SoMachine
SnmpManager
Library Guide

06/2017

EIO0000002429.01

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 SnmpManager.

The SnmpManager library implements a manager for the Internet-standard protocol SNMP (Simple Network Management Protocol). The SNMP manager allows your controller to collect and organize information about managed devices on IP (Internet Protocol) networks and to modify this information in order to change the configuration of the device.

The SnmpManager library uses system functions and resources which are supported on specific controller platforms available in SoMachine.

The following controllers are supported:
- Modicon M241 Logic Controller
- Modicon M251 Logic Controller
- Modicon M258 Logic Controller
- Modicon LMC058 Motion Controller
- Modicon LMC078 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 SnmpManager library. Personnel working with the SnmpManager 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 SnmpManager library implements a manager for the Internet-standard protocol SNMP (Simple Network Management Protocol). The SNMP manager allows your controller to collect and organize information about managed devices on IP (Internet Protocol) networks and to modify this information in order to change the configuration of the device.

The SNMP manager is able to communicate to managed devices over SNMPv1-protocol and SNMPv2c-protocol.

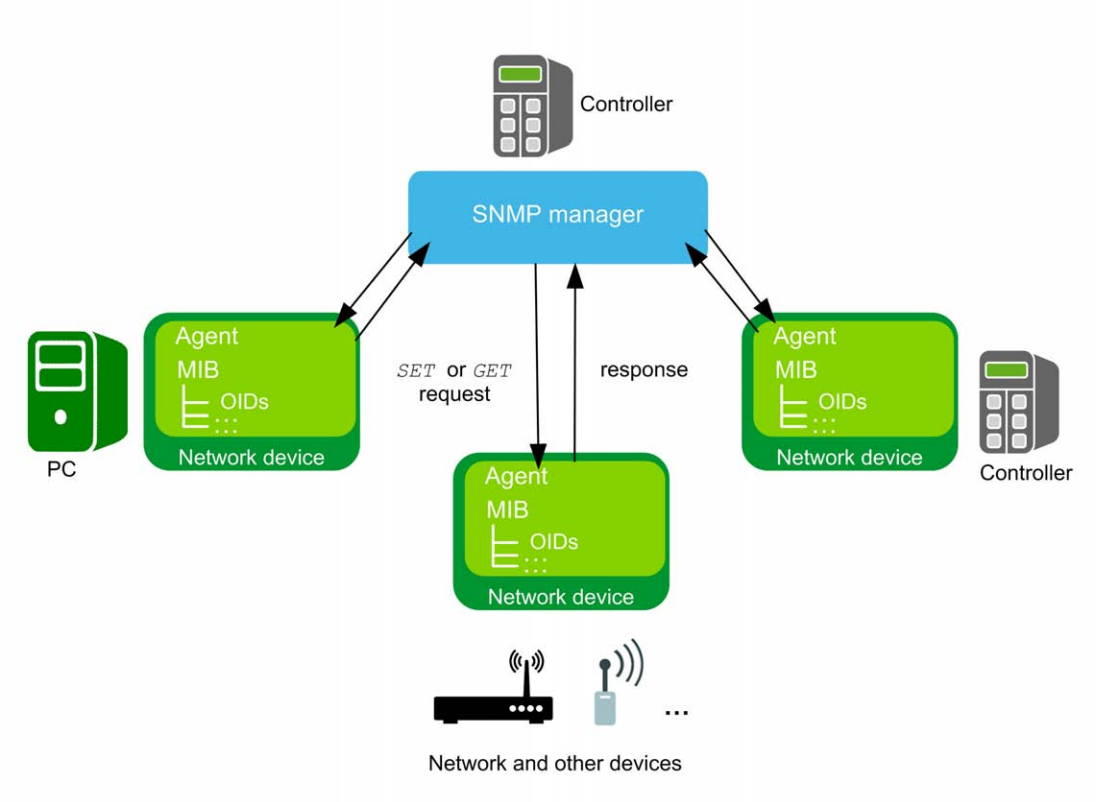
The SNMP manager complies to version 1 of the SNMP protocol defined in the IETF RFC 1157.

The SNMP manager complies to version 2 (Community-based) of the SNMP protocol defined in the IETF RFC 1901...1907.

It provides the following functions:

- Generating and transmitting a GET request for one OID (Object IDentifier).
- Generating and transmitting a SET request for one OID.
- Receiving and structuring the response of the agent.
- Providing the decoded value from a response in a suitable system type.
- Encoding a given value for a corresponding SET request.
- Managing detected errors.

An agent can be any network device providing SNMP functionality such as switches, routers, printers, or other controllers. You have to provide the information about the specific MIB (Management Information Base) structure and the corresponding OIDs for each agent.



The following table indicates the characteristics of the library:

Characteristic	Value
Library title	SnmpManager
Company	Schneider Electric
Category	**Communication**
Component	**Internet protocol suite**
Default namespace	**SE_SNMP**
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_SNMP**.

General Considerations

Consider the following limitations for SNMP communication:

- Only IPv4 (Internet Protocol version 4) is supported.
- Only one request to one SNMP agent is allowed at a time.
- Only one OID can be processed per request.
- The SnmpManager library incorporates pointers on addresses.

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 SNMP Manager Library is based on the UDP communications.

NOTE: The UDP protocol does not set up a dedicated end-to-end connection. The communication between the peers is achieved by transmitting information unidirectional from source to destination. It does not allow you to verify if a message reached its destination peer or information was lost on route. The UDP protocol does not provide any concept of acknowledgment, retransmission, or timeout.

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 SnmpManager library functions do not support secure connections such as TLS (Transport Layer Security) or SSL (Secure Socket Layer). Since the SNMP telegrams are not encrypted and authentication is not required to get or set information on an agent. 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 / function	Use
`FB_SnmpManager` *(see page 57)*	Implements an SNMP manager, which performs requests to managed devices.
`FC_EtResultToString` *(see page 61)*	Converts an enumeration element of type `ET_Result` to a string value.

Overview of the Structures in the Module-Specific Interface

Structure	Use
`ST_RequestInformation` *(see page 44)*	Contains one structure `i_stRequest` for user input and one `q_stResponse` for output data.
`ST_Request` *(see page 45)*	Contains the user-specific information for sending a request to an agent.
`ST_Response` *(see page 47)*	Contains the user information received from an agent upon a request.

Overview of the Enumerations

Enumeration	Use
`ET_SnmpTag` *(see page 32)*	Describes the possible types of values which can be sent with a SET request.
`ET_Result` *(see page 33)*	Contains the possible values that indicate the result of operations executed by the function block.
`ET_SnmpRequest` *(see page 41)*	Allows you to define the types of requests (Protocol Data Units - PDUs) which can be executed by the function block `FB_SNMPManager` via `i_etRequest`.
`ET_SnmpProtocolVersion` *(see page 42)*	Allows you to define the type for the protocol version, which can be executed by the function block `FB_SNMPManager` via `i_etVersion`.

Chapter 3
How to Use SnmpManager to Manage Devices/Clients in a Network

How Does It Work?

System Requirements

SNMP agent running on a device/client supporting protocol version v1 or v2c

Information Requirements

- MIB structure with its OID (Object IDentifier).
- Port and IP address of the device/client on which the SNMP agent is running.
- The different community names configured in the SNMP agent for reading and writing.

How to Set a New Value in an OID (Example)

Declaration:

```
PROGRAM SetRequest
VAR
       sNewSysName         : STRING[60];
       fbSnmpManager       : SE_SNMP.FB_SnmpManager;
       stRequestInfo       : SE_SNMP.ST_RequestInformation;
       etRequest           : SE_SNMP.ET_SnmpRequest;
       etProtocolVersion   : SE_SNMP.ET_SnmpProtocolVersion;
       xError              : BOOL;
       xDone               : BOOL;
       xBusy               : BOOL;
       xActive             : BOOL;
       xReady              : BOOL;

       etResult            : SE_SNMP.ET_Result;
       sResultMsg          : STRING(255);
END_VAR
```

Implementation:

```
sNewSysName := 'SchneiderSystem';
stRequestInfo.i_stRequest.pbyValueBuffer := ADR(sNewSysName);
stRequestInfo.i_stRequest.dwNumBytesValue := 60;
stRequestInfo.i_stRequest.sAgentIp := '10.128.154.41';
stRequestInfo.i_stRequest.sOid := '1.3.6.1.2.1.1.5.0';     //SysName
stRequestInfo.i_stRequest.uiAgentSnmpPort := 161;
stRequestInfo.i_stRequest.etValueType := SE_SNMP.ET_SnmpTag.OctetString
;
etRequest := SE_SNMP.ET_SnmpRequest.SetRequest;
etProtocolVersion := SE_SNMP.ET_SnmpProtocolVersion.Version2c

fbSnmpManager( i_xEnable := TRUE,
               i_xExecute := TRUE,
               i_etRequest := etRequest,
               i_etVersion := etProtocolVersion
               iq_stRequestInfo := stRequestInfo,
               q_xActive => xActive,
               q_xReady => xReady,
               q_xBusy => xBusy,
               q_xDone => xDone,
               q_xError => xError,
               q_etResult => etResult,
               q_sResultMsg => sResultMsg

)
```

Description of the parameters used:

Step	Action
1	In the `stRequestInfo.i_stRequest` *(see page 58)*, set the IP address (`sAgentIp` *(see page 45)*) of your device where the SNMP agent is running, for example, 10.128.154.47.
2	In the `stRequestInfo.i_stRequest`, set the port (`uiAgentSnmpPort` *(see page 45)*) of the SNMP agent. Standard port of SNMP and default port of the library are 161.
3	In the `stRequestInfo.i_stRequest`, set the OID (`sOid` *(see page 45)*) of the value to modify, for example, 1.3.6.1.2.1.1.5.0 (`SysName`).
4	In the `stRequestInfo.i_stRequest`, set the pointer (`pbyValueBuffer` *(see page 45)*) which points to the value to set in the OID (in case of a `GetRequest`: set the pointer to the buffer where the received value is stored).
5	In the `stRequestInfo.i_stRequest`, set the value type (`etValueType` *(see page 45)*), for example, OctetString (in case of a `GetRequest`: this parameter is not used).
6	In the `stRequestInfo.i_stRequest`, set the size of your value (`dwNumBytesValue` *(see page 45)*), for example, 60 (in case of a `GetRequest`: set the size of your value buffer). **NOTE:** For string types add an additional byte for the end of string character.
7	In the `ET_SnmpProtocolVersion` *(see page 42)*, set the type of the SNMP protocol version. The default version of the library is set to Version2c.
8	In the `ET_SnmpRequest` *(see page 41)*, set the type of request, for example, `SetRequest`.
9	Call the `FB_SnmpManager` *(see page 57)* with the settings/parameters/variables above.

Part II
Enumerations and Structures

What Is in This Part?

This part contains the following chapters:

Chapter 4
Enumerations

What Is in This Chapter?

This chapter contains the following topics:

ET_SnmpTag

Overview

Type:	Enumeration
Available as of:	V1.0.0.0

Description

The enumeration ET_SnmpTag describes the possible types of values which can be sent with a SET request.

NOTE: The type of value Counter64 is not supported in SNMPv1.

The suggested buffer type (see table below) indicates the variable type that is to be referenced by the pointer iq_stRequestInfo.i_stRequest.pbyValueBuffer.

- For a SET request, these variable types are mandatory.
- For value types received by a GET request, the suggested buffer type refers to the value type that is used by the function block to represent the received data.

Enumeration Elements

Name	Data type	Value	Description
Integer	BYTE	2	Value of type signed integer (32-bit wide). Suggested buffer type: DINT
OctetString	BYTE	4	Value of type string with 8-bit characters. Suggested buffer type: STRING
ObjectID	BYTE	6	Value of type ObjectID. Suggested buffer type: STRING (numbered sequence separated by dots)
IpAddress	BYTE	64	Value of type OctetString(4). Suggested buffer type: STRING[15]
Counter32	BYTE	65	Value of type unsigned integer (32-bit wide). Suggested buffer type: UDINT
Gauge32	BYTE	66	Value of type unsigned integer (32-bit wide). Suggested buffer type: UDINT
TimeTicks	BYTE	67	Value of type unsigned integer (32-bit wide). Suggested buffer type: UDINT
Counter64	BYTE	70	Value of type unsigned integer (64-bit wide). Suggested buffer type: ULINT Not supported in SNMPv1.

Used By

- FB_SnmpManager

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 and extended error codes of operations executed by the function block.

In case of detected errors received in the response telegram from the agent, the outputs `iq_stRequestInfo.q_stResponse.sOID` and `iq_stRequestInfo.q_stResponse.sAgentIp` are still set with the data of the request which caused the error.

Enumeration Elements

Name	Data type	Value	Description
If `q_xError` of the function block is FALSE, the following status messages are shown:			
`Ok`	UDINT	0	The operation was completed successfully and the function block is idle.
`NotReady`	UDINT	2	The requested operation cannot be executed in the present state.
`Disabled`	UDINT	3	The function block is disabled.
`Sending`	UDINT	4	The function block sends a request telegram with the information specified with the input `iq_stRequestInfo.i_stRequest`.
`Listening`	UDINT	5	The function block is waiting for a response from the agent.
`Evaluating`	UDINT	6	The function block processes the received telegram.
If `q_xError` of the function block is TRUE, the following status messages are shown:			
`UnknownState`	UDINT	1	The operation was not completed successfully due to an indeterminable error.
`GplMaxUserDataTooBig`	UDINT	15	`GPL.Gc_dwMaxSizeUserData` is out of range. Verify the settings of Global Parameter List *(see page 53)*.

Name	Data type	Value	Description
GplMaxUserDataTooSmall	UDINT	16	GPL.Gc_dwMaxSizeUserData is out of range. Verify the settings of Global Parameter List *(see page 53)*.
GplMaxOidTooBig	UDINT	17	GPL.Gc_dwMaxSizeOid is out of range. Verify the settings of Global Parameter List *(see page 53)*.
GplMaxOidTooSmall	UDINT	18	GPL.Gc_dwMaxSizeOid is out of range. Verify the settings of Global Parameter List *(see page 53)*.
GplTimeoutZero	UDINT	19	GPL.Gc_udiTimeout is out of range. Verify the settings of Global Parameter List *(see page 53)*.
InputMissingOid	UDINT	20	Mandatory input iq_stRequestInfo.i_stRequest.sOid is invalid. Verify that the input is assigned.
InputInvalidBufferSize	UDINT	21	Buffer has to be larger than zero. The buffer size is provided by the value iq_stRequestInfo.i_stRequest.dwNumBytesValue.
InputInvalidRequest	UDINT	22	Only ET_SnmpRequest.GetRequest and ET_SnmpRequest.SetRequest are supported.
InputInvalidOid	UDINT	23	OID must be provided as a STRING containing a sequence of numbers separated by dots. The first number can only be 0,1, or 2.
InputInvalidIp	UDINT	24	IP address is invalid. Verify that the IP address is provided as a STRING containing a sequence of 4 sets of numbers separated by dots. The values represented by the sets of numbers have to be in the range of 0...255.
InputInvalidPointer	UDINT	25	The buffer referenced by the pointer iq_stRequestInfo.i_stRequest.pbyValueBuffer is invalid. Verify address of pointer referencing the buffer, the size of buffer and if it is writable.
InputSize32BitValue	UDINT	26	The data type referenced by iq_stRequestInfo.i_stRequest.etValueType allows values with a maximum size of 4 bytes.

Name	Data type	Value	Description
GplMaxUserDataTooSmall	UDINT	16	GPL.Gc_dwMaxSizeUserData is out of range. Verify the settings of Global Parameter List *(see page 53)*.
GplMaxOidTooBig	UDINT	17	GPL.Gc_dwMaxSizeOid is out of range. Verify the settings of Global Parameter List *(see page 53)*.
GplMaxOidTooSmall	UDINT	18	GPL.Gc_dwMaxSizeOid is out of range. Verify the settings of Global Parameter List *(see page 53)*.
GplTimeoutZero	UDINT	19	GPL.Gc_udiTimeout is out of range. Verify the settings of Global Parameter List *(see page 53)*.
InputMissingOid	UDINT	20	Mandatory input iq_stRequestInfo.i_stRequest.sOid is invalid. Verify that the input is assigned.
InputInvalidBufferSize	UDINT	21	Buffer has to be larger than zero. The buffer size is provided by the value iq_stRequestInfo.i_stRequest.dwNumBytesValue.
InputInvalidRequest	UDINT	22	Only ET_SnmpRequest.GetRequest and ET_SnmpRequest.SetRequest are supported.
InputInvalidOid	UDINT	23	OID must be provided as a STRING containing a sequence of numbers separated by dots. The first number can only be 0,1, or 2.
InputInvalidIp	UDINT	24	IP address is invalid. Verify that the IP address is provided as a STRING containing a sequence of 4 sets of numbers separated by dots. The values represented by the sets of numbers have to be in the range of 0...255.
InputInvalidPointer	UDINT	25	The buffer referenced by the pointer iq_stRequestInfo.i_stRequest.pbyValueBuffer is invalid. Verify address of pointer referencing the buffer, the size of buffer and if it is writable.
InputSize32BitValue	UDINT	26	The data type referenced by iq_stRequestInfo.i_stRequest.etValueType allows values with a maximum size of 4 bytes.

Name	Data type	Value	Description
InputSize64BitValue	UDINT	27	The data type referenced by iq_stRequestInfo.i_stRequest.etValueType allows values with maximum size of 8 bytes.
InputInvalidValueIp	UDINT	28	The data type referenced by iq_stRequestInfo.i_stRequest.etValueType requires a specified format: a STRING containing a sequence of 4 sets of numbers separated by dots. The values represented by the sets of numbers have to be in the range 0...255.
InputInvalidValueOid	UDINT	29	The data type referenced by iq_stRequestInfo.i_stRequest.etValueType requires a specified format: a STRING containing a sequence of numbers separated by dots. The first number can only be 0,1, or 2.
InputInvalidValueVersion	UDINT	30	The input data type i_etVersion of the function block only allows the values Version1 or Version2c.
UdpIssue	UDINT	40	Generic TCP issue. Refer to TcpUdpCommunication Library Guide *(see SoMachine, TcpUdpCommunication, Library Guide)*.
UdpNotSupported	UDINT	41	The requested UDP operation is not supported by this controller.
UdpSocketMngListTooSmall	UDINT	42	UDP socket could not be opened because the internal socket management list is full. GPL.Gc_uiSocketManagementListSize needs to be increased in TcpUdp library.
UdpNumBytesSendOutOfRange	UDINT	43	Number of bytes to send exceeds valid range. Contact your local Schneider Electric support.
UdpInvalidBufferAddress	UDINT	44	The address of the telegram buffer is invalid. Contact your local Schneider Electric support.
UdpInvalidIp	UDINT	45	Provided IP address is invalid. Verify the given input iq_stRequestInfo.i_stRequest.sAgentIp.
UdpReceiveBufferSizeOutOfRange	UDINT	46	The size of the receive buffer exceeds the valid range. Contact your local Schneider Electric support.

Name	Data type	Value	Description
UdpNotReady	UDINT	47	UDP request cannot be executed at the moment.
UdpTimeout	UDINT	48	No response received from agent in the given time with the given retries. Try to increase GPL.Gc_udiTimeout and / or GPL.Gc_uiMaxNumberRetries.
ValidateRequestIdNoMatch	UDINT	60	Request ID received in response from agent does not match the request ID sent to the agent. Refer to iq_stRequestInfo.q_stResponse for information about which agent (IP Address) and which OID provoked this message.
ValidateOidNoMatch	UDINT	61	OID received in response from agent does not match the requested OID. Refer to iq_stRequestInfo.q_stResponse for information about which agent (IP address) and which OID provoked this message.
ValidateBufferTooSmall	UDINT	62	Provided buffer in iq_stRequestInfo.i_stRequest.pbyValueBuffer is insufficient for the value received from the agent. Refer to iq_stRequestInfo.q_stResponse for information about which agent (IP Address) and which OID provoked this message and the number of bytes received for the value. **NOTE:** If the received value is of type ObjectId the decoded value is usually more than twice as large as the received number of bytes, plan buffer size accordingly.
ValidateVersionNotSupported	UDINT	63	Protocol version mismatch in request/ response.
ValidateNoValue	UDINT	69	The received telegram from the agent contained no value and no error message. Refer to iq_stRequestInfo.q_stResponse for information about which agent (IP Address) and which OID provoked this message.
SnmpResponseGenErr	UDINT	70	Generic error message received from SNMP agent. Any other issue but the ones provided in this list. Refer to iq_stRequestInfo.q_stResponse for information about which agent (IP address) and which OID provoked this message.

Name	Data type	Value	Description
SnmpResponseTooBig	UDINT	71	Response from agent on a GET request or a SET request. The determined response message is too large for agent to send. Refer to iq_stRequestInfo.q_stResponse for information about which agent (IP address) and which OID provoked this message.
SnmpResponseSetNoAccess	UDINT	72	Response from agent on a SET request. Access denied to requested OID. Verify community name in iq_stRequestInfo.i_stRequest.sCommunityName. Refer to iq_stRequestInfo.q_stResponse for information about which agent (IP address) and which OID provoked this message.
SnmpResponseSetWrongType	UDINT	73	Response from agent on a SET request. Given value type does not match the type of the value specified by the OID. Refer to iq_stRequestInfo.q_stResponse for information about which agent (IP address) and which OID provoked this message.
SnmpResponseSetWrongLength	UDINT	74	Response from agent on a SET request: Length of value is inconsistent with defined length of OID value: Verify length of the OID value if error message persists contact your local Schneider Electric support. Refer to iq_stRequestInfo.q_stResponse for information about which agent (IP address) and which OID provoked this message.
SnmpResponseSetWrongEncoding	UDINT	75	Response from agent on a SET request. Incorrect encoding used for the value. Contact your local Schneider Electric support. Refer to iq_stRequestInfo.q_stResponse for information about which agent (IP address) and which OID provoked this message.
SnmpResponseSetWrongValue	UDINT	76	Response from agent on a SET request. Given value for this OID is invalid or not supported in any other way. Verify input data in iq_stRequestInfo.i_stRequest. Refer to iq_stRequestInfo.q_stResponse for information about which agent (IP address) and which OID provoked this message.

Name	Data type	Value	Description
SnmpResponseSetNotWritable	UDINT	77	Response from agent on a SET request. Given value for this OID is invalid or not supported in any other way. Verify input data in iq_stRequestInfo.i_stRequest. Refer to iq_stRequestInfo.q_stResponse for information about which agent (IP address) and which OID provoked this message.
SnmpResponseSetNoSuchName	UDINT	78	Response from agent on a SET request. Name is inconsistent or does not exist and cannot be created at the moment. Refer to iq_stRequestInfo.q_stResponse for information about which agent (IP address) and which OID provoked this message.
SnmpResponseSetInconsistentValue	UDINT	79	Response from agent on a SET request. Value cannot be set at the moment or resource not available to set this value at the moment. Refer to iq_stRequestInfo.q_stResponse for information about which agent (IP address) and which OID provoked this message.
SnmpResponseSetNoCreation	UDINT	80	Response from agent on a SET request. Value does not exist, neither can it be created. Refer to iq_stRequestInfo.q_stResponse for information about which agent (IP address) and which OID provoked this message.
SnmpResponseGetNoSuchObject	UDINT	81	Response from agent on a GET request. OID prefix does not match any OID prefix for accessible variables. Verify OID in iq_stRequestInfo.i_stRequest.sOid. Refer to iq_stRequestInfo.q_stResponse for information about which agent (IP address) and which OID provoked this message.
SnmpResponseGetNoSuchInstance	UDINT	82	Response from agent on a GET request. No exact match for OID found. Verify OID in iq_stRequestInfo.i_stRequest.sOid. Refer to iq_stRequestInfo.q_stResponse for information about which agent (IP address) and which OID provoked this message.

Name	Data type	Value	Description
SnmpResponseEndofMibView	UDINT	83	Response from agent on a GET request. No further variables in lexicographical order. Refer to iq_stRequestInfo.q_stResponse for information about which agent (IP address) and which OID provoked this message.
SnmpResponseV1WrongErrorState	UDINT	84	The responded value of ErrorState cannot be handled by the protocol version 1.
SnmpVersion1SetCounter64	UDINT	88	Counter64 is not available for SNMPv1.
ValueNotSupported	UDINT	90	Response from agent contains indeterminable value. Only value types provided by ET_SnmpTag *(see page 32)* are supported. Refer to iq_stRequestInfo.q_stResponse for information about which agent (IP address) and which OID provoked this message.
UnknownResult	UDINT	99	Feedback of FC_EtResultToString in case of an indeterminable EtResult value.

In case of recurring timeouts and other communication issues:

- Verify physical connections of controller, SNMP agent, and so on.
- Verify connection configuration of controller and SNMP agent.
- Verify communication route between controller and SNMP agent, for example, switch and router settings, firewall settings (UDP communication and SNMP ports need to be allowed).
- Verify provided IP address and SNMP port in iq_stRequestInfo.i_stRequest.
- Verify provided community name in iq_stRequestInfo.i_stRequest.

Used By

- FB_SnmpManager

ET_SnmpRequest

Overview

Type:	Enumeration
Available as of:	V1.0.0.0

Description

The enumeration ET_SnmpRequest defines the types of requests (Protocol Data Units - PDUs) which can be executed by the function block *(see page 57)* FB_SnmpManager via i_etRequest.

Enumeration Elements

Name	Data type	Value	Description
NoCommand	BYTE	0	No type of request is defined. Initial state and default value.
GetRequest	BYTE	1	GET requests ask an agent for the value referenced by the OID.
SetRequest	BYTE	4	SET requests set a value referenced by the OID on an agent.

Used By

- FB_SnmpManager

ET_SnmpProtocolVersion

Overview

Type:	Enumeration
Available as of:	V1.0.0.0

Description

The enumeration `ET_SnmpProtocolVersion` defines the types of the SNMP protocol types, a SNMP agent can handle. The enumeration can be executed be the function block *(see page 57)* `FB_SnmpManager` via the input `i_etVersion`.

Enumeration Elements

Name	Data type	Value	Description
`Version1`	BYTE	0	SNMP protocol type SNMPv1 used for communication SNMP manager and agent
`Version2c`	BYTE	1	SNMP protocol type SNMPv2c used for communication SNMP manager and agent

Chapter 5
Structures

What Is in This Chapter?

This chapter contains the following topics:

ST_RequestInformation

Overview

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

Description

The structure ST_RequestInformation contains two structures, one for the input data necessary to send a request and one for the output data to present the response to the user.

Structure Elements

Name	Data type	Description
i_stRequest	ST_Request *(see page 45)*	Structure containing required user input data
q_stResponse	ST_Response *(see page 47)*	Structure containing response to user request

Used By

- FB_SnmpManager

ST_Request

Overview

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

Description

The structure ST_Request contains the user-specific information for sending an SNMP request to an agent.

Structure Elements

Name	Data type	Description
sOid	STRING[GPL.Gc_dwMaxSizeOid]	Requested OID Format: sequence of numbers separated by dots, for example, 1.3.6.1.2.1.1.5.0.
sAgentIp	STRING[15]	IP address of the SNMP agent
pbyValueBuffer	POINTER TO BYTE	The pointer to the first byte of the buffer. For SET requests, this is the buffer containing the user data to set on the agent. For GET requests, this is the buffer where the function block stores the response from the agent.
dwNumBytesValue	DWORD	Size of the buffer or the value in the buffer The buffer is referenced by the pointer pbyValueBuffer. For GET requests, this defines the size of the buffer where the received values are stored. For SET requests, this defines the size of the value in the buffer to be sent. **NOTE:** For string types add an additional byte for the end of string character.
etValueType	ET_SnmpTag *(see page 32)*	Type of the value to be set by the SET requests (not used for GET requests).
uiAgentSnmpPort	UINT	SNMP port of the agent Default value: 161
sCommunityName	STRING[255]	Community name for request Default values: • SET request: private • GET request: public

Used By

- FB_SnmpManager

ST_Response

Overview

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

Description

The structure ST_Response provides information received from an SNMP agent upon a request.

Structure Elements

Name	Data type	Description
sOid	STRING[GPL.Gc_dwMaxSizeOid]	The OID the agent responded to. This is additional information for you to match your request to the response.
sAgentIp	STRING[15]	The IP address of the agent sending the response. This is additional information for you to match your request to the response.
pbyValueBuffer	POINTER TO BYTE	The pointer to the buffer containing the value referenced by the OID received on a GET request (buffer provided by iq_stRequestInfo.i_stRequest.pbyValueBuffer). In case of a SET request, the buffer is not written.
dwNumBytesValue	DWORD	Size of the value referenced by the OID which was set/gotten by the request before.
etProtocolVersion	ET_SnmpProtocolVersion	SNMP protocol version sent from agent.

Used By

- FB_SnmpManager

Part III
Global Variables

What Is in This Part?

This part contains the following chapters:

Chapter 6
Global Constants List

Global Constants List (GCL)

Overview

Type:	Global constants
Available as of:	V1.0.0.0

Description

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

Global Constants

Variable	Data type	Value	Description
`Gc_dwMaxLimitUserData`	DWORD	1500	Maximum size for the combination of OID and value in bytes, in the SNMP telegram. This value limits the `Gc_MaxSizeUserData` parameter.
`Gc_uiDefaultPortSnmp`	UINT	161	Port number of agent for sending/receiving SNMP requests. The default value is the standard port `161`.
`Gc_sLibraryVersion`	STRING[80]	Vx.x.x.0[1]	Library version
[1] This value varies to indicate the version of the library.			

Chapter 7
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 SnmpManager library. They can be overwritten specifically for your project in the **Library Manager**.

Global Parameters

Variable	Data type	Value range	Description
`Gc_udiTimeout`	UDINT	> 0	Maximum time to wait for a response from the agent (in milliseconds). Default value: 10000
`Gc_uiMaxNumberRetries`	UINT	0..65355	Maximum number of retries to send a request after a timeout has occurred. Default value: 2
`Gc_dwMaxSizeOid`	DWORD	1... `GPL.Gc_dwMaxSizeUserData`	Maximum size for STRING `iq_stRequestInfo.i_stRequest.sOid`. Default value: 255
`Gc_dwMaxSizeUserData`	DWORD	1... `GCL.GC_MaxLimitUserData`.	Maximum size for the combination of OID and value in bytes, in the SNMP telegram. This value is limited by `Gc_dwMaxLimitUserData`. Default value: 1500

Part IV
Program Organization Units (POU)

What Is in This Part?

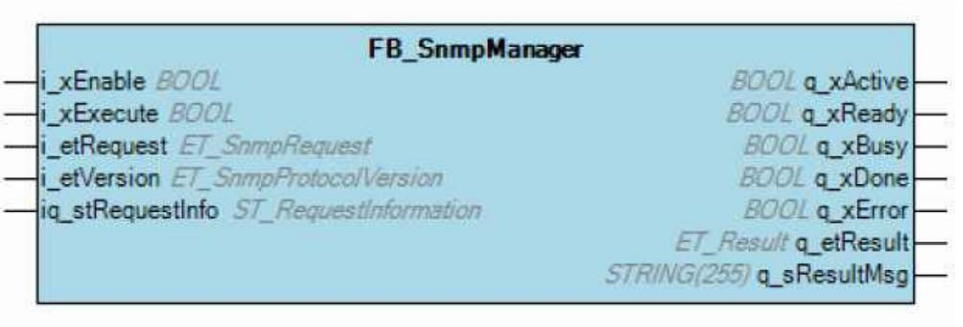
This part contains the following chapters:

Chapter 8
Function Blocks

FB_SnmpManager

Overview

Type:	Function block
Available as of:	V1.0.0.0



Task

The FB_SnmpManager function block is used to perform SNMP requests to managed devices on IP (Internet Protocol) networks.

Functional Description

The FB_SnmpManager function block is the user interface for SNMP communications.

Only one request to one agent can be managed at a time.

The function block needs to be enabled and ready to perform a request. When starting the execution of a request, the inputs `i_etRequest` and `iq_stRequestInfo` need to be set. The information from the inputs is used to build a SNMP telegram containing the request, and then to send it via UDP to the agent. The function block waits for a response from the agent, processes it and presents the received information at `iq_stRequestInfo.stResponse`. As long as the function block is executing a request, the output `q_xBusy` is set to TRUE and `q_etResult` shows the state of operation. The output `q_xDone` signals a successful execution and `q_xError` shows if the function block detects an error during execution with `q_etResult` and `q_sResultMsg` presenting further information on the nature or cause of the detected error. If an error is detected, the function block needs to be reset by disabling and re-enabling it.

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_etRequest` is executed upon rising edge of this input.
`i_etRequest`	`ET_SnmpRequest`	The SNMP request that is executed if the input `i_xExecute` is TRUE. Make sure that `iq_stRequestInfo` is available before the SNMP request is executed.
`i_etVersion`	`ET_SnmpProtocolVersion`	Specifies protocol version for communication with the agent. Default value is `ET_SnmpProtocolVersion.Version2c`.

Input / Output	Data type	Description
`iq_stRequestInfo`	`ST_RequestInformation` *(see page 44)*	Used to pass the structure containing information for sending a request to an agent and the structure to present the received response from the agent.

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 is executing the command specified at the input i_etRequest.
q_xDone	BOOL	If this output is set to TRUE, the function block has successfully completed the command specified at the input i_etRequest. Additional data is provided at iq_stRequestInfo.q_stResponse.
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 9
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 string value containing the diagnostic and status information.

Functional Description

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

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 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.

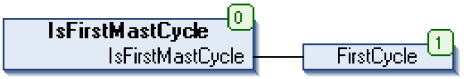
Functions `IsFirstMastCycle` and `SetRTCDrift` and Function Block `TON` are used as examples to show implementations.

Using a Function in IL Language

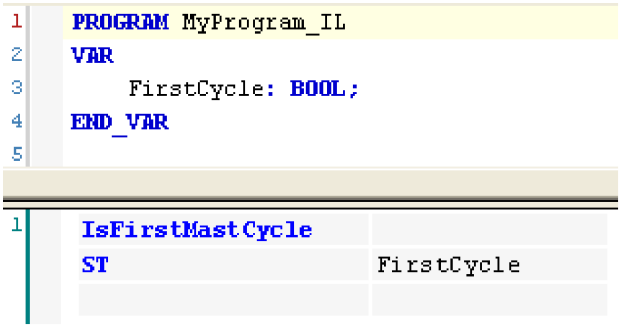
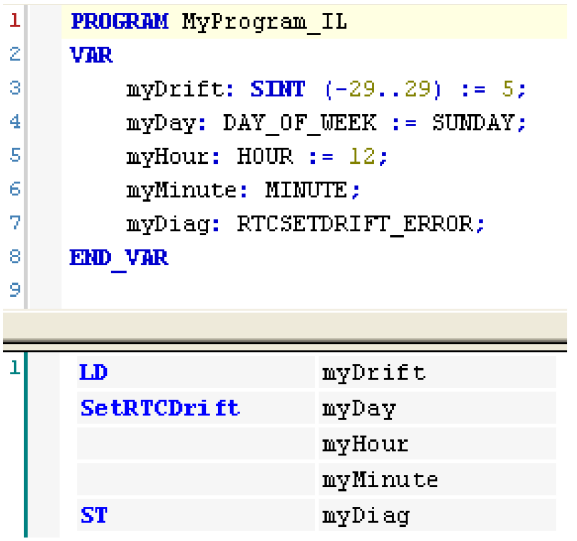
This procedure describes how to insert a function in IL language:

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 SoMachine 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:

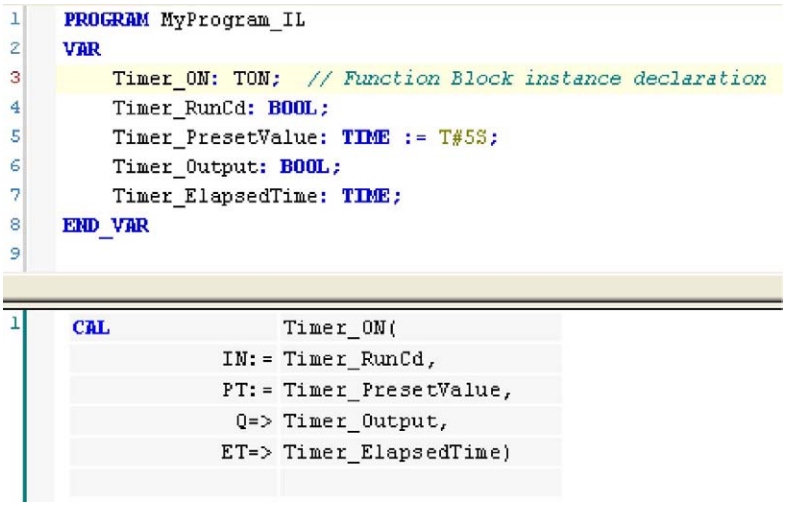
Step	Action
1	Open or create a new POU in Instruction List language. **NOTE:** The procedure to create a POU is not detailed here. For more information, refer to Adding and Calling POUs *(see SoMachine, Programming Guide)*.

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 SoMachine 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 SoMachine 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

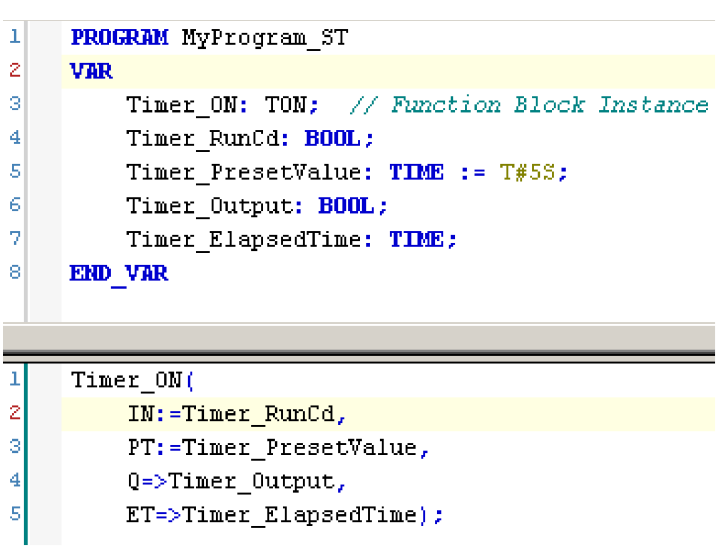
This procedure describes how to insert a function block in ST language:

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 SoMachine 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.

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.

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

(*Internet protocol* Part of the TCP/IP protocol family that tracks the Internet addresses of devices, routes outgoing messages, and recognizes incoming messages.

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).

M

MIB

(*management information base*) An object database that is monitored by a network management system like SNMP. SNMP monitors devices are defined by their MIBs. Schneider Electric has obtained a private MIB, groupeschneider (3833).

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

SNMP

(*simple network management protocol*) A protocol that can control a network remotely by polling the devices for their status and viewing information related to data transmission. You can also use it to manage software and databases remotely. The protocol also permits active management tasks, such as modifying and applying a new configuration.

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.

U

UDP

(*user datagram protocol*) A connectionless mode protocol (defined by IETF RFC 768) in which messages are delivered in a datagram (data telegram) to a destination computer on an IP network. The UDP protocol is typically bundled with the Internet protocol. UDP/IP messages do not expect a response, and are therefore ideal for applications in which dropped packets do not require retransmission (such as streaming video and networks that demand real-time performance).

V

variable

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

Index

G

L

Q

S