SoMachine
EMailHandling Library Guide

06/2017

EIO0000002423.01

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.

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 EmailHandling.

The library allows your controller to send and receive an email, including attachments, to or from one or several recipients with the possibility to customize the content.

The EmailHandling 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 EmailHandling library. Personnel working with the EmailHandling 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.

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.

Care must be taken and provisions made for use of this library for machine control to avoid inadvertent consequences of commanded machine operation, state changes, or alteration of data memory or machine operating elements.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Place operator devices of the control system near the machine or in a place where you have full view of the machine.
- Protect operator commands against unauthorized access.
- If remote control is a necessary design aspect of the application, ensure that there is a local, competent, and qualified observer present when operating from a remote location.
- Configure and install the Run/Stop input, if so equipped, or, other external means within the application, so that local control over the starting or stopping of the device can be maintained regardless of the remote commands sent to it.

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

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Do not put any application program file (receipt file, G-Code file, firmware file) that you received as an email attachment into service unless you have thoroughly tested it for proper operation.

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

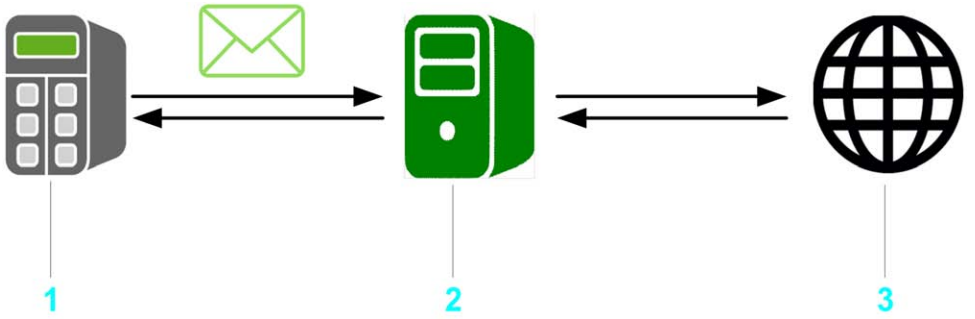
Chapter 2
Presentation of the Library

General Information

Introduction

The EmailHandling library provides email client functions that allow your controller to send emails to one or several recipients with the possibility to customize the content. The protocol type used is TCP as standard for email traffic. It is also possible to receive or delete emails from a server using the Post Office Protocol 3 (POP3).

You can connect your controller to an email server to send emails that inform about your machine status or report on key performance indicators.



1 Controller
2 Email server
3 Recipients/Sender

The following table indicates the characteristics of the library:

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

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_EMail**.

General Considerations

Consider the following limitations for email transfer:

- Only ASCII symbols are supported.
- Only IPv4 IP addresses are supported.
- The EmailHandling incorporates pointers on addresses.
- Receive acknowledgement is not supported.
- Sending or receiving files via email leads to loss of file attributes.
- In case the address of a recipient does not exist, it depends on the configuration of the server whether a feedback mail is created or whether the `FB_SendEMail` is able to create a diagnostic message.
- Archiving emails (sent and received items) has to be performed in the application program. Automatic storage of email traffic on the controller file system is not implemented because emails are mainly handled in the controller RAM (Random Access Memory).

Executing the **Online Change** command can change the contents of addresses.

⚠ CAUTION

INVALID POINTER

Verify the validity of the pointers when using pointers on addresses and executing the Online Change command.

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

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 functions provided with this library do not support secure connections such as TLS (Transport Layer Security) or SSL (Secure Socket Layer). Since the emails are not encrypted, a specific email server is required for this communication. 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.

By configuring a white list with the input `i_pbyWhiteListSender`, the entries of this list will be compared to the sender email specified in the header of the received email. This feature does not provide any protection against intentional manipulation of the header field `From:` inside the received email.

No antivirus program is available on the controller. Consider configuring one on server side.

Overview of the POUs

Function block / function	Use
`FB_SendEMail` *(see page 48)*	Implements an email client, which connects to an email server for sending emails.
`FB_Pop3EMailClient` *(see page 51)*	Implements an email client, which connects to an email server for receiving and deleting emails using POP3.

Function block / function	Use
FC_EtResultToString *(see page 57)*	Converts an enumeration element of type ET_Result to a string value.

Overview of the Structures in the Module-Specific Interface

Structure	Use
ST_CredentialsSendEMail *(see page 38)*	Contains the user-specific information for connecting to an external email server for sending emails.
ST_CredentialsReceiveEMail *(see page 39)*	Contains the user-specific information for connecting to an external email server for receiving and deleting emails.
ST_EMail *(see page 40)*	Contains the information of a received email.

Overview of the Enumerations

Enumeration	Use
ET_AuthenticationMode *(see page 26)*	Defines the mode of authentication that is required to log into an email server.
ET_Command *(see page 27)*	Indicates the command that is executed.
ET_EMailStatus *(see page 28)*	Provides information on the status of the received email.
ET_Priority *(see page 30)*	Defines the priority level that is assigned to the email.
ET_Protocol *(see page 31)*	Defines the type of protocol that is used for email transfer.
ET_Result *(see page 32)*	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_AuthenticationMode

Overview

Type:	Enumeration
Available as of:	V1.0.0.0

Description

The enumeration ET_AuthenticationMode defines the mode of authentication that is required to log into an email server.

Enumeration Elements

Name	Data type	Value	Description
NoAuthentication	INT	0	No user authentication required.
Login	INT	10	LOGIN mechanism used for authentication.
Plain	INT	20	PLAIN mechanism used for authentication.

Used By

- FB_SendEMail

ET_Command

Overview

Type:	Enumeration
Available as of:	1.1.2.0

Description

The enumeration ET_Command indicates the command that is executed.

Enumeration Elements

Name	Data type	Value	Description
NoCommand	INT	0	No command is executed.
CheckInbox	INT	1	The number of emails available on the server is verified.
Receive	INT	2	The emails are received from the server.
Delete	INT	3	The email specified with the unique ID at the input i_sUniqueID is deleted from the server.

Used By

- FB_Pop3EMailClient

ET_EMailStatus

Overview

Type:	Enumeration
Available as of:	1.1.2.0

Description

The enumeration ET_EMailStatus provides status information of the email.

Enumeration Elements

Name	Data type	Value	Description
Empty	INT	0	The structure is empty or the email data is not valid.
Ok	INT	1	The email has been successfully received.
SenderBlocked	INT	2	The sender address is blocked by the white list, or is not found in the header of the email, or exceeds the maximum length of 200 bytes.
NotSupported	INT	3	The format of the email is not supported. Supported Content-Types (not valid for the file attachment part): ● text/plain ● multipart/mixed Supported Content-Transfer-Encoding schemes: ● 7-bit ● quoted-printable ● base 64
InvalidHeader	INT	4	The header of the email is invalid.
InvalidAttachmentPath	INT	5	The path to the attachment of the email is invalid: ● The path exceeds the maximum length of 255 bytes, or ● No file name is provided.

Name	Data type	Value	Description
`InvalidAttachmentExtension`	INT	6	The attachment of the email has an invalid extension. It is not possible to receive an attachment with one of the following extensions: ● .ap_ ● .app ● .cf_ ● .cfg ● .crc ● .err ● .frc ● .log ● .prj ● .rcp ● .rsi ● .urf

Used By

- `FB_Pop3EMailClient`

ET_Priority

Overview

Type:	Enumeration
Available as of:	1.1.2.0

Description

The enumeration `ET_Priority` defines the priority level that is assigned to the email.

Enumeration Elements

Name	Data type	Value	Description
`Normal`	INT	0, 3	The email priority level is set to normal.
`VeryHigh`	INT	1	The email priority level is set to very high.
`High`	INT	2	The email priority level is set to high.
`Low`	INT	4	The email priority level is set to low.
`VeryLow`	INT	5	The email priority level is set to very low.

Used By

- `FB_SendEMail`

ET_Protocol

Overview

Type:	Enumeration
Available as of:	V1.0.0.0

Description

The enumeration ET_Protocol defines the type of protocol that is used for email transfer.

Enumeration Elements

Name	Data type	Value	Description
SMTP	INT	0	SMTP (Simple Mail Transfer Protocol) is used for email transfer.
eSMTP	INT	1	eSMTP (extended Simple Mail Transfer Protocol) is used for email transfer. Select this protocol to use additional protocol extensions, for example authentication, attachment, or email priority.

Used By

- FB_SendEMail

ET_Result

Overview

Type:	Enumeration
Available as of:	V1.0.0.0

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 48)* is FALSE, one of the following status messages is shown.			
`Ok`	UDINT	0	The operation completed successfully.
`Disabled`	UDINT	1	The function block is disabled.
`Initializing`	UDINT	2	The function block is initializing.
`Ready`	UDINT	4	The function block is ready.
`ConnectingToServer`	UDINT	5	The client is connecting to the server.
`DisconnectingFromServer`	UDINT	6	The client is disconnecting from the server.
`NoCommand`	UDINT	7	No command selected.
`SendingEMail`	UDINT	8	An email is being sent to the server.
`ConnectedToServer`	UDINT	50	A connection to the server has been established.
`MessageFromServerReceived`	UDINT	52	Data exchange with server: A message has been received.
`MessageToServerSent`	UDINT	54	Data exchange with server: A message has been sent to the server.
`SendingMessageToServer`	UDINT	55	Data exchange with server: A message is being sent to the server.
`Busy`	UDINT	56	The function block is busy.
`WaitingForExpectedMessage`	UDINT	58	Data exchange with server: Waiting for expected message to continue.
`OpeningAttachment`	UDINT	60	The attachment is opened on the file system of the controller.
`SendingAttachment`	UDINT	62	The process of sending the attachment is in progress.

Name	Data type	Value	Description
DownloadingEMail	UDINT	64	The email is downloaded from the server.
SavingAttachment	UDINT	66	The process of saving the attachment in the file system of the controller is in progress.
DeletingEMail	UDINT	68	The email is deleted from the server.
ServerOk	UDINT	70	The server is in operational state.
DeletingFile	UDINT	72	The invalid file is deleted from the system.
If q_xError of a function block *(see page 48)* is TRUE, one of the following status messages is shown.			
InvalidAuthenticationMode	UDINT	101	It has been detected that the value of i_etAuthenticationMode is not valid.
InvalidServerIP	UDINT	102	It has been detected that i_sServerIP is empty. This is not valid.
InvalidDomainName	UDINT	104	It has been detected that i_sDomainName is empty. This is not valid.
InvalidSenderEMail	UDINT	106	It has been detected that i_sSenderEMail is empty. This is not valid.
InvalidRecipientEMail	UDINT	108	It has been detected that i_sRecipientEMail is empty. This is not valid.
InvalidProtocol	UDINT	110	It has been detected that the value of i_etProtocol is not valid.
InvalidMessageAddress	UDINT	111	It has been detected that the value of i_pbyMessage is not valid.
AuthenticationFailed	UDINT	112	Authentication has not been successful: Username (i_sUsername) or password (i_sPassword) are not correct.
AuthenticationRequired	UDINT	113	Authentication is required: Enable an authentication mode.
InvalidUsername	UDINT	114	It has been detected that i_sUsername is empty. This is not valid.
InvalidPassword	UDINT	116	It has been detected that i_sPassword is empty. This is not valid.
FunctionNotSupported	UDINT	118	The selected function, such as the selected authentication mode, is not supported. Set i_etProtocol to ET_Protocol.eSMTP or try to use another i_etAuthenticationMode.
SyntaxError	UDINT	120	A syntax error, such as empty recipient address, has been detected.

Name	Data type	Value	Description
RecipientAddressTooLong	UDINT	122	It has been detected that the recipient address exceeds the allowed length of 200 bytes.
MailboxUnavailable	UDINT	124	The requested action has not been performed because the mailbox is not available.
OpenAttachmentFailed	UDINT	126	Unable to open the attachment.
ReadAttachmentFailed	UDINT	128	Unable to read the attachment.
CloseAttachmentFailed	UDINT	130	Unable to close the attachment.
MessageRejectedFromRecipient	UDINT	132	The email has been rejected by the recipient.
EMailNotFound	UDINT	133	Corresponding email has not been found on the server. Verify the value of i_sUniqueId.
InvalidPriority	UDINT	134	The value of i_etPriority is invalid.
InvalidNumberEMailsToReceive	UDINT	135	It has been detected that the value of i_uiEMailsToReceive is not valid. It must be greater than 0 and less or equal to GPL.Gc_uiInboxSize.
ServerError	UDINT	136	The server has detected an error.
InvalidWhiteListSize	UDINT	137	It has been detected that the value of i_udiWhiteListSize is not valid. It must be greater than 0.
InvalidFilePath	UDINT	138	It has been detected that the value of i_sFilePath is not valid.
InvalidMaxNumberOfAttachments	UDINT	139	The value of GPL.Gc_udiMaxNumberOfAttachments is not valid. It must be greater than 0.
InvalidMailboxBuffer	UDINT	140	The value of i_pbyMailboxBuffer is not valid.
InvalidCommand	UDINT	141	The value of i_etCommand is not valid.
InvalidBufferSize	UDINT	142	The value of i_udiBufferSize is not valid. It must be greater than 0.
NotSupported	UDINT	143	This function is not supported.
InvalidInboxSize	UDINT	144	The value of Gc_udiInboxSize is not valid. It must be greater than 0.
InvalidUniqueId	UDINT	145	The value of i_sUniqueID is not valid.
SaveAttachmentFailed	UDINT	146	The attempt to save the attachment to the file system of the controller was not successful.

Name	Data type	Value	Description
InvalidWhiteList	UDINT	147	The value of i_pyWhiteListSender is not valid. As i_udiWhiteListSize is greater than 0, i_pyWhiteListSender must have a valid address (>0).
MailboxBufferFull	UDINT	148	The buffer of the mailbox is full. The number of received bytes is greater than i_udiBufferSize.
DecodingError	UDINT	149	An internal error has been detected while decoding an email.
FailedToConnectToServer	UDINT	150	The connection to the server has not been established. Verify the Ethernet connection or the parameters i_sServerIP / i_sServerPort.
InvalidMessageFromServer	UDINT	152	It has been detected that the message received from the server is invalid.
UnexpectedMessageFromServer	UDINT	153	An unexpected message has been received from the server.
FailedToReceiveMessageFromServer	UDINT	154	A timeout has expired: No answer has been received from the server within the given time.
FailedToSendMessageToServer	UDINT	156	A communication error has been detected: Unable to send message to server.
UnexpectedProgramBehavior	UDINT	200	An internal error has been detected.

Used By

- FB_SendEMail
- FB_PopEMailClient

Chapter 4
Structures

What Is in This Chapter?

This chapter contains the following topics:

ST_CredentialsSendEMail

Overview

Type:	Structure
Available as of:	V1.0.0.0
Inherits from:	–

Description

The structure ST_CredentialsSendEMail contains the user-specific information for connecting to an external email server for sending emails.

Structure Elements

Name	Data type	Description
i_sServerIP	STRING[GPL.Gc_uiIpStringSize]	The IP address of the external email server.
i_uiServerPort	UINT	The port of the external email server.
i_etProtocol	ET_Protocol	Enumeration which indicates the protocol.
i_sDomainName	STRING[200]	The domain of the client.
i_sSenderEMail	STRING[200]	The email address of the sender.
i_sUsername	STRING[60]	The username to access the external email server.
i_sPassword	STRING[60]	The password to access the external email server.
i_etAuthenticationMode	ET_AuthenticationMode	Enumeration which indicates the authentication mode.

Used By

- FB_SendEMail

ST_CredentialsReceiveEMail

Overview

Type:	Structure
Available as of:	V1.1.2.0
Inherits from:	–

Description

The structure ST_CredentialsReceiveEMail contains the user-specific information for connecting to an external email server for receiving and deleting emails using POP3.

Structure Elements

Name	Data type	Description
i_sServerIP	STRING[GPL.Gc_uiIpStringSize]	The IP address of the external email server.
i_uiServerPort	UINT	The port of the external email server.
i_sUsername	STRING[200]	The username to access the external email server.
i_sPassword	STRING[60]	The password to access the external email server.
i_pbyWhiteListSender	POINTER TO BYTE	The start address of the string containing the address(es) of the white list. If this list contains more than one entry, the email addresses must be separated by a semicolon. The maximum size of a single address is restricted to 200 bytes. An empty string blocks all emails. Enter an asterisk in combination with a domain (*@yourdomain.com) to allow receiving emails from senders of this domain. Emails from other domains are blocked.
i_udiWhiteListSize	UDINT	Size of the white list.
i_xOverwriteAttachment	BOOL	If TRUE, attachments stored in the EMailAttachments folder are overwritten if an attachment with the same name is downloaded from the server. If FALSE, the receive process is aborted as soon as a second attachment with identical name is to be downloaded. The output q_sResultMsg indicates the file name that generated the error.

Used By

- FB_Pop3EMailClient

ST_EMail

Overview

Type:	Structure
Available as of:	V1.1.2.0
Inherits from:	–

Description

The structure ST_EMail contains the information of a received email.

Structure Elements

Name	Data type	Description
q_etEMailStatus	ET_EMailStatus	Status of the email *(see page 28)*.
q_pbyDate	POINTER TO BYTE	The start address of the string containing the date.
q_udiLengthOfDate	UDINT	Length of the date string.
q_pbySenderEMail	POINTER TO BYTE	The start address of the string containing the sender email address.
q_udiLengthOfSenderEMail	UDINT	Length of the sender email address string.
q_pbySubject	POINTER TO BYTE	The start address of the string containing the subject of the email.
udiLengthOfSubject	UDINT	Length of the subject string.
q_pbyMessage	POINTER TO BYTE	The start address of the string containing the email message.
q_udiLengthOfMessage	UDINT	Length of the message string.
q_asAttachmentPath	ARRAY [1..GPL.Gc_udiMaxNumberOfAttachments] OF STRING[255]	Relative or absolute path(s) to the attachment file(s) on the file system.
q_audiSizeOfAttachment	ARRAY [1..GPL.Gc_udiMaxNumberOfAttachments] OF UDINT	Size(s) of the attachment file(s) on the file system.
q_sUniqueID	STRING[70]	Unique ID of the email.

Used By

- FB_Pop3EMailClient

Part III
Global Variables

Chapter 5
Global Parameter List

Global Parameter List (GPL)

Overview

Type:	Global parameters
Available as of:	V1.0.0.0

Description

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

Global Parameters

Variable	Data type	Default value	Description
`Gc_udiRecipientEMailSize`	UDINT	200	Maximum number of bytes of `i_sRecipientEMail`.
`Gc_tTimeOut`	TIME	TIME#10m0s0ms	Maximum time to receive an answer from the server.
`Gc_uiIpStringSize`	UINT	15	Maximum size for IP addresses, such as `i_sServerIP`.
`Gc_udiReceiveBufferSize`	UDINT	1000	Maximum size of the receive buffer that is required to communicate with the server.
`Gc_xUseControllerTime`	BOOL	FALSE	If TRUE, the controller time is used for creating the time stamp indicating when the email was sent.
`Gc_udiInboxSize`	UDINT	10	Maximum size of the inbox. This value determines the maximum number of emails that can be downloaded with one receive command.

Variable	Data type	Default value	Description
Gc_udiMaxNumberOfAttachmentPaths	UDINT	1	Maximum size of the array ST_EMail.asAttachmentPath. It contains absolute or relative path(s) on the file system of the controller where downloaded attachments are stored. This variable does not limit the number of downloaded attachments.
Gc_xUseTop	BOOL	TRUE	If TRUE, internally the TOP0 command is used to download only the header of the email. This is useful if the white list feature is used to verify the email address of the sender before the entire message is downloaded. If the TOP command is not supported by your email server (indicated by q_sResultMsg), set this variable to FALSE.
Gc_xUseUniqueId	BOOL	TRUE	If TRUE, the UIDL command is used to retrieve the unique ID of the email. If the UIDL command is not supported by your email server (indicated by q_sResultMsg), set this variable to FALSE.

Part IV
Program Organization Units (POU)

What Is in This Part?

This part contains the following chapters:

Chapter 6
Function Blocks

What Is in This Chapter?

This chapter contains the following topics:

FB_SendEMail

Overview

Type:	Function block
Available as of:	V1.0.0.0

Task

The FB_SendEMail function block includes the related functions for sending emails. Each instance handles one SMTP connection.

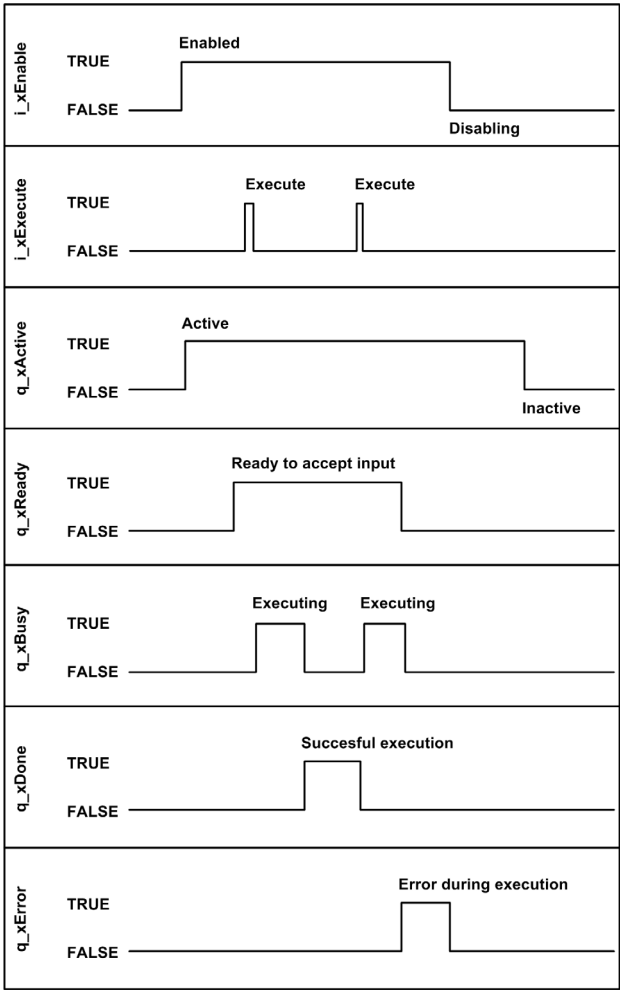
Functional Description

The FB_SendEMail function block is the user-interface to interact with an external email server. It allows you to send emails.

After the function block has been enabled and is being executed, a TCP connection to the email server is established using the user credentials that have been submitted using iq_stCredentials. As soon as the connection has been established, email data is transferred to the server. When the data transfer is completed, the TCP connection to the email server is automatically closed by the function block.

As long as the function block is executed, the output q_xBusy is set to TRUE. The output q_xDone is set to TRUE after the function block has been executed successfully.

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



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 resend an email.

Interface

Input	Data type	Description
i_xEnable	BOOL	Activation and initialization of the function block.
i_xExecute	BOOL	The function block sends an email upon rising edge of this input.
i_sRecipientEMail	STRING [GPL.Gc_udiRecipientEMailSize]	The string containing the recipient email address(es). If more than one recipient, the email addresses must be separated by a semicolon. The maximum size of a single address is restricted to 200 bytes.
i_sSubject	STRING[200]	Subject of the email.
i_pbyMessage	POINTER TO BYTE	Start address of the string in which the message is stored.
i_udiSizeOfMessage	UDINT	Size of message data.
i_etPriority	ET_Priority	The enumeration indicating the priority level that is assigned to the email *(see page 30)*.
i_sAttachmentPath	STRING[255]	Absolute or relative path to the attachment located on the controller file system. If this string is empty, no attachment is sent.

Input / Output	Data type	Description
iq_stCredentials	ST_CredentialsSendEMail	Used to pass the structure containing user settings, such as user name or password.

Output	Data type	Description
q_xActive	BOOL	If the function block is active, this output is set to TRUE.
q_xReady	BOOL	If the initialization is successful, this output signals a 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 execution is in progress.
q_xDone	BOOL	If this output is set to TRUE, the execution has been completed successfully.
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[80]	Provides additional diagnostic and status information.

FB_Pop3EMailClient

Overview

Type:	Function block
Available as of:	V1.1.2.0

Task

The `FB_Pop3EMailClient` function block includes the related functions for receiving and deleting emails using POP3. Each instance handles one POP3 connection.

Functional Description

The `FB_Pop3EMailClient` function block is the user-interface to interact with an external POP3 (email) server. It allows you to receive and delete emails. By means of attachments of received emails you are able to get input for several system features which are based on files located on the system memory. Certain file extensions are not allowed to be stored on the controller file system via `FB_Pop3EMailClient` (refer to the `ET_EmailStatus.InvalidAttachmentExtension` parameter *(see page 28)*). This applies to files that are handled automatically by the controller and to system files, such as the controller firmware to help to prevent unintended overwriting.

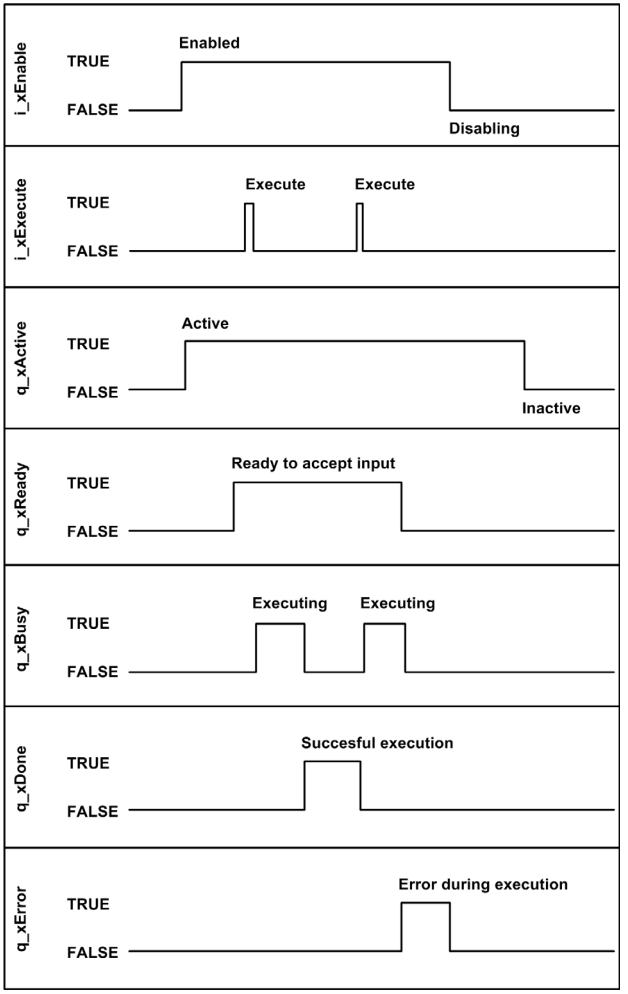
After the function block has been enabled and is being executed, a TCP connection to the POP3 server is established using the user credentials that have been submitted using `iq_stCredentials`. As soon as the connection has been established, the command specified with `i_etCommand` will be executed. When the data transfer is completed, the TCP connection to the POP3 server is closed by the function block. Received emails are deleted from the POP3 server. You can manually delete emails by specifying the email with the unique ID at the input `i_sUniqueId` and executing the delete command with `i_etCommand`. By executing further commands, the inbox structure available at `q_astInbox` containing the references to the email data are reset.

Received emails are held in volatile memory. The volatile memory is cleared when power is removed, and all held emails are therefore deleted.

NOTICE
LOSS OF DATA
Save incoming emails to non-volatile memory.
Failure to follow these instructions can result in equipment damage.

As long as the function block is executed, the output `q_xBusy` is set to TRUE. The output `q_xDone` is set to TRUE after the function block has been executed successfully.

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



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 receive or delete an email.

Interface

Input	Data type	Description
i_xEnable	BOOL	Activation and initialization of the function block.
i_xExecute	BOOL	The function block receives or deletes an email upon rising edge of this input.
i_etCommand	ET_Command	The enumeration indicating the command to be executed *(see page 27)*.
i_pbyMailboxBuffer	POINTER TO BYTE	Start address of the first byte in which the incoming emails are stored.
i_udiBufferSize	UDINT	Size of the mailbox buffer.
i_uiEMailsToReceive	UINT	Number of emails to receive from the server.
i_sFilePath	STRING[200]	Path to the folder in the controller file system where the folder EMailAttachments is created. Inside this folder the attachments of the received emails are stored. The file extension defined with the ET_EMailStatus.InvalidAttachmentExtension parameter *(see page 28)* cannot be stored. **NOTE:** If you receive a second attachment with identical name as an already available attachment in this folder, the older file may be overwritten if the global parameter ST_CredentialsReceiveEMail.i_xOverwriteAttachment is set to TRUE. If this string is empty, the folder EMailAttachments is created at the default file path of the controller.
i_sUniqueID	STRING[70]	The unique ID that is required to delete an email. After the email has been received from the server, the unique ID is displayed at the output q_astInbox.

Input / Output	Data type	Description
iq_stCredentials	ST_CredentialsReceiveEMail	Used to pass the structure containing user settings, such as user name or password.
iq_astInbox	ARRAY [1…GPL.Gc_udiInboxSize] OF ST_EMail	Structure *(see page 40)* which contains the information of received emails.

Output	Data type	Description
q_xActive	BOOL	If the function block is active, this output is set to TRUE.
q_xReady	BOOL	If the initialization is successful, this output signals a TRUE as long as the function block is operational.
q_xBusy	BOOL	If this output is set to TRUE, the function block execution is in progress.
q_xDone	BOOL	If this output is set to TRUE, the execution has been completed successfully.
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[80]	Provides additional diagnostic and status information.
q_udiNumberOfEmails	UDINT	Depends on the executed i_etCommand: • ET_Command.CheckInbox: Indicates the number of emails available on the server. • ET_Command.Receive: Indicates the number of emails received from the server. If an error has been detected, this output provides the number of emails downloaded successfully. • ET_Command.Delete: Indicates the number of emails deleted.

Chapter 7
Functions

FC_EtResultToString

Overview

Type:	Function
Available as of:	V1.0.0.0
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.

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.

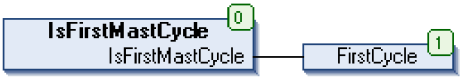
Functions `IsFirstMastCycle` and `SetRTCDrift` and Function Block `TON` are used as examples to show implementations.

Using a Function in IL Language

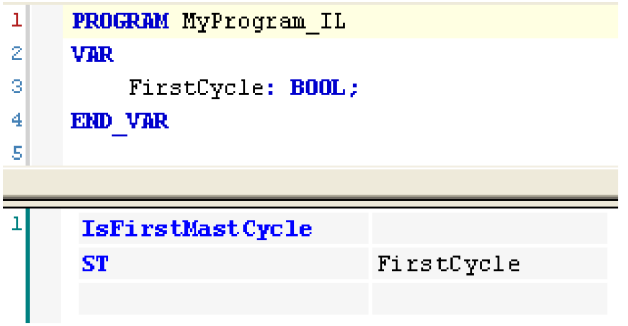
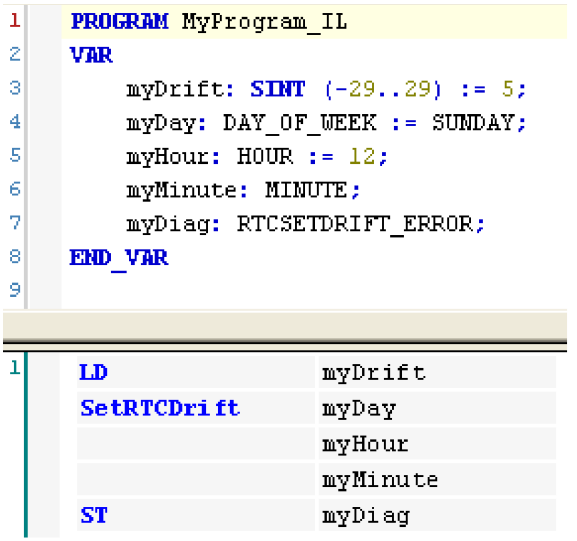
This procedure describes how to insert a function in IL language:

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

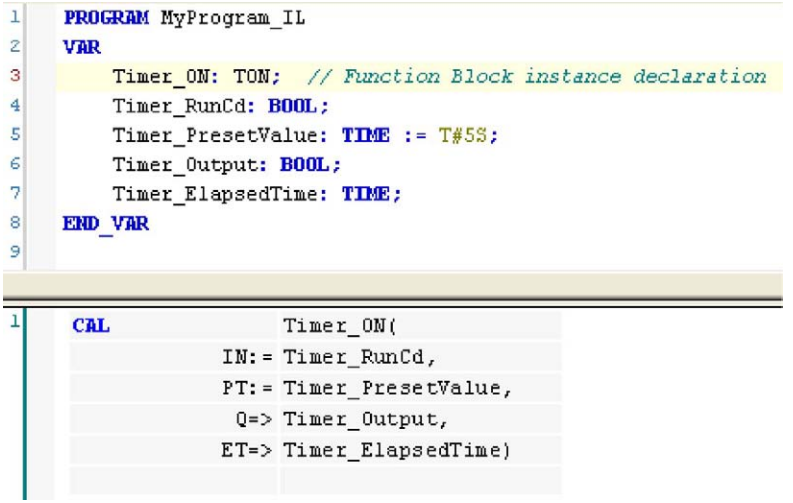
This procedure describes how to insert a function block in IL language:

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)*.
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	

```
1  PROGRAM MyProgram_IL
2  VAR
3      Timer_ON: TON;  // Function Block instance declaration
4      Timer_RunCd: BOOL;
5      Timer_PresetValue: TIME := T#5S;
6      Timer_Output: BOOL;
7      Timer_ElapsedTime: TIME;
8  END_VAR
9
```

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

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

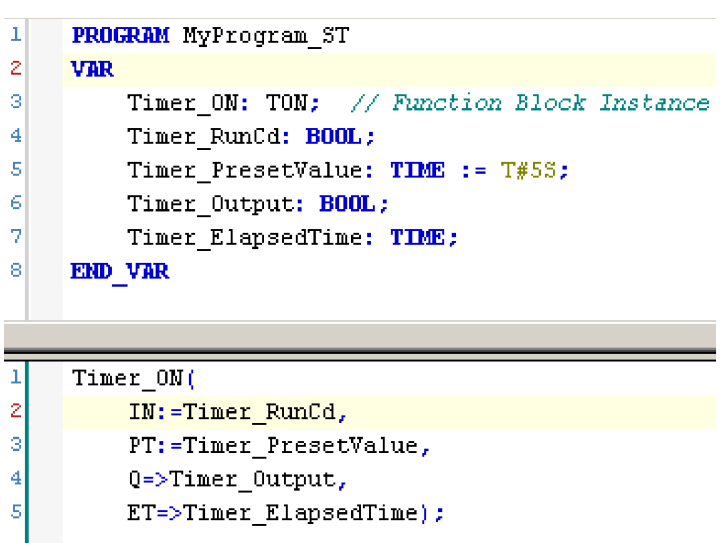
This procedure describes how to insert a function block in ST language:

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.

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.

FCL

(*forward compatible library*) A forward compatible library is developed in such a way that its functionalities are forward compatible. This means that every version of a forward compatible library contains all functionalities of the previous version and a newer library version can be easily used in already existing projects without any changes.

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.

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

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.

T

TCP

TCP (Transmission Control Protocol) is a transmission protocol used in IP networks.

V

variable

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

Index

E

F

G

L

Q

S