

Used By

- FB_FtpClient

Chapter 4
Structures

What Is in This Chapter?

This chapter contains the following topics:

ST_Credentials

Overview

Type:	Structure
Available as of:	V1.0.4.0
Inherits from:	–

Description

The structure ST_Credentials contains the user-specific information for connecting to an external FTP server.

NOTE: Commands to handle a login are not included in the enumeration ET_FtpCommand. The user credentials (Username, Password) are used to establish a connection to the specified host automatically after the function block has been enabled. Port 21 is the default, monitored port for the FTP server. In order to modify these credentials, disable the function and re-enable it with the new information. For more information, refer to ST_Credentials *(see page 30)*.

Structure Elements

Name	Data type	Description
i_sHostIp	STRING [15]	The IP address of the external FTP server.
i_sUsername	STRING [255]	The username to access the external FTP server.
i_sPassword	STRING [255]	The password to access the external FTP server.
i_sAccount	STRING [255]	The account to access the external FTP server. This parameter is not mandatory for all FTP servers.

Used By

- FB_FtpClient

ST_Content

Overview

Type:	Structure
Available as of:	V1.0.4.0
Inherits from:	–

Description

The structure ST_Content contains the user information related to the selected directory.

Structure Elements

Name	Data type	Description
i_sCommandArg1	STRING [255]	In case an FTP command requires an input argument, it has to be transferred using this variable. For usage, refer to the next table.
i_sCommandArg2	STRING [255]	In case an FTP command requires an additional / second input argument, it has to be transferred using this variable. For usage, refer to the next table.
q_sCurrentDirectory	STRING [255]	Name of the working directory.
q_astElements	ARRAY[0..Gc_udiMaxNumberOfListEntries] OF ST_Element	Lists directory content and shows element-specific information.
q_udiAvailableElements	UDINT	Number of elements in the directory.

Detailed information about the usage of `i_sCommandArg1` and `i_sCommandArg2`:

Command	`i_sCommandArg1`	`i_sCommandArg2`
`ChangeWorkingDirectory`	Name of the directory on the external FTP server.	–
`Retrieve`	Name of the file on the external FTP server.	Name of the file in the file system of the controller.
`Store`	Name of the file in the file system of the controller.	Name of the file on the external FTP server.
`Rename`	Present name on the external FTP server.	New name on the external FTP server.
`Delete`	Name of the file on the external FTP server.	–
`RemoveDirectory`	Name of the directory on the external FTP server.	–
`MakeDirectory`	Name of the directory on the external FTP server.	–
`List`	–	–

NOTE: The `Retrieve` and `Store` commands overwrite existing elements if they have an identical name.

Programming Example for Copying a File

To copy the file *task.txt* from the directory */jobs* on the external FTP server to the local root directory without modifying the file name, do the following before sending a positive edge via `i_xExecute`:

```
i_sCommandArg1 := 'jobs/task.txt';
i_sCommandArg2 := 'task.txt';
```

Used By

- `FB_FtpClient`

ST_Element

Overview

Type:	Structure
Available as of:	V1.0.4.0
Inherits from:	–

Description

The structure ST_Element is part of ST_Content. It contains specific information for each element in the selected external FTP server directory after ET_FtpCommand.List has been executed.

Structure Elements

Name	Data type	Description
sName	STRING [255]	Name of the element.
xIsDirectory	BOOL	TRUE: Element is a directory. FALSE: Element is a file.
udiSize	UDINT	File size in bytes.
sDate	STRING [255]	Last modification date.

Used By

- FB_FtpClient

Part III
Global Variables

What Is in This Part?

This part contains the following chapters:

Chapter 5
Global Constants List

Global Constants List (GCL)

Overview

Type:	Global constants
Available as of:	V1.0.4.0

Description

The global constants list contains the global constants of the FtpRemoteFileHandling library.

Global Constants

Variable	Data type	Value	Description
`Gc_sLibraryVersion`	STRING[80]	Vx.x.x.0[1]	Library version
[1] This value varies to indicate the version of the library.			

Chapter 6
Global Parameter List

Global Parameter List (GPL)

Overview

Type:	Global parameters
Available as of:	V1.0.4.0

Description

The global parameter list contains the global parameters of the FtpRemoteFileHandling library. They can be overwritten specifically for your project in the **Library Manager**.

Global Parameters

Variable	Data type	Value	Description
`Gc_udiMaxNumberOfListEntries`	UDINT	20	Maximum number of elements in a directory to be displayed.
`Gc_udiListBufferSize`	UDINT	2550	Maximum number of bytes to be buffered when calling `ET_FtpCommand.List`. It is a good practice to multiply the value of the parameter `Gc_udiMaxNumberOfListEntries` by 255 and use the resulting value for this variable.

Part IV
Program Organization Units (POU)

What Is in This Part?

This part contains the following chapters:

Chapter 7
Function Blocks

FB_FtpClient

Overview

Type:	Function block
Available as of:	V1.0.4.0

Task

The `FB_FtpClient` function block includes the related FTP functionalities for operations on files and directories. Each instance handles one FTP connection. In case you attempt to establish a second transfer, the function block responds with `ET_Result.UnableToEstablishMutlipleConnections`.

Functional Description

The `FB_FtpClient` function block is the user-interface to interact with the external FTP server.

After the function block has been enabled, an FTP connection is established using the user credentials that are submitted using `iq_stCredentials`. As soon as the connection has been established, the function block is capable of processing commands submitted by `i_etCommand` and a rising edge detected at `i_xExecute`.

As long as commands are executed, the output `q_xBusy` is set to TRUE. After a command has been successfully completed, `q_xDone` is set to TRUE.

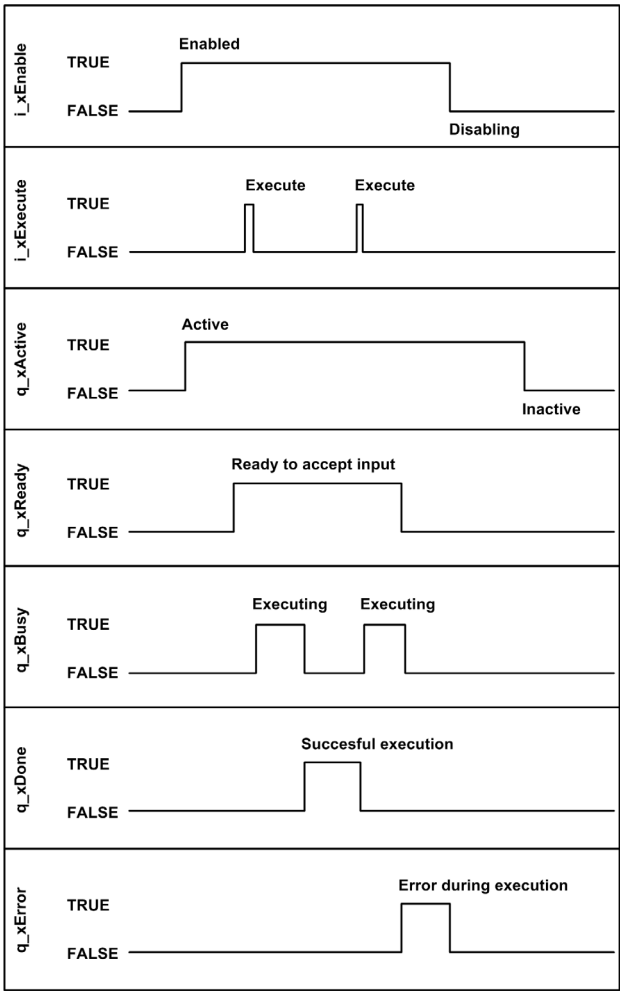
Status messages and diagnostic information are provided using the outputs `q_xError` (TRUE if an error has been detected), `q_etResult` and `q_etResultMsg`.

To acknowledge detected errors, disable and re-enable the function block to be able to send a command.

When disabling the function block (`i_xEnable` = FALSE), it must be called as long as `q_xActive` = TRUE in order to complete the internal cleanup routines. Afterwards it can be re-enabled.

If a timeout is exceeded after the connection has been established, the execution of the next FTP command (`ET_FtpCommand`) is detected as an error. To avoid this behavior, enable the function block just before performing the related operations and disable it afterwards.

The diagram shows the signal behavior of the inputs and outputs of the function block:



Interface

Input	Data type	Description
i_xEnable	BOOL	Activation and initialization of the function block.
i_xExecute	BOOL	The command specified with the input i_etCommand is executed upon rising edge of this input.
i_etCommand	ET_FtpCommand	The FTP command that is executed if the input i_xExecute is TRUE.

Input/Output	Data type	Description
iq_stCredentials	ST_Credentials	Used to pass the structure containing user settings, such as user name or password.
iq_stContent	ST_Content	Used to pass the working directory and, if applicable, the amount and names of files in this directory.

Output	Data type	Description
q_xActive	BOOL	If the function block is active, the output is set to TRUE.
q_xReady	BOOL	If the initialization is successful, the output is set to TRUE as long as the function block is capable of accepting inputs.
q_xBusy	BOOL	If this output is set to TRUE, the function block executes the command specified at i_etCommand.
q_xDone	BOOL	If this output is set to TRUE, the function block has successfully completed the command specified at i_etCommand.
q_xError	BOOL	If this output is set to TRUE, an error has been detected. For details, refer to q_etResult and q_etResultMsg.
q_etResult	ET_Result	Provides diagnostic and status information.
q_sResultMsg	STRING[255]	Provides additional diagnostic and status information.

Chapter 8
Functions

What Is in This Chapter?

This chapter contains the following topics:

FC_EtResultToString

Overview

Type:	Function
Available as of:	V1.0.1.0
Inherits from:	–
Implements:	–

Task

Convert an enumeration element of type ET_Result to a variable of type STRING.

Functional Description

Using the function `FC_EtResultToString`, you can convert an enumeration element of type ET_Result to a variable of type STRING.

Interface

Input	Data type	Description
`i_etResult`	ET_Result	Enumeration with the result.

Return Value

Data type	Description
STRING(80)	The ET_Result converted to text.

FC_EtFtpCommandToString

Overview

Type:	Function
Available as of:	V1.0.1.0
Inherits from:	–
Implements:	–

Task

Convert an enumeration element of type `ET_FtpCommand` to a string value containing the FTP command.

Functional Description

Using the function `FC_EtFtpCommandToString`, you can convert an enumeration element of type `ET_FtpCommand` to a string value.

Interface

Input	Data type	Description
`i_etFtpCommand`	`ET_FtpCommand`	Enumeration with the result.

Return Value

Data type	Description
STRING(80)	The `ET_FtpCommand` converted to a string value.

Appendices

Appendix A
Function and Function Block Representation

Overview

Each function can be represented in the following languages:

- IL: Instruction List
- ST: Structured Text
- LD: Ladder Diagram
- FBD: Function Block Diagram
- CFC: Continuous Function Chart

This chapter provides functions and function blocks representation examples and explains how to use them for IL and ST languages.

What Is in This Chapter?

This chapter contains the following topics:

Differences Between a Function and a Function Block

Function

A function:

- is a POU (Program Organization Unit) that returns one immediate result.
- is directly called with its name (not through an instance).
- has no persistent state from one call to the other.
- can be used as an operand in other expressions.

Examples: boolean operators (`AND`), calculations, conversion (`BYTE_TO_INT`)

Function Block

A function block:

- is a POU (Program Organization Unit) that returns one or more outputs.
- needs to be called by an instance (function block copy with dedicated name and variables).
- each instance has a persistent state (outputs and internal variables) from one call to the other from a function block or a program.

Examples: timers, counters

In the example, `Timer_ON` is an instance of the function block `TON`:

```
PROGRAM MyProgram_ST
VAR
    Timer_ON: TON;  // Function Block Instance
    Timer_RunCd: BOOL;
    Timer_PresetValue: TIME := T#5S;
    Timer_Output: BOOL;
    Timer_ElapsedTime: TIME;
END_VAR
```

```
Timer_ON(
    IN:=Timer_RunCd,
    PT:=Timer_PresetValue,
    Q=>Timer_Output,
    ET=>Timer_ElapsedTime);
```

How to Use a Function or a Function Block in IL Language

General Information

This part explains how to implement a function and a function block in IL language.

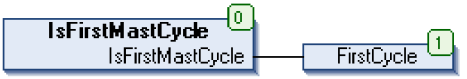
Functions `IsFirstMastCycle` and `SetRTCDrift` and Function Block `TON` are used as examples to show implementations.

Using a Function in IL Language

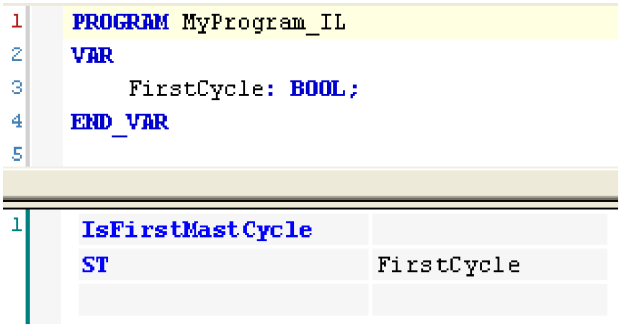
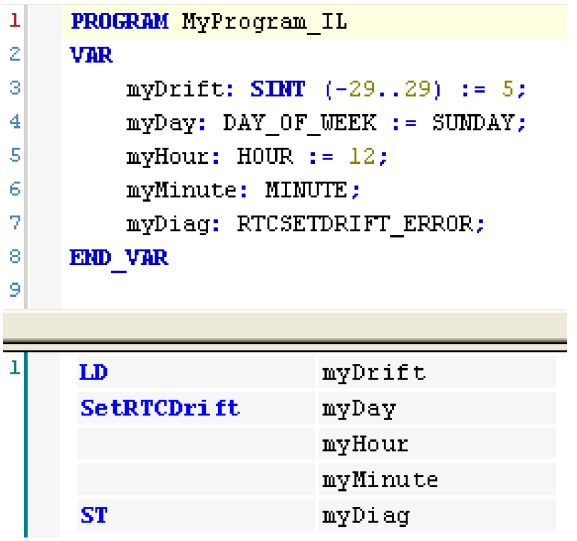
This procedure describes how to insert a function in IL language:

Step	Action
1	Open or create a new POU in Instruction List language. **NOTE:** The procedure to create a POU is not detailed here. For more information, refer to Adding and Calling POUs *(see SoMachine, Programming Guide)*.
2	Create the variables that the function requires.
3	If the function has 1 or more inputs, start loading the first input using LD instruction.
4	Insert a new line below and: ● type the name of the function in the operator column (left field), or ● use the **Input Assistant** to select the function (select **Insert Box** in the context menu).
5	If the function has more than 1 input and when Input Assistant is used, the necessary number of lines is automatically created with ??? in the fields on the right. Replace the ??? with the appropriate value or variable that corresponds to the order of inputs.
6	Insert a new line to store the result of the function into the appropriate variable: type ST instruction in the operator column (left field) and the variable name in the field on the right.

To illustrate the procedure, consider the Functions `IsFirstMastCycle` (without input parameter) and `SetRTCDrift` (with input parameters) graphically presented below:

Function	Graphical Representation
without input parameter: `IsFirstMastCycle`	
with input parameters: `SetRTCDrift`	

In IL language, the function name is used directly in the operator column:

Function	Representation in POU IL Editor
IL example of a function without input parameter: `IsFirstMastCycle`	
IL example of a function with input parameters: `SetRTCDrift`	

Using a Function Block in IL Language

This procedure describes how to insert a function block in IL language:

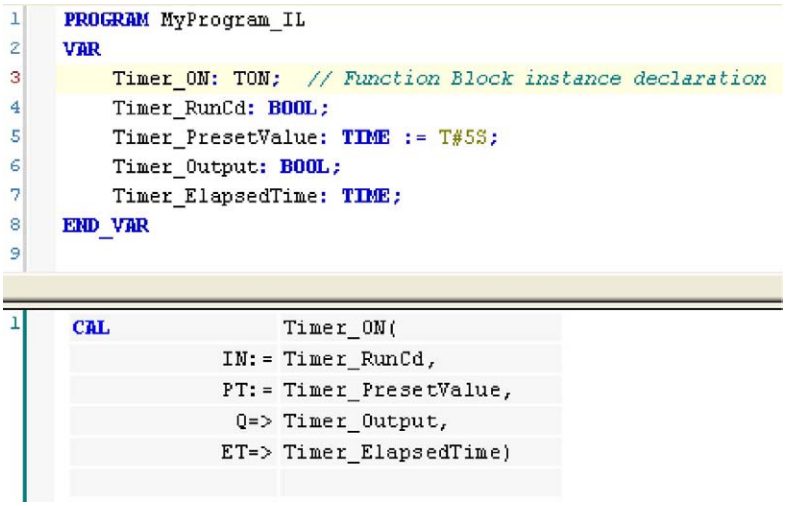
Step	Action
1	Open or create a new POU in Instruction List language. **NOTE:** The procedure to create a POU is not detailed here. For more information, refer to Adding and Calling POUs *(see SoMachine, Programming Guide)*.

Step	Action
2	Create the variables that the function block requires, including the instance name.
3	Function Blocks are called using a CAL instruction: ● Use the **Input Assistant** to select the FB (right-click and select **Insert Box** in the context menu). ● Automatically, the CAL instruction and the necessary I/O are created. Each parameter (I/O) is an instruction: ● Values to inputs are set by ":=". ● Values to outputs are set by "=>".
4	In the CAL right-side field, replace ??? with the instance name.
5	Replace other ??? with an appropriate variable or immediate value.

To illustrate the procedure, consider this example with the TON Function Block graphically presented below:

Function Block	Graphical Representation
TON	

In IL language, the function block name is used directly in the operator column:

Function Block	Representation in POU IL Editor
TON	

How to Use a Function or a Function Block in ST Language

General Information

This part explains how to implement a Function and a Function Block in ST language.

Function `SetRTCDrift` and Function Block `TON` are used as examples to show implementations.

Using a Function in ST Language

This procedure describes how to insert a function in ST language:

Step	Action
1	Open or create a new POU in Structured Text language. **NOTE:** The procedure to create a POU is not detailed here. For more information, refer to Adding and Calling POUs *(see SoMachine, Programming Guide)*.
2	Create the variables that the function requires.
3	Use the general syntax in the **POU ST Editor** for the ST language of a function. The general syntax is: `FunctionResult:= FunctionName(VarInput1, VarInput2,.. VarInputx);`

To illustrate the procedure, consider the function `SetRTCDrift` graphically presented below:

Function	Graphical Representation
`SetRTCDrift`	

The ST language of this function is the following:

Function	Representation in POU ST Editor

`SetRTCDrift`	`PROGRAM MyProgram_ST`
	`VAR myDrift: SINT(-29..29) := 5;`
	`myDay: DAY_OF_WEEK := SUNDAY;`
	`myHour: HOUR := 12;`
	`myMinute: MINUTE;`
	`myRTCAdjust: RTCDRIFT_ERROR;`
	`END_VAR`
	`myRTCAdjust:= SetRTCDrift(myDrift, myDay, myHour, myMinute);`

Using a Function Block in ST Language

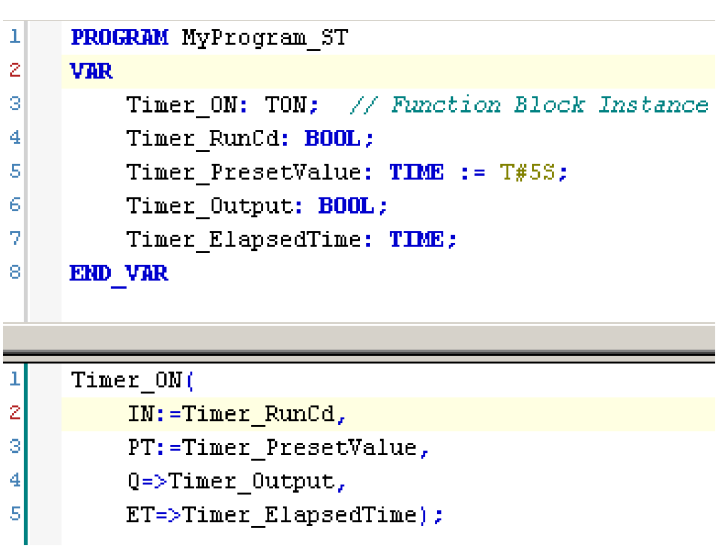
This procedure describes how to insert a function block in ST language:

Step	Action
1	Open or create a new POU in Structured Text language. **NOTE:** The procedure to create a POU is not detailed here. For more information on adding, declaring and calling POUs, refer to the related documentation *(see SoMachine, Programming Guide)*.
2	Create the input and output variables and the instance required for the function block: ● Input variables are the input parameters required by the function block ● Output variables receive the value returned by the function block
3	Use the general syntax in the **POU ST Editor** for the ST language of a Function Block. The general syntax is: `FunctionBlock_InstanceName(Input1:=VarInput1, Input2:=VarInput2,... Ouput1=>VarOutput1, Ouput2=>VarOutput2,...);`

To illustrate the procedure, consider this example with the `TON` function block graphically presented below:

Function Block	Graphical Representation
`TON`	

This table shows examples of a function block call in ST language:

Function Block	Representation in POU ST Editor
TON	

```
PROGRAM MyProgram_ST
VAR
    Timer_ON: TON;  // Function Block Instance
    Timer_RunCd: BOOL;
    Timer_PresetValue: TIME := T#5S;
    Timer_Output: BOOL;
    Timer_ElapsedTime: TIME;
END_VAR
```

```
Timer_ON(
    IN:=Timer_RunCd,
    PT:=Timer_PresetValue,
    Q=>Timer_Output,
    ET=>Timer_ElapsedTime);
```

Glossary

A

application

A program including configuration data, symbols, and documentation.

ASCII

(*American standard code for Information Interchange*) A protocol for representing alphanumeric characters (letters, numbers, certain graphics, and control characters).

B

byte

A type that is encoded in an 8-bit format, ranging from 00 hex to FF hex.

C

CFC

(*continuous function chart*) A graphical programming language (an extension of the IEC 61131-3 standard) based on the function block diagram language that works like a flowchart. However, no networks are used and free positioning of graphic elements is possible, which allows feedback loops. For each block, the inputs are on the left and the outputs on the right. You can link the block outputs to the inputs of other blocks to create complex expressions.

client

A client is a part of a communications application. The initially active part establishes a connection (TCP) or sends data to the server.

configuration

The arrangement and interconnection of hardware components within a system and the hardware and software parameters that determine the operating characteristics of the system.

controller

Automates industrial processes (also known as programmable logic controller or programmable controller).

E

expansion bus

An electronic communication bus between expansion I/O modules and a controller.

F

FB

(*function block*) A convenient programming mechanism that consolidates a group of programming instructions to perform a specific and normalized action, such as speed control, interval control, or counting. A function block may comprise configuration data, a set of internal or external operating parameters and usually 1 or more data inputs and outputs.

FTP

(*file transfer protocol*) A standard network protocol built on a client-server architecture to exchange and manipulate files over TCP/IP based networks regardless of their size.

function block diagram

One of the 5 languages for logic or control supported by the standard IEC 61131-3 for control systems. Function block diagram is a graphically oriented programming language. It works with a list of networks where each network contains a graphical structure of boxes and connection lines representing either a logical or arithmetic expression, the call of a function block, a jump, or a return instruction.

I

I/O

(*input/output*)

IL

(*instruction list*) A program written in the language that is composed of a series of text-based instructions executed sequentially by the controller. Each instruction includes a line number, an instruction code, and an operand (refer to IEC 61131-3).

INT

(*integer*) A whole number encoded in 16 bits.

IP address

The IP address of IPv4 (Internet Protocol version 4) is a value of 4 bytes which identifies the devices connected to an IP network.

L

LD

(*ladder diagram*) A graphical representation of the instructions of a controller program with symbols for contacts, coils, and blocks in a series of rungs executed sequentially by a controller (refer to IEC 61131-3).

P

POU

(*program organization unit*) A variable declaration in source code and a corresponding instruction set. POUs facilitate the modular re-use of software programs, functions, and function blocks. Once declared, POUs are available to one another.

program

The component of an application that consists of compiled source code capable of being installed in the memory of a logic controller.

S

server

A server is a part of a communications application. At first, the server is passive. It waits until clients initialize communication. A server runs on a defined port number and the clients know its address.

ST

(*structured text*) A language that includes complex statements and nested instructions (such as iteration loops, conditional executions, or functions). ST is compliant with IEC 61131-3.

V

variable

A memory unit that is addressed and modified by a program.

Index