EcoStruxure Machine Expert

OpcUaHandling

Library Guide

EIO0000004021.05
06/2024

www.se.com

Legal Information

The information provided in this document contains general descriptions, technical characteristics and/or recommendations related to products/solutions.

This document is not intended as a substitute for a detailed study or operational and site-specific development or schematic plan. It is not to be used for determining suitability or reliability of the products/solutions for specific user applications. It is the duty of any such user to perform or have any professional expert of its choice (integrator, specifier or the like) perform the appropriate and comprehensive risk analysis, evaluation and testing of the products/solutions with respect to the relevant specific application or use thereof.

The Schneider Electric brand and any trademarks of Schneider Electric SE and its subsidiaries referred to in this document are the property of Schneider Electric SE or its subsidiaries. All other brands may be trademarks of their respective owner.

Schneider Electric does not grant any right or license for commercial use of the document or its content, except for a non-exclusive and personal license to consult it on an "as is" basis.

Schneider Electric reserves the right to make changes or updates with respect to or in the content of this document or the format thereof, at any time without notice.

To the extent permitted by applicable law, no responsibility or liability is assumed by Schneider Electric and its subsidiaries for any errors or omissions in the informational content of this document, as well as any non-intended use or misuse of the content thereof.

Table of Contents

Safety Information

Important Information

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.

Qualification of Personnel

A qualified person is one who has the following qualifications:

- Skills and knowledge related to the construction and operation of electrical equipment and the installation.
- Knowledge and experience in industrial control programming.
- 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

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 design of this overall system (for example, machine design).

Any other use is not intended and may be hazardous.

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 are 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:

(In case of divergence or contradiction between any translation and the English original, the original text in the English language will prevail.)

- 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 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

Document Scope

This document describes the library OpcUaHandling.

The OpcUaHandling library provides function blocks to access OPC UA client functionality running on the controller.

Validity Note

This document has been updated for the release of EcoStruxure™ Machine Expert V2.2.2.

Related Documents

Document title	Reference
Cybersecurity Best Practices	CS-Best-Practices-2019-340
Cybersecurity Guidelines for EcoStruxure Machine Expert, Modicon and PacDrive Controllers and Associated Equipment	EIO0000004242
EcoStruxure Machine Expert, Functions and Libraries User Guide	EIO0000002829 (ENG); EIO0000002830 (FRE); EIO0000002831 (GER); EIO0000002832 (ITA); EIO0000002833 (SPA); EIO0000002834 (CHS);
EcoStruxure Machine Expert, Programming Guide	EIO0000002854 (ENG); EIO0000002855 (FRE); EIO0000002856 (GER); EIO0000002857 (ITA); EIO0000002858 (SPA); EIO0000002859 (CHS);
How To Manage Certificates for OPC UA Client, User Guide	EIO0000004065 (ENG); EIO0000004066 (FRE); EIO0000004067 (GER); EIO0000004068 (ITA); EIO0000004069 (SPA); EIO0000004070 (CHS);
OPC 10000-x: UA Part x (Specification documents for OPC UA provided by the OPC Foundation)	https://reference.opcfoundation.org
PLCopen OPC UA Client Function Blocks for IEC 61131-3 version 1.02	https://plcopen.org/system/files/downloads/plcopen_opc_ua_client_function_blocks_for_iec_61131-3_version_1.02.pdf
EcoStruxure Machine Expert OPCUAHandling Example Guide	EIO0000004099 (ENG)

To find documents online, visit the Schneider Electric download center (www.se.com/ww/en/download/).

Product Related Information

⚠WARNING

LOSS OF CONTROL

- Perform a Failure Mode and Effects Analysis (FMEA), or equivalent risk analysis, of your application, and apply preventive and detective controls before implementation.
- Provide a fallback state for undesired control events or sequences.
- Provide separate or redundant control paths wherever required.
- Supply appropriate parameters, particularly for limits.
- Review the implications of transmission delays and take actions to mitigate them.
- Review the implications of communication link interruptions and take actions to mitigate them.
- Provide independent paths for control functions (for example, emergency stop, over-limit conditions, and error conditions) according to your risk assessment, and applicable codes and regulations.
- Apply local accident prevention and safety regulations and guidelines.[1]
- Test each implementation of a system for proper operation before placing it 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 PROGRAM ORGANIZATION UNITS

- Perform a safety-related analysis for the application and the devices installed.
- Ensure that the Program Organization Units (POUs) are compatible with the devices in the system and have no unintended effects on the proper functioning of the system.
- Ensure that the axis is homed and that the homing is valid before usage of absolute movements or POUs using absolute movements.
- 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 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 product as a control device to avoid inadvertent consequences of commanded machine operation, state changes, or alteration of data memory or machine operating parameters.

⚠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.

Information on Non-Inclusive or Insensitive Terminology

As a responsible, inclusive company, Schneider Electric is constantly updating its communications and products that contain non-inclusive or insensitive terminology. However, despite these efforts, our content may still contain terms that are deemed inappropriate by some customers.

Terminology Derived from Standards

The technical terms, terminology, symbols and the corresponding descriptions in the information contained herein, or that appear in or on the products themselves, are generally derived from the terms or definitions of international standards.

In the area of functional safety systems, drives and general automation, this may include, but is not limited to, terms such as *safety*, *safety function*, *safe state*, *fault*, *fault reset*, *malfunction*, *failure*, *error*, *error message*, *dangerous*, etc.

Among others, these standards include:

Standard	Description
IEC 61131-2:2007	Programmable controllers, part 2: Equipment requirements and tests.
ISO 13849-1:2023	Safety of machinery: Safety related parts of control systems. General principles for design.
EN 61496-1:2020	Safety of machinery: Electro-sensitive protective equipment. Part 1: General requirements and tests.
ISO 12100:2010	Safety of machinery - General principles for design - Risk assessment and risk reduction
EN 60204-1:2006	Safety of machinery - Electrical equipment of machines - Part 1: General requirements
ISO 14119:2013	Safety of machinery - Interlocking devices associated with guards - Principles for design and selection
ISO 13850:2015	Safety of machinery - Emergency stop - Principles for design
IEC 62061:2021	Safety of machinery - Functional safety of safety-related electrical, electronic, and electronic programmable control systems
IEC 61508-1:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: General requirements.
IEC 61508-2:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: Requirements for electrical/electronic/programmable electronic safety-related systems.
IEC 61508-3:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: Software requirements.
IEC 61784-3:2021	Industrial communication networks - Profiles - Part 3: Functional safety fieldbuses - General rules and profile definitions.
2006/42/EC	Machinery Directive
2014/30/EU	Electromagnetic Compatibility Directive
2014/35/EU	Low Voltage Directive

In addition, terms used in the present document may tangentially be used as they are derived from other standards such as:

Standard	Description
IEC 60034 series	Rotating electrical machines
IEC 61800 series	Adjustable speed electrical power drive systems
IEC 61158 series	Digital data communications for measurement and control – Fieldbus for use in industrial control systems

Finally, the term *zone of operation* may be used in conjunction with the description of specific hazards, and is defined as it is for a *hazard zone* or *danger zone* in the *Machinery Directive* (*2006/42/EC*) and *ISO 12100:2010*.

NOTE: The aforementioned standards may or may not apply to the specific products cited in the present documentation. For more information concerning the individual standards applicable to the products described herein, see the characteristics tables for those product references.

General Information

What’s in This Part

Presentation of the Library

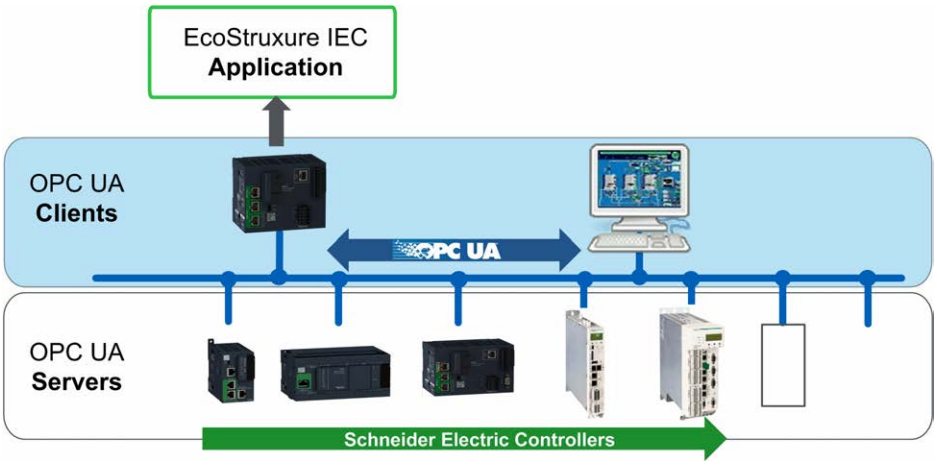
What's in This Chapter

General Information

Library Overview

The OpcUaHandling library provides you the function blocks to implement the OPC UA client functionality in your application. The OPC UA client itself is a component in the runtime system (controller firmware). The function blocks are applied to control and monitor the client from the application, and thus to implement the data exchange with the OPC UA server.

For more information about the OPC especially OPC UA, refer to the official webpage of the OPC Foundation at https://opcfoundation.org.



Also refer to *General Information on Function Blocks*, page 65.

Characteristics of the Library

The table indicates the characteristics of the library:

Characteristic	Value
Library title	OpcUaHandling
Company	Schneider Electric
Category	**Communication**
Component	**OPC UA Handling**
Default namespace	**SE_OPC**
Language model attribute	Qualified-access-only (see EcoStruxure Machine Expert, 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_OPC**.

Controller Platform

The OpcUaHandling library is supported by:

- Modicon M262 Logic/Motion Controllers (TM262L20MESE8T, TM262M25MESS8T, TM262M35MESS8T)
- PacDrive LMC controllers

Compatibility/Conformity

The function blocks and data types provided with the library are compliant with the PLCopen specification *PLCopen OPC-UA Client for IEC61131-3 version 1.1*.

Example Project

In conjunction with the library, an example project is provided. The example project demonstrates how to implement the components from the OpcUaHandling library.

The example project is installed on your PC along with the programming software. To open the project example, proceed as follows:

Step	Action	Comment
1	In the EcoStruxure Machine Expert Logic Builder, execute the command **New Project**.	–
2	In the **New Project** dialog box, select **From Example** from the **Project type** list.	–
3	On the right-hand side of the **New Project** dialog box, click the button **Toggle Filter**.	**Result**: Available examples are listed in the drop down menu.
4	Select your example from the drop down menu.	–
5	Select your controller from the **Controllers** list.	–
6	Enter a name for the new project, and select the file location.	–
7	Click the **OK** button.	**Result**: A new project is created based on the selected example.

General Considerations

⚠WARNING

EXCHANGED DATA INCOMPATIBILITY

Verify that the exchanged data are compatible because data structure alignments are not the same for all devices.

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

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.

For more information on organizational measures and rules covering access to infrastructures, refer to ISO/IEC 27000 series, Common Criteria for Information Technology Security Evaluation, ISO/IEC 15408, IEC 62351, ISA/IEC 62443, NIST Cybersecurity Framework, Information Security Forum - Standard of Good Practice for Information Security and refer to Cybersecurity Guidelines for EcoStruxure Machine Expert, Modicon and PacDrive Controllers and Associated Equipment.

Common Inputs and Outputs

What's in This Chapter

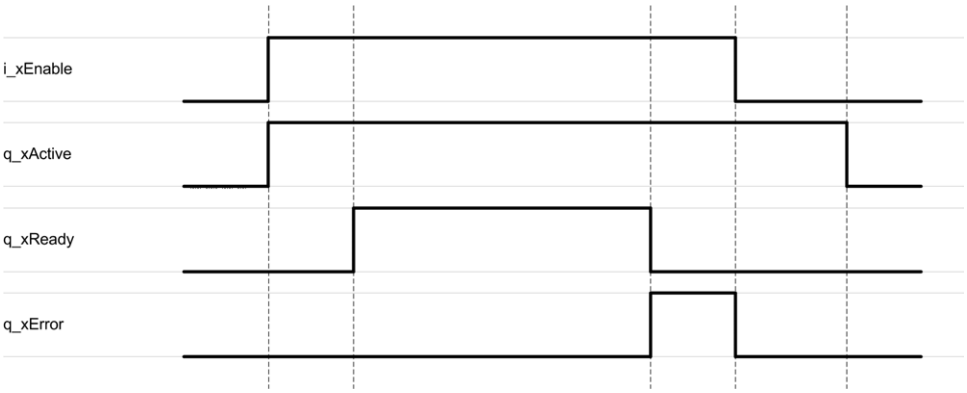
Behavior of Function Blocks with the Input *i_xEnable*

General Information

By setting the input *i_xEnable* to TRUE, the function block starts the enabling process. The function block continues initialization and the output *q_xActive* is set to TRUE. Once the initialization is finished, the output *q_xReady* is set to TRUE.

In case an error is detected, the output *q_xError* remain TRUE until the function block is disabled.

Example



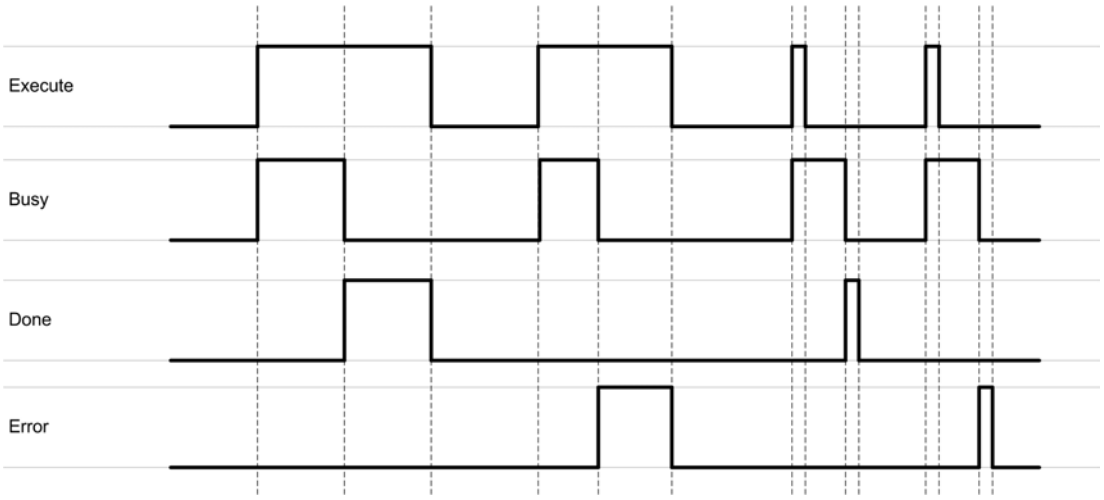
Behavior of Function Blocks with the Input *Execute*

General Information

A rising edge of the input *Execute* starts the execution of the function block. The function block continues execution and the output *Busy* is set to TRUE. Additional rising edges at the input *Execute* are ignored while the function block is being executed.

Once the execution is finished, the outputs *Done* or *Error* remain TRUE until the input *Execute* is set to FALSE. If the input is reset before the execution is finished, the outputs *Done* or *Error* are set to TRUE for one cycle.

Example



Data Unit Types

What's in This Part

Enumerations

What's in This Chapter

ET_ArrayType

Overview

Type:	Enumeration
Available as of:	V2.2.4.0

Description

The enumeration *ET_ArrayType* specifies the type of the variable used as argument in a method call.

Enumeration Elements

Name	Value	Description
UATypeScalar	16#0	The variable is of type scalar.
UATypeArray	16#1	The variable is of type array.
UATypeMatrix	16#2	The variable is of type matrix.

Used By

- *ST_Argument*

ET_Result

Overview

Type:	Enumeration
Available as of:	V1.0.0.0

Description

The enumeration *ET_Result* specifies the possible values that indicate the result of operations executed by the POUs of this library.

NOTE: In case the returned error ID is not part of the enumeration, refer to the PLCopen specification for more details (https://opcfoundation.org).

Enumeration Elements

Schneider Electric-specific values

Name	Value	Description
InternalError	16#B0000001	An internal error of the client is detected.
InvalidNodeHdlCount	16#B0000003	The value of *NodeHdlCount* is invalid. Valid range: *1..GPL.MAX_ELEMENTS_NODELIST*
InvalidServerEndpointUrl	16#B0000004	The specified *ServerEndpointUrl* must not be a null string.
InvalidNodeIDCount	16#B0000006	The value of *NodeIdCount* is invalid. Valid range: *1..GPL.MAX_ELEMENTS_NODELIST*
CorruptedResponse	16#B0000007	Invalid or incorrect response from the OPC UA server.
InvalidTransportProfile	16#B0000011	The value of *TransportProfile* is unsupported.
InvalidSessionTimeout	16#B0000012	The value of *SessionTimeout* is unsupported. Valid range: 30 s...1 d
InvalidMonitorConnection	16#B0000013	The value of *MonitorConnection* is unsupported. Valid range: 500 ms...10 min
InvalidUserIdentityTokenType	16#B0000014	The value of *UAUserIdentityTokenType* is unsupported.
InvalidIndexRangeCount	16#B0000016	The value of *IndexRangeCount* is unsupported. Valid range: 0...3
InvalidIndexRange	16#B0000017	The value of *StartIndex* must not be greater than the value of *EndIndex*.
InvalidIdentifierType	16#B0000018	The value of *IdentifierType* is unsupported.
InvalidIdentifier	16#B0000019	The specified *Identifier* must not be a null string.
DimensionsMismatch	16#B0000021	The specified data type dimensions of the node do not match the dimensions of the node on the OPC UA server.
InvalidBufferAddress	16#B0000022	The specified *BufferAddress* must not be 0.
BufferSizeMismatch	16#B0000024	The specified data size of the buffer does not match the data size of the targeted node on the server.
InvalidUserIdentityTokenParam	16#B0000025	The specified *UserIdentityTokenParam* is invalid. With *UserIdentityTokenType* = *UAUITT_Username*, none of the *ST_UserIdentyTokenParams* may be a null string.
InvalidSecurityPolicy	16#B0000026	The specified *InvalidSecurityPolicy* is invalid.
InvalidSecurityMsgMode	16#B0000027	The specified *InvalidSecurityMsgMode* is invalid.
InvalidAttributeID	16#B0000028	The specified *AttributeID* is unsupported.
InvalidNamespaceUrisCount	16#B0000029	The value of *NamespaceUrisCount* is invalid. Valid range: *1...GPL.MAX_ELEMENTS_NAMESPACES*
InvalidNamespaceUri	16#B0000030	The value of *NamespaceUri* must not be a null string.
InvalidBrowsePathCount	16#B0000031	The specified *BrowsePathCount* is invalid.
InvalidNumberOfElements	16#B0000032	The specified *NoOfElements* is invalid.
InvalidRelativePathTargetName	16#B0000033	The value of *Name* must not be a null string.
InvalidBrowseDescriptionDirection	16#B0000034	The specified *Direction* is invalid.
InvalidResultMask	16#B0000035	The specified *ResultMask* must not be 0.
InvalidMonitoredItemHdlCount	16#B0000037	The specified *MonitoredItemHdlCount* is invalid.
PublishingNotEnabledNotSupported	16#B0000038	The setting *PublishingEnabled* = FALSE is unsupported.
InvalidQueueSize	16#B0000039	The specified *QueueSize* is invalid.
InvalidDeadbandType	16#B0000040	The specified *DeadbandType* is invalid.

Name	Value	Description
IncompabileNodeDataTypeWithSync-Mode	16#B0000041	The specified *NodeDataType* is unsupported. The node data type using firmware synchronization must be *IecSymbol*.
InvalidVarInfo	16#B0000042	The specified variable information is invalid.
InvalidVarInfocount	16#B0000043	The specified number of variables is outside the valid range of 1...128.
VarNotFound	16#B0000044	The specified variable is not part of the symbol configuration or is not accessible. If symbol sets are enabled, verify the user name and the password combination.
InitAsyncManagerFailed	16#B0000045	The initialization of the asynchronous manager is unsuccessful.
AsyncManagerNotInitialized	16#B0000046	The asynchronous manager has not been initialized. Initialize the asynchronous manager inside your application or set *i_InitAsyncManager* to TRUE.
AuthenticationFailed	16#B0000047	Unsuccessful authentication for accessing variables protected by the symbol sets.
InvalidNumberOfArguments	16#B0000048	The specified number of arguments is invalid.
IecTimeout	16#B0000099	The timeout has elapsed while waiting for a response from the client (firmware component).
Timeout	16#B0000100	The timeout has elapsed while waiting for a response from the OPC UA server.
DataTypeMismatch	16#B0000102	The specified *DataType* of the node does not match the data type of the node on the OPC UA server.
BufferSizeTooSmall	16#B0000103	The specified *BufferSize* is too small.
NotEnoughMemory	16#B0000104	The controller does not provide enough memory to process the received data.
OnlineChangeDetected	16#B0000105	After an online change was detected, the OPC UA client was reinitialized and the function block execution was aborted.
SessionNodeHdlAlreadyExist	16#B0000106	Node handles already retrieved for this session. Release already retrieved node handles before getting new ones.
BrowseFailed	16#B0000107	The browse operation was unsuccessful because the client is not connected.
TranslatePathListFailed	16#B0000108	The translate path list operation was unsuccessful because the client is not connected.
SubscriptionExists	16#B0000109	The requested subscription already exists.
SubscriptionNotExists	16#B000010A	The given subscription handle does not exist.
MonitoredItemMaxReached	16#B000010B	The maximum number of monitored items (100) has been reached.
MonitoredItemsNotExists	16#B000010C	The specified *MonitoredItemHdl* does not exist.
IncompatibleDataTypeOfIecSymbol	16#B000010D	The data type of the variable referenced by *IecSymbol* is unsupported. The data type must be a base data type with a maximum size of 8 bytes.
MaxBrowseResultExceeded	16#B000010E	The OPC UA server is returning too many browse results.
StackNotEnabled	16#B000010F	The OPC UA stack has not been enabled. Verify that you have enabled the OPC UA server in EcoStruxure Machine Expert.
MonitoredItemIdAlreadyUsed	16#B0000110	An item with the same ID is already available in the list.
MonitoredItemIdInvalid	16#B0000111	The ID of the monitored item must not be 0.
QueueSizeInvalid	16#B0000112	The *QueueSize* must not be 0.
AnotherInstanceActive	16#B0000113	Only one instance of the function block *FB_TimeStamper* must be active.
InvalidContinuationPointIn	16#B0000114	An invalid continuation point is specified at the function block input.
MaxContinuationPointStorageReached	16#B0000115	The maximum number of stored continuation points has been reached.
InvalidMatrixDimension	16#B0000116	The dimension of the matrix must be greater than 0.
InvalidNumberOfMatrixDimensions	16#B0000117	The number of matrix dimensions must be 2 or 3.

Name	Value	Description
InvalidArrayLength	16#B0000118	The length of the array must be greater than 0.
InvalidArrayType	16#B0000119	The specified array type is invalid.
NotEnoughMethodHandles	16#B000011B	The maximum number of method handles has been exceeded. Release method handles before getting new ones.
InvalidByteString	16#B000011A	The specified byte string is invalid. *pbyBuffer* and *udiBufferSize* must be greater than 0.
NumberOfOutputArgumentsMismatch	16#B000011C	The number of output arguments returned by the server does not match the number of outputs specified at the function block interface.
InvalidMethodHandle	16#B000011D	An invalid method handle has been specified.
OpcUa_API_BadSignatureInvalid	16#A0010000	The message signature is invalid. Verify the security settings.
OpcUa_API_ BadExtensibleParameterInvalid	16#A0040000	The extensible parameter provided is invalid for the service. Re-execute the function block.
OpcUa_API_ BadExtensibleParameterUnsupported	16#A0050000	The server does not support the parameter provided. Verify the function block inputs.
OpcUa_API_BadHostUnknown	16#A0060000	The host name could not be resolved.
OpcUa_API_BadTooManyPosts	16#A0070000	Too many posts have been made to a semaphore. Re-execute the function block.
OpcUa_API_BadSecurityConfig	16#A0080000	The security configuration is invalid. Verify that client and server certificates are valid.
OpcUa_API_BadFileNotFound	16#A0090000	The specified file name is invalid. Verify that the certificate file (PEM, CRL, DER, key file, certificate chain) has a valid name.
OpcUa_API_BadContinue	16#A00A0000	Accept invalid result and continue.
OpcUa_API_BadHttpMethodNotAllowed	16#A00B0000	An invalid http method is used in the server endpoint call.
OpcUa_API_BadFileExists	16#A00C0000	The log file or the log directory already exists.
OpcUa_API_BadCryptoKeyTooShort	16#A0120000	The key received for the secured channel is too short.
OpcUa_API_BadCryptoKeyTooLong	16#A0130000	The key received for the secured channel is too long.
OpcUa_API_ BadSignatureAlgorithmNotAllowed	16#A0140000	The certificate signature algorithm is not allowed.
OpcUa_API_ BadWrongReceiverCertificate	16#A0150000	The sender provided an incorrect receiver certificate thumbprint.
OpcUa_API_BadMessageTooLarge	16#A0160000	The message is too large and cannot be sent.

For more information on PLCopen-specific values, refer to the PLCopen webpage (https://www.plcopen.org).

For more information on the OPC UA diagnostic codes, refer to the OPC Foundation webpage (https://opcfoundation.org).

Used By

- Used by all function blocks.

ET_TimeStampVarType

Overview

Type:	Enumeration
Available as of:	V2.1.6.0

Description

The enumeration *ET_TimeStampVarType* specifies the variable types supported by the function block *FB_TimeStamper*.

Enumeration Elements

Name	Value	Description
TSTypeNotSet	0	-
TsTypeBool	1	BOOL
TSTypeSInt	2	SINT
TSTypeUSInt	3	USINT
TSTypeInt	4	INT
TSTypeUInt	5	UINT
TSTypeDInt	6	DINT
TSTypeUDInt	7	UDINT
TSTypeLInt	8	LINT
TSTypeULInt	9	ULINT
TSTypeReal	10	REAL
TSTypeLReal	11	LREAL
TSTypeWord	14	WORD
TSTypeLWord	15	LWORD
TSTypeDWord	16	DWORD

Used By

- *FB_TimeStamper*

ET_VarType

Overview

Type:	Enumeration
Available as of:	V1.0.0.0

Description

The enumeration *ET_VarType* specifies the OPC variable types.

Enumeration Elements

Name	Value	Description **This OPC variable types correspond to the IEC 61131-3 data types:**
UATypeNull	0	-
UATypeBool	1	BOOL
UATypeSByte	2	SINT
UATypeByte	3	USINT
UATypeInt16	4	INT
UATypeUInt16	5	UINT
UATypeInt32	6	DINT
UATypeUInt32	7	UDINT
UATypeInt64	8	LINT
UATypeUInt64	9	ULINT
UATypeFloat	10	REAL
UATypeDouble	11	LREAL
UATypeString	12	STRING (Maximum length of string equals to 255 bytes).
UATypeDateTime	13	DT
UATypeByteString	17	ARRAY OF BYTE (Exclusive to *UA_MethodCall*, page 76.)
UATypeLocalizedText	21	The type *UATypeLocalizedText* corresponds to the data type *UALocalizedText*. The variable described using this type must be of data type *UALocalizedText*, page 45. **NOTE:** Variables with data type *ET_VarType.UATypeLocalizedText* are exclusive to Modicon M262 Logic/Motion Controllers with firmware version V5.2.8.27 or greater.
UATypeSubString	98	STRING (Maximum length of string equals to 255 bytes).
UATypeIECSymbol	99	The value *UATypeIECSymbol* allows you to configure the client by using the **Symbol configuration**: • Publish the buffer allocated in the application in the **Symbol configuration**. • Address a string containing the buffer name at *pbyBuffer* and specify the size at *udiBufferSize*. For a code example, refer to *ST_Variable*, Code Example, page 41.

Used By

- *ST_Variable*

UAAttributeID

Overview

Type:	Enumeration
Available as of:	V1.0.0.0

Description

The enumeration *UAAttributeID* specifies theOPC attributes of a node.

Enumeration Elements

Name	Value	Description
UAAI_NodeID	1	Approved identifier for the node.
UAAI_NodeClass	2	Class of the node.
UAAI_BrowseName	3	Non-localized name for the node.
UAAI_DisplayName	4	Localized name for the node.
UAAI_Description	5	Localized description for the node.
UAAI_WriteMask	6	Indicates which attributes can be written.
UAAI_UserWriteMask	7	Indicates which attributes can be written manually.
UAAI_IsAbstract	8	Indicates that a node type may not be instantiated.
UAAI_Symmetric	9	Indicates that forward and inverse references have the same meaning.
UAAI_InverseName	10	Browse name for an inverse reference.
UAAI_ContainsNoLoops	11	Indicates that following forward references within a view will not cause a loop.
UAAI_EventNotifier	12	Indicates that the node can be used to subscribe to events.
UAAI_Value	13	Value of a variable.
UAAI_DataType	14	Node Id of the data type for the variable value.
UAAI_ValueRank	15	Number of dimensions in the value.
UAAI_ArrayDimensions	16	Length for each dimension of an array value.
UAAI_AccessLevel	17	Indicates how a variable can be accessed.
UAAI_UserAccessLevel	18	Indicates how a variable can be accessed after taking the access rights of the user into account.
UAAI_MinimumSamplingInterval	19	Specifies how fast the server can sample the value for modifications [ms].
UAAI_Historizing	20	Specifies whether the server is actively collecting data for the variable.
UAAI_Executable	21	Specifies whether the method can be called.
UAAI_UserExecutable	22	Specifies whether the method can be called manually.

Used By

- *UANodeAdditionalInfo*

UABrowseDirection

Overview

Type:	Enumeration
Available as of:	V2.0.0.0

Description

The enumeration *UABrowseDirection* specifies the direction of references to return.

Enumeration Elements

Name	Value	Description
UABD_Forward	0	Select forward references.
UABD_Inverse	1	Select inverse references.
UABD_Both	2	Select forward and inverse references.

Used By

- *UABrowseDescription*

UAConnectionStatus

Overview

Type:	Enumeration
Available as of:	V1.0.0.0

Description

The enumeration *UAConnectionStatus* specifies the connection state between the OPC UA server and the OPC UA client.

Enumeration Elements

Name	Value	Description
UACS_Connected	0	OPC UA client is connected to OPC UA server.
UACS_ConnectionError	1	For the connection from the OPC UA client to the OPC UA server, an error was detected.
UACS_Shutdown	2	The OPC UA client was disconnected from the OPC UA server.

Used By

- *UA_ConnectionGetStatus*

UAIdentifierType

Overview

Type:	Enumeration
Available as of:	V1.0.0.0

Description

The enumeration *UAIdentifierType* specifies the types of the OPC UA identifier.

Enumeration Elements

Name	Value	Description
UAIT_Numeric	0	Identifier specified with a numeric value.
UAIT_String	1	Identifier specified with a string value.
UAIT_GUID	2	Identifier specified with a GUID (Globally Unique Identifier) value.
UAIT_Opaque	3	Not supported.

Used By

- *UANodeID*

UAMonitoringSyncMode

Overview

Type:	Enumeration
Available as of:	V2.0.0.0

Description

The enumeration *UAMonitoringSyncMode* specifies the synchronization mode.

Enumeration Elements

Name	Value	Description
UAMS_Unknown	0	Unknown synchronization mode.
UAMS_ControllerSync	1	Controller synchronization mode.
UAMS_FwSync	2	Firmware synchronization mode.

Used By

- *UA_MonitoredItemAddList*

UASecurityMsgMode

Overview

Type:	Enumeration
Available as of:	V1.0.0.0

Description

The enumeration *UASecurityMsgMode* specifies the message security mode applied to the connection to the OPC UA server.

Enumeration Elements

Name	Value	Description
UASMM_BestAvailable	0	Best available message security mode to the OPC UA server. The client receives the available message security modes from the server and selects the best. Note that this could also result in level *UASMM_None*.
UASMM_None	1	No security is applied.
UASMM_Sign	2	Messages are signed but not encrypted.
UASMM_SignEncrypt	3	Messages are signed and encrypted.

Used By

- *UASessionConnectInfo*

UASecurityPolicy

Overview

Type:	Enumeration
Available as of:	V1.0.0.0

Description

The enumeration *UASecurityPolicy* specifies the security profile applied to the connection to the OPC UA server.

Enumeration Elements

Name	Value	Description
UASP_BestAvailable	0	Best available security connection to the OPC UA server. The client receives the available policies from the server and selects the best. Note that this can also result in level *UASP_None*.
UASP_None	1	No security profile is applied.
UASP_Basic128Rsal5	2	Basic128Rsa15 profile.
UASP_Basic256	3	Basic256 profile.
UASP_Basic256Sha256	4	Basic256Sha256 profile.

Used By

- *UASessionConnectInfo*

UAServerState

Overview

Type:	Enumeration
Available as of:	V1.0.0.0

Description

The enumeration *UAServerState* specifies the present OPC UA server state.

Enumeration Elements

Name	Value	Description
UASS_Running	0	The server is in an operational state.
UASS_Failed	1	The server is in a non-operational state.
UASS_NoConfiguraiton	2	The server is running but has no configuration information loaded and therefore does not transfer data.
UASS_Suspended	3	The server has been temporarily suspended and is not receiving or sending data.
UASS_Shutdown	4	The server is stopped or is in the process of stopping.
UASS_Test	5	The server is in test mode.
UASS_CommunicatonFault	6	The server is running properly but cannot access data from its data sources.
UASS_Unknown	7	This state is used only to indicate that the OPC UA server gets no information on the state underlaying servers.

Used By

- *UA_ConnectionGetStatus*

UATransportProfile

Overview

Type:	Enumeration
Available as of:	V1.0.0.0

Description

The enumeration *UATransportProfile* specifies the transport profile applied to the connection to the OPC UA server.

Enumeration Elements

Name	Value	Description
UATP_UATcp	1	TCP profile.
UATP_WSHttpBinary	2	Not supported.
UATP_WSHttpXmlOrBinary	3	Not supported.
UATP_WSHttpXml	4	Not supported.

Used By

- *UASessionConnectInfo*

UAUserIdentityTokenType

Overview

Type:	Enumeration
Available as of:	V1.0.0.0

Description

The enumeration *UAUserIdentityTokenType* specifies the type of authentication.

Enumeration Elements

Name	Value	Description
UAUITT_Anonymous	0	No authentication is applied.
UAUITT_Username	1	Authentication by user name and password.

Name	Value	Description
UAUITT_x509	2	Authentication by X.509 certificate. For further information, see Authentication by X.509 Certificates, page 35. Exclusive to Modicon M262 Logic/Motion Controllers with firmware version V5.2.8.27 or greater.
UAUITT_IssuedToken	3	Not supported.

Authentication by X.509 Certificates

For the authentication using a client certificate, the following preconditions need to be fulfilled:

- The client certificate is available in the certificate store of the controller in the folder **Own Certificates**(1).

  NOTE: An externally created certificate can be downloaded as PKCS12 container file to the controller from the **Security Screen** editor.

- The **Common name** of the certificate is assigned to the element *TokenParam2* of the structure *UAUserIdentityToken*, page 56.

(1) The certificate store of the controller can be accessed using the **Security Screen** editor in EcoStruxure Machine Expert (refer to the **Software > Programming and Configuring > CODESYS Security Agent** part of the EcoStruxure Machine Expert online help).

Used By

- *UAUserIdentityToken*

Structures

What's in This Chapter

ST_Argument

Overview

Type:	Structure
Available as of:	V2.2.4.0
Inherits from:	-
Attribute:	`'pack_mode' := '1'`

Description

The structure *ST_Argument* contains the information about an argument used in a method call.

Structure Elements

Name	Data type	Description
etNodeDataType	ET_VarType, page 27	Data type of the node. **NOTE:** This value must match the data type definition of the node on the server. In case of an array, specify the base data type of the array.
etArrayType	*ET_ArrayType*, page 22	Defines the type of the node: scalar value, array, or matrix.
udiArrayLength	UDINT	Defines the number of elements of the array. The parameter is considered if *etArrayType* is equal to *UATypeArray*.
udiNumberOfMatrixDimensions	UDINT	Defines the number of dimensions. The parameter is considered if *etArrayType* is equal to *UATypeMatrix*.
audiMatrixDimensions	ARRAY [1..3] OF UDINT	Defines the number of elements of every dimension. The parameter is considered if *etArrayType* is equal to *UATypeMatrix*.
pbyBuffer	POINTER TO BYTE	Address of the buffer allocated in the application.
udiBufferSize	UDINT	Size of the buffer allocated in your application. **NOTE:** In case the value of *etNodeDataType* indicates *UATypeString*, page 28 and *UANodeAdditionalInfo*, page 47 an array or matrix, assign the size of the array or matrix of the application to this parameter. This value has to match the node definition on the server. **NOTE:** Use the `SIZEOF` operator to get the size of the variable inside the application.
etErrorID	*ET_Result*, page 22	Indicates diagnostic information as a numeric value.

ST_Arguments

Overview

Type:	Structure
Available as of:	V2.2.4.0
Inherits from:	-
Attribute:	`'pack_mode' := '1'`

Description

The structure *ST_Arguments* contains the list of arguments used in a method call.

Structure Elements

Name	Data type	Description
uiNumberOfArguments	UINT	The number of arguments in the *astArguments* array. It must not exceed the value of *GPL.Gc_uiMaxArgumentsMethod*, page 63.
astArguments	ARRAY [1..*GPL.Gc_uiMaxArgumentsMethod*] OF *ST_Argument*	Array containing the arguments.

ST_ByteString

Overview

Type:	Structure
Available as of:	V2.2.4.0
Inherits from:	-
Attribute:	`'pack_mode' := '1'`

Description

The structure *ST_ByteString* is used to assign the variable specified inside your application to an argument of type ByteString of a method.

Structure Elements

Name	Data type	Description
pbyBuffer	POINTER TO BYTE	Address of the buffer allocated in the application.
udiBufferSize	UDINT	Size of the buffer allocated in your application. **NOTE:** Use the `SIZEOF` operator to get the size of the variable inside the application.

ST_Credentials

Overview

Type:	Structure
Available as of:	V2.1.6.0
Inherits from:	-
Attribute:	`'pack_mode' := '1'`

Description

The structure *ST_Credentials* contains the credentials required to access variables inside the symbol configuration in case symbol sets are enabled.

Structure Elements

Name	Data type	Description
sUsername	STRING [255]	The user name for accessing variables if symbol sets are enabled.
sPassword	STRING [255]	The password for accessing variables if symbol sets are enabled.

ST_TimeStampResult

Overview

Type:	Structure
Available as of:	V2.1.6.0
Inherits from:	-
Attribute:	`'pack_mode' := '1'`

Description

The structure *ST_TimeStampResult* is used to provide the information about the monitored variable including the value and timestamp.

Structure Elements

Name	Data type	Description
sVarName	STRING [*GPL.Gc_uiMaxVarNameLengthTimeStamper*]	Name of the variable.
etTimeStampVarType	*ET_TimeStampVarType*	Data type of the variable.
pbyVarAddress	POINTER TO BYTE	Address of the variable located in the application.
lwValue	LWORD	Value of the variable.
ldtTimeStamp	LDT	Timestamp of the last value change.

ST_Variable

Overview

Type:	Structure
Available as of:	V1.0.0.0
Inherits from:	-
Attribute:	`'pack_mode' := '1'`

Description

The structure *ST_Variable* is used to provide the information about the variable and the memory area.

Structure Elements

Name	Data type	Description
etNodeDataType	ET_VarType, page 27	Data type of the node. **NOTE:** This value must match the data type definition of the node on the server. In case of an array, specify the base data type of the array.
pbyBuffer	POINTER TO BYTE	Address of the buffer allocated in the application.
udiBufferSize	UDINT	Size of the buffer allocated in your application. **NOTE:** In case the value of *etNodeDataType* indicates *UATypeString*, page 28 and *UANodeAdditionalInfo*, page 47 indicates an array or matrix, assign the size of the array or matrix of the application to the *udiBufferSize* parameter. This value has to match the node definition on the server. **NOTE:** Use the `SIZEOF` operator to get the size of the variable inside the application.

NOTE: The value *UATypeIECSymbol* allows the client configuration through the symbol configuration:

- Publish the buffer allocated in the application inside the symbol configuration.
- Address a string containing the buffer name at *pbyBuffer* and specify its size at *udiBufferSize*.

Code Example

The code example indicates how to configure the function block *UA_MonitoredItemAddList* for synchronization mode (input *SyncMode*) *UAMonitoringSyncMode.UAMS_FwSync*.

```
VAR
    fbUaMonitoredItemAddList     : SE_Opc.UA_MonitoredItemAddList;

    adwNodeHdls                  : ARRAY [1..2] OF DWORD; //Node handles got from
UA_NodeGetHandleList
    sSymbolReadSetVelocity       : STRING := 'Application.GVL_Client.G_
uiReadSetVelocity'; //Symbol of the variable published in the Symbol configuration
    sSymbolReadVelocity          : STRING := 'Application.GVL_Client.G_
uiReadVelocity'; //Symbol of the variable published in the Symbol configuration

    dwSubscriptionHdl            : DWORD; //Subscription handle got from UA_
SubscriptionCreate
    astSubVariables              : ARRAY[1..SE_OPC.GPL.MAX_ELEMENTS_MONITORLIST]
OF SE_OPC.UAMonitoredVariables;
    astMonitoringParameters      : ARRAY[1..SE_OPC.GPL.MAX_ELEMENTS_MONITORLIST]
OF SE_OPC.UAMonitoringParameter;
    axValuesChanged              : ARRAY[1..SE_OPC.GPL.MAX_ELEMENTS_MONITORLIST]
OF BOOL;
    auiMinLostValueCount         : ARRAY[1..SE_OPC.GPL.MAX_ELEMENTS_MONITORLIST]
OF UINT;
END_VAR
```

```
//Node description
fbUaMonitoredItemAddList.NodeHdls[1]                := adwNodeHdls[1];
astSubVariables[1].Values[1].etNodeDataType         := SE_Opc.ET_VarType.
UATypeIECSymbol;
astSubVariables[1].Values[1].pbyBuffer              := ADR(sSymbolReadSetVelocity);
astSubVariables[1].Values[1].udiBufferSize          := SIZEOF
(sSymbolReadSetVelocity);
astMonitoringParameters[1].SamplingTime             := T#1S;

fbUaMonitoredItemAddList.NodeHdls[2]                := adwNodeHdls[2];
astSubVariables[2].Values[1].etNodeDataType         := SE_Opc.ET_VarType.
UATypeIECSymbol;
astSubVariables[2].Values[1].pbyBuffer              := ADR(sSymbolReadVelocity);
astSubVariables[2].Values[1].udiBufferSize          := SIZEOF(sSymbolReadVelocity);
astMonitoringParameters[2].SamplingTime             := T#1S;

//Function block call
fbUaMonitoredItemAddList(    Execute                := TRUE,
                             SubscriptionHdl        := dwSubscriptionHdl,
                             NodeHdlCount           := 2,
                             SyncMode               := SE_Opc.UAMonitoringSyncMode.
UAMS_FwSync,
                             Variables              := astSubVariables,
                             MonitoringParameters   := astMonitoringParameters,
                             ValuesChanged          := axValuesChanged,
                             MinLostValueCount      := auiMinLostValueCount);
```

Used By

- *UA_ReadList*
- *UA_WriteList*

ST_VarInfo

Overview

Type:	Structure
Available as of:	V2.1.6.0
Inherits from:	-
Attribute:	`'pack_mode' := '1'`

Description

The structure *ST_VarInfo* is used to provide the information about the variable that is time-stamped.

Structure Elements

Name	Data type	Description
sVarName	STRING [*GPL.Gc_uiMaxVarNameLengthTimeStamper*]	Name of the variable.
etTimeStampVarType	*ET_TimeStampVarType*	Data type of the variable.
pbyVarAddress	POINTER TO BYTE	Address of the variable located in the application.

UABrowseDescription

Overview

Type:	Structure
Available as of:	V2.0.0.0
Inherits from:	-
Attribute:	`'pack_mode' := '1'`

Description

The structure *UABrowseDescription* is used to provide information required for browsing.

Structure Elements

Name	Data type	Description
StartingNodeID	UANodeID, page 49	Identifier for the starting node for browsing.
Direction	*UABrowseDirection*, page 29	Defines the browse direction.
ReferenceTypeID	UANodeID, page 49	Identifier for the reference type the server should follow.
IncludeSubtypes	BOOL	Defines whether subtypes of the reference type should be returned.
NodeClass	UANodeClassMask, page 58	Node class of a returned node such as object, variable, or method.
ResultMask	*UABrowseResultMask*, page 57	Defines the fields of the *UAReferenceDescription*, page 52 that are requested from the server.

Used By

- *UA_Browse*
- *UANodeID*

UABrowsePath

Overview

Type:	Structure
Available as of:	V2.0.0.0
Inherits from:	-
Attribute:	`'pack_mode' := '1'`

Description

The structure *UABrowsePath* is used to provide information to specify nodes using relative paths.

Structure Elements

Name	Data type	Description
StartingNode	UANodeID, page 49	Identifier for the starting node from which the relative path will be evaluated.
RelativePath	*UARelativePath*, page 52	Defines the relative path which will be evaluated.

Used By

- *UA_TranslatePathList*

UAExpandedNodeID

Overview

Type:	Structure
Available as of:	V1.0.0.0
Inherits from:	-
Attribute:	`'pack_mode' := '1'`

Description

The structure *UAExpandedNodeID* is used to provide the extended information about the node.

Structure Elements

Name	Data type	Description
ServerIndex	UDINT	Server index formatted as a base 10 number.
NamespaceUri	STRING[255]	Namespace URI formatted as a string. Any reserved characters in the URI are replaced with a `%` followed by its 8-bit ANSI value encoded as two hexadecimal digits.
ID	UANodeID, page 49	Identifier for a node in the address space of an OPC UA server.

Used By

- *UAReferenceDescription*

UAIndexRange

Overview

Type:	Structure
Available as of:	V1.0.0.0
Inherits from:	-
Attribute:	`'pack_mode' := '1'`

Description

The structure *UAIndexRange* is used to provide the information about the index range of a dimension of a variable published by the OPC UA server.

Structure Elements

Name	Data type	Description
StartIndex	UINT	Start index (first element has index value 0).
EndIndex	UINT	End index.

NOTE: For each dimension:

1. *StartIndex* and *EndIndex* must be assigned.
2. The value for *StartIndex* must be less than the value for *EndIndex*.
3. To access all the elements in a dimension, you must assign *StartIndex* and *EndIndex* depending on the total number of elements in the dimension.
4. A single element in a dimension can be selected by specifying the same *StartIndex* and *EndIndex*.

Used By

- *UANodeAdditionalInfo*

UALocalizedText

Overview

Type:	Structure
Available as of:	V1.0.0.0
Inherits from:	-
Attribute:	-

Description

The structure *UALocalizedText* corresponds to the built-in data type *LocalizedText* as defined in the specification *OPC 10000-3: UA Part 3: Address Space Model*, page 10.

Structure Elements

Name	Data type	Description
Locale	STRING [6]	Language and country/region identifier. Format: `<language>-<country/region>` Example: `en-US; zh-CHS`
Text	STRING [255]	Indicates localized text as a string.

Used By

- *UAReferenceDescription*
- *UANodeInformation*

UAMonitoredVariables

Overview

Type:	Structure
Available as of:	V2.0.0.0
Inherits from:	-
Attribute:	`'pack_mode' := '1'`

Description

The structure *UAMonitoredVariables* is used to provide information about monitored variables.

Structure Elements

Name	Data type	Description
Values	ARRAY [1..*GPL.MAX_ELEMENTS_QUEUELIST*] OF *ST_Variable*, page 39	Array containing information about the variables and the corresponding memory areas.
TimeStamps	ARRAY [1..*GPL.MAX_ELEMENTS_QUEUELIST*] OF DT	Source time stamp returned by the server.
NodeQualityIDs	ARRAY [1..*GPL.MAX_ELEMENTS_QUEUELIST*] OF DWORD	Contains an error code for each valid element of the *Variable* array.
NewValuesCount	UINT	Counts the values which where updated starting from the lowest element of the values.

Used By

- *UA_MonitoredItemAddList*

UAMonitoringParameter

Overview

Type:	Structure
Available as of:	V2.0.0.0
Inherits from:	-
Attribute:	`'pack_mode' := '1'`

Description

The structure *UAMonitoringParameter* provides parameters to configure the monitoring.

Structure Elements

Name	Data type	Description
SamplingInterval	TIME	The interval (in milliseconds) the server polls the underlying data source for changes.
QueueSize	UINT	The queue size for the monitoring item. Default value: 1
DiscardOldest	BOOL	Determines the discard policy in case of queue overflow.
DeadbandType	*UADeadbandType*	Indicates the type of deadband that is applied, if any.
Deadband	REAL	The deadband value (in %).

Used By

- *UA_MonitoredItemAddList*
- *UA_MonitoredItemModifyList*
- *UA_MonitoredItemOperateList*
- *UA_MonitoredVariables*

UANodeAdditionalInfo

Overview

Type:	Structure
Available as of:	V1.0.0.0
Inherits from:	-
Attribute:	`'pack_mode' := '1'`

Description

The structure *UANodeAdditionalInfo* is used to provide additional information about the node.

Structure Elements

Name	Data type	Description
AttributeID	UAAttributeID, page 29	Attribute to be accessed. Default value: *UAAI_Value*
IndexRangeCount	UINT	Number of valid index ranges specified. Each index range specifies one dimension. **NOTE:** An elementary data type element is defined by *IndexRangeCount* = 0. Value range: 0...*GPL.MAX_ELEMENTS_INDEXRANGE* **NOTE:** If the value of *etNodeDataType* indicates *UATypeIECSymbol*, then *IndexRangeCount* 0 defines the index ranges according to the variable declaration inside the symbol configuration. **NOTE:** If the value of *etNodeDataType* indicates *UATypeByte*, then *IndexRangeCount* 0 defines the index ranges according to the variable declaration inside the server configuration.
IndexRange	ARRAY [1..GPL.MAX_ELEMENTS_INDEXRANGE] OF UAIndexRange, page 45	List of index ranges. **NOTE:** For *ET_VarType UATypeSubString*, the last *IndexRange* specifies the start and end index of the characters to read from each element.

Used By

- *UA_ReadList*
- *UA_WriteList*

UANodeID

Overview

Type:	Structure
Available as of:	V1.0.0.0
Inherits from:	-
Attribute:	`'pack_mode' := '1'`

Description

The structure *UANodeID* is used to provide the information to identify the target node on the OPC UA server.

Structure Elements

Name	Data type	Description
NamespaceIndex	UINT	Name space of the OPC UA server.
Identifier	STRING [255]	Identifier. A null string is not allowed.
IdentifierType	UAIdentifierType, page 31	Type of identifier. Default value: *UAAI_String* Only *UAAI_Numeric* and *UAAI_String* are supported.

Used By

- *UA_NodeGetHandleList*
- *UA_NodeGetInformation*
- *UAReferenceDescription*
- *UAExpandedNodeID*
- *UABrowseDescription*
- *UANodeInformation*

UANodeInformation

Overview

Type:	Structure
Available as of:	V1.0.0.0
Inherits from:	-
Attribute:	`'pack_mode' := '1'`

Description

The structure *UANodeInformation* is used to provide information about the node.

Structure Elements

Name	Data type	Description		
AccessLevel	BYTE	A bitmask indicating whether the current value of the value *Attribute* is readable and writable as well as whether the previous values are readable and can be modified:		
		Bit	**Value**	**Access level**
		-	16#0	*None*
		0	16#1	*CurrentRead*
		1	16#2	*CurrentWrite*
		2	16#4	*HistoryRead*
		3	16#8	*HistoryWrite*
ArrayDimension	ARRAY [1..GPL.MAX_ELEMENTS_ARRAYDIMENSION] OF UDINT	Length for each dimension of an array value.		
BrowseName	UAQualifiedName, page 51	Element name.		
ContainsNoLoops	BOOL	Indicates that following forward references within a view will not cause a loop.		
DataType	UANodeID, page 49	Node Id of the data type for the variable value.		
Description	UALocalizedText, page 45	Localized description for the node.		
DisplayName	UALocalizedText, page 45	Localized name for the node.		
EventNotifier	BYTE	This attribute represents a bitmask that identifies whether the object can be used to subscribe to events and whether the previous events are accessible and can be modified:		
		Bit	**Value**	**EventNotifier**
		-	16#0	The object does not produce events and has no event history (*SubscribeToEvents*)
		0	16#1	*CurrentRead*
		1	16#2	*CurrentWrite*
		2	16#4	*HistoryRead*
		3	16#8	*HistoryWrite*
Executable	BOOL	Indicates whether the method can be called.		
Historizing	BOOL	Indicates whether the server is actively collecting previous data for the variable.		
InverseName	STRING [255]	Browse name for an inverse reference.		
IsAbstract	BOOL	Indicates that a type node may not be instantiated.		
MinimumSamplingInterval	TIME	Indicates how fast the server can sample the value for modifications.		
NodeClass	UANodeClassMask, page 58	Node class of a node such as object, variable, or method.		
Symmetric	BOOL	Indicates that forward and inverse references have the same meaning.		

Name	Data type	Description		
UserAccessLevel	BYTE	Same information as the *AccessLevel* but takes user access rights into account:		
		Bit	**Value**	**Access level**
		-	16#0	*None*
		0	16#1	*CurrentRead*
		1	16#2	*CurrentWrite*
		2	16#4	*HistoryRead*
		3	16#8	*HistoryWrite*
UserExecutable	BOOL	Indicates whether the method can be called manually.		
UserWriteMask	UDINT	Indicates which attributes can be written manually.		
ValueRank	DINT	Number of dimensions in the value.		
WriteMask	UDINT	Indicates which attributes can be written.		

Used By

- *UA_NodeGetInformation*

UAQualifiedName

Overview

Type:	Structure
Available as of:	V1.0.0.0
Inherits from:	-
Attribute:	`'pack_mode' := '1'`

Description

The structure *UAQualifiedName* is used to provide the information about the name and the corresponding namespace index.

Structure Elements

Name	Data type	Description
NamespaceIndex	UINT	The namespace index where *Name* is provided.
Name	STRING [255]	Name of qualified name.

Used By

- *UANodeInformation*

UAReferenceDescription

Overview

Type:	Structure
Available as of:	V2.0.0.0
Inherits from:	-
Attribute:	`'pack_mode' := '1'`

Description

The structure *UAReferenceDescription* provides information about references and target nodes.

Structure Element

Name	Data type	Description
ReferenceTypeID	UANodeID, page 49	Identifier for the reference type from the starting node to the target node.
IsForward	BOOL	Indicates whether a forward reference is followed.
NodeID	*UAExpandedNodeID*, page 44	Indicates the node ID of the target node. This can also be a node in another server.
BrowseName	STRING[255]	Indicates the qualified name of the target node.
DisplayName	*UALocalizedText*, page 45	Indicates the localized name of the target node.
NodeClass	UANodeClassMask, page 58	Node class of a returned node such as object, variable, or method.
TypeDefinition	*UAExpandedNodeID*, page 44	Indicates the node ID of the object or variable type of the target node.

Used By

- *UA_Browse*
- *UANodeID*

UARelativePath

Overview

Type:	Structure
Available as of:	V2.0.0.0
Inherits from:	-
Attribute:	`'pack_mode' := '1'`

Description

The structure *UARelativePath* is used to provide information about a list of relative path elements.

Structure Elements

Name	Data type	Description
NoOfElements	UINT	Defines the valid number of elements in the *Elements* array.
Elements	ARRAY [1..*GPL. MAX_ ELEMENTS_ RELATIVEPATH*] OF *UARelativePathElement*	Array containing relative path elements.

Used By

- *UABrowsePath*

UARelativePathElement

Overview

Type:	Structure
Available as of:	V2.0.0.0
Inherits from:	-
Attribute:	`'pack_mode' := '1'`

Description

The structure *UARelativePathElement* is used to provide information about a relative path element.

Structure Elements

Name	Data type	Description
ReferenceTypeID	UANodeID, page 49	Identifier for the reference type.
IsInverse	BOOL	If TRUE, the inverse references will be evaluated.
IncludeSubtypes	BOOL	Defines whether subtypes of the reference type should be returned. Default value: TRUE
TargetName	*UAQualifiedName*, page 51	Defines the name of the target element.

Used By

- *UARelativePath*

UASessionConnectInfo

Overview

Type:	Structure
Available as of:	V1.0.0.0
Inherits from:	-
Attribute:	`'pack_mode' := '1'`

Description

The structure *UASessionConnectInfo* is used to provide the connection information required to create an OPC UA session.

Structure Elements

Name	Data type	Description
SessionName	STRING [255]	Name of the session assigned by the client. The string can be empty.
ApplicationName	STRING [255]	Readable name of the OPC UA client application. The string can be empty.
SecurityMsgMode	UASecurityMsgMode, page 32	Security mode of the communication with the OPC UA server. Default value: *UASMM_BestAvailable*
SecurityPolicy	UASecurityPolicy, page 33	Security profile of the communication with the OPC UA server. Default value: *UASP_BestAvailable*
CertificateStore	STRING [255]	Not supported, value is ignored.
ClientCertificateName	STRING [255]	Not supported, value is ignored.
ServerUri	STRING [255]	Server URI of the server to connect through a gateway server. The string can be empty.
CheckServerCertificate	BOOL	Flag indicating whether the server certificate is verified with the trust list of the client application. Default value: TRUE For more details refer to the *How To Manage Certificates with OPC UA Clients, User Guide* (see How To Manage Certificates for OPC UA Client, User Guide).
TransportProfile	UATransportProfile, page 34	Transport profile. Only *UATP_UATcp* is supported.
UserIdentityToken	UAUserIdentityToken, page 56	User authentication data.
SessionTimeout	TIME	Time period for which the session is maintained if the communication is interrupted [ms]. Default value: 20 minutes (*T#20m*) Value range: 30 s...1 day
MonitorConnection	TIME	Interval time to verify the connection. Default value: 5 s (*T#5s*) Value range: 0,5 s...10 minutes
LocaleIDs	ARRAY [1..5] OF STRING [6]	Language and country/region identifier. Format: `<language>-<country/region>` Example: `en-US; zh-CHS` The string can be empty.

Used By

- *UA_Connect*

UAUserIdentityToken

Overview

Type:	Structure
Available as of:	V1.0.0.0
Inherits from:	-
Attribute:	`'pack_mode' := '1'`

Description

The structure *UAUserIdentityToken* is used to provide the information to authenticate at the OPC UA server while establishing connection.

Structure Elements

Name	Data type	Description
UserIdentityTokenType	UAUserIdentityToken-Type, page 34	Identity token to authenticate a user during the creation of a session. Default value: *UAUITT_Username*
TokenParam1	STRING [255]	Parameter for token type: • *UAUITT_Username*: Username • *UAUITT_x509*: The given value is ignored.
TokenParam2	STRING [255]	Parameter for token type: • *UAUITT_Username*: Password • *UAUITT_x509*: Name of the certificate file Also refer to Authentication by X.509 Certificates, page 35.

Used By

- *UASessionConnectInfo*

Aliases

What's in This Chapter

UABrowseResultMask

Overview

Type:	Alias
Available as of:	V2.0.0.0

Description

The alias *UABrowseResultMask* specifies which fields of the *UAReferenceDescription* are requested from the server.

UABrowseResultMask is a bitmask and every bit of the byte represents a field.

Supported Bitmasks

Bitmask	Description
16#1	*ReferenceType*
16#2	*IsForward*
16#4	*NodeClass*
16#8	*BrowseName*
16#10	*DisplayName*
16#20	*TypeDefinition*

Used By

- *UABrowseDescription*

UADeadbandType

Overview

Type:	Alias
Available as of:	V2.0.0.0

Description

The alias *UADeadbandType* specifies which type of deadband is applied, if any.

UADeadbandType is a bitmask and every bit of the byte represents a deadband.

Supported Bitmasks

Bitmask	Description
16#0	No deadband calculation is applied.
16#1	Absolute deadband calculation is applied.
16#2	Percent deadband calculation is applied.

Used By

- *UAMonitoringParameter*

UANodeClassMask

Overview

Type:	Alias
Available as of:	V1.0.0.0

Description

The alias *UANodeClassMask* specifies the *NodeClass* of a node.

UANodeClassMask is a bitmask and every bit of the byte represents a node class.

Supported Bitmasks

Bitmask	Description
16#0	No node class (unspecified)
16#1	Node class: Object
16#2	Node class: Variable
16#4	Node class: Method
16#8	Node class: Object type.
16#10	Node class: Variable type
16#20	Node class: Reference type
16#40	Node class: Data type
16#80	Node class: View
16#FF	All node classes combined

Used By

- *UAReferenceDescription*
- *UABrowseDescription*
- *UANodeInformation*

Global Variables

What’s in This Part

Global Constants List

What's in This Chapter

Global Constants List (GCL)

Overview

Type:	Global constants
Available as of:	V1.0.0.0

Description

The Global Constants List (GCL) contains the global constants of the OpcUaHandling library.

Global Constants

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

Global Parameter List

What's in This Chapter

Global Parameter List (GPL)

Overview

Type:	Global parameters
Available as of:	V1.0.0.0

Description

The global parameter list (GPL) contains global constants which are used by certain components of this library. The parameters can be edited individually for each application where the library is used. The modification must be done within the **Library Manager** of the project where the library is referenced.

The global parameter list (GPL) has the attribute *qualified access only*.

NOTE: Do not modify global parameters while the OPC UA client is connected.

Global Parameters

Variable	Data type	Default value	Range	Description
Gc_uiMaxArgumentsMethod	UINT	20	2..20	Limits the maximum number of arguments for a method call.
Gc_uiMaxVariablesTimeStamper	UINT	64	1..128	Limits the maximum number of variables to be time-stamped.
Gc_uiMaxVarNameLengthTimeStamper	UINT	80	20..255	Limits the maximum length of variable names of the variables to be time-stamped.
MAX_ELEMENTS_ARRAYDIMENSION	UINT	3	3..3	Limits the maximum dimensions of a node.
MAX_ELEMENTS_BROWSERESULT	UINT	5	1..30	Limits the number of browse results, which could be used by the browse function block.
MAX_ELEMENTS_INDEXRANGE	UINT	4	3..4	Limits the maximum defined index range.
MAX_ELEMENTS_METHOD	UINT	50	1..100	Limits the number of method handles which could be handled by the list function blocks.
MAX_ELEMENTS_MONITORLIST	UINT	10	1..100	Limits the number of monitored items, which could be used by the monitored items function blocks for each connection.
MAX_ELEMENTS_NAMESPACES	UINT	20	1..100	Limits the number of namespaces.
MAX_ELEMENTS_NODELIST	UINT	100	1..5000	Limits the number of nodes, which could be used by the list function blocks.
MAX_ELEMENTS_QUEUELIST	UINT	1	1..10	Limits the number of items inside the queue for every monitored item.
MAX_ELEMENTS_RELATIVEPATH	UINT	5	1..120	Limits the number of relative path elements in a relative path.
TIMEOUT	TIME	TIME#30s	-	Defines the default timeout in case no specific value is assigned to the function block input *Timeout*.

Program Organization Units (POU)

What’s in This Part

Function Block Description

What's in This Chapter

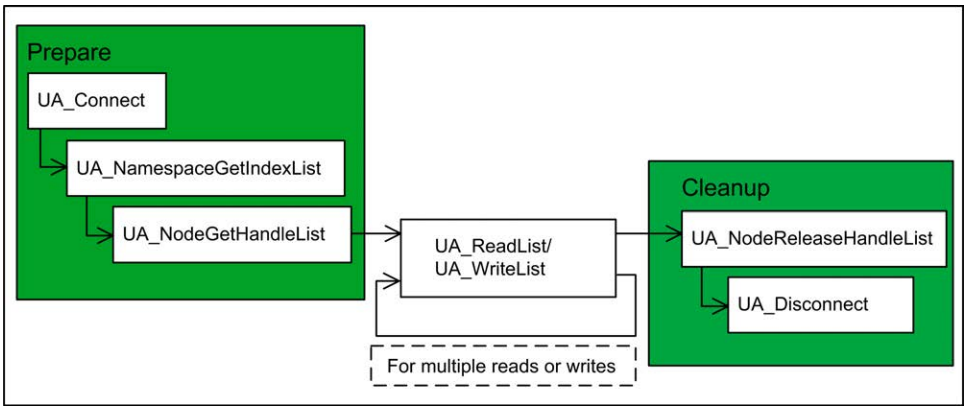
General Information on Function Blocks

Sequences for Communication

The diagrams below indicate the correlation of the function blocks to implement.

The simultaneous execution of function blocks using the same connection handle is not allowed. The function block executed using an occupied connection handle reports the error ID 16#A0000002, which stands for *PLCopenUA_Bad_FW_TempError*.

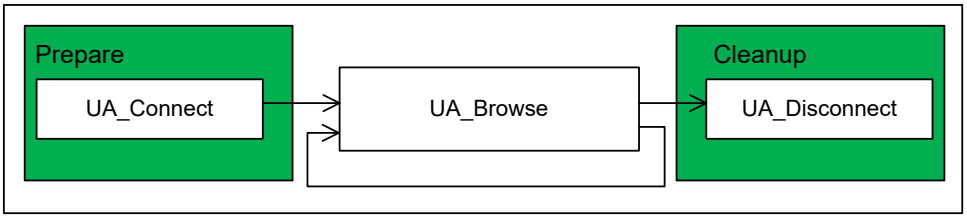
The following steps must be executed in your application to read or write a list of variables:



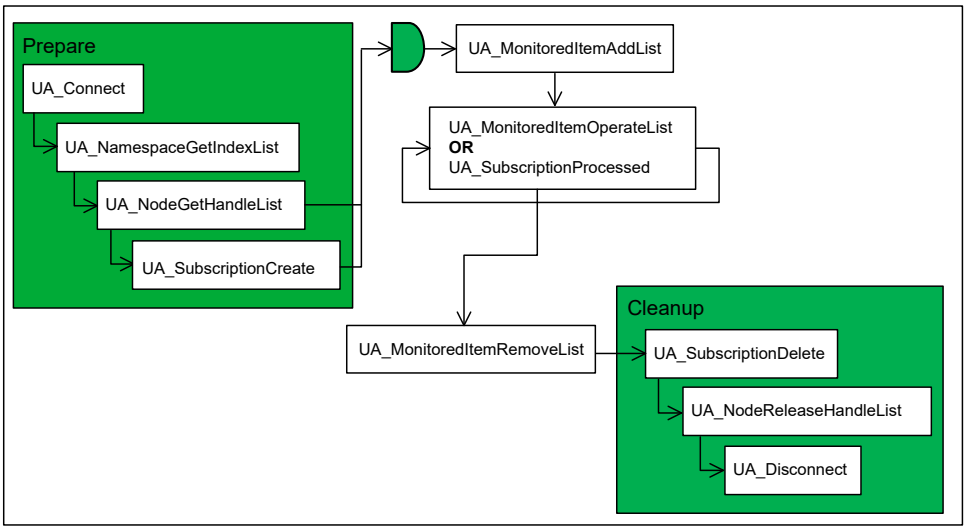
The *UA_Connect* function block is used to create a transport connection of an OPC UA session. The function block *UA_NamespaceGetIndexList* must be executed to get the namespaces of the connected OPC UA server. They are required to retrieve the node handles by executing the function block *UA_NodeGetHandleList*. Reading and writing attributes of the nodes can be performed multiple times. After the communication has been finished, the node handles are no longer required. It is a good practice to release them by executing the *UA_NodeReleaseHandleList* function block for all handles. Execute the *UA_Disconnect* function block to release the connection handle.

NOTE: In case of an online change of the application, the OPC UA client will be reinitialized. Consequently, the ongoing processes will be aborted and all connections closed.

The following steps must be executed in your application to browse the address space on the server:

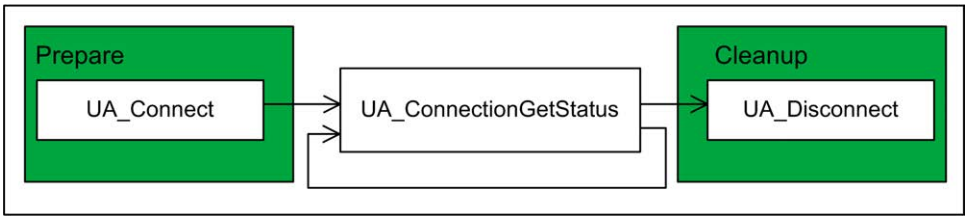


The following steps must be executed to process a subscription of a list of nodes:



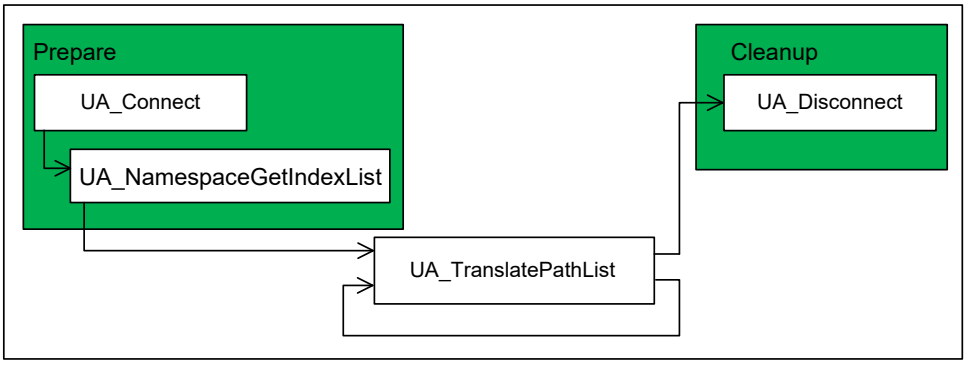
The client application initiates the communication and the values are published by the OPC UA server. As indicated in the block diagram above, a subscription and monitored items must be set up.

The following steps must be executed to get the present connection status:



You can call the function block *UA_ConnectionGetStatus* periodically. For performance reasons, do not execute it in every control program cycle.

The following steps must be executed to get the node parameters using the node path:



Code Example

Declaration

```
PROGRAM Example
VAR
    fbUaConnect : SE_Opc.UA_Connect;
    fbUaNamespaceGetIndexList       : SE_Opc.UA_NamespaceGetIndexList;
    fbUaNodeGetHandleList           : SE_Opc.UA_NodeGetHandleList;
    fbUaReadList                    : SE_Opc.UA_ReadList;
    fbUaNodeReleaseHandleList       : SE_Opc.UA_NodeReleaseHandleList;
    fbUaDisconnect                  : SE_Opc.UA_Disconnect;
    fbUaConnectionGetStatus         : SE_Opc.UA_ConnectionGetStatus;

    stSessionConnectInfo            : SE_Opc.UASessionConnectInfo;
    dwConnectionHdl                 : DWORD;
    auiNamespaceIndexes             : ARRAY [1..SE_Opc.GPL.MAX_ELEMENTS_
NAMESPACES] OF UINT;
    asNamespaceUris                 : ARRAY [1..SE_Opc.GPL.MAX_ELEMENTS_
NAMESPACES] OF STRING(255);
    astNodeIds                      : ARRAY [1..SE_Opc.GPL.MAX_ELEMENTS_NODELIST]
OF SE_Opc.UANodeID;
    adwNodeHdls                     : ARRAY [1..SE_Opc.GPL.MAX_ELEMENTS_NODELIST]
OF DWORD;
    astNodeAddInfos                 : ARRAY [1..SE_Opc.GPL.MAX_ELEMENTS_NODELIST]
OF SE_Opc.UANodeAdditionalInfo;
    astVariables                    : ARRAY [1..SE_Opc.GPL.MAX_ELEMENTS_NODELIST]
OF SE_Opc.ST_Variable;
    iReadValue                      : INT;
    xOpcCommunicationOk             : BOOL;
    iState                          : INT;
    xCmdConnect                     : BOOL;
    xCmdReadVariable                : BOOL;
    xCmdCyclicReadVariable          : BOOL;
    xCmdDisconnect                  : BOOL;
END_VAR
```

Program

```
CASE iState OF
    0:  // Idle
        IF xCmdConnect THEN
            xCmdConnect := FALSE;

           stSessionConnectInfo.UserIdentityToken.UserIdentityTokenType := SE_Opc.
UAUserIdentityTokenType.UAUITT_Username;
           stSessionConnectInfo.UserIdentityToken.TokenParam1 := 'Administrator';
           stSessionConnectInfo.UserIdentityToken.TokenParam2 := 'Password';

           fbUaConnect(    Execute:= TRUE,
                             ServerEndpointUrl:= 'opc.tcp://10.128.154.220:4840',
                                  SessionConnectInfo:= stSessionConnectInfo);

           iState := 10;
        END_IF

     10: // Connecting
        fbUaConnect();
        IF fbUaConnect.Done THEN
            dwConnectionHdl := fbUaConnect.ConnectionHdl;
            fbUaConnect (Execute := FALSE);

            asNamespaceUris[1] := 'http://www.unifiedautomation.com/
customprovider/';

           fbUaNamespaceGetIndexList( Execute:= TRUE,
                                      ConnectionHdl:= dwConnectionHdl,
                                      NamespaceUrisCount:= 1,
                                      NamespaceUris:= asNamespaceUris,
                                      NamespaceIndexes=> auiNamespaceIndexes);
```

```
            iState := 20;
        ELSIF fbUaConnect.Error THEN
             ; // Add error handling here
        END_IF

     20: // Wait for IndexList
        fbUaNamespaceGetIndexList();
        IF fbUaNamespaceGetIndexList.Done THEN

            IF fbUaNamespaceGetIndexList.ErrorIDs[1] = SE_Opc.ET_Result.OpcUa_Good
THEN
                auiNamespaceIndexes := fbUaNamespaceGetIndexList.NamespaceIndexes;
                fbUaNamespaceGetIndexList(Execute := FALSE);

                astNodeIds[1].IdentifierType := SE_Opc.UAIdentifierType.UAIT_
String;
                astNodeIds[1].Identifier := 'Application.GVL.G_iVariableToRead';
                astNodeIds[1].NamespaceIndex := auiNamespaceIndexes[1];

                fbUaNodeGetHandleList( Execute:= TRUE,
                                       ConnectionHdl:= dwConnectionHdl,
                                       NodeIDCount:= 1,
                                       NodeIDs:= astNodeIds);

                iState := 30;
           ELSE
                ; // Add error handling here
           END_IF
        ELSIF fbUaNamespaceGetIndexList.Error THEN
            ; // Add error handling here
        END_IF

    30: // Wait for HandleList
        fbUaNodeGetHandleList();
        IF fbUaNodeGetHandleList.Done THEN
            IF fbUaNodeGetHandleList.NodeErrorIDs[1] = SE_Opc.ET_Result.OpcUa_Good
THEN
                adwNodeHdls := fbUaNodeGetHandleList.NodeHdls;
                fbUaNodeGetHandleList(Execute := FALSE);
                iState := 100;
            ELSE
                ; // Add error handling here
            END_IF
        ELSIF fbUaNodeGetHandleList.Error THEN
                ; // Add error handling here
        END_IF

    100:// client is connected, ready for read & write variables
        IF xCmdDisconnect THEN
           xCmdDisconnect := FALSE;

           fbUaNodeReleaseHandleList( Execute:= TRUE,#
                                      ConnectionHdl:= dwConnectionHdl,
                                      NodeHdlCount:= 1,
                                      NodeHdls:= adwNodeHdls);
           iState := 200;
        ELSIF xCmdReadVariable OR xCmdCyclicReadVariable THEN
            xCmdReadVariable := FALSE;

            astNodeAddInfos[1].AttributeID := SE_Opc.UAAttributeID.UAAI_Value;
            astNodeAddInfos[1].IndexRangeCount := 0;

            astVariables[1].etNodeDataType := SE_Opc.ET_VarType.UATypeInt16;
            astVariables[1].pbyBuffer := ADR(iReadValue);
            astVariables[1].udiBufferSize := SIZEOF(iReadValue);

            fbUaReadList( Execute:= TRUE,
```

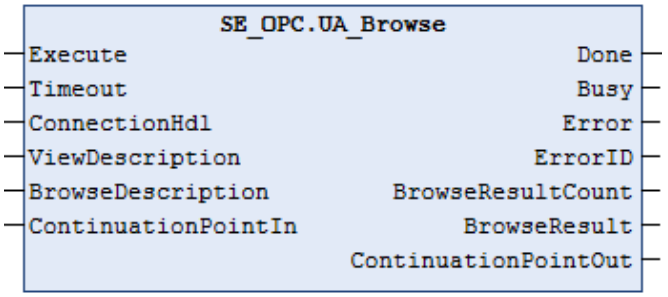
```
                            ConnectionHdl:= dwConnectionHdl,
                            NodeHdlCount:= 1,
                            NodeHdls:= adwNodeHdls,
                            NodeAddInfos:= astNodeAddInfos,
                            Variables:= astVariables);

            iState := 110;
        ELSE
            fbUaConnectionGetStatus( Execute:= TRUE,
                                     ConnectionHdl:= dwConnectionHdl);

            iState := 120;
        END_IF

    110:// Wait for read operatin done
        fbUaReadList(Variables:= astVariables);
        IF fbUaReadList.Done THEN
            IF fbUaReadList.NodeErrorIDs[1] = SE_Opc.ET_Result.OpcUa_Good THEN
              ; // Read value is valid and can be processed here
            ELSE
              ; // Add error handling here
            END_IF
            fbUaReadList(Execute := FALSE, Variables:= astVariables);
            iState := 100;
        ELSIF fbUaReadList.Error THEN
            ; // Add error handling here
        END_IF

    120:// Wait for connection status
        fbUaConnectionGetStatus();
        IF fbUaConnectionGetStatus.Done THEN
            IF fbUaConnectionGetStatus.ServerState = SE_Opc.UAServerState.UASS_
Running AND
                fbUaConnectionGetStatus.ConnectionStatus = SE_Opc.
UAConnectionStatus.UACS_Connected THEN
                xOpcCommunicationOk := TRUE;
            ELSE
                // Add handling of communication interruption here
                xOpcCommunicationOk := FALSE;
            END_IF
            fbUaConnectionGetStatus (Execute := FALSE);
            iState := 100;
        ELSIF fbUaConnectionGetStatus.Error THEN
            ; // Add error handling here
        END_IF

    200:// Wait for HandleList is released
        fbUaNodeReleaseHandleList();
        IF fbUaNodeReleaseHandleList.Done THEN
            IF fbUaNodeReleaseHandleList.NodeErrorIDs[1] = SE_Opc.ET_Result.OpcUa_
Good THEN
               fbUaNodeReleaseHandleList(Execute := FALSE);

               fbUaDisconnect( Execute:= TRUE,
                               ConnectionHdl:= dwConnectionHdl);

               iState := 210;
            ELSE
                ; // Add error handling here
            END_IF
        ELSIF fbUaNodeReleaseHandleList.Error THEN
            ; // Add error handling here
        END_IF

    210:// Wait for disconnection done
        fbUaDisconnect();
        IF fbUaDisconnect.Done THEN
            fbUaDisconnect( Execute:= FALSE);
```

```
            iState := 0;
        ELSIF fbUaDisconnect.Error THEN
            ; // Add error handling here
        END_IF
END_CASE
```

UA_Browse

Overview

Type:	Function block
Available as of:	V2.0.0.0



Functional Description

The function block *UA_Browse* is used to navigate through the address space. After a starting node, the server returns a list of nodes by references. If not all nodes are returned, assign the output *ContinuationPointOut* to the input *ContinuationPointIn* and re-execute the function block.

After successful function block execution, the output *ContinuationPointOut* is set to 0 to indicate that the browse request has been completed and that the results have been returned. Up to 10 ongoing browse requests can be handled in the internal buffer of the function block.

To delete the internal buffer, execute one of the following actions:

- Call *FB_Browse* by setting the input *ContinuationPointIn* to `16#FFFFFFFF`: No browse request is executed, but the internal buffer is reset.
- Call *UA_Disconnect* to disconnect the OPC UA client.
- Execute the **Online > Reset Cold** command to reset the application.

Interface

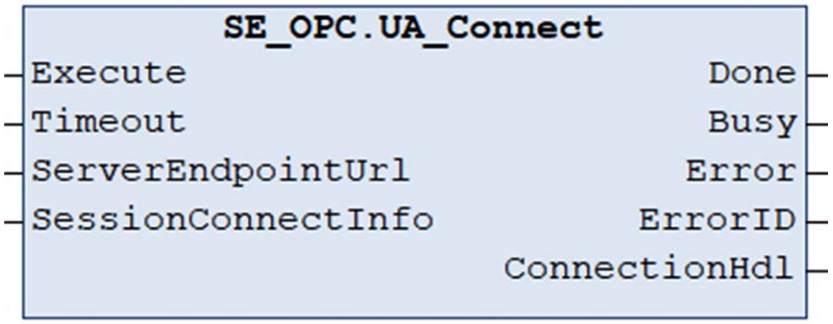
Input	Data type	Description
Execute	BOOL	Upon a rising edge, the function block is being executed. Also refer to *Behavior of Function Blocks with the Input Execute*, page 19.
Timeout	TIME	Maximum time to respond. Value range: 2 s...60 s If the value is out of range the upper or lower limit is applied. Default value: *GPL.Timeout*
ConnectionHdl	DWORD	Connection handle.
ViewDescription	*UAViewDescription*	Not used. Default value: empty
BrowseDescription	*UABrowseDescription*	Starting node and other information for navigation. **NOTE:** This parameter is ignored if the input *ContinuationPointIn* is not 0.
ContinuationPointIn	DWORD	• If set to 0, the browse process starts with the starting node. • If set to *ContinuationPointOut*, a browse next service can be performed. **NOTE:** PacDrive LMC controllers support *ContinuationPointIn* from the server if it is limited to unsigned integer of 32 bits.

Output	Data type	Description
Done	BOOL	Indicates that the execution of the function block was completed successfully.
Busy	BOOL	Indicates that the execution of the function block is in progress.
Error	BOOL	Indicates that an error was detected during execution. **NOTE:** Even if *Error* indicates FALSE, verify the corresponding *ErrorIDs* before processing the namespace indexes.
ErrorID	ET_Result, page 24	Provides additional diagnostic information as a numeric value. For each specified namespace URI, a separate result is provided.
BrowseResultCount	UINT	Indicates the number of entries inside the *BrowseResult* array.
BrowseResult	ARRAY [1..*GPL. MAX_ELEMENTS_NODELIST*] OF *UAReferenceDescription*	Contains references and target node information for the nodes passing the filter criteria in the request.
ContinuationPointOut	DWORD	Indicates that the server was not able to deliver all the results. Can be copied to *ContinuationPointIn* for browse next service. **NOTE:** PacDrive LMC controllers support *ContinuationPointOut* from the server if it is limited to unsigned integer of 32 bits.

UA_Connect

Overview

Type:	Function block
Available as of:	V1.0.0.0



Functional Description

The function block *UA_Connect* is used to create a transport connection and an OPC UA session.

NOTE:

Before executing the function block *UA_Connect*, enable the OPC UA stack on your controller:

- For Modicon M262 Logic/Motion Controllers, activate the option **OPC UA Server enabled** in the **OPC UA Server Configuration** tab of the device editor.
- For PacDrive controllers, call the function *SystemInterface.FC_OpcUaStart* once in the initialization of your application.

The function block *UA_Connect* must be executed once for each connection. The connection is terminated by calling the *UA_Disconnect* function block.

NOTE: The connection monitoring and the reconnect handling must be implemented separately within your application. Refer to OPC UA specification part 4.

Interface

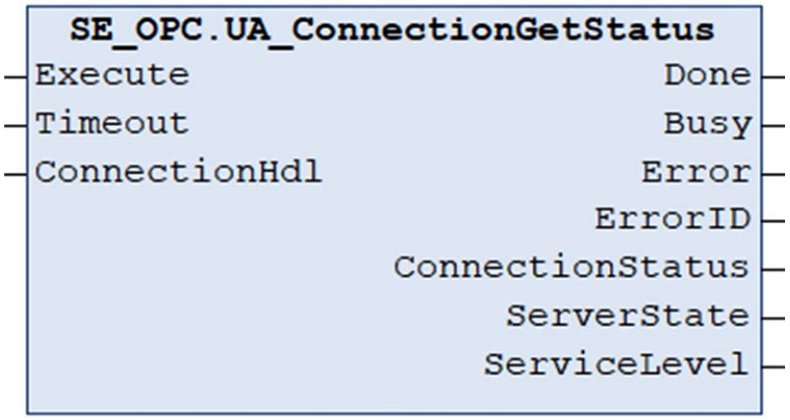
Input	Data type	Description
Execute	BOOL	Upon a rising edge, the function block is being executed. Also refer to *Behavior of Function Blocks with the Input Execute*, page 19.
Timeout	TIME	Maximum time to respond. Value range: 2 s...60 s If the value is out of range the upper or lower limit is applied. Default value: *GPL.Timeout*
ServerEndpointUrl	STRING [255]	URL of the server to connect to. For example, *opc.tcp://10.128.154.220:4840*. This string must not be a null string.
SessionConnectInfo	UASessionConnectInfo, page 54	Structure to specify the connection information required to create an OPC UA session.

Output	Data type	Description
Done	BOOL	Indicates that the execution of the function block was completed successfully.
Busy	BOOL	Indicates that the execution of the function block is in progress.
Error	BOOL	Indicates that an error was detected during execution. **NOTE:** Even if *Error* indicates FALSE, verify the corresponding *ErrorIDs* before processing the namespace indexes.
ErrorID	ET_Result, page 24	Provides additional diagnostic information as a numeric value. For each specified namespace URI, a separate result is provided.
ConnectionHdl	DWORD	Connection handle valid until *UA_Disconnect* is called.

UA_ConnectionGetStatus

Overview

Type:	Function block
Available as of:	V1.0.0.0



Functional Description

The function block *UA_ConnectionGetStatus* is used to get the connection status of a specified connection.

Interface

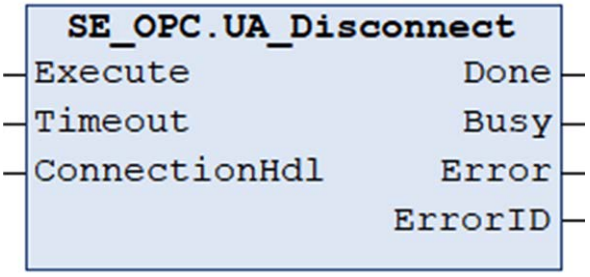
Input	Data type	Description
Execute	BOOL	Upon a rising edge, the function block is being executed. Also refer to *Behavior of Function Blocks with the Input Execute*, page 19.
Timeout	TIME	Maximum time to respond. Value range: 2 s...60 s If the value is out of range the upper or lower limit is applied. Default value: *GPL.Timeout*
ConnectionHdl	DWORD	Connection handle.

Output	Data type	Description
Done	BOOL	Indicates that the execution of the function block was completed successfully.
Busy	BOOL	Indicates that the execution of the function block is in progress.
Error	BOOL	Indicates that an error was detected during execution. **NOTE:** Even if *Error* indicates FALSE, verify the corresponding *ErrorIDs* before processing the namespace indexes.
ErrorID	ET_Result, page 24	Provides additional diagnostic information as a numeric value. For each specified namespace URI, a separate result is provided.
ConnectionStatus	UAConnectionStatus, page 30	Indicates the connection status.
ServerState	UAServerState, page 33	Indicates the server status.
ServiceLevel	BYTE	Indicates the ability of the server to provide its data to the client. The value range is from 0...255, where 0 indicates the lowest and 255 indicates the highest level (best). The intent is to provide the clients an indication of availability among redundant servers.

UA_Disconnect

Overview

Type:	Function block
Available as of:	V1.0.0.0



Functional Description

The function block *UA_Disconnect* is used to close the specified transport connection of an OPC UA session.

NOTE: Regardless of the result of executing, the function block *UA_Disconnect* releases the connection handle and all node handles of the specified connection.

Interface

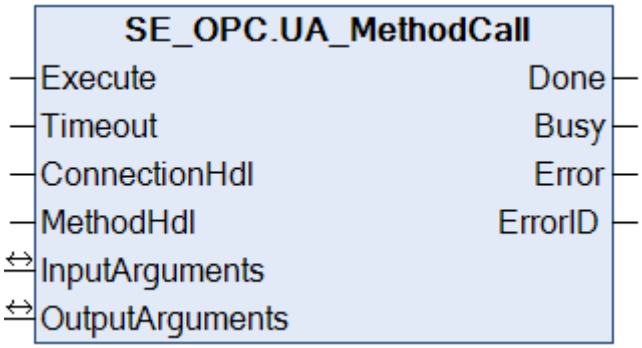
Input	Data type	Description
Execute	BOOL	Upon a rising edge, the function block is being executed. Also refer to *Behavior of Function Blocks with the Input Execute*, page 19.
Timeout	TIME	Maximum time to respond. Value range: 2 s...60 s If the value is out of range the upper or lower limit is applied. Default value: *GPL.Timeout*
ConnectionHdl	DWORD	Connection handle.

Output	Data type	Description
Done	BOOL	Indicates that the execution of the function block was completed successfully.
Busy	BOOL	Indicates that the execution of the function block is in progress.
Error	BOOL	Indicates that an error was detected during execution. **NOTE:** Even if *Error* indicates FALSE, verify the corresponding *ErrorIDs* before processing the namespace indexes.
ErrorID	ET_Result, page 24	Provides additional diagnostic information as a numeric value. For each specified namespace URI, a separate result is provided.

UA_MethodCall

Overview

Type:	Function block
Available as of:	V2.2.4.0



Functional Description

The function block *UA_MethodCall* is used to execute a method call with the possibility to send and to receive a set of arguments. The implementation of the method is done on the OPC UA server.

NOTE: To help avoid an inconsistent response, do not modify parameters while the function block is executing (*Busy* = TRUE).

⚠WARNING

UNINTENDED EQUIPMENT OPERATION

Do not modify input parameters while the *Busy* output is equal to TRUE.

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

NOTE: Arguments with data type *ET_VarType.UATypeLocalizedText* are exclusive to Modicon M262 Logic/Motion Controllers with firmware version V5.2.8.27 or greater.

Interface

Input	Data type	Description
Execute	BOOL	Upon a rising edge, the function block is being executed. Also refer to *Behavior of Function Blocks with the Input Execute*, page 19.
Timeout	TIME	Maximum time to respond. Value range: 2 s...60 s If the value is out of range the upper or lower limit is applied. Default value: *GPL.Timeout*
ConnectionHdl	DWORD	Connection handle.
MethodHdl	DWORD	Method handle returned by the function block *UA_MethodGetHandleList*, page 78.

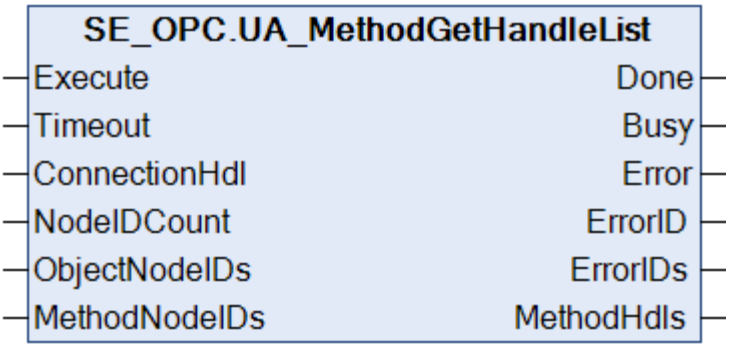
Inputs / Outputs	Data type	Description
InputArguments	*ST_Arguments*, page 37	Structure containing the arguments to send to the OPC UA server.
OutputArguments	*ST_Arguments*, page 37	Structure containing the arguments received from the OPC UA server.

Output	Data type	Description
Done	BOOL	Indicates that the execution of the function block was completed successfully.
Busy	BOOL	Indicates that the execution of the function block is in progress.
Error	BOOL	Indicates that an error was detected during execution. **NOTE:** Even if *Error* indicates FALSE, verify the corresponding *ErrorIDs* before processing the namespace indexes.
ErrorID	ET_Result, page 24	Provides additional diagnostic information as a numeric value. For each specified namespace URI, a separate result is provided.

UA_MethodGetHandleList

Overview

Type:	Function block
Available as of:	V2.2.4.0



Functional Description

The function block *UA_MethodGetHandleList* is used to get the method handles for a method call.

NOTE: To help avoid an inconsistent response, do not modify parameters while the function block is executing (*Busy* = TRUE).

⚠WARNING

UNINTENDED EQUIPMENT OPERATION

Do not modify input parameters while the *Busy* output is equal to TRUE.

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

Interface

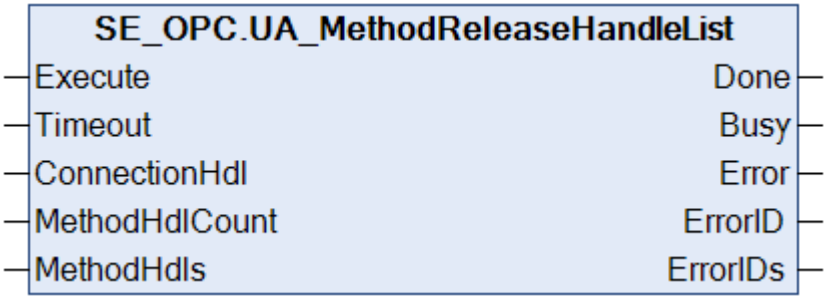
Input	Data type	Description
Execute	BOOL	Upon a rising edge, the function block is being executed. Also refer to *Behavior of Function Blocks with the Input Execute*, page 19.
Timeout	TIME	Maximum time to respond. Value range: 2 s...60 s If the value is out of range the upper or lower limit is applied. Default value: *GPL.Timeout*
ConnectionHdl	DWORD	Connection handle.
NodeIDCount	UINT	Number of node paths in the arrays *ObjectNodeIDs* and *MethodNodeIDs*. It must not exceed the value of *GPL.MAX_ELEMENTS_METHOD*, page 63.
ObjectNodeIDs	ARRAY [1..*GPL. MAX_ELEMENTS_METHOD*] OF *UANodeID*	Array containing paths of object nodes.
MethodNodeIDs	ARRAY [1..*GPL. MAX_ELEMENTS_METHOD*] OF *UANodeID*	Array containing paths of method nodes.

Output	Data type	Description
Done	BOOL	Indicates that the execution of the function block was completed successfully.
Busy	BOOL	Indicates that the execution of the function block is in progress.
Error	BOOL	Indicates that an error was detected during execution. **NOTE:** Even if *Error* indicates FALSE, verify the corresponding *ErrorIDs* before processing the namespace indexes.
ErrorID	ET_Result, page 24	Provides additional diagnostic information as a numeric value. For each specified namespace URI, a separate result is provided.
ErrorIDs	ARRAY [1..*GPL. MAX_ELEMENTS_METHOD*] OF *ET_Result*	Array containing error codes for each node ID specified inside the arrays at the function block input.
MethodHdls	ARRAY [1..*GPL.MAX_ELEMENTS_NODELIST*] OF DWORD	Array containing method handles for each node ID specified inside the arrays at the function block input.

UA_MethodReleaseHandleList

Overview

Type:	Function block
Available as of:	V2.2.4.0



Functional Description

The function block *UA_MethodReleaseHandleList* is used to release a set of the method handles.

NOTE: To help avoid an inconsistent response, do not modify parameters while the function block is executing (*Busy* = TRUE).

⚠WARNING

UNINTENDED EQUIPMENT OPERATION

Do not modify input parameters while the *Busy* output is equal to TRUE.

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

Interface

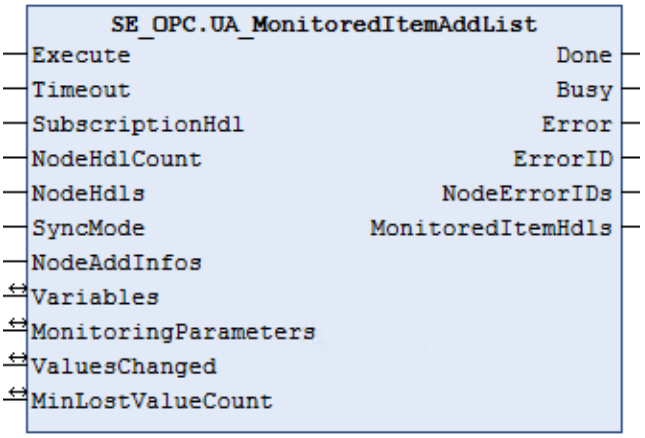
Input	Data type	Description
Execute	BOOL	Upon a rising edge, the function block is being executed. Also refer to *Behavior of Function Blocks with the Input Execute*, page 19.
Timeout	TIME	Maximum time to respond. Value range: 2 s...60 s If the value is out of range the upper or lower limit is applied. Default value: *GPL.Timeout*
ConnectionHdl	DWORD	Connection handle.
MethodHdlCount	UINT	Number of method handles in the array *MethodHdls*. It must not exceed the value of *GPL.MAX_ELEMENTS_METHOD*, page 63.
MethodHdls	ARRAY [1..*GPL. MAX_ELEMENTS_METHOD*] OF DWORD	Array containing method handles.

Output	Data type	Description
Done	BOOL	Indicates that the execution of the function block was completed successfully.
Busy	BOOL	Indicates that the execution of the function block is in progress.
Error	BOOL	Indicates that an error was detected during execution. **NOTE:** Even if *Error* indicates FALSE, verify the corresponding *ErrorIDs* before processing the namespace indexes.
ErrorID	ET_Result, page 24	Provides additional diagnostic information as a numeric value. For each specified namespace URI, a separate result is provided.
ErrorIDs	ARRAY [1..*GPL. MAX_ELEMENTS_METHOD*] OF *ET_Result*	Array containing error codes for each method handle specified inside the array at the function block input.

UA_MonitoredItemAddList

Overview

Type:	Function block
Available as of:	V2.0.0.0



Functional Description

The function block *UA_MonitoredItemAddList* is used to add monitored items to a subscription using a list of node handles.

The function block updates the values of *SamplingTime* and *QueueSize* by using the input/output parameter *MonitoringParameters*.

The remaining input/output parameters are updated separately depending on the selected *SyncMode*, page 31:

- For *SyncMode UAMS_ControllerSync*, call the function block *UA_MonitoredItemOperateList* for updating values.
- For *SyncMode UAMS_FwSync*, the firmware updates the values automatically according to the interval configured with the input/output *PublishingInterval*, page 94 for the subscription. The function block *UA_SubscriptionProcessed* can be called optionally to verify whether the new values have been published.

  NOTE: The *SyncMode UAMS_FwSync* is only compatible using the *NodeDataType UATypeIECSymbol* to reference variables of base data types with a maximum size of 8 bytes. For further information on *UATypeIECSymbol*, refer to *ET_VarType*, page 27.

  NOTE: A data type mismatch between an item to be monitored specified at the input/output variables and the corresponding variable declared on the server side cannot be detected. When a data type mismatch occurs, an implicit conversion is performed.

  NOTE: *ByteString* is represented as a one-dimensional ARRAY OF BYTE on the client side. If *ByteString* is declared on the server side, use a buffer of type ARRAY OF BYTE and *NodeDataType UATypeByte*.

  NOTE: The function block does not support the *MaxAge* feature specified by the OPC UA protocol.

Interface

Input	Data type	Description
Execute	BOOL	Upon a rising edge, the function block is being executed. Also refer to *Behavior of Function Blocks with the Input Execute*, page 19.
Timeout	TIME	Maximum time to respond. Value range: 2 s...60 s If the value is out of range the upper or lower limit is applied. Default value: *GPL.Timeout*
SubscriptionHdl	DWORD	Subscription handle.
NodeHdlCount	UINT	Number of node handles in the *NodeHdls* array. Value range: *1..GPL.MAX_ELEMENTS_NODELIST]*
NodeHdls	ARRAY [1..*GPL. MAX_ELEMENTS_ MONITORLIST*] OF DWORD	Array containing node handles.
SyncMode	*UAMonitoringSyncMode*	Synchronization mode.
NodeAddInfos	ARRAY [1..*GPL. MAX_ELEMENTS_ MONITORLIST*] OF *UANodeAdditionalInfo*	Array containing additional node information like attribute and index range.

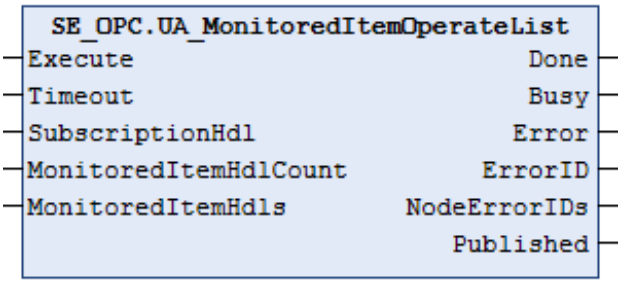
Input/Output	Data type	Description
Variables	ARRAY [1..*GPL. MAX_ELEMENTS_ MONITORLIST*] OF *UAMonitoredVariables*	Array containing information about the variables to read and the corresponding memory areas. **NOTE:** Do not process the variables until the function block indicates *Done*.
MonitoringParameters	ARRAY [1..*GPL. MAX_ELEMENTS_ MONITORLIST*] OF *UAMonitoringParameter*	Array containing monitoring parameters for each valid element of the *NodeHdls* array.
ValuesChanged	ARRAY [1..*GPL. MAX_ELEMENTS_ MONITORLIST*] OF BOOL	Indicates that the values of the monitored item have been modified.
MinLostValueCount	ARRAY [1..*GPL. MAX_ELEMENTS_ MONITORLIST*] OF UINT	Indicates the number of missed values if the queue size is greater than 1 in case the queue size on the client side is smaller than the queue size on the server side.

Output	Data type	Description
Done	BOOL	Indicates that the execution of the function block was completed successfully.
Busy	BOOL	Indicates that the execution of the function block is in progress.
Error	BOOL	Indicates that an error was detected during execution. **NOTE:** Even if *Error* indicates FALSE, verify the corresponding *ErrorIDs* before processing the namespace indexes.
ErrorID	ET_Result, page 24	Provides additional diagnostic information as a numeric value. For each specified namespace URI, a separate result is provided.
NodeErrorIDs	ARRAY [1..*GPL. MAX_ELEMENTS_ MONITORLIST*] OF *ET_Result*	Contains an error value for each element of the *NodeHdls* array.
MonitoredItemHdls	ARRAY [1..*GPL. MAX_ELEMENTS_ NODELIST*] OF DWORD	Contains monitored item handles for each valid element of the *NodeHdls* array.

UA_MonitoredItemOperateList

Overview

Type:	Function block
Available as of:	V2.0.0.0



Functional Description

The function block *UA_MonitoredItemOperateList* is used to update values of multiple monitored items using a list of monitored item handles and the corresponding subscription handle.

NOTE:

- The *UA_MonitoredItemOperateList* function block can only be used in combination with the controller synchronization mode (*UAMS_ControllerSync*), page 31.
- The values of the variables *Variables*, *ValuesChanged*, *TimeStamps* and *NodeQualityIDs* of the corresponding monitored item specified with the function block *UA_MonitoredItemAddList*, page 81 are only valid after the function block *UA_MonitoredItemOperateList* returns *Done* = TRUE and no *NodeErrorID*.
- If the *UA_MonitoredItemOperateList* function block is not called, the values of the monitored items are not updated.

Interface

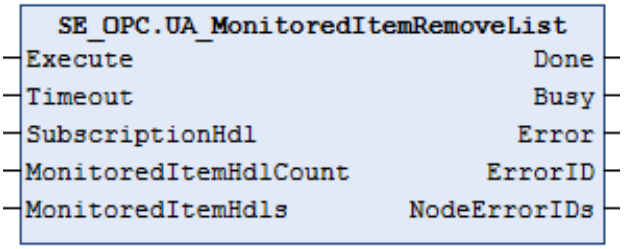
Input	Data type	Description
Execute	BOOL	Upon a rising edge, the function block is being executed. Also refer to *Behavior of Function Blocks with the Input Execute*, page 19.
Timeout	TIME	Maximum time to respond. Value range: 2 s...60 s If the value is out of range the upper or lower limit is applied. Default value: *GPL.Timeout*
SubscriptionHdl	DWORD	Subscription handle.
MonitoredItemHdlCount	UINT	Number of monitored item handles in the *MonitoredItemHdls* array.
MonitoredItemHdls	ARRAY [1..*GPL. MAX_ELEMENTS_MONITORLIST*] OF DWORD	Array containing the monitored item handles to be updated.

Output	Data type	Description
Done	BOOL	Indicates that the execution of the function block was completed successfully.
Busy	BOOL	Indicates that the execution of the function block is in progress.
Error	BOOL	Indicates that an error was detected during execution. **NOTE:** Even if *Error* indicates FALSE, verify the corresponding *ErrorIDs* before processing the namespace indexes.
ErrorID	ET_Result, page 24	Provides additional diagnostic information as a numeric value. For each specified namespace URI, a separate result is provided.
NodeErrorIDs	ARRAY [1..*GPL. MAX_ELEMENTS_MONITORLIST*] OF *ET_Result*	Contains an error value for each element of the *MonitoredItemHdls* array.
Published	BOOL	Indicates that variables have been published since the last call. At least one element of the array *ValuesChanged*, page 82 is TRUE.

UA_MonitoredItemRemoveList

Overview

Type:	Function block
Available as of:	V2.0.0.0



Functional Description

The function block *UA_MonitoredItemRemoveList* is used to remove monitored items from a subscription using a list of monitored item handles and the corresponding subscription handle.

Interface

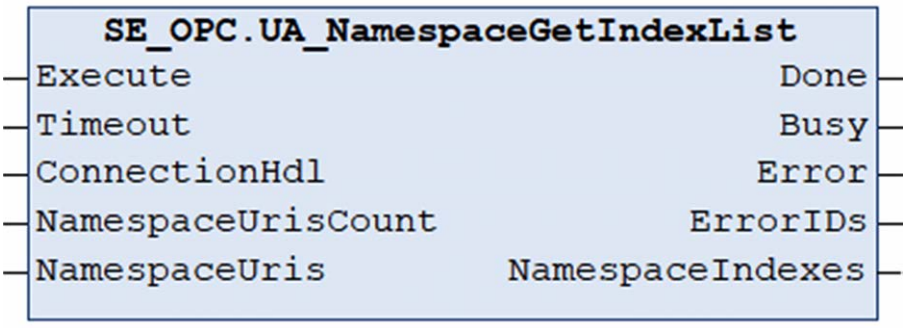
Input	Data type	Description
Execute	BOOL	Upon a rising edge, the function block is being executed. Also refer to *Behavior of Function Blocks with the Input Execute*, page 19.
Timeout	TIME	Maximum time to respond. Value range: 2 s...60 s If the value is out of range the upper or lower limit is applied. Default value: *GPL.Timeout*
SubscriptionHdl	DWORD	Subscription handle.
MonitoredItemHdlCount	UINT	Number of monitored item handles in the *MonitoredItemHdls* array.
MonitoredItemHdls	ARRAY [1..*GPL. MAX_ELEMENTS_MONITORLIST*] OF DWORD	Array containing the monitored item handles to be removed.

Output	Data type	Description
Done	BOOL	Indicates that the execution of the function block was completed successfully.
Busy	BOOL	Indicates that the execution of the function block is in progress.
Error	BOOL	Indicates that an error was detected during execution. **NOTE:** Even if *Error* indicates FALSE, verify the corresponding *ErrorIDs* before processing the namespace indexes.
ErrorID	ET_Result, page 24	Provides additional diagnostic information as a numeric value. For each specified namespace URI, a separate result is provided.
NodeErrorIDs	ARRAY [1..*GPL. MAX_ELEMENTS_MONITORLIST*] OF *ET_Result*	Contains an error value for each element of the *MonitoredItemHdls* array.

UA_NamespaceGetIndexList

Overview

Type:	Function block
Available as of:	V1.0.0.0



Functional Description

The function block *UA_NamespaceGetIndexList* is used to get the namespace indexes of several namespace URIs.

NOTE: The function block reads the server object *NamespaceArray* and returns the indexes of the requested elements. The namespace index is an element within the structure *UANodeID*, which is required for subsequently used function blocks, page 65.

NOTE: To help avoid an inconsistent response, do not modify parameters while the function block is executing (*Busy* = TRUE).

⚠WARNING

UNINTENDED EQUIPMENT OPERATION

Do not modify input parameters while the *Busy* output is equal to TRUE.

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

Interface

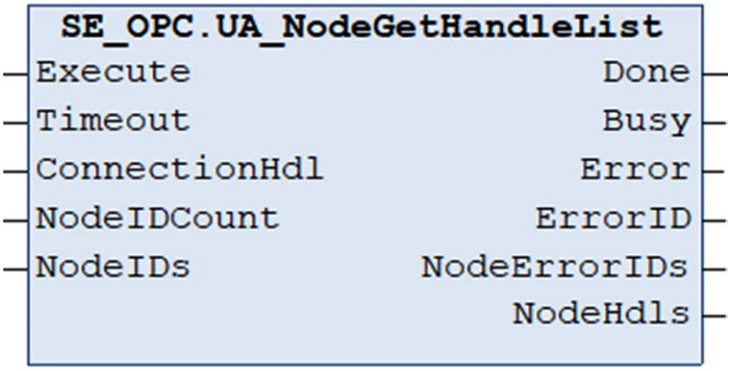
Input	Data type	Description
Execute	BOOL	Upon a rising edge, the function block is being executed. Also refer to *Behavior of Function Blocks with the Input Execute*, page 19.
Timeout	TIME	Maximum time to respond. Value range: 2 s...60 s If the value is out of range the upper or lower limit is applied. Default value: *GPL.Timeout*
ConnectionHdl	DWORD	Connection handle.
NamespaceUrisCount	UINT	Number of namespace URIs in *NamespaceUris* array. Value range: *1.. GPL.MAX_ELEMENTS_ NAMESPACES*
NamespaceUris	ARRAY [1..GPL.MAX_ELEMENTS_ NAMESPACES] OF STRING [255]	Array containing namespace URIs.

Output	Data type	Description
Done	BOOL	Indicates that the execution of the function block was completed successfully.
Busy	BOOL	Indicates that the execution of the function block is in progress.
Error	BOOL	Indicates that an error was detected during execution. **NOTE:** Even if *Error* indicates FALSE, verify the corresponding *ErrorIDs* before processing the namespace indexes.
ErrorIDs	ARRAY [1..GPL.MAX_ELEMENTS_ NAMESPACES] OF ET_Result, page 24	Provides additional diagnostic information as a numeric value. For each specified namespace URI, a separate result is provided.
NamespaceIndexes	ARRAY [1..GPL.MAX_ELEMENTS_ NAMESPACES] OF UINT	Provides retrieved namespace indexes.

UA_NodeGetHandleList

Overview

Type:	Function block
Available as of:	V1.0.0.0



Functional Description

The function block *UA_NodeGetHandleList* is used to get multiple node handles.

NOTE: To help avoid an inconsistent response, do not modify parameters while the function block is executing (*Busy* = TRUE).

⚠WARNING

UNINTENDED EQUIPMENT OPERATION

Do not modify input parameters while the *Busy* output is equal to TRUE.

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

Interface

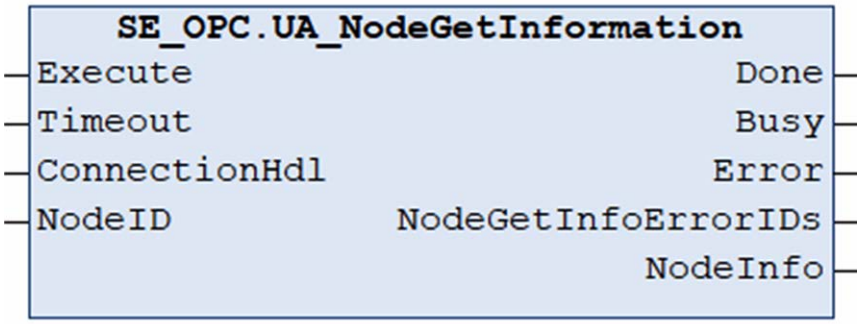
Input	Data type	Description
Execute	BOOL	Upon a rising edge, the function block is being executed. Also refer to *Behavior of Function Blocks with the Input Execute*, page 19.
Timeout	TIME	Maximum time to respond. Value range: 2 s...60 s If the value is out of range the upper or lower limit is applied. Default value: *GPL.Timeout*
ConnectionHdl	DWORD	Connection handle.
NodeIDCount	UINT	Number of node Ids in *NodeIDs* array.
NodeIDs	ARRAY [1..GPL.MAX_ELEMENTS_NODELIST] OF UANodeID	Array containing node Ids. Value range: *1.. GPL.MAX_ELEMENTS_NODELIST]*

Output	Data type	Description
Done	BOOL	Indicates that the execution of the function block was completed successfully.
Busy	BOOL	Indicates that the execution of the function block is in progress.
Error	BOOL	Indicates that an error was detected during execution. **NOTE:** Even if *Error* indicates FALSE, verify the corresponding *ErrorIDs* before processing the namespace indexes.
ErrorID	ET_Result, page 24	Provides additional diagnostic information as a numeric value. For each specified namespace URI, a separate result is provided.
NodeErrorIDs	ARRAY [1..GPL.MAX_ELEMENTS_NODELIST] OF ET_Result, page 24	Contains an error value for each element of the *NodeHdls* array.
NodeHdls	ARRAY [1..GPL.MAX_ELEMENTS_NODELIST] OF DWORD	Provides retrieved node handles.

UA_NodeGetInformation

Overview

Type:	Function block
Available as of:	V1.0.0.0



Functional Description

The function block *UA_NodeGetInformation* is used to get node information.

NOTE: To help avoid an inconsistent response, do not modify parameters while the function block is executing (*Busy* = TRUE).

⚠WARNING

UNINTENDED EQUIPMENT OPERATION

Do not modify input parameters while the *Busy* output is equal to TRUE.

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

Interface

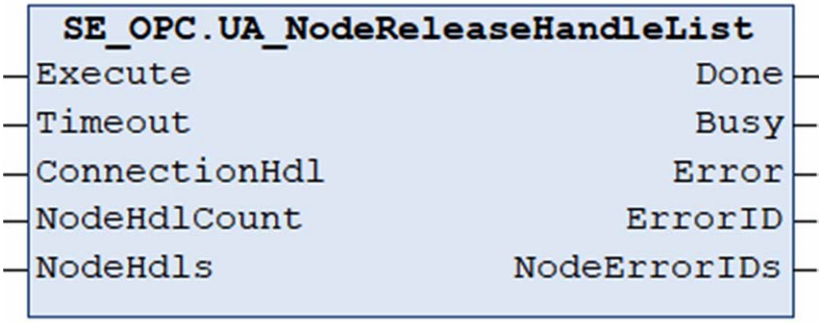
Input	Data type	Description
Execute	BOOL	Upon a rising edge, the function block is being executed. Also refer to *Behavior of Function Blocks with the Input Execute*, page 19.
Timeout	TIME	Maximum time to respond. Value range: 2 s...60 s If the value is out of range the upper or lower limit is applied. Default value: *GPL.Timeout*
ConnectionHdl	DWORD	Connection handle.
NodeID	UANodeID, page 49	Node Id.

Output	Data type	Description
Done	BOOL	Indicates that the execution of the function block was completed successfully.
Busy	BOOL	Indicates that the execution of the function block is in progress.
Error	BOOL	Indicates that an error was detected during execution. **NOTE:** Even if *Error* indicates FALSE, verify the corresponding *ErrorIDs* before processing the namespace indexes.
NodeGetInfoErrorIDs	ARRAY [0..21] OF ET_Result, page 24	Provides an error value for each element of the *NodeHdls* array.
NodeInfo	UANodeInformation, page 50	Provides the node information.

UA_NodeReleaseHandleList

Overview

Type:	Function block
Available as of:	V1.0.0.0



Functional Description

The function block *UA_NodeReleaseHandleList* is used to release multiple node handles.

NOTE: To help avoid an inconsistent response, do not modify parameters while the function block is executing (*Busy* = TRUE).

⚠WARNING

UNINTENDED EQUIPMENT OPERATION

Do not modify input parameters while the *Busy* output is equal to TRUE.

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

Interface

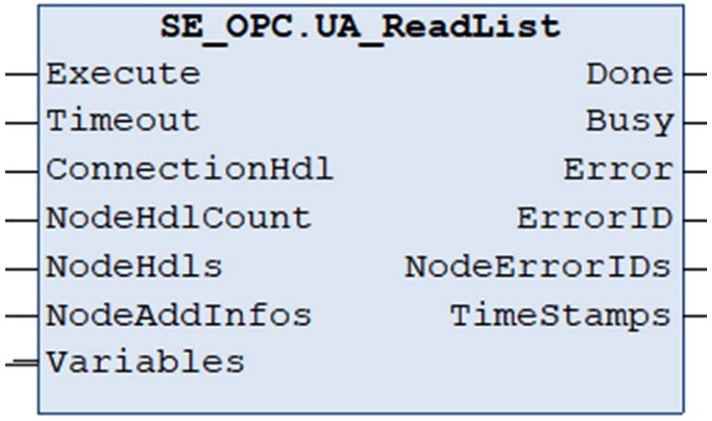
Input	Data type	Description
Execute	BOOL	Upon a rising edge, the function block is being executed. Also refer to *Behavior of Function Blocks with the Input Execute*, page 19.
Timeout	TIME	Maximum time to respond. Value range: 2 s...60 s If the value is out of range the upper or lower limit is applied. Default value: *GPL.Timeout*
ConnectionHdl	DWORD	Connection handle.
NodeHdlCount	UINT	Number of node handles in the *NodeHdls* array. Value range: *1..GPL.MAX_ELEMENTS_NODELIST]*
NodeHdls	ARRAY [1..GPL.MAX_ELEMENTS_NODELIST] OF DWORD	Array containing node handles.

Output	Data type	Description
Done	BOOL	Indicates that the execution of the function block was completed successfully.
Busy	BOOL	Indicates that the execution of the function block is in progress.
Error	BOOL	Indicates that an error was detected during execution. **NOTE:** Even if *Error* indicates FALSE, verify the corresponding *ErrorIDs* before processing the namespace indexes.
ErrorID	ET_Result, page 24	Provides additional diagnostic information as a numeric value. For each specified namespace URI, a separate result is provided.
NodeErrorIDs	ARRAY [1..GPL.MAX_ELEMENTS_NODELIST] OF ET_Result, page 24	Contains an error value for each element of the *NodeHdls* array.

UA_ReadList

Overview

Type:	Function block
Available as of:	V1.0.0.0



Functional Description

The function block *UA_ReadList* is used to read values of multiple nodes using a list of node handles.

NOTE: To help avoid an inconsistent response, do not modify parameters while the function block is executing (*Busy* = TRUE).

⚠WARNING

UNINTENDED EQUIPMENT OPERATION

Do not modify input parameters while the *Busy* output is equal to TRUE.

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

NOTE: *ByteString* is represented as a one-dimensional ARRAY OF BYTE on the client side. If *ByteString* is declared on the server side, use a buffer of type ARRAY OF BYTE and *NodeDataType UATypeByte*.

NOTE: The function block does not support the *MaxAge* feature specified by the OPC UA protocol.

Interface

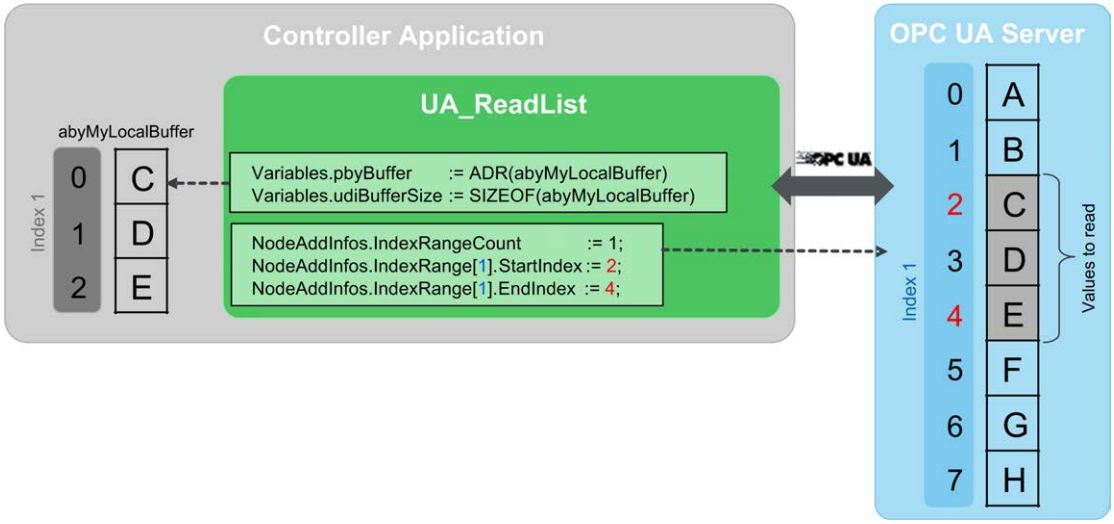
Input	Data type	Description
Execute	BOOL	Upon a rising edge, the function block is being executed. Also refer to *Behavior of Function Blocks with the Input Execute*, page 19.
Timeout	TIME	Maximum time to respond. Value range: 2 s...60 s If the value is out of range the upper or lower limit is applied. Default value: *GPL.Timeout*
ConnectionHdl	DWORD	Connection handle.
NodeHdlCount	UINT	Number of node handles in the *NodeHdls* array. Value range: *1..GPL.MAX_ELEMENTS_NODELIST]*
NodeHdls	ARRAY [1..GPL.MAX_ELEMENTS_NODELIST] OF DWORD	Array containing node handles.
NodeAddInfos	ARRAY [1..GPL.MAX_ELEMENTS_NODELIST] OF UANodeAdditionalInfo, page 47	Array containing additional node information like attribute and index range.

Input/Output	Data type	Description
Variables	ARRAY [1..GPL.MAX_ELEMENTS_NODELIST] OF ST_Variable, page 39	Array containing information about the variables to read and the corresponding memory areas. **NOTE:** Do not process the variables until the function block indicates *Done*.

Output	Data type	Description
Done	BOOL	Indicates that the execution of the function block was completed successfully.
Busy	BOOL	Indicates that the execution of the function block is in progress.
Error	BOOL	Indicates that an error was detected during execution. **NOTE:** Even if *Error* indicates FALSE, verify the corresponding *ErrorIDs* before processing the namespace indexes.
ErrorID	ET_Result, page 24	Provides additional diagnostic information as a numeric value. For each specified namespace URI, a separate result is provided.
NodeErrorIDs	ARRAY [1..GPL.MAX_ELEMENTS_NODELIST] OF ET_Result, page 24	Contains an error value for each element of the *NodeHdls* array.
TimeStamps	ARRAY [1..GPL.MAX_ELEMENTS_NODELIST] OF DATE_AND_TIME	Contains a time stamp for each valid element of the *NodeHdls* array.

Example

The following example illustrates how to read elements from an array published by the OPC UA server.



The inputs *Variables.pbyBuffer* and *Variables.udiBufferSize* describe the memory allocated inside the controller application where the read elements are written.

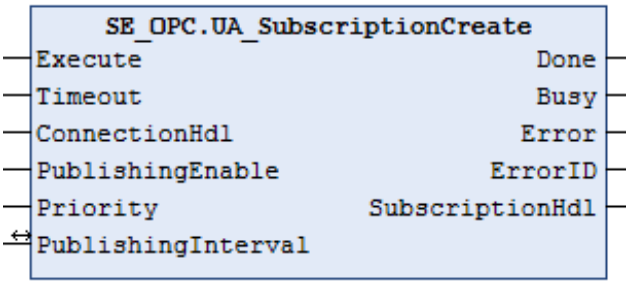
The input *NodeAddInfos* describes the elements to read from the OPC UA server.

NOTE: In case of an array of string, the buffer size must match the number of bytes returned by the OPC UA server, which depend on the data type definition on the server and the provided additional node information. For the other data types, the buffer size must be at least as large as the number of bytes received.

UA_SubscriptionCreate

Overview

Type:	Function block
Available as of:	V2.0.0.0



Functional Description

The function block *UA_SubscriptionCreate* is used to create a subscription.

Interface

Input	Data type	Description
Execute	BOOL	Upon a rising edge, the function block is being executed. Also refer to *Behavior of Function Blocks with the Input Execute*, page 19.
Timeout	TIME	Maximum time to respond. Value range: 2 s...60 s If the value is out of range the upper or lower limit is applied. Default value: *GPL.Timeout*
ConnectionHdl	DWORD	Connection handle.
PublishingEnable	BOOL	Activate the publishing. Default value: TRUE
Priority	BYTE	Priority of the subscription in the server relative to the other subscriptions created by this client. The greater the value the higher the priority.

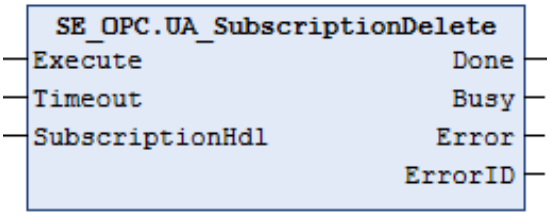
Input/Output	Data type	Description
PublishingInterval	TIME	Defines the interval the server publishes new values. The publishing interval can be modified by the server.

Output	Data type	Description
Done	BOOL	Indicates that the execution of the function block was completed successfully.
Busy	BOOL	Indicates that the execution of the function block is in progress.
Error	BOOL	Indicates that an error was detected during execution. **NOTE:** Even if *Error* indicates FALSE, verify the corresponding *ErrorIDs* before processing the namespace indexes.
ErrorID	ET_Result, page 24	Provides additional diagnostic information as a numeric value. For each specified namespace URI, a separate result is provided.
SubscriptionHdl	DWORD	Subscription handle.

UA_SubscriptionDelete

Overview

Type:	Function block
Available as of:	V2.0.0.0



Functional Description

The function block *UA_SubscriptionDelete* is used to delete a subscription.

Interface

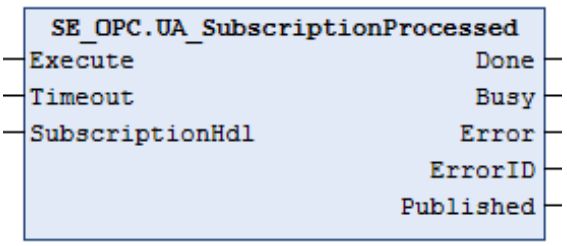
Input	Data type	Description
Execute	BOOL	Upon a rising edge, the function block is being executed. Also refer to *Behavior of Function Blocks with the Input Execute*, page 19.
Timeout	TIME	Maximum time to respond. Value range: 2 s...60 s If the value is out of range the upper or lower limit is applied. Default value: *GPL.Timeout*
SubscriptionHdl	DWORD	Subscription handle.

Output	Data type	Description
Done	BOOL	Indicates that the execution of the function block was completed successfully.
Busy	BOOL	Indicates that the execution of the function block is in progress.
Error	BOOL	Indicates that an error was detected during execution. **NOTE:** Even if *Error* indicates FALSE, verify the corresponding *ErrorIDs* before processing the namespace indexes.
ErrorID	ET_Result, page 24	Provides additional diagnostic information as a numeric value. For each specified namespace URI, a separate result is provided.

UA_SubscriptionProcessed

Overview

Type:	Function block
Available as of:	V2.0.0.0



Functional Description

The function block *UA_SubscriptionProcessed* can optionally be called to verify whether the values of multiple monitored items out of a subscription have been published.

NOTE:

The function block *UA_SubscriptionProcessed* can only be used in combination with the firmware synchronization mode *SyncMode UAMS_FwSync*, page 31. With this synchronization mode selected, the values are published automatically by the underlying system.

Interface

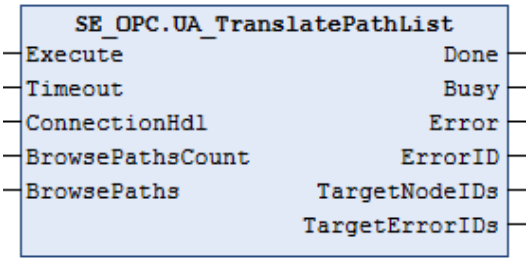
Input	Data type	Description
Execute	BOOL	Upon a rising edge, the function block is being executed. Also refer to *Behavior of Function Blocks with the Input Execute*, page 19.
Timeout	TIME	Maximum time to respond. Value range: 2 s...60 s If the value is out of range the upper or lower limit is applied. Default value: *GPL.Timeout*
SubscriptionHdl	DWORD	Subscription handle.

Output	Data type	Description
Done	BOOL	Indicates that the execution of the function block was completed successfully.
Busy	BOOL	Indicates that the execution of the function block is in progress.
Error	BOOL	Indicates that an error was detected during execution. **NOTE:** Even if *Error* indicates FALSE, verify the corresponding *ErrorIDs* before processing the namespace indexes.
ErrorID	ET_Result, page 24	Provides additional diagnostic information as a numeric value. For each specified namespace URI, a separate result is provided.
Published	BOOL	Indicates that variables have been published since the last call.

UA_TranslatePathList

Overview

Type:	Function block
Available as of:	V2.0.0.0



NOTE: To help avoid an inconsistent response, do not modify parameters while the function block is executing (*Busy* = TRUE).

⚠WARNING

UNINTENDED EQUIPMENT OPERATION

Do not modify input parameters while the *Busy* output is equal to TRUE.

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

Functional Description

The function block *UA_TranslatePathList* is used to get node parameters of a node using the path of the node for multiple nodes.

Interface

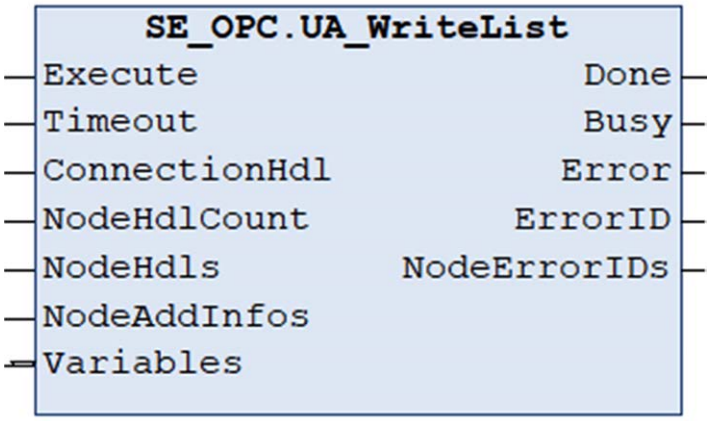
Input	Data type	Description
Execute	BOOL	Upon a rising edge, the function block is being executed. Also refer to *Behavior of Function Blocks with the Input Execute*, page 19.
Timeout	TIME	Maximum time to respond. Value range: 2 s...60 s If the value is out of range the upper or lower limit is applied. Default value: *GPL.Timeout*
ConnectionHdl	DWORD	Connection handle.
BrowsePathsCount	UINT	Number of paths of nodes in the *BrowsePaths* array. Must not exceed the size defined with *GPL. MAX_ ELEMENTS_RELATIVEPATH* or *GPL.MAX_ ELEMENTS_NODELIST*.
BrowsePaths	ARRAY [1..*GPL. MAX_ELEMENTS_ RELATIVEPATH*] OF *UABrowsePath*	Array containing paths of nodes.

Output	Data type	Description
Done	BOOL	Indicates that the execution of the function block was completed successfully.
Busy	BOOL	Indicates that the execution of the function block is in progress.
Error	BOOL	Indicates that an error was detected during execution. **NOTE:** Even if *Error* indicates FALSE, verify the corresponding *ErrorIDs* before processing the namespace indexes.
ErrorID	ET_Result, page 24	Provides additional diagnostic information as a numeric value. For each specified namespace URI, a separate result is provided.
TargetNodeIDs	ARRAY [1..*GPL. MAX_ELEMENTS_NODELIST*] OF *UANodeID*	Contains node parameters for each target node indicated inside the *BrowsePaths* array.
TargetErrorIDs	ARRAY [1..*GPL. MAX_ELEMENTS_NODELIST*] OF *ET_Result*	Contains an error value for each element of the *BrowsePaths* array.

UA_WriteList

Overview

Type:	Function block
Available as of:	V1.0.0.0



Functional Description

The function block *UA_WriteList* is used to write values of multiple nodes using a list of node handles.

NOTE: To help avoid an inconsistent response, do not modify parameters while the function block is executing (*Busy* = TRUE).

⚠WARNING

UNINTENDED EQUIPMENT OPERATION

Do not modify input parameters while the *Busy* output is equal to TRUE.

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

NOTE: *ByteString* is represented as a one-dimensional ARRAY OF BYTE on the client side. If *ByteString* is declared on the server side, use a buffer of type ARRAY OF BYTE and *NodeDataType UATypeByte*.

NOTE: The function block does not support the *MaxAge* feature specified by the OPC UA protocol.

Interface

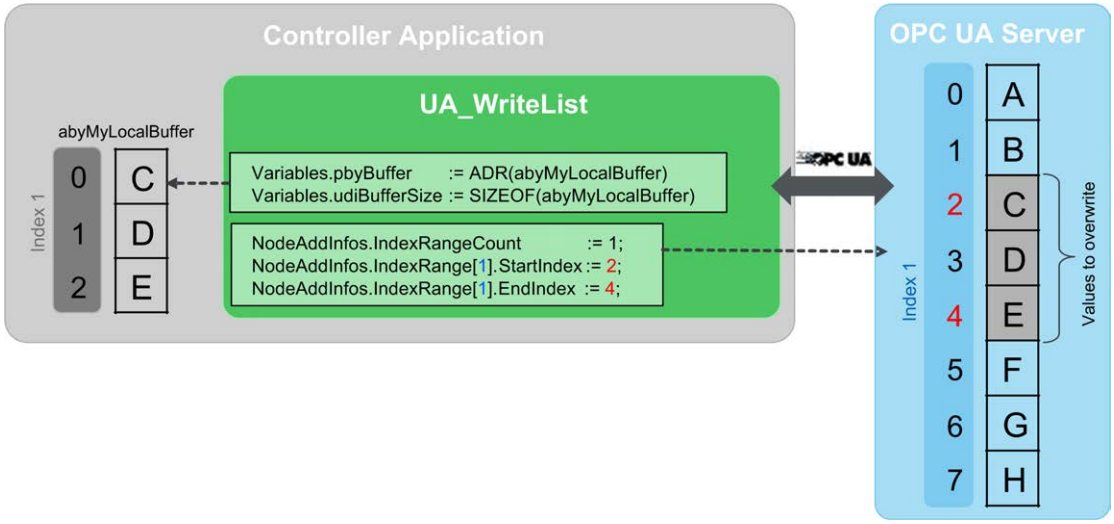
Input	Data type	Description
Execute	BOOL	Upon a rising edge, the function block is being executed. Also refer to *Behavior of Function Blocks with the Input Execute*, page 19.
Timeout	TIME	Maximum time to respond. Value range: 2 s...60 s If the value is out of range the upper or lower limit is applied. Default value: *GPL.Timeout*
ConnectionHdl	DWORD	Connection handle.
NodeHdlCount	UINT	Number of node handles in the *NodeHdls* array. Value range: *1..GPL.MAX_ELEMENTS_NODELIST]*
NodeHdls	ARRAY [1..GPL.MAX_ELEMENTS_NODELIST] OF DWORD	Array containing node handles.
NodeAddInfos	ARRAY [1..GPL.MAX_ELEMENTS_NODELIST] OF UANodeAdditionalInfo, page 47	Array containing additional node information like attribute and index range.

Input/Output	Data type	Description
Variables	ARRAY [1..GPL.MAX_ELEMENTS_NODELIST] OF ST_Variable, page 39	Array containing information about the variables to read and the corresponding memory areas. **NOTE:** Do not process the variables until the function block indicates *Done*.

Output	Data type	Description
Done	BOOL	Indicates that the execution of the function block was completed successfully.
Busy	BOOL	Indicates that the execution of the function block is in progress.
Error	BOOL	Indicates that an error was detected during execution. **NOTE:** Even if *Error* indicates FALSE, verify the corresponding *ErrorIDs* before processing the namespace indexes.
ErrorID	ET_Result, page 24	Provides additional diagnostic information as a numeric value. For each specified namespace URI, a separate result is provided.
NodeErrorIDs	ARRAY [1..GPL.MAX_ELEMENTS_NODELIST] OF ET_Result, page 24	Contains an error value for each element of the *NodeHdls* array.

Example

Following example illustrates how to write elements to an array published by the OPC UA server.



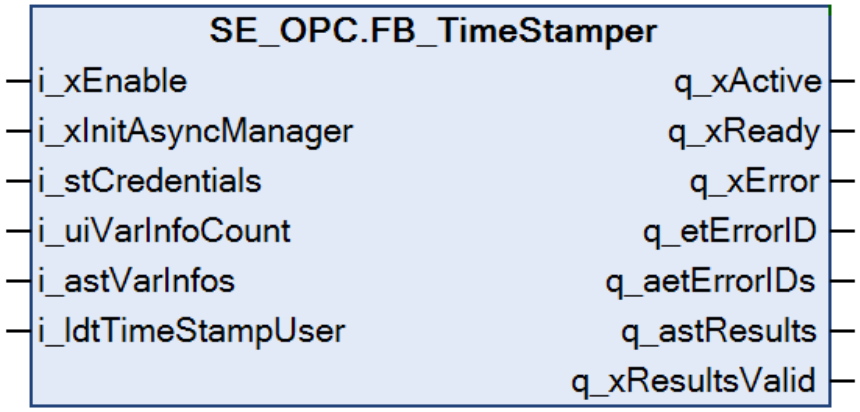
The inputs *Variables.pbyBuffer* and *Variables.udiBufferSize* describe the memory allocated inside the controller application containing the elements to write.

The input *NodeAddInfos* describes the elements of the OPC UA server to overwrite.

FB_TimeStamper

Overview

Type:	Function block
Available as of:	V2.1.6.0



Functional Description

The function block *FB_TimeStamper* is used to create a list of time-stamped variables which will be stored in the RAM for the OPC UA server. Only one function block instance is allowed at a time.

NOTE: The function block *FB_TimeStamper* is not supported by PacDrive LMC controllers.

NOTE: As the variables are time-stamped according to the real-time clock (RTC) of the controller, set the RTC to the correct time. As an alternative, you can provide a user-specific timestamp at the input *i_ldtTimeStampUser*.

By enabling the function block, it is verified whether the variables are available inside the symbol configuration and the timestamp is set. A maximum of 128 variables can be time-stamped.

NOTE: The function block uses an asynchronous task to verify the existence of the variables inside the symbol configuration. If the asynchronous manager is not initialized inside the application and the input *i_xInitAsyncManager* is set to TRUE, the function block initializes the asynchronous manager. This process increases the execution time of the function block for one cycle. Consider this in the watchdog configuration of the calling task.

NOTE: Do not modify the variable information at the input of the function block while the function block is active. To update the variables that will be time-stamped, you must disable and re-enable the function block.

The cycle time of the task calling the function block corresponds to the sample rate to detect value changes of the variables to time stamp. With each function block call, the values of the variables referenced by the pointer inside the input structure *i_astVarInfos* are compared with the respective values at the previous function block call. Only value changes slower than the interval between two function block calls can be detected. If a value change has been detected, the output *q_astResults* is updated with the new values and the new timestamp. Process the values provided by *q_astResults* only if the respective variables inside *q_aetErrorIDs* indicate no error and *q_xResultsValid* is TRUE.

NOTE: The output *q_astResults* can be updated with every function block call. Consider this if you access *q_astResults* from another task.

The function block *FB_TimeStamper* must be available on the same controller as the OPC UA server. The OPC UA server gets the reference to the function block output *q_astResults* when the function block is enabled and with every online change of the application. If the function block is enabled, the OPC UA server accesses the output *q_astResults* to retrieve, for example, the source timestamp of a variable for the OPC UA client.

NOTE: While the OPC UA server accesses the output *q_astResults*, the comparison of the values of the variables to time stamp inside the function block is skipped and the data of *q_astResults* is not updated.

Interface

Input	Data type	Description
i_xEnable	BOOL	Activation and initialization of the function block.
i_xInitAsyncManager	BOOL	If TRUE, the asynchronous manager is initialized by the function block during the first initialization (if not yet done). If FALSE, an error message is returned in case the asynchronous manager has not been initialized.
i_stCredentials	*ST_Credentials*	Credentials to access variables of the symbol configuration if symbol sets are enabled.
i_uiVarInfoCount	UINT	Number of variables to time stamp. Value range: 1...128
i_astVarInfos	ARRAY [1..*GPL.Gc_uiMaxVariablestimeStamper*] OF *ST_VarInfo*	Structure containing information of variables to time stamp.
i_ldtTimeStampUser	LDATE_AND_TIME	User-specific time stamp. If this value is not assigned, the RTC of the controller is used.

Output	Data type	Description
q_xActive	BOOL	If the function block is active, the output is set to TRUE.
q_xReady	BOOL	If the initialization is successful, the output is set to TRUE as long as the function block is operating.
q_xError	BOOL	If this output is set to TRUE, an error has been detected.
q_etErrorID	*ET_Result*	Provides diagnostic and status information.
q_aetErrorIDs	ARRAY [1..*GPL.Gc_uiMaxVariablesTimeStamper*] OF *ET_Result*	Provides diagnostic and status information for each monitored variable.
q_astResults	ARRAY [1..*GPL.Gc_uiMaxVariablesTimeStamper*] OF *ST_TimeStampResults*	Structure containing information about monitored variables including the timestamp.
q_xResultsValid	BOOL	Indicates if value-timestamp pairs available at the output *q_astResults* are valid and can be processed. Verify this output before you access *q_astResults* from another task.

Functions

What's in This Chapter

FC_•••ToString

Overview

Example of one of the functions that convert an enumeration, in this case *EtResult*, to a string variable. To convert other enumerations, use the name of the enumeration as part of the function name preceded by *FC_*.

Task

Converts a variable of the corresponding enumeration type, page 22 to a variable of type STRING.

Interface

Input	Data type	Description
i_etResult	Corresponding enumeration of this library.	Enumeration to be converted.

Return Value

Data type	Description
STRING(80)	Provides the corresponding text.

Glossary

F

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.

O

OPC UA:

OPC Unified Architecture: OPC UA is an interoperability standard for the secured and reliable exchange of data in the industrial automation space. It is a platform independent communication protocol using the server/client model. The connection between client and server is commonly based on the reliable transport layer protocol (TCP, Transmission Control Protocol).

For more information about the OPC especially OPC UA refer to the official webpage of the OPC Foundation at https://opcfoundation.org.

T

TCP:

(*transmission control protocol*) A connection-based transport layer protocol that provides a simultaneous bi-directional transmission of data. TCP is part of the TCP/IP protocol suite.

Index

C

E

X

Schneider Electric
35 rue Joseph Monier
92500 Rueil Malmaison
France

+ 33 (0) 1 41 29 70 00

www.se.com

As standards, specifications, and design change from time to time, please ask for confirmation of the information given in this publication.

EIO0000004021.05