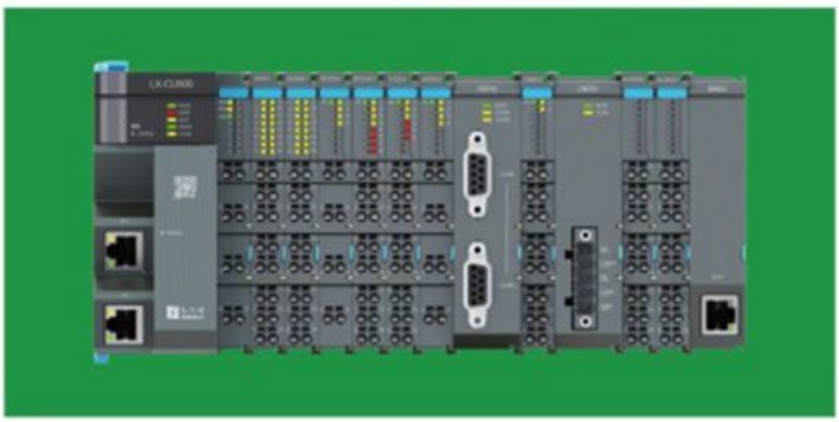


Basic Instruction Manual for
LX Series Programmable Logic Controllers

Version:2026.06

Copyright Notice

Address: Di Sheng Middle Road, No.2,
Economic-Technological Development Area, 100176, Beijing, China
Tel: +86 010-5898 1588
Consulting Hotline: 400-811-1999
Technology Request: +86 010-58981514
Fax: +86 010-5898 1558
Web: http://www.hollysys.com
FA Web: https://fa.hollysys.com/
Email: PLC@hollysys.com
Sina weibo: http://weibo.com/hollysysplc

Table of Contents

Chapter 1 Foreword

1.1 Purpose

This document will introduce the use of basic commang in the LX system, including product introduction, module components, technical parameters, indicators, wiring instructions, diagnostic information, etc. Please use it in conjunction with other supporting product materials to help you have a more comprehensive understanding of how to use the modules.

1.2 Object

This manual is intended for use by engineers or related professional technicians with expertise in automation and control systems.

1.3 Target Products

This manual is applicable to the following products:

- LX series PLC products

1.4 Statement of Use

- The contents of this manual have been tested according to regulations, and the diagrams are consistent with the software and hardware products described. However, errors are inevitable, and complete consistency cannot be guaranteed. If you have any suggestions for improving or correcting the contents of this manual, please let us know in time and we will make corrections in the next version.
- Due to product iteration updates, the relevant parameters and information in this manual may change without prior notice.
- Since the devices described in this manual can be used in a variety of ways, the user and the person responsible for the use of the devices must ensure that each method is admissible. HollySys will not be legally responsible for any direct or indirect losses resulting from the use or misuse of these devices.
- Due to uncertainties in the actual application, HollySys assumes no responsibility for the direct use of the data provided in this manual.

1.5 Safety Precautions

1.5.1 Safety Statement

1. Please read and comply with these safety precautions before using this product.
2. To ensure personal and device safety, please strictly abide by the safety precautions on the product label and in the manual when using this product.
3. The words "Note", "Caution", "Warning" and "Danger" in this manual do not represent all the safety precautions that shall be observed, but only serve as a supplement to all precautions.
4. This product shall be used in an environment that meets the design specifications. Otherwise, it may cause faults. Device damage caused by failure to comply with relevant regulations is not covered by the product quality guarantee.
5. HollySys shall not be responsible for any personal safety accidents or property losses caused by any operation that does not comply with the provisions of this manual or does not meet the requirements.

1.5.2 Warning

For your safety and to avoid property losses, please pay attention to the warnings in this manual. The following warnings are indicated according to the hazard level from high to low. When there are multiple hazard levels, only the highest level is prompted. If the highest level is a warning that may cause personal injury, there may also be a warning that may cause property loss.

Warnings on personal safety: Indicated by a warning triangle  .

Warnings on property loss: Without any warning triangle.

 Danger
It indicates that death or serious personal injury may be caused if operations are not carried out as specified.

 Warning
It indicates that death or serious personal injury may be caused if operations are not carried out as specified.

 Caution
It indicates that minor personal injuries may be caused if operations are not carried out as specified.

Note
It indicates that property losses may be caused if operations are not carried out as specified.

Chapter 2 Document Guide

2.1 Related Manuals

The available manuals for this product are displayed in the table below. Please choose and refer to them based on your specific usage requirement.

Manual Name	Purpose	Content
Module Manual for LX Series Programmable Logic Controllers	Learn about the hardware modules of the LX system	Describes all hardware modules, including: Module function Hardware interface Wiring Indicator Technical parameters
Communication Manual for LX Series Programmable Logic Controllers	Guide users to build the LX system communication network	Describe the communication networks supported by the LX system, including: Introduction to communication protocols Function usage
Programming Manual for LX Series Programmable Logic Controllers	Learn about the use of FA-AutoThink programming software	Describe the interface, functions, and related operations of FA-AutoThink programming software, including: Software interface introduction Project creation Programming Online commissioning function
Motion Instruction Manual for LX Series Programmable Logic Controllers	Learn about the use of motion control instructions of the LX system	Describe the use of motion instructions, including: Instruction function Parameter Example
PLCopen Instruction Manual for LX Series Programmable Logic Controllers	Learn about the use of PLCopen motion control instructions of the LX system	Describe the use of PLCopen instructions, including: Parameter Function Example Notes for Using See Synonyms at force
Feedback Control Algorithm Instruction Manual for LX Series Programmable Logic Controllers	Learn about the use of feedback control algorithm instructions of the LX system	Describe the use of feedback control algorithm instructions, including: Parameter Function Example Notes for Using

2.2 Usage Convention

The following symbols are used in this manual:

-  Tip: This symbol indicates helpful tips or suggestions for using the product more efficiently.

2.3 How to Get the Manual

If you need the electronic PDF file of this manual, kindly access it through our website.

- The official website of HollySys (https://www.hollyfa.com/LX-CU500.html#c_relevant_052-17326776315060) to download the file.

Chapter 3 Revision History

Version	Revision Date	Revision Content
V1.0.0	July 2023	Created
V1.1.0	October 2023	Added instructions: CTUD (Up-down Counter); IEC_SFC; UDP_CONNECT_EXT2 (Creating UDP Communication (Supporting Multicast/Broadcast)); TaskSetWdtTimeout (Setting Task Watchdog Timeout Period); EtherCAT_SetAxisOperationMode (Setting Axis Mode of Operation), EtherCAT_ SetAxisCtrWord (Setting Axis Control Word), EtherCAT_GetSlaveODOffset (Getting Slave Station Object Dictionary IQ Area Offset), EtherCAT_GetSlaveLinkInfo (Getting Slave Station Port Link Information); EtherNetIp Protocol Related Instruction Library Deleted instructions: SAFE_COPY_BLOCK (safe copy of data block) instruction
V1.1.1	December 2023	Modified the data in parameter tables in sections 6.6.1.2 and 6.6.2.2
V1.1.2	January 2024	Added the description on full-text instructions, parameter table default value, and power-failure protection
V1.2.0	October 2024	Added new instructions: MB_CLIENT, MbSetComFrameEndDelay, TaskCycleGet, sysGetDataAreaAddrById,TCP_CONNECT2,UDP_CONNECT2
V1.3.0	January 2025	Update TP command and DataMemCmp command examples
V1.4.0	March 2025	Added: DeviceNet_Atrribute_WriteRead (Write Read DeviceNet Slave Atrribute), DeviceNet_DiagInfo_Read (Read DeviceNet Slave Diag Info), PN_Get_Channel_Diag (Read PROFINET Slave Channel Diag Info), PN_Get_Warning_Diag (Read PROFINET Slave Warning Diag Info), PN_Get_Mismatch_Diag (Read PROFINET Slave Mismatch Diag Info), SlSetConfig (Set the parameters of the SL protocol server), SlGetDiag (Get SL protocol server pDiagnose Information)
V1.5.0	September 2025	Added: LXCANopenReadDict(Read CANopen Slave EDS), LXCANopenWriteDict(Write CANopen Slave EDS), LXCANopenNodeNMT(Send NMT command specifier), CIPOpen (CIP Class3 Create Connection), CIPRead (CIP Class3 Read Tag), CIPWrite (CIP Class3 Write Tag), CIPClose (CIP Class3 Close Connection), SLConfig (Set the parameters of the SL peotocol), SLConnect (Set SL protocol server connection), SLDiag (Get SL protocol server Diagnose Information)
V1.6.0	June 2026	New commands: MbMappingConfig (Set/Get Modbus Slave Mapping Offset), LXComCapture (Serial Port Capture), LXComMBMaster (MODBUS Master (capture)), LXComMBSlave (MODBUS Slave(capture)), LXComFreePortSend (Free Protocol Communication Data Transmission(capture)),LXComFreePortReceive (Free Protocol Communication Data Reception(capture)), sysSetEthIp (Set IP Address), sysGetEthIp (Get IP Address) Updated commands: LX-CU433 / LX-CU500 / LX-CU501 / LX-CU510 / LX-CU511 support CIP Class3 related commands, and LX-CU430 / LX-CU433 / LX-CU500 / LX-CU501 / LX-CU510 support SLConfig, SLConnect, and SLDiag. Updated EtherCAT error codes.

Chapter 4 Overview of Instructions

4.1 List of Instructions

Instruction Library	Instruction	Name	Function Description
Arithmetical Operation	ADD	Add	Implement the sum operation of two or more variables/constants.
	SUB	Subtract	Implement subtraction between two variables or constants.
	MUL	Multiply	Implement multiplication of two or more variables or constants.
	DIV	Divide	Divide variables or constants
	MOD	Take remainder	Take remainder after division of variables or constants.
	SIZEOF	Byte length	Number of bytes required for getting the data type.
	ABS	Absolute value	Assign the absolute value of the input data to the output variable.
	SQRT	Square root	Find the square root of the input data.
	LOG	Common logarithm	Find the logarithm of the input data in base 10.
	LN	Natural logarithm	Find the natural logarithm of the input data.
	EXP	Exponent	Calculate by raising the input data to the power of the exponent.
	SIN	Sine	Find the sine of the input data.
	COS	Cosine	Find the cosine of the input data.
	TAN	Tangent	Find the tangent of the input data
	ASIN	Arcsine	Find the arcsine of the input data.
	ACOS	Arc cosine	Find the arc cosine of the input data.
	ATAN	Arc tangent	Find the arc tangent of the input data.
	EXPT	Power	Calculate the power of x as the base and y as the exponent.
Logic operation	AND	AND	AND operation of variables or constants.
	OR	Or	OR operation of variables or constants.
	XOR	XOR	XOR operation of variables or constants.
	NOT	No	NOT operation of variables or constants.
Comparison operation	GT	Greater than	Determine the size of two operands.
	LT	Smaller than	Determine the size of two operands.
	GE	Greater Than or Equal to	Determine the size of two operands.

	LE	Smaller than or equal to	Determine the size of two operands.
	EQ	Equal to	Determine whether two operands are equal.
	NE	Not equal to	Determine whether two operands are not equal.
Selection operation	SEL	Either-or	Use the selector switch to select one of the two input data as the output.
	MAX	Take the maximum value	Select the maximum value among the two input data as the output.
	MIN	Take the minimum value	Select the minimum value among the two input data as the output.
	MUX	Multiple choice	Use the control number to select one of multiple input data as the output.
	LIMIT	Amplitude limiting	Determine whether the input data is between the minimum and maximum values.
Shift operation	SHL	Shift left	Shift operands to the left by bit.
	SHR	Shift right	Shift operands to the right by bit.
	ROL	Ring shift left	Ring shift operands to the left by bit.
	ROR	Ring shift right	Ring shift operands to the right by bit.
Data type conversion	BOOL_TO_TYPE	BOOL type conversion	Convert a Boolean data type to another data type.
	BYTE_TO_TYPE	BYTE type conversion	Convert the byte type to other data types.
	DATE_TO_TYPE	DATE type conversion	Convert a date type data to another data type.
	DINT_TO_TYPE	DINT type conversion	Convert the double integer type to other data types.
	DT_TO_TYPE	DT type conversion	Convert the date/time type data to other data types.
	DWORD_TO_TYPE	DWORD type conversion	Convert the double word type to other data types.
	INT_TO_TYPE	INT type conversion	Convert the integer data type to other data types.
	LREAL_TO_TYPE	LREAL type conversion	Convert the long real number to other data types.
	REAL_TO_TYPE	REAL type conversion	Convert the real number to other data types.
	SINT_TO_TYPE	SINT type conversion	Convert the short integer type to other data types.
	STRING_TO_TYPE	STRING type conversion	Convert variables of a STRING type to another data type.
	TIME_TO_TYPE	TIME type conversion	Convert the time type data to other data types.
	TOD_TO_TYPE	TOD type conversion	Convert the time type data to other data types.
	UDINT_TO_TYPE	UDINT type conversion	Convert unsigned double integer type to another data type.
	UINT_TO_TYPE	UINT type conversion	Convert the unsigned integer type to other data types.
	USINT_TO_TYPE	USINT type conversion	Convert the unsigned short integer type to other data types.
	WORD_TO_TYPE	WORD type conversion	Convert the word type to other data types.

	TRUNC	Floating number to integer	Implement the trunc function.
Pointer	ADR	Getting address	Get the memory address of the input variable.
	VAL	Value	Get the data at the address pointed by the pointer.
String	Left	String left	Specify a substring consisting of the leftmost count characters of the specified string.
	Right	String right	Specify a substring consisting of the rightmost count characters of the specified string.
	Concat	String concatenation	Concatenate two strings.
	Delete	String deletion	Delete len characters starting from the pos character of str.
	Insert	String insertion	Insert str2 into the pos character of str1.
	Mid	String middle	A substring consisting of len consecutive characters starting from the pos character in the string.
	Replace	String replacement	Use str2 to replace len characters starting from the pos character in str1
	Find	String search	Find the first character position of str2 in str1.
	Len	String length	Enter the length of the string
Edge trigger instruction	F_TRIG	Falling edge detection trigger	Detect falling edge
	R_TRIG	Rising edge detection trigger	Detect rising edge.
Bistable instruction	RS	Reset priority bistable device	Implement the reset priority RS trigger function.
	SR	Setting priority bistable device	Implement the setting priority RS trigger function.
Counter instruction	CTD	Count-down counter	Implement the function of decrementing the count value when the input CD has a rising edge.
	CTU	Count-up counter	Implement the function of increasing the count value when the input CU has a rising edge.
Timer instruction	TP	Ordinary timer	Set the output to a preset period of time.
	TON	Power-on delay timer	Set the output to delay for a specified period of time.
	TOF	Off-delay timer	Reset the output to a preset period of time.
	TONR	Hold power-on delay timer	Accumulate the time value within the set time period.
IEC_SFC	SFCActionControlBlock	SFC action control block	This instruction realizes the control of IEC steps in SFC.
BCD conversion instruction	BCD_TO_INT	BCD code to integer	Convert BCD code to an integer value algorithm.
	INT_TO_BCD	Integer to BCD code	Convert an integer value to BCD code.
Bit_byte operation instruction	UNPACK	Bit split instruction	Take out the 7th to 0th bits of the input and assign them to the output.
	EXTRACT	Bit extraction instruction	Output the specified Nth bit of input variable X.

	PACK	Bit integration instruction	Combine B0~B7 into a BYTE variable in order from bit 0 to bit 7.
	PUTBIT	Bit assignment command	Take the Nth bit specified by the input value X as the specified value B.
Mathematical auxiliary function	DERIVATIVE	Differential	Perform differential operations on continuous input variables.
	INTEGRAL	Integral	Approximately solve the integral of the input function.
	STATISTICS_INT	Integer statistics	Count the minimum, maximum, and average values of all INT type input variables.
	STATISTICS_REAL	Real statistics	Count the maximum, minimum, and average values of all REAL type input variables.
	VARIANCE	Square deviation	Calculate the variance of all input variables.
Controller instruction	PID	PID controller	Perform PID adjustment control on the input.
Signal generator instruction	BLINK	Pulse signal generator	Output high-level and low-level rectangular waves at specified times.
	GEN	Typical periodic signal generator	Generate a signal specifying the type, period, and amplitude of a periodic function.
	RAND	Random number generator	Generate random numbers.
Analog processing instruction	RAMP_INT	Integral speed limit	Limit the rising and falling speed of an integer input function.
	RAMP_REAL	Real speed limit	Limit the rising and falling speed of a real input function.
	CHARCURVE	Characteristic curve	Generate a characteristic row in the DINT type arrays PX[0..10] and PY[0..10].
	HYSTERESIS	Hysteresis	Compare the input value to the upper and lower limits.
	LIMITALARM	Upper and lower limit alarm	Compare the input value to the upper and lower limits.
	HEX_ENGIN	Hexadecimal numbers to engineering quantity data	Convert the input code value WH into the corresponding engineering quantity AV value.
	ENGIN_HEX	Engineering quantity data to hexadecimal WORD data	Convert the input engineering quantity AV into the corresponding code value WH.
	ENGIN_HEX2	Engineering quantity data to hexadecimal INT data	Convert the input engineering quantity AV into the corresponding code value WH.
Data block transfer instruction	MOVE_BLOCK	Data block transfer	Transfer data in the M area from one location to another location in the M area in batches.
	MOVE_BLOCK_EXT	Data block transfer extension instruction	Transfer data in the M/I/Q/S area to the M/I/Q area in batches.
	SAFE_COPY_BLOCK	Safe copy of data block	Copy data in the G/M buffer to the corresponding I/O channel address at one time.
Communication instruction	MB_CLIENT	Modbus TCP master communication	Establish connections between clients and servers, send Modbus requests, and receive responses.
	GENERATE_CRC	CRC check calculation	Generate a 16-bit CRC check value.

	ComConfig	Set serial port communication parameters	Configure the communication parameters of each channel online.
	ComRecv	Receiving serial port data	The serial port receives data.
	ComSend	Sending serial port data	The serial port sends data.
	ComClearRecvFIFO	Clearing serial port received data	Clear serial port Rx data.
	TCP_CONNECT	Establishing TCP communication connection	Establishing TCP communication connection
	TCP_CONNECT2	Establishing a TCP Communication Connection(support communication by specific nices)	Establish TCP communication connection.
	TCP_SEND	TCP data sending	Use TCP protocol for Ethernet data sending.
	TCP_RECV	TCP data reception	Use the TCP protocol for Ethernet data reception.
	UDP_CONNECT2	Establishing a UDP Communication Connection(support communication by specific nices)	Establish UDP communication connection.
	UDP_CONNECT_EXT	Creating UDP communication (supporting multicast/broadcast)	Create a passive connection using UDP protocol.
	UDP_SEND	UDP data sending	UDP protocol and connections created via UDP_CONNECT.
	UDP_RECV	UDP data reception	UDP protocol and connections created via UDP_CONNECT.
	ETH_DISCONNECT	Closing Ethernet communication connection	Terminate the existing communication connection.
	MbSetComFrameEndDelay	Set serial port Modbus frame end delay time	Used to set the frame end delay time for Modbus RTU communications on COM ports.
	MbMappingConfig	Set/Get Modbus Slave Mapping Offset	Set or obtained the address mapping relationship of Modbus TCP and Modbus RTU in the controller.
	LXComModbusRTUMaster	MODBUS master	Realize the Modbus RTU master station function together with the LX-CM series serial port communication module.
	LXComModbusRTUSlave	MODBUS slave	Realize the Modbus RTU slave station function together with the LX-CM series serial port communication module.
	LXComSerialLineControl	Free protocol	This functional block, together with the Rx/Tx functional block and the LX-CM series serial port communication module, realizes free protocol communication and can complete the Rx and Tx functions.
	LXComReceive	Receiving data through free protocol	Receive third-party module data transmitted through the serial port module together with LXComSerialLineControl.

	LXComSend	Sending data through free protocol	Send data to a third-party module through the serial port module together with LXComSerialLineControl.
	LXComClearBuf	Clearing buffer through free protocol	Clear the internal PLC communication buffer.
	LXComClearCom	Clearing serial port module buffer	Clear the Tx and Rx buffers of the current serial port channel.
	SlSetConfig	Set the parameters of the SL protocol server	Set the SL protocol server parameters.
	SlGetDiag	Get SL protocol server pDiagnose Information	Retrieve SL Protocol Server Diagnostic Information.
	SLConfig	Set the parameters of the SL peotocol	Through this function block, you can set, retrieve, or reset parameters for soft components.
	SLConnect	Set SL protocol server connection	Set SL protocol server connection.
	SLDiag	Get SL protocol server Diagnose Information	Get SL protocol server status.
	LXComCapture	Serial port packet capture	Obtain the baud rate, data bits, stop bits, parity bits, and frame interval information of the corresponding channel of the LX-CM series serial module for packet capturing.
	LXComMBMaster	MODBUS Master(capture)	View the status of Modbus RTU master station messages and use the "Serial Port Packet Capture" tool to parse received or sent slave station messages.
	LXComMBSlave	MODBUS Slave(capture)	View the status of Modbus RTU slave station messages and use the "Serial Port Packet Capture" tool to parse the messages.
	LXComFreePortSend	Free Protocol Communication Data Transmission(capture)	Implement the function of sending data via a free protocol.
	LXComFreePortReceive	Free Protocol Communication Data Reception(capture)	Implement the function of receiving data via a free protocol.
Task instruction	TaskStart	Running task	Run the specified task.
	TaskStop	Stopping task	Stop the specified task.
	TaskStatus	Return to task status	Return to the current status of the task.
	TaskTestCancel	Setting task cancellation point	Set the task cancellation point function.
	TaskExit	Exiting task	Users can call this function in task configuration to end the execution of this task.
	TaskLock	System lock locking (blocking)	When the lock is not acquired, the task thread is blocked and suspended until other tasks release the lock.
	TaskTryLock	System lock locking (non-blocking)	Users need to decide whether to access resources (data) shared between different tasks based on the return value of this interface. It can be unlocked only after the lock is locked successfully.
	TaskUnlock	System lock unlocking	After the system lock is locked, TaskUnlock provides an unlocking interface.

	TaskCycleGet	Get the task cycle run by the feature block	Period time used to get the IEC task where the feature block is located.
File operation	FileOpen	Opening file	Open a file according to the method set by the user.
	FileClose	Closing file	Close a file.
	FileRead	Reading file	Read a file.
	FileWrite	Writing file	Write a file.
	FileRead	Reading file 2	Read file 2.
	FileWrite	Writing file 2	Write file 2.
	FileLseek	Moving file read/write pointer	Move the file read/write pointer.
	FileListScan	Scanning directory files	Scan all files contained in the specified directory.
	FindFirstFile	Searching first file	Open a file handle that traverses the directory and obtain the first file name in the directory.
	FindNextFile	Find the next file	Traverse the directory and get the next file name in the specified directory.
	FindClose	Search end	Close the traversal handle.
	FileRemove	Deleting files	Delete a file.
Time related instruction	GetTickCnt	Getting boot count	Get the time since the controller was powered on.
	TimeDelay	Time delay	Achieve the effect of time delay.
	SET_RTC	Setting real-time clock	Set the user-specified time into the PLC real-time clock.
	GET_RTC	Reading real-time clock	Read the time in the PLC hardware real-time clock, which is the UTC+08:00 time.
	sysGetTimepulse_1min	Getting 1-minute pulse status	Output a square wave pulse signal with a period of 1 minute, with the state flipping every 30 seconds.
	sysGetTimepulse_1s	Getting 1-second pulse status	Output a square wave pulse signal with a period of 1 second. The return value status flips every 500 milliseconds.
	sysGetTimepulse_200ms	Getting 200-millisecond pulse status	Output a square wave pulse signal with a period of 200 milliseconds. The return value status flips every 100 milliseconds.
	sysGetTimepulse_100ms	Getting 100-millisecond pulse status	Output a square wave pulse signal with a period of 100 milliseconds. The return value status flips every 50 milliseconds.
	sysGetTimepulse_20ms	Getting 20-millisecond pulse status	Output a square wave pulse signal with a period of 20 milliseconds. The return value status flips every 10 milliseconds.
	sysGetTimepulse_10ms	Getting 10-millisecond pulse status	Output a square wave pulse signal with a period of 10 milliseconds. The return value status flips every 5 milliseconds.
	sysGetTimepulse_1ms	Getting 1-millisecond pulse status	Output a square wave pulse signal with a period of 1 millisecond. The return value status flips every 500 microseconds.
	sysGetTimepulse_100us	Getting 100-microsecond pulse status	Output a square wave pulse signal with a period of 100 microseconds. The return value status flips every 50

			microseconds.
Module information instruction	sysGetModel	Getting module model	Get the module model.
	sysGetCPUModuleVern	Get the firmware version information of the controller	Get the firmware version information of the controller.
	sysGetModuleSN	Getting module serial number	Get the module serial number.
	sysGetCheckTime	Getting inspection time	Get the detection time.
	sysGetEthMac	Getting Mac address	Get the Mac address.
	sysSetEthIp	Set IP Address	Set the controller IP address.
	sysGetEthIp	Get Eth IP Address	Get the controller IP address.
	sysSetCpuStatusAddress	Setting CPU diagnostic address	The input parameter DataAddr points to an address used to store diagnostic data.
	sysCleanQDataArea	Clearing Q area address data	Clear all data in Q area.
	sysGetDataAreaAddrById	Get data area header address	Use to get the start address of the data area based on the ID of the user data area (M, I, Q, R, G, S).
ETHERCAT Functions	EtherCAT_BusReset	Resetting EtherCAT bus	By resetting the comX protocol board, reconfiguring the slave station, and initializing the protocol stack, the entire EtherCAT bus is finally reset.
	EtherCAT_AxisFaultReset	Resetting slave station error	By clearing the error code displayed by the slave station through the EtherCAT protocol message, and then initializing the slave station, the slave station is finally ready to enter the enabled state.
	EtherCAT_DriveEnable	Servo enable	By turning on/off the servo enable switch, the enable status of each slave station can be controlled independently.
	EtherCAT_GetDriveEnableState	Getting servo enable status	Get the servo enable status of a slave station axis.
	EtherCAT_Read	Reading slave station parameters	Read the parameter length and parameter content of the specified parameters of the specified slave station.
	EtherCAT_Write	Writing slave station parameters	Read the content of the specified parameters of the specified slave station.
	EtherCAT_Read2	Reading slave station parameters2	It is to quickly read the parameter length and parameter content of the specified parameters of the specified slave station.
	EtherCAT_Write2	Writing slave station parameters2	It is to write the parameter content of the specified parameters of the specified slave station.
	EtherCAT_GetSlaveInOutOffset	Getting slave station input/output starting offset	Get the offset of the input/output starting address of a slave station.
	EtherCAT_SetAxisOperationMode	Set the Axis Mode of Operation	It is to set the Mode of Operation (control mode) of the specified EtherCAT slave station. The input is the specified axis number and the control mode value to be set. The axis number must be consistent with

	EtherCAT_SetAxisCtrWord	Set the Axis Control Word	It is to set the control word of the specified EtherCAT slave station.
	EtherCAT_GetSlaveODOffset	Get Slave Station OD Offset of IQ area	Get the IQ area offset of the slave station object dictionary.
	EtherCAT_GetSlaveLinkInfo	Get Slave link info	Get the slave station port link information.
EtherCat Touch Probe	EtherCAT_CaptureConfig	Touch Probe function configuration	Capture the high-precision position of the EtherCAT bus driver
	EtherCAT_CaptureGetStatus	Get Touch Probe Status	Return the status of the high-speed capture function of the specified slave station.
	EtherCAT_CaptureGetMpos	Get Touch Probe Result	Return Mpos obtained from the high-speed capture channel of the specified slave station.
EtherNetIp Functions	EIP_SCANNER_GET_IDENTITY	Get save identity information	Get the slave station identity information.
	EIP_SCANNER_GET_ASSEMBLY	Get slave assembly information	Get the slave station assembly information.
	EIP_SCANNER_SET_ASSEMBLY	Set slave assembly information	Set the slave station assembly information.
	CIPOpen	CIP Class3 Create Connection	Open a connection.
	CIPRead	CIP Class3 Read Tag	Write the read peer tag data into the source tag.
	CIPWrite	CIP Class3 Write Tag	Write the source tag value to the peer tag.
	CIPClose	CIP Class3 Close Connection	Close connection.
	CIPUCMMSend	Send CIP UCMM Explicit Message	Send connectionless explicit messages to a specified EtherNet/IP Adaptor.
CANopen Functions(LX)	LXCANopenReadDict	Read CANopen slave EDS	Used to retrieve parameter values from the object dictionary of a CANopen slave node.
	LXCANopenWriteDict	Write CANopen slave EDS	Used to write values to the object dictionary of a CANopen slave node.
	LXCANopenNodeNMT	Send NMT command specifier	Sends an NMT command to a CANopen slave node to switch its state.
DeviceNet Functions	DeviceNet_Atrribute_WriteRead	Write read DeviceNet slave atrribute	Use the DeviceNet_Attribute_WriteRead function block to set and read the attribute values of DeviceNet slave stations.
	DeviceNet_DiagInfo_Read	Read DeviceNet slave diag info	Use the DeviceNet_DiagInfo_Read function block to read the polling time, number of packets sent, number of packets received, and number of packets lost from the DeviceNet slave device.
PROFINET Functions	PN_Get_Channel_Diag	Read the PROFINET slave channel diagnostic values	Read the PROFINET slave channel diagnostic values.

	PN_Get_Warning_Diag	Read the PROFINET slave warning diagnostic values	Read the PROFINET slave warning diagnostic values.
	PN_Get_Mismatch_Diag	Read the PROFINET slave slot mismatch diagnostic values	Read the PROFINET slave slot mismatch diagnostic values.

4.2 Use of Instructions

Before using the instructions, please first understand the relevant information involved in the instructions, so that you can better understand and use the instructions.

Information Item	Description
Instruction/Functional block	Indicate the instruction name, for example: ADD
Name	Indicate the Chinese name of the instruction, for example, addition instruction
FB/FUN	Indicate whether the instruction is a functional block (FB) or function (FUN) Functional block instructions: can be called by programs or FBs. Function instructions: can be called by any one of the programs, FBs, and FUNs.
Function	Explain the instruction function
Parameter	Describe the input, output and other parameter information of the instruction.
Example	Instruction application examples are described by programming examples in LD and ST.

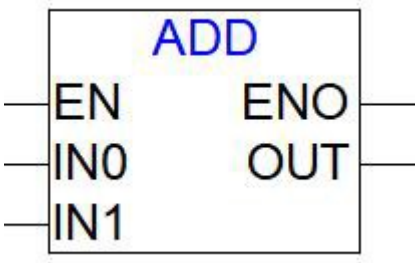
Chapter 5 Standard Library

5.1 Arithmetical Operation

5.1.1 ADD (Add)

5.1.1.1 Function

It is to add two (or multiple) variables or constants. If the calculation result exceeds the value range of the output variable, the overflow value will be displayed. For REAL type variables, if the operation overflows the range, the output will display the invalid value "1.#INF" or "-1.#INF".



ADD Functional Block

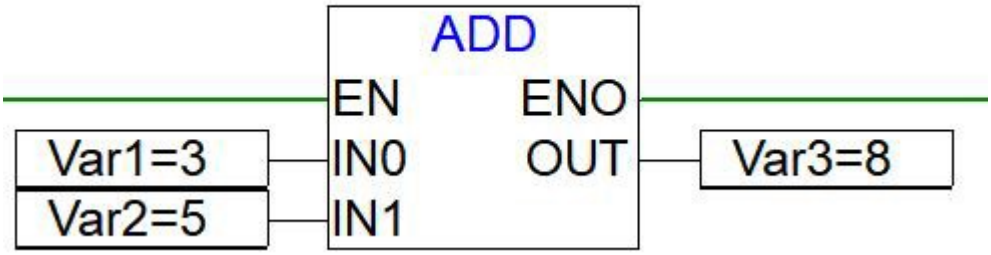
5.1.1.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
In0~InN	Input	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, TIME.	The 1st~(N+1)th input values	0	FALSE
OUT	Output	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, TIME.	1 output value	0	FALSE

5.1.1.3 Example

The following example uses the ADD instruction to implement the addition of multiple operands. When the input EN detects a rising edge, the ADD instruction starts the addition function and stores the calculation result in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			INT	3	FALSE	Not Public	
0002	Var2			INT	5	FALSE	Not Public	
0003	Var3			INT	8	FALSE	Not Public	

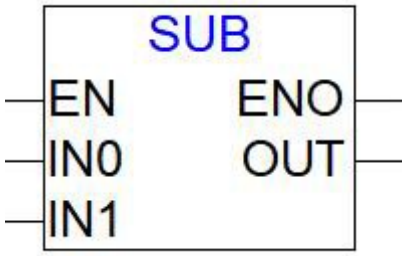



```
1   Var1 := Var2  + Var3;      |Var1 = 8                Var2 = 3                Var3 = 5
```

5.1.2 SUB (Subtract)

5.1.2.1 Function

Implement subtraction between two variables or constants.



SUB Functional Block

5.1.2.2 Parameter

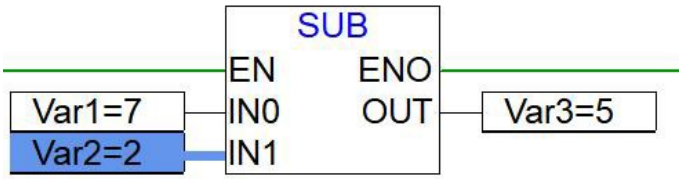
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
In0~In1	Input	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, TIME	The 1st~2nd input values (when the input data is of TIME type, please refer to the help description in the AT software for the operation rules)	0	FALSE
OUT	Output	BYTE, WORD, DWORD, SINT, USINT, INT, UINT,	1 output value	0	FALSE

		DINT, UDINT, REAL, TIME.			

5.1.2.3 Example

For example, the following example implements the subtraction operation of two operands through the SUB instruction. When the input EN detects a rising edge, the SUB instruction starts the subtraction function and stores the calculation result in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1	%IW0		INT	7	FALSE	Not Public	
0002	Var2	%MW100		INT	2	FALSE	Not Public	
0003	Var3	%QW0		INT	5	FALSE	Not Public	

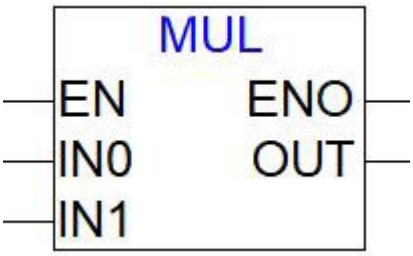



```
1   Var3 := Var1 - Var2 ;   Var3 = 5        Var1 = 7        Var2 = 2
```

5.1.3 MUL (Multiply)

5.1.3.1 Function

It is to implement multiplication of two (or multiple) variables or constants.



MUL Functional Block

5.1.3.2 Parameter

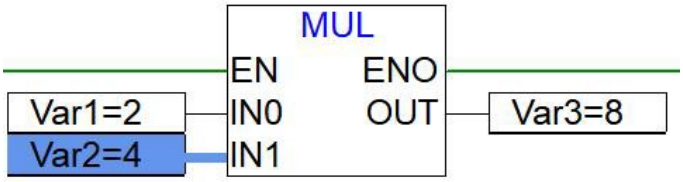
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
In1~InN	Input	BYTE, WORD, DWORD,	The 1st~Nth input values (when the input	0	FALSE

		SINT, USINT, INT, UINT, DINT, UDINT, REAL, TIME.	data is of TIME type, please refer to the help description in the AT software for the operation rules)		
OUT	Output	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, TIME.	1 output value	0	FALSE

5.1.3.3 Example

For example, the following example uses the MUL instruction to implement the multiplication operation of multiple operands. When the input EN detects a rising edge, the MUL instruction starts the multiplication function and stores the calculation result in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1	%MW0		INT	2	FALSE	Not Public	
0002	Var2	%MW100		INT	4	FALSE	Not Public	
0003	Var3	%QW0		INT	5	FALSE	Not Public	



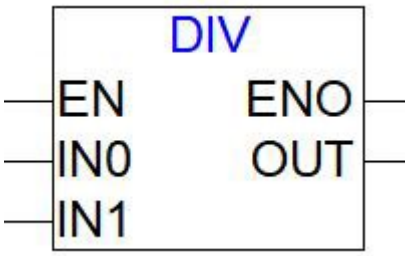

```
1  Var3 := Var1 * Var2 ;
```

Var3 = 8 Var1 = 2 Var2 = 4

5.1.4 DIV (Divide)

5.1.4.1 Function

Divide variables or constants. When the divisor is 0, the result is equal to the dividend.



DIV Functional Block

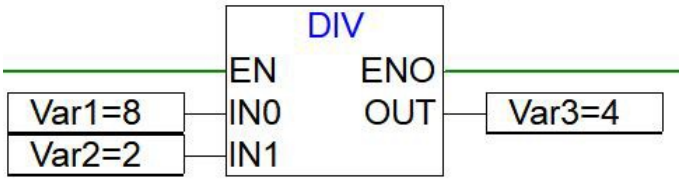
5.1.4.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
In1~In2	Input	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL.	The 1st~2nd input values	0	FALSE
OUT	Output	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL.	1 output value	0	FALSE

5.1.4.3 Example

For example, the following example implements the division operation of two operands through the DIV instruction. When the input EN detects a rising edge, the DIV instruction starts the division function and stores the calculation result in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1	%IW0		INT	8	FALSE	Not Public	
0002	Var2	%MW100		INT	2	FALSE	Not Public	
0003	Var3	%QW0		INT	4	FALSE	Not Public	



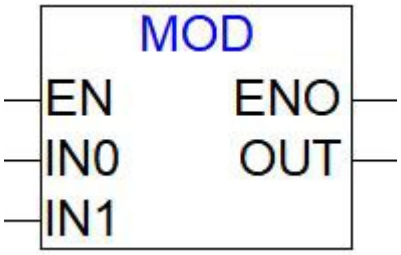

```
1  Var3 := Var1 / Var2 ;
```

Var3 = 4 Var1 = 8 Var2 = 2

5.1.5 MOD (Modulus)

5.1.5.1 Function

The remainder after the division of a variable or constant is an integer. When the divisor is 0, the result is equal to the dividend, and the sign of the result is the same as the dividend.



MOD Functional Block

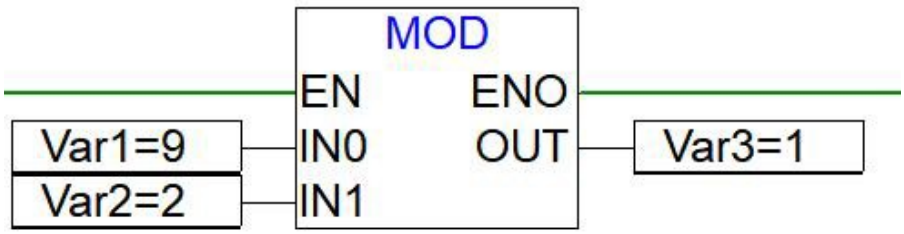
5.1.5.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
In1~In2	Input	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL.	The 1st~2nd input values	0	FALSE
OUT	Output	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL.	1 output value	0	FALSE

5.1.5.3 Example

For example, the following example implements the remainder operation of two operands through the MOD instruction. When the input EN detects a rising edge, the MOD instruction starts the remainder operation function and stores the calculation result in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1	%IW0		INT	9	FALSE	Not Public	
0002	Var2	%MW100		INT	2	FALSE	Not Public	
0003	Var3	%QW0		INT	1	FALSE	Not Public	

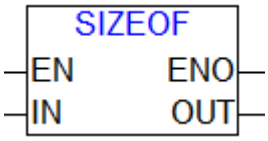



```
1  Var3 := Var1 MOD Var2 ;
```
Var3 = 1 Var1 = 9 Var2 = 2

5.1.6 SIZEOF (Variable Size)

5.1.6.1 Function

Number of bytes required for getting the data type.



SIZEOF Functional Block

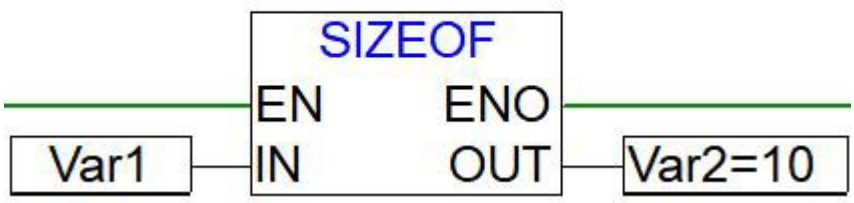
5.1.6.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
In1	Input	ARRAY	1 input value		FALSE
OUT	Output	DINT, DWORD, INT, REAL, UDINT, UINT, WORD	1 output value	0	FALSE

5.1.6.3 Example

For example, the following example uses the SIZEOF instruction to obtain the number of bytes required for the data type. When the input EN detects a rising edge, the SIZEOF instruction starts the byte length function and stores the calculation result in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			ARRAY[0..4] OF INT		FALSE	Not Public	
0002	Var2			INT	10	FALSE	Not Public	




```
1  Var2 := SIZEOF(Var1);
```

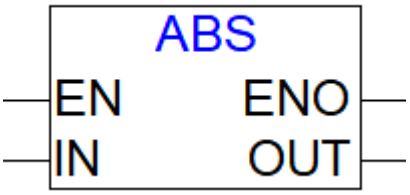
Var2 = 10

Var1

5.1.7 ABS (Absolute Value)

5.1.7.1 Function

Assign the absolute value of the input data to the output variable.



ABS Functional Block

5.1.7.2 Parameter

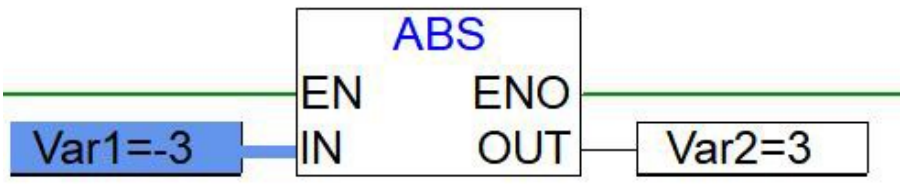
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
In1	Input	INT, REAL, LREAL, BYTE, WORD, DWORD, SINT, USINT, UINT, DINT, UDINT.	1 input value	0	FALSE
OUT	Output	①INT, WORD, DWORD, DINT, UINT, REAL, LREAL ②REAL ③LREAL ④INT, BYTE, WORD, DWORD, DINT, UINT, REAL, LREAL ⑤WORD, DWORD, DINT, REAL, LREAL	1 output value	0	FALSE

		⑥DWORD, DINT, REAL, LREAL ⑦INT, BYTE, WORD, DWORD, DINT, UINT, REAL, LREAL ⑧INT, BYTE, WORD, DWORD, DINT, UINT, REAL, LREAL ⑨WORD, DWORD, DINT, UINT, REAL, LREAL ⑩DWORD, DINT, REAL, LREAL ⑪DWORD, DINT, UDINT, REAL, LREAL ⑫LREAL ⑬LREAL ⑭LREAL			

5.1.7.3 Example

For example, the following example implements an absolute value assignment operation for an operand through the ABS instruction. When the input EN detects a rising edge, the ABS instruction assigns the absolute value of the input data to the output variable and stores the calculation result in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1	%IW10		INT	0	FALSE	Not Public	
0002	Var2	%MW100		REAL	0	FALSE	Not Public	




```
1  Var2 := ABS(Var1);
```

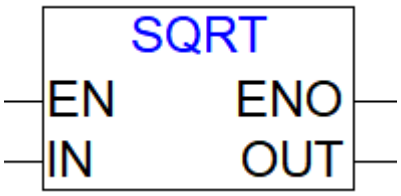
Var2 = 3

Var1 = -3

5.1.8 SQRT (Square Root)

5.1.8.1 Function

It is to find the square root of the input data, where the input data must be non-negative.



SQRT Functional Block

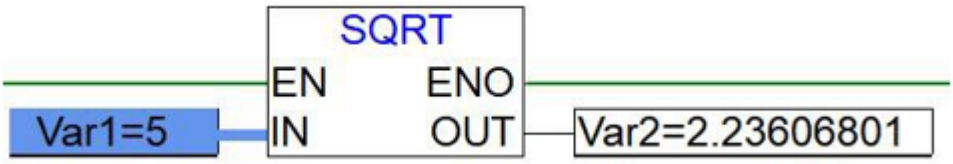
5.1.8.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
In1	Input	BYTE, WORD, DWORD, INT, DINT, REAL, LREAL, SINT, USINT, UINT, UDINT	1 input value	0	FALSE
OUT	Output	REAL, LREAL	1 output value	0	FALSE

5.1.8.3 Example

For example, the following example implements the square root operation of an operand through the SQRT instruction. When the input EN detects a rising edge, the SQRT instruction takes the square root of the input data and stores the calculation result in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1	%IW10		INT	5	FALSE	Not Public	
0002	Var2	%MW100		REAL	2.236068	FALSE	Not Public	

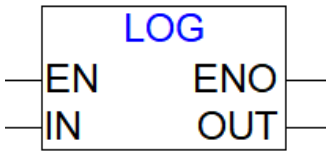



```
1   Var2 := SQRT(Var1);        Var2 = 2.23606801        Var1 = 5
```

5.1.9 LOG (Common Logarithm)

5.1.9.1 Function

Find the base 10 logarithm of the input data, where the input data must be a positive number.



LOG Functional Block

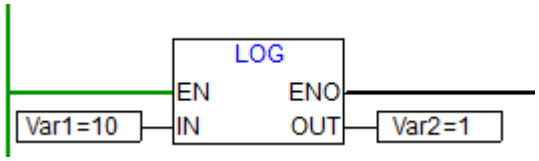
5.1.9.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
In1	Input	BYTE, WORD, DWORD, INT, DINT, REAL, LREAL, SINT, USINT, UINT, UDINT,	1 input value	0	FALSE
OUT	Output	REAL, LREAL	1 output value	0	FALSE

5.1.9.3 Example

For example, the following example implements the logarithm operation of an operand through the LOG instruction. When the input EN detects a rising edge, the LOG instruction starts to calculate the base-10 logarithm of the input data (the input data must be a positive number), and stores the calculation result in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1	%IW10		INT	10	FALSE	Not Public	
0002	Var2	%MW100		REAL	1	FALSE	Not Public	




```
1  Var2 := LOG(Var1);
```

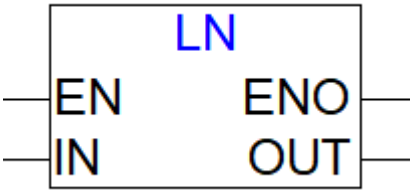
Var2 = 1

Var1 = 10

5.1.10 LN (Natural Logarithm)

5.1.10.1.1 Function

It is to find the natural logarithm of the input data, where the input data must be a positive number.



LN Functional Block

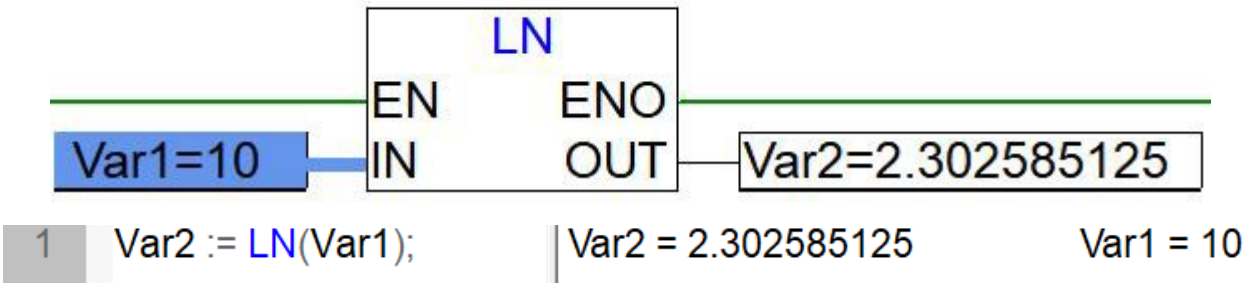
5.1.10.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
In1	Input	BYTE, WORD, DWORD, INT, DINT, REAL, LREAL, SINT, USINT, UINT, UDINT	1 input value	0	FALSE
OUT	Output	REAL, LREAL	1 output value	0	FALSE

5.1.10.3 Example

For example, the following example implements the natural logarithm operation of an operand through the LN instruction. When the input EN detects a rising edge, the LN instruction calculates the natural logarithm of the input data (the input data must be a positive number), and stores the calculation result in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1	%IW10		INT	10	FALSE	Not Public	
0002	Var2	%MW100		REAL	2.302585	FALSE	Not Public	



5.1.11 EXP (Exponent)

5.1.11.1 Function

It is calculated using the input data as the power of the exponent, that is, y=eX, where x is the input and y is the output.

EXP Functional Block

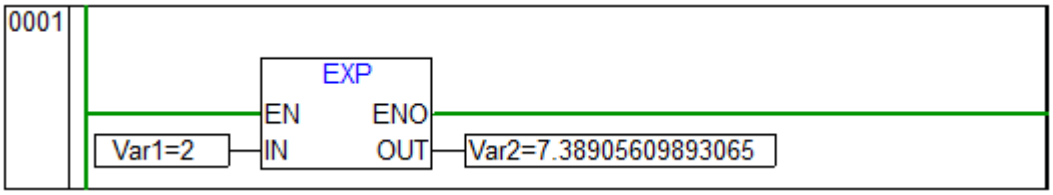
5.1.11.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
In1	Input	BYTE, WORD, DWORD, INT, DINT, REAL, LREAL, SINT, USINT, UINT, UDINT	1 input value	0	FALSE
OUT	Output	REAL, LREAL	1 output value	0	FALSE

5.1.11.3 Example

For example, the following example implements the power operation of an operand through the EXP instruction. When the input EN detects a rising edge, the EXP instruction calculates by raising the input data to the power of the exponent and stores the calculation result in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			INT	2	FALSE	Not Public	
0002	Var2			LREAL	7.38905609893...	FALSE	Not Public	



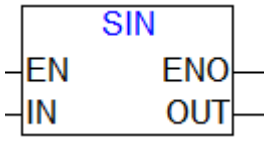

```
1    Var2 := EXP(Var1);    | Var2 = 7.38905609893065    Var1 = 2
```

5.1.12 SIN (Sine)

5.1.12.1 Function

Find the sine of the input data, where the input data is expressed in radians.

$$\text{Radian (rad)} = \text{angle}^* \frac{\pi}{180^\circ}$$



SIN Functional Block

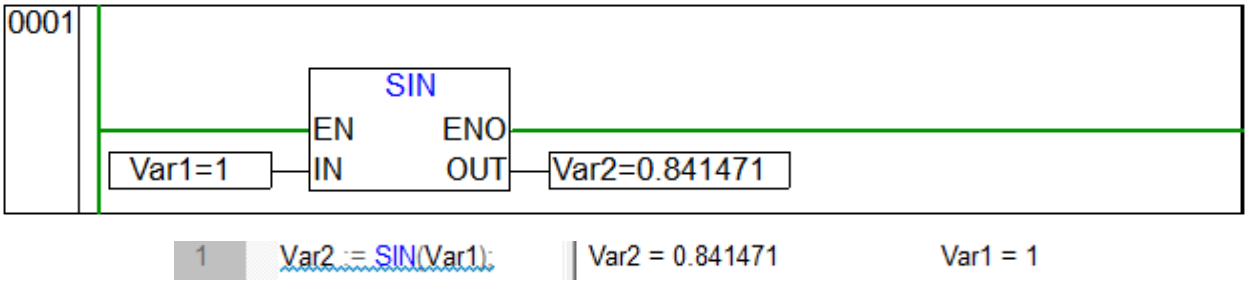
5.1.12.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
In1	Input	BYTE, WORD, DWORD, INT, DINT, REAL, LREAL, SINT, USINT, UINT, UDINT	1 input value	0	FALSE
OUT	Output	REAL, LREAL	1 output value	0	FALSE

5.1.12.3 Example

For example, the following example implements the sine operation of an operand through the SIN instruction. When the input EN detects a rising edge, the SIN instruction finds the sine value of the input data, which is expressed in radians, and stores the calculation result in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1	%IW10		INT	1	FALSE	Not Public	
0002	Var2	%MW100		REAL	0.841471	FALSE	Not Public	



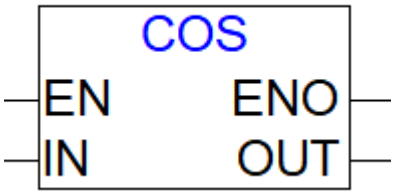

```
1   Var2 := SIN(Var1);   | Var2 = 0.841471   Var1 = 1
```

5.1.13 COS (Cosine)

5.1.13.1 Function

It is to find the cosine of the input data, where the input data is expressed in radians.

$$\text{Radian (rad)} = \text{angle}* \frac{\pi}{180^\circ}$$



COS Functional Block

5.1.13.2 Parameter

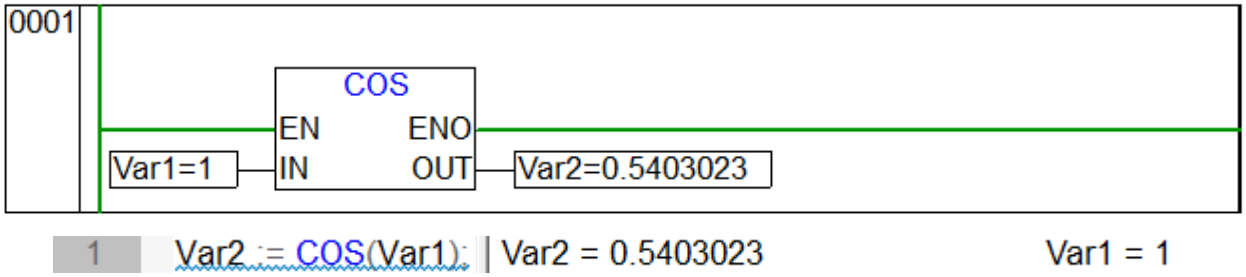
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
In1	Input	BYTE, WORD, DWORD, INT, DINT, REAL, LREAL, SINT, USINT, UINT, UDINT	1 input value	0	FALSE
OUT	Output	REAL, LREAL	1 output value	0	FALSE

5.1.13.3 Example

For example, the following example implements the cosine operation of an operand through the COS

instruction. When the input EN detects a rising edge, the COS instruction finds the cosine value of the input data, which is expressed in radians, and stores the calculation result in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1	%IW10		INT	1	FALSE	Not Public	
0002	Var2	%MW100		REAL	0.5403023	FALSE	Not Public	

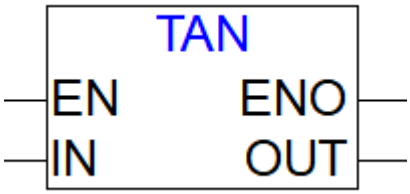


5.1.14 TAN (Tangent)

5.1.14.1 Function

It is to find the tangent value of the input data, where the input data is expressed in radians.

$$\text{Radian (rad)} = \text{angle}^* \frac{\pi}{180^\circ}$$



TAN Functional Block

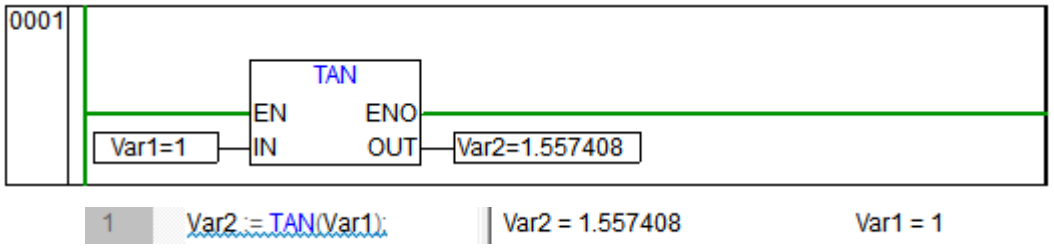
5.1.14.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
In1	Input	BYTE, WORD, DWORD, INT, DINT, REAL, LREAL, SINT, USINT, UINT, UDINT	1 input value	0	FALSE
OUT	Output	REAL, LREAL	1 output value	0	FALSE

5.1.14.3 Example

For example, the following example implements the tangent operation of an operand through the TAN instruction. When the input EN detects a rising edge, the TAN instruction finds the tangent value of the input data, which is expressed in radians, and stores the calculation result in the output parameter OUT.

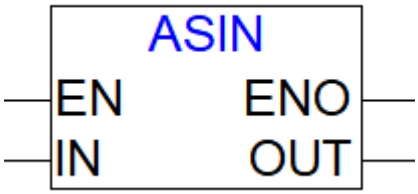
No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1	%IW10		INT	1	FALSE	Not Public	
0002	Var2	%MW100		REAL	1.557408	FALSE	Not Public	



5.1.15 ASIN (Arc Sine)

5.1.15.1 Function

Find the arcsine of the input data. Input value: [-1,+1].



ASIN Functional Block

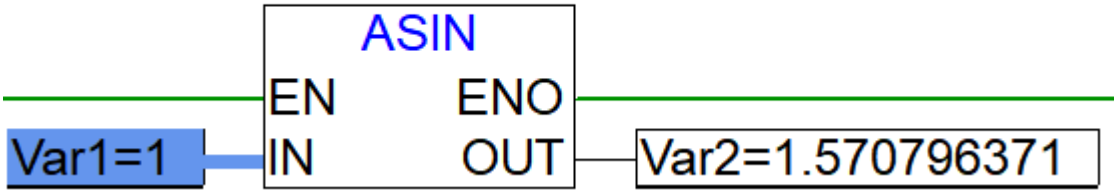
5.1.15.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
In1	Input	BYTE, WORD, DWORD, INT, DINT, REAL, LREAL, SINT, USINT, UINT, UDINT	1 input value	0	FALSE
OUT	Output	REAL, LREAL	1 output value	0	FALSE

5.1.15.3 Example

For example, the following example implements the arcsine operation of an operand through the ASIN instruction. When the input EN detects a rising edge, the ASIN instruction finds the arcsine value of the input data. The input value is [-1,+1], and the calculation result is stored in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			INT	1	FALSE	Not Public	
0002	Var2			LREAL	1.570796371	FALSE	Not Public	



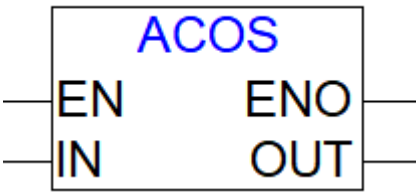

```
0001 Var2 := ASIN (Var1 );
```

Var2 = 1.570796371 Var1 = 1

5.1.16 ACOS (Arc Cosine)

5.1.16.1 Function

Find the arc cosine of the input data. Input value: [-1,+1].



ACOS Functional Block

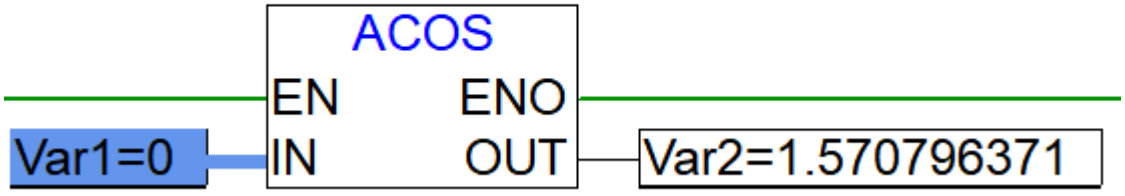
5.1.16.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
In1	Input	BYTE, WORD, DWORD, INT, DINT, REAL, LREAL, SINT, USINT, UINT, UDINT	1 input value	0	FALSE
OUT	Output	REAL, LREAL	1 output value	0	FALSE

5.1.16.3 Example

For example, the following example implements the arc cosine operation of an operand through the ACOS instruction. When the input EN detects a rising edge, the ACOS instruction finds the arc cosine value of the input data. The input value is [-1,+1], and the calculation result is stored in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			INT	0	FALSE	Not Public	
0002	Var2			LREAL	1.570796371	FALSE	Not Public	

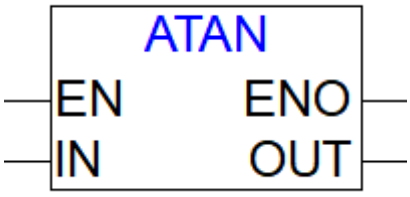



```
0001 Var2 := ACOS (Var1);
```
Var2 = 1.570796371 Var1 = 0

5.1.17 ATAN (Arc Tangent)

5.1.17.1 Function

Find the arc tangent of the input data.



ATAN Functional Block

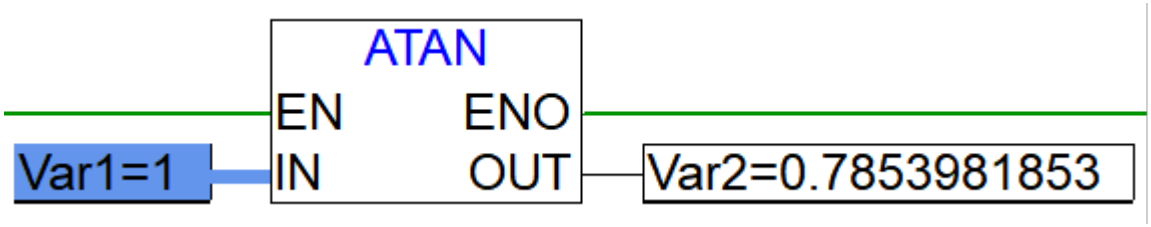
5.1.17.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
In1	Input	BYTE, WORD, DWORD, INT, DINT, REAL, LREAL, SINT, USINT, UINT, UDINT	1 input value	0	FALSE
OUT	Output	REAL, LREAL	1 output value	0	FALSE

5.1.17.3 Example

For example, the following example implements the arc tangent operation of an operand through the ATAN instruction. When the input EN detects a rising edge, the ATAN instruction takes the arc tangent value of the input data and stores the calculation result in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			INT	1	FALSE	Not Public	
0002	Var2			LREAL	0.7853981853	FALSE	Not Public	



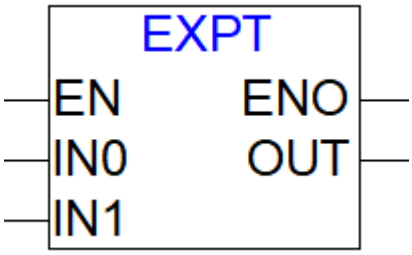

```
0001 Var2:= ATAN (Var1 );
```

Var2 = 0.7853981853 Var1 = 1

5.1.18 EXPT (Power)

5.1.18.1 Function

Calculate the power of x as the base and y as the exponent. x is IN1 and y is IN2.



EXPT Functional Block

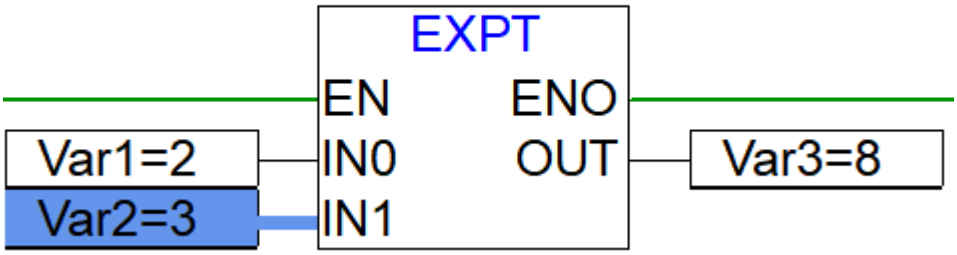
5.1.18.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
In1~In2	Input	BYTE, WORD, DWORD, INT, DINT, REAL, LREAL, SINT, USINT, UINT, UDINT	The 1st~2nd input values	0	FALSE
OUT	Output	REAL, LREAL	1 output value	0	FALSE

5.1.18.3 Example

For example, the following example implements the power operation of 2 operands through the EXPT instruction. When the input EN detects a rising edge, the EXPT instruction calculates the power with In1 as the base and In2 as the exponent and stores the calculation result in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1	%IW10		INT	2	FALSE	Not Public	
0002	Var2	%MW100		REAL	3	FALSE	Not Public	
0003	Var3	%QW0		LREAL	8	FALSE	Not Public	

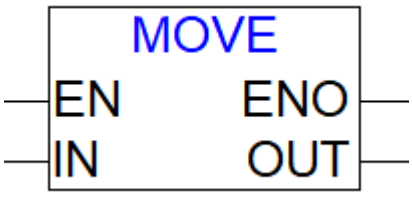



```
1  Var3 := EXPT(Var1, Var2);   Var3 = 8        Var1 = 2        Var2 = 3
```

5.1.19 MOVE (Assignment)

5.1.19.1 Function

Assign the value of a constant or variable to another variable.



MOVE Functional Block

5.1.19.2 Parameter

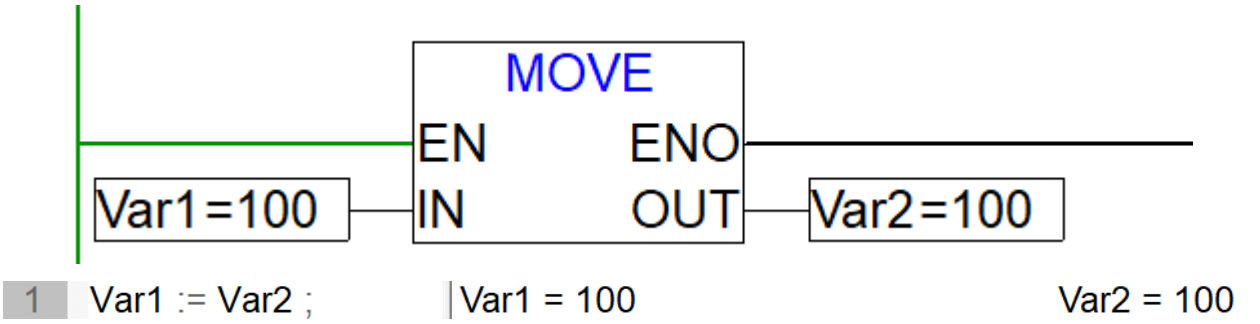
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
In1	Input	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT,	1 input value	0	FALSE

		UDINT, REAL, TIME, DT, BOOL, ARRAY, LREAL, STRING, TOD, DATE			
OUT	Output	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, TIME, DT, BOOL, ARRAY, LREAL, STRING, TOD, DATE	1 output value	0	FALSE

5.1.19.3 Example

For example, the following example implements the assignment of an operand through the MOVE instruction. When the input EN detects a rising edge, the MOVE instruction assigns the value of a constant or variable to another variable, and stores the calculation result in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1	%IW10		INT	100	FALSE	Not Public	
0002	Var2	%MW100		REAL	100	FALSE	Not Public	

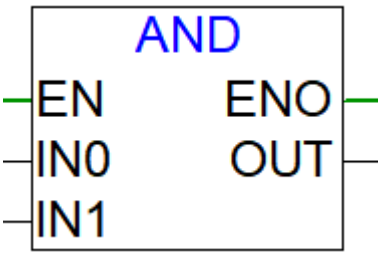


5.2 Logic Operation

5.2.1 AND (And)

5.2.1.1 Function

AND operation of variables or constants.



AND Functional Block

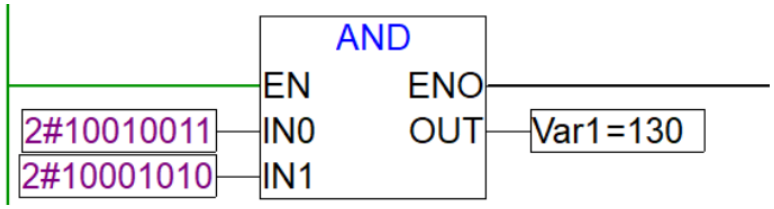
5.2.1.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
In1~InN	Input	BOOL, BYTE, WORD, DWORD	It is to perform logic operation of the 1st~Nth input values		
OUT	Output	BOOL, BYTE, WORD, DWORD	Instruction result	0	FALSE

5.2.1.3 Example

The example uses the "AND" operation instruction to perform an "AND" operation on the value of input IN0 and the value of input IN1 and queries the result in the output OUT. When this instruction is executed, bit 0 of the input IN0 value and bit 0 of the input IN1 value perform an "AND" operation. The result is stored in bit 0 of the output OUT. The same logic operation is performed on all other bits of the specified value, and the signal status of the result bit is "1" only when both bits in the logic operation have the signal status "1". If one of the two bits of the logic operation has the signal status "0", the corresponding result bit is reset.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			INT	130	FALSE	Not Public	

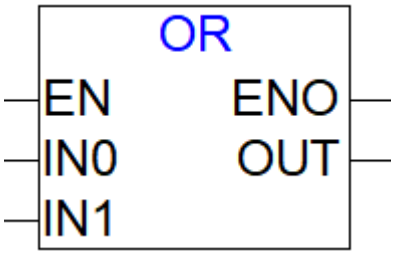



```
1   Var1 := 2#10010011 & 2#10001010 ;        Var1 = 130
```

5.2.2 OR (Or)

5.2.2.1 Function

OR operation of variables or constants.



OR Functional Block

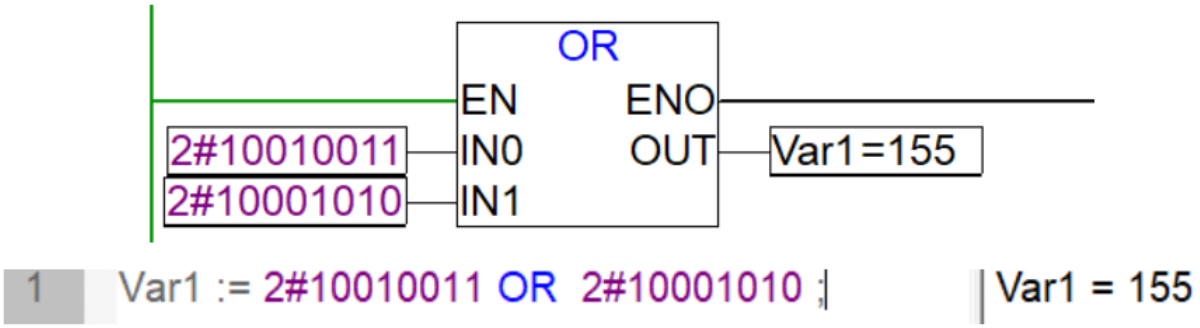
5.2.2.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
In1~InN	Input	BOOL, WORD, DWORD, BYTE	It is to perform logic operation of the 1st~Nth input values		
OUT	Output	BOOL, WORD, DWORD, BYTE	Instruction result	0	FALSE

5.2.2.3 Example

The example uses the "OR" operation instruction to perform an "OR" operation on the value of input IN0 and the value of input IN1, and queries the result in the output OUT. After this instruction is executed, it is to perform an "OR" operation on bit 0 of the value input by IN0 and bit 0 of the value input by IN1. The result is stored in bit 0 of the output OUT. The same logic operation is performed on all bits of the specified variable. As long as the signal status of one of the two bits in this logic operation is "1", the signal status of the result bit is "1". If the signal status of both bits of the logic operation is "0", the corresponding result bit is reset.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			BYTE	155	FALSE	Not Public	



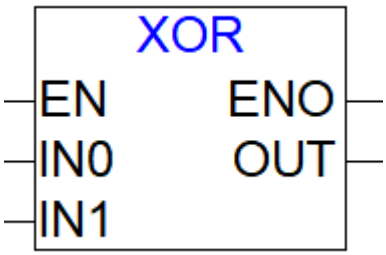

```
1   Var1 := 2#10010011 OR  2#10001010 ;
```

Var1 = 155

5.2.3 XOR (Exclusive Or)

5.2.3.1 Function

XOR operation of variables or constants.



XOR Functional Block

5.2.3.2 Parameter

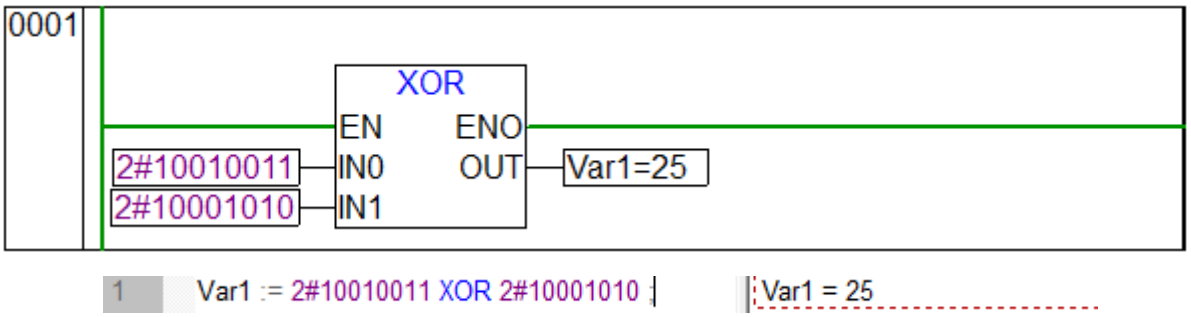
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
In1~InN	Input	BOOL, BYTE, WORD, DWORD	It is to perform logic operation of the 1st~Nth input values		
OUT	Output	BOOL, BYTE, WORD, DWORD	Instruction result	0	FALSE

5.2.3.3 Example

The example uses the "XOR" operation instruction to perform an "XOR" operation on the value of input IN0 and the value of input IN1, and queries the result in the output OUT. After this instruction is executed, it is to perform an "XOR" operation on bit 0 of the value input by IN1 and bit 0 of the value input by IN2.

The result is stored in bit 0 of the output OUT. The same logic operation is performed on all other bits of the specified value. When the signal status of one of the two bits in this logic operation is "1", the signal status of the result bit is "1". If the signal status of both bits of the logic operation is "1" or "0", the corresponding result bit is reset.

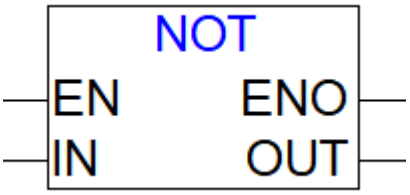
No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			BYTE	25	FALSE	Not Public	



5.2.4 NOT (Not)

5.2.4.1 Function

NOT operation of variables or constants bit by bit.



NOT Functional Block

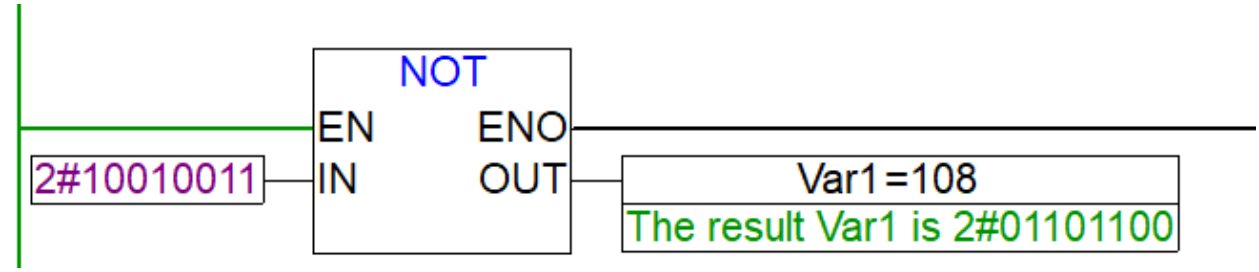
5.2.4.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
IN	Input	BOOL, WORD, DWORD, BYTE	Logic operation of input values		
OUT	Output	BOOL, WORD, DWORD, BYTE	Instruction result	0	FALSE

5.2.4.3 Example

The example uses the "NOT" instruction to invert the signal status of each bit of the input IN. When processing the instruction, it inverts the signal status of the individual bits and stores the result in the output OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			BYTE	108	FALSE	Not Public	




```
1  Var1:=2#10010011 XOR 2#10001010;
2  (*The result Var1 is 2#01101100*)
```

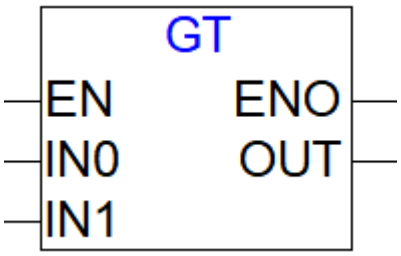
Var1 = 108

5.3 Comparison Operation

5.3.1 GT (Greater Than)

5.3.1.1 Function

Determine the size of the two operands. When the first number is greater than the second number, TRUE is output, otherwise FALSE is output.



GT Functional Block

5.3.1.2 Parameter

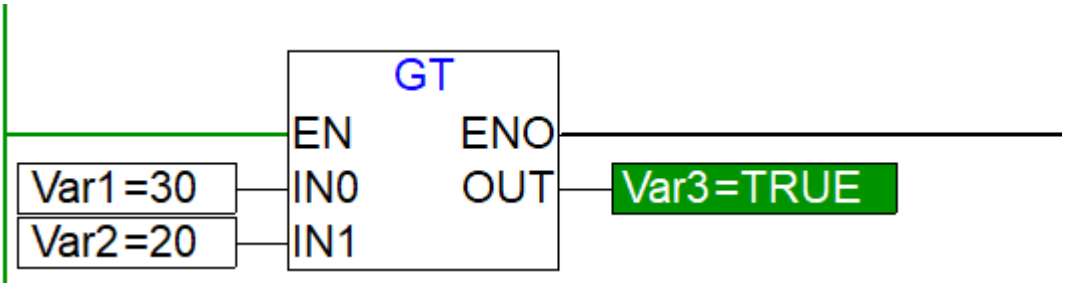
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard

EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
IN0, IN1	Input	BOOL, BYTE, WORD, DT, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, TIME, DATE, TOD, DT, LREAL, STRING	Enter two operands	0	FALSE
OUT	Output	BOOL	Instruction result	FALSE	FALSE

5.3.1.3 Example

The example uses the GT instruction to compare two operands. When the operand IN0 is greater than the operand IN1, the OUT output is TRUE, otherwise the OUT output is FALSE.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			INT	20	FALSE	Not Public	
0002	Var2			INT	30	FALSE	Not Public	
0003	Var3			BOOL	TRUE	FALSE	Not Public	

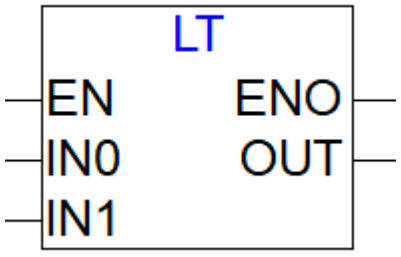


1 Var3:=Var1>Var2; | Var3 = TRUE Var1 = 30 Var2 = 20

5.3.2 LT (Less Than)

5.3.2.1 Function

It is to determine the size of the two operands. When the first number is lesser than the second number, TRUE is returned, otherwise, the result is FALSE.



LT Functional Block

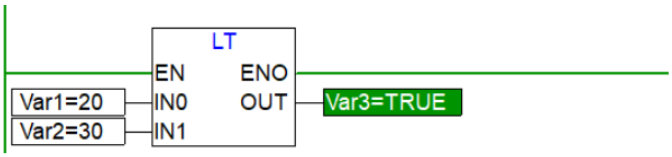
5.3.2.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
IN0, IN1	Input	BOOL, BYTE, WORD, DT, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, TIME, DATE, TOD, DT, LREAL, STRING	Enter two operands	0	FALSE
OUT	Output	BOOL	Instruction result	FALSE	FALSE

5.3.2.3 Example

The example uses the LT instruction to compare two operands. When the operand IN0 is smaller than the operand IN1, the OUT output is TRUE, otherwise, the OUT output is FALSE.

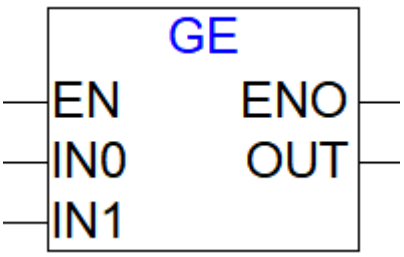
No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			INT	20	FALSE	Not Public	
0002	Var2			INT	30	FALSE	Not Public	
0003	Var3			BOOL	TRUE	FALSE	Not Public	



5.3.3 GE (Greater or Equal)

5.3.3.1 Function

It is to determine the size of the two operands. When the first number is greater than or equal to the second number, TRUE is returned, otherwise, the result is FALSE.



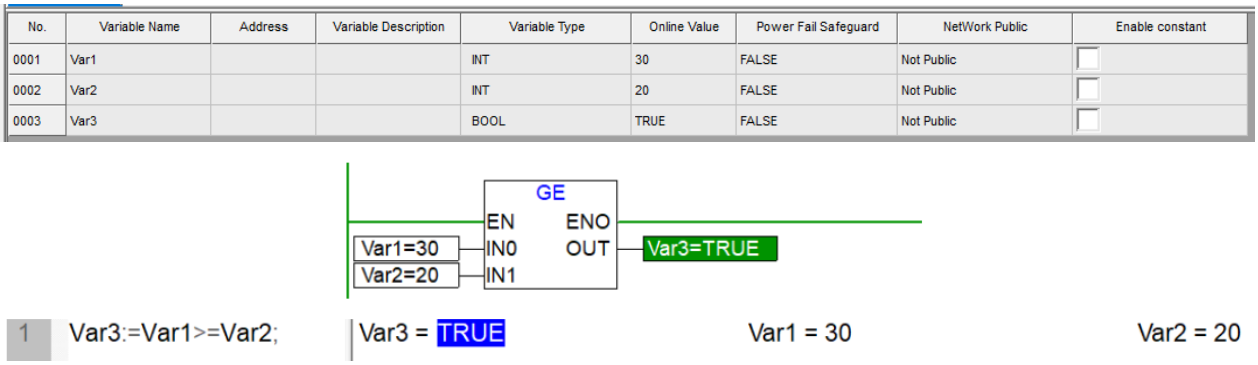
GE Functional Block

5.3.3.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
IN0, IN1	Input	BOOL, BYTE, WORD, DT, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, TIME, DATE, TOD, DT, LREAL, STRING	Enter two operands	0	FALSE
OUT	Output	BOOL	Instruction result	FALSE	FALSE

5.3.3.3 Example

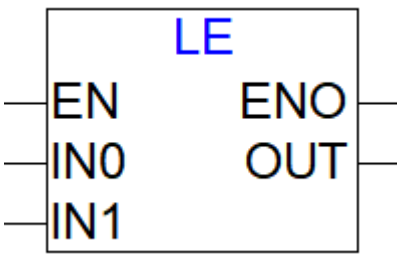
The example uses the GE instruction to compare two operands. When the operand IN0 is greater than or equal to the operand IN1, the OUT output is TRUE, otherwise the OUT output is FALSE.



5.3.4 LE (Less or Equal)

5.3.4.1 Function

It is to determine the size of the two operands. When the first number is lesser than or equal to the second number, TRUE is returned, otherwise, the result is FALSE.



LE Functional Block

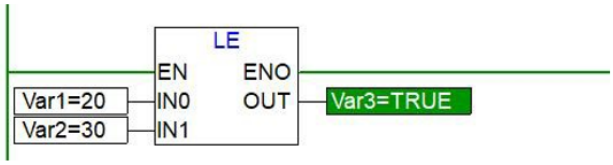
5.3.4.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
IN0, IN1	Input	BYTE, WORD, DT, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, TIME, DATE, TOD, DT, LREAL, STRING	Enter two operands	0	FALSE
OUT	Output	BOOL	Instruction result	FALSE	FALSE

5.3.4.3 Example

The example uses the LE instruction to compare two operands. When the operand IN0 is smaller than or equal to the operand IN1, the OUT output is TRUE, otherwise, the OUT output is FALSE.

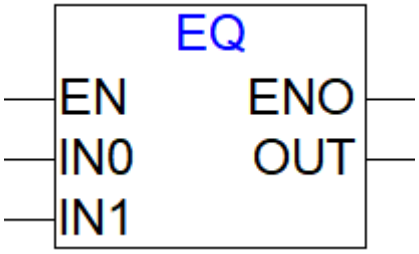
No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			INT	20	FALSE	Not Public	
0002	Var2			INT	30	FALSE	Not Public	
0003	Var3			BOOL	TRUE	FALSE	Not Public	



5.3.5 EQ (Equal)

5.3.5.1 Function

It is to determine whether the two operands are equal to each other. When the first number is equal to the second number, TRUE is returned, otherwise the result is FALSE.



EQ Functional Block

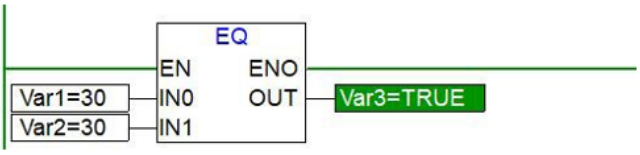
5.3.5.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
IN0, IN1	Input	BOOL, BYTE, WORD, DT, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, TIME, DATE, TOD, LREAL, STRING	Enter two operands	0	FALSE
OUT	Output	BOOL	Instruction result	FALSE	FALSE

5.3.5.3 Example

The example uses the EQ instruction to compare two operands. When the operand IN0 is equal to the operand IN1, the OUT output is TRUE, otherwise the OUT output is FALSE.

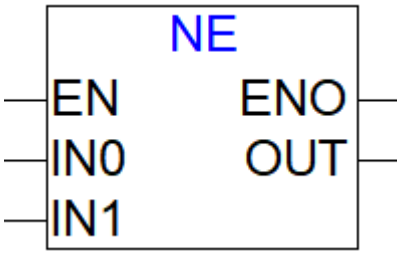
No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			INT	30	FALSE	Not Public	
0002	Var2			INT	30	FALSE	Not Public	
0003	Var3			BOOL	TRUE	FALSE	Not Public	



5.3.6 NE (Not Equal)

5.3.6.1 Function

It is to determine whether the two operands are not equal to each other. When the first number is not equal to the second number, TRUE is returned, otherwise the result is FALSE.



NE Functional Block

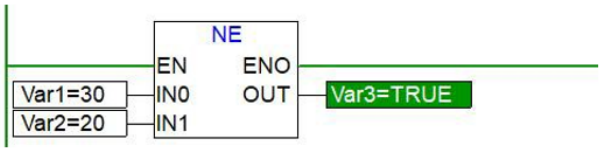
5.3.6.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
IN0, IN1	Input	BOOL, BYTE, WORD, DT, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, TIME, DATE, TOD, LREAL, STRING	Enter two operands	0	FALSE
OUT	Output	BOOL	Instruction result	FALSE	FALSE

5.3.6.3 Example

The example uses the NE instruction to compare two operands. When the operand IN0 is not equal to the operand IN1, the OUT output is TRUE, otherwise the OUT output is FALSE.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			INT	30	FALSE	Not Public	
0002	Var2			INT	20	FALSE	Not Public	
0003	Var3			BOOL	TRUE	FALSE	Not Public	

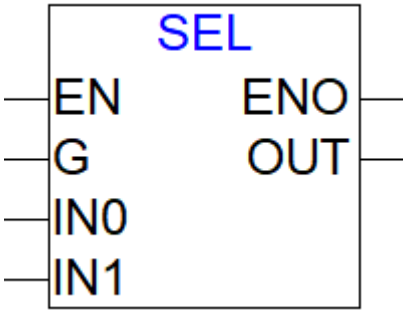


5.4 Selection Operation

5.4.1 SEL (One Out of Two)

5.4.1.1 Function

Use the selector switch to select one of the two input data as the output. When the selector switch is FALSE, the output is the first input data. When the selector switch is TRUE, the output is the second input data.



SEL Functional Block

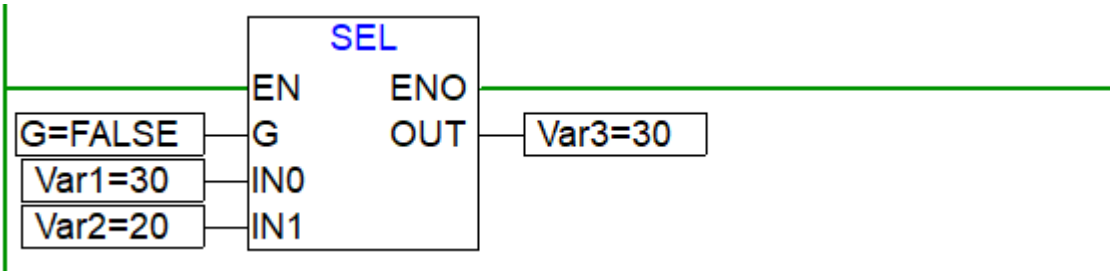
5.4.1.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
G	Input	BOOL	Selector switch	FALSE	FALSE
IN0, IN1	Input	BOOL, BYTE, WORD, INT, DWORD, SINT, USINT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TOD, DT	Enter two operands	0	FALSE
OUT	Output	BOOL, BYTE, WORD, INT, DWORD, SINT, USINT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TOD, DT	Instruction result	0	FALSE

5.4.1.3 Example

The example is a SEL instruction OUT := SEL(G, IN0, IN1), where G is the selector switch, and IN0 and IN1 are the first and the second input data respectively. When the selector switch G is FALSE, the OUT output is IN0, and when the selector switch G is TRUE, the OUT output is IN1.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	G			BOOL	FALSE	FALSE	Not Public	
0002	Var1			INT	30	FALSE	Not Public	
0003	Var2			INT	20	FALSE	Not Public	
0004	Var3			INT	30	FALSE	Not Public	



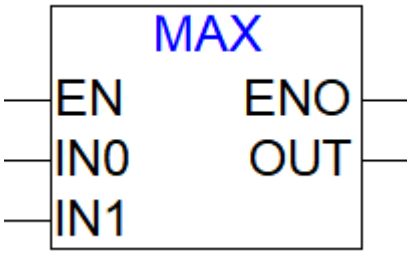

```
1  Var3 := SEL  (G, Var1, Var2);
```

Var3 = 30 G = FALSE Var1 = 30 Var2 = 20

5.4.2 MAX (Maximum)

5.4.2.1 Function

Select the maximum value among the two input data as the output.



MAX Functional Block

5.4.2.2 Parameter

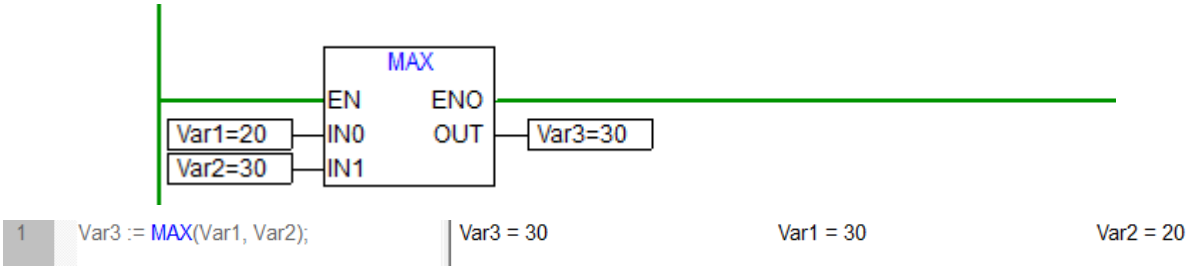
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
IN0, IN1	Input	BYTE, WORD, INT, DWORD, SINT, USINT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TOD, DT	Enter two operands	0	FALSE
OUT	Output	BYTE, WORD, INT, DWORD, SINT, USINT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TOD, DT	Instruction result	0	FALSE

5.4.2.3 Example

The example is a MAX instruction OUT:=MAX(IN0, IN1), where IN0 and IN1 are the first input data and the

second input data respectively, and OUT is the output data. When the MAX instruction is executed, the maximum value out of the two input data IN0 and IN1 is selected to output to OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			INT	20	FALSE	Not Public	
0002	Var2			INT	30	FALSE	Not Public	
0003	Var3			INT	30	FALSE	Not Public	



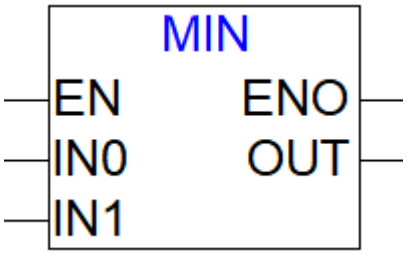

```
1  Var3 := MAX(Var1, Var2);
```

Var3 = 30 Var1 = 30 Var2 = 20

5.4.3 MIN (Minimum)

5.4.3.1 Function

Select the minimum value among the two input data as the output.



MIN Functional Block

5.4.3.2 Parameter

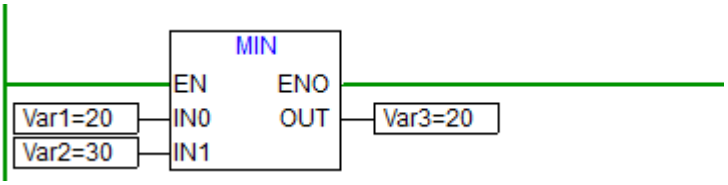
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
IN0, IN1	Input	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TOD, DT	Enter two operands	0	FALSE
OUT	Output	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TOD, DT	Instruction result	0	FALSE

5.4.3.3 Example

The example is a MIN instruction OUT:=MIN(IN0, IN1), where IN0 and IN1 are the first input data and the

second input data respectively, and OUT is the output data. When the MIN instruction is executed, the minimum value out of the two input data IN0 and IN1 is selected to output to OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			INT	20	FALSE	Not Public	
0002	Var2			INT	30	FALSE	Not Public	
0003	Var3			INT	20	FALSE	Not Public	



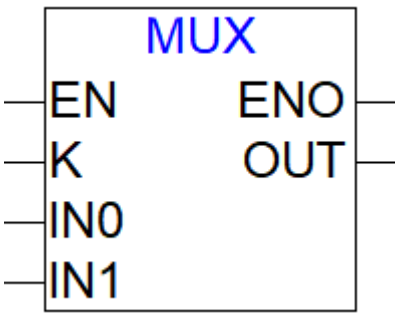

```
1  Var3 := MIN (Var1, Var2);
```

Var3 = 20 Var1 = 30 Var2 = 20

5.4.4 MUX (One Out of Multiple)

5.4.4.1 Function

Use the control number to select one of multiple input data as the output.



MUX Functional Block

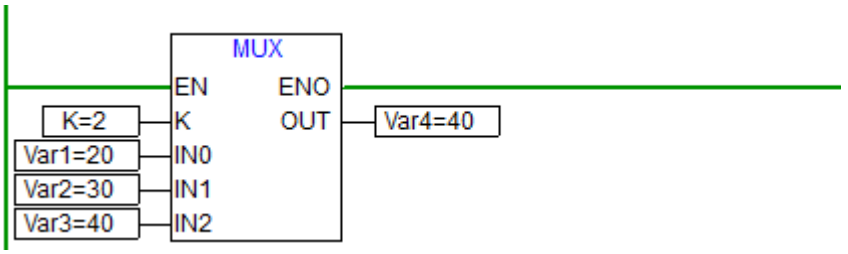
5.4.4.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
K	Input	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT	Select control number	0	FALSE
IN0~INn	Input	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TOD, DT	Enter 1~N operands	0	FALSE
OUT	Output	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TOD, DT	Instruction result	0	FALSE

5.4.4.3 Example

The example is an instruction OUT:=MUX(K,IN0,…,INn), where K is the control number, IN0,…,INn is the input data, and OUT is the output result. When the control number K is 0, the IN0th input data is selected as the OUT output. When the control number K is n, the INnth input data is selected as the OUT output.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			INT	20	FALSE	Not Public	
0002	Var2			INT	30	FALSE	Not Public	
0003	Var3			INT	40	FALSE	Not Public	
0004	Var4			INT	40	FALSE	Not Public	
0005	K			INT	2	FALSE	Not Public	

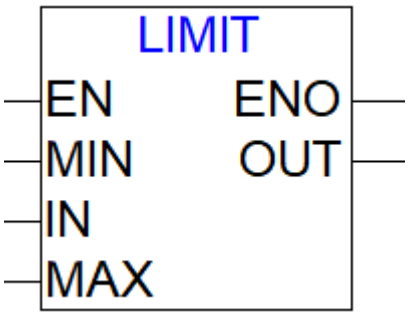



```
1  Var4 := MUX(K, Var1, Var2, Var3);   Var4 = 40   K = 2   Var1 = 20   Var2 = 30   Var3 = 40
```

5.4.5 LIMIT (Limit)

5.4.5.1 Function

It is to determine whether the input data is between the minimum and the maximum values. If the input data is between the two, the input data is directly output as output data. If the input data is greater than the maximum value, the maximum value is used as the output value. If the input data is smaller than the minimum value, the minimum value is used as the output value.



LIMIT Functional Block

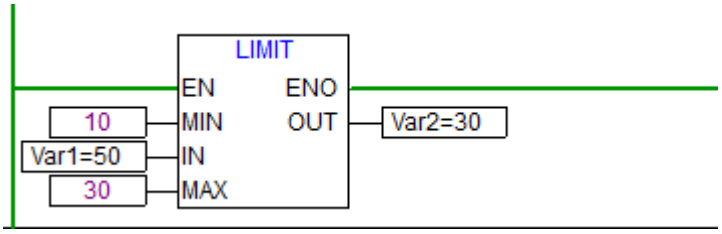
5.4.5.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
MIN	Input	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TOD, DT	Lower limits of operand	0	
MAX	Input	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TOD, DT	Upper limits of operand	0	
IN	Input	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TOD, DT	Enter one operand	0	FALSE
OUT	Output	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TOD, DT	Instruction result	0	FALSE

5.4.5.3 Example

The example uses the LIMIT instruction OUT := LIMIT(Min, IN, Max). To determine whether the value of the input data IN is between MIN and MAX, if the input data IN is between the two, the OUT output is IN; if IN is smaller than MIN, then the OUT output is MIN, and; if IN is greater than MAX, the OUT output is MAX.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			LREAL	50	FALSE	Not Public	
0002	Var2			LREAL	30	FALSE	Not Public	




```
1   Var2 := LIMIT(10, Var1, 30);
```

Var2 = 30

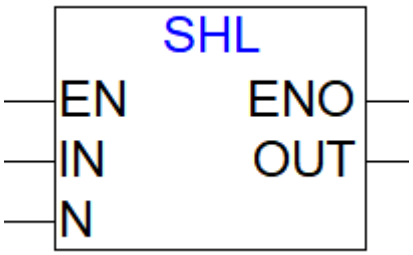
Var1 = 50

5.5 Shift Operation

5.5.1 SHL (Shift Left)

5.5.1.1 Function

It is to shift left the operand bit by bit. The bits shifted out on the left are not processed, and 0 is automatically added to the right.



SHL Functional Block

5.5.1.2 Parameter

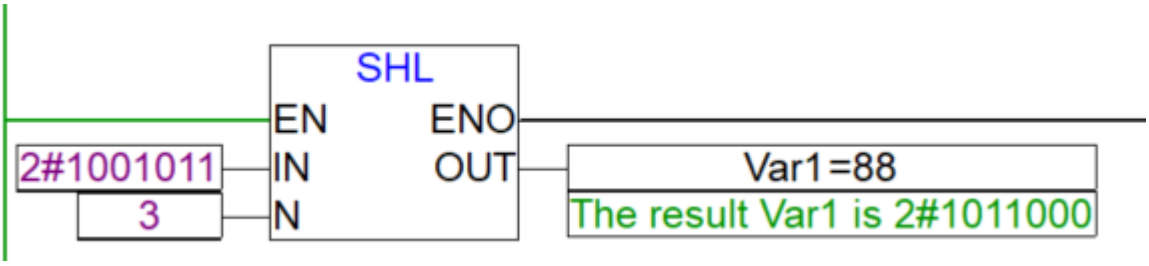
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
IN	Input	BYTE, INT, WORD, SINT, UINT, USINT, DINT, DWORD	Enter one operand	0	FALSE
N	Input	BYTE, INT, WORD, SINT, UINT, USINT, DINT, DWORD	Number of left shifts	0	FALSE
OUT	Output	BYTE, INT, WORD, SINT, UINT, USINT,	Instruction result	0	FALSE

		DINT, DWORD			

5.5.1.3 Example

The example uses the SHL instruction to implement a bitwise left shift of each bit of the input operand IN by shifting it to the left by N bits and discarding it, and automatically filling the vacated bits on the right with 0.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			INT	88	FALSE	Not Public	

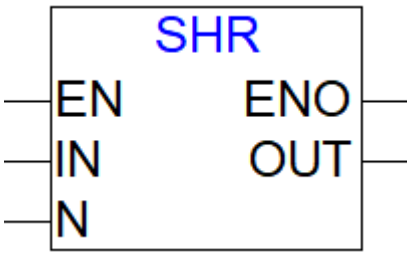



```
1  Var1:= SHL (2#1001011,3);                    Var1 = 88
2  (*The result Var1 is 2#1011000*)
```

5.5.2 SHR (Shift Right)

5.5.2.1 Function

It is to shift right the operand bit by bit. The bits shifted out on the right are not processed, and 0 is automatically added to the left.



SHR Functional Block

5.5.2.2 Parameter

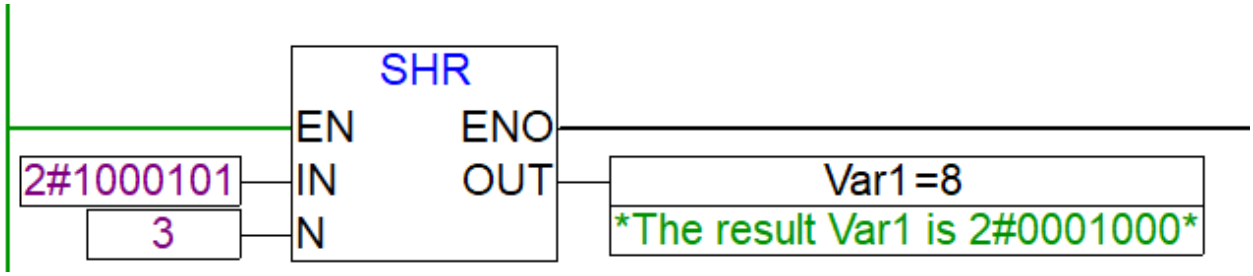
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		

ENO	Output	BOOL	Enable output		
IN	Input	BYTE, INT, WORD, SINT, UINT, USINT, DINT, DWORD	Enter one operand	0	FALSE
N	Input	BYTE, INT, WORD, SINT, UINT, USINT, DINT, DWORD	Number of right shifts	0	FALSE
OUT	Output	BYTE, INT, WORD, SINT, UINT, USINT, DINT, DWORD	Instruction result	0	FALSE

5.5.2.3 Example

The example uses the SHR instruction to implement a bitwise right shift of each bit of the input operand IN by shifting it to the right by N bits and discarding it, and automatically filling the vacated bits on the left with 0.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			BYTE	8	FALSE	Not Public	

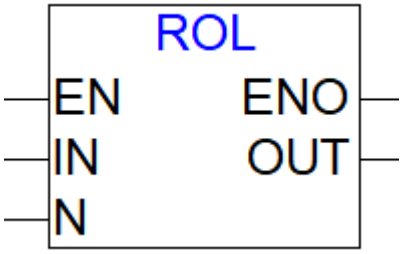



```
1  Var1:= SHR (2#1000101,3);                    Var1 = 8
2  (*The result Var1 is 2#0001000*)
```

5.5.3 ROL (Ring Shift Left)

5.5.3.1 Function

Ring shift operands to the left bit by bit, and the bits shifted out on the left are directly added to the lowest bit on the right.



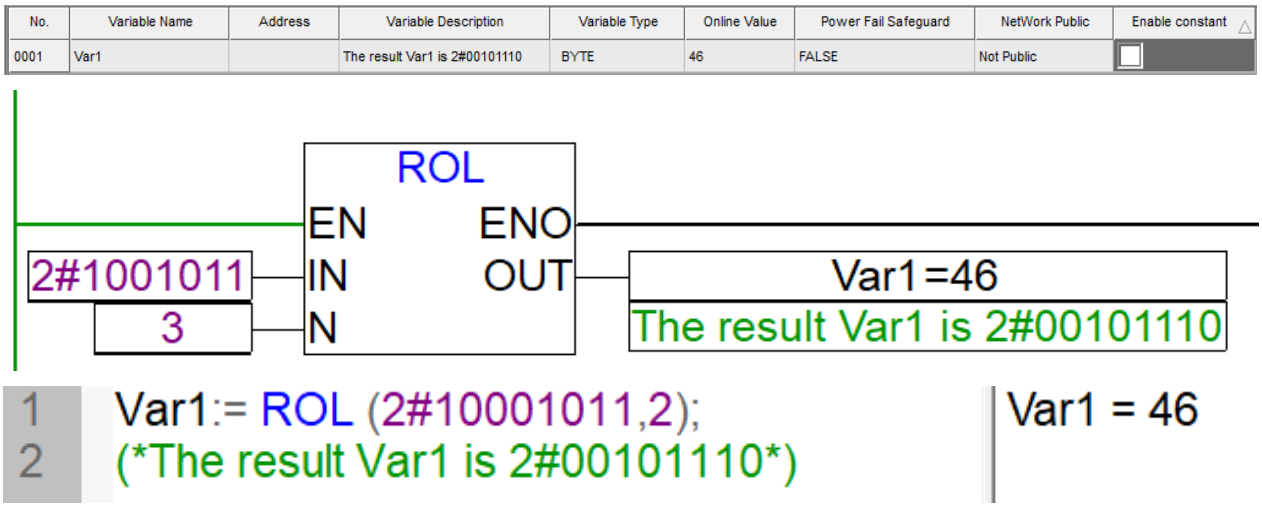
ROL Functional Block

5.5.3.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
IN	Input	BYTE, INT, WORD, SINT, UINT, USINT, DINT, DWORD	Enter one operand	0	FALSE
N	Input	BYTE, INT, WORD, SINT, UINT, USINT, DINT, DWORD	Number of left shifts	0	FALSE
OUT	Output	BYTE, INT, WORD, SINT, UINT, USINT, DINT, DWORD	Instruction result	0	FALSE

5.5.3.3 Example

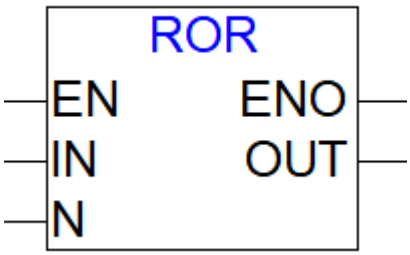
The example uses the ROL instruction to realize a bit-wise ring shift left of each bit of the operand IN, and the bits shifted out on the left are directly added to the lowest bit on the right for n times.



5.5.4 ROR (Ring Shift Right)

5.5.4.1 Function

It is to ring shift operands to the right bit by bit, and the bits shifted out on the right are directly added to the lowest bit on the left.



ROR Functional Block

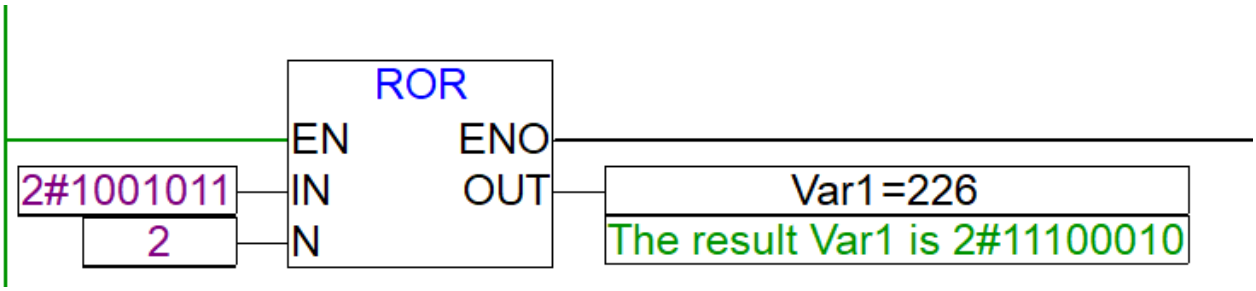
5.5.4.2 Parameter

Parameter	Statement	Data Type	Description	Default Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
IN	Input	BYTE, INT, WORD, SINT, UINT, USINT, DINT, DWORD	Enter one operand	0	FALSE
N	Input	BYTE, INT, WORD, SINT, UINT, USINT, DINT, DWORD	Number of right shifts	0	FALSE
OUT	Output	BYTE, INT, WORD, SINT, UINT, USINT, DINT, DWORD	Instruction result	0	FALSE

5.5.4.3 Example

The example uses the ROR instruction to realize a bit-wise ring shift right of each bit of the operand IN, and the bits shifted out on the right are directly added to the lowest bit on the left for n times.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1		The result Var1 is 2#11100010	BYTE	226	FALSE	Not Public	




```
1  Var1 := ROR(2#10001011, 2);
2  (*The result Var1 is 2#11100010*)
```

Var1 = 226

5.6 Data Type Conversion

5.6.1 BOOL_TO_TYPE (Boolean Type Conversion Instruction)

5.6.1.1 Function

Convert a Boolean data type to another data type.

When the output is a numeric type, if the input is TRUE, the output is 1; if the input is FALSE, the output is 0.

When the output is a string type, if the input is TRUE, the output is the string 'TRUE'; if the input is FALSE, the output is the string 'FALSE'.

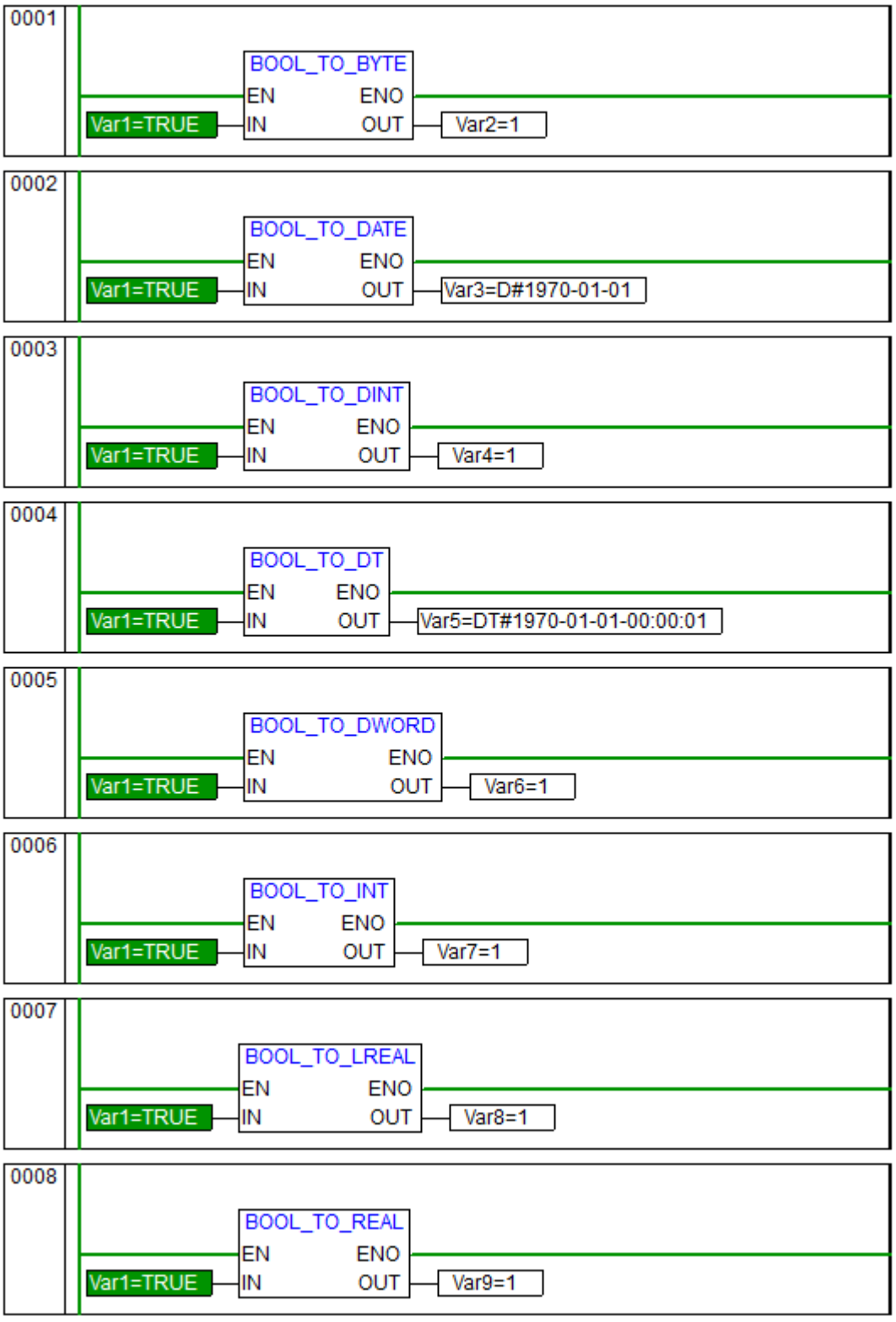
5.6.1.2 Parameter

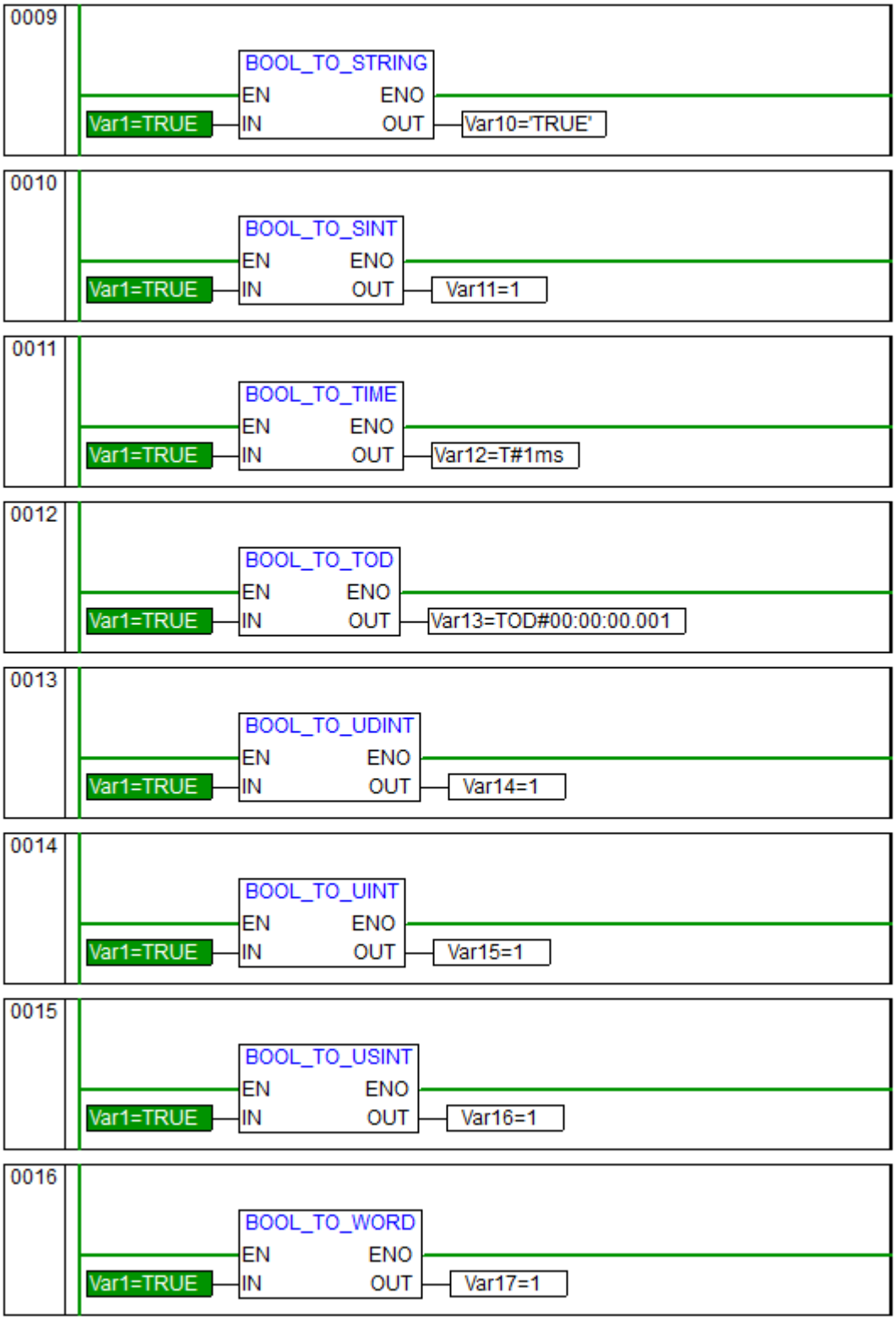
Parameter	Statement	Data Type	Description
IN	Input	BOOL	Enter the Boolean variable to be converted
OUT	Output	BYTE, DATE, DINT, DT, DWORD, INT, LREAL, REAL, STRING, SINT, TIME, TOD, UDINT, UINT, USINT, WORD	Converted data type

5.6.1.3 Example

The following example uses the BOOL conversion instruction to convert BOOL variables to other data types. When the input EN detects a rising edge, the BOOL conversion instruction starts the conversion function and stores the calculation result in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			BOOL	FALSE	FALSE	Not Public	
0002	Var2			BYTE	0	FALSE	Not Public	
0003	Var3			DATE	D#1970-01-01	FALSE	Not Public	
0004	Var4			DINT	0	FALSE	Not Public	
0005	Var5			DT	DT#1970-01-01...	FALSE	Not Public	
0006	Var6			DWORD	0	FALSE	Not Public	
0007	Var7			INT	0	FALSE	Not Public	
0008	Var8			LREAL	0	FALSE	Not Public	
0009	Var9			REAL	0	FALSE	Not Public	
0010	Var10			STRING(80)	''	FALSE	Not Public	
0011	Var11			SINT	0	FALSE	Not Public	
0012	Var12			TIME	T#0MS	FALSE	Not Public	
0013	Var13			TOD	TOD#00:00:00	FALSE	Not Public	
0014	Var14			UDINT	0	FALSE	Not Public	
0015	Var15			UINT	0	FALSE	Not Public	
0016	Var16			USINT	0	FALSE	Not Public	
0017	Var17			WORD	0	FALSE	Not Public	


0001
BOOL_TO_BYTE
EN
ENO
Var1=TRUE
IN
OUT
Var2=1
0002
BOOL_TO_DATE
EN
ENO
Var1=TRUE
IN
OUT
Var3=D#1970-01-01
0003
BOOL_TO_DINT
EN
ENO
Var1=TRUE
IN
OUT
Var4=1
0004
BOOL_TO_DT
EN
ENO
Var1=TRUE
IN
OUT
Var5=DT#1970-01-01-00:00:01
0005
BOOL_TO_DWORD
EN
ENO
Var1=TRUE
IN
OUT
Var6=1
0006
BOOL_TO_INT
EN
ENO
Var1=TRUE
IN
OUT
Var7=1
0007
BOOL_TO_LREAL
EN
ENO
Var1=TRUE
IN
OUT
Var8=1
0008
BOOL_TO_REAL
EN
ENO
Var1=TRUE
IN
OUT
Var9=1


0009
BOOL_TO_STRING
EN
ENO
Var1=TRUE
IN
OUT
Var10='TRUE'
0010
BOOL_TO_SINT
EN
ENO
Var1=TRUE
IN
OUT
Var11=1
0011
BOOL_TO_TIME
EN
ENO
Var1=TRUE
IN
OUT
Var12=T#1ms
0012
BOOL_TO_TOD
EN
ENO
Var1=TRUE
IN
OUT
Var13=TOD#00:00:00.001
0013
BOOL_TO_UDINT
EN
ENO
Var1=TRUE
IN
OUT
Var14=1
0014
BOOL_TO_UINT
EN
ENO
Var1=TRUE
IN
OUT
Var15=1
0015
BOOL_TO_USINT
EN
ENO
Var1=TRUE
IN
OUT
Var16=1
0016
BOOL_TO_WORD
EN
ENO
Var1=TRUE
IN
OUT
Var17=1

```
1   Var2:= BOOL_TO_BYTE(Var1);
2   Var3 := BOOL_TO_DATE(Var1);
3   Var4 := BOOL_TO_DINT(Var1);
4   Var5 := BOOL_TO_DT(Var1);
5   Var6 := BOOL_TO_DWORD(Var1);
6   Var7 := BOOL_TO_INT(Var1);
7   Var8 := BOOL_TO_LREAL(Var1);
8   Var9 := BOOL_TO_REAL(Var1);
9   Var10 := BOOL_TO_STRING(Var1);
10  Var11 := BOOL_TO_SINT(Var1);
11  Var12 := BOOL_TO_TIME(Var1);
12  Var13 := BOOL_TO_TOD(Var1);
13  Var14 := BOOL_TO_UDINT(Var1);
14  Var15 := BOOL_TO_UINT(Var1);
15  Var16 := BOOL_TO_USINT(Var1);
16  Var17 := BOOL_TO_WORD(Var1);
```

```
Var2 = 1                          Var1 = TRUE
Var3 = D#1970-01-01               Var1 = TRUE
Var4 = 1                          Var1 = TRUE
Var5 = DT#1970-01-01-00:00:01     Var1 = TRUE
Var6 = 1                          Var1 = TRUE
Var7 = 1                          Var1 = TRUE
Var8 = 1                          Var1 = TRUE
Var9 = 1                          Var1 = TRUE
Var10 = 'TRUE'                    Var1 = TRUE
Var11 = 1                         Var1 = TRUE
Var12 = T#1ms                     Var1 = TRUE
Var13 = TOD#00:00:00.001          Var1 = TRUE
Var14 = 1                         Var1 = TRUE
Var15 = 1                         Var1 = TRUE
Var16 = 1                         Var1 = TRUE
Var17 = 1                         Var1 = TRUE
```

5.6.2 BYTE_TO_TYPE (Byte Type Conversion Instruction)

5.6.2.1 Function

Convert the byte type to other data types.

In the case of BYTE_TO_BOOL, the output is TRUE when the input is not equal to 0, and FALSE when the input is equal to 0.

In the case of BYTE_TO_TIME and BYTE_TO_TOD, the input will be converted in milliseconds.

In the case of BYTE_TO_DATE and BYTE_TO_DT, the input will be converted in seconds.

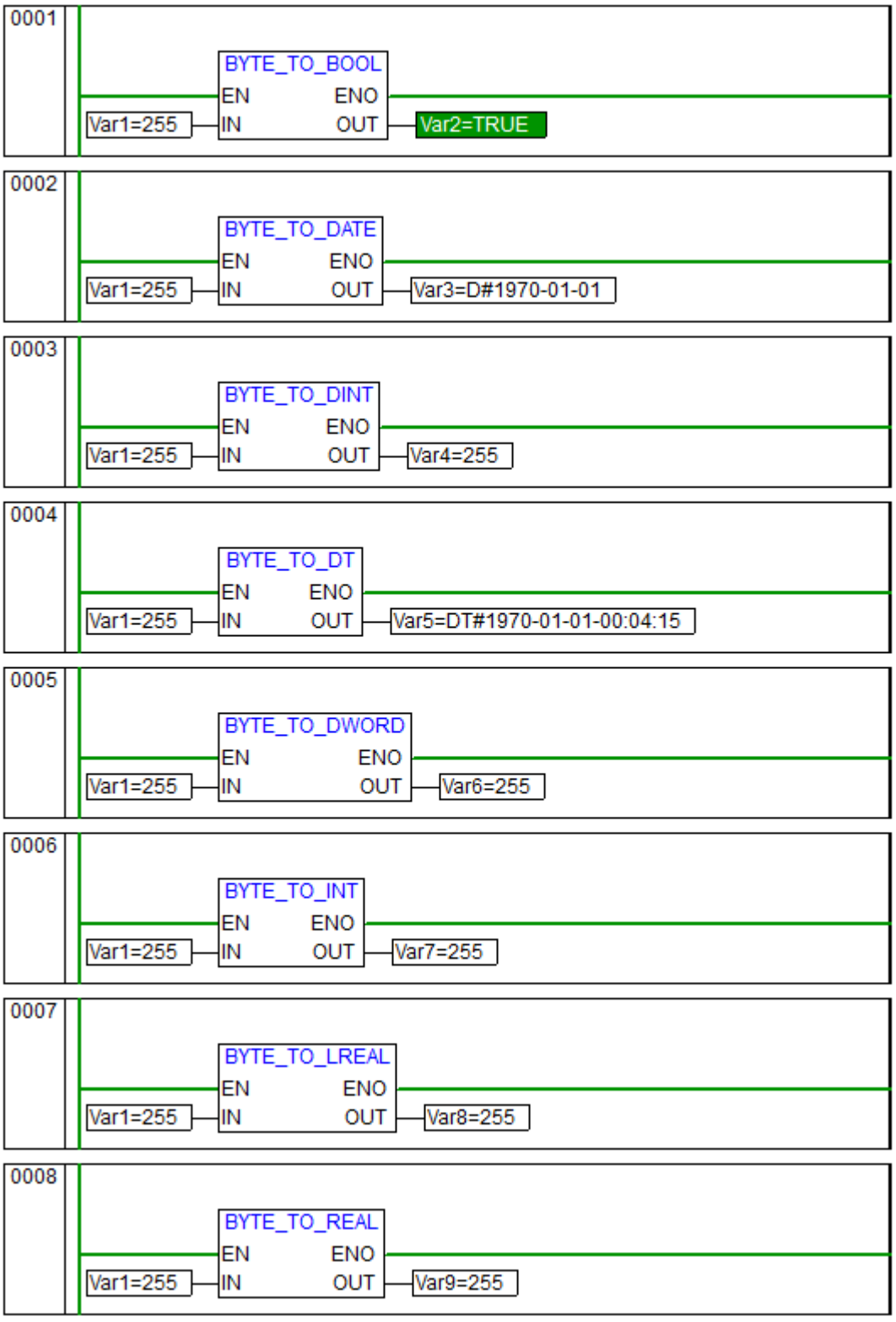
5.6.2.2 Parameter

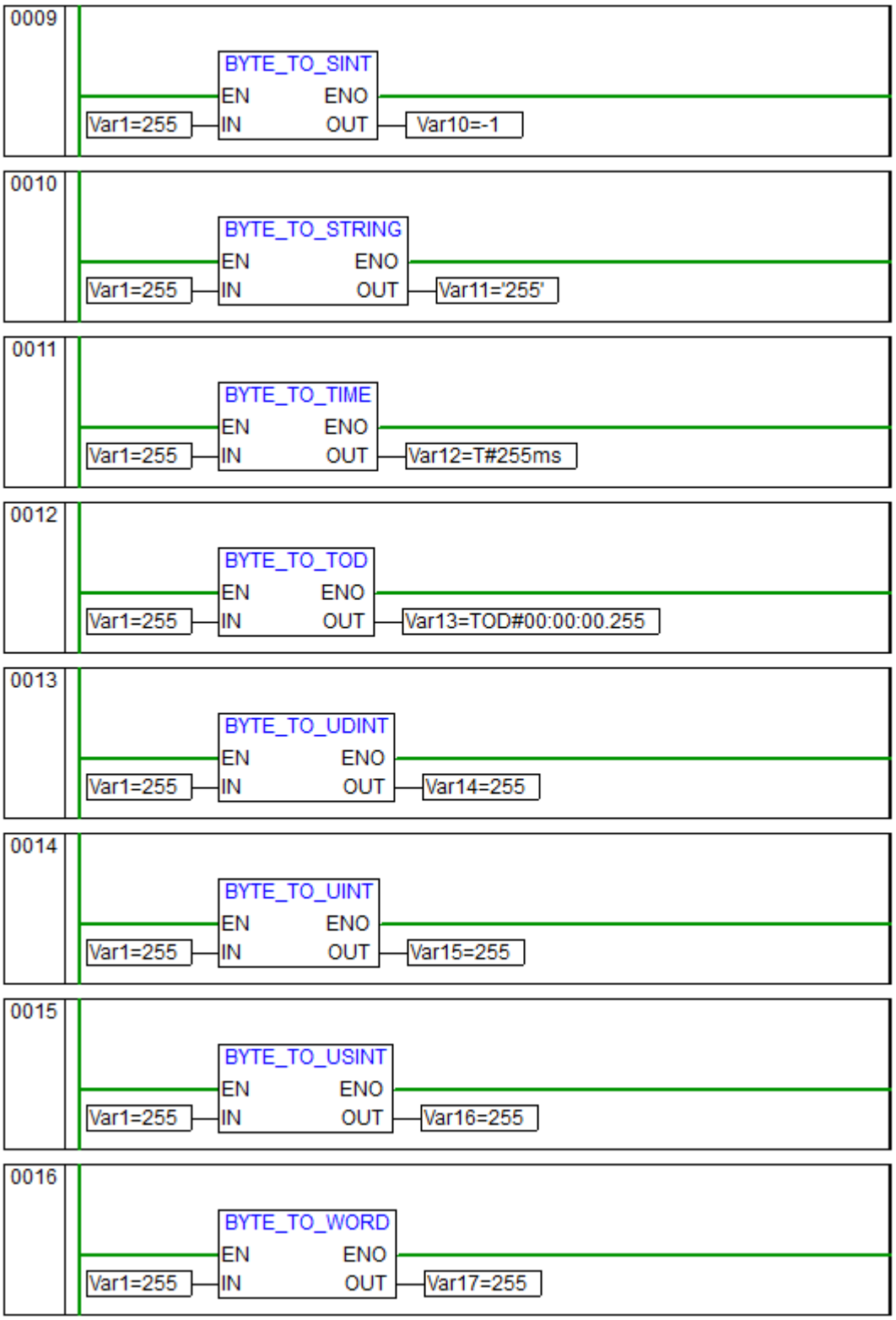
Parameter	Statement	Data Type	Description
IN	Input	BYTE	Enter the byte variables to be converted
OUT	Output	BOOL, DATE, DINT, DT, DWORD, INT, LREAL, REAL, STRING, SINT, TIME, TOD, UDINT, UINT, USINT, WORD	Converted data type

5.6.2.3 Example

The following example uses the BYTE conversion instruction to convert BYTE variables to other data types. When the input EN detects a rising edge, the BYTE conversion instruction starts the conversion function and stores the calculation result in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			BYTE	255	FALSE	Not Public	
0002	Var2			BOOL	TRUE	FALSE	Not Public	
0003	Var3			DATE	D#1970-01-01	FALSE	Not Public	
0004	Var4			DINT	255	FALSE	Not Public	
0005	Var5			DT	DT#1970-01-01...	FALSE	Not Public	
0006	Var6			DWORD	255	FALSE	Not Public	
0007	Var7			INT	255	FALSE	Not Public	
0008	Var8			LREAL	255	FALSE	Not Public	
0009	Var9			REAL	255	FALSE	Not Public	
0010	Var10			STRING(80)	''	FALSE	Not Public	
0011	Var11			SINT	0	FALSE	Not Public	
0012	Var12			TIME	T#0MS	FALSE	Not Public	
0013	Var13			TOD	TOD#00:00:00	FALSE	Not Public	
0014	Var14			UDINT	255	FALSE	Not Public	
0015	Var15			UINT	255	FALSE	Not Public	
0016	Var16			USINT	255	FALSE	Not Public	
0017	Var17			WORD	255	FALSE	Not Public	


0001
BYTE_TO_BOOL
EN
ENO
Var1=255
IN
OUT
Var2=TRUE
0002
BYTE_TO_DATE
EN
ENO
Var1=255
IN
OUT
Var3=D#1970-01-01
0003
BYTE_TO_DINT
EN
ENO
Var1=255
IN
OUT
Var4=255
0004
BYTE_TO_DT
EN
ENO
Var1=255
IN
OUT
Var5=DT#1970-01-01-00:04:15
0005
BYTE_TO_DWORD
EN
ENO
Var1=255
IN
OUT
Var6=255
0006
BYTE_TO_INT
EN
ENO
Var1=255
IN
OUT
Var7=255
0007
BYTE_TO_LREAL
EN
ENO
Var1=255
IN
OUT
Var8=255
0008
BYTE_TO_REAL
EN
ENO
Var1=255
IN
OUT
Var9=255


0009
BYTE_TO_SINT
EN
ENO
Var1=255
IN
OUT
Var10=-1
0010
BYTE_TO_STRING
EN
ENO
Var1=255
IN
OUT
Var11='255'
0011
BYTE_TO_TIME
EN
ENO
Var1=255
IN
OUT
Var12=T#255ms
0012
BYTE_TO_TOD
EN
ENO
Var1=255
IN
OUT
Var13=TOD#00:00:00.255
0013
BYTE_TO_UDINT
EN
ENO
Var1=255
IN
OUT
Var14=255
0014
BYTE_TO_UINT
EN
ENO
Var1=255
IN
OUT
Var15=255
0015
BYTE_TO_USINT
EN
ENO
Var1=255
IN
OUT
Var16=255
0016
BYTE_TO_WORD
EN
ENO
Var1=255
IN
OUT
Var17=255

```
1   Var2 := BYTE_TO_BOOL(Var1);
2   Var3 := BYTE_TO_DATE(Var1);
3   Var4 := BYTE_TO_DINT(Var1);
4   Var5 := BYTE_TO_DT(Var1);
5   Var6 := BYTE_TO_DWORD(Var1);
6   Var7 := BYTE_TO_INT(Var1);
7   Var8 := BYTE_TO_LREAL(Var1);
8   Var9 := BYTE_TO_REAL(Var1);
9   Var10 := BYTE_TO_SINT(Var1);
10  Var11 := BYTE_TO_STRING(Var1);
11  Var12 := BYTE_TO_TIME(Var1);
12  Var13 := BYTE_TO_TOD(Var1);
13  Var14 := BYTE_TO_UDINT(Var1);
14  Var15 := BYTE_TO_UINT(Var1);
15  Var16 := BYTE_TO_USINT(Var1);
16  Var17 := BYTE_TO_WORD(Var1);
```

```
Var2 = TRUE                      Var1 = 255
Var3 = D#1970-01-01              Var1 = 255
Var4 = 255                       Var1 = 255
Var5 = DT#1970-01-01-00:04:15    Var1 = 255
Var6 = 255                       Var1 = 255
Var7 = 255                       Var1 = 255
Var8 = 255                       Var1 = 255
Var9 = 255                       Var1 = 255
Var10 = -1                       Var1 = 255
Var11 = '255'                    Var1 = 255
Var12 = T#255ms                  Var1 = 255
Var13 = TOD#00:00:00.255         Var1 = 255
Var14 = 255                      Var1 = 255
Var15 = 255                      Var1 = 255
Var16 = 255                      Var1 = 255
Var17 = 255                      Var1 = 255
```

5.6.3 DATE_TO_TYPE (Date Type Conversion Instruction)

5.6.3.1 Function

It is to convert date data to other types of data. The date is stored internally in seconds, starting from January 1, 1970.

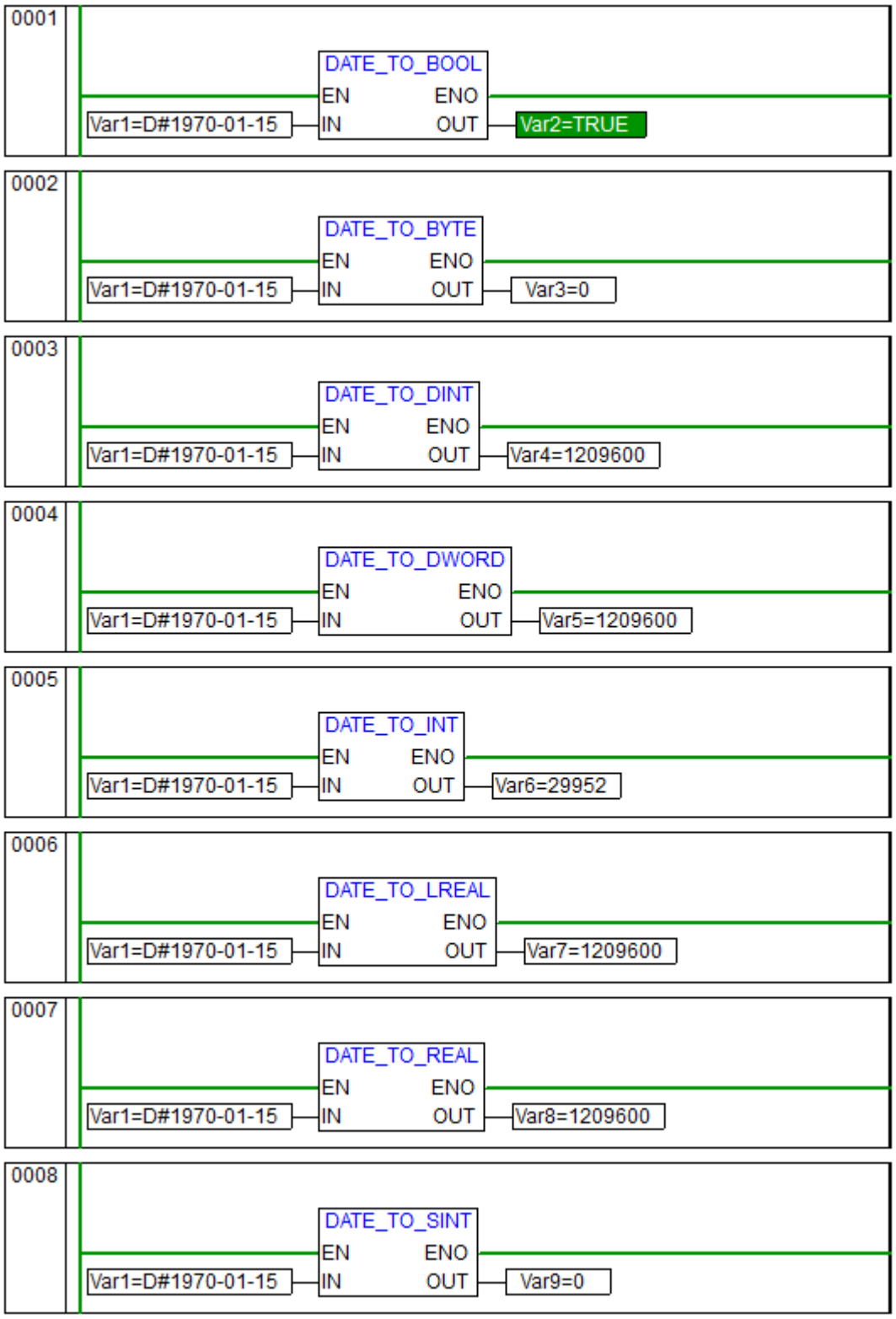
5.6.3.2 Parameter

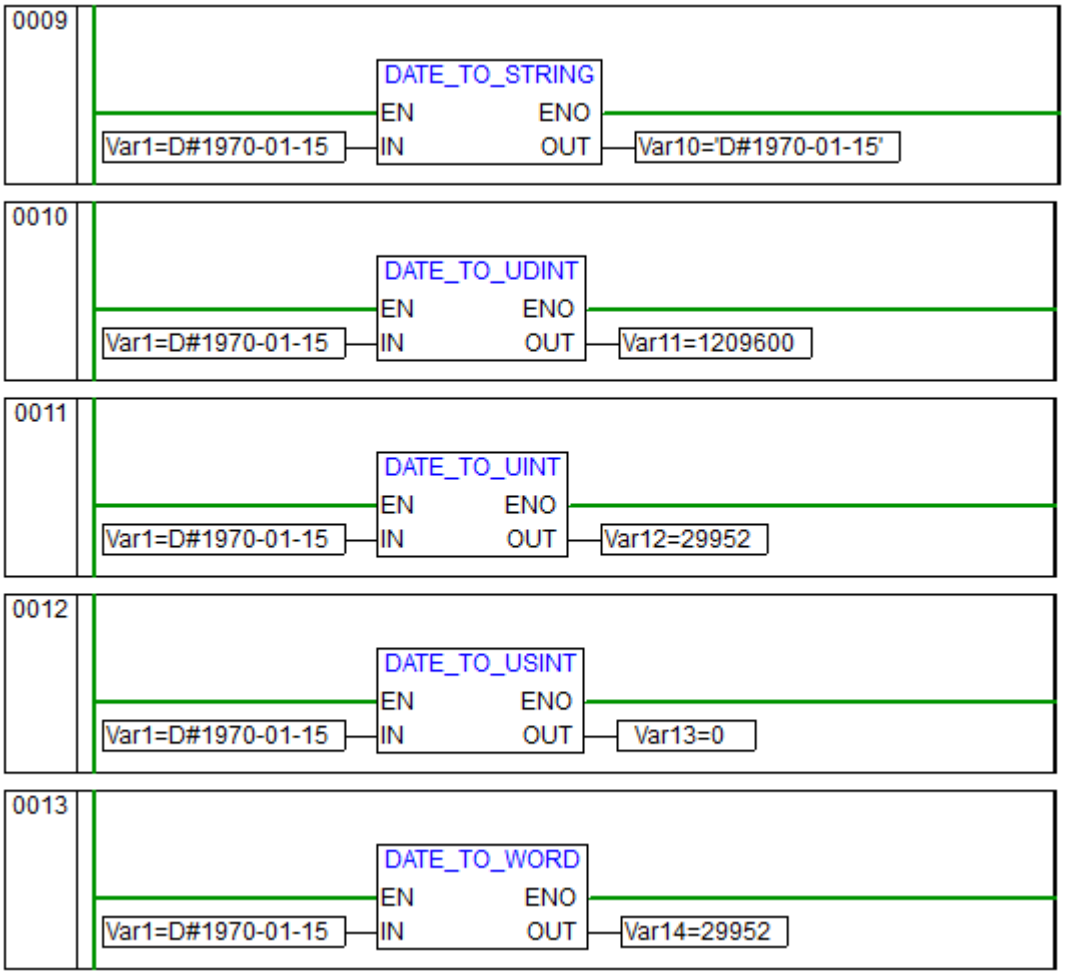
Parameter	Statement	Data Type	Description
IN	Input	DATE	Enter the date variables to be converted
OUT	Output	BOOL, BYTE, DINT, DWORD, INT, LREAL, REAL, STRING, SINT, UDINT, UINT, USINT, WORD	Converted data type

5.6.3.3 Example

The following example uses the DATE conversion instruction to convert DATE variables to other data types. When the input EN detects a rising edge, the DATE conversion instruction starts the conversion function and stores the calculation result in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			DATE	D#1970-01-01	FALSE	Not Public	
0002	Var2			BOOL	TRUE	FALSE	Not Public	
0003	Var3			BYTE	0	FALSE	Not Public	
0004	Var4			DINT	1209600	FALSE	Not Public	
0005	Var5			DWORD	1209600	FALSE	Not Public	
0006	Var6			INT	29952	FALSE	Not Public	
0007	Var7			LREAL	1209600	FALSE	Not Public	
0008	Var8			REAL	1209600	FALSE	Not Public	
0009	Var9			SINT	0	FALSE	Not Public	
0010	Var10			STRING(80)	'D#1970-01-01'	FALSE	Not Public	
0011	Var11			UDINT	1209600	FALSE	Not Public	
0012	Var12			UINT	29952	FALSE	Not Public	
0013	Var13			USINT	0	FALSE	Not Public	
0014	Var14			WORD	29952	FALSE	Not Public	


0001
DATE_TO_BOOL
EN
ENO
Var1=D#1970-01-15
IN
OUT
Var2=TRUE
0002
DATE_TO_BYTE
EN
ENO
Var1=D#1970-01-15
IN
OUT
Var3=0
0003
DATE_TO_DINT
EN
ENO
Var1=D#1970-01-15
IN
OUT
Var4=1209600
0004
DATE_TO_DWORD
EN
ENO
Var1=D#1970-01-15
IN
OUT
Var5=1209600
0005
DATE_TO_INT
EN
ENO
Var1=D#1970-01-15
IN
OUT
Var6=29952
0006
DATE_TO_LREAL
EN
ENO
Var1=D#1970-01-15
IN
OUT
Var7=1209600
0007
DATE_TO_REAL
EN
ENO
Var1=D#1970-01-15
IN
OUT
Var8=1209600
0008
DATE_TO_SINT
EN
ENO
Var1=D#1970-01-15
IN
OUT
Var9=0




```
1   Var2 := DATE_TO_BOOL(Var1);
2   Var3 := DATE_TO_BYTE(Var1);
3   Var4 := DATE_TO_DINT(Var1);
4   Var5 := DATE_TO_DWORD(Var1);
5   Var6 := DATE_TO_INT(Var1);
6   Var7 := DATE_TO_LREAL(Var1);
7   Var8 := DATE_TO_REAL(Var1);
8   Var9 := DATE_TO_SINT(Var1);
9   Var10 := DATE_TO_STRING(Var1);
10  Var11 := DATE_TO_UDINT(Var1);
11  Var12 := DATE_TO_UINT(Var1);
12  Var13 := DATE_TO_USINT(Var1);
13  Var14 := DATE_TO_WORD(Var1);
```

```
Var2 = TRUE                Var1 = D#1970-01-15
Var3 = 0                   Var1 = D#1970-01-15
Var4 = 1209600             Var1 = D#1970-01-15
Var5 = 1209600             Var1 = D#1970-01-15
Var6 = 29952               Var1 = D#1970-01-15
Var7 = 1209600             Var1 = D#1970-01-15
Var8 = 1209600             Var1 = D#1970-01-15
Var9 = 0                   Var1 = D#1970-01-15
Var10 = 'D#1970-01-15'     Var1 = D#1970-01-15
Var11 = 1209600            Var1 = D#1970-01-15
Var12 = 29952              Var1 = D#1970-01-15
Var13 = 0                  Var1 = D#1970-01-15
Var14 = 29952              Var1 = D#1970-01-15
```

5.6.4 DINT_TO_TYPE (Double Integer Type Conversion Instruction)

5.6.4.1 Function

Convert the double integer type to other data types.

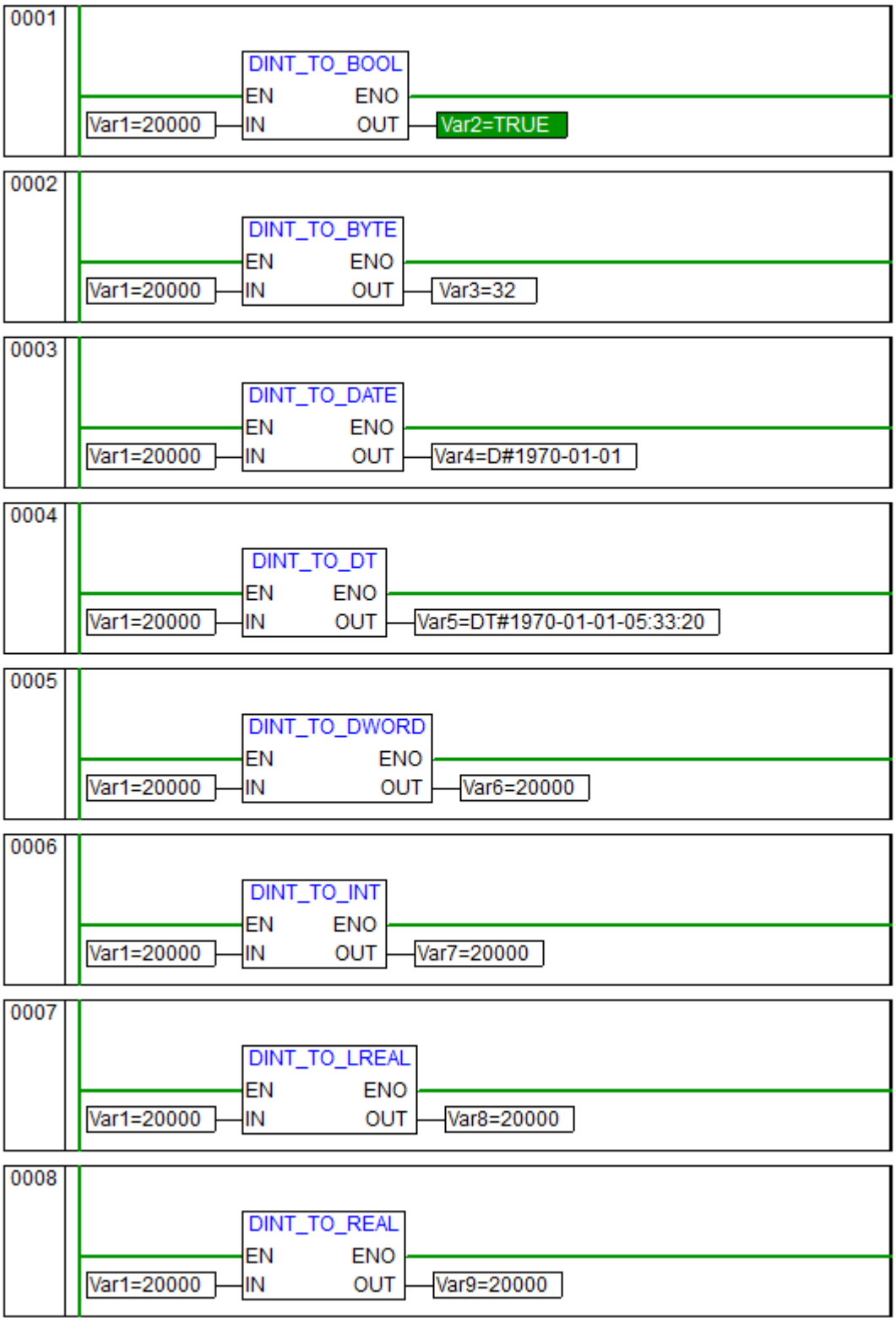
5.6.4.2 Parameter

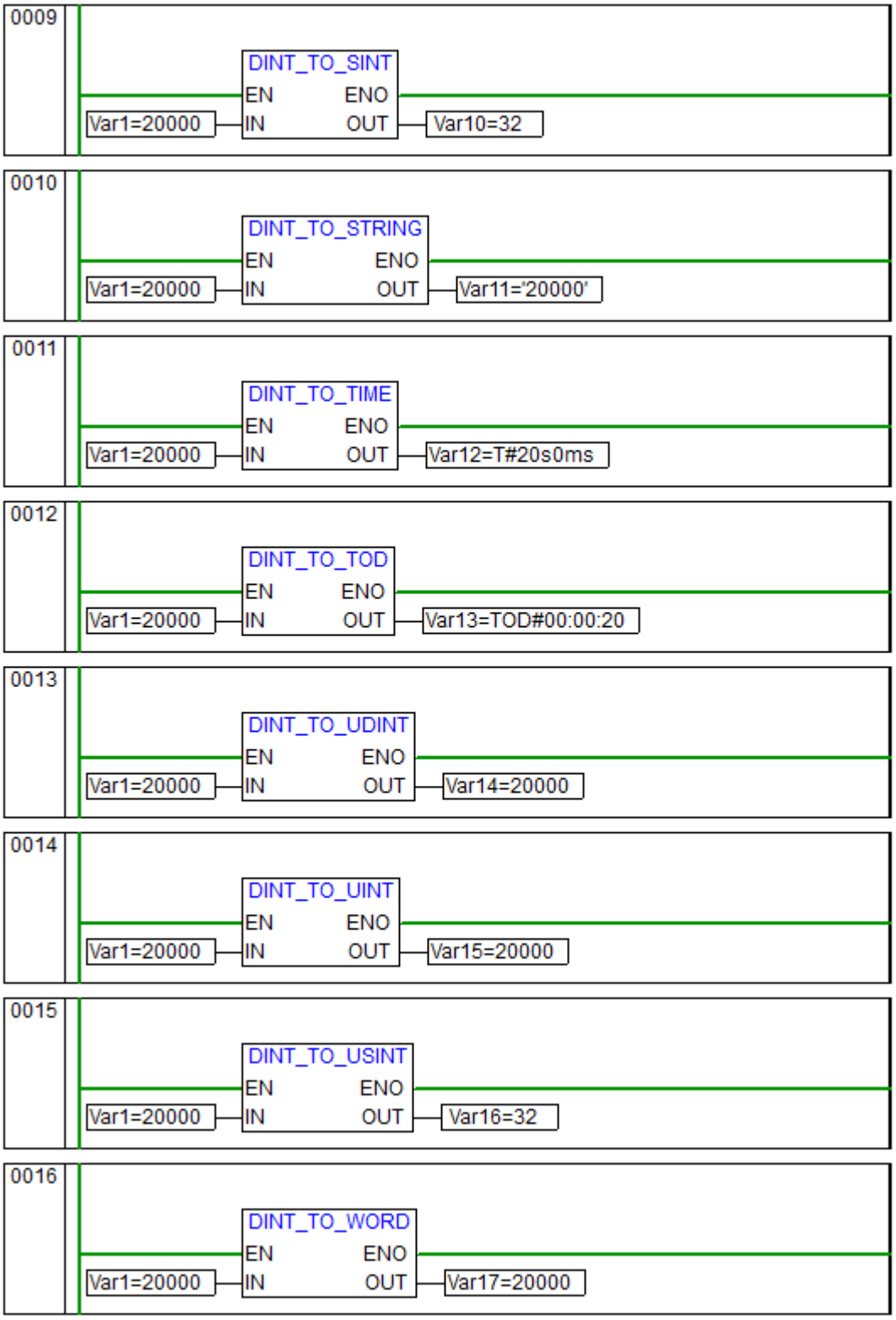
Parameter	Statement	Data Type	Description
IN	Input	DINT	Enter the double integer variable to be converted
OUT	Output	BOOL, BYTE, DATE, DT, DWORD, INT, LREAL, REAL, STRING, SINT, TIME, TOD, UDINT, UINT, USINT, WORD	Converted data type

5.6.4.3 Example

The following example uses the DINT conversion instruction to convert DINT variables to other data types. When the input EN detects a rising edge, the DINT conversion instruction starts the conversion function and stores the calculation result in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			DINT	20000	FALSE	Not Public	
0002	Var2			BOOL	TRUE	FALSE	Not Public	
0003	Var3			BYTE	32	FALSE	Not Public	
0004	Var4			DATE	D#1970-01-01	FALSE	Not Public	
0005	Var5			DT	DT#1970-01-01...	FALSE	Not Public	
0006	Var6			DWORD	20000	FALSE	Not Public	
0007	Var7			INT	20000	FALSE	Not Public	
0008	Var8			LREAL	20000	FALSE	Not Public	
0009	Var9			REAL	20000	FALSE	Not Public	
0010	Var10			SINT	32	FALSE	Not Public	
0011	Var11			STRING(80)	'20000'	FALSE	Not Public	
0012	Var12			TIME	T#20s0MS	FALSE	Not Public	
0013	Var13			TOD	TOD#00:00:00	FALSE	Not Public	
0014	Var14			DINT	20000	FALSE	Not Public	
0015	Var15			UINT	20000	FALSE	Not Public	
0016	Var16			USINT	32	FALSE	Not Public	
0017	Var17			WORD	20000	FALSE	Not Public	


0001
DINT_TO_BOOL
EN
ENO
Var1=20000
IN
OUT
Var2=TRUE
0002
DINT_TO_BYTE
EN
ENO
Var1=20000
IN
OUT
Var3=32
0003
DINT_TO_DATE
EN
ENO
Var1=20000
IN
OUT
Var4=D#1970-01-01
0004
DINT_TO_DT
EN
ENO
Var1=20000
IN
OUT
Var5=DT#1970-01-01-05:33:20
0005
DINT_TO_DWORD
EN
ENO
Var1=20000
IN
OUT
Var6=20000
0006
DINT_TO_INT
EN
ENO
Var1=20000
IN
OUT
Var7=20000
0007
DINT_TO_LREAL
EN
ENO
Var1=20000
IN
OUT
Var8=20000
0008
DINT_TO_REAL
EN
ENO
Var1=20000
IN
OUT
Var9=20000


0009
DINT_TO_SINT
EN
ENO
Var1=20000
IN
OUT
Var10=32
0010
DINT_TO_STRING
EN
ENO
Var1=20000
IN
OUT
Var11='20000'
0011
DINT_TO_TIME
EN
ENO
Var1=20000
IN
OUT
Var12=T#20s0ms
0012
DINT_TO_TOD
EN
ENO
Var1=20000
IN
OUT
Var13=TOD#00:00:20
0013
DINT_TO_UDINT
EN
ENO
Var1=20000
IN
OUT
Var14=20000
0014
DINT_TO_UINT
EN
ENO
Var1=20000
IN
OUT
Var15=20000
0015
DINT_TO_USINT
EN
ENO
Var1=20000
IN
OUT
Var16=32
0016
DINT_TO_WORD
EN
ENO
Var1=20000
IN
OUT
Var17=20000

```
1   Var2 := DINT_TO_BOOL(Var1);
2   Var3 := DINT_TO_BYTE(Var1);
3   Var4 := DINT_TO_DATE(Var1);
4   Var5 := DINT_TO_DT(Var1);
5   Var6 := DINT_TO_DWORD(Var1);
6   Var7 := DINT_TO_INT(Var1);
7   Var8 := DINT_TO_LREAL(Var1);
8   Var9 := DINT_TO_REAL(Var1);
9   Var10 := DINT_TO_SINT(Var1);
10  Var11 := DINT_TO_STRING(Var1);
11  Var12 := DINT_TO_TIME(Var1);
12  Var13 := DINT_TO_TOD(Var1);
13  Var14 := DINT_TO_UDINT(Var1);
14  Var15 := DINT_TO_UINT(Var1);
15  Var16 := DINT_TO_USINT(Var1);
16  Var17 := DINT_TO_WORD(Var1);
```

```
Var2 = TRUE                    Var1 = 20000
Var3 = 32                      Var1 = 20000
Var4 = D#1970-01-01            Var1 = 20000
Var5 = DT#1970-01-01-05:33:20  Var1 = 20000
Var6 = 20000                   Var1 = 20000
Var7 = 20000                   Var1 = 20000
Var8 = 20000                   Var1 = 20000
Var9 = 20000                   Var1 = 20000
Var10 = 32                     Var1 = 20000
Var11 = '20000'                Var1 = 20000
Var12 = T#20s0ms               Var1 = 20000
Var13 = TOD#00:00:20           Var1 = 20000
Var14 = 20000                  Var1 = 20000
Var15 = 20000                  Var1 = 20000
Var16 = 32                     Var1 = 20000
Var17 = 20000                  Var1 = 20000
```

5.6.5 DT_TO_TYPE (Date/Time Type Conversion Instruction)

5.6.5.1 Function

It is to convert date/time data to other types of data. The date is stored internally in seconds, starting from January 1, 1970.

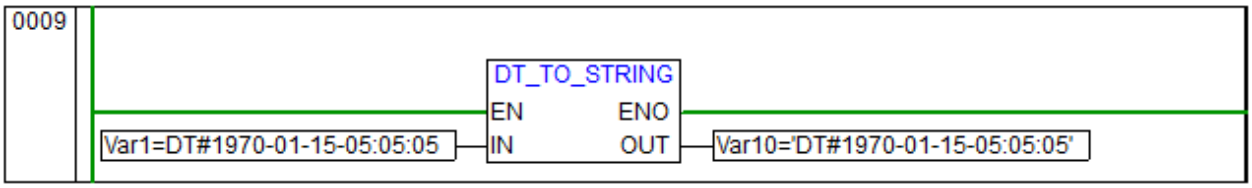
5.6.5.2 Parameter

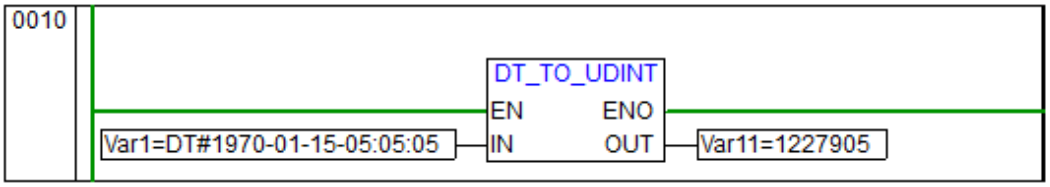
Parameter	Statement	Data Type	Description
IN	Input	DT	Enter the date/time variable to be converted
OUT	Output	BOOL, BYTE, DINT, DWORD, INT, LREAL, REAL, STRING, SINT, UDINT, UINT, USINT, WORD	Converted data type

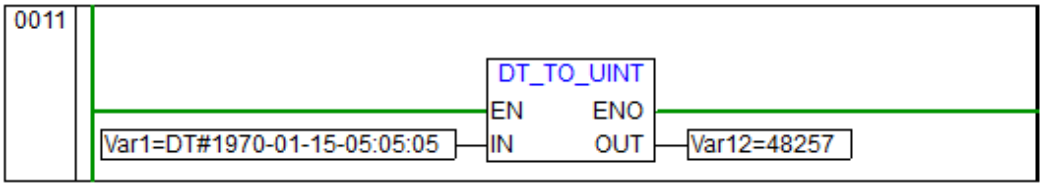
5.6.5.3 Example

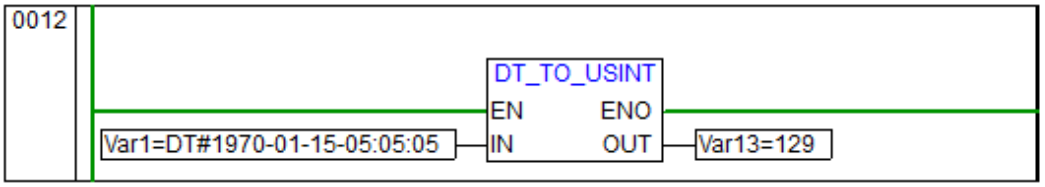
The following example uses the DT conversion instruction to convert DT variables to other data types. When the input EN detects a rising edge, the DT conversion instruction starts the conversion function and stores the calculation result in the output parameter OUT.

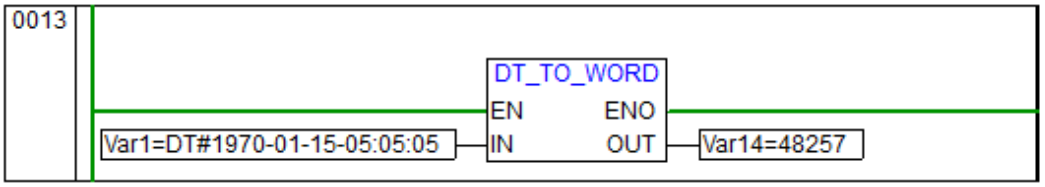
No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			DT	DT#1970-01-01...	FALSE	Not Public	
0002	Var2			BOOL	TRUE	FALSE	Not Public	
0003	Var3			BYTE	129	FALSE	Not Public	
0004	Var4			DINT	1227905	FALSE	Not Public	
0005	Var5			DWORD	1227905	FALSE	Not Public	
0006	Var6			INT	17279	FALSE	Not Public	
0007	Var7			LREAL	1227905	FALSE	Not Public	
0008	Var8			REAL	1227905	FALSE	Not Public	
0009	Var9			SINT	127	FALSE	Not Public	
0010	Var10			STRING(80)	'DT#1970-15-0...	FALSE	Not Public	
0011	Var11			USINT	0	FALSE	Not Public	
0012	Var12			UINT	048257	FALSE	Not Public	
0013	Var13			USINT	129	FALSE	Not Public	
0014	Var14			WORD	48257	FALSE	Not Public	

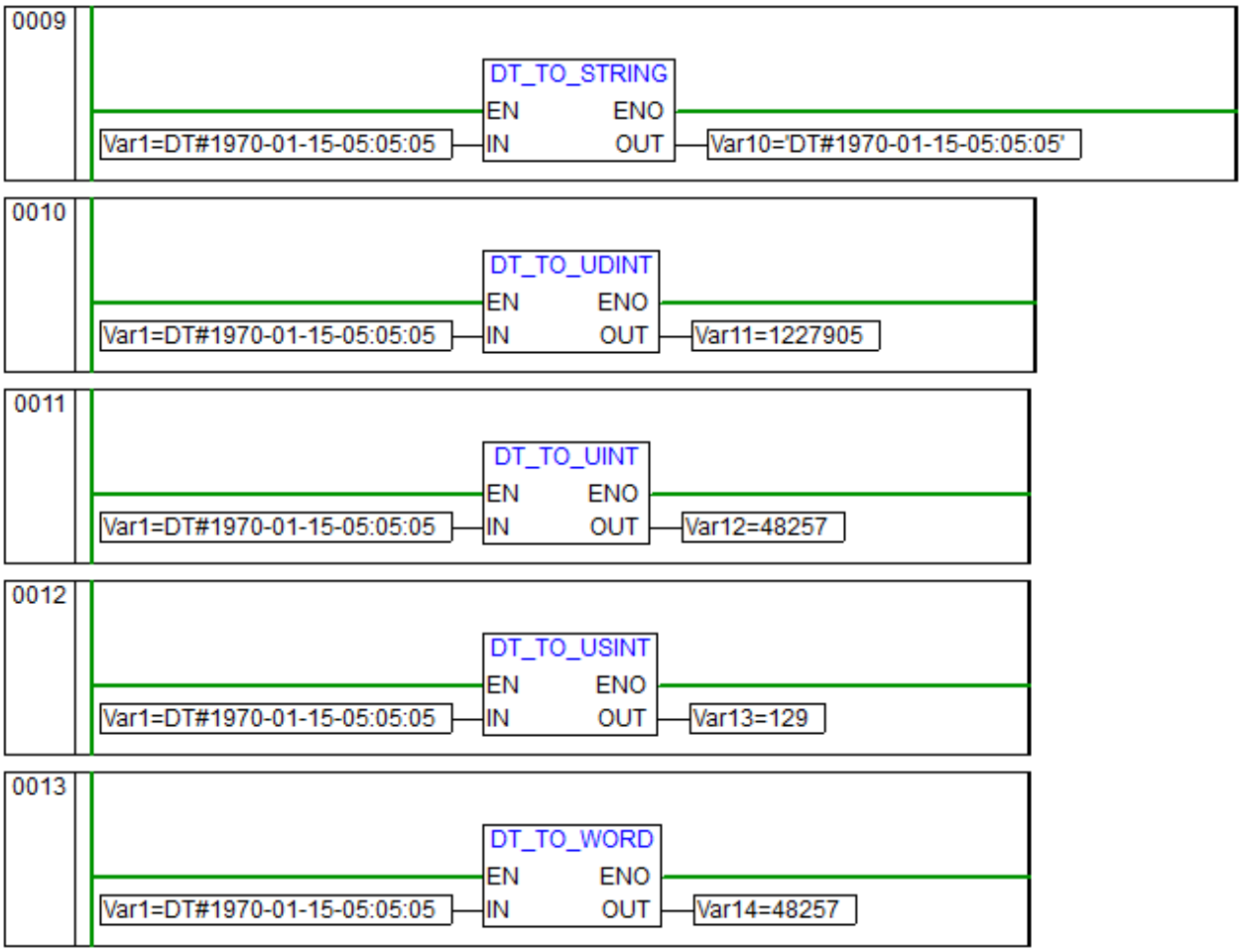













```
1  Var2 := DT_TO_BOOL(Var1);
2  Var3 := DT_TO_BYTE(Var1);
3  Var4 := DT_TO_DINT(Var1);
4  Var5 := DT_TO_DWORD(Var1);
5  Var6 := DT_TO_INT(Var1);
6  Var7 := DT_TO_LREAL(Var1);
7  Var8 := DT_TO_REAL(Var1);
8  Var9 := DT_TO_SINT(Var1);
9  Var10 := DT_TO_STRING(Var1);
10 Var11 := DT_TO_UDINT(Var1);
11 Var12 := DT_TO_UINT(Var1);
12 Var13 := DT_TO_USINT(Var1);
13 Var14 := DT_TO_WORD(Var1);
```

Var2 = TRUE	Var1 = DT#1970-01-15-05:05:05
Var3 = 129	Var1 = DT#1970-01-15-05:05:05
Var4 = 1227905	Var1 = DT#1970-01-15-05:05:05
Var5 = 1227905	Var1 = DT#1970-01-15-05:05:05
Var6 = -17279	Var1 = DT#1970-01-15-05:05:05
Var7 = 1227905	Var1 = DT#1970-01-15-05:05:05
Var8 = 1227905	Var1 = DT#1970-01-15-05:05:05
Var9 = -127	Var1 = DT#1970-01-15-05:05:05
Var10 = 'DT#1970-01-15-05:05:...	Var1 = DT#1970-01-15-05:05:05
Var11 = 1227905	Var1 = DT#1970-01-15-05:05:05
Var12 = 48257	Var1 = DT#1970-01-15-05:05:05
Var13 = 129	Var1 = DT#1970-01-15-05:05:05
Var14 = 48257	Var1 = DT#1970-01-15-05:05:05

5.6.6 DWORD_TO_TYPE (Double Word Type Conversion Instruction)

5.6.6.1 Function

Convert the double word type to other data types.

In the case of DWORD_TO_BOOL, the output is TRUE when the input is not equal to 0, and FALSE when

the input is equal to 0.

In the case of DWORD_TO_TIME and DWORD_TO_TOD, the input will be converted in milliseconds.

In the case of DWORD_TO_DATE and DWORD_TO_DT, the input will be converted in seconds.

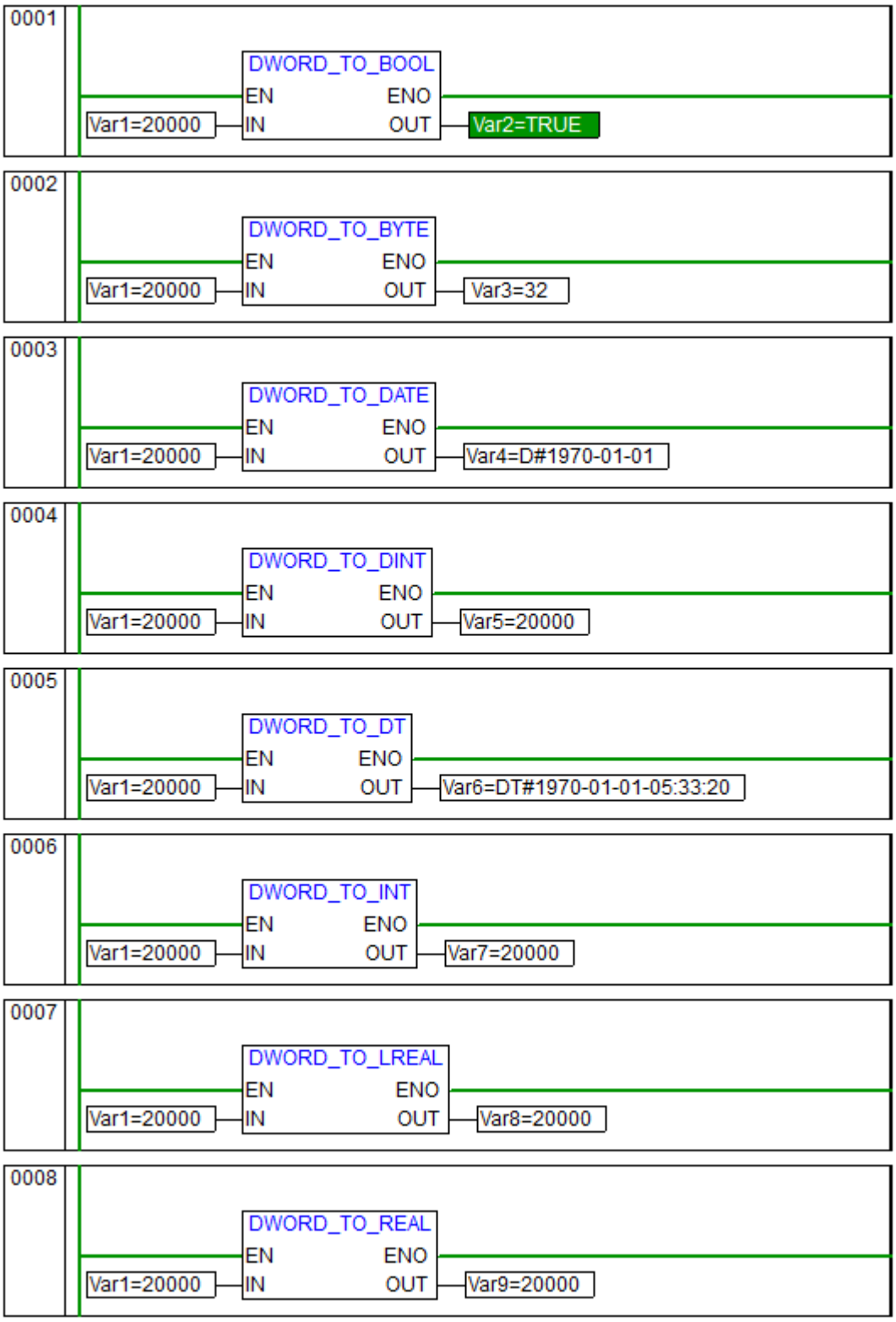
5.6.6.2 Parameter

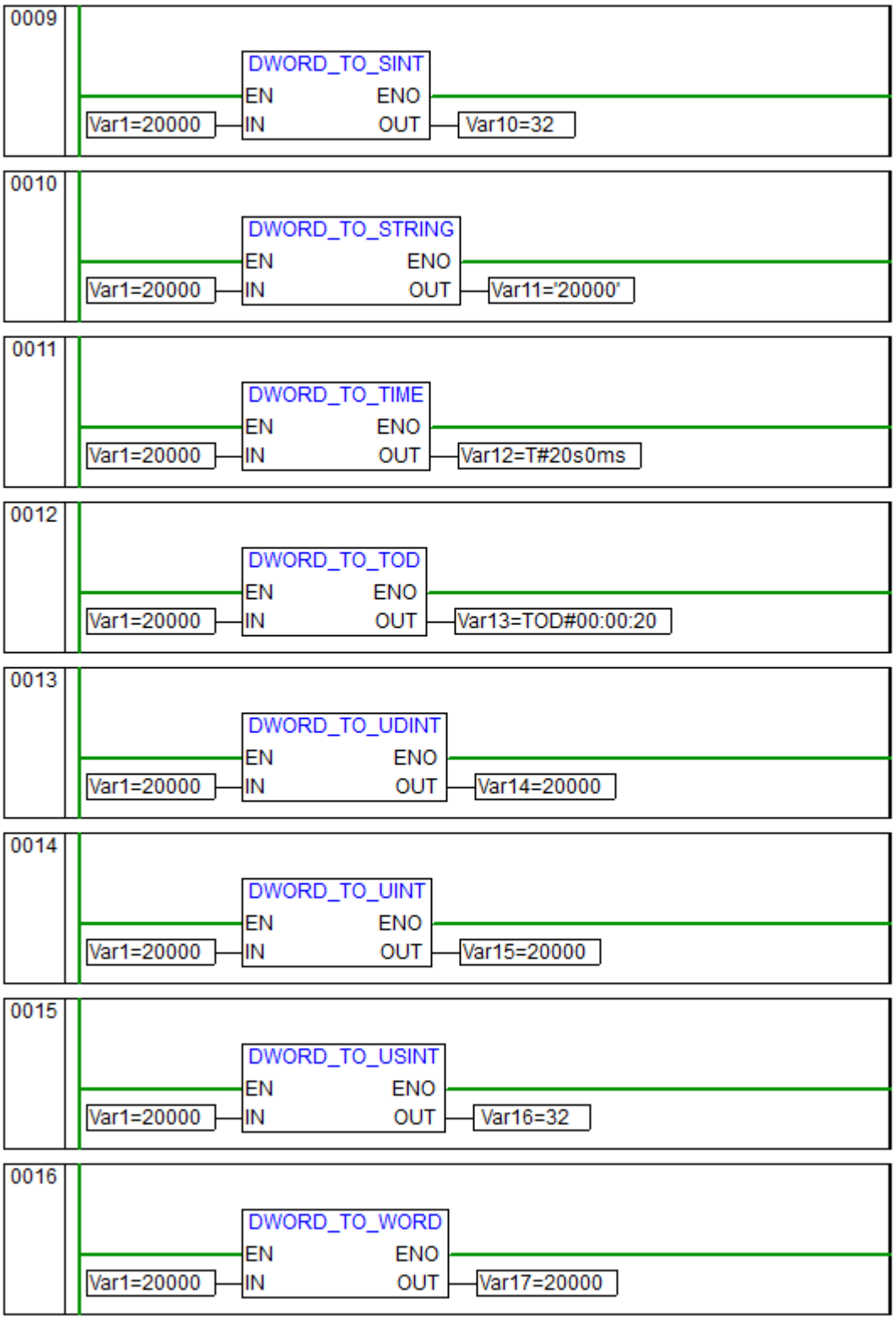
Parameter	Statement	Data Type	Description
IN	Input	DWORD	Enter the double-word variable to be converted
OUT	Output	BOOL, BYTE, DATE, DINT, DT, INT, LREAL, REAL, STRING, SINT, TIME, TOD, UDINT, UINT, USINT, WORD	Converted data type

5.6.6.3 Example

The following example uses the DWORD conversion instruction to convert DWORD variables to other data types. When the input EN detects a rising edge, the DWORD conversion instruction starts the conversion function and stores the calculation result in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			DWORD	20000	FALSE	Not Public	
0002	Var2			BOOL	TRUE	FALSE	Not Public	
0003	Var3			BYTE	32	FALSE	Not Public	
0004	Var4			DATE	D#1970-01-01	FALSE	Not Public	
0005	Var5			DINT	20000	FALSE	Not Public	
0006	Var6			DT	DT#1970-01-01...	FALSE	Not Public	
0007	Var7			INT	20000	FALSE	Not Public	
0008	Var8			LREAL	20000	FALSE	Not Public	
0009	Var9			REAL	20000	FALSE	Not Public	
0010	Var10			SINT	32	FALSE	Not Public	
0011	Var11			STRING(80)	'20000'	FALSE	Not Public	
0012	Var12			TIME	T#20s0MS	FALSE	Not Public	
0013	Var13			TOD	TOD#00:00:20	FALSE	Not Public	
0014	Var14			UDINT	20000	FALSE	Not Public	
0015	Var15			UINT	20000	FALSE	Not Public	
0016	Var16			USINT	32	FALSE	Not Public	
0017	Var17			WORD	20000	FALSE	Not Public	


0001
DWORD_TO_BOOL
EN
ENO
Var1=20000
IN
OUT
Var2=TRUE
0002
DWORD_TO_BYTE
EN
ENO
Var1=20000
IN
OUT
Var3=32
0003
DWORD_TO_DATE
EN
ENO
Var1=20000
IN
OUT
Var4=D#1970-01-01
0004
DWORD_TO_DINT
EN
ENO
Var1=20000
IN
OUT
Var5=20000
0005
DWORD_TO_DT
EN
ENO
Var1=20000
IN
OUT
Var6=DT#1970-01-01-05:33:20
0006
DWORD_TO_INT
EN
ENO
Var1=20000
IN
OUT
Var7=20000
0007
DWORD_TO_LREAL
EN
ENO
Var1=20000
IN
OUT
Var8=20000
0008
DWORD_TO_REAL
EN
ENO
Var1=20000
IN
OUT
Var9=20000


0009
DWORD_TO_SINT
EN
ENO
Var1=20000
IN
OUT
Var10=32
0010
DWORD_TO_STRING
EN
ENO
Var1=20000
IN
OUT
Var11='20000'
0011
DWORD_TO_TIME
EN
ENO
Var1=20000
IN
OUT
Var12=T#20s0ms
0012
DWORD_TO_TOD
EN
ENO
Var1=20000
IN
OUT
Var13=TOD#00:00:20
0013
DWORD_TO_UDINT
EN
ENO
Var1=20000
IN
OUT
Var14=20000
0014
DWORD_TO_UINT
EN
ENO
Var1=20000
IN
OUT
Var15=20000
0015
DWORD_TO_USINT
EN
ENO
Var1=20000
IN
OUT
Var16=32
0016
DWORD_TO_WORD
EN
ENO
Var1=20000
IN
OUT
Var17=20000

```
1  Var2 := DWORD_TO_BOOL(Var1);
2  Var3 := DWORD_TO_BYTE(Var1);
3  Var4 := DWORD_TO_DATE(Var1);
4  Var5 := DWORD_TO_DINT(Var1);
5  Var6 := DWORD_TO_DT(Var1);
6  Var7 := DWORD_TO_INT(Var1);
7  Var8 := DWORD_TO_LREAL(Var1);
8  Var9 := DWORD_TO_REAL(Var1);
9  Var10 := DWORD_TO_SINT(Var1);
10 Var11 := DWORD_TO_STRING(Var1);
11 Var12 := DWORD_TO_TIME(Var1);
12 Var13 := DWORD_TO_TOD(Var1);
13 Var14 := DWORD_TO_UDINT(Var1);
14 Var15 := DWORD_TO_UINT(Var1);
15 Var16 := DWORD_TO_USINT(Var1);
16 Var17 := DWORD_TO_WORD(Var1);
```

```
Var2 = TRUE                    Var1 = 20000
Var3 = 32                      Var1 = 20000
Var4 = D#1970-01-01            Var1 = 20000
Var5 = 20000                   Var1 = 20000
Var6 = DT#1970-01-01-05:33:20  Var1 = 20000
Var7 = 20000                   Var1 = 20000
Var8 = 20000                   Var1 = 20000
Var9 = 20000                   Var1 = 20000
Var10 = 32                     Var1 = 20000
Var11 = '20000'                Var1 = 20000
Var12 = T#20s0ms               Var1 = 20000
Var13 = TOD#00:00:20           Var1 = 20000
Var14 = 20000                  Var1 = 20000
Var15 = 20000                  Var1 = 20000
Var16 = 32                     Var1 = 20000
Var17 = 20000                  Var1 = 20000
```

5.6.7 INT_TO_TYPE (Integer Type Conversion Instruction)

5.6.7.1 Function

Convert the integer data type to other data types.

In the case of INT_TO_BOOL, the output is TRUE when the input is not equal to 0, and FALSE when the input is equal to 0. In the case of INT_TO_TIME and INT_TO_TOD, the input will be converted in milliseconds.

In the case of INT_TO_DATE and INT_TO_DT, the input will be converted in seconds.

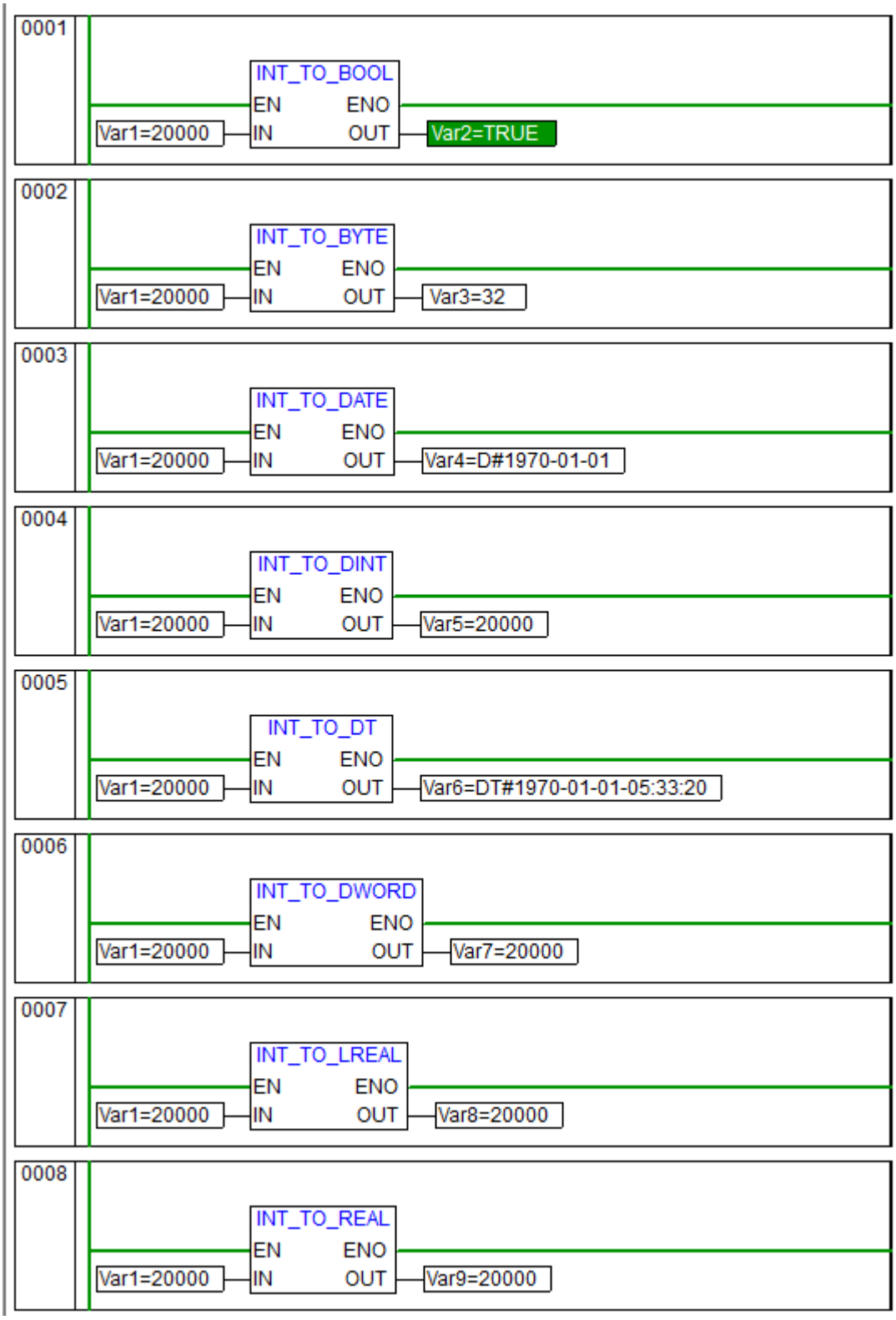
5.6.7.2 Parameter

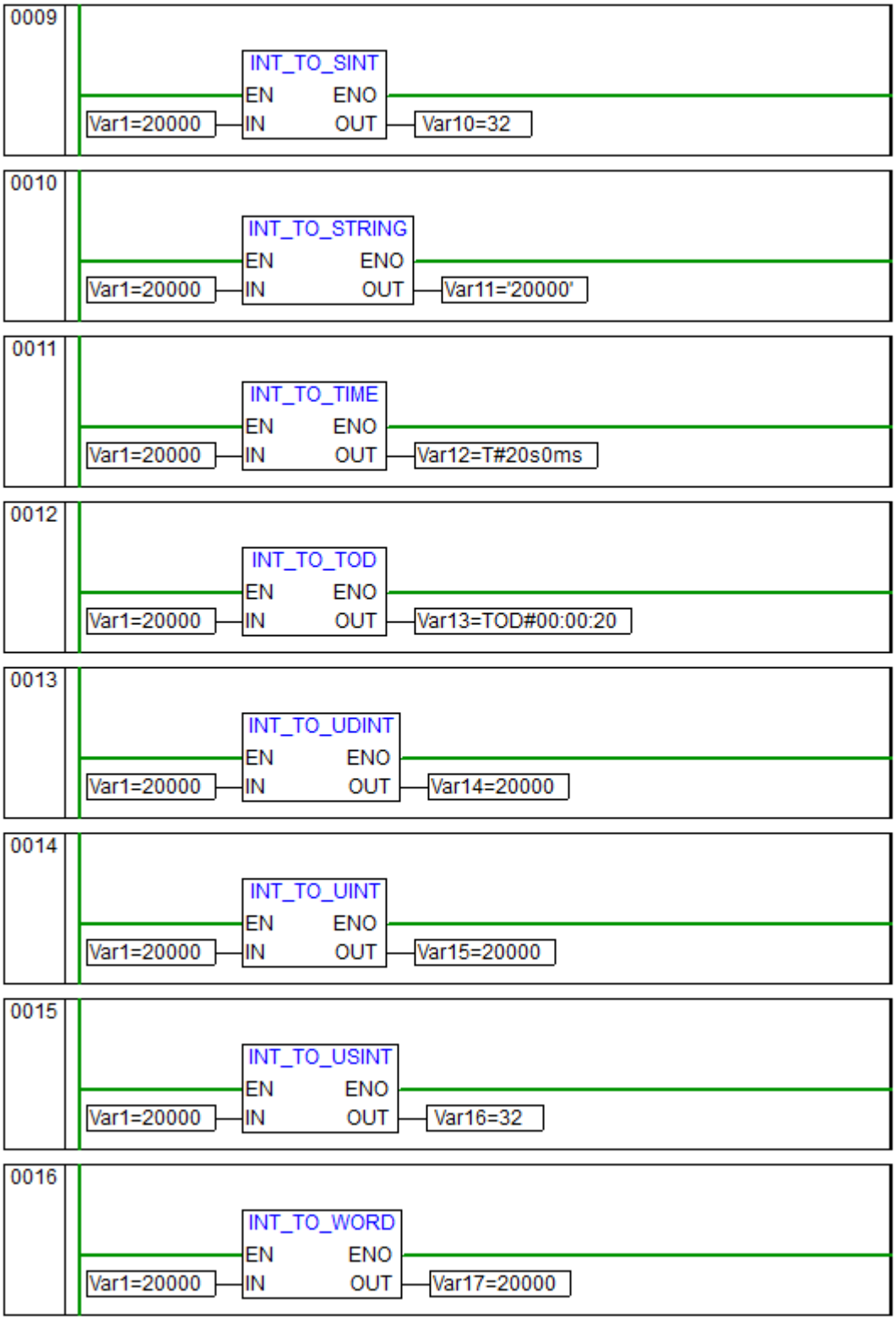
Parameter	Statement	Data Type	Description
IN	Input	INT	Enter the double-word variable to be converted
OUT	Output	BOOL, BYTE, DATE, DINT, DT, DWORD, LREAL, REAL, STRING, SINT, TIME, TOD, UDINT, UINT, USINT, WORD	Converted data type

5.6.7.3 Example

The following example uses the INT conversion instruction to convert INT variables to other data types. When the input EN detects a rising edge, the INT conversion instruction starts the conversion function and stores the calculation result in the output parameter OUT.

No.	Variable Name	Address	/ariable Descriptio	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			INT	20000	FALSE	Not Public	
0002	Var2			BOOL	TRUE	FALSE	Not Public	
0003	Var3			BYTE	32	FALSE	Not Public	
0004	Var4			DATE	D#1970-01-01	FALSE	Not Public	
0005	Var5			DINT	20000	FALSE	Not Public	
0006	Var6			DT	DT#1970-01-01...	FALSE	Not Public	
0007	Var7			DWORD	20000	FALSE	Not Public	
0008	Var8			LREAL	20000	FALSE	Not Public	
0009	Var9			REAL	20000	FALSE	Not Public	
0010	Var10			SINT	32	FALSE	Not Public	
0011	Var11			STRING(80)	'20000'	FALSE	Not Public	
0012	Var12			TIME	T#20s0ms	FALSE	Not Public	
0013	Var13			TOD	TOD#00:00:20	FALSE	Not Public	
0014	Var14			UDINT	20000	FALSE	Not Public	
0015	Var15			UINT	20000	FALSE	Not Public	
0016	Var16			USINT	32	FALSE	Not Public	
0017	Var17			WORD	20000	FALSE	Not Public	


0001
INT_TO_BOOL
EN
ENO
Var1=20000
IN
OUT
Var2=TRUE
0002
INT_TO_BYTE
EN
ENO
Var1=20000
IN
OUT
Var3=32
0003
INT_TO_DATE
EN
ENO
Var1=20000
IN
OUT
Var4=D#1970-01-01
0004
INT_TO_DINT
EN
ENO
Var1=20000
IN
OUT
Var5=20000
0005
INT_TO_DT
EN
ENO
Var1=20000
IN
OUT
Var6=DT#1970-01-01-05:33:20
0006
INT_TO_DWORD
EN
ENO
Var1=20000
IN
OUT
Var7=20000
0007
INT_TO_LREAL
EN
ENO
Var1=20000
IN
OUT
Var8=20000
0008
INT_TO_REAL
EN
ENO
Var1=20000
IN
OUT
Var9=20000


0009
INT_TO_SINT
EN
ENO
Var1=20000
IN
OUT
Var10=32
0010
INT_TO_STRING
EN
ENO
Var1=20000
IN
OUT
Var11='20000'
0011
INT_TO_TIME
EN
ENO
Var1=20000
IN
OUT
Var12=T#20s0ms
0012
INT_TO_TOD
EN
ENO
Var1=20000
IN
OUT
Var13=TOD#00:00:20
0013
INT_TO_UDINT
EN
ENO
Var1=20000
IN
OUT
Var14=20000
0014
INT_TO_UINT
EN
ENO
Var1=20000
IN
OUT
Var15=20000
0015
INT_TO_USINT
EN
ENO
Var1=20000
IN
OUT
Var16=32
0016
INT_TO_WORD
EN
ENO
Var1=20000
IN
OUT
Var17=20000

```
1   Var2 := INT_TO_BOOL(Var1);
2   Var3 := INT_TO_BYTE(Var1);
3   Var4 := INT_TO_DATE(Var1);
4   Var5 := INT_TO_DINT(Var1);
5   Var6 := INT_TO_DT(Var1);
6   Var7 := INT_TO_DWORD(Var1);
7   Var8 := INT_TO_LREAL(Var1);
8   Var9 := INT_TO_REAL(Var1);
9   Var10 := INT_TO_SINT(Var1);
10  Var11 := INT_TO_STRING(Var1);
11  Var12 := INT_TO_TIME(Var1);
12  Var13 := INT_TO_TOD(Var1);
13  Var14 := INT_TO_UDINT(Var1);
14  Var15 := INT_TO_UINT(Var1);
15  Var16 := INT_TO_USINT(Var1);
16  Var17 := INT_TO_WORD(Var1);
```

```
Var2 = TRUE     Var1 = 20000
Var3 = 32       Var1 = 20000
Var4 = D#1970-01-01     Var1 = 20000
Var5 = 20000    Var1 = 20000
Var6 = DT#1970-01-01-05:33:20   Var1 = 20000
Var7 = 20000    Var1 = 20000
Var8 = 20000    Var1 = 20000
Var9 = 20000    Var1 = 20000
Var10 = 32      Var1 = 20000
Var11 = '20000'     Var1 = 20000
Var12 = T#20s0ms    Var1 = 20000
Var13 = TOD#00:00:20    Var1 = 20000
Var14 = 20000   Var1 = 20000
Var15 = 20000   Var1 = 20000
Var16 = 32      Var1 = 20000
Var17 = 20000   Var1 = 20000
```

5.6.8 LREAL_TO_TYPE (Long Real Number Type Conversion Instruction)

5.6.8.1 Function

Convert the long real number to other data types.

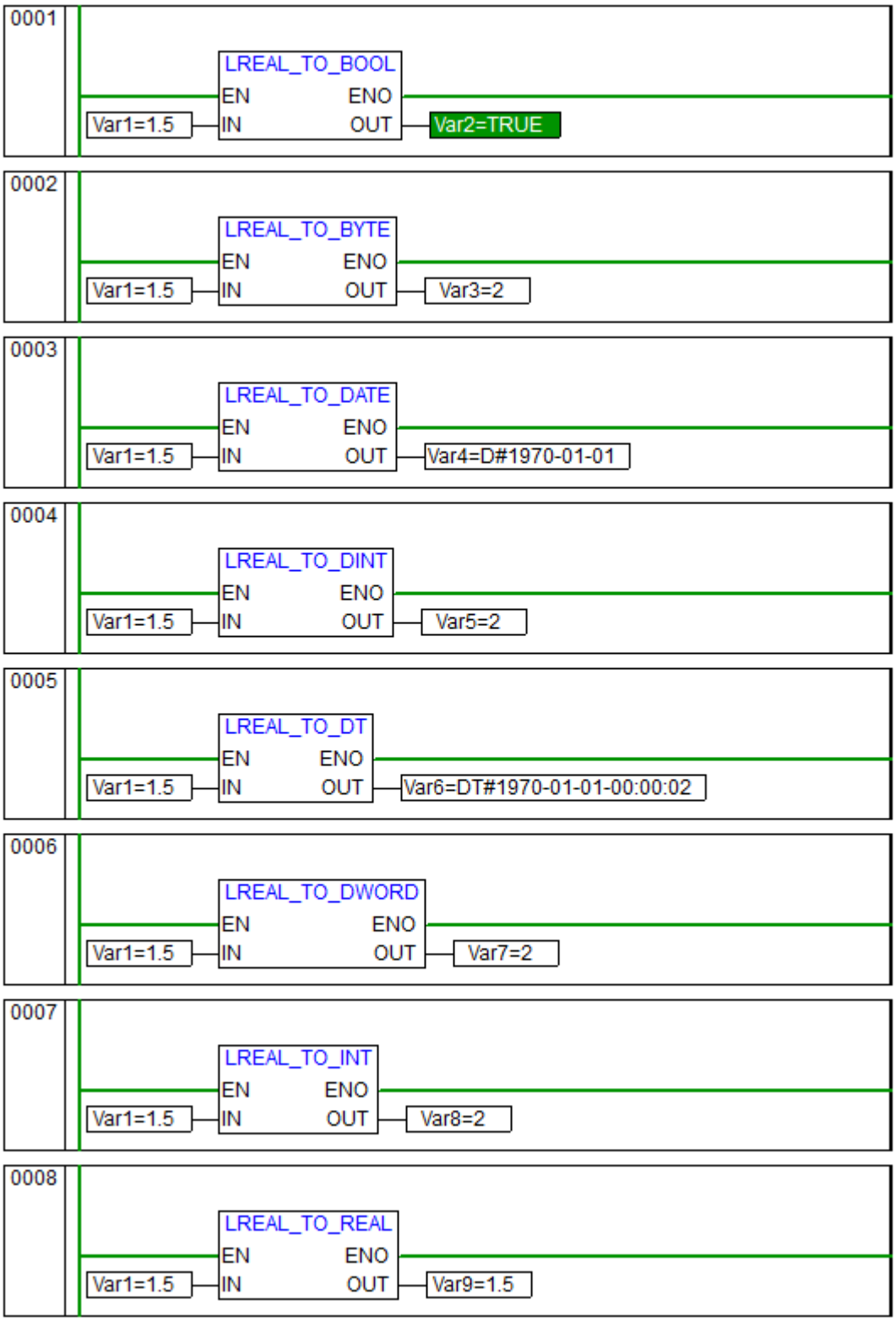
5.6.8.2 Parameter

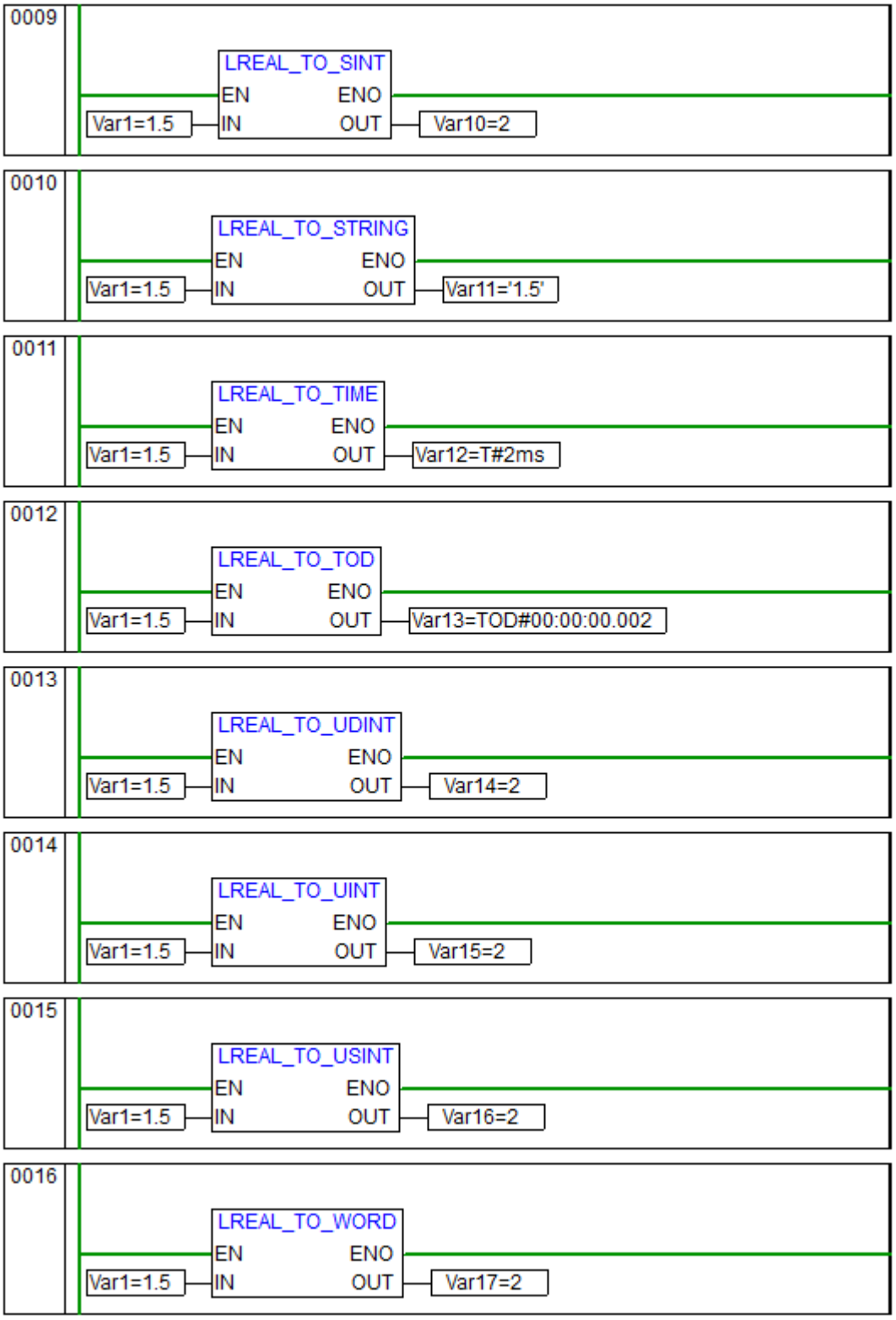
Parameter	Statement	Data Type	Description
IN	Input	LREAL	Enter the long real number variable to be converted
OUT	Output	BOOL, BYTE, DATE, DINT, DT, DWORD, INT, REAL, STRING, SINT, TIME, TOD, UDINT, UINT, USINT, WORD	Converted data type

5.6.8.3 Example

The following example uses the LREAL conversion instruction to convert LREAL variables to other data types. When the input EN detects a rising edge, the LREAL conversion instruction starts the conversion function and stores the calculation result in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			LREAL	1.5	FALSE	Not Public	
0002	Var2			BOOL	TRUE	FALSE	Not Public	
0003	Var3			BYTE	2	FALSE	Not Public	
0004	Var4			DATE	D#1970-01-01	FALSE	Not Public	
0005	Var5			DINT	2	FALSE	Not Public	
0006	Var6			DT	DT#1970-01-01...	FALSE	Not Public	
0007	Var7			DWORD	2	FALSE	Not Public	
0008	Var8			LREAL	2	FALSE	Not Public	
0009	Var9			REAL	1.5	FALSE	Not Public	
0010	Var10			SINT	2	FALSE	Not Public	
0011	Var11			STRING(80)	'1.5'	FALSE	Not Public	
0012	Var12			TIME	T#2MS	FALSE	Not Public	
0013	Var13			TOD	TOD#00:00:00....	FALSE	Not Public	
0014	Var14			UDINT	2	FALSE	Not Public	
0015	Var15			UINT	2	FALSE	Not Public	
0016	Var16			USINT	2	FALSE	Not Public	
0017	Var17			WORD	2	FALSE	Not Public	


0001
LREAL_TO_BOOL
EN
ENO
Var1=1.5
IN
OUT
Var2=TRUE
0002
LREAL_TO_BYTE
EN
ENO
Var1=1.5
IN
OUT
Var3=2
0003
LREAL_TO_DATE
EN
ENO
Var1=1.5
IN
OUT
Var4=D#1970-01-01
0004
LREAL_TO_DINT
EN
ENO
Var1=1.5
IN
OUT
Var5=2
0005
LREAL_TO_DT
EN
ENO
Var1=1.5
IN
OUT
Var6=DT#1970-01-01-00:00:02
0006
LREAL_TO_DWORD
EN
ENO
Var1=1.5
IN
OUT
Var7=2
0007
LREAL_TO_INT
EN
ENO
Var1=1.5
IN
OUT
Var8=2
0008
LREAL_TO_REAL
EN
ENO
Var1=1.5
IN
OUT
Var9=1.5


0009
LREAL_TO_SINT
EN
ENO
Var1=1.5
IN
OUT
Var10=2
0010
LREAL_TO_STRING
EN
ENO
Var1=1.5
IN
OUT
Var11='1.5'
0011
LREAL_TO_TIME
EN
ENO
Var1=1.5
IN
OUT
Var12=T#2ms
0012
LREAL_TO_TOD
EN
ENO
Var1=1.5
IN
OUT
Var13=TOD#00:00:00.002
0013
LREAL_TO_UDINT
EN
ENO
Var1=1.5
IN
OUT
Var14=2
0014
LREAL_TO_UINT
EN
ENO
Var1=1.5
IN
OUT
Var15=2
0015
LREAL_TO_USINT
EN
ENO
Var1=1.5
IN
OUT
Var16=2
0016
LREAL_TO_WORD
EN
ENO
Var1=1.5
IN
OUT
Var17=2

```
1   Var2 := LREAL_TO_BOOL(Var1);
2   Var3 := LREAL_TO_BYTE(Var1);
3   Var4 := LREAL_TO_DATE(Var1);
4   Var5 := LREAL_TO_DINT(Var1);
5   Var6 := LREAL_TO_DT(Var1);
6   Var7 := LREAL_TO_DWORD(Var1);
7   Var8 := LREAL_TO_INT(Var1);
8   Var9 := LREAL_TO_REAL(Var1);
9   Var10 := LREAL_TO_SINT(Var1);
10  Var11 := LREAL_TO_STRING(Var1);
11  Var12 := LREAL_TO_TIME(Var1);
12  Var13 := LREAL_TO_TOD(Var1);
13  Var14 := LREAL_TO_UDINT(Var1);
14  Var15 := LREAL_TO_UINT(Var1);
15  Var16 := LREAL_TO_USINT(Var1);
16  Var17 := LREAL_TO_WORD(Var1);
```

```
Var2 = TRUE                    Var1 = 1.5
Var3 = 2                       Var1 = 1.5
Var4 = D#1970-01-01            Var1 = 1.5
Var5 = 2                       Var1 = 1.5
Var6 = DT#1970-01-01-00:00:02  Var1 = 1.5
Var7 = 2                       Var1 = 1.5
Var8 = 2                       Var1 = 1.5
Var9 = 1.5                     Var1 = 1.5
Var10 = 2                      Var1 = 1.5
Var11 = '1.5'                  Var1 = 1.5
Var12 = T#2ms                  Var1 = 1.5
Var13 = TOD#00:00:00.002       Var1 = 1.5
Var14 = 2                      Var1 = 1.5
Var15 = 2                      Var1 = 1.5
Var16 = 2                      Var1 = 1.5
Var17 = 2                      Var1 = 1.5
```

5.6.9 REAL_TO_TYPE (Real Number Type Conversion Instruction)

5.6.9.1 Function

Convert the real number to other data types.

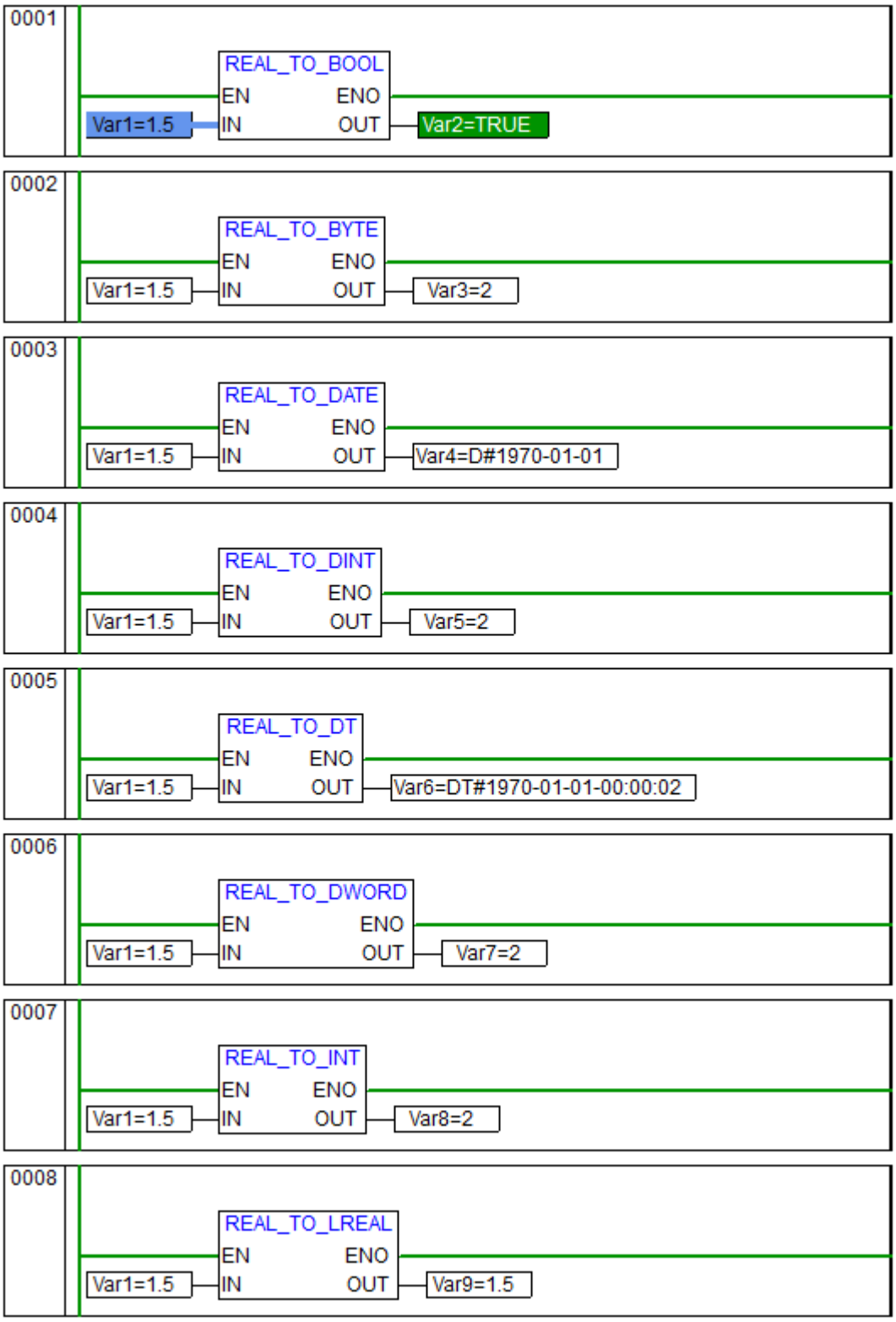
5.6.9.2 Parameter

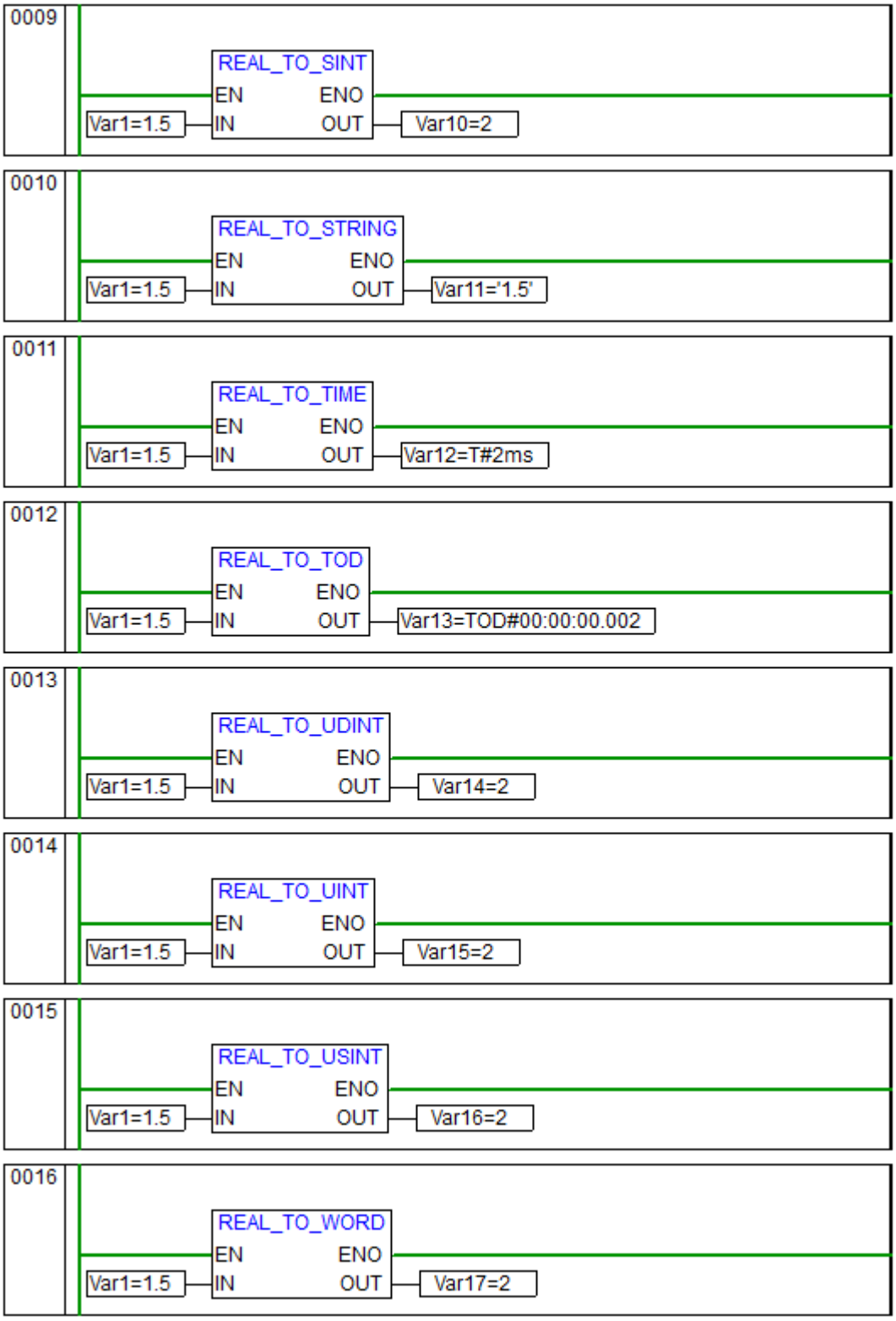
Parameter	Statement	Data Type	Description
IN	Input	REAL	Enter the real number variable to be converted
OUT	Output	BOOL, BYTE, DATE, DINT, DT, DWORD, INT, LREAL, STRING, SINT, TIME, TOD, UDINT, UINT, USINT, WORD	Converted data type

5.6.9.3 Example

The following example uses the REAL conversion instruction to convert REAL variables to other data types. When the input EN detects a rising edge, the REAL conversion instruction starts the conversion function and stores the calculation result in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			REAL	1.5	FALSE	Not Public	
0002	Var2			BOOL	TRUE	FALSE	Not Public	
0003	Var3			BYTE	2	FALSE	Not Public	
0004	Var4			DATE	D#1970-01-01	FALSE	Not Public	
0005	Var5			DINT	2	FALSE	Not Public	
0006	Var6			DT	DT#1970-01-01...	FALSE	Not Public	
0007	Var7			DWORD	2	FALSE	Not Public	
0008	Var8			INT	2	FALSE	Not Public	
0009	Var9			LREAL	1.5	FALSE	Not Public	
0010	Var10			SINT	2	FALSE	Not Public	
0011	Var11			STRING(80)	'1.5'	FALSE	Not Public	
0012	Var12			TIME	T#2MS	FALSE	Not Public	
0013	Var13			TOD	TOD#00:00:002	FALSE	Not Public	
0014	Var14			UDINT	2	FALSE	Not Public	
0015	Var15			UINT	2	FALSE	Not Public	
0016	Var16			USINT	2	FALSE	Not Public	
0017	Var17			WORD	2	FALSE	Not Public	


0001
REAL_TO_BOOL
EN
ENO
Var1=1.5
IN
OUT
Var2=TRUE
0002
REAL_TO_BYTE
EN
ENO
Var1=1.5
IN
OUT
Var3=2
0003
REAL_TO_DATE
EN
ENO
Var1=1.5
IN
OUT
Var4=D#1970-01-01
0004
REAL_TO_DINT
EN
ENO
Var1=1.5
IN
OUT
Var5=2
0005
REAL_TO_DT
EN
ENO
Var1=1.5
IN
OUT
Var6=DT#1970-01-01-00:00:02
0006
REAL_TO_DWORD
EN
ENO
Var1=1.5
IN
OUT
Var7=2
0007
REAL_TO_INT
EN
ENO
Var1=1.5
IN
OUT
Var8=2
0008
REAL_TO_LREAL
EN
ENO
Var1=1.5
IN
OUT
Var9=1.5


0009
REAL_TO_SINT
EN
ENO
Var1=1.5
IN
OUT
Var10=2
0010
REAL_TO_STRING
EN
ENO
Var1=1.5
IN
OUT
Var11='1.5'
0011
REAL_TO_TIME
EN
ENO
Var1=1.5
IN
OUT
Var12=T#2ms
0012
REAL_TO_TOD
EN
ENO
Var1=1.5
IN
OUT
Var13=TOD#00:00:00.002
0013
REAL_TO_UDINT
EN
ENO
Var1=1.5
IN
OUT
Var14=2
0014
REAL_TO_UINT
EN
ENO
Var1=1.5
IN
OUT
Var15=2
0015
REAL_TO_USINT
EN
ENO
Var1=1.5
IN
OUT
Var16=2
0016
REAL_TO_WORD
EN
ENO
Var1=1.5
IN
OUT
Var17=2

```
1  Var2 := REAL_TO_BOOL(Var1);
2  Var3 := REAL_TO_BYTE(Var1);
3  Var4 := REAL_TO_DATE(Var1);
4  Var5 := REAL_TO_DINT(Var1);
5  Var6 := REAL_TO_DT(Var1);
6  Var7 := REAL_TO_DWORD(Var1);
7  Var8 := REAL_TO_INT(Var1);
8  Var9 := REAL_TO_LREAL(Var1);
9  Var10 := REAL_TO_SINT(Var1);
10 Var11 := REAL_TO_STRING(Var1);
11 Var12 := REAL_TO_TIME(Var1);
12 Var13 := REAL_TO_TOD(Var1);
13 Var14 := REAL_TO_UDINT(Var1);
14 Var15 := REAL_TO_UINT(Var1);
15 Var16 := REAL_TO_USINT(Var1);
16 Var17 := REAL_TO_WORD(Var1);
```

```
Var2 = TRUE                    Var1 = 1.5
Var3 = 2                       Var1 = 1.5
Var4 = D#1970-01-01            Var1 = 1.5
Var5 = 2                       Var1 = 1.5
Var6 = DT#1970-01-01-00:00:02  Var1 = 1.5
Var7 = 2                       Var1 = 1.5
Var8 = 2                       Var1 = 1.5
Var9 = 1.5                     Var1 = 1.5
Var10 = 2                      Var1 = 1.5
Var11 = '1.5'                  Var1 = 1.5
Var12 = T#2ms                  Var1 = 1.5
Var13 = TOD#00:00:00.002       Var1 = 1.5
Var14 = 2                      Var1 = 1.5
Var15 = 2                      Var1 = 1.5
Var16 = 2                      Var1 = 1.5
Var17 = 2                      Var1 = 1.5
```

5.6.10 SINT_TO_TYPE (Short Integer Type Conversion Instruction)

5.6.10.1 Function

Convert the short integer type to other data types.

In the case of SINT_TO_BOOL, the output is TRUE when the input is not equal to 0, and FALSE when the input is equal to 0.

In the case of SINT_TO_TIME and SINT_TO_TOD, the input will be converted in milliseconds.

In the case of SINT_TO_DATE and SINT_TO_DT, the input will be converted in seconds.

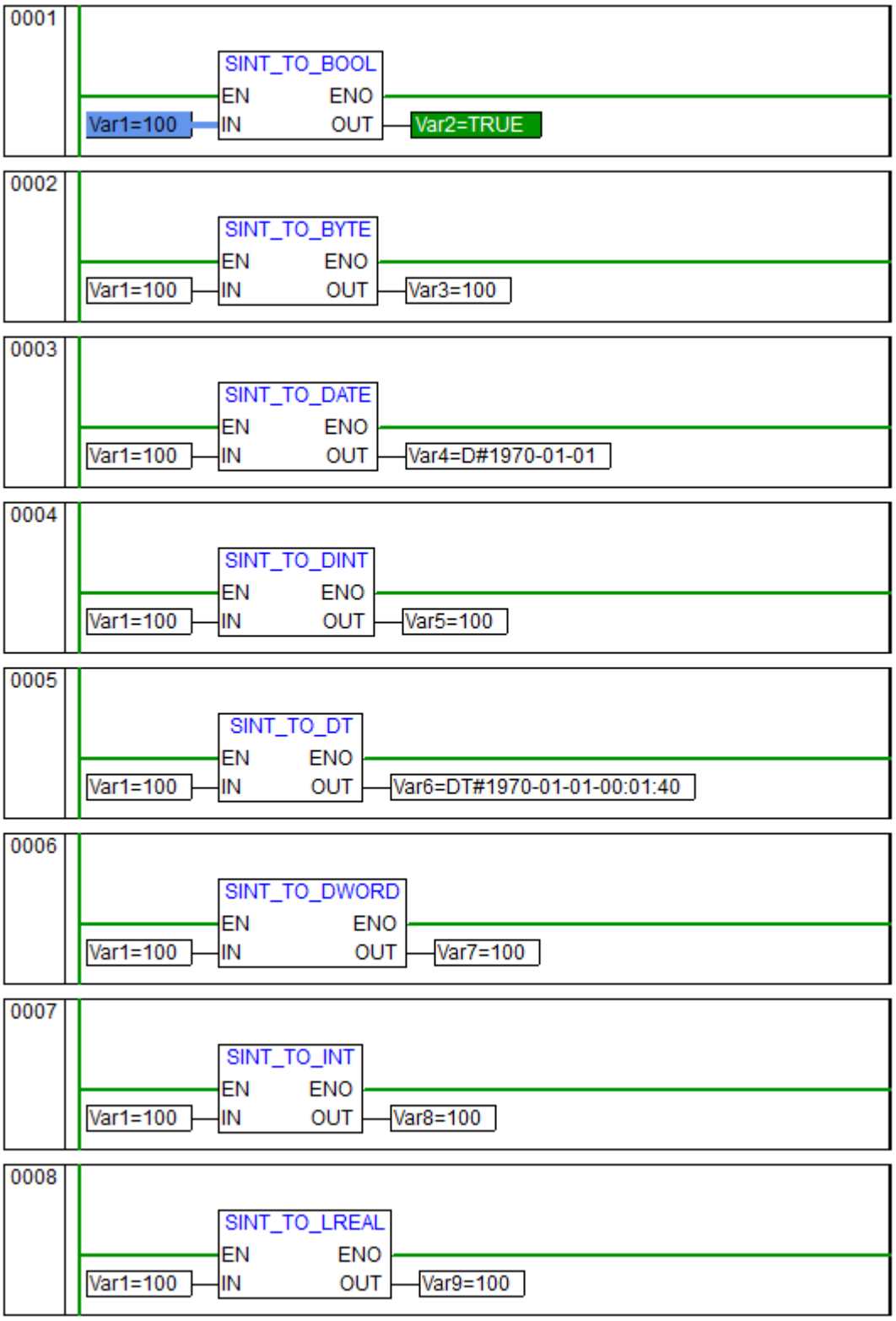
5.6.10.2 Parameter

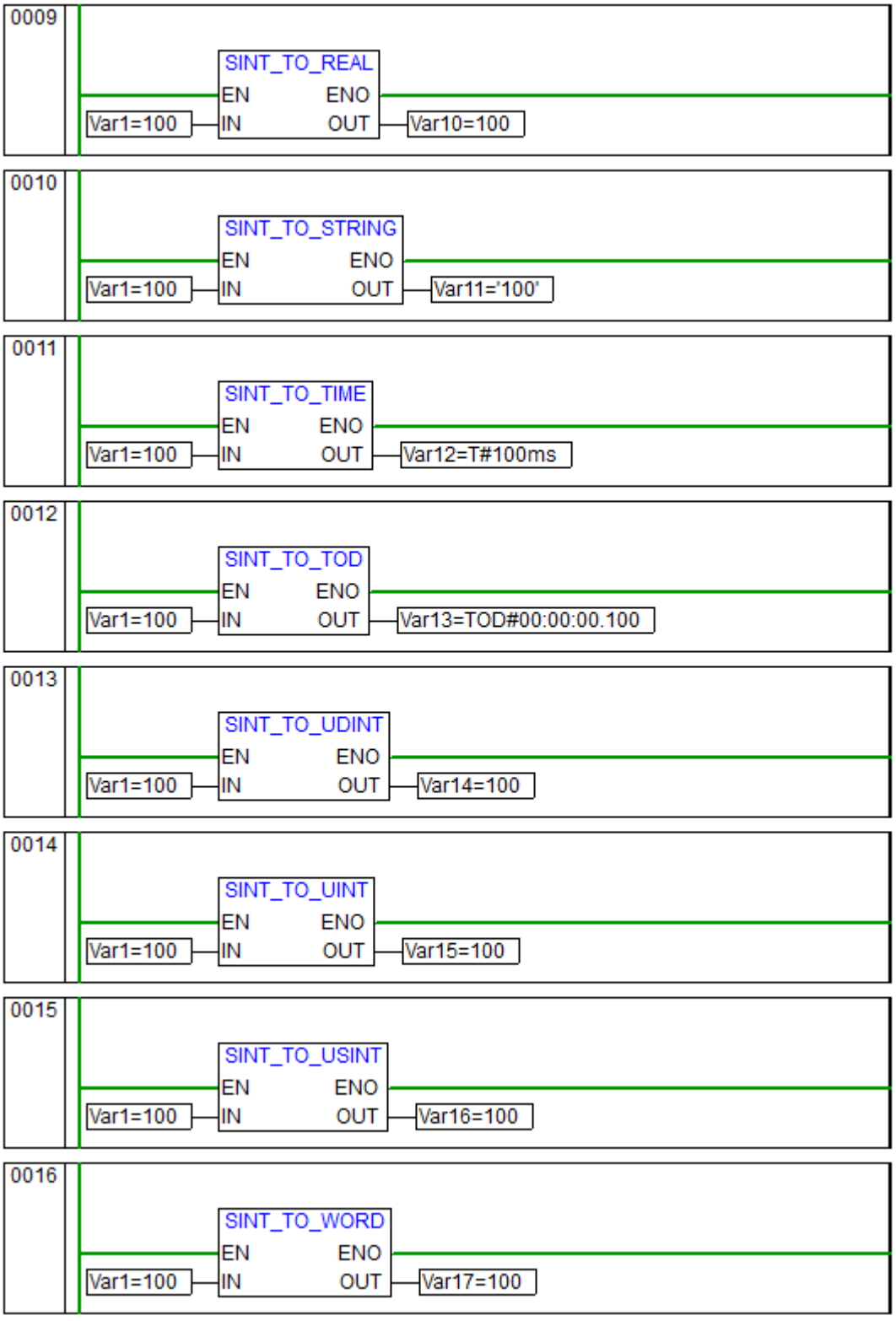
Parameter	Statement	Data Type	Description
IN	Input	SINT	Enter the short integer variable to be converted
OUT	Output	BOOL, BYTE, DATE, DINT, DT, DWORD, INT, LREAL, STRING, REAL, TIME, TOD, UDINT, UINT, USINT, WORD	Converted data type

5.6.10.3 Example

The following example uses the SINT conversion instruction to convert SINT variables to other data types. When the input EN detects a rising edge, the SINT conversion instruction starts the conversion function and stores the calculation result in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			SINT	100	FALSE	Not Public	
0002	Var2			BOOL	TRUE	FALSE	Not Public	
0003	Var3			BYTE	100	FALSE	Not Public	
0004	Var4			DATE	D#1970-01-01	FALSE	Not Public	
0005	Var5			DINT	100	FALSE	Not Public	
0006	Var6			DT	DT#1970-01-...	FALSE	Not Public	
0007	Var7			DWORD	100	FALSE	Not Public	
0008	Var8			INT	100	FALSE	Not Public	
0009	Var9			LREAL	100	FALSE	Not Public	
0010	Var10			REAL	100	FALSE	Not Public	
0011	Var11			STRING(80)	'100'	FALSE	Not Public	
0012	Var12			TIME	T#100MS	FALSE	Not Public	
0013	Var13			TOD	TOD#00:00:0...	FALSE	Not Public	
0014	Var14			UDINT	100	FALSE	Not Public	
0015	Var15			UINT	100	FALSE	Not Public	
0016	Var16			USINT	100	FALSE	Not Public	
0017	Var17			WORD	100	FALSE	Not Public	


0001
SINT_TO_BOOL
EN
ENO
Var1=100
IN
OUT
Var2=TRUE
0002
SINT_TO_BYTE
EN
ENO
Var1=100
IN
OUT
Var3=100
0003
SINT_TO_DATE
EN
ENO
Var1=100
IN
OUT
Var4=D#1970-01-01
0004
SINT_TO_DINT
EN
ENO
Var1=100
IN
OUT
Var5=100
0005
SINT_TO_DT
EN
ENO
Var1=100
IN
OUT
Var6=DT#1970-01-01-00:01:40
0006
SINT_TO_DWORD
EN
ENO
Var1=100
IN
OUT
Var7=100
0007
SINT_TO_INT
EN
ENO
Var1=100
IN
OUT
Var8=100
0008
SINT_TO_LREAL
EN
ENO
Var1=100
IN
OUT
Var9=100


0009
SINT_TO_REAL
EN
ENO
Var1=100
IN
OUT
Var10=100
0010
SINT_TO_STRING
EN
ENO
Var1=100
IN
OUT
Var11='100'
0011
SINT_TO_TIME
EN
ENO
Var1=100
IN
OUT
Var12=T#100ms
0012
SINT_TO_TOD
EN
ENO
Var1=100
IN
OUT
Var13=TOD#00:00:00.100
0013
SINT_TO_UDINT
EN
ENO
Var1=100
IN
OUT
Var14=100
0014
SINT_TO_UINT
EN
ENO
Var1=100
IN
OUT
Var15=100
0015
SINT_TO_USINT
EN
ENO
Var1=100
IN
OUT
Var16=100
0016
SINT_TO_WORD
EN
ENO
Var1=100
IN
OUT
Var17=100

```
1   Var2 := SINT_TO_BOOL(Var1);
2   Var3 := SINT_TO_BYTE(Var1);
3   Var4 := SINT_TO_DATE(Var1);
4   Var5 := SINT_TO_DINT(Var1);
5   Var6 := SINT_TO_DT(Var1);
6   Var7 := SINT_TO_DWORD(Var1);
7   Var8 := SINT_TO_INT(Var1);
8   Var9 := SINT_TO_LREAL(Var1);
9   Var10 := SINT_TO_REAL(Var1);
10  Var11 := SINT_TO_STRING(Var1);
11  Var12 := SINT_TO_TIME(Var1);
12  Var13 := SINT_TO_TOD(Var1);
13  Var14 := SINT_TO_UDINT(Var1);
14  Var15 := SINT_TO_UINT(Var1);
15  Var16 := SINT_TO_USINT(Var1);
16  Var17 := SINT_TO_WORD(Var1);
```

```
Var2 = TRUE                    Var1 = 100
Var3 = 100                     Var1 = 100
Var4 = D#1970-01-01            Var1 = 100
Var5 = 100                     Var1 = 100
Var6 = DT#1970-01-01-00:01:40  Var1 = 100
Var7 = 100                     Var1 = 100
Var8 = 100                     Var1 = 100
Var9 = 100                     Var1 = 100
Var10 = 100                    Var1 = 100
Var11 = '100'                  Var1 = 100
Var12 = T#100ms                Var1 = 100
Var13 = TOD#00:00:00.100       Var1 = 100
Var14 = 100                    Var1 = 100
Var15 = 100                    Var1 = 100
Var16 = 100                    Var1 = 100
Var17 = 100                    Var1 = 100
```

5.6.11 STRING_TO_TYPE (String Type Conversion Instruction)

5.6.11.1 Function

Convert variables of a STRING type to another data type.

For the STRING_TO_BOOL instruction, if the string 'TRUE' or 'true' is input, TRUE is output; otherwise, FALSE is output.

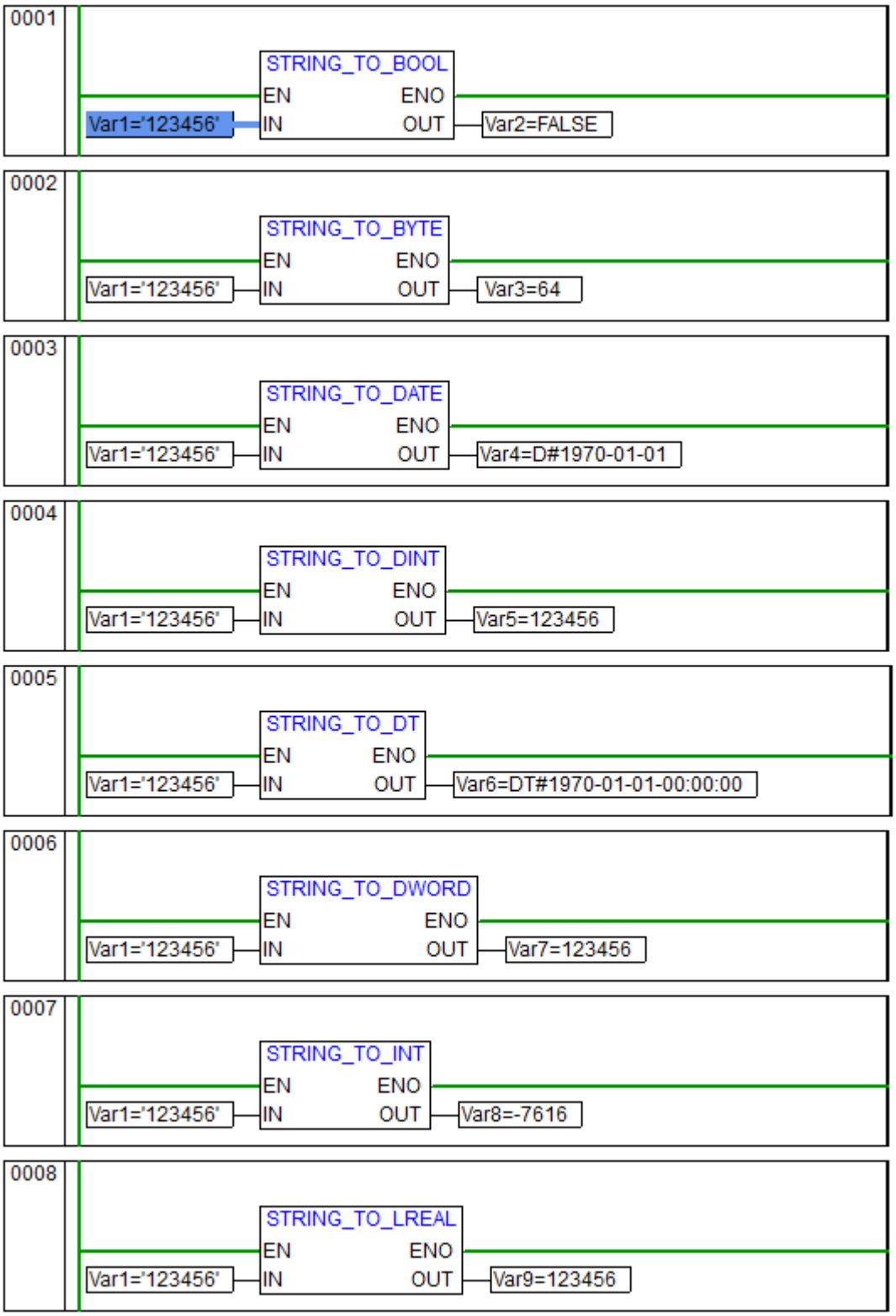
5.6.11.2 Parameter

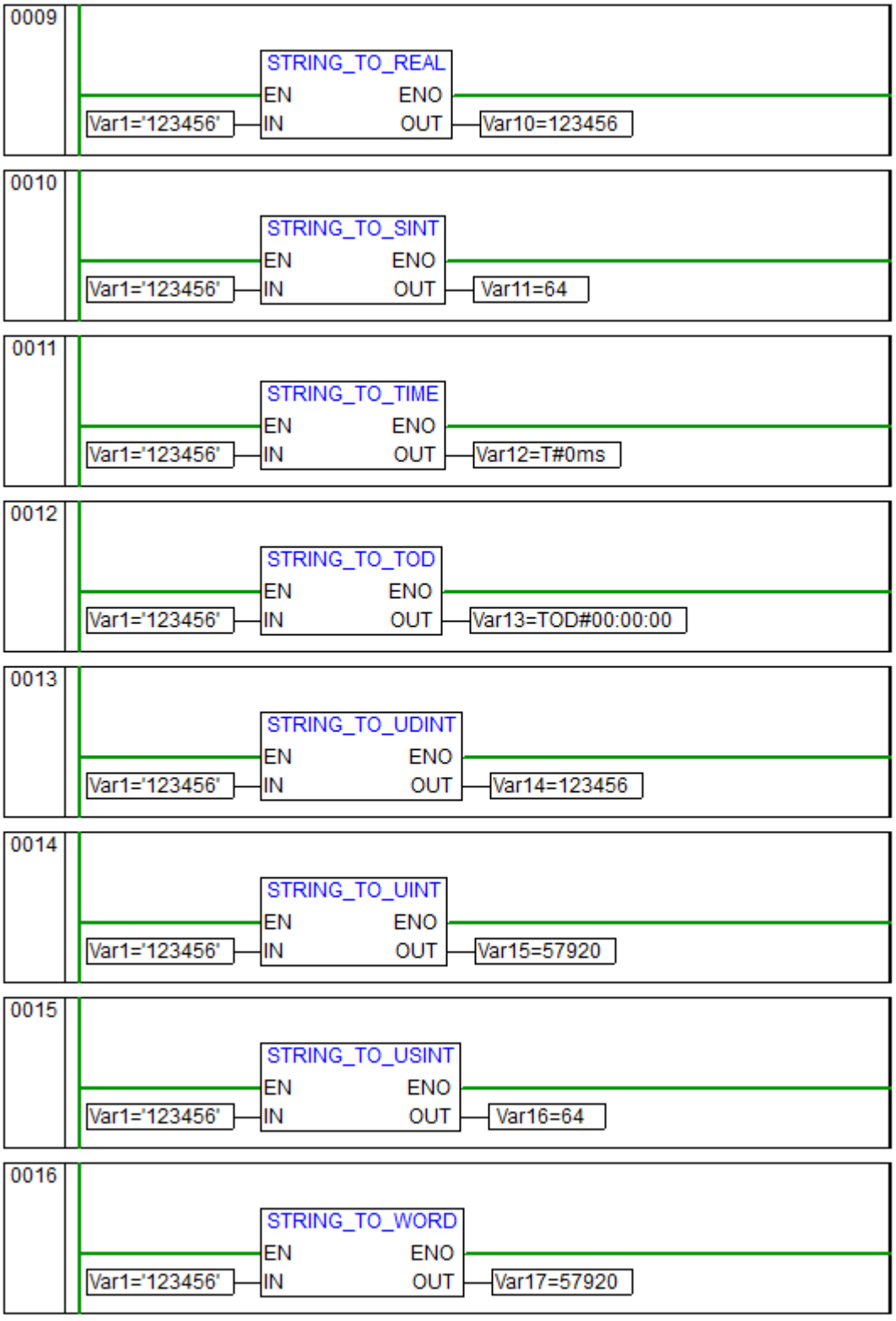
Parameter	Statement	Data Type	Description
IN	Input	STRING	Enter the string variable to be converted
OUT	Output	BOOL, BYTE, DATE, DINT, DT, DWORD, INT, LREAL, REAL, SINT, TIME, TOD, UDINT, UINT, USINT, WORD	Converted data type

5.6.11.3 Example

The following example uses the STRING conversion instruction to convert STRING variables to other data types. When the input EN detects a rising edge, the STRING conversion instruction starts the conversion function and stores the calculation result in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard
0001	Var1			STRING(80)	'123456'	FALSE
0002	Var2			BOOL	FALSE	FALSE
0003	Var3			BYTE	64	FALSE
0004	Var4			DATE	D#1970-01-01	FALSE
0005	Var5			DINT	100	FALSE
0006	Var6			DT	DT#1970-01-01...	FALSE
0007	Var7			DWORD	123456	FALSE
0008	Var8			INT	7616	FALSE
0009	Var9			LREAL	123456	FALSE
0010	Var10			REAL	123456	FALSE
0011	Var11			SINT	64	FALSE
0012	Var12			TIME	T#0MS	FALSE
0013	Var13			TOD	TOD#00:00:00....	FALSE
0014	Var14			UDINT	123456	FALSE
0015	Var15			UINT	57920	FALSE
0016	Var16			USINT	64	FALSE
0017	Var17			WORD	57920	FALSE


0001
STRING_TO_BOOL
EN
ENO
Var1='123456'
IN
OUT
Var2=FALSE
0002
STRING_TO_BYTE
EN
ENO
Var1='123456'
IN
OUT
Var3=64
0003
STRING_TO_DATE
EN
ENO
Var1='123456'
IN
OUT
Var4=D#1970-01-01
0004
STRING_TO_DINT
EN
ENO
Var1='123456'
IN
OUT
Var5=123456
0005
STRING_TO_DT
EN
ENO
Var1='123456'
IN
OUT
Var6=DT#1970-01-01-00:00:00
0006
STRING_TO_DWORD
EN
ENO
Var1='123456'
IN
OUT
Var7=123456
0007
STRING_TO_INT
EN
ENO
Var1='123456'
IN
OUT
Var8=-7616
0008
STRING_TO_LREAL
EN
ENO
Var1='123456'
IN
OUT
Var9=123456


0009
STRING_TO_REAL
EN
ENO
Var1='123456'
IN
OUT
Var10=123456
0010
STRING_TO_SINT
EN
ENO
Var1='123456'
IN
OUT
Var11=64
0011
STRING_TO_TIME
EN
ENO
Var1='123456'
IN
OUT
Var12=T#0ms
0012
STRING_TO_TOD
EN
ENO
Var1='123456'
IN
OUT
Var13=TOD#00:00:00
0013
STRING_TO_UDINT
EN
ENO
Var1='123456'
IN
OUT
Var14=123456
0014
STRING_TO_UINT
EN
ENO
Var1='123456'
IN
OUT
Var15=57920
0015
STRING_TO_USINT
EN
ENO
Var1='123456'
IN
OUT
Var16=64
0016
STRING_TO_WORD
EN
ENO
Var1='123456'
IN
OUT
Var17=57920

```
1   Var2 := STRING_TO_BOOL(Var1);
2   Var3 := STRING_TO_BYTE(Var1);
3   Var4 := STRING_TO_DATE(Var1);
4   Var5 := STRING_TO_DINT(Var1);
5   Var6 := STRING_TO_DT(Var1);
6   Var7 := STRING_TO_DWORD(Var1);
7   Var8 := STRING_TO_INT(Var1);
8   Var9 := STRING_TO_LREAL(Var1);
9   Var10 := STRING_TO_REAL(Var1);
10  Var11 := STRING_TO_SINT(Var1);
11  Var12 := STRING_TO_TIME(Var1);
12  Var13 := STRING_TO_TOD(Var1);
13  Var14 := STRING_TO_UDINT(Var1);
14  Var15 := STRING_TO_UINT(Var1);
15  Var16 := STRING_TO_USINT(Var1);
16  Var17 := STRING_TO_WORD(Var1);
```

```
Var2 = FALSE                    Var1 = '123456'
Var3 = 64                       Var1 = '123456'
Var4 = D#1970-01-01             Var1 = '123456'
Var5 = 123456                   Var1 = '123456'
Var6 = DT#1970-01-01-00:00:00   Var1 = '123456'
Var7 = 123456                   Var1 = '123456'
Var8 = -7616                    Var1 = '123456'
Var9 = 123456                   Var1 = '123456'
Var10 = -1.#IND                 Var1 = '123456'
Var11 = 64                      Var1 = '123456'
Var12 = T#0ms                   Var1 = '123456'
Var13 = TOD#00:00:00            Var1 = '123456'
Var14 = 123456                  Var1 = '123456'
Var15 = 57920                   Var1 = '123456'
Var16 = 64                      Var1 = '123456'
Var17 = 57920                   Var1 = '123456'
```

5.6.12 TIME_TO_TYPE (Time Type Conversion Instruction)

5.6.12.1 Function

It is to convert time data to other types of data. The time is stored internally in milliseconds as a DWORD type (for the TIME_OF_DAY variable, it starts at 00:00 am).

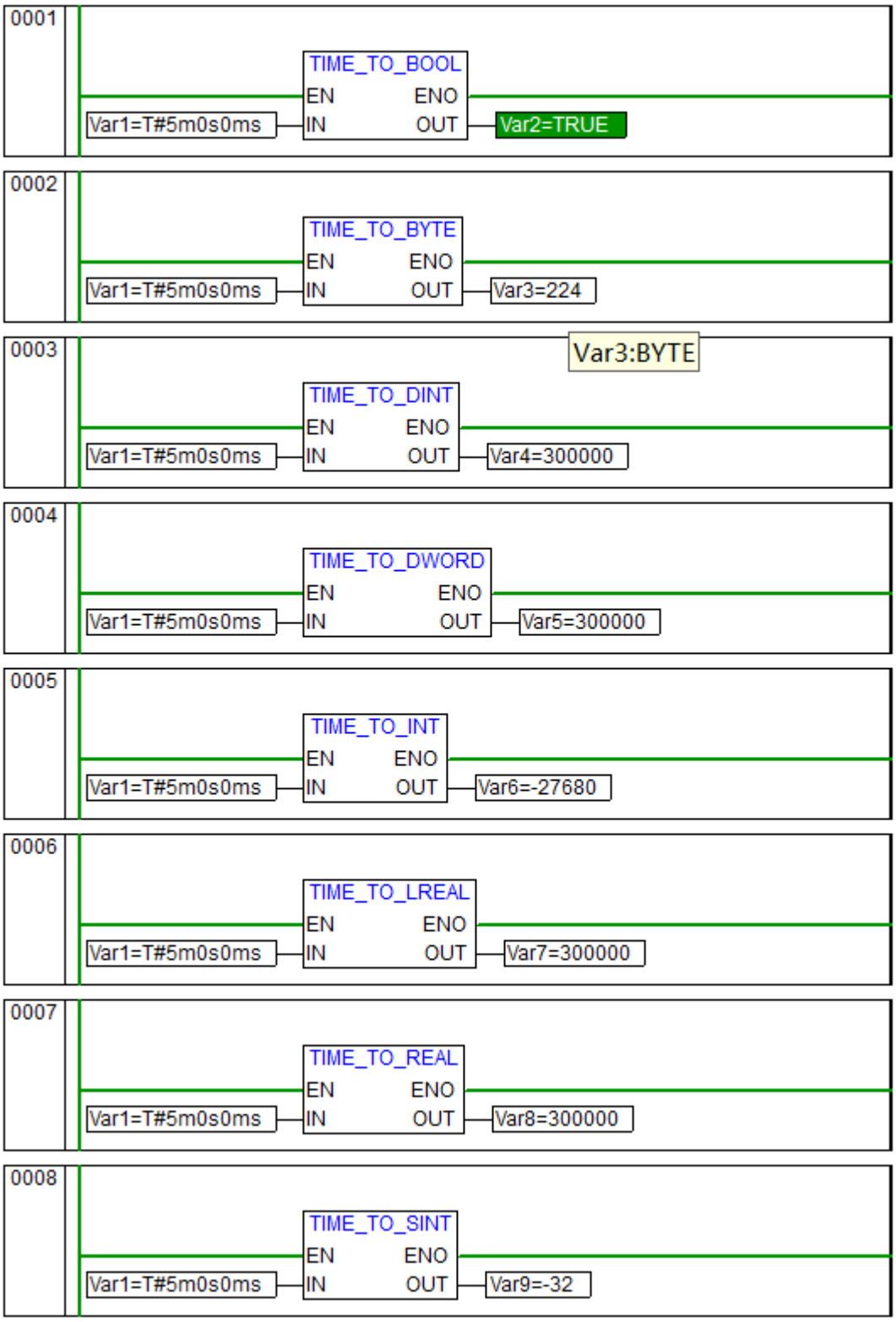
5.6.12.2 Parameter

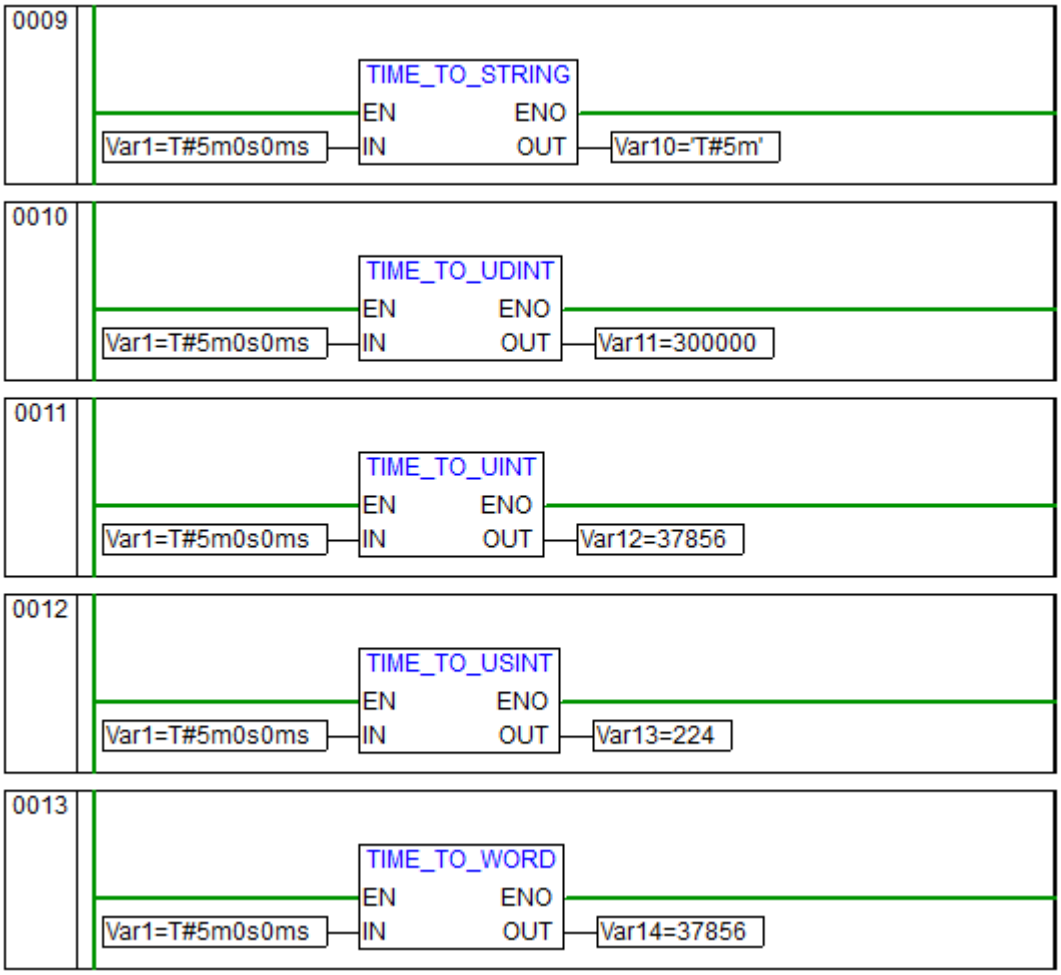
Parameter	Statement	Data Type	Description
IN	Input	TIME	Enter the time variable to be converted
OUT	Output	BOOL, BYTE, DINT, DWORD, INT, LREAL, REAL, STRING, SINT, UDINT, UINT, USINT, WORD	Converted data type

5.6.12.3 Example

The following example uses the TIME conversion instruction to convert TIME variables to other data types. When the input EN detects a rising edge, the TIME conversion instruction starts the conversion function and stores the calculation result in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			TIME	T#5m0s0ms	FALSE	Not Public	
0002	Var2			BOOL	TRUE	FALSE	Not Public	
0003	Var3			BYTE	224	FALSE	Not Public	
0004	Var4			DINT	300000	FALSE	Not Public	
0005	Var5			DWORD	300000	FALSE	Not Public	
0006	Var6			INT	27680	FALSE	Not Public	
0007	Var7			LREAL	300000	FALSE	Not Public	
0008	Var8			INT	100	FALSE	Not Public	
0009	Var9			LREAL	32	FALSE	Not Public	
0010	Var10			STRING(80)	'T#5m'	FALSE	Not Public	
0011	Var11			UDINT	300000	FALSE	Not Public	
0012	Var12			UINT	37856	FALSE	Not Public	
0013	Var13			USINT	224	FALSE	Not Public	
0014	Var14			WORD	37856	FALSE	Not Public	


0001
TIME_TO_BOOL
EN
ENO
Var1=T#5m0s0ms
IN
OUT
Var2=TRUE
0002
TIME_TO_BYTE
EN
ENO
Var1=T#5m0s0ms
IN
OUT
Var3=224
0003
Var3:BYTE
TIME_TO_DINT
EN
ENO
Var1=T#5m0s0ms
IN
OUT
Var4=300000
0004
TIME_TO_DWORD
EN
ENO
Var1=T#5m0s0ms
IN
OUT
Var5=300000
0005
TIME_TO_INT
EN
ENO
Var1=T#5m0s0ms
IN
OUT
Var6=-27680
0006
TIME_TO_LREAL
EN
ENO
Var1=T#5m0s0ms
IN
OUT
Var7=300000
0007
TIME_TO_REAL
EN
ENO
Var1=T#5m0s0ms
IN
OUT
Var8=300000
0008
TIME_TO_SINT
EN
ENO
Var1=T#5m0s0ms
IN
OUT
Var9=-32




```
1  Var2 := TIME_TO_BOOL(Var1);
2  Var3 := TIME_TO_BYTE(Var1);
3  Var4 := TIME_TO_DINT(Var1);
4  Var5 := TIME_TO_DWORD(Var1);
5  Var6 := TIME_TO_INT(Var1);
6  Var7 := TIME_TO_LREAL(Var1);
7  Var8 := TIME_TO_REAL(Var1);
8  Var9 := TIME_TO_SINT(Var1);
9  Var10 := TIME_TO_STRING(Var1);
10 Var11 := TIME_TO_UDINT(Var1);
11 Var12 := TIME_TO_UINT(Var1);
12 Var13 := TIME_TO_USINT(Var1);
13 Var14 := TIME_TO_WORD(Var1);
```

```
Var2 = TRUE
Var3 = 224
Var4 = 300000
Var5 = 300000
Var6 = -27680
Var7 = 300000
Var8 = 300000
Var9 = -32
Var10 = 'T#5m'
Var11 = 300000
Var12 = 37856
Var13 = 224
Var14 = 37856
```

```
Var1 = T#5m0s0ms
Var1 = T#5m0s0ms
Var1 = T#5m0s0ms
Var1 = T#5m0s0ms
Var1 = T#5m0s0ms
Var1 = T#5m0s0ms
Var1 = T#5m0s0ms
Var1 = T#5m0s0ms
Var1 = T#5m0s0ms
Var1 = T#5m0s0ms
Var1 = T#5m0s0ms
Var1 = T#5m0s0ms
Var1 = T#5m0s0ms
```

5.6.13 TOD_TO_TYPE (Time Type Conversion Instruction)

5.6.13.1 Function

It is to convert time data to other types of data. The date is converted internally in milliseconds.

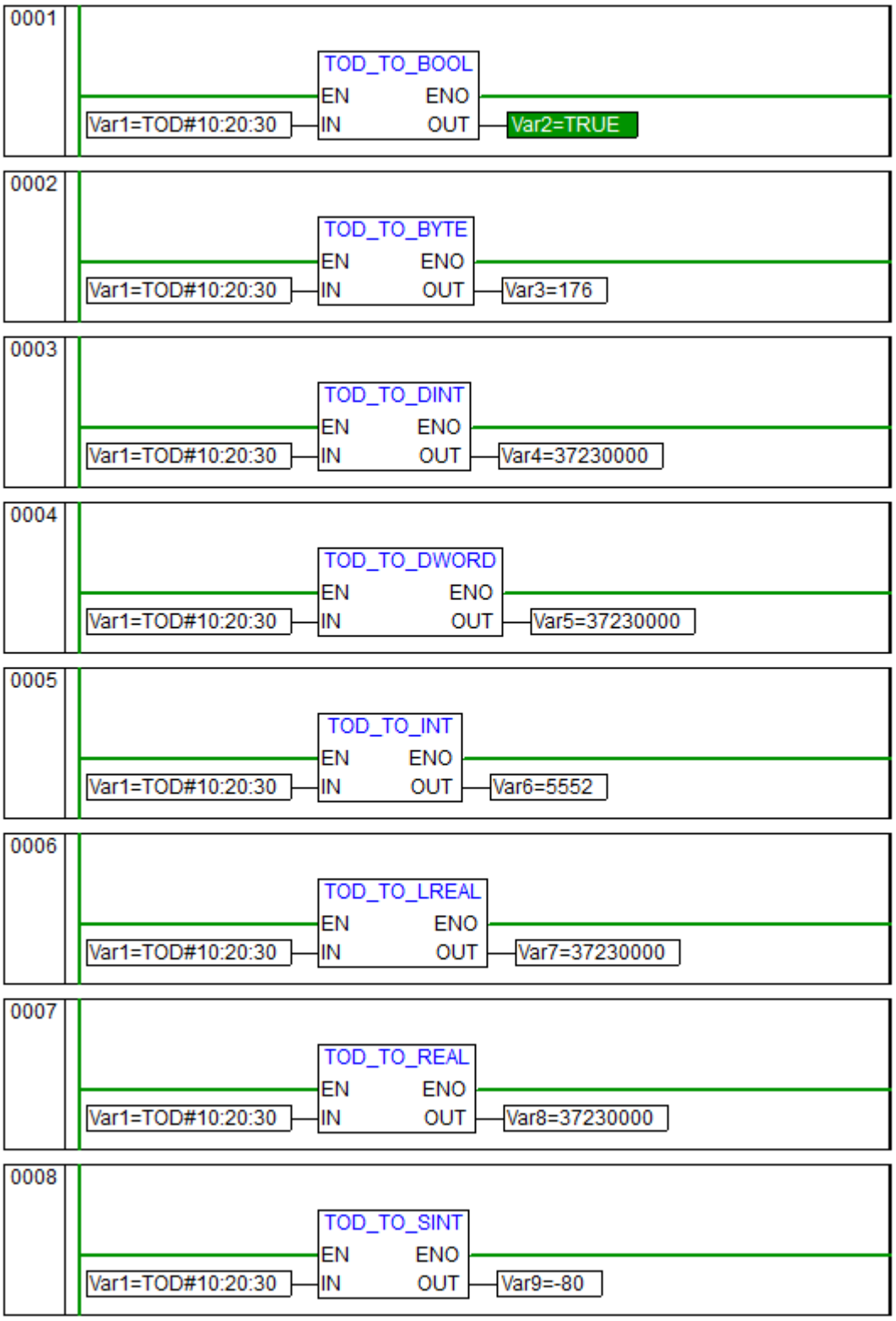
5.6.13.2 Parameter

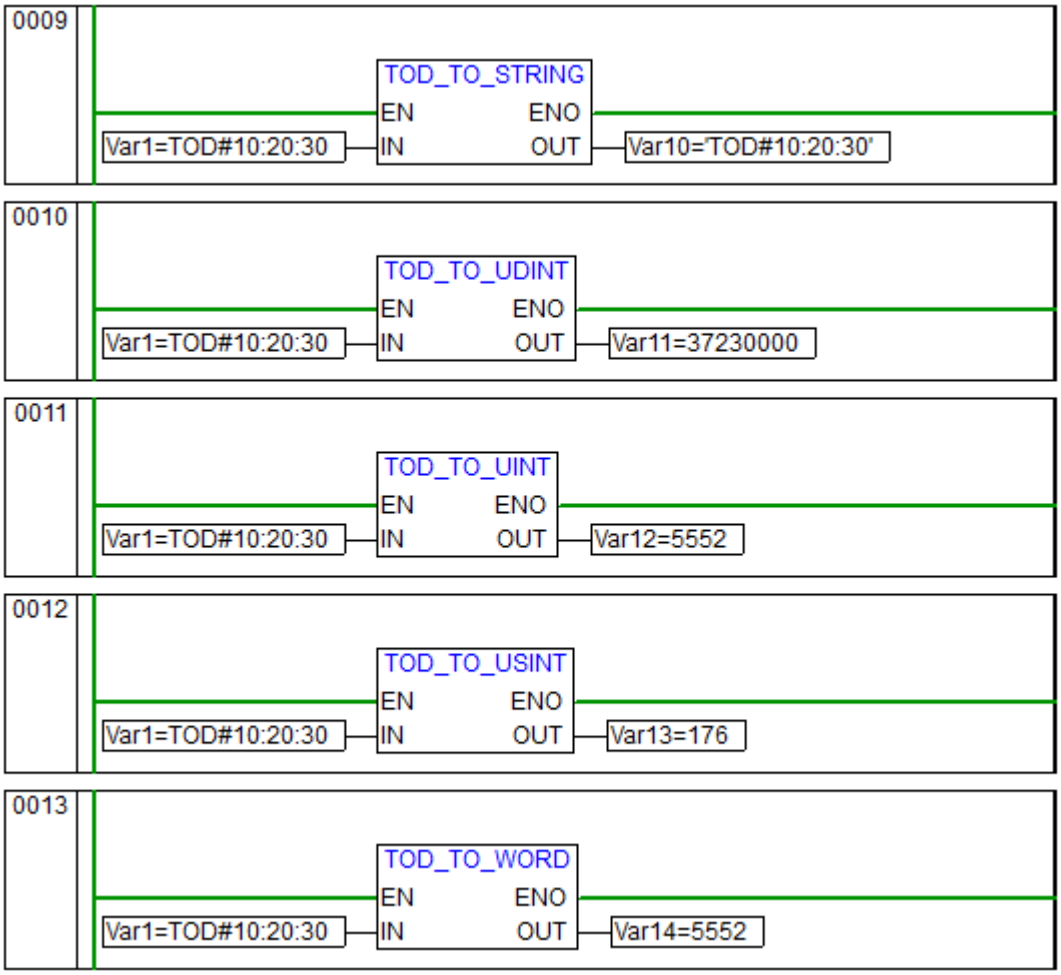
Parameter	Statement	Data Type	Description
IN	Input	TOD	Enter the TOD variable to be converted
OUT	Output	BOOL, BYTE, DINT, DWORD, INT, LREAL, REAL, STRING, SINT, UDINT, UINT, USINT, WORD	Converted data type

5.6.13.3 Example

The following example uses the TOD conversion instruction to convert TOD variables to other data types. When the input EN detects a rising edge, the TOD conversion instruction starts the conversion function and stores the calculation result in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			TOD	TOD#10:20:30	FALSE	Not Public	
0002	Var2			BOOL	TRUE	FALSE	Not Public	
0003	Var3			BYTE	176	FALSE	Not Public	
0004	Var4			DINT	37230000	FALSE	Not Public	
0005	Var5			DWORD	37230000	FALSE	Not Public	
0006	Var6			INT	0	FALSE	Not Public	
0007	Var7			LREAL	37230000	FALSE	Not Public	
0008	Var8			REAL	3723000	FALSE	Not Public	
0009	Var9			SINT	80	FALSE	Not Public	
0010	Var10			STRING(80)	'TOD#10:20:30'	FALSE	Not Public	
0011	Var11			UDINT	37230000	FALSE	Not Public	
0012	Var12			UINT	5552	FALSE	Not Public	
0013	Var13			USINT	176	FALSE	Not Public	
0014	Var14			WORD	5552	FALSE	Not Public	


0001
TOD_TO_BOOL
EN
ENO
Var1=TOD#10:20:30
IN
OUT
Var2=TRUE
0002
TOD_TO_BYTE
EN
ENO
Var1=TOD#10:20:30
IN
OUT
Var3=176
0003
TOD_TO_DINT
EN
ENO
Var1=TOD#10:20:30
IN
OUT
Var4=37230000
0004
TOD_TO_DWORD
EN
ENO
Var1=TOD#10:20:30
IN
OUT
Var5=37230000
0005
TOD_TO_INT
EN
ENO
Var1=TOD#10:20:30
IN
OUT
Var6=5552
0006
TOD_TO_LREAL
EN
ENO
Var1=TOD#10:20:30
IN
OUT
Var7=37230000
0007
TOD_TO_REAL
EN
ENO
Var1=TOD#10:20:30
IN
OUT
Var8=37230000
0008
TOD_TO_SINT
EN
ENO
Var1=TOD#10:20:30
IN
OUT
Var9=-80




```
1  Var2 := TOD_TO_BOOL(Var1);
2  Var3 := TOD_TO_BYTE(Var1);
3  Var4 := TOD_TO_DINT(Var1);
4  Var5 := TOD_TO_DWORD(Var1);
5  Var6 := TOD_TO_INT(Var1);
6  Var7 := TOD_TO_LREAL(Var1);
7  Var8 := TOD_TO_REAL(Var1);
8  Var9 := TOD_TO_SINT(Var1);
9  Var10 := TOD_TO_STRING(Var1);
10 Var11 := TOD_TO_UDINT(Var1);
11 Var12 := TOD_TO_UINT(Var1);
12 Var13 := TOD_TO_USINT(Var1);
13 Var14 := TOD_TO_WORD(Var1);
```

```
Var2 = TRUE
Var3 = 176
Var4 = 37230000
Var5 = 37230000
Var6 = 5552
Var7 = 37230000
Var8 = 37230000
Var9 = -80
Var10 = 'TOD#10:20:30'
Var11 = 37230000
Var12 = 5552
Var13 = 176
Var14 = 5552
```

```
Var1 = TOD#10:20:30
Var1 = TOD#10:20:30
Var1 = TOD#10:20:30
Var1 = TOD#10:20:30
Var1 = TOD#10:20:30
Var1 = TOD#10:20:30
Var1 = TOD#10:20:30
Var1 = TOD#10:20:30
Var1 = TOD#10:20:30
Var1 = TOD#10:20:30
Var1 = TOD#10:20:30
Var1 = TOD#10:20:30
Var1 = TOD#10:20:30
```

5.6.14 UDINT_TO_TYPE (Unsigned Double Integer Type Conversion Instruction)

5.6.14.1 Function

Convert unsigned double integer type to another data type.

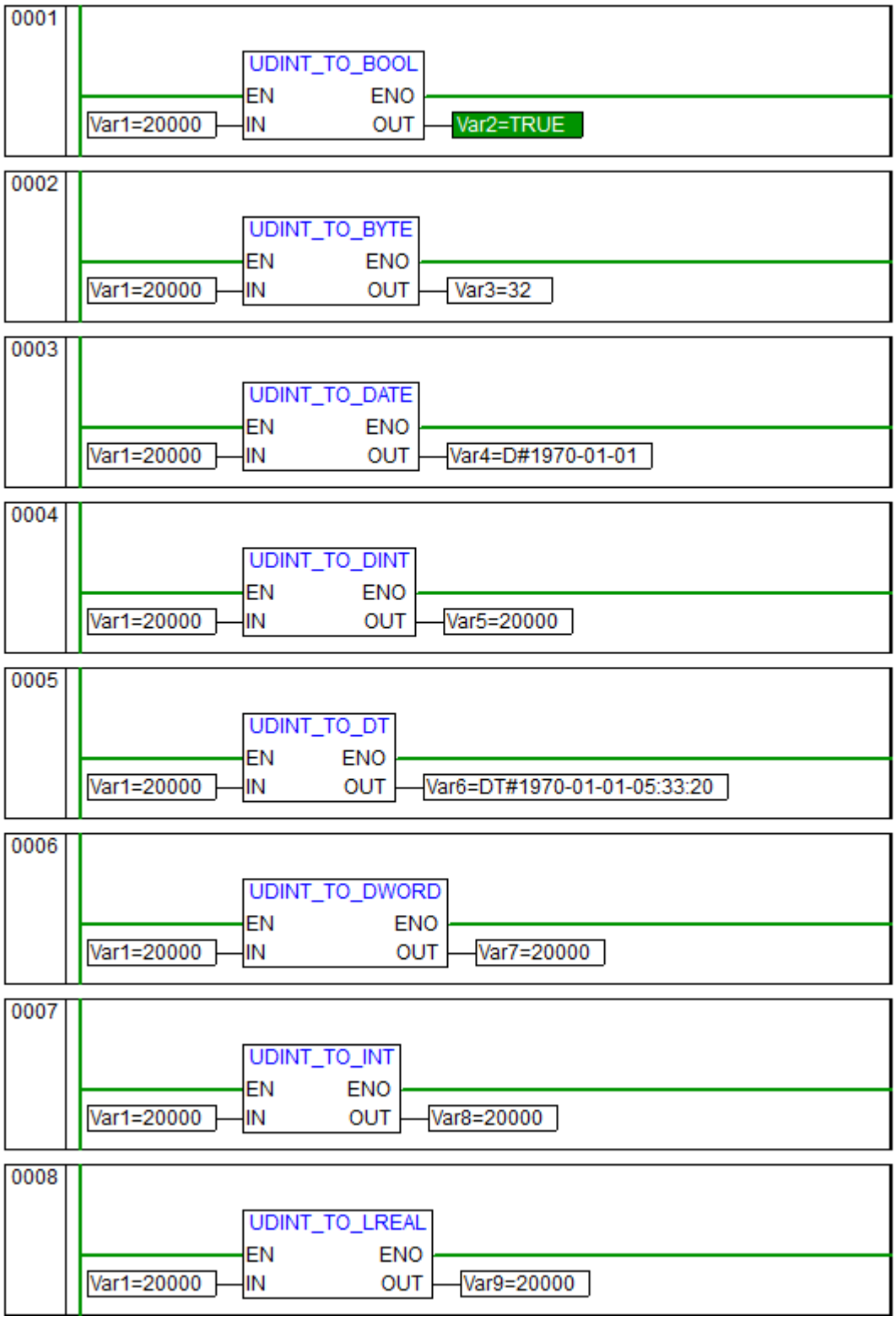
5.6.14.2 Parameter

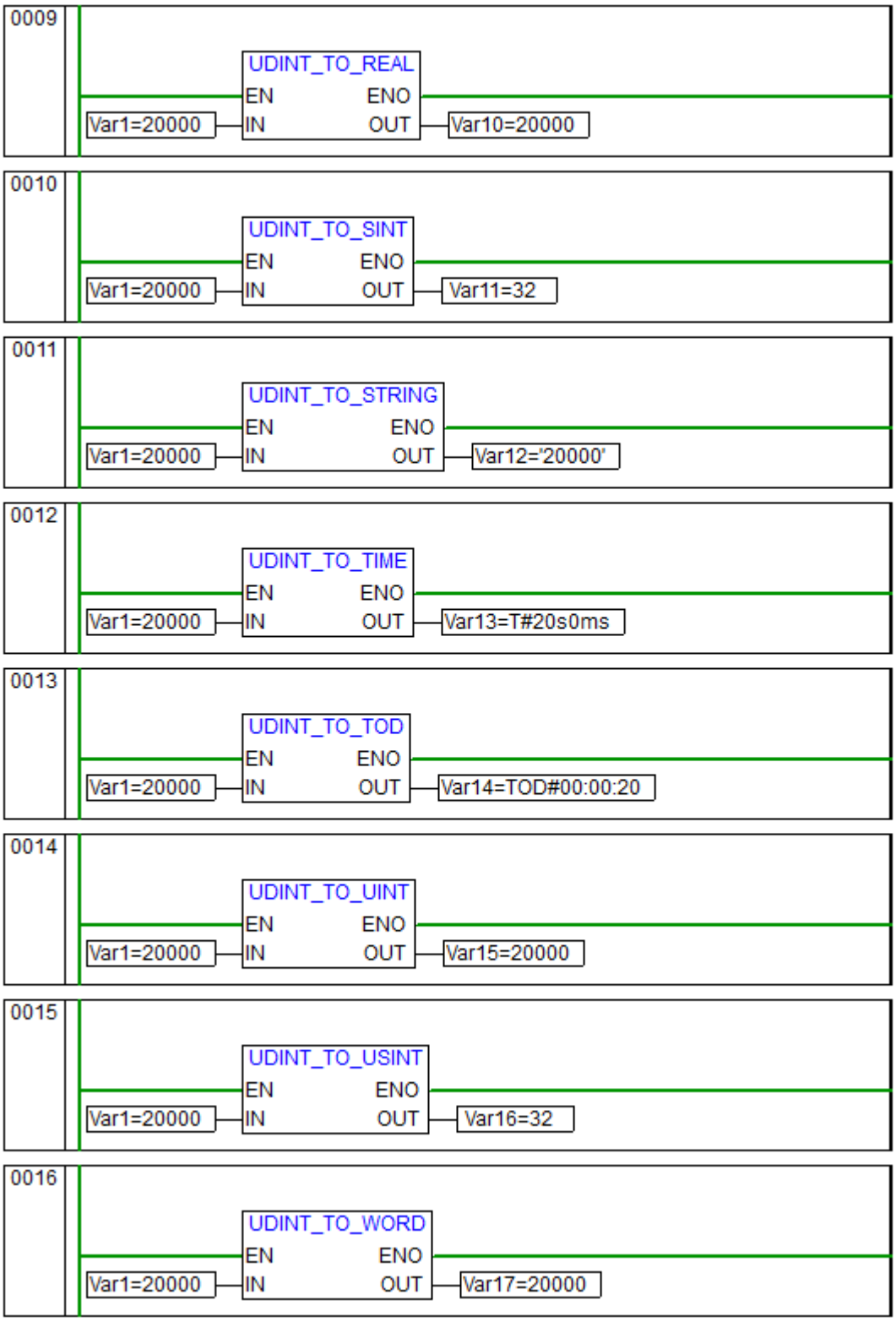
Parameter	Statement	Data Type	Description
IN	Input	UDINT	Enter the unsigned double integer variable to be converted
OUT	Output	BOOL, BYTE, DATE, DINT, DT, DWORD, INT, LREAL, REAL, SINT, STRING, TIME, TOD, UINT, USINT, WORD	Converted data type

5.6.14.3 Example

The following example uses the UDINT conversion instruction to convert UDINT variables to other data types. When the input EN detects a rising edge, the UDINT conversion instruction starts the conversion function and stores the calculation result in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			UDINT	20000	FALSE	Not Public	
0002	Var2			BOOL	TRUE	FALSE	Not Public	
0003	Var3			BYTE	32	FALSE	Not Public	
0004	Var4			DATE	D#1970-01-01	FALSE	Not Public	
0005	Var5			DINT	20000	FALSE	Not Public	
0006	Var6			DT	DT#1970-01-...	FALSE	Not Public	
0007	Var7			DWORD	20000	FALSE	Not Public	
0008	Var8			INT	20000	FALSE	Not Public	
0009	Var9			LREAL	20000	FALSE	Not Public	
0010	Var10			REAL	20000	FALSE	Not Public	
0011	Var11			SINT	32	FALSE	Not Public	
0012	Var12			STRING(80)	'20000'	FALSE	Not Public	
0013	Var13			TIME	T#20s0MS	FALSE	Not Public	
0014	Var14			TOD	TOD#00:00:20	FALSE	Not Public	
0015	Var15			UINT	20000	FALSE	Not Public	
0016	Var16			USINT	32	FALSE	Not Public	
0017	Var17			WORD	20000	FALSE	Not Public	


0001
UDINT_TO_BOOL
EN
ENO
Var1=20000
IN
OUT
Var2=TRUE
0002
UDINT_TO_BYTE
EN
ENO
Var1=20000
IN
OUT
Var3=32
0003
UDINT_TO_DATE
EN
ENO
Var1=20000
IN
OUT
Var4=D#1970-01-01
0004
UDINT_TO_DINT
EN
ENO
Var1=20000
IN
OUT
Var5=20000
0005
UDINT_TO_DT
EN
ENO
Var1=20000
IN
OUT
Var6=DT#1970-01-01-05:33:20
0006
UDINT_TO_DWORD
EN
ENO
Var1=20000
IN
OUT
Var7=20000
0007
UDINT_TO_INT
EN
ENO
Var1=20000
IN
OUT
Var8=20000
0008
UDINT_TO_LREAL
EN
ENO
Var1=20000
IN
OUT
Var9=20000


0009
UDINT_TO_REAL
EN
ENO
Var1=20000
IN
OUT
Var10=20000
0010
UDINT_TO_SINT
EN
ENO
Var1=20000
IN
OUT
Var11=32
0011
UDINT_TO_STRING
EN
ENO
Var1=20000
IN
OUT
Var12='20000'
0012
UDINT_TO_TIME
EN
ENO
Var1=20000
IN
OUT
Var13=T#20s0ms
0013
UDINT_TO_TOD
EN
ENO
Var1=20000
IN
OUT
Var14=TOD#00:00:20
0014
UDINT_TO_UINT
EN
ENO
Var1=20000
IN
OUT
Var15=20000
0015
UDINT_TO_USINT
EN
ENO
Var1=20000
IN
OUT
Var16=32
0016
UDINT_TO_WORD
EN
ENO
Var1=20000
IN
OUT
Var17=20000

```
1   Var2 := UDINT_TO_BOOL(Var1);
2   Var3 := UDINT_TO_BYTE(Var1);
3   Var4 := UDINT_TO_DATE(Var1);
4   Var5 := UDINT_TO_DINT(Var1);
5   Var6 := UDINT_TO_DT(Var1);
6   Var7 := UDINT_TO_DWORD(Var1);
7   Var8 := UDINT_TO_INT(Var1);
8   Var9 := UDINT_TO_LREAL(Var1);
9   Var10 := UDINT_TO_REAL(Var1);
10  Var11 := UDINT_TO_SINT(Var1);
11  Var12 := UDINT_TO_STRING(Var1);
12  Var13 := UDINT_TO_TIME(Var1);
13  Var14 := UDINT_TO_TOD(Var1);
14  Var15 := UDINT_TO_UINT(Var1);
15  Var16 := UDINT_TO_USINT(Var1);
16  Var17 := UDINT_TO_WORD(Var1);
```

```
Var2 = TRUE                    Var1 = 20000
Var3 = 32                      Var1 = 20000
Var4 = D#1970-01-01            Var1 = 20000
Var5 = 20000                   Var1 = 20000
Var6 = DT#1970-01-01-05:33:20  Var1 = 20000
Var7 = 20000                   Var1 = 20000
Var8 = 20000                   Var1 = 20000
Var9 = 20000                   Var1 = 20000
Var10 = 20000                  Var1 = 20000
Var11 = 32                     Var1 = 20000
Var12 = '20000'                Var1 = 20000
Var13 = T#20s0ms               Var1 = 20000
Var14 = TOD#00:00:20           Var1 = 20000
Var15 = 20000                  Var1 = 20000
Var16 = 32                     Var1 = 20000
Var17 = 20000                  Var1 = 20000
```

5.6.15 UINT_TO_TYPE (Unsigned Integer Type Conversion Instruction)

5.6.15.1 Function

Convert unsigned integer type to other data types.

In the case of UINT_TO_BOOL, the output is TRUE when the input is not equal to 0, and FALSE when the input is equal to 0.

In the case of UINT_TO_TIME and UINT_TO_TOD, the input will be converted in milliseconds.

In the case of UINT_TO_DATE and UINT_TO_DT, the input will be converted in seconds.

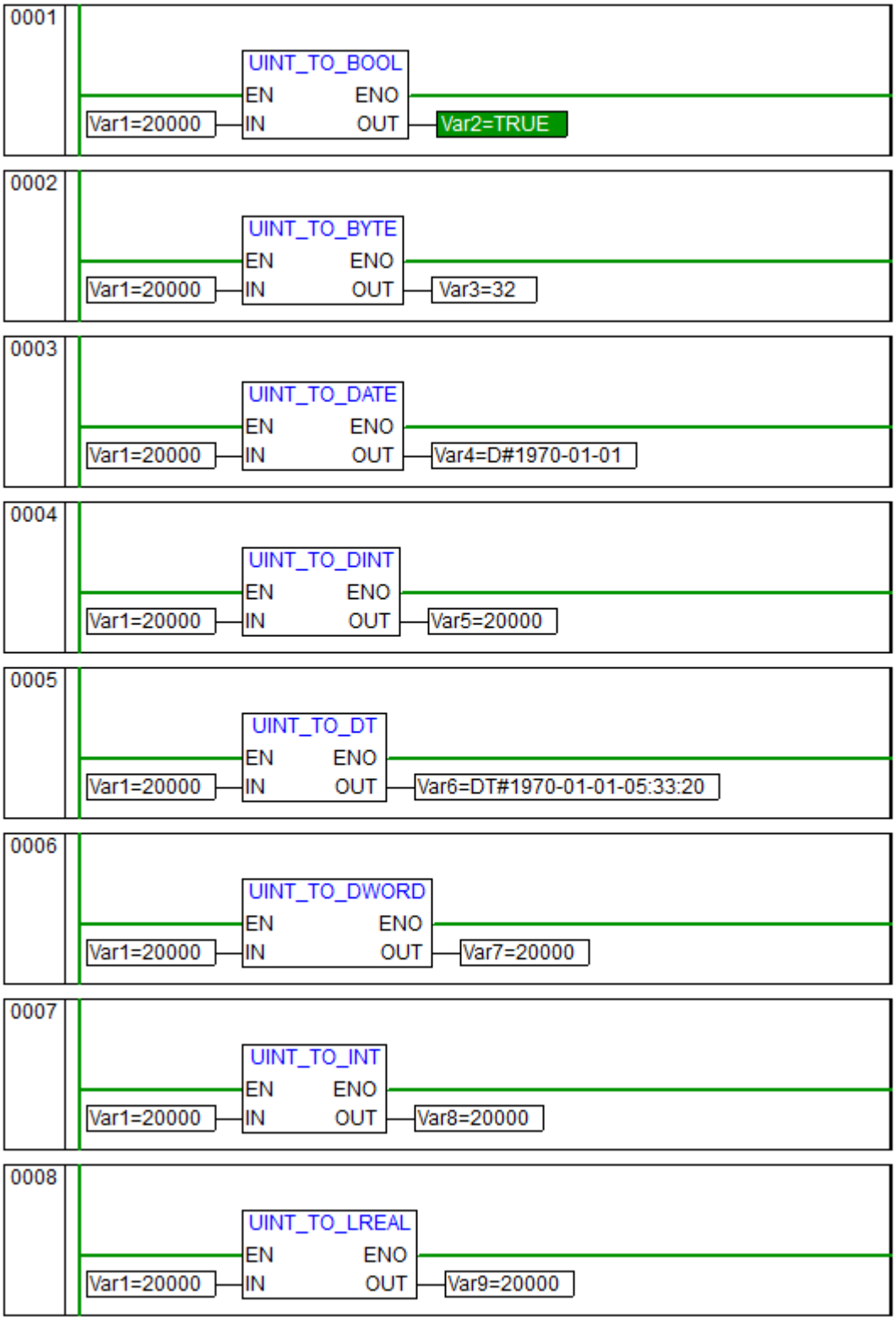
5.6.15.2 Parameter

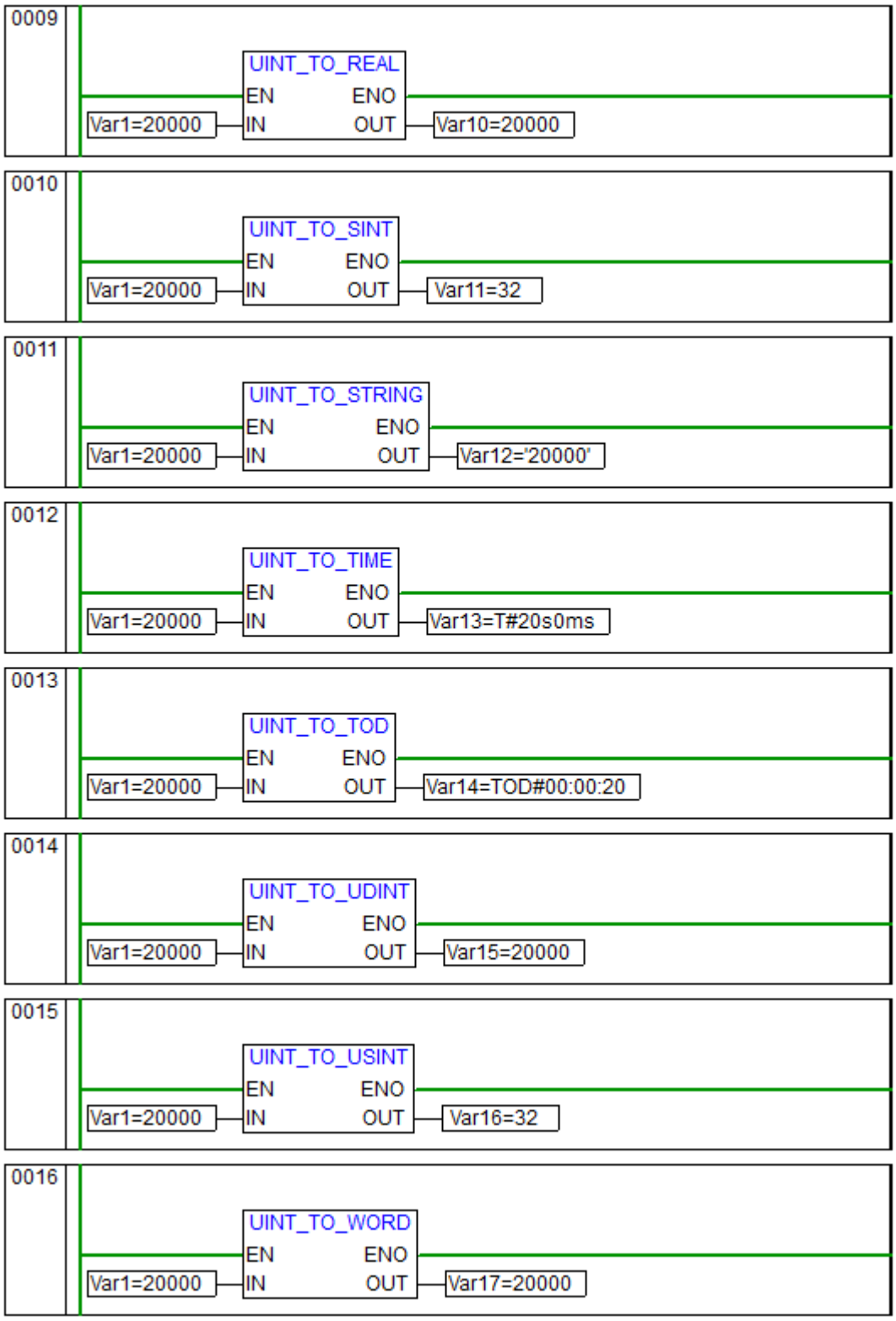
Parameter	Statement	Data Type	Description
IN	Input	UINT	Enter the unsigned integer variable to be converted
OUT	Output	BOOL, BYTE, DATE, DINT, DT, DWORD, INT, LREAL, REAL, SINT, STRING, TIME, TOD, UDINT, USINT, WORD	Converted data type

5.6.15.3 Example

The following example uses the UINT conversion instruction to convert UINT variables to other data types. When the input EN detects a rising edge, the UINT conversion instruction starts the conversion function and stores the calculation result in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			UINT	20000	FALSE	Not Public	
0002	Var2			BOOL	FALSE	FALSE	Not Public	
0003	Var3			BYTE	32	FALSE	Not Public	
0004	Var4			DATE	D#1970-01-01	FALSE	Not Public	
0005	Var5			DINT	20000	FALSE	Not Public	
0006	Var6			DT	DT#1970-01-01...	FALSE	Not Public	
0007	Var7			DWORD	20000	FALSE	Not Public	
0008	Var8			INT	20000	FALSE	Not Public	
0009	Var9			LREAL	20000	FALSE	Not Public	
0010	Var10			REAL	20000	FALSE	Not Public	
0011	Var11			SINT	32	FALSE	Not Public	
0012	Var12			STRING(80)	'20000'	FALSE	Not Public	
0013	Var13			TIME	T#20s0ms	FALSE	Not Public	
0014	Var14			TOD	TOD#00:00:20	FALSE	Not Public	
0015	Var15			UDINT	20000	FALSE	Not Public	
0016	Var16			USINT	32	FALSE	Not Public	
0017	Var17			WORD	20000	FALSE	Not Public	


0001
UINT_TO_BOOL
EN
ENO
Var1=20000
IN
OUT
Var2=TRUE
0002
UINT_TO_BYTE
EN
ENO
Var1=20000
IN
OUT
Var3=32
0003
UINT_TO_DATE
EN
ENO
Var1=20000
IN
OUT
Var4=D#1970-01-01
0004
UINT_TO_DINT
EN
ENO
Var1=20000
IN
OUT
Var5=20000
0005
UINT_TO_DT
EN
ENO
Var1=20000
IN
OUT
Var6=DT#1970-01-01-05:33:20
0006
UINT_TO_DWORD
EN
ENO
Var1=20000
IN
OUT
Var7=20000
0007
UINT_TO_INT
EN
ENO
Var1=20000
IN
OUT
Var8=20000
0008
UINT_TO_LREAL
EN
ENO
Var1=20000
IN
OUT
Var9=20000


0009
UINT_TO_REAL
EN
ENO
Var1=20000
IN
OUT
Var10=20000
0010
UINT_TO_SINT
EN
ENO
Var1=20000
IN
OUT
Var11=32
0011
UINT_TO_STRING
EN
ENO
Var1=20000
IN
OUT
Var12='20000'
0012
UINT_TO_TIME
EN
ENO
Var1=20000
IN
OUT
Var13=T#20s0ms
0013
UINT_TO_TOD
EN
ENO
Var1=20000
IN
OUT
Var14=TOD#00:00:20
0014
UINT_TO_UDINT
EN
ENO
Var1=20000
IN
OUT
Var15=20000
0015
UINT_TO_USINT
EN
ENO
Var1=20000
IN
OUT
Var16=32
0016
UINT_TO_WORD
EN
ENO
Var1=20000
IN
OUT
Var17=20000

```
1   Var2 := UINT_TO_BOOL(Var1);
2   Var3 := UINT_TO_BYTE(Var1);
3   Var4 := UINT_TO_DATE(Var1);
4   Var5 := UINT_TO_DINT(Var1);
5   Var6 := UINT_TO_DT(Var1);
6   Var7 := UINT_TO_DWORD(Var1);
7   Var8 := UINT_TO_INT(Var1);
8   Var9 := UINT_TO_LREAL(Var1);
9   Var10 := UINT_TO_REAL(Var1);
10  Var11 := UINT_TO_SINT(Var1);
11  Var12 := UINT_TO_STRING(Var1);
12  Var13 := UINT_TO_TIME(Var1);
13  Var14 := UINT_TO_TOD(Var1);
14  Var15 := UINT_TO_UDINT(Var1);
15  Var16 := UINT_TO_USINT(Var1);
16  Var17 := UINT_TO_WORD(Var1);
```

```
Var2 = TRUE                    Var1 = 20000
Var3 = 32                      Var1 = 20000
Var4 = D#1970-01-01            Var1 = 20000
Var5 = 20000                   Var1 = 20000
Var6 = DT#1970-01-01-05:33:20  Var1 = 20000
Var7 = 20000                   Var1 = 20000
Var8 = 20000                   Var1 = 20000
Var9 = 20000                   Var1 = 20000
Var10 = 20000                  Var1 = 20000
Var11 = 32                     Var1 = 20000
Var12 = '20000'                Var1 = 20000
Var13 = T#20s0ms               Var1 = 20000
Var14 = TOD#00:00:20           Var1 = 20000
Var15 = 20000                  Var1 = 20000
Var16 = 32                     Var1 = 20000
Var17 = 20000                  Var1 = 20000
```

5.6.16 USINT_TO_TYPE (Unsigned Short Integer Type Conversion Instruction)

5.6.16.1 Function

Convert the unsigned short integer type to other data types.

In the case of USINT_TO_BOOL, the output is TRUE when the input is not equal to 0, and FALSE when the input is equal to 0.

In the case of USINT_TO_TIME and USINT_TO_TOD, the input will be converted in milliseconds.

In the case of USINT_TO_DATE and USINT_TO_DT, the input will be converted in seconds.

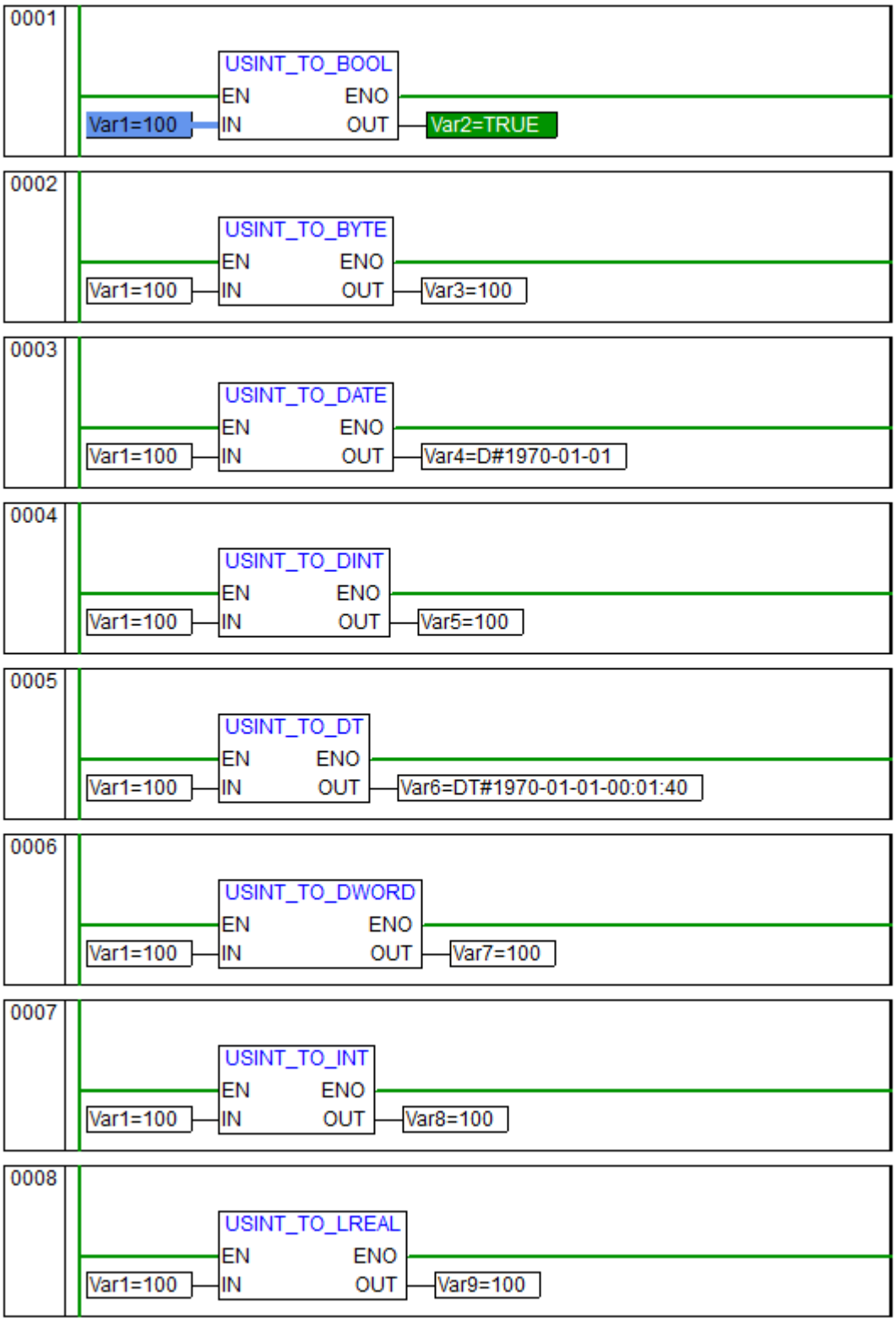
5.6.16.2 Parameter

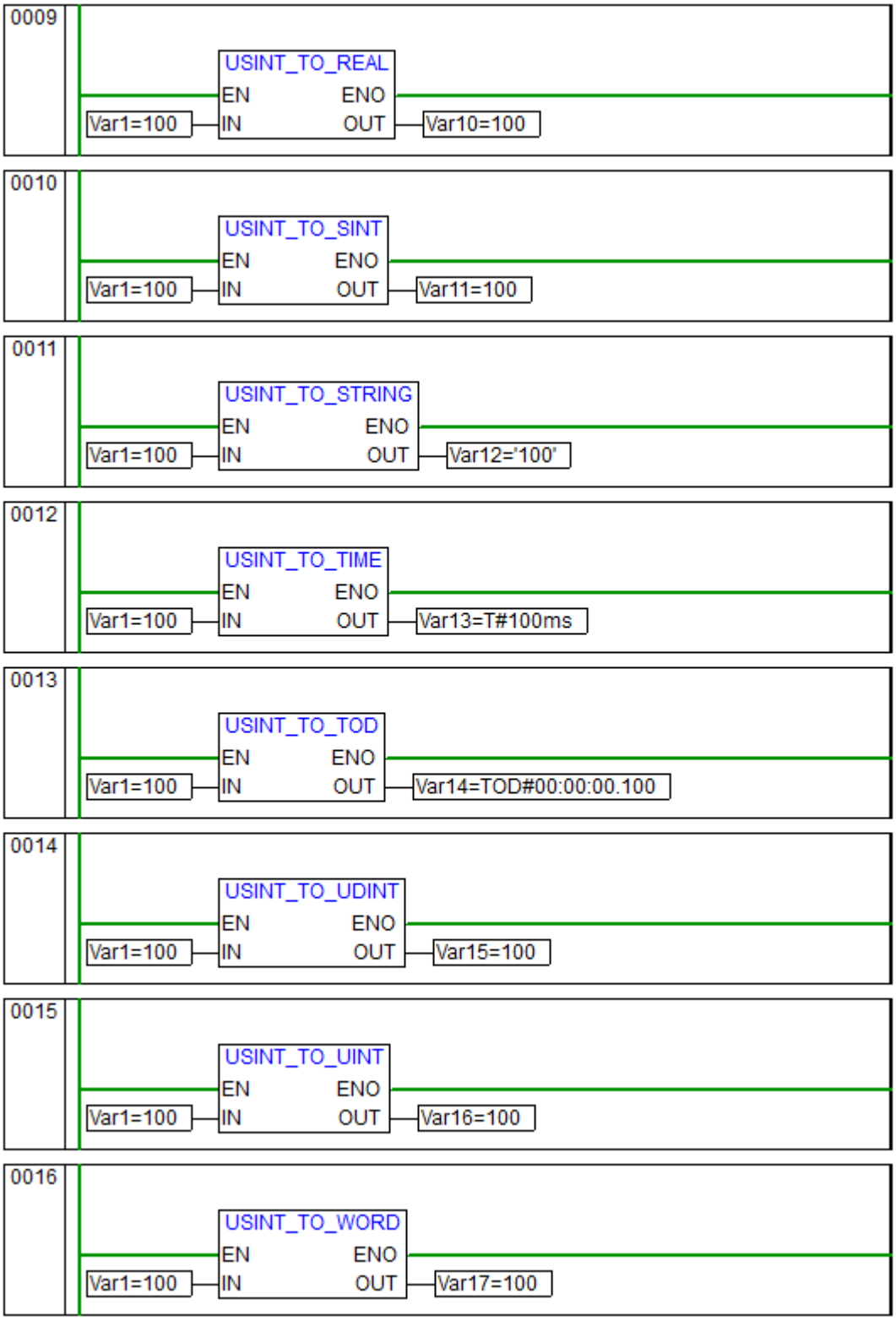
Parameter	Statement	Data Type	Description
IN	Input	USINT	Enter the unsigned short integer variable to be converted
OUT	Output	BOOL, BYTE, DATE, DINT, DT, DWORD, INT, LREAL, REAL, SINT, STRING, TIME, TOD, UDINT, UINT, WORD	Converted data type

5.6.16.3 Example

The following example uses the USINT conversion instruction to convert USINT variables to other data types. When the input EN detects a rising edge, the USINT conversion instruction starts the conversion function and stores the calculation result in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			USINT	100	FALSE	Not Public	
0002	Var2			BOOL	TRUE	FALSE	Not Public	
0003	Var3			BYTE	100	FALSE	Not Public	
0004	Var4			DATE	D#1970-01-01	FALSE	Not Public	
0005	Var5			DINT	100	FALSE	Not Public	
0006	Var6			DT	DT#1970-01-01...	FALSE	Not Public	
0007	Var7			DWORD	100	FALSE	Not Public	
0008	Var8			INT	100	FALSE	Not Public	
0009	Var9			LREAL	100	FALSE	Not Public	
0010	Var10			REAL	100	FALSE	Not Public	
0011	Var11			SINT	100	FALSE	Not Public	
0012	Var12			STRING(80)	'100'	FALSE	Not Public	
0013	Var13			TIME	T#1s0ms	FALSE	Not Public	
0014	Var14			TOD	TOD#00:00:00....	FALSE	Not Public	
0015	Var15			UDINT	100	FALSE	Not Public	
0016	Var16			UINT	100	FALSE	Not Public	
0017	Var17			WORD	100	FALSE	Not Public	


0001
USINT_TO_BOOL
EN
ENO
Var1=100
IN
OUT
Var2=TRUE
0002
USINT_TO_BYTE
EN
ENO
Var1=100
IN
OUT
Var3=100
0003
USINT_TO_DATE
EN
ENO
Var1=100
IN
OUT
Var4=D#1970-01-01
0004
USINT_TO_DINT
EN
ENO
Var1=100
IN
OUT
Var5=100
0005
USINT_TO_DT
EN
ENO
Var1=100
IN
OUT
Var6=DT#1970-01-01-00:01:40
0006
USINT_TO_DWORD
EN
ENO
Var1=100
IN
OUT
Var7=100
0007
USINT_TO_INT
EN
ENO
Var1=100
IN
OUT
Var8=100
0008
USINT_TO_LREAL
EN
ENO
Var1=100
IN
OUT
Var9=100


0009
USINT_TO_REAL
EN
ENO
Var1=100
IN
OUT
Var10=100
0010
USINT_TO_SINT
EN
ENO
Var1=100
IN
OUT
Var11=100
0011
USINT_TO_STRING
EN
ENO
Var1=100
IN
OUT
Var12='100'
0012
USINT_TO_TIME
EN
ENO
Var1=100
IN
OUT
Var13=T#100ms
0013
USINT_TO_TOD
EN
ENO
Var1=100
IN
OUT
Var14=TOD#00:00:00.100
0014
USINT_TO_UDINT
EN
ENO
Var1=100
IN
OUT
Var15=100
0015
USINT_TO_UINT
EN
ENO
Var1=100
IN
OUT
Var16=100
0016
USINT_TO_WORD
EN
ENO
Var1=100
IN
OUT
Var17=100

```
1   Var2 := USINT_TO_BOOL(Var1);
2   Var3 := USINT_TO_BYTE(Var1);
3   Var4 := USINT_TO_DATE(Var1);
4   Var5 := USINT_TO_DINT(Var1);
5   Var6 := USINT_TO_DT(Var1);
6   Var7 := USINT_TO_DWORD(Var1);
7   Var8 := USINT_TO_INT(Var1);
8   Var9 := USINT_TO_LREAL(Var1);
9   Var10 := USINT_TO_REAL(Var1);
10  Var11 := USINT_TO_SINT(Var1);
11  Var12 := USINT_TO_STRING(Var1);
12  Var13 := USINT_TO_TIME(Var1);
13  Var14 := USINT_TO_TOD(Var1);
14  Var15 := USINT_TO_UDINT(Var1);
15  Var16 := USINT_TO_UINT(Var1);
16  Var17 := USINT_TO_WORD(Var1);
```

```
Var2 = TRUE                    Var1 = 100
Var3 = 100                     Var1 = 100
Var4 = D#1970-01-01            Var1 = 100
Var5 = 100                     Var1 = 100
Var6 = DT#1970-01-01-00:01:40  Var1 = 100
Var7 = 100                     Var1 = 100
Var8 = 100                     Var1 = 100
Var9 = 100                     Var1 = 100
Var10 = 100                    Var1 = 100
Var11 = 100                    Var1 = 100
Var12 = '100'                  Var1 = 100
Var13 = T#100ms                Var1 = 100
Var14 = TOD#00:00:00.100       Var1 = 100
Var15 = 100                    Var1 = 100
Var16 = 100                    Var1 = 100
Var17 = 100                    Var1 = 100
```

5.6.17 WORD_TO_TYPE (Word Type Conversion Instruction)

5.6.17.1 Function

Convert the word type to other data types.

In the case of WORD_TO_BOOL, the output is TRUE when the input is not equal to 0, and FALSE when the input is equal to 0.

In the case of WORD_TO_TIME and WORD_TO_TOD, the input will be converted in milliseconds.

In the case of WORD_TO_DATE and WORD_TO_DT, the input will be converted in seconds.

5.6.17.2 Parameter

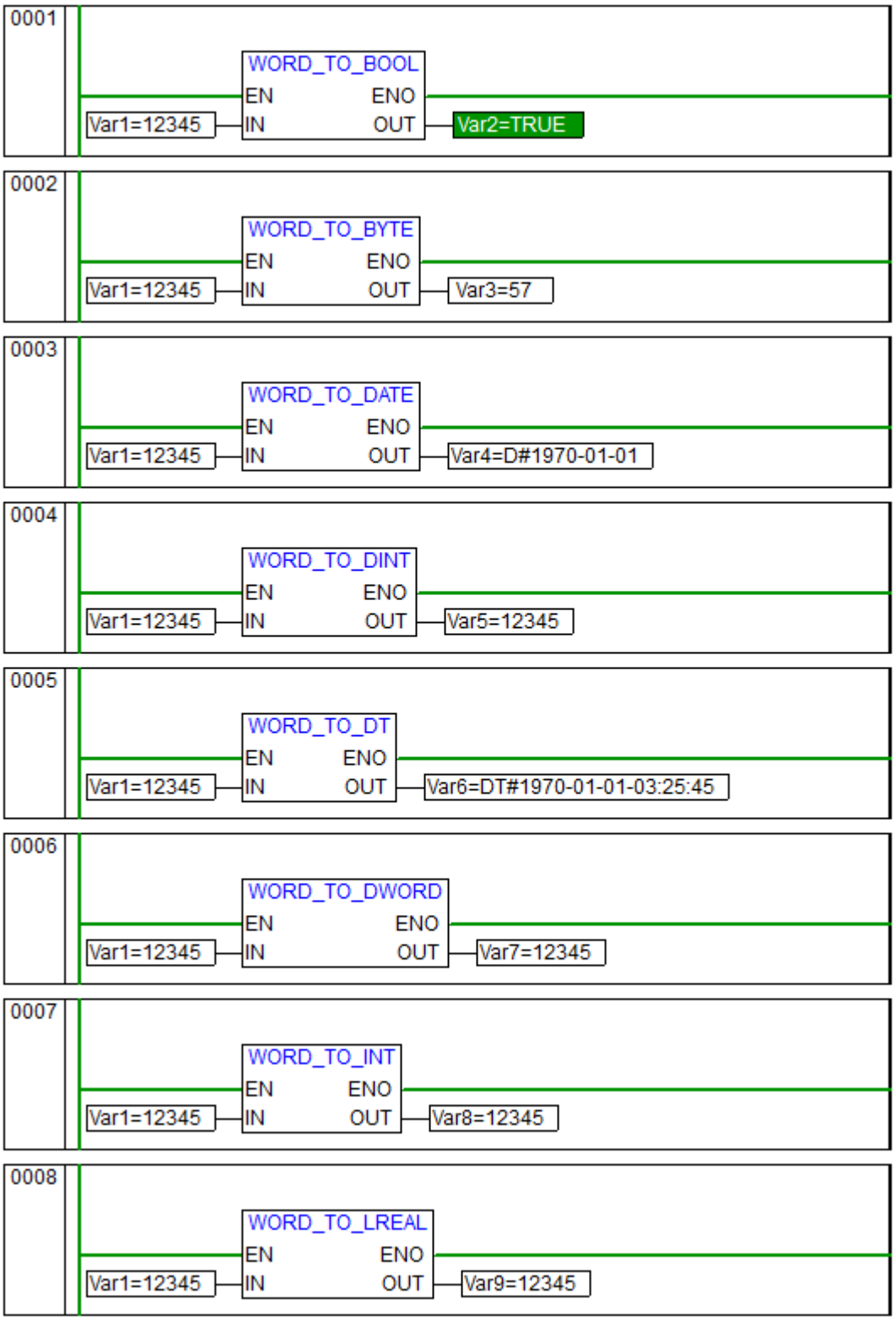
Parameter	Statement	Data Type	Description
IN	Input	WORD	Enter the word variable to be converted
OUT	Output	BOOL, BYTE, DATE, DINT, DT, DWORD, INT, LREAL, REAL, SINT, STRING, TIME, TOD, UDINT, UINT, USINT	Converted data type

5.6.17.3 Example

The following example uses the WORD conversion instruction to convert WORD variables to other data types. When the input EN detects a rising edge, the WORD conversion instruction starts the conversion function and stores the calculation result in the output parameter OUT.

Main(PRG).ld

No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			WORD	12345	FALSE	Not Public	
0002	Var2			BOOL	TRUE	FALSE	Not Public	
0003	Var3			BYTE	57	FALSE	Not Public	
0004	Var4			DATE	D#1970-01-01	FALSE	Not Public	
0005	Var5			DINT	12345	FALSE	Not Public	
0006	Var6			DT	DT#1970-01-...	FALSE	Not Public	
0007	Var7			DWORD	12345	FALSE	Not Public	
0008	Var8			INT	12345	FALSE	Not Public	
0009	Var9			LREAL	12345	FALSE	Not Public	
0010	Var10			REAL	12345	FALSE	Not Public	
0011	Var11			SINT	57	FALSE	Not Public	
0012	Var12			STRING(80)	'12345'	FALSE	Not Public	
0013	Var13			TIME	T#20s0MS	FALSE	Not Public	
0014	Var14			TOD	TOD#00:00:12	FALSE	Not Public	
0015	Var15			UDINT	12345	FALSE	Not Public	
0016	Var16			UINT	12345	FALSE	Not Public	
0017	Var17			UDINT	57	FALSE	Not Public	


0001
WORD_TO_BOOL
EN
ENO
Var1=12345
IN
OUT
Var2=TRUE
0002
WORD_TO_BYTE
EN
ENO
Var1=12345
IN
OUT
Var3=57
0003
WORD_TO_DATE
EN
ENO
Var1=12345
IN
OUT
Var4=D#1970-01-01
0004
WORD_TO_DINT
EN
ENO
Var1=12345
IN
OUT
Var5=12345
0005
WORD_TO_DT
EN
ENO
Var1=12345
IN
OUT
Var6=DT#1970-01-01-03:25:45
0006
WORD_TO_DWORD
EN
ENO
Var1=12345
IN
OUT
Var7=12345
0007
WORD_TO_INT
EN
ENO
Var1=12345
IN
OUT
Var8=12345
0008
WORD_TO_LREAL
EN
ENO
Var1=12345
IN
OUT
Var9=12345

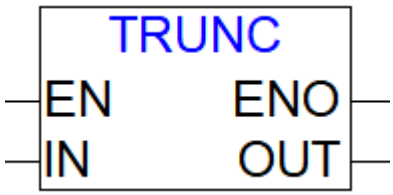
```
1   Var2 := WORD_TO_BOOL(Var1);
2   Var3 := WORD_TO_BYTE(Var1);
3   Var4 := WORD_TO_DATE(Var1);
4   Var5 := WORD_TO_DINT(Var1);
5   Var6 := WORD_TO_DT(Var1);
6   Var7 := WORD_TO_DWORD(Var1);
7   Var8 := WORD_TO_INT(Var1);
8   Var9 := WORD_TO_LREAL(Var1);
9   Var10 := WORD_TO_REAL(Var1);
10  Var11 := WORD_TO_SINT(Var1);
11  Var12 := WORD_TO_STRING(Var1);
12  Var13 := WORD_TO_TIME(Var1);
13  Var14 := WORD_TO_TOD(Var1);
14  Var15 := WORD_TO_UDINT(Var1);
15  Var16 := WORD_TO_UINT(Var1);
16  Var17 := WORD_TO_USINT(Var1);
```

```
Var2 = TRUE                    Var1 = 12345
Var3 = 57                      Var1 = 12345
Var4 = D#1970-01-01            Var1 = 12345
Var5 = 12345                   Var1 = 12345
Var6 = DT#1970-01-01-03:25:45  Var1 = 12345
Var7 = 12345                   Var1 = 12345
Var8 = 12345                   Var1 = 12345
Var9 = 12345                   Var1 = 12345
Var10 = 12345                  Var1 = 12345
Var11 = 57                     Var1 = 12345
Var12 = '12345'                Var1 = 12345
Var13 = T#12s345ms             Var1 = 12345
Var14 = TOD#00:00:12.345       Var1 = 12345
Var15 = 12345                  Var1 = 12345
Var16 = 12345                  Var1 = 12345
Var17 = 57                     Var1 = 12345
```

5.6.18 TRUNC (Truncate)

5.6.18.1 Function

Implement the trunc function.



TRUNC Functional Block

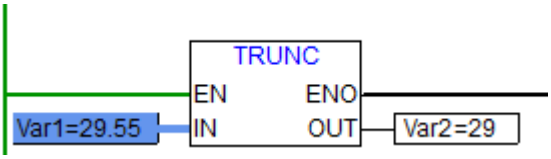
5.6.18.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
IN	Input	REAL	Enter the floating point variable to be rounded	0	FALSE
OUT	Output	DINT	Rounded integer	0	FALSE

5.6.18.3 Example

The following example uses the TRUNC instruction to round floating-point variables into integers. When the input EN detects a rising edge, the TRUNC instruction starts the trunc function and stores the calculation result in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			REAL	29.55	FALSE	Not Public	
0002	Var2			DINT	29	FALSE	Not Public	




```
1  Var2 :=TRUNC(Var1);
```

Var2 = 29

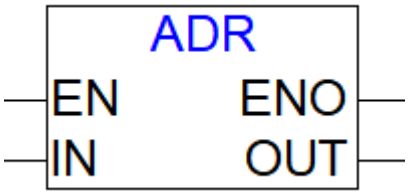
Var1 = 29.55

5.7 Pointer

5.7.1 ADR (Address)

5.7.1.1 Function

It is to get and output the memory address of the input variable. This address can be used as a pointer within the program or passed to a function as a pointer.



ADR Functional Block

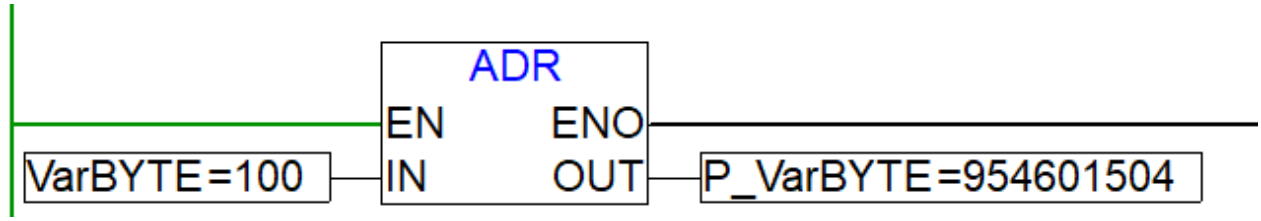
5.7.1.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
IN	Input	POINTER TO BYTE	Enter a variable to get its address	0	FALSE
OUT	Output	BYTE	Pointer variable, used to store the address of the input variable in memory	0	FALSE

5.7.1.3 Example

For example, the following example uses the ADR instruction to obtain the memory address of the IN input variable, and the result is output through OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0004	P_VarBYTE			POINTER TO BYTE	954601504	FALSE	Not Public	
0005	VarBYTE			BYTE	100	FALSE	Not Public	

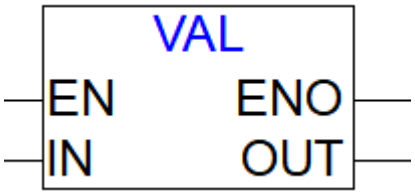



```
1  P_VarBYTE := ADR(VarBYTE);      | P_VarBYTE = 954601504      VarBYTE = 100
```

5.7.2 VAL (Value)

5.7.2.1 Function

Get the data at the address pointed by the pointer.



VAL Functional Block

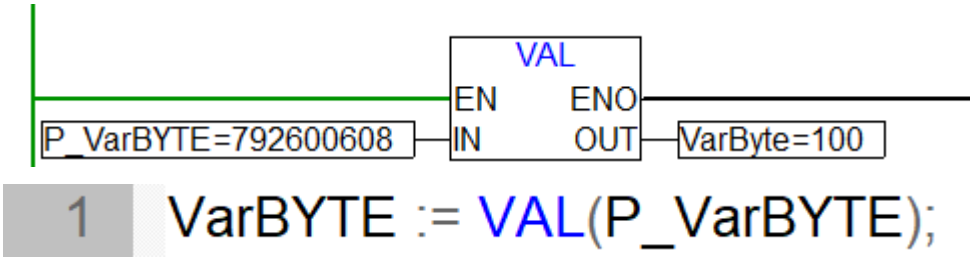
5.7.2.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
IN	Input	POINTER TO BYTE	Pointer variable	0	FALSE
OUT	Output	BYTE	Value of the address pointed to by the pointer	0	FALSE

5.7.2.3 Example

For example, the following example uses the VAL instruction to obtain the value of the address pointed to by the pointer.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	P_VarBYTE			POINTER TO BYTE	792600608	FALSE	Not Public	
0002	VarByte			BYTE	100	FALSE	Not Public	

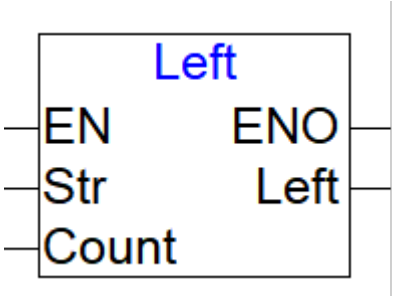


5.8 String

5.8.1 Left (Gets the string to the left of the string)

5.8.1.1 Function

It is to return the substring consisting of the leftmost count characters of the specified string.



Left Functional Block

5.8.1.2 Parameter

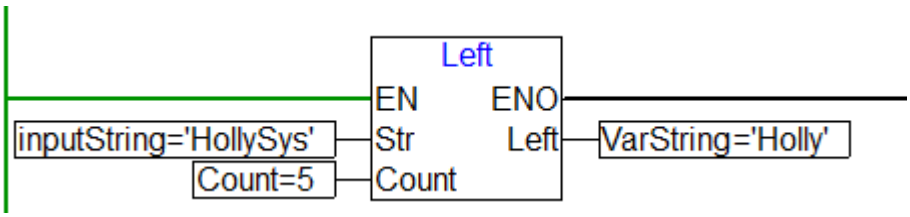
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
Str	Input	STRING	Input string, with a maximum length of 128		FALSE
Count	Input	BYTE, WORD, DWORD, SINT, USINT, INT, UINTNT and UDINT	Length of the substring to return	0	FALSE
Left	Return	STRING	Return substring. (1) If Str is empty, return empty; (2) If count is smaller than or equal to 0, return empty; (3) If the actual length of str is greater		FALSE

			than 128, take the first 128 characters; (4) If count is greater than or equal to the length of str, then return str;		

5.8.1.3 Example

For example, starting from the first character on the left, a partial string of STRING data type is extracted from the string.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	inputString			STRING(128)	'HollySys'	FALSE	Not Public	
0002	Count			DINT	5	FALSE	Not Public	
0003	VarString			STRING(128)	'Holly'	FALSE	Not Public	

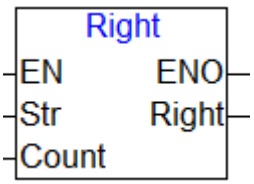



```
1  VarString := Left(inputString, Count);
```
VarString = 'Holly' inputString = 'HollySys' Count = 5

5.8.2 Right (Gets the string to the right of the string)

5.8.2.1 Function

It is to return the substring consisting of the rightmost count characters of the specified string.



Right Functional Block

5.8.2.2 Parameter

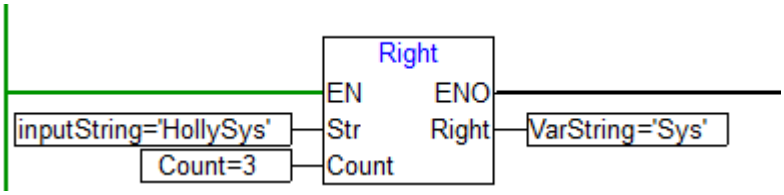
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
Str	Input	STRING	Input string, with a maximum length of 128		FALSE
Count	Input	BYTE, WORD, DWORD, SINT, USINT, INT, UINTNT and	Length of the substring to return	0	FALSE

		UDINT			
Right	Return	STRING	Return substring. (1) If Str is empty, return empty; (2) If count is smaller than or equal to 0, return empty; (3) If the actual length of str is greater than 128, take the first 128 characters; (4) If count is greater than or equal to the length of str, then return str;		FALSE

5.8.2.3 Example

For example, starting from the first character on the right, a partial string of STRING data type is extracted from the string.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	inputString			STRING(128)	'HollySys'	FALSE	Not Public	
0002	Count			DINT	3	FALSE	Not Public	
0003	VarString			STRING(128)	'Sys'	FALSE	Not Public	

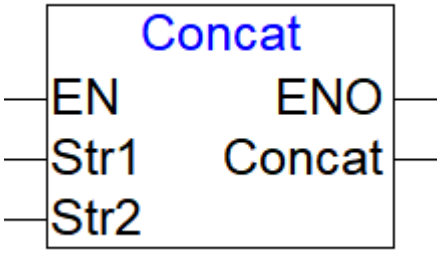



```
1  VarString := Right(inputString, Count);   VarString = 'HollySys'   inputString = 'Sys'   Count = 3
```

5.8.3 Concat (String concatenation)

5.8.3.1 Function

Concatenate two strings.



Concat Functional Block

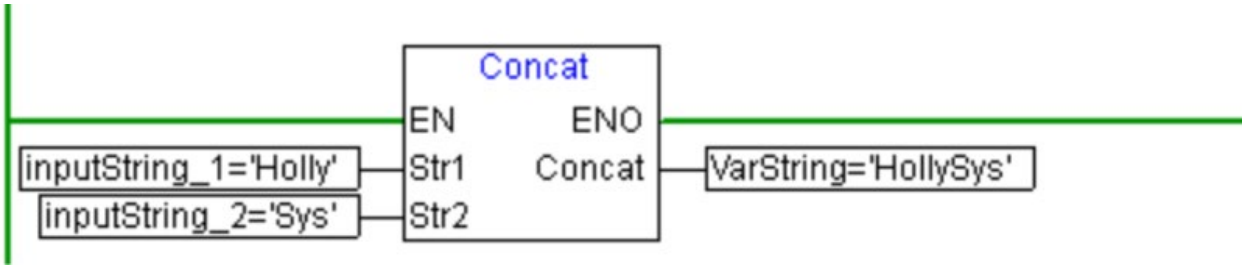
5.8.3.2 Parameter

Parameter	Statement	Data Type	Description	Power Fail Safeguard
EN	Input	BOOL	Enable input	
ENO	Output	BOOL	Enable output	
Str1	Input	STRING	Input string, with a maximum length of 128	FALSE
Str2	Input	STRING	Input string, with a maximum length of 128	FALSE
Concat	Return	STRING	Return substring. (1) When the length of the input string variable is greater than 128, take the first 128 characters; (2) If the sum of length of str1 and str2 is greater than 128, return empty.	FALSE

5.8.3.3 Example

In the following example, two strings of the STRING data type are concatenated together.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	inputString_1			STRING(128)	'Holly'	FALSE	Not Public	
0002	inputString_2			STRING(128)	'Sys'	FALSE	Not Public	
0003	VarString			STRING(128)	'HollySys'	FALSE	Not Public	

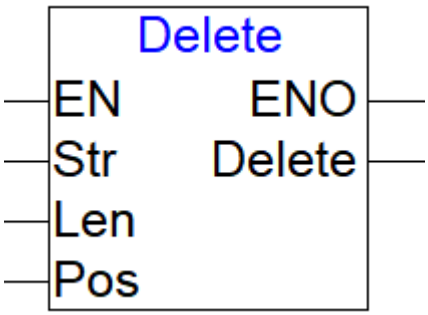



```
1  VarString := Concat(inputString_1, inputString_2);   VarString = 'HollySys'   inputString_1 = 'Holly'   inputString_2 = 'Sys'
```

5.8.4 Delete (String remove)

5.8.4.1 Function

Delete len characters starting from the pos character of str, and return the result.



Delete Functional Block

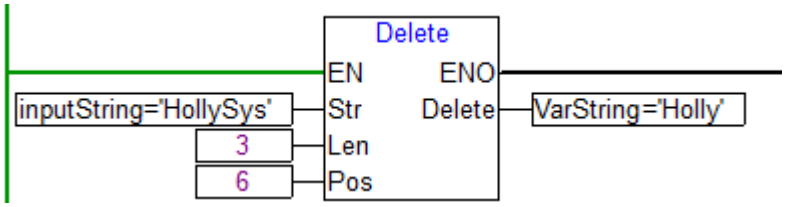
5.8.4.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
Str	Input	STRING	Input string, with a maximum length of 128		FALSE
Len	Input	BYTE, WORD, SINT, USINT, INT, UINT	Number of characters to delete	0	FALSE
Pos	Input	BYTE, WORD, SINT, USINT, INT, UINT	Starting position of the character to be deleted in the str string	0	FALSE
Delete	Return	STRING	(1) If the actual length of str is greater than 128, take the first 128 characters; (2) If len is smaller than 0, then pos=pos+len, len=-len; (3) If pos is smaller than or equal to 0, then len=len+pos-1, pos=1; (4) If pos+len is greater than the length of str, then len = str length-pos+1; (5) If pos is greater than the length of str or len is smaller than or equal to 0, then str is returned (if its length is greater than 128, the first 128 characters are returned)		FALSE

5.8.4.3 Example

In the following example, 3 characters starting from the 6th character of str are deleted.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	inputString			STRING(128)	'HollySys'	FALSE	Not Public	
0002	VarString			STRING(128)	'Holly'	FALSE	Not Public	



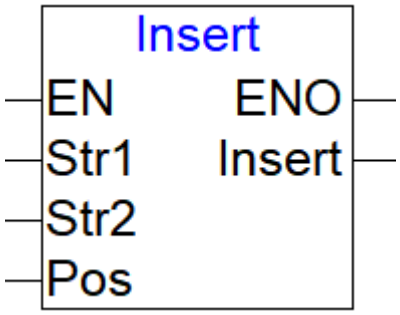

```
1  VarString := Delete(inputString, 3, 6);
```

VarString = 'HollySys' inputString = 'Holly'

5.8.5 Insert (String insert)

5.8.5.1 Function

Insert str2 into a position after the pos character of str1, and return the result.



Insert Functional Block

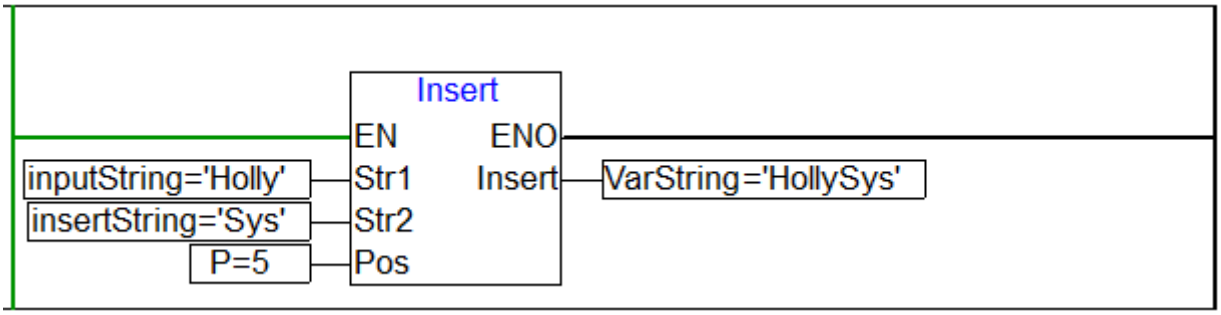
5.8.5.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
Str1	Input	STRING	Input string		FALSE
Str2	Input	STRING	Inserted string		FALSE
Pos	Input	BYTE, WORD, SINT, USINT, INT, UINT	Insertion position	0	FALSE
Insert	Return	STRING	Generated string		FALSE

5.8.5.3 Example

In the following example, one string is inserted into another string.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	inputString			STRING(128)	'Holly'	FALSE	Not Public	
0002	insetString			STRING(128)	'Sys'	FALSE	Not Public	
0003	P			INT	5	FALSE	Not Public	
0004	VarString			STRING(128)	'HollySys'	FALSE	Not Public	

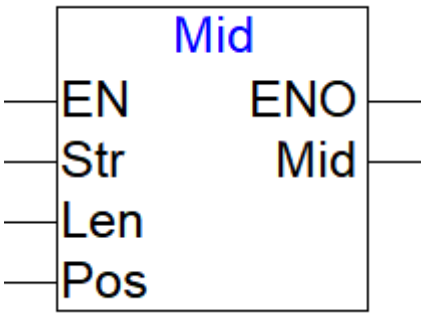



```
VarString := Insert(inputString, insertString, P);
```

5.8.6 Mid (Gets the string to the middle of the string)

5.8.6.1 Function

Return the substring consisting of len consecutive characters starting from the pos character in the string.



Mid Functional Block

5.8.6.2 Parameter

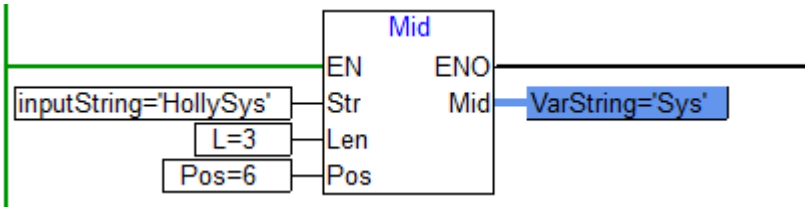
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
Str	Input	STRING	Input string		FALSE
Len	Input	BYTE, WORD, SINT, USINT, INT, UINT	String length to extract	0	FALSE
Pos	Input	BYTE, WORD, SINT, USINT, INT, UINT	Position of the string to extract	0	FALSE

Mid	Return	STRING	Extracted string		FALSE

5.8.6.3 Example

In the following example, a partial string of the STRING data type is extracted starting from the middle of the string.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	inputString			STRING(128)	'HollySys'	FALSE	Not Public	
0002	L			INT	3	FALSE	Not Public	
0003	Pos			INT	6	FALSE	Not Public	
0004	VarString			STRING(128)	'Sys'	FALSE	Not Public	

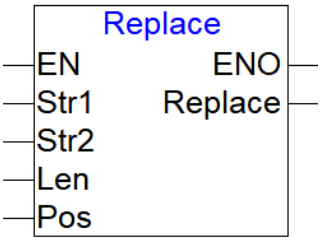



```
1  VarString := Mid(inputString, L, Pos);
```
VarString = 'Sys' inputString = 'HollySys' L = 3 Pos = 6

5.8.7 Replace (String replace)

5.8.7.1 Function

Use str2 to replace len characters starting from the pos character in str1, and return the result.



Replace Functional Block

5.8.7.2 Parameter

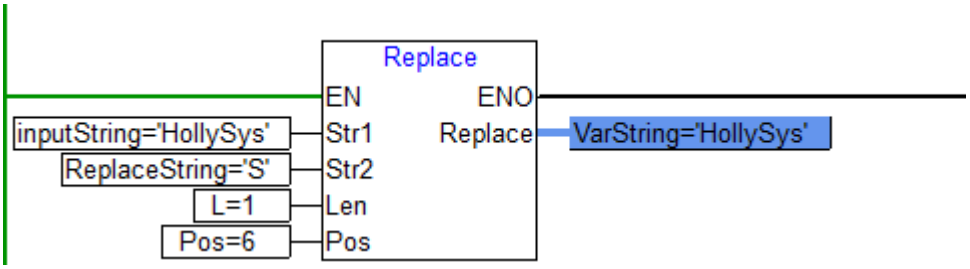
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
Str1	Input	STRING	String whose characters are to be replaced		FALSE
Str2	Input	STRING	String containing the characters to be inserted		FALSE

Pos	Input	BYTE, WORD, SINT, USINT, INT, UINT	Position of the first character to replace	0	FALSE
Len	Input	BYTE, WORD, SINT, USINT, INT, UINT	Number of characters to replace	0	FALSE
Replace	Return	STRING	Generated string		FALSE

5.8.7.3 Example

In the following example, part of a string is replaced with another string.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	inputString			STRING(128)	'HollySys'	FALSE	Not Public	
0002	ReplaceString			STRING(80)	'S'	FALSE	Not Public	
0003	L			INT	1	FALSE	Not Public	
0004	Pos			INT	6	FALSE	Not Public	
0005	VarString			STRING(128)	'HollySys'	FALSE	Not Public	

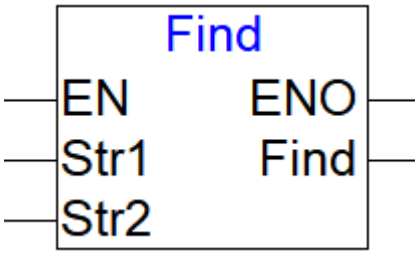



```
1  VarString := Replace(inputString, ReplaceString, L, Pos);
```

5.8.8 Find (String find)

5.8.8.1 Function

Find the first character position of str2 in str1, and return the result.



Find Functional Block

5.8.8.2 Parameter

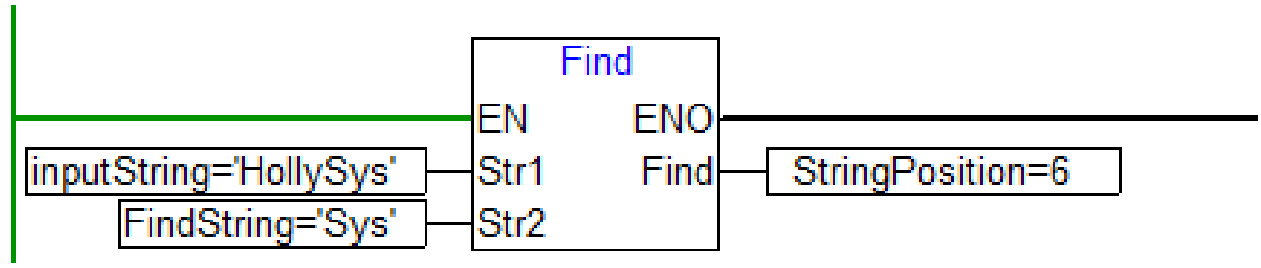
Parameter	Statement	Data Type	Description	Power Fail Safeguard

EN	Input	BOOL	Enable input	
ENO	Output	BOOL	Enable output	
Str1	Input	STRING	String being searched	FALSE
Str2	Input	STRING	String to be searched	FALSE
Find	Return	WORD, DWORD, INT, UINT, DINT, UDINT	Character position	FALSE

5.8.8.3 Example

In the following example, one string is searched in another string.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	inputString			STRING(128)	'HollySys'	FALSE	Not Public	
0002	FindString			STRING(128)	'Sys'	FALSE	Not Public	
0003	StringPosition			INT	6	FALSE	Not Public	

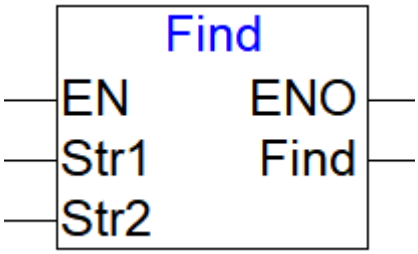



```
1  StringPosition := Find(inputString, FindString);   StringPosition = 6        inputString = 'HollySys'        FindString = 'Sys'
```

5.8.9 Len (Gets the length of the string)

5.8.9.1 Function

Return the length of the input string.



Len Functional Block

5.8.9.2 Parameter

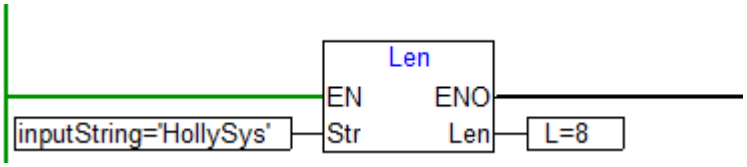
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		

ENO	Output	BOOL	Enable output		
Str	Input	STRING	Input string		FALSE
Len	Return	WORD, DWORD, INT, UINT, DINT, UDINT	Return the length of the input string.	0	FALSE

5.8.9.3 Example

In the following example, the length of the input string in Str1 is determined.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	inputString			STRING(128)	'HollySys'	FALSE	Not Public	
0002	L			INT	8	FALSE	Not Public	



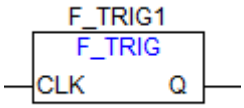

```
1  L := Len(inputString);  L = 8                    inputString = 'HollySys'
```

5.9 Edge Trigger Instruction

5.9.1 F_TRIG (Falling Edge Detection Trigger)

5.9.1.1 Function

Used to detect falling edges.



F_TRIG Functional Block

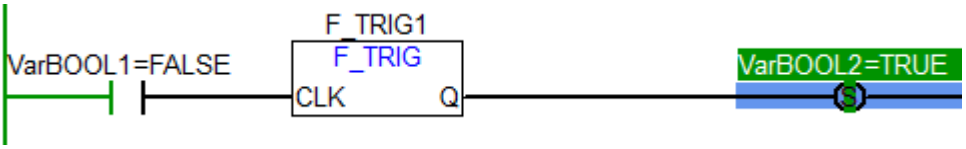
5.9.1.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
CLK	Input	BOOL	Falling edge trigger signal	TRUE	FALSE
Q	Output	BOOL	If a falling edge is detected, Q=TRUE when Q=TRUE calls the instruction every time.	FALSE	FALSE

5.9.1.3 Example

For example, the following example uses the F_TRIG instruction to detect the change of input CLK from "1" to "0".

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	F_TRIG1			F_TRIG		FALSE	Not Public	
0002	VarBOOL1			BOOL	FALSE	FALSE	Not Public	
0003	VarBOOL2			BOOL	TRUE	FALSE	Not Public	



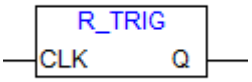

```
1  F_TRIG1(
2  CLK := VarBOOL1,
3  Q=>VarBOOL2);
```

```
F_TRIG1
VarBOOL1 = FALSE
VarBOOL2 = TRUE
```

5.9.2 R_TRIG (Rising Edge Detection Trigger)

5.9.2.1 Function

It is used to detect rising edges.



R_TRIG Functional Block

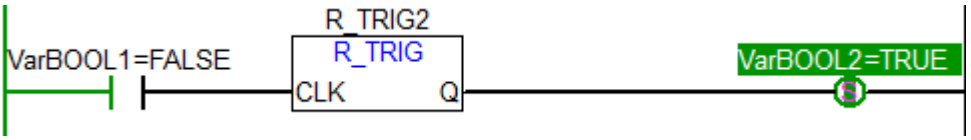
5.9.2.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
CLK	Input	BOOL	Rising edge trigger signal	FALSE	FALSE
Q	Output	BOOL	If a rising edge is detected, Q=TRUE when Q=TRUE calls the instruction every time.	FALSE	FALSE

5.9.2.3 Example

For example, the following example uses the R_TRIG instruction to detect the change of the input CLK from "0" to "1".

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	R_TRIG2			R_TRIG		FALSE	Not Public	
0002	VarBOOL1			BOOL	FALSE	FALSE	Not Public	
0003	VarBOOL2			BOOL	TRUE	FALSE	Not Public	



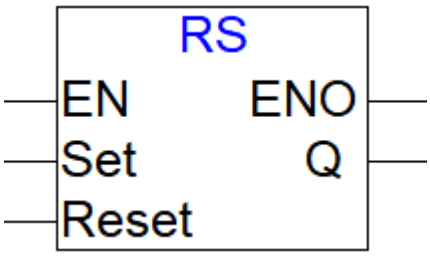

```
1  R_TRIG1(
2  CLK := VarBOOL1,
3  Q=>VarBOOL2);
```

5.10 Bistable Instruction

5.10.1 RS (Reset Priority Bistable)

5.10.1.1 Function

The reset bistable function algorithm (RS) implements the reset-priority RS trigger function.



RS Functional Block

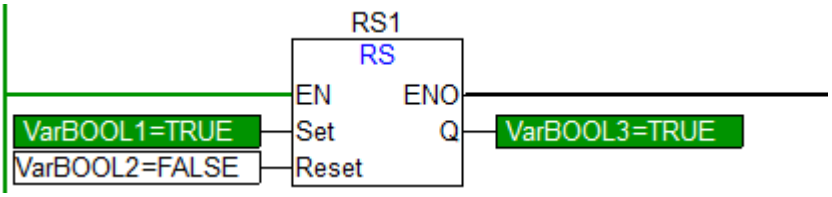
5.10.1.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
SET	Input	BOOL	Setting	FALSE	FALSE
RESET	Input	BOOL	Reset	FALSE	FALSE
Q	Output	BOOL	Signal status of the operand	FALSE	FALSE

5.10.1.3 Example

For example, the following example uses the RS instruction to implement the reset-priority RS trigger function. When VarBOOL1 is "1" and VarBOOL2 is "0", set VarBOOL3.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	RS1			RS		FALSE	Not Public	
0002	VarBOOL3			BOOL	TRUE	FALSE	Not Public	
0003	VarBOOL1			BOOL	TRUE	FALSE	Not Public	
0004	VarBOOL2			BOOL	FALSE	FALSE	Not Public	

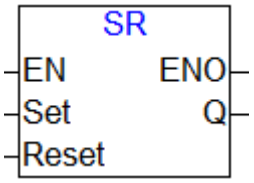



```
1  RS1(
2  Set := VarBOOL1,
3  Reset := VarBOOL2,
4  Q=>VarBOOL3);
5
```

5.10.2 SR (Setting Priority Bistable)

5.10.2.1 Function

The bistable function algorithm (SR) implements the setting-priority RS trigger function.



SR Functional Block

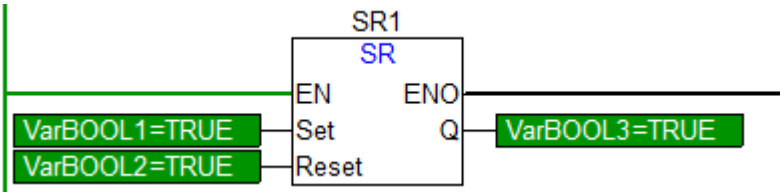
5.10.2.2 Parameter

Parameter	Statement	Data Type	Description
EN	Input	BOOL	Enable input
ENO	Output	BOOL	Enable output
SET	Input	BOOL	Setting
RESET	Input	BOOL	Reset
Q	Output	BOOL	Signal status of the operand

5.10.2.3 Example

For example, the following example uses the SR instruction to implement the setting priority SR trigger function. When VarBOOL1 is "1" and VarBOOL2 is "1", set VarBOOL3 to 1.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	SR1			SR		FALSE	Not Public	
0002	VarBOOL1			BOOL	TRUE	FALSE	Not Public	
0003	VarBOOL2			BOOL	TRUE	FALSE	Not Public	
0004	VarBOOL3			BOOL	TRUE	FALSE	Not Public	



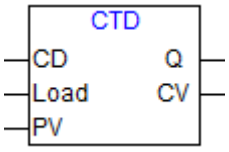

```
1  SR1(
2  Set := VarBOOL1,
3  Reset := VarBOOL2,
4  Q=>VarBOOL3);
5
```

5.11 Counter Instruction

5.11.1 CTD (Decrement Counter)

5.11.1.1 Function

The count-down algorithm (CTD) implements the down-counting function when there is a rising edge on the input CD.



CTD Functional Block

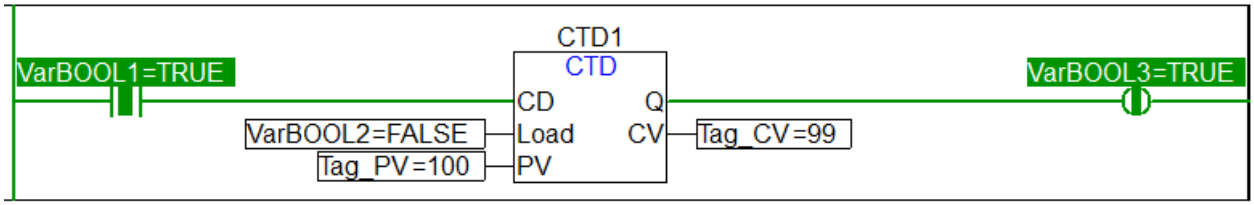
5.11.1.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
CD	Input	BOOL	Count input	FALSE	FALSE
Load	Input	BOOL	Load input	FALSE	FALSE
PV	Input	WORD	Set the target value of the output CV using LD = 1	0	FALSE
Q	Output	BOOL	Counter status	TRUE	FALSE
CV	Output	WORD	Current counter value	0	FALSE

5.11.1.3 Example

For example, the following example uses the CTD instruction to implement incremental counting. When the input CD detects a rising edge, the CTD instruction starts the down-counting function and returns the current value through CV.

No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
0002	VarBOOL1			BOOL	TRUE	FALSE	Not Public	
0003	VarBOOL2			BOOL	TRUE	FALSE	Not Public	
0004	Tag_PV			WORD	100	FALSE	Not Public	
0005	VarBOOL3			BOOL	FALSE	FALSE	Not Public	
0006	Tag_CV			WORD	99	FALSE	Not Public	



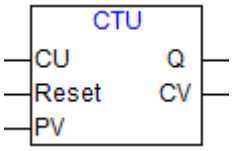

```
1  CTD1(
2  CD := VarBOOL1,
3  Load := VarBOOL2,
4  PV := Tag_PV,
5  Q=>VarBOOL3,
6  CV=>Tag_CV);
```

```
CTD1
VarBOOL1 = TRUE
VarBOOL2 = FALSE
Tag_PV = 100
VarBOOL3 = TRUE
Tag_CV = 99
```

5.11.2 CTU (Increment Counter)

5.11.2.1 Function

The count-up algorithm (CTU) implements the up-counting function when there is a rising edge on the input CU.



CTU Functional Block

5.11.2.2 Parameter

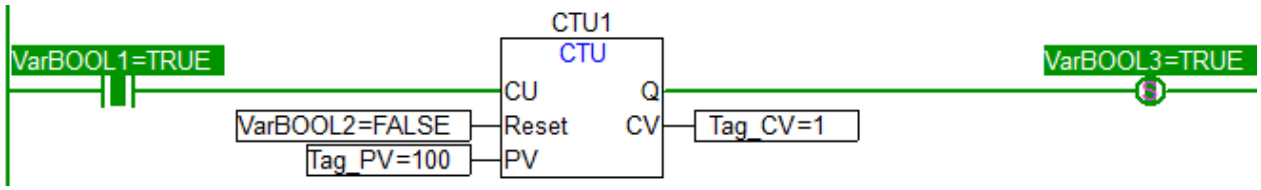
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
CU	Input	BOOL	Count input	FALSE	FALSE
Reset	Input	BOOL	Reset input	FALSE	FALSE

PV	Input	WORD	Set value	0	FALSE
Q	Output	BOOL	Counter status	TRUE	FALSE
CV	Output	WORD	Current counter value	0	FALSE

5.11.2.3 Example

For example, the following example uses the CTU instruction to implement incremental counting. When the input CU detects a rising edge, the CTU instruction starts the up-counting function and returns the current value through CV. When CV is greater than or equal to the set value PV, the output Q returns TRUE.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	CTU1			CTU		FALSE	Not Public	
0002	VarBOOL1			BOOL	TRUE	FALSE	Not Public	
0003	VarBOOL2			BOOL	FALSE	FALSE	Not Public	
0004	VarBOOL3			BOOL	TRUE	FALSE	Not Public	
0005	Tag_PV			WORD	100	FALSE	Not Public	
0006	Tag_CV			WORD	1	FALSE	Not Public	



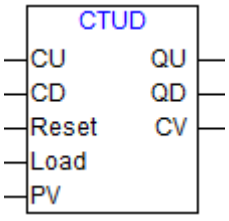

```
1  CTU1(
2  CU := VarBOOL1,
3  Reset := VarBOOL2,
4  PV := Tag_PV,
5  Q=>VarBOOL3,
6  CV=>Tag_CV);
```

```
CTU1
VarBOOL1 = TRUE
VarBOOL2 = FALSE
Tag_PV = 100
VarBOOL3 = TRUE
Tag_CV = 1
```

5.11.3 CTUD (Increment/ Decrement Counter)

5.11.3.1 Function

The count-up algorithm (CTU) implements the up-counting function when there is a rising edge on the input CU.



CTUD Instruction Schematic Diagram

5.11.3.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
CU	Input	BOOL	Rising edge detection terminal 1	FALSE	FALSE
CD	Input	BOOL	Rising edge detection terminal 2	FALSE	FALSE
Reset	Input	BOOL	Reset	FALSE	FALSE
Load	Input	BOOL	When LOAD is TRUE, CV is initialized to PV.	FALSE	FALSE
PV	Input	WORD	Set value	0	FALSE
QU	Output	BOOL	When CV=PV, QU=TRUE	TRUE	FALSE
QD	Output	BOOL	When CV=0, QD=TRUE	TRUE	FALSE
CV	Output	WORD	Count output	0	FALSE

5.11.3.3 Example

(1) Initialization Function

When RESET is set to 1, the count output CV is initialized to 0. When LOAD is set to 1, the count output CV is initialized to PV.

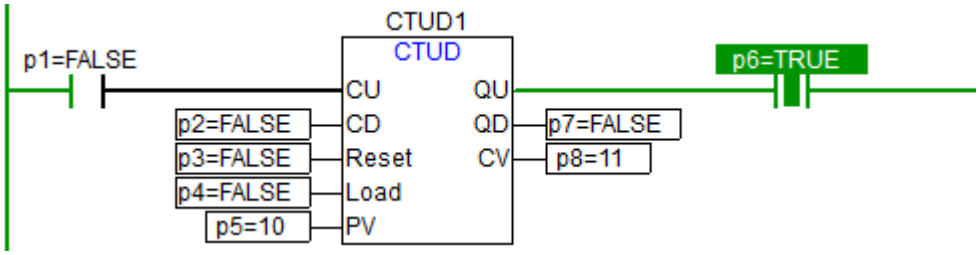
(2) Counting Function

If there is a rising edge on CU (transition from FALSE to TRUE), CV is incremented by 1. If there is a rising edge on CD (transition from FALSE to TRUE), CV is decremented by 1 (assuming it will not make CV less than 0).

When CV is greater than or equal to PV, QU returns TRUE.

When CV equals 0, QD returns TRUE.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	CTUD1			CTUD		FALSE	Not Public	
0002	p1			BOOL	FALSE	FALSE	Not Public	
0003	p2			BOOL	FALSE	FALSE	Not Public	
0004	p3			BOOL	FALSE	FALSE	Not Public	
0005	p4			BOOL	FALSE	FALSE	Not Public	
0006	p5			WORD	10	FALSE	Not Public	
0007	p6			BOOL	TRUE	FALSE	Not Public	
0008	p7			BOOL	FALSE	FALSE	Not Public	
0009	p8			WORD	11	FALSE	Not Public	




```
1  CTUD2(
2  CU := p1,
3  CD := p2,
4  Reset := p3,
5  Load := p4,
6  PV := p5,
7  QU=>p6,
8  QD=>p7,
9  CV=>p8);
```

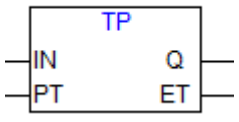
```
CTUD2
p1 = FALSE
p2 = FALSE
p3 = FALSE
p4 = FALSE
p5 = 10
p6 = TRUE
p7 = FALSE
p8 = 11
```

5.12 Timer Instruction

5.12.1 TP (Normal Timer)

5.12.1.1 Function

By using the TP instruction, the output Q can be set as a preset period of time.



TP Functional Block

5.12.1.2 Parameter

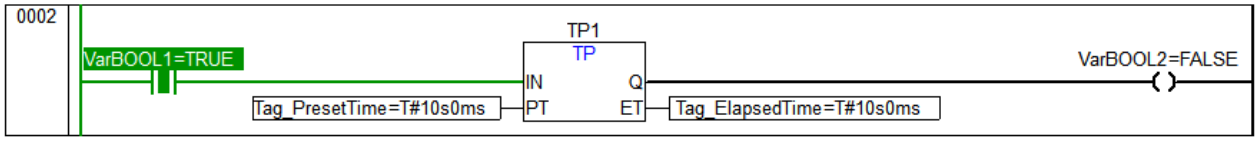
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
IN	Input	BOOL	Start input	FALSE	FALSE

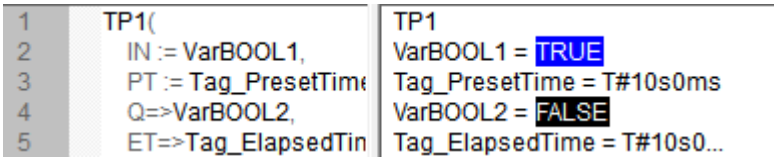
PT	Input	TIME	Pulse duration	T#0S	FALSE
Q	Output	BOOL	Pulse output	FALSE	FALSE
ET	Output	TIME	Current time value	T#0S	FALSE

5.12.1.3 Example

For example, the following example uses the TP instruction to implement timing. When IN changes from "0" to "1", the timing starts, the Q output is "True", When the time reaches the preset value of PT, the Q output is "False".

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	TP1			TP		FALSE	Not Public	
0002	VarBOOL1			BOOL	TRUE	FALSE	Not Public	
0003	VarBOOL2			BOOL	FALSE	FALSE	Not Public	
0004	Tag_PresetTime			TIME	T#10s0ms	FALSE	Not Public	
0005	Tag_ElapsedTime			TIME	T#10s0ms	FALSE	Not Public	

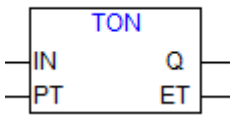




5.12.2 TON (On Delay Timer)

5.12.2.1 Function

By using the TON instruction, the setting of the Q output can be delayed for a period of time specified in PT.



TON Functional Block

5.12.2.2 Parameter

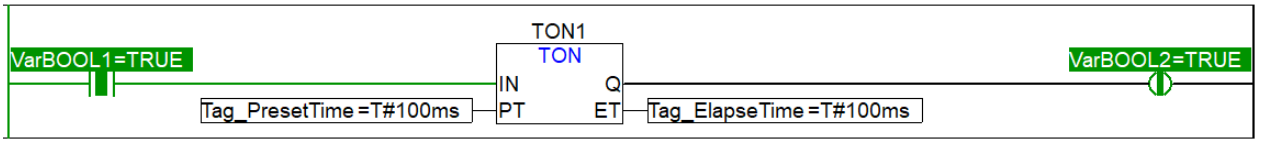
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
IN	Input	BOOL	Start input	FALSE	FALSE
PT	Input	TIME	On-delay duration	T#0S	FALSE

Q	Output	BOOL	Set output after the time PT has elapsed	FALSE	FALSE
ET	Output	TIME	Current time value	T#0S	FALSE

5.12.2.3 Example

For example, the following example uses the TON instruction to implement the on-delay setting function. When IN changes from "0" to "1", the timing starts. When the time reaches the preset value of PV, the Q output is "True". Operand VarBOOL2 remains set to "1" as long as the signal status of operand VarBOOL1 is "1".

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	TON1			TON		FALSE	Not Public	
0002	VarBOOL1			BOOL	TRUE	FALSE	Not Public	
0003	VarBOOL2			BOOL	TRUE	FALSE	Not Public	
0004	Tag_PresetTime			TIME	T#100ms	FALSE	Not Public	
0005	Tag_ElapseTime			TIME	T#100ms	FALSE	Not Public	



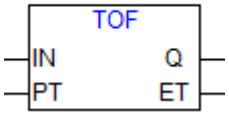

```
1  TON1(
2  IN := Var_BOOL1,
3  PT := Tag_PresetTime,
4  Q=>Var_BOOL2,
5  ET=>Tag_ElapseTime);
```

```
TON1
Var_BOOL1 = TRUE
Tag_PresetTime = T#100ms
Var_BOOL2 = TRUE
Tag_ElapseTime = T#100ms
```

5.12.3 TOF (Off Delay Timer)

5.12.3.1 Function

By using the TOF instruction, the Q output can be reset for a preset period of time PT.



TOF Functional Block

5.12.3.2 Parameter

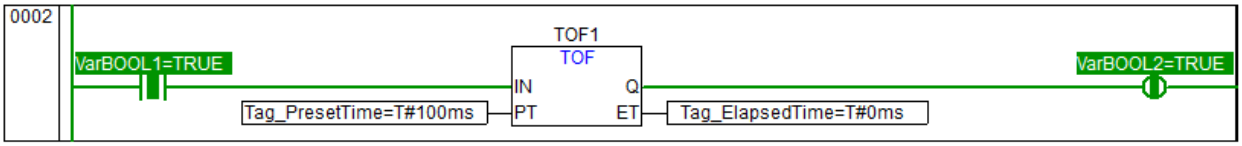
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
IN	Input	BOOL	Start input	FALSE	FALSE
PT	Input	TIME	On-delay duration	T#0S	FALSE
Q	Output	BOOL	Set output after the time PT has elapsed	FALSE	FALSE

ET	Output	TIME	Current time value	T#0S	FALSE

5.12.3.3 Example

For example, the following example uses the TOF instruction to implement the off-delay setting function. When IN changes from "0" to "1", the signal status of operand VarBOOL2 is set to "1". When the signal status of operand VarBOOL1 changes from "1" to "0", the time preset by the PT parameter starts counting. Operand VarBOOL2 remains set to TRUE as long as the time is ticking. After this time has elapsed, operand VarBOOL2 is reset to FALSE. The current time value is stored in operand Tag_ElapsedTime.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	TOF1			TOF		FALSE	Not Public	
0002	VarBOOL1			BOOL	TRUE	FALSE	Not Public	
0003	Tag_PresetTime			TIME	T#100ms	FALSE	Not Public	
0004	VarBOOL2			BOOL	TRUE	FALSE	Not Public	
0005	Tag_ElapsedTime			TIME	T#0ms	FALSE	Not Public	



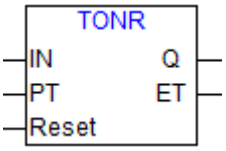

```
10  TOF1(
11  IN := VarBOOL1,
12  PT := Tag_PresetTime,
13  Q=>VarBOOL2,
14  ET=>Tag_ElapsedTime);
```

```
TOF1
VarBOOL1 = TRUE
Tag_PresetTime = T#100ms
VarBOOL2 = TRUE
Tag_ElapsedTime = T#0ms
```

5.12.4 TONR (Retentive On Delay Timer)

5.12.4.1 Function

By using the TONR instruction, the time value within the time period set by parameter PT can be accumulated.



TONR Functional Block

5.12.4.2 Parameter

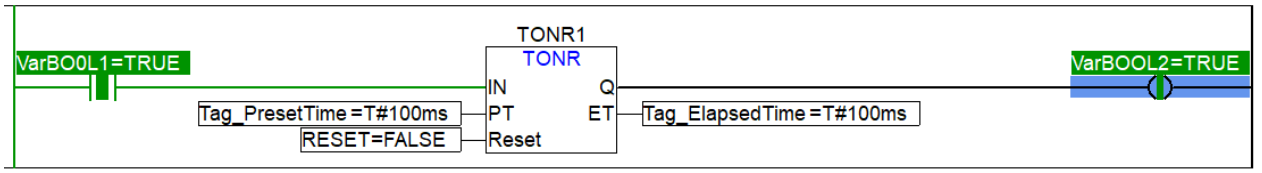
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard

IN	Input	BOOL	Start input	FALSE	FALSE
R	Input	BOOL	Reset input	FALSE	FALSE
PT	Input	TIME	On-delay duration	T#0MS	FALSE
Q	Output	BOOL	Set output after the time PT has elapsed	FALSE	FALSE
ET	Output	TIME	Accumulated time value	T#0MS	FALSE

5.12.4.3 Example

For example, the following example uses the TONR instruction to implement time accumulation. When the signal status of operand VarBOOL1 changes from "0" to "1", the time preset by the PT parameter starts counting. This time continues as long as the signal status of operand VarBOOL1 is "1". When the signal status of operand VarBOOL1 changes from "1" to "0", the timing is stopped and the current time value in operand Tag_ElapsedTime is recorded.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	TONR1			TONR		FALSE	Not Public	
0002	VarBOOL1			BOOL	TRUE	FALSE	Not Public	
0003	VarBOOL2			BOOL	TRUE	FALSE	Not Public	
0004	Tag_PresetTime			TIME	T#100ms	FALSE	Not Public	
0005	RESET			BOOL	FALSE	FALSE	Not Public	
0006	Tag_ElapsedTime			TIME	T#100ms	FALSE	Not Public	




```
10  TONR1(
11  IN := VarBO0L1,
12  PT := Tag_PresetTime,
13  Reset := RESET,
14  Q=>VarBOOL2,
15  ET=>Tag_ElapsedTime);
```

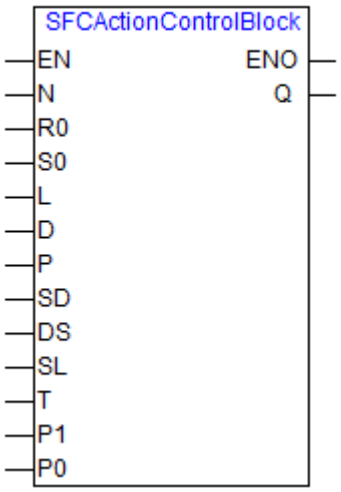
```
TONR1
VarBO0L1 = TRUE
Tag_PresetTime = T#100ms
RESET = FALSE
VarBOOL2 = TRUE
Tag_ElapsedTime = T#100ms
```

5.13 IEC_SFC

5.13.1 SFCActionControlBlock(SFC Action Control Block)

5.13.1.1 Function

This instruction realizes the control of IEC step related actions in the SFC language.



IEC Action Instruction

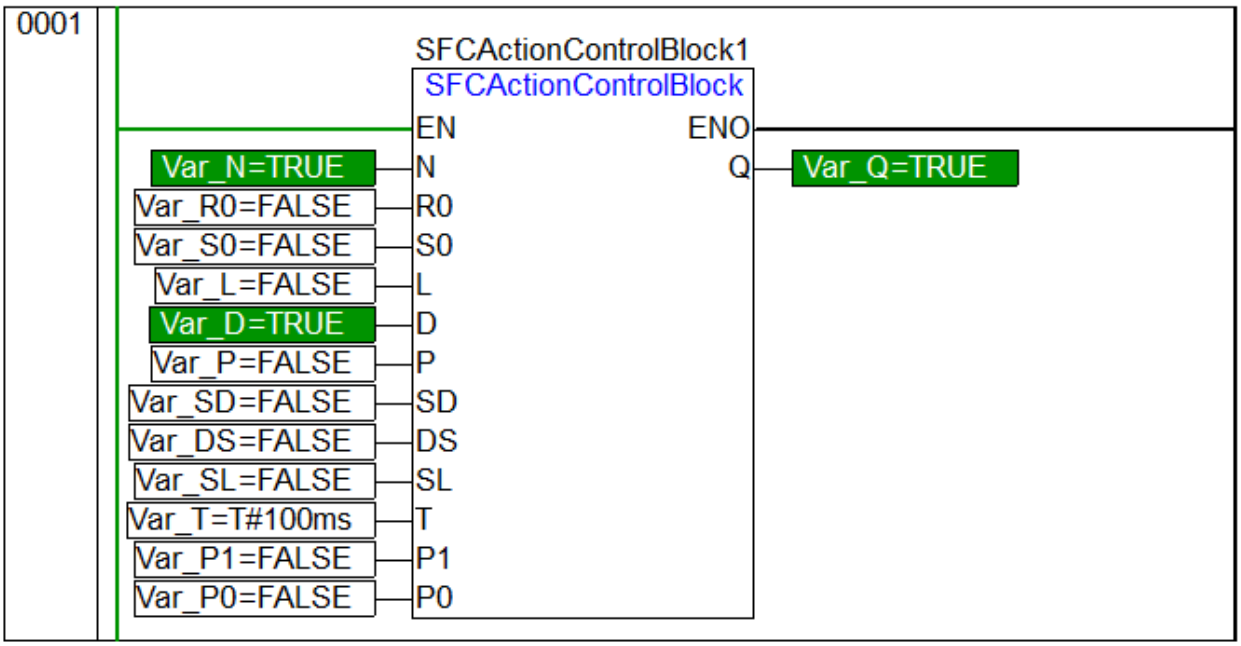
- When the IEC step is used in the SFC programming language, this library must be added, otherwise the compilation prompts an error. When the IEC step has associated actions, the system automatically calls the SFCActionControl instruction.

5.13.1.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
N	Input	BOOL	Non stored	FALSE	FALSE
R0	Input	BOOL	Storage reset	FALSE	FALSE
S0	Input	BOOL	Set stored	FALSE	FALSE
L	Input	BOOL	Time limited	FALSE	FALSE
D	Input	BOOL	Time delayed	FALSE	FALSE
P	Input	BOOL	Pulse	FALSE	FALSE
SD	Input	BOOL	Stored and time delayed	FALSE	FALSE
DS	Input	BOOL	Delayed and stored	FALSE	FALSE
SL	Input	BOOL	Stored and time limited	FALSE	FALSE
T	Input	TIME	Current time	T#0MS	FALSE
P1	Input	BOOL	Rise edge pulse	FALSE	FALSE
P0	Input	BOOL	Descending edge pulse	FALSE	FALSE
Q	Output	BOOL	Associated action is executed, if Q equals TRUE	FALSE	FALSE

5.13.1.3 Example

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	SFCActionControlBlock1			SFCACTIONCONTROL...		FALSE	Not Public	
0002	Var_N			BOOL	TRUE	FALSE	Not Public	
0003	Var_R0			BOOL	FALSE	FALSE	Not Public	
0004	Var_S0			BOOL	FALSE	FALSE	Not Public	
0005	Var_L			BOOL	FALSE	FALSE	Not Public	
0006	Var_D			BOOL	TRUE	FALSE	Not Public	
0007	Var_P			BOOL	FALSE	FALSE	Not Public	
0008	Var_SD			BOOL	FALSE	FALSE	Not Public	
0009	Var_DS			BOOL	FALSE	FALSE	Not Public	
0010	Var_SL			BOOL	FALSE	FALSE	Not Public	
0011	Var_T			TIME	T#100ms	FALSE	Not Public	
0012	Var_P1			BOOL	FALSE	FALSE	Not Public	
0013	Var_P0			BOOL	FALSE	FALSE	Not Public	
0014	Var_Q			BOOL	TRUE	FALSE	Not Public	



```
1   SFCActionControlBlock1(
2   N := Var_N,
3   R0 := Var_R0,
4   S0 := Var_S0,
5   L := Var_L,
6   D := Var_D,
7   P := Var_P,
8   SD := Var_SD,
9   DS := Var_DS,
10  SL := Var_SL,
11  T := Var_T,
12  P1 := Var_P1,
13  P0 := Var_P0,
14  Q=>Var_Q);
```

```
SFCActionControlBlock1
Var_N = TRUE
Var_R0 = FALSE
Var_S0 = FALSE
Var_L = FALSE
Var_D = TRUE
Var_P = FALSE
Var_SD = FALSE
Var_DS = FALSE
Var_SL = FALSE
Var_T = T#100ms
Var_P1 = FALSE
Var_P0 = FALSE
Var_Q = TRUE
```

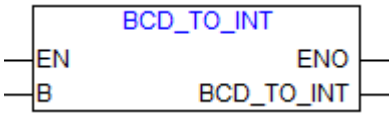
Chapter 6 Basic Application Library

6.1 BCD Conversion Functions

6.1.1 BCD_TO_INT (BCD Code to Integer)

6.1.1.1 Function

Convert BCD code to an integer value algorithm.



BCD_TO_INT Functional Block

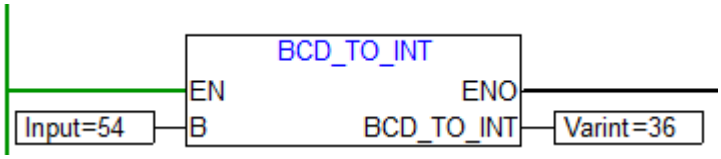
6.1.1.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
Input	Input	BYTE	To convert the BCD code, enter the binary form of the BCD code (or the decimal and hexadecimal code corresponding to the binary code). For example, the BCD code 49 is expressed as 2#100 1001 (or corresponding to 10#73 and 16#49), then enter 2#100 1001 here (or 10#73 or 16#49).	0	FALSE
Varint	Output	INT	For the converted INT value, if the input byte is not BCD code, the output is -1.	0	FALSE

6.1.1.3 Example

The BCD code to integer conversion algorithm (BCD_TO_INT) converts the variable Input whose input value is the BCD code into the variable Varint whose output is the INT type.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Input			BYTE	54	FALSE	Not Public	
0002	Varint			INT	36	FALSE	Not Public	

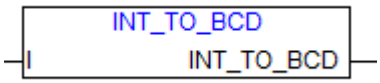



```
1  Varint := BCD_TO_INT(Input);
```

6.1.2 INT_TO_BCD (Integer to BCD Code)

6.1.2.1 Function

Convert an integer value to BCD code.



INT_TO_BCD Functional Block

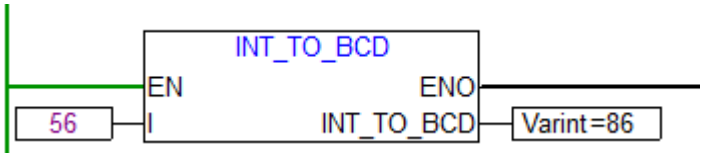
6.1.2.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
I	Input	INT	If the integer data value to be converted is 49, enter the integer data 49 here	0	FALSE
INT_TO_BCD	Output	BYTE	Converted BCD code For example, if 49 is converted into BCD code 2#100 1001, then 2#1001001 is output here (or 10#73 or 16#49 corresponding to the binary value)	0	FALSE

6.1.2.3 Example

The integer to BCD code value algorithm (INT_TO_BCD) converts the input value, which is an INT type constant 56, into the variable Varint whose output is the BCD code.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Varint			INT	86	FALSE	Not Public	



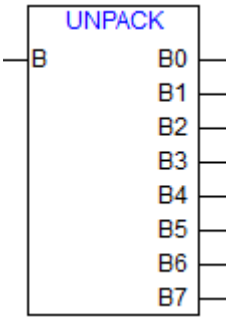

```
2  Varint := INT_TO_BCD(56);
```

6.2 Bit_byte Functions

6.2.1 UNPACK (Bit Split)

6.2.1.1 Function

It is to take out the 7th to 0th bits of the input B and assign them to B7~B0. It is opposite to the PACK instruction.



UNPACK Functional Block

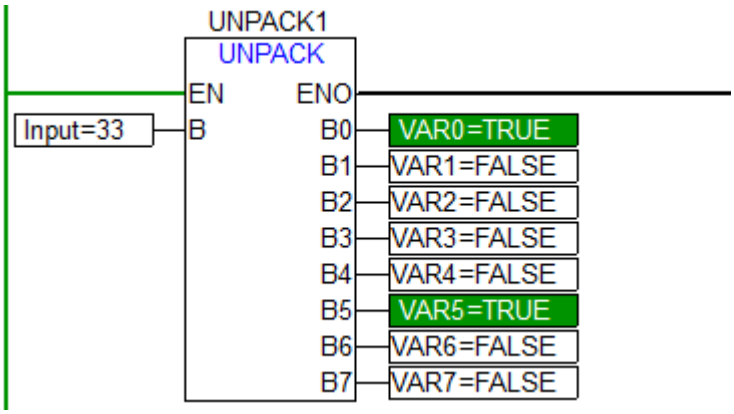
6.2.1.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
B	Input	BYTE	BYTE type input variable	0	FALSE
B0	Output	BOOL	BOOL type output variable	FALSE	FALSE
B1	Output	BOOL	BOOL type output variable	FALSE	FALSE
B2	Output	BOOL	BOOL type output variable	FALSE	FALSE
B3	Output	BOOL	BOOL type output variable	FALSE	FALSE
B4	Output	BOOL	BOOL type output variable	FALSE	FALSE
B5	Output	BOOL	BOOL type output variable	FALSE	FALSE
B6	Output	BOOL	BOOL type output variable	FALSE	FALSE
B7	Output	BOOL	BOOL type output variable	FALSE	FALSE

6.2.1.3 Example

The UNPACK instruction takes out bits 7~0 of the input and assigns them to B7~B0.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	UNPACK1			UNPACK		FALSE	Not Public	
0002	Input			BYTE	33	FALSE	Not Public	
0003	VAR0			BOOL	TRUE	FALSE	Not Public	
0004	VAR1			BOOL	FALSE	FALSE	Not Public	
0005	VAR2			BOOL	FALSE	FALSE	Not Public	
0006	VAR3			BOOL	FALSE	FALSE	Not Public	
0007	VAR4			BOOL	FALSE	FALSE	Not Public	
0008	VAR5			BOOL	TRUE	FALSE	Not Public	
0009	VAR6			BOOL	FALSE	FALSE	Not Public	
0010	VAR7			BOOL	FALSE	FALSE	Not Public	



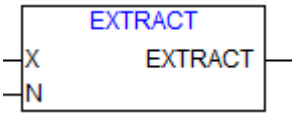

```
1  UNPACK1(
2  B := Input,
3  B0=>VAR0,
4  B1=>VAR1,
5  B2=>VAR2,
6  B3=>VAR3,
7  B4=>VAR4,
8  B5=>VAR5,
9  B6=>VAR6,
10 B7=>VAR7);
```

```
UNPACK1
Input = 33
VAR0 = TRUE
VAR1 = FALSE
VAR2 = FALSE
VAR3 = FALSE
VAR4 = FALSE
VAR5 = TRUE
VAR6 = FALSE
VAR7 = FALSE
```

6.2.2 EXTRACT (Bit Extraction)

6.2.2.1 Function

Output the specified Nth bit of input variable X.



EXTRACT Functional Block

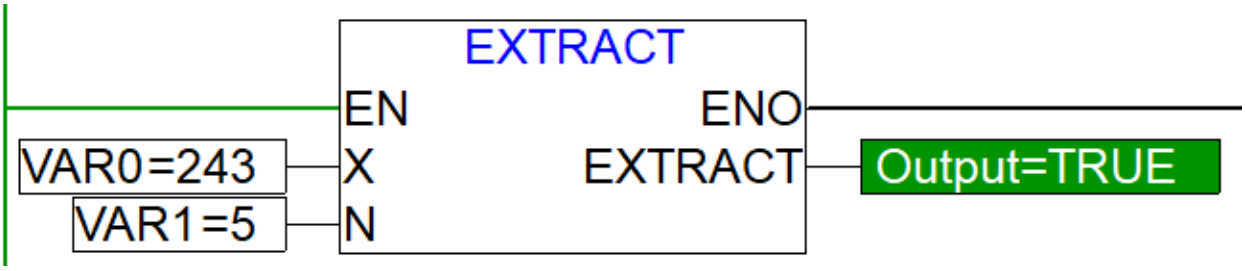
6.2.2.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
X	Input	DWORD	Enter the manipulated variable	0	FALSE
N	Input	BYTE	Specify the input position	0	FALSE
EXTRACT	Output	BOOL	Nth bit status	FALSE	FALSE

6.2.2.3 Example

When the input EN detects a rising edge, the EXTRACT instruction starts the bit extraction function and stores the result in the output parameter EXTRACT.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	VAR0			DWORD	243	FALSE	Not Public	
0002	VAR1			BYTE	5	FALSE	Not Public	
0003	Output			BOOL	TRUE	FALSE	Not Public	



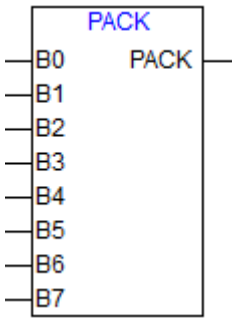

```
1  Output := EXTRACT(VAR0, VAR1);
```

Output = TRUE VAR0 = 243 VAR1 = 5

6.2.3 PACK (Bit Merger)

6.2.3.1 Function

Combine B0~B7 into a BYTE variable in order from bit 0 to bit 7. The corresponding instruction to this instruction is UNPACK.



PACK Functional Block

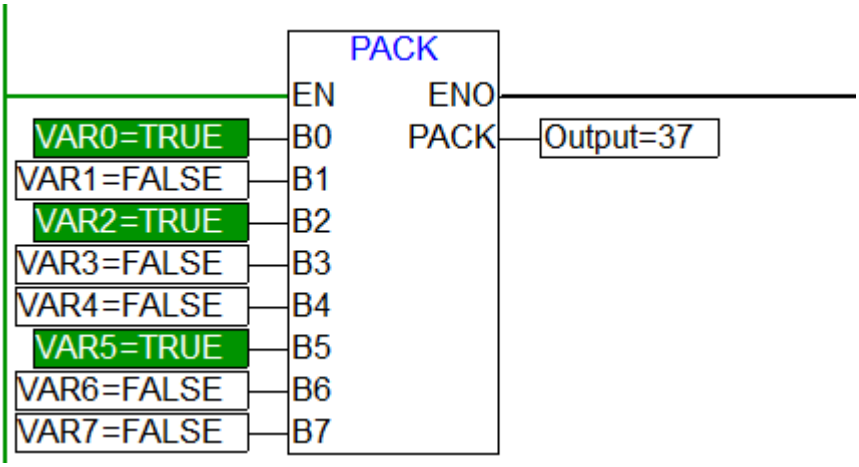
6.2.3.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
VAR0	Input	BOOL	Bit 0 data	FALSE	FALSE
VAR1	Input	BOOL	Bit 1 data	FALSE	FALSE
VAR2	Input	BOOL	Bit 2 data	FALSE	FALSE
VAR3	Input	BOOL	Bit 3 data	FALSE	FALSE
VAR4	Input	BOOL	Bit 4 data	FALSE	FALSE
VAR5	Input	BOOL	Bit 5 data	FALSE	FALSE
VAR6	Input	BOOL	Bit 6 data	FALSE	FALSE
VAR7	Input	BOOL	Bit 7 data	FALSE	FALSE
Output	Output	BYTE	BYTEs from high to low: B7~B0	0	FALSE

6.2.3.3 Example

The PACK instruction combines B0~B7 into a BYTE variable in order from bit 0 to bit 7.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	VAR0			BOOL	TRUE	FALSE	Not Public	
0002	VAR1			BOOL	FALSE	FALSE	Not Public	
0003	VAR2			BOOL	TRUE	FALSE	Not Public	
0004	VAR3			BOOL	FALSE	FALSE	Not Public	
0005	VAR4			BOOL	FALSE	FALSE	Not Public	
0006	VAR5			BOOL	TRUE	FALSE	Not Public	
0007	VAR6			BOOL	FALSE	FALSE	Not Public	
0008	VAR7			BOOL	FALSE	FALSE	Not Public	
0009	Output			BYTE	37	FALSE	Not Public	



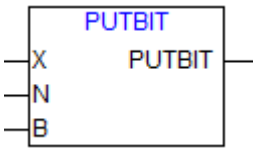

```
1  Output := PACK(VAR0, VAR1, VAR2, VAR3,
2  VAR4, VAR5, VAR6, VAR7);
```

Output = 37 VAR0 = TRUE VAR1 = FALSE VAR2 = TRUE
VAR4 = FALSE VAR5 = TRUE VAR6 = FALSE VAR7 = FALSE

6.2.4 PUTBIT (Bit Assignment)

6.2.4.1 Function

The algorithm (PUTBIT) of setting the specified bit of a variable to the specified value sets the Nth bit specified by the input value X to the specified value B.



PUTBIT Functional Block

6.2.4.2 Parameter

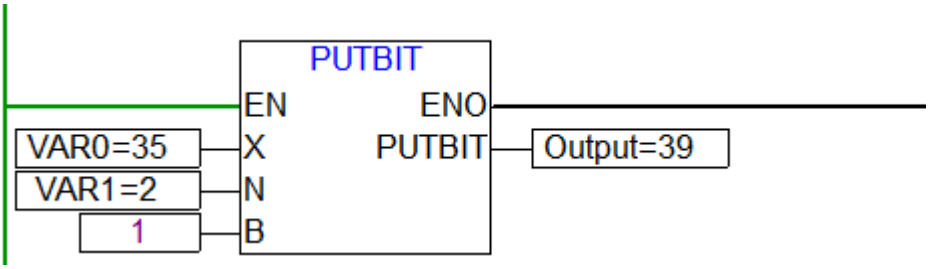
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
X	Input	DWORD	Manipulated data to be converted	0	FALSE
N	Input	BYTE	The Nth bit to be converted specified by the input	0	FALSE

			data		
B	Input	BOOL	Specified value to be replaced at the Nth bit	FALSE	FALSE
PUTBIT	Output	DWORD	Final output result of the converted data	0	FALSE

6.2.4.3 Example

When the input EN detects a rising edge, the PUTBIT instruction starts the bit assignment function and stores the calculation result in the output parameter PUTBIT.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	VAR0			DWORD	35	FALSE	Not Public	
0002	VAR1			BYTE	2	FALSE	Not Public	
0003	Output			DWORD	39	FALSE	Not Public	




```
1  Output := PUTBIT(VAR0, VAR1, 1);
```

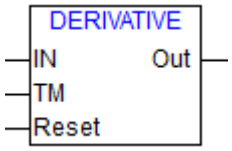
Output = 39 VAR0 = 35 VAR1 = 2

6.3 Mathematical Functions

6.3.1 DERIVATIVE (Derivative Function)

6.3.1.1 Function

This instruction performs differential operations on continuous input variables.



DERIVATIVE Functional Block

6.3.1.2 Parameter

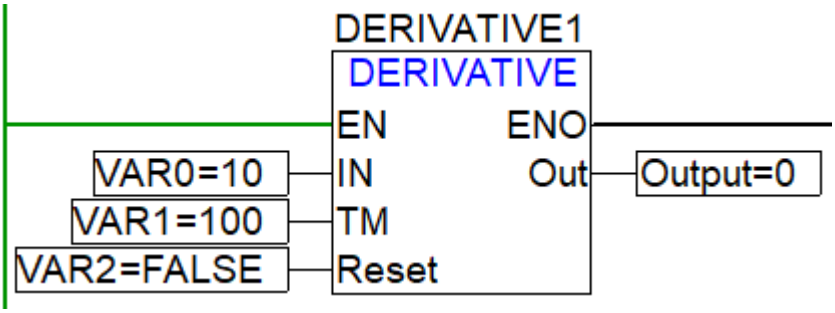
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
IN	Input	REAL	Differential instruction input data	0	FALSE
TM	Input	DWORD	Differential iteration formula time constant	0	FALSE
RESET	Input	BOOL	Instruction reset restart signal	FALSE	FALSE

Output	Output	REAL	Output data of differential algorithm	0	FALSE

6.3.1.3 Example

When the input EN detects a rising edge, the DERIVATIVE instruction starts the differential calculation function and stores the calculation result in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	DERIVATIVE1			DERIVATIVE		FALSE	Not Public	
0002	VAR0			REAL	10	FALSE	Not Public	
0003	VAR1			DWORD	100	FALSE	Not Public	
0004	VAR2			BOOL	FALSE	FALSE	Not Public	
0005	Output			REAL	0	FALSE	Not Public	



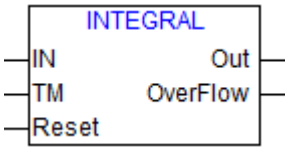

```
1  DERIVATIVE1(
2  IN := VAR0,
3  TM := VAR1,
4  Reset := VAR2,
5  Out=>OUTPUT);
```

```
DERIVATIVE1
VAR0 = 10
VAR1 = 100
VAR2 = FALSE
OUTPUT = 0
```

6.3.2 INTEGRAL (Integral Function)

6.3.2.1 Function

The integration algorithm (INTEGRAL) approximately solves the integral of the input function.



INTEGRAL Functional Block

6.3.2.2 Parameter

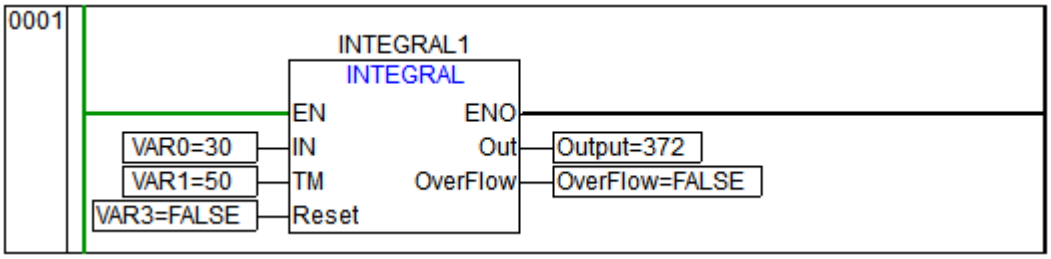
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
IN	Input	REAL	Integral instruction input data	0	FALSE

TM	Input	DWORD	Integral iteration formula time constant	0	FALSE
Reset	Input	BOOL	Instruction reset restart signal	FALSE	FALSE
Out	Output	REAL	Approximate integral	0	FALSE
OverFlow	Output	BOOL	Overflow flag	FALSE	FALSE

6.3.2.3 Example

When the input EN detects a rising edge, the INTEGRAL instruction starts the integral function and stores the calculation result in the output parameter OUT.

No.	Variable Name	Address	/ariable Descriptio	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	INTEGRAL1			INTEGRAL		FALSE	Not Public	
0005	Output			REAL	372	FALSE	Not Public	
0003	VAR1			DWORD	50	FALSE	Not Public	
0002	VAR0			REAL	30	FALSE	Not Public	
0006	OverFlow			BOOL	FALSE	FALSE	Not Public	
0004	VAR3			BOOL	FALSE	FALSE	Not Public	

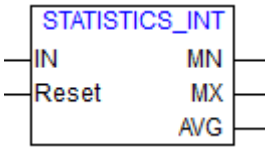



```
1  INTEGRAL1(                    INTEGRAL1
2    IN := VAR0,                 VAR0 = 30
3    TM := VAR1,                 VAR1 = 50
4    Reset := VAR3,              VAR3 = FALSE
5    Out=>Output,                Output = 372
6    OverFlow=>OverFlow);        OverFlow = FALSE
```

6.3.3 STATISTICS_INT (Int Statistics)

6.3.3.1 Function

The standard statistical algorithm (STATISTICS_INT) counts the minimum, maximum, and average values of all INT type input variables.



STATISTICS_INT Functional Block

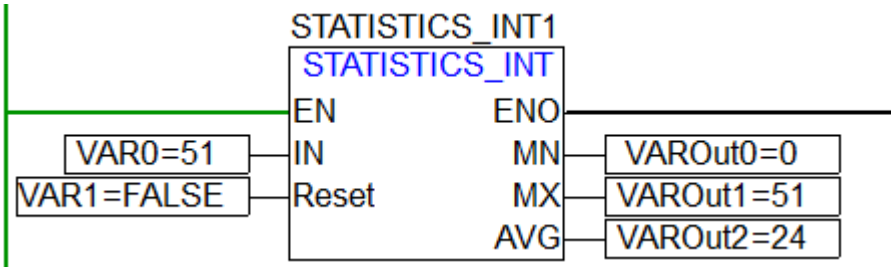
6.3.3.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
IN	Input	INT	Input operation data	0	FALSE
RESET	Input	BOOL	Instruction initialization signal	FALSE	FALSE
MN	Output	INT	The instruction operates on the input data, and the minimum value of the input data is output here	32767	FALSE
MX	Output	INT	The instruction operates on the input data, and the maximum value of the input data is output here	-32768	FALSE
AVG	Output	INT	The instruction operates on the input data, and the average value of the input data is output here	0	FALSE

6.3.3.3 Example

When the input EN detects a rising edge, the STATISTICS_INT instruction starts the integer statistics function and stores the calculation results in the corresponding output parameters MN, MX, and AVG.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	STATISTICS_INT1			STATISTICS_INT		FALSE	Not Public	
0002	VAR0			INT	51	FALSE	Not Public	
0003	VAR1			BOOL	FALSE	FALSE	Not Public	
0004	VAROut0			INT	0	FALSE	Not Public	
0005	VAROut1			INT	51	FALSE	Not Public	
0006	VAROut2			INT	24	FALSE	Not Public	



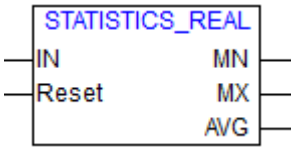

```
1  STATISTICS_INT1(
2  IN := VAR0,
3  Reset := VAR1,
4  MN=>VAROut0,
5  MX=>VAROut1,
6  AVG=>VAROut2);
```

```
STATISTICS_INT1
VAR0 = 51
VAR1 = FALSE
VAROut0 = 0
VAROut1 = 51
VAROut2 = 24
```

6.3.4 STATISTICS_REAL (Real Statistics)

6.3.4.1 Function

Count the maximum, minimum, and average values of all REAL type input variables.



STATISTICS_ REAL Functional Block

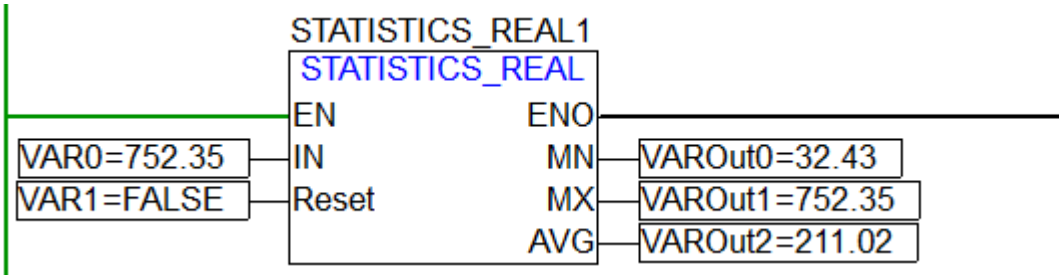
6.3.4.2 Parameter

Parameter	Statement	Data Type	Description	Default Value	Power Fail Safeguard
IN	Input	REAL	Input operation data	0	FALSE
Reset	Input	BOOL	Instruction initialization signal	FALSE	FALSE
MN	Output	REAL	The instruction operates on the input data, and the minimum value of the input data is output here	3E+38	FALSE
MX	Output	REAL	The instruction operates on the input data, and the maximum value of the input data is output here	1E-37	FALSE
AVG	Output	REAL	The instruction operates on the input data, and the average value of the input data is output here	0	FALSE

6.3.4.3 Example

When the input EN detects a rising edge, the STATISTICS_REAL instruction starts the integer statistics function and stores the calculation results in the corresponding output parameters MN, MX, and AVG.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	STATISTICS_RE...			STATISTICS_RE...		FALSE	Not Public	
0002	VAR0			REAL	752.35	FALSE	Not Public	
0003	VAR1			BOOL	FALSE	FALSE	Not Public	
0004	VAROut0			REAL	32.43	FALSE	Not Public	
0005	VAROut1			REAL	752.35	FALSE	Not Public	
0006	VAROut2			REAL	211.02	FALSE	Not Public	



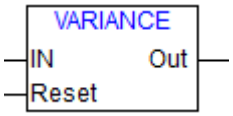

```
1  STATISTICS_REAL1(
2  IN := VAR0,
3  Reset := VAR1,
4  MN=>VAROut0,
5  MX=>VAROut1,
6  AVG=>VAROut2);
```

```
STATISTICS_REAL1
VAR0 = 752.35
VAR1 = FALSE
VAROut0 = 32.43
VAROut1 = 752.35
VAROut2 = 211.02
```

6.3.5 VARIANCE (Calculation of Variance)

6.3.5.1 Function

The algorithm for calculating the square deviation of input values (VARIANCE) calculates the square deviation of all input variables. The standard deviation can be found from the square root of the square deviation.



VARIANCE Functional Block

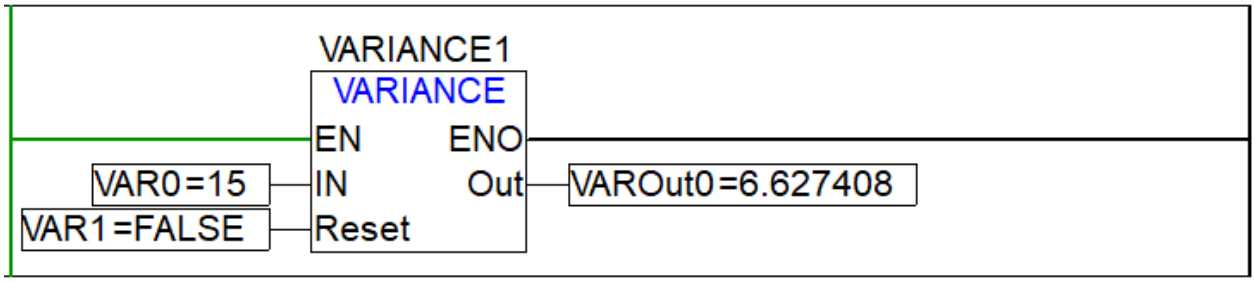
6.3.5.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
IN	Input	REAL	Enter the operand	0	FALSE
Reset	Input	BOOL	Instruction restart signal	FALSE	FALSE
Out	Output	REAL	Square deviation value output	0	FALSE

6.3.5.3 Example

When the input EN detects a rising edge, the VARIANCE instruction starts the square deviation function, calculates the squared difference of all input variables, and stores the calculation results in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	VARIANCE1			VARIANCE		FALSE	Not Public	
0002	VAR0			REAL	15	FALSE	Not Public	
0003	VAR1			BOOL	FALSE	FALSE	Not Public	
0004	VAROut0			REAL	6.627408	FALSE	Not Public	




```
1  VARIANCE1(
2  IN := VAR0,
3  Reset := VAR1,
4  Out=>VAROut0);
```

```
VARIANCE1
VAR0 = 15
VAR1 = FALSE
VAROut0 = 6.627408
```

6.4 Controller Functions

6.4.1 PID (PID Controller)

6.4.1.1 Function

1. Enable

   When the enabled terminal EN_R is true, the program runs normally, otherwise the program returns without operation.

2. Determining setting values

   - The output upper and lower limits MVMax and MVMin are set to determine that the output lower limit must be smaller than the output upper limit, otherwise the over limit and under limit flags of the output value are set, and the program returns without operation.
   - Settings and judgment of integral time Ti, differential time Td and dead band range DeadBand

     The integral time Ti, differential time Td and dead band range DeadBand must be non-negative

numbers, otherwise the operation will be based on the initial values.

3. Calculating

□ Manual mode

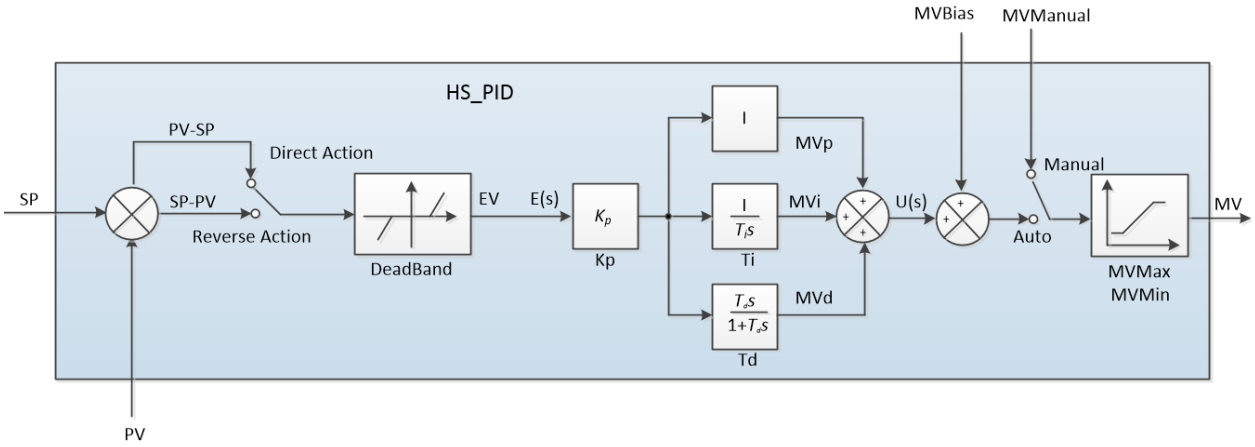
The output MV is equal to the manual input value MVManual, and the SP tracks the PV.

□ Auto mode

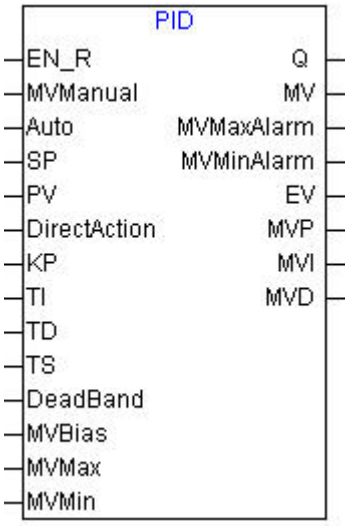
It calculates the deviation based on the direct/reverse action mode DerectAction, set value SP, measured value PV and dead band range DeadBand. Then it calculates the integral saturation coefficient based on the output MV, output upper and lower limits, and whether the deviation is positive or negative.

It calculates the proportional component MVp, integral component MVi, differential component MVd, and control output MV of the control output based on the proportional gain Kp, integral time Ti, differential time Td, and integral saturation coefficient.

The logic structure of the PID functional block is shown in the figure.



Logic Structure Diagram of PID Functional Block



PID Functional Block

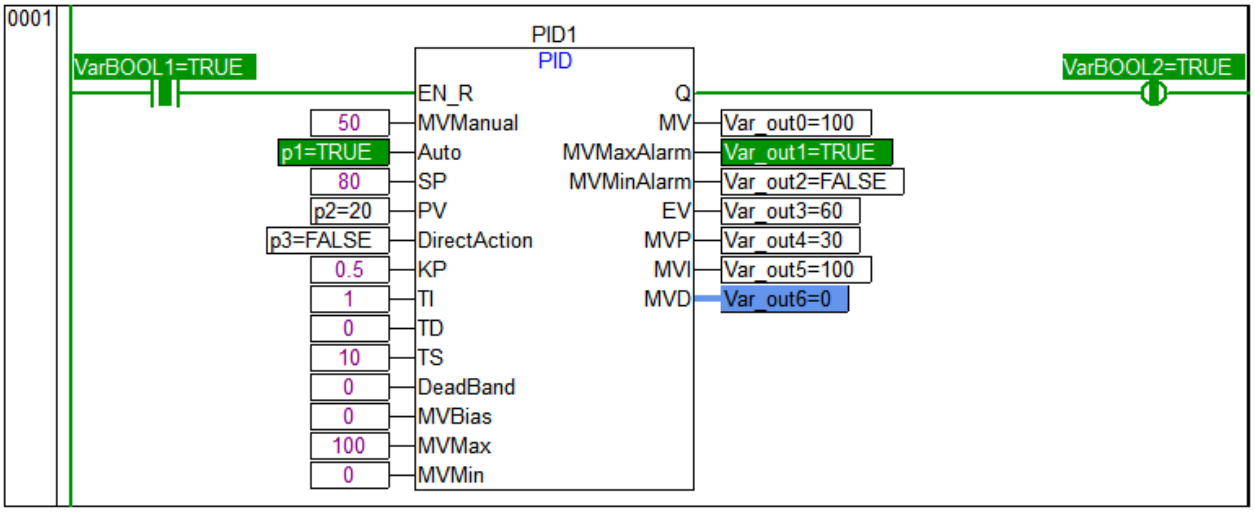
6.4.1.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
MVManual	Input	REAL	Manual input value, which is effective in manual status	0	FALSE
Auto	Input	BOOL	"Auto mode" selection: FALSE=manual TRUE=automatic	FALSE	FALSE
SP	Input	REAL	Set value	0	FALSE
PV	Input	REAL	Process value	0	FALSE
DirectAction	Input	BOOL	"Direct action" mode selection FALSE: reverse action (EV=SP-PV) TRUE: direct action (EV=PV-SP)	FALSE	FALSE
KP	Input	REAL	Proportional gain	1	FALSE
TI	Input	REAL	Integral time (S) TI>=0; if TI=0, it means there is no integration link	1	FALSE
TD	Input	REAL	Differential time (S) TD>=0; if TD=0, it means there is no differential link	0	FALSE
TS	Input	REAL	Operation cycle (S) TS>0	0	FALSE
DeadBand	Input	REAL	Deviation dead band limit If DeadBand=0, it means there is no dead band deviation. It is valid when the absolute value is greater than this value, otherwise, the deviation is 0.	0	FALSE
MVBias	Input	REAL	Feedforward control quantity	0	FALSE
MVMax	Input	REAL	Output value upper limit	100.0	FALSE
MVMin	Input	REAL	The lower limit of the input value must be smaller than the upper limit of the input, otherwise, the over-limit and under-limit flags of the output value are set, and the program returns without operation.	-100.0	FALSE
MV	Output	REAL	Output control	0	FALSE
MVMaxAlarm	Output	BOOL	Over limit alarm FALSE: The upper limit has not been exceeded TRUE: Over limit	FALSE	FALSE
MVMinAlarm	Output	BOOL	Under limit alarm FALSE: The lower limit has not been exceeded TRUE: Under limit	FALSE	FALSE
EV	Output	REAL	If there is a dead band in the deviation between the set value and the process value, EV is the deviation value after dead band processing.	0	FALSE
MVP	Output	REAL	Proportional component	0	FALSE
MVI	Output	REAL	Integral component	0	FALSE
MVD	Output	REAL	Differential component	0	FALSE
EN_R	Input	BOOL	Enable	0	FALSE
Q	Output	BOOL	Enable status flag	FALSE	FALSE

6.4.1.3 Example

The PID controller compares the on-site process value input with the set value and performs PID adjustment control. It outputs the adjustment instruction to the field device to achieve timely response adjustment tasks. At the same time, it limits the output value to ensure that the output operates within the permitted range, and sends alarms for over-limit values in a timely manner.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	PID1			PID		FALSE	Not Public	
0002	VarBOOL1			BOOL	TRUE	FALSE	Not Public	
0003	VarBOOL2			BOOL	TRUE	FALSE	Not Public	
0004	p1			BOOL	TRUE	FALSE	Not Public	
0005	p2			REAL	20	FALSE	Not Public	
0006	p3			BOOL	FALSE	FALSE	Not Public	
0007	Var_out0			REAL	100	FALSE	Not Public	
0008	Var_out1			BOOL	TRUE	FALSE	Not Public	
0009	Var_out2			BOOL	FALSE	FALSE	Not Public	
0010	Var_out3			REAL	60	FALSE	Not Public	
0011	Var_out4			REAL	30	FALSE	Not Public	
0012	Var_out5			REAL	100	FALSE	Not Public	
0013	Var_out6			REAL	0	FALSE	Not Public	



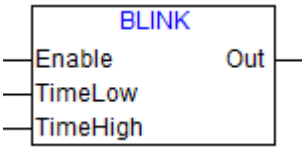
```
1   PID1(                                   PID1
2     EN_R := VarBOOL1,                     VarBOOL1 = TRUE
3     MVManual := 50,
4     Auto := p1,                           p1 = TRUE
5     SP := 80,
6     PV := p2,                             p2 = 20
7     DirectAction := p3,                   p3 = FALSE
8     KP := 0.5,
9     TI := 1,
10    TD := 0,
11    TS := 10,
12    DeadBand := 0,
13    MVBias := 0,
14    MVMax := 100,
15    MVMin := 0,
16    Q=>VarBOOL2,                          VarBOOL2 = TRUE
17    MV=>Var_out0,                         Var_out0 = 100
18    MVMaxAlarm=>Var_out1,                 Var_out1 = TRUE
19    MVMinAlarm=>Var_out2,                 Var_out2 = FALSE
20    EV=>Var_out3,                         Var_out3 = 60
21    MVP=>Var_out4,                        Var_out4 = 30
22    MVI=>Var_out5,                        Var_out5 = 400
23    MVD=>Var_out6);                       Var_out6 = 0
```

6.5 Signal Generator Instruction

6.5.1 BLINK (Pulse Signal Generator)

6.5.1.1 Function

The pulse signal generator algorithm (BLINK) completes the output of high-level and low-level rectangular waves at a specified time.



BLINK Functional Block

6.5.1.2 Parameter

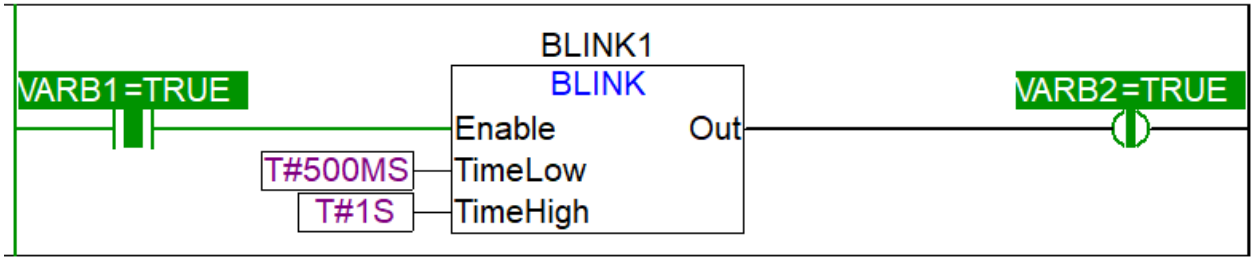
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
TimeLow	Input	TIME	Output signal low level time	0	FALSE
TimeHigh	Input	TIME	Output signal high level time	0	FALSE

Enable	Input	BOOL	Enable	FALSE	FALSE
Out	Output	BOOL	Output	FALSE	FALSE

6.5.1.3 Example

When the input ENABLE detects a rising edge, the BLINK instruction starts the pulse signal output function and stores the result in the output parameter OUT. The input the time is 1 second, which can be set to T#1S.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	BLINK1			BLINK		FALSE	Not Public	
0002	VARB1			BOOL	TRUE	FALSE	Not Public	
0003	VARB2			BOOL	TRUE	FALSE	Not Public	




```
1  BLINK1(
2  Enable := VARB1,
3  TimeLow := T#500MS,
4  TimeHigh := T#1S,
5  Out=>VARB2);
```

```
BLINK1
VARB1 = TRUE

VARB2 = TRUE
```

6.5.2 GEN (Typical Signal Generator)

6.5.2.1 Function

- Initialization function

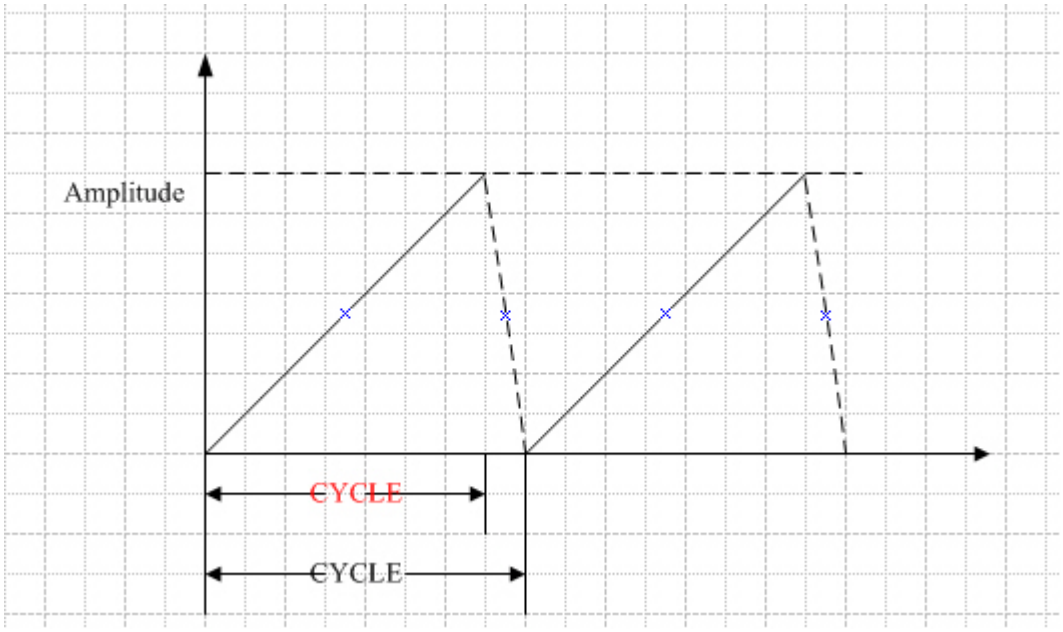
  When RESET=1, the output and related count variables are cleared.

- To calculate and generate waveforms

  When BASE=0, a MODE type waveform is generated according to the period CYCLES and the assignment AMPLITUDE; when BASE=1, only the period is different, and the period becomes PERIOD.

- Precautions

  When BASE=0 and the function type is SAWTOOTH_RISE or SAWTOOTH_FALL, the cycle period shown in CYCLE is shown in black font in the figure below, not in red font. (Because in actual signal generation, there will be no straight up and down waveforms, that is, one point in time corresponds to two values.)

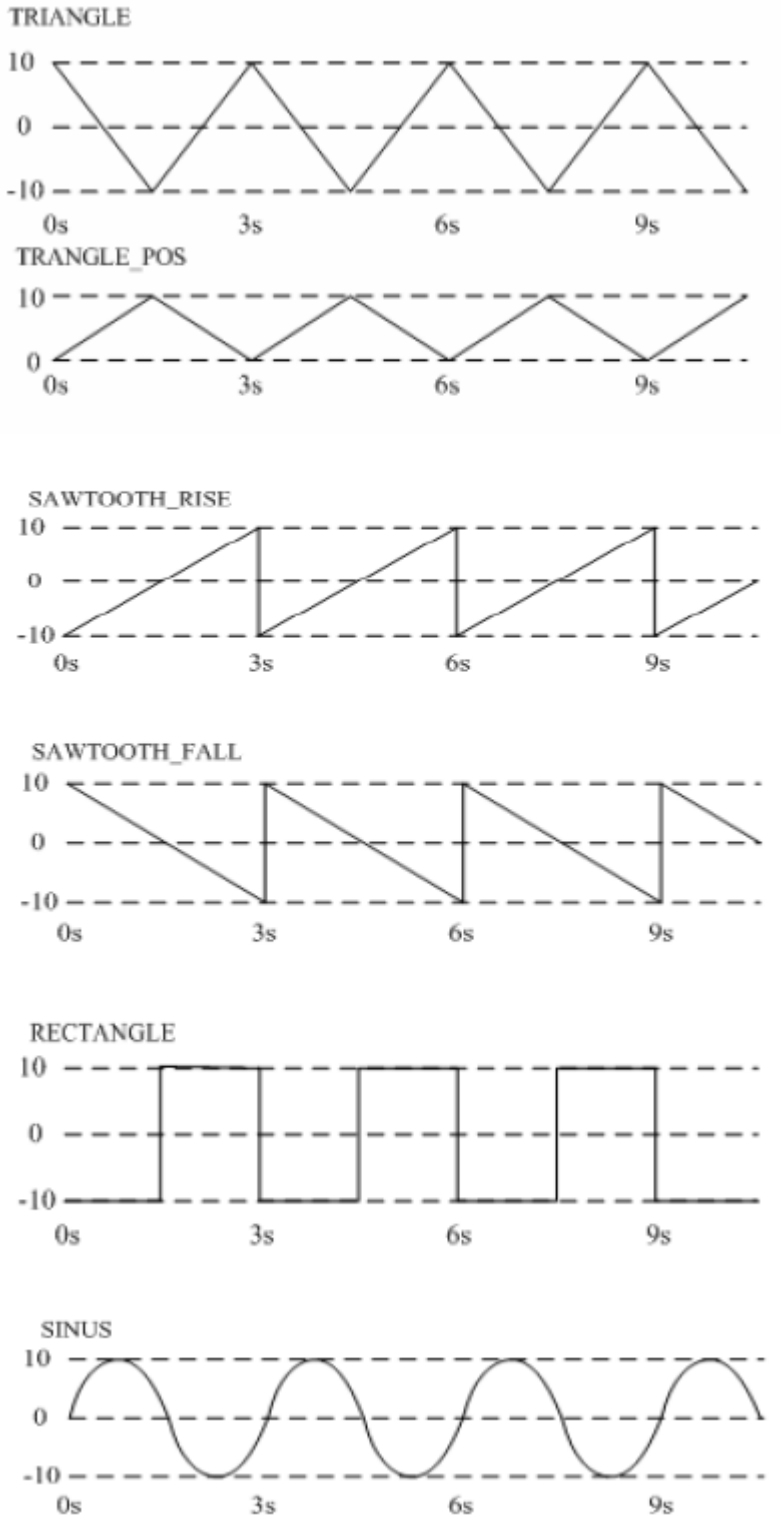


Cycle Chart

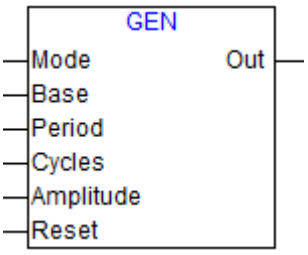
When BASE=1 and the function type is SINUS or COSINUS, if the ratio of the IEC operation period to the waveform cycle period PERIOD is within 0.1, full waveform acquisition can be achieved;

If the ratio is greater than 0.1, the peak and trough values may not be collected, and the maximum and minimum values that can be collected are related to the ratio.

Depending on the input at Mode, the generated waveforms are as shown in the figure below.



Waveform Graphs



GEN Functional Block

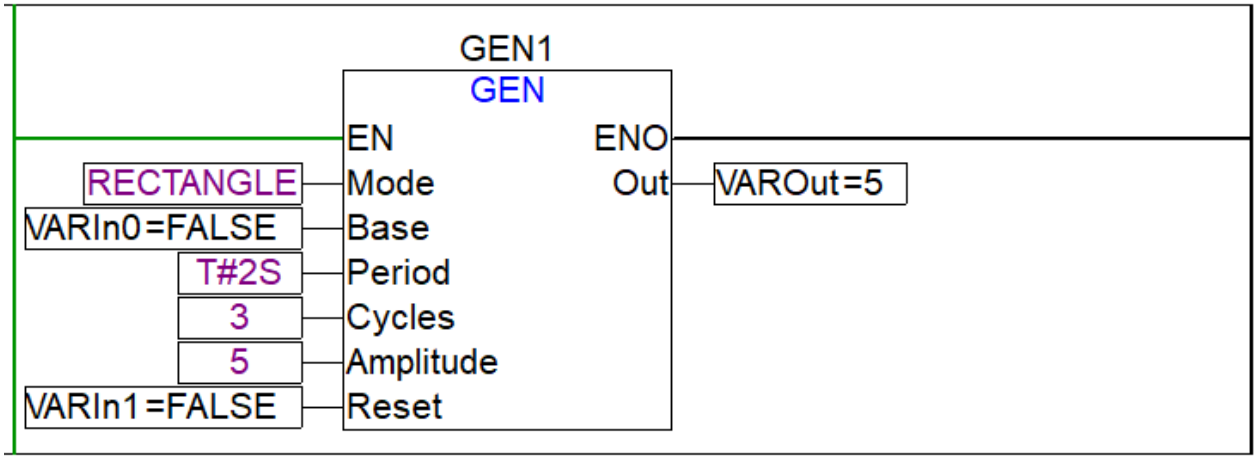
6.5.2.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
Mode	Input	GEN_MODE	Directly input TRIANGLE, TRIANGLE_POS, SAWTOOTH_RISE, SAWTOOTH_FALL, RECTANGLE, SINUS, COSINUS into MODE to generate the corresponding waveform TRIANGLE: triangle wave TRIANGLE_POS: zero-based triangle wave SAWTOOTH_RISE: rising sawtooth wave SAWTOOTH_FALL: falling sawtooth wave RECTANGLE: square wave SINUS: sine wave COSINUS: cosine wave	TRAINGLE	TRUE
Base	Input	BOOL	Cycle mode selection If BASE =TRUE, the signal generator is related to the defined cycle period; If BASE =FALSE, the signal generator is related to a specific number of occurrences	FALSE	TRUE
Period	Input	TIME	Define the cycle period (valid when Base is true)	T#1S	TRUE
Cycles	Input	INT	Define the cycle period (valid when Base is false)	1000	TRUE
Amplitude	Input	INT	Signal amplitude	0	TRUE
Reset	Input	BOOL	Initialize If RESET=TRUE, the signal generator is reset to 0	FALSE	TRUE
Out	Output	INT	Waveform signal output value	0	TRUE

6.5.2.3 Example

When the input EN detects a rising edge, the GEN instruction starts the periodic signal function, calculates according to the relevant waveform parameters set by the user, and stores the calculation results in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	GEN1			GEN		FALSE	Not Public	
0002	VARIn0			BOOL	FALSE	FALSE	Not Public	
0003	VARIn1			BOOL	FALSE	FALSE	Not Public	
0004	VAROut			INT	5	FALSE	Not Public	



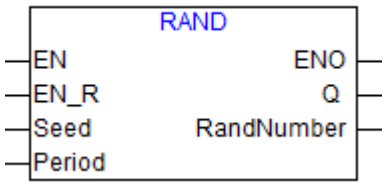

```
1  GEN1(
2  Mode := RECTANGLE,
3  Base := VARIn0,
4  Period := T#2S,
5  Cycles := 3,
6  Amplitude := 5,
7  Reset := VARIn1,
8  Out=>VAROut);
```

```
GEN1

VARIn0 = FALSE


VARIn1 = FALSE
VAROut = 5
```

6.5.3 RAND (Random Number Generator)

6.5.3.1 Function

Generate random numbers.



RAND Functional Block

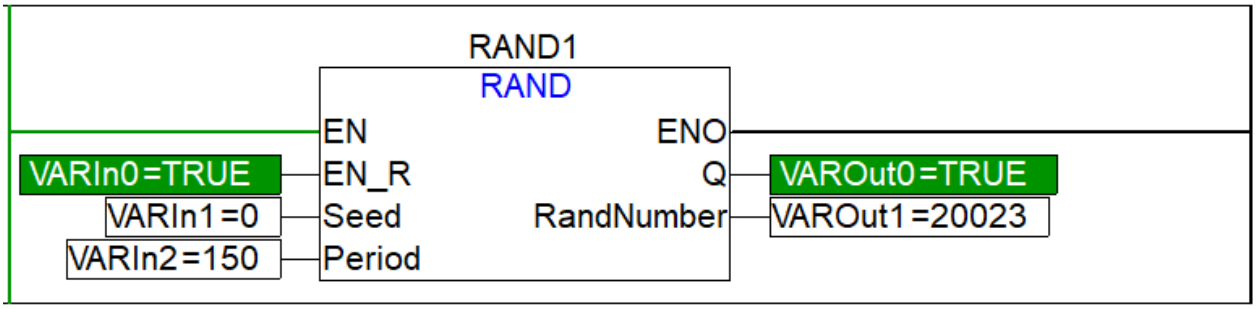
6.5.3.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN_R	Input	BOOL	Enable 0: Invalid, in which case it stops generating random numbers and the output remains unchanged from the last generated random number; 1: Rising edge enabled, and high level valid	FALSE	FALSE
Seed	Input	UINT	Initial value of the seed calculation random sequence, range: 0~65535 To ensure the continuity of random number generation, the seed value is only valid in the following scenarios: full installation, controller reset, AT executing "cold reset", and "reset".	0	FALSE
Period	Input	UDINT	0: A random number is generated in every scan cycle Non-0: If it is set to 200, a random number will be generated every 200 ms Value range: 0~4294967295	0	FALSE
Q	Output	BOOL	0: Invalid data 1: Valid data	FALSE	FALSE
RandNumber	Output	UINT	0-65535	0	FALSE

6.5.3.3 Example

When the input EN_R detects a rising edge, the RAND instruction starts the random number function and stores the calculation results in the output parameter RandNumber.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	RAND1			RAND		FALSE	Not Public	
0002	VARIn0			BOOL	TRUE	FALSE	Not Public	
0003	VARIn1			UINT	0	FALSE	Not Public	
0004	VARIn2			UDINT	150	FALSE	Not Public	
0005	VAROut0			BOOL	TRUE	FALSE	Not Public	
0006	VAROut1			UINT	20023	FALSE	Not Public	




```
1  RAND1(
2  EN_R := VARIn0,
3  Seed := VARIn1,
4  Period := VARIn2,
5  Q=>VAROut0,
6  RandNumber=>VAROut1);
```

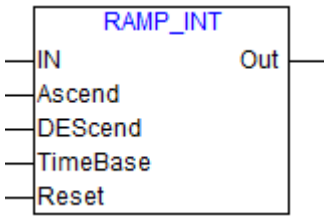
```
RAND1
VARIn0 = TRUE
VARIn1 = 0
VARIn2 = 150
VAROut0 = TRUE
VAROut1 = 20023
```

6.6 Analog Processing Instruction

6.6.1 RAMP_INT (Int Rate Limit)

6.6.1.1 Function

Limit the rising and falling speed of an integer input function.



RAMP_INT Functional Block

6.6.1.2 Parameter

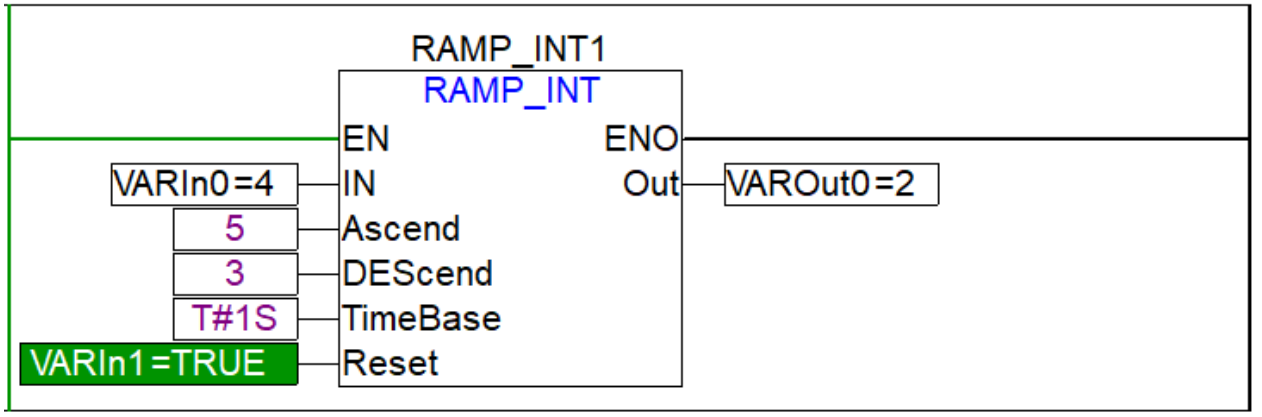
Parameter	Statement	Data Type	Function Description	Initial Value	Power Fail Safeguard
IN	Input	INT	Manipulated variable	0	FALSE

Ascend	Input	INT	Maximum rising value within time interval	0	TRUE
DEScend	Input	INT	Maximum falling value within time interval	0	TRUE
TimeBase	Input	TIME	Time interval	T#0S	TRUE
Reset	Input	BOOL	Restart signal	FALSE	TRUE
Out	Output	INT	Value after rising and falling limits	0	TRUE

6.6.1.3 Example

When the input EN detects a rising edge, the RAMP_INT instruction starts the integral speed limit function and stores the calculation result in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	RAMP_INT1			RAMP_INT		FALSE	Not Public	
0002	VARIn0			INT	4	FALSE	Not Public	
0003	VARIn1			BOOL	TRUE	FALSE	Not Public	
0004	VAROut0			INT	2	FALSE	Not Public	



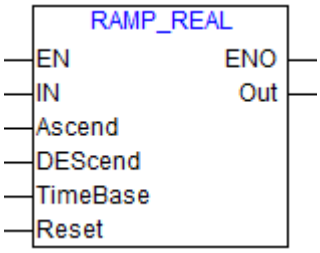

```
1  RAMP_INT1(
2  IN := VARIn0,
3  Ascend := 5,
4  DEScend := 3,
5  TimeBase := T#1S,
6  Reset := VARIn1,
7  Out=>VAROut0);
```

```
RAMP_INT1
VARIn0 = 4

VARIn1 = TRUE
VAROut0 = 2
```

6.6.2 RAMP_REAL (Real Rate Limit)

6.6.2.1 Function

The RAMP_REAL instruction can limit the rising and falling speed of real input functions.



RAMP_REAL Functional Block

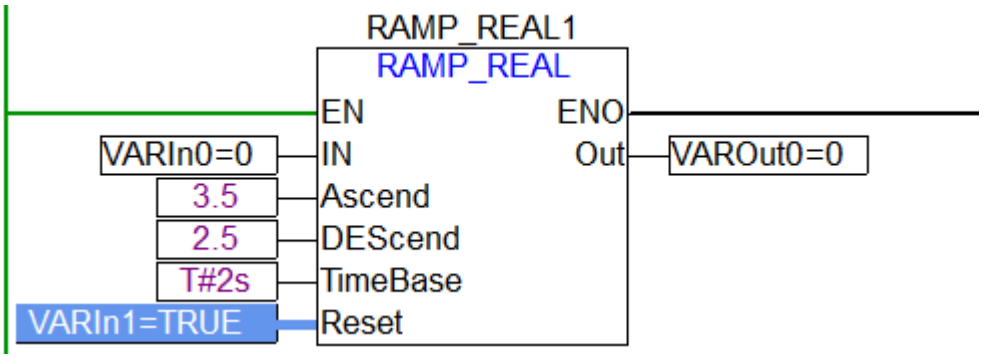
6.6.2.2 Parameter

Parameter	Statement	Data Type	Function Description	Initial Value	Power Fail Safeguard
IN	Input	REAL	Manipulated variable	0	FALSE
Ascend	Output	REAL	Maximum rising value within time interval	0	TRUE
DEScend	Input	REAL	Maximum falling value within time interval	0	TRUE
TimeBase	Input	TIME	Time interval	T#0S	TRUE
Reset	Input	BOOL	Restart signal	FALSE	TRUE
Out	Output	REAL	Value after rising and falling limits	0	TRUE

6.6.2.3 Example

When the input EN detects a rising edge, the RAMP_REAL instruction starts the real speed limit function and stores the calculation result in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	RAMP_REAL1			RAMP_REAL		FALSE	Not Public	
0002	VARIn0			REAL	0	FALSE	Not Public	
0003	VARIn1			BOOL	TRUE	FALSE	Not Public	
0004	VAROut0			REAL	0	FALSE	Not Public	




```
1  RAMP_REAL1(
2  IN := VARIn0,
3  Ascend := 3.5,
4  DEScend := 2.5,
5  TimeBase := T#2S,
6  Reset := VARIn1,
7  Out=>VAROut0);
```

```
RAMP_REAL1
VARIn0 = 0


VARIn1 = TRUE
VAROut0 = 0
```

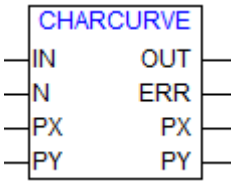
6.6.3 CHARCURVE (Characteristic Curve)

6.6.3.1 Function

IN is the DINT type, where the value to be operated is entered. N is the byte type, which determines the number of points of the characteristic curve and 2=<N<=11. Generate a characteristic row in the DINT type arrays PX[0..10] and PY[0..10]. The DINT type output OUT is the value after operation. The byte type variable ERR outputs error information.

The points PX[0]..PX[N-1] in the array must be stored from small to large, otherwise, the value of ERR is 1. If IN is not between PX[0] and PX[N-1], ERR=2, and the output contains the corresponding limit value PY[0] or PY[N-1].

If N exceeds the allowed value between 2 and 11, ERR=4.



CHARCURVE Functional Block

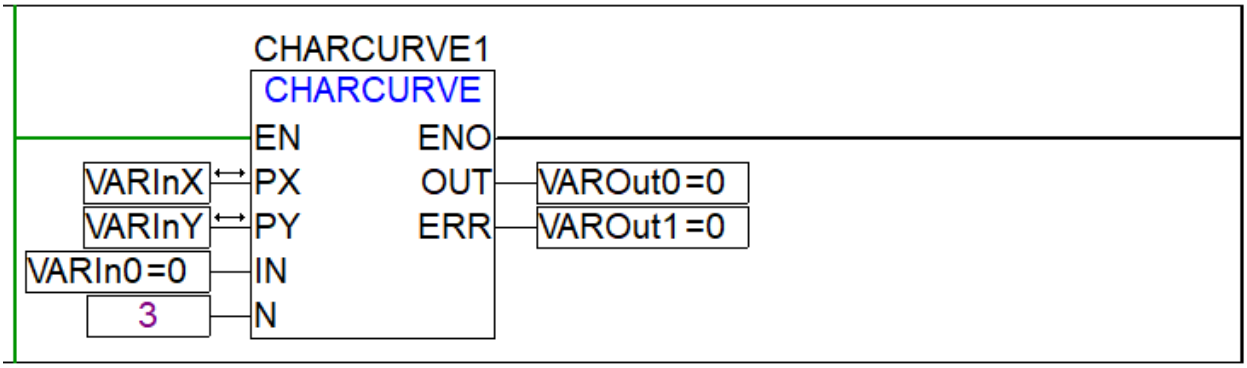
6.6.3.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
IN	Input	DINT	Input operand	0	FALSE
N	Input	BYTE	The actual number of points of the characteristic curve 2=<N<=11	0	FALSE
OUT	Output	DINT	Output value corresponding to the input X on the characteristic curve	0	FALSE
ERR	Output	BYTE	The points PX[0]..PX[N-1] in the array must be stored from small to large, otherwise, the value of ERR is 1; If IN is not between PX[0] and PX[N-1], ERR=2; If N exceeds the allowed value between 2 and 11, ERR=4;	0	FALSE
PX	Other parameters	ARRAY[0..10] OF DINT	X-axis point array of characteristic curve		FALSE
PY	Other parameters	ARRAY[0..10] OF DINT	Y-axis point array of characteristic curve		FALSE

6.6.3.3 Example

When the input EN detects a rising edge, the CHARCURVE instruction starts the characteristic curve function, finds the output corresponding to the input on the characteristic curve based on the characteristic curve points and quantity of the input, and stores the calculation results in the output parameter OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	CHARCURVE1			CHARCURVE		FALSE	Not Public	
0002	VARInX			ARRAY[0..10] OF DINT		FALSE	Not Public	
0003	VARIn0			BYTE	0	FALSE	Not Public	
0004	VARInY			ARRAY[0..10] OF DINT		FALSE	Not Public	
0005	VAROut0			DINT	0	FALSE	Not Public	
0006	VAROut1			BYTE	0	FALSE	Not Public	

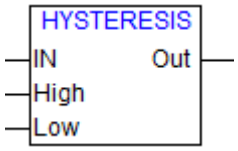



```
1  CHARCURVE1(          CHARCURVE1
2  IN := VARIn0,        VARIn0 = 0
3  N := 3,
4  PX := VARInX,        VARInX
5  PY := VARInY,        VARInY
6  OUT=>VAROut0,        VAROut0 = 0
7  ERR=>VAROut1);       VAROut1 = 0
```

6.6.4 HYSTERESIS (Lag)

6.6.4.1 Function

When the input value is smaller than the lower limit, the output is TRUE; when the input value is greater than the upper limit, the output is FALSE, when the input value is between the upper and lower limits, the output remains.



HYSTERESIS Functional Block

6.6.4.2 Parameter

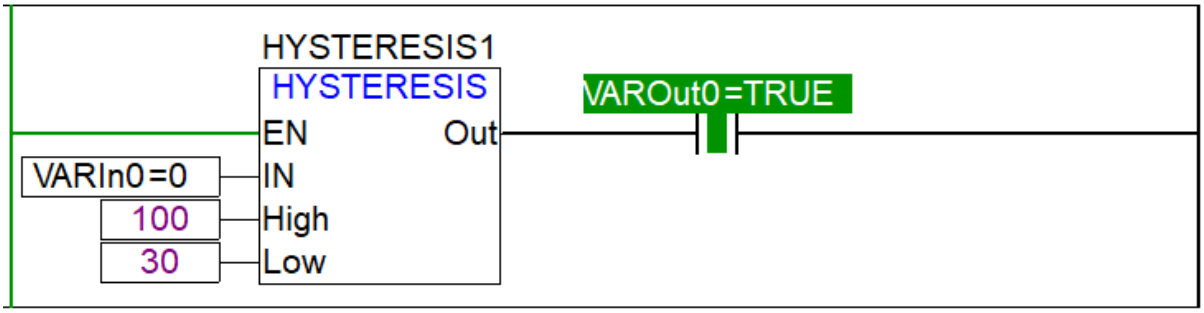
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
IN	Input	INT	Enter the operand	0	FALSE

HIGH	Input	INT	Upper limit settings	0	FALSE
LOW	Input	INT	Lower limit settings	0	FALSE
OUT	Output	BOOL	If OUT=FALSE, the input value is greater than the upper limit; If OUT=TRUE, the input value is smaller than the lower limit; The output remains when the input value is between the upper and lower limits.	FALSE	FALSE

6.6.4.3 Example

When the input EN detects a rising edge, the HYSTERESUS instruction starts the lag function and stores the corresponding results in the output parameter OUT.

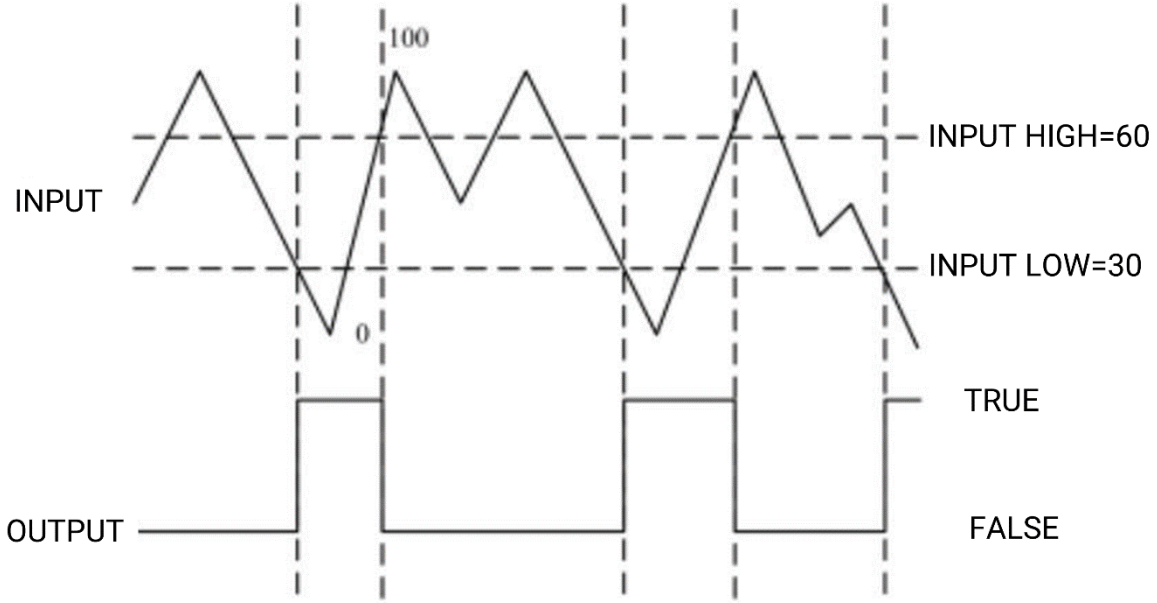
No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	HYSTERESIS1			HYSTERESIS		FALSE	Not Public	
0002	VARIn0			INT	0	FALSE	Not Public	
0003	VAROut0			BOOL	TRUE	FALSE	Not Public	




```
1  HYSTERESIS1(
2  IN := VARIn0,
3  High := 100,
4  Low := 30,
5  Out=>VAROut0);
```

```
HYSTERESIS1
VARIn0 = 0

VAROut0 = TRUE
```

In the above example, when the instruction is executed, according to the change of the input value, the corresponding output value is as shown in the figure.

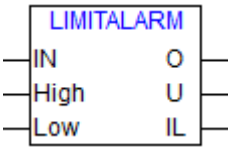


Relationship Between Input and Output Based on Changes

6.6.5 LIMITALARM (Limit Alarm)

6.6.5.1 Function

When the input value is smaller than the lower limit, the output U is TRUE; when the input value is greater than the upper limit, the output O is TRUE, and; when the input value is between the upper and lower limits, the output IL is TRUE.



LIMITALARM Functional Block

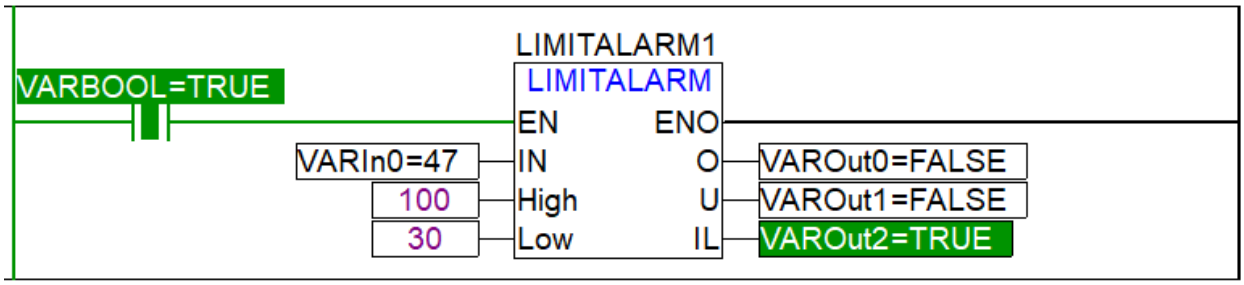
6.6.5.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
IN	Input	INT	Enter the operand	0	FALSE
HIGH	Input	INT	Upper limit settings	0	FALSE
LOW	Input	INT	Lower limit settings	0	FALSE
O	Output	BOOL	TRUE when the input value is greater than the upper limit	FALSE	FALSE
U	Output	BOOL	TRUE when the input is smaller than the lower limit	FALSE	FALSE
IL	Output	BOOL	TRUE when the input value is between the upper and lower limits	FALSE	FALSE

6.6.5.3 Example

When the input EN detects a rising edge, the LIMITALARM instruction starts the upper and lower limit functions and stores the calculation results in the output parameters based on the input data.

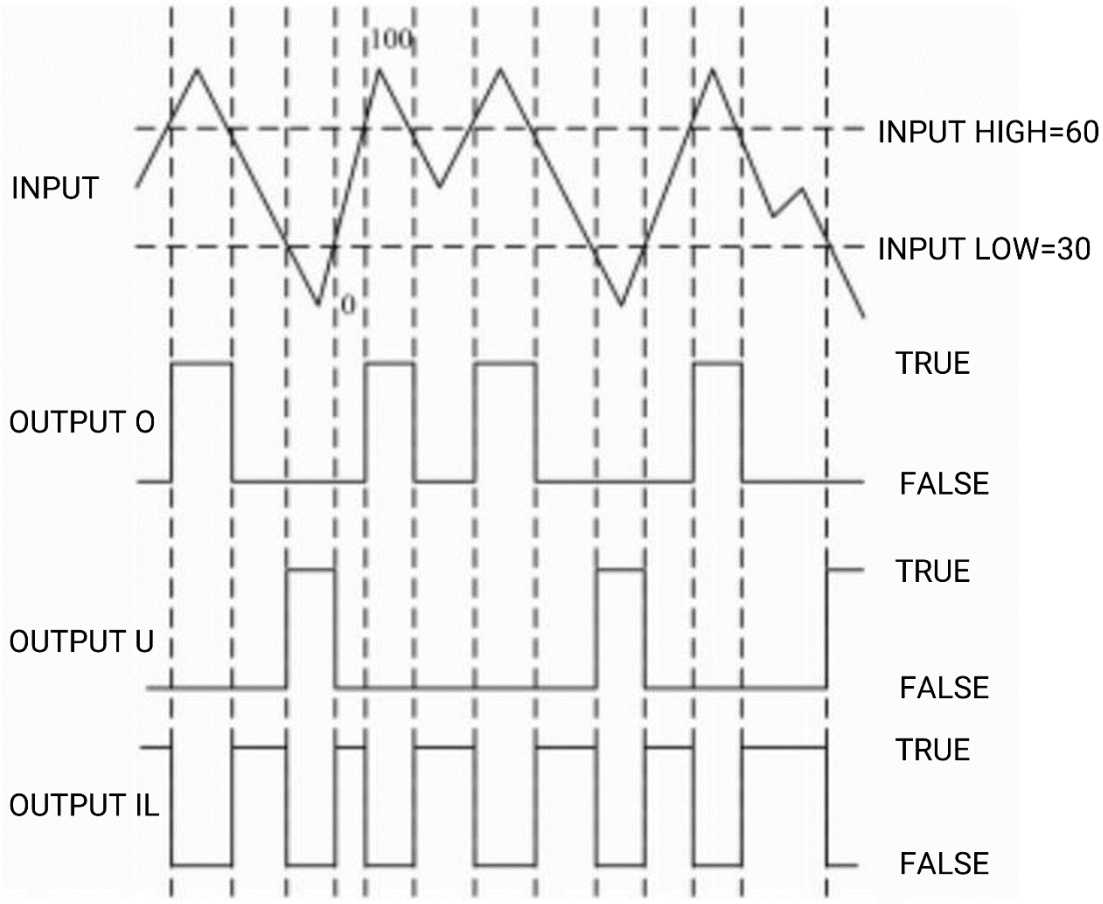
No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	LIMITALARM1			LIMITALARM		FALSE	Not Public	
0002	VARBOOL			BOOL	TRUE	FALSE	Not Public	
0003	VARIn0			INT	47	FALSE	Not Public	
0004	VAROut0			BOOL	FALSE	FALSE	Not Public	
0005	VAROut1			BOOL	FALSE	FALSE	Not Public	
0006	VAROut2			BOOL	TRUE	FALSE	Not Public	




```
1  LIMITALARM1(
2  IN := VARIn0,
3  High := 100,
4  Low := 30,
5  O=>VAROut0,
6  U=>VAROut1,
7  IL=>VAROut2);
```

```
LIMITALARM1
VARIn0 = 47

VAROut0 = FALSE
VAROut1 = FALSE
VAROut2 = TRUE
```

In the above example, when the instruction is executed, according to the change of the input value, the corresponding output value is as shown in the figure.

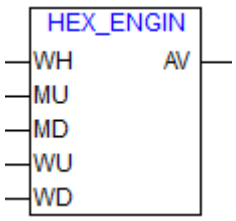


Relationship Between Input and Output Based on Changes

6.6.6 HEX_ENGIN (Hexadecimal Conversion to Engineering Data)

6.6.6.1 Function

Convert the input code value WH into the corresponding engineering quantity AV value.



HEX_ENGIN Functional Block

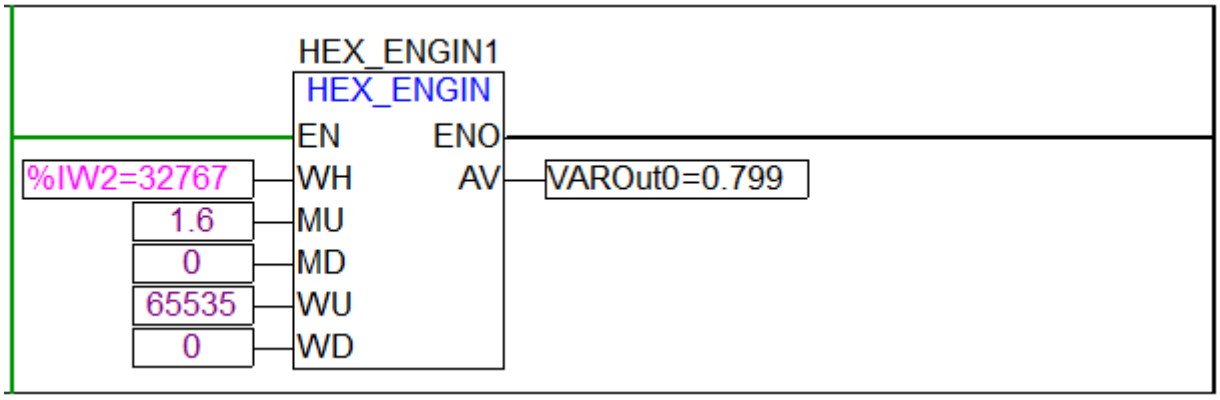
6.6.6.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
WH	Input	DINT	Enter the channel value to be processed	0	FALSE
MU	Input	REAL	Upper limit of engineering quantity	0	FALSE
MD	Input	REAL	Lower limit of engineering quantity	0	FALSE
WU	Input	DINT	Upper limit of analog input code value of PLC	0	FALSE
WD	Input	DINT	Lower limit of analog input code value of PLC	0	FALSE
AV	Output	REAL	Converted engineering quantity output	0	FALSE
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		

6.6.6.3 Example

When the input EN detects a rising edge, the HEX_ENGIN instruction starts the engineering quantity conversion function and stores the calculation results in the output parameter AV.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	HEX_ENGIN1			HEX_ENGIN		FALSE	Not Public	
0002	VAROut0			REAL	0.799	FALSE	Not Public	




```
1  HEX_ENGIN1(                    HEX_ENGIN1
2  WH := %IW2,                    %IW2 = 32767
3  MU := 1.6,
4  MD := 0,
5  WU := 65535,
6  WD := 0,
7  AV=>VAROut0);                  VAROut0 = 0.799
```

Program description:

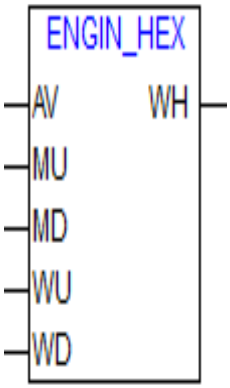
This instruction is generally used for analog input data processing;

When EN is set, conversion is performed, for which the input data is %IW2, the upper and lower limits of engineering quantity are 1.6 MPa and 0 Mpa, and the corresponding module code value range is 0-65535. When the value of %IW2 is 32767, the corresponding output value is 0.799.

6.6.7 ENGIN_HEX(Engineering Data Conversion to Hexadecimal (WORD))

6.6.7.1 Function

Convert the input engineering quantity AV into the corresponding code value WH.



ENGIN_HEX Functional Block

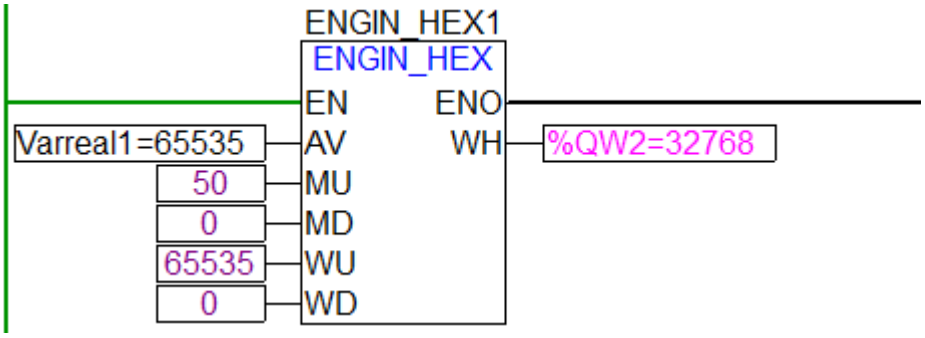
6.6.7.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
AV	Input	REAL	Enter the engineering quantity to be processed	0	FALSE
MU	Input	REAL	Upper limit of engineering quantity	0	FALSE
MD	Input	REAL	Lower limit of engineering quantity	0	FALSE
WU	Input	WORD	Upper limit of analog input code value of PLC	0	FALSE
WD	Input	WORD	Lower limit of analog input code value of PLC	0	FALSE
WH	Output	WORD	Converted output code value	0	FALSE

6.6.7.3 Example

When the input EN detects a rising edge, the ENGIN_HEX instruction starts the engineering quantity conversion function and stores the calculation results in the output parameter WH.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	ENGIN_HEX1			ENGIN_HEX		FALSE	Not Public	
0002	Varreal1			REAL	65535	FALSE	Not Public	




```
1  ENGIN_HEX1(
2  AV := Varreal1,
3  MU := 50,
4  MD := 0,
5  WU := 65535,
6  WD := 0,
7  WH=>%QW2);
```

```
ENGIN_HEX1
Varreal1 = 65535


%QW2 = 32768
```

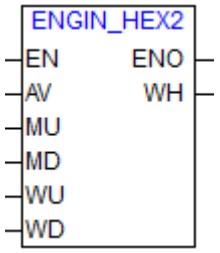
Program description:

This instruction is generally used for analog output data processing. When EN is set, conversion is performed, for which the actual engineering quantity data is Varreal1, the upper and lower limits of engineering quantity are 50 Hz and 0 Hz, and the corresponding module range is 0~65535. When the value of Varreal1 is 25 Hz, the corresponding output value is 32768. When 32768 is assigned to %QW2, then the analog output value of this channel is 25 Hz.

6.6.8 ENGIN_HEX2 (Engineering Data Conversion to Hexadecimal (INT))

6.6.8.1 Function

Convert the input engineering quantity AV into the corresponding code value WH.



ENGIN_HEX2 Functional Block

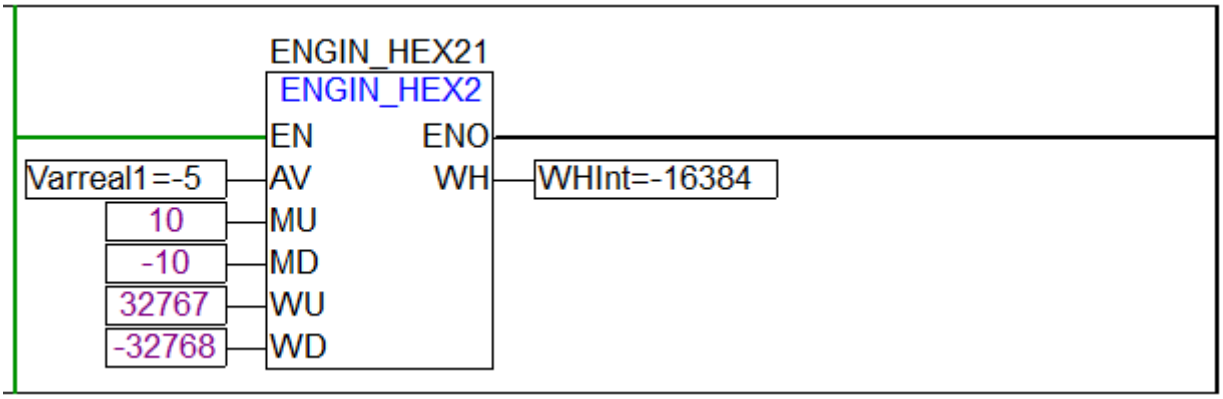
6.6.8.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
AV	Input	REAL	Enter the engineering quantity to be processed	0	FALSE
MU	Input	REAL	Upper limit of engineering quantity	0	FALSE
MD	Input	REAL	Lower limit of engineering quantity	0	FALSE
WU	Input	INT	Upper limit of analog input code value of PLC	0	FALSE
WD	Input	INT	Lower limit of analog input code value of PLC	0	FALSE
WH	Output	INT	Converted output code value	0	FALSE

6.6.8.3 Example

When the input EN detects a rising edge, the ENGIN_HEX2 instruction starts the engineering quantity conversion function and stores the calculation results in the output parameter WH.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	ENGIN_HEX21			ENGIN_HEX2		FALSE	Not Public	
0002	Varreal1			REAL	-5	FALSE	Not Public	
0003	WHInt	%QW4		INT	16384	FALSE	Not Public	




```
1  ENGIN_HEX21(                ENGIN_HEX21
2  AV := Varreal1,             Varreal1 = -5
3  MU := 10,
4  MD := -10,
5  WU := 32767,
6  WD := -32767,
7  WH=>WHInt);                 WHInt = -16384
```

Program description:

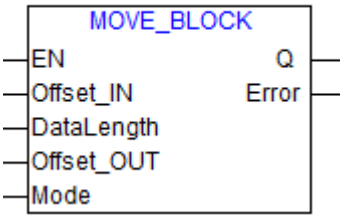
This instruction is generally used for analog output data processing. When EN is set, conversion is performed, for which the actual engineering quantity data is Varreal1, the upper and lower limits of engineering quantity are 10 V and -10 V, and the corresponding module range is -32768~32767. When the value of Varreal1 is -5 V, the corresponding output value is -16384. When -16384 is assigned to WHInt (%QW4), then the analog output value of this channel is -5 V.

6.7 Data Block Transfer Instruction

6.7.1 MOVE_BLOCK (Data Block Move)

6.7.1.1 Function

Transfer data in the M area from one location to another location in the M area in batches.



MOVE_BLOCK Functional Block

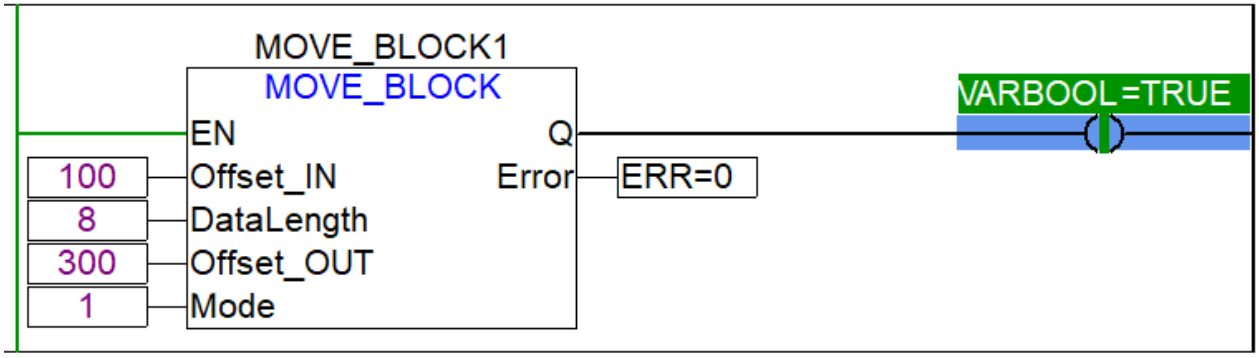
6.7.1.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
Offset_In	Input	DWORD	Offset word address of data area	0	FALSE
DataLength	Input	BYTE	Length of data transferred: 0~255	0	FALSE
Offset_Out	Input	DWORD	Offset word address of data area	0	FALSE
Mode	Input	BYTE	Data transfer method: =0: It indicates byte transfer; =1: It indicates word transfer; =2: It indicates double-word transfer;	0	FALSE
Q	Output	BOOL	Enable output: When the data is not transferred or it is transferred incorrectly, Q=FALSE. When the data is transferred correctly, Q=TRUE.	FALSE	FALSE
Error	Output	BYTE	Error message: =0: The data is transferred correctly =1: The offset address (Offset_IN) of the data source data area is incorrect =2: The offset address (Offset_OUT) of the data destination data area is incorrect =3: The data transfer mode (Mode) is incorrect	0	FALSE

6.7.1.3 Example

For example, when EN is set, the N bytes of data pointed to by Offset_IN are transferred to the destination address pointed by Offset_OUT.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	MOVE_BLOCK1			MOVE_BLOCK		FALSE	Not Public	
0002	ERR			BYTE	0	FALSE	Not Public	
0003	VARBOOL			BOOL	TRUE	FALSE	Not Public	



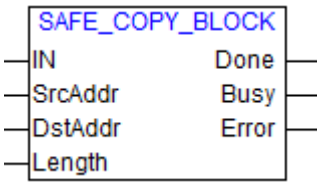

```
2  MOVE_BLOCK1(
3  Offset_IN := 100,
4  DataLength := 8,
5  Offset_OUT := 300,
6  Mode := 1,
7  Q=>VARBOOL,
8  Error=>ERR);
```

```
MOVE_BLOCK1


VARBOOL = TRUE
ERR = 0
```

6.7.2 SAFE_COPY_BLOCK (Data Block Safe Copy)

6.7.2.1 Function

It provides the SafeCopyBlock library instruction, which is the I/O data used by the general IEC task during the calculation process. To prevent changes during the IEC task calculation process, this library instruction needs to be used to copy the corresponding I area data to the G or M area when the IEC task starts running. To ensure the consistency of the data in the Q area of the output module, the user shall avoid directly operating the I/O channel address, and, instead, use this instruction to copy the data in the G/M buffer to the corresponding I/O channel address at one time after the IEC task is completed.



SAFE_COPY_BLOCK Functional Block

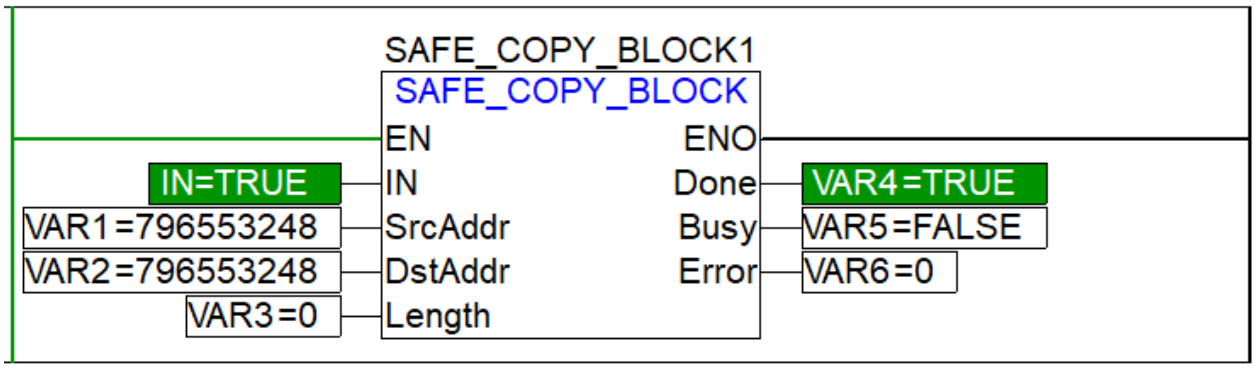
6.7.2.2 Parameter

Parameter	Statement	Data Type	Function Description	Description of Parameter Values	Initial Value	Power Fail Safeguard

IN	Input	BOOL	Data copy trigger signal	High level trigger copy	FALSE	FALSE
SrcAddr	Input	POINTER TO BYTE	Offset byte address of data source	Default value 0	0	FALSE
DstAddr	Input	POINTER TO BYTE	Offset byte address of data destination	Default value 0	0	FALSE
Length	Input	DWORD	Copied data length	0~256	0	FALSE
Done	Output	BOOL	Data copy completion flag	0: Not completed 1: Completed	FALSE	FALSE
Busy	Output	BOOL	Data copy status	0: Not started or ended 1: In progress	FALSE	FALSE
Error	Output	WORD	Error message	= 0: No error = 1: The source data byte address is empty = 2: The destination data byte address is empty = 3: The copy length is 0 or greater than 256 = 4: The source data byte address + copy length exceeds the source data area (I/Q/M/G/S) = 5: The destination data byte address is in area I or S = 6: The destination data word address + copy length exceeds the destination data area (Q/M/G)	0	FALSE

6.7.2.3 Example

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	SAFE_COPY_BLO...			SAFE_COPY_BLOCK		FALSE	Not Public	
0002	IN			BOOL	TRUE	FALSE	Not Public	
0003	VAR1			POINTER TO BYTE	808239136	FALSE	Not Public	
0004	VAR2			POINTER TO BYTE	808239136	FALSE	Not Public	
0005	VAR3			DWORD	0	FALSE	Not Public	
0006	VAR4			BOOL	FALSE	FALSE	Not Public	
0007	VAR5			BOOL	TRUE	FALSE	Not Public	
0008	VAR6			WORD	0	FALSE	Not Public	




```
1  SAFE_COPY_BLOCK1(
2  IN := IN,
3  SrcAddr := VAR1,
4  DstAddr := VAR2,
5  Length := VAR3,
6  Done=>VAR4,
7  Busy=>VAR5,
8  Error=>VAR6);
```

```
SAFE_COPY_BLOCK1
IN = TRUE
VAR1 = 796553248
VAR2 = 796553248
VAR3 = 0
VAR4 = TRUE
VAR5 = FALSE
VAR6 = 0
```

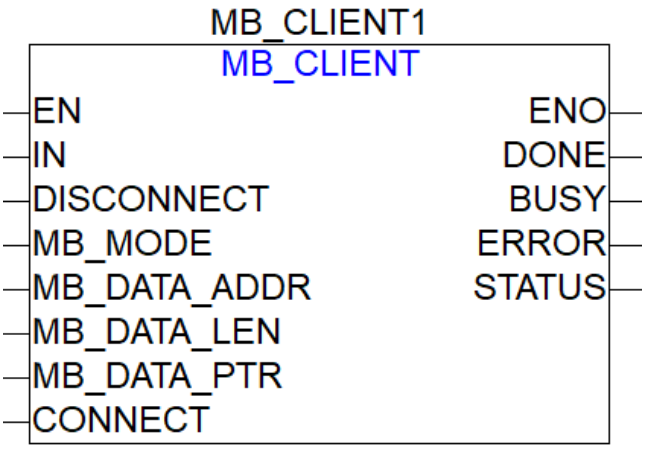
Chapter 7 System Instruction

7.1 Communication Functions

7.1.1 MB_CLIENT (Modbus TCP Master Communication)

7.1.1.1 Function

The "MB_CLIENT" directive acts as a Modbus TCP client to communicate with the Modbus TCP server over an Ethernet network. The "MB_CLIENT" directive allows you to establish a connection between the client and server, send a Modbus request, and receive a response. Support for 64 master stations.



MB_CLIENT Functional Block

7.1.1.2 Parameter

Parameter	Statement	Data Type	Function Description	Initial Value
IN	Input	BOOL	High level trigger	FALSE
DISCONNECT	Input	BOOL	Control establishing and terminating connections with the Modbus server: 0: The default value. Establishes a communication connection with a connection partner configured via the CONNECT parameter 1: Disconnect TCP connection. No other functions are performed during the termination of the connection. After the connection is successfully terminated, the STATUS parameter outputs a value of 0003	FALSE

MB_MODE	Input	USINT	Select the Modbus request pattern, preferably 0, 1, 2, 101, 102, 103, 104, 105, 106, 115, 116	0
MB_DATA_ADDR	Input	UDINT	The Modbus TCP server data starting address to be accessed depends on MB_MODE	0
MB_DATA_LEN	Input	UINT	Data length: The number of digits or words of data to access. See MB_MODE, MB_DATA_ADDR, and MB_DATA_LEN parameters for more details	0
MB_DATA_PTR	Input	PINTER TO BYTE	A pointer to the data buffer where the data to be received from or sent to the Modbus server is located	0
CONNECT	Input	CONNECT_IPV4	Pointer to connection description structure: see parameter CONNECT_IPV4 for details	
DONE	Output	BOOL	If the last Modbus job completes successfully, the bit in the output parameter DONE is immediately set to 1, which defaults to 0	FALSE
BUSY	Output	BOOL	Job status bit: 0: No Modbus requests in progress 1: Processing Modbus request	FALSE
ERROR	Output	BOOL	Error bit: 0: No errors 1: An error occurred, the details of the cause of which are specified by the STATUS parameter	FALSE
STATUS	Output	WORD	Communication status word (see error code for details)	0

■ Parameter description:

1. MB_DATA_PTR parameter

   The parameter MB_DATA_PTR is a pointer to the data buffer that will receive data from the Modbus server or where the data to be sent to the Modbus server is located.

2. MB_MODE, MB_DATA_ADDR and MB_DATA_LEN parameters

   The combination of parameters MB_MODE, MB_DATA_ADDR and MB_DATA_LEN defines the Modbus function code:

   (1) MB_MODE contains information about read and write operations: MB_MODE = 0: read, MB_MODE = 1 and 2: write (NOTE: MB_MODE = 2, Modbus functions 15 and 05 or Modbus functions 16 and 06 are indistinguishable).

   (2) MB_DATA_ADD contains information about the target to be read/written, as well as address information for the "MB_CLIENT" directive to calculate the remote address.

   (3) MB_DATA_LEN contains the number to read/write.

   The combination of parameters MB_MODE, MB_DATA_ADDR and MB_DATA_LEN defines the Modbus function code:

MB_MODE, MB_DATA_ADDR and MB_DATA_LEN and Modbus function code relationship description table

MB_MODE	MB_DATA_ADDR	MB_DATA_LEN	Modbus function code	Function and data type
0	1~9999	1~2000	01	Read 1 ~ 2000 output bits at remote addresses 0 ~ 9998
0	10001~19999	1~2000	02	Read 1 ~ 2000 input bits at remote addresses 0 ~ 9998
0	40001~49999 400001~ 465535	1~125	03	1、 Read 1 ~ 125 holding registers at remote addresses 0 ~ 9998

				2、 Read 1 ~ 125 holding registers at remote addresses 0 ~ 65534
0	30001~39999	1~125	04	Read 1 to 125 input words at remote addresses 0 to 9998
1	1~ 9999	1	05	Write 1 output bit at remote addresses 0-9998
1	40001~49999 400001~ 465535	1	06	1、 Write 1 holding register at remote addresses 0-9998 2、 Write 1 holding register at remote addresses 0 ~ 65534
1	1~9999	2~1968	15	Write 2-1968 output bits at remote addresses 0-9998
1	40001~49999 400001~ 465535	2~123	16	1、 Write 2 ~ 123 holding registers at remote addresses 0 ~ 9998 2、 Write 2 ~ 123 holding registers at the remote address 0 ~ 65534
2	1~9999	1~1968	15	Write 1 ~ 1968 output bits at remote addresses 0 ~ 9998
2	40001~49999 400001~ 465535	1~123	16	1、 Write 1 ~ 123 holding registers at the remote address 0 ~ 9998 2、 Write 1 ~ 123 holding registers at the remote address 0 ~ 65534
101	0~65535	1~2000	01	Read 1 ~ 2000 output bits at remote addresses 0 ~ 65535
102	0~65535	1~2000	02	Read 1 ~ 2000 input bits at remote addresses 0 ~ 65535
103	0~65535	1~125	03	Read 1 to 125 holding registers at remote addresses 0 to 65535
104	0~65535	1~125	04	Read 1 to 125 input words at remote addresses 0 to 65535
105	0~65535	1	05	Write 1 output bit at remote addresses 0-65535
106	0~65535	1	06	Write 1 holding register at remote addresses 0-65535
115	0~65535	1~1968	15	Write 1 ~ 1968 output bits at remote address 0 ~ 6553
116	0~65535	1~123	16	Write 1 ~ 123 holding registers at remote addresses 0 ~ 6553

■ CONNECT_IPV4 data type:

For CONNECT parameter settings, use CONNECT_IPV4 of the following structure to describe the connection:

(1) Ensure that only TCP -type connections are specified in the CONNECT_IPV4 structure.

(2) The following TCP port numbers cannot be used for this connection: 20, 21, 25, 80, 102, 135, 161, 162, 443, 1200, 34962, 34963, and 34964.

Parameters for CONNECT_IPV4

Parameter	Data Type	Initial Value	Function Description
InterfaceID	UINT	0	Hardware identifier for local interface (network card number): 0-1, default 0
ID	WORD	0	Connection ID (value range: 1 ~ 64) This parameter uniquely determines the connection within the CPU, and

			each MB_CLIENT instance must use a unique ID
ConnectionType	BYTE	11	Connection type: MB_CLIENT fixed to 11 for TCP
ActiveEstablished	BOOL	1	Active connection establishment, should be set to TRUE
RemoteAddress	ARRAY [0..3] of BYTE	0.0.0.0	The IP address of the Modbustcp server, for example 192.168.0.250: addr[0] = 192 addr[1] = 168 addr[2] = 0 addr[3] = 250
RemotePort	UINT	502	Modbustcp server communication port number (default: 502) Port number: 1 ~ 49151 (except for unusable ports listed above)
LocalPort	UINT	0	Local port number (client need not specify) Port number: 1 ~ 49151 (except for unusable ports listed above) Any port: 0; default 0, no port number specified

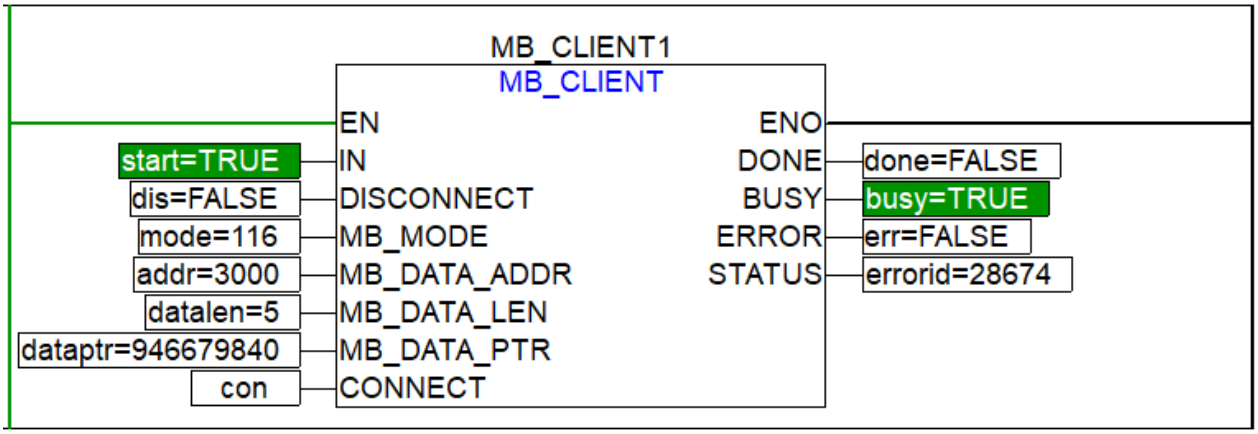
7.1.1.3 Error code

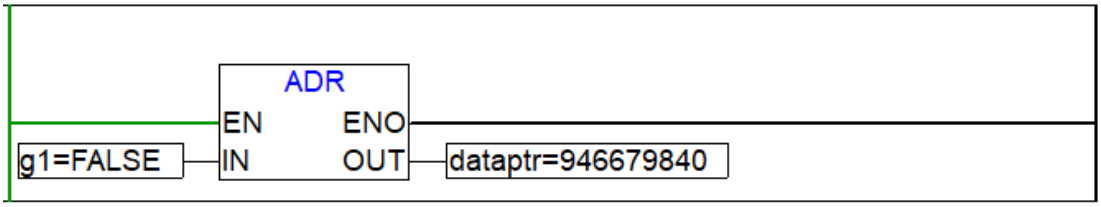
STATUS(16#)	Description
Parameter STATUS (general status information)	
0000	Instruction executed without any errors
0001	Connection established
0003	Connection terminated
7000	Job not activated and connection not established (IN = 0, DISCONNECT = 1)
7001	Connection establishment operation triggered
7002	Intermediate call, establishing connection
7004	The connection was established and is monitored. No job processing activated
7005	Sending data
7006	Receiving data
Parameter STATUS（Protocol error）	
809B	Incorrect interfaceId in connection description
80A4	The IP address of the remote connection endpoint is invalid, that is, it is a duplicate of the IP address of the local partner
80A5	Connection ID is already taken The connection ID is invalid
80C5	The communication partner has actively closed the connection
80C8	1、 During the specified time period, the server did not respond. Check connectivity to the Modbus server. The error is reported only after the number of repeatable attempts for the configuration 2、 Output the error code if the "MB_CLIENT" instruction does not receive a reply from the initial transmission transaction ID within the specified time limit (see the static variable MB_TRANSACTION_ID)
8380	Modbus frame format error or too few bytes received
8383	Error reading/writing data or accessing beyond MB_DATA_PTR address
8384	1、 An invalid exception code was received 2、 The data values received are different from those initially sent by the client (functions 5, 6 and 8) 3、 Received invalid status value (function 11)
8386	The function code received is inconsistent with the code initially sent
8388	The Modbus server sends a different length of data than requested. This error occurs only if you use Modbus Function Code 5, 6, 15, or 16
Parameter STATUS（Parameter error）	
80B6	Invalid connection type, only TCP connections are supported
80BB	Invalid value for ActiveEstablished parameter (identifier to establish such a connection, see 'CONNECT

	parameter'): 1、 Only passive connections to servers are allowed (ActiveEstablished = FALSE) 2、 Only active connections to clients are allowed (ActiveEstablished = TRUE)
8188	Invalid value for parameter MB_MODE
8189	Invalid data address in parameter MB_DATA_ADDR
818A	Invalid data length in parameter MB_DATA_LEN
818B	Invalid pointer at parameter MB_DATA_PTR. In addition, check the values of the parameters MB_DATA_ADDR and MB_DATA_LEN
818C	The parameters BLOCKKED_PROC_TIMEOUT or RCV_TIMEOUT timeout (see static variables for instructions) must be between 0.5 s and 55 s
9000	Remote Port Error
9001	Local Port Error
9002	Transaction ID error
9003	Incorrect unit ID
9004	Function block and hardware configuration cannot be used together

7.1.1.4 Example

No.	Variable Name	Address	ariable Descriptio	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	MB_CLIENT1			MB_CLIENT		FALSE	Not Public	
0002	start			BOOL	TRUE	FALSE	Not Public	
0003	dis			BOOL	FALSE	FALSE	Not Public	
0004	mode			USINT	116	FALSE	Not Public	
0005	addr			UDINT	3000	FALSE	Not Public	
0006	datalen			UINT	5	FALSE	Not Public	
0007	dataptr			POINTER TO BYTE	946679840	FALSE	Not Public	
0008	con			CONNECT_IPV4		FALSE	Not Public	
0009	done			BOOL	FALSE	FALSE	Not Public	
0010	busy			BOOL	TRUE	FALSE	Not Public	
0011	err			BOOL	FALSE	FALSE	Not Public	
0012	errorid			WORD	28674	FALSE	Not Public	
0013	g1			BOOL	FALSE	FALSE	Not Public	





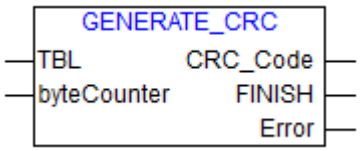

```
1  MB_CLIENT1(
2  IN := start,
3  DISCONNECT := dis,
4  MB_MODE := mode,
5  MB_DATA_ADDR := addr,
6  MB_DATA_LEN := datalen,
7  MB_DATA_PTR := dataptr,
8  CONNECT := con,
9  DONE=>done,
10 BUSY=>busy,
11 ERROR=>err,
12 STATUS=>errorid);
13 dataptr := ADR(g1);
```

```
MB_CLIENT1
start = TRUE
dis = FALSE
mode = 116
addr = 3000
datalen = 5
dataptr = 946679840
con
done = FALSE
busy = TRUE
err = FALSE
errorid = 28674
dataptr = 946679840          g1 = FALSE
```

7.1.2 GENERATE_CRC (Cyclic Redundancy Check)

7.1.2.1 Function

Generate a 16-bit CRC check value.



GENERATE_CRC Functional Block

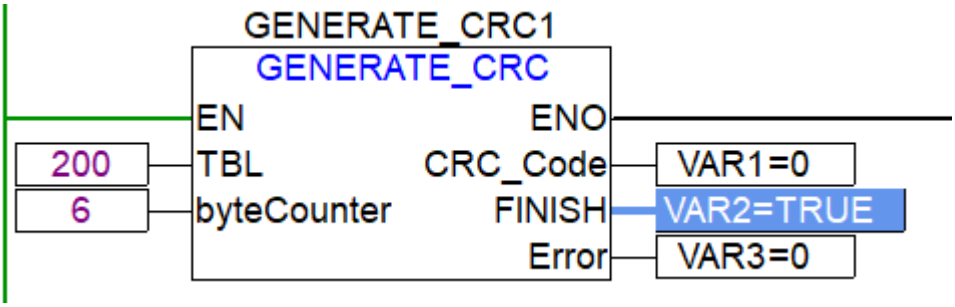
7.1.2.2 Parameter

Parameter	Statement	Data Type	Function Description	Description of Parameters	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input			
ENO	Output	BOOL	Enable output			
TBL	Input	WORD	First byte address of the data to be checked	If it is set to 200, the data to be checked is stored in the address starting from %MB200	0	FALSE
byteCounter	Input	WORD	Number of bytes of the data to be checked	0 is an invalid value	0	FALSE
CRC_Code	Output	WORD	Check result	16-bit	0	FALSE
FINISH	Output	BOOL	Check completed or not	0: Not completed 1: Completed	FALS E	FALSE
Error	Output	BYTE	Error message	0: Correct 1: Number of bytes of the data to be checked byteCounter=0 2: The data to be checked exceeds the range of area M	0	FALSE

7.1.2.3 Example

For example, we can use the CRC check calculation instruction to calculate the 6-byte CRC check code starting from %MB200. Then, according to actual communication needs, we can send the CRC check code after other data to be sent.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	GENERATE_CRC1			GENERATE_CRC		FALSE	Not Public	
0002	VAR1			WORD	0	FALSE	Not Public	
0003	VAR2			BOOL	TRUE	FALSE	Not Public	
0004	VAR3			BYTE	0	FALSE	Not Public	




```
1  GENERATE_CRC1(
2  TBL := 200,
3  byteCounter := 6,
4  CRC_Code=>VAR1,
5  FINISH=>VAR2,
6  Error=>VAR3);
```

```
GENERATE_CRC1

VAR1 = 0
VAR2 = TRUE
VAR3 = 0
```

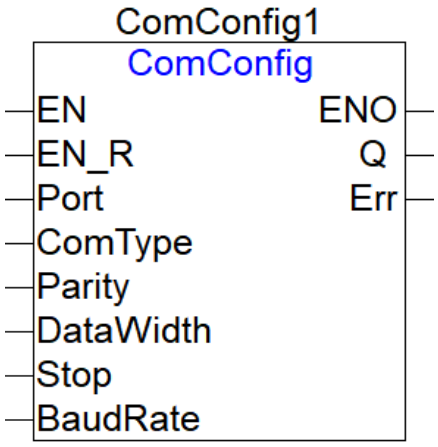
Program description:

We can use the CRC check calculation instruction to calculate the 6-byte CRC check code starting from %MB200. Then, according to actual communication needs, we can send the CRC check code after other data to be sent.

7.1.3 ComConfig (Setting Serial Port Communication Parameter)

7.1.3.1 Function

Set the controller serial port communication parameters. This function currently only supports online configuration of communication parameters for each channel.



ComConfig Functional Block

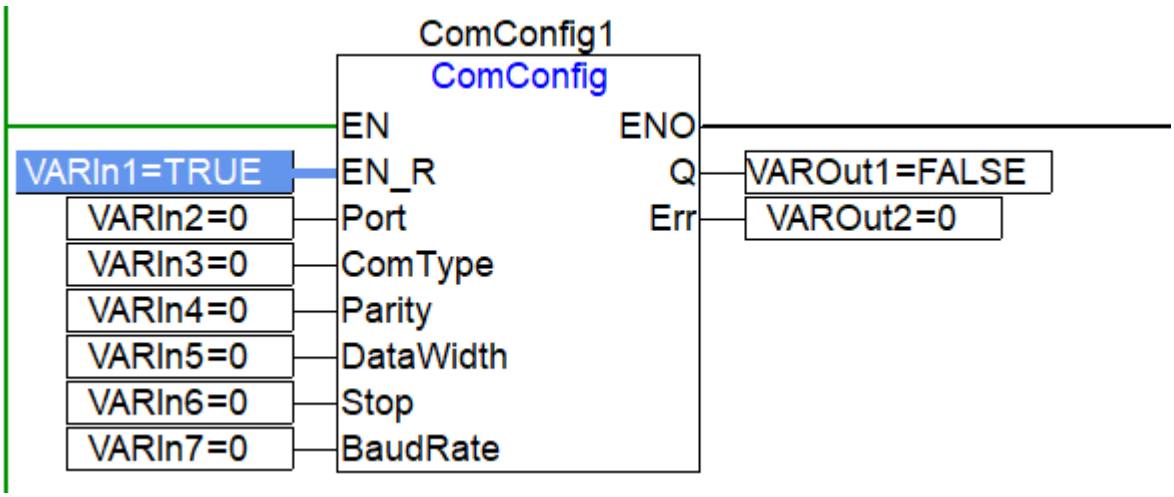
7.1.3.2 Parameter

Parameter	Statement	Data Type	Function Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
EN_R	Input	BOOL	Rising edge enable	0	FALSE
Port	Input	USINT	Selective serial port 0: UART1	0	FALSE
ComType	Input	USINT	Communication type 0: RS-485	0	FALSE
Parity	Input	USINT	Check mode 0: No check 1: Even parity check 2: Odd parity check	0	FALSE
Data Width	Input	USINT	Data width (supporting 8 bits) 8: Data width of 8 bits	0	FALSE
Stop	Input	USINT	Stop bit (supporting 1 and 2) 1: Stop bit 1 2: Stop bit 2	0	FALSE
Baud Rate	Input	UDINT	Baud rate (bps) 1200bps/2400bps/4800bps/9600bps/19200bps/38400bps/57600bps/115200bps	0	FALSE
Q	Output	BOOL	Output terminal Terminal Q is TRUE, which is only used to indicate that the instruction has been executed in this task cycle.	FALSE	FALSE
Err	Output	USINT	Error code 0: No error 1: ComType error 2: Port error 4: Parity error 5: DataWidth error 6: Stop error 7: BaudRate error	0	FALSE

7.1.3.3 Example

For example, the following example uses the ComConfig instruction to set serial port communication parameters. When the input EN_R detects a rising edge, the communication parameters of each channel are written.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	ComConfig1			COMCONFIG		FALSE	Not Public	
0002	VARIn1			BOOL	TRUE	FALSE	Not Public	
0003	VARIn2			USINT	0	FALSE	Not Public	
0004	VARIn3			USINT	0	FALSE	Not Public	
0005	VARIn4			USINT	0	FALSE	Not Public	
0006	VARIn5			USINT	0	FALSE	Not Public	
0007	VARIn6			USINT	0	FALSE	Not Public	
0008	VARIn7			UDINT	0	FALSE	Not Public	
0009	VAROut1			BOOL	FALSE	FALSE	Not Public	
0010	VAROut2			USINT	0	FALSE	Not Public	



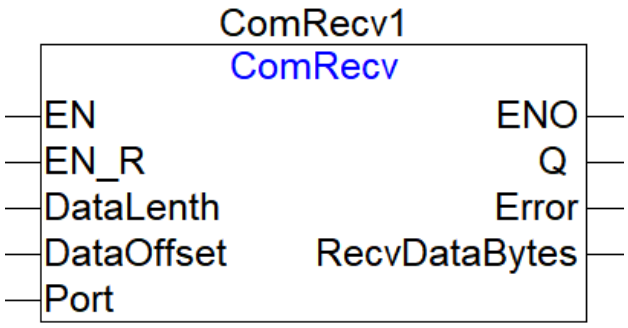

```
1  ComConfig1(
2  EN_R := VARIn1,
3  Port := VARIn2,
4  ComType := VARIn3,
5  Parity := VARIn4,
6  DataWidth := VARIn5,
7  Stop := VARIn6,
8  BaudRate := VARIn7,
9  Q=>VAROut1,
10 Err=>VAROut2);
```

```
ComConfig1
VARIn1 = TRUE
VARIn2 = 0
VARIn3 = 0
VARIn4 = 0
VARIn5 = 0
VARIn6 = 0
VARIn7 = 0
VAROut1 = FALSE
VAROut2 = 0
```

7.1.4 ComRecv (Receive Serial Port Data)

7.1.4.1 Function

Used to receive controller serial port data.



ComRecv Functional Block

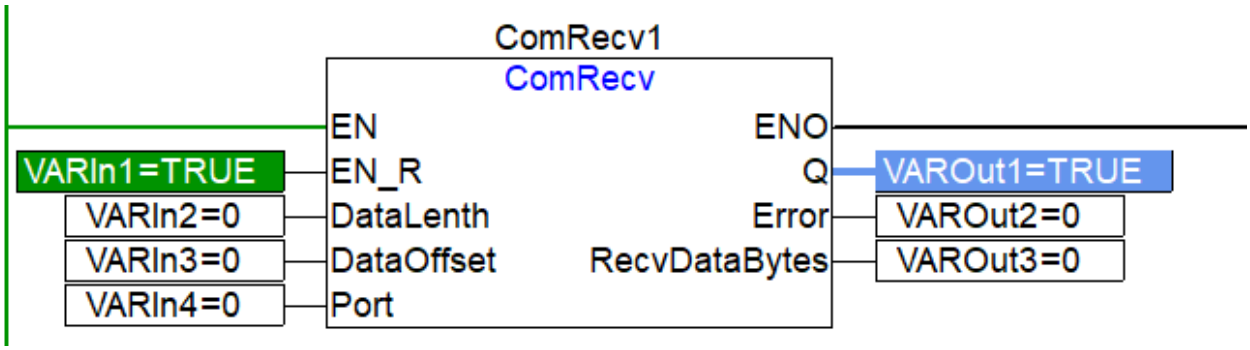
7.1.4.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
EN_R	Input	BOOL	High level valid	FALSE	FALSE
DataLenth	Input	UDINT	Data length to be received (in Byte)	0	FALSE
DataOffset	Input	UDINT	Offset of area M where the data address is located (in Byte), which needs to increase the number of bytes of the received data	0	FALSE
Port	Input	USINT	0: UART1, supports RS-485 serial port	0	FALSE
Q	Output	BOOL	The output terminal Q is valid, which is only used to indicate that the instruction has been executed in this task cycle.	FALSE	FALSE
Err	Output	USINT	Error code 0: No error 1: The input parameter configuration is incorrect 2: This port is not configured as a free port protocol	0	FALSE
RecvData Bytes	Output	UDINT	Actual number of bytes received	0	FALSE

7.1.4.3 Example

When EN_R is set, the offset data of area M where the data address is located is received according to the set data length to be received.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	ComRecv1			COMRECV		FALSE	Not Public	
0002	VARIn1			BOOL	TRUE	FALSE	Not Public	
0003	VAROut2			USINT	0	FALSE	Not Public	
0004	VARIn2			UDINT	0	FALSE	Not Public	
0005	VARIn3			UDINT	0	FALSE	Not Public	
0006	VARIn4			USINT	0	FALSE	Not Public	
0007	VAROut1			BOOL	TRUE	FALSE	Not Public	
0008	VAROut3			UDINT	0	FALSE	Not Public	



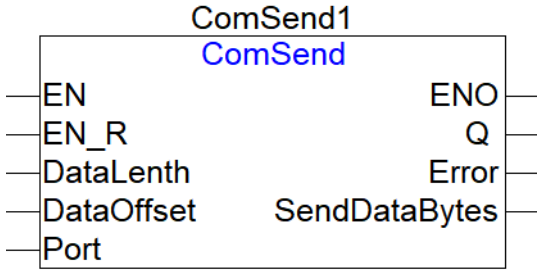

```
1  ComRecv1(
2  EN_R := VARIn1,
3  DataLenth := VARIn2,
4  DataOffset := VARIn3,
5  Port := VARIn4,
6  Q=>VAROut1,
7  Error=>VAROut2,
8  RecvDataBytes=>VAROut3);
```

```
ComRecv1
VARIn1 = TRUE
VARIn2 = 0
VARIn3 = 0
VARIn4 = 0
VAROut1 = TRUE
VAROut2 = 0
VAROut3 = 0
```

7.1.5 ComSend (Send Serial Port Data)

7.1.5.1 Function

Used to send controller serial port data.



ComSend Functional Block

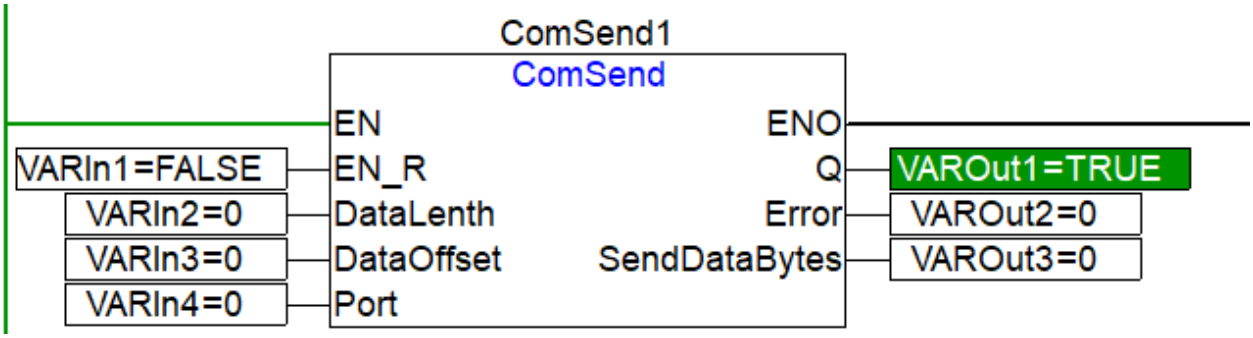
7.1.5.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
En_R	Input	BOOL	Rising edge enable	FALSE	FALSE
DataLenth	Input	UDINT	Data length to send	0	FALSE
DataOffset	Input	UDINT	Offset of area M where the data address is located	0	FALSE
Port	Input	USINT	0: UART1, supports RS-485 serial port	0	FALSE
Q	Output	BOOL	The output terminal Q is valid, which is only used to indicate that the instruction has been executed in this task cycle.	FALSE	FALSE
Err	Output	USINT	Error code 0: No error 1: The input parameter configuration is incorrect 2: This port is not configured as a free port protocol	0	FALSE
SendDataBytes	Output	UDINT	Actual number of bytes sent	0	FALSE

7.1.5.3 Example

When EN_R is set, the offset data of area M where the data address is located is sent according to the set data length to be sent.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	ComSend1			COMSEND		FALSE	Not Public	
0002	VARIn1			BOOL	FALSE	FALSE	Not Public	
0003	VARIn2			UDINT	0	FALSE	Not Public	
0004	VARIn3			UDINT	0	FALSE	Not Public	
0005	VARIn4			USINT	0	FALSE	Not Public	
0006	VAROut1			BOOL	TRUE	FALSE	Not Public	
0007	VAROut2			USINT	0	FALSE	Not Public	
0008	VAROut3			UDINT	0	FALSE	Not Public	



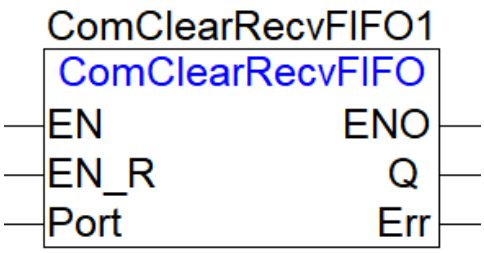

```
1  ComSend1(
2  EN_R := VARIn1,
3  DataLenth := VARIn2,
4  DataOffset := VARIn3,
5  Port := VARIn4,
6  Q=>VAROut1,
7  Error=>VAROut2,
8  SendDataBytes=>VAROut3);
```

```
ComSend1
VARIn1 = FALSE
VARIn2 = 0
VARIn3 = 0
VARIn4 = 0
VAROut1 = TRUE
VAROut2 = 0
VAROut3 = 0
```

7.1.6 ComClearRecvFIFO (Clear Serial Port Data)

7.1.6.1 Function

Clear controller serial port Rx data.



ComClearRecvFIFO Functional Block

7.1.6.2 Parameter

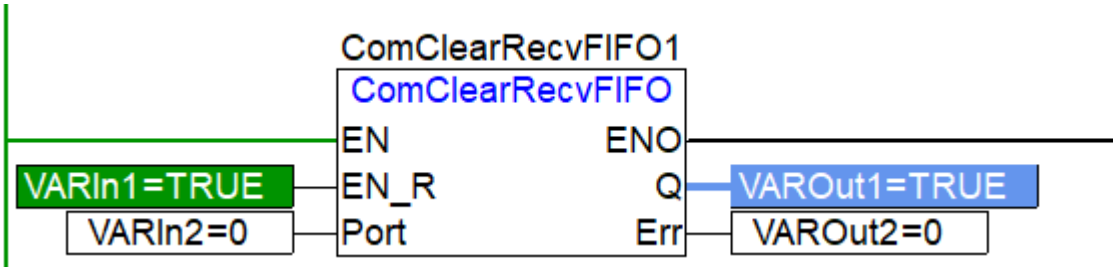
Parameter	Statement	Data	Description	Initial	Power Fail

		Type		Value	Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
EN_R	Input	BOOL	Rising edge enable	FALSE	FALSE
Port	Input	BYTE	0: UART1, supports RS-485 serial port	0	FALSE
Q	Output	BOOL	Output	FALSE	FALSE
Err	Output	USINT	Error 0: No error 1: Port configuration error 14: Backboard communication failed or clearing failed	0	FALSE

7.1.6.3 Example

Clear the selected serial port data when EN_R is set.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	ComClearRecvFIFO1			COMCLEARRECVFIFO		FALSE	Not Public	
0002	VARIn1			BOOL	TRUE	FALSE	Not Public	
0003	VARIn2			USINT	0	FALSE	Not Public	
0004	VAROut1			BOOL	TRUE	FALSE	Not Public	
0005	VAROut2			USINT	0	FALSE	Not Public	



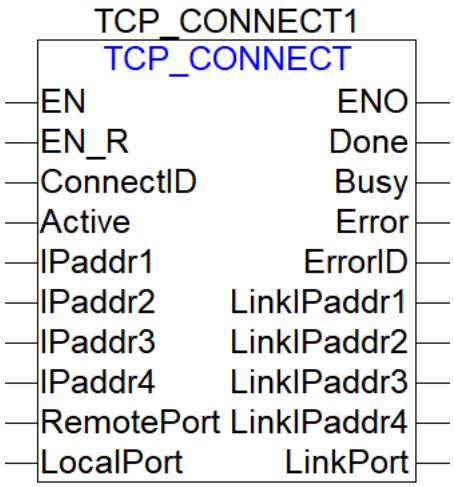

```
1  ComClearRecvFIFO1(
2  EN_R := VARIn1,
3  Port := VARIn2,
4  Q=>VAROut1,
5  Err=>VAROut2);
```

```
ComClearRecvFIFO1
VARIn1 = TRUE
VARIn2 = 0
VAROut1 = TRUE
VAROut2 = 0
```

7.1.7 TCP_CONNECT (Establishing a TCP Communication Connection)

7.1.7.1 Function

Establish TCP communication connection. Independent network card configuration is not supported.



TCP_CONNECT Functional Block

The connection operation is asynchronous and may require several scans to complete. When a connection operation is pending, the Busy output value is TRUE. Once the CPU completes the operation, it sets the Done or Error output. If an error occurs, the ErrorID output displays the error code.

While the command is busy, the input parameters of TCP_CONNECT must not be changed. The CPU relies on this to understand that this is a continuation of the call to initiate the connection process.

Assign the connection ID (ConnectID) input to the connection, then use this ConnectID to reference the connection when sending, receiving, or disconnecting.

The Active input bit determines whether this is an active connection (Active set to TRUE) or a passive connection (Active set to FALSE).

When the Active input is set to TRUE, the CPU creates an active (client) connection, attempting to contact and establish a connection to the specified IP address and remote port number (RemotePort). The CPU opens the local port (LocalPort) to receive messages from the remote device.

When the Active input is set to FALSE, the CPU creates a passive (server) connection. In this case, the CPU opens the requested local port (LocalPort) and accepts connection requests from remote devices. To accept connection requests from any remote IP address, set the IP address to 0.0.0.0. If the IP address is not zero, the CPU only accepts connection requests from the specified IP address. For a passive connection, the CPU ignores the remote port number (RemotePort), which can be set to zero.

The TCP_CONNECT instruction can be called at any time to determine the current status of the connection. Setting the EN_R input to FALSE and providing a valid connection ID (ConnectID), TCP_CONNECT returns the following:

- Busy: If the connection process is still in progress.
- Done: If the connection is active and ready for sending or receiving data.
- Error: If the connection is not available. The ErrorID will display one of the error codes, indicating the issue encountered.
- LinkIPaddr1: Outputs the first octet of the IP address of the currently connected device.
- LinkIPaddr2: Outputs the second octet of the IP address of the currently connected device.

- LinkIPaddr3: Outputs the third octet of the IP address of the currently connected device.
- LinkIPaddr4: Outputs the fourth octet of the IP address of the currently connected device.
- LinkPort: Outputs the port number of the currently connected device.

- Please Note:

  An active connection may take a minimum of 30 seconds to determine whether the remote device allows the connection. A passive connection will show the Busy status until the remote device attempts to connect to the CPU.

  For an active connection, if the EN_R remains high and the remote device disconnects, the CPU will automatically attempt to reconnect to the remote device.

  After a connection is closed, the CPU will not automatically attempt to reconnect to the device, regardless of whether it was an active or passive connection.

  Active Connection:

  - If the LocalPort is set to 0, the connection can be re-established immediately after it is actively disconnected, without any waiting time.
  - If the LocalPort is set to a non-zero value, the connection must wait at least 30 seconds after it is actively disconnected before it can be re-established successfully.

7.1.7.2 Parameter

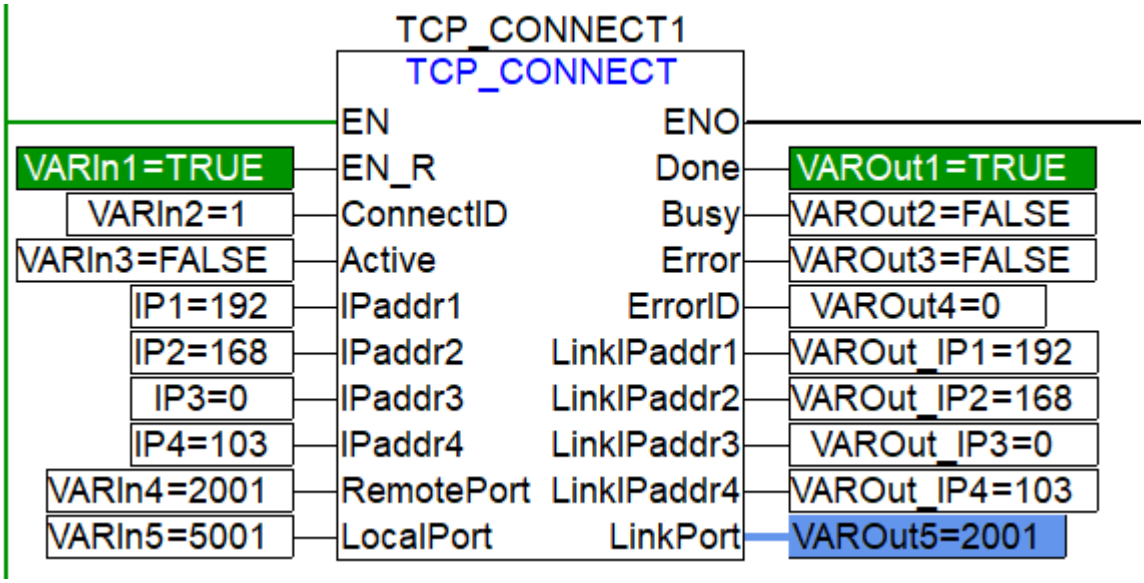
Parameter	Statement	Data Type	Parameter Description	Initial Value	Power Fail Safeguard
EN_R	Input	BOOL	Rising edge: The CPU starts connection operation. When EN_R is FALSE, the output displays the current connection status	0	FALSE
ConnectID	Input	BYTE	Socket connection number 1~16	0	FALSE
Active	Input	BOOL	TRUE: Create an active client connection FALSE: Create a passive server connection	0	FALSE
IPaddr1	Input	BYTE	IP address digit 1	0	FALSE
IPaddr2	Input	BYTE	IP address digit 2	0	FALSE
IPaddr3	Input	BYTE	IP address digit 3	0	FALSE
IPaddr4	Input	BYTE	IP address digit 4	0	FALSE
RemotePort	Input	WORD	Port number on the remote device. The range of remote port numbers is from 1 to 49151. For passive connections, zero is used.	0	FALSE
LocalPort	Input	WORD	Port number on the local device. The range of local port numbers is 1 to 49151, but there are some restrictions. For details, see the parameter LocalPort in the Common parameters of the Ethernet free protocol library instruction.	0	FALSE
Done	Output	BOOL	If the connection is active and ready to send or receive	0	FALSE
Busy	Output	BOOL	If the connection process is still in progress	0	FALSE
Error	Output	BOOL	If the connection is not available. ErrorID displays one of the error codes to indicate the problem	0	FALSE
ErrorID	Output	WORD	If the instruction sets the Error output, the ErrorID output will display the error code. If the instruction sets the Busy or Done output, ErrorID is zero (no error).	0	FALSE

			For details, see the Error codes of Ethernet free protocol communication instructions.		
LinkIPaddr1	Output	BYTE	Four octets of the IP address of the connected remote device. LinkIPaddr1 is the most significant byte of the IP address, and LinkIPaddr4 is the least significant byte of the IP address.	Hidden by default	FALSE
LinkIPaddr2	Output	BYTE			FALSE
LinkIPaddr3	Output	BYTE			FALSE
LinkIPaddr4	Output	BYTE			FALSE
LinkPort	Output	WORD	Port number of connected remote device	0	FALSE

7.1.7.3 Example

When EN_R is on, the remote device is connected according to the set IP address and port number of the device.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	TCP_CONNECT1			TCP_CONNECT		FALSE	Not Public	
0002	VARIn1			BOOL	TRUE	FALSE	Not Public	
0003	VARIn2			BYTE	1	FALSE	Not Public	
0004	VARIn3			BOOL	FALSE	FALSE	Not Public	
0005	IP1			BYTE	192	FALSE	Not Public	
0006	IP2			BYTE	168	FALSE	Not Public	
0007	IP3			BYTE	0	FALSE	Not Public	
0008	IP4			BYTE	103	FALSE	Not Public	
0009	VARIn4			WORD	2001	FALSE	Not Public	
0010	VARIn5			WORD	5001	FALSE	Not Public	
0011	VAROut1			BOOL	TRUE	FALSE	Not Public	
0012	VAROut2			BOOL	FALSE	FALSE	Not Public	
0013	VAROut3			BOOL	FALSE	FALSE	Not Public	
0014	VAROut4			WORD	0	FALSE	Not Public	
0015	VAROut_IP1			BYTE	192	FALSE	Not Public	
0016	VAROut_IP2			BYTE	168	FALSE	Not Public	
0017	VAROut_IP3			BYTE	0	FALSE	Not Public	
0018	VAROut_IP4			BYTE	103	FALSE	Not Public	
0019	VAROut5			WORD	2001	FALSE	Not Public	



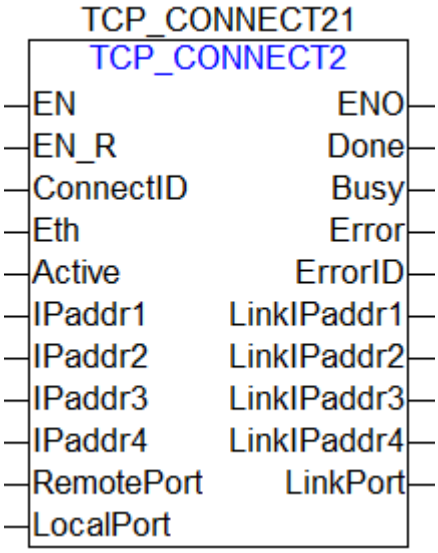
```
1  TCP_CONNECT1(
2  EN_R := VARIn1,
3  ConnectID := VARIn2,
4  Active := VARIn3,
5  IPaddr1 := IP1,
6  IPaddr2 := IP2,
7  IPaddr3 := IP3,
8  IPaddr4 := IP4,
9  RemotePort := VARIn4,
10 LocalPort := VARIn5,
11 Done=>VAROut1,
12 Busy=>VAROut2,
13 Error=>VAROut3,
14 ErrorID=>VAROut4,
15 LinkIPaddr1=>VAROut_IP1,
16 LinkIPaddr2=>VAROut_IP2,
17 LinkIPaddr3=>VAROut_IP3,
18 LinkIPaddr4=>VAROut_IP4,
19 LinkPort=>VAROut5);
```

```
TCP_CONNECT1
VARIn1 = TRUE
VARIn2 = 1
VARIn3 = FALSE
IP1 = 192
IP2 = 186
IP3 = 0
IP4 = 103
VARIn4 = 2001
VARIn5 = 5002
VAROut1 = TRUE
VAROut2 = FALSE
VAROut3 = FALSE
VAROut4 = 0
VAROut_IP1 = 192
VAROut_IP2 = 168
VAROut_IP3 = 0
VAROut_IP4 = 103
VAROut5 = 2001
```

7.1.8 TCP_CONNECT2(Establishing a TCP Communication Connection(support communication by specific nices))

7.1.8.1 Function

When setting up a TCP communication connection and using a separate network card, it is necessary to specify the network card and configure the free protocol under the corresponding network card.



TCP_CONNECT2 Functional Block

- LX-CU430 do not support network card independent configuration.

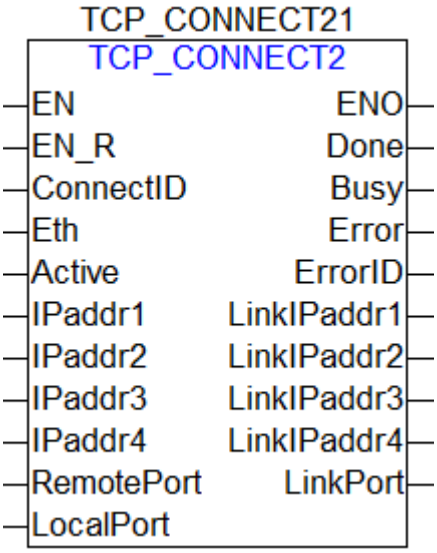
7.1.8.2 Parameter

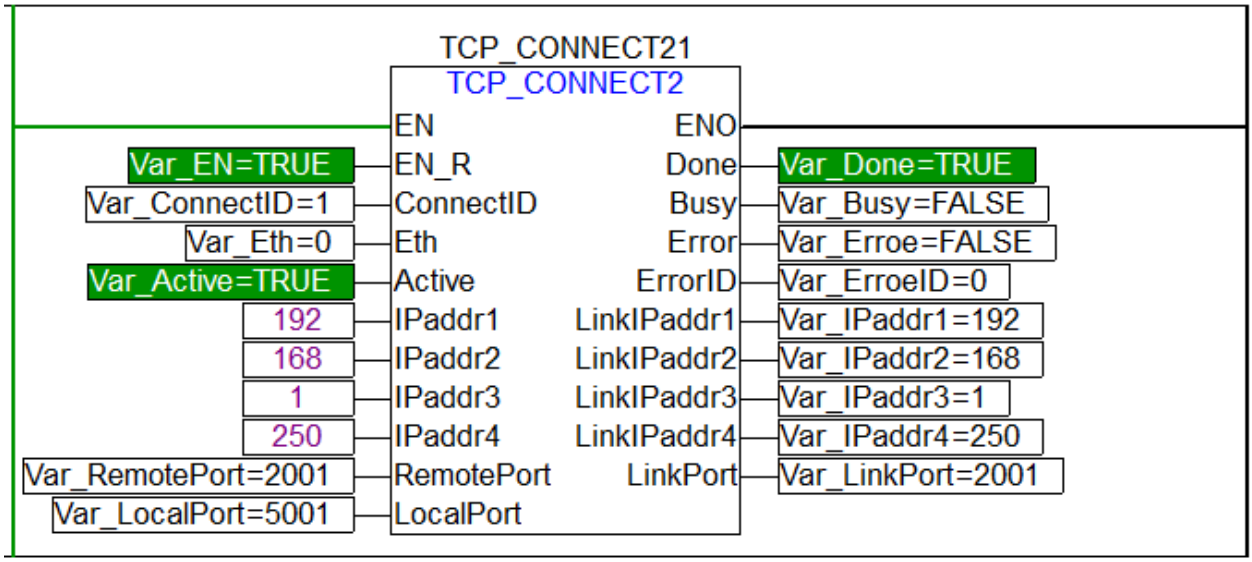
Parameter	Statement	Data Type	Parameter Description	Initial Value	Power Fail Safeguard
EN_R	Input	BOOL	Rising edge: The CPU starts connection operation.	FALSE	FALSE
ConnectID	Input	BYTE	Socket connection number 1~16	0	FALSE
Eth	Input	BYTE	Network card ID number 0: P1 Interface Card, EtherNet1 1: P2 interface card, EtherNet2	0	FALSE
Active	Input	BOOL	TRUE: Create an active client connection FALSE: Create a passive server connection	0	FALSE
IPaddr1	Input	BYTE	IP address digit 1	0	FALSE
IPaddr2	Input	BYTE	IP address digit 2	0	FALSE
IPaddr3	Input	BYTE	IP address digit 3	0	FALSE
IPaddr4	Input	BYTE	IP address digit 4	0	FALSE
RemotePort	Input	WORD	Port number on the remote device. The range of remote port numbers is from 1 to 49151. For passive connections, zero is used.	0	FALSE
LocalPort	Input	WORD	Port number on the local device. The range of local port numbers is 1 to 49151, but there are some restrictions. For details, see the parameter LocalPort in the Common parameters of the Ethernet free protocol library instruction.	0	FALSE
Done	Output	BOOL	If the connection is active and ready to send or receive	0	FALSE
Busy	Output	BOOL	If the connection process is still in progress	0	FALSE
Error	Output	BOOL	If the connection is not available. ErrorID displays one of the error codes to indicate the problem	0	FALSE
ErrorID	Output	WORD	If the instruction sets the Error output, the ErrorID output will display the error code. If the instruction sets the Busy or Done output, ErrorID is zero (no error).	0	FALSE

			For details, see the Error codes of Ethernet free protocol communication instructions.		
LinkIPaddr1	Output	BYTE	Four octets of the IP address of the connected remote device. LinkIPaddr1 is the most significant byte of the IP address, and LinkIPaddr4 is the least significant byte of the IP address.	Hidden by default	FALSE
LinkIPaddr2	Output	BYTE			FALSE
LinkIPaddr3	Output	BYTE			FALSE
LinkIPaddr4	Output	BYTE			FALSE
LinkPort	Output	WORD	Port number of connected remote device	0	FALSE

7.1.8.3 Example

When EN_R is on, the remote device is connected according to the set IP address and port number of the device.


TCP_CONNECT21
TCP_CONNECT2
EN
EN_R
ConnectID
Eth
Active
IPaddr1
IPaddr2
IPaddr3
IPaddr4
RemotePort
LocalPort
ENO
Done
Busy
Error
ErrorID
LinkIPaddr1
LinkIPaddr2
LinkIPaddr3
LinkIPaddr4
LinkPort

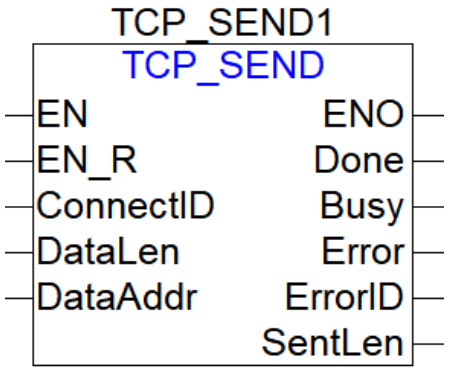

TCP_CONNECT21
TCP_CONNECT2
Var_EN=TRUE
Var_ConnectID=1
Var_Eth=0
Var_Active=TRUE
192
168
1
250
Var_RemotePort=2001
Var_LocalPort=5001
EN
EN_R
ConnectID
Eth
Active
IPaddr1
IPaddr2
IPaddr3
IPaddr4
RemotePort
LocalPort
ENO
Done
Busy
Error
ErrorID
LinkIPaddr1
LinkIPaddr2
LinkIPaddr3
LinkIPaddr4
LinkPort
Var_Done=TRUE
Var_Busy=FALSE
Var_Erroe=FALSE
Var_ErroeID=0
Var_IPaddr1=192
Var_IPaddr2=168
Var_IPaddr3=1
Var_IPaddr4=250
Var_LinkPort=2001

```
1  TCP_CONNECT21(                    TCP_CONNECT21
2  EN_R := Var_EN,                   Var_EN = TRUE
3  ConnectID := Var_ConnectID,       Var_ConnectID = 1
4  Eth := Var_Eth,                   Var_Eth = 0
5  Active := Var_Active,             Var_Active = FALSE
6  IPaddr1 := 192,
7  IPaddr2 := 168,
8  IPaddr3 := 1,
9  IPaddr4 := 250,
10 RemotePort := Var_RemotePort,     Var_RemotePort = 2001
11 LocalPort := Var_LocalPort,       Var_LocalPort = 5001
12 Done=>Var_Done,                   Var_Done = TRUE
13 Busy=>Var_Busy,                   Var_Busy = FALSE
14 Error=>Var_Erroe,                 Var_Erroe = FALSE
15 ErrorID=>Var_ErroeID,             Var_ErroeID = 0
16 LinkIPaddr1=>Var_IPaddr1,         Var_IPaddr1 = 192
17 LinkIPaddr2=>Var_IPaddr2,         Var_IPaddr2 = 168
18 LinkIPaddr3=>Var_IPaddr3,         Var_IPaddr3 = 1
19 LinkIPaddr4=>Var_IPaddr4,         Var_IPaddr4 = 250
20 LinkPort=>Var_LinkPort);          Var_LinkPort = 2001
```

7.1.9 TCP_SEND (Send Data via tcp protocol)

7.1.9.1 Function

Use TCP protocol for Ethernet data sending.



TCP_SEND Functional Block

The TCP_SEND command initiates the operation to send a specified number of bytes when the following conditions occur:

- The program calls the instruction by setting the EN_R input to TRUE.
- The connection is not currently being used to perform another send operation.
- The EN_R input uses a rising-edge trigger to prevent the instruction from initiating unintended send operations. When TCP_SEND is in a busy state, the program ignores the EN_R input. The Done, Busy,

and Error outputs, along with the ErrorID output byte, indicate the status of each TCP_SEND call.

- After the send operation completes, the instruction indicates the Done or Error status for the single call of TCP_SEND. Thereafter, TCP_SEND responds with an error code (error description: "No pending operations") if called with EN_R set to FALSE. If EN_R is set to TRUE, the program initiates another send operation. The relationship between input and output parameters is described below:
- EN_R is set to TRUE to start the message send operation. Busy is set to TRUE.
- Message sending is complete. Done is set, and Busy is cleared.
- EN_R is FALSE, meaning no message send operation is being performed.
- EN_R is set to TRUE again, so another message send operation begins. Busy is set to TRUE.
- Message sending is complete. Done is set, and Busy is cleared within one scan cycle.

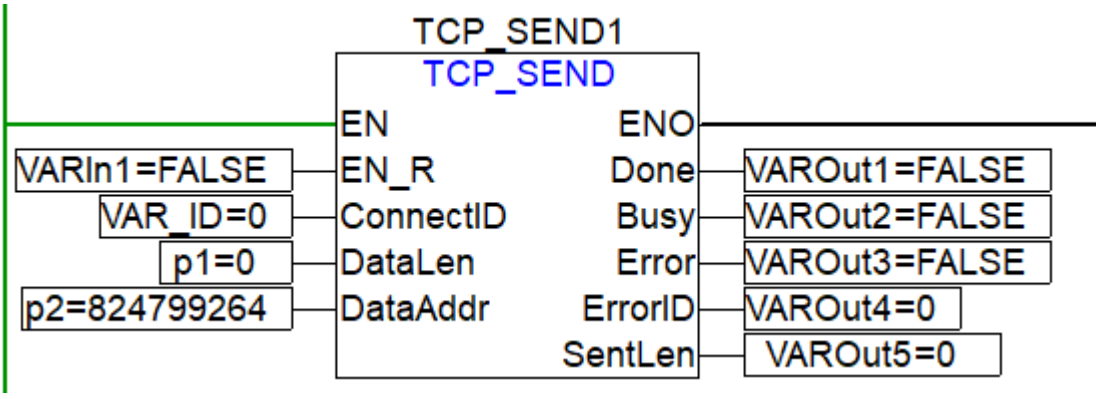
7.1.9.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN_R	Input	BOOL	Rising edge: The CPU starts sending operation.	FALSE	FALSE
ConnectID	Input	BYTE	The connection ID (ConnectID) is the number of the connection used by this send operation. Use the ConnectID you selected for the TCP_CONNECT operation.	0	FALSE
DataLen	Input	WORD	Data length requested to send, (1~32*1024)Byte	0	FALSE
DataAddr	Input	POINTER TO BYTE	DataAddr is a pointer to the storage location of the sent data	0	FALSE
Done	Output	BOOL	When the send operation is completed without errors, the instruction sets the Done output	FALSE	FALSE
Busy	Output	BOOL	When the send operation is in progress, the instruction sets the Busy output	FALSE	FALSE
Error	Output	BOOL	When the sending operation is completed but an error occurs, the instruction sets the Error output	FALSE	FALSE
ErrorID	Output	WORD	If the instruction sets the Error output, the ErrorID output will display the error code. If the instruction sets the Busy or Done output, ErrorID is zero (no error). For details, see the Error codes of Ethernet free protocol communication instructions.	0	FALSE
SentLen	Output	WORD	SentLen is the actual number of bytes sent. SentLen is valid only when the instruction sets the Done or Error output. If the instruction sets the Done output, the entire message information is sent.	0	FALSE

7.1.9.3 Example

When EN_R is set, the data at the data address pointed to by DataAddr is sent according to the set sending length.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	TCP_SEND1			TCP_SEND		FALSE	Not Public	
0002	VARIn1			BOOL	FALSE	FALSE	Not Public	
0003	VAR_ID			BYTE	0	FALSE	Not Public	
0004	p1			WORD	0	FALSE	Not Public	
0005	p2			POINTER TO BYTE	824799264	FALSE	Not Public	
0006	VAROut1			BOOL	FALSE	FALSE	Not Public	
0007	VAROut2			BOOL	FALSE	FALSE	Not Public	
0008	VAROut3			BOOL	FALSE	FALSE	Not Public	
0009	VAROut4			WORD	0	FALSE	Not Public	
0010	VAROut5			WORD	0	FALSE	Not Public	



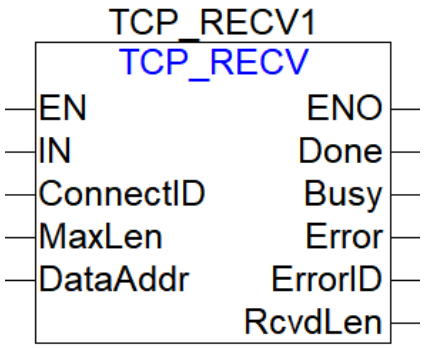

```
1  TCP_SEND1(
2  EN_R := VARIn1,
3  ConnectID := VAR_ID,
4  DataLen := p1,
5  DataAddr := p2,
6  Done=>VAROut1,
7  Busy=>VAROut2,
8  Error=>VAROut3,
9  ErrorID=>VAROut4,
10 SentLen=>VAROut5);
11
```

```
TCP_SEND1
VARIn1 = FALSE
VAR_ID = 0
p1 = 0
p2 = 824799264
VAROut1 = FALSE
VAROut2 = FALSE
VAROut3 = FALSE
VAROut4 = 0
VAROut5 = 0
```

7.1.10 TCP_RECV (Receive Data via tcp protocol)

7.1.10.1 Function

Use the TCP protocol for Ethernet data reception.



TCP_RECV Functional Block

7.1.10.2 Parameter

Parameter	Statement	Data Type	Parameter Description	Initial Value	Power Fail Safeguard
IN	Input	BOOL	High level enabled continuous reception	FALSE	FALSE
ConnectID	Input	BYTE	The connection ID (ConnectID) is the number of the connection used by this send operation. Use the ConnectID you selected for the TCP_CONNECT operation.	0	FALSE
MaxLen	Input	WORD	Size of the buffer in DataPtr in the MaxLen description (1~32 *1024) Byte	0	FALSE
DataAddr	Input	POINTER TO BYTE	DataAddr is a pointer to the storage location of the Rx data	0	FALSE
Done	Output	BOOL	When the Rx operation is completed without errors, the instruction sets the Done output. When the instruction sets the Done output, the RcvdLen output is valid	FALSE	FALSE
Busy	Output	BOOL	When the Rx operation is in progress, the instruction sets the Busy output	FALSE	FALSE
Error	Output	BOOL	When the receive operation is completed but an error occurs, the instruction sets the Error output	FALSE	FALSE
ErrorID	Output	WORD	If the instruction sets the Error output, the ErrorID output will display the error code. If the instruction sets the Busy or Done output, ErrorID is zero (no error). For details, see Error codes of Ethernet free protocol communication instructions.	0	FALSE
RcvdLen	Output	WORD	RcvdLen is the actual number of bytes received. RcvdLen is valid only when the instruction sets the Done or Error output. If the instruction sets the Done output, the instruction receives the entire message. If the instruction sets the Error output, the message exceeds the buffer size (MaxLen) and is truncated	0	FALSE

The TCP_RECV instruction has an IN (enable) input, which is triggered by a high level. After the TCP_RECV instruction is executed for the first time, the status bit shows that the instruction is busy. A busy status will be shown for subsequent calls to TCP_RECV until the CPU receives data over the specified connection.

After the CPU receives the message through the specified connection, the next time it executes the TCP_RECV instruction, it will perform the following tasks:

- Copy the message data to the program's data area (DataAddr).
- Set the RcvdLen output to the number of bytes received.

- Set the Done output, clear the Busy and Error outputs, and set the ErrorID output byte value to zero.

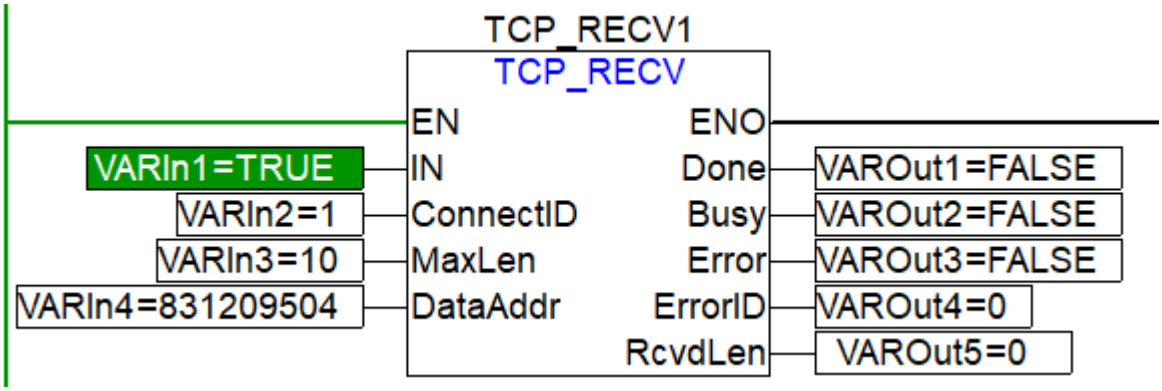
When the TCP protocol is used, the TCP_RECV instruction returns all bytes received by the CPU through the specified connection since the last time the program called the TCP_RECV instruction. The program must call the TCP_RECV instruction frequently enough to depict messages correctly because TCP acts as a "streaming" protocol. There is no message description (no start or end tags) in the TCP protocol. Therefore, the CPU does not know when a message starts or ends.

For example, let's assume that there is a TCP client that sends 10 pieces of 10-byte messages to the CPU one after another, and the program does not call the TCP_RECV instruction during this period. When the program calls the TCP_RECV instruction after the CPU has accepted all 10 messages, the TCP_RECV instruction returns this data in a 100-byte received message. It is the program's responsibility to call the TCP_RECV instruction frequently enough to receive each message exactly as it was sent.

7.1.10.3 Example

When IN is set, the data will be received according to the set received data length, and the received data will be placed in the address pointed to by DataAddr.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	TCP_RECV1			TCP_RECV		FALSE	Not Public	
0002	VARIn1			BOOL	TRUE	FALSE	Not Public	
0003	VARIn2			BYTE	1	FALSE	Not Public	
0004	VARIn3			WORD	10	FALSE	Not Public	
0005	VARIn4			POINTER TO BYTE	831209504	FALSE	Not Public	
0006	VAROut1			BOOL	FALSE	FALSE	Not Public	
0007	VAROut2			BOOL	FALSE	FALSE	Not Public	
0008	VAROut3			BOOL	FALSE	FALSE	Not Public	
0009	VAROut4			WORD	0	FALSE	Not Public	
0010	VAROut5			WORD	0	FALSE	Not Public	



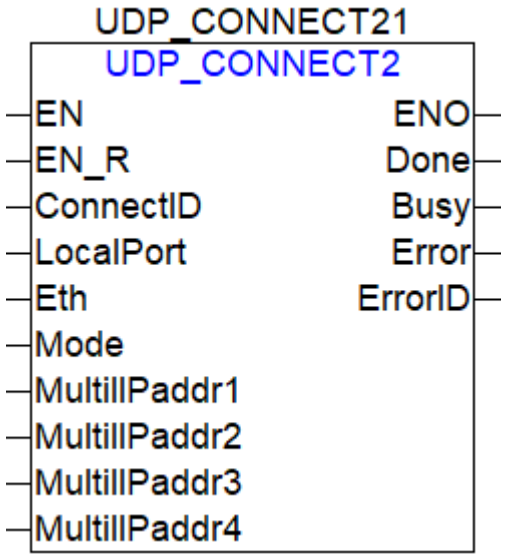

```
1  TCP_RECV1(
2  IN := VARIn1,
3  ConnectID := VARIn2,
4  MaxLen := VARIn3,
5  DataAddr := VARIn4,
6  Done=>VAROut1,
7  Busy=>VAROut2,
8  Error=>VAROut3,
9  ErrorID=>VAROut4,
10 RcvdLen=>VAROut5);
```

```
TCP_RECV1
VARIn1 = TRUE
VARIn2 = 1
VARIn3 = 10
VARIn4 = 831209504
VAROut1 = FALSE
VAROut2 = FALSE
VAROut3 = FALSE
VAROut4 = 0
VAROut5 = 0
```

7.1.11 UDP_CONNECT2(Creating UDP Protocol Communication (support communication by specific nices))

7.1.11.1 Function

When setting up a UDP communication connection and using a separate network card, it is necessary to specify the network card and configure the free protocol under the corresponding network card.



UDP_CONNECT2 Functional Block

- LX-CU430 do not support network card independent configuration.

7.1.11.2 Parameter

Parameter	Statement	Data Type	Parameter Description	Initial Value	Power Fail Safeguard
EN_R	Input	BOOL	Rising edge: The CPU starts connection operation.	FALSE	FALSE
ConnectID	Input	BYTE	Socket connection number 1~16	0	FALSE
LocalPort	Input	WORD	Port number on the local device. The range of local port numbers is 1 to 49151, but there are some restrictions. For details, see the parameter LocalPort in the Common parameters of the Ethernet free protocol library instruction.	0	FALSE
Eth	Input	BYTE	Network card ID number 0: P1 Interface Card, EtherNet1 1: P2 interface card, EtherNet2	0	FALSE
Mode	Input	BYTE	0: Unicast 1: Broadcast 2: Multicast 0 3: Multicast 1	0	FALSE
MultiIIPaddr1	Input	BYTE	Byte 1 of multicast IP address	0	FALSE
MultiIIPaddr2	Input	BYTE	Byte 2 of multicast IP address	0	FALSE
MultiIIPaddr3	Input	BYTE	Byte 3 of multicast IP address	0	FALSE
MultiIIPaddr4	Input	BYTE	Byte 4 of multicast IP address	0	FALSE
Done	Output	BOOL	If the connection is active and ready to send or receive	0	FALSE
Busy	Output	BOOL	If the connection process is still in progress	0	FALSE
Error	Output	BOOL	If the connection is not available. ErrorID displays one of the error codes to indicate the problem	0	FALSE
ErrorID	Output	WORD	If the instruction sets the Error output, the ErrorID output will display the error code. If the instruction sets the Busy or Done output, ErrorID is zero (no error). For details, see the Error codes of Ethernet free protocol communication instructions.	0	FALSE

- Parameter Supplement Explanation:

  The UDP_CONNECT2 instruction only requires the connection ID and local port number to create a connection. A single UDP connection can send messages to any number of other devices, as the IP address and remote port are provided with each UDP_SEND instruction. Multiple UDP connections are only needed if multiple local ports are required. The same local port number cannot be used for multiple UDP connections. All local port numbers must be unique.

  The connection operation is asynchronous and may take several scans to complete. There is no active connection established with the remote device, nor is there a wait for another device to connect to the CPU. While the connection operation is pending, the Busy output is set. When the connection operation is complete, the Done output is set. The Error output is set only if there is an issue with the input parameters or if a passive connection is not available. If the Error output is set, the ErrorID output byte will contain the error code.

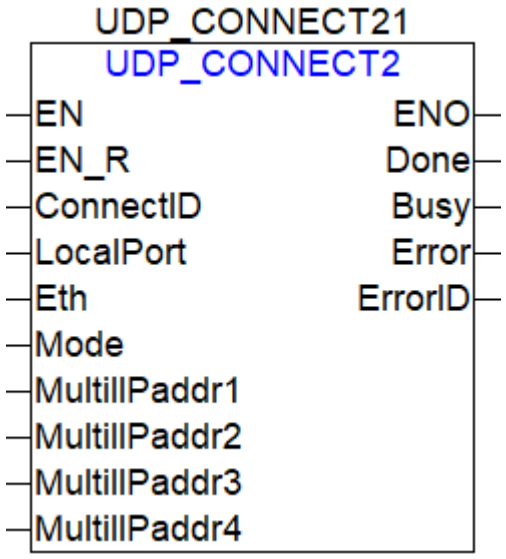
  When the instruction is busy, do not change the parameters of UDP_CONNECT2, as the CPU uses this to understand that it is a continuation of the call to initiate the connection process.

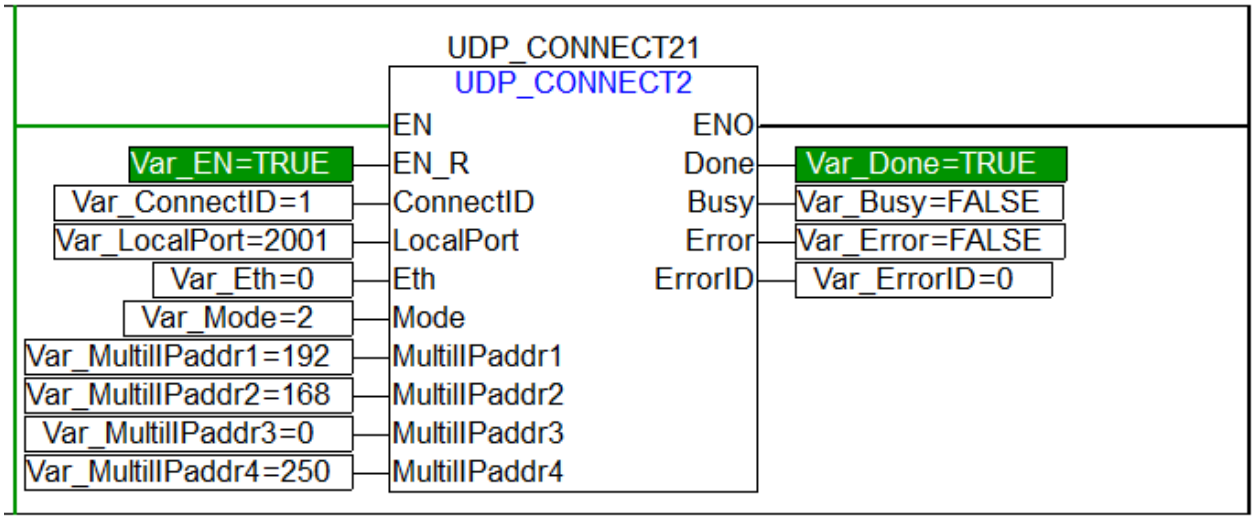
  The instruction can be called to determine the current status of the connection. Set the EN_R input to FALSE and provide a valid connection ID (ConnectID). The UDP_CONNECT2 instruction will return the following:

  - If the connection is active and ready for sending or receiving, the Done output is set. This only indicates that the connection is usable and does not mean that any remote device is present.
  - If the connection is still in progress, the Busy output is set.
  - If the connection is not available, the Error output is set. The ErrorID output byte will display one of the error codes, indicating the issue encountered.

7.1.11.3 Example

Creates a passive connection using the UDP protocol when EN _ R is set.





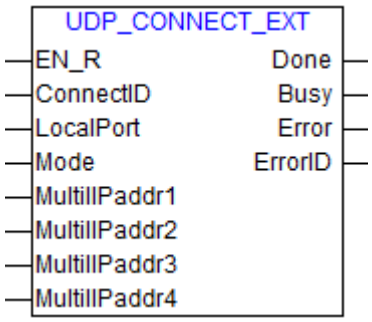

```
1   UDP_CONNECT21(
2   EN_R := Var_EN,
3   ConnectID := Var_ConnectID,
4   LocalPort := Var_LocalPort,
5   Eth := Var_Eth,
6   Mode := Var_Mode,
7   MultiIIPaddr1 := Var_MultiIIPaddr1,
8   MultiIIPaddr2 := Var_MultiIIPaddr2,
9   MultiIIPaddr3 := Var_MultiIIPaddr3,
10  MultiIIPaddr4 := Var_MultiIIPaddr4,
11  Done=>Var_Done,
12  Busy=>Var_Busy,
13  Error=>Var_Error,
14  ErrorID=>Var_ErrorID);
```

```
UDP_CONNECT21
Var_EN = TRUE
Var_ConnectID = 1
Var_LocalPort = 2001
Var_Eth = 0
Var_Mode = 2
Var_MultiIIPaddr1 = 192
Var_MultiIIPaddr2 = 168
Var_MultiIIPaddr3 = 0
Var_MultiIIPaddr4 = 250
Var_Done = TRUE
Var_Busy = FALSE
Var_Error = FALSE
Var_ErrorID = 0
```

7.1.12 UDP_CONNECT_EXT (Create UDP Protocol Communication (Support Multicast and broadcast))

7.1.12.1 Function

Create a passive connection using UDP protocol. Independent network card configuration is not supported. Unicast, multicast, and broadcast modes are supported.



UDP_CONNECT_EXT Functional Block

7.1.12.2 Parameter

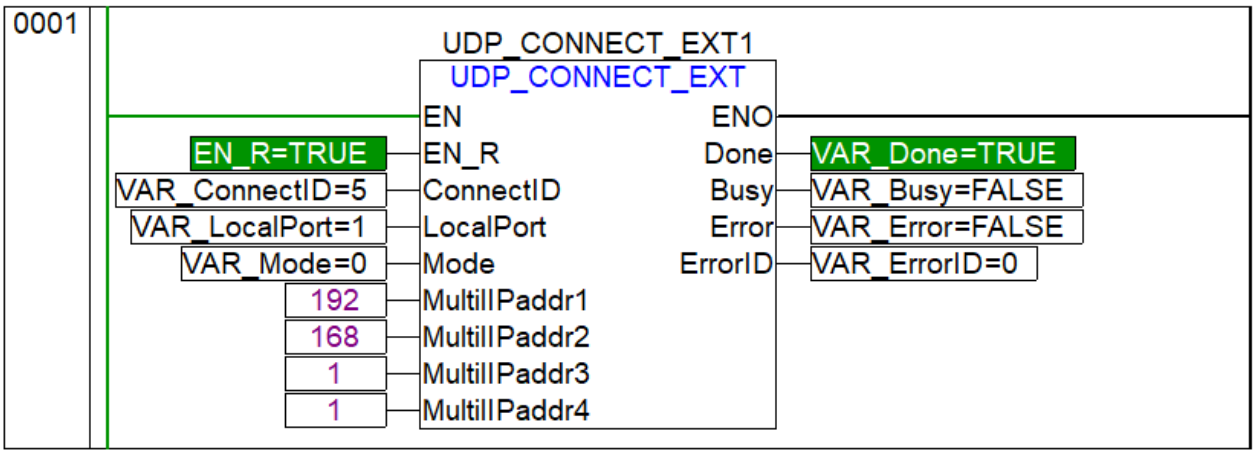
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN_R	Input	BOOL	Rising edge trigger connection	FALSE	FALSE
ConnectID	Input	BYTE	Socket connection number 1~16	0	FALSE
LocalPort	Input	WORD	Port number on the local device. The range of local port numbers is 1 to 49151, but there are some restrictions, see the parameter LocalPort in the Common parameters of the Ethernet free protocol library instruction.	0	FALSE
Mode	Input	BYTE	0: Unicast 1: Broadcast 2: Multicast 0 3: Multicast 1	0	FALSE
MultiIIPaddr1	Input	BYTE	Bit 1 of multicast IP address	0	FALSE
MultiIIPaddr2	Input	BYTE	Bit 2 of multicast IP address	0	FALSE
MultiIIPaddr3	Input	BYTE	Bit 3 of multicast IP address	0	FALSE
MultiIIPaddr4	Input	BYTE	Bit 4 of multicast IP address	0	FALSE
Done	Output	BOOL	When the connection operation is completed without errors, the instruction sets the Done output	FALSE	FALSE
Busy	Output	BOOL	When the connection operation is in progress, the instruction sets the Busy output	FALSE	FALSE
Error	Output	BOOL	If the connection is not available. ErrorID displays one of the error codes to indicate the problem	FALSE	FALSE
ErrorID	Output	WORD	If the instruction sets the Error output, the ErrorID output will	0	FALSE

			display the error code. If the instruction sets the Busy or Done output, ErrorID is zero (no error)		

7.1.12.3 Example

When EN_R is set, a passive connection is created using the UDP protocol.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	UDP_CONNECT_EXT1			UDP_CONNECT_EXT		FALSE	Not Public	
0002	EN_R			BOOL	TRUE	FALSE	Not Public	
0003	VAR_ConnectID			BYTE	5	FALSE	Not Public	
0004	VAR_Mode			BYTE	0	FALSE	Not Public	
0005	VAR_LocalPort			WORD	1	FALSE	Not Public	
0006	VAR_Done			BOOL	TRUE	FALSE	Not Public	
0007	VAR_Busy			BOOL	FALSE	FALSE	Not Public	
0008	VAR_Error			BOOL	FALSE	FALSE	Not Public	
0009	VAR_ErrorID			WORD	0	FALSE	Not Public	



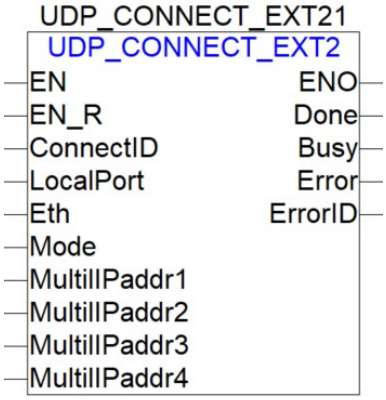

```
1  UDP_CONNECT_EXT1(
2  EN_R := EN_R,
3  ConnectID := VAR_ConnectID,
4  LocalPort := VAR_LocalPort,
5  Mode := VAR_Mode,
6  MultiIIPaddr1 := 192,
7  MultiIIPaddr2 := 168,
8  MultiIIPaddr3 := 1,
9  MultiIIPaddr4 := 1,
10 Done=>VAR_Done,
11 Busy=>VAR_Busy,
12 Error=>VAR_Error,
13 ErrorID=>VAR_ErrorID);
```

```
UDP_CONNECT_EXT1
EN_R = TRUE
VAR_ConnectID = 5
VAR_LocalPort = 1
VAR_Mode = 0

VAR_Done = TRUE
VAR_Busy = FALSE
VAR_Error = FALSE
VAR_ErrorID = 0
```

7.1.13 UDP_CONNECT_EXT2 (Creating UDP Protocol Communication (Support for specified network card multicast/broadcast))

7.1.13.1 Function

It is to create UDP communication. It supports unicast, multicast, and broadcast modes. When using an independent network card configuration, you need to specify the network card and configure the free protocol under the corresponding network card to enable communication.



UDP_CONNECT_EXT2 Functional Block

- LX-CU430 do not support network card independent configuration.

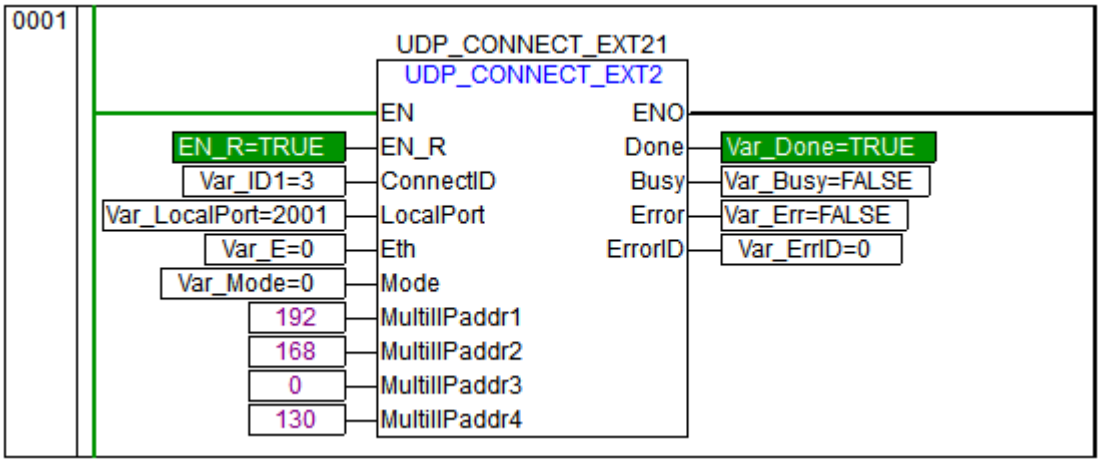
7.1.13.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN_R	Input	BOOL	Rising edge trigger connection	FALSE	FALSE
ConnectID	Input	BYTE	Socket connection number 1~16	0	FALSE
LocalPort	Input	WORD	Port number on the local device. The range of local port numbers is 1 to 49151, but there are some restrictions, see the parameter LocalPort in the Common parameters of the Ethernet free protocol library instruction.	0	FALSE
Eth	Input	BYTE	Network card ID number (default value: 0)	0	FALSE
Mode	Input	BYTE	0: Unicast 1: Broadcast 2: Multicast 0 3: Multicast 1	0	FALSE
MultiIIPaddr1	Input	BYTE	Bit 1 of multicast IP address	0	FALSE
MultiIIPaddr2	Input	BYTE	Bit 2 of multicast IP address	0	FALSE
MultiIIPaddr3	Input	BYTE	Bit 3 of multicast IP address	0	FALSE
MultiIIPaddr4	Input	BYTE	Bit 4 of multicast IP address	0	FALSE
Done	Output	BOOL	When the connection operation is completed without errors, the instruction sets the Done output	FALSE	FALSE
Busy	Output	BOOL	When the connection operation is in progress, the instruction sets the Busy output	FALSE	FALSE
Error	Output	BOOL	If the connection is not available. ErrorID displays one of the error codes to indicate the problem	FALSE	FALSE
ErrorID	Output	WORD	If the instruction sets the Error output, the ErrorID output will display the error code. If the instruction sets the Busy or Done output, ErrorID is zero (no error)	0	FALSE

7.1.13.3 Example

Creates a passive connection using the UDP protocol when EN _ R is set.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	UDP_CONNECT_EXT21			UDP_CONNECT_EXT2		FALSE	Not Public	
0002	EN_R			BOOL	TRUE	FALSE	Not Public	
0003	Var_ID1			BYTE	3	FALSE	Not Public	
0004	Var_LocalPort			WORD	2001	FALSE	Not Public	
0005	Var_E			BYTE	0	FALSE	Not Public	
0006	Var_Mode			BYTE	0	FALSE	Not Public	
0007	Var_Done			BOOL	TRUE	FALSE	Not Public	
0008	Var_Busy			BOOL	FALSE	FALSE	Not Public	
0009	Var_Err			BOOL	FALSE	FALSE	Not Public	
0010	Var_ErrID			WORD	0	FALSE	Not Public	




```
1  UDP_CONNECT_EXT21(
2    EN_R := EN_R,
3    ConnectID := Var_ID1,
4    LocalPort := Var_LocalPort,
5    Eth := Var_E,
6    Mode := Var_Mode,
7    MultiIIPaddr1 := 192,
8    MultiIIPaddr2 := 168,
9    MultiIIPaddr3 := 0,
10   MultiIIPaddr4 := 130,
11   Done=>Var_Done,
12   Busy=>Var_Busy,
13   Error=>Var_Err,
14   ErrorID=>Var_ErrID);
```

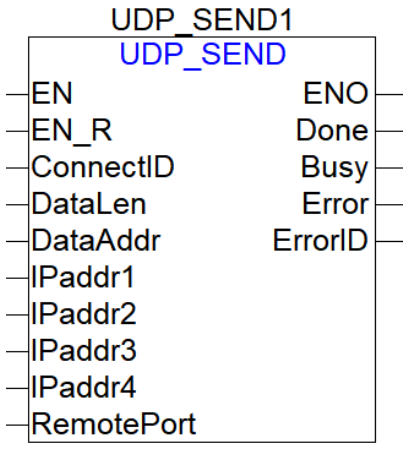
```
UDP_CONNECT_EXT21
EN_R = TRUE
Var_ID1 = 3
Var_LocalPort = 2001
Var_E = 0
Var_Mode = 0


Var_Done = TRUE
Var_Busy = FALSE
Var_Err = FALSE
Var_ErrID = 0
```

7.1.14 UDP_SEND (Send data via udp protocol)

7.1.14.1 Function

The UDP_SEND instruction transfers the requested number of bytes (DataLen) from the requested buffer

location (DataAddr) to the device specified by IP address (IPaddr1–IPaddr4) and port (RemPort). This instruction is only used for the UDP protocol and connections created via UDP_CONNECT.



UDP_SEND Functional Block

7.1.14.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
EN_R	Input	BOOL	Rising edge: The CPU starts sending operation.	FALSE	FALSE
ConnectID	Input	BYTE	The connection ID (ConnectID) is the number of the connection used by this send operation (defined by UDP_CONNECT during the connection process)	0	FALSE
DataLen	Input	WORD	Sent data length, (1~32*1024) Byte	0	FALSE
DataAddr	Input	POINTER TO BYTE	DataAddr is a pointer to the storage location of the sent data	0	FALSE
IPaddr1	Input	BYTE	IP address digit 1	0	FALSE
IPaddr2	Input	BYTE	IP address digit 2	0	FALSE
IPaddr3	Input	BYTE	IP address digit 3	0	FALSE
IPaddr4	Input	BYTE	IP address digit 4	0	FALSE
RemotePort	Input	WORD	Port number on the remote device. The range of remote port numbers is 1 to 49151. For passive connections, zero is used	0	FALSE
Done	Output	BOOL	When the send operation is completed without errors, the instruction sets the Done output	FALSE	FALSE
Busy	Output	BOOL	When the send operation is in progress, the instruction sets the Busy output	FALSE	FALSE
Error	Output	BOOL	When the sending operation is completed but an error occurs, the instruction sets the Error output	FALSE	FALSE
ErrorID	Output	WORD	If the instruction sets the Error output, the ErrorID output will display the error code. If the instruction sets the Busy or Done output, ErrorID is zero (no error). For details, see the Error codes of Ethernet free protocol communication instructions.	0	FALSE

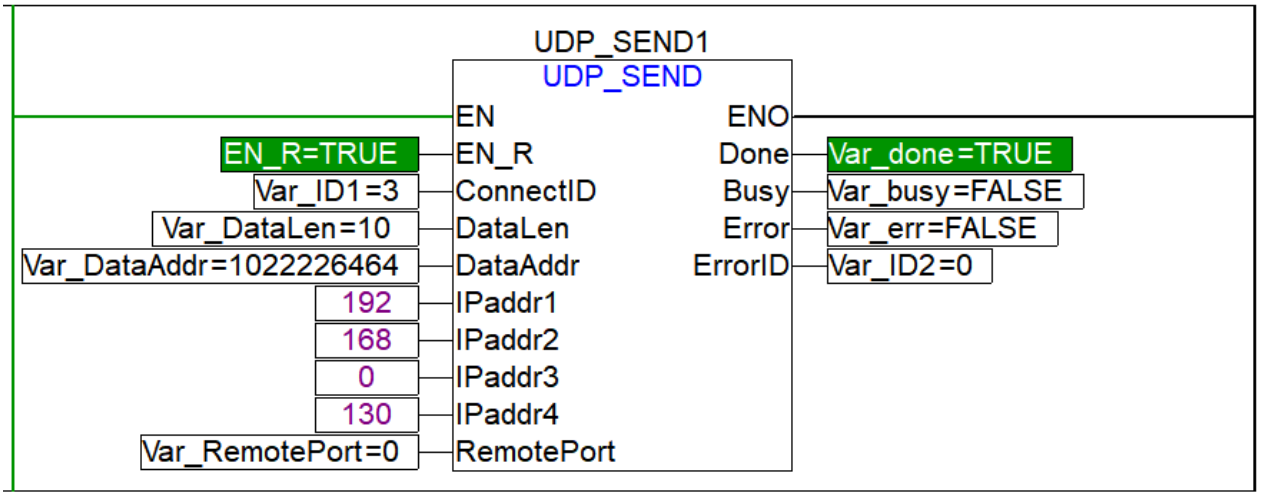
When the following conditions are met, the UDP_SEND instruction initiates the operation of sending a specified number of bytes:

- The instruction is called by setting the EN_R input to TRUE.
- The connection is not currently being used for another send operation.
- The EN_R input is level-triggered. It is recommended to use a rising-edge trigger for the EN_R input so that the instruction does not initiate unintended send operations. When UDP_SEND is busy, the program ignores the EN_R input. The Done, Busy, and Error outputs, as well as the ErrorID output byte, show the status of each call to UDP_SEND.
- Once the send operation is complete, the instruction shows the Done or Error status for one call to UDP_SEND. Afterward, UDP_SEND responds with an error code, indicating that there is no pending operation (if called with EN_R set to FALSE). If EN_R is set to TRUE, the program starts another send operation. The relationship between the input and output parameters is described below.
- EN_R is set to TRUE to start the message send operation. Busy is set to TRUE.
- Message send operation is complete. Done is set, and Busy is cleared.
- EN_R is FALSE, but no message send operation is in progress.
- EN_R is set to TRUE again, so another message send operation begins. Busy is set to TRUE.
- Message send operation is complete. Done is set, and Busy is cleared within one scan cycle.

7.1.14.3 Example

When EN_R is set, the data at the address pointed to by DataAddr is sent according to the set sending length

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	UDP_SEND1			UDP_SEND		FALSE	Not Public	
0002	EN_R			BOOL	TRUE	FALSE	Not Public	
0003	Var_ID1			BYTE	3	FALSE	Not Public	
0004	Var_done			BOOL	TRUE	FALSE	Not Public	
0005	Var_busy			BOOL	FALSE	FALSE	Not Public	
0006	Var_err			BOOL	FALSE	FALSE	Not Public	
0007	Var_ID2			WORD	0	FALSE	Not Public	
0008	Var_DataLen			WORD	10	FALSE	Not Public	
0009	Var_DataAddr			POINTER TO BYTE	1022226464	FALSE	Not Public	
0010	Var_RemotePort			WORD	0	FALSE	Not Public	



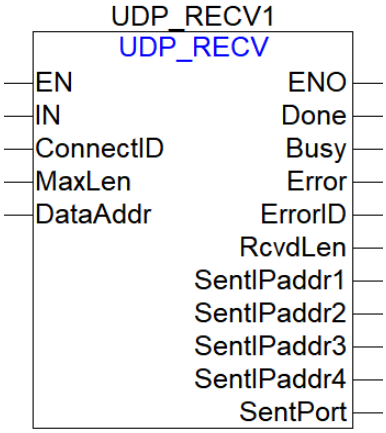

```
1  UDP_SEND1(
2  EN_R := EN_R,
3  ConnectID := Var_ID1,
4  DataLen := Var_DataLen,
5  DataAddr := Var_DataAddr,
6  IPaddr1 := 192,
7  IPaddr2 := 168,
8  IPaddr3 := 0,
9  IPaddr4 := 130,
10 RemotePort := Var_RemotePort,
11 Done=>Var_done,
12 Busy=>Var_busy,
13 Error=>Var_err,
14 ErrorID=>Var_ID2);
```

```
UDP_SEND1
EN_R = TRUE
Var_ID1 = 3
Var_DataLen = 10
Var_DataAddr = 1022226464

Var_RemotePort = 0
Var_done = TRUE
Var_busy = FALSE
Var_err = FALSE
Var_ID2 = 0
```

7.1.15 UDP_RECV (Receive data via udp protocol)

7.1.15.1 Function

The UDP_RECV instruction retrieves data over an existing connection. This instruction is only used for the UDP protocol and connections created via UDP_CONNECT.



UDP_RECV Functional Block

7.1.15.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
IN	Input	BOOL	High level enable	FALSE	FALSE
ConnectID	Input	BYTE	The CPU uses the connection ID (ConnectID) for this receive operation (defined during the UDP_CONNECT connection process)	0	FALSE
MaxLen	Input	WORD	MaxLen is the maximum number of bytes to be received (for example, the size of the buffer in DataPtr (1~32*1024)Byte)	0	FALSE
DataAddr	Input	POINTER TO BYTE	DataAddr is a pointer to the storage location of the Rx data	0	FALSE
Done	Output	BOOL	When the Rx operation is completed without errors, the instruction sets the Done output	FALSE	FALSE
Busy	Output	BOOL	When the Rx operation is in progress, the instruction sets the Busy output	FALSE	FALSE
Error	Output	BOOL	When the receive operation is completed but an error occurs, the instruction sets the Error output	FALSE	FALSE
ErrorID	Output	WORD	If the instruction sets the Error output, the ErrorID output will display the error code. If the instruction sets the Busy or Done output, ErrorID is zero (no error). For details, see the Error codes of Ethernet free protocol communication instructions	0	FALSE
RcvdLen	Output	WORD	RcvdLen is the actual number of bytes received. RcvdLen is valid only when the instruction sets the Done or Error output. If the instruction sets the Done output, the instruction receives the entire message. If the instruction sets the Error output, the message exceeds the buffer size (MaxLen) and is truncated	0	FALSE
SentIPaddr1	Output	BYTE	Four octets of the IP address of the remote device sending messages. SentIPaddr1 is the most significant byte of the IP address, and SentIPaddr4 is the least significant byte of the	0	FALSE
SentIPaddr2	Output	BYTE		0	FALSE
SentIPaddr3	Output	BYTE		0	FALSE

SentIPaddr4	Output	BYTE	IP address.	0	FALSE
SentPort	Output	WORD	Port number of the remote device sending messages	0	FALSE

The UDP_RECV instruction has only an EN (Enable) input. The UDP_RECV instruction is active-high. After the first execution of the UDP_RECV instruction, the status outputs indicate that the instruction is busy. Subsequent calls to UDP_RECV will show a busy status until the CPU receives data through the specified connection.

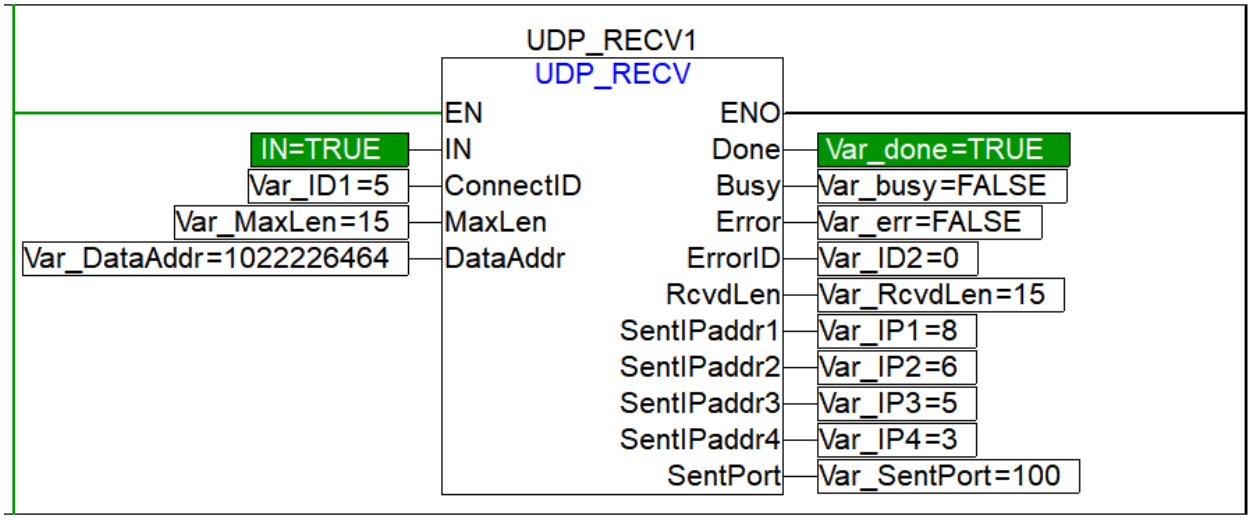
Once the CPU receives a message through the specified connection, the next execution of the UDP_RECV instruction will perform the following tasks:

- Copy the message data to the program's data area (DataAddr).•Set the returned RcvdLen to the number of bytes received.
- Set the SentIPaddr1 to SentIPaddr4 addresses to the address of the remote device that sent the message.
- Set the remote port number (SentPort) to the port of the remote device.
- Set the Done output, clear the Busy and Error outputs, and set the ErrorID output byte value to zero.

7.1.15.3 Example

When IN is on, the data will be received according to the set Rx data length, and the received data will be placed in the address pointed to by DataAddr.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	UDP_RECV1			UDP_RECV		FALSE	Not Public	
0002	IN			BOOL	TRUE	FALSE	Not Public	
0003	Var_ID1			BYTE	5	FALSE	Not Public	
0004	Var_MaxLen			WORD	15	FALSE	Not Public	
0005	Var_DataAddr			POINTER TO BYTE	1022226464	FALSE	Not Public	
0006	Var_done			BOOL	TRUE	FALSE	Not Public	
0007	Var_busy			BOOL	FALSE	FALSE	Not Public	
0008	Var_err			BOOL	FALSE	FALSE	Not Public	
0009	Var_ID2			WORD	0	FALSE	Not Public	
0010	Var_RcvdLen			WORD	15	FALSE	Not Public	
0011	Var_IP1			BYTE	8	FALSE	Not Public	
0012	Var_IP2			BYTE	6	FALSE	Not Public	
0013	Var_IP3			BYTE	5	FALSE	Not Public	
0014	Var_IP4			BYTE	3	FALSE	Not Public	
0015	Var_SentPort			WORD	100	FALSE	Not Public	



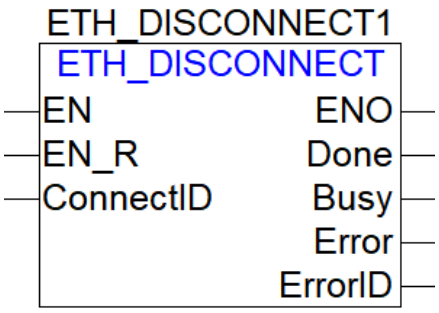
```
1  UDP_RECV1(
2  IN := IN,
3  ConnectID := Var_ID1,
4  MaxLen := Var_MaxLen,
5  DataAddr := Var_DataAddr,
6  Done=>Var_done,
7  Busy=>Var_busy,
8  Error=>Var_err,
9  ErrorID=>Var_ID2,
10 RcvdLen=>Var_RcvdLen,
11 SentIPaddr1=>Var_IP1,
12 SentIPaddr2=>Var_IP2,
13 SentIPaddr3=>Var_IP3,
14 SentIPaddr4=>Var_IP4,
15 SentPort=>Var_SentPort);
```

```
UDP_RECV1
IN = TRUE
Var_ID1 = 5
Var_MaxLen = 15
Var_DataAddr = 1022226464
Var_done = TRUE
Var_busy = FALSE
Var_err = FALSE
Var_ID2 = 0
Var_RcvdLen = 15
Var_IP1 = 8
Var_IP2 = 6
Var_IP3 = 5
Var_IP4 = 3
Var_SentPort = 100
```

7.1.16 ETH_DISCONNECT (Closing Ethernet Communication Connection)

7.1.16.1 Function

The ETH_DISCONNECT instruction is used to terminate existing communication connections.



ETH_DISCONNECT Functional Block

7.1.16.2 Parameter

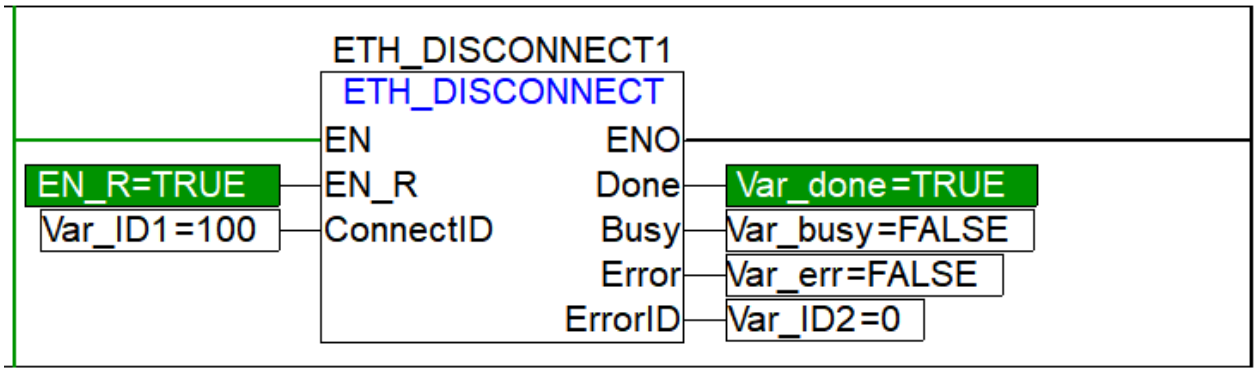
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		

EN_R	Input/Output	BOOL	Rising edge trigger closing connection	FALSE	FALSE
ConnectID	Input	BYTE	The CPU uses the connection ID (ConnectID) to identify the connection to be terminated (defined during the connection process)	0	FALSE
Done	Output	BOOL	When the disconnection operation is completed without errors, the instruction sets the Done output	FALSE	FALSE
Busy	Output	BOOL	When the disconnection operation is in progress, the instruction sets the Busy output	FALSE	FALSE
Error	Output	BOOL	When the disconnection operation is completed but an error occurs, the instruction sets the Error output	FALSE	FALSE
ErrorID	Output	WORD	If the instruction sets the Error output, the ErrorID output will display the error code. If the instruction sets the Busy or Done output, ErrorID is zero (no error). For details, see the Error codes of Ethernet free protocol communication instructions	0	FALSE

7.1.16.3 Example

When EN_R is set, the Ethernet communication link is closed.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	ETH_DISCONNECT1			ETH_DISCONNECT		FALSE	Not Public	
0002	EN_R			BOOL	TRUE	FALSE	Not Public	
0003	Var_ID1			BYTE	10	FALSE	Not Public	
0004	Var_done			BOOL	TRUE	FALSE	Not Public	
0005	Var_busy			BOOL	FALSE	FALSE	Not Public	
0006	Var_err			BOOL	FALSE	FALSE	Not Public	
0007	Var_ID2			WORD	0	FALSE	Not Public	




```
1 ETH_DISCONNECT1(
2 EN_R := EN_R,
3 ConnectID := Var_ID1,
4 Done=>Var_done,
5 Busy=>Var_busy,
6 Error=>Var_err,
7 ErrorID=>Var_ID2);
```

```
ETH_DISCONNECT1
EN_R = TRUE
Var_ID1 = 100
Var_done = TRUE
Var_busy = FALSE
Var_err = FALSE
Var_ID2 = 0
```

7.1.16.4 Common parameters of Ethernet free protocol library instructions

Common Parameters	Description
EN_R	This input is used to initiate an operation, and it is triggered by level. The EN_R input shall be connected to the library instruction via a rising edge instruction so that the operation is initiated only once. When the instruction is Busy, the program will ignore the EN_R input.
Active	The Active input is used to specify whether the connection instruction creates an active client connection (Active=TRUE) or a passive server connection (Active=FALSE). In an active connection, the local CPU initiates communication with the remote device. In a passive connection, the local CPU waits for the remote device to start communication.
Done	When the operation is completed without errors, the OUC instruction sets the Done output. If the instruction sets the Done output, the Busy, Error, and ErrorID outputs are zero. Other outputs (e.g. number of bytes received) are valid only when the Done output is set.
Busy	The Busy output indicates that an operation is in progress. When an operation is initiated by setting EN_R to TRUE, the OUC instruction sets the Busy output. For all subsequent calls to the instruction, the Busy output remains set until the operation is completed.
Error	The Error output indicates that the operation is completed with errors. If the OUC instruction sets the Error output, the Done and Busy outputs will be set to FALSE. If the instruction sets the Error

	output, the ErrorID output will display the error cause. If the Error output is set, all other outputs are invalid.
ConnectID	The ConnectID number is the identifier of the connection. When a connection is created via TCP_CONNECT, ConnectID is created which can be chosen to be any value in the range 1 to 5. Each connection must have a unique ConnectID. The program uses ConnectID to specify the connection required for subsequent send, receive, and disconnect operations.
IPaddr1~IPaddr4	These are four octets of the remote device's IP address. IPaddr1 is the most significant byte of the IP address, and IPaddr4 is the least significant byte of the IP address. For example, for the IP address 192.168.2.15, the following values are set: IPaddr1=192 IPaddr2=168 IPaddr3=2 IPaddr4=15 The IP address cannot be the following values: 0.0.0.0 (for active connections) Any broadcast IP address (for example, 255.255.255.255) Any multicast address IP address of local CPU The IP address 0.0.0.0 can be used for passive connections. By selecting the IP address 0.0.0.0, the CPU accepts connections from any remote IP address. If a non-zero IP address is selected for a passive connection, the CPU will only accept connections from the specified address.
RemotePort	Port number on the remote device. Port numbers can be used for TCP and UDP protocols, thus routing messages within the device. The rules for remote port numbers are as follows: Valid port number range: 1 to 49151 The recommended port number range is 2000 to 49151. For passive connections, the CPU ignores the remote port number (it can be set to zero).
LocalPort	It is the port number on the local CPU. Port numbers can be used for TCP and UDP protocols, thus routing messages within the device. For all passive connections, the local port number must be unique. The rules for local port numbers are as follows: Valid port number range: 1 to 49151 Port numbers 20, 21, 25, 80, 102, 135, 161, 162, 443, 1200, and 34962 to 34964 cannot be used. These ports are used for specific purposes. The recommended port number range is 2000 to 49151. For passive connections, the local port number must be unique (not duplicated).

7.1.16.5 Error Codes of Ethernet Free Protocol Communication Instructions

Error code	Error Message Description
0	No error
1	Wrong input ConnectID
2	Invalid input IP address
3	Invalid input remote port number
4	The input local port number is invalid
5	The input parameters changed while trying to connect using the TCP_CONNECT instruction
6	The input information for this active connection is duplicated with other connection information while trying to use the TCP_CONNECT instruction to create an active connection
7	Trying to use the TCP_CONNECT instruction to create a passive connection, but the input information for this connection is duplicated with other connection information
8	Connection closed
9	Non-TCP type connection
10	Failed to establish connection
11	The connection object refused the connection
12	Received a connection request from a non-configured IP address

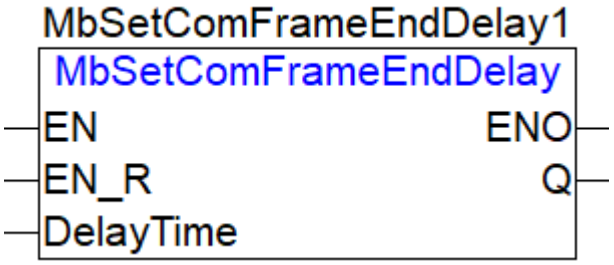
13	Ethernet protocol link is not configured with the Ethernet free protocol function
14	Incorrect data length or address information while sending or receiving data using the TCP or UDP free protocol
15	TCP_SEND data sending failed
16	TCP_RECV data receiving failed
17	Non-UDP type connection
18	ETH_DISCONNECT instruction is being used to close the closed link
19	UDP_CONNECT connection input information is duplicated with other connection information
20	Trying to connect using the UDP_CONNECT instruction, but the input parameters changed
21	An error occurred during TCP_CONNECT setting socket options
22	TCP_CONNECT port binding process failed
23	TCP_CONNECT link listening process failed
24	An error occurred during UDP_CONNECT setting socket options
25	UDP_CONNECT port binding process failed
26	Multiple tasks are performing the same connection operation at the same time

7.1.17 MbSetComFrameEndDelay (Set Serial Port Modbus Frame End Delay Time)

7.1.17.1 Function

The MbSetComFrameEndDelay function block is used to set the Modbus RTU frame end delay time for the LX master control COM port during Modbus RTU communication. When not configured, the master control COM port strictly adheres to the standard 3.5-character interval time during Modbus RTU communication. That is, when the interval between received message characters exceeds 3.5 characters, the message is considered ended. When communicating with third-party non-standard devices, if the other party does not strictly comply with the 3.5-character timing and the interval between characters in a message frame exceeds 3.5 characters, it may be treated as two separate message frames, leading to errors.

To ensure compatibility with third-party devices, an additional delay time is introduced on top of the standard 3.5-character interval. This helps distinguish between different message frames and ensures that a complete Modbus RTU data frame can be received and processed correctly.



MbSetComFrameEndDelay Functional Block

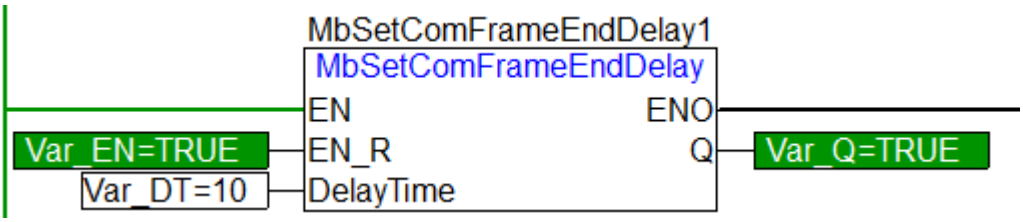
- The function block is applicable to the LX-CU500, LX-CU510, and LX-CU430 master controllers. It only needs to be configured once after powering on. The LX-CU501 and LX-CU511 do not have a COM port, so the settings are invalid for these models.
- The function block is specifically used when the master controller's COM port is operating as a Modbus RTU master or slave for Modbus communication. It does not affect the COM port when it is used for free communication.

7.1.17.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value
EN_R	Input	BOOL	Enabled on rising edge	FALSE
DelayTime	Input	BYTE	Delay time to set (ms)	0
Q	Output	BOOL	Set success flag with TRUE success	FALSE

7.1.17.3 Example

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	MbSetComFrameEndDelay1			MBSETCOMFRAMEENDDELAY		FALSE	Not Public	
0002	Var_EN			BOOL	TRUE	FALSE	Not Public	
0003	Var_DT			BYTE	10	FALSE	Not Public	
0004	Var_Q			BOOL	TRUE	FALSE	Not Public	




```
1 MbSetComFrameEndDelay1(
2 EN_R := Var_EN,
3 DelayTime := Var_DT,
4 Q=>Var_Q);
```

```
MbSetComFrameEndDelay1
Var_EN = TRUE
Var_DT = 10
Var_Q = TRUE
```

7.1.18 MbMappingConfig (Set/Get Modbus Slave Mapping Offset)

7.1.18.1 Version

- AutoThink software version: V3.2.3 SP3 and above.
- Controller firmware version is as follows:

Controller Model	Firmware Version
LX-CU430	LX-CU430_V1.2.0 and above.

LX-CU433	LX-CU433_V1.0.0 and above.
LX-CU500	LX-CU500_V1.4.0 and above.
LX-CU501	LX-CU501_V1.2.0 and above.
LX-CU510	LX-CU510_V1.1.0 and above.
LX-CU511	LX-CU511_V1.1.0 and above.

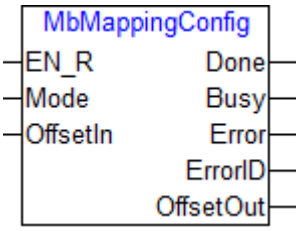
7.1.18.2 Function

When the controller acts as a slave station for Modbus communication, the Modbus address of the controller needs to be filled in on the master station side for data reading and writing. Therefore, it is necessary to know the mapping relationship between the Modbus address and the memory address of the controller. By default, the Modbus address mapping offset of the M area in the controller is 3000.

When OffsetIn is 0, function codes 02 and 04 can only access the I area; function codes 01, 05, and 15 can only access the Q area; function codes 03, 06, 16, and 23 can only access the M area, and the M area only supports word access, not bit access.

When OffsetIn is non-zero, for function codes 01, 05, 15, 03, 06, 16, and 23, if the address offset is greater than this parameter value, the M area is accessed; otherwise, the Q area is accessed. For function codes 02 and 04, if the address offset is greater than this parameter value, the M area is accessed; otherwise, the I area is accessed.

Through this function block, the address mapping relationship of Modbus TCP and Modbus RTU in the controller can be set or obtained. After modification, Modbus TCP and Modbus RTU take effect simultaneously; separate settings are not supported.



MbMappingConfig function block

7.1.18.3 Precautions

In setting mode, after the function block is configured successfully, a power cycle or a key switch restart of the controller is required for the changes to take effect.

The set mapping offset address returns to the default value only when the controller is restored to factory settings.

7.1.18.4 Parameters

Parameter	Declaration	Data Type	Description	Default Value
EN_R	INPUT	BOOL	Function block enable, rising edge triggers.	FALSE

Mode	INPUT	BOOL	Operating mode: FALSE: Get. In this mode, input parameter OffsetIn can be empty. TRUE: Set. In this mode, output parameter OffsetOut can be ignored.	FALSE
OffsetIn	INPUT	WORD	In setting mode, the Modbus mapping offset address to be modified shall be in the range of 0 to 65535, default 3000, and must be a multiple of 8.	0
Done	OUTPUT	BOOL	Set or get successful.	FALSE
Busy	OUTPUT	BOOL	Set or get in progress.	FALSE
Error	OUTPUT	BOOL	Set or get error.	FALSE
ErrorID	OUTPUT	WORD	Error codes: 0: No error 1: Offset error — must be a multiple of 8 2: Failed to create write request 3: Write aborted abnormally 4: Write returned failure	0
OffsetOut	OUTPUT	WORD	When the operating mode is Get, this is the obtained Modbus mapping offset address.	0

7.1.18.5 Example

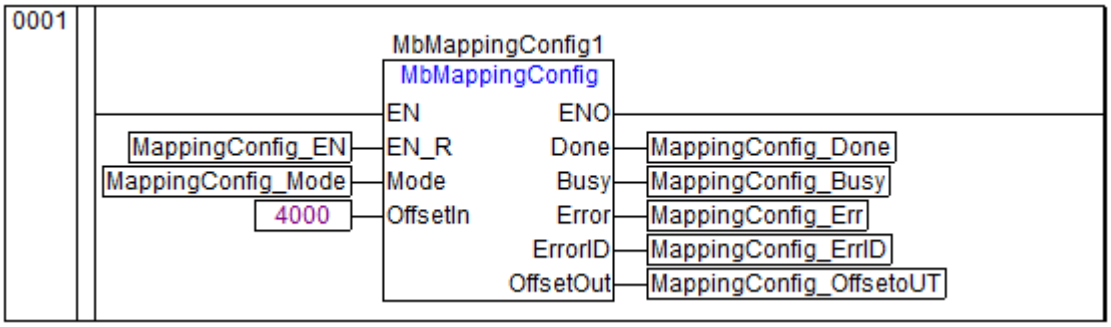
This example changes the Modbus mapping offset address of the controller from the default value of 3000 to 4000. The detailed operations are as follows.

1. Variable Definition

Main(PRG).ld

No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	MbMappingConfig1			MBMAPPINGCONFIG		FALSE	Not Public	
0002	MappingConfig_EN			BOOL	FALSE	FALSE	Not Public	
0003	MappingConfig_Mode			BOOL	FALSE	FALSE	Not Public	
0004	MappingConfig_Done			BOOL	FALSE	FALSE	Not Public	
0005	MappingConfig_Busy			BOOL	FALSE	FALSE	Not Public	
0006	MappingConfig_Err			BOOL	FALSE	FALSE	Not Public	
0007	MappingConfig_ErrID			WORD	0	FALSE	Not Public	
0008	MappingConfig_OffsetOut			WORD	0	FALSE	Not Public	

2. LD Configuration Example



3. ST Configuration Example

```
MbMappingConfig1(
   EN_R := MappingConfig_EN,
   Mode := MappingConfig_Mode,
```

OffsetIn := 4000,

Done=>MappingConfig_Done,

Busy=>MappingConfig_Busy,

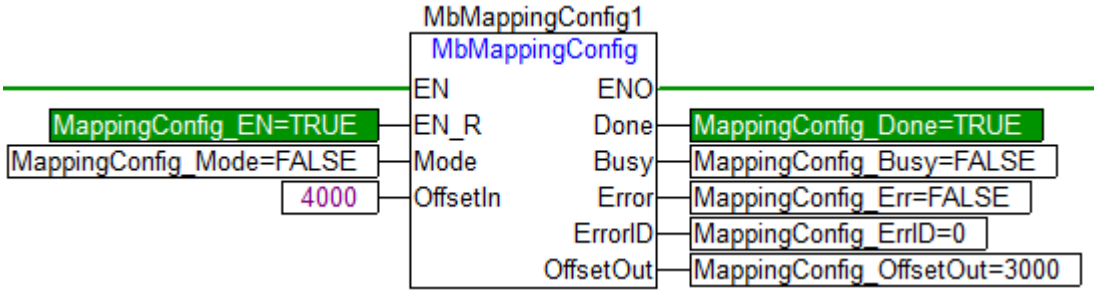
Error=>MappingConfig_Err,

ErrorID=>MappingConfig_ErrID,

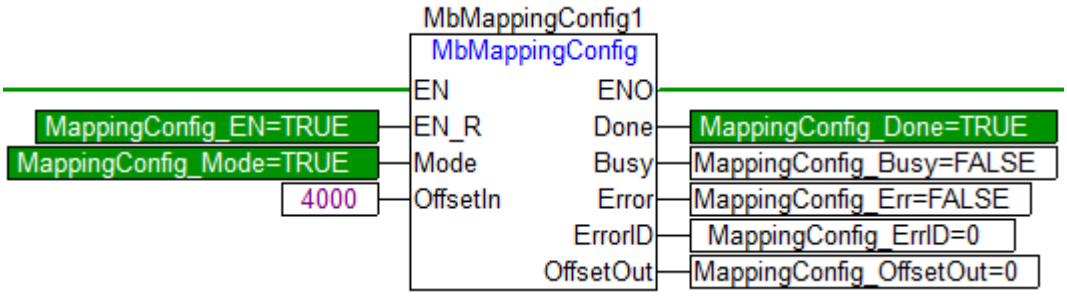
OffsetOut=>MappingConfig_OffsetOut);

4. Taking the LD program as an example, get and set the Modbus slave mapping offset.

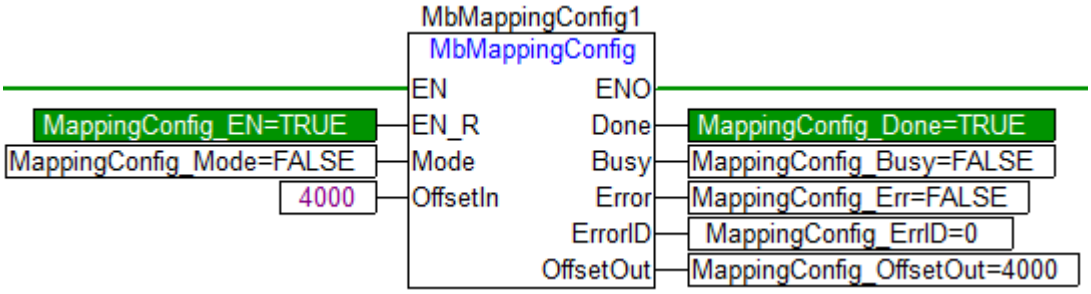
(1) After the controller is successfully connected and the project is online, set Mode to FALSE (Get mode), enable the MbMappingConfig function block, and the Done output becomes TRUE (get successful). At this time, the Modbus slave mapping offset OffsetOut is 3000.



(2) Set Mode to TRUE (Set mode), change the Modbus mapping offset address OffsetIn to 4000, enable the MbMappingConfig function block again, and the Done output becomes TRUE (set successful).



(3) Power cycle the controller. Set the Mode of the MbMappingConfig function block to FALSE (Get mode) and enable it. Obtain the modified Modbus mapping offset address OffsetIn value, which is 4000.

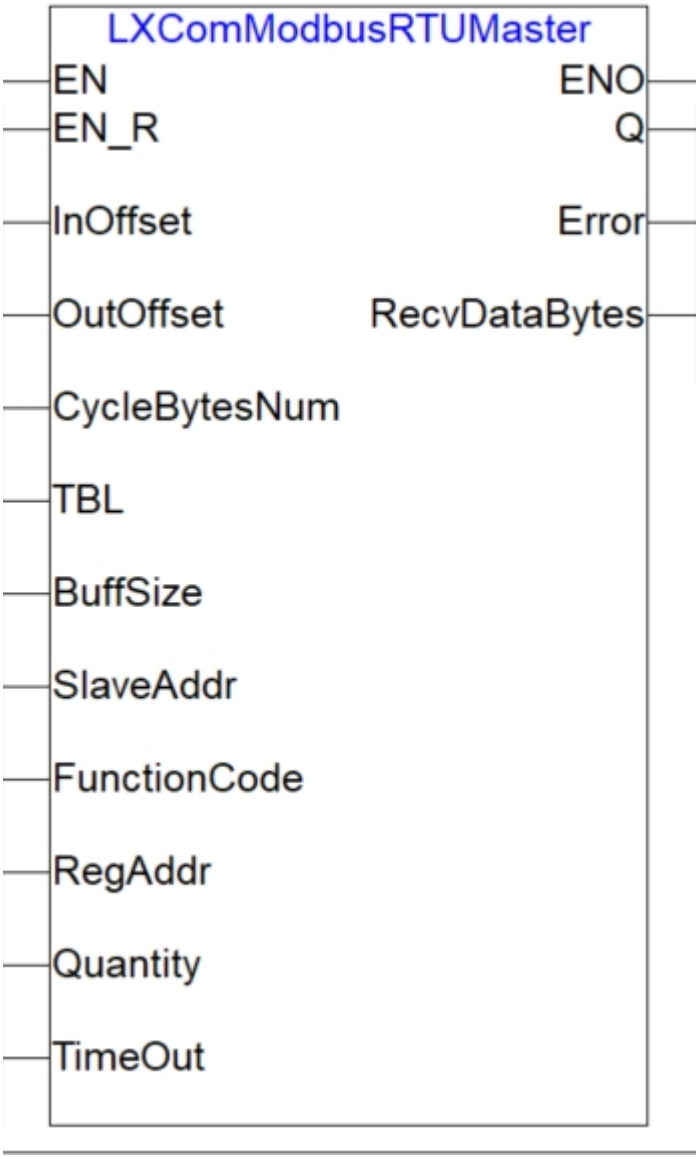


7.1.19 LXComModbusRTUMaster (MODBUS Master)

7.1.19.1 Function

This functional block realizes the MODBUS RTU master station function together with the LX-CM series serial port communication module.

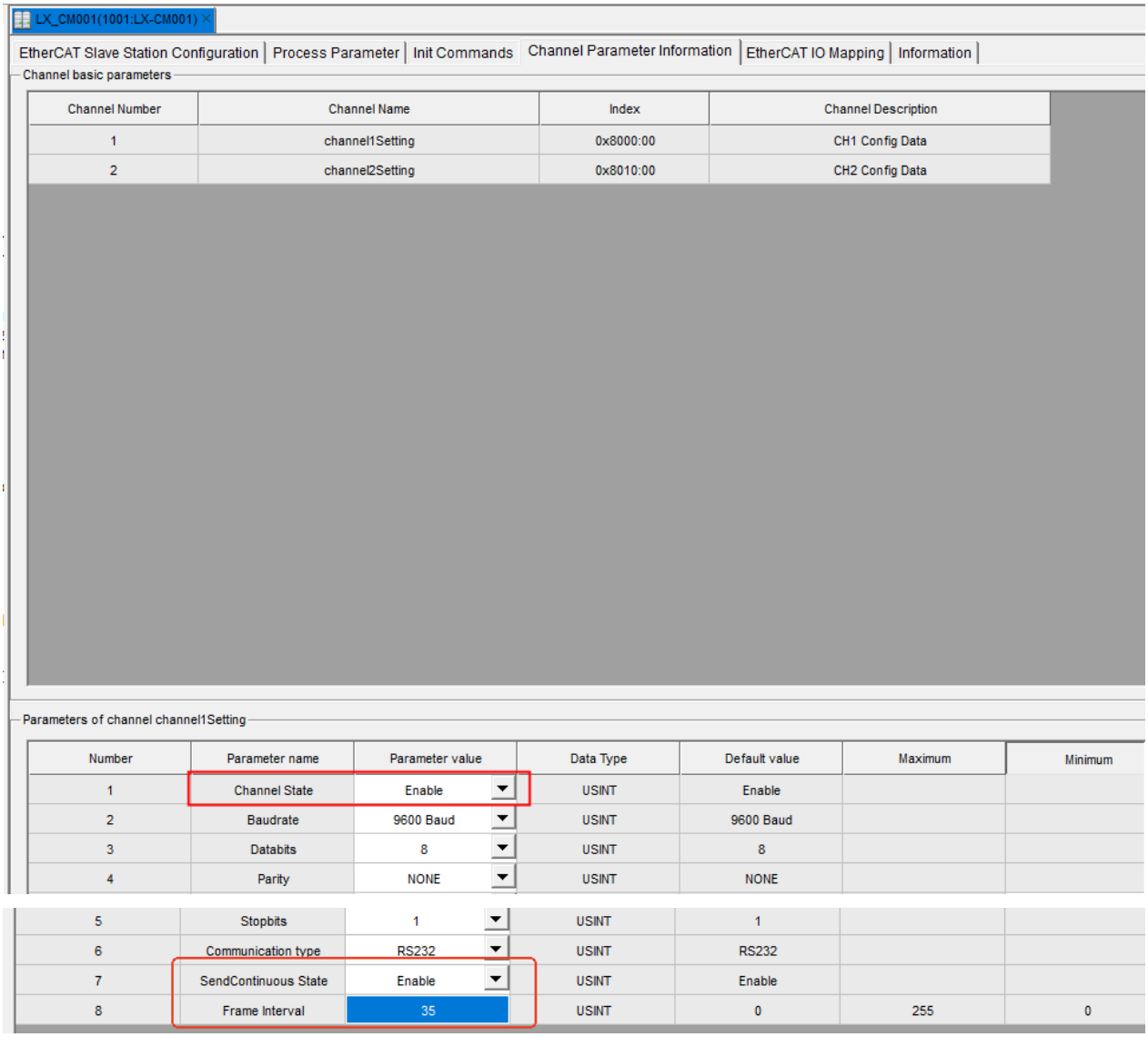
This function can also be implemented via LXComMBMaster (MODBUS Master with packet capture).



LXComModbusRTUMaster Functional Block

Channel parameter configuration:

- In addition to the channel parameter information, attention shall also be paid to the process parameters in the hardware configuration and the channel address in the EtherCAT IO mapping. For details, see the description of functional block parameters.



7.1.19.2 Parameter

Parameter	Statement	Data Type	Function Description	Description	Initial Value	Power Fail Safeguard
EN_R	Input	BOOL	Enabled, valid on rising edge	Each time an instruction action is executed, EN_R shall be set from FALSE to TRUE. When it is checked that Q changes from FALSE to TRUE, it means that the execution of this action is completed, and EN_R can be set to FALSE.	FALSE	FALSE
InOffset	Input	DWORD	Area I offset address of periodic input data	INOFFSET depends on the channel address in the EtherCAT IO mapping page in the hardware configuration. The channel address corresponding to the first status is the INOFFSET of the first channel, and the channel address corresponding to the second status is the INOFFSET of the second channel.	0	FALSE

OutOffset	Input	DWORD	Area Q offset address of periodic output data	OUTOFFSET depends on the channel address in the EtherCAT IO mapping page in the hardware configuration. The channel address corresponding to the first ctrl is the OUTOFFSET of the first channel, and the channel address corresponding to the second ctrl is the OUTOFFSET of the second channel.	0	FALSE
CycleBytesNum	Input	LXCYCLECOMBYTESNUM_E	Data size transmitted periodically	For CYCLEBYTESNUM, the enumeration variable of type LXCYCLECOMBYTESNUM_E shall be filled in, and the value depends on the process parameters in the hardware configuration. On the left side of the process parameter page, 16#1A00/16#1A01/16#1A02 respectively correspond to the data of 20/40/80 bytes input per cycle in channel 1, and 16#1A03/16#1A04/16#1A05 respectively correspond to the data of 20/40/80 bytes input per cycle in channel 2. On the right side, 16#1600/16#1601/16#1602 respectively correspond to the data of 20/40/80 bytes output per cycle in channel 1, and 16#1603/16#1604/16#1605 respectively correspond to the data of 20/40/80 bytes output per cycle in channel 2. When checked, the input and output of each channel must be consistent. For example, if 40 is selected for the input of channel 1, 40 must also be selected for the output, and if 20 is selected for the input of channel 2, 20 must also be selected for the output.	LXCycleComBytesNum_20	FALSE
TBL	Input	DWORD	First byte address of the master station to store data	It is the first byte address of area M for data storage, such as 200, which means that the storage address is a space starting from %MW200. If it is a read-type instruction, the data value read back is stored in this data area. If it is a write type instruction, the data value to be written is stored in this data area	0	FALSE
BuffSize	Input	DWORD	Byte size of the data memory stored in the master station	Byte size of the data memory to store data by the master station	0	FALSE
SlaveAddr	Input	BYTE	Slave station address	The address of MODBUS slave station is 1~247, and 0 is the broadcast address.	0	FALSE
Function Code	Input	BYTE	Instruction number	MODBUS instruction number, supporting instructions 01, 02, 03, 04, 05, 06, 15, and 16	0	FALSE
RegAddr	Input	WORD	Register start address	MODBUS register start address	0	FALSE
Quantity	Input	WORD	Number of registers	Number of MODBUS registers	0	FALSE
TimeOut	Input	DWORD	Slave station response timeout (ms)	Starting from the time when EN_R is enabled, if the correct slave station response frame is not received within the specified time, the receiving process will be aborted. Q is set to 1, and Error prompt times out.	0	FALSE
Q	Output	BOOL	Completion flag	1: Completed 0: Not completed	FALSE	FALSE
Error	Output	BYTE	Error	0: No error	0	FALSE

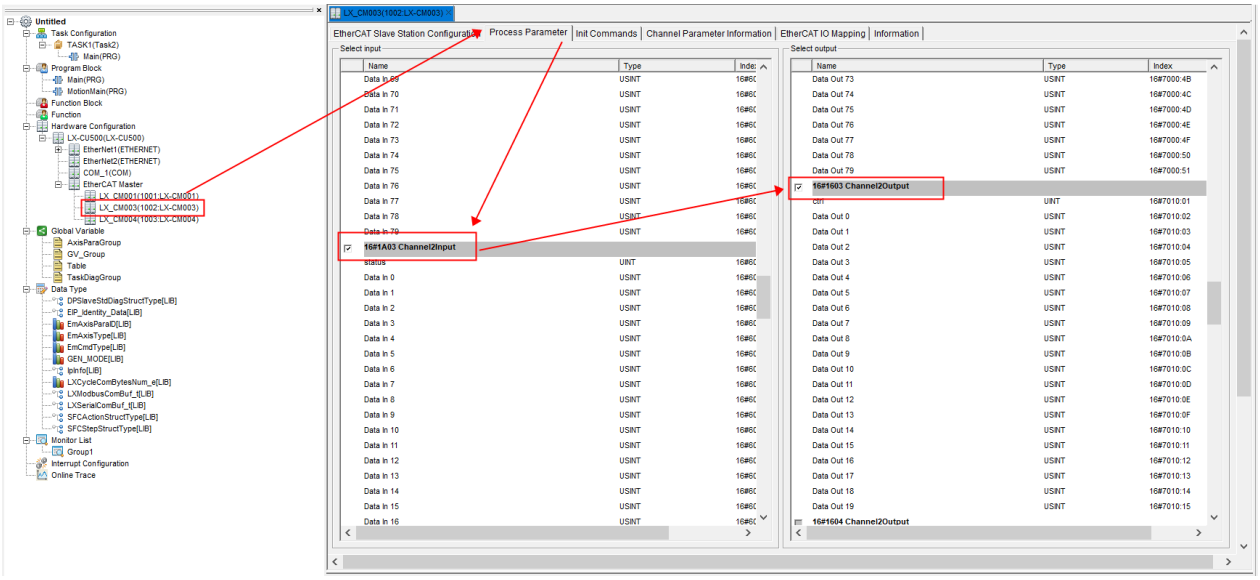
				2: Timeout 4: This module does not support function codes 10: Respond to unrequested responses 16: Incorrect slave station address 33: TBL or BuffSize out of range 34: BuffSize input 0 35: Timeout input 0 64: The slave station returns an invalid instruction 65: The slave station returns an invalid data address 66: The slave station returns an invalid data value 67: The slave station returns a slave station device fault 68: The slave station returns other error diagnosis 129: InOffset out of range 130: OutOffset out of range 131: CycleBytesNum is illegal 132: Error occurred in data acquisition		
RecvDataBytes	Output	WORD	Number of data bytes read	Number of data bytes actually read by the read operation	0	FALSE

7.1.19.3 Example

Using the first serial port module channel 2 as the configuration method of the MODBUS RTU master station, it reads the 10 analog output registers at the first address 3000 to 3009 of the No. 2 slave station register through the No. 3 instruction. Store them in the %MW200 address of the master station PLC. The communication parameters are baud rate 9600, 8 data bits, 1 stop bit, and no parity. The bus transmits 20 bytes of data per cycle.

- Setting channel parameters of master station

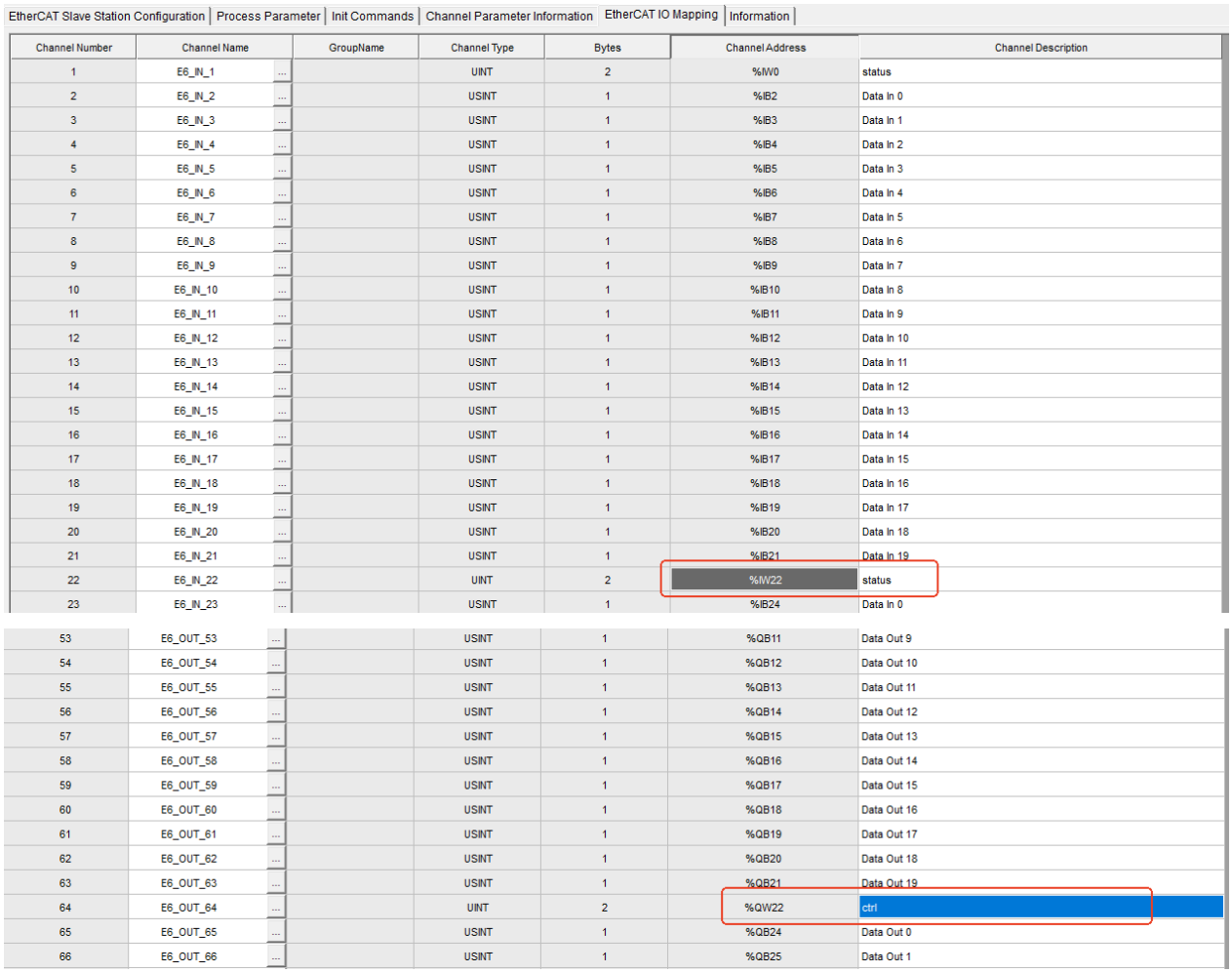
(1) Double-click to open the first serial port module, and check 16#1A03 and 16#1603 on the process parameter page.



(2) Click to enter channel parameter information, and click channel2Setting to enter channel 2 parameter configuration. The configuration results are shown in the figure below.

Number	Parameter name	Parameter value	Data Type	Default value	Maximum	Minimum
1	Channel State	Enable	USINT	Enable		
2	Baudrate	9600 Baud	USINT	9600 Baud		
3	Databits	8	USINT	8		
4	Parity	NONE	USINT	NONE		
5	Stopbits	1	USINT	1		
6	Communication type	RS485	USINT	RS485		
7	SendContinuous State	Enable	USINT	Enable		
8	Frame Interval	0	USINT	0	255	0

(3) Find the second status channel address in the EtherCAT IO mapping, which is 22, and the second ctrl channel address, which is also 22.



EtherCAT Slave Station Configuration | Process Parameter | Init Commands | Channel Parameter Information | EtherCAT IO Mapping | Information

Channel Number	Channel Name	GroupName	Channel Type	Bytes	Channel Address	Channel Description
1	E6_IN_1		UINT	2	%IW0	status
2	E6_IN_2		USINT	1	%IB2	Data In 0
3	E6_IN_3		USINT	1	%IB3	Data In 1
4	E6_IN_4		USINT	1	%IB4	Data In 2
5	E6_IN_5		USINT	1	%IB5	Data In 3
6	E6_IN_6		USINT	1	%IB6	Data In 4
7	E6_IN_7		USINT	1	%IB7	Data In 5
8	E6_IN_8		USINT	1	%IB8	Data In 6
9	E6_IN_9		USINT	1	%IB9	Data In 7
10	E6_IN_10		USINT	1	%IB10	Data In 8
11	E6_IN_11		USINT	1	%IB11	Data In 9
12	E6_IN_12		USINT	1	%IB12	Data In 10
13	E6_IN_13		USINT	1	%IB13	Data In 11
14	E6_IN_14		USINT	1	%IB14	Data In 12
15	E6_IN_15		USINT	1	%IB15	Data In 13
16	E6_IN_16		USINT	1	%IB16	Data In 14
17	E6_IN_17		USINT	1	%IB17	Data In 15
18	E6_IN_18		USINT	1	%IB18	Data In 16
19	E6_IN_19		USINT	1	%IB19	Data In 17
20	E6_IN_20		USINT	1	%IB20	Data In 18
21	E6_IN_21		USINT	1	%IB21	Data In 19
22	E6_IN_22		UINT	2	%IW22	status
23	E6_IN_23		USINT	1	%IB24	Data In 0
53	E6_OUT_53		USINT	1	%QB11	Data Out 9
54	E6_OUT_54		USINT	1	%QB12	Data Out 10
55	E6_OUT_55		USINT	1	%QB13	Data Out 11
56	E6_OUT_56		USINT	1	%QB14	Data Out 12
57	E6_OUT_57		USINT	1	%QB15	Data Out 13
58	E6_OUT_58		USINT	1	%QB16	Data Out 14
59	E6_OUT_59		USINT	1	%QB17	Data Out 15
60	E6_OUT_60		USINT	1	%QB18	Data Out 16
61	E6_OUT_61		USINT	1	%QB19	Data Out 17
62	E6_OUT_62		USINT	1	%QB20	Data Out 18
63	E6_OUT_63		USINT	1	%QB21	Data Out 19
64	E6_OUT_64		UINT	2	%QW22	ctrl
65	E6_OUT_65		USINT	1	%QB24	Data Out 0
66	E6_OUT_66		USINT	1	%QB25	Data Out 1

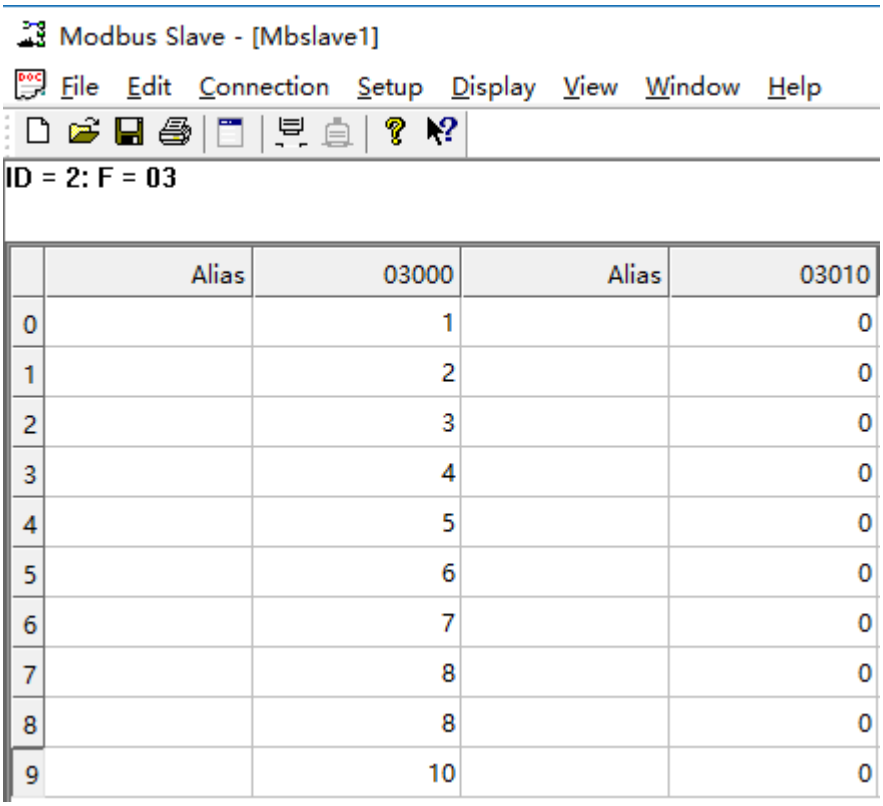
(4) After the functional block establishes a connection with the slave station device as a Modbus master station, it can set the enable signal EN_R from FALSE to TRUE, and then execute the current instruction. Wait for Q to become TRUE, when ReadLen equals 20, then read the module data of 10 registers, including a total of 20 bytes, and store it in %MW200.

The following is the schematic diagram of data storage in the slave station and the schematic diagram of data storage read by the master station.

Channel Number	Channel Name	Index	Channel Description
1	channel1Setting	0x8000:00	CH1 Config Data
2	channel2Setting	0x8010:00	CH2 Config Data

Parameters of channel channel2Setting

Number	Parameter name	Parameter value	Data Type	Default value	Maximum	Minimum
1	Channel State	Enable	USINT	Enable		
2	Baudrate	9600 Baud	USINT	9600 Baud		
3	Databits	8	USINT	8		
4	Parity	NONE	USINT	NONE		
5	Stopbits	1	USINT	1		
6	Communication type	RS232	USINT	RS232		
7	SendContinuous State	Enable	USINT	Enable		
8	Frame Interval	35	USINT	0	255	0

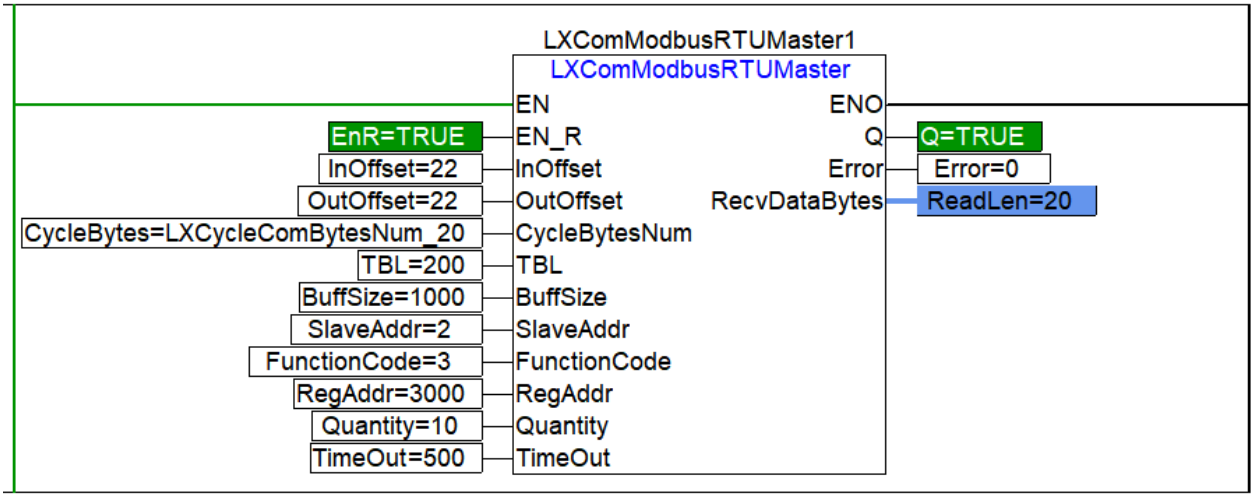


	Alias	03000	Alias	03010
0		1		0
1		2		0
2		3		0
3		4		0
4		5		0
5		6		0
6		7		0
7		8		0
8		8		0
9		10		0

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Buffer[0]	%MW200		WORD	1	FALSE	Not Public	
0002	Buffer[1]	%MW202		WORD	2	FALSE	Not Public	
0003	Buffer[2]	%MW204		WORD	3	FALSE	Not Public	
0004	Buffer[3]	%MW206		WORD	4	FALSE	Not Public	
0005	Buffer[4]	%MW208		WORD	5	FALSE	Not Public	
0006	Buffer[5]	%MW210		WORD	6	FALSE	Not Public	
0007	Buffer[6]	%MW212		WORD	7	FALSE	Not Public	
0008	Buffer[7]	%MW214		WORD	8	FALSE	Not Public	
0009	Buffer[8]	%MW216		WORD	8	FALSE	Not Public	
0010	Buffer[9]	%MW218		WORD	10	FALSE	Not Public	

- Configuration example:

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	LXComModbusRTUMaster1			LXCOMMODBUSRTUMASTER		FALSE	Not Public	
0002	EnR			BOOL	TRUE	FALSE	Not Public	
0003	InOffset			DWORD	22	FALSE	Not Public	
0004	OutOffset			DWORD	22	FALSE	Not Public	
0005	CycleBytes			LXCYCLECOMBYTESNUM_E	LXCycleComBytesNum_20	FALSE	Not Public	
0006	TBL			DWORD	200	FALSE	Not Public	
0007	BuffSize			DWORD	1000	FALSE	Not Public	
0008	SlaveAddr			BYTE	2	FALSE	Not Public	
0009	FunctionCode			BYTE	3	FALSE	Not Public	
0010	RegAddr			WORD	3000	FALSE	Not Public	
0011	Quantity			WORD	10	FALSE	Not Public	
0012	TimeOut			DWORD	500	FALSE	Not Public	
0013	Q			BOOL	TRUE	FALSE	Not Public	
0014	Error			BYTE	0	FALSE	Not Public	
0015	ReadLen			WORD	20	FALSE	Not Public	




```
1  LXComModbusRTUMaster1(
2  EN_R := EnR,
3  InOffset := InOffset,
4  OutOffset := OutOffset,
5  CycleBytesNum := CycleBytes,
6  TBL := TBL,
7  BuffSize := BuffSize,
8  SlaveAddr := SlaveAddr,
9  FunctionCode := FunctionCode,
10 RegAddr := RegAddr,
11 Quantity := Quantity,
12 TimeOut := TimeOut,
13 Q=>Q,
14 Error=>Error,
15 RecvDataBytes=>ReadLen);
```

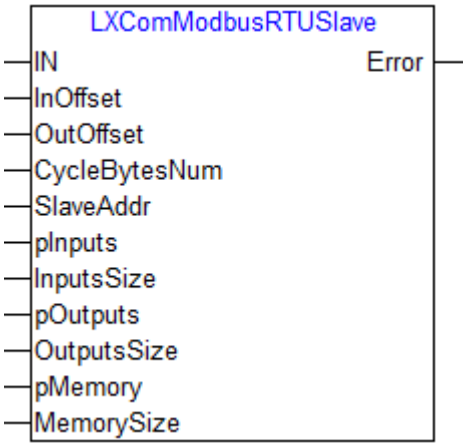
```
LXComModbusRTUMaster1
EnR = TRUE
InOffset = 22
OutOffset = 22
CycleBytes = LXCycleComBytesNu...
TBL = 200
BuffSize = 1000
SlaveAddr = 2
FunctionCode = 3
RegAddr = 3000
Quantity = 10
TimeOut = 500
Q = TRUE
Error = 0
ReadLen = 20
```

7.1.20 LXComModbusRTUSlave (MODBUS Slave)

7.1.20.1 Function

This functional block realizes the MODBUS RTU slave station function together with the LX-CM series serial port communication module.

This function can also be implemented via LXComMBSlave (MODBUS Slave(capture)).



LXComModbusRTUSlave Functional Block

7.1.20.2 Parameter

Parameter	Statement	Data Type	Function Description	Description of Parameter Values	Default Value	Power Fail Safeguard
IN	Input	BOOL	Enable, valid at high level	Communication takes effect when the level is high	FALSE	FALSE
InOffset	Input	DWORD	Area I offset address of periodic input data	INOFFSET depends on the channel address in the EtherCAT IO mapping page in the hardware configuration. The channel address corresponding to the first status is the INOFFSET of the first channel, and the channel address corresponding to the second status is the INOFFSET of the second channel.	0	FALSE
OutOffset	Input	DWORD	Area Q offset address of periodic output data	OUTOFFSET depends on the channel address in the EtherCAT IO mapping page in the hardware configuration. The channel address corresponding to the first ctrl is the OUTOFFSET of the first channel, and the channel address corresponding to the second ctrl is the OUTOFFSET of the second channel.	0	FALSE
CYCLEBYTESNUM	Input	LXCYCLECOMBYTESNUM_E	Data size transmitted periodically	For CYCLEBYTESNUM, the enumeration variable of type LXCYCLECOMBYTESNUM_E shall be filled in, and this value depends on the process parameters in the hardware configuration. On the left side of the process parameter page, 16#1A00/16#1A01/ 16#1A02 respectively correspond to the data of 20/40/80 bytes input per cycle in channel 1, and 16#1A03/ 16#1A04/16 #1A05 respectively correspond to the data of 20/40/80 bytes input per cycle in channel 2. On the right	LXCycleComBytesNum_20	FALSE

				side, 16#1600/ 16#1601/16#1602 respectively correspond to the data of 20/40/80 bytes output per cycle in channel 1, and 16#1603/ 16#1604/ 16#1605 respectively correspond to the data of 20/40/80 bytes output per cycle in channel 2. When checked, the input and output of each channel must be consistent. For example, if 40 is selected for the input of channel 1, 40 must also be selected for the output, and if 20 is selected for the input of channel 2, 20 must also be selected for the output.		
SlaveAddr	Input	BYTE	Slave station address	MODBUS slave station address, 1~247	0	FALSE
pInputs	Input	POINTER TO BYTE	Pointer address of area I (if not filled in, the address of area I will be used by default)	If the pointer is filled in, the area pointed to by the pointer is considered to be area I when the communication is performed. If it is not filled in or filled with 0, the default address of area I is used for communication.	0	FALSE
InputsSize	Input	DWORD	Length of the filled-in address of area I	Byte size of area I filled in	0	FALSE
pOutputs	Input	POINTER TO BYTE	Pointer address of area Q (if not filled in, the address of area Q will be used by default)	If the pointer is filled in, the area pointed to by the pointer is considered to be area Q when the communication is executed. If it is not filled in or filled with 0, the default address of area Q is used for communication.	0	FALSE
OutputsSize	Input	DWORD	Length of the filled-in address of area Q	Byte size of area Q filled in	0	FALSE
pMemory	Input	POINTER TO BYTE	Pointer address of area M (if not filled in, the address of area M will be used by default)	If the pointer is filled in, the area pointed to by the pointer is considered to be area M when the communication is executed. If it is not filled in or filled with 0, the default address of area M is used for communication.	0	FALSE
MemorySize	Input	DWORD	Length of the filled-in address of area M	Byte size of area M filled in	0	FALSE
Error	Output	BYTE	Error	0: No error 1: Received a data packet that failed the CRC check 2: Received a data packet from a non-local address 16: Incorrect slave station address 36: pInputs and InputsSize are enabled but the address area is invalid 37: pOutputs and OutputsSize are enabled but the address area is invalid	0	FALSE

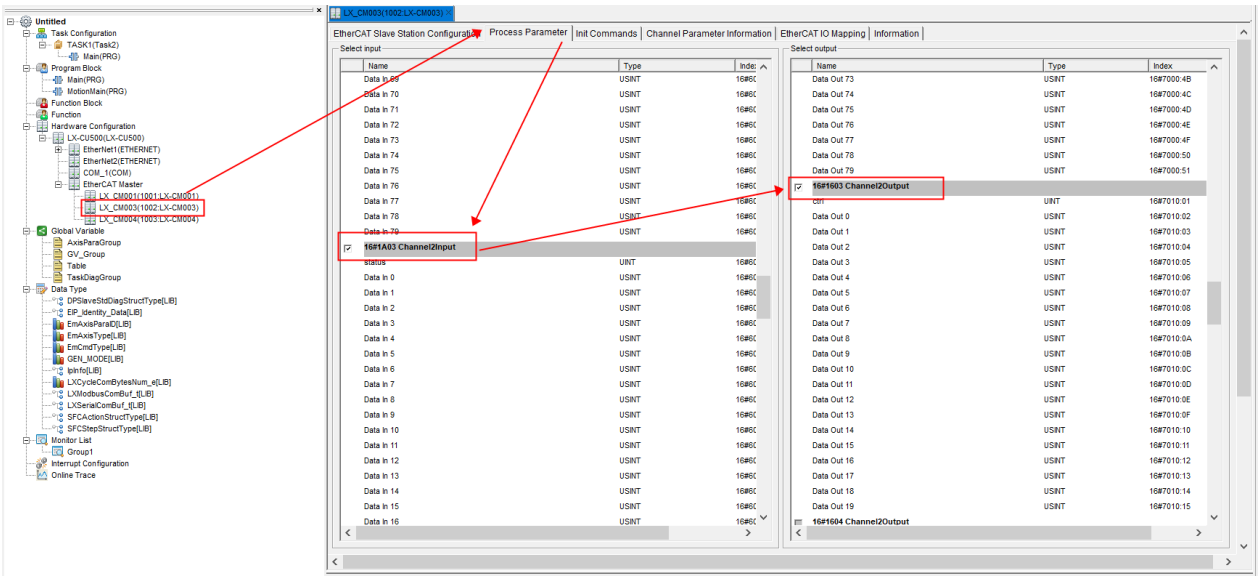
				38: pMemory and MemorySize are enabled but the address area is invalid 129: InOffset out of range 130: OutOffset out of range 131: CycleBytesNum is illegal 132: Error occurred in data acquisition		

7.1.20.3 Example

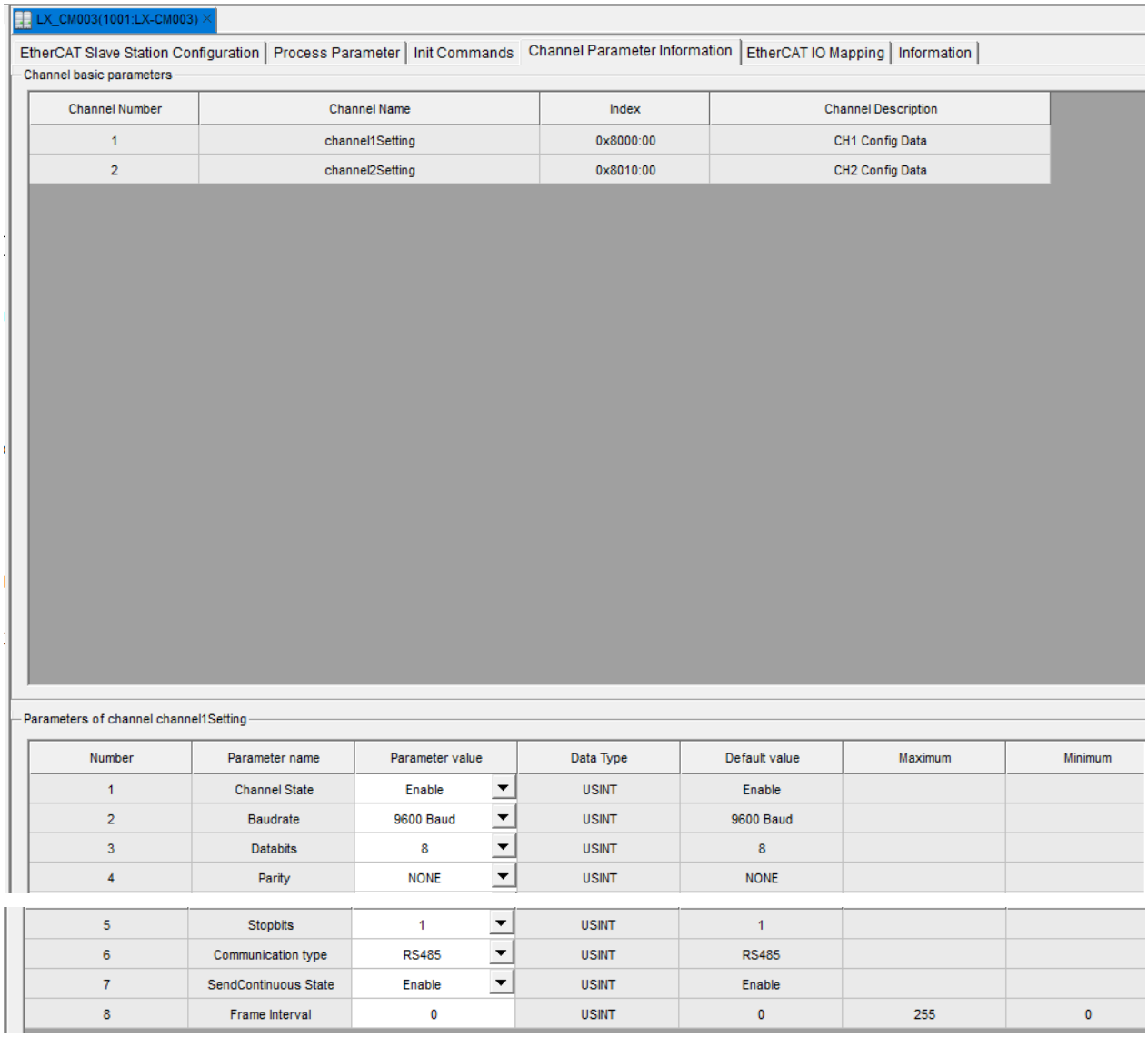
Using the first serial port module channel 2 as the configuration method of the MODBUS RTU slave station, the master station reads the 10 analog output registers at the %MW200 address of the station as the No. 2 slave station through the No. 3 instruction. The communication parameters are baud rate 9600, 8 data bits, 1 stop bit, and no parity. The bus transmits 20 bytes of data per cycle.

■ Setting channel parameters of slave station

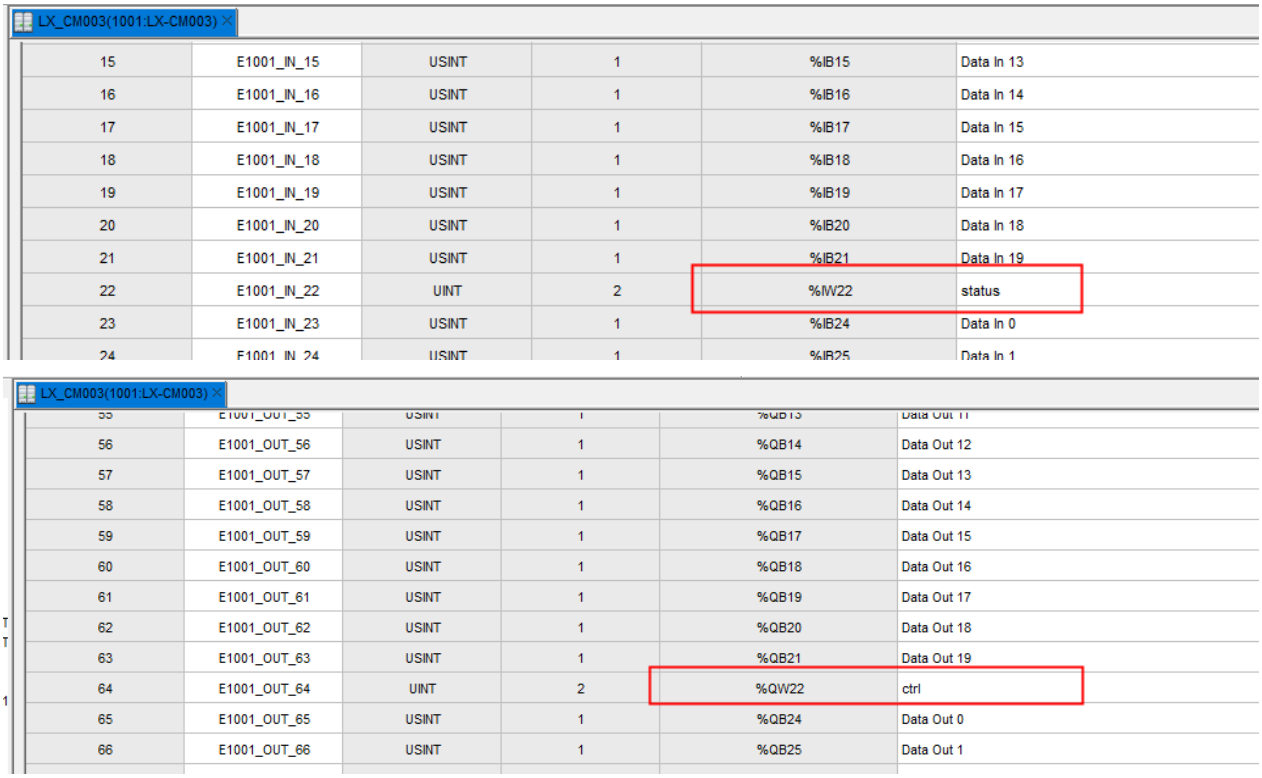
(1) Double-click to open the first serial port module, and check 16#1A03 and 16#1603 on the process parameter page.



(2) Click to enter channel parameter information, and click channel2Setting to enter channel 2 parameter configuration. The configuration results are shown in the figure below.



(3) Find the second status channel address in the EtherCAT IO mapping, which is 22, and the second ctrl channel address, which is also 22.



LX_CM003(1001:LX-CM003)

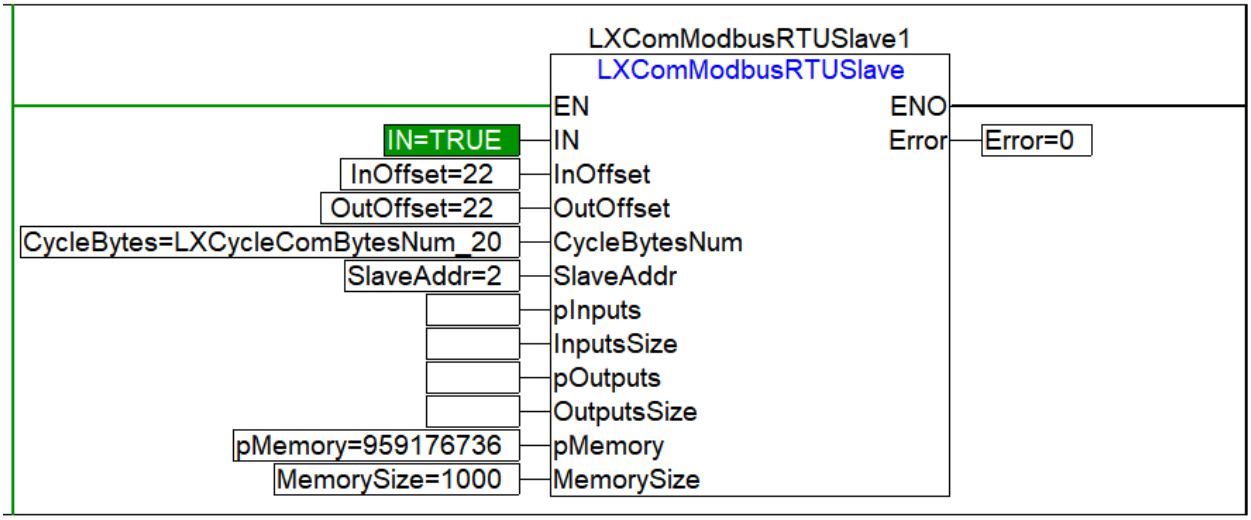
15	E1001_IN_15	USINT	1	%IB15	Data In 13
16	E1001_IN_16	USINT	1	%IB16	Data In 14
17	E1001_IN_17	USINT	1	%IB17	Data In 15
18	E1001_IN_18	USINT	1	%IB18	Data In 16
19	E1001_IN_19	USINT	1	%IB19	Data In 17
20	E1001_IN_20	USINT	1	%IB20	Data In 18
21	E1001_IN_21	USINT	1	%IB21	Data In 19
22	E1001_IN_22	UINT	2	%IW22	status
23	E1001_IN_23	USINT	1	%IB24	Data In 0
24	E1001_IN_24	USINT	1	%IB25	Data In 1

LX_CM003(1001:LX-CM003)

56	E1001_OUT_56	USINT	1	%QB14	Data Out 12
57	E1001_OUT_57	USINT	1	%QB15	Data Out 13
58	E1001_OUT_58	USINT	1	%QB16	Data Out 14
59	E1001_OUT_59	USINT	1	%QB17	Data Out 15
60	E1001_OUT_60	USINT	1	%QB18	Data Out 16
61	E1001_OUT_61	USINT	1	%QB19	Data Out 17
62	E1001_OUT_62	USINT	1	%QB20	Data Out 18
63	E1001_OUT_63	USINT	1	%QB21	Data Out 19
64	E1001_OUT_64	UINT	2	%QW22	ctrl
65	E1001_OUT_65	USINT	1	%QB24	Data Out 0
66	E1001_OUT_66	USINT	1	%QB25	Data Out 1

- Configuration example:

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	LXComModbusRTUSlave1			LXCOMMODBUSRTUSLAVE		FALSE	Not Public	
0002	Memory			LXMODBUSCOMBUF_T		FALSE	Not Public	
0003	pMemory			POINTER TO BOOL	959176736	FALSE	Not Public	
0004	IN			BOOL	TRUE	FALSE	Not Public	
0005	InOffset			DWORD	22	FALSE	Not Public	
0006	OutOffset			DWORD	22	FALSE	Not Public	
0007	CycleBytes			LXCYCLECOMBYTESNUM_E	LXCycleComBy...	FALSE	Not Public	
0008	SlaveAddr			BYTE	2	FALSE	Not Public	
0009	MemorySize			DWORD	1000	FALSE	Not Public	
0010	Error			BYTE	0	FALSE	Not Public	




```
1  pMemory := ADR(Memory);
2  LXComModbusRTUSlave1(
3  IN := IN,
4  InOffset := InOffset,
5  OutOffset := OutOffset,
6  CycleBytesNum := CycleBytes,
7  SlaveAddr := SlaveAddr,
8  pMemory := pMemory,
9  MemorySize := MemorySize,
10 Error=>Error);
```

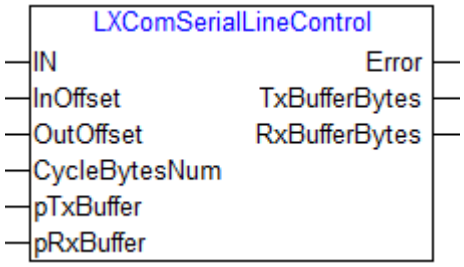
```
pMemory = 959176736                Memory
LXComModbusRTUSlave1
IN = TRUE
InOffset = 22
OutOffset = 22
CycleBytes = LXCycleComBytesN...
SlaveAddr = 2
pMemory = 959176736
MemorySize = 1000
Error = 0
```

7.1.21 LXComSerialLineControl (Free Protocol)

7.1.21.1 Function

When LX PLC communicates with other devices via serial ports, if the protocol supported by these third-party devices is not the standard MODBUS RTU protocol, it can be implemented through free protocol configuration. This functional block, together with the Rx/Tx functional block and the LX-2CM series serial port communication module, realizes free protocol communication and can complete the Rx and Tx

functions.



LXComSerialLineControl Functional Block

7.1.21.2 Parameter

Parameter	Statement	Data Type	Function Description	Description	Initial Value	Power Fail Safeguard
IN	Input	BOOL	Enable, valid at high level	Communication takes effect when the level is high	FALSE	FALSE
InOffset	Input	DWORD	Area I offset address of periodic input data	INOFFSET depends on the channel address in the EtherCAT IO mapping page in the hardware configuration. The channel address corresponding to the first status is the INOFFSET of the first channel, and the channel address corresponding to the second status is the INOFFSET of the second channel.	0	FALSE
OutOffset	Input	DWORD	Area Q offset address of periodic output data	OUTOFFSET depends on the channel address in the EtherCAT IO mapping page in the hardware configuration. The channel address corresponding to the first ctrl is the OUTOFFSET of the first channel, and the channel address corresponding to the second ctrl is the OUTOFFSET of the second channel.	0	FALSE
CycleBytesNum	Input	LXCYCLECOMBYTESNUM _E	Data size transmitted periodically	For CYCLEBYTESNUM, the enumeration variable of type LXCYCLECOMBYTESNUM_E shall be filled in, and this value depends on the process parameters in the hardware configuration. On the left side of the process parameter page, 16#1A00/ 16#1A01/ 16#1A02 respectively correspond to the data of 20/40/80 bytes input per cycle in channel 1, and 16#1A03/ 16#1A04/ 16#1A05 respectively correspond to the data of 20/40/80 bytes input per cycle in channel 2. On the right side, 16#1600/ 16#1601/ 16#1602 respectively correspond to the data of 20/40/80 bytes output per cycle in channel 1, and 16#1603/ 16#1604/ 16#1605 respectively correspond to the data of 20/40/80 bytes output per cycle in channel 2. When checked, the input and output of each channel must be consistent. For example, if 40 is selected for the input of channel 1, 40 must also be selected for	LXCycleComBytesNum_20	FALSE

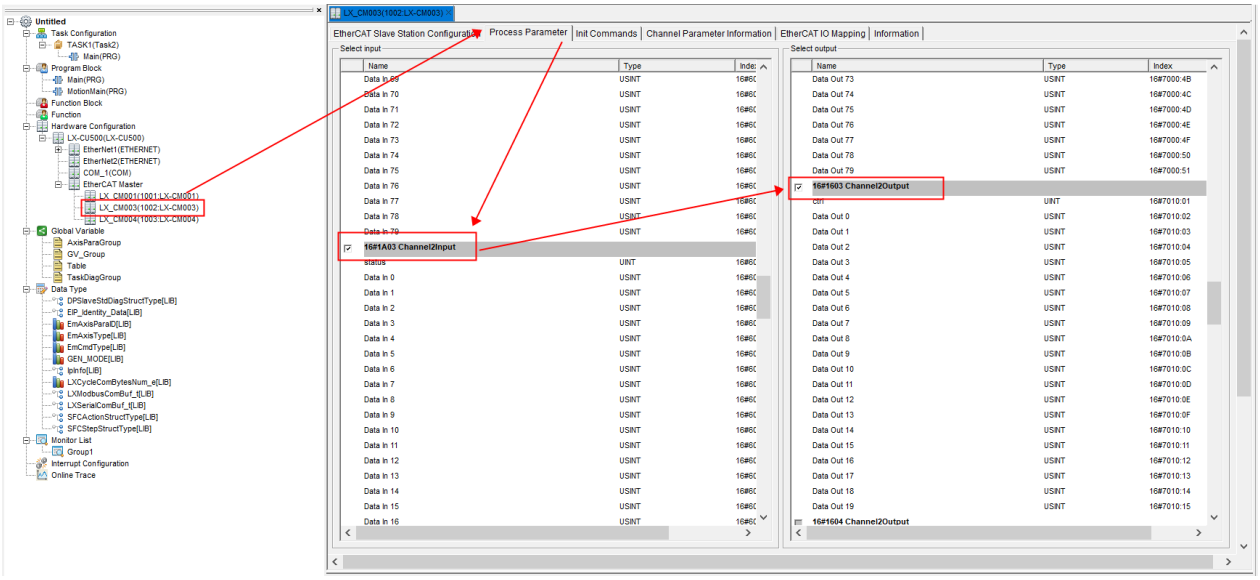
				the output, and if 20 is selected for the input of channel 2, 20 must also be selected for the output.		
pTxBuffer	Input	POINTE R TO LXSERIA LCOMBUF_T	Tx communication buffer pointer	Pointer to Tx buffer	0	FALSE
pRxBuffer	Input	POINTE R TO LXSERIA LCOMBUF_	Rx communication buffer pointer	Pointer to Rx buffer	0	FALSE
Error	Output	BYTE	Error	0: No error 1: Incorrect buffer pointer input 129: InOffset out of range 130: OutOffset out of range 131: CycleBytesNum is illegal 132: Incorrect cycle data acquisition	0	FALSE
TXBUFFE RBYTES	Output	DWORD	Existing data length of Tx communication buffer	Existing data length of Tx communication buffer	0	FALSE
RXBUFFE RBYTES	Output	DWORD	Existing data length of Rx communication buffer	Existing data length of Rx communication buffer	0	FALSE

7.1.21.3 Example

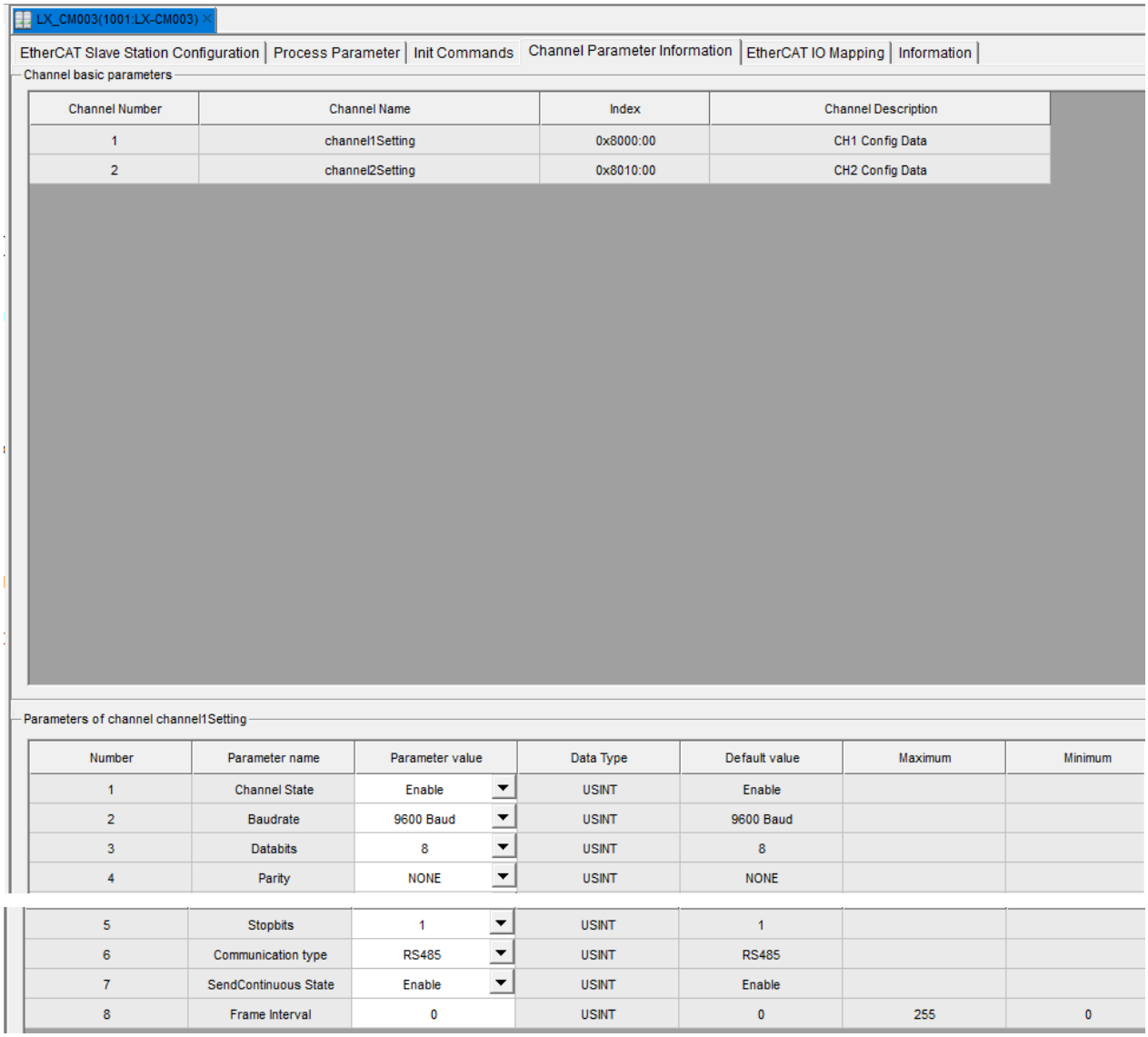
Using the first serial port module channel 2 as the configuration method of the free communication protocol for communication, it performs continuous transmission and automatic summary off. The communication parameters are baud rate 9600, 8 data bits, 1 stop bit, and no parity. The bus transmits 20 bytes of data per cycle.

- Setting channel parameters of slave station

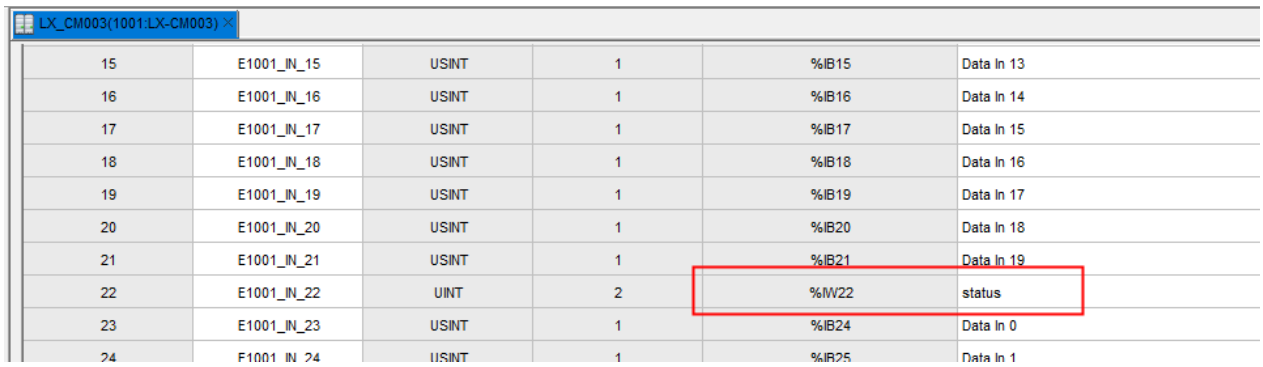
(1) Double-click to open the first serial port module, and check 16#1A03 and 16#1603 on the process parameter page.

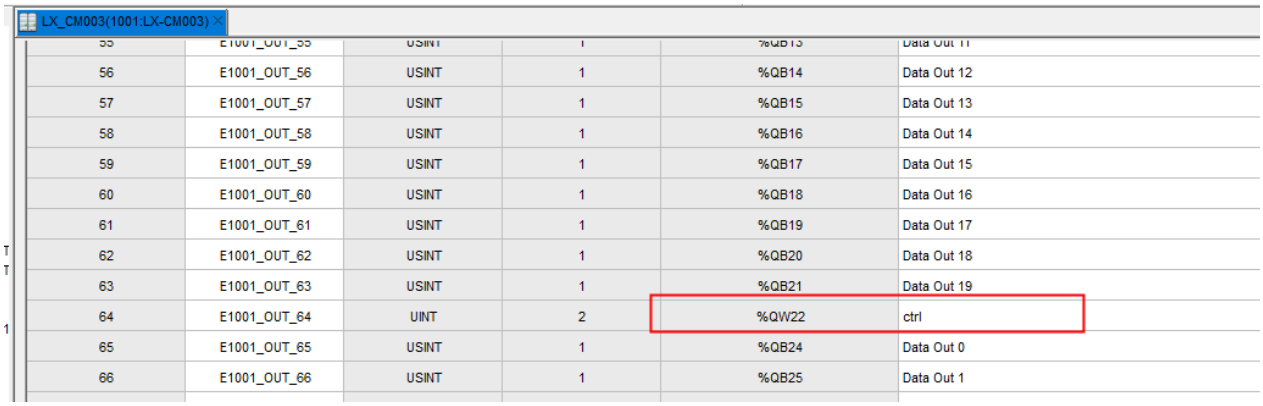


(2) Click to enter channel parameter information, and click channel2Setting to enter channel 2 parameter configuration. The configuration results are shown in the figure below.



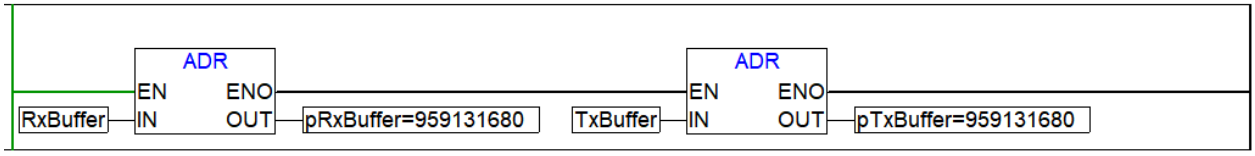
(3) Find the second status channel address in the EtherCAT IO mapping, which is 22, and the second ctrl channel address, which is also 22.

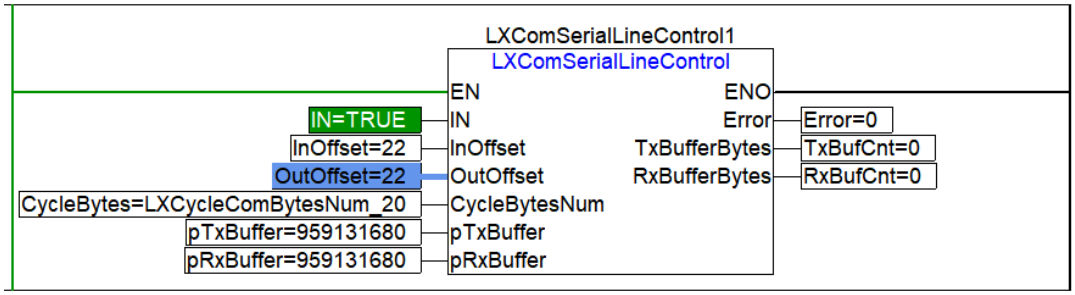




- Configuration example:

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	RxBuffer			LXSERIALCOMBUF_T		FALSE	Not Public	
0002	pRxBuffer			POINTER TO LXSERIALCOMBUF_T	959131680	FALSE	Not Public	
0003	TxBuffer			LXSERIALCOMBUF_T		FALSE	Not Public	
0004	pTxBuffer			POINTER TO LXSERIALCOMBUF_T	959131680	FALSE	Not Public	
0005	LXComSerialLineControl1			LXCOMSERIALLINECONTROL		FALSE	Not Public	
0006	IN			BOOL	TRUE	FALSE	Not Public	
0007	InOffset			DWORD	22	FALSE	Not Public	
0008	OutOffset			DWORD	22	FALSE	Not Public	
0009	CycleBytes			LXCYCLECOMBYTESNUM_E	LXCycleComBy...	FALSE	Not Public	
0010	Error			BYTE	0	FALSE	Not Public	
0011	TxBufCnt			DWORD	0	FALSE	Not Public	
0012	RxBufCnt			DWORD	0	FALSE	Not Public	






```
1   pRxBuffer := ADR(RxBuffer);
2   pTxBuffer := ADR(TxBuffer);
3
4   LXComSerialLineControl1(
5   IN := IN,
6   InOffset := InOffset,
7   OutOffset := OutOffset,
8   CycleBytesNum := CycleBytes,
9   pTxBuffer := pTxBuffer,
10  pRxBuffer := pRxBuffer,
11  Error=>Error,
12  TxBufferBytes=>TxBufCnt,
13  RxBufferBytes=>RxBufCnt);
```

```
pRxBuffer = 959131680                RxBuffer
pTxBuffer = 959131680                TxBuffer

LXComSerialLineControl1
IN = TRUE
InOffset = 22
OutOffset = 22
CycleBytes = LXCycleComBytesNu...
pTxBuffer = 959131680
pRxBuffer = 959131680
Error = 0
TxBufCnt = 0
RxBufCnt = 0
```

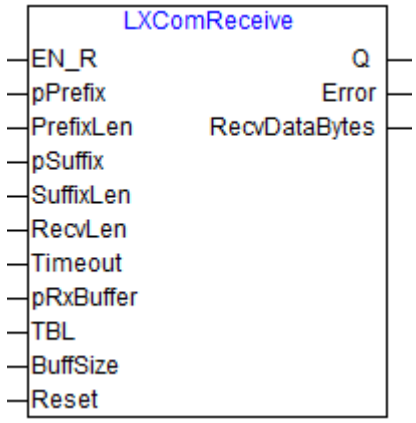
After the functional block establishes a connection as a free protocol party, it can set the enable signal IN from FALSE to TRUE, and then perform communication. When a third-party module sends data to this channel through the serial port module, the data is temporarily stored in RxBuffer, and RxBufCnt indicates the data size to be retrieved in the current RxBuffe. When there is data in TxBuffer, the data starts to be sent to the third-party module through the serial port module.

7.1.22 LXComReceive (Free Protocol Communication Data Reception)

7.1.22.1 Function

This functional block receives third-party module data transmitted through the serial port module together with LXComSerialLineControl.

This function can also be implemented via LXComFreePortReceive (Free Protocol Communication Data Reception(capture)).



LXComReceive Functional Block

7.1.22.2 Parameter

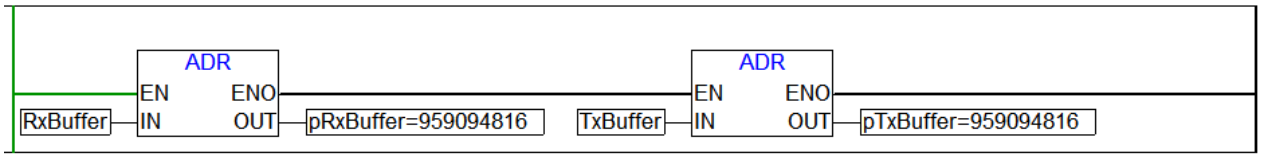
Parameter	Statement	Data Type	Function Description	Description	Initial Value	Power Fail Safeguard
EN_R	Input	BOOL	Enabled, valid on rising edge	Each time an Rx action is executed, EN_R shall be set from FALSE to TRUE. When it is checked that Q changes from FALSE to TRUE, it means that the execution of this reception is completed, and EN_R can be set to FALSE.	FALSE	FALSE
pPrefix	Input	POINTER TO BYTE	Prefix pointer	If a variable pointer is passed to an input variable prefix, the characters of the received data must be the same as this prefix, while other characters are discarded. After the prefix is used, if no suffix is specified, it will be received with the incoming RecvLen data length. If no RecvLen is passed in, it will be received with the incoming Timeout. If no Timeout is passed in, it will be considered an invalid Rx mode.	0	FALSE
PrefixLen	Input	BYTE	Prefix length	Number of data bytes in the prefix	0	FALSE
pSuffix	Input	POINTER TO	Suffix pointer	If a variable pointer is passed to the input	0	FALSE

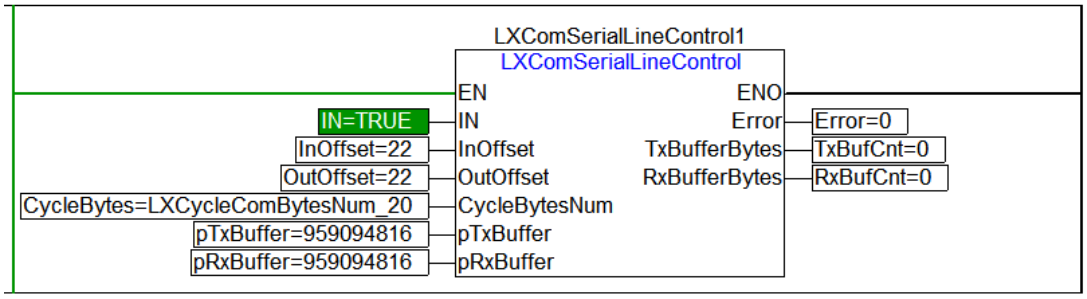
		BYTE		variable suffix, the input data is read until the end of the received data coincides with the suffix.		
SuffixLen	Input	BYTE	Suffix length	Number of data bytes in the suffix	0	FALSE
RecvLen	Input	DWORD	Receiving length	If no prefix or suffix is specified, the data of a RecvLen length is received starting from the first data.	0	FALSE
Timeout	Input	DWORD	Receiving timeout	If the prefix and suffix and the Rx data length are not specified, the data will be received within the timeout period. If there is no new data after the timeout period after the completion of one byte reception, the reception is considered completed.	0	FALSE
pRxBuffer	Input	POINTER TO LXSERIALCOMBUF_T	Rx communication buffer pointer	Pointer to Rx buffer	0	FALSE
TBL	Input	DWORD	Rx data buffer address	Offset of area M where the data address is to be stored, such as 200, which means that the storage address is a space starting from %MB200.	0	FALSE
BuffSize	Input	DWORD	Rx data buffer length	Byte size of memory area to store data	0	FALSE
Reset	Input	BYTE	Reset	Reset signal to restore functional block status	0	FALSE
Q	Output	BOOL	Completion flag	1: Completed 0: Not completed	FALSE	FALSE
Error	Output	BYTE	Error	0: No error 1: Incorrect buffer pointer input 2: No valid Rx mode 3: Incorrect input of prefix or suffix pointer 4: Sum of prefix and suffix exceeds 1000 5: RecvLen exceeds 1000 6: TBL or BuffSize out of range 7: Actual data length is greater than BuffSize	0	FALSE
RecvData Bytes	Output	WORD	Received data length	Actual length of data received after the reception is completed	0	FALSE

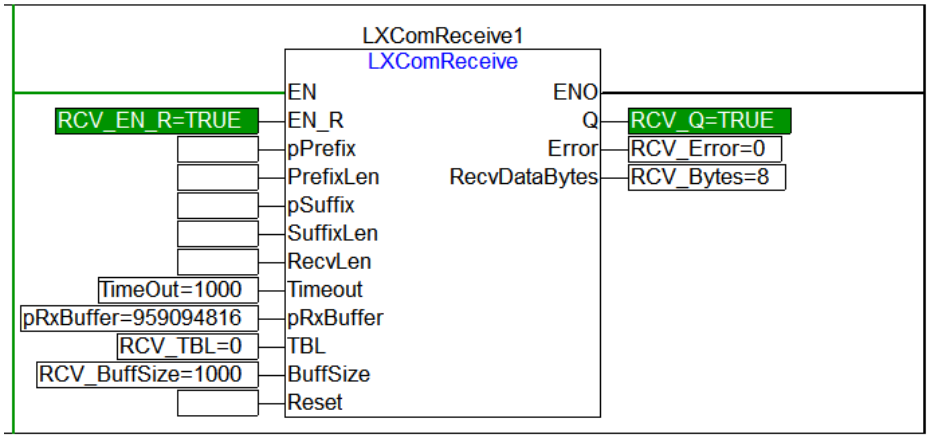
7.1.22.3 Example

When EN_R is set, the received data is placed in the 1000 bytes starting from MB0.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	RxBuffer			LXSERIALCOMBUF_T		FALSE	Not Public	
0002	pRxBuffer			POINTER TO LXSERIALCOMBUF_T	959094816	FALSE	Not Public	
0003	TxBuffer			LXSERIALCOMBUF_T		FALSE	Not Public	
0004	pTxBuffer			POINTER TO LXSERIALCOMBUF_T	959094816	FALSE	Not Public	
0005	LXComSerialLineControl1			LXCOMSERIALLINECONTROL		FALSE	Not Public	
0006	IN			BOOL	TRUE	FALSE	Not Public	
0007	InOffset			DWORD	22	FALSE	Not Public	
0008	OutOffset			DWORD	22	FALSE	Not Public	
0009	CycleBytes			LXCYCLECOMBYTESNUM_E	LXCycleComBy...	FALSE	Not Public	
0010	Error			BYTE	0	FALSE	Not Public	
0011	TxBufCnt			DWORD	0	FALSE	Not Public	
0012	RxBufCnt			DWORD	0	FALSE	Not Public	
0013	LXComReceive1			LXCOMRECEIVE		FALSE	Not Public	
0014	RCV_EN_R			BOOL	TRUE	FALSE	Not Public	
0015	TimeOut			DWORD	1000	FALSE	Not Public	
0016	RCV_TBL			DWORD	0	FALSE	Not Public	
0017	RCV_BuffSize			DWORD	1000	FALSE	Not Public	
0018	RCV_Q			BOOL	TRUE	FALSE	Not Public	
0019	RCV_Error			BYTE	0	FALSE	Not Public	
0020	RCV_Bytes			WORD	8	FALSE	Not Public	







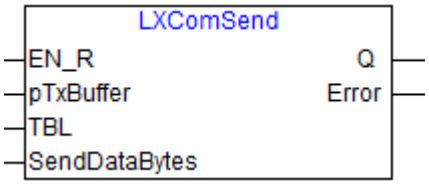
```
1  pRxBuffer := ADR(RxBuffer);                pRxBuffer = 959094816          RxBuffer
2  pTxBuffer := ADR(TxBuffer);                pTxBuffer = 959094816          TxBuffer
3
4  LXComSerialLineControl1(                   LXComSerialLineControl1
5  IN := IN,                                  IN = TRUE
6  InOffset := InOffset,                      InOffset = 22
7  OutOffset := OutOffset,                    OutOffset = 22
8  CycleBytesNum := CycleBytes,               CycleBytes = LXCycleComBytes...
9  pTxBuffer := pTxBuffer,                    pTxBuffer = 959094816
10 pRxBuffer := pRxBuffer,                    pRxBuffer = 959094816
11 Error=>Error,                              Error = 0
12 TxBufferBytes=>TxBufCnt,                   TxBufCnt = 0
13 RxBufferBytes=>RxBufCnt);                  RxBufCnt = 0
14
15 LXComReceive1(                             LXComReceive1
16 EN_R := RCV_EN_R,                          RCV_EN_R = TRUE
17 Timeout := TimeOut,                        TimeOut = 1000
18 pRxBuffer := pRxBuffer,                    pRxBuffer = 959094816
19 TBL := RCV_TBL,                            RCV_TBL = 0
20 BuffSize := RCV_BuffSize,                  RCV_BuffSize = 1000
21 Q=>RCV_Q,                                  RCV_Q = TRUE
22 Error=>RCV_Error,                          RCV_Error = 0
23 RecvDataBytes=>RCV_Bytes);                 RCV_Bytes = 8
```

7.1.23 LXComSend (Free Protocol Communication Data Transmission)

7.1.23.1 Function

This functional block sends data to a third-party module through the serial port module together with LXComSerialLineControl.

This function can also be implemented via LXComFreePortSend (Free Protocol Communication Data Transmission(capture)).



LXComSend Functional Block

7.1.23.2 Parameter

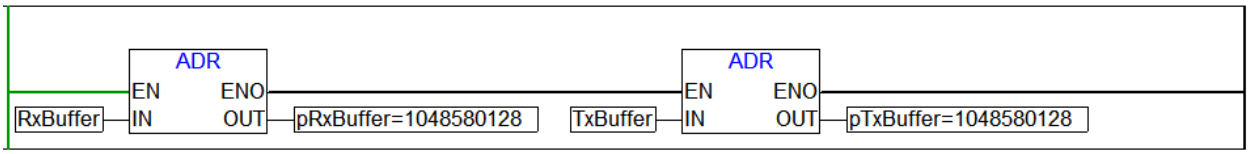
Parameter	Statement	Data Type	Function	Description of Parameter	Initial	Power Fail

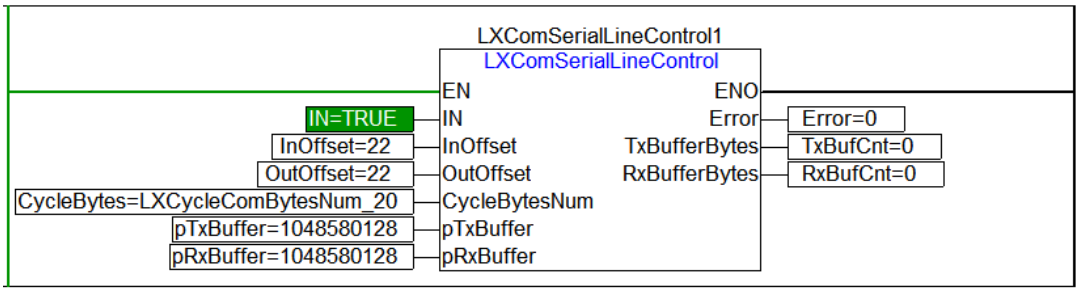
			Description	Values	Value	Safeguard
EN_R	Input	BOOL	Enabled, valid on rising edge	Each time a Tx action is executed, EN_R shall be set from FALSE to TRUE. When it is checked that Q changes from FALSE to TRUE, it means that the execution of this transmission is completed, and EN_R can be set to FALSE.	FALSE	FALSE
pTxBuffer	Input	POINTER TO LXSERIALCOMBUF_T	Tx communication buffer pointer	Pointer to Tx buffer	0	FALSE
TBL	Input	DWORD	Tx data buffer address	Offset of area M where the data address is to be sent, such as 200, which means that the address to send data is a space starting from %MB200.	0	FALSE
SendData Bytes	Input	DWORD	Tx data length	Length of data to be sent	0	FALSE
Q	Output	BOOL	Completion flag	1: Completed 0: Not completed	FALSE	FALSE
Error	Output	BYTE	Error	0: No error 1: Incorrect buffer pointer input 2: TBL or BuffSize out of range 3: SendDataBytes exceeds 1000	0	FALSE

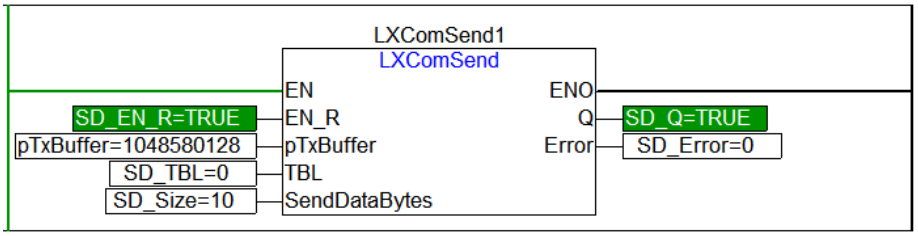
7.1.23.3 Example

When EN_R is set, 10 bytes of data starting from MB0 will be sent.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	RxBuffer			LXSERIALCOMBUF_T		FALSE	Not Public	
0002	pRxBuffer			POINTER TO LXSERIALCOMBUF_T	1048580128	FALSE	Not Public	
0003	TxBuffer			LXSERIALCOMBUF_T		FALSE	Not Public	
0004	pTxBuffer			POINTER TO LXSERIALCOMBUF_T	1048580128	FALSE	Not Public	
0005	LXComSerialLineControl1			LXCOMSERIALLINECONTROL		FALSE	Not Public	
0006	IN			BOOL	TRUE	FALSE	Not Public	
0007	InOffset			DWORD	22	FALSE	Not Public	
0008	OutOffset			DWORD	22	FALSE	Not Public	
0009	CycleBytes			LXCYCLECOMBYTESNUM_E	LXCycleComBy...	FALSE	Not Public	
0010	Error			BYTE	0	FALSE	Not Public	
0011	TxBufCnt			DWORD	0	FALSE	Not Public	
0012	RxBufCnt			DWORD	0	FALSE	Not Public	
0013	LXComSend1			LXCOMSEND		FALSE	Not Public	
0014	SD_EN_R			BOOL	TRUE	FALSE	Not Public	
0015	SD_TBL			DWORD	0	FALSE	Not Public	
0016	SD_Size			DWORD	10	FALSE	Not Public	
0017	SD_Q			BOOL	TRUE	FALSE	Not Public	
0018	SD_Error			BYTE	0	FALSE	Not Public	







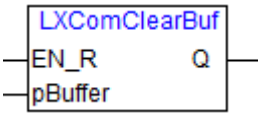
```
1   pRxBuffer := ADR(RxBuffer);
2   pTxBuffer := ADR(TxBuffer);
3
4   LXComSerialLineControl1(
5   IN := IN,
6   InOffset := InOffset,
7   OutOffset := OutOffset,
8   CycleBytesNum := CycleBytes,
9   pTxBuffer := pTxBuffer,
10  pRxBuffer := pRxBuffer,
11  Error=>Error,
12  TxBufferBytes=>TxBufCnt,
13  RxBufferBytes=>RxBufCnt);
14
15  LXComSend1(
16  EN_R := SD_EN_R,
17  pTxBuffer := pTxBuffer,
18  TBL := SD_TBL,
19  SendDataBytes :=SD_Size,
20  Q=>SD_Q,
21  Error=>SD_Error);
```

```
pRxBuffer = 1048580128          RxBuffe
pTxBuffer = 1048580128          TxBuffer

LXComSerialLineControl1
IN = TRUE
InOffset = 22
OutOffset = 22
CycleBytes = LXCycleComByt...
pTxBuffer = 1048580128
pRxBuffer = 1048580128
Error = 0
TxBufCnt = 0
RxBufCnt = 0

LXComSend1
SD_EN_R = TRUE
pTxBuffer = 1048580128
SD_TBL = 0
SD_Size = 10
SD_Q = TRUE
SD_Error = 0
```

7.1.24 LXComClearBuf (Free Protocol Communication Clear Buffer)

7.1.24.1 Function

Clear the internal PLC communication buffer (LXSERIALCOMBUF_T type).



LXComClearBuf Functional Block

7.1.24.2 Parameter

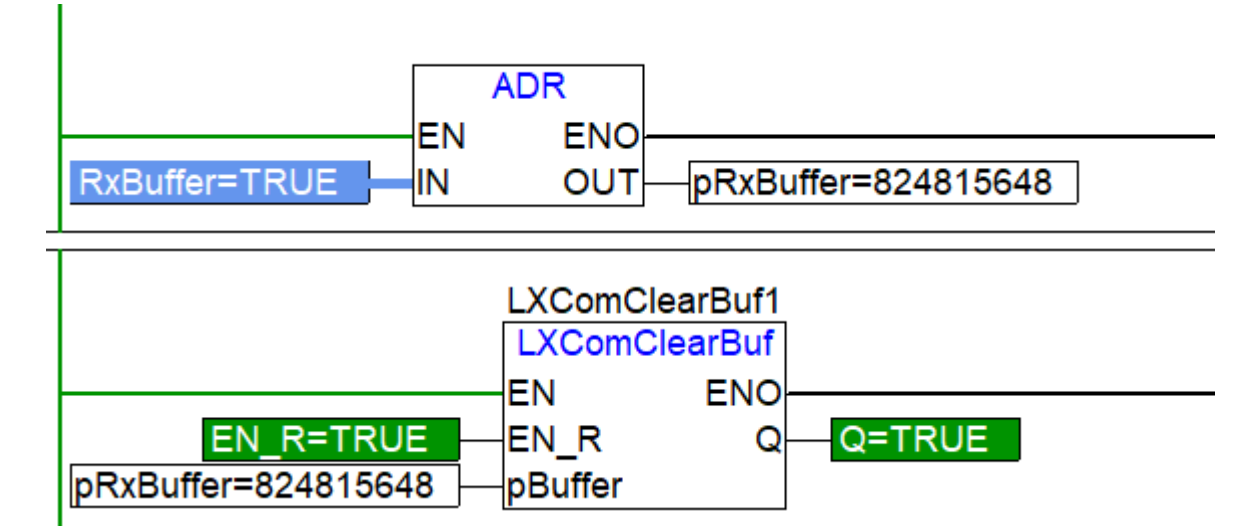
Parameter	Statement	Data Type	Function Description	Description	Default Value	Power Fail Safeguard
EN_R	Input	BOOL	Enabled, valid on rising edge	Each time a clearing action is executed, EN_R shall be set from FALSE to TRUE. When it is checked that Q changes from FALSE to TRUE, it means that the execution of this clearing is completed, and EN_R can be set to FALSE.	FALSE	FALSE
pBuffer	Input	POINTER TO LXSERIALCO	Rx or Tx communication	Pointer to Rx or Tx buffer	0	FALSE

		MBUF_T	buffer pointer			
Q	Output	BOOL	Completion flag	1: Completed 0: Not completed	FALSE	FALSE

7.1.24.3 Example

It is to clear the buffer data pointed to by the pointer when EN_R is set.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	LXComClearBuf1			LXCOMCLEARBUF		FALSE	Not Public	
0002	EN_R			BOOL	TRUE	FALSE	Not Public	
0003	Q			BOOL	TRUE	FALSE	Not Public	
0004	RxBuffer			BOOL	TRUE	FALSE	Not Public	
0005	pRxBuffer			POINTER TO LXSERIALCOMBUF_T	824815648	FALSE	Not Public	



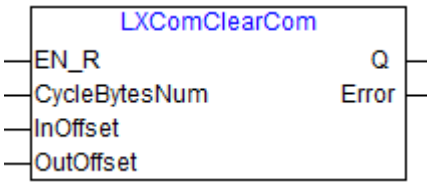

```
1  LXComClearBuf1(
2  EN_R := EN_R,
3  pBuffer := pRxBuffer,
4  Q=>Q);
```

```
LXComClearBuf1
EN_R = TRUE
pRxBuffer = 824815648
Q = TRUE
```

7.1.25 LXComClearCom (Clearing Serial Port Module Buffer)

7.1.25.1 Function

Clear the Tx and Rx buffers of the current serial port channel, which is equivalent to the channel reset operation, without clearing the PLC Tx and Rx buffers.



LXComClearCom Functional Block

7.1.25.2 Parameter

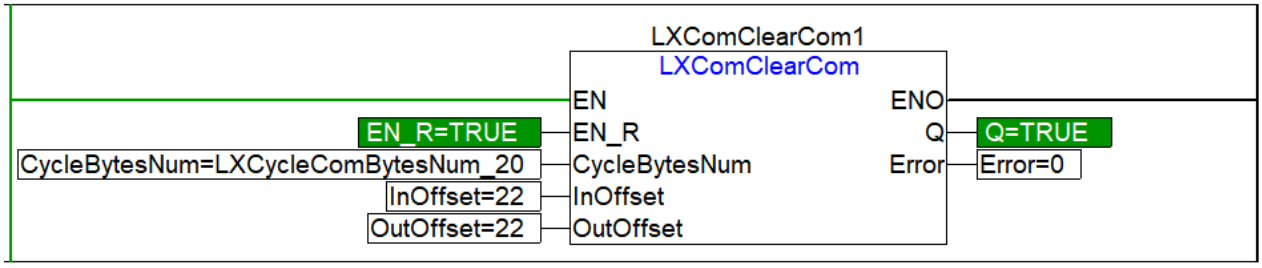
Parameter	Statement	Data Type	Function Description	Description	Initial Value	Power Fail Safeguard
EN_R	Input	BOOL	Enabled, valid on rising edge	Each time a clearing action is performed, EN_R shall be set from FALSE to TRUE. When it is detected that Q changes from FALSE to TRUE, it means that the clearing action is completed, and EN_R can be set to FALSE. After the clearing is completed, it is recommended not to enable this pin.	FALSE	FALSE
CycleBytesNum	Input	LXCYCLECOMBYTESNUM_E	Data size transmitted periodically	For CYCLEBYTESNUM, the enumeration variable of type LXCYCLECOMBYTESNUM_E shall be filled in, and the value depends on the process parameters in the hardware configuration. On the left side of the process parameter page, 16#1A00/ 16#1A01/ 16#1A02 respectively correspond to the data of 20/40/80 bytes input per cycle in channel 1, and 16#1A03/ 16#1A04/ 16#1A05 respectively correspond to the data of 20/40/80 bytes input per cycle in channel 2. On the right side, 16#1600/ 16#1601/ 16#1602 respectively correspond to the data of 20/40/80 bytes output per cycle in channel 1, and 16#1603/ 16#1604/ 16#1605 respectively correspond to the data of 20/40/80 bytes output per cycle in channel 2. When checked, the input and output of each channel must be consistent. For example, if 40 is selected for the input of channel 1, 40 must also be selected for the output, and if 20 is selected for the input of channel 2, 20 must also be selected for the output.	LXCycleComBytesNum_20	FALSE
InOffset	Input	DWORD	Area I offset address of periodic input data	INOFFSET depends on the channel address in the EtherCAT IO mapping page in the hardware configuration. The channel address corresponding to the first status is the INOFFSET of the first channel, and the channel address corresponding to the second status is the INOFFSET of the second channel.	0	FALSE
OutOffset	Input	DWORD	Area Q offset address of periodic output data	OUTOFFSET depends on the channel address in the EtherCATIO mapping page in the hardware configuration. The channel address corresponding to the first ctrl is the OUTOFFSET of the first channel, and the channel address corresponding to the	0	FALSE

				second ctrl is the OUTOFFSET of the second channel.		
Q	Output	BOOL	Completion flag	1: Completed 0: Not completed	FALSE	FALSE
Error	Output	BYTE	Error	0: No error 129: InOffset out of range 130: OutOffset out of range 131: CycleBytesNum is illegal	0	FALSE

7.1.25.3 Example

Clearing the Tx and Rx buffers of the current serial port channel is equivalent to the channel reset operation, without clearing the PLC Tx and Rx buffers.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	LXComClearCom1			LXCOMCLEARCOM		FALSE	Not Public	
0002	EN_R			BOOL	TRUE	FALSE	Not Public	
0003	CycleBytesNum			LXCYCLECOMBYTESNUM_E	LXCycleComBytesNum_20	FALSE	Not Public	
0004	InOffset			DWORD	22	FALSE	Not Public	
0005	OutOffset			DWORD	22	FALSE	Not Public	
0006	Q			BOOL	TRUE	FALSE	Not Public	
0007	Error			BYTE	0	FALSE	Not Public	




```
1  LXComClearCom1(
2  EN_R := EN_R,
3  CycleBytesNum := CycleBytesNum,
4  InOffset := InOffset,
5  OutOffset := OutOffset,
6  Q=>Q,
7  Error=>Error);
```

```
LXComClearCom1
EN_R = TRUE
CycleBytesNum = LXCycleComB...
InOffset = 22
OutOffset = 22
Q = TRUE
Error = 0
```

7.1.26 SISetConfig (Set the parameters of the SL protocol server)

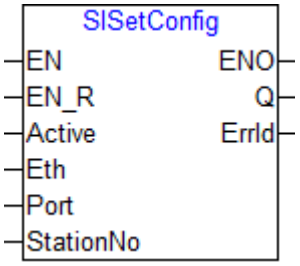
7.1.26.1 Version

- AutoThink Software Version: V3.2.3 and above.
- Firmware Version

Controller Type	Firmware Version
LX-CU430	LX-CU430_V1.1.0 and above.
LX-CU433	LX-CU433_V1.0.0 and above.
LX-CU500	LX-CU500_V1.3.0 and above.
LX-CU501	LX-CU501_V1.1.0 and above.
LX-CU510	LX-CU510_V1.0.0 and above.
LX-CU511	LX-CU511_V1.0.0 and above.

7.1.26.2 Function

Set the SL protocol server parameters.



SISetConfig Functional Block

7.1.26.3 Parameter

Parameter	Statement	Data Type	Description	Default Value	Power Fail Safeguard
EN_R	Input	BOOL	Rising edge trigger configuration	FALSE	FALSE
Active	Input	BOOL	Server settings FALSE: Deactivate TRUE: Activate	FALSE	FALSE
Eth	Input	BYTE	Specify network card 0: eth0 1: eth1	0	FALSE
Port	Input	DINT	Port number Default 5002, range 0~1, 1023, 1200 cannot be configured	0	FALSE
StationNo	Input	BYTE	Request target station number Default 255, configurable range 1~255	0	FALSE
Q	Output	BOOL	Output result FALSE: Not completed TRUE: Completed	FALSE	FALSE
ErrId	Output	DINT	Output error code 0: Success 0x1: Invalid network card number 0x2: Invalid station number 0x4: Port number cannot be configured 0x8: Server creation failed 0x10: The instance already exists	0	FALSE

7.1.26.4 Precautions for use

The SL protocol server supports only one client connection. When another client connects, the server actively closes the old connection.

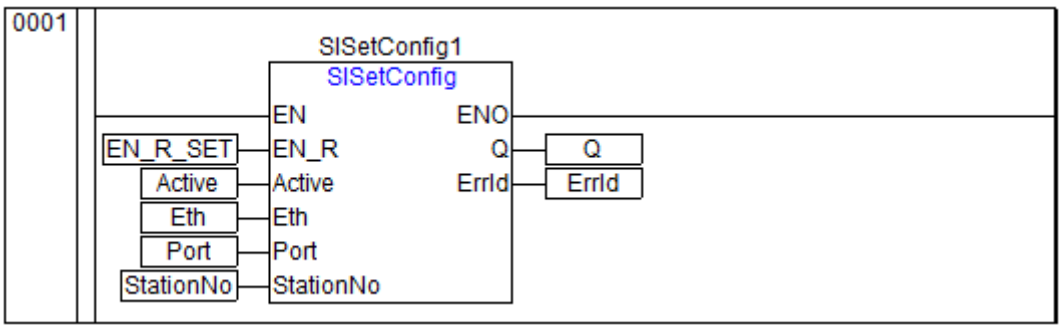
The D-area soft components are D0~D4999.

7.1.26.5 Example

1. Variable Definition

No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	SISetConfig1			SLSETCONFIG		FALSE	Not Public	
0002	EN_R_SET			BOOL	FALSE	FALSE	Not Public	
0003	Active			BOOL	FALSE	FALSE	Not Public	
0004	Eth			BYTE	0	FALSE	Not Public	
0005	Port			DINT	0	FALSE	Not Public	
0006	StationNo			BYTE	0	FALSE	Not Public	
0007	Q			BOOL	FALSE	FALSE	Not Public	
0008	ErrId			DINT	0	FALSE	Not Public	

2. LD Configuration Example

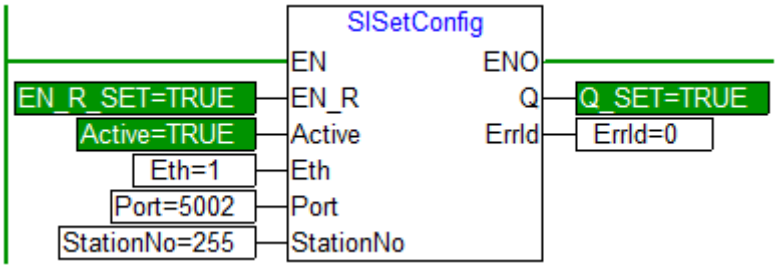


3. ST Configuration Example

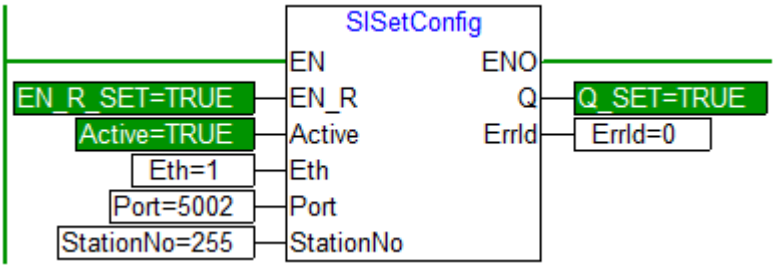
```
SISetConfig1(
    EN_R := EN_R_SET,
    Active := Active,
    Eth := Eth,
    Port := Port,
    StationNo := StationNo,
    Q=>Q,
    ErrId=>ErrId);
```

4. Operation Example

The rising edge of EN_R triggers the write configuration. The Active, Eth, Port, and StationNo fields must be written in advance.When reconfiguring the server, EN_R must first be written from a high level to a low level. When the configuration is successful, the output ErrId is 0, and Q is 1. When Enr is written to a low level, ErrId and Q are cleared to 0. An example of server configuration is shown in the figure below.



An ErrId value of 0x10 indicates that when configuring multiple function blocks, if one function block's EN_R input is at a high level, other function blocks are prohibited from being configured until the configured function block's EN_R input goes to a low level. During a full download, cold reset, or hot reset, all function blocks reset their configuration protection state. An example of the "Configuration in Progress" error report is shown in the figure below.



7.1.27 SlGetDiag (Get SL protocol server pDiagnose Information)

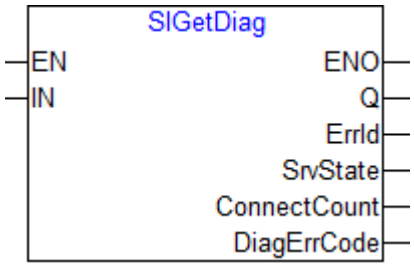
7.1.27.1 Version

- AutoThink Software Version: V3.2.3 and above.
- Firmware Version

Controller Type	Firmware Version
LX-CU430	LX-CU430_V1.1.0 and above.
LX-CU433	LX-CU433_V1.0.0 and above.
LX-CU500	LX-CU500_V1.3.0 and above.
LX-CU501	LX-CU501_V1.1.0 and above.
LX-CU510	LX-CU510_V1.0.0 and above.
LX-CU511	LX-CU511_V1.0.0 and above.

7.1.27.2 Function

Retrieve SL Protocol Server Diagnostic Information.



SlGetDiag Functional Block

7.1.27.3 Parameter

Parameter	Statement	Data Type	Description	Default Value	Power Fail Safeguard
IN	Input	BOOL	Function block enable, active high	FALSE	FALSE
Q	Output	BOOL	Output Result FALSE: Not Completed TRUE: Completed	FALSE	FALSE
ErrId	Output	DINT	Output Result 0: Success Not 0: Failure	0	FALSE
SrvState	Output	BOOL	Server Status FALSE: Server Deactivated TRUE: Server Activated	FALSE	FALSE
ConnectCount	Output	BYTE	Number of Client Connections 0: Not Connected	0	FALSE
DiagErrCode	Output	UDINT	Diagnostic Error Codes 0: Success 0x1: Server Receive Error 0x2: Server Send Error 0x4: Received Message Length Error	0	FALSE

7.1.27.4 Precautions for use

The SL protocol server supports only one client connection. When another client connects, the server actively closes the old connection.

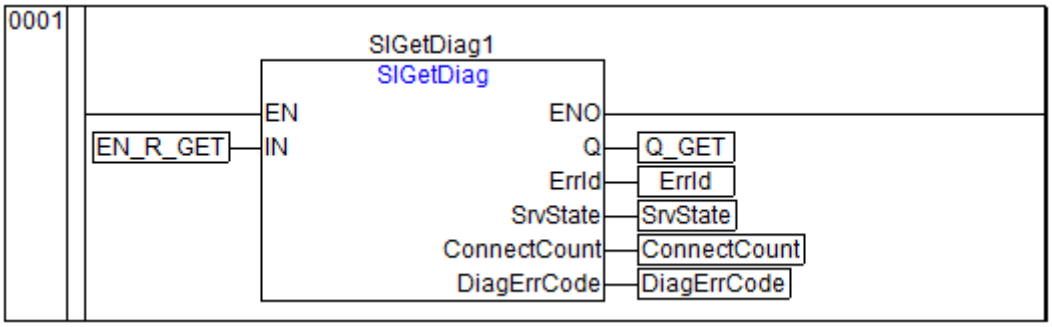
The D-area soft components are D0~D4999.

7.1.27.5 Example

1. Variable Definition

No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	SlGetDiag1			SLGETDIAG		FALSE	Not Public	
0002	EN_R_GET			BOOL	FALSE	FALSE	Not Public	
0003	Q_GET			BOOL	FALSE	FALSE	Not Public	
0004	ErrId			DINT	0	FALSE	Not Public	
0005	SrvState			BOOL	FALSE	FALSE	Not Public	
0006	ConnectCount			BYTE	0	FALSE	Not Public	
0007	DiagErrCode			UDINT	0	FALSE	Not Public	

2. LD Configuration Example

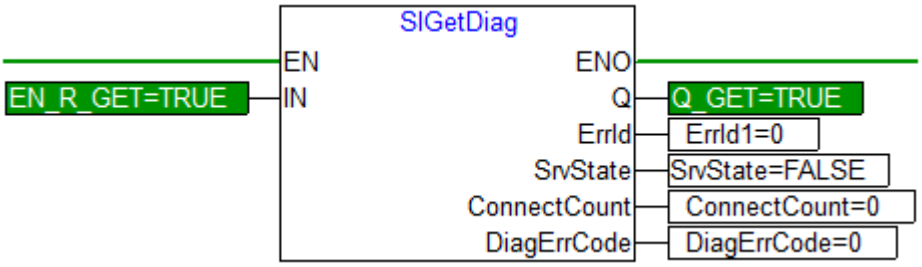


3. ST Configuration Example

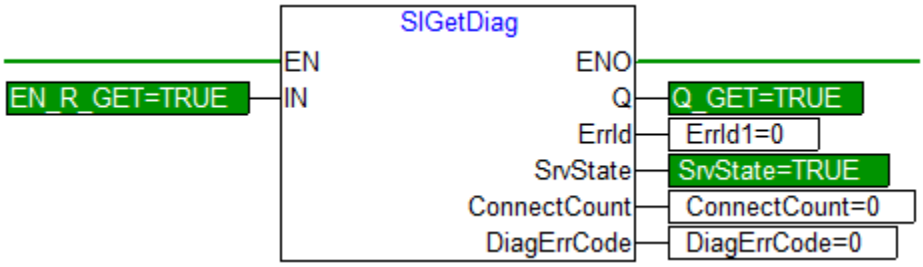
```
SlGetDiag1(
  IN := EN_R_GET,
  Q=>Q_GET,
  ErrId=>ErrId,
  SrvState=>SrvState,
  ConnectCount=>ConnectCount,
  DiagErrCode=>DiagErrCode);
```

4. Operation Example

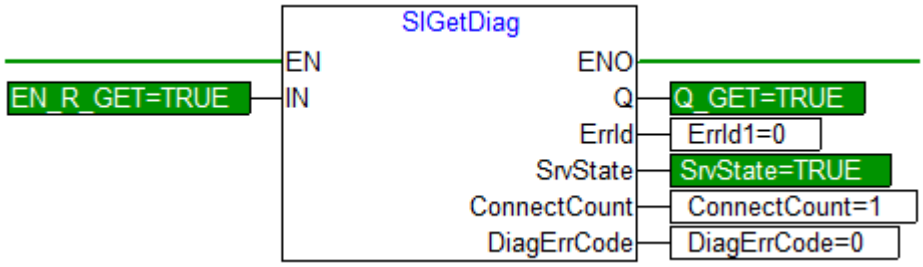
- When the IN input is high, output diagnostic information, as shown in the example below.



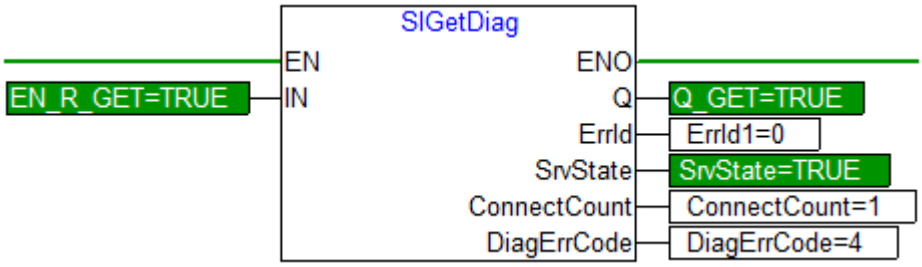
- SrvState diagnoses the server status, including activation and deactivation states. When deactivated, SrvState is FALSE. When activated, SrvState is TRUE. An example is shown in the figure below.



- ConnectCount diagnoses the number of connected clients. 0 indicates no connection. An example with 1 connection is shown in the figure below.



- An example with 1 connection is shown in thDiagErrCode diagnoses server faults, such as reporting a received message length error. An example is shown in the figure below.e figure below.



7.1.28 SLConfig (Set the parameters of the SL peotocol)

7.1.28.1 Version

- AutoThink Software Version: V3.2.3SP3 and above.
- Firmware Version

Controller Type	Firmware Version
LX-CU430	LX-CU430_V1.2.0 and above.
LX-CU433	LX-CU433_V1.1.0 and above.
LX-CU500	LX-CU500_V1.4.0 and above.
LX-CU501	LX-CU501_V1.2.0 and above.
LX-CU510	LX-CU510_V1.1.0 and above.
LX-CU511	LX-CU511_V1.0.1 and above.

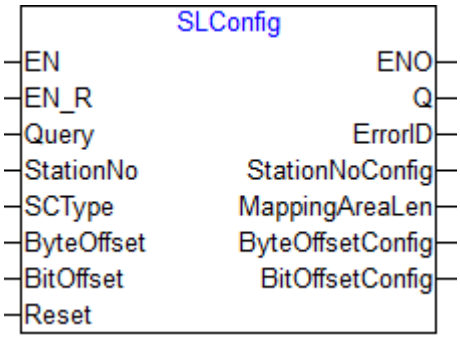
7.1.28.2 Function

Through this function block, you can set, retrieve, or reset parameters for soft components. The reset operation takes priority over setting and retrieving parameters.

When Query is FALSE, you can configure soft component parameters. The parameters to configure include StationNo, SCType, ByteOffset, and BitOffset.

When Query is TRUE, you can retrieve the configured soft component parameters. The configured parameters can be obtained through ByteOffsetConfig, BitOffsetConfig, and StationNoConfig.

When Reset is TRUE, the configured soft component parameters in the Mitsubishi server will be reset to their default values.



SLConfig Functional Block

7.1.28.3 Precautions

Once the function block configuration takes effect, it will remain unchanged even after power-off and restart, until the parameters are reconfigured or reset.

When the function block is triggered by the rising edge of EN_R, other SLConfig function blocks must wait until the EN_R of this function block is set to FALSE before they can be used. Otherwise, error code 2 (an instance is currently being configured) will be reported.

During the configuration process of the function block, power-off is strictly prohibited, as it may cause the operation to fail.

The access address of a word soft component = starting offset byte + number of bytes occupied by a single soft component * soft component number. During configuration, ensure that this address does not exceed the size of the output parameter MappingAreaLen to avoid memory access out-of-bounds issues.

7.1.28.4 Parameter

Parameter	Statement	Data Type	Description	Default Value	Power Fail Safeguard
EN_R	INPUT	BOOL	Rising edge trigger configuration.	FALSE	FALSE

Query	INPUT	BOOL	FALSE: Set soft component parameters TRUE: Get soft component parameters	FALSE	FALSE
StationNo	INPUT	BYTE	Request station No.: 1~255	0	FALSE
SCType	INPUT	SL_SC_TYPE	Enumeration type: eSC_D: D soft component (data register) eSC_W: W soft component (link register)	eSC_D	FALSE
ByteOffset	INPUT	UDINT	Starting offset byte in M area for the soft element selected by SCType.	0	FALSE
BitOffset	INPUT	Byte	Starting bit of the offset byte in M area for the soft element selected by SCType (range: 0~7, default: 0).	0	FALSE
Reset	INPUT	BOOL	Reset configuration: FALSE: No reset / Reset failed TRUE: Reset	FALSE	FALSE
Q	OUTPUT	BOOL	Completion status: FALSE: Not completed TRUE: Completed	FALSE	FALSE
ErrorID	OUTPUT	UDINT	Error code definitions: See the table below.	0	FALSE
StationNoConfig	OUTPUT	BYTE	Get configured station number.	0	FALSE
MappingAreaLen	OUTPUT	UDINT	Get total size of M area.	0	FALSE
ByteOffsetConfig	OUTPUT	UDINT	Retrieve the byte offset configuration of the M area corresponding to the soft component selected by SCType.	0	FALSE
BitOffsetConfig	OUTPUT	BYTE	Retrieve the starting bit configuration of the starting offset byte in the M area corresponding to the soft component selected by SCType.	0	FALSE

7.1.28.5 Error Code

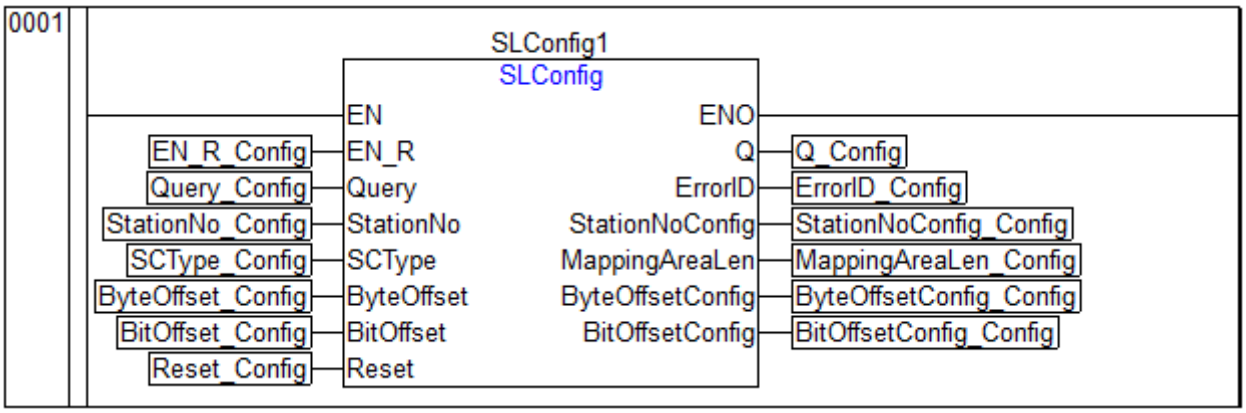
Error code	Description
0	Success.
1	Invalid request station number.
2	Configuration instance already in progress.
3	Soft element mapping offset out of range.
4	Illegal soft element bit offset..
5	Configuration file write failed.

7.1.28.6 Example

1. Variable Definition

No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	SLConfig1			SLCONFIG		FALSE	Not Public	
0002	EN_R_Config			BOOL	FALSE	FALSE	Not Public	
0003	Query_Config			BOOL	FALSE	FALSE	Not Public	
0004	StationNo_Config			BYTE	0	FALSE	Not Public	
0005	SCType_Config			SL_SC_TYPE	eSC_D	FALSE	Not Public	
0006	ByteOffset_Config			UDINT	0	FALSE	Not Public	
0007	BitOffset_Config			BYTE	0	FALSE	Not Public	
0008	Reset_Config			BOOL	FALSE	FALSE	Not Public	
0009	Q_Config			BOOL	FALSE	FALSE	Not Public	
0010	ErrorID_Config			UDINT	0	FALSE	Not Public	
0011	StationNoConfig_Config			BYTE	0	FALSE	Not Public	
0012	MappingAreaLen_Config			UDINT	0	FALSE	Not Public	
0013	ByteOffsetConfig_Config			UDINT	0	FALSE	Not Public	
0014	BitOffsetConfig_Config			BYTE	0	FALSE	Not Public	

2. LD Configuration Example



3. ST Configuration Example

```
SLConfig1(
    EN_R := EN_R_Config,
    Query := Query_Config,
    StationNo := StationNo_Config,
    SCType := SCType_Config,
    ByteOffset := ByteOffset_Config,
    BitOffset := BitOffset_Config,
    Reset := Reset_Config,
    Q=>Q_Config,
    ErrorID=>ErrorID_Config,
    StationNoConfig=>StationNoConfig_Config,
    MappingAreaLen=>MappingAreaLen_Config,
    ByteOffsetConfig=>ByteOffsetConfig_Config,
```

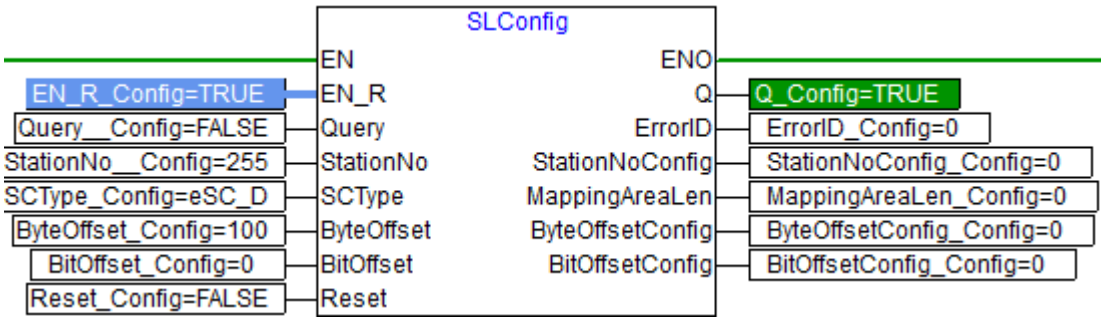
BitOffsetConfig=>BitOffsetConfig_Config);

4. Next, take the LD program as an example to demonstrate SLConfig parameter configuration, parameter query, parameter reset, configuration parameter check, configuration parameter power-down retention, and multi-instance protection.

■ Parameter Configuration

Set Query and Reset parameters to FALSE. Input the request station number, soft component type, starting byte of the soft component M-area mapping, and the bit starting position of the starting byte. A rising edge on EN_R triggers the parameter configuration. The output pins will have the following states:

□ Configuration successful: Q is TRUE, ErrorID is 0, and ByteOffsetConfig, BitOffsetConfig, StationNoConfig, and MappingAreaLen are all 0.

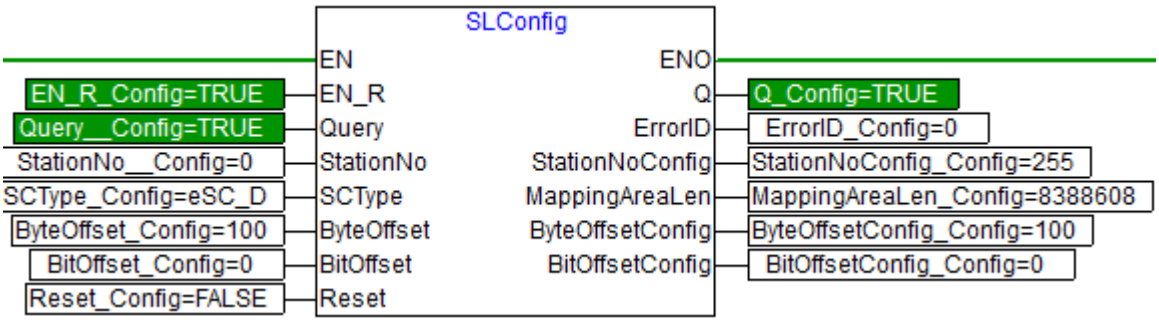


□ Configuration failed: Q is FALSE, ErrorID is a non-zero value, and ByteOffsetConfig, BitOffsetConfig, StationNoConfig, and MappingAreaLen are all 0.

■ Parameter Query

Set the Query parameter to TRUE and the Reset parameter to FALSE. Select the soft component type to query in SCType, and ignore other input parameters. A rising edge on EN_R triggers the parameter query. The output pins will have the following states:

□ Query successful: Q is TRUE, ErrorID is 0, and the output parameters ByteOffsetConfig, BitOffsetConfig, StationNoConfig, and MappingAreaLen are valid.



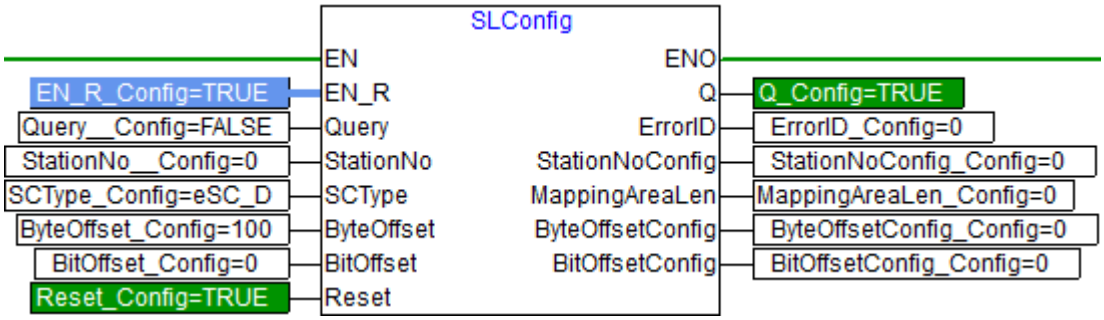
□ Query failed: Q is FALSE, ErrorID is a non-zero value, and the output parameters ByteOffsetConfig, BitOffsetConfig, StationNoConfig, and MappingAreaLen are all 0.

■ Parameter Reset

Set the Reset parameter to TRUE (other input parameters are ignored). A rising edge on EN_R triggers the parameter reset. The request station number becomes 255. All soft components' M-

area mapping byte offsets and starting bit offsets are reset to 0. The output pins will have the following states:

- Reset successful: Q is TRUE, ErrorID is 0, and ByteOffsetConfig, BitOffsetConfig, StationNoConfig, MappingAreaLen are all 0.

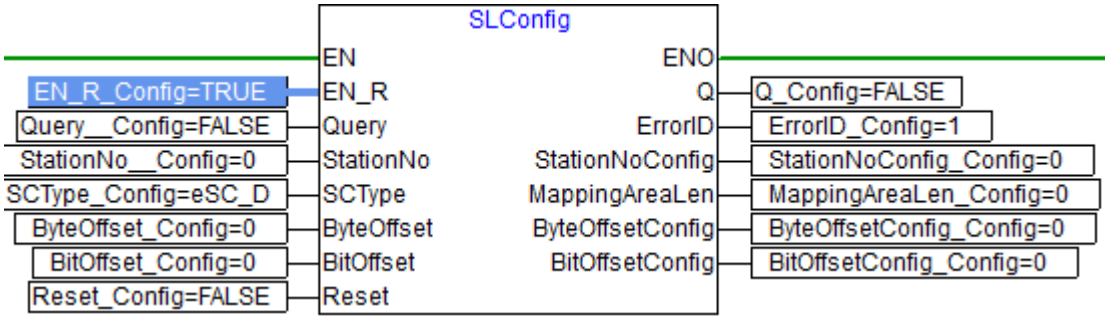


- Reset failed: Q is FALSE, ErrorID is a non-zero value, and ByteOffsetConfig, BitOffsetConfig, StationNoConfig, MappingAreaLen are all 0.

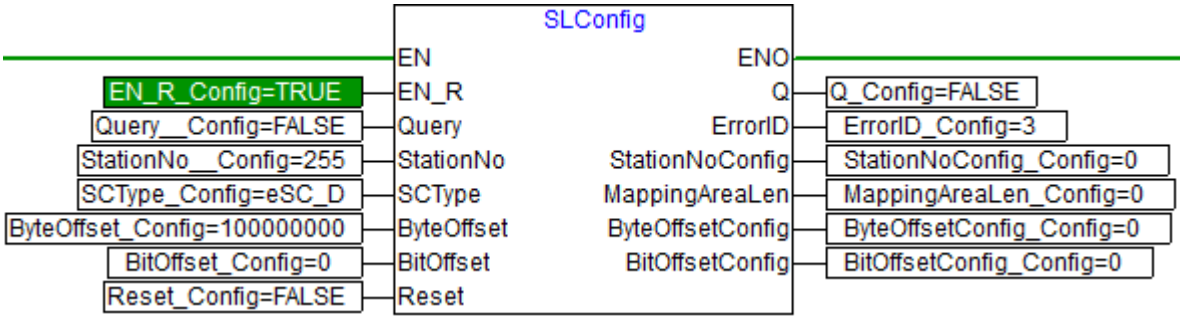
■ Configuration Parameter Validation

Configure the function block's input parameters with both Query and Reset set to FALSE. A rising edge on EN_R triggers validation checks on the input parameters, including:

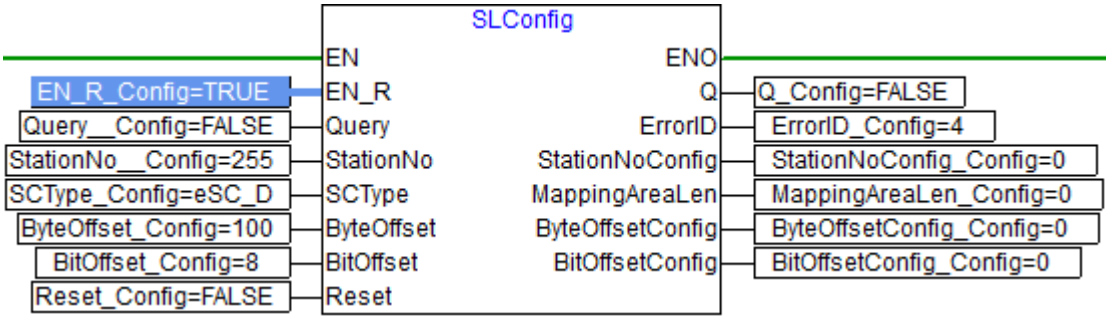
- Request station number range: Must be 1~255. If out of range, ErrorID reports 1 (Invalid station number).



- ByteOffset exceeds M-area size: Reports error code 3 (Soft component mapping offset out of range).



- BitOffset range (0~7) violation: Reports error code 4 (Invalid soft component bit offset).



7.1.29 SLConnect (Set SL protocol server connection)

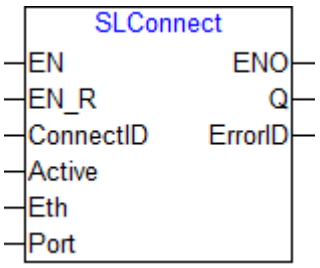
7.1.29.1 Version

- AutoThink Software Version: V3.2.3SP3 and above.
- Firmware Version

Controller Type	Firmware Version
LX-CU430	LX-CU430_V1.2.0 and above.
LX-CU433	LX-CU433_V1.1.0 and above.
LX-CU500	LX-CU500_V1.4.0 and above.
LX-CU501	LX-CU501_V1.2.0 and above.
LX-CU510	LX-CU510_V1.1.0 and above.
LX-CU511	LX-CU511_V1.0.1 and above.

7.1.29.2 Function

Set SL protocol server connection.



SLConnect Functional Block

7.1.29.3 Parameter

Parameter	Statement	Data Type	Description	Default Value	Power Fail Safeguard
EN_R	INPUT	BOOL	Rising edge trigger configuration.	FALSE	FALSE
ConnectID	INPUT	BYTE	Connection ID: Range 1~8, corresponding to 8 channels of connections.	1	FALSE

Active	INPUT	BOOL	Server activation/deactivation settings: FALSE: Deactivate TRUE: Activate	FALSE	FALSE
Eth	INPUT	BYTE	Port Number: 0: P1 1: P2	0	FALSE
Port	INPUT	UINT	Port number configuration: Valid range: 1025~65534 Ports 1200 and 1201 cannot be configured.	0	FALSE
Q	OUTPUT	BOOL	Completion Status: FALSE: Incomplete TRUE: Complete	FALSE	FALSE
ErrorID	OUTPUT	DINT	For error code definitions, please refer to the table below.	0	FALSE

7.1.29.4 Error Code

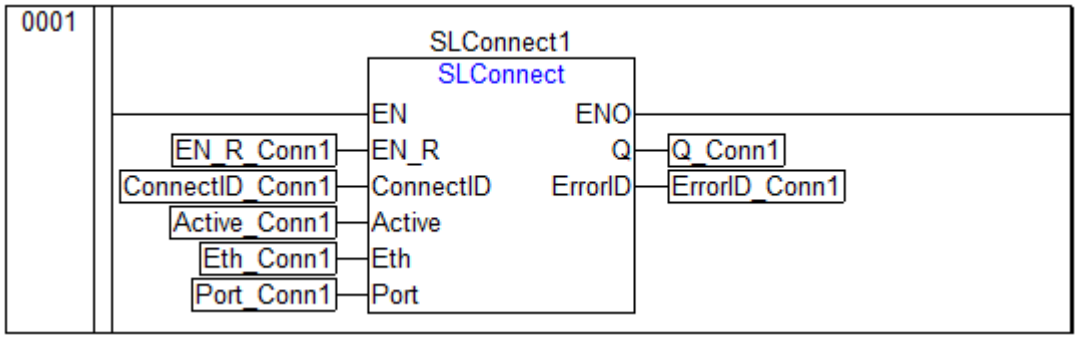
Error code	Description
0	Success.
2	Configuration instance already in progress.
6	Invalid connection ID.
7	Invalid port number.
8	Port number cannot be configured.
9	Server startup failed (e.g. port number already exists).
14	Server already exists.

7.1.29.5 Example

1. Variable Definition

No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	SLConnect1			SLCONNECT		FALSE	Not Public	
0002	EN_R_Conn1			BOOL	FALSE	FALSE	Not Public	
0003	ConnectID_Conn1			BYTE	0	FALSE	Not Public	
0004	Eth_Conn1			BYTE	0	FALSE	Not Public	
0005	Active_Conn1			BOOL	FALSE	FALSE	Not Public	
0006	Port_Conn1			UINT	0	FALSE	Not Public	
0007	Q_Conn1			BOOL	FALSE	FALSE	Not Public	
0008	ErrorID_Conn1			UDINT	0	FALSE	Not Public	

2. LD Configuration Example



3. ST Configuration Example

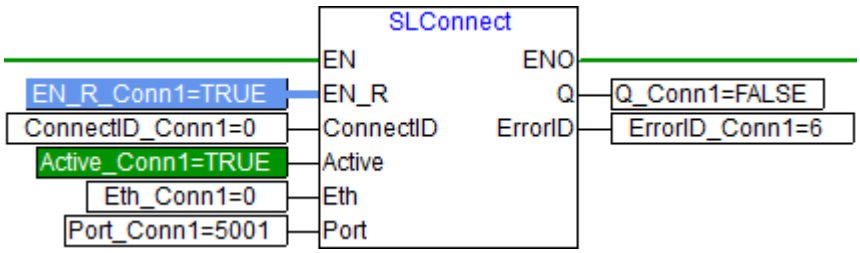
```
SLConnect1(
  EN_R := EN_R_Conn1,
  ConnectID := ConnectID_Conn1,
  Active := Active_Conn1,
  Eth := Eth_Conn1,
  Port := Port_Conn1,
  Q=>Q_Conn1,
  ErrorID=>ErrorID_Conn1);
```

4. The following demonstrates SLConnect parameter configuration, multi-instance protection for the same handle, and connection triggering using an LD program as an example.

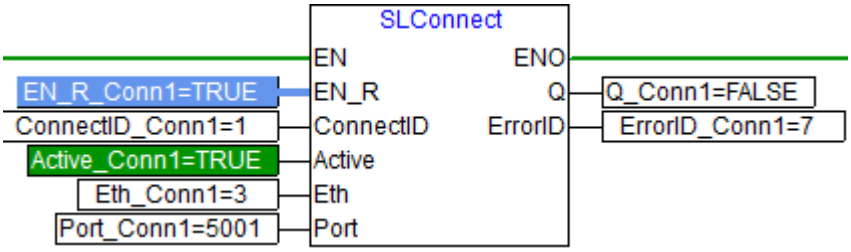
■ Parameter Validation

When the connection function block's EN_R receives a rising edge, it validates input parameters. When Active is FALSE, only ConnectID is checked as follows:

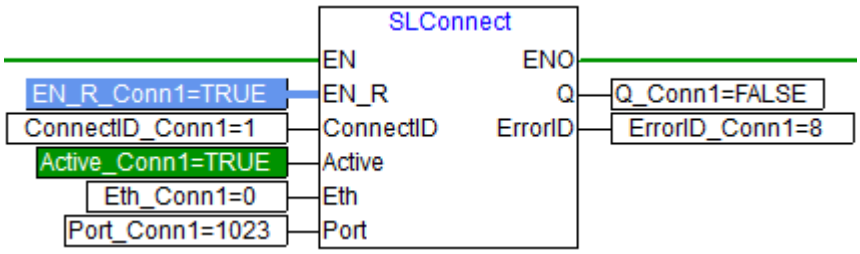
□ ConnectID range: 1~8 (corresponding to 8 connections), out-of-range reports error 6 (Invalid Connection ID)



□ Eth range: 0~1, out-of-range reports error 7 (Invalid Network Port Number)



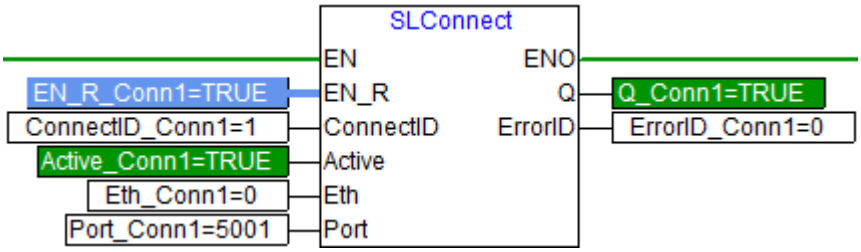
- Port range: 1025~65534 (ports 1200 and 1201 cannot be configured), out-of-range reports error 8 (Unconfigurable Port Number)



■ Connection Triggering

With correct input parameters, a rising edge on EN_R triggers the connection. Output pin states are:

- Connection successful: Q=TRUE, ErrorID=0.



- Connection failed: Q=FALSE, ErrorID=non-zero value.

7.1.30 SLDiag (Get SL protocol server Diagnose Information)

7.1.30.1 Version

■ AutoThink Software Version: V3.2.3SP3 and above.

■ Firmware Version

Controller Type	Firmware Version
LX-CU430	LX-CU430_V1.2.0 and above.
LX-CU433	LX-CU433_V1.1.0 and above.
LX-CU500	LX-CU500_V1.4.0 and above.
LX-CU501	LX-CU501_V1.2.0 and above.

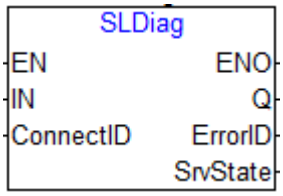
LX-CU510	LX-CU510_V1.1.0 and above.
LX-CU511	LX-CU511_V1.0.1 and above.

7.1.30.2 Function

Get SL protocol server status.

7.1.30.3 Precautions

During the configuration process of the function block, power-off is strictly prohibited, as it may cause the connection to fail.



SLDiag Functional Block

7.1.30.4 Parameter

Parameter	Statement	Data Type	Description	Default Value	Power Fail Safeguard
IN	INPUT	BOOL	Function block enable, high level trigger.	FALSE	FALSE
ConnectID	INPUT	BYTE	Connection ID, range: 1~8 (corresponding to 8 connection channels).	0	FALSE
Q	OUTPUT	BOOL	Completion Status: FALSE: Not completed TRUE: Completed	FALSE	FALSE
SrvState	OUTPUT	SL_SRV_STATE	Server Status: eInit(0): IN disabled display value eStop(1): Server stopped eStart(2): Server started eConnect(3): Server connected	eInit	FALSE
ErrorID	OUTPUT	UDINT	For error code definitions, please refer to the table below.	0	FALSE

7.1.30.5 Error Code

Error code	Description
0	Success.
6	Invalid connection ID.
10	Service reception error.
11	Server transmission error.
12	Remote client actively closed connection.

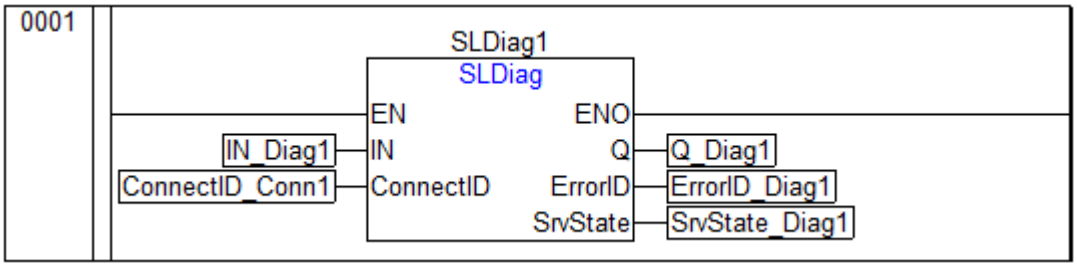
13	Received message length error.

7.1.30.6 Example

1. Variable Definition

No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	SLDiag1			SLDIAG		FALSE	Not Public	
0002	IN_Diag1			BOOL	FALSE	FALSE	Not Public	
0003	ConnectID_Conn1			BYTE	0	FALSE	Not Public	
0004	Q_Diag1			BOOL	FALSE	FALSE	Not Public	
0005	ErrorID_Diag1			UDINT	0	FALSE	Not Public	
0006	SrvState_Diag1			SL_SRV_STATE	eInit	FALSE	Not Public	

2. LD Configuration Example



3. ST Configuration Example

```
SLDiag1(
  IN := IN_Diag1,
  ConnectID := ConnectID_Conn1,
  Q=>Q_Diag1,
  ErrorID=>ErrorID_Diag1,
  SrvState=>SrvState_Diag1);
```

4. The following demonstrates SLDiag parameter verification and diagnostic acquisition example using an LD program as an example.

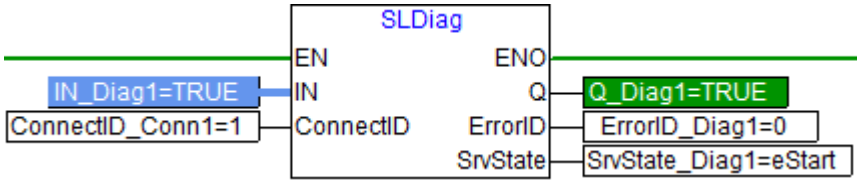
   ■ Parameter Check

   The connection ID range is 1~8, corresponding to 8 channels. If the range is exceeded, error code 6 (Invalid Connection ID) will be reported.

   ■ Diagnostic Acquisition

   Input ConnectID, and when IN is at a high level, diagnostics are reported in real-time with the following two states:

   □ Normal Diagnosis: Q is TRUE, and ErrorID is 0.



- Abnormal Diagnosis: Q is FALSE, and ErrorID is a non-zero value.

7.1.31 LXComCapture (Serial port packet capture)

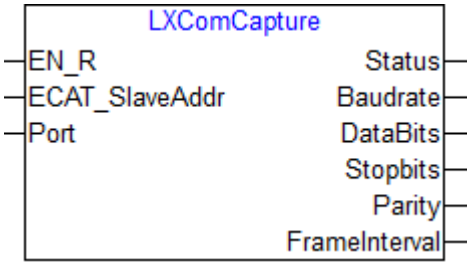
7.1.31.1 Version

- AutoThink software version: V3.2.3SP3 and above.
- Controller firmware versions are as follows:

Controller Model	Firmware Version
LX-CU430	LX-CU430_V1.2.0 and above.
LX-CU433	LX-CU433_V1.1.0 and above.
LX-CU500	LX-CU500_V1.4.0 and above.
LX-CU501	LX-CU501_V1.2.0 and above.
LX-CU510	LX-CU510_V1.1.0 and above.
LX-CU511	LX-CU511_V1.0.1 and above.

7.1.31.2 Function

Obtains the baud rate, data bits, stop bits, parity, and frame interval information for the corresponding channel of the LX-CM series serial port module to be captured.



LXComCapture Function Block

7.1.31.3 Parameters

Parameter	Declaration	Data Type	Description	Default Value
EN_R	INPUT	BOOL	Enable on rising edge.	FALSE
ECAT_SlaveAddr	INPUT	UINT	EtherCAT slave address of the serial port module for capture. Input range 1~65535.	0
Port	INPUT	USINT	Serial port module channel number: 1: Channel 1 (channel1Setting)	0

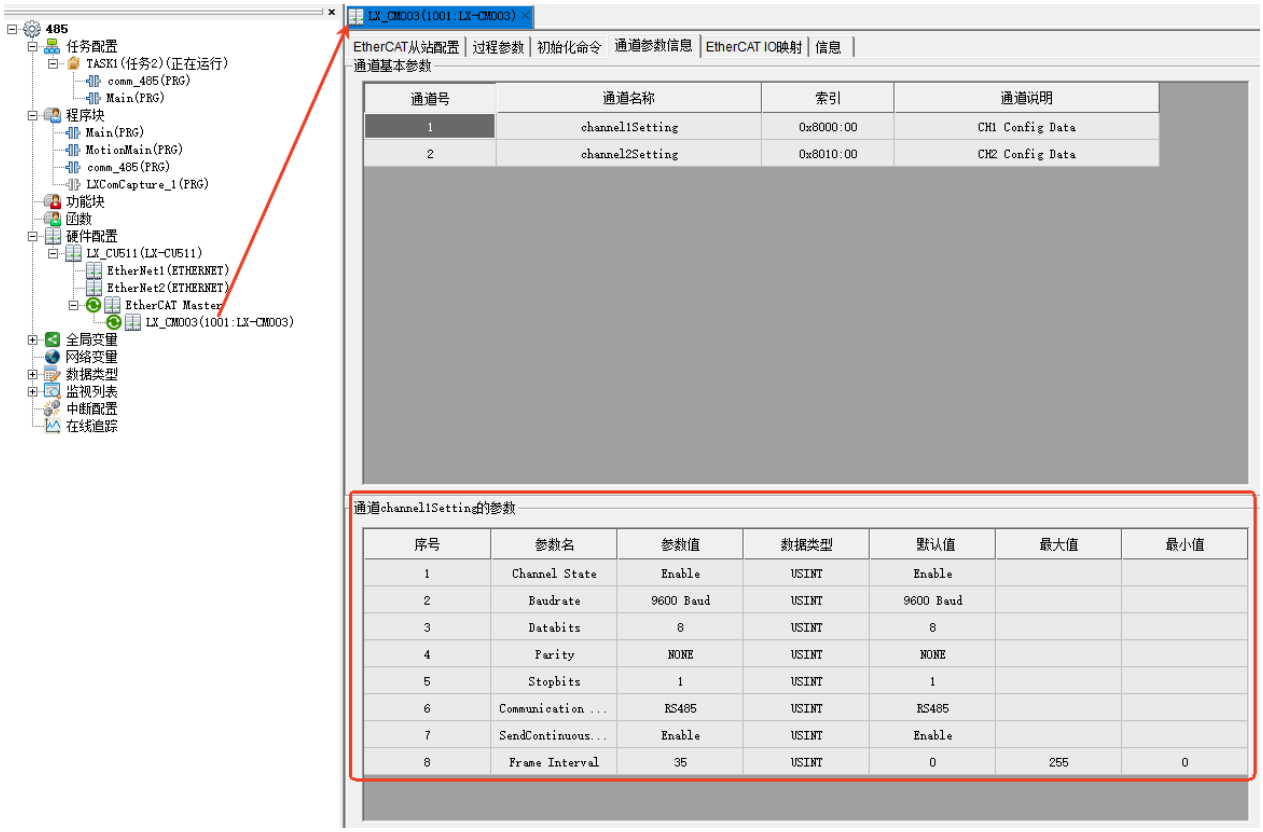
			2: Channel 2 (channel2Setting)	
Status	OUTPUT	USINT	Capture status 0: No error 1: Invalid ECAT_SlaveAddr value input 4: Invalid Port value input 200: Not a HollySys LX series serial port module 201: Module is not a serial port module 202: Channel of the capture module is not enabled 203: Module status abnormal or input slave address is not a configured serial port slave 204: AT capture function already enabled, function block capture is disabled 205: Capture function block stopped 206: AT has stopped function block capture, waiting for AT to start capture 207: Capturing with new parameters 208: Capture function block called repeatedly 209: The serial port parameter pointer for the packet capture function is null	0
Baudrate	OUTPUT	USINT	Baud rate of the serial port module: 1: 1200 2: 2400 3: 4800 4: 9600 5: 19200 6: 38400 7: 57600 8: 115200	0
Databits	OUTPUT	USINT	Data bits of the serial port module: 1: 8 bits 2: 7 bits	0
Parity	OUTPUT	USINT	Parity of the serial port module: 1: None 2: Odd 3: Even	0
Stopbits	OUTPUT	USINT	Stop bits of the serial port module: 1: 1 bit 2: 2 bits	0
FrameInterval	OUTPUT	USINT	Frame interval of the serial port module: 0~255.	0

7.1.31.4 Example

Taking the acquisition of configuration parameters for channel 1 of the LX-CM003 serial port module as an example.

1. Configure Module Channel Parameters

   Double-click the LX_CM003 node in the left configuration tree, and open the "Channel Parameter Information" tab to view the configured parameters for LX-CM003 channel 1, as shown below.

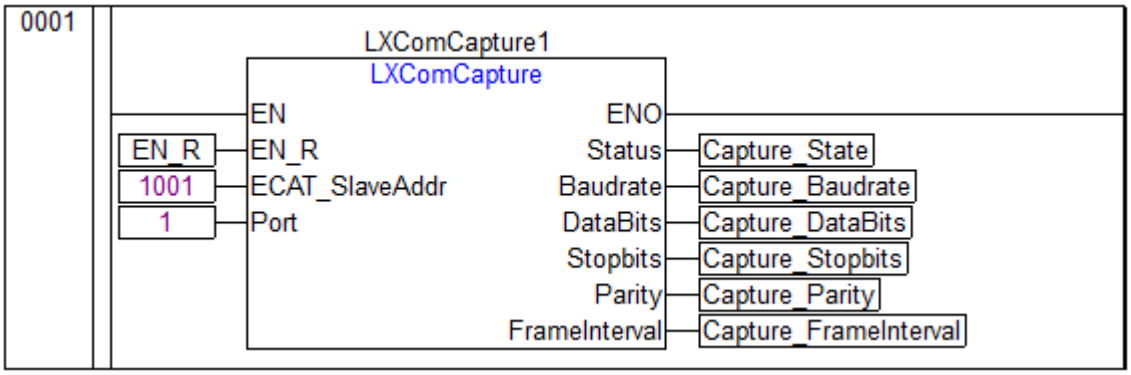


2. Configuration Example

- Variable Declaration

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护	网络公开	是否常量
0001	LXComCapture1			LXCOMCAPTURE		FALSE	不公开	
0002	EN_R			BOOL	FALSE	FALSE	不公开	
0003	Capture_State			USINT	0	FALSE	不公开	
0004	Capture_Baudrate			USINT	0	FALSE	不公开	
0005	Capture_DataBits			USINT	0	FALSE	不公开	
0006	Capture_Stopbits			USINT	0	FALSE	不公开	
0007	Capture_Parity			USINT	0	FALSE	不公开	
0008	Capture_FrameInterval			USINT	0	FALSE	不公开	

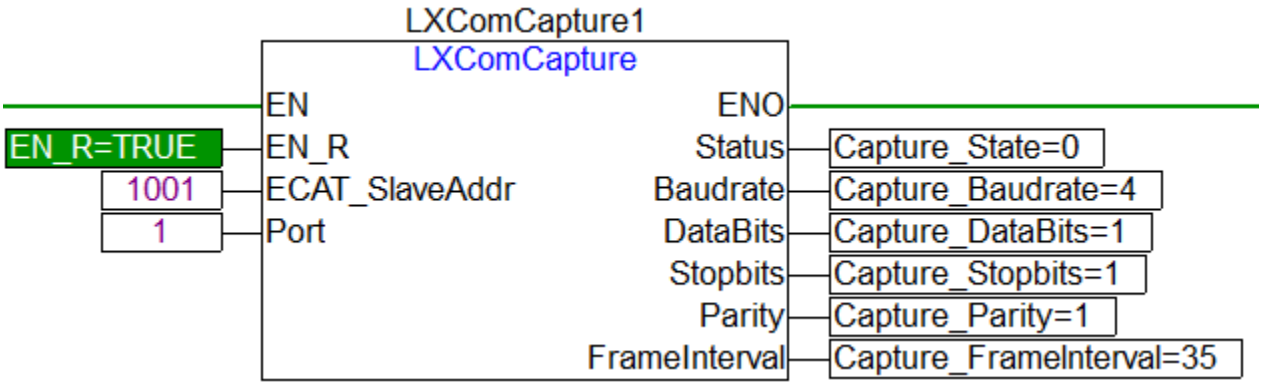
- LD Configuration Example



■ ST Configuration Example

```
LXComCapture1(
    EN_R := EN_R,
    ECAT_SlaveAddr := 1001,
    Port := 1,
    Status=>Capture_State,
    Baudrate=>Capture_Baudrate,
    DataBits=>Capture_DataBits,
    Stopbits=>Capture_Stopbits,
    Parity=>Capture_Parity,
    FrameInterval=>Capture_FrameInterval);
```

3. Taking the LD program as an example, when the function block enable signal EN_R is set from FALSE to TRUE, the baud rate, data bits, stop bits, parity, and frame interval of LX-CM003 channel 1 can be obtained.



7.1.32 LXComMBMaster (MODBUS Master(capture))

7.1.32.1 Version

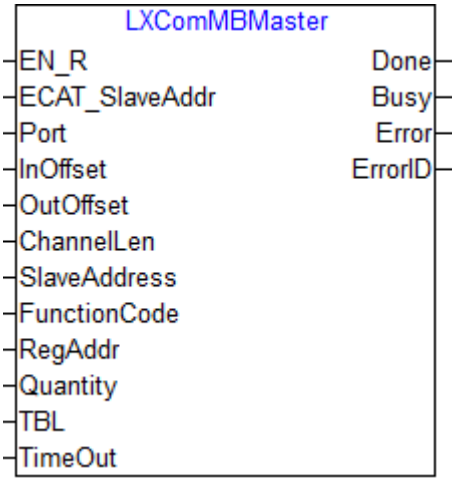
- AutoThink software version: V3.2.3SP3 and above.
- Controller firmware versions are as follows:

Controller Model	Firmware Version
LX-CU430	LX-CU430_V1.2.0 and above.
LX-CU433	LX-CU433_V1.1.0 and above.
LX-CU500	LX-CU500_V1.4.0 and above.
LX-CU501	LX-CU501_V1.2.0 and above.
LX-CU510	LX-CU510_V1.1.0 and above.
LX-CU511	LX-CU511_V1.0.1 and above.

7.1.32.2 Function

When the LX-CM series serial communication module acts as a Modbus RTU master, this function block allows checking the message status and using the "Serial Port Capture" tool to parse received or sent slave messages.

This function block replaces the previous LXComModbusRTUMaster (MODBUS Master).



LXComMBMaster Function Block

7.1.32.3 Precautions

Configure the serial port module channel parameters with Channel State set to Enable, SendContinuous State set to Continuous Send Enable (Enable), and Frame Interval must be configured to at least 35.

7.1.32.4 Parameters

Parameter	Declaration	Data Type	Description	Default Value
EN_R	INPUT	BOOL	Rising edge enable.	FALSE
ECAT_SlaveAddr	INPUT	WORD	EtherCAT slave address of the serial port module for capture. Input range 1~65535.	0
Port	INPUT	BYTE	Serial port module channel number: 1: Channel 1 (channel1Setting) 2: Channel 2 (channel2Setting)	0
InOffset	INPUT	DWORD	Offset address in the EtherCAT cyclic input data I area.	0
OutOffset	INPUT	DWORD	Offset address in the EtherCAT cyclic output data Q area.	0
ChannelLen	INPUT	LXCHANNELLENNUM_E	Amount of data transmitted cyclically via EtherCAT.	LXChannelLenNum_20
SlaveAddress	INPUT	BYTE	Modbus RTU slave address.	0
FunctionCode	INPUT	BYTE	Modbus RTU slave function code.	0
RegAddr	INPUT	WORD	Starting address of the slave register.	0
Quantity	INPUT	WORD	Number of registers.	0
TBL	INPUT	DWORD	Offset address in the M area where the master stores data.	0
TimeOut	INPUT	WORD	Slave response timeout (ms).	0
Done	OUTPUT	BOOL	FALSE: Slave message processing not completed or not started TRUE: Slave message processing completed	FALSE

Busy	OUTPUT	BOOL	FALSE: Idle TRUE: Receiving or processing the slave's response message	FALSE
Error	OUTPUT	BOOL	FALSE: No error TRUE: Error occurred	FALSE
ErrorID	OUTPUT	WORD	Error code: 0: No error Other: Refer to Error Codes	0

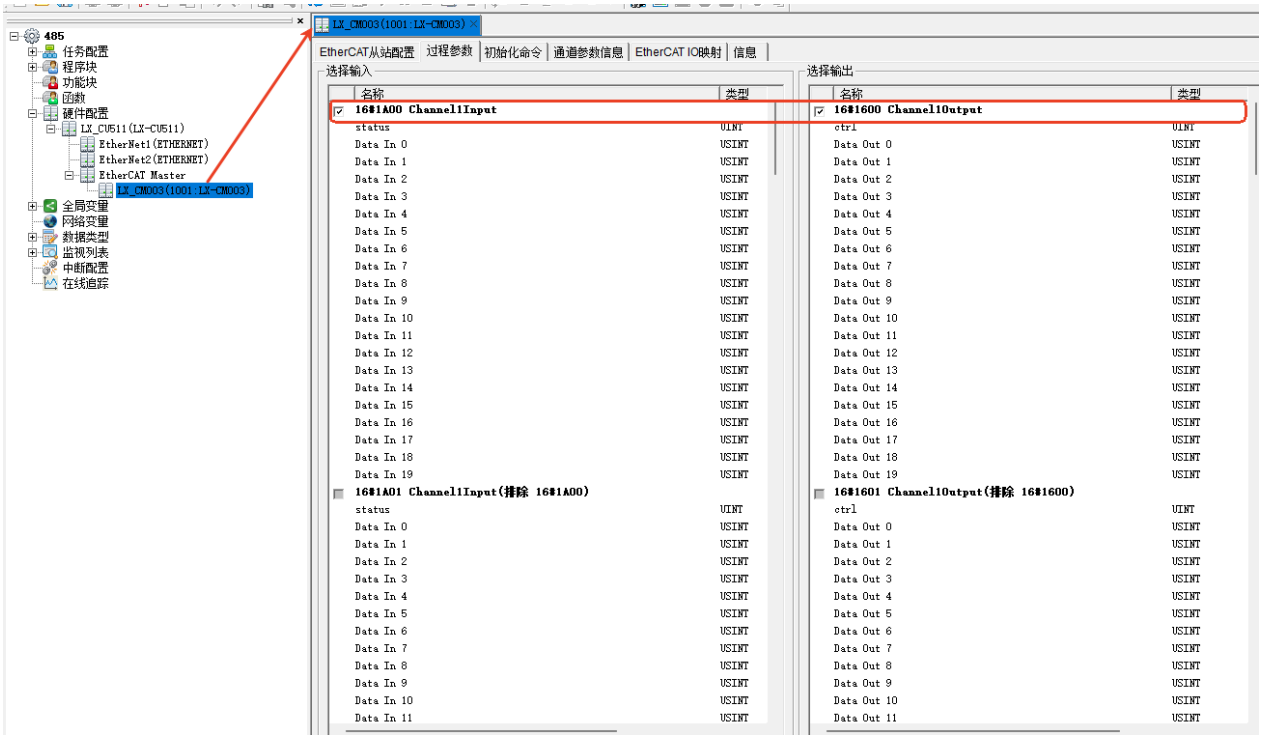
7.1.32.5 Example

This example uses channel 1 of the LX-CM003 serial module configured as a Modbus RTU master. It reads 10 holding registers from address 3000 to 3009 of a slave (LX-CU500) with SlaveID 1 using function code 03. The data is stored in the master PLC at address %MW100. Communication parameters are baud rate 9600, 8 data bits, 1 stop bit, no parity, with 20 bytes of data transmitted per bus cycle.

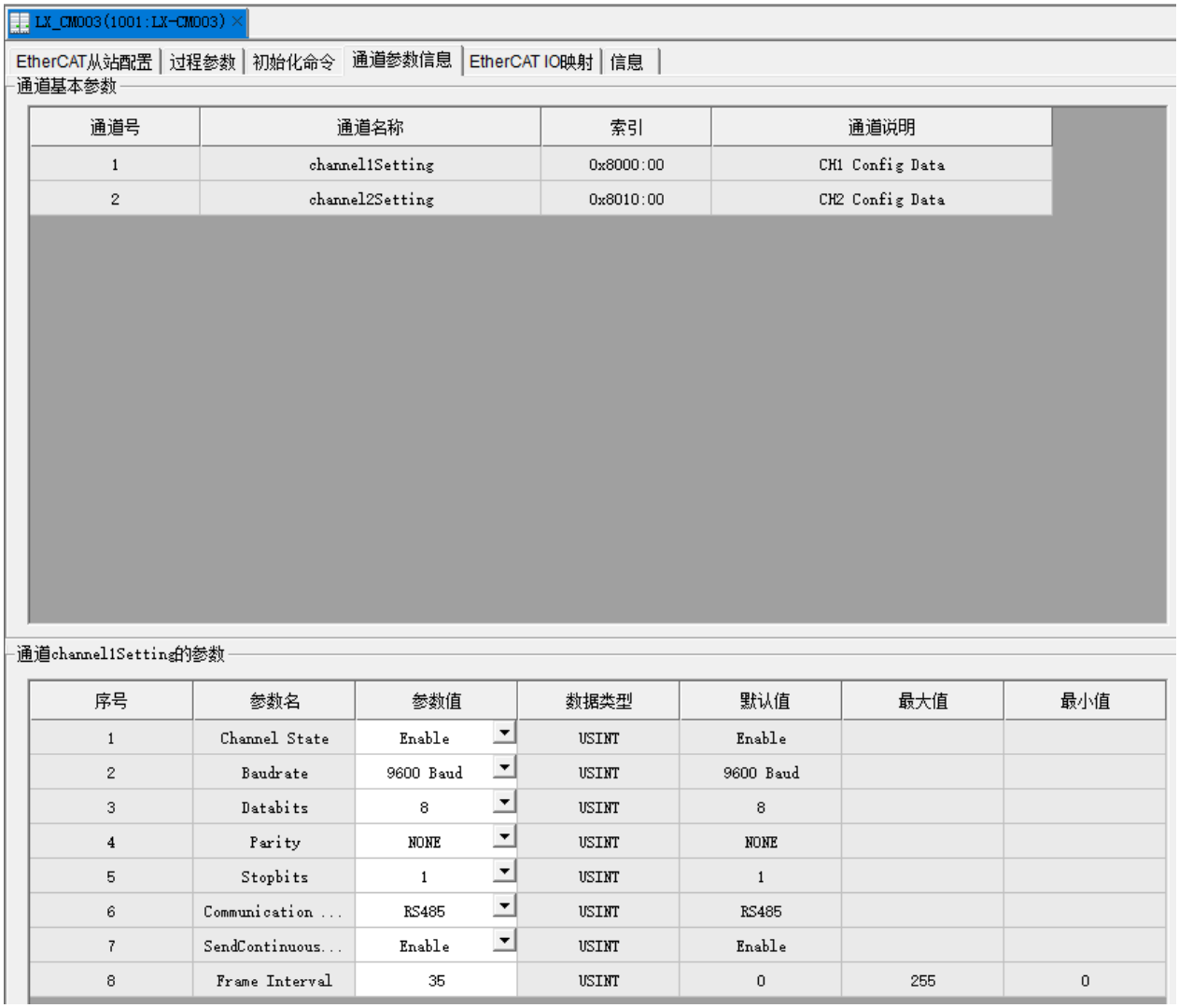
1. Configure Serial Module Parameters

(1) Double-click the corresponding serial module, and check 16#1A00 and 16#1600 in the Process Parameters page.

The cyclic data length can be 20/40/80 bytes, selectable in the module's "Process Parameters" tab.



(2) In the "Channel Parameter Information" tab, click channel1Setting to enter channel 1 parameter configuration. The configuration results are shown below.



(3) In the "EtherCAT IO Mapping" tab, find the first status channel address is 0 (i.e., INOFFSET is 0), and the first ctrl channel address is 0 (i.e., OUTOFFSET is 0).

通道号	通道名称	组名	通道类型	字节数	通道地址	通道说明
1	E6_IN_1		UINT	2	%IW0	status
2	E6_IN_2		USINT	1	%IB2	Data In 0
3	E6_IN_3		USINT	1	%IB3	Data In 1
4	E6_IN_4		USINT	1	%IB4	Data In 2
5	E6_IN_5		USINT	1	%IB5	Data In 3
6	E6_IN_6		USINT	1	%IB6	Data In 4

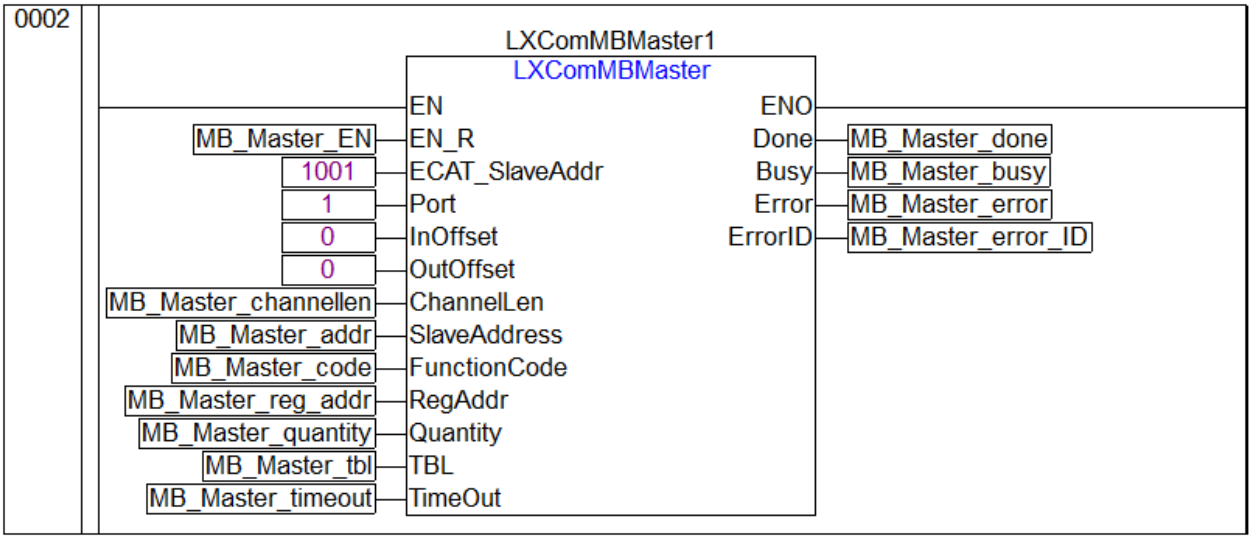
42	E6_IN_42		USINT	1	%IB43	Data In 19
43	E6_OUT_1		UINT	2	%QW0	ctrl
44	E6_OUT_2		USINT	1	%QB2	Data Out 0
45	E6_OUT_3		USINT	1	%QB3	Data Out 1
46	E6_OUT_4		USINT	1	%QB4	Data Out 2
47	E6_OUT_5		USINT	1	%QB5	Data Out 3

2. Configuration Example

- Variable Declaration

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护	网络公开	是否常量
0001	rs485_master_comm	%MW100		ARRAY[0..10] OF WORD		FALSE	不公开	
0002	LXComMBMaster1			LXCOMMBMASTER		FALSE	不公开	
0003	MB_Master_EN			BOOL	FALSE	FALSE	不公开	
0004	MB_Master_channellen			LXCHANNELLENNUM_E	LXChan...	FALSE	不公开	
0005	MB_Master_addr			BYTE	0	FALSE	不公开	
0006	MB_Master_code			BYTE	0	FALSE	不公开	
0007	MB_Master_reg_addr			WORD	3000	FALSE	不公开	
0008	MB_Master_quantity			WORD	0	FALSE	不公开	
0009	MB_Master_tbl			DWORD	0	FALSE	不公开	
0010	MB_Master_timeout			WORD	0	FALSE	不公开	
0011	MB_Master_done			BOOL	FALSE	FALSE	不公开	
0012	MB_Master_busy			BOOL	FALSE	FALSE	不公开	
0013	MB_Master_error			BOOL	FALSE	FALSE	不公开	
0014	MB_Master_error_ID			WORD	0	FALSE	不公开	

- LD Configuration Example



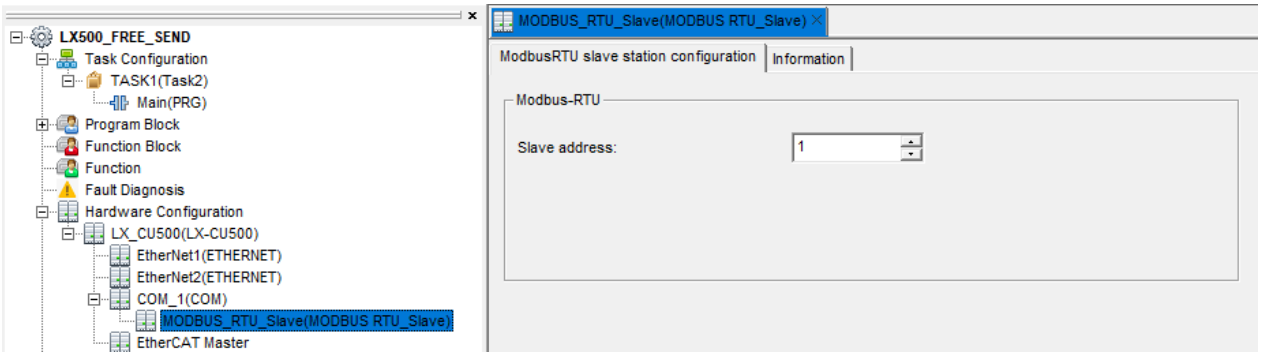
■ ST Configuration Example

```
LXComMBMaster1(
  EN_R := MB_Master_EN,
  ECAT_SlaveAddr := 1001,
  Port := 1,
  InOffset := 0,
  OutOffset := 0,
  ChannelLen := MB_Master_channellen,
  SlaveAddress := MB_Master_addr,
  FunctionCode := MB_Master_code,
  RegAddr := MB_Master_reg_addr,
  Quantity := MB_Master_quantity,
  TBL := MB_Master_tbl,
  TimeOut := MB_Master_timeout,
  Done=>MB_Master_done,
  Busy=>MB_Master_busy,
  Error=>MB_Master_error,
  ErrorID=>MB_Master_error_ID);
```

3. Taking the LD program as an example, after the Modbus master establishes a connection with the slave device (In the example, the LX-CU500 is used as a Modbus RTU slave, and the LX-CM003 mounted on the LX-CU511 serves as the Modbus RTU master.), when the function block enable signal EN_R is set from FALSE to TRUE, the current command is executed. Wait for Done to become TRUE, the data of 10 registers is read and stored in %MW100.

■ **Slave Side Configuration**

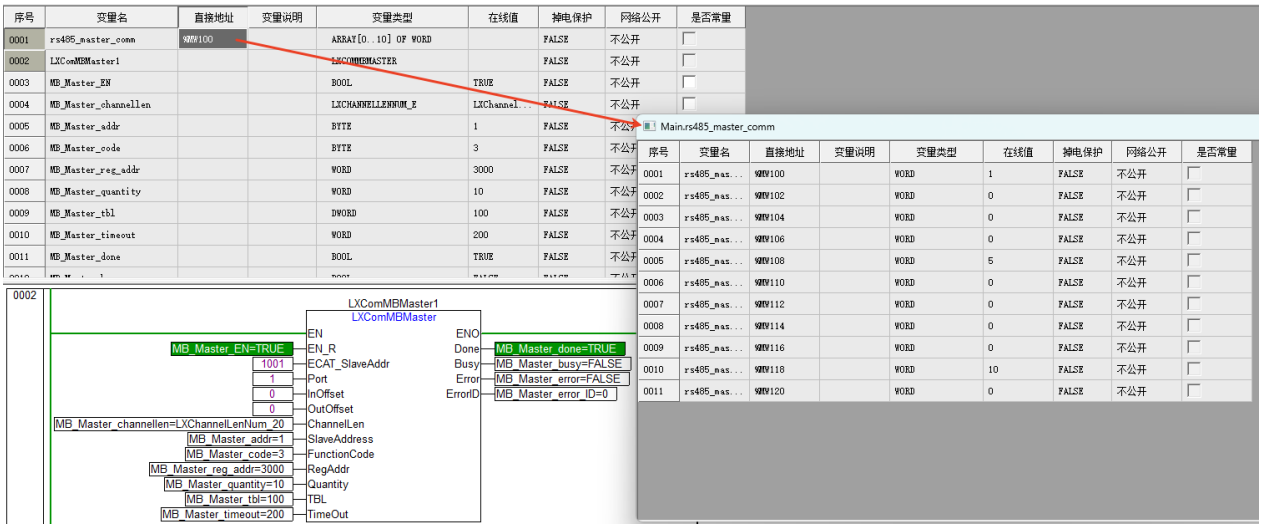
(a) In the slave configuration software, add the serial Modbus RTU slave protocol and configure the slave address.



(b) Double-click "GV_Group" under "Global Variables" to open it, then add a new array variable. Set the Modbus mapping address to %MW0, which will be used as the data transmitted to the master. This is shown in the figure below.

■ **Master Side Configuration:**

After the master-slave communication is established, the master configuration software receives the data transmitted from the slave.



7.1.33 LXComMBSlave (MODBUS Slave(capture))

7.1.33.1 Version

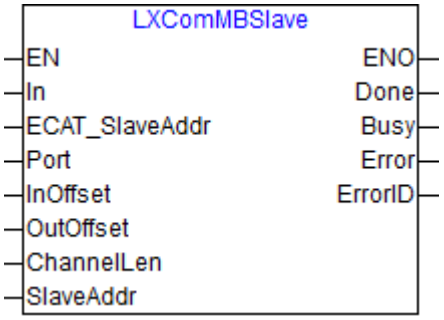
- AutoThink software version: V3.2.3SP3 and above.
- Controller firmware versions are as follows:

Controller Model	Firmware Version
LX-CU430	LX-CU430_V1.2.0 and above.
LX-CU433	LX-CU433_V1.1.0 and above.
LX-CU500	LX-CU500_V1.4.0 and above.
LX-CU501	LX-CU501_V1.2.0 and above.
LX-CU510	LX-CU510_V1.1.0 and above.
LX-CU511	LX-CU511_V1.0.1 and above.

7.1.33.2 Function

When the LX-CM series serial communication module acts as a Modbus RTU slave, this function block allows checking the message status and using the "Serial Port Capture" tool to parse messages.

This function block replaces the previous LXComModbusRTUSlave (MODBUS Slave).



LXComMBSlave Function Block

7.1.33.3 Precautions

Configure the serial port module channel parameters with Channel State set to Enable, SendContinuous State set to Continuous Send (Enable), and Frame Interval must be configured to at least 35.

7.1.33.4 Parameters

Parameter	Declaration	Data Type	Description	Default Value
In	INPUT	BOOL	Enable on high level.	FALSE
ECAT_SlaveAddr	INPUT	WORD	EtherCAT slave address of the serial port module for capture. Input range 1~65535.	0
Port	INPUT	BYTE	Serial port module channel number: 1: Channel 1 (channel1Setting) 2: Channel 2 (channel2Setting)	0
InOffset	INPUT	DWORD	Offset address in the EtherCAT cyclic input data I area.	0
OutOffset	INPUT	DWORD	Offset address in the EtherCAT cyclic output data Q area.	0
ChannelLen	INPUT	LXCHANNELLENNUM_E	Amount of data transmitted cyclically via EtherCAT.	LXChannelLenNum_20
SlaveAddress	INPUT	BYTE	Modbus RTU slave address.	0
Done	OUTPUT	BOOL	FALSE: Slave message processing not completed or not started TRUE: Message processing completed	FALSE
Busy	OUTPUT	BOOL	FALSE: Idle TRUE: Receiving or processing message	FALSE
Error	OUTPUT	BOOL	FALSE: No error TRUE: Error occurred	FALSE
ErrorID	OUTPUT	WORD	Error code: 0: No error Other: Refer to Error Codes	0

7.1.33.5 Example

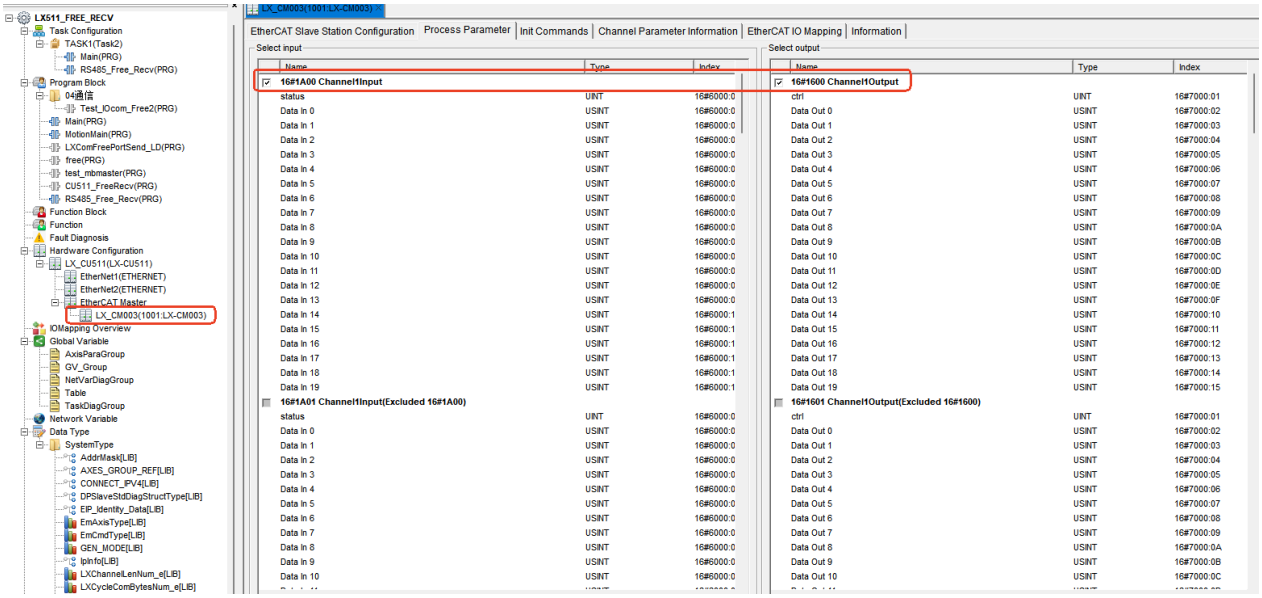
This example uses channel 1 of the LX-CM003 serial module configured as a Modbus RTU slave. The

master reads 10 holding registers from address 400036 to 400045 of this slave (SlaveID configured as 1) using function code 03. Communication parameters are baud rate 9600, 8 data bits, 1 stop bit, no parity, with 20 bytes of data transmitted per bus cycle.

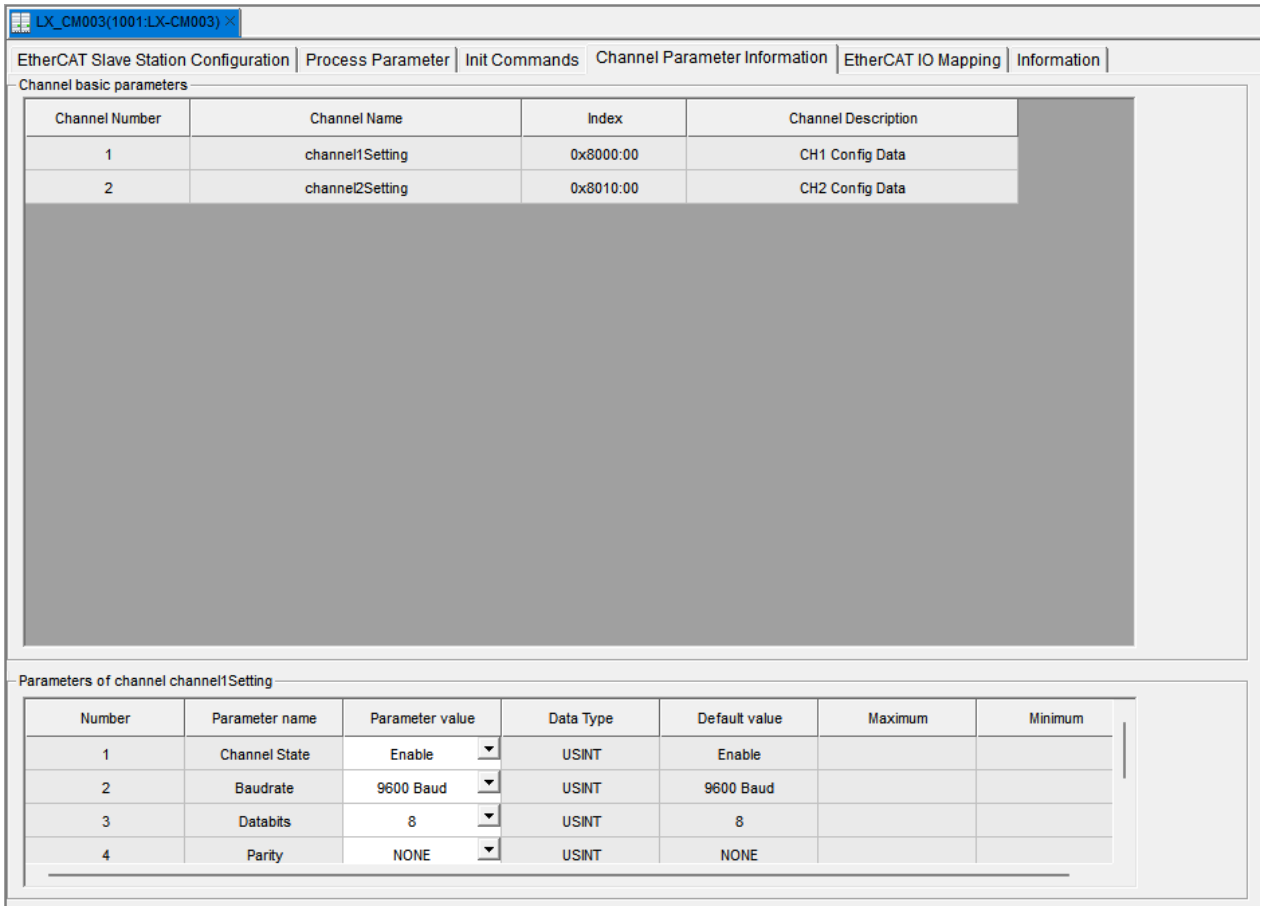
1. Configure Serial Module Parameters

(1) Double-click the corresponding serial module, and check 16#1A00 and 16#1600 in the "Process Parameters" page.

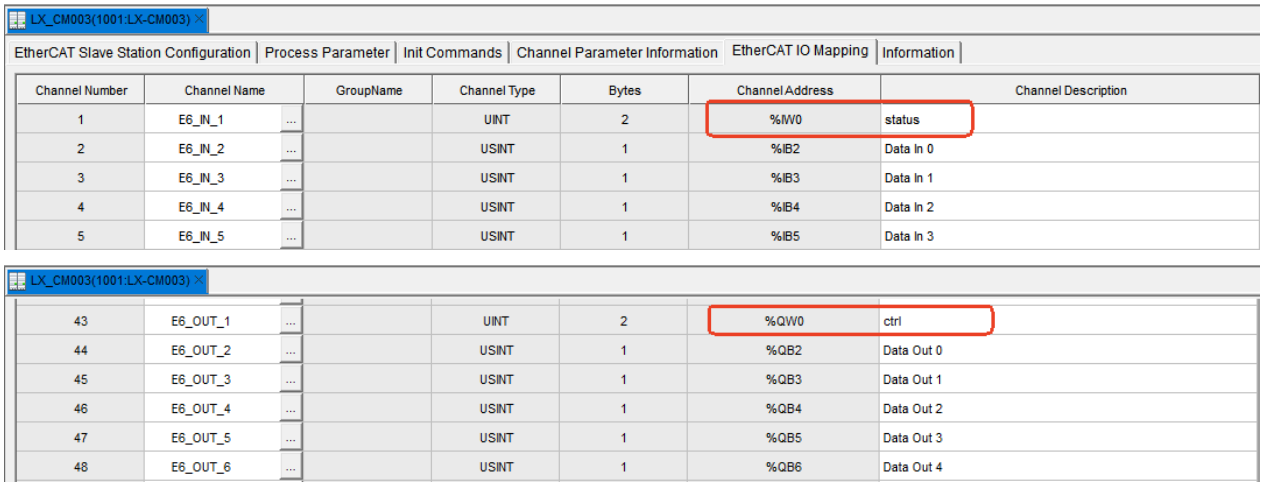
The cyclic data length can be 20/40/80 bytes, selectable in the module's "Process Parameters" tab.



(2) In the "Channel Parameter Information" tab, click channel1Setting to enter channel 1 parameter configuration. The configuration results are shown below.



(3) In the EtherCAT IO Mapping, find the first status channel address is 0 (i.e., INOFFSET is 0), and the first ctrl channel address is 0 (i.e., OUTOFFSET is 0).

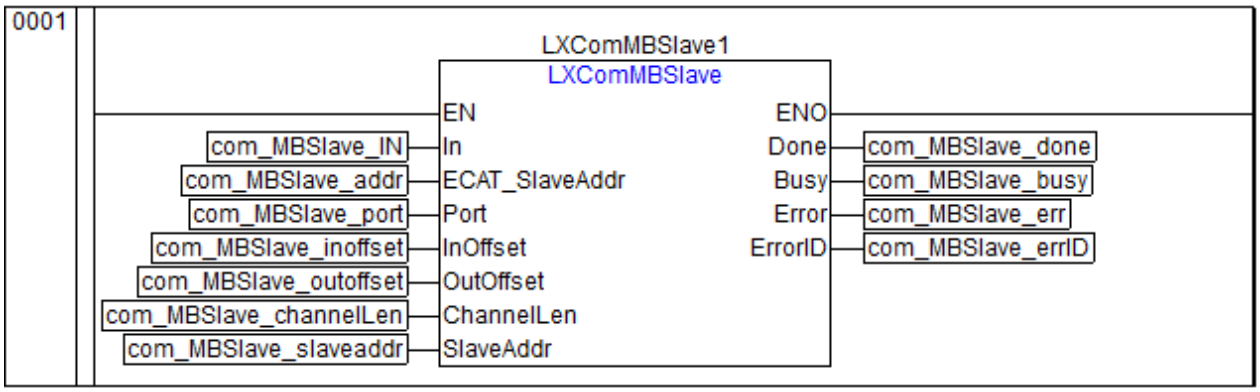


2. Configuration Example

■ Variable Declaration

No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	LXComMBSlave1			LXCOMMBSLAVE		FALSE	Not Public	
0002	com_MBSlave_IN			BOOL	FALSE	FALSE	Not Public	
0003	com_MBSlave_addr			WORD	0	FALSE	Not Public	
0004	com_MBSlave_port			BYTE	0	FALSE	Not Public	
0005	com_MBSlave_inoffset			DWORD	0	FALSE	Not Public	
0006	com_MBSlave_outoffset			DWORD	0	FALSE	Not Public	
0007	com_MBSlave_channelLen			LXCHANNELLENNUM_E	LXChannelLenNum_20	FALSE	Not Public	
0008	com_MBSlave_slaveaddr			BYTE	0	FALSE	Not Public	
0009	com_MBSlave_done			BOOL	FALSE	FALSE	Not Public	
0010	com_MBSlave_busy			BOOL	FALSE	FALSE	Not Public	
0011	com_MBSlave_err			BOOL	FALSE	FALSE	Not Public	
0012	com_MBSlave_errID			WORD	0	FALSE	Not Public	
0013	p1	%QW70		ARRAY[0..9] OF WORD		FALSE	Not Public	

- LD Configuration Example



- ST Configuration Example

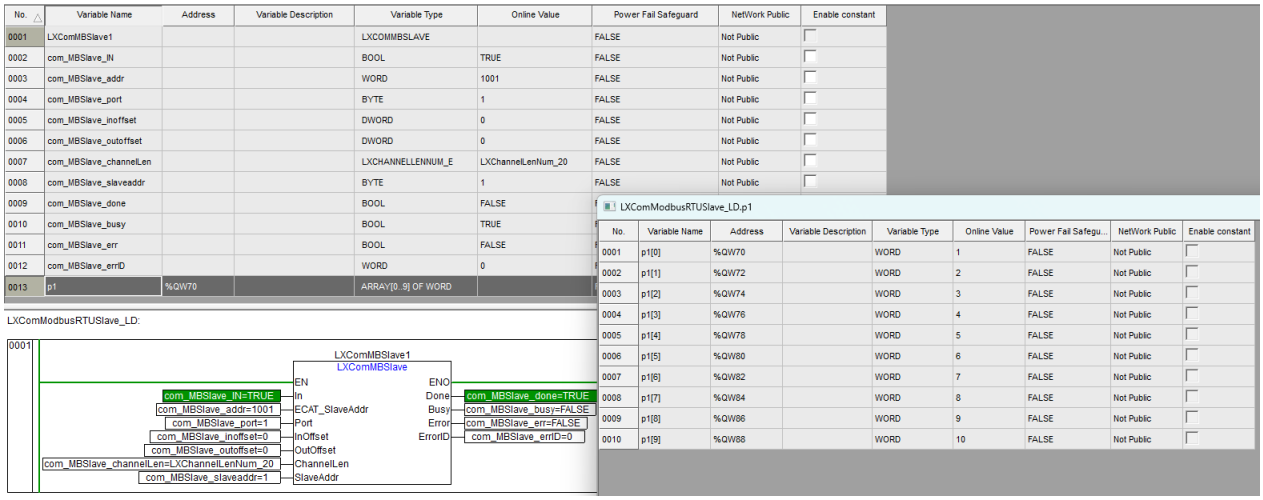
```
LXComMBSlave1(
    EN_R := com_MBSlave_IN,
    ECAT_SlaveAddr := com_MBSlave_addr,
    Port := com_MBSlave_port,
    InOffset := com_MBSlave_inoffset,
    OutOffset := com_MBSlave_outoffset,
    ChannelLen := com_MBSlave_channelLen,
    SlaveAddress := com_MBSlave_slaveaddr,
    Done=>com_MBSlave_done,
    Busy=>com_MBSlave_busy,
    Error=>com_MBSlave_err,
    ErrorID=>com_MBSlave_errID);
```

3. Taking the LD program as an example, after the Modbus slave establishes a connection with the master device (LX-CU500 is used as the Modbus RTU master, and the LX-CM003 mounted on the LX-CU511 is the Modbus RTU slave), when the function block enable signal EN_R is set from FALSE to TRUE,

the current command is executed. The master reads register data from the slave starting at %IW70.

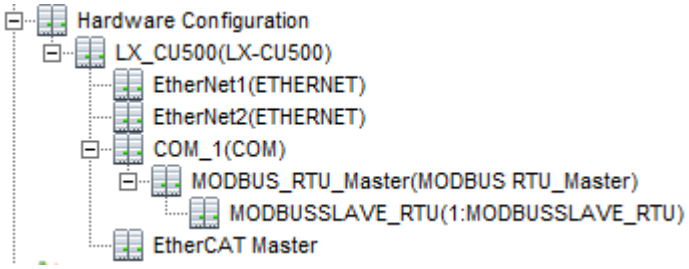
■ **Slave Side Configuration**

In the Modbus RTU slave configuration software, write the data to be sent to address %QW70 and enable the LXComMBSlave function block, as shown below.

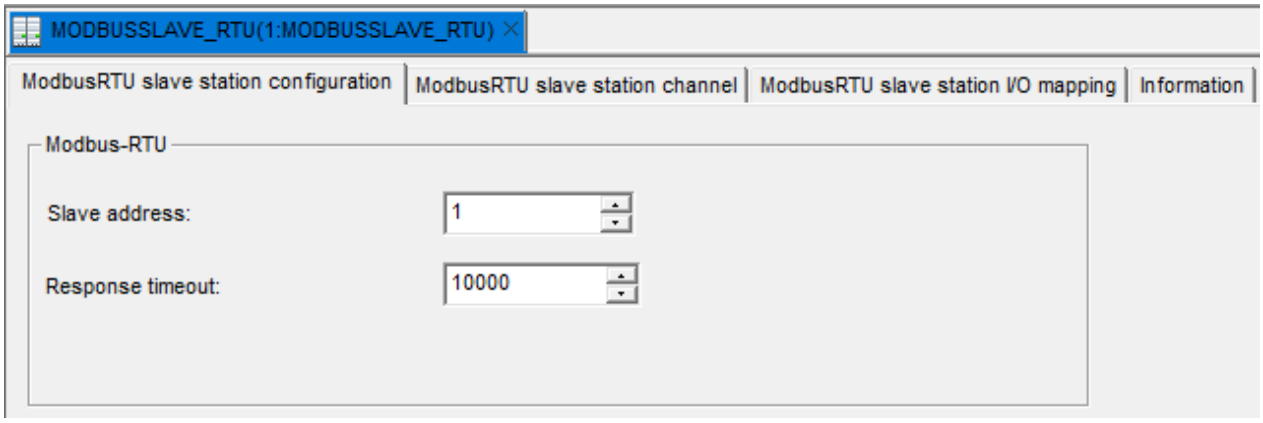


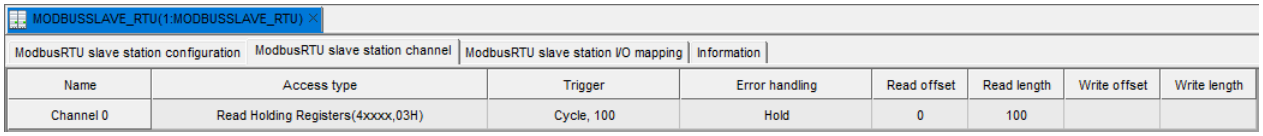
■ **Master Side Configuration**

(a) In the Modbus RTU master configuration software, add the Modbus RTU master protocol and device. As shown below.

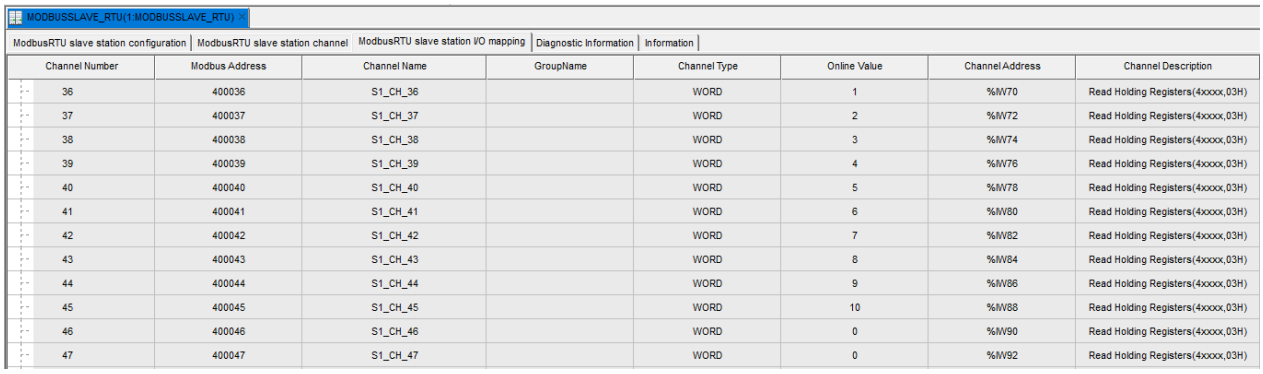


(b) Double-click the "MODBUSSLAVE_RTU" node. In the "Modbus RTU Slave Configuration" tab, configure the slave address and response timeout. In the "Modbus RTU Slave Channel" tab, add a "Read Holding Register" operation.





(c) After the Modbus RTU master project is online, go to the "MODBUSSLAVE_RTU - Modbus RTU Slave I/O Mapping" tab to view the data exchanged between the Modbus master and slave.



7.1.34 LXComFreePortSend (Free Protocol Communication Data Transmission(capture))

7.1.34.1 Version

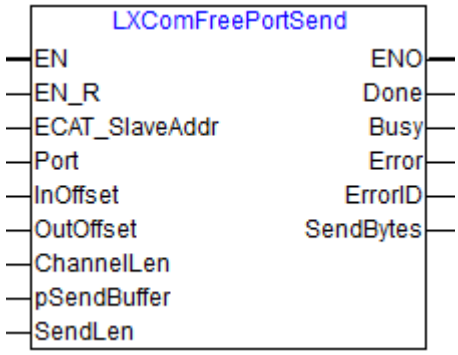
- AutoThink software version: V3.2.3SP3 and above.
- Controller firmware versions are as follows:

Controller Model	Firmware Version
LX-CU430	LX-CU430_V1.2.0 and above.
LX-CU433	LX-CU433_V1.1.0 and above.
LX-CU500	LX-CU500_V1.4.0 and above.
LX-CU501	LX-CU501_V1.2.0 and above.
LX-CU510	LX-CU510_V1.1.0 and above.
LX-CU511	LX-CU511_V1.0.1 and above.

7.1.34.2 Function

This function block, together with the LX-CM series serial communication module, implements the free protocol send functionality.

This function block replaces the previous LXComSend (Free Protocol Communication Data Transmission).



LXComFreePortSend Function Block

7.1.34.3 Precautions

Configure the serial port module channel parameters with Channel State set to Enable, and SendContinuous State set to Continuous Send (Enable).

7.1.34.4 Parameters

Parameter	Declaration	Data Type	Description	Default Value
EN_R	INPUT	BOOL	Rising edge enable.	FALSE
ECAT_SlaveAddr	INPUT	WORD	EtherCAT slave address of the serial port module for capture. Input range 1~65535.	0
Port	INPUT	BYTE	Serial port module channel number: 1: Channel 1 (channel1Setting) 2: Channel 2 (channel2Setting)	0
InOffset	INPUT	DWORD	Offset address in the EtherCAT cyclic input data I area.	0
OutOffset	INPUT	DWORD	Offset address in the EtherCAT cyclic output data Q area.	0
ChannelLen	INPUT	LXCHANNELLENNUM_E	Amount of data transmitted cyclically via EtherCAT.	LXChannelLenNum_20
pSendBuffer	INPUT	POINTER TO BYTE	Send data storage buffer.	0
SendLen	INPUT	WORD	Send data length.	
Done	OUTPUT	BOOL	Send complete flag.	FALSE
Busy	OUTPUT	BOOL	Sending flag.	FALSE
Error	OUTPUT	BOOL	FALSE: No error TRUE: Error occurred	FALSE

ErrorID	OUTPUT	WORD	Error code: 0: No error Other: Refer to Error Codes	0
SendBytes	OUTPUT	WORD	Actual length of data sent.	

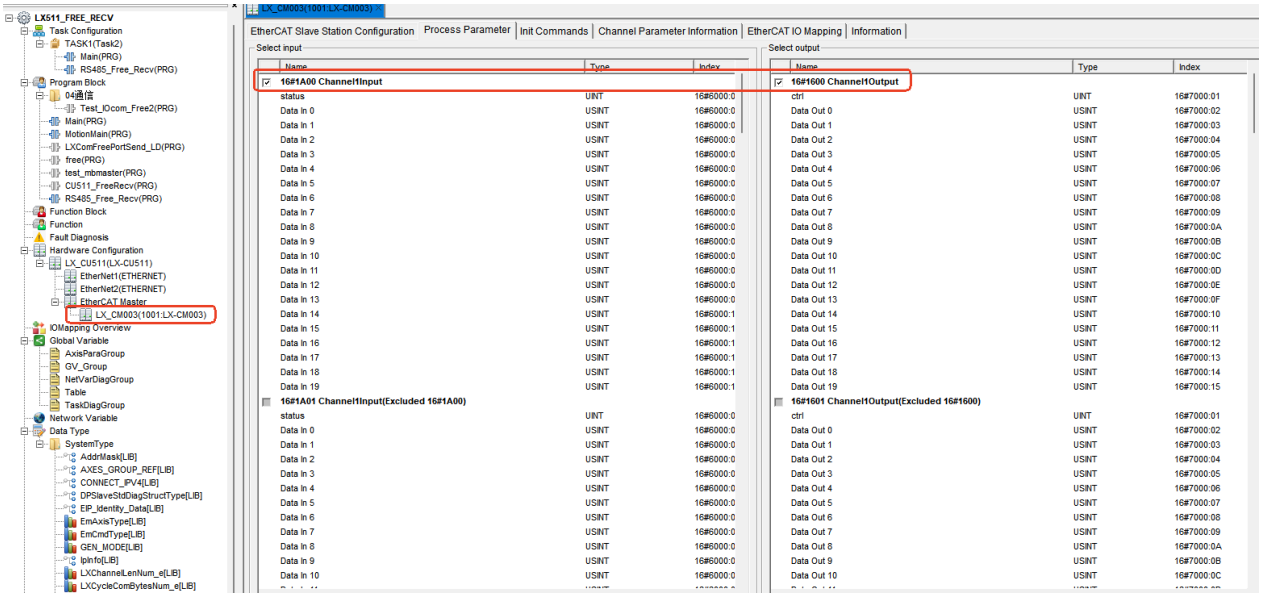
7.1.34.5 Example

This example takes the LXcomFreePortSend function block and uses serial port 1 of the LX-CM003 as the sender in free protocol mode for communication. It sends 10 bytes of data to the peer. Communication parameters are baud rate 9600, 8 data bits, 1 stop bit, no parity, with 10 bytes of data transmitted per bus cycle.

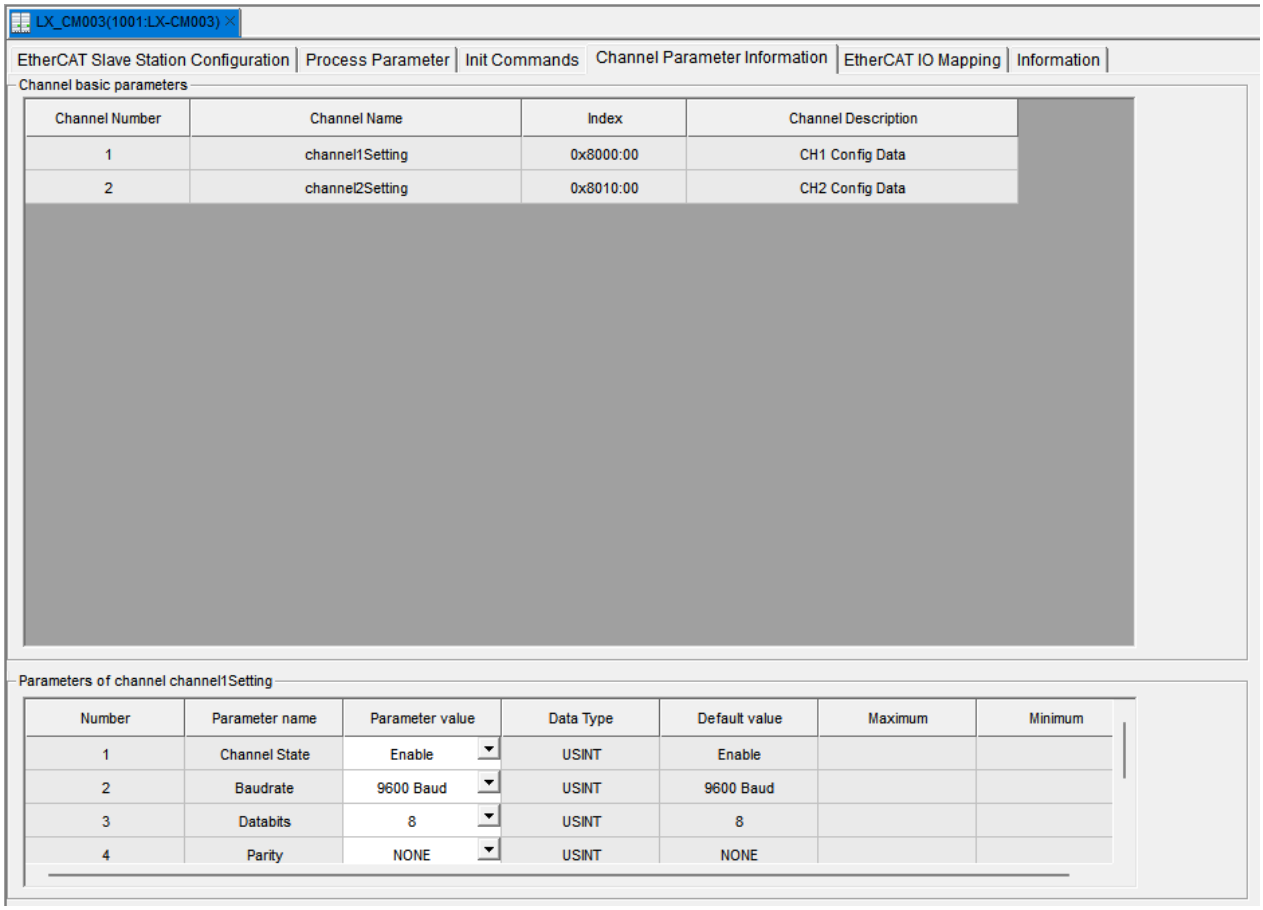
1. Configure Serial Module Parameters

(1) Double-click the corresponding serial module, and check 16#1A00 and 16#1600 in the "Process Parameters" page.

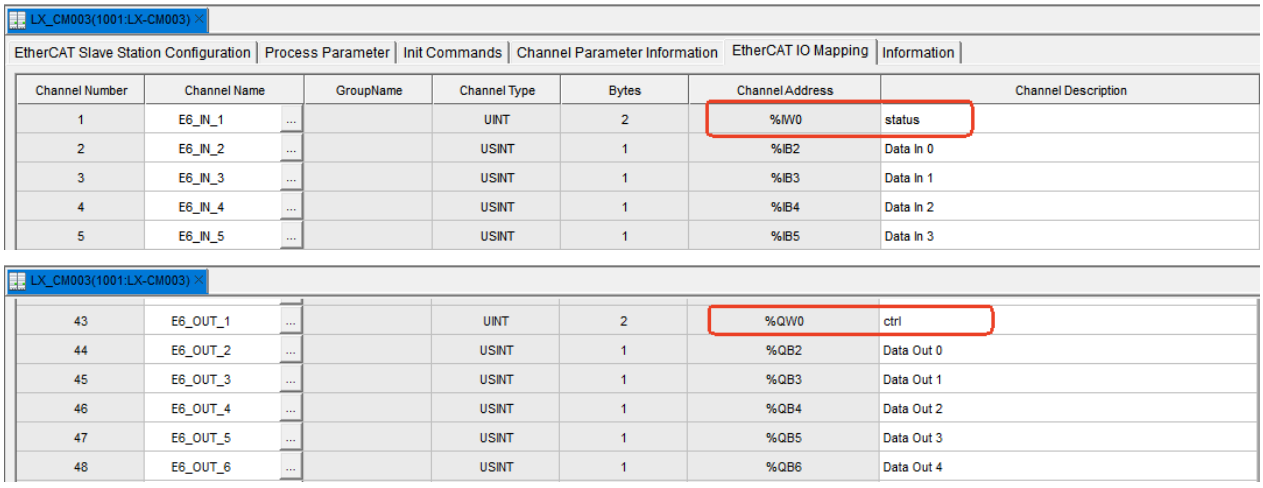
The cyclic data length can be 20/40/80 bytes, selectable in the module's "Process Parameters" tab.



(2) In the "Channel Parameter Information" tab, click channel1Setting to enter channel 1 parameter configuration. The configuration results are shown below.



(3) In the EtherCAT IO Mapping, find the first status channel address is 0 (i.e., INOFFSET is 0), and the first ctrl channel address is 0 (i.e., OUTOFFSET is 0).



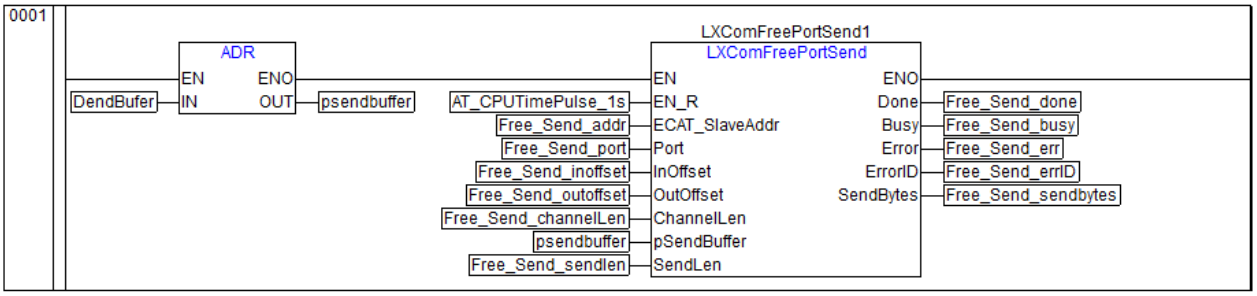
After the function block (acting as the free protocol receiver) establishes a connection with the sending device, set the enable signal EN_R from FALSE to TRUE to execute the current command. Wait for Done to become TRUE, then 10 bytes of data have been send, and RecvBytes will be 10 (the length of data send).

2. Configuration Example

■ Variable Declaration

No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	LXComFreePortSend1			LXCOMFREEPORTSEND		FALSE	Not Public	
0002	DendBufer			ARRAY[0..999] OF BYTE		FALSE	Not Public	
0003	psendbuffer			POINTER TO ARRAY[0..999] OF BYTE	0	FALSE	Not Public	
0004	Free_Send_addr			WORD	0	FALSE	Not Public	
0005	Free_Send_port			BYTE	0	FALSE	Not Public	
0006	Free_Send_inoffset			DWORD	0	FALSE	Not Public	
0007	Free_Send_outoffset			DWORD	0	FALSE	Not Public	
0008	Free_Send_channelLen			LXCHANNELLENNUM_E	LXChannelL...	FALSE	Not Public	
0009	Free_Send_sendlen			WORD	0	FALSE	Not Public	
0010	Free_Send_done			BOOL	FALSE	FALSE	Not Public	
0011	Free_Send_busy			BOOL	FALSE	FALSE	Not Public	
0012	Free_Send_err			BOOL	FALSE	FALSE	Not Public	
0013	Free_Send_errID			WORD	0	FALSE	Not Public	
0014	Free_Send_sendbytes			WORD	0	FALSE	Not Public	

■ LD Configuration Example



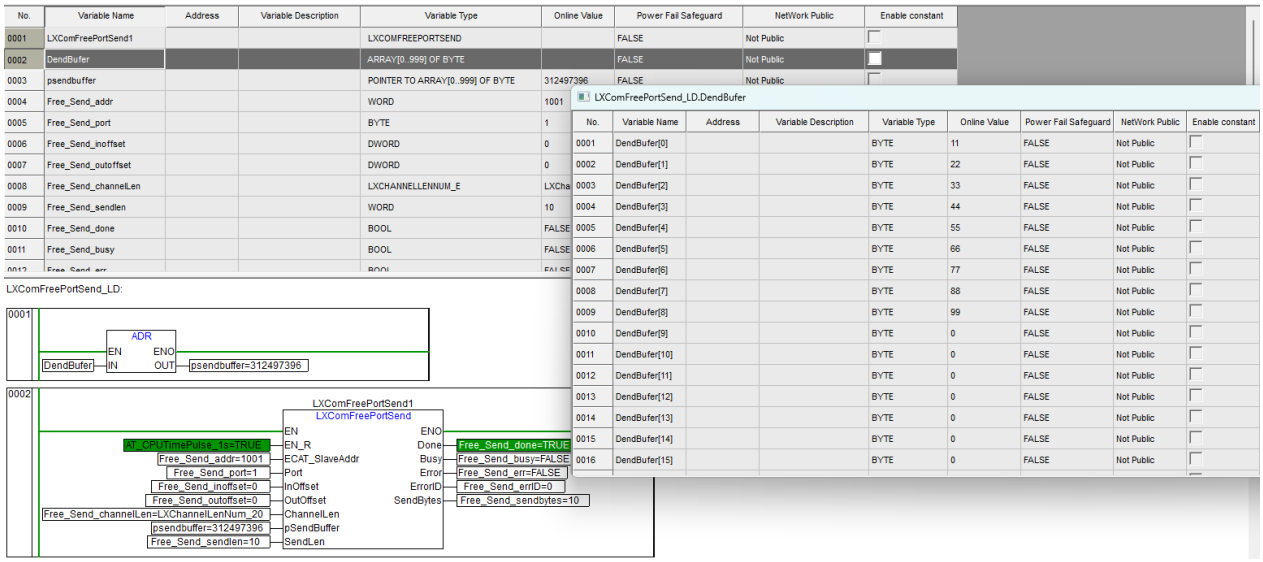
■ ST Configuration Example

```
psendbuffer := ADR(SendBuffer);
LXComFreePortSend1(
    EN_R := AT_CPUTimePulse_1s,
    ECAT_SlaveAddr := Free_Send_addr,
    Port := Free_Send_port,
    InOffset := Free_Send_inoffset,
    OutOffset := Free_Send_outoffset,
    ChannelLen := Free_Send_channelLen,
    pSendBuffer := psendbuffer,
    SendLen := Free_Send_sendlen,
    Done=>Free_Send_done,
    Busy=>Free_Send_busy,
    Error=>Free_Send_err,
    ErrorID=>Free_Send_errID,
    SendBytes=>Free_Send_sendbytes);
```

3. Taking the LD program as an example, after the free protocol sender establishes a connection with the receiver device (In the example, the LX-CU500 is used as the receiver, and the LX-CM003 mounted on the LX-CU511 serves as the transmitter.), when the function block enable signal EN_R is set from FALSE to TRUE, the current command is executed. When Busy is TRUE, the sender is sending data. Wait for Done to become TRUE and SendBytes equals the length of data sent (e.g., 10 bytes).

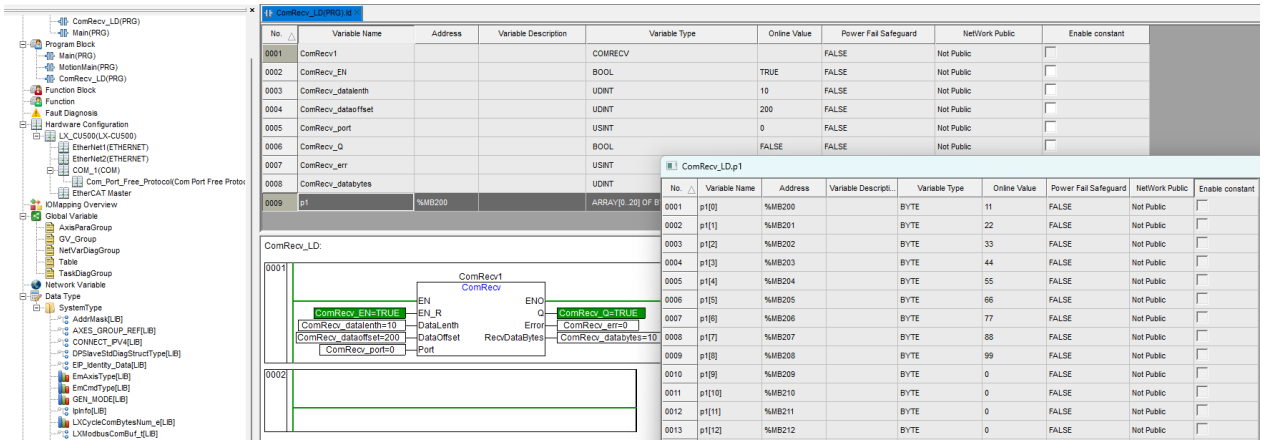
■ Sender Configuration

In the sender configuration software, write the data to be sent and enable the LXComFreePortSend function block, as shown below.



■ Receiver Configuration

In the receiver configuration software, add a free protocol under the COM node in the controller configuration, and create a POU for receiving serial port free protocol data. As shown below.



7.1.35 LXComFreePortReceive (Free Protocol Communication Data Reception(capture))

7.1.35.1 Version

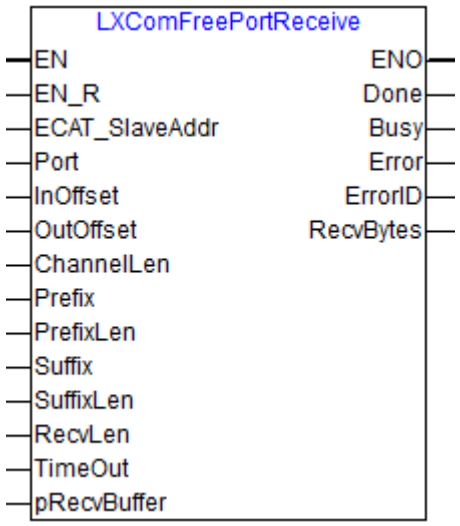
- AutoThink software version: V3.2.3SP3 and above.
- Controller firmware versions are as follows:

Controller Model	Firmware Version
LX-CU430	LX-CU430_V1.2.0 and above.
LX-CU433	LX-CU433_V1.1.0 and above.
LX-CU500	LX-CU500_V1.4.0 and above.
LX-CU501	LX-CU501_V1.2.0 and above.
LX-CU510	LX-CU510_V1.1.0 and above.
LX-CU511	LX-CU511_V1.0.1 and above.

7.1.35.2 Function

This function block, together with the LX-CM series serial communication module, implements the free protocol receive functionality.

This function block replaces the previous LXComReceive (Free Protocol Communication Data Reception).



LXComFreePortReceive Function Block

7.1.35.3 Precautions

Configure the serial port module channel parameters with Channel State set to Enable, and SendContinuous State set to Continuous Send (Enable).

Supported receive modes (timeout is optional for the first five modes):

- Prefix + Suffix + Timeout
- Prefix + Length + Timeout
- Prefix + Timeout
- Suffix + Timeout
- Length + Timeout
- Timeout

7.1.35.4 Parameters

Parameter	Declaration	Data Type	Description	Default Value
EN_R	INPUT	BOOL	Enable on rising edge.	FALSE
ECAT_SlaveAddr	INPUT	WORD	EtherCAT slave address of the serial port module for capture. Input range 1~65535.	0
Port	INPUT	BYTE	Serial port module channel number: 1: Channel 1 (channel1Setting) 2: Channel 2 (channel2Setting)	0
InOffset	INPUT	DWORD	Offset address in the EtherCAT cyclic input data I area.	0
OutOffset	INPUT	DWORD	Offset address in the EtherCAT cyclic output data Q area.	0

ChannelLen	INPUT	LXCHANNELLENNUM_E	Amount of data transmitted cyclically via EtherCAT.	LXChannelLenNum_20
Prefix	INPUT	POINTER TO BYTE	Pointer to prefix..	0
PrefixLen	INPUT	BYTE	Prefix length in bytes.	0
Suffix	INPUT	POINTER TO BYTE	Pointer to suffix..	0
SuffixLen	INPUT	BYTE	Suffix length in bytes.	0
RecvLen	INPUT	DWORD	Specifies the length of data to receive.	0
TimeOut	INPUT	DWORD	Specifies the timeout period (ms).	0
pRecvBuffer	INPUT	POINTER TO BYTE	Pointer to the receive data buffer.	0
Done	OUTPUT	BOOL	FALSE: Slave message processing not completed or not started TRUE: Message processing completed	FALSE
Busy	OUTPUT	BOOL	FALSE: Idle TRUE: Receiving or processing message	FALSE
Error	OUTPUT	BOOL	FALSE: No error TRUE: Error occurred	FALSE
ErrorID	OUTPUT	WORD	Error code: 0: No error Other: Refer to Error Codes	0
RecvBytes	OUTPUT	WORD	Actual length of data received.	0

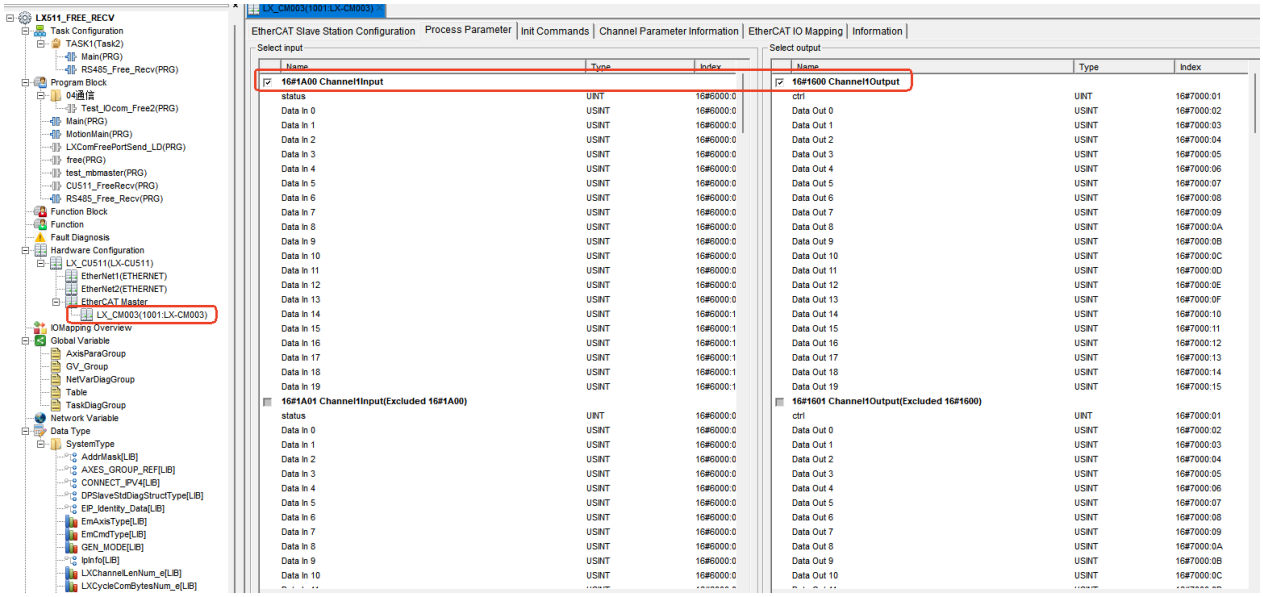
7.1.35.5 Example

This example takes the LXcomFreePortSend function block and uses serial port 1 of the LX-CM003 as the reception in free protocol mode for communication. It is configured in length mode and receives 10 bytes of data from the sender. Communication parameters are baud rate 9600, 8 data bits, 1 stop bit, no parity, with 20 bytes of data transmitted per bus cycle.

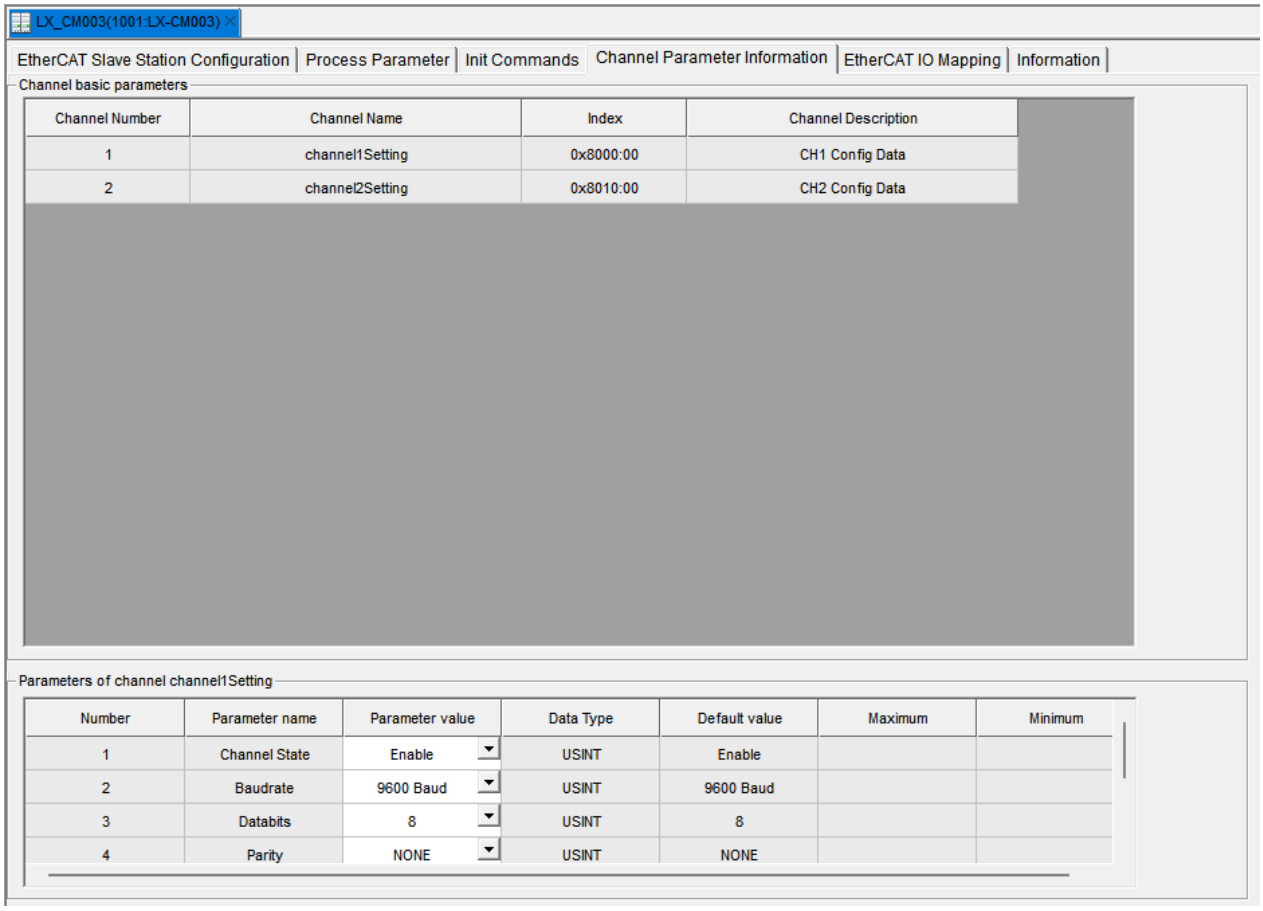
1. Configure Serial Module Parameters

   (1) Double-click the corresponding serial module, and check 16#1A00 and 16#1600 in the "Process Parameters" page.

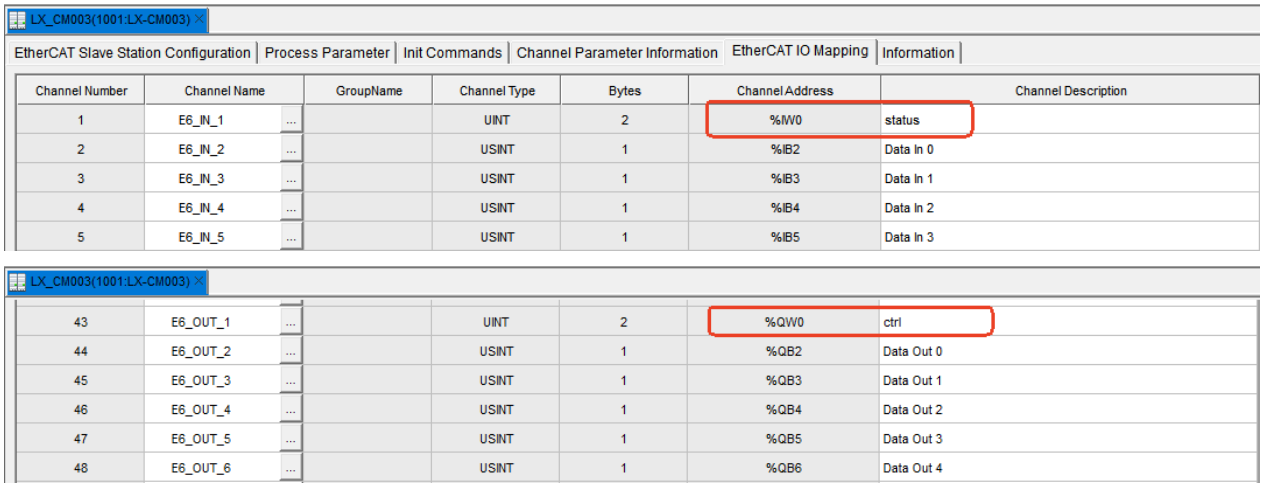
   The cyclic data length can be 20/40/80 bytes, selectable in the module's "Process Parameters" tab.



(2) In the "Channel Parameter Information" tab, click channel1Setting to enter channel 1 parameter configuration. The configuration results are shown below.



(3) In the EtherCAT IO Mapping, find the first status channel address is 0 (i.e., INOFFSET is 0), and the first ctrl channel address is 0 (i.e., OUTOFFSET is 0).



LX_CM003(1001:LX-CM003)

EtherCAT Slave Station Configuration | Process Parameter | Init Commands | Channel Parameter Information | EtherCAT IO Mapping | Information

Channel Number	Channel Name	GroupName	Channel Type	Bytes	Channel Address	Channel Description
1	E6_IN_1		UINT	2	%IW0	status
2	E6_IN_2		USINT	1	%IB2	Data In 0
3	E6_IN_3		USINT	1	%IB3	Data In 1
4	E6_IN_4		USINT	1	%IB4	Data In 2
5	E6_IN_5		USINT	1	%IB5	Data In 3

LX_CM003(1001:LX-CM003)

43	E6_OUT_1		UINT	2	%QW0	ctrl
44	E6_OUT_2		USINT	1	%QB2	Data Out 0
45	E6_OUT_3		USINT	1	%QB3	Data Out 1
46	E6_OUT_4		USINT	1	%QB4	Data Out 2
47	E6_OUT_5		USINT	1	%QB5	Data Out 3
48	E6_OUT_6		USINT	1	%QB6	Data Out 4

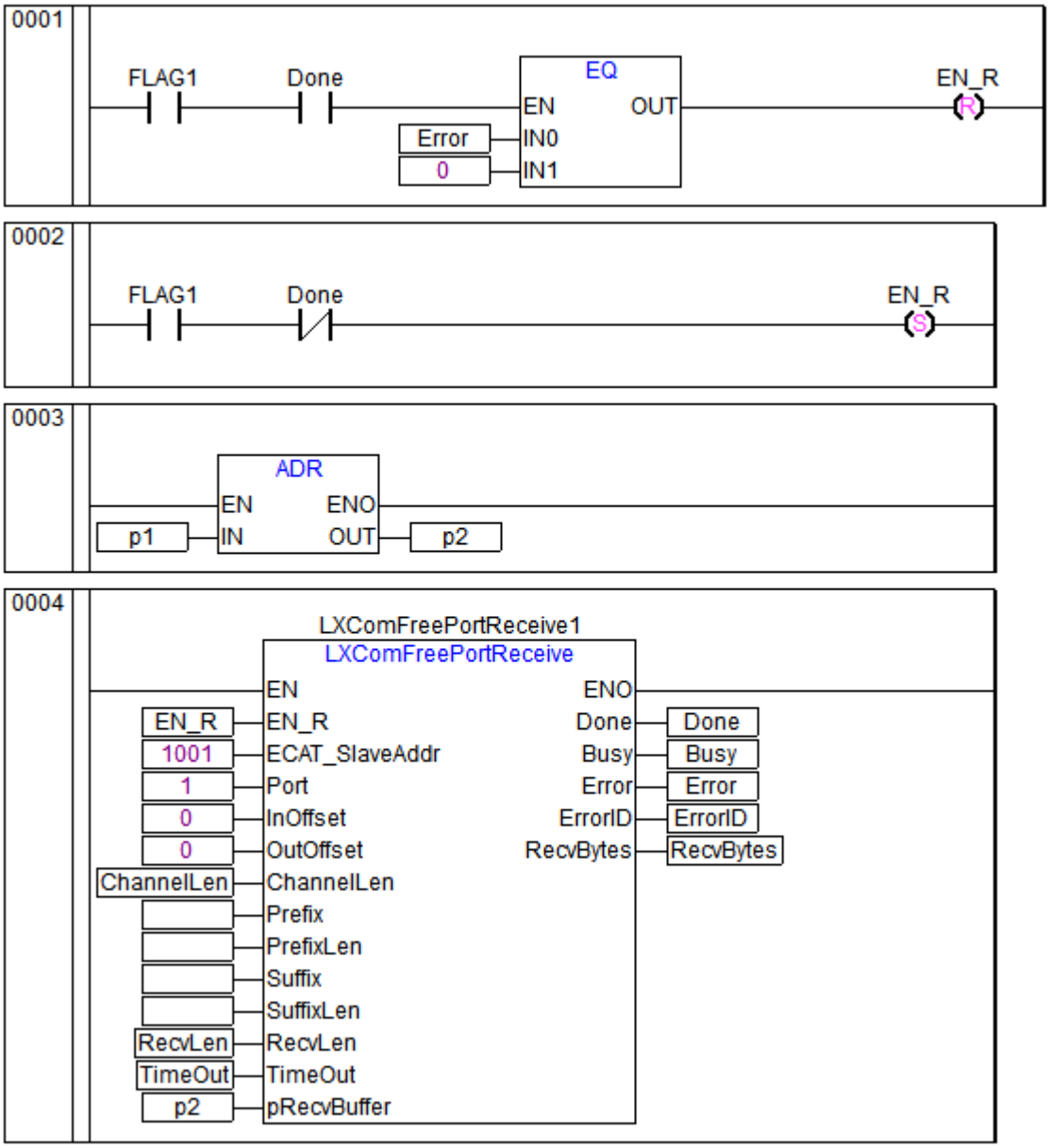
After the function block (acting as the free protocol receiver) establishes a connection with the sending device, set the enable signal EN_R from FALSE to TRUE to execute the current command. Wait for Done to become TRUE, then 10 bytes of data have been received, and RecvBytes will be 10 (the length of data received).

2. Configuration Example

- Variable Declaration

No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
0010	p1			ARRAY[0..1000] OF BYTE		FALSE	Not Public	
0012	FLAG1			BOOL	FALSE	FALSE	Not Public	
0013	EN_R			BOOL	FALSE	FALSE	Not Public	
0007	Error			BOOL	FALSE	FALSE	Not Public	
0005	Done			BOOL	FALSE	FALSE	Not Public	
0006	Busy			BOOL	FALSE	FALSE	Not Public	
0004	TimeOut			DWORD	1000	FALSE	Not Public	
0003	RecvLen			DWORD	0	FALSE	Not Public	
0002	ChannelLen			LXCHANNELLENNUM_E	LXChannelL...	FALSE	Not Public	
0001	LXComFreePortReceive1			LXCOMFREEPORTRECEIVE		FALSE	Not Public	
0011	p2			POINTER TO ARRAY[0..1000] OF BYTE	0	FALSE	Not Public	
0009	RecvBytes			WORD	0	FALSE	Not Public	
0008	ErrorID			WORD	0	FALSE	Not Public	

- LD Configuration Example



■ ST Configuration Example

```
p2 := ADR(p1);
LXComFreePortReceive1(
   EN_R :=EN_R ,
   ECAT_SlaveAddr :=1001 ,
   Port :=1 ,
   InOffset :=0 ,
   OutOffset :=0 ,
   ChannelLen :=ChannelLen ,
```

RecvLen :=RecvLen ,

TimeOut :=TimeOut ,

pRecvBuffer :=p2 ,

Done=>Done,

Busy=>Busy,

Error=>Error,

ErrorID=>ErrorID,

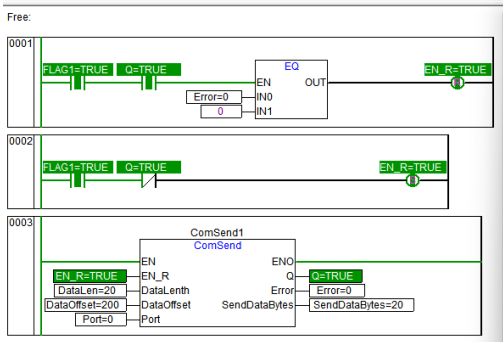
RecvBytes=>RecvBytes);

3. Taking the LD program as an example, after the free protocol receiver establishes a connection with the sender device (In the example, the LX-CU500 is used as the transmitter, and the LX-CM003 mounted on the LX-CU511 serves as the receiver.), when the function block enable signal EN_R is set from FALSE to TRUE, the current command is executed. When Busy is TRUE, the receiver is receiving data. Wait for Done to become TRUE and RecvBytes equals the length of data written to be received; 10 bytes of data are then read.

■ **Sender Configuration**

In the slave configuration software, add a free protocol under the COM node in the controller configuration, and create a POU for sending serial port free protocol data. As shown below.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0004	ComSend1			COMSEND		FALSE	Not Public	
0005	EN_R			BOOL	FALSE	FALSE	Not Public	
0006	DataLen			UDINT	20	FALSE	Not Public	
0007	DataOffset			UDINT	200	FALSE	Not Public	
0008	Port			USINT	0	FALSE	Not Public	
0009	Q			BOOL	FALSE	FALSE	Not Public	
0010	Error			USINT	0	FALSE	Not Public	
0011	SendDataBytes			UDINT	0	FALSE	Not Public	
0012	p1	%MB200		ARRAY[0..1000] OF BYTE		FALSE	Not Public	
0013	FLAG1			BOOL	TRUE	FALSE	Not Public	

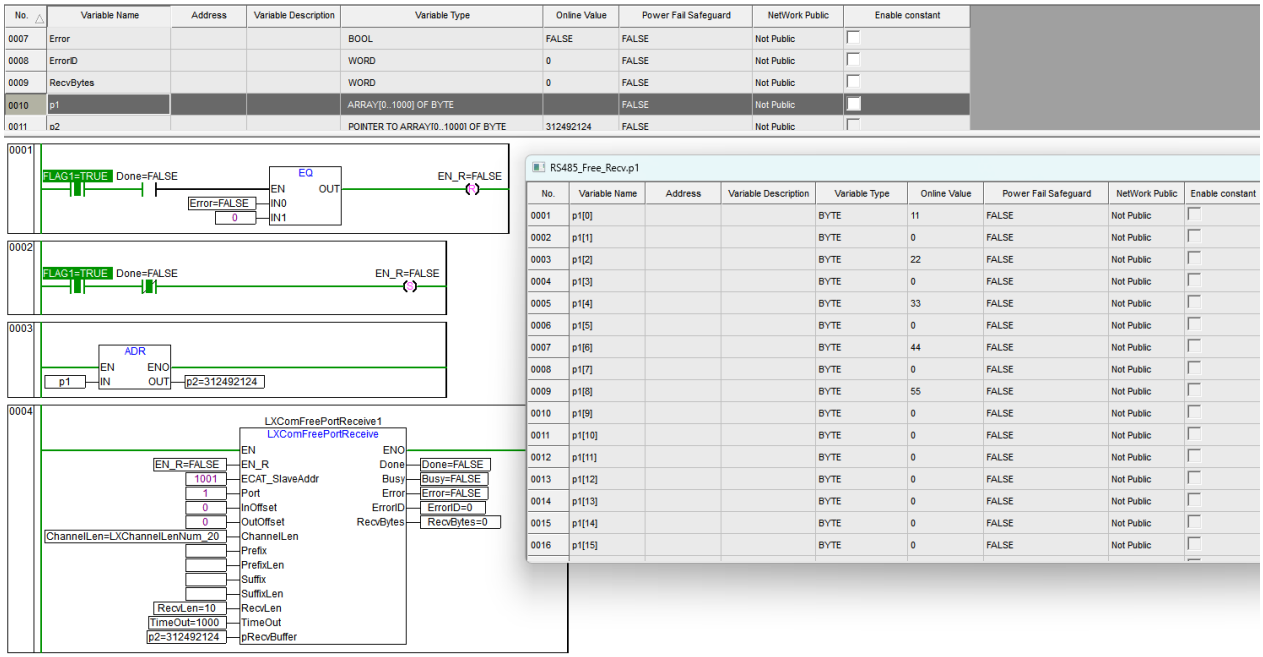


Free.p1

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	p1[0]	%MB200		BYTE	11	FALSE	Not Public	
0002	p1[1]	%MB201		BYTE	0	FALSE	Not Public	
0003	p1[2]	%MB202		BYTE	22	FALSE	Not Public	
0004	p1[3]	%MB203		BYTE	0	FALSE	Not Public	
0005	p1[4]	%MB204		BYTE	33	FALSE	Not Public	
0006	p1[5]	%MB205		BYTE	0	FALSE	Not Public	
0007	p1[6]	%MB206		BYTE	44	FALSE	Not Public	
0008	p1[7]	%MB207		BYTE	0	FALSE	Not Public	
0009	p1[8]	%MB208		BYTE	55	FALSE	Not Public	
0010	p1[9]	%MB209		BYTE	0	FALSE	Not Public	
0011	p1[10]	%MB210		BYTE	0	FALSE	Not Public	
0012	p1[11]	%MB211		BYTE	0	FALSE	Not Public	
0013	p1[12]	%MB212		BYTE	0	FALSE	Not Public	

■ **Receiver Configuration**

After the communication between the sender and the receiver is established, the receiver configuration software receives the data transmitted from the sender.

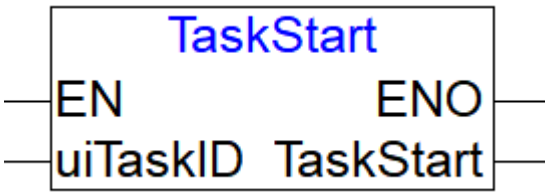


7.2 Task Operation Instruction

7.2.1 TaskStart (Task Status)

7.2.1.1 Function

Run the specified task.



TaskStart Functional Block

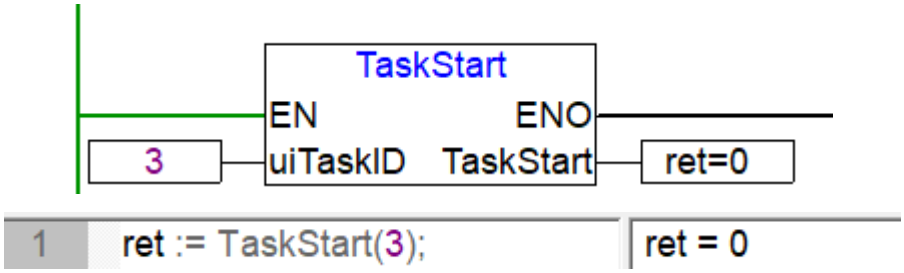
7.2.1.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
uiTaskID	Input	UDINT	Task ID number	0	FALSE
TaskStart	output	DINT	0: returned if succeeded -1: returned if failed	0	FALSE

7.2.1.3 Example

When the conditions meet the requirements, the TaskStart (running task) statement is executed, and the set task ID program is started. The output value of TaskStart is 0 if it is successfully started, and -1 if failed.

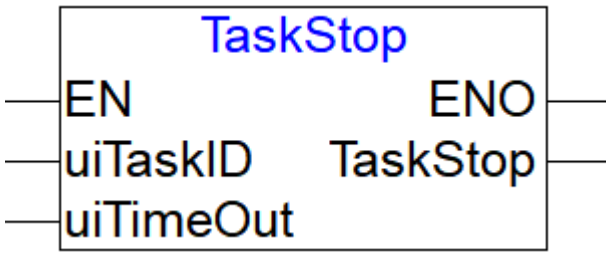
No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	ret			DINT	0	FALSE	Not Public	



7.2.2 TaskStop (Task Stop)

7.2.2.1 Function

Stop the specified task.



TaskStop Functional Block

7.2.2.2 Parameter

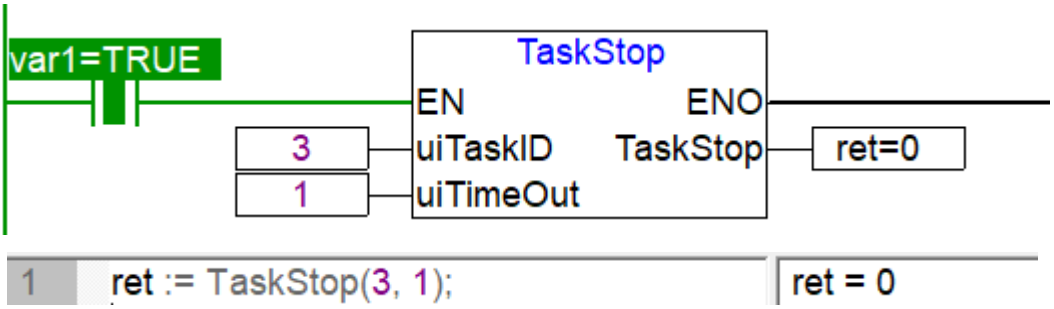
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
uiTaskID	Input	UDINT	Task ID number	0	FALSE
uiTimeout	Input	UDINT	Stop task timeout in ms	0	FALSE
TaskStop	output	DINT	0 returned if succeeded; -1 returned if failed	0	FALSE

7.2.2.3 Example

When the conditions meet the requirements, the TaskStop (stopping task) statement is executed. The task

to stop shall be set for its ID and timeout time (ms). If the task time is exceeded, the task will be terminated. The output value of TaskStart is 0 if the task is successfully stopped, and -1 if failed.

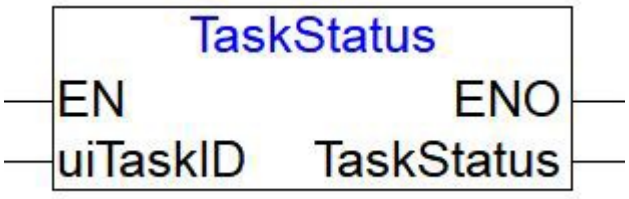
No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	ret			DINT	0	FALSE	Not Public	
0002	var1			BOOL	TRUE	FALSE	Not Public	



7.2.3 TaskStatus (Return Task Status)

7.2.3.1 Function

Return to the current status of the task.



TaskStatus Functional Block

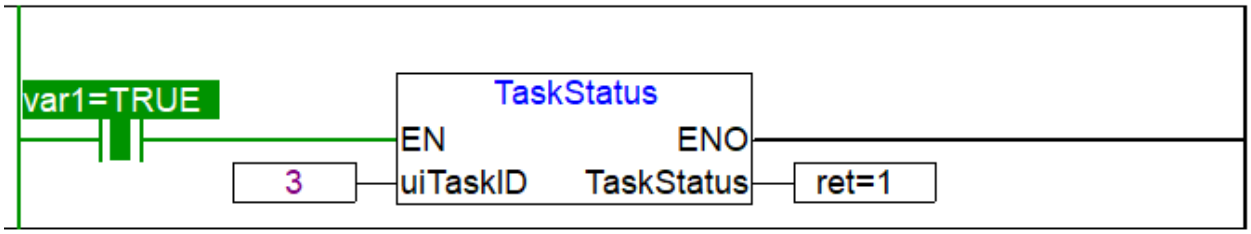
7.2.3.2 Parameter

Parameter	Statement	Data Type	Description	Default Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
uiTaskID	Input	UDINT	Task ID number	0	FALSE
TaskStatus	output	DINT	Task status: Running 1 Suspended 2 Stopped 5 Failed: -1	0	FALSE

7.2.3.3 Example

The current task status can be viewed through the TaskStatus (returning the task status) instruction. The output values are 1 for Running, 2 for Suspended, 3 for Stopped, and -1 for Failed.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	var1			BOOL	TRUE	FALSE	Not Public	
0002	ret			DINT	1	FALSE	Not Public	

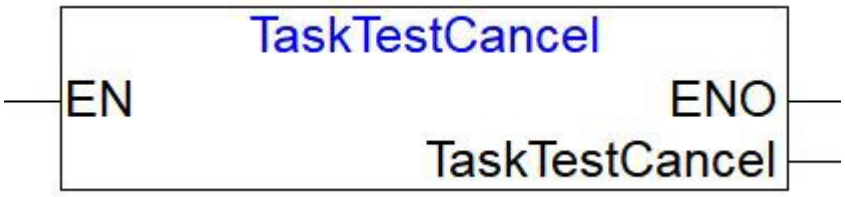



```
6   ret := TaskStatus(3);          ret = 1
```

7.2.4 TaskTestCancel (Set Task Cancel Point)

7.2.4.1 Function

Set the task cancellation point, which is similar to the thread cancellation point function in the operating system.



TaskTestCancel Functional Block

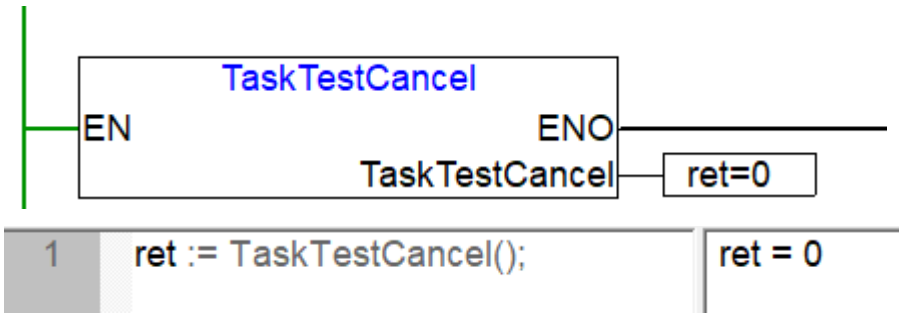
7.2.4.2 Parameter

Parameter	Statement	Data Type	Description	Default Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
TaskTestCancel	Output	DINT	0 returned if succeeded; -1 returned if failed.	0	FALSE

7.2.4.3 Example

Set the task cancellation point, which is similar to the thread cancellation point function in the operating system. The user can call this interface to set the cancellation point in the configuration where they believe it is safe to exit. When other tasks stop this task, this task will not exit immediately. Instead, it will continue to execute to the cancellation point and then exit the task thread, to ensure the safe exit of the task to the greatest extent.

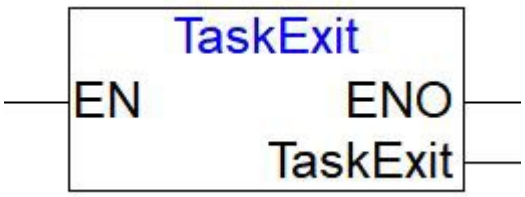
No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	ret			DINT	0	FALSE	Not Public	



7.2.5 TaskExit (Task Exit)

7.2.5.1 Function

The task exit instruction provides an interface for users to exit threads in configuration logic. Users can call this function in task configuration to end the execution of this task.



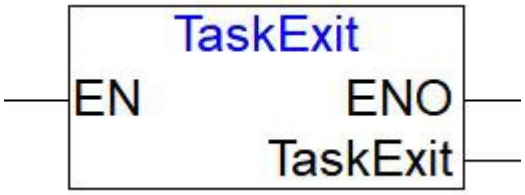
TaskExit Functional Block

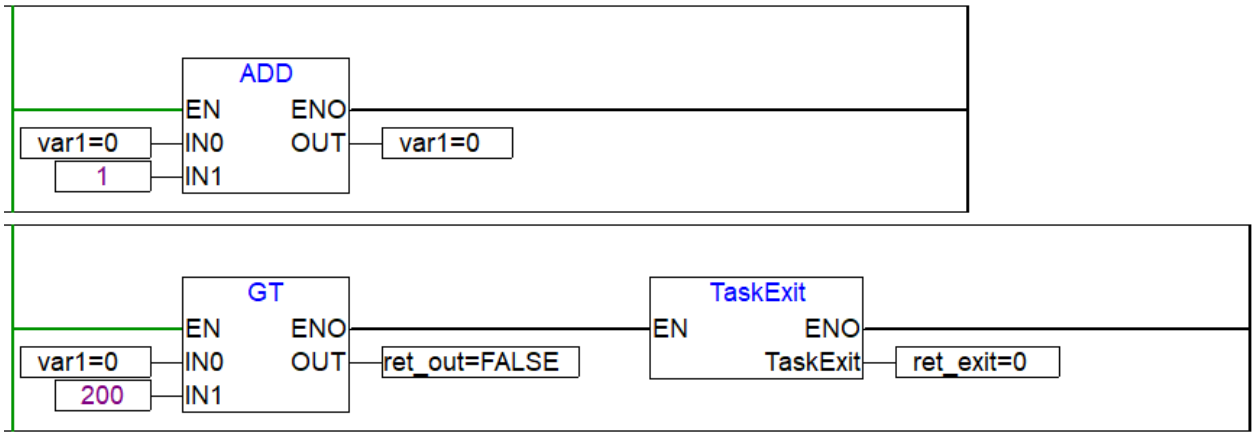
7.2.5.2 Parameter

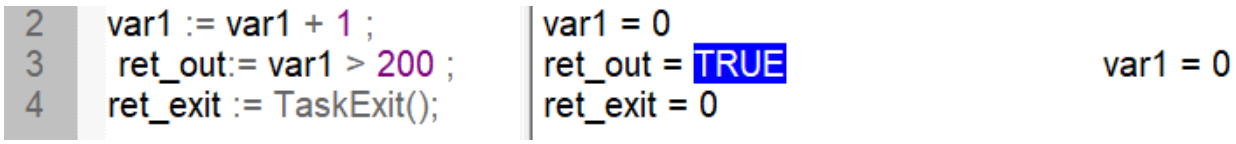
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
TaskExit	output	DINT	0 returned if succeeded; -1 returned if failed.	0	FALSE

7.2.5.3 Example

When the value of Var1 is added by 1 and output to var1, if var1 is greater than 200, TaskExit is executed (exiting task). If succeeded, ret=0, and if failed, ret=-1.



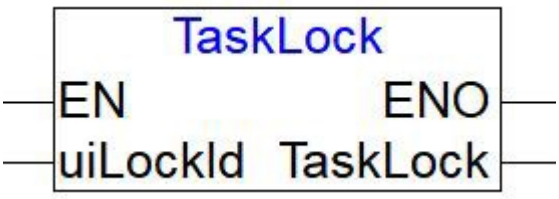




7.2.6 TaskLock (Task Lock)

7.2.6.1 Function

The system provides a total of 32 system locks and 2 locking methods (blocking and non-blocking) so that users can protect shared resources (data) between different tasks. Among them, TaskLock is a blocking locking interface. When the lock is not acquired, the task thread is blocked and suspended until other tasks release the lock.



TaskLock Functional Block

7.2.6.2 Parameter

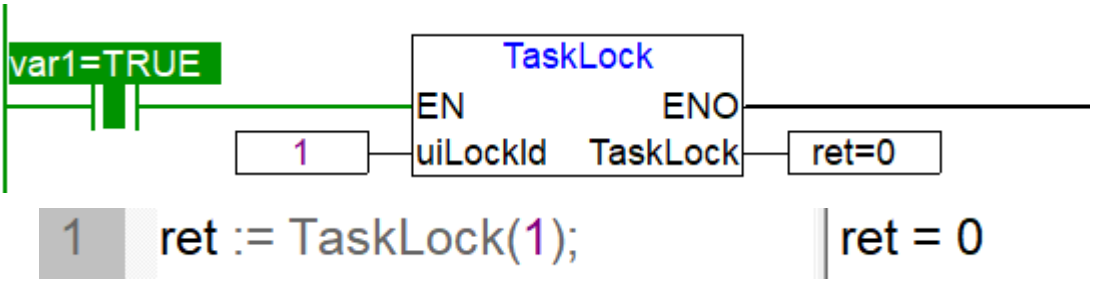
Parameter	Statement	Data Type	Description	Default Value	Power Fail Safeguard

EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
Input port	uilockId	BYTE	Task lock index number, 0~31	0	FALSE
Output terminal	TaskLock	DINT	0 returned if locking failed, and 1 returned if locking succeeded	0	FALSE

7.2.6.3 Example

TaskLock is a blocking locking interface. When the lock is not acquired, the task thread is blocked and suspended until other tasks release the lock. If the TaskLock instruction runs successfully, the output value is 1, and if it fails, the output value is 0.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	ret			DINT	0	FALSE	Not Public	
0002	var1			BOOL	TRUE	FALSE	Not Public	



7.2.7 TaskTryLock (Task Try Lock)

7.2.7.1 Function

TaskTryLock is a non-blocking locking interface. This interface will not block the task thread when the lock is not acquired and will return 0 immediately. Users need to decide whether to access resources (data) shared between different tasks based on the return value of this interface. It can be unlocked only after the lock is locked successfully.

TaskTryLock Functional Block

7.2.7.2 Parameter

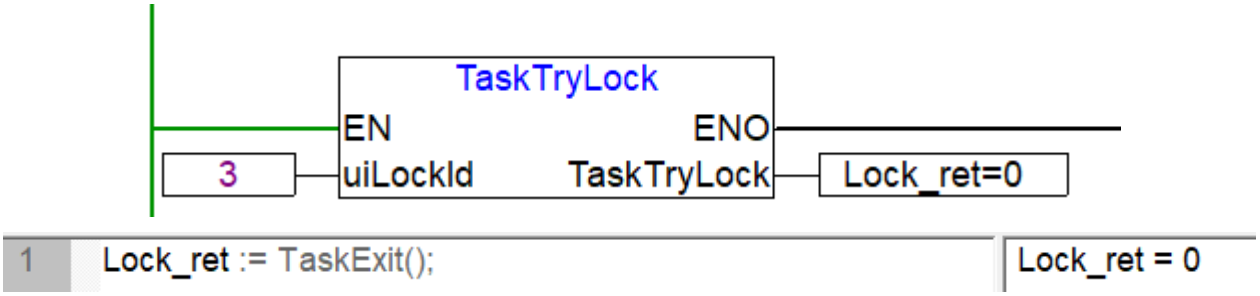
Parameter	Statement	Data	Description	Initial	Power Fail

		Type		Value	Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
uilockId	Input	BYTE	Task lock index number, 0~31	0	FALSE
TaskTryLock	Output	DINT	0 returned if locking failed, and 1 returned if locking succeeded	0	FALSE

7.2.7.3 Example

TaskTryLock is a non-blocking locking interface. This interface will not block the task thread when the lock is not acquired and will return 0 immediately. The output port is 0 for locking failure and 1 for successful locking.

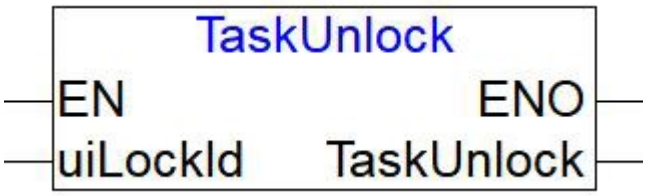
No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Lock_ret			DINT	0	FALSE	Not Public	



7.2.8 TaskUnlock (Task Unlock)

7.2.8.1 Function

After the system lock is locked, TaskUnlock provides an unlocking interface.



TaskUnlock Functional Block

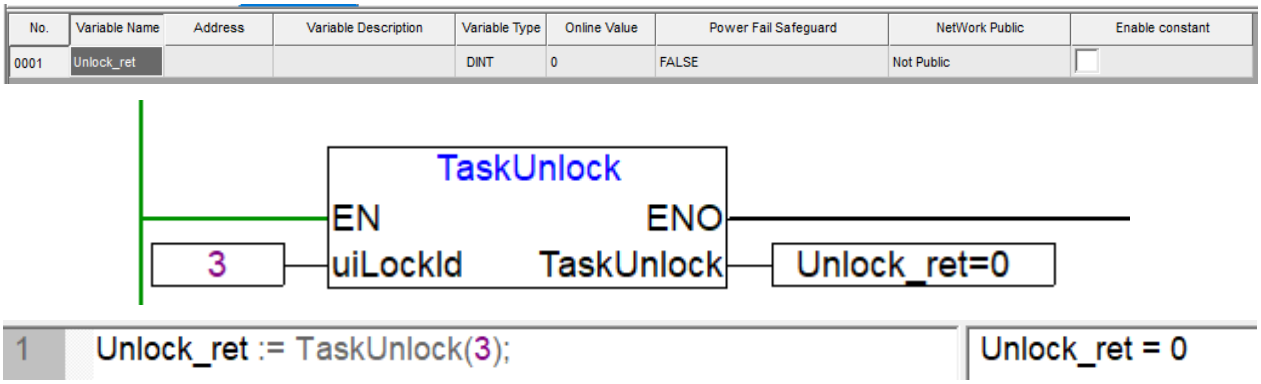
7.2.8.2 Parameter

Parameter	Statement	Data Type	Description	Default Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
uilockId	Input	BYTE	Task lock index number, 0~31	0	FALSE

TaskUnlock	Output	DINT	0 returned if unlocking failed, and 1 returned if the unlocking succeeded	0	FALSE

7.2.8.3 Example

After the system lock is locked, TaskUnlock provides an unlocking interface. If TASKID is successfully unlocked, the output value is 1, and if failed, it is 0.



7.2.9 TaskSetWdtTimeout (Set up the task door dog timeout time)

7.2.9.1 Function

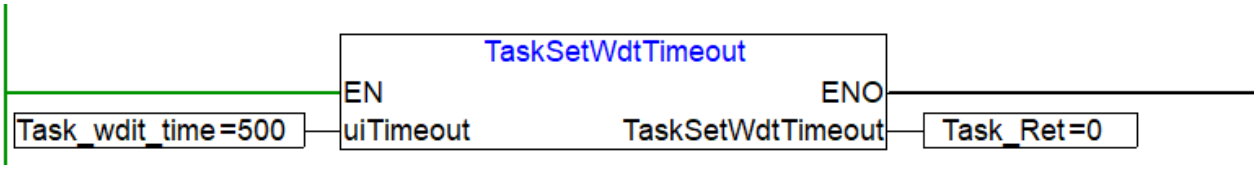
Set the timeout period of the system watchdog. When the system is abnormal, it enters the abnormal status after waiting for the Timeout period.

7.2.9.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
UITIMEOUT	Input	DINT	Watchdog timeout period (ms)	0	FALSE
TaskSetWdtTimeout	Output	UDINT	0 returned if locking failed, and 1 returned if the locking succeeded	0	FALSE

7.2.9.3 Example

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Task_wdit_time			UDINT	500	FALSE	Not Public	
0002	Task_Ret			DINT	0	FALSE	Not Public	

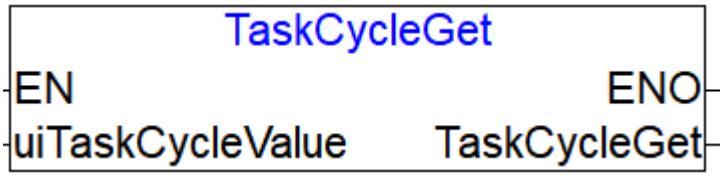



```
3   Task_Ret := TaskSetWdtTimeout(Task_wdit_time);   Task_Ret = 0   Task_wdit_time = 500
```

7.2.10 TaskCycleGet (get current task period)

7.2.10.1 Function

Get the cycle time used by the IEC task where thef unction block is located



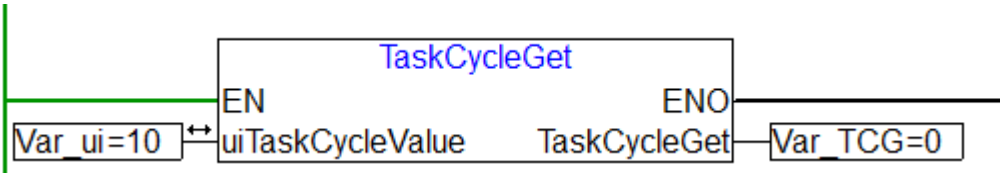
TaskCycleGet Functional Block

7.2.10.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value
uiTaskCycleValue	InputOutput	UDINT	Acquired cycle time	0
TaskCycleGet	Output	DINT	Return value: Get Task cycle success; 1: Task cycle type is a single run aperiodic task, can not get the task cycle value; 2: Task cycle type is event running aperiodic task, can not get task cycle value; 3: Task cycle type is full speed running aperiodic task, can not get the task cycle value; 4: The current task information was not obtained.	0

7.2.10.3 Example

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var_ui			UDINT	10	FALSE	Not Public	
0002	Var_TCG			DINT	0	FALSE	Not Public	




```
5
6   Var_TCG := TaskCycleGet(Var_ui);
```

Var_TCG = 0

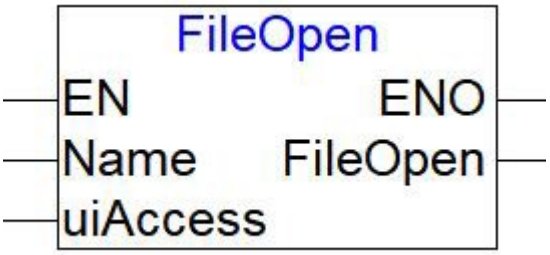
Var_ui = 10

7.3 File Operation Instruction

7.3.1 FileOpen (Open File)

7.3.1.1 Function

Open a file according to the method set by the user.



FileOpen Functional Block

7.3.1.2 Parameter

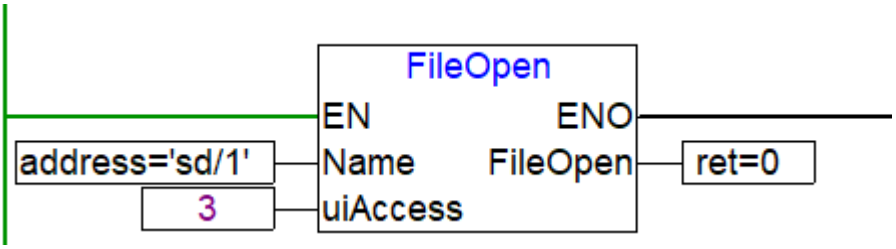
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
Name	Input	STRING(63)	File name, which supports up to 63 characters and only supports full path. "sd/" is the SD card directory "flash/" is the Flash directory		FALSE
uiAccess	Input	UDINT	File operation mode 0: Read and write (open a file in read/write mode by	0	FALSE

			default) 1: Read-only (open a file in read-only mode) 2: Write-only (open a file in write-only mode) 8: Create (create and open a file) 16: Clear (open a file and clear the contents of it) 32: Append (open a file and locate the cursor to the end of it) Note: 1) Except for the append method, other methods will locate the cursor at the file header. 2) The write operation will overwrite the content after the cursor. 3) The cursor can be moved through the FileLseek instruction.		
FileOpen	Output	DINT	Success: Return file access ID Failed: -1 returned	0	FALSE

7.3.1.3 Example

According to the method set by the user, the Name of the open file is the path to be opened and uiAccess is the opening method selected. The output value is the file ID if the file is opened successfully, and -1 if it fails to open the file.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	address			STRING(63)	'sd/1'	FALSE	Not Public	
0002	ret			DINT	0	FALSE	Not Public	




```
1  ret := FileOpen(address, 3);
```

ret = 0 address = 'sd/1'

7.3.2 FileClose (Close File)

7.3.2.1 Function

Close a file.

FileClose Functional Block

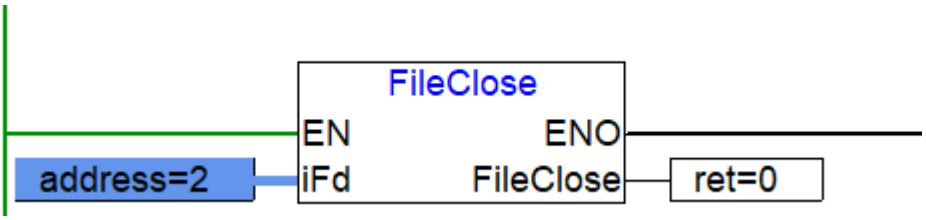
7.3.2.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
iFd	Input	DINT	FileOpen returns file ID	0	FALSE
FileClose	Output	DINT	Succeeded: 0 returned Failed: -1 returned	0	FALSE

7.3.2.3 Example

The input of FileClose is the address of the file to be closed. The output result is 0 for successful closing and -1 for failure.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	address			DINT	2	FALSE	Not Public	
0002	ret			DINT	0	FALSE	Not Public	




```
1  ret := FileClose(address);
```

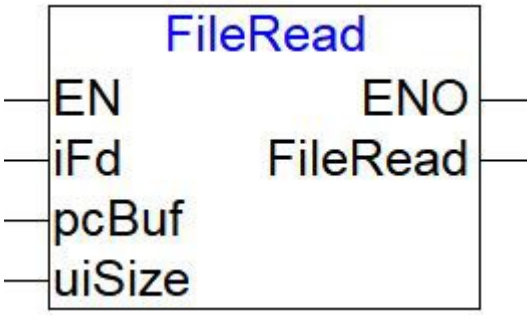
ret = 0

address = 2

7.3.3 FileRead (Read File)

7.3.3.1.1 Function

Read a file.



FileRead Functional Block

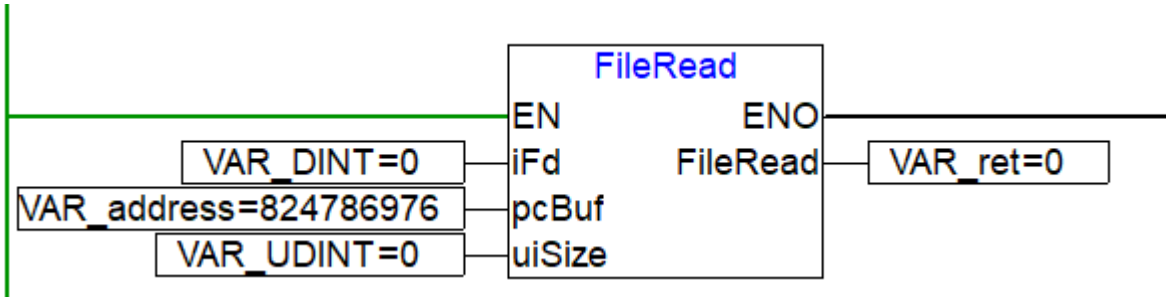
7.3.3.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
iFd	Input	DINT	FileOpen returns file ID	0	FALSE
pcBuf	Input	POINTER TO USINT	Memory address of read data	0	FALSE
uiSize	Input	UDINT	Size of read file (byte)	0	FALSE
FileRead	Output	DINT	Success: Return the number of bytes actually read from the file Failed: -1 returned	0	FALSE

7.3.3.3 Example

It is to set the ID, memory address, and size of the file to be read. The output value is the number of bytes of the file if the reading is successful, and -1 if it fails.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	VAR_DINT			DINT	0	FALSE	Not Public	
0002	VAR_address			POINTER T...	824786976	FALSE	Not Public	
0003	VAR_UDINT			UDINT	0	FALSE	Not Public	
0004	VAR_ret			DINT	0	FALSE	Not Public	

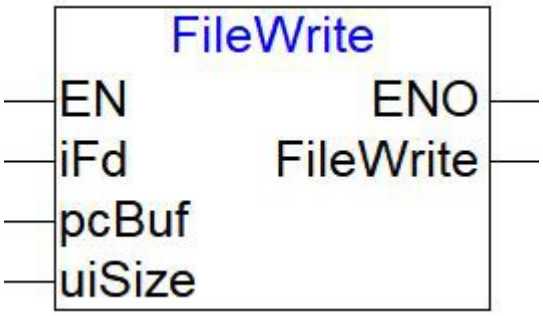



```
1  VAR_ret := FileRead(VAR_DINT, VAR_adress, VAR_UDINT);
```

7.3.4 FileWrite (Write File)

7.3.4.1 Function

Write a file.



FileWrite Functional Block

7.3.4.2 Parameter

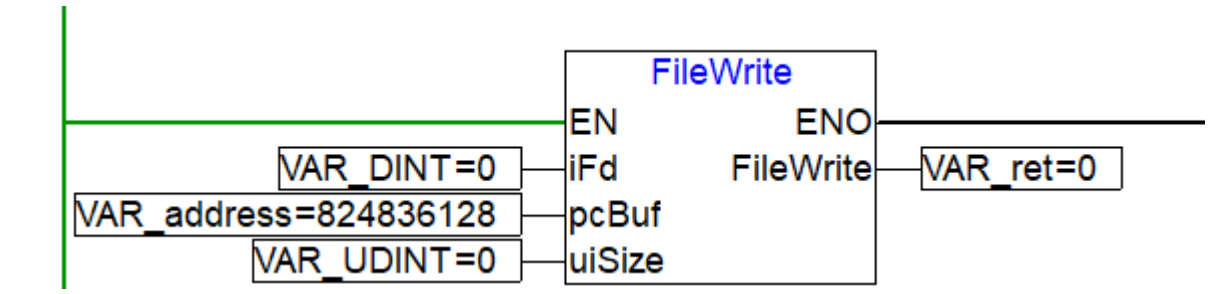
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
iFd	Input	DINT	FileOpen returns file ID	0	FALSE
pcBuf	Input	POINTER TO USINT	Memory address of written data	0	FALSE
uiSize	Input	UDINT	Size of written file (byte)	0	FALSE
FileRead	Output	DINT	Success: Return the actual size of the data written	0	FALSE

			Failed: -1 returned		

7.3.4.3 Example

It is to set the ID, memory address, and the size of the file to be read. If successful, the size of the actual written data will be returned. If failed, -1 will be returned.

No.	Variable Name	Address	/ariable Descriptior	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	VAR_DINT			DINT	0	FALSE	Not Public	
0002	VAR_address			POINTER TO USINT	824836128	FALSE	Not Public	
0003	VAR_UDINT			UDINT	0	FALSE	Not Public	
0004	VAR_ret			DINT	0	FALSE	Not Public	

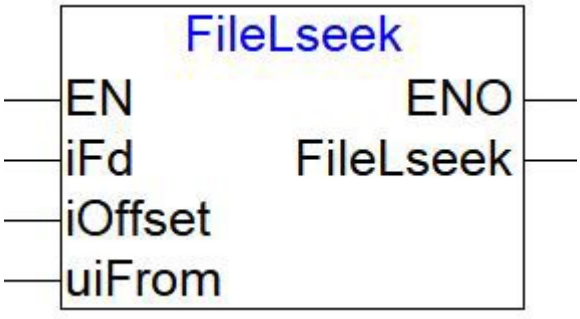



```
1  VAR_ret := FileWrite(VAR_DINT, VAR_adress, VAR_UDINT);
```

7.3.5 FileLseek (Move File to Read/Write Pointer)

7.3.5.1 Function

Move the file read/write pointer.



FileLseek Functional Block

7.3.5.2 Parameter

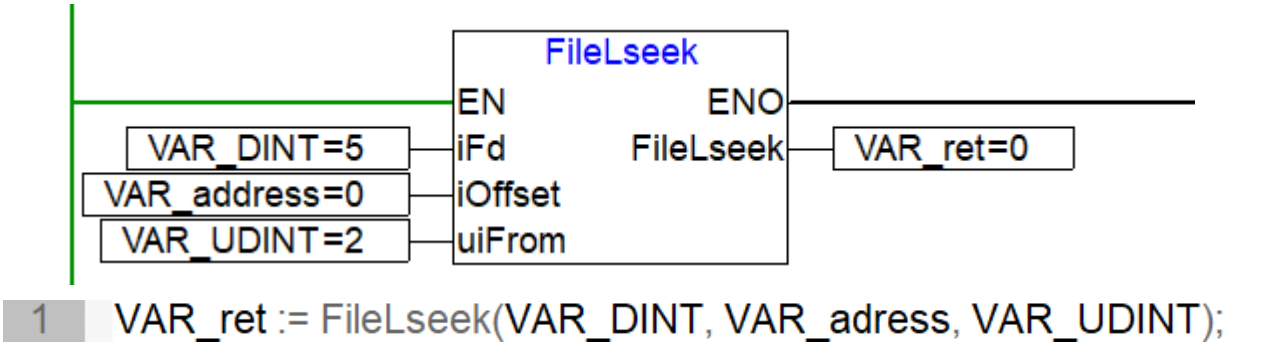
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard

EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
iFd	Input	DINT	FileOpen returns file ID	0	FALSE
iOffset	Input	DINT	Offset of moving file pointer	0	FALSE
uiFrom	Input	UDINT	0: Locate to the file header 1: Locate to the current position of the file 2: Locate to the file end	0	FALSE
FileLseek	Output	DINT	Success: Return the current read/write position of the file Failed: -1 returned	0	FALSE

7.3.5.3 Example

It is to move file read/write pointer vardint1 writes the file ID, vardint2 is the offset, and var3udint locates the file position. If successful, the current read/write position of the file will be output, and if it fails, -1 will be output.

No.	Variable Name	Address	/ariable Descriptior	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	VAR_DINT			DINT	5	FALSE	Not Public	
0002	VAR_address			DINT	0	FALSE	Not Public	
0003	VAR_UDINT			UDINT	2	FALSE	Not Public	
0004	VAR_ret			DINT	0	FALSE	Not Public	




```
1  VAR_ret := FileLseek(VAR_DINT, VAR_adress, VAR_UDINT);
```

7.3.6 FileListScan (Scan Directory File)

7.3.6.1 Function

Scan all files contained in the specified directory.

FileListScan Functional Block

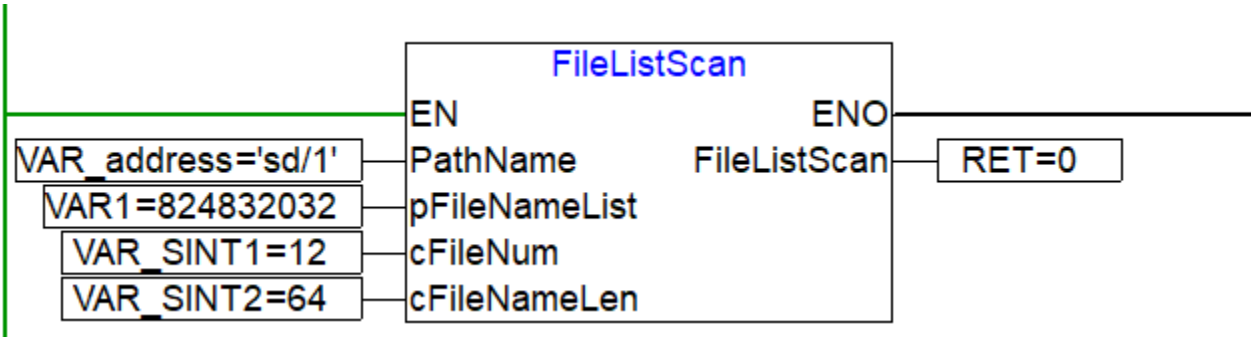
7.3.6.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
PathName	Input	STRING(63)	Name of the directory to be scanned, which supports up to 63 characters and only supports full path mode. "sd/" is the SD card directory, and "flash/" is the Flash directory.		FALSE
pFileNameList	Input	POINTER TO USINT	Address of string pointer array, which returns a list of scanned file names	0	FALSE
cFileNum	Input	SINT	Number of files to be scanned, which currently supports up to 32 files	0	FALSE
cFileNameLen	Input	SINT	Size of storage space opened for each file name, of which the string ends with '\0', (including the terminator '\0', up to 64 characters supported) and is recommended to set to 64	0	FALSE
FileListScan	Output	SINT	Success: Return the number of scanned files Failed: -1 returned	0	FALSE

7.3.6.3 Example

It is to scan all files contained in the specified directory. PathName is the path of the file to be scanned, pFileNameList is the file address, cFileNum is the number of files, and cFileNameLen selects the storage space size of the file to be scanned. If successful, the output value is the number of files scanned. If it fails, -1 is returned.

No.	Variable Name	Address	/ariable Descriptior	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	VAR_address			STRING(63)	'sd/1'	FALSE	Not Public	
0002	VAR1			POINTER TO USINT	824832032	FALSE	Not Public	
0003	VAR_SINT1			SINT	12	FALSE	Not Public	
0004	VAR_SINT2			SINT	64	FALSE	Not Public	
0005	RET			SINT	0	FALSE	Not Public	




```
1  RET := FileListScan(VAR_adress, VAR1, VAR_SINT1, VAR_SINT2);
```

7.3.7 FindFirstFile (Find First File)

7.3.7.1 Function

Open a file handle that traverses the directory and obtain the first file name in the directory.

FindFirstFile Functional Block

7.3.7.2 Parameter

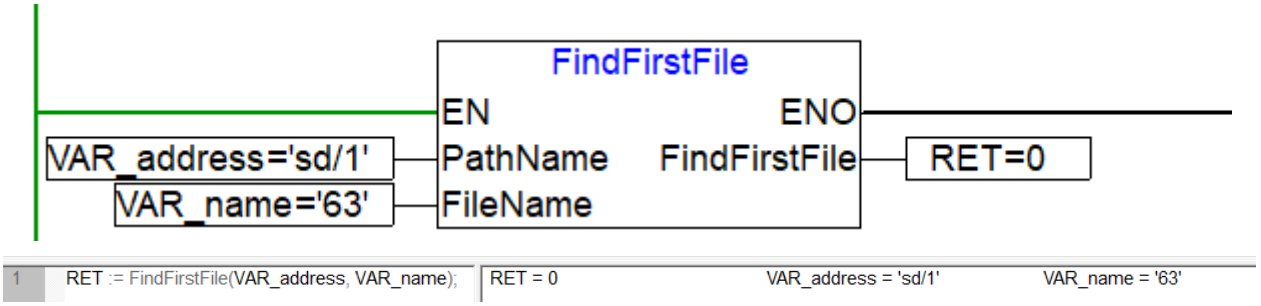
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
PathName	Input	STRING(63)	Name of the directory to be traversed, which supports up to 63 characters and only supports full path mode. "sd/" is the SD card directory and "flash/" is the Flash directory.		FALSE
FileName	Input	STRING(63)	Return the traversed file name (excluding path, up to 63 characters supported)		FALSE
FindFirstFile	Output	DINT	Succeeded: Return the handle of scanned files	0	FALSE

			Failed: Return (-1)		

7.3.7.3 Example

It is to find the first file. PathName is the path to find, and FileName returns the traversed file name. If successful, the scanned file handle is returned. If it fails, -1 is returned.

No.	Variable Name	Address	/ariable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	VAR_address			STRING(63)	'sd/1'	FALSE	Not Public	
0002	VAR_name			STRING(63)	'63'	FALSE	Not Public	
0003	RET			DINT	0	FALSE	Not Public	



7.3.8 FindNextFile (Find Next File)

7.3.8.1 Function

Traverse the directory and get the next file name in the specified directory.

FindNextFile Functional Block

7.3.8.2 Parameter

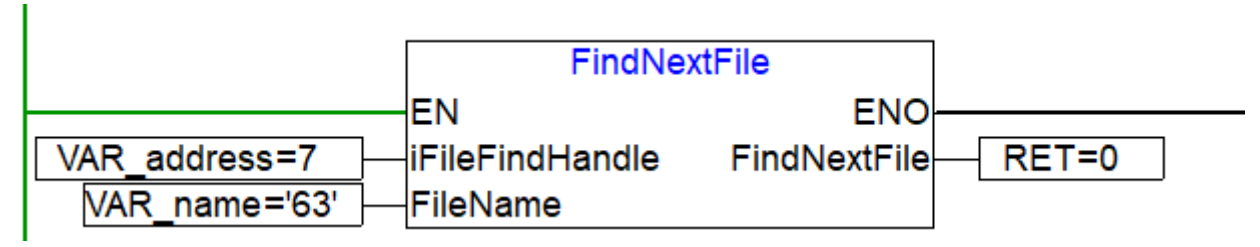
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
iFileFindHandle	Input	DINT	File traversal handle (return value of FindFirstFile)	0	FALSE
FileName	Input	STRING(63)	Return the traversed file name (excluding path,		FALSE

			up to 63 characters supported)		
FindNextFile	Output	SINT	Succeeded: 0 returned Failed: -1 returned	0	FALSE

7.3.8.3 Example

It is to traverse the directory and obtain the next file name of the specified directory by entering the file traversal handle and the traversed file name. If successful, the output value is 0, and if failed, -1 is returned.

No.	Variable Name	Address	/ariable Descriptior	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	VAR_address			DINT	7	FALSE	Not Public	
0002	VAR_name			STRING(63)	'63'	FALSE	Not Public	
0003	RET			DINT	0	FALSE	Not Public	



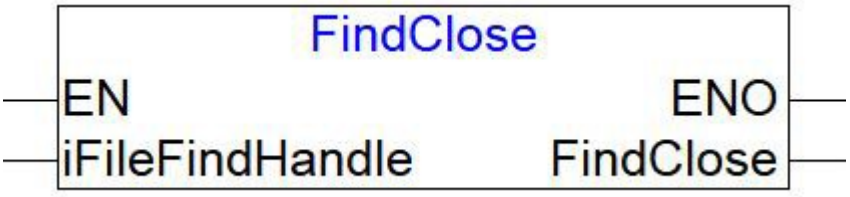

```
RET := FindNextFile(VAR_address, VAR_name);
```

RET = 0 VAR_address = 7 VAR_name = '63'

7.3.9 FindClose (Finding End)

7.3.9.1 Function

Close the traversal handle.



FindClose Functional Block

7.3.9.2 Parameter

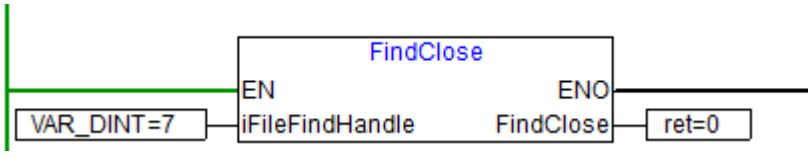
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
iFileFindHandle	Input	DINT	File traversal handle (return value of	0	FALSE

			FindFirstFile)		
FindClose	Output	SINT	Succeeded: 0 returned Failed: -1 returned	0	FALSE

7.3.9.3 Example

It is to close the traversal handle by entering the file traversal handle instruction. If successful, the output value is 0, and if failed, it is -1.

No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard
0001	var1dint			DINT	7	FALSE
0002	ret			SINT	0	FALSE




```
1  ret := FindClose(VAR_DINT);
```

ret = 0

VAR_DINT = 7

7.3.10 FileRemove (Deleting File)

7.3.10.1 Function

Delete a file.

FileRemove Functional Block

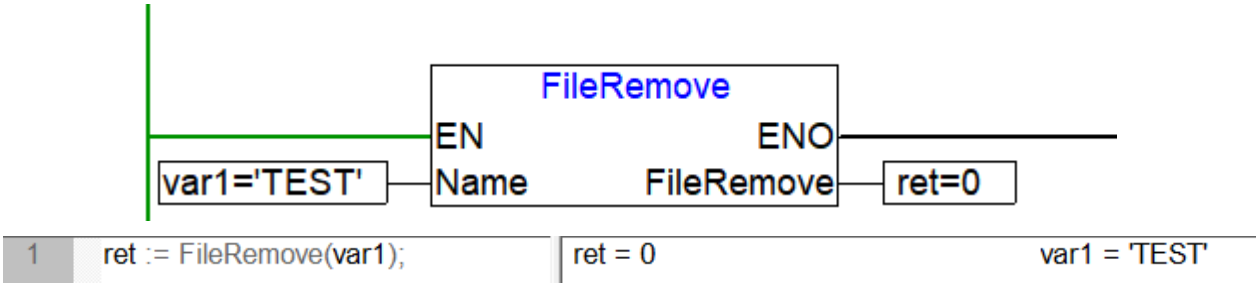
7.3.10.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
Name	Input	STRING(63)	File name to be traversed, which supports up to 63 characters and only supports full path mode. "sd/" is the SD card directory and "flash/" is the Flash directory.		FALSE
FileRemove	Output	DINT	Succeeded: 0 returned Failed: -1 returned	0	FALSE

7.3.10.3 Example

It is to delete a file by entering the path of the file to delete. If the instruction is executed successfully, the output value is 0, and if it fails, it is -1.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	var1			STRING(63)	'TEST'	FALSE	Not Public	
0002	ret			DINT	0	FALSE	Not Public	



7.4 Time Correlation Instruction

7.4.1 GetTickCnt (Get Tick Count)

7.4.1.1 Function

Get the time(us) since the controller was powered on.

GetTickCnt Functional Block

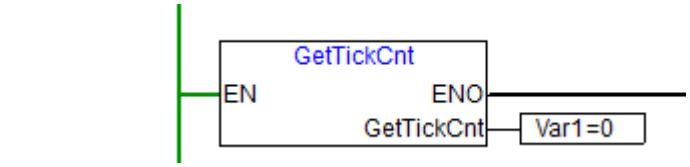
7.4.1.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
GetTickCnt	Output	UDINT	Get the time since the controller was powered on, and restart the time from zero after overflow (us)	0	FALSE

7.4.1.3 Example

The following example gets the time since the controller was powered on by calling the GetTickCnt instruction, which gets the time from the controller and stores the results in the output parameter GetTickCnt.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1			UDINT	0	FALSE	Not Public	




```
1  Var1 := GetTickCnt();        Var1 = 0
```

7.4.2 TimeDelay (Time Delay)

7.4.2.1 Function

After this function is called, it returns after executing uiTimeDelay milliseconds and finally achieve the effect of time delay.

TimeDelay Functional Block

7.4.2.2 Parameter

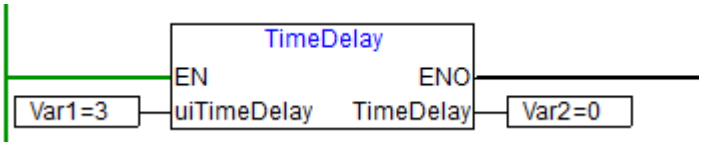
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
uiTimeDelay	Input	UDINT	Delay time (ms)	0	FALSE
TimeDelay	Output	DINT	Execution status 0: Delay executed successfully; -1: Delay execution failed	0	FALSE

7.4.2.3 Example

The following example uses the TimeDelay instruction to implement time delay. After this function is called,

it returns after executing uiTimeDelay milliseconds, finally achieving the effect of time delay, and stores the status parameter of whether the execution is successful into TimeDelay.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var1		Delay time	UDINT	3	FALSE	Not Public	
0002	Var2		Execution status	DINT	0	FALSE	Not Public	




```
1  Var2 := TimeDelay(Var1);
```

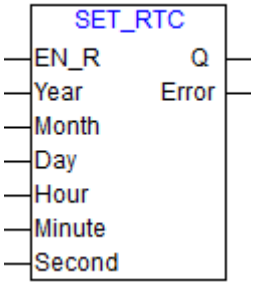
Var2 = 0

Var1 = 3

7.4.3 SET_RTC (Set Real Time Clock)

7.4.3.1 Function

Set the user-specified time into the PLC real-time clock.



SET_RTC Functional Block

7.4.3.2 Parameter

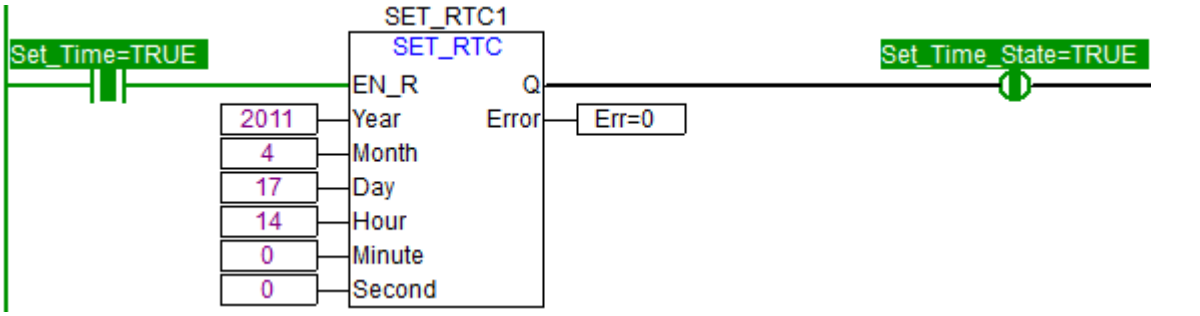
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN_R	Input	BOOL	Enable 0: Invalid 1: Valid on rising edge	FALSE	FALSE
Year	Input	WORD	Year LX platform: 2000~2036	2011	FALSE
Month	Input	BYTE	Month 1-12	1	FALSE
Day	Input	BYTE	Day 1-31	1	FALSE
Hour	Input	BYTE	Hour 0-23	0	FALSE
Minute	Input	BYTE	Minute 0-59	0	FALSE
Second	Input	BYTE	Ssecond	0	FALSE

			0-59		
Q	Output	BOOL	Setup completed or not 0: Not completed 1: Completed	FALSE	FALSE
Error	Output	BYTE	Parameter setting error =0: There is no error in setting the RTC time If Not=0: It indicates an error occurred in setting the RTC time =1: The set year exceeds the specified range =2: The set month is unreasonable =3: The set date is unreasonable =4: The set hour is unreasonable =5: The set minute is unreasonable =6: The set second is unreasonable =200: The module is repeatedly defined and called	0	FALSE

7.4.3.3 Example

The following example uses the SET_RTC instruction to set the user-specified time to the PLC real-time clock, and stores the setting results and error codes into the output parameters Q and Error.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	SET_RTC1			SET_RTC		FALSE	Not Public	
0002	Set_Time			BOOL	TRUE	FALSE	Not Public	
0003	Err			BYTE	0	FALSE	Not Public	
0004	Set_Time_State			BOOL	TRUE	FALSE	Not Public	



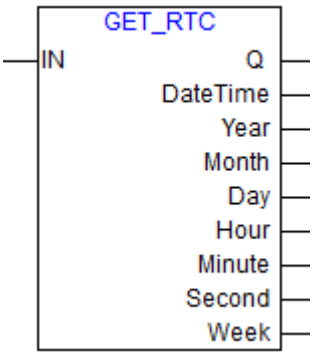

```
1  SET_RTC1(
2  EN_R := Set_Time,
3  Year := 2011,
4  Month := 4,
5  Day := 17,
6  Hour := 14,
7  Minute := 0,
8  Second := 0,
9  Q=>Set_Time_State,
10 Error=>Err);
```

```
SET_RTC1
Set_Time = TRUE


Set_Time_State = TRUE
Err = 0
```

7.4.4 GET_RTC (Read Real Time Clock)

7.4.4.1 Function

This functional block reads the time in the PLC hardware real-time clock, which is UTC+08:00.



GET_RTC Functional Block

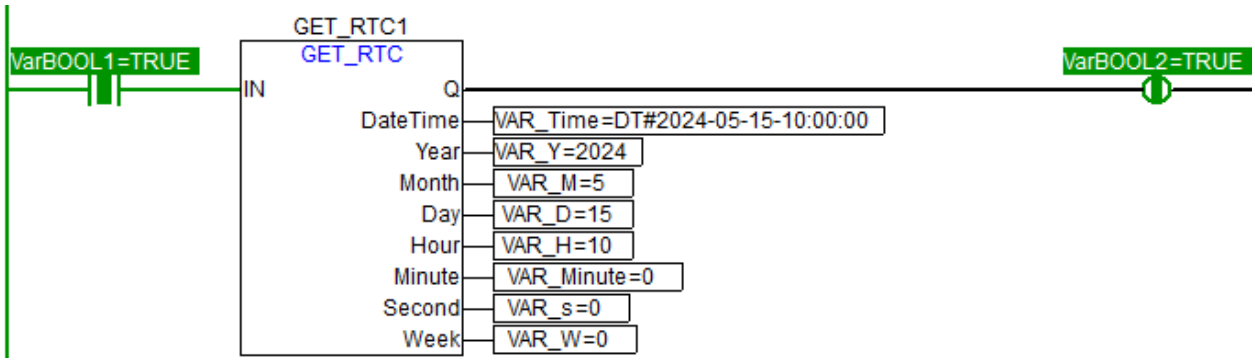
7.4.4.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
IN	Input	BOOL	0: Invalid 1: Valid	FALSE	FALSE
Q	Output	BOOL	0: No data read out 1: Data read correctly	FALSE	FALSE
DateTime	Output	DT	Date/time	DT#1970-01-01-00:00:00	FALSE
Year	Output	WORD	Year	0	FALSE
Month	Output	BYTE	Month	0	FALSE
Day	Output	BYTE	Day	0	FALSE
Hour	Output	BYTE	Hour	0	FALSE
Minute	Output	BYTE	Minute	0	FALSE
Second	Output	BYTE	Ssecond	0	FALSE
Week	Output	BYTE	Week	0	FALSE

7.4.4.3 Example

The following example uses the GET_RTC instruction to read the time of the PLC hardware real-time clock, which is the UTC+08:00 time.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	GET_RTC1			GET_RTC		FALSE	Not Public	
0002	varBOOL			BOOL	FALSE	FALSE	Not Public	
0003	VAR_Time			DT	DT#2024-05-15...	FALSE	Not Public	
0004	VAR_Y			WORD	2024	FALSE	Not Public	
0005	VAR_M			BYTE	5	FALSE	Not Public	
0006	VAR_D			BYTE	15	FALSE	Not Public	
0007	VAR_H			BYTE	10	FALSE	Not Public	
0008	VAR_Minute			BYTE	0	FALSE	Not Public	
0009	VAR_s			BYTE	0	FALSE	Not Public	
0010	VAR_W			BYTE	0	FALSE	Not Public	
0011	VarBOOL2			BOOL	FALSE	FALSE	Not Public	



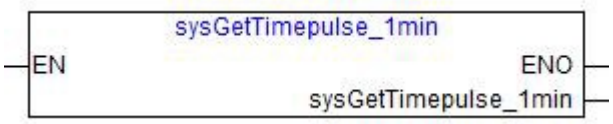

```
1  GET_RTC1(
2  IN := varBOOL,
3  Q=>VarBOOL2,
4  DateTime=>VAR_Time,
5  Year=>VAR_Y,
6  Month=>VAR_M,
7  Day=>VAR_D,
8  Hour=>VAR_H,
9  Minute=>VAR_Minute,
10 Second=>VAR_s,
11 Week=>VAR_W);
```

```
GET_RTC1
varBOOL = TRUE
VarBOOL2 = TRUE
VAR_Time = DT#2024-05-15-1...
VAR_Y = 2024
VAR_M = 5
VAR_D = 15
VAR_H = 10
VAR_Minute = 0
VAR_s = 0
VAR_W = 0
```

7.4.5 sysGetTimepulse_1min (Get 1 minute time pulse)

7.4.5.1 Function

Output a square wave pulse signal with a period of 1 minute, with the state flipping every 30 seconds.



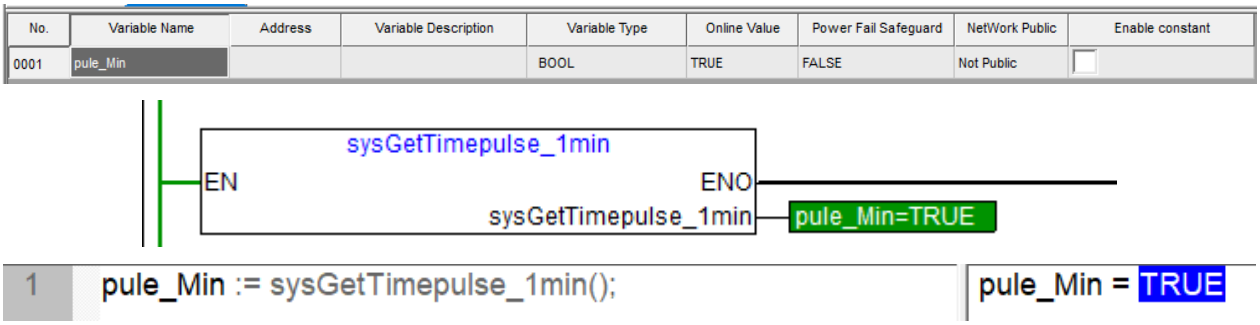
sysGetTimepulse_1min Functional Block

7.4.5.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
sysGetTimepulse_1min	Output	BOOL	Current status of the 1-minute pulse	FALSE	FALSE

7.4.5.3 Example

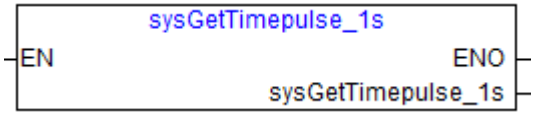
The following example uses the sysGetTimepulse_1min instruction to implement the function of outputting a square wave pulse signal with a period of 1 minute. The return value status flips every 30 seconds.



7.4.6 sysGetTimepulse_1s (Get 1 second time pulse)

7.4.6.1 Function

Output a square wave pulse signal with a period of 1 second. The return value status flips every 500 milliseconds.



sysGetTimepulse_1s Functional Block

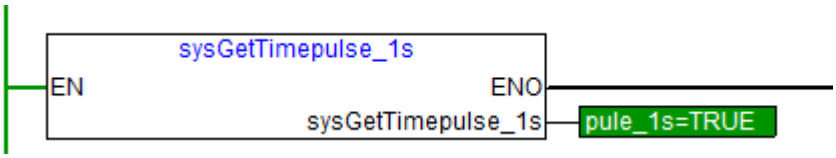
7.4.6.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
sysGetTimepulse_1s	Output	BOOL	Current status of the 1-second pulse	FALSE	FALSE

7.4.6.3 Example

The following example uses the sysGetTimepulse_1s instruction to implement the function of outputting a square wave pulse signal with a period of 1 second. The return value status flips every 500 milliseconds.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	pule_1s			BOOL	TRUE	FALSE	Not Public	

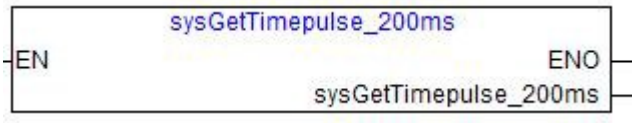



```
3   pule_1s := sysGetTimepulse_1s();        pule_1s = TRUE
```

7.4.7 sysGetTimepulse_200ms (Get 200 millisecond time pulse)

7.4.7.1 Function

Output a square wave pulse signal with a period of 200 milliseconds. The return value status flips every 100 milliseconds.



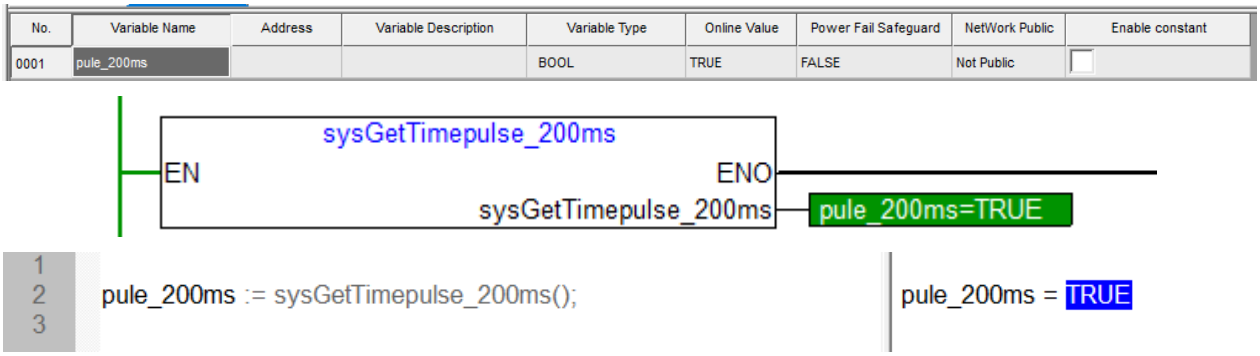
sysGetTimepulse_200ms Functional Block

7.4.7.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
sysGetTimepulse_200ms	Output	BOOL	Current status of 200 ms pulse	FALSE	FALSE

7.4.7.3 Example

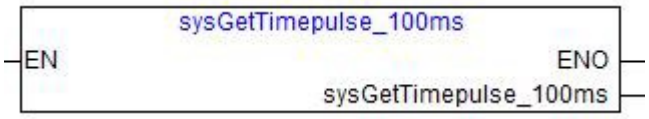
The following example uses the sysGetTimepulse_200ms instruction to implement the function of outputting a square wave pulse signal with a period of 200 milliseconds. The return value status flips every 100 milliseconds.



7.4.8 sysGetTimepulse_100ms (Get 100 millisecond time pulse)

7.4.8.1 Function

Output a square wave pulse signal with a period of 100 milliseconds. The return value status flips every 50 milliseconds.



sysGetTimepulse_100ms Functional Block

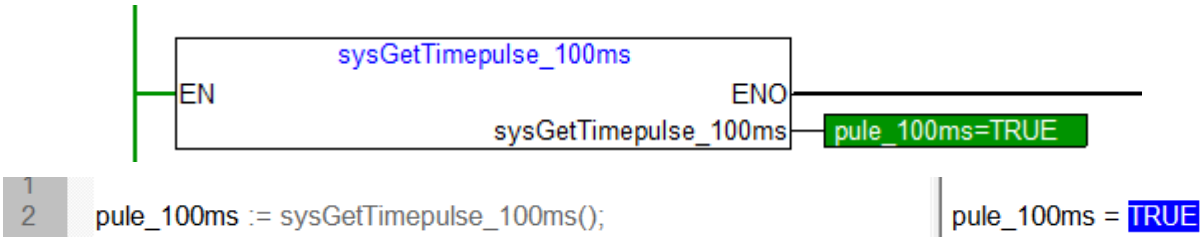
7.4.8.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
sysGetTimepulse_100ms	Output	BOOL	Current status of 100 ms pulse	FALSE	FALSE

7.4.8.3 Example

The following example uses the sysGetTimepulse_100ms instruction to implement the function of outputting a square wave pulse signal with a period of 100 milliseconds. The return value status flips every 50 milliseconds.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	pule_100ms			BOOL	TRUE	FALSE	Not Public	




```
1
2  pule_100ms := sysGetTimepulse_100ms();
```

pule_100ms = TRUE

7.4.9 sysGetTimepulse_20ms (Get 20 millisecond time pulse)

7.4.9.1 Function

Output a square wave pulse signal with a period of 20 milliseconds. The return value status flips every 10 milliseconds.

sysGetTimepulse_20ms Functional Block

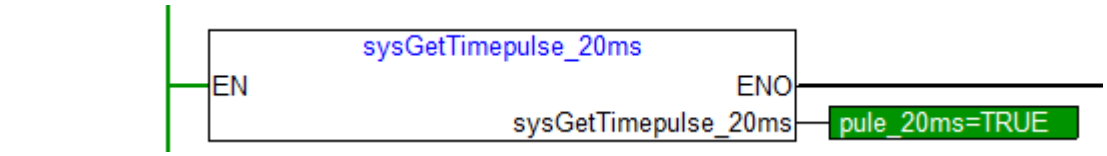
7.4.9.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
sysGetTimepulse_20ms	Output	BOOL	Current status of 20 ms pulse	FALSE	FALSE

7.4.9.3 Example

The following example uses the sysGetTimepulse_20ms instruction to implement the function of outputting a square wave pulse signal with a period of 20 milliseconds. The return value status flips every 10 milliseconds.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	pule_20ms			BOOL	TRUE	FALSE	Not Public	

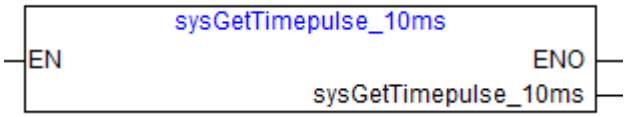



```
3   pule_20ms := sysGetTimepulse_20ms();        pule_20ms = TRUE
```

7.4.10 sysGetTimepulse_10ms (Get 10 millisecond time pulse)

7.4.10.1 Function

Output a square wave pulse signal with a period of 10 milliseconds. The return value status flips every 5 milliseconds.



sysGetTimepulse_10ms Functional Block

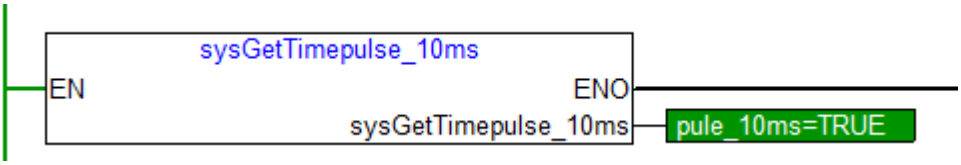
7.4.10.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
sysGetTimepulse_10ms	Output	BOOL	Current status of 10 ms pulse	FALSE	FALSE

7.4.10.3 Example

The following example uses the sysGetTimepulse_10ms instruction to implement the function of outputting a square wave pulse signal with a period of 10 milliseconds. The return value status flips every 5 milliseconds.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	pule_10ms			BOOL	TRUE	FALSE	Not Public	




```
4
5   pule_10ms := sysGetTimepulse_10ms();
```

pule_10ms = TRUE

7.4.11 sysGetTimepulse_1ms (Get 1 millisecond time pulse)

7.4.11.1 Function

Output a square wave pulse signal with a period of 1 millisecond. The return value status flips every 500 microseconds.

sysGetTimepulse_1ms Functional Block

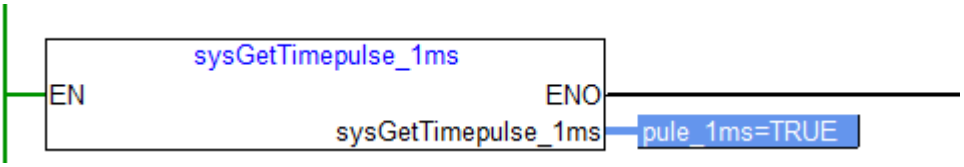
7.4.11.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
sysGetTimepulse_1ms	Output	BOOL	Current status of 1 ms pulse	FALSE	FALSE

7.4.11.3 Example

The following example uses the sysGetTimepulse_1ms instruction to implement the function of outputting a square wave pulse signal with a period of 1 millisecond. The return value status flips every 500 microseconds.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	pule_1ms			BOOL	TRUE	FALSE	Not Public	



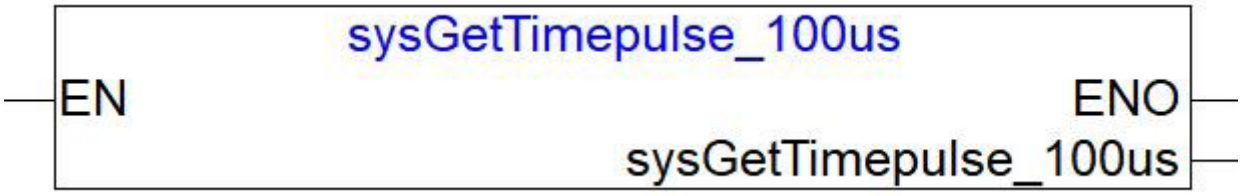

```
6
7   pule_1ms := sysGetTimepulse_1ms();
```

pule_1ms = TRUE

7.4.12 sysGetTimepulse_100us (Get 100 microsecond time pulse)

7.4.12.1 Function

Output a square wave pulse signal with a period of 100 microseconds. The return value status flips every 50 microseconds.



sysGetTimepulse_100us Functional Block

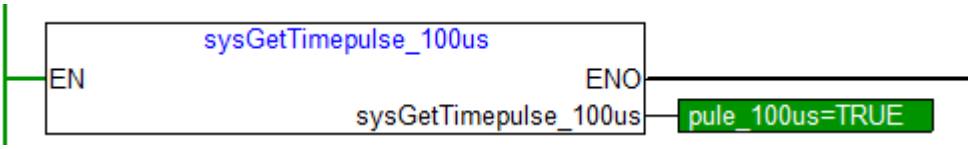
7.4.12.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
sysGetTimepulse_100us	Output	BOOL	Current status of 100 us pulse	FALSE	FALSE

7.4.12.3 Example

The following example uses the sysGetTimepulse_100us instruction to implement the function of outputting a square wave pulse signal with a period of 100 microseconds. The return value status flips every 50 microseconds.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	pule_100us			BOOL	TRUE	FALSE	Not Public	



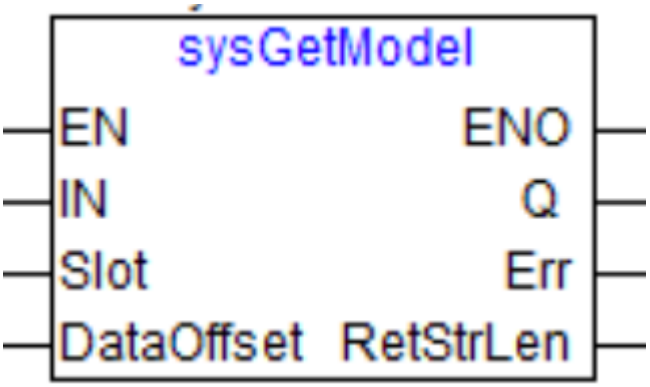

```
8
9   pule_100us := sysGetTimepulse_100us();        pule_100us = TRUE
```

7.5 Module Information Instruction

7.5.1 sysGetModel (Get Module Type)

7.5.1.1 Function

Get the module model.



sysGetModel Functional Block

7.5.1.2 Parameter

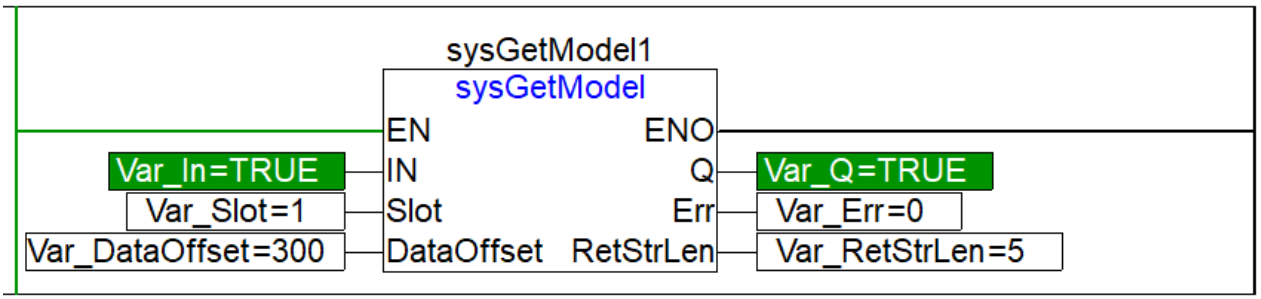
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN	Input	BOOL	Enable input		
ENO	Output	BOOL	Enable output		
IN	Input	BOOL	High level enable	FALSE	FALSE
Slot	Input	BYTE	Slot number (1 by default) The slot number of the controller module is fixed at 1 The slot number of other modules increases to the right according to the controller position To use it, enter the slot number of the module for which you want to get the version information	1	FALSE
DataOffset	Input	DWORD	Offset of area M (The module model is stored at the	0	FALSE

			specified offset position in string format) (0 by default, valid value range: 0-1048575)		
Q	Output	BOOL	Output	FALSE	FALSE
Err	Output	BYTE	Error (0 by default) 1: Slot number error 2: DataOffet offset configuration error 128: No module in the corresponding slot 129: The returned string exceeds the range of area M 130: Returned information error 131: The module type of this slot is unknown	0	FALSE
RetStrLen	Output	BYTE	Module model string length	0	FALSE

7.5.1.3 Example

The following example uses the sysGetModel instruction to obtain the module model.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	sysGetModel1			SYSGETMODEL		FALSE	Not Public	
0002	Var_In			BOOL	TRUE	FALSE	Not Public	
0003	Var_Slot			BYTE	1	FALSE	Not Public	
0004	Var_DataOffset			DWORD	300	FALSE	Not Public	
0005	Var_Q			BOOL	TRUE	FALSE	Not Public	
0006	Var_Err			BYTE	0	FALSE	Not Public	
0007	Var_RetStrLen			BYTE	5	FALSE	Not Public	



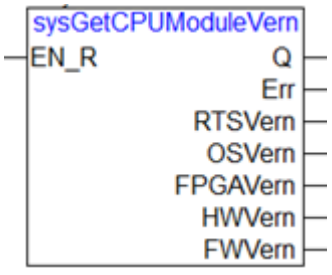

```
1  sysGetModel1(
2  IN := Var_In,
3  Slot := Var_Slot,
4  DataOffset := Var_DataOffset,
5  Q=>Var_Q,
6  Err=>Var_Err,
7  RetStrLen=>Var_RetStrLen);
```

```
sysGetModel1
Var_In = TRUE
Var_Slot = 1
Var_DataOffset = 300
Var_Q = TRUE
Var_Err = 0
Var_RetStrLen = 5
```

7.5.2 sysGetCPUModuleVern (Getting Firmware Version of CPU)

7.5.2.1 Function

Get the firmware version information of the controller, which is displayed in decimal format.



sysGetCPUModuleVern Functional Block

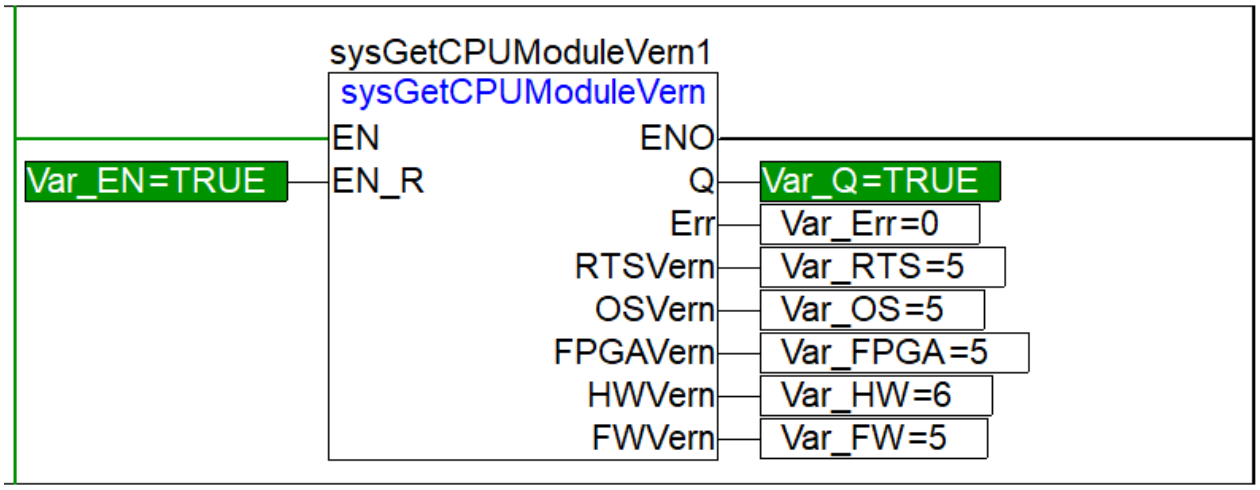
7.5.2.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN_R	Input	BOOL	Enable FALSE: Invalid (default value) TRUE: Valid on rising edge	FALSE	FALSE
Q	Output	BOOL	Output (FALSE by default)	FALSE	FALSE
Err	Output	BYTE	Error (default value 0) 128: Failed to obtain version	0	FALSE
RTSVern	Output	DWORD	RTS firmware version number The high 16 bits are the RTS version number on the fast core HEROS side, corresponding to the gadget RTOS_RTS version Vx.y.z. The low 16 bits are the RTS version number of the slow core LINUX side, corresponding to the gadget LINUX_RTS version Vx.y.z. The high 5 bits represent x, the middle 6 bits represent y, and the last 5 bits represent z.	0	FALSE
OSVern	Output	DWORD	OS version number The high 16 bits are the OS version number on the fast core HEROS side, corresponding to the gadget HEROS version Vx.y.z. The lower 16 bits are the OS version number on the slow core LINUX side, corresponding to the gadget LINUX version Vx.y.z. Among them, the high 5 bits represent x, the middle 6 bits represent y, and the last 5 bits represent z.	0	FALSE
FPGAVern	Output	WORD	FPGA version number, corresponding to the gadget FPGA version Vx.y.z There are 16 bits in total, where the high 5 bits represent x, the middle 6 bits represent y, and the last 5 bits represent z.	0	FALSE
HWVern	Output	WORD	Hardware version number Return value is 1~26, corresponding to A~Z	0	FALSE
FWVern	Output	WORD	Module firmware version number, corresponding to the gadget SW version Vx.y.z There are 16 bits in total, where the high 5 bits represent x, the middle 6 bits represent y, and the last 5 bits represent z.	0	FALSE

7.5.2.3 Example

The following example uses the sysGetCPUModuleVern instruction to obtain the firmware version information of the controller, which is displayed in decimal format.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	sysGetCPUModuleVern1			SYSGETCPUMODULEVERN		FALSE	Not Public	
0002	Var_EN			BOOL	TRUE	FALSE	Not Public	
0003	Var_Q			BOOL	TRUE	FALSE	Not Public	
0004	Var_Err			BYTE	0	FALSE	Not Public	
0005	Var_RTS			DWORD	5	FALSE	Not Public	
0006	Var_OS			DWORD	5	FALSE	Not Public	
0007	Var_FPGA			WORD	5	FALSE	Not Public	
0008	Var_HW			WORD	6	FALSE	Not Public	
0009	Var_FW			WORD	5	FALSE	Not Public	



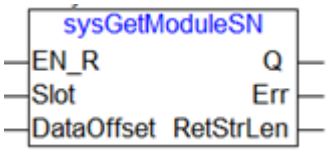

```
1  sysGetCPUModuleVern1(
2  EN_R := Var_EN,
3  Q=>Var_Q,
4  Err=>Var_Err,
5  RTSVern=>Var_RTS,
6  OSVern=>Var_OS,
7  FPGAVern=>Var_FPGA,
8  HWVern=>Var_HW,
9  FWVern=>Var_FW);
```

```
sysGetCPUModuleVern1
Var_EN = TRUE
Var_Q = TRUE
Var_Err = 0
Var_RTS = 5
Var_OS = 5
Var_FPGA = 5
Var_HW = 6
Var_FW = 5
```

7.5.3 sysGetModuleSN (Get Serial Number of Module)

7.5.3.1 Function

Get the module serial number.



sysGetModuleSN Functional Block

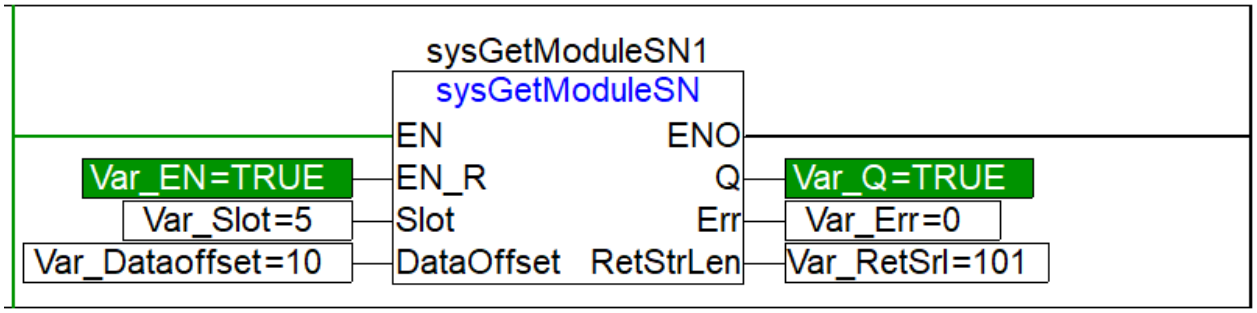
7.5.3.2 Parameter

Parameter	Statement	Data Type	Description	Default Value	Power Fail Safeguard
EN_R	Input	BOOL	Enable FALSE: Invalid TRUE: Valid on rising edge	FALSE	FALSE
Slot	Input	BYTE	Slot number (default: 1, valid value: 1-5) The slot number of the controller module is fixed at 1 The slot number of other modules increases to the right according to the controller position To use it, enter the slot number of the module for which you want to get the version information	1	FALSE
DataOffset	Input	DWORD	Offset of area M (The module serial number is stored at the specified offset position in string format) (0 by default, valid value range: 0-1048575)	0	FALSE
Q	Output	BOOL	Output	FALSE	FALSE
Err	Output	BYTE	Error (0 by default) 0: Correct 1: Slot number error 2: DataOffet offset configuration error 128: No module in the corresponding slot 129: The returned string exceeds the range of area M 130: Returned information error or no correct data obtained temporarily during the reading process (it will become 0 after the final reading is successful) 200: The module is repeatedly defined and called	0	FALSE
RetStrLen	Output	BYTE	Module serial number string length (default value: 0), up to 18-bit ASCII code supported, with the excess part to be automatically truncated	0	FALSE

7.5.3.3 Example

The following example uses the sysGetModuleSN instruction to obtain the module serial number.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	sysGetModuleSN1			SYSGETMODULESN		FALSE	Not Public	
0002	Var_EN			BOOL	TRUE	FALSE	Not Public	
0003	Var_Slot			BYTE	5	FALSE	Not Public	
0004	Var_Dataoffset			DWORD	10	FALSE	Not Public	
0005	Var_Q			BOOL	TRUE	FALSE	Not Public	
0006	Var_Err			BYTE	0	FALSE	Not Public	
0007	Var_RetSrl			BYTE	101	FALSE	Not Public	



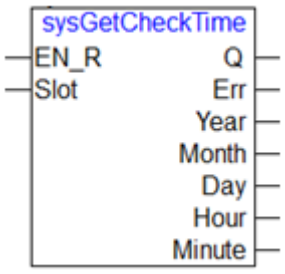

```
1  sysGetModuleSN1(
2  EN_R := Var_EN,
3  Slot := Var_Slot,
4  DataOffset := Var_Dataoffset,
5  Q=>Var_Q,
6  Err=>Var_Err,
7  RetStrLen=>Var_RetSrl);
```

```
sysGetModuleSN1
Var_EN = TRUE
Var_Slot = 5
Var_Dataoffset = 10
Var_Q = TRUE
Var_Err = 0
Var_RetSrl = 101
```

7.5.4 sysGetCheckTime (Get Check Time)

7.5.4.1 Function

Get the detection time.



sysGetCheckTime Functional Block

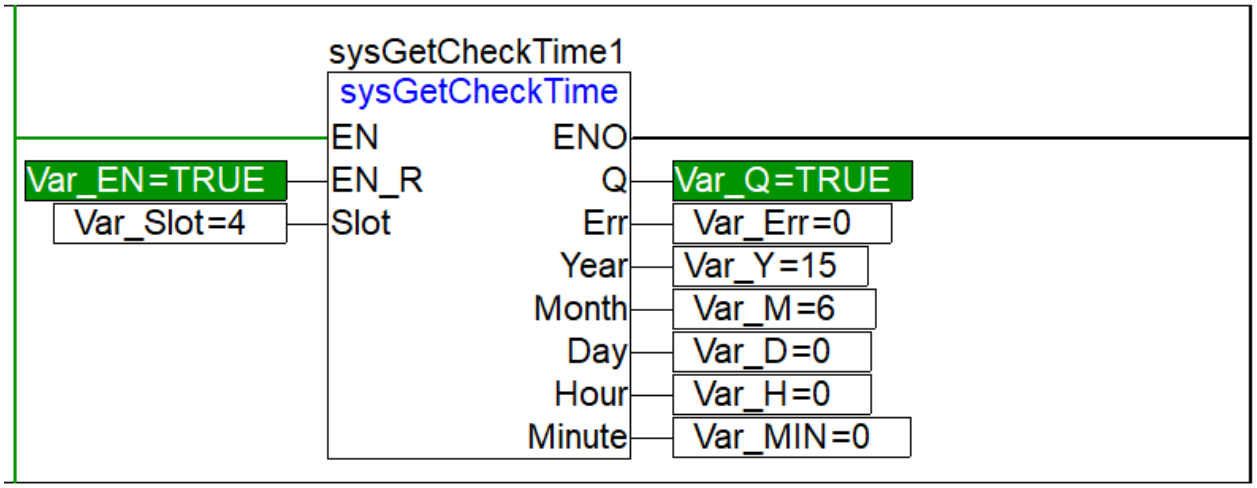
7.5.4.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN_R	Input	BOOL	Enable FALSE: Invalid (default value) TRUE: Valid on rising edge	FALSE	FALSE
Slot	Input	BYTE	Slot number (default: 1, valid value: 1-5) The slot number of the controller module is fixed at 1 The slot number of other modules increases to the right according to the controller position To use it, enter the slot number of the module for which you want to get the version information	1	FALSE
Q	Output	BOOL	Output (FALSE by default)	FALSE	FALSE
Err	Output	BYTE	Error (0 by default) 0: Correct 1: Slot configuration error 128: No module in the corresponding slot position 129: Failed to get 130: Incorrect year (2015-2099) 131: Incorrect month 132: Incorrect date 133: Incorrect hour 134: Incorrect minute 200: The module is repeatedly defined and called	0	FALSE
Year	Output	WORD	Year	0	FALSE
Month	Output	BYTE	Month	0	FALSE
Day	Output	BYTE	Day	0	FALSE
Hour	Output	BYTE	Hour	0	FALSE
Minute	Output	BYTE	Minute	0	FALSE

7.5.4.3 Example

The following example uses the sysGetCheckTime instruction to obtain the detection time.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	sysGetCheckTime1			SYSGETCHECKTIME		FALSE	Not Public	
0002	Var_EN			BOOL	TRUE	FALSE	Not Public	
0003	Var_Slot			BYTE	4	FALSE	Not Public	
0004	Var_Q			BOOL	TRUE	FALSE	Not Public	
0005	Var_Err			BYTE	0	FALSE	Not Public	
0006	Var_Y			WORD	15	FALSE	Not Public	
0007	Var_M			BYTE	6	FALSE	Not Public	
0008	Var_D			BYTE	0	FALSE	Not Public	
0009	Var_H			BYTE	0	FALSE	Not Public	
0010	Var_MIN			BYTE	0	FALSE	Not Public	



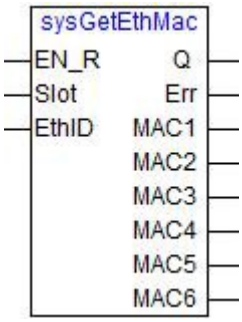

```
1  sysGetCheckTime1(
2  EN_R := Var_EN,
3  Slot := Var_Slot,
4  Q=>Var_Q,
5  Err=>Var_Err,
6  Year=>Var_Y,
7  Month=>Var_M,
8  Day=>Var_D,
9  Hour=>Var_H,
10 Minute=>Var_MIN);
```

```
sysGetCheckTime1
Var_EN = TRUE
Var_Slot = 4
Var_Q = TRUE
Var_Err = 0
Var_Y = 15
Var_M = 6
Var_D = 0
Var_H = 0
Var_MIN = 0
```

7.5.5 sysGetEthMac (Get MAC Address)

7.5.5.1 Function

Get the Mac address.



sysGetEthMac Functional Block

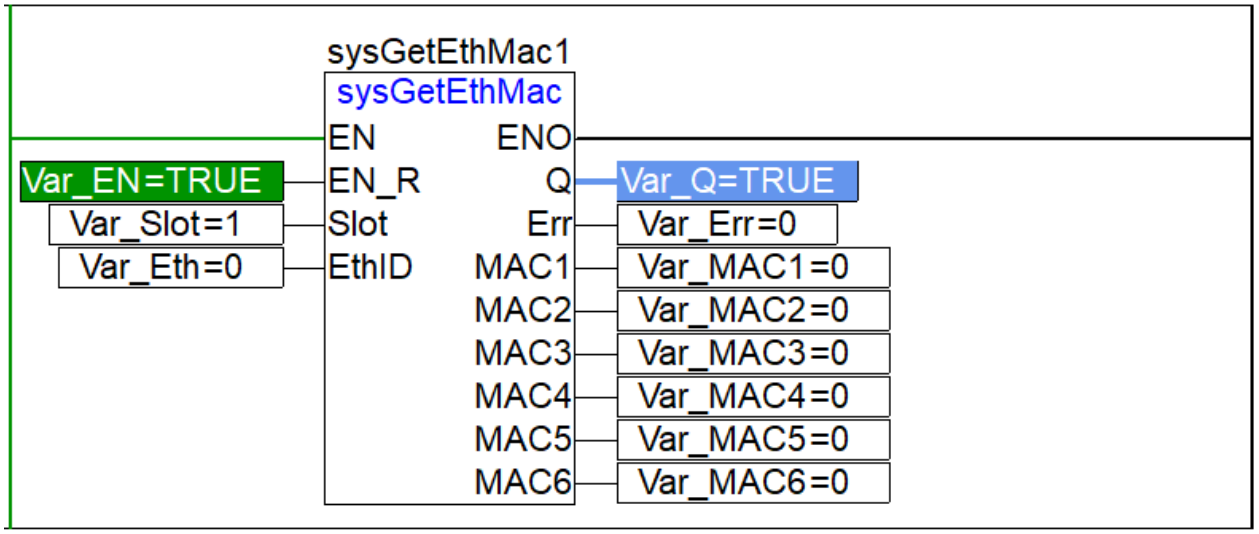
7.5.5.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN_R	Input	BOOL	Enable FALSE: Invalid (default value) TRUE: Valid on rising edge	FALSE	FALSE
Slot	Input	BYTE	Slot number 1: Local machine 2-5: CP card in slot 2-5	1	FALSE
EthID	Input	BYTE	Network card ID number (default value: 0, valid value: -1) 0: First network card 1: Second network card	0	FALSE
Q	Output	BOOL	Output (FALSE by default)	FALSE	FALSE
Err	Output	BYTE	Error (default value 0) 1: EthID configuration error 128: Failed to get MAC 200: The module is repeatedly defined and called	0	FALSE
MAC1	Output	BYTE	Part 1 of MAC address	0	FALSE
MAC2	Output	BYTE	Part 2 in MAC address	0	FALSE
MAC3	Output	BYTE	Part 3 in MAC address	0	FALSE
MAC4	Output	BYTE	Part 4 in MAC address	0	FALSE
MAC5	Output	BYTE	Part 5 in MAC address	0	FALSE
MAC6	Output	BYTE	Part 6 in MAC address	0	FALSE

7.5.5.3 Example

The following example uses the sysGetEthMac instruction to obtain the MAC address.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	sysGetEthMac1			SYSGETETHMAC		FALSE	Not Public	
0002	Var_EN			BOOL	TRUE	FALSE	Not Public	
0003	Var_Slot			BYTE	1	FALSE	Not Public	
0004	Var_Eth			BYTE	0	FALSE	Not Public	
0005	Var_Q			BOOL	TRUE	FALSE	Not Public	
0006	Var_Err			BYTE	0	FALSE	Not Public	
0007	Var_MAC1			BYTE	0	FALSE	Not Public	
0008	Var_MAC2			BYTE	0	FALSE	Not Public	
0009	Var_MAC3			BYTE	0	FALSE	Not Public	
0010	Var_MAC4			BYTE	0	FALSE	Not Public	
0011	Var_MAC5			BYTE	0	FALSE	Not Public	
0012	Var_MAC6			BYTE	0	FALSE	Not Public	




```
1   sysGetEthMac1(
2   EN_R := Var_EN,
3   Slot := Var_Slot,
4   EthID := Var_Eth,
5   Q=>Var_Q,
6   Err=>Var_Err,
7   MAC1=>Var_MAC1,
8   MAC2=>Var_MAC2,
9   MAC3=>Var_MAC3,
10  MAC4=>Var_MAC4,
11  MAC5=>Var_MAC5,
12  MAC6=>Var_MAC6);
```

```
sysGetEthMac1
Var_EN = TRUE
Var_Slot = 1
Var_Eth = 0
Var_Q = TRUE
Var_Err = 0
Var_MAC1 = 0
Var_MAC2 = 0
Var_MAC3 = 0
Var_MAC4 = 0
Var_MAC5 = 0
Var_MAC6 = 0
```

7.5.6 sysSetEthIp (Set Eth IP Address)

7.5.6.1 Version

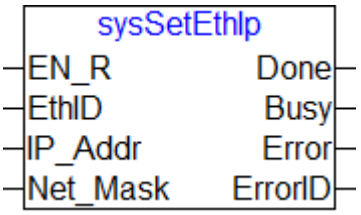
- AutoThink software version: V3.2.4 and above.
- Controller firmware version is as follows:

Controller Model	Firmware Version
LX-CU430	LX-CU430_V1.2.0 and above.
LX-CU433	LX-CU433_V1.1.0 and above.
LX-CU500	LX-CU500_V1.4.0 and above.
LX-CU501	LX-CU501_V1.2.0 and above.
LX-CU510	LX-CU510_V1.1.0 and above.
LX-CU511	LX-CU511_V1.1.0 and above.

7.5.6.2 Function

This function block is used to set the controller IP address.

The setting takes effect immediately after success.



sysSetEthIp function block

7.5.6.3 Parameters

Parameter	Declaration	Data Type	Description	Initial Value	Retain
EN_R	INPUT	BOOL	Rising edge trigger	FALSE	FALSE
EthID	INPUT	BYTE	Network port number to set the IP address for: 0: P1 port 1: P2 port	0	FALSE
IP_Addr	INPUT	ADDRMASK	IP address	-	FALSE
Net_Mask	INPUT	ADDRMASK	Subnet mask	-	FALSE
Done	OUTPUT	BOOL	IP address set successfully	FALSE	FALSE
Busy	OUTPUT	BOOL	IP address setting in progress	FALSE	FALSE
Error	OUTPUT	BOOL	IP address setting error	FALSE	FALSE
ErrorID	OUTPUT	DWORD	Error code: 0: No error 2: Network port IP address conflict 32769: Network card ID out of range 32770: Input IP address invalid 32771: Input subnet mask invalid	0	FALSE

7.5.6.4 Example

Take setting the IP address of the controller network port 1 (P1 port) to 192.168.0.201 as an example.

1. Variable Definition

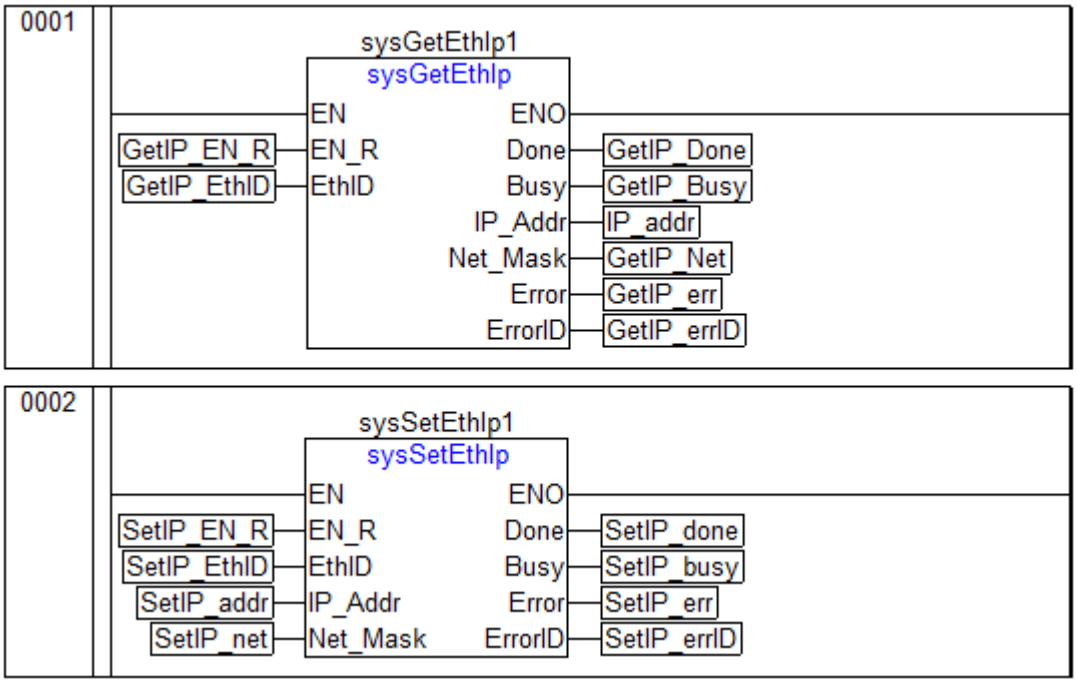
- Get IP Address

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护	网络公开	是否常量
0001	sysGetEthIp1			SYSGETETHIP		FALSE	不公开	
0002	GetIP_EN_R			BOOL	FALSE	FALSE	不公开	
0003	GetIP_EthID			BYTE	0	FALSE	不公开	
0004	GetIP_Done			BOOL	FALSE	FALSE	不公开	
0005	GetIP_Busy			BOOL	FALSE	FALSE	不公开	
0006	IP_addr			ADDRMASK		FALSE	不公开	
0007	GetIP_Net			ADDRMASK		FALSE	不公开	
0008	GetIP_err			BOOL	FALSE	FALSE	不公开	
0009	GetIP_errID			WORD	0	FALSE	不公开	

- Set IP Address

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护	网络公开	是否常量
0010	sysSetEthIp1			SYSSETETHIP		FALSE	不公开	
0011	SetIP_EN_R			BOOL	FALSE	FALSE	不公开	
0012	SetIP_EthID			BYTE	0	FALSE	不公开	
0013	SetIP_addr			ADDRMASK		FALSE	不公开	
0014	SetIP_net			ADDRMASK		FALSE	不公开	
0015	SetIP_done			BOOL	FALSE	FALSE	不公开	
0016	SetIP_busy			BOOL	FALSE	FALSE	不公开	
0017	SetIP_err			BOOL	FALSE	FALSE	不公开	
0018	SetIP_errID			WORD	0	FALSE	不公开	

2. LD Configuration Example

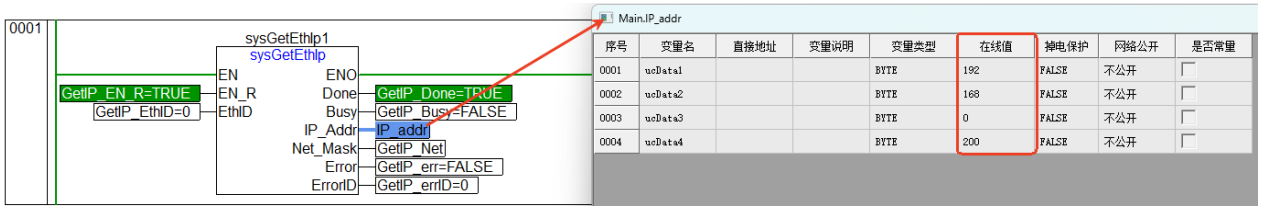


3. ST Configuration Example

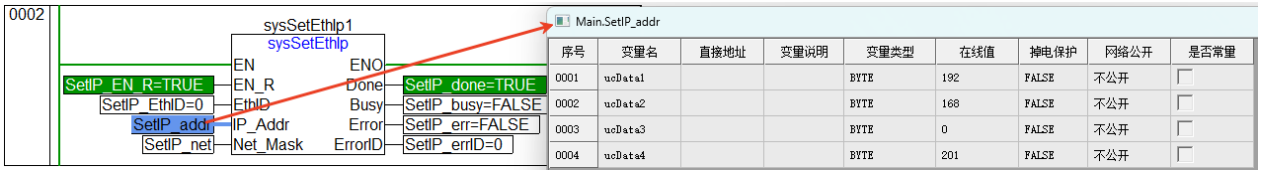
```
sysGetEthIp1(
    EN_R :=GetIP_EN_R ,
    EthID :=GetIP_EthID ,
    Done=>GetIP_Done,
    Busy=>GetIP_Busy,
    IP_Addr=>IP_addr,
    Net_Mask=>GetIP_Net,
    Error=>GetIP_err,
    ErrorID=>GetIP_errID);
sysSetEthIp1(
    EN_R :=SetIP_EN_R ,
    EthID :=SetIP_EthID ,
    IP_Addr :=SetIP_addr ,
    Net_Mask :=SetIP_net ,
    Done=>SetIP_done,
    Busy=>SetIP_busy,
    Error=>SetIP_err,
    ErrorID=>SetIP_errID);
```

4. Taking the LD program as an example, illustrate setting and getting the controller IP address.

(1) Enable the sysGetEthIp function block, set EthID to 0. When Done changes from FALSE to TRUE, the IP address (192.168.0.200) of the controller network port 1 is successfully retrieved.

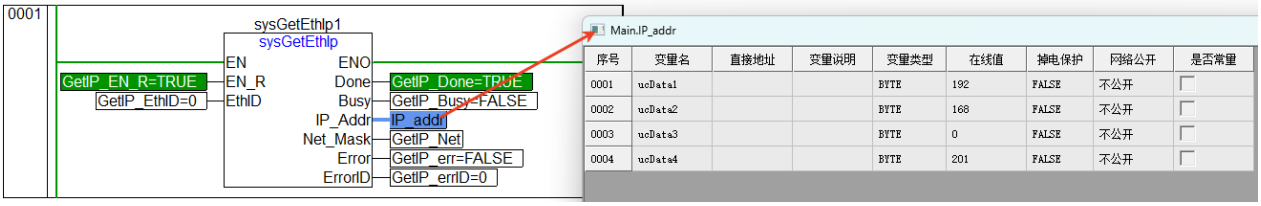


(2) Enable the sysSetEthIp function block, set EthID to 0, set the IP address (IP_Addr) to 192.168.0.201. A message indicating successful online value write will be displayed in the "Communication Information" window, and the IP address of the controller network port 1 is set successfully.



(3) After the IP address is set successfully, the controller will automatically exit the monitoring state. To reconnect to the controller, you need to enter the modified IP address in "Online - Communication Settings" on the menu bar.

(4) Reconnect to the controller and enable the sysGetEthIp function block again. When Done changes from FALSE to TRUE, the modified IP address of the controller network port 1 is retrieved.



7.5.7 sysGetEthIp (Get Eth IP Address)

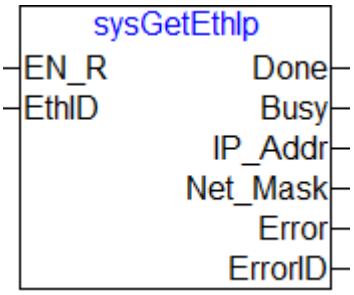
7.5.7.1 Version

- AutoThink software version: V3.2.4 and above.
- Controller firmware version is as follows:

Controller Model	Firmware Version
LX-CU430	LX-CU430_V1.2.0 and above.
LX-CU433	LX-CU433_V1.1.0 and above.
LX-CU500	LX-CU500_V1.4.0 and above.
LX-CU501	LX-CU501_V1.2.0 and above.
LX-CU510	LX-CU510_V1.1.0 and above.
LX-CU511	LX-CU511_V1.1.0 and above.

7.5.7.2 Function

This function block is used to get the controller IP address.



sysGetEthIp function block

7.5.7.3 Parameters

Parameter	Declaration	Data Type	Description	Initial Value	Retain
EN_R	INPUT	BOOL	Rising edge trigger	FALSE	FALSE
EthID	INPUT	BYTE	Network port number to get the IP address from: 0: P1 port 1: P2 port	0	FALSE
Done	OUTPUT	BOOL	IP address retrieval successful	FALSE	FALSE
Busy	OUTPUT	BOOL	IP address retrieval in progress	FALSE	FALSE
IP_Addr	OUTPUT	ADDRMASK	IP address	-	FALSE
Net_Mask	OUTPUT	ADDRMASK	Subnet mask	-	FALSE
Error	OUTPUT	BOOL	IP address retrieval error	FALSE	FALSE
ErrorID	OUTPUT	DWORD	Error code 0: No error 1: Failed to get IP address 32769: Network card ID out of range	0	FALSE

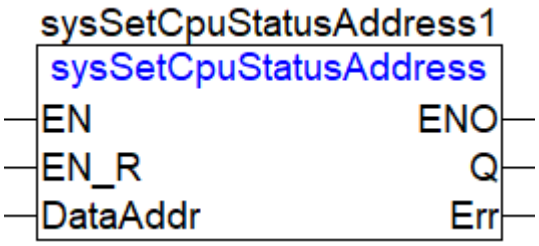
7.5.7.4 Example

Refer to sysSetEthIp (Set IP Address) Example Description.

7.5.8 sysSetCpuStatusAddress (Set CPU Diagnostic Status Address)

7.5.8.1 Function

Get CPU diagnostic data, including project running status and toggle switch status. When this instruction is used, the input parameter DataAddr points to an address used to store diagnostic data.



sysSetCpuStatusAddress Functional Block

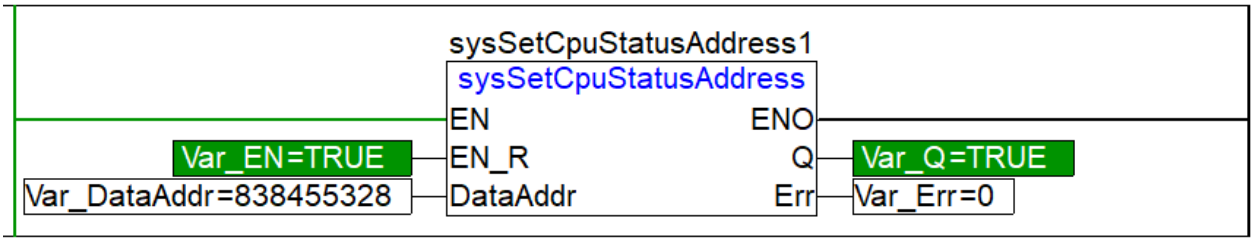
7.5.8.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN_R	Input	BOOL	Input (valid on rising edge)	FALSE	FALSE
DataAddr	Input	POINTER TO BYTE	Data area address, used to save CPU diagnostic information BYTE0: Project running status: 0: Unknown 1: RUN 2: STOP BYTE1: Controller operating mode switch status: 0: Unknown 1: RUN 2: STOP	0	FALSE
Q	Output	Q	Output (FALSE by default) TRUE: Output successful FALSE: No output	FALSE	FALSE
Err	Output	BYTE	Error 0: No error 1: The incoming address is empty, or the address is invalid (the address is not in the data area)	0	FALSE

7.5.8.3 Example

The following example uses the sysSetCpuStatusAddress instruction to set the CPU diagnostic address.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	sysSetCpuStatusAddress1			SYSSETCPUSTATUSADDRESS		FALSE	Not Public	
0002	Var_EN			BOOL	TRUE	FALSE	Not Public	
0003	Var_DataAddr			POINTER TO BYTE	838455328	FALSE	Not Public	
0004	Var_Q			BOOL	TRUE	FALSE	Not Public	
0005	Var_Err			USINT	0	FALSE	Not Public	



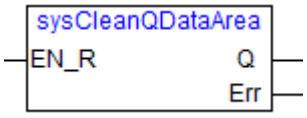

```
1  sysSetCpuStatusAddress1(
2  EN_R := Var_EN,
3  DataAddr := Var_DataAddr,
4  Q=>Var_Q,
5  Err=>Var_Err);
```

```
sysSetCpuStatusAddress1
Var_EN = TRUE
Var_DataAddr = 838455328
Var_Q = TRUE
Var_Err = 0
```

7.5.9 sysCleanQDataArea (Clearing Q data of area)

7.5.9.1 Function

Clear all data in Q area.



sysCleanQDataArea Functional Block

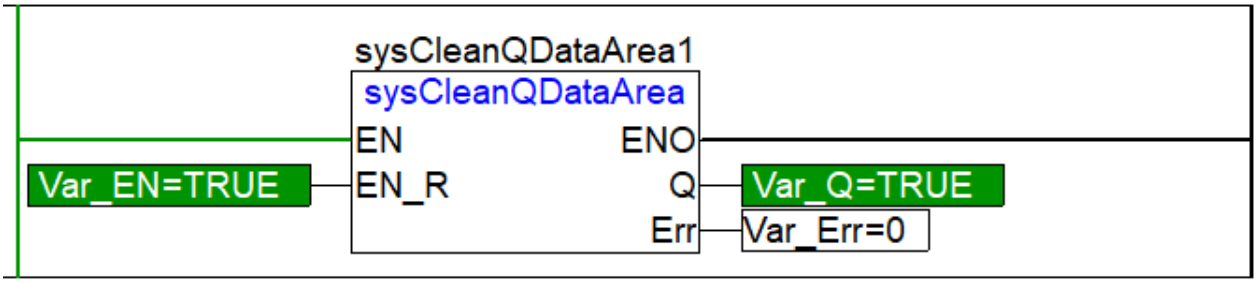
7.5.9.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN_R	Input	BOOL	Enabled, valid on rising edge	FALSE	FALSE
Q	Output	BOOL	Output (FALSE by default) TRUE: Output successful FALSE: No output	FALSE	FALSE
Err	Output	BYTE	Error (0 by default) 1: Failed to clear data in area Q	0	FALSE

7.5.9.3 Example

The following example uses the sysCleanQDataArea instruction to clear all data in area Q.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	sysCleanQDataArea1			SYSCLEANQDATAAREA		FALSE	Not Public	
0002	Var_EN			BOOL	TRUE	FALSE	Not Public	
0003	Var_Q			BOOL	TRUE	FALSE	Not Public	
0004	Var_Err			USINT	0	FALSE	Not Public	



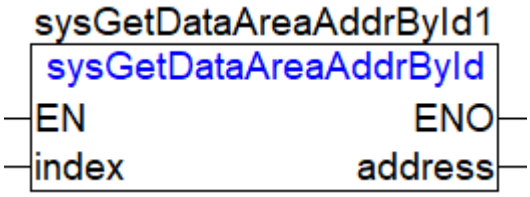

```
1  sysCleanQDataArea1(
2  EN_R := Var_EN,
3  Q=>Var_Q,
4  Err=>Var_Err);
```

```
sysCleanQDataArea1
Var_EN = TRUE
Var_Q = TRUE
Var_Err = 0
```

7.5.10 sysGetDataAreaAddrById (Get Address of data area)

7.5.10.1 Function

Use to get the start address of the data area based on the ID of the user data area (M, I, Q, R, G, S).



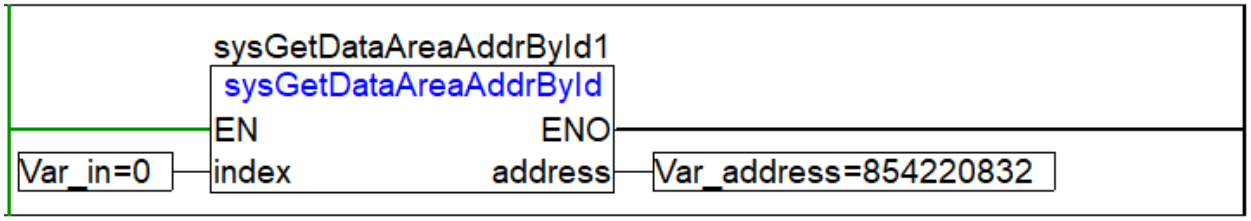
sysGetDataAreaAddrById Functional Block

7.5.10.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value
index	Input	BYTE	User data area ID: M (Middle Zone) ID: 0 I (input zone) ID: 1 Q (output zone) ID: 2 R (Holding Zone) ID: 3 G (random zone) ID: 4 S (special register zone) ID: 6	0
address	Output	POINTER TO BYTE	Get start address: Returns a pointer to the BYTE variable normally. Parameter error returned NULL.	0

7.5.10.3 Example

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	sysGetDataAreaAddrById1			SYSGETDATAAREAADDRBYID		FALSE	Not Public	
0002	Var_in			BYTE	0	FALSE	Not Public	
0003	Var_address			POINTER TO BYTE	854220832	FALSE	Not Public	




```
1  sysGetDataAreaAddrById1(
2  index := Var_in,
3  address=>Var_address);
```

```
sysGetDataAreaAddrById1
Var_in = 0
Var_address = 854220832
```

7.6 Memory Operation Instruction

7.6.1 DataMemCmp (Compare the values of Two different memory areas variables)

7.6.1.1 Function

Compare data variables in two different memory areas in bytes.

The instruction first gets the memory addresses of the source data and destination data, and after the conditions are met, it compares the variable values of certain lengths in the two different memory areas.

7.6.1.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
pDst	Input	POINTER TO BYTE	Valid address of data area	0	FALSE
pSrc	Input	POINTER TO BYTE	Valid address of data area	0	FALSE
Length	Input	DWOED	In 1 byte	0	FALSE
DataMemCmp	Output	BYTE	0: Succeeded, with pDst data equal to pSrc data 1: Source address error 2: Destination address error 3: Data length input 0 4: Source address + data length, access	0	FALSE

			out-of-range error 5: Destination address + data length, access out-of-range error 6: Success, with pDst data greater than pSrc data 7: Success, with pDst data lesser than pSrc data		

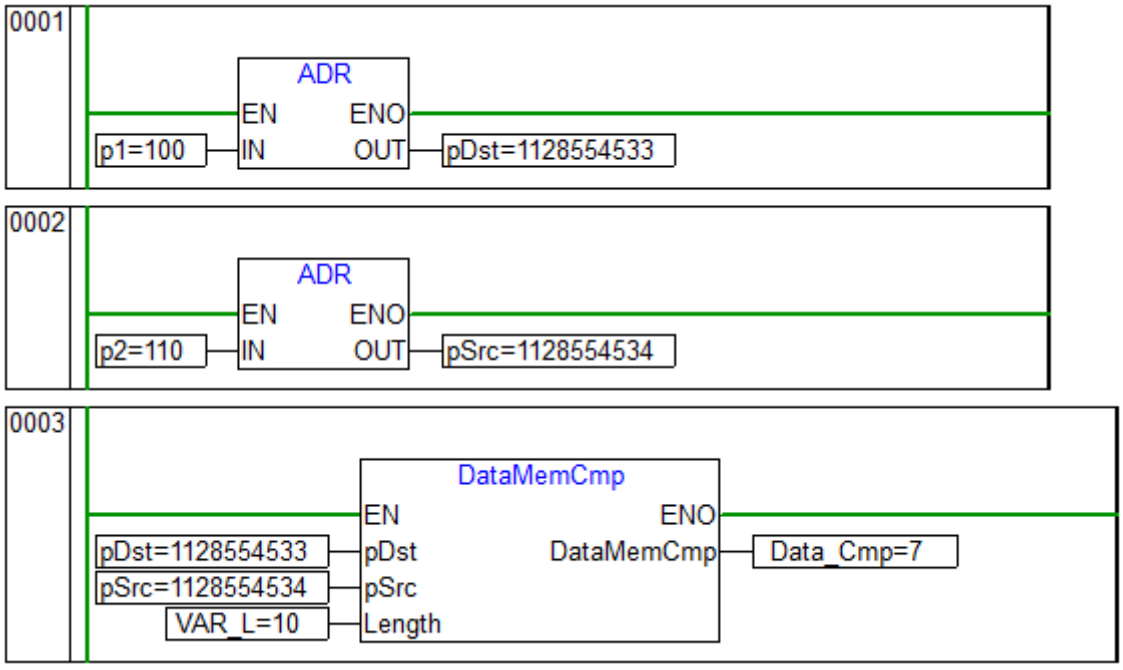
7.6.1.3 Example

The following example uses the DataMemCmp instruction to copy the memory of the data area. When the DataMemCmp instruction is started, it compares the first Length bytes of the storage areas pDst and pSrc one by one, and feeds the comparison results back to the return value of Output.

(1) Variable definition:

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	p1			BYTE	100	FALSE	Not Public	
0002	pDst			POINTER TO BYTE	1128554533	FALSE	Not Public	
0003	p2			BYTE	110	FALSE	Not Public	
0004	pSrc			POINTER TO BYTE	1128554534	FALSE	Not Public	
0005	VAR_L			DWORD	10	FALSE	Not Public	
0006	Data_Cmp			BYTE	7	FALSE	Not Public	

(2) LD program implementation:



(3) ST program implementation:

```
1   pDst := ADR(p1);                                  pDst = 1128554536   p1 = 100
2   pSrc := ADR(p2);                                  pSrc = 1128554537   p2 = 110
3   Data_Cmp := DataMemCmp(pDst, pSrc, VAR_L);        Data_Cmp = 7        pDst = 1128554536   pSrc = 1128554537   VAR_L = 10
```

7.6.1.4 Precautions for use

- The data length shall not be 0.
- The data length must be less than or equal to the size of the source data and destination data pointed to by the pointer.

7.6.2 DataMemCpy (Copy Data from Source Address to Destination Address)

7.6.2.1 Function

Copy data from source address to destination address in bytes.

The instruction first gets the memory addresses of the source data and destination data, and after the conditions are met, it transfers the data of a certain length starting from the source data memory address to the destination memory address segment.

- When the length is set too large, data may be stored in an unpredictable location, leading to data corruption and a risk of incorrect output.

7.6.2.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
pDst	Input	POINTER TO BYTE	Valid address of data area	0	FALSE
pSrc	Input	POINTER TO BYTE	Valid address of data area	0	FALSE
Length	Input	DWORD	In 1 byte	0	FALSE
DataMemCpy	Output	BYTE	0: Succeeded 1: Source address error 2: Destination address error 3: Data length input 0 4: Source address + data length, access out-of-range error 5: Destination address + data length, access out-of-range error	0	FALSE

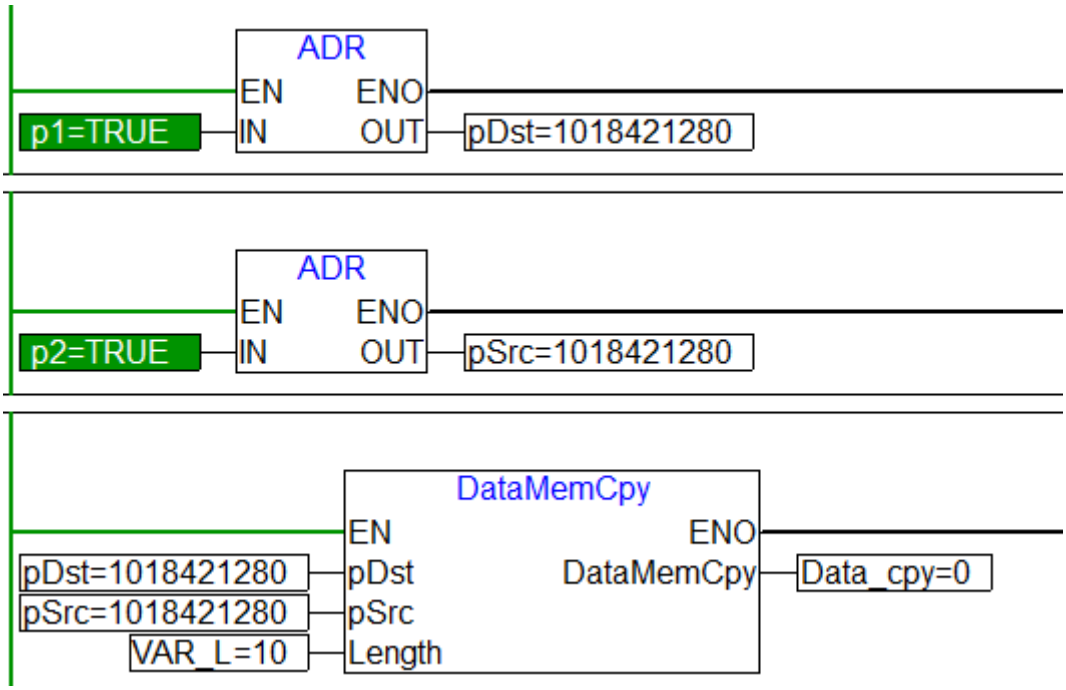
7.6.2.3 Example

The following example uses the DataMemCpy instruction to copy the memory of the data area. When the DataMemCpy instruction is started, it copies the Length data length from the address starting from the pSrc memory to the pDst memory address segment, and feeds the copy results back to the return value of Output.

(1) Variable definition:

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	p1			BOOL	TRUE	FALSE	Not Public	
0002	pDst			POINTER TO BYTE	1018421280	FALSE	Not Public	
0003	p2			BOOL	TRUE	FALSE	Not Public	
0004	pSrc			POINTER TO BYTE	1018421280	FALSE	Not Public	
0005	VAR_L			DWORD	10	FALSE	Not Public	
0006	Data_cpy			BYTE	0	FALSE	Not Public	

(2) LD program implementation:



(3) ST program implementation:

```
2
3   pDst := ADR(p1);
4   pSrc := ADR(p2);
5   Data_cpy := DataMemCpy(pDst, pSrc, VAR_L);
```

pDst = 1018421280
pSrc = 1018421280
Data_cpy = 0

p1 = TRUE
p2 = TRUE
pDst = 1018421280 pSrc = 1018421280 VAR_L = 10

7.6.2.4 Precautions for Use

- The data length shall not be 0.
- The data length must be less than or equal to the size of the source data and destination data pointed

to by the pointer.

- Memory overlap is not supported.

7.6.3 DataMemMove (Copy Data from Source Address to Destination Address)

7.6.3.1 Function

Copy data from source address to destination address in bytes.

The instruction first gets the memory addresses of the source data and destination data, and after the conditions are met, it transfers the data of a certain length starting from the source data memory address to the destination memory address segment.

- When the length is set too large, data may be stored in an unpredictable location, leading to data corruption and a risk of incorrect output.

7.6.3.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
pDst	Input	POINTER TO BYTE	Valid address of data area	0	FALSE
pSrc	Input	POINTER TO BYTE	Valid address of data area	0	FALSE
Length	Input	DWOED	In 1 byte	0	FALSE
DataMemMove	Output	BYTE	0: Succeeded 1: Source address error 2: Destination address error 3: Data length input 0 4: Source address + data length, access out-of-range error 5: Destination address + data length, access out-of-range error	0	FALSE

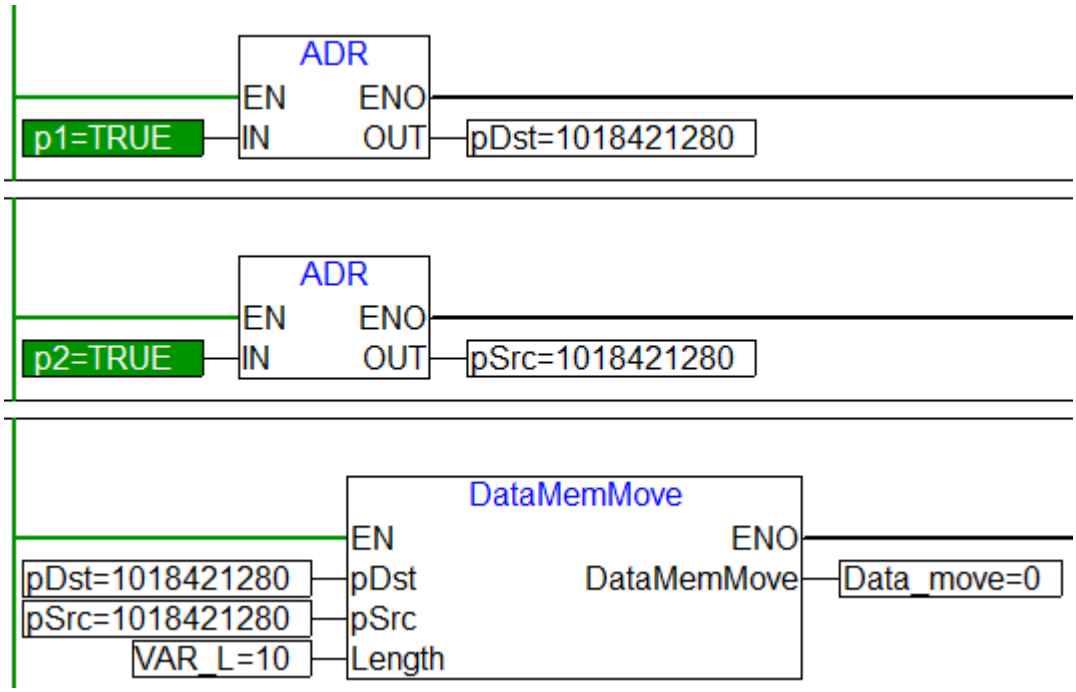
7.6.3.3 Example

The following example uses the DataMemMove instruction to copy the memory of the data area. When the DataMemMove instruction is started, it copies the Length data length from the address starting from the pSrc memory to the pDst memory address segment, and feeds the copy results back to the return value Output.

(1) Variable definition:

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	p1			BOOL	TRUE	FALSE	Not Public	
0002	pDst			POINTER TO BYTE	1018421280	FALSE	Not Public	
0003	p2			BOOL	TRUE	FALSE	Not Public	
0004	pSrc			POINTER TO BYTE	1018421280	FALSE	Not Public	
0005	VAR_L			DWORD	10	FALSE	Not Public	
0006	Data_move			BYTE	0	FALSE	Not Public	

(2) LD program implementation:



(3) ST program implementation:

```
3  pDst := ADR(p1);
4  pSrc := ADR(p2);
5  Data_move := DataMemMove(pDst, pSrc, VAR_L);
```

pDst = 1018421280
pSrc = 1018421280
Data_move = 0

p1 = TRUE
p2 = TRUE
pDst = 1018421280
pSrc = 1018421280
VAR_L = 10

7.6.3.4 Precautions for Use

- The data length shall not be 0.
- The data length must be lesser than or equal to the size of the source data and destination data pointed to by the pointer.
- Memory overlap is supported.

7.6.4 DataMemSet (Set Data from Source Address to Destination Address)

7.6.4.1 Function

It allows to set a variable in a specific memory area to a specific value in bytes.

The instruction first gets the memory addresses of the source data and destination data, and after the conditions are met, it copies a certain length of the specified value in bytes to the destination memory address segment.

7.6.4.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
pDst	Input	POINTER TO BYTE	Valid address of data area	0	FALSE
Value	Input	BYTE	Set the specified data value	0	FALSE
Length	Input	DWOED	In 1 byte	0	FALSE
DataMemSet	Output	BYTE	0: Succeeded 2: Destination address error 3: Data length input 0 5: Destination address + data length, access out-of-range error	0	FALSE

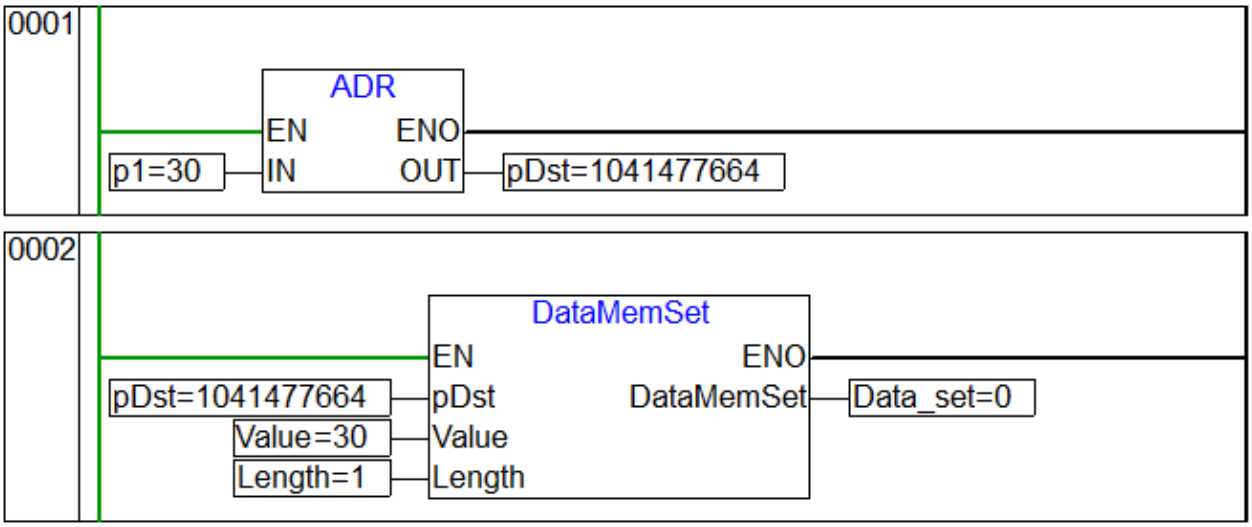
7.6.4.3 Example

The following example uses the DataMemSet instruction to implement specified values in the data area. When the DataMemSet instruction is started, it replaces the first Length bytes of the pDst address with Value, and feeds the copy results back to the return value of Output.

(1) Variable definition:

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	p1			BYTE	30	FALSE	Not Public	
0002	pDst			POINTER TO BYTE	1041477664	FALSE	Not Public	
0003	Value			BYTE	30	FALSE	Not Public	
0004	Length			DWORD	1	FALSE	Not Public	
0005	Data_set			BYTE	0	FALSE	Not Public	

(2) LD program implementation:



(3) ST program implementation:

```
1  pDst := ADR(p1);
2  Data_set := DataMemSet(pDst, Value, Length);
```

pDst = 1041477664
Data_set = 0

p1 = 30
pDst = 1041477664

Value = 30

Length = 1

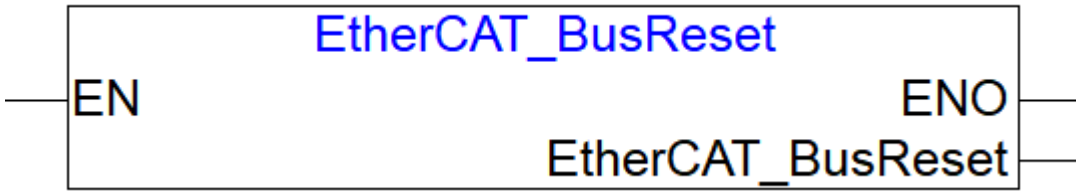
Chapter 8 Fieldbus Instruction

8.1 ETHERCAT Functions

8.1.1 EtherCAT_BusReset (Reset EtherCAT_bus)

8.1.1.1 Function

It is to reset the EtherCAT bus through an external library by first resetting the comX protocol board, then reconfiguring the slave station, initializing the protocol stack, and finally achieving the purpose of resetting the entire EtherCAT bus.



EtherCAT_BusReset Functional Block

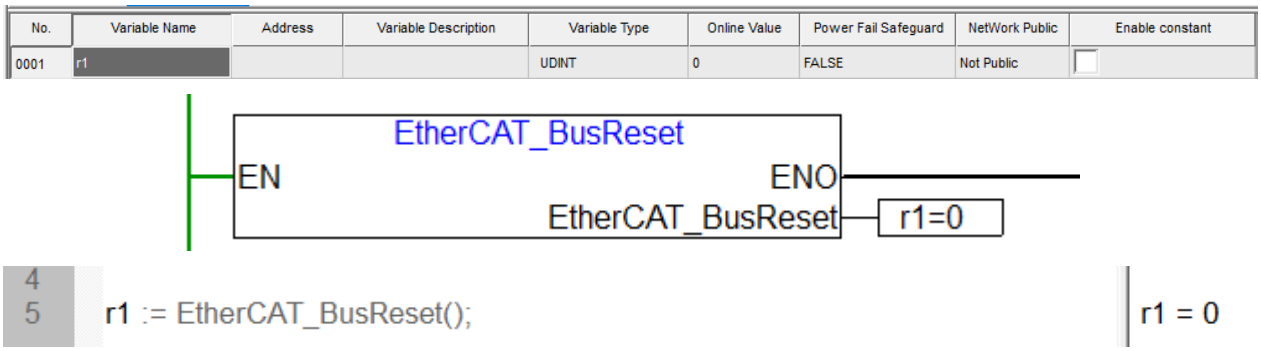
8.1.1.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
Input parameters	EN	BOOL	Rising edge trigger module reset function		
Output parameters	ENO	BOOL	/		
	EtherCAT_BusRest	UDINT	0: EtherCAT bus reset successfully 0x800A0001: Invalid comX communication channel (internal error, no attention required from users) 0x800C0011: Invalid comX communication channel status (internal error, no attention required from users) 0x800A000A: Invalid EtherCA configuration file (internal error, no attention required from users) 0x800A0004: Failed to load EtherCA configuration file (internal error, no attention required from users) 0x800A0019: Failed to reset comX communication channel 0x800C0020: comX communication channel	0	FALSE

			reset timeout		

8.1.1.3 Example

Reset EtherCAT bus.



8.1.2 EtherCAT_AxisFaultReset (Reset slave axis error)

8.1.2.1 Function

It is to clear the error code displayed by the EtherCAT slave station through an external function library by first clearing the error code displayed by the slave station through the EtherCAT protocol message, then initializing the slave station, and finally getting the slave station ready to enter the enabled status.

EtherCAT_AxisFaultReset Functional Block

8.1.2.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
Input parameters	EN	BOOL	Rising edge trigger module reset function		
	ucAxis	USINT	Axis number	0	FALSE
Output parameters	ENO	/	/		
	EtherCAT_ SlaveFaultReset	UDINT	0: Slave station error code cleared successfully 1: Failed to clear slave station error code	0	FALSE

The successful execution of clearing the slave error code means that the instruction is executed

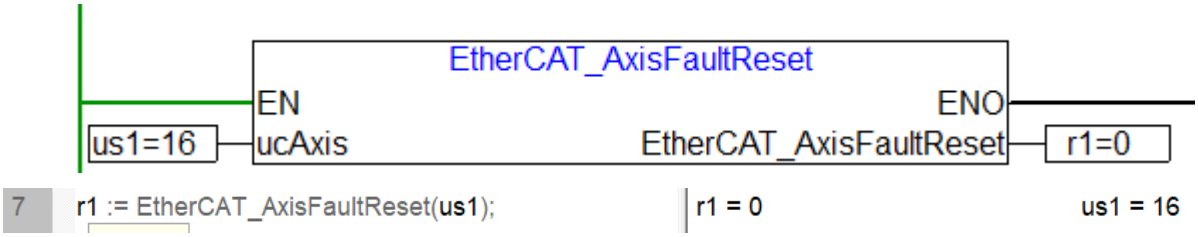
successfully, but it does not mean that the slave station error is cleared successfully. Some slave station errors cannot be cleared. Please refer to the slave station device manual for details.

The reason for the failure to clear the slave error code may be that the slave station number is incorrect or the EtherCAT bus communication is interrupted.

8.1.2.3 Example

The following example shows how to clear axis errors.

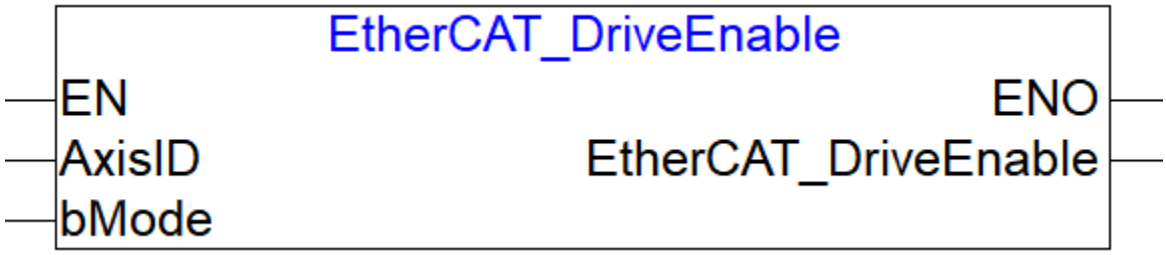
No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	us1			USINT	16	FALSE	Not Public	
0002	r1			UDINT	0	FALSE	Not Public	



8.1.3 EtherCAT_DriveEnable (Enable Drive)

8.1.3.1 Function

By turning on/off the servo enable switch, the enable status of each slave station can be controlled independently.



EtherCAT_DriveEnable Functional Block

8.1.3.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
Input	EN	BOOL	Rising edge trigger module reset function		
	AxisID	USINT	Axis number	0	FALSE
	bMODE	BOOL	Enable values	FALSE	FALSE

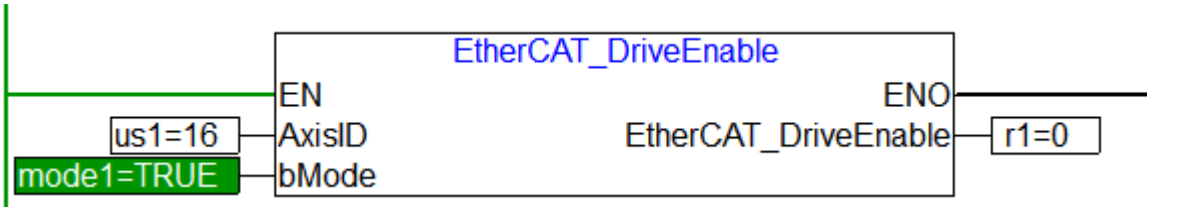
			1: Enable 0: Disabled (default)		
Output	ENO	/	/		
	EtherCAT_DriveEnable	UDINT	0: Set successfully 1: Set failed	0	FALSE

- Note:
  - Blocking instructions, non-axis motion cache instructions.
  - Only when the main servo enable switch is turned on, can the servo enable sub-switch be valid.

8.1.3.3 Example

The following example shows how to set the servo enable switch of axis 16#10 to On status.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	us1			USINT	16	FALSE	Not Public	
0002	mode1			BOOL	TRUE	FALSE	Not Public	
0003	r1			UDINT	0	FALSE	Not Public	

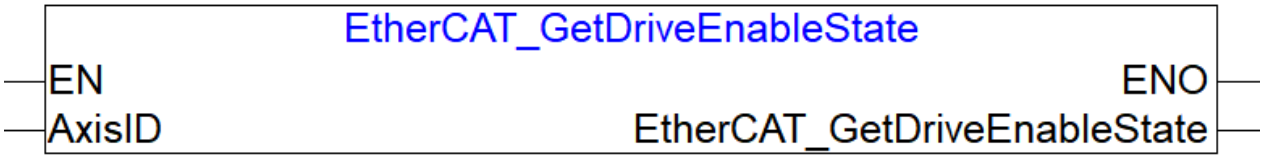



```
1  r1 := EtherCAT_DriveEnable(us1, mode1);   r1 = 0   us1 = 16   mode1 = TRUE
```

8.1.4 EtherCAT_GetDriveEnableState (Getting Enabled Drive State)

8.1.4.1 Function

It is to get the servo enable status of a slave station axis.



EtherCAT_GetDriveEnableState Functional Block

8.1.4.2 Parameter

Parameter	Statement	Data	Description	Initial	Power Fail

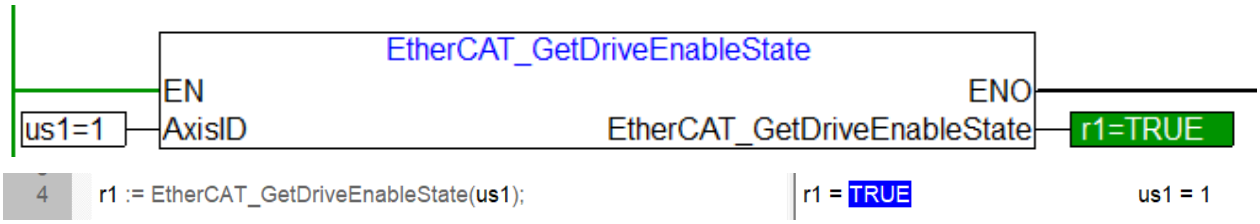
		Type		Value	Safeguard
Input	EN	BOOL	Rising edge trigger module reset function		
	AxisID	USINT	Axis number	0	FALSE
Output	ENO	/	/		
	EtherCAT_DriveEnableState	UDINT	0: Servo enable off 1: Servo enable on	FALSE	FALSE

- Note:
- Blocking instructions, non-axis motion cache instructions.
- Only when the main servo enable switch and the corresponding slave servo enable sub-switch are turned on at the same time, can the servo enable status of the slave station be On, otherwise it is Off.

8.1.4.3 Example

For example, get the enable status of axis xxx.

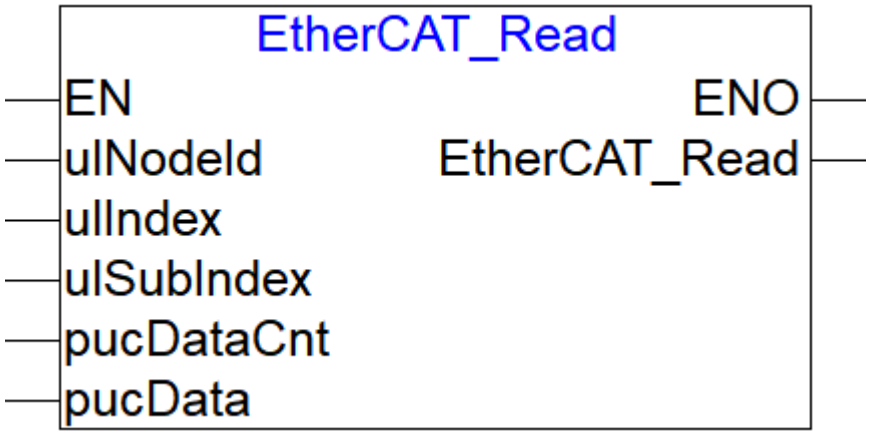
No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	us1			USINT	1	FALSE	Not Public	
0002	r1			BOOL	TRUE	FALSE	Not Public	



8.1.5 EtherCAT_Read (Read Slave Parameter)

8.1.5.1 Function

Read the parameter length and parameter content of the specified parameters of the specified slave station.



EtherCAT_Read Functional Block

8.1.5.2 Parameter

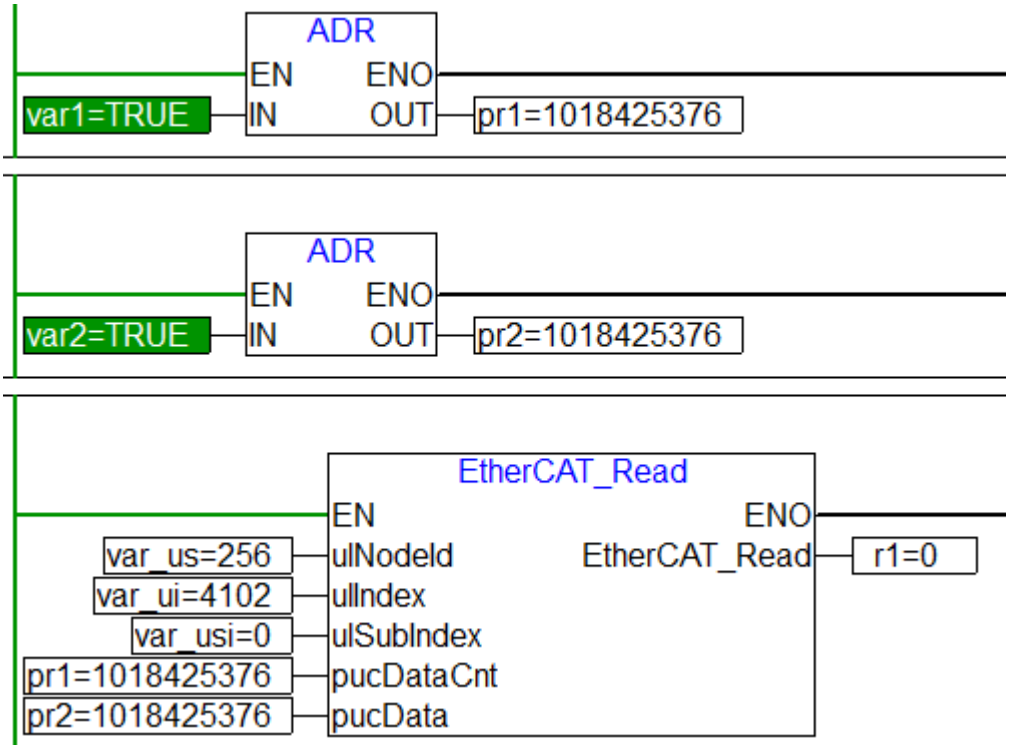
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
Input	EN	BOOL	Rising edge trigger module reset function		
	ulIndesID	UDINT	Slave station number	0	FALSE
	ulINdex	UDINT	Object index corresponding to the parameter	0	FALSE
	ulSubIndex	UDINT	Object sub-index corresponding to the parameter	0	FALSE
	pucDateCn t	POINTER TO BYTE	Pointer to parameter length	FALSE	FALSE
	pucData	POINTER TO BYTE	Pointer to parameter content	FALSE	FALSE
Output	ENO	/	/		
	EtherCAT_ Read	UDINT	0: Read successfully Others: For details, see Appendix 3: Error Codes of EtherCAT Read/Write Parameters	0	FALSE

- Note: To call the EtherCAT_Read instruction in an IEC periodic task, the task cycle must be set to be greater than the actual time of task execution.

8.1.5.3 Example

The following example shows how to read the device name of station 16#100.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	var1			BOOL	TRUE	FALSE	Not Public	
0002	pr1			POINTER TO BYTE	1018425376	FALSE	Not Public	
0003	var2			BOOL	TRUE	FALSE	Not Public	
0004	pr2			POINTER TO BYTE	1018425376	FALSE	Not Public	
0005	var_us			DWORD	256	FALSE	Not Public	
0006	var_ui			DWORD	4102	FALSE	Not Public	
0007	var_usi			DWORD	0	FALSE	Not Public	
0008	r1			DWORD	0	FALSE	Not Public	




```
2  pr1 := ADR(var1);
3  pr2 := ADR(var2);
4  r1 := EtherCAT_Read(var_us, var_ui, var_usi, pr1, pr2);
```

pr1 = 1018425376 pr2 = 1018425376 r1 = 0 var1 = TRUE var2 = TRUE var_us = 256 var_ui = 4102 var_usi = 0 pr1 = 1018425376 pr2 = 1018425376

8.1.6 EtherCAT_Write (Write Slave Parameter)

8.1.6.1 Function

It is to write the parameter content of the specified parameters of the specified slave station.

EtherCAT_Write Functional Block

8.1.6.2 Parameter

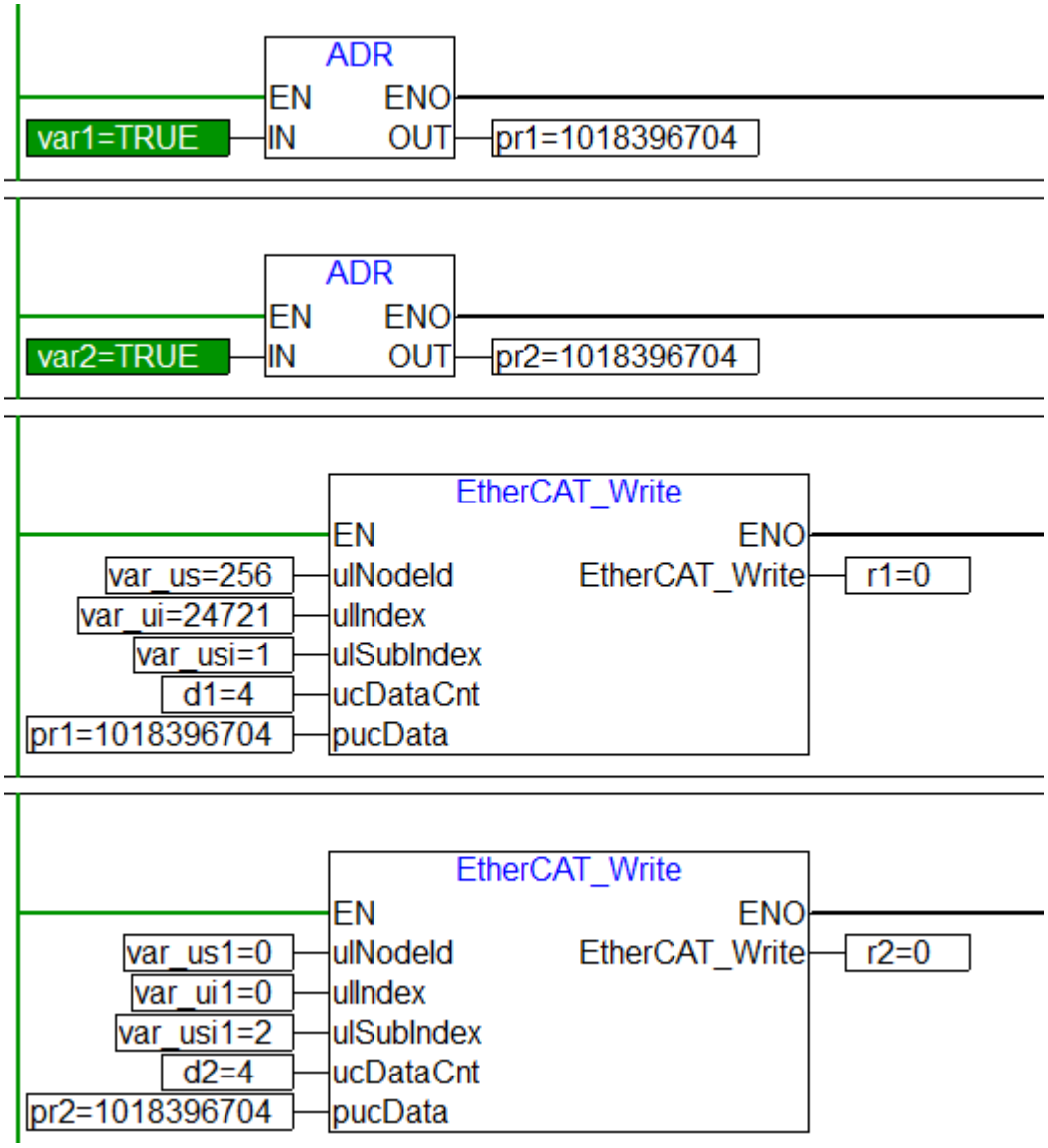
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
Input	EN	BOOL	Rising edge trigger module reset function		
	ulNodeId	UDINT	Slave station number	0	FALSE
	ulIndex	UDINT	Object index corresponding to the parameter	0	FALSE
	ulSubIndex	UDINT	Object sub-index corresponding to the parameter	0	FALSE
	ucDataCnt	ByTe	Length of the parameter to be written	FALSE	FALSE
	pucData	POINTER TO BYTE	Pointer to the content of the parameter to be written	FALSE	FALSE
Output	ENO	/	/		
	EtherCAT_Write	UDINT	0: Written successfully Others: For details, see Appendix: Error Codes of EtherCAT Read/Write Parameters	0	FALSE

- Note:
- The actual length of the parameter content to be written must be strictly consistent with the value of ucDataCnt.
- To call the EtherCAT_Write instruction in an IEC periodic task, the task cycle must be set to be greater than the actual time of task execution.

8.1.6.3 Example

The following example shows how to set the gear ratio of slave station 16#100 (Omron as an example).

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	var1			BOOL	TRUE	FALSE	Not Public	
0002	pr1			POINTER TO BYTE	1018396704	FALSE	Not Public	
0003	var2			BOOL	TRUE	FALSE	Not Public	
0004	pr2			POINTER TO BYTE	1018396704	FALSE	Not Public	
0005	var_us			DWORD	256	FALSE	Not Public	
0006	var_ui			DWORD	24721	FALSE	Not Public	
0007	var_usi			DWORD	1	FALSE	Not Public	
0008	var_pr1			BOOL	FALSE	FALSE	Not Public	
0009	r1			DWORD	0	FALSE	Not Public	
0010	d1			BYTE	4	FALSE	Not Public	
0011	var_us1			DWORD	0	FALSE	Not Public	
0012	var_ui1			DWORD	0	FALSE	Not Public	
0013	var_usi1			DWORD	2	FALSE	Not Public	
0014	d2			BYTE	4	FALSE	Not Public	
0015	r2			DWORD	0	FALSE	Not Public	

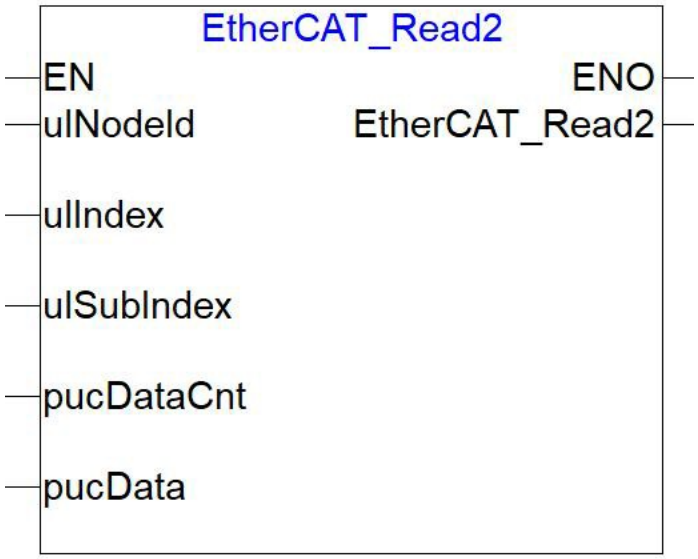


```
2   pr1 := ADR(var1);                                          pr1 = 1018396704    var1 = TRUE
3   pr2 := ADR(var2);                                          pr2 = 1018396704    var2 = TRUE
4   r1 := EtherCAT_Write(var_us, var_ui, var_usi, d1, pr1);    r1 = 0              var_us = 256    var_ui = 24721    var_usi = 1    d1 = 4    pr1 = 1018396704
5   r2 := EtherCAT_Write(var_us1, var_ui1, var_usi1, d2, pr2); r2 = 0              var_us1 = 0     var_ui1 = 0       var_usi1 = 2   d2 = 4    pr2 = 1018396704
```

8.1.7 EtherCAT_Read2 (Read Slave Parameter 2)

8.1.7.1 Function

It is to quickly read the parameter length and parameter content of the specified parameters of the specified slave station. The EtherCAT_Read2 instruction reads PDO process parameters faster than the EtherCAT_Read (read slave station parameters) instruction.



EtherCAT_Read Functional Block

8.1.7.2 Parameter

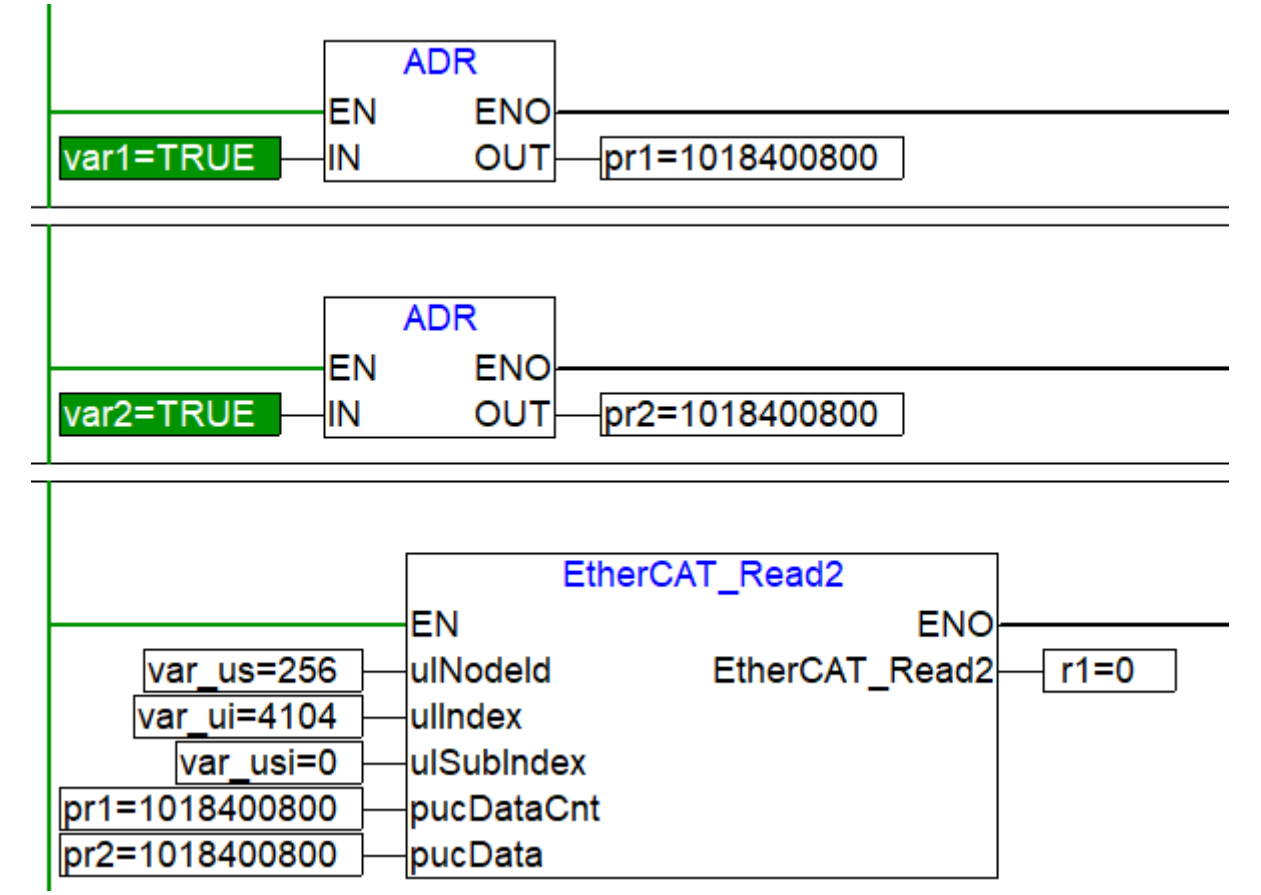
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
Input	EN	/	Rising edge trigger module reset function		
	ulNodeId	DWORD	Slave station number	0	FALSE
	ullNdex	DWORD	Object index corresponding to the parameter	0	FALSE
	ulSubIndex	DWORD	Object sub-index corresponding to the parameter	0	FALSE
	pucDateCnt	POINTER TO BYTE	Pointer to parameter length	FALSE	FALSE
	pucData	POINTER TO BYTE	Pointer to parameter content	FALSE	FALSE
Output	ENO	/	/		
	EtherCAT_ Read2	DWORD	0: Read successfully Others: For details, see Appendix: Error Codes of EtherCAT Read/Write Parameters	0	FALSE

- Note: To call the EtherCAT_Read instruction in an IEC periodic task, the task cycle must be set to be greater than the actual time of task execution.

8.1.7.3 Example

The following example shows how to read the device name of station 16#100 (Omron as an example).

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	var1			BOOL	FALSE	FALSE	Not Public	
0002	pr1			POINTER TO BYTE	1018400800	FALSE	Not Public	
0003	var2			BOOL	FALSE	FALSE	Not Public	
0004	pr2			POINTER TO BYTE	1018400800	FALSE	Not Public	
0005	var_us			DWORD	256	FALSE	Not Public	
0006	var_ui			DWORD	4104	FALSE	Not Public	
0007	var_usi			DWORD	0	FALSE	Not Public	
0008	r1			DWORD	0	FALSE	Not Public	



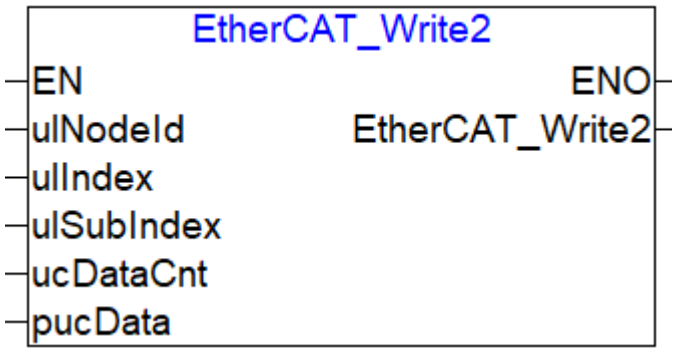

```
7  pr1 := ADR(var1);
8  pr2 := ADR(var2);
9  r1 := EtherCAT_Read2(var_us, var_ui, var_usi, pr1, pr2);
```

pr1 = 1018400800 pr2 = 1018400800 r1 = 0 var1 = TRUE var2 = TRUE var_us = 256 var_ui = 4104 var_usi = 0 pr1 = 1018400800 pr2 = 1018400800

8.1.8 EtherCAT_Write2 (Write Slave Parameter 2)

8.1.8.1 Function

It is to write the parameter content of the specified parameters of the specified slave station.



EtherCAT_Write2 Functional Block

8.1.8.2 Parameter

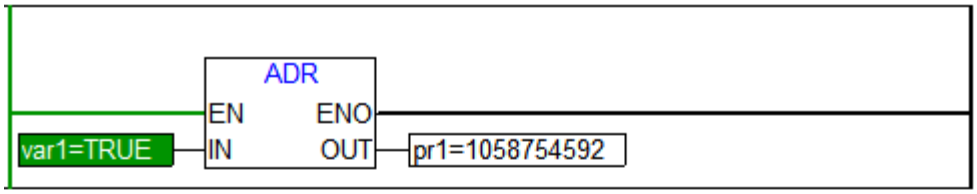
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
Input	EN	BOOL	Rising edge trigger module reset function		
	ulNodeId	UDINT	Slave station number	0	FALSE
	ulIndex	UDINT	Object index corresponding to the parameter	0	FALSE
	ulSubIndex	UDINT	Object sub-index corresponding to the parameter	0	FALSE
	ucDataCnt	ByTe	Length of the parameter to be written	FALSE	FALSE
	pucData	POINTER TO BYTE	Pointer to the content of the parameter to be written	FALSE	FALSE
Output	ENO	/	/		
	EtherCAT_Write2	UDINT	0: Written successfully Others: For details, see Appendix: Error Codes of EtherCAT Read/Write Parameters	0	FALSE

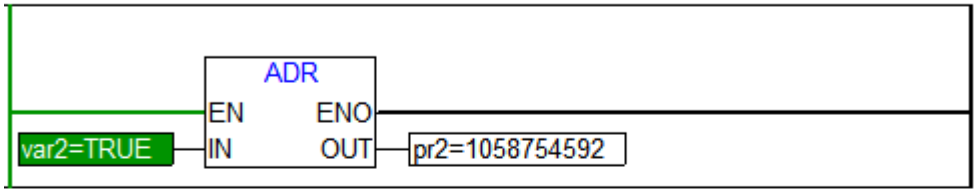
- Note:
- The actual length of the parameter content to be written must be strictly consistent with the value of ucDataCnt.
- To call the EtherCAT_Write instruction in an IEC periodic task, the task cycle must be set to be greater than the actual time of task execution.

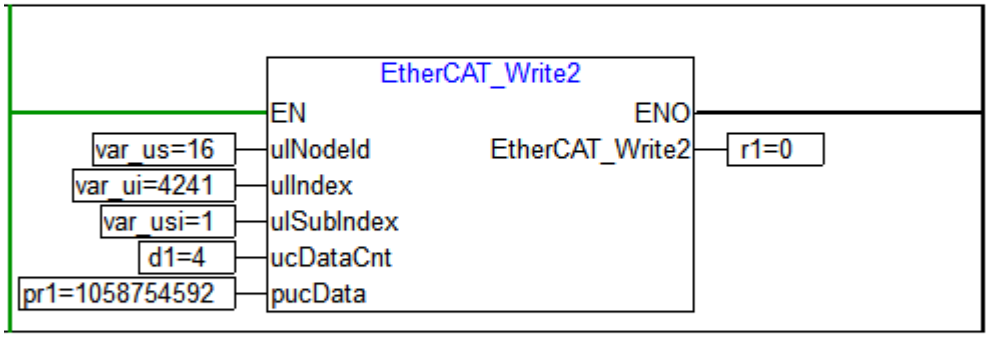
8.1.8.3 Example

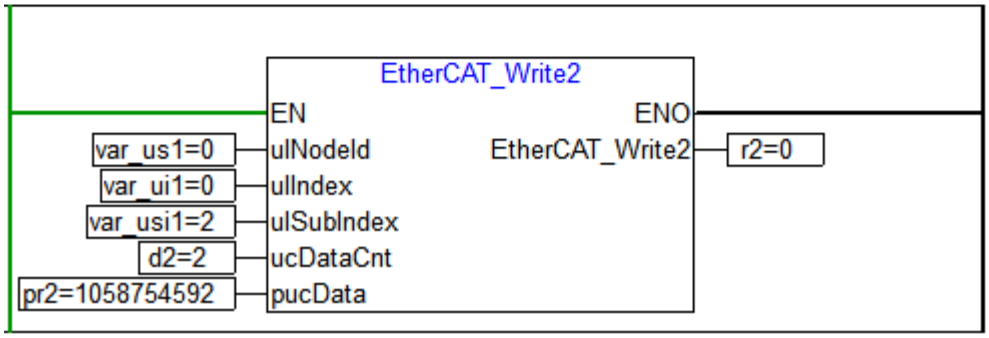
The following example shows how to set the gear ratio of slave station 16#10 (Omron as an example).

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	var1			BOOL	TRUE	FALSE	Not Public	
0002	var2			BOOL	TRUE	FALSE	Not Public	
0003	pr1			POINTER TO BYTE	1058754592	FALSE	Not Public	
0004	pr2			POINTER TO BYTE	1058754592	FALSE	Not Public	
0005	var_us			DWORD	16	FALSE	Not Public	
0006	var_ui			DWORD	4241	FALSE	Not Public	
0007	var_usi			DWORD	1	FALSE	Not Public	
0008	d1			BYTE	4	FALSE	Not Public	
0009	r1			DWORD	0	FALSE	Not Public	
0010	var_us1			DWORD	0	FALSE	Not Public	
0011	var_ui1			DWORD	0	FALSE	Not Public	
0012	var_usi1			DWORD	2	FALSE	Not Public	
0013	d2			BYTE	2	FALSE	Not Public	
0014	r2			DWORD	0	FALSE	Not Public	










```
1  pr1 := ADR(var1);
2  pr1 := ADR(var2);
3  r1 := EtherCAT_Write2(var_us, var_ui, var_usi, d1, pr1);
4  r2 := EtherCAT_Write2(var_us1, var_ui1, var_usi1, d2, pr2);
```

```
pr1 = 1058754592    var1 = TRUE
pr1 = 1058754592    var2 = TRUE
r1 = 0              var_us = 16     var_ui = 4241    var_usi = 1    d1 = 4    pr1 = 1058754592
r2 = 0              var_us1 = 0     var_ui1 = 0      var_usi1 = 2   d2 = 2    pr2 = 1058754592
```

8.1.9 EtherCAT_GetSlaveInOutOffset (Get Slave Input and Output Start Offset)

8.1.9.1 Function

By using the "Get Slave Input/Output" instruction, you can obtain the first address of the channel assigned to I and Q of the specified EtherCAT slave station. The instruction is a function that returns a result each time it is called. When the execution is successful, the return value is 1, and when the execution fails, the return value is 0.

When the execution is successful, if the slave station actually has channels allocated to areas I and Q, the parameter returns the first address of areas I and Q. If the slave station only has channels to area I or Q, another parameter returns 0.

When the input slave station address does not exist, the execution fails and the parameter return address is 0xFFFFFFFF.

EtherCAT_GetSlaveInOutOffset Functional Block

8.1.9.2 Parameter

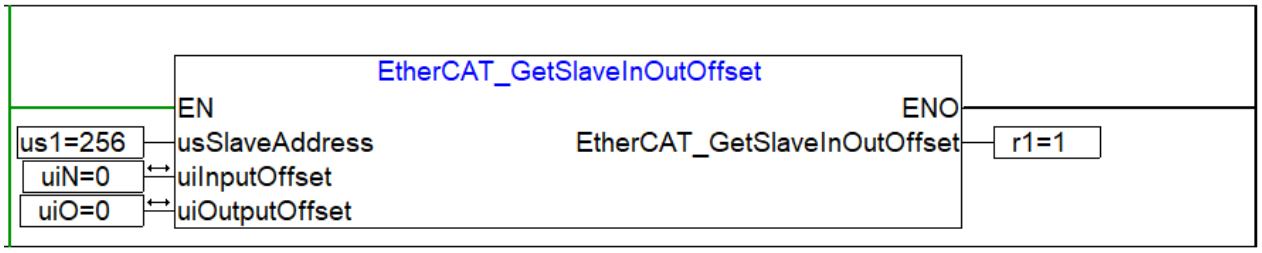
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
Input	EN	BOOL	Rising edge trigger module reset function		
	usSlaveAddress	UINT	Slave station number setting	16#100	FALSE
	uinputOffset	Dword	Input position	0	FALSE
	uiOutputOffset	Dword	Output location	0	FALSE
Output	ENO	/	/		
	EtherCAT_ GetSlaveInOutOffset	Dword	When the execution is successful, the return value is 1; When the execution fails, the return value is 0.	0	FALSE

- Note that the slave station address shall exist in the actual configuration.

8.1.9.3 Example

The EtherCAT_GetSlaveInOutOffset instruction can be used to get the address of areas I and Q of the specified slave station. When this instruction is called, the usSlaveAddress pin passes in the station address of the slave station. After successful execution, the first address of the channel assigned by the slave station to areas I and Q is stored in the uiInputOffset and uiOutputOffset pins.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	us1			UINT	256	FALSE	Not Public	
0002	uiN			DWORD	0	FALSE	Not Public	
0003	uiO			DWORD	0	FALSE	Not Public	
0004	r1			DWORD	1	FALSE	Not Public	




```
1
2    r1 := EtherCAT_GetSlaveInOutOffset(us1, uin, uio);
```

8.1.10 EtherCAT_SetAxisOperationMode (Set the Axis Mode of Operation)

8.1.10.1 Function

It is to set the Mode of Operation (control mode) of the specified EtherCAT slave station. The input is the specified axis number and the control mode value to be set. The axis number must be consistent with the actual configured axis number when setting. Setting control mode means setting the value of 0x6060 in the object dictionary of the EtherCAT slave station to configure the operating mode of the EtherCAT slave station. If the setting is successful, 0 is returned. If an error occurs, a non-0 value is returned. The error code is as follows:

Error code	Description
2498	Axis number error
2499	The slave station address corresponding to this axis is invalid
2500	Failed to set the slave station control mode through SDO
2501	An error occurred in communication with the slave station

8.1.10.2 Example

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
ucAxisID	Input	USINT	Axis number	0	FALSE
cMode	Input	SINT	Set control mode value	0	FALSE
Return value	Output	UDINT	0: Set successfully Non-0: Setting error	0	FALSE

8.1.10.3 Example

The following example uses the EtherCAT_SetAxisOperationMode and EtherCAT_SetAxisCtrWord instructions to set the control mode and control word of the slave station to control the EtherCAT slave station to run in synchronous speed mode. The target speed of the slave station can be modified in real time through the 0x60FF object dictionary. Set the control mode to 9 and define the control words as 6, 7, and 15 in order to enable the servo system. Then, modify the target speed variable E1004_CH_13 online, that is, the target speed object dictionary 0x60FF, to change the servo rotation speed in real time.

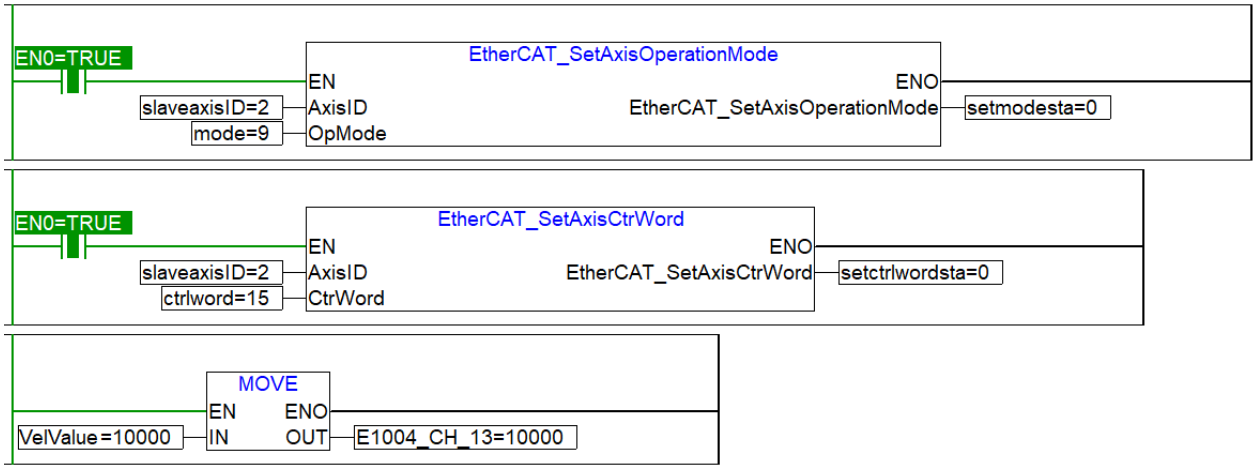
EtherCAT Slave Station Configuration | Process Parameter | Init Commands | EtherCAT IO Mapping | Information

Channel Number	Channel Name	Channel Type	Bytes	Channel Address	Online Value	Channel Description
1	E1003_IN_1	UINT	2	%IW4	0	Error code
2	E1003_IN_2	UINT	2	%IW6	5687	Statusword
3	E1003_IN_3	DINT	4	%ID8	1841185970	Position actual value
4	E1003_IN_4	INT	2	%IW12	-5	Torque actual value
5	E1003_IN_5	SINT	1	%IB14	9	Modes of operation display
6	E1003_IN_6	DINT	4	%ID16	0	Following error actual value
7	E1003_IN_7	UINT	2	%IW20	0	Touch probe status
8	E1003_IN_8	DINT	4	%ID24	0	Touch probe pos1 pos value
9	E1003_IN_9	DINT	4	%ID28	0	Touch probe pos2 pos value
10	E1003_IN_10	DINT	4	%ID32	-64000	Velocity actual value
11	E1003_OUT_11	UINT	2	%QW32	15	Controlword
12	E1003_OUT_12	DINT	4	%QD36	1840429949	Target position
13	E1003_OUT_13	DINT	4	%QD40	10000	Target velocity
14	E1003_OUT_14	INT	2	%QW44	0	Target torque
15	E1003_OUT_15	SINT	1	%QB46	9	Modes of operation
16	E1003_OUT_16	UINT	2	%QW48	0	Touch probe function
17	E1003_OUT_17	UDINT	4	%QD52	104857600	Max profile velocity

Variable definition:

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	EN0			BOOL	FALSE	FALSE	Not Public	
0002	slaveAxisID			USINT	2	FALSE	Not Public	
0003	mode			SINT	9	FALSE	Not Public	
0004	setmodesta			UDINT	0	FALSE	Not Public	
0005	ctrlword			UINT	15	FALSE	Not Public	
0006	setctrwordsta			UDINT	0	FALSE	Not Public	
0007	VelValue			INT	10000	FALSE	Not Public	
0008	E1004_CH_13			INT	10000	FALSE	Not Public	

LD program implementation:



ST program implementation:

```
1  IF EN0 THEN                                  EN0 = FALSE
2      Var_out := EtherCAT_SetAxisOperationMode(Var_ID, Var_mode);   Var_out = 0   Var_ID = 2   Var_mode = 9
3      EN0:=FALSE;                              EN0 = FALSE
4  END_IF
5
6  IF EN1 THEN                                  EN1 = TRUE
7      Var_out1 := EtherCAT_SetAxisCtrWord(Var_ID, Var_CW);   Var_out1 = 0   Var_ID = 2   Var_CW = 15
8      E1004_CH_13:=Var_in;                     E1004_CH_13 = 10000   Var_in = 10000
9  END_IF
```

8.1.11 EtherCAT_SetAxisCtrWord (Set the Axis Control Word)

8.1.11.1 Function

It is to set the control word of the specified EtherCAT slave station. The input is the specified axis number and the control word value to be set. The axis number must be consistent with the actual configured axis number when setting. Setting of axis control word means setting the value of 0x6040 in the object dictionary of the EtherCAT slave station to control the status machine transition of the EtherCAT slave station. If the setting is successful, 0 is returned, and if it fails, a non-0 value is returned. The error code is as follows:

Error code	Description
2502	Axis number error
2503	The slave station address corresponding to this axis is invalid
2504	Failed to set the slave station control word through sdo
2505	An error occurred in communication with the slave station

8.1.11.2 Example

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
ucAxisID	Input	USINT	Axis number	0	FALSE
cMode	Input	USINT	Set control word value	0	FALSE
Return value	Output	UDINT	0: Set successfully	0	FALSE

			Non-0: Setting error		

8.1.11.3 Example

See the example in EtherCAT_SetAxisOperationMode (Set the Axis Mode of Operation).

8.1.12 EtherCAT_GetSlaveODOffset (Get Slave Station OD Offset of IQ area)

8.1.12.1 Function

Get the IQ area offset of the slave station object dictionary.

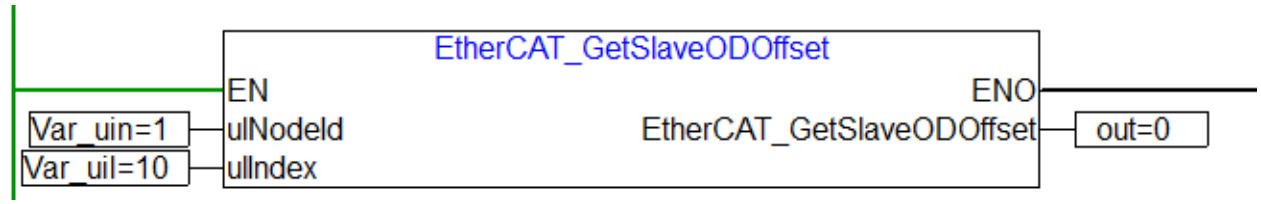
EtherCAT_GetSlaveODOffset Functional Block

8.1.12.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
ulNodeId	Input	DWORD	Slave station number	0	FALSE
ulIndex	Input	DWORD	Parameter index value	0	FALSE
EtherCAT_GetSlaveODOffset	Output	DWORD	Return Value: 0xFFFFFFFF: Indicates a failure. On success, returns the corresponding I/O area (Input or Output) offset address in the object dictionary.	0	FALSE

8.1.12.3 Example

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var_uin			DWORD	1	FALSE	Not Public	
0002	Var_uil			DWORD	10	FALSE	Not Public	
0003	out			DWORD	0	FALSE	Not Public	



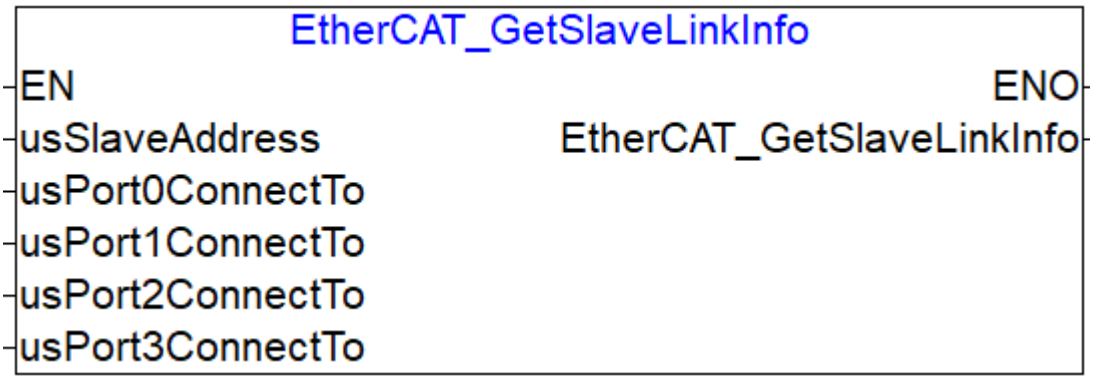

```
10
11  out := EtherCAT_GetSlaveODOffset(Var_uin, Var_uil);
```

out = 0 Var_uin = 1 Var_uil = 10

8.1.13 EtherCAT_GetSlaveLinkInfo (Get Slave link info)

8.1.13.1 Function

Get the slave station port link information.



EtherCAT_GetSlaveLinkInfo Functional Block

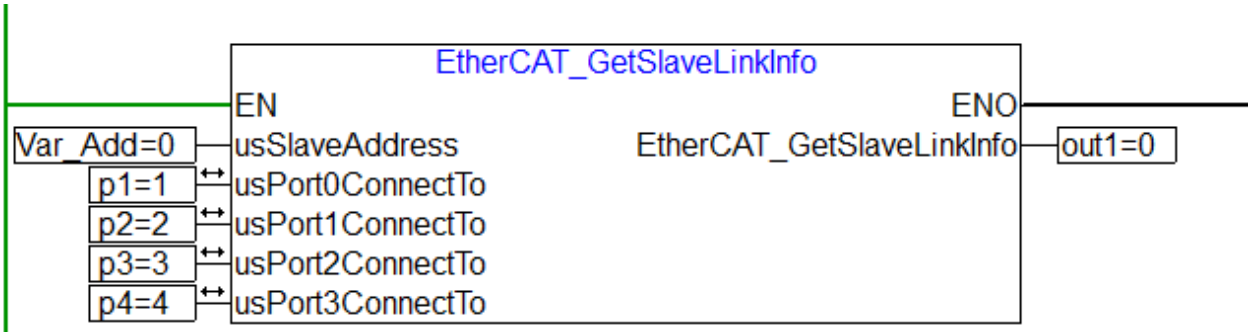
8.1.13.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
usSlaveAddress	Input	UINT	Slave station address (such as: 1001)	0	FALSE
usPort0ConnectTo	Input/Output	UINT	Slave Connection Port 0: 0: Connected to the master station 1 to n: Station address of the connected peer slave 0xFFFF: Not connected	0	FALSE
usPort1ConnectTo	Input/Output	UINT	Slave Connection Port 1: 0: Connected to the master station 1 to n: Station address of the connected peer slave	0	FALSE

			0xFFFF: Not connected		
usPort2ConnectTo	Input/Output	UINT	Slave Connection Port 2: 0: Connected to the master station 1 to n: Station address of the connected peer slave 0xFFFF: Not connected	0	FALSE
usPort3ConnectTo	Input/Output	UINT	Slave Connection Port 3: 0: Connected to the master station 1 to n: Station address of the connected peer slave 0xFFFF: Not connected	0	FALSE
EtherCAT_ GetSlaveLinkInfo	Output	DWORD	Get the slave station port Link information	0	FALSE

8.1.13.3 Example

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	Var_Add			UINT	0	FALSE	Not Public	
0002	p1			UINT	1	FALSE	Not Public	
0003	p2			UINT	2	FALSE	Not Public	
0004	p3			UINT	3	FALSE	Not Public	
0005	p4			UINT	4	FALSE	Not Public	
0006	out1			DWORD	0	FALSE	Not Public	




```
12
13  out1 := EtherCAT_GetSlaveLinkInfo(Var_Add, p1, p2, p3, p4);
```

out1 = 0 Var_Add = 0 p1 = 1 p2 = 2 p3 = 3 p4 = 4

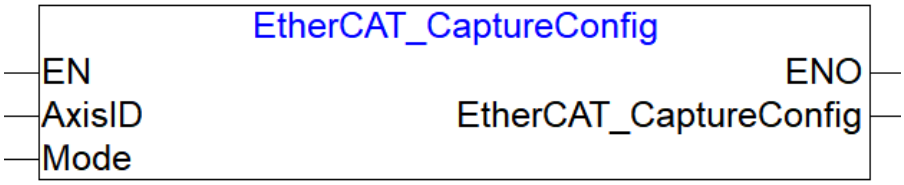
8.1.14 EtherCat Touch Probe

It is to configure the high-speed capture function of the EtherCAT bus servo slave station. After the capture function is configured, it is better to call EtherCAT_CaptureGetStatus in IEC to scan the capture status cyclically. When it is detected that the capture is completed, immediately call EtherCAT_CaptureGetMpos to read the required capture position.

8.1.14.1 EtherCAT_CaptureConfig (Touch Probe Configuration)

8.1.14.1.1 Function

This function enables high-precision position capture of the EtherCAT bus driver. The capture is completed by hardware without software delay deviation. The capture trigger source can be triggered by the IO input corresponding to the driver's touch probe function, or triggered by the Z-phase 0 pulse of the encoder.



EtherCAT_CaptureConfig Functional Block

- Note: After configuring capture, it is a good idea to call EtherCAT _ CaptureGetStatus in the IEC to loop through the capture status and immediately call EtherCAT _ CaptureGetMpos to read the capture location as soon as the capture is detected to be complete.

8.1.14.1.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
Input	EN	BOOL	Rising edge trigger module reset function		
	AxisID	USINT	Axis number	0	FALSE
	Mode	UINT	In capture mode, the servo mode parameters are configured in the following table (for more details, see the related object for the probe function of the servo drive used, index: 0x60B8)	0	FALSE
Output	ENO	/	/		
	EtherCAT_CaptureConfig	UDINT	Configure high-speed location capture 0: Configured successfully Others: error code	0	FALSE
Capture Channel	**Bit**		**Parameter Description**		
Touch probe1	0		0: Close capture channel 1 1: Open capture channel 1		
	1		0: Single trigger 1: Multiple triggers		
	2		0: Input signal trigger of channel 1 (touch probe1 input) 1: Z phase 0 pulse trigger		
	3		Reserved		
	4		0: Close rising edge capture 1: Open rising edge capture		
	5		0: Close falling edge capture 1: Open falling edge capture		
	6		Reserved		
	7		Reserved		

Touch probe2	8	0: Close capture channel 2 1: Open capture channel 2
	9	0: Single trigger 1: Multiple triggers
	10	0: Input signal trigger of channel 2 (touch probe2 input) 1: Z phase 0 pulse trigger
	11	Reserved
	12	0: Close rising edge capture 1: Open rising edge capture
	13	0: Close falling edge capture 1: Open falling edge capture
	14	Reserved
	15	Reserved

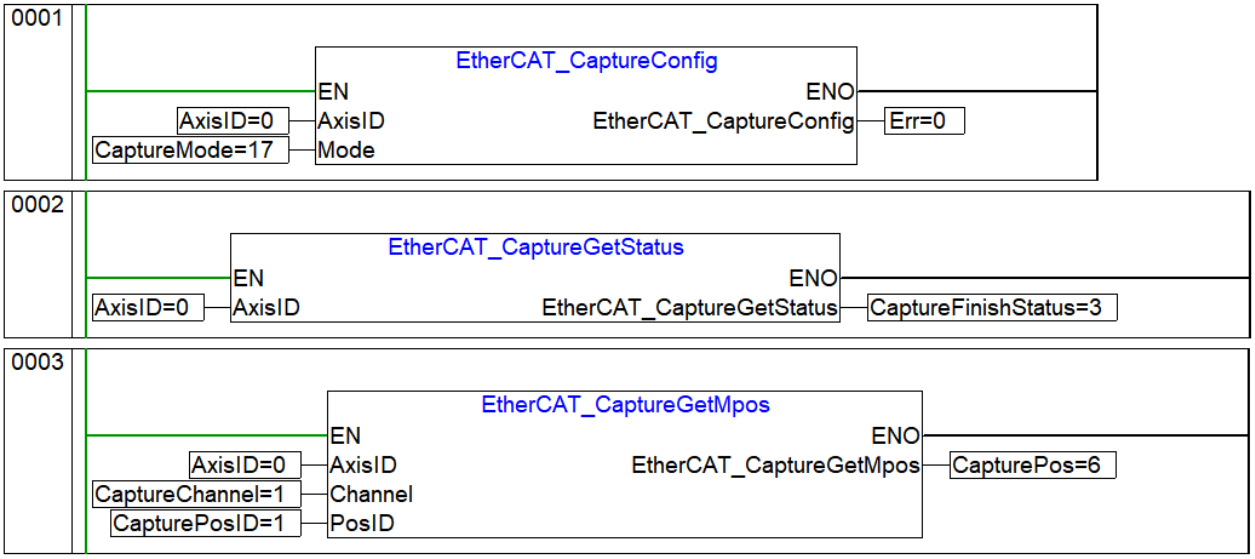
8.1.14.1.3 Instruction type

Non-axis motion cache instruction.

8.1.14.1.4 Example

When the touch probe1 pin of the servo driver corresponding to axis 0 inputs a rising edge, the position value of the axis is captured.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	AxisID		Trapping axis	USINT	0	FALSE	Not Public	
0002	CaptureMode		Capture mode	UINT	17	FALSE	Not Public	
0003	CaptureFinishStatus		State on capture completion	WORD	3	FALSE	Not Public	
0004	Err		Error code	UDINT	0	FALSE	Not Public	
0005	CaptureChannel		Capture channel 1: where ascending edge captured; 2: where descending edge captured	USINT	1	FALSE	Not Public	
0006	CapturePosID		Location captured	USINT	1	FALSE	Not Public	
0007	CapturePos		Capture position	LREAL	6	FALSE	Not Public	

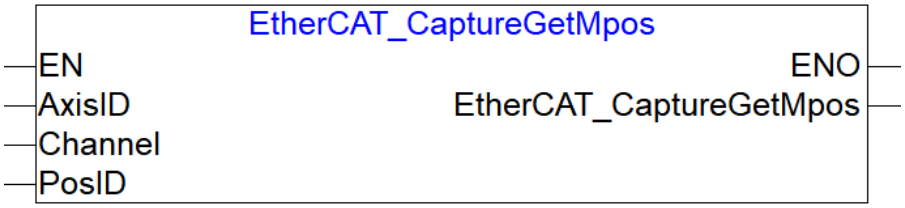



```
AxisID :=0;(*Trapping axis*)
CaptureMode :=16#11;(*Capture mode configuration: Using channel 1, single trigger, trigger signal from touch probe1 input, capture only ascending edge*)
CaptureFinishStatus :=16#3;(*State on capture completion*)
Err := EtherCAT_CaptureConfig(AxisID, CaptureMode);(*Configuration capture: returns 0 on successful configuration*)
IF 0 =Err THEN(*Configuration succeeded*)
    WHILE TRUE DO;
        IF CaptureFinishStatus = (UINT_TO_WORD(EtherCAT_CaptureGetStatus(AxisID))AND CaptureFinishStatus)THEN(*Query capture complete*)
            CaptureChannel :=1;(*Capture channel 1: touch probe1; 2: touch probe2*)
            CapturePosID :=1;(*Capture channel 1: where ascending edge captured; 2: where descending edge captured*)
            CapturePos := EtherCAT_CaptureGetMpos(AxisID, CaptureChannel, CapturePosID);(*Get the location captured*)
            EXIT;(*Exit loop after getting capture location*)
        END_IF
    END_WHILE
END_IF
```

8.1.14.2 EtherCAT_CaptureGetStatus (Get Touch Probe Status)

8.1.14.2.1 Function

Return the status of the high-speed capture function of the specified slave station.



EtherCAT_CaptureGetStatus Functional Block

8.1.14.2.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
Input	EN	BOOL	Rising edge trigger module reset function		
	AxisID	USINT	Axis number	2	FALSE
Output	ENO	/	/		
	EtherCAT_CaptureGetStatus	UINT	High-speed capture function status(Return 0 if instruction call fails)	0	FALSE
Capture Channel	**Bit**		**Parameter Description**		
Touch probe1	0		0: Capture channel 1 closed 1: Capture channel 1 opened		
	1		0: Rising edge capture of capture channel 1 is not completed 1: Rising edge capture of capture channel 1 completed		
	2		0: Falling edge capture of capture channel 1 not completed 1: Falling edge capture of capture channel 1 completed		
	3		Reserved		
	4		Reserved		
	5		Reserved		
	6		For testing		
	7		For testing		
Touch probe2	8		0: Capture channel 2 closed 1: Capture channel 2 opened		
	9		0: Rising edge capture of capture channel 2 not completed 1: Rising edge capture of capture channel 2 completed		
	10		0: Falling edge capture of capture channel 2 is not completed 1: Falling edge capture of capture channel 2 completed		
	11		Reserved		
	12		Reserved		
	13		Reserved		
	14		For testing		
	15		For testing		

8.1.14.2.3 Instruction type

Non-axis motion cache instruction.

8.1.14.2.4 Example

See EtherCAT_CaptureConfig.

8.1.14.3 EtherCAT_CaptureGetMpos (Get Touch Probe Result)

8.1.14.3.1 Function

Return Mpos obtained from the high-speed capture channel of the specified slave station.

EtherCAT_CaptureGetMpos Functional Block

8.1.14.3.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
Input	EN	BOOL	Rising edge trigger module reset function		
	AxisID	USINT	Axis number	1	FALSE
	Channel	USINT	Channel number: 1: Channel 1 (touch probel 1) 2: Channel 2 (touch probel 2)	1	FALSE
	PosID	USINT	Location ID to be captured 1: Rising edge 2: Falling edge (some drives do not support falling edge capture, such as Panasonic A5B)	1	FALSE
Output	ENO	/	/		
	EtherCAT_CaptureGetMpos	LREAL	Captured Mpos value (0 is returned when it fails to call the instruction)	0	FALSE

8.1.14.3.3 Instruction type

Non-axis motion cache instruction.

8.1.14.3.4 Example

See EtherCAT_CaptureConfig.

8.1.14.3.5 Additional information

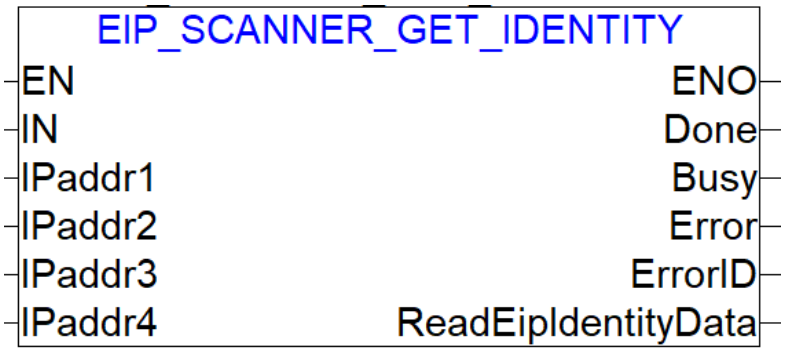
To get this value, please first use the EtherCAT_CaptureGetStatus instruction to check the capture status and confirm that the value to get has been captured.

8.2 EtherNetIp Functions

8.2.1 EIP_SCANNER_GET_IDENTITY (Get save identity information)

8.2.1.1 Function

Get the slave station identity information.



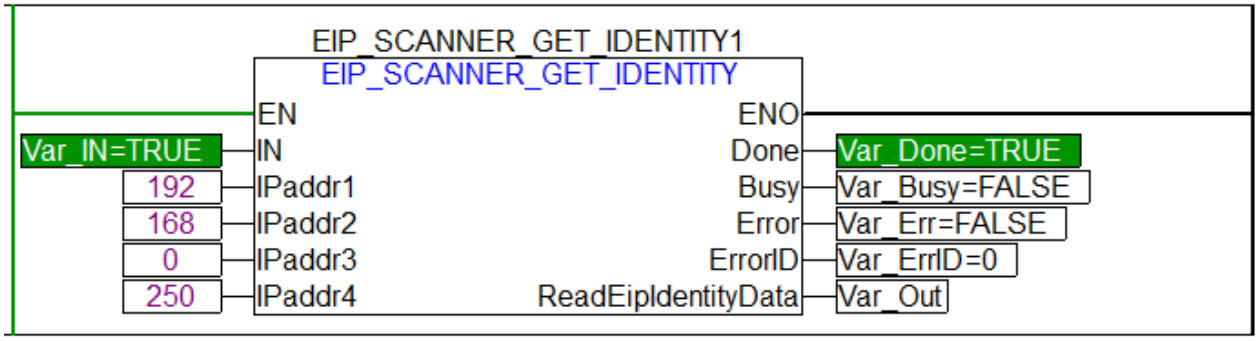
EIP_SCANNER_GET_IDENTITY Functional Block

8.2.1.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
IN	Input	BOOL	High level enable (default: FALSE)	FALSE	FALSE
IPaddr1	Input	BYTE	First bit of IP address	0	FALSE
IPaddr2	Input	BYTE	Bit 2 of IP address	0	FALSE
IPaddr3	Input	BYTE	Bit 3 of IP address	0	FALSE
IPaddr4	Input	BYTE	Bit 4 of IP address	0	FALSE
Done	Output	BOOL	Data acquisition succeeded	FALSE	FALSE
Busy	Output	BOOL	Obtaining data	FALSE	FALSE
Error	Output	BOOL	Failed to obtain data	FALSE	FALSE
ErrorID	Output	WORD	Error code 0: No error 1: IP address error 2: Data address error 3: Data retrieval failure 4: Data setting failure	0	FALSE
ReadEipIdentityData	Output	EIP_IDENTITY_DATA	Read EIP identity data		FALSE

8.2.1.3 Example

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	EIP_SCANNER_GET_IDENTITY1			EIP_SCANNER_GET_IDENTITY		FALSE	Not Public	
0002	Var_IN			BOOL	TRUE	FALSE	Not Public	
0003	Var_Done			BOOL	TRUE	FALSE	Not Public	
0004	Var_Busy			BOOL	FALSE	FALSE	Not Public	
0005	Var_Err			BOOL	FALSE	FALSE	Not Public	
0006	Var_ErrID			WORD	0	FALSE	Not Public	
0007	Var_Out			EIP_IDENTITY_DATA		FALSE	Not Public	



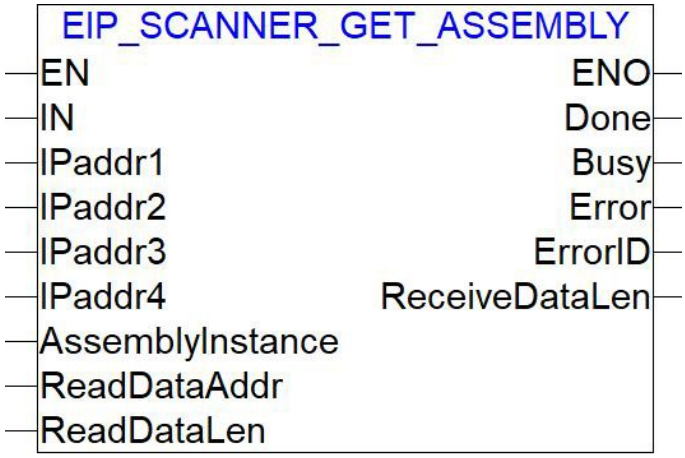

```
1  EIP_SCANNER_GET_IDENTITY1(
2  IN := Var_IN,
3  IPaddr1 := 192,
4  IPaddr2 := 168,
5  IPaddr3 := 0,
6  IPaddr4 := 250,
7  Done=>Var_Done,
8  Busy=>Var_Busy,
9  Error=>Var_Err,
10 ErrorID=>Var_ErrID,
11 ReadEipIdentityData=>Var_Out);
```

```
EIP_SCANNER_GET_IDENTITY1
Var_IN = TRUE

Var_Done = TRUE
Var_Busy = FALSE
Var_Err = FALSE
Var_ErrID = 0
Var_Out
```

8.2.2 EIP_SCANNER_GET_ASSEMBLY (Get slave assembly information)

8.2.2.1 Function

Get the slave station assembly information.



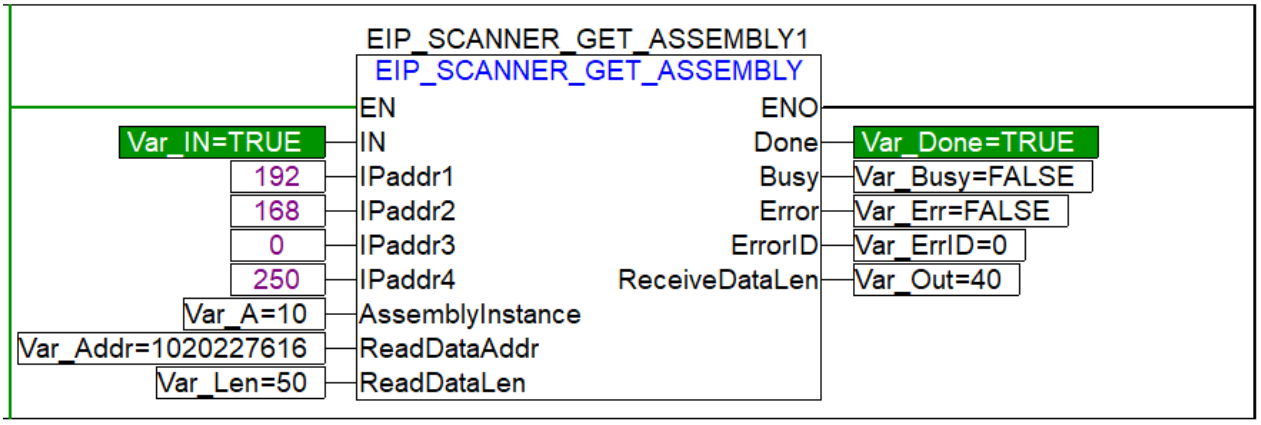
EIP_SCANNER_GET_ASSEMBLY Functional Block

8.2.2.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
IN	Input	BOOL	High level enable (default: FALSE)	FALSE	FALSE
IPaddr1	Input	BYTE	Bit 1 of IP address	0	FALSE
IPaddr2	Input	BYTE	Bit 2 of IP address	0	FALSE
IPaddr3	Input	BYTE	Bit 3 of IP address	0	FALSE
IPaddr4	Input	BYTE	Bit 4 of IP address	0	FALSE
AssemblyInstance	Input	DWORD	Instance number	0	FALSE
ReadDataAddr	Input	POINTER TO BYTE	Obtain data storage address	0	FALSE
ReadDataLen	Input	WORD	Obtain data length	0	FALSE
Done	Output	BOOL	Data acquisition succeeded	FALSE	FALSE
Busy	Output	BOOL	Obtaining data	FALSE	FALSE
Error	Output	BOOL	Failed to obtain data	FALSE	FALSE
ErrorID	Output	WORD	Error code 0: No error 1: IP address error 2: Data address error 3: Data retrieval failure 4: Data setting failure	0	FALSE
ReceiveDataLen	Output	WORD	Data receiving length	0	FALSE

8.2.2.3 Example

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	EIP_SCANNER_GET_ASSEMBLY1			EIP_SCANNER_GET_ASSEMBLY		FALSE	Not Public	
0002	Var_IN			BOOL	TRUE	FALSE	Not Public	
0003	Var_A			DWORD	10	FALSE	Not Public	
0004	Var_Addr			POINTER TO BYTE	1020227616	FALSE	Not Public	
0005	Var_Len			WORD	50	FALSE	Not Public	
0006	Var_Done			BOOL	TRUE	FALSE	Not Public	
0007	Var_Busy			BOOL	FALSE	FALSE	Not Public	
0008	Var_Err			BOOL	FALSE	FALSE	Not Public	
0009	Var_ErrID			WORD	0	FALSE	Not Public	
0010	Var_Out			WORD	40	FALSE	Not Public	



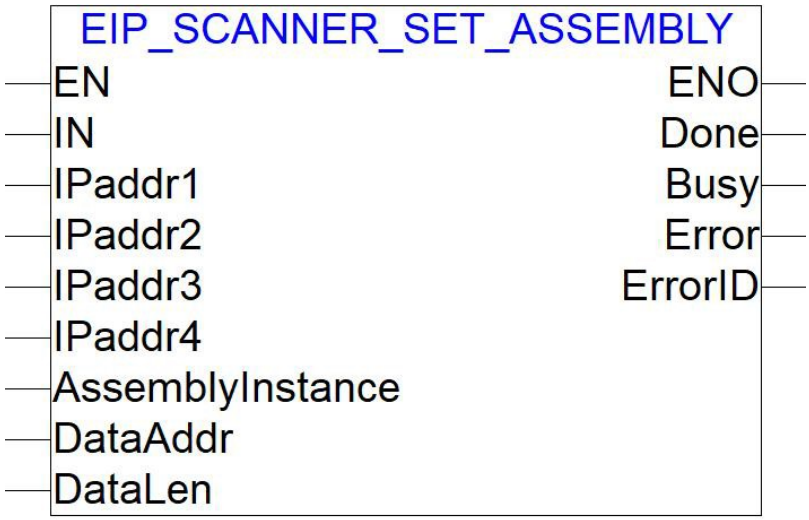

```
1  EIP_SCANNER_GET_ASSEMBLY1(
2  IN := Var_IN,
3  IPaddr1 := 192,
4  IPaddr2 := 168,
5  IPaddr3 := 0,
6  IPaddr4 := 250,
7  AssemblyInstance := Var_A,
8  ReadDataAddr := Var_Addr,
9  ReadDataLen := Var_Len,
10 Done=>Var_Done,
11 Busy=>Var_Busy,
12 Error=>Var_Err,
13 ErrorID=>Var_ErrID,
14 ReceiveDataLen=>Var_Out);
```

```
EIP_SCANNER_GET_ASSEMBLY1
Var_IN = TRUE

Var_A = 10
Var_Addr = 1020227616
Var_Len = 50
Var_Done = TRUE
Var_Busy = FALSE
Var_Err = FALSE
Var_ErrID = 0
Var_Out = 40
```

8.2.3 EIP_SCANNER_SET_ASSEMBLY (Set slave assembly information)

8.2.3.1 Function

Set the slave station assembly information.



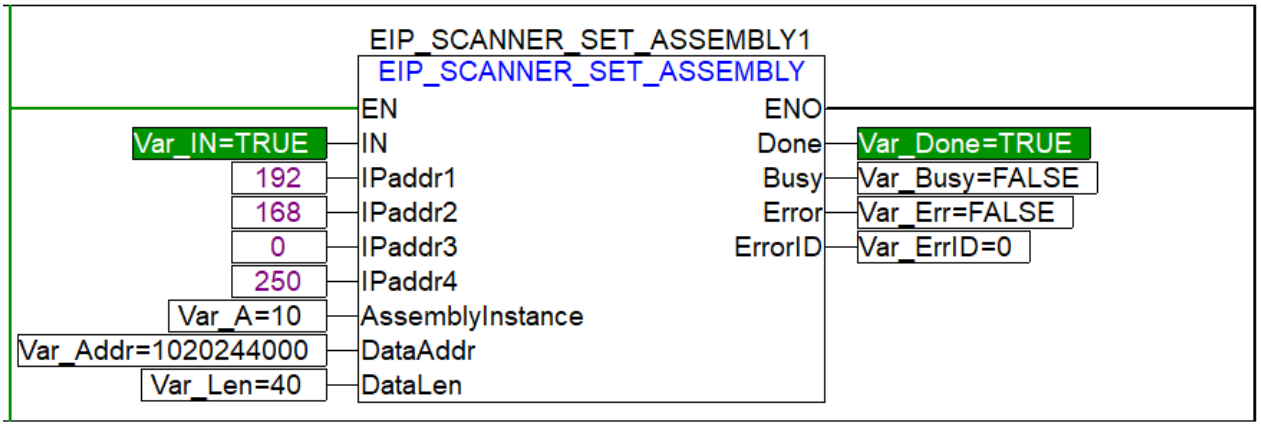
EIP_SCANNER_SET_ASSEMBLY Functional Block

8.2.3.2 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
IN	Input	BOOL	High level enable (default: FALSE)	FALSE	FALSE
IPaddr1	Input	BYTE	First bit of IP address	0	FALSE
IPaddr2	Input	BYTE	Bit 2 of IP address	0	FALSE
IPaddr3	Input	BYTE	Bit 3 of IP address	0	FALSE
IPaddr4	Input	BYTE	Bit 4 of IP address	0	FALSE
AssemblyInstance	Input	DWORD	Instance number	0	FALSE
DataAddr	Input	POINTER TO BYTE	Data storage address	0	FALSE
DataLen	Input	WORD	Data length	0	FALSE
Done	Output	BOOL	Data written successfully	FALSE	FALSE
Busy	Output	BOOL	Data writing in progress	FALSE	FALSE
Error	Output	BOOL	Failed to write data	FALSE	FALSE
ErrorID	Output	WORD	Error code 0: No error 1: IP address error 2: Data address error 3: Data retrieval failure 4: Data setting failure	0	FALSE

8.2.3.3 Example

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	EIP_SCANNER_SET_ASSEMBLY1			EIP_SCANNER_SET_ASSEMBLY		FALSE	Not Public	
0002	Var_IN			BOOL	TRUE	FALSE	Not Public	
0003	Var_A			DWORD	10	FALSE	Not Public	
0004	Var_Addr			POINTER TO BYTE	10202440...	FALSE	Not Public	
0005	Var_Len			WORD	40	FALSE	Not Public	
0006	Var_Done			BOOL	TRUE	FALSE	Not Public	
0007	Var_Busy			BOOL	FALSE	FALSE	Not Public	
0008	Var_Err			BOOL	FALSE	FALSE	Not Public	
0009	Var_ErrID			WORD	0	FALSE	Not Public	




```
1  EIP_SCANNER_SET_ASSEMBLY1(
2  IN := Var_IN,
3  IPaddr1 := 192,
4  IPaddr2 := 168,
5  IPaddr3 := 0,
6  IPaddr4 := 250,
7  AssemblyInstance := Var_A,
8  DataAddr := Var_Addr,
9  DataLen := Var_Len,
10 Done=>Var_Done,
11 Busy=>Var_Busy,
12 Error=>Var_Err,
13 ErrorID=>Var_ErrID);
```

```
EIP_SCANNER_SET_ASSEMBLY1
Var_IN = TRUE

Var_A = 10
Var_Addr = 1020244000
Var_Len = 40
Var_Done = TRUE
Var_Busy = FALSE
Var_Err = FALSE
Var_ErrID = 0
```

8.2.4 CIPOpen (CIP Class3 Create Connection)

8.2.4.1 Version

- AutoThink Software Version: V3.2.3SP2 and above.
- The controller firmware version is as follows:

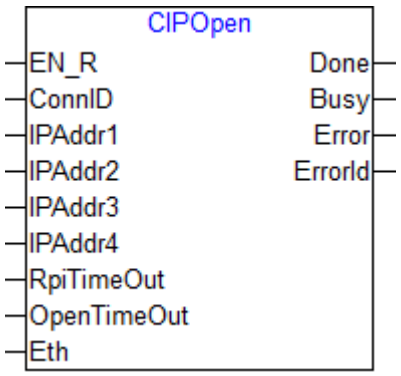
Controller Type	Firmware Version

LX-CU430	LX-CU430_V1.1.1 and above.
LX-CU433	LX-CU433_V1.1.0 and above.
LX-CU500	LX-CU500_V1.4.0 and above.
LX-CU501	LX-CU501_V1.2.0 and above.
LX-CU510	LX-CU510_V1.1.0 and above.
LX-CU511	LX-CU511_V1.0.1 and above.

8.2.4.2 Function

Open a connection.

Reading tags, writing tags, and closing the connection all require opening the connection first.



CIPOpen Functional Block

8.2.4.3 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN_R	INPUT	BOOL	When not connected, a rising edge triggers a Class 3 connection.	FALSE	FALSE
ConnID	INPUT	UINT	Valid ID, range: 1–32.	1	FALSE
IPAddr1	INPUT	BYTE	Connected object's IP address 1	0	FALSE
IPAddr2	INPUT	BYTE	Connected object's IP address 2	0	FALSE
IPAddr3	INPUT	BYTE	Connected object's IP address 3	0	FALSE
IPAddr4	INPUT	BYTE	Connected object's IP address 4	0	FALSE
RpiTimeOut	INPUT	UINT	RPI timeout, unit: 100ms, valid range: 1~65535.	1	FALSE
OpenTimeOut	INPUT	UINT	Open command execution timeout, unit: s, valid range: 1~600.	1	FALSE
Eth	INPUT	BYTE	Port number. 0 represents P1, 1 represents P2.	0	FALSE
Done	OUTPUT	BOOL	Command completion status. FALSE: Not completed. TRUE: Completed.	FALSE	FALSE
Busy	OUTPUT	BOOL	Command processing status FALSE: Idle. TRUE: Processing.	FALSE	FALSE
Error	OUTPUT	BOOL	Command error status FALSE: No error. TRUE: Error occurred.	FALSE	FALSE
ErrorId	OUTPUT	DINT	Error Code	0	FALSE

			0: Success. Non-zero: Specific values refer to the error code table.		

8.2.4.4 Error Code

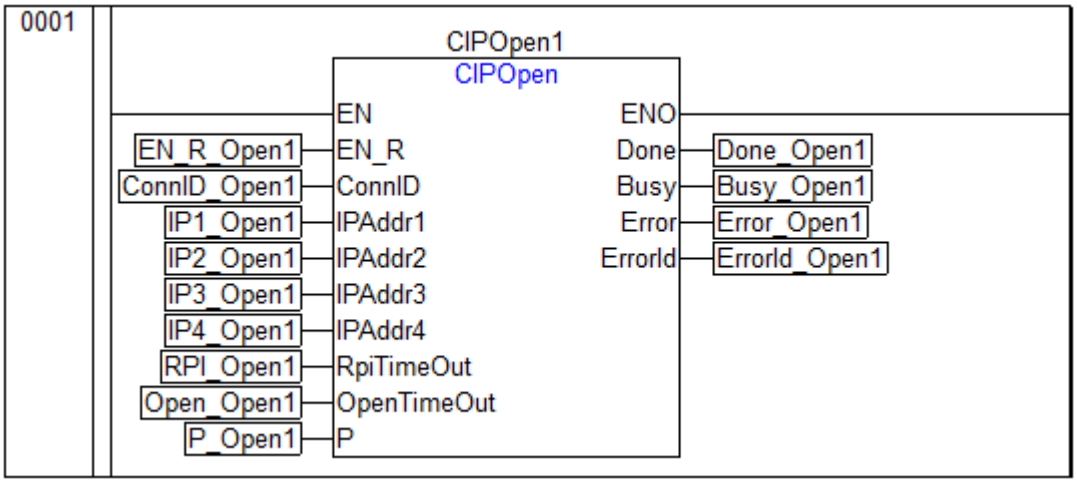
Error code	Description
0	Open successful.
2	Invalid target IP (IP address is not in the same subnet as the current network card).
5	Connection timeout interrupted.
6	Connection closed.
207	Connection limit exceeded.
503	Connection failed (timeout).
601	EIP tag communication error: Specified network card does not exist in hardware configuration.
602	Invalid network card number (Valid range: 0–1, where 0 = P1, 1 = P2).
603	Invalid IP address (Must be non-zero).
604	Invalid RPI connection timeout value (Valid range: 1–65535).
605	Invalid Open command timeout value (Valid range: 1–600).
606	Invalid connection ID.
607	Connection is currently in use.
608	Modification error: ConnID is not in closed state (ConnID changed).
609	Modification error: ConnID is not in closed state (Network interface changed).
610	Modification error: ConnID is not in closed state (IP address changed).
611	Modification error: ConnID is not in closed state (RPI timeout changed).
612	Modification error: ConnID is not in closed state (Open timeout changed).
614	Connection already open.

8.2.4.5 Example

1. Variable Definition

No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	CIPOpen1			CIPOPEN		FALSE	Not Public	
0002	EN_R_Open1			BOOL	FALSE	FALSE	Not Public	
0003	ConnID_Open1			UINT	0	FALSE	Not Public	
0004	IP1_Open1			BYTE	0	FALSE	Not Public	
0005	IP2_Open1			BYTE	0	FALSE	Not Public	
0006	IP3_Open1			BYTE	0	FALSE	Not Public	
0007	IP4_Open1			BYTE	0	FALSE	Not Public	
0008	RPI_Open1			UINT	0	FALSE	Not Public	
0009	Open_Open1			UINT	0	FALSE	Not Public	
0010	P_Open1			BYTE	0	FALSE	Not Public	
0011	Done_Open1			BOOL	FALSE	FALSE	Not Public	
0012	Busy_Open1			BOOL	FALSE	FALSE	Not Public	
0013	Error_Open1			BOOL	FALSE	FALSE	Not Public	
0014	ErrorId_Open1			DINT	0	FALSE	Not Public	

2. LD Configuration Example



3. ST Configuration Example

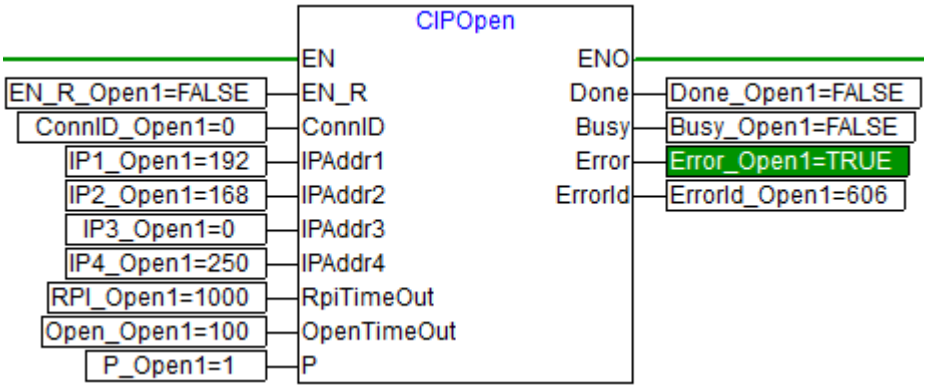
```
CIPOpen1(
   EN_R := EN_R_Open1,
   ConnID := ConnID_Open1,
   IPAddr1 := IP1_Open1,
   IPAddr2 := IP2_Open1,
   IPAddr3 := IP3_Open1,
   IPAddr4 := IP4_Open1,
   RpiTimeOut := RPI_Open1,
   OpenTimeOut := Open_Open1,
   P := P_Open1,
   Done=>Done_Open1,
   Busy=>Busy_Open1,
   Error=>Error_Open1,
   ErrorId=>ErrorId_Open1);
```

4. This example demonstrates how to use the CIPOpen function block.

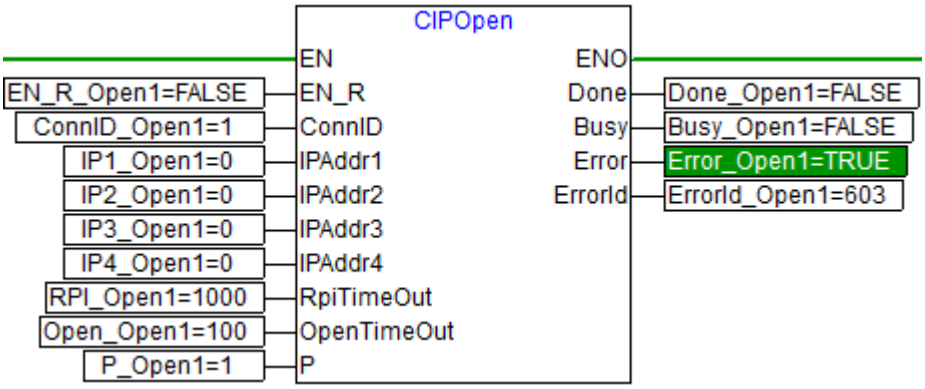
   ■ Parameter Check

   The function block triggers parameter inspection on the rising edge of EN_R.

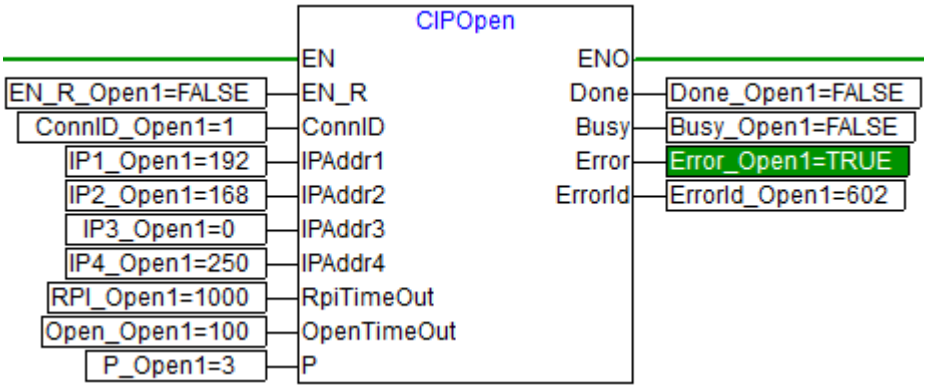
   □ Valid ConnID range: 1 to 32. If ConnID is outside valid range, report diagnostic error code 606 (Invalid Connection ID).



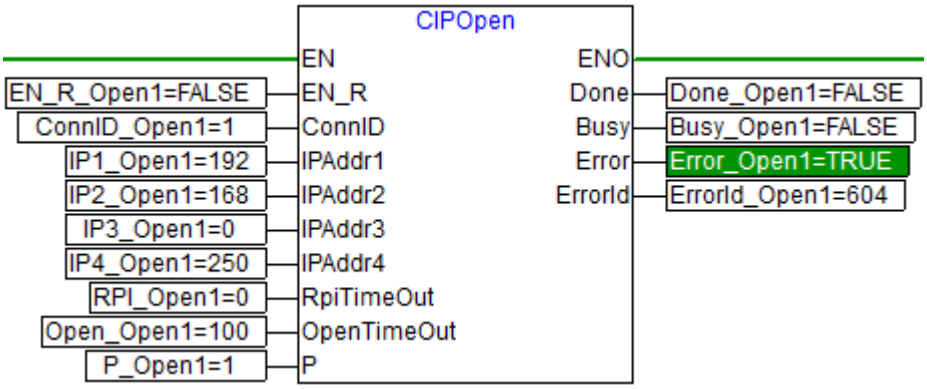
- The IP address must not be 0, or report error code 603 (Invalid IP Address).



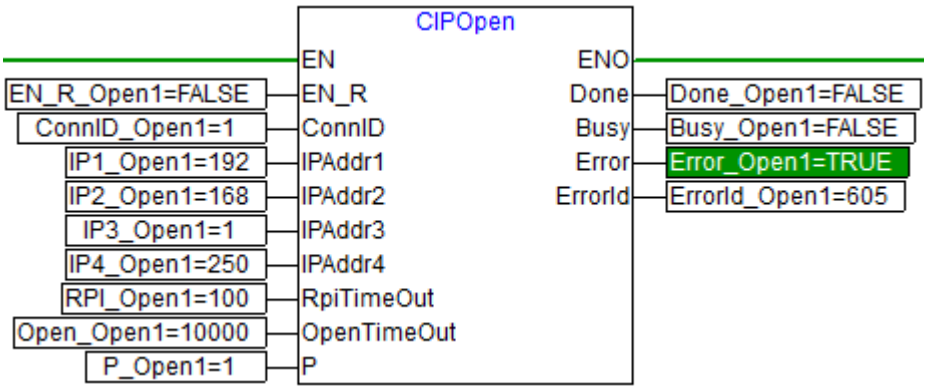
- Valid NIC number range: 0 to 1, 0 = Port1, 1 = Port2. If out of range, report error code 602 (Invalid NIC Number).



- RPI timeout valid range: 1–65535, if the value is outside this range, report Error 604 (Invalid RPI Timeout).



□ Open connection timeout valid range: 1 second to 10 minutes, if the value is outside this range, report Error 605 (Invalid Open Timeout)

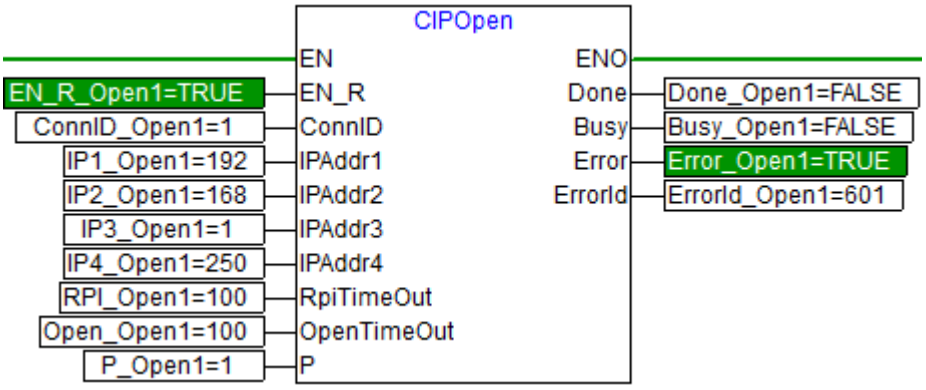


■ EtherNet/IP tag communication hardware configuration diagnostic.

The function block triggers hardware configuration validation on the rising edge of EN_R during connection establishment.

□ Verifies whether the network port has proper EtherNet/IP tag communication hardware nodes configured. If the designated network interface lacks required hardware configuration, Reports Error 601 (EtherNet/IP Tag Communication Hardware Configuration Missing for Specified Network Port)

□ After successful validation, reporting standard connection status and results.



■ Message Transmission

When the EN_R of the function block is FALSE, set parameters such as ConnID. After the setup

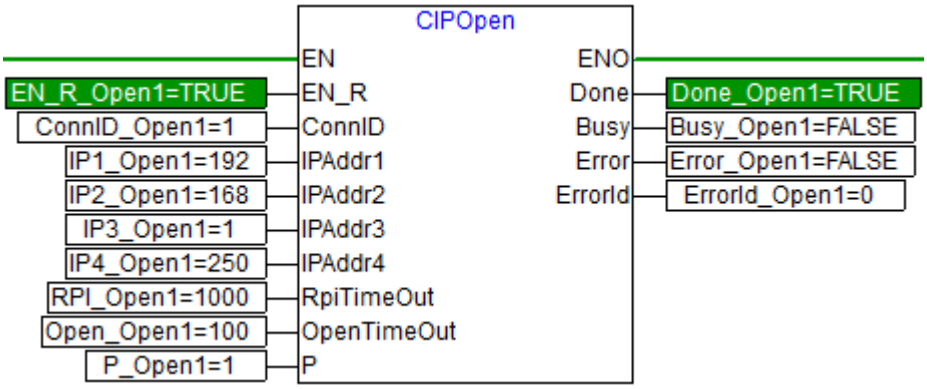
is completed, set EN_R to TRUE to trigger the message transmission of the function block on the rising edge.

The function block providing three status indicators:

Output parameter states:

- Command is processing: Busy = TRUE, Done = FALSE, Error = FALSE.
- Command is completion: Done = TRUE, Busy, Error = FALSE.
- Command is erroe: Error = TRUE, Done = FALSE, Busy = FALSE, ErrorID populated with specific error code.

Successful opening sequence:



■ Parameter modification Rules during active connection

Input parameters cannot be modified while the function block is connected (regardless of EN_R state). Any unauthorized changes trigger errors until parameters are corrected, or connection is closed

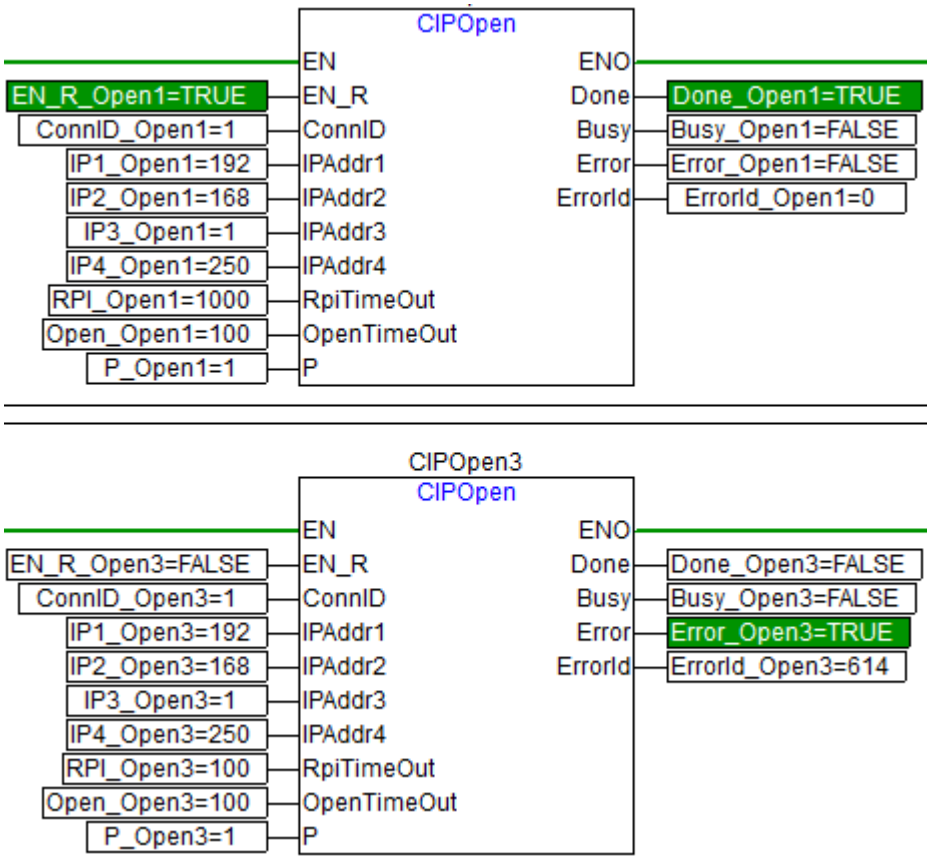
Error Codes:

- 608: ConnID modified during active connection
- 609: Network card number modified during active connection
- 610: IP address modified during active connection
- 611: RPI timeout modified during active connection
- 612: Open timeout modified during active connection

■ Real-Time Error Detection

The function block continuously monitors connection status (regardless of EN_R state), detecting open connections and Closed connections.

- 614 (Connection Already Open): Triggered when a non-opening function block attempts to use an already open ConnID



- When connection is OPEN, all function blocks cannot trigger another open command with the same ConnID.
- When connection is CLOSED, all function blocks using the same ConnID report Error 6 (Connection Closed).

■ Download-Triggered Task Initialization

During initialization triggered by a download operation, all active connections are forcibly closed. Any open function block detecting its assigned ConnID report Error 6 (Connection Closed)

8.2.5 CIPRead (CIP Class3 Read Tag)

8.2.5.1 Version

■ AutoThink Software Version: V3.2.3SP2 and above.

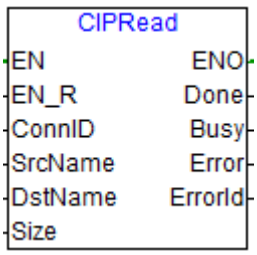
■ The controller firmware version is as follows:

Controller Type	Firmware Version
LX-CU430	LX-CU430_V1.1.1 and above.
LX-CU433	LX-CU433_V1.1.0 and above.
LX-CU500	LX-CU500_V1.4.0 and above.
LX-CU501	LX-CU501_V1.2.0 and above.
LX-CU510	LX-CU510_V1.1.0 and above.

LX-CU511	LX-CU511_V1.0.1 and above.

8.2.5.2 Function

Write the read peer tag data into the source tag. SrcName specifies the source tag name, DstName specifies the peer tag name.



CIPRead Functional Block

8.2.5.3 Precautions

- Data types such as DATE, DT, TIME, and TOD are not supported for read/write operations with Omron devices.
- A single BOOL variable in Omron occupies two bytes, while each BOOL in a BOOL array occupies one byte. Therefore, when communicating BOOL-type data with Omron devices, only arrays can be read or written, and the number of elements must be greater than one. Otherwise, Omron will determine the data length as mismatched and reject the read/write operation.
- The STRING data type does not support communication in array form.
- Writing an illegal address to a pointer can easily cause controller exceptions. Please use pointer types with caution.
- When the function block's Done signal becomes TRUE indicates that the read operation has completed successfully. Verify if the local DstName data area contains the newly written values. If values are present, operation successful. If no values are written, operation failed.

8.2.5.4 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN_R	INPUT	BOOL	Enable, rising edge triggers operation.	FALSE	FALSE
ConnID	INPUT	UINT	Valid ID, range: 1–32.	1	FALSE
SrcName	INPUT	STRING(255)	Peer tag name, max length 255, excluded '\0'.	-	FALSE
DstName	INPUT	STRING(255)	Source tag name, max length 255, excluded '\0'.	-	FALSE
Size	INPUT	UINT	Number of elements read from the peer tag Typically, the Size is 1. For array or string types, the Size indicates the actual number of elements read or written.	1	FALSE
Done	OUTPUT	BOOL	Command completion status.	FALSE	FALSE

			FALSE: Not completed. TRUE: Completed.		
Busy	OUTPUT	BOOL	Command processing status FALSE: Idle. TRUE: Processing.	FALSE	FALSE
Error	OUTPUT	BOOL	Command error status FALSE: No error. TRUE: Error occurred.	FALSE	FALSE
ErrorId	OUTPUT	DINT	Error Code 0: Success. Non-zero: Specific values refer to the error code table.	0	FALSE

8.2.5.5 Error Code

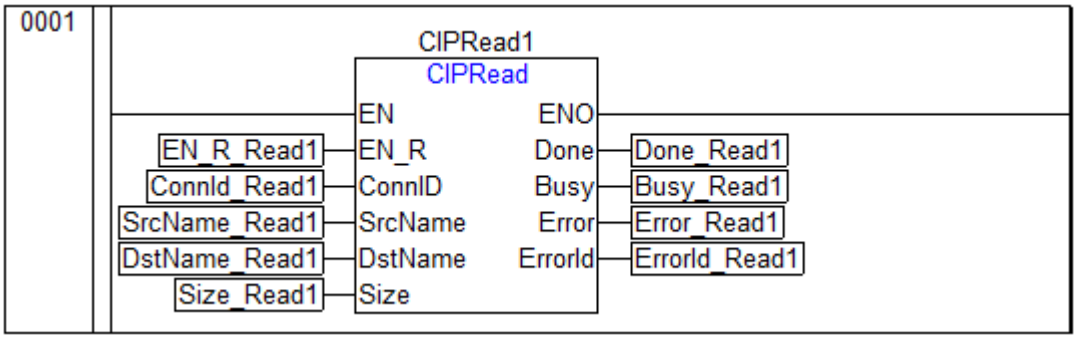
Error code	Description
0	No diagnostic information available.
501	The read target connection is closed.
502	Class 3 read limit exceeded (only 1 read allowed at a time).
503	Source tag array error. 1.Array tag - Invalid element format (cannot convert to number). 2.Array tag - Zero-length element. 3.Array tag - Missing closing bracket. 4.Standard tag - Tag does not exist or length is 0.
504	Requested read size exceeds limit or is zero.
505	Target tag array error. 1.Array tag - Invalid element format (cannot convert to number). 2.Array tag - Zero-length element. 3.Array tag - Missing closing bracket. 4.Standard tag - Tag does not exist or length is 0.
506	Tag data type is not supported.
507	Target tag information not found.
508	Received an error response (peer returned an exception: 1. Peer tag does not exist; 2. Peer considers the Size incorrect).
509	Data type mismatch.
606	Invalid connection ID.
607	The connection is currently busy.

8.2.5.6 Example

1. Variable Definition

No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	CIPRead1			CIPREAD		FALSE	Not Public	
0002	EN_R_Read1			BOOL	FALSE	FALSE	Not Public	
0003	ConnId_Read1			UINT	0	FALSE	Not Public	
0004	SrcName_Read1			STRING(255)	''	FALSE	Not Public	
0005	DstName_Read1			STRING(255)	''	FALSE	Not Public	
0006	Size_Read1			UINT	0	FALSE	Not Public	
0007	Done_Read1			BOOL	FALSE	FALSE	Not Public	
0008	Busy_Read1			BOOL	FALSE	FALSE	Not Public	
0009	Error_Read1			BOOL	FALSE	FALSE	Not Public	
0010	ErrorId_Read1			DINT	0	FALSE	Not Public	

2. LD Configuration Example



3. ST Configuration Example

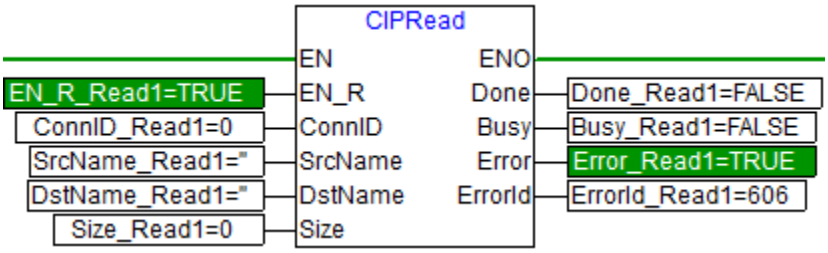
```
CIPRead1(
    EN_R := EN_R_Read1,
    ConnID := ConnId_Read1,
    SrcName := SrcName_Read1,
    DstName := DstName_Read1,
    Size := Size_Read1,
    Done=>Done_Read1,
    Busy=>Busy_Read1,
    Error=>Error_Read1,
    ErrorId=>ErrorId_Read1);
```

4. This example demonstrates how to use the CIPRead function block.

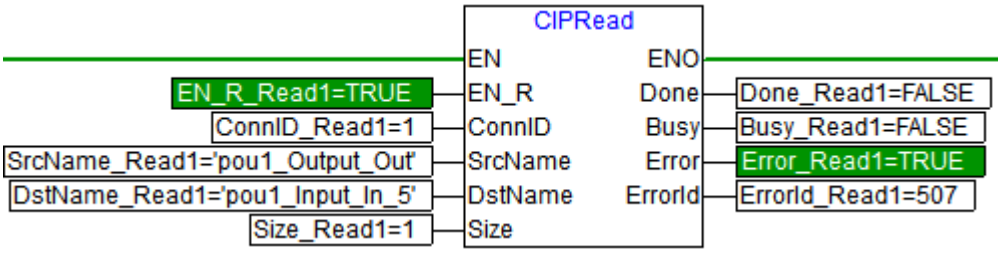
■ Parameter Check

The function block triggers parameter inspection on the rising edge of EN_R.

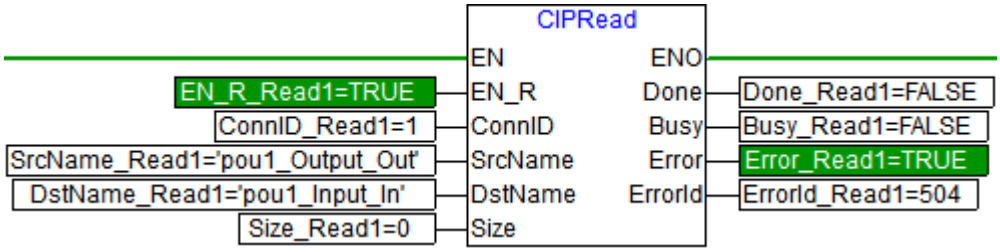
□ Valid ConnID range: 1 to 32. If ConnID is outside valid range, report diagnostic error code 606 (Invalid Connection ID).



- DstName is set incorrectly, reporting a 507 error (target tag information not found) diagnosis.



- Size is set to 0, which is out of range, reporting a 504 error (read Size exceeds limit or is 0) diagnosis.



- Message Transmission

  The function block requires the input parameter ConnID which must be in a connected state. When EN_R is FALSE, accept writes to ConnID, SrcName, DstName. Upon setting EN_R to TRUE, the function block will send an write request transmission.

  Output parameter states:

  - During execution: Done = FALSE, Busy = TRUE, Error = FALSE, ErrorId = 0.
  - Upon completion: Done = TRUE, Busy = FALSE, Error = FALSE, ErrorId = 0.
  - On error: Done = FALSE, Busy = FALSE, Error = TRUE, ErrorId = Non-zero.

8.2.6 CIPWrite (CIP Class3 Write Tag)

8.2.6.1 Version

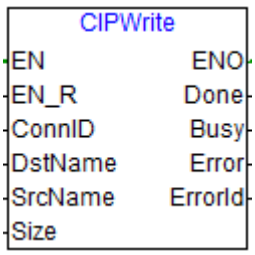
- AutoThink Software Version: V3.2.3SP2 and above.
- The controller firmware version is as follows:

Controller Type	Firmware Version

LX-CU430	LX-CU430_V1.1.1 and above.
LX-CU433	LX-CU433_V1.1.0 and above.
LX-CU500	LX-CU500_V1.4.0 and above.
LX-CU501	LX-CU501_V1.2.0 and above.
LX-CU510	LX-CU510_V1.1.0 and above.
LX-CU511	LX-CU511_V1.0.1 and above.

8.2.6.2 Function

Write the source tag value to the peer tag. SrcName specifies the source tag name, DstName specifies the peer tag name.



CIPWrite Functional Block

8.2.6.3 Precautions

- Data types such as DATE, DT, TIME, and TOD are not supported for read/write operations with Omron devices.
- A single BOOL variable in Omron occupies two bytes, while each BOOL in a BOOL array occupies one byte. Therefore, when communicating BOOL-type data with Omron devices, only arrays can be read or written, and the number of elements must be greater than one. Otherwise, Omron will determine the data length as mismatched and reject the read/write operation.
- The STRING data type does not support communication in array form.
- Writing an illegal address to a pointer can easily cause controller exceptions. Please use pointer types with caution.
- When the function block's Done signal becomes TRUE indicates that the write operation has completed successfully. Verify if the local DstName data area contains the newly written values. If values are present, operation successful. If no values are written, operation failed.

8.2.6.4 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN_R	INPUT	BOOL	Enable, rising edge triggers operation.	FALSE	FALSE
ConnID	INPUT	UINT	Valid ID, range: 1–32.	1	FALSE
DstName	INPUT	STRING(255)	Peer tag name, max length 255, excluded '\0'.	-	FALSE
SrcName	INPUT	STRING(255)	Source tag name, max length 255, excluded '\0'.	-	FALSE

Size	INPUT	UINT	Number of elements write from the peer tag Typically, the Size is 1. For array or string types, the Size indicates the actual number of elements read or written.	1	FALSE
Done	OUTPUT	BOOL	Command completion status. FALSE: Not completed. TRUE: Completed.	FALSE	FALSE
Busy	OUTPUT	BOOL	Command processing status FALSE: Idle. TRUE: Processing.	FALSE	FALSE
Error	OUTPUT	BOOL	Command error status FALSE: No error. TRUE: Error occurred.	FALSE	FALSE
ErrorId	OUTPUT	DINT	Error Code 0: Success. Non-zero: Specific values refer to the error code table.	0	FALSE

8.2.6.5 Error Code

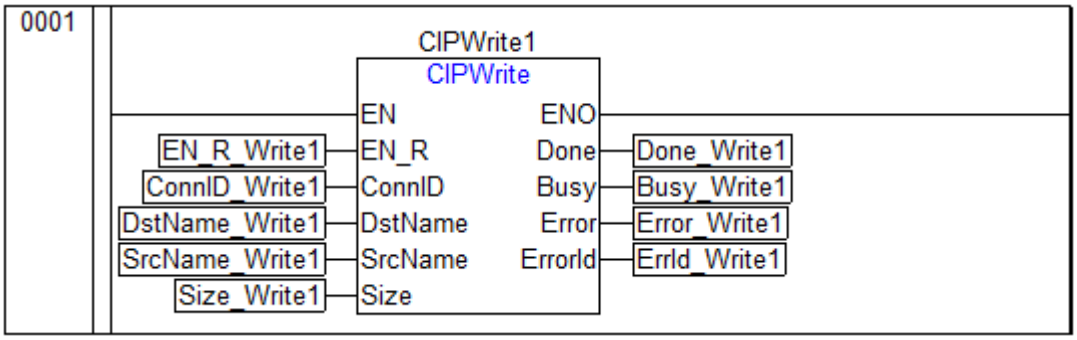
Error code	Description
0	No diagnostic information available.
501	The read target connection is closed.
502	Class 3 read limit exceeded.
503	Source tag array error. 1.Array tag - Invalid element format (cannot convert to number). 2.Array tag - Zero-length element. 3.Array tag - Missing closing bracket. 4.Standard tag - Tag does not exist or length is 0.
504	No source tag data available.
505	Unsupported tag data format.
506	Write size exceeds limit or is zero.
507	Received error response (peer returned exception): 1.Target tag not found on remote device. 2.Remote side detected size mismatch. 3.Unsupported data type by remote endpoint.
508	Target tag array error. 1.Array tag - Invalid element format (cannot convert to number). 2.Array tag - Zero-length element. 3.Array tag - Missing closing bracket. 4.Standard tag - Tag does not exist or length is 0.
509	Data type mismatch.
606	Invalid connection ID.
607	The connection is currently busy.

8.2.6.6 Example

1. Variable Definition

No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	CIPWrite1			CIPWRITE		FALSE	Not Public	
0002	EN_R_Write1			BOOL	FALSE	FALSE	Not Public	
0003	ConnID_Write1			UINT	0	FALSE	Not Public	
0004	DstName_Write1			STRING(255)	''	FALSE	Not Public	
0005	SrcName_Write1			STRING(255)	''	FALSE	Not Public	
0006	Size_Write1			UINT	0	FALSE	Not Public	
0007	Done_Write1			BOOL	FALSE	FALSE	Not Public	
0008	Busy_Write1			BOOL	FALSE	FALSE	Not Public	
0009	Error_Write1			BOOL	FALSE	FALSE	Not Public	
0010	ErrId_Write1			DINT	0	FALSE	Not Public	

2. LD Configuration Example



3. ST Configuration Example

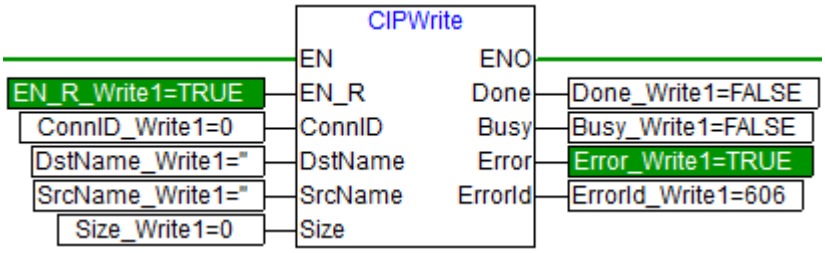
```
CIPWrite1(
  EN_R := EN_R_Write1,
  ConnID := ConnID_Write1,
  DstName := DstName_Write1,
  SrcName := SrcName_Write1,
  Size := Size_Write1,
  Done=>Done_Write1,
  Busy=>Busy_Write1,
  Error=>Error_Write1,
  ErrorId=>ErrId_Write1);
```

4. This example demonstrates how to use the CIPWrite function block.

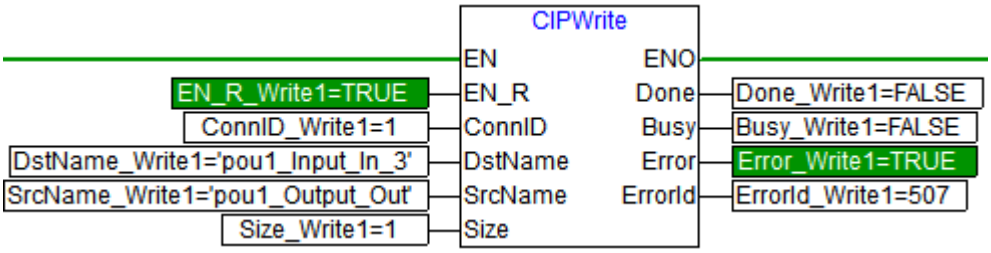
■ Parameter Check

The function block triggers parameter inspection on the rising edge of EN_R.

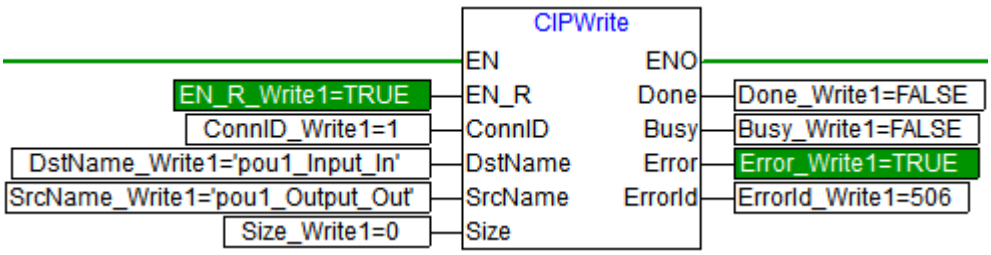
□ Valid ConnID range: 1 to 32. If ConnID is outside valid range, report diagnostic error code 606 (Invalid Connection ID).



- The destination tag DstName does not exist, reporting a 507 error (received incorrect response) diagnosis.



- The Size parameter is set to 0, which is out of range, reporting a 506 error (read Size exceeds limit or is 0) diagnosis.



- Message Transmission

The function block requires the input parameter ConnID which must be in a connected state. When EN_R is FALSE, accept writes to ConnID, SrcName, DstName. Upon setting EN_R to TRUE, the function block will send an write request transmission.

Output parameter states:

- During execution: Done = FALSE, Busy = TRUE, Error = FALSE, ErrorId = 0.
- Upon completion: Done = TRUE, Busy = FALSE, Error = FALSE, ErrorId = 0.
- On error: Done = FALSE, Busy = FALSE, Error = TRUE, ErrorId = Non-zero.

8.2.7 CIPClose (CIP Class3 Close Connection)

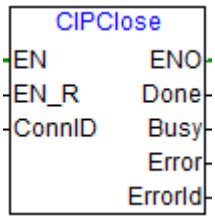
8.2.7.1 Version

- AutoThink Software Version: V3.2.3SP2 and above.
- The controller firmware version is as follows:

Controller Type	Firmware Version
LX-CU430	LX-CU430_V1.1.1 and above.
LX-CU433	LX-CU433_V1.1.0 and above.
LX-CU500	LX-CU500_V1.4.0 and above.
LX-CU501	LX-CU501_V1.2.0 and above.
LX-CU510	LX-CU510_V1.1.0 and above.
LX-CU511	LX-CU511_V1.0.1 and above.

8.2.7.2 Function

Close connection.



CIPClose Functional Block

8.2.7.3 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN_R	INPUT	BOOL	Enable, rising edge triggers operation.	FALSE	FALSE
ConnID	INPUT	UINT	Valid ID, range: 1–32.	1	FALSE
Done	OUTPUT	BOOL	Command completion status. FALSE: Not completed. TRUE: Completed.	FALSE	FALSE
Busy	OUTPUT	BOOL	Command processing status FALSE: Idle. TRUE: Processing.	FALSE	FALSE
Error	OUTPUT	BOOL	Command error status FALSE: No error. TRUE: Error occurred.	FALSE	FALSE
ErrorId	OUTPUT	DINT	Error Code 0: Success. Non-zero: Specific values refer to the error code table.	0	FALSE

8.2.7.4 Error Code

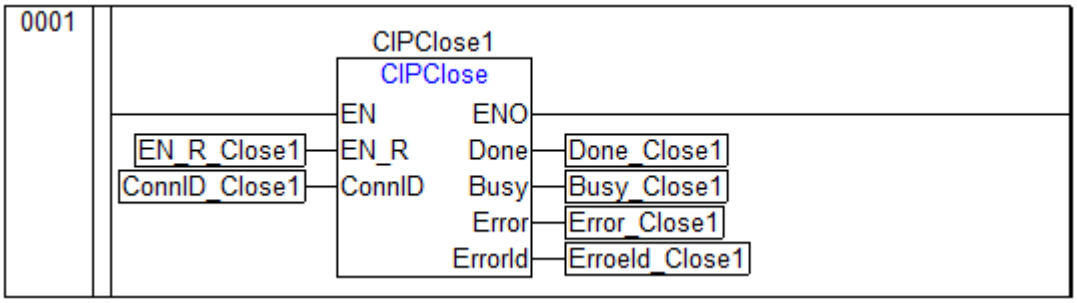
Error code	Description
0	Connection closed successfully.
606	The specified connection ID does not exist.
607	Connection is locked by another thread/process.
613	Duplicate connection closure attempt.

8.2.7.5 Example

1. Variable Definition

No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	CIPClose1			CIPCLOSE		FALSE	Not Public	
0002	EN_R_Close1			BOOL	FALSE	FALSE	Not Public	
0003	ConnID_Close1			UINT	0	FALSE	Not Public	
0004	Done_Close1			BOOL	FALSE	FALSE	Not Public	
0005	Busy_Close1			BOOL	FALSE	FALSE	Not Public	
0006	Error_Close1			BOOL	FALSE	FALSE	Not Public	
0007	ErroeId_Close1			DINT	0	FALSE	Not Public	

2. LD Configuration Example



3. ST Configuration Example

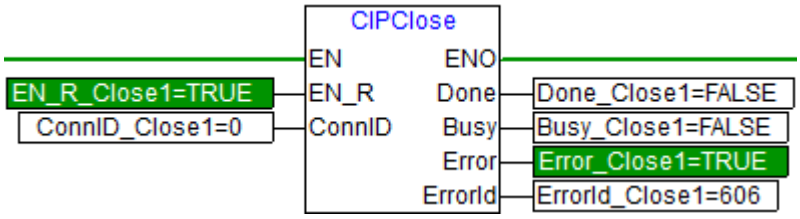
```
CIPClose1(
  EN_R := EN_R_Close1,
  ConnID := ConnID_Close1,
  Done=>Done_Close1,
  Busy=>Busy_Close1,
  Error=>Error_Close1,
  ErrorId=>ErroeId_Close1);
```

4. This example demonstrates how to use the CIPClose function block.

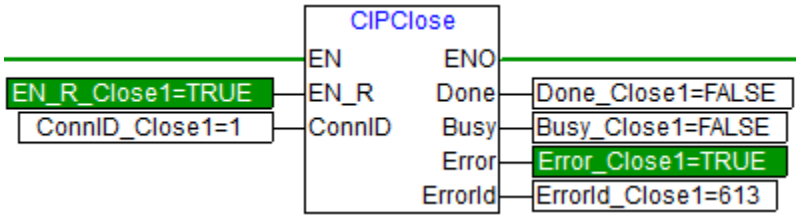
- Parameter Check

  The function block triggers parameter inspection on the rising edge of EN_R.

  - ConnID valid range: 1~32. Report diagnostic code 606 for out-of-range ConnID (Invalid Connection ID).



- When operating an already closed ConnID, report error 613 (Duplicate connection closure).



5. Message Transmission

   When the function block's EN_R is FALSE, input the ConnID, then set EN_R to TRUE—the rising edge triggers a close request.

   Output parameter states:

   - During execution: Done = FALSE, Busy = TRUE, Error = FALSE, ErrorId = 0.
   - Upon completion: Done = TRUE, Busy = FALSE, Error = FALSE, ErrorId = 0.
   - On error: Done = FALSE, Busy = FALSE, Error = TRUE, ErrorId = Non-zero.

8.2.8 CIPUCMMSend (Send CIP UCMM Explicit Message)

8.2.8.1 Version

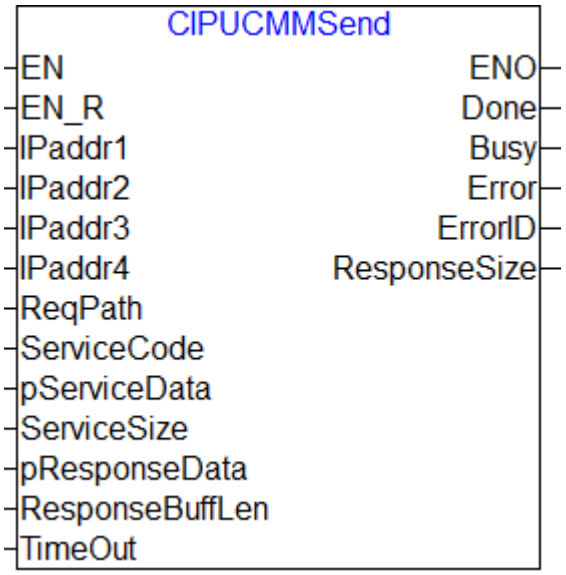
- AutoThink software version: V3.2.3 SP3 and above.
- Controller firmware version is as follows:

Controller Model	Firmware Version
LX-CU430	LX-CU430_V1.2.0 and above.
LX-CU433	LX-CU433_V1.1.0 and above.
LX-CU500	LX-CU500_V1.4.0 and above.
LX-CU501	LX-CU501_V1.2.0 and above.
LX-CU510	LX-CU510_V1.1.0 and above.
LX-CU511	LX-CU511_V1.1.0 and above.

8.2.8.2 Function

The CIPUCMMSend function block is used for connectionless, acyclic communication between the LX

controller acting as an EtherNet/IP Scanner and an EtherNet/IP Adaptor. This function block can send connectionless explicit messages to a specified EtherNet/IP Adaptor.



CIPUCMMSend function block

8.2.8.3 Precautions

The slave's response data is stored in a user-specified buffer. If the response data exceeds the buffer size, it will be truncated.

If the rising edge enable is cancelled while waiting for the execution result before completion, the execution will be aborted.

The maximum number of instances that can be configured to execute simultaneously does not exceed 32.

In AutoThink, add the EtherNet/IP Scanner protocol under the network port that needs to communicate with the slave station.

8.2.8.4 Parameters

Parameter	Declaration	Data Type	Description	Default Value	Retain
EN_R	INPUT	BOOL	Function block enable, rising edge trigger	FALSE	FALSE
IPaddr1	INPUT	BYTE	Remote IP address octet 1	0	FALSE
IPaddr2	INPUT	BYTE	Remote IP address octet 2	0	FALSE
IPaddr3	INPUT	BYTE	Remote IP address octet 3	0	FALSE
IPaddr4	INPUT	BYTE	Remote IP address octet 4	0	FALSE
ReqPath	INPUT	EIP_REQPATH	Request Path: ClassID, InstanceID, AttributeID.	-	FALSE
ServiceCode	INPUT	BYTE	Service code	0	FALSE

pServiceData	INPUT	POINTER TO BYTE	Address of service data (send data); If this is just a read operation, the content of the service data address will be ignored and not sent to the slave station.	0	FALSE
ServiceSize	INPUT	UINT	Length of service data (send data), maximum 500 bytes.	0	FALSE
pResponseData	INPUT	POINTER TO BYTE	Address of response data; Address of the buffer to store the data returned from the slave station.	0	FALSE
ResponseBuffLen	INPUT	UINT	Length of the response data buffer; If the length of data returned from the slave station exceeds the buffer length, the data will be truncated and an error will be reported.	0	FALSE
TimeOut	INPUT	UINT	Timeout time Unit: 0.1s Range: 1~65535	0	FALSE
Done	OUTPUT	BOOL	Command completion status, FALSE indicates not complete, TRUE indicates complete	FALSE	FALSE
Busy	OUTPUT	BOOL	Command processing status, FALSE indicates idle, TRUE indicates processing	FALSE	FALSE
Error	OUTPUT	BOOL	Command error status, FALSE indicates no error, TRUE indicates error	FALSE	FALSE
ErrorId	OUTPUT	WORD	Error code: 0: No error 1: Communication connection error 2: Resources required for the object to execute the requested service are unavailable 3: Invalid parameter value 4: Path segment error 5: Path destination unknown 6: Only part of the expected data was transmitted 7: Message connection lost 8: Requested service is not implemented or defined in this object class/instance 9: Invalid attribute value 10: An attribute in the Get_Attribute_List or Set_Attribute_List response has a non-zero status 11: Object is already in the mode/state requested by the service 12: Object state conflict 13: The object instance requested to be created already exists 14: Attribute cannot be set 15: Permission check failed 16: Device state conflict 256: Remote IP address error, cannot be 0.0.0.0 or *.*.*.255 or multicast address (224.0.0.255~239.255.255.255) 257: Service data (send data) address is invalid 258: Service data (send data) address length error (must not exceed 500 bytes) 259: Response data address is invalid 260: Timeout time error (cannot be 0) 261: Exceeds maximum number of instances (32) 262: System error 263: UCMM not in use (EIP Scanner protocol not configured) 264: Communication timeout 265: Response data exceeds the receive buffer size	0	FALSE
ResponseSize	OUTPUT	UINT	Response data length	0	FALSE

8.3 CANopen Functions(LX)

The LX-CM007 module functions as a CANopen master module, capable of connecting to CANopen slave devices and converting the CANopen protocol to the EtherCAT protocol. The LX-CM007 supports three commands: LXCANopenReadDict, LXCANopenWriteDict, and LXCANopenNodeNMT. These commands enable read/write operations on the object dictionary of CANopen slave devices and facilitate switching the state of slave nodes.

8.3.1 LXCANopenReadDict(Read CANopen Slave EDS)

8.3.1.1 Version

- AutoThink Software Version: V3.2.3SP1 and above.
- The controller firmware version is as follows:

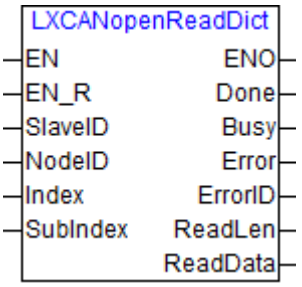
Controller Type	Firmware Version
LX-CU430	LX-CU430_V1.1.0 and above.
LX-CU433	LX-CU433_V1.0.0 and above.
LX-CU500	LX-CU500_V1.3.0 and above.
LX-CU501	LX-CU501_V1.1.0 and above.
LX-CU510	LX-CU510_V1.0.0 and above.
LX-CU511	LX-CU511_V1.0.0 and above.

8.3.1.2 Function

Used to retrieve parameter values from the object dictionary of a CANopen slave node.

When EN_R is set to TRUE, the instruction is triggered on the rising edge. After execution is completed, EN_R must be manually set to FALSE. If EN_R is set to FALSE during the instruction execution process, the operation will be interrupted.

Simply setting EN_R to TRUE will trigger the instruction to execute once.



LXCANopenReadDict Functional Block

8.3.1.3 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN_R	INPUT	BOOL	Rising edge triggered	FALSE	FALSE
SlaveID	INPUT	WORD	EtherCAT address of the CANopen master	0	FALSE
NodeID	INPUT	BYTE	CANopen slave nodes: 1~126	0	FALSE
Index	INPUT	WORD	CANopen slave parameter index: 0x1000~0x9FFF	0	FALSE
SubIndex	INPUT	BYTE	CANopen slave parameter subindex	0	FALSE
Done	OUTPUT	BOOL	Whether reading the CANopen slave node object dictionary parameters is completed: FALSE: Not completed TRUE: Completed	FALSE	FALSE
Busy	OUTPUT	BOOL	Operation status of reading CANopen slave node object dictionary parameters: FALSE: Not started or already finished TRUE: In progress	FALSE	FALSE
Error	OUTPUT	BOOL	Error Indicator: FALSE: No error TRUE: Error detected	FALSE	FALSE
ErrorID	OUTPUT	DWORD	Error codes	0	FALSE
ReadLen	OUTPUT	WORD	Size of the read object dictionary parameter: 1: The read object dictionary parameter size is 1 byte 2: The read object dictionary parameter size is 2 bytes 4: The read object dictionary parameter size is 4 bytes	0	FALSE
ReadData	OUTPUT	DWORD	Value of the read object dictionary parameter	0	FALSE

8.3.2 LXCANopenWriteDict(Write CANopen Slave EDS)

8.3.2.1 Version

- AutoThink Software Version: V3.2.3SP1 and above.
- The controller firmware version is as follows:

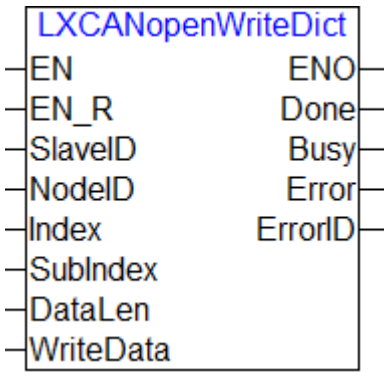
Controller Type	Firmware Version
LX-CU430	LX-CU430_V1.1.0 and above.
LX-CU433	LX-CU433_V1.0.0 and above.
LX-CU500	LX-CU500_V1.3.0 and above.
LX-CU501	LX-CU501_V1.1.0 and above.
LX-CU510	LX-CU510_V1.0.0 and above.
LX-CU511	LX-CU511_V1.0.0 and above.

8.3.2.2 Function

Used to write values to the object dictionary of a CANopen slave node.

When EN_R is set to TRUE, the instruction is triggered on the rising edge. After execution is completed, EN_R must be manually set to FALSE. If EN_R is set to FALSE during the instruction execution process, the operation will be interrupted.

Simply setting EN_R to TRUE will trigger the instruction to execute once.



LXCANopenWriteDict Functional Block

8.3.2.3 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN_R	INPUT	BOOL	Rising edge triggered	FALSE	FALSE
SlaveID	INPUT	WORD	EtherCAT address of the CANopen master	0	FALSE
NodeID	INPUT	BYTE	CANopen slave nodes: 1~126	0	FALSE
Index	INPUT	WORD	CANopen slave parameter index: 0x1000~0x9FFF	0	FALSE
SubIndex	INPUT	BYTE	CANopen slave parameter subindex	0	FALSE
DataLen	INPUT	WORD	Size of the write object dictionary parameter: 1: The read object dictionary parameter size is 1 byte 2: The read object dictionary parameter size is 2 bytes 4: The read object dictionary parameter size is 4 bytes	0	FALSE
WriteData	INPUT	DWORD	Value of the write object dictionary parameter	0	FALSE
Done	OUTPUT	BOOL	Whether updating the object dictionary parameters of the CANopen slave node is completed: FALSE: Not completed TRUE: Completed	FALSE	FALSE
Busy	OUTPUT	BOOL	Status of updating the object dictionary parameters of the CANopen slave node: FALSE: Not started or has ended TRUE: In progress	FALSE	FALSE
Error	OUTPUT	BOOL	Error flag: FALSE: No error TRUE: Error present	FALSE	FALSE
ErrorID	OUTPUT	DWORD	Error codes	0	FALSE

8.3.3 LXCANopenNodeNMT(Send NMT command specifier)

8.3.3.1 Version

- AutoThink Software Version: V3.2.3SP1 and above.
- The controller firmware version is as follows:

Controller Type	Firmware Version
LX-CU430	LX-CU430_V1.1.0 and above.
LX-CU433	LX-CU433_V1.0.0 and above.
LX-CU500	LX-CU500_V1.3.0 and above.
LX-CU501	LX-CU501_V1.1.0 and above.
LX-CU510	LX-CU510_V1.0.0 and above.
LX-CU511	LX-CU511_V1.0.0 and above.

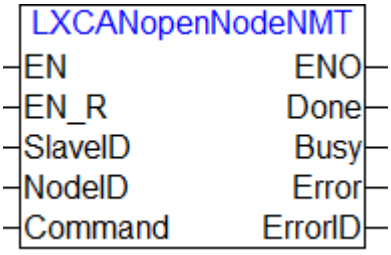
8.3.3.2 Function

Sends an NMT command to a CANopen slave node to switch its state.

When EN_R is set to TRUE, the instruction is triggered on the rising edge. After execution is completed, EN_R must be manually set to FALSE. If EN_R is set to FALSE during the instruction execution, the operation will be interrupted.

Simply setting EN_R to TRUE will trigger the instruction to execute once.

Completion of the NMT command transmission only indicates that the CANopen master has sent the NMT command to the specified slave node. It does not guarantee that the slave has successfully executed the command. The execution status can be verified through error codes and the connection state in the slave diagnostic information.



LXCANopenNodeNMT Functional Block

8.3.3.3 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN_R	INPUT	BOOL	Rising edge triggered	FALSE	FALSE
SlaveID	INPUT	WORD	EtherCAT address of the CANopen master	0	FALSE
NodeID	INPUT	BYTE	CANopen slave nodes: 1~126	0	FALSE
Command	INPUT	DWORD	NMT command word settings: 0x01: Start remote node 0x02: Stop remote node 0x80: Enter pre-operational state 0x81: Application reset 0x82: Communication reset	0	FALSE
Done	OUTPUT	BOOL	Whether the command has been sent completely: FALSE: Not completed TRUE: Completed	FALSE	FALSE
Busy	OUTPUT	BOOL	Command send status: FALSE: Not started or has ended TRUE: In progress	FALSE	FALSE
Error	OUTPUT	BOOL	Error flag: FALSE: No error TRUE: Error present	FALSE	FALSE
ErrorID	OUTPUT	DWORD	Error codes	0	FALSE

8.4 DeviceNet Functions

8.4.1 DeviceNet_Atrribute_WriteRead (Write Read DeviceNet Slave Atrribute)

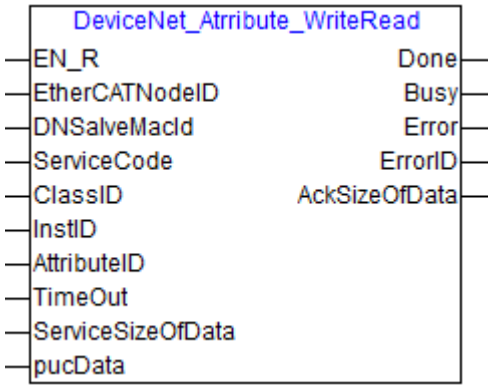
8.4.1.1 Version

- AutoThink Software Version: V3.2.3 and above.
- The controller firmware version is as follows:

Controller Type	Firmware Version
LX-CU430	LX-CU430_V1.1.0 and above.
LX-CU433	LX-CU433_V1.0.0 and above.
LX-CU500	LX-CU500_V1.3.0 and above.
LX-CU501	LX-CU501_V1.1.0 and above.
LX-CU510	LX-CU510_V1.0.0 and above.
LX-CU511	LX-CU511_V1.0.0 and above.

8.4.1.2 Function

Use the DeviceNet_Attribute_WriteRead function block to set and read the attribute values of DeviceNet slave stations.



DeviceNet_Atrribute_WriteRead Functional Block

8.4.1.3 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN_R	Input	BOOL	Enable	FALSE	FALSE
EtherCATNodeID	Input	UDINT	The EtherCAT slave address for the DeviceNet master	0	FALSE
DNSalveMacID	Input	BYTE	Device Netslave MAC ID（0-63）	0	FALSE
ServiceCode	Input	BOOL	Service Code: 0: Set a single attribute value (0x10) 1: Read a single attribute value (0x0E)	FALSE	FALSE
ClassID	Input	WORD	Class ID (the DeviceNet slave class ID to be set or read)	0	FALSE
InstID	Input	WORD	Instance ID (the DeviceNet slave instance ID to be set or read)	0	FALSE
AttributeID	Input	BYTE	Attribute（0-255）	0	FALSE
TimeOut	Input	WORD	Timeout for reading or setting attributes: The timeout period for waiting for a response after sending data. It is recommended to set this to 500ms (for service data lengths less than 4).	0	FALSE
ServiceSizeOfData	Input	BYTE	Service data length（0-230）	0	FALSE
pucData	Input	POINTER TO BYTE	When setting an attribute value: The attribute value to be set. When reading an attribute value: The attribute value that has been read (if the same data buffer is used for both setting and reading, the values in the buffer should be cleared before reading).	0	FALSE
Done	Output	BOOL	Set the flag after receiving a correct response from the DeviceNet slave.	FALSE	FALSE
Busy	Output	BOOL	Enter the Busy state if no response is received within the timeout period.	FALSE	FALSE
Error	Output	BOOL	Setting/Getting the attribute value failed.	FALSE	FALSE

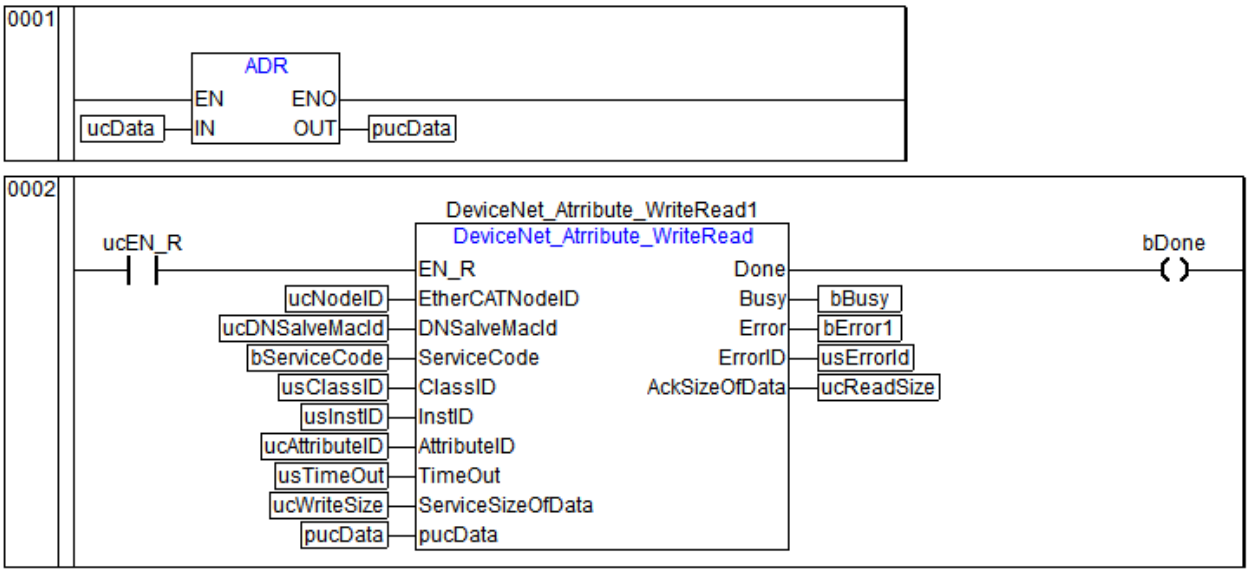
ErrorID	Output	WORD	Error Codes: 0: Setting/Reading attribute value succeeded. 1: Invalid DeviceNet slave MAC ID setting. 2: Invalid service data length setting. 3: EtherCAT read/write parameter failed. 4: Failed to copy the attribute data for reading. 5: Explicit message sending failed although the set/read attribute function block did not time out. 6: The connection between the master and the slave is in an abnormal state (not in IO communication mode). 7: Explicit message timed out although the set/read attribute function block did not time out. 8: The DeviceNet slave does not support setting/reading this class/instance/attribute value. 9: The set/read attribute function block timed out.	0	FALSE
AckSizeOfData	Output	BYTE	Response data length.	0	FALSE

8.4.1.4 Example

1. Variable Definition

No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	ucData			ARRAY[0..254] OF BYTE		FALSE	Not Public	
0002	pucData			POINTER TO BYTE	0	FALSE	Not Public	
0003	DeviceNet_Atrribute_WriteRead1			DEVICENET_ATRRIBUTE_WRITEREAD		FALSE	Not Public	
0004	ucEN_R			BOOL	FALSE	FALSE	Not Public	
0005	ucNodeID			UDINT	0	FALSE	Not Public	
0006	ucDNSalveMacId			BYTE	0	FALSE	Not Public	
0007	bServiceCode			BOOL	FALSE	FALSE	Not Public	
0008	usClassID			WORD	0	FALSE	Not Public	
0009	usInstID			WORD	0	FALSE	Not Public	
0010	ucAttributeID			BYTE	0	FALSE	Not Public	
0011	usTimeOut			WORD	0	FALSE	Not Public	
0012	ucWriteSize			BYTE	0	FALSE	Not Public	
0013	bDone			BOOL	FALSE	FALSE	Not Public	
0014	bBusy			BOOL	FALSE	FALSE	Not Public	
0015	bError1			BOOL	FALSE	FALSE	Not Public	
0016	usErrorId			WORD	0	FALSE	Not Public	
0017	ucReadSize			BYTE	0	FALSE	Not Public	

2. LD Configuration Example



3. ST Configuration Example

```
pucData := ADR(ucData);
DeviceNet_Atrribute_WriteRead1(
   EN_R := ucEN_R,
   EtherCATNodeID := ucNodeID,
   DNSalveMacId := ucDNSalveMacId,
   ServiceCode := bServiceCode,
   ClassID := usClassID,
   InstID := usInstID,
   AttributeID := ucAttributeID,
   TimeOut := usTimeOut,
   ServiceSizeOfData := ucWriteSize,
   pucData := pucData,
   Done=>bDone,
   Busy=>bBusy,
   Error=>bError1,
   ErrorID=>usErrorId,
   AckSizeOfData=>ucReadSize);
```

4. Through the LD program example, setting and reading attribute values of a deviceNet slave device. The operation steps are as follows:

   ■ Setting Attribute Values of a DeviceNet Slave Device

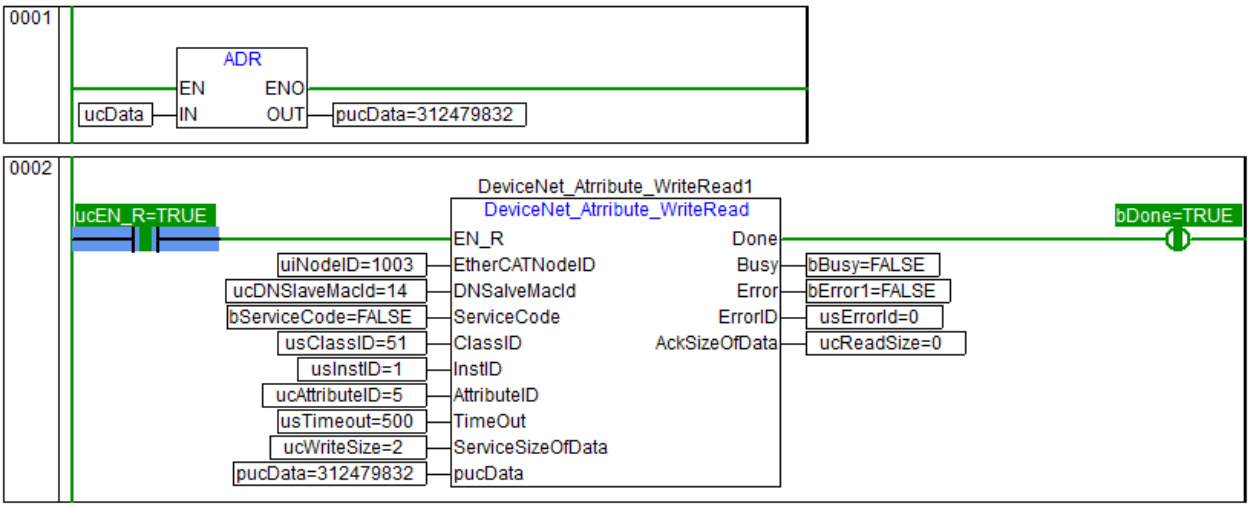
   (a) Double-click the ucData index column to set the attribute values of the DeviceNet slave device.

No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	ucData			ARRAY[0..254] OF BYTE		FALSE	Not Public	
0002	pucData			POINTER TO BYTE	0	FALSE	Not Public	

Main.ucData

No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	ucData[0]			BYTE	51	FALSE	Not Public	
0002	ucData[1]			BYTE	68	FALSE	Not Public	
0003	ucData[2]			BYTE	0	FALSE	Not Public	
0004	ucData[3]			BYTE	0	FALSE	Not Public	

(b) Enable EN_R after downloading the monitoring program. The figure below shows that the attribute values have been set successfully.



■ Reading Attribute Values of a DeviceNet Slave Device

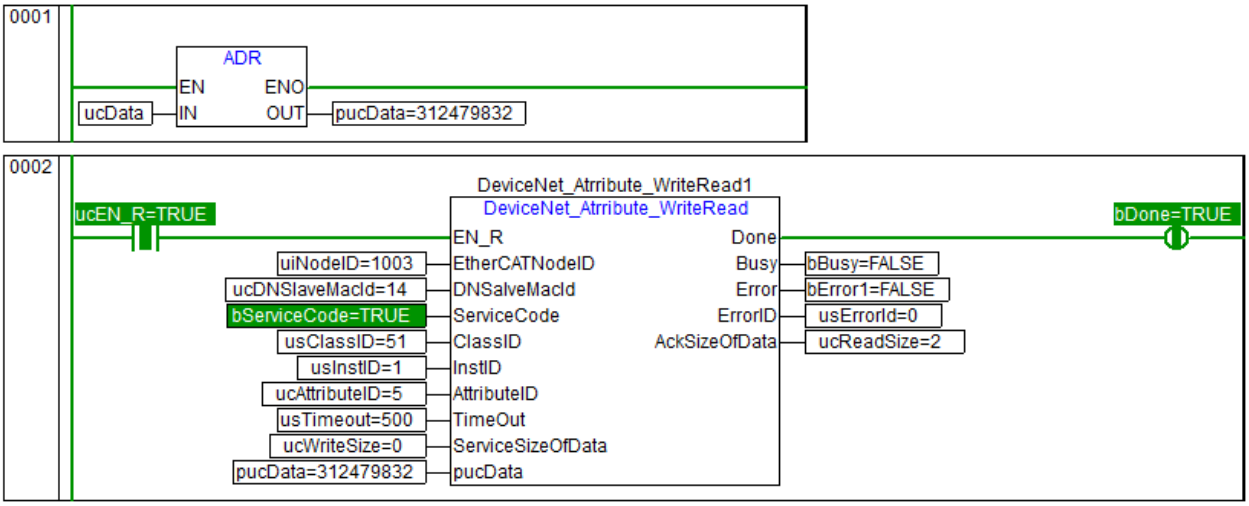
(a) Set the value of ucWriteSize to 0, set bServiceCode to TRUE, and clear the data in ucData.

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	ucData			ARRAY[0..254] OF BYTE		FALSE	Not Public	
0002	pucData			POINTER TO BYTE	1543901216	FALSE	Not Public	
0003	DeviceNet_Atrribute_WriteRead1			DEVICENET_ATRRIBUTE_WRITEREAD		FALSE	Not Public	

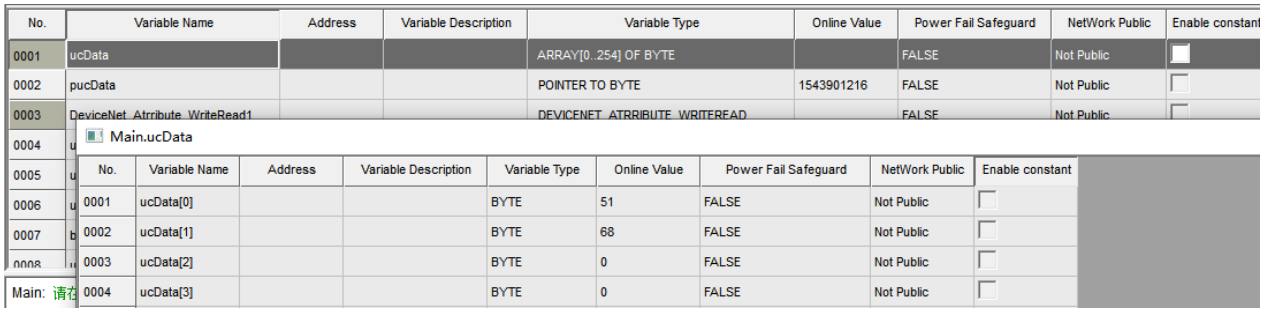
Main.ucData

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	ucData[0]			BYTE	0	FALSE	Not Public	
0002	ucData[1]			BYTE	0	FALSE	Not Public	
0003	ucData[2]			BYTE	0	FALSE	Not Public	
0004	ucData[3]			BYTE	0	FALSE	Not Public	

(b) Re-enable EN_R to successfully read the attribute values, as shown in the figure below.



(c) Set ucReadSize to 2, and the data in ucData contains the read attribute values.



No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	ucData			ARRAY[0..254] OF BYTE		FALSE	Not Public	
0002	pucData			POINTER TO BYTE	1543901216	FALSE	Not Public	
0003	DeviceNet_Atrribute_WriteRead1			DEVICENET_ATRRIBUTE_WRITEREAD		FALSE	Not Public	

Main.ucData

No.	Variable Name	Address	Variable Description	Variable Type	Online Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	ucData[0]			BYTE	51	FALSE	Not Public	
0002	ucData[1]			BYTE	68	FALSE	Not Public	
0003	ucData[2]			BYTE	0	FALSE	Not Public	
0004	ucData[3]			BYTE	0	FALSE	Not Public	

8.4.2 DeviceNet_DiagInfo_Read (Read DeviceNet Slave Diag Info)

8.4.2.1 Version

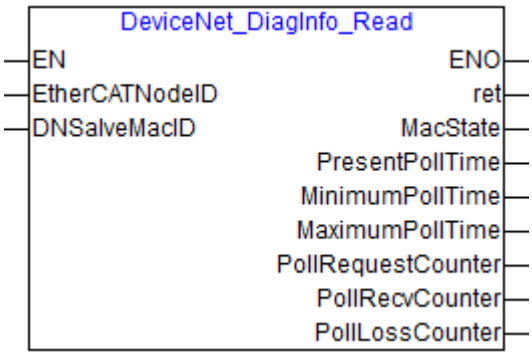
- AutoThink Software Version: V3.2.3 and above.
- The controller firmware version is as follows:

Controller Type	Firmware Version
LX-CU430	LX-CU430_V1.1.0 and above.
LX-CU433	LX-CU433_V1.0.0 and above.
LX-CU500	LX-CU500_V1.3.0 and above.
LX-CU501	LX-CU501_V1.1.0 and above.
LX-CU510	LX-CU510_V1.0.0 and above.
LX-CU511	LX-CU511_V1.0.0 and above.

8.4.2.2 Function

Use the DeviceNet_DiagInfo_Read function block to read the polling time, number of packets sent,

number of packets received, and number of packets lost from the DeviceNet slave device.



DeviceNet_DiagInfo_Read Functional Block

8.4.2.3 Parameter

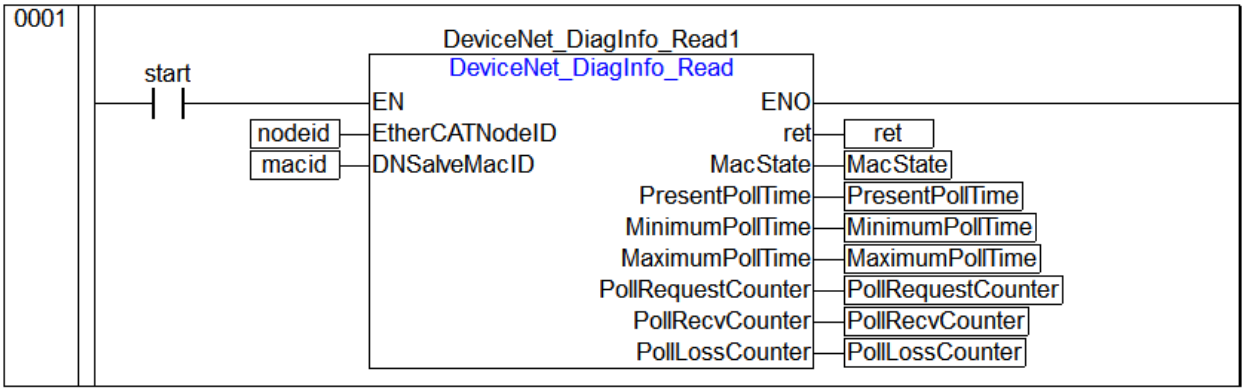
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EtherCATNodeID	Input	UDINT	The EtherCAT slave address for the DeviceNet master	0	FALSE
DNSlaveMacID	Input	BYTE	DeviceNet slave MacID	0	FALSE
ret	Output	UDINT	Return Values: 0: Read successful 1: Slave MacID is greater than 63 Others: See EtherCAT error codes	0	FALSE
MacState	Output	BYTE	Slave Status	0	FALSE
PresentPollTime	Output	UINT	Current Polling Time of Slave (ms)	0	FALSE
MinimumPollTime	Output	UINT	Minimum Polling Time of Slave (ms)	0	FALSE
MaximumPollTime	Output	UINT	Maximum Polling Time of Slave (ms)	0	FALSE
PollRequestCounter	Output	UDINT	Number of Slave Polling Requests	0	FALSE
PollRecvCounter	Output	UDINT	Number of Slave Polling Responses	0	FALSE
PollLossCounter	Output	UDINT	Number of Slave Polling Packet Losses	0	FALSE

8.4.2.4 Example

1. Variable Definition

No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	DeviceNet_DiagInfo_Read1			DEVICENET_DIAGINFO_READ		FALSE	Not Public	
0002	start			BOOL	FALSE	FALSE	Not Public	
0003	nodeid			UDINT	0	FALSE	Not Public	
0004	macid			BYTE	0	FALSE	Not Public	
0005	ret			UDINT	0	FALSE	Not Public	
0006	MacState			BYTE	0	FALSE	Not Public	
0007	PresentPollTime			UINT	0	FALSE	Not Public	
0008	MinimumPollTime			UINT	0	FALSE	Not Public	
0009	MaximumPollTime			UINT	0	FALSE	Not Public	
0010	PollRequestCounter			UDINT	0	FALSE	Not Public	
0011	PollRecvCounter			UDINT	0	FALSE	Not Public	
0012	PollLossCounter			UDINT	0	FALSE	Not Public	

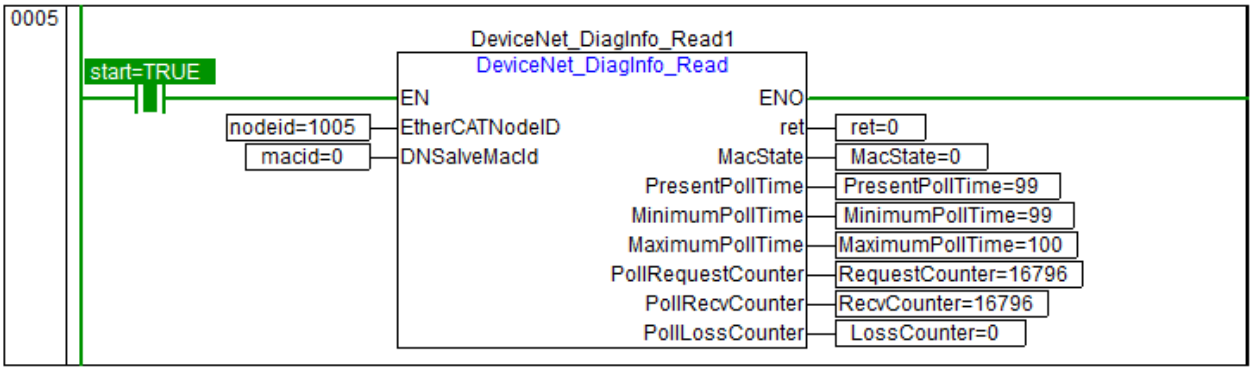
2. LD Configuration Example



3. ST Configuration Example

```
DeviceNet_DiagInfo_Read1(
    EtherCATNodeID := nodeid,
    DNSalveMacID := macid,
    ret=>ret,
    MacState=>MacState,
    PresentPollTime=>PresentPollTime,
    MinimumPollTime=>MinimumPollTime,
    MaximumPollTime=>MaximumPollTime,
    PollRequestCounter=>PollRequestCounter,
    PollRecvCounter=>PollRecvCounter,
    PollLossCounter=>PollLossCounter);
```

4. After enabling, the function block will read the polling time, number of packets sent, number of packets received, and number of packet losses for the DeviceNet slave. When the slave polling cycle time is 100ms, the information read by DeviceNet_DiagInfo_Read is as follows.



8.5 PROFINET Functions

8.5.1 PN_Get_Channel_Diag (Read PROFINET Slave Channel Diag Info)

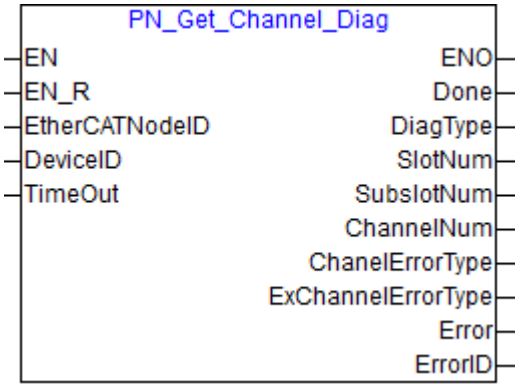
8.5.1.1 Version

- AutoThink Software Version: V3.2.3 and above.
- The controller firmware version is as follows:

Controller Type	Firmware Version
LX-CU430	LX-CU430_V1.1.0 and above.
LX-CU433	LX-CU433_V1.0.0 and above.
LX-CU500	LX-CU500_V1.3.0 and above.
LX-CU501	LX-CU501_V1.1.0 and above.
LX-CU510	LX-CU510_V1.0.0 and above.
LX-CU511	LX-CU511_V1.0.0 and above.

8.5.1.2 Function

Read the PROFINET slave channel diagnostic values.



PN_Get_Channel_Diag Functional Block

8.5.1.3 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN_R	Input	BOOL	Rising edge enable	FALSE	FALSE
EtherCATNodeID	Input	UDINT	EtherCAT slave address where the LX-CM022 module is located	0	FALSE
DeviceID	Input	WORD	PN slave device number	0	FALSE
TimeOut	Input	WORD	Timeout for waiting for a response (unit: ms), recommended to set to 100ms	0	FALSE
Done	Output	BOOL	FALSE: The channel is either fault-free or faulty, but the channel diagnosis acquisition has failed TRUE: Channel diagnostic acquisition succeeded	FALSE	FALSE
DiagType	Output	WORD	Diagnostic type 0: No fault has occurred in the slave channel 1: Channel diagnosis 2: Extended channel diagnosis	0	FALSE
SlotNum	Output	WORD	Slot number of the faulty PROFINET slave device	0	FALSE
SubSlotNum	Output	WORD	Sub-slot number of the faulty PROFINET slave device	0	FALSE
ChannelNum	Output	WORD	Channel number of the faulty PROFINET slave device	0	FALSE
ChannelErrorType	Output	WORD	Channel error type	0	FALSE
ExChannelErrorType	Output	WORD	Channel extended error type	0	FALSE
Error	Output	BOOL	FALSE: Channel diagnostic acquisition succeeded TRUE: The channel is either fault-free or faulty, but the channel diagnosis acquisition has failed	FALSE	FALSE
ErrorID	Output	BYTE	Error code 0: No error 1: The PROFINET slave channel diagnosis was successfully read 2: EtherCAT write failed 3: EtherCAT read failed 4: DeviceID is not within the range of 1~2047 5: No channel fault has occurred in the slave 6: Timeout	0	FALSE

8.5.2 PN_Get_Warning_Diag (Read PROFINET Slave Warning Diag Info)

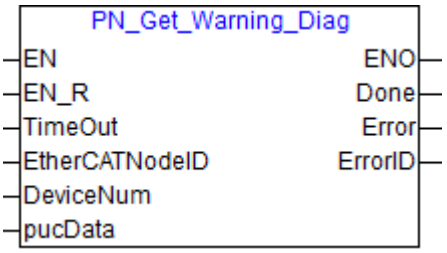
8.5.2.1 Version

- AutoThink Software Version: V3.2.3 and above.
- The controller firmware version is as follows:

Controller Type	Firmware Version
LX-CU430	LX-CU430_V1.1.0 and above.
LX-CU433	LX-CU433_V1.0.0 and above.
LX-CU500	LX-CU500_V1.3.0 and above.
LX-CU501	LX-CU501_V1.1.0 and above.
LX-CU510	LX-CU510_V1.0.0 and above.
LX-CU511	LX-CU511_V1.0.0 and above.

8.5.2.2 Function

Read the PROFINET slave warning diagnostic values.



PN_Get_Warning_Diag Functional Block

8.5.2.3 Parameter

Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN_R	Input	BOOL	Rising edge enable	FALSE	FALSE
TimeOut	Input	WORD	Timeout for waiting for a response (unit: ms), recommended to set to 100ms	0	FALSE
EtherCATNodeID	Input	UDINT	EtherCAT slave address where the LX-CM022 module is located	0	FALSE
DeviceNum	Input	WORD	Number of configured PN slaves	0	FALSE
pucData	Input	POINTER TO BYTE	Warning diagnostic data	0	FALSE
Done	Output	BOOL	FALSE: Warning diagnostic acquisition failed TRUE: Warning diagnostic acquisition succeeded	FALSE	FALSE
Error	Output	BOOL	FALSE: Warning diagnostic acquisition	FALSE	FALSE

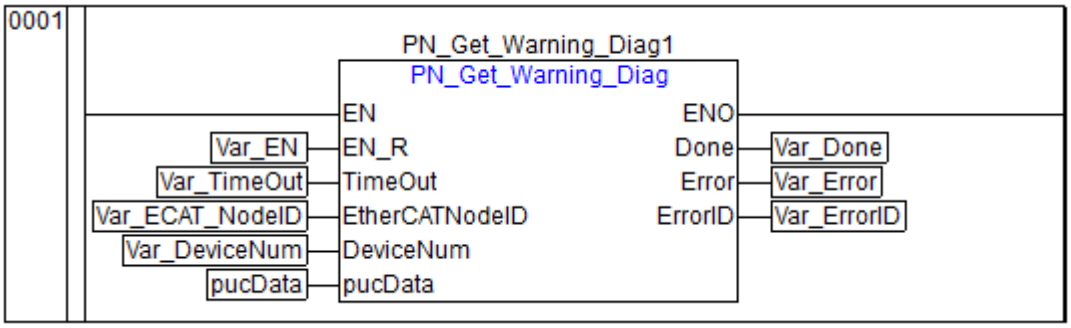
			succeeded TRUE: Warning diagnostic acquisition failed		
ErrorID	Output	BYTE	Error code 0: No error 1: DeviceNum is greater than 25 2: EtherCAT write failed 3: EtherCAT read failed 4: Data copy failed 6: Timeout	0	FALSE

8.5.2.4 Example

1. Variable Definition

No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	PN_Get_Warning_Diag1			PN_GET_WARNING_DIAG		FALSE	Not Public	
0002	Var_EN			BOOL	FALSE	FALSE	Not Public	
0003	Var_TimeOut			WORD	0	FALSE	Not Public	
0004	Var_ECAT_NodeID			UDINT	0	FALSE	Not Public	
0005	Var_DeviceNum			WORD	0	FALSE	Not Public	
0006	pucData			POINTER TO BOOL	0	FALSE	Not Public	
0007	Var_Done			BOOL	FALSE	FALSE	Not Public	
0008	Var_Error			BOOL	FALSE	FALSE	Not Public	
0009	Var_ErrorID			BYTE	0	FALSE	Not Public	

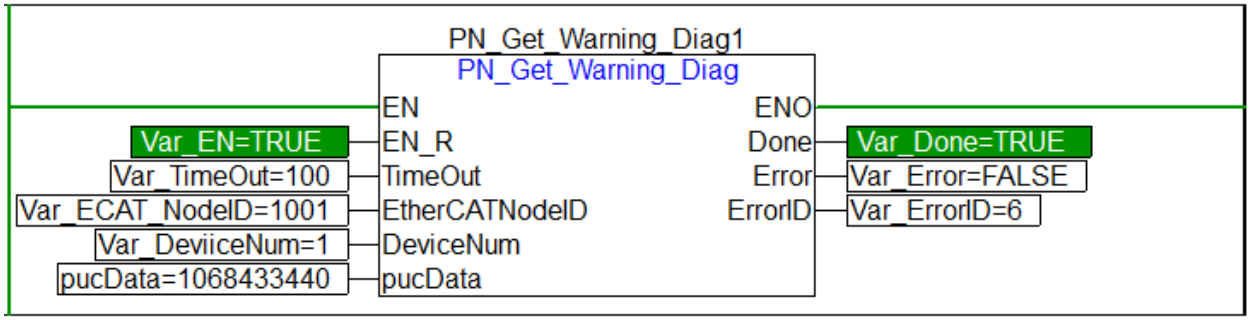
2. LD Configuration Example



3. ST Configuration Example

```
PN_Get_Warning_Diag1(
    EN_R := Var_EN,
    TimeOut := Var_TimeOut,
    EtherCATNodeID := Var_ECAT_NodeID,
    DeviceNum := Var_DeviceNum,
    pucData := pucData,
    Done=>Var_Done,
    Error=>Var_Error,
    ErrorID=>Var_ErrorID);
```

4. Use the LD program to read the PROFINET slave alarm diagnostic errors. As shown in the figure below, a timeout error occurred when reading the channel diagnostic information of the PROFINET slave.



8.5.3 PN_Get_Mismatch_Diag (Read PROFINET Slave Mismatch Diag Info)

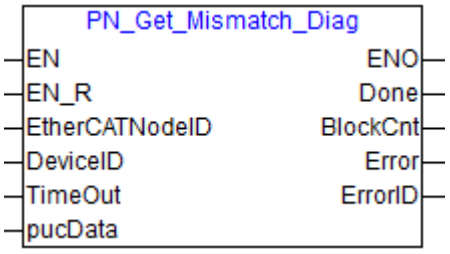
8.5.3.1 Version

- AutoThink Software Version: V3.2.3 and above.
- The controller firmware version is as follows:

Controller Type	Firmware Version
LX-CU430	LX-CU430_V1.1.0 and above.
LX-CU433	LX-CU433_V1.0.0 and above.
LX-CU500	LX-CU500_V1.3.0 and above.
LX-CU501	LX-CU501_V1.1.0 and above.
LX-CU510	LX-CU510_V1.0.0 and above.
LX-CU511	LX-CU511_V1.0.0 and above.

8.5.3.2 Function

Read the PROFINET slave slot mismatch diagnostic values.



PN_Get_Mismatch_Diag Functional Block

8.5.3.3 Parameter

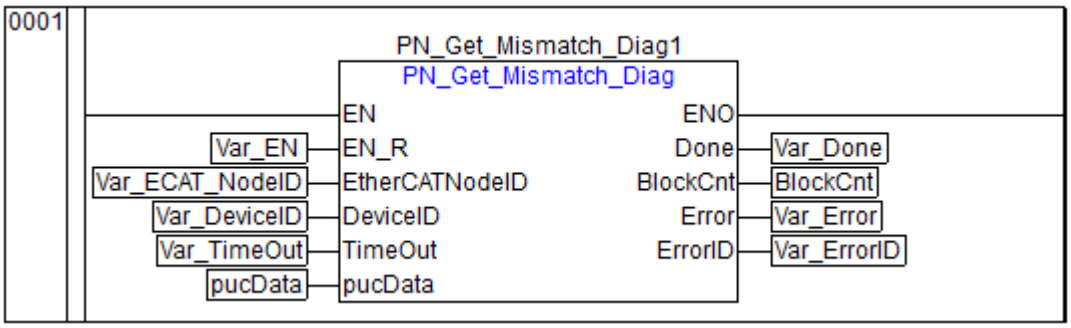
Parameter	Statement	Data Type	Description	Initial Value	Power Fail Safeguard
EN_R	Input	BOOL	Rising edge enable	FALSE	FALSE
EtherCATNodeID	Input	UDINT	EtherCAT slave address where the LX-CM022 module is located	0	FALSE
DeviceID	Input	WORD	PN slave device number	0	FALSE
TimeOut	Input	WORD	Timeout for waiting for a response (unit: ms), recommended to set to 100ms	0	FALSE
pucData	Input	POINTER TO BYTE	Module mismatch diagnostic data	0	FALSE
Done	Output	BOOL	FALSE: No slot mismatch fault in the slave, or a slot mismatch fault exists but the fault diagnosis acquisition has failed TRUE: Slot mismatch fault in the slave diagnostic acquisition succeeded	FALSE	FALSE
BlockCnt	Output	WORD	Number of mismatched slots	0	FALSE
Error	Output	BOOL	FALSE: Slot mismatch fault in the slave diagnostic acquisition succeeded TRUE: No slot mismatch fault in the slave, or a slot mismatch fault exists but the fault diagnosis acquisition has failed	FALSE	FALSE
ErrorID	Output	BYTE	Error code 0: No slot mismatch fault in the slave 1: The read device does not match the diagnostic returned device number 2: EtherCAT write failed 3: EtherCAT read failed 4: Data copy failed 5: The slave has not undergone module mismatch diagnostics 6: Timeout 7: BlockCnt exceeds 128 or BlockCnt is 0 8: DeviceID is not within the range of 1~2047	0	FALSE

8.5.3.4 Example

1. Variable Definition

No.	Variable Name	Address	Variable Description	Variable Type	Initial Value	Power Fail Safeguard	NetWork Public	Enable constant
0001	PN_Get_Mismatch_Diag1			PN_GET_MISMATCH_DIAG		FALSE	Not Public	
0002	Var_EN			BOOL	FALSE	FALSE	Not Public	
0003	Var_ECAT_NodeID			UDINT	0	FALSE	Not Public	
0004	Var_DeviceID			WORD	0	FALSE	Not Public	
0005	Var_TimeOut			WORD	0	FALSE	Not Public	
0006	pucData			POINTER TO BYTE	0	FALSE	Not Public	
0007	Var_Done			BOOL	FALSE	FALSE	Not Public	
0008	BlockCnt			WORD	0	FALSE	Not Public	
0009	Var_Error			BOOL	FALSE	FALSE	Not Public	
0010	Var_ErrorID			BYTE	0	FALSE	Not Public	

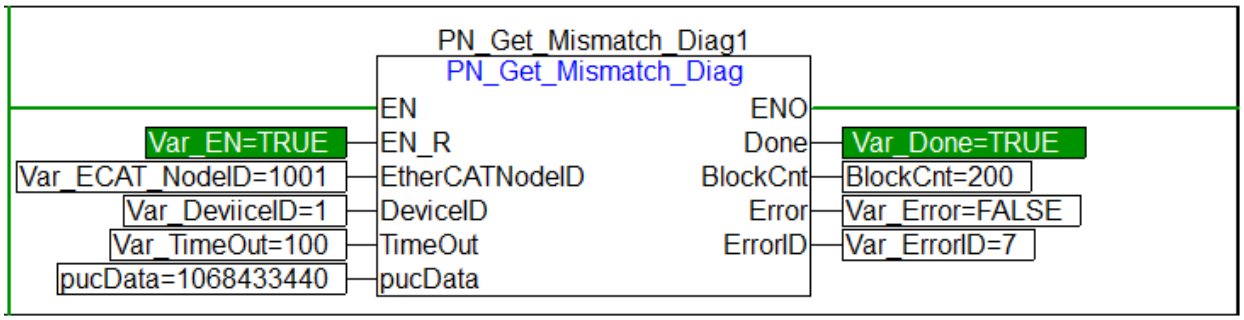
2. LD Configuration Example



3. ST Configuration Example

```
PN_Get_Mismatch_Diag1(
    EN_R := Var_EN,
    EtherCATNodeID := Var_ECAT_NodeID,
    DeviceID := Var_DeviceID,
    TimeOut := Var_TimeOut,
    pucData := pucData,
    Done=>Var_Done,
    BlockCnt=>BlockCnt,
    Error=>Var_Error,
    ErrorID=>Var_ErrorID);
```

4. Use the LD program to read the PROFINET slave slot mismatch diagnostic errors. As shown in the figure below, when reading the slot diagnostic information of the PROFINET slave, BlockCnt exceeds 128, and the ErrorID is 7.



Chapter 9 Appendix

9.1 EtherCAT error codes

Controller Read/Write Parameter Error Codes

Error code (HEX)	Error code (DEC)	Error message description
0xC0CD0020	3234660384	When the slave loses connection and EtherCAT-Read/EtherCAT-Write reads or writes, this error code is returned when the slave is offline
0xC065000F	3227844623	Invalid slave number
0xC0650012	3227844626	Invalid object index
0xC0650013	3227844627	Bus not in communication state
0xC0650023	3227844643	Slave does not support mailbox communication
0xC0650028	3227844648	SDO protocol timeout
0xC065002E	3227844654	Access to object not supported
0xC0650032	3227844658	Object does not exist in PDO
0xC0650038	3227844664	Parameter data type mismatch, length too large
0xC0650039	3227844665	Parameter data type mismatch, length too short
0xC065003B	3227844667	Parameter value out of range
0xC065003C	3227844668	Parameter value exceeds upper limit
0xC065003D	3227844669	Parameter value below lwer limit
0xC065003E	3227844670	Maximum value lower than parameter lower limit
0xC0650044	3227844676	Unknown error
0xFFFFFFFF	4294967295	Internal error

EtherCAT diagnostic error codes

Error code (HEX)	Error code (DEC)	Error message description
0xC0650005	3227844613	The software configuration topology is inconsistent with the physical connection of the slave station
0xC0CD	323466040	Check if the specified slave is consistent with the project configuration

0034	4	
0xC0CD0043	3234660419	Check if the network cable of the substation is connected correctly, such as connecting the network cable that should be connected to IN to OUT
0xC02B4242	3224060482	The ENI configuration file of the master station exceeds the maximum capacity
0xC0640001	3227779073	Slave initialization command is unresponsive
0xCFFF0000	3489595392	The hot link identifier of the slave station is set incorrectly, with the lower 16 bits being the hot link identifier in the slave station EEPROM
0xCFFFFFFA	3489660922	The protocol is in auto-start mode, but the scan failed
0xCFFFFFFB	3489660923	No link on the network interface.
0xCFFFFFFC	3489660924	The hot link identifier for the slave station in the ENI file is 0
0xCFFFFFFD	3489660925	Duplicate hot link identifiers in ENI files
0xCFFFFFFE	3489660926	The number of hot link groups from the ENI file exceeds 64
0xCFFFFFFF	3489660927	The watchdog is triggered from the slave station, and the controller diagnosis is 13, indicating insufficient cycle during the download process

EtherCAT slave ALStatusCode error codes

Error code (HEX)	Error code (DEC)	Error message description
0xC0D50001	3235184641	unknown error
0xC0D50002	3235184642	insufficient memory
0xC0D50003	3235184643	Slave station startup error
0xC0D50011	3235184657	Invalid slave state transition request
0xC0D50012	3235184658	Unknown state transition request
0xC0D50013	3235184659	Bootstrap mode is not supported
0xC0D50014	3235184660	Wireless firmware
0xC0D50015	3235184661	Invalid email configuration in BOOT mode
0xC0D50016	3235184662	Invalid email configuration in PREOP mode
0xC0D50017	3235184663	Invalid synchronization management configuration
0xC0D50018	3235184664	Invalid input
0xC0D50019	3235184665	Invalid output
0xC0D5001A	3235184666	Synchronization error
0xC0D5001B	3235184667	Synchronization Manager Watchdog Trigger
0xC0D5001C	3235184668	Invalid synchronization manager type
0xC0D5001D	3235184669	Invalid output configuration
0xC0D5001E	3235184670	Invalid input configuration
0xC0D5001F	3235184671	Invalid watchdog configuration
0xC0D50020	3235184672	The slave station requires a cold start
0xC0D50021	3235184673	The slave station requires INIT
0xC0D50022	3235184674	The slave station requires PREOP
0xC0D50023	3235184675	The slave station requires SAFEOP
0xC0D50024	3235184676	Invalid input mapping
0xC0D50025	3235184677	Invalid output mapping
0xC0D50026	3235184678	Inconsistent configuration
0xC0D50027	3235184679	Not supporting FreeRun
0xC0D50028	3235184680	SyncMode is not supported
0xC0D50029	3235184681	FreeRun mode requires 3 Buffer modes
0xC0D5002B	3235184683	Invalid output and input

0xC0D5002D	3235184685	SYNC loss error
0xC0D50030	3235184688	Invalid DC synchronization configuration
0xC0D50031	3235184689	Invalid DC Latch configuration
0xC0D50032	3235184690	PLL error
0xC0D50033	3235184691	DC synchronous I/O error
0xC0D50034	3235184692	DC synchronization timeout error
0xC0D50035	3235184693	Invalid DC synchronization cycle time
0xC0D50036	3235184694	DC SYNC 0 cycle time does not meet application layer requirements
0xC0D50037	3235184695	DC SYNC 1 cycle time does not meet application layer requirements
0xC0D50041	3235184705	AOE protocol specific error
0xC0D50042	3235184706	EOE protocol specific error
0xC0D50043	3235184707	COE protocol specific error
0xC0D50044	3235184708	FOE protocol specific error
0xC0D50045	3235184709	SOE protocol specific error
0xC0D5004F	3235184719	VOE protocol specific error
0xC0D50050	3235184720	PDI does not have EEPROM access permission
0xC0D50051	3235184721	EEPROM error
0xC0D50060	3235184736	Restart occurred at the slave station
0xC0D50061	3235184737	Device identifier value update
0xC0D50070	3235184752	The detected module (0xF030) list is inconsistent with the configured module (0xF050)
0xC0D50080	3235184768	The voltage or temperature of the equipment is too low
0xC0D50081	3235184769	The voltage or temperature of the equipment is too high

9.2 CANopen error codes

Master Station Emergency Error Codes

Error Register	Emergency Error Code	Master Custom Error Code
0x0	0x0: Error reset	0
0x11	0x8130: Slave offline	Slave's nodeID

Master Custom Error Codes

Error Code	Description
0x00	No error.
0x01	NodeID error.
0x02	Index error.
0x04	Write object dictionary parameter length error.
0x05	NMT command word error.
0x06	EtherCAT bus read/write error.
0x1105	Object dictionary read/write operation - slave no response.
0x1106	CAN bus error passive state.
0x1107	Master information storage memory error.
0x1108	Master state Eerror.
0x1109	Master configuration data error.
0x110A	Master initialization error.
0x110B	Master object dictionary read/write error.

0x110C	Slave node does not exist.
0x110D	Slave state error.
0x110E	Slave configuration return error.
0x110F	Slave PDO parameter error.
0x1110	Slave SDO read/write return error.
0x1111	Slave SDO read/write parameter error.
0x1112	CAN bus disconnected.
0x1113	CAN bus reconnection restored.
0x1114	CAN bus data transmission failed.
0x1115	CAN bus receive overflow.
0x1116	Slave device type mismatch error.
0x1117	Slave vendor code mismatch error.
0x1118	Slave product code mismatch error.
0x1119	Slave product version mismatch error.
0x111A	Slave heartbeat exception.
0x111B	The total number of RPDOs exceeds 256.
0x111C	The total number of TPDOs exceeds 256.
0x111D	The total number of SDOs exceeds 8192.
0x111E	The number of slave PDOs exceeds 64.
0x111F	The number of slave SDOs exceeds 1000.

SDO Error Codes

Error Code	Description
0x05030000	Toggle bit not alternated.
0x05040000	SDO protocol timed out.
0x05040001	Client/server command specifier not valid or unknown.
0x05040002	Invalid block size (block mode only).
0x05040003	Invalid sequence number (block mode only).
0x05040004	CRC error (block mode only).
0x05040005	Out of memory.
0x06010000	Unsupported access to an object.
0x06010001	Attempt to read a write only object.
0x06010002	Attempt to write a read only object.
0x06020000	Object does not exist in the object dictionary.
0x06040041	Object cannot be mapped to the PDO.
0x06040042	The number and length of the objects to be mapped would exceed PDO length.
0x06040043	General parameter incompatibility reason.
0x06040047	General internal incompatibility in the device.
0x06060000	Access failed due to an hardware error.
0x06070010	Data type does not match, length of service parameter does not match.
0x06070012	Data type does not match, length of service parameter too high.
0x06070013	Data type does not match, length of service parameter too low.
0x06090011	Sub-index does not exist.
0x06090030	Value range of parameter exceeded (only for write access).
0x06090031	Value of parameter written too high.
0x06090032	Value of parameter written too low.
0x06090036	Maximum value is less than minimum value.
0x08000000	General error.
0x08000020	Data cannot be transferred or stored to the application.
0x08000021	Data cannot be transferred or stored to the application because of local control.
0x08000022	Data cannot be transferred or stored to the application because of the present device state.
0x08000023	Object dictionary dynamic generation fails or no object dictionary is present (e.g. object dictionary is

	generated from file and generation fails because of an file error).

9.3 Serial Port Capture Error Codes

ErrorID	Description
0	No error
1	Invalid EtherCAT slave address input (1~65535)
2	Input EtherCAT address does not exist
3	Input EtherCAT address is not a serial port module
4	Invalid communication serial port number input (1, 2)
5	Input In Offset out of bounds
6	Input Out Offset out of bounds
7	When serial port number is 1, input offset does not match the module's actual offset
8	When serial port number is 1, output offset does not match the module's actual offset
9	Input cyclic data length is not 20/40/80
10	Invalid Modbus slave address input (1~247)
11	Invalid timeout input (must be an integer greater than 0)
20	Invalid function code input
21	Invalid number of slave registers input
40	Input pSendBuffer pointer error
41	Input send data length exceeds the 1000-byte limit
60	Input pRecvBuffer pointer error
61	Input prefix length exceeds receive length
62	Input prefix pointer is null
63	Input suffix pointer is null
64	Input receive length exceeds the 1000-byte limit
100	Communication timeout
120	Master send process error
121	Master receive process error
122	Received message CRC check error
123	Data length or register address in slave response does not match the request
124	Slave returned illegal instruction
125	Slave returned illegal data address
126	Slave returned illegal data value
127	Slave returned slave device failure
128	Slave returned other error diagnostic
140	Error during EtherCAT cyclic message reception
141	Received message CRC check error
142	Error sending data from controller (as slave) to master
160	Free port send process error
180	Error during data reply process after free port reception
181	Free port receive buffer full or prefix pointer mismatch
182	Received data length is zero, receive timeout
183	Receive mode not supported
184	Free port cyclic data reception error
200	Not a HollySys LX series serial port module

201	Not a serial port module
202	Capture port not enabled
203	EtherCAT_Read failed. Check module status or confirm if input address is a configured slave
204	Capture via AT already enabled, function block capture is disabled
205	Capture function block stopped
206	Capture via AT has stopped the function block, waiting for AT to start capture
207	Capturing with new parameters
208	Capture function block called repeatedly
209	Capture function serial port parameter is empty
255	Maximum error code value

Beijing HollySys Intelligent Technologies Co., Ltd..

Di Sheng Middle Road, No.2

Economic-Technological Development Area

100176 Beijing, China

Tel: 010-5898 1588

Hotline: 400-811-1999

Fax: 010-5898 1558

Web: http://www.hollysys.com

FA Web: https://fa.hollysys.com/