



Add-On Instructions for PSx-3__ with EtherNet/IP

halstrup-walcher GmbH

Stegener Straße 10
D-79199 Kirchzarten

Phone: +49 (0) 76 61/39 63-0
Fax: +49 (0) 76 61/39 63-99

E-Mail: info@halstrup-walcher.de
Internet: www.halstrup-walcher.de

Table of Contents

1	Safety precautions	4
1.1	Appropriate use	4
1.2	Symbols	4
2	Data Structure DRIVE_DATA	5
3	Error Description (Error ID)	8
4	Description of the Add-On Instructions.....	10
4.1	Commonalities of all Add-On Instructions	10
4.2	MC_Move	12
4.3	MC_Error	15
4.4	MC_ReadParameter	16
4.5	MC_WriteParameter	18
4.6	MC_Parametrization	20
4.7	MC_PositionParametrization.....	23
5	How to Add Drives to a project	27
5.1	Registering the appropriate EDS file with the help of RSNetWorx.....	27
5.2	Registering the appropriate EDS file with the help of Logix Designer	29
5.3	Adding a drive.....	33
6	How to Add AOIs to a project	37
6.1	Import an AOI	37
6.2	Representation of drives and AOIs in the Controller Organizer	40
7	How to Add an AOI and Controller Tags to a Ladder.....	41

Purpose of instruction manual

This instruction manual describes the Add-On Instructions for the PSx-3__-EI (with EtherNet/IP interface).

Improper use of these devices or failure to follow these instructions may cause injury or equipment damage. Every person who uses the devices must therefore read the manual and understand the possible risks. The instruction manual, and in particular the safety precautions contained therein, must be followed carefully. **Contact the manufacturer if you do not understand any part of this instruction manual.**

The manufacturer reserves the right to continue developing these Add-On Instructions without documenting such development in each individual case. The manufacturer will be happy to determine whether this manual is up-to-date.

© 2015

The manufacturer owns the copyright to this instruction manual. It must not be copied either wholly or in part or made available to third parties.

1 Safety precautions

1.1 Appropriate use

The positioning systems PSx-3__-EI are especially suitable for automatically setting tools, stops or spindles for wood-processing equipment, packing lines, printing equipment, filling units and other types of special machines.

PSx3__-EI positioning systems are not stand-alone devices and may only be used if coupled to another machine.

1.2 Symbols

The symbols given below are used throughout this manual to indicate instances when improper operation could result in the following hazards:



WARNING!

This warns you of a potential hazard that could lead to bodily injury up to and including death if the corresponding instructions are not followed.



CAUTION!

This warns you of a potential hazard that could lead to significant property damage if corresponding instructions are not followed.



INFORMATION!

This indicates that the corresponding information is important for operating the Add-On Instructions properly.

2 Data Structure DRIVE_DATA

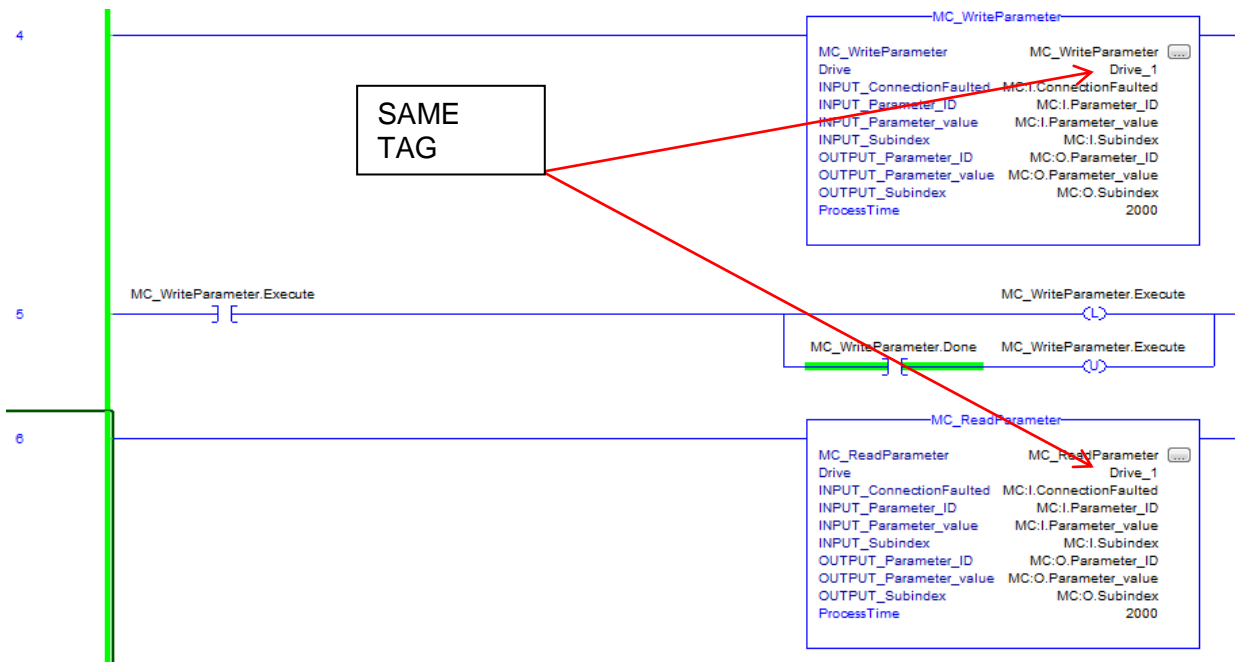
For each drive there's a data structure, in which data is placed that details the drive's state, the name of the axis (optional), and a description of the drive (optional). For each drive an instance of this structure is required. This instance must be provided to each AOI that operates on the corresponding drive. For every one drive there are six AOIs that can be used. The drive's state prevents two or more AOIs from accesses the parameter interface of a single drive at the same time.

Parameter name	Data type	Written by	Description
Name	STRING[16]	User (optional)	Name of axis
Description	STRING[32]	User (optional)	Description (e.g. function, task of this axis)
State	DINT	AOIs	Actual state

In this library, AOIs are available in order to perform the following tasks on each drive:

- A run command can be sent with an AOI "MC_Move".
- Error messages can be retrieved with an AOI "MC_Error".
- Single parameters can be read with an AOI "MC_ReadParameter".
- Single parameters can be written with an AOI "MC_WriteParameter".
- Multiple parameters can be written with an AOI "MC_Parametrization".
- Position data can be written with an AOI "MC_PosParametrization".

In the example below there are two different instructions for Drive 1. The Drive Parameter on the AOI instructions needs to be the same for each instruction writing to Drive 1.



Depending on the AOI instructions that are currently running on a certain drive, the state of this drive will be as follows:

- MC_Move → bit 0 of "state" (running: bit 0 set; not running: bit 0 reset)

- MC_Error → no influence on “state”
- MC_ReadParameter → state = 30
- MC_WriteParameter → state = 40
- MC_Parametrization → state = 50
- MC_PosParametrization → state = 60

When a single AOI from above is enabled it will populate the Drive_1 tag with the state. The variable state at each time shows which AOIs are running currently.

Examples:

- state = 0 → All AOIs are idle.
- state = 30 → Only AOI “MC_ReadParameter” is running.
- state = 41 → AOIs “MC_WriteParameter” and “MC_Move” are running.

The following rules apply for the simultaneous running of more than one AOI on the same axis:

- “MC_Error” command is possible together with all other AOIs.
- “MC_Move” command shall only be started if “MC_Parametrization” and “MC_PositionParametrization” are idle. (“MC_ReadParameter” or “MC_WriteParameter” may run at the same time.)
- “MC_ReadParameter” shall only be started if “MC_WriteParameter”, “MC_Parametrization” and “MC_PositionParametrization” are idle. (“MC_Move” doesn’t matter.)
- “MC_WriteParameter” shall only be started if “MC_ReadParameter”, “MC_Parametrization” and “MC_PositionParametrization” are idle. (“MC_Move” doesn’t matter.)
- “MC_Parametrization” shall only be started if “MC_Move”, “MC_ReadParameter”, “MC_WriteParameter” and “MC_PositionParametrization” are idle.
- “MC_PositionParametrization” shall only be started if “MC_Move”, “MC_ReadParameter”, “MC_WriteParameter” and “MC_Parametrization” are idle.

Example:

In the project there are three drives. Each drive shall be moved with an AOI MC_Move, additionally it should be possible to determine the state of each drive with MC_Error and for each drive it should be possible to perform any user-defined writing and reading accesses.

To achieve this, altogether three variables of the type DRIVE_DATA are necessary, these have to be generated in the Controller Tags section:

- Drive_1
- Drive_2
- Drive_3

For the execution of the required functions for each drive also an instance of the AOI MC_Move, MC_Error, MC_ReadParameter and MC_WriteParameter is required:

- MC_Move_1, MC_Error_1, MC_Read_1, MC_Write_1:
VAR_IN_OUT "Drive" → enter "Drive_1"
- MC_Move_2, MC_Error_2, MC_Read_2, MC_Write_2:
VAR_IN_OUT "Drive" → enter "Drive_2"
- MC_Move_3, MC_Error_3, MC_Read_3, MC_Write_3:
VAR_IN_OUT "Drive" → enter "Drive_3"

3 Error Description (Error ID)

Below is a listing of the Error Codes, which can be accessed from the AOIs controller tags:

ErrorID (hex)	Description
16xF000 (mask)	AOI
16#1xxx	Error in MC_Move
16#2xxx	Error in MC_Error
16#3xxx	Error in MC_ReadParameter
16#4xxx	Error in MC_WriteParameter
16#5xxx	Error in MC_Parametrization
16#6xxx	Error in MC_PositionParametrization
16#0F00 (mask)	Internal AOI and PD errors
16#x1xx	Error in state machine or other AOI internal error
16#x6xx	Unallowed input data change
16#x7xx	Connection Faulted
16#00F0 (mask)	Parameter errors
16#xx1x	Parameter: communication timeout (1000 ms)
16#xx2x	Parameter: invalid parameter number
16#xx3x	Parameter: value is read only
16#xx4x	Parameter: lower or upper limit exceeded
16#xx5x	Parameter: faulty sub-index
16#xx6x	Parameter: not an array
16#xx7x	Parameter: incorrect data type
16#xx8x	Parameter: setting not allowed (resetting only)
16#xx9x	Parameter: request cannot be processed due to operating state
16#xxAx	Other error
16#000F (mask)	Drive errors
16#xxx1	Drag error
16#xxx2	Under- or overvoltage motor supply
16#xxx3	Positioning run aborted
16#xxx4	Temperature exceeded
16#xxx5	Absolute measuring system error
16#xxx6	Block or overcurrent error
16#xxx7	Manual displacement
16#xxx8	Incorrect target value
16#xxx9	Under- or overvoltage during run
16#xxxA	Lower position limit exceeded
16#xxxB	Upper position limit exceeded

In the table above "Drive Errors" are a copy of the error bits in the status word of the PSx.

Examples:

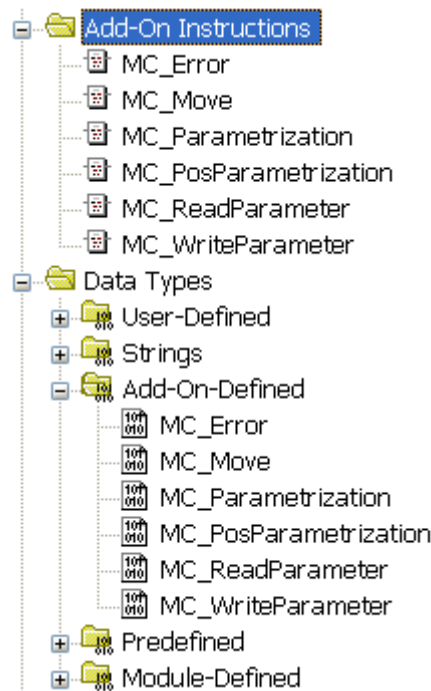
- Run command (MC_Move) with incorrect target value → ErrorID = 16#1008

- Writing a parameter (MC_WriteParameter) with invalid parameter number → ErrorID = 16#4020

4 Description of the Add-On Instructions

Initially the Add-On instructions have to be included in an own RSLogix or Studio 5000 project. This happens in the Controller Organizer with right click on “Add-On Instructions”, then “Import Add-On Instruction” and importing the desired Add-On instructions individually.

As result the Add-On instructions present itself in the following way in the Controller Organizer:



4.1 Commonalities of all Add-On Instructions

The AOIs respectively use a part of the following variables, all of them of the type VAR_IN_OUT. These inputs and outputs must be connected in either case, otherwise there will be error messages when downloading the project.

Drive

Reference to the desired drive (see also chapter 2)

- Type: DRIVE_DATA
- Nature: VAR_IN_OUT

ErrorDescription

Error description

- Type: STRING
- Default value: ""
- connect with any string variable in the Controller Tags section

INPUT_ConnectionFaulted

Signals if the input data of the corresp. drive are valid

- Type: BOOL

- connect with I.ConnectionFaulted of the corresp. drive

INPUT Status Word

Status word of the corresp. drive

- Type: INT
- connect with I.status_word of the corresp. drive

INPUT Actual Position

Actual position of the corresp. drive

- Type: DINT
- connect with I.actual_position of the corresp. drive

INPUT Parameter ID

Parameter number of the parameter interface of the corresp. drive

- Type: INT
- connect with I.Parameter_ID of the corresp. drive

INPUT Parameter value

Parameter value of the parameter interface of the corresp. drive

- Type: DINT
- connect with I.Parameter_value of the corresp. drive

INPUT Subindex

Array subindex of the parameter interface of the corresp. drive

- Type: INT
- connect with I.Subindex of the corresp. drive

OUTPUT Control Word

Control word of the corresp. drive

- Type: INT
- connect with O.control_word of the corresp. drive

OUTPUT Target Position

Target position of the corresp. drive

- Type: DINT
- connect with O.target_position of the corresp. drive

OUTPUT Parameter ID

Parameter number of the parameter interface of the corresp. drive

- Type: INT
- connect with O.Parameter_ID of the corresp. drive

OUTPUT Parameter value

Parameter value of the parameter interface of the corresp. drive

- Type: DINT
- connect with O.Parameter_value of the corresp. drive

OUTPUT Subindex

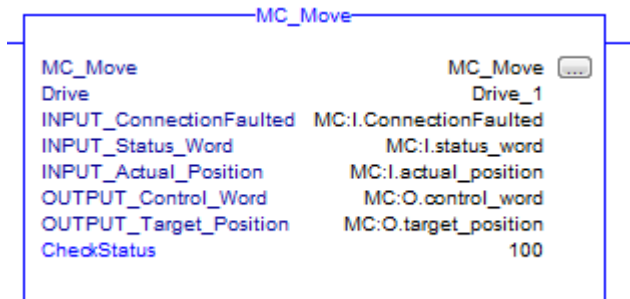
Array subindex of the parameter interface of the corresp. drive

- Type: INT
- connect with O.Subindex of the corresp. drive

In the following descriptions of the particular AOIs these variables of the type VAR_IN_OUT are not listed separately.

4.2 MC_Move

This AOI serves to send run commands to the drive.



Release

Release of the drive

- Type: BOOL
- Initial value: FALSE
- Nature: VAR_INPUT

Description:

- Run commands will only be executed if this bit is set.
- This input directly controls the release bit (bit 4) in the control word. If this input stays activated and e.g. the readjustment in the drive is activated, the drive readjusts automatically.
- If the input is activated and the target position is changed, the drive immediately moves to that position. An edge is not necessary.
- If the input is de-asserted during the run, the drive stops.

Position

Target position to be approached

- Type: DINT
- Initial value: 0
- Nature: VAR_INPUT

Description:

- If during a run a new target position is sent, this target position is approached immediately.
- If the release bit is still set after the end of a run and the target position is changed, the drive immediately approaches to that position.



INFORMATION!

In order to move to the same target position e.g. after a blocking condition, the release has to be de-asserted and asserted again.

ManualRunToLargerValues

Manual run to larger values

- Type: BOOL
- Initial value: FALSE

- Nature: VAR_INPUT

Description:

- Manual run to larger values, finishing at the positive range limit.
- Additionally the input "Release" has to be on resp. set.



CAUTION!

When de-asserting the input "ManualRunToLargerValues", additionally the release input has to be de-asserted. Otherwise the drive will move to the target position (AOI input "Position").

ManualRunToSmallerValues

Manual run to smaller values

- Type: BOOL
- Initial value: FALSE
- Nature: VAR_INPUT

Description:

- Manual run to larger values, finishing at the negative range limit.
- Additionally the input "Release" has to be on resp. set.



CAUTION!

When de-asserting the input "ManualRunToSmallerValues", additionally the release input has to be de-asserted. Otherwise the drive will move to the target position (AOI input "Position").

CheckStatus

Delay in checking status of the Input Status

- Type: DINT
- Initial value: 100
- Nature: VAR_INPUT

Description:

- Logical delay in checking the status word bits .6 and .0 (in [ms]). The delay is used in order to check if the drive is moving within a certain time frame after giving a release command. It shall prevent that the AOI sees status bit .6 ("drive is running") low and bit .0 ("target position reached") high before the drive has received a new run command.
- It is recommended to leave this at 100 for most applications. If the RPI is more than the "CheckStatus" value the user will run the risk of the timer expiring and resetting the instruction even before the move has started.

Active

Run command or run is active

- Type: BOOL
- Nature: VAR_OUTPUT

This output is asserted, if:

- the release bit is set from 0 to 1 (rising edge)
- the release is already present and the target position is changing
- the bit "drive is running" in the status word of the drive is set (e.g. when the drive is readjusting its position)

This output is de-asserted, if:

- at the end of a run the bit "drive is running" in the status word of the drive is no longer set

- a communication error occurs

InPosition

Target position reached

- Type: BOOL
- Nature: VAR_OUTPUT

This output is a copy of the status bit “target position reached”. If a communication error occurs, it will be de-asserted.

Actual position

Actual value of the position

- Type: DINT
- Nature: VAR_OUTPUT

This value is a copy of the actual position. If a communication error occurs, the value will be set to 0.

Error

Error while executing the AOI or error in drive

- Type: BOOL
- Nature: VAR_OUTPUT

The error bit also might be set during a move of the drive (e.g. drag error).

ErrorID

Error code

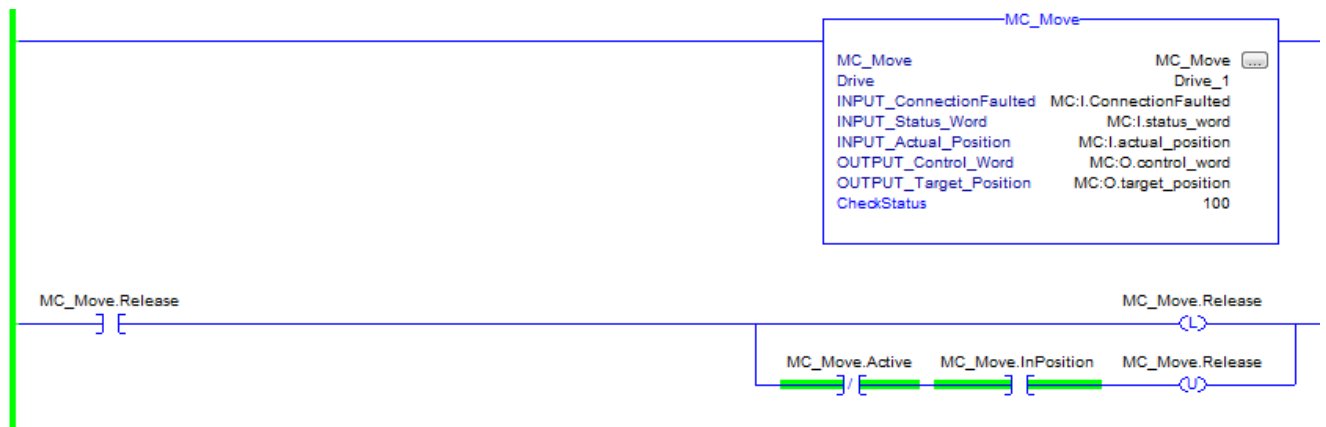
- Type: INT
- Nature: VAR_OUTPUT

The error bit also might be set during a move of the drive (e.g. drag error). In case of no error, the value is set to 0.

If the drive reports multiple errors, the ErrorID with the highest priority is shown. This priority corresponds to the order in the following table (highest priority has 16#x1xx):

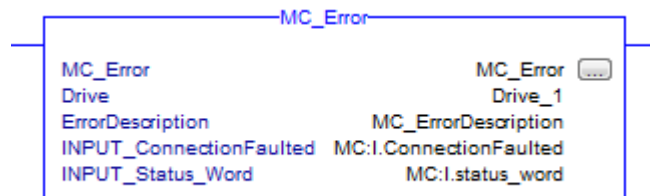
ErrorID	Description
16#x1xx	AOI internal error
16#x2xx	Invalid PD input address
16#x3xx	Invalid PD output address
16#x4xx	Error while reading PD
16#x5xx	Error while writing PD
16#xxx2	Under- or overvoltage motor supply
16#xxx4	Temperature exceeded
16#xxx5	Absolute measuring system error
16#xxx8	Incorrect target value
16#xxx9	Under- or overvoltage during run
16#xxx6	Block or overcurrent error
16#xxx7	Manual displacement
16#xxxA	Lower position limit exceeded
16#xxxB	Upper position limit exceeded
16#xxx3	Positioning run aborted
16#xxx1	Drag error

Simple ladder logic example with the AOI “MC_Move”:



4.3 MC_Error

This AOI reports the state of the drive and the AOI as error bit, error code (“ErrorID”) and as text.



Enable

The outputs Error, ErrorID and ErrorDescription permanently are updated by the drive, as long as Enable is set. If Enable is de-asserted, these outputs switch to their default values.

- Type: BOOL
- Initial value: FALSE
- Nature: VAR_INPUT

Error

Error while executing the AOI or error in drive

- Type: BOOL
- Default value: FALSE
- Nature: VAR_OUTPUT

ErrorID

Error code (see following table “ErrorID”)

- Type: INT
- Default value: 0
- Nature: VAR_OUTPUT

ErrorDescription

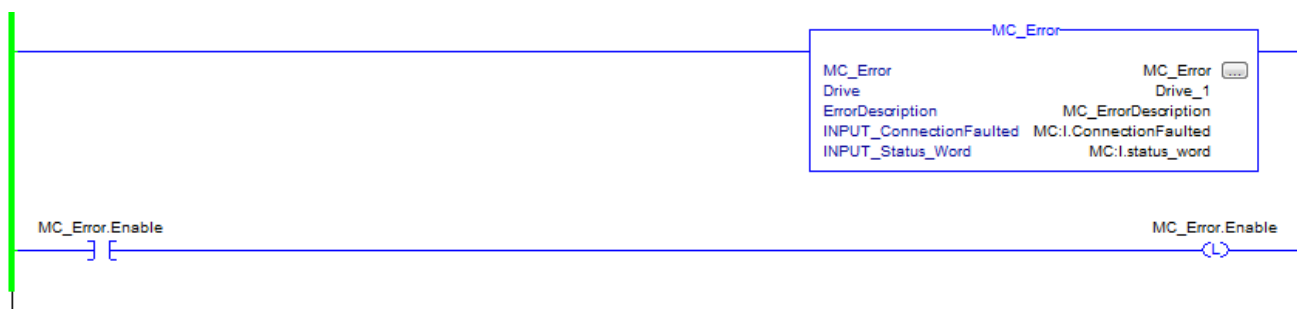
Error Description as text

- Type: STRING
- Default value: “”
- Nature: VAR_IN_OUT

The priority corresponds to the order in the following table (highest priority has 16#x1xx).

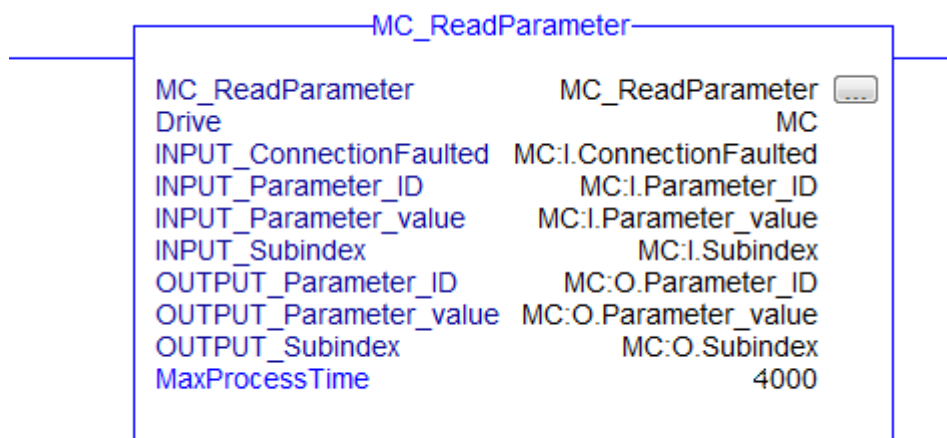
ErrorID	ErrorDescription
16#x1xx	AOI internal error
16#x2xx	Invalid PD input address
16#x4xx	Error while reading PD
16#xxx2	Under- or overvoltage motor supply
16#xxx4	Temperature exceeded
16#xxx5	Absolute measuring system error
16#xxx8	Incorrect target value
16#xxx9	Under- or overvoltage during run
16#xxx6	Block or overcurrent error
16#xxx7	Manual displacement
16#xxxA	Lower position limit exceeded
16#xxxB	Upper position limit exceeded
16#xxx3	Positioning run aborted
16#xxx1	Drag error

Simple ladder logic example with the AOI "MC_Error":



4.4 MC_ReadParameter

With this AOI values of parameters can be read by the drive. Each parameter can be read except par. 23 ("device model as string").



Execute

Start of a reading process

- Type: BOOL
- Initial value: FALSE
- Nature: VAR_INPUT

Description:

When issuing a rising edge, a reading process of the parameter which is specified by "ParameterNumber" and "Subindex" is started. For a new reading process, a new rising edge has to be generated. When de-asserting the bit, the outputs fall back to their specified default value.

ParameterNumber

Parameter number of the parameter to be read

- Type: INT
- Initial value: 0
- Nature: VAR_INPUT

SubIndex

Array subindex of the parameter

- Type: INT
- Initial value: 0
- Nature: VAR_INPUT

MaxProcessTime

Maximum time for the AOI to finish the communication with the drive (in [ms]).

- Type: DINT
- Initial value: 4000
- Nature: VAR_INPUT

Description:

The AOI must complete within this time frame or the AOI will produce an error. It is recommended to leave this at 4000 for most applications.

Active

Bit is set as long as the reading process is active

- Type: BOOL
- Default value: FALSE
- Nature: VAR_OUTPUT

The bit is being de-asserted as soon as the value has been read or an error occurred (check "Done" and "Error").

Done

Bit is set as soon as the parameter has been read successfully and is available in "Value"

- Type: BOOL
- Default value: FALSE
- Nature: VAR_OUTPUT

The bit is being de-asserted when starting a reading process.

Error

Bit is set if an error occurred during the execution of the AOI

- Type: BOOL
- Default value: FALSE
- Nature: VAR_OUTPUT

ErrorID

Error code (see table "ErrorID" in chapter 3)

- Type: INT
- Default value: 0
- Nature: VAR_OUTPUT

Drive errors are not considered when reading a parameter.

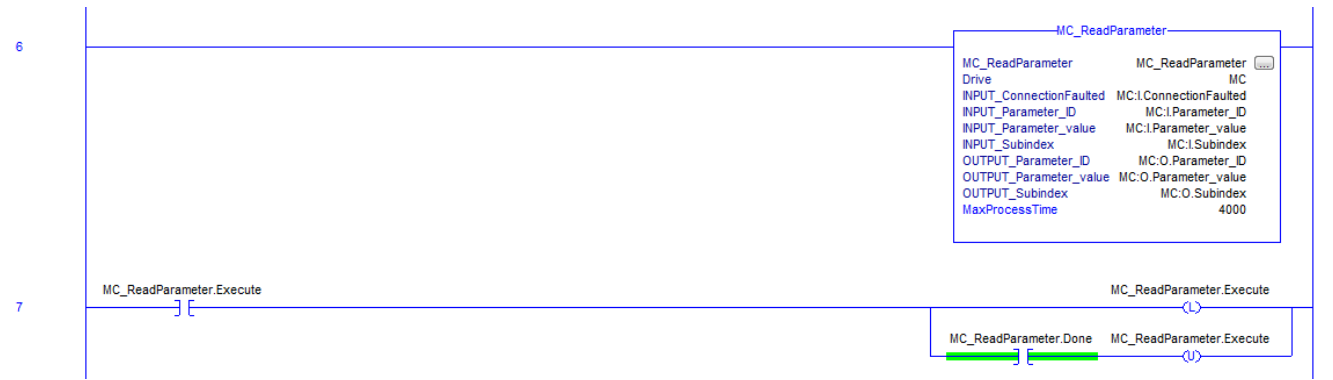
Value

Actual value of the parameter which has been read

- Type: DINT
- Default value: 0
- Nature: VAR_OUTPUT

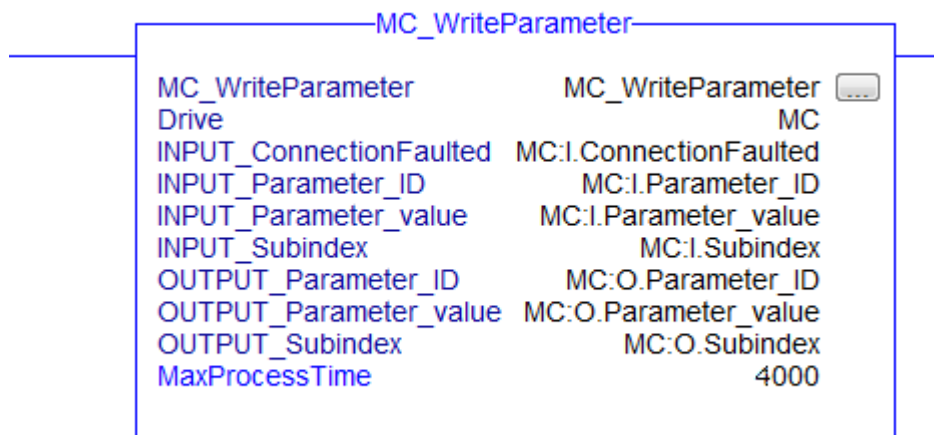
When an error occurred, the value will be 0.

Simple ladder logic example with the AOI "MC_ReadParameter":



4.5 MC_WriteParameter

With this AOI values of parameters can be written into the drive.



Execute

Start of a writing process

- Type: BOOL

- Initial value: FALSE
- Nature: VAR_INPUT

Description:

When issuing a rising edge, a writing process of the parameter which is specified by "ParameterNumber" and "Subindex" with the value which is specified by the input "Value" is started. For a new writing process, a new rising edge has to be generated. When de-asserting the bit, the outputs fall back to their specified default value.

ParameterNumber

Parameter number of the parameter to be written

- Type: INT
- Initial value: 0
- Nature: VAR_INPUT

Value

Value to be written to the parameter

- Type: DINT
- Initial value: 0
- Nature: VAR_INPUT

MaxProcessTime

Maximum time for the AOI to finish the communication with the drive (in [ms]).

- Type: DINT
- Initial value: 4000
- Nature: VAR_INPUT

Description:

The AOI must complete within this time frame or the AOI will produce an error. It is recommended to leave this at 4000 for most applications.

Active

Bit is set as long as the writing process is active

- Type: BOOL
- Default value: FALSE
- Nature: VAR_OUTPUT

The bit is being de-asserted as soon as the value has been written or an error occurred (check "Done" and "Error").

Done

Bit is set as soon as the parameter has been written successfully

- Type: BOOL
- Default value: FALSE
- Nature: VAR_OUTPUT

The bit is being de-asserted when starting a writing process.

Error

Bit is set if an error occurred during the execution of the AOI

- Type: BOOL
- Default value: FALSE
- Nature: VAR_OUTPUT

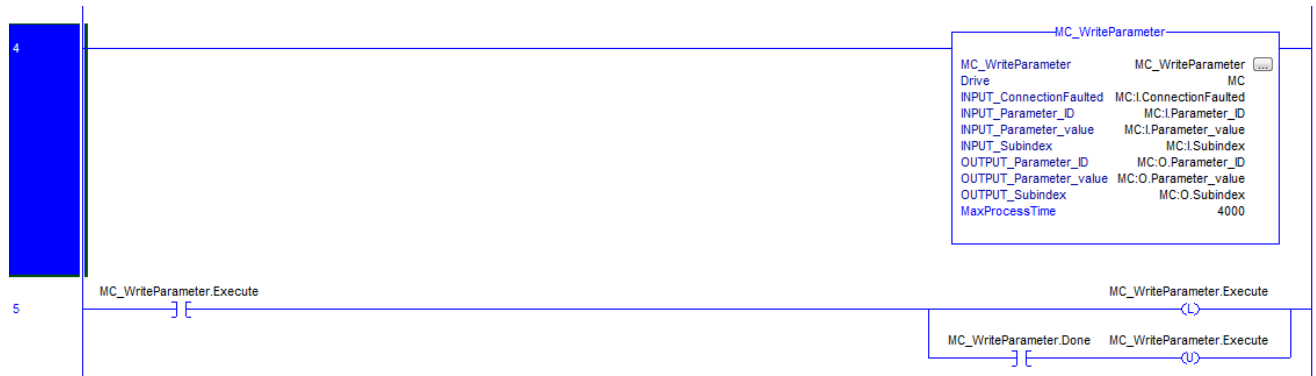
ErrorID

Error code (see table "ErrorID" in chapter 3)

- Type: INT
- Default value: 0
- Nature: VAR_OUTPUT

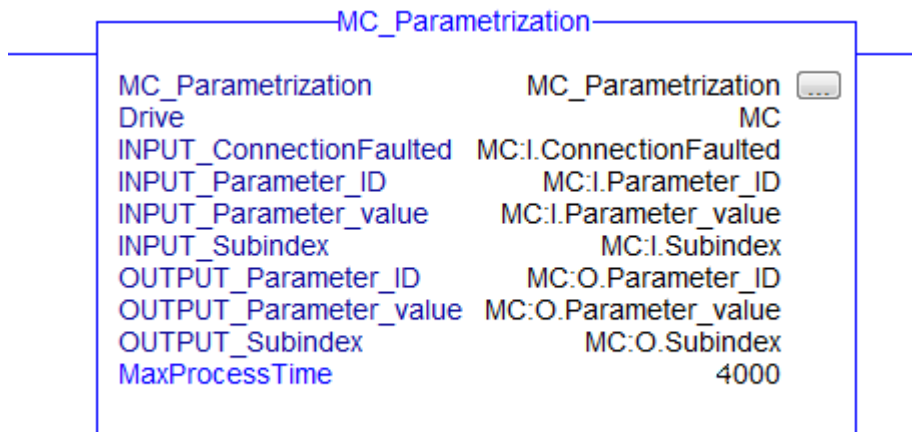
Drive errors are not considered when writing a parameter.

Simple ladder logic example with the AOI “MC_WriteParameter”:



4.6 MC_Parametrization

With this AOI the following parameters of the drive can be written with one single AOI call: 10, 26, 28, 30, 32, 40, 42, 44, 46, 48, 52, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 84, 86, 108, 110 and 113



The following items have to be considered when using this AOI:

- For each parameter value there's additionally an enable tag in order to determine whether the parameter shall be written or not.
Example: DirRotation_Enable = 1 → DirRotation_Value is written
- The parameters are being written in the following order:
 - “DeliveryState” (Par. 113) →
 - “DirRotation” (Par. 26) →
 - “PosScaleNumerator” (Par. 28) →
 - ... →
 - “MaxTemperature” (Par. 110) →

- "SaveSettings" (Par. 113)

In between the parameters are being written in ascending order.

- Optionally a delivery state might be commanded before setting a certain number of parameters. To do this, the input "DeliveryState_113" has to be set to TRUE before the execution of the AOI. Thus the values of each parameter are set to the delivery state (initially without saving).
- If there are parameters that are set to "enable" (→ "x_Enable" is set) after the delivery state they will change from the delivery state to the values entered in the "x_Value".
- Optionally at the end additionally the written values might be saved permanently. To do this, the input "SaveSettings_113" has to be set to TRUE before the execution of the AOI.
- In case of an error while writing a parameter, the subsequent parameters are not written any more. Also no saving of the values is carried out, if the input "SaveSettings" is set.

Execute

Start of a parametrization process

- Type: BOOL
- Initial value: FALSE
- Nature: VAR_INPUT

Description:

When issuing a rising edge, a parametrization process with the given values is started. For a new parametrization process, a new rising edge has to be generated. When de-asserting the bit, the outputs fall back to their specified default value.

DeliveryState

Loading of the delivery state (initially without saving)

- Type: BOOL
- Initial value: FALSE
- Nature: VAR_INPUT

However, IP address and address assigning method stay unaffected.

x_Enable

If set, the corresp. parameter is written

- Type: BOOL
- Initial value: FALSE
- Nature: VAR_INPUT

x_Value

Desired value of the parameter

- Initial value: 0
- Nature: VAR_INPUT

The parameter number is given after the parameter name. The data type, a description as well as the value range can be extracted of the instruction manual of the PSx-3__-EI.

SaveSettings

Saving the settings permanently

- Type: BOOL
- Initial value: FALSE
- Nature: VAR_INPUT

MaxProcessTime

Maximum time for the AOI to finish the communication with the drive (in [ms]).

- Type: DINT
- Initial value: 4000
- Nature: VAR_INPUT

Description:

The AOI must complete within this time frame or the AOI will produce an error. It is recommended to leave this at 4000 for most applications.

Active

Bit is set as long as the parametrization process is active

- Type: BOOL
- Default value: FALSE
- Nature: VAR_OUTPUT

The bit is being de-asserted as soon as the parametrization has been finished successfully or an error occurred (check “Done” and “Error”).

Done

Bit is set as soon as the parametrization has been finished successfully

- Type: BOOL
- Default value: FALSE
- Nature: VAR_OUTPUT

The bit is being de-asserted when starting a parametrization process.

Error

Bit is set if an error occurred during the execution of the AOI

- Type: BOOL
- Default value: FALSE
- Nature: VAR_OUTPUT

ErrorID

Error code (see table “ErrorID” in chapter 3)

- Type: INT
- Default value: 0
- Nature: VAR_OUTPUT

Drive errors are not considered during a parametrization.

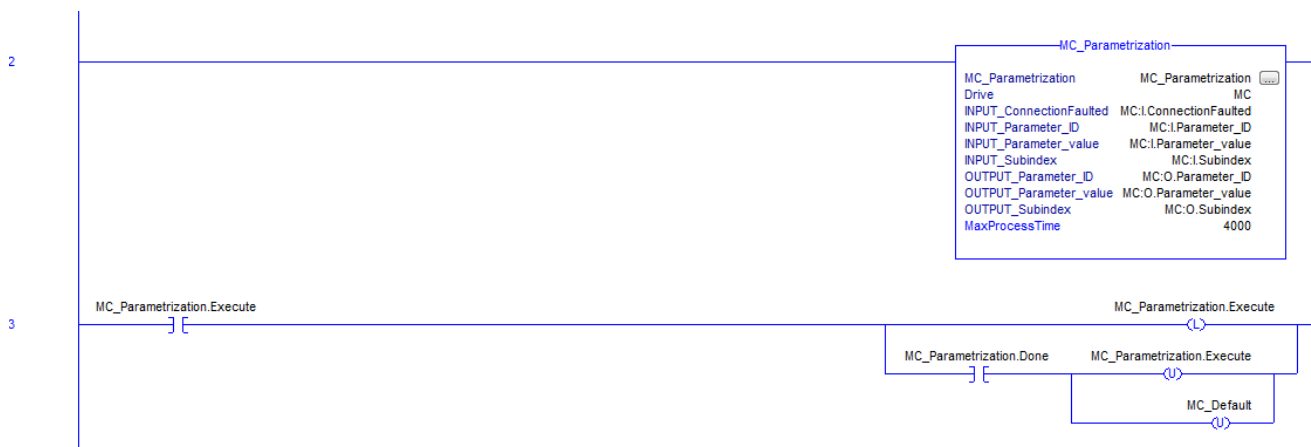
ErrorParameter

Parameter number that caused the error (in case of an error)

- Type: INT
- Default value: 0
- Nature: VAR_OUTPUT

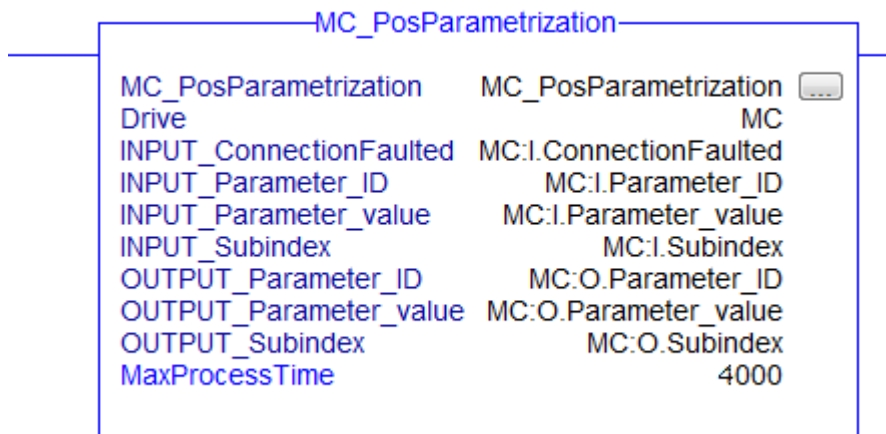
If no error occurred, this value is 0.

Simple ladder logic example with the AOI “MC_Parametrization”:



4.7 MC_PositionParametrization

With this AOI the parametrization of the position data can be carried out (parameters having an influence on the value of the displayed actual position).



The following items have to be considered when using this AOI:

- Each value has to be written and the values have to have a reasonable relation to each other. Each value is processed, after that the following parameters are written in the order stated below:
 - Direction of rotation (Par. 26) = Direction
 - Position scaling, numerator (Par. 28) = 400
 - Position scaling, denominator (Par. 30) = StepsPerTurn
 - Actual value (Par. 10) = SetPoint
 - If (SetPoint > UpperLimit):
 - Upper mapping end (Par. 34) = SetPoint + (3 x StepsPerTurn)
 - otherwise:
 - Upper mapping end (Par. 34) = UpperLimit + (3 x StepsPerTurn)
 - Upper limit (Par. 36) = UpperLimit
 - Lower limit (Par. 38) = LowerLimit

- The number of steps per revolution “StepsPerTurn” directly results in the value of the parameter “Position scaling, denominator” (Par. 30). Thereby it is assumed that the value of “Position scaling, numerator” (Par. 28) is in delivery state, thus 400.
- Before writing the parameters, the entered values are checked for validity.

Subsequently the conditions and error codes which are displayed if a condition is not satisfied.

Condition	ErrorID	ErrorParameter
StepsPerTurn ≥ 1	16#6140	39
StepsPerTurn ≤ 10000	16#6140	39
LowerLimit \leq UpperLimit	16#6140	42
$(\text{UpperLimit} - \text{LowerLimit}) / \text{StepsPerTurn} \leq 250$	16#6140	43
If SetPoint < LowerLimit: $(\text{UpperLimit} - \text{SetPoint}) / \text{StepsPerTurn} \leq 250$	16#6140	3
If SetPoint > UpperLimit: $(\text{SetPoint} - \text{LowerLimit}) / \text{StepsPerTurn} \leq 250$	16#6140	3

- Optionally at the end additionally the written values might be saved permanently. To do this, the input “SaveSettings” has to be set to TRUE before the execution of the AOI.
- In case of an error while writing a parameter, the subsequent parameters are not written any more. Also no saving of the values is carried out, if the input “SaveSettings” is set.

Execute

Start of a parametrization process

- Type: BOOL
- Initial value: FALSE
- Nature: VAR_INPUT

Description:

When issuing a rising edge, a parametrization process with the given values is started. For a new parametrization process, a new rising edge has to be generated. When de-asserting the bit, the outputs fall back to their specified default value.

Direction

Direction in which the drive shall turn with larger values (if looking at the output shaft):

0 \rightarrow CW, 1 \rightarrow CCW

- Type: INT
- Initial value: 0
- Nature: VAR_INPUT

StepsPerTurn

Number of steps per revolution at the output shaft (resolution)

- Type: INT
- Initial value: 0
- Nature: VAR_INPUT

LowerLimit

Lower limit

- Type: DINT
- Initial value: 0

- Nature: VAR_INPUT

UpperLimit

Upper limit

- Type: DINT
- Initial value: 0
- Nature: VAR_INPUT

SetPoint

Value on which the measuring system is referenced (new actual value at the actual position)

- Type: DINT
- Initial value: 0
- Nature: VAR_INPUT

SaveSettings

Saving the settings permanently

- Type: BOOL
- Initial value: FALSE
- Nature: VAR_INPUT

MaxProcessTime

Maximum time for the AOI to finish the communication with the drive (in [ms]).

- Type: DINT
- Initial value: 4000
- Nature: VAR_INPUT

Description:

The AOI must complete within this time frame or the AOI will produce an error. It is recommended to leave this at 4000 for most applications.

Active

Bit is set as long as the parametrization process is active

- Type: BOOL
- Default value: FALSE
- Nature: VAR_OUTPUT

The bit is being de-asserted as soon as the parametrization has been finished successfully or an error occurred (check "Done" and "Error").

Done

Bit is set as soon as the parametrization has been finished successfully

- Type: BOOL
- Default value: FALSE
- Nature: VAR_OUTPUT

The bit is being de-asserted when starting a parametrization process.

Error

Bit is set if an error occurred during the execution of the AOI

- Type: BOOL
- Default value: FALSE
- Nature: VAR_OUTPUT

ErrorID

Error code (see table “ErrorID” in chapter 3)

- Type: INT
- Default value: 0
- Nature: VAR_OUTPUT

Drive errors are not considered during a parametrization.

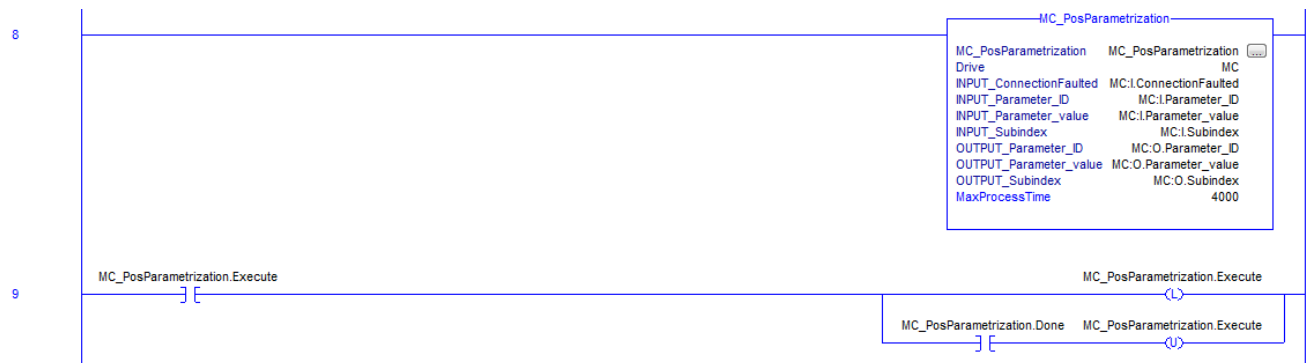
ErrorParameter

Parameter number that caused the error (in case of an error)

- Type: INT
- Default value: 0
- Nature: VAR_OUTPUT

If no error occurred, this value is 0.

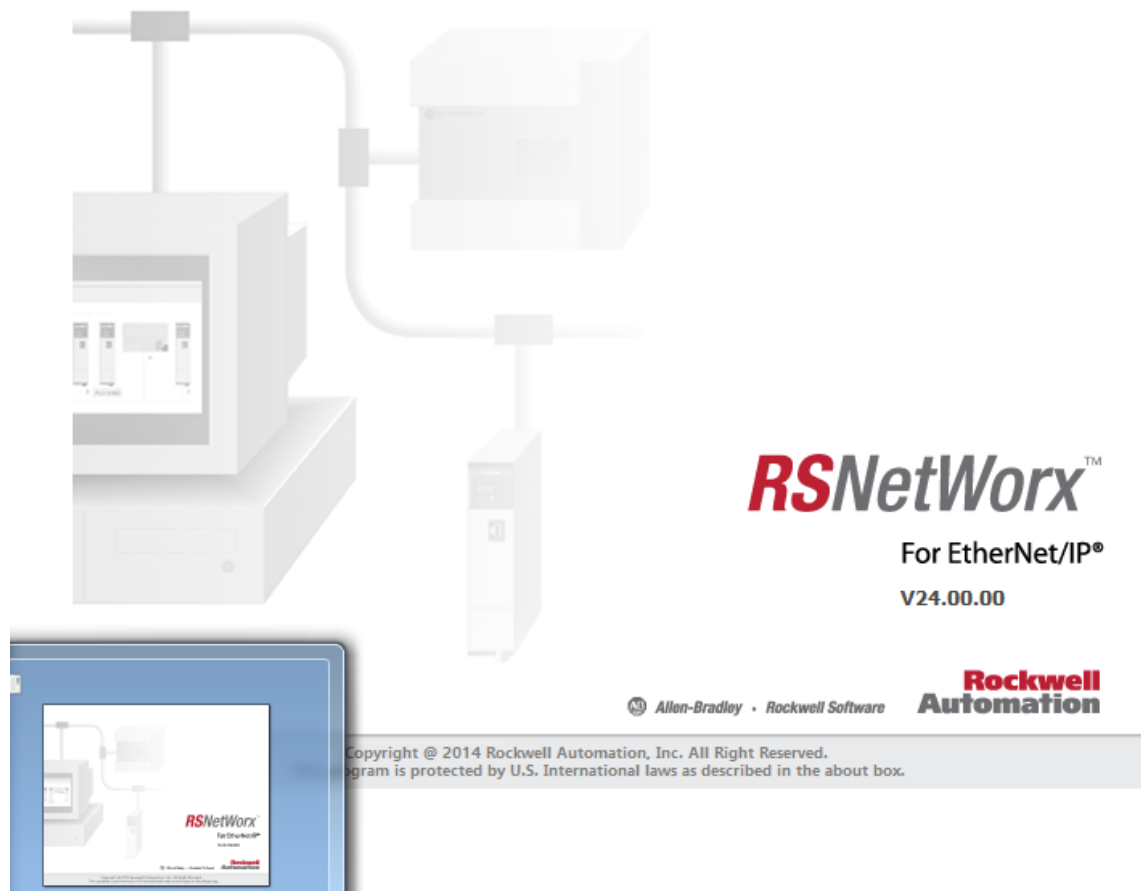
Simple ladder logic example with the AOI “MC_PositionParametrization”:



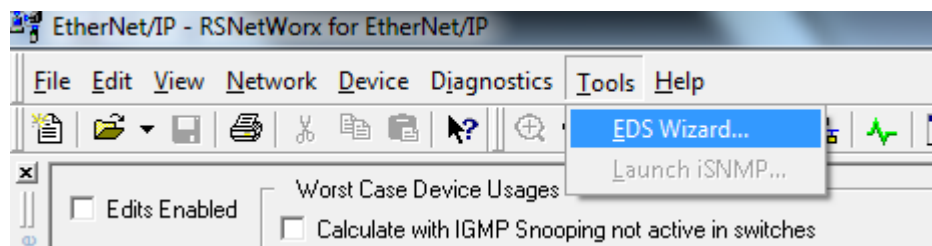
5 How to Add Drives to a project

5.1 Registering the appropriate EDS file with the help of RSNetWorx

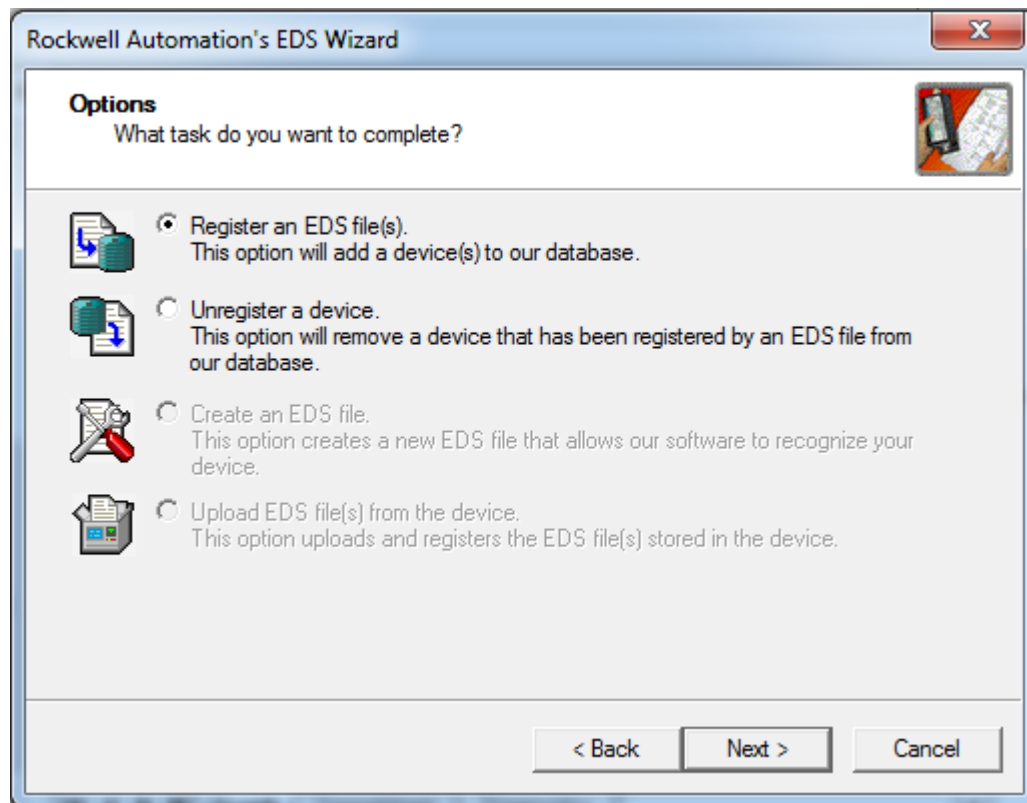
- First you must register the EDS file. To do that, open RSNetWorx for Ethernet I/P:



- Then navigate to “Tools”, then “EDS Wizard”:



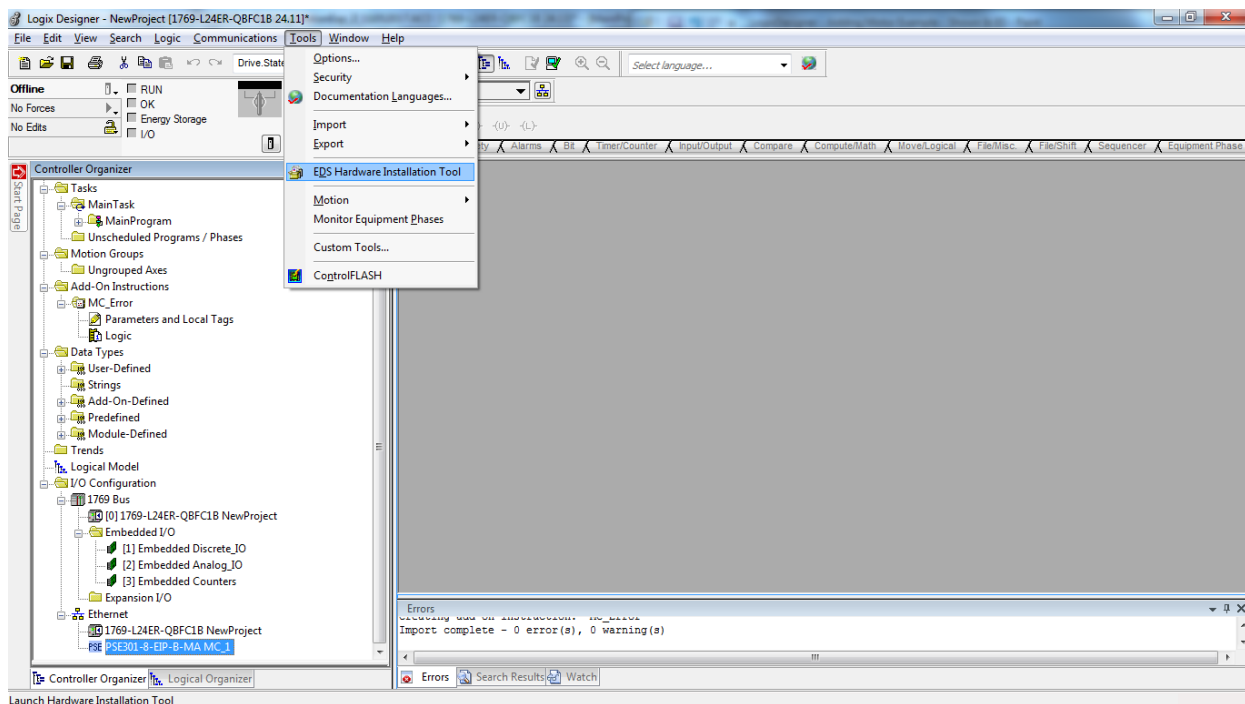
- Follow the Wizard's steps to register the files:



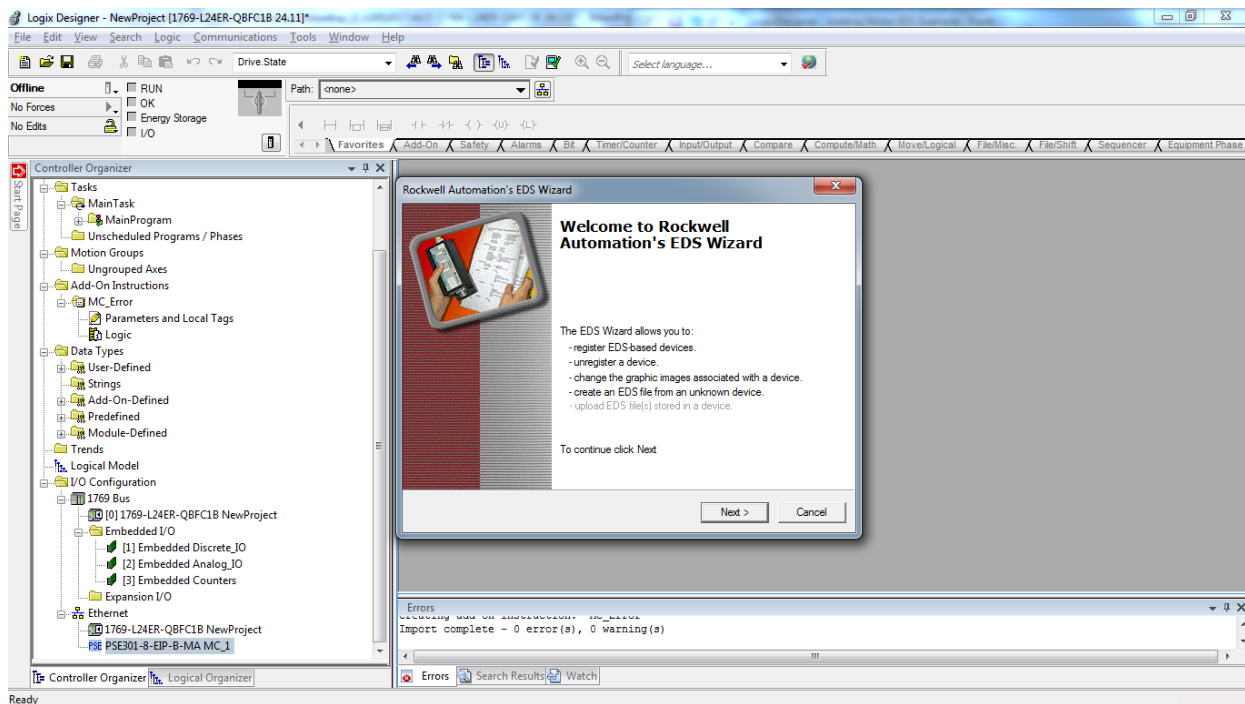
- When finished, close RSNetWorx.

5.2 Registering the appropriate EDS file with the help of Logix Designer

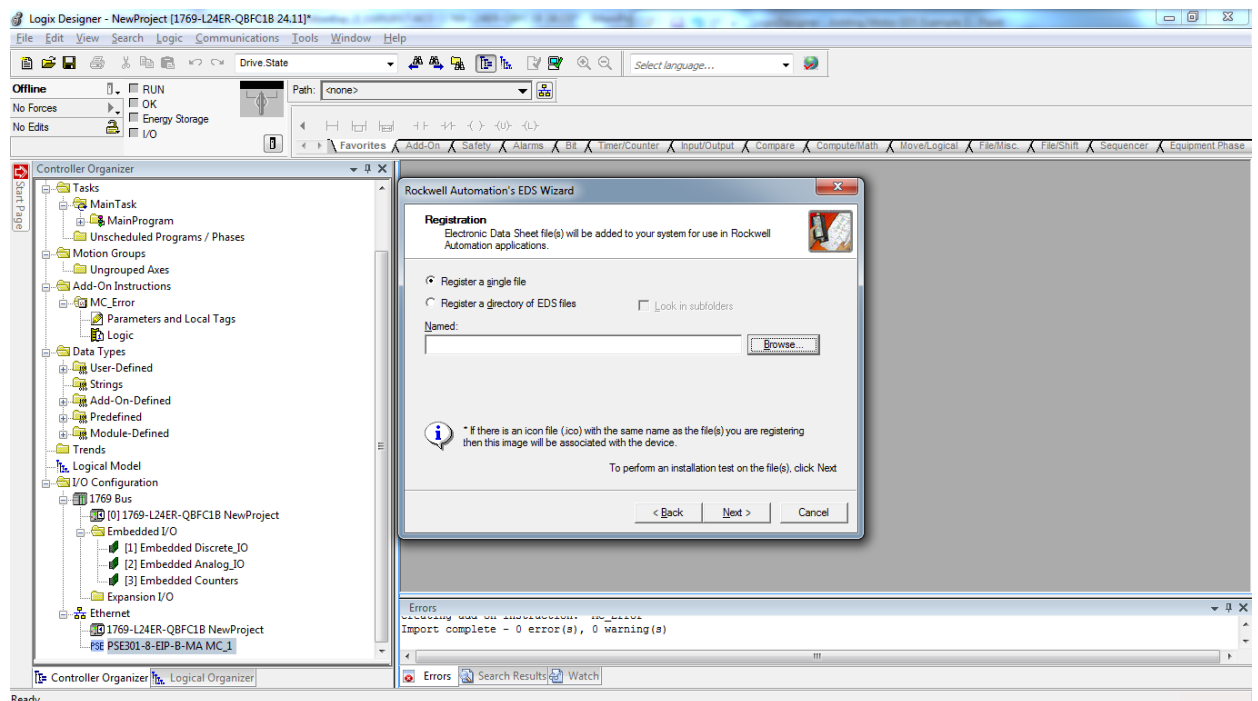
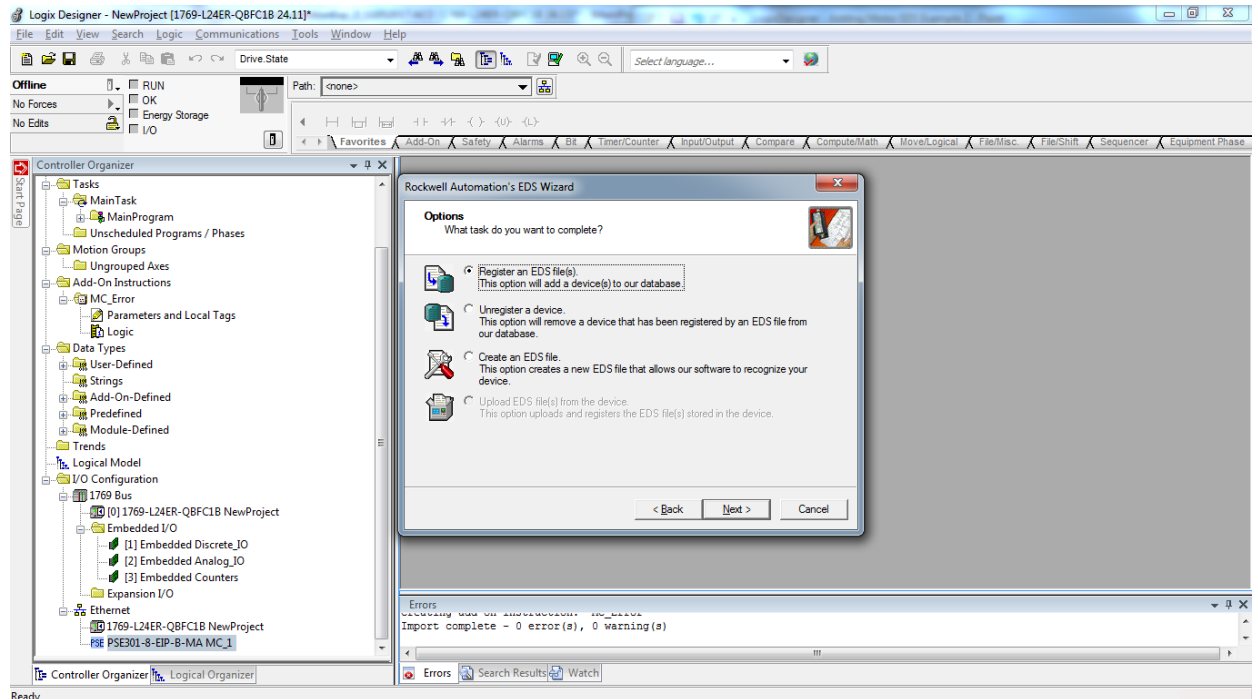
- Alternatively, you might register the EDS file with the help of the Logix Designer. First, navigate to “Tools”, then “EDS Hardware Installation Tool”:

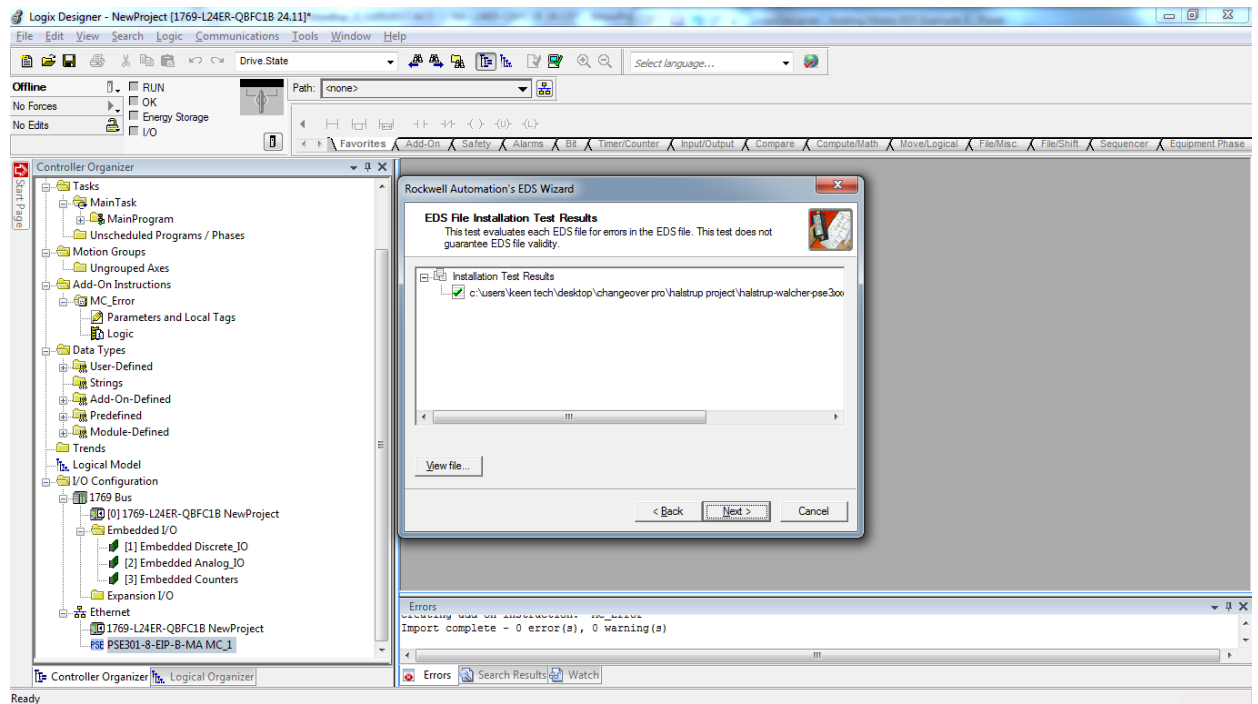
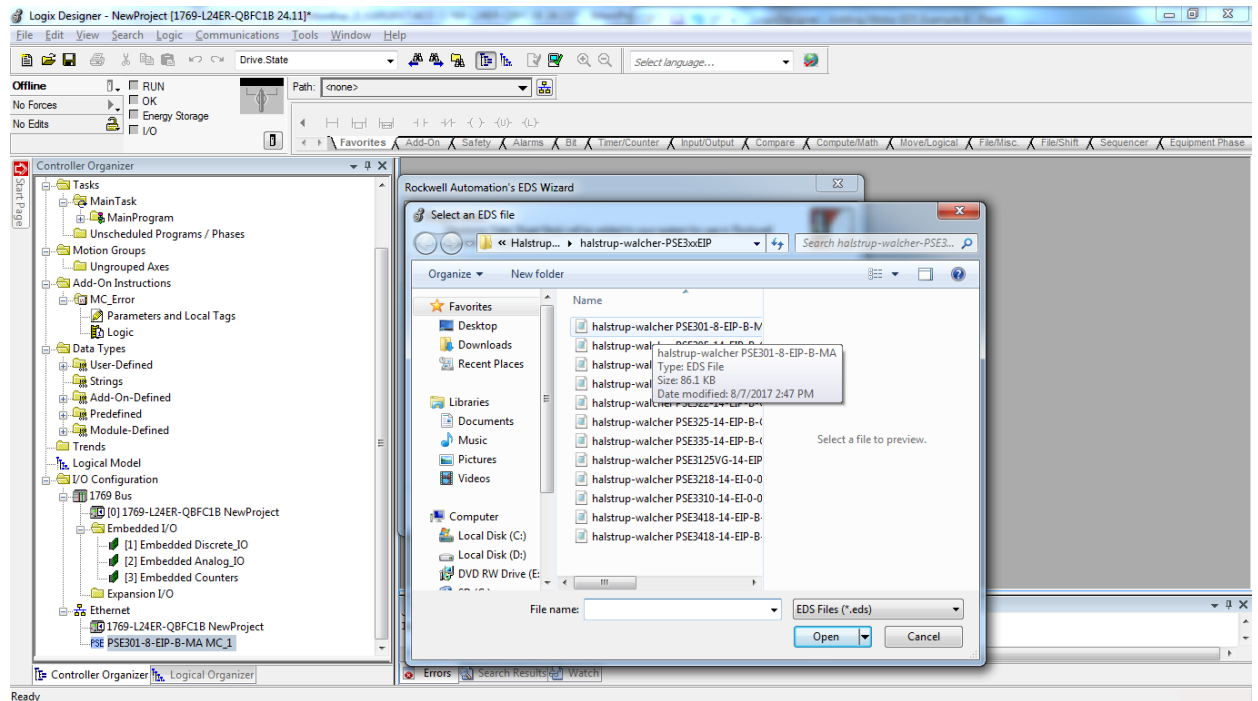


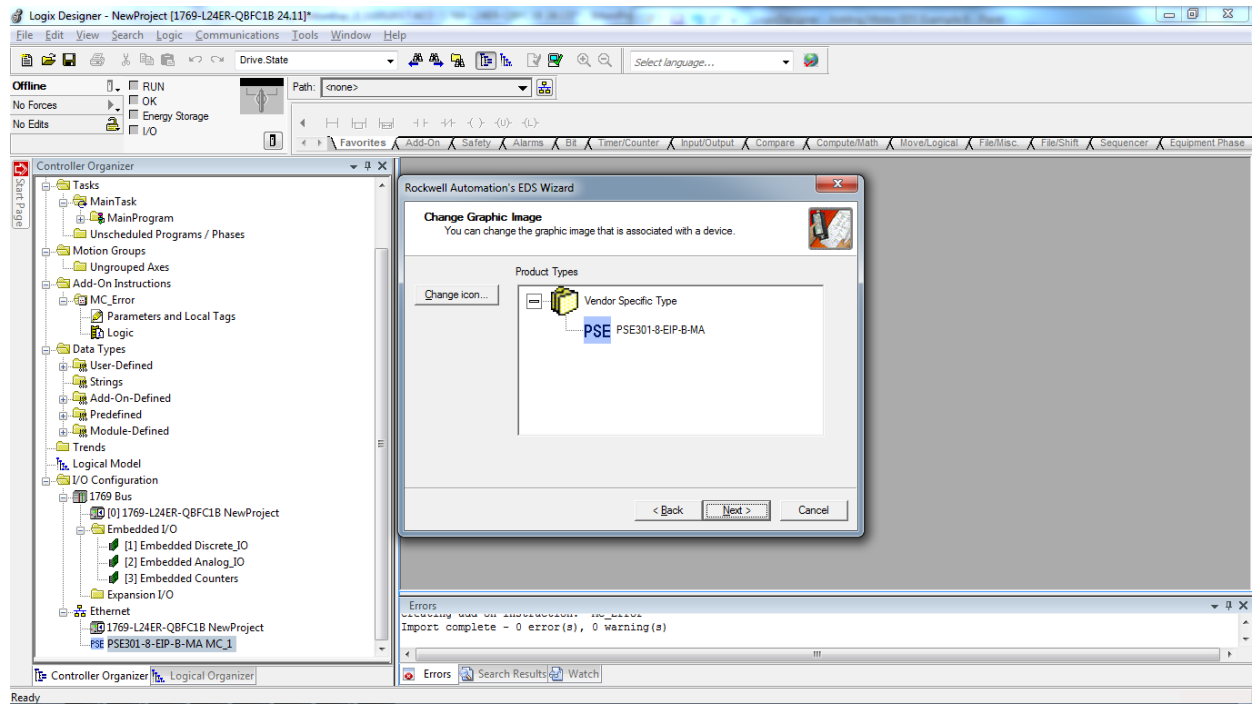
- Then select “Next” to start the EDS Wizard:



- Follow the Wizard's steps to register the files:

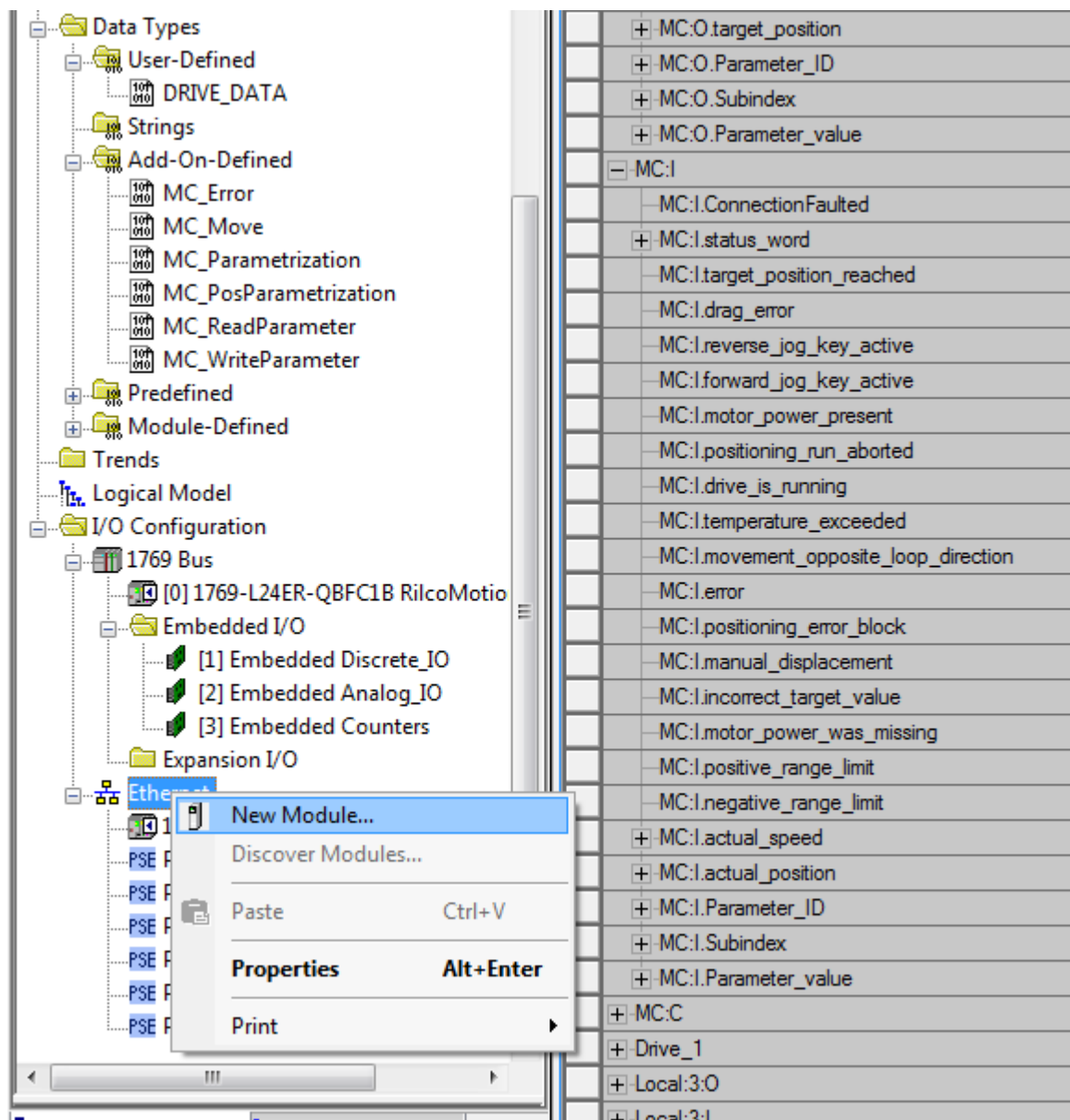




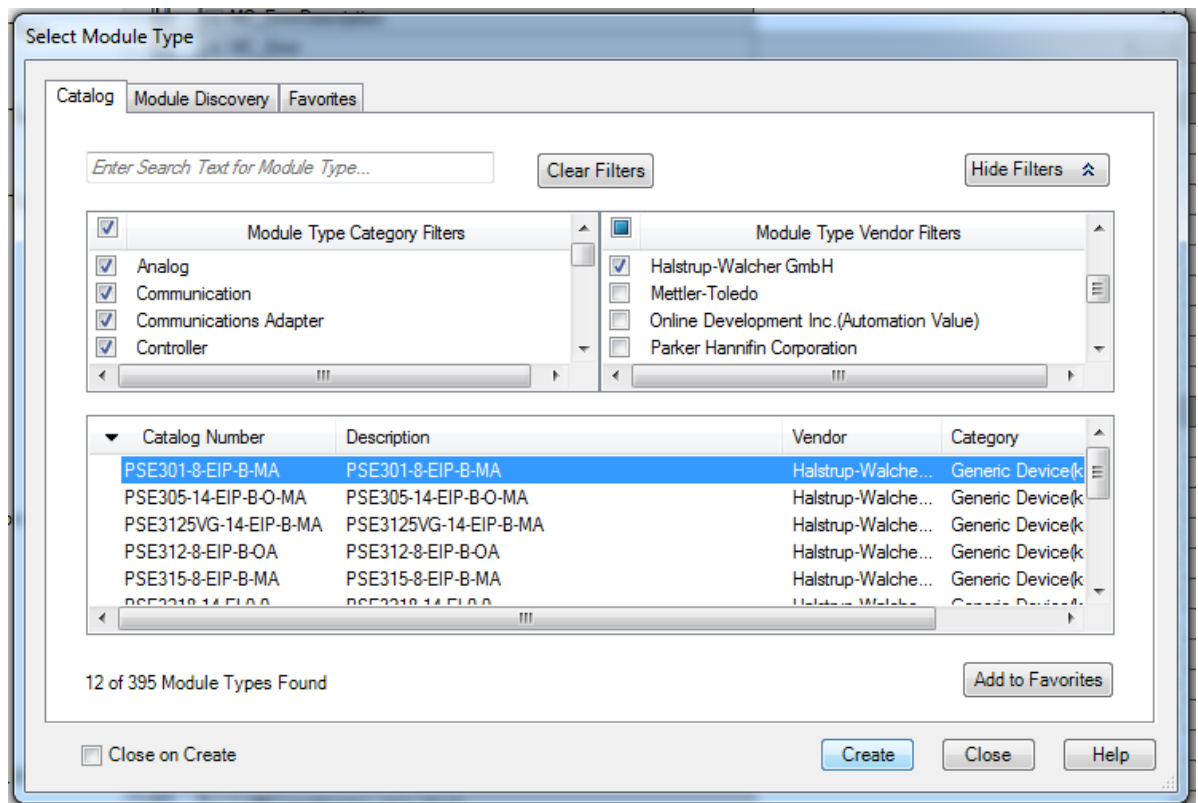


5.3 Adding a drive

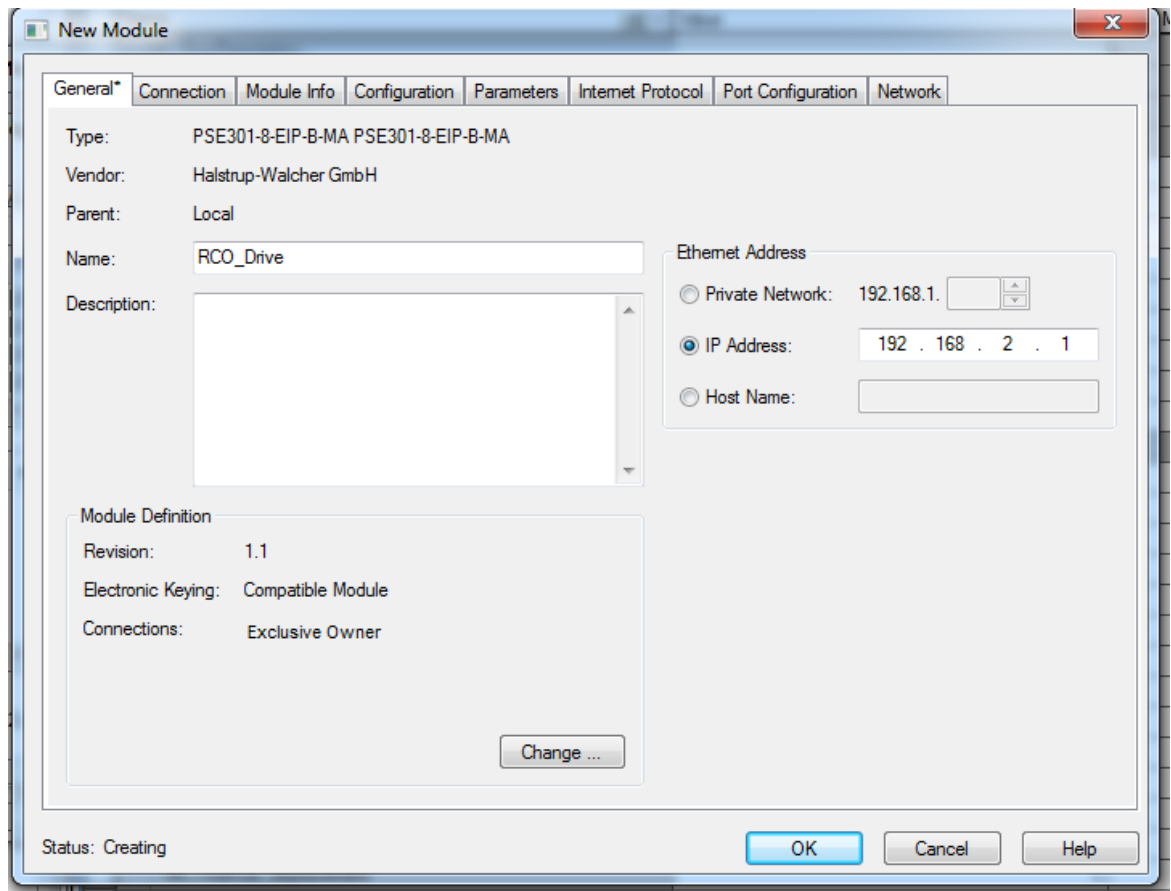
- In order to add a drive, open up Studio 5000 or RS Logix. At the bottom of the IO tree, select “Ethernet → “New Module”:



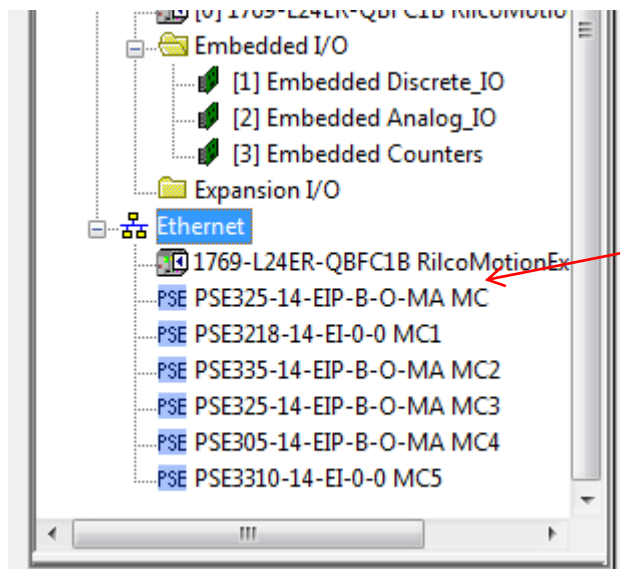
- The “Select Module Type” dialogue box appears. Choose your drive and click “Create”:



- Enter a name and IP Address and click “OK”:



- Next look at the IO tree under “Ethernet”:



→ The new module will appear in the module list.

- Next look at the Controller tags. There will be the input and output data for the drives.

This is displaying the OUTPUT data – this is the data that is going out to the drive:

MC:O	{...}	{...}		_03F2:P
MC:O.control_word	2#0000_0000_0000_0000		Binary	INT
MC:O.manual_run_to_larger_values	0		Decimal	BOOL
MC:O.manual_run_to_smaller_values	0		Decimal	BOOL
MC:O.transfer_target_position	0		Decimal	BOOL
MC:O.release_the_axle_will_only_run_if_the_bi	0		Decimal	BOOL
MC:O.target_position	12000		Decimal	DINT
MC:O.Parameter_ID	2#0000_0000_0000_0000		Binary	INT
MC:O.Subindex	16#0000		Hex	INT
MC:O.Parameter_value	0		Decimal	DINT

This is displaying the INPUT data – this is the data that is coming in from the drive:

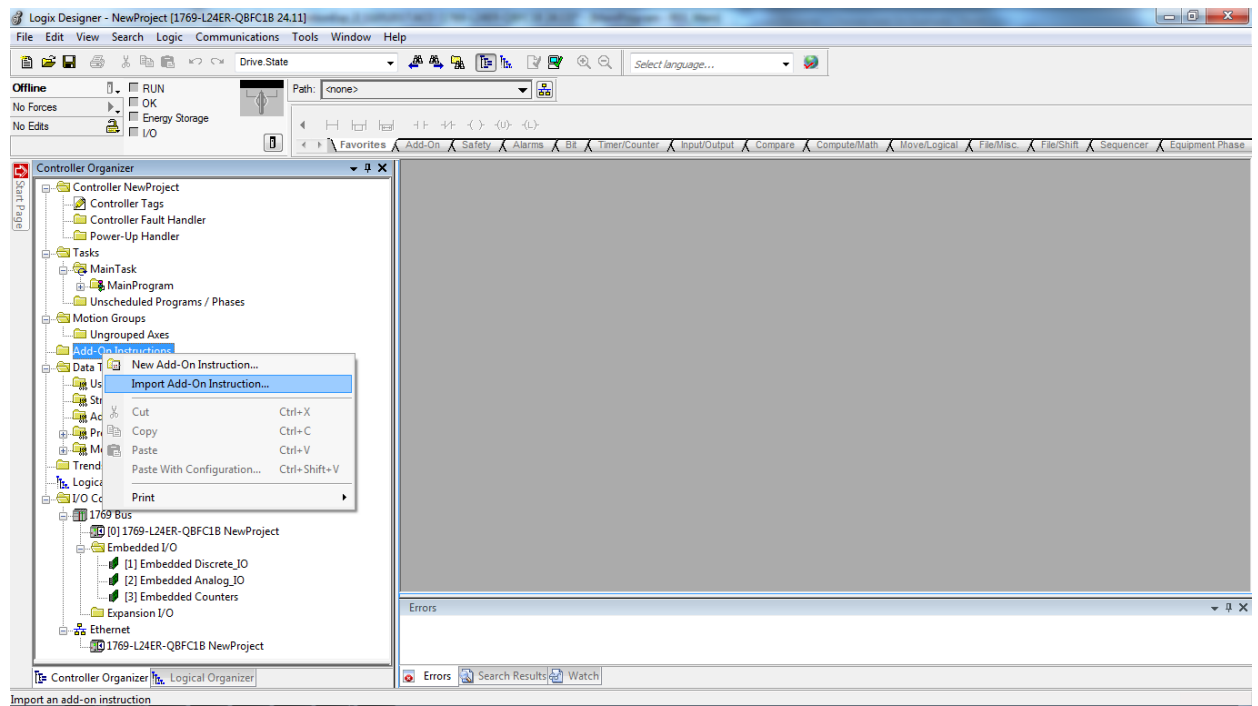
MC:I	{...}	{...}		_03F2:I
MC:I.ConnectionFaulted	0		Decimal	BOOL
MC:I.status_word	2#1000_1001_0001_0000		Binary	INT
MC:I.target_position_reached	0		Decimal	BOOL
MC:I.drag_error	0		Decimal	BOOL
MC:I.reverse_jog_key_active	0		Decimal	BOOL
MC:I.forward_jog_key_active	0		Decimal	BOOL
MC:I.motor_power_present	1		Decimal	BOOL
MC:I.positioning_run_aborted	0		Decimal	BOOL
MC:I.drive_is_running	0		Decimal	BOOL
MC:I.temperature_exceeded	0		Decimal	BOOL
MC:I.movement_opposite_loop_direction	1		Decimal	BOOL
MC:I.error	0		Decimal	BOOL
MC:I.positioning_error_block	0		Decimal	BOOL
MC:I.manual_displacement	1		Decimal	BOOL
MC:I.incorrect_target_value	0		Decimal	BOOL
MC:I.motor_power_was_missing	0		Decimal	BOOL
MC:I.positive_range_limit	0		Decimal	BOOL
MC:I.negative_range_limit	1		Decimal	BOOL
MC:I.actual_speed	0		Decimal	INT
MC:I.actual_position	167200		Decimal	DINT
MC:I.Parameter_ID	16#0000		Hex	INT
MC:I.Subindex	0		Decimal	INT
MC:I.Parameter_value	0		Decimal	DINT

(The style and type of these tags is determined by the EDS file associated with the drive.)

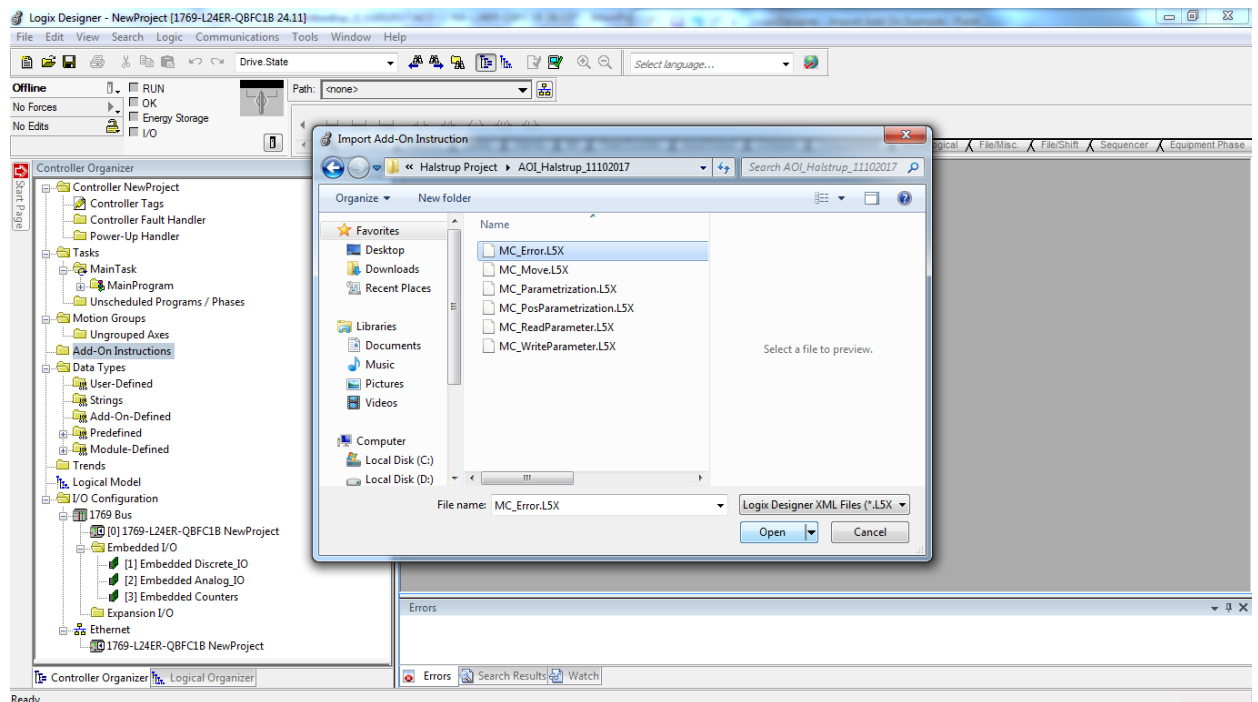
6 How to Add AOIs to a project

6.1 Import an AOI

- In the Controller Organizer, right-click on “Add-On Instructions”, then click on “Import Add-On Instruction”:

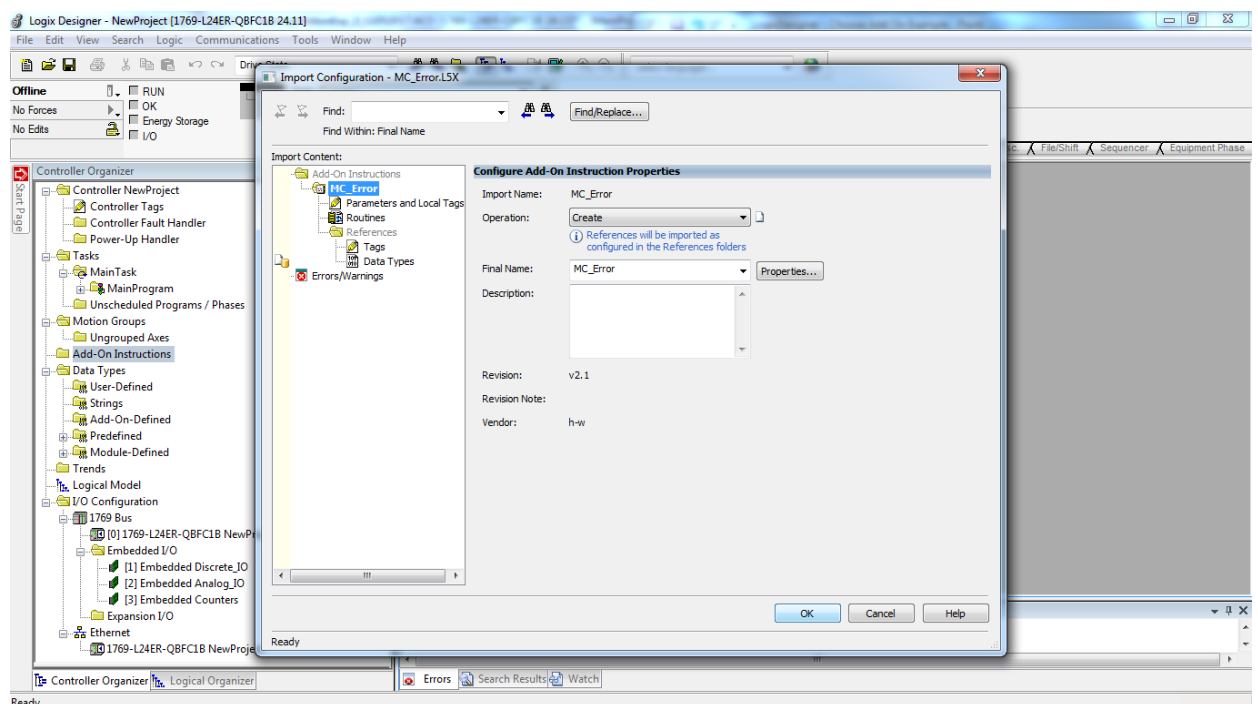


- Navigate to the halstrup-walcher library and select the desired AOI:



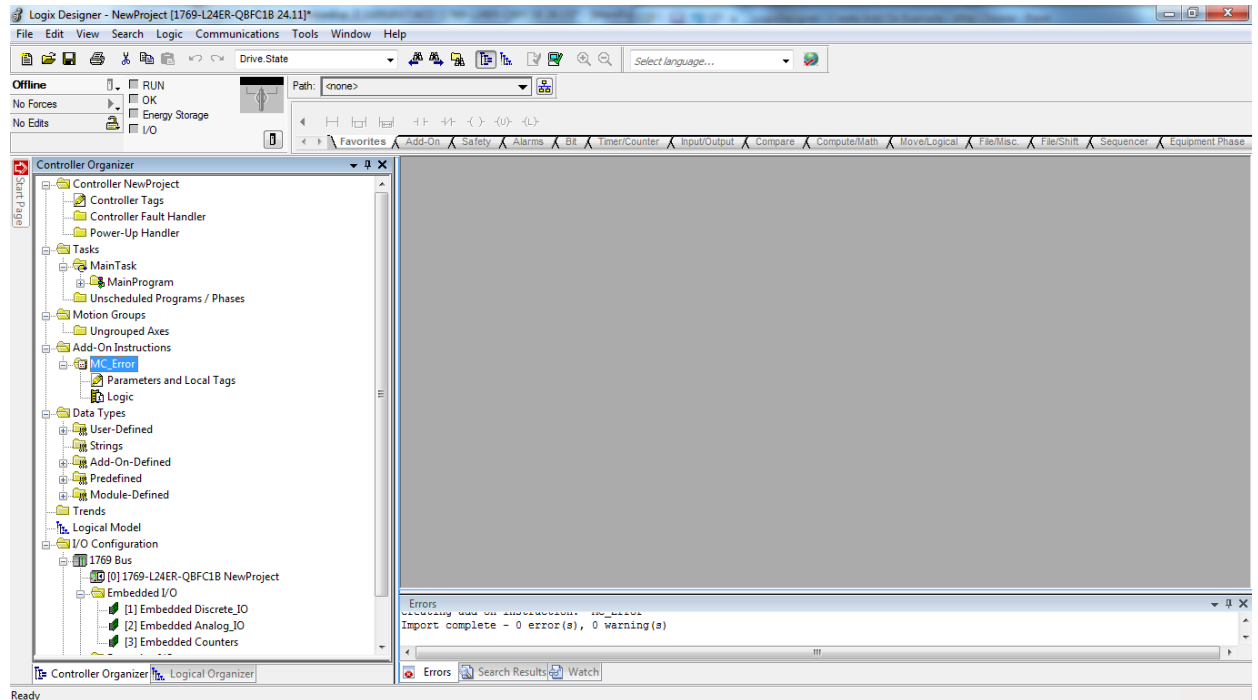
Then click “Open”.

- Then configure the AOI:



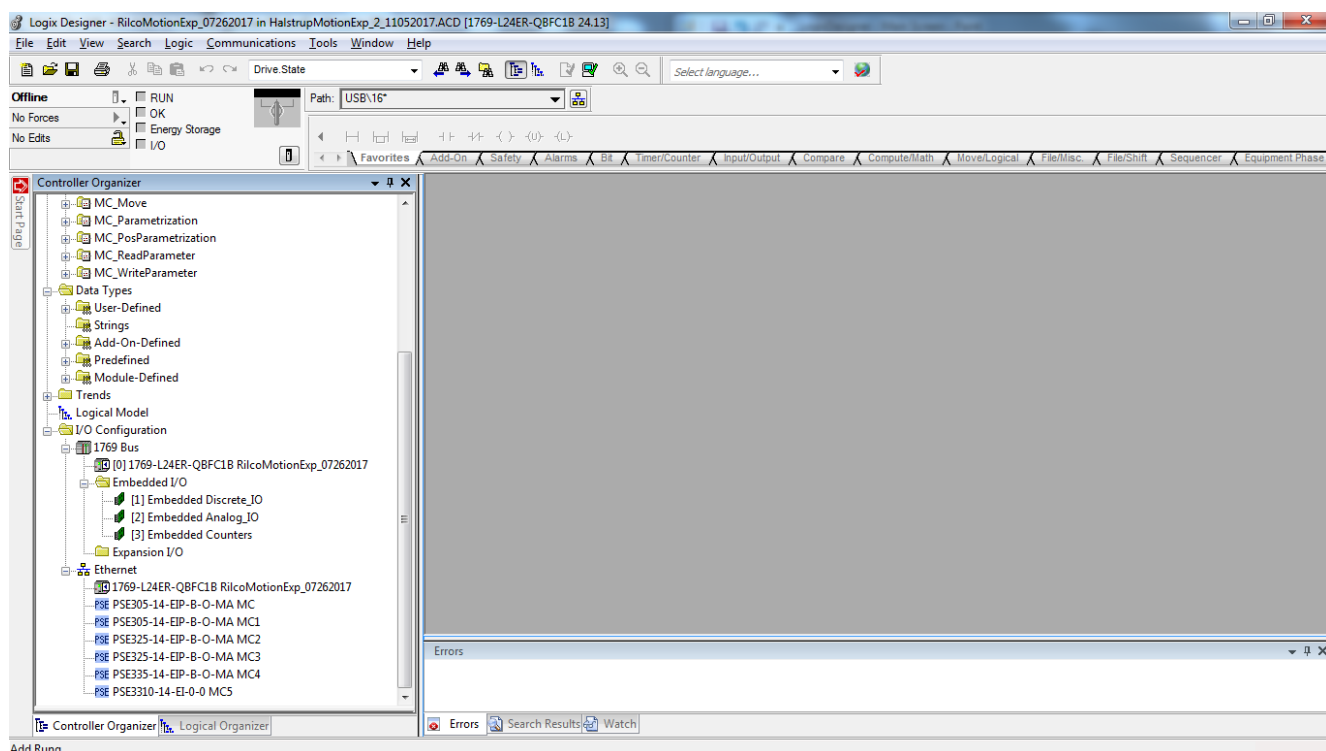
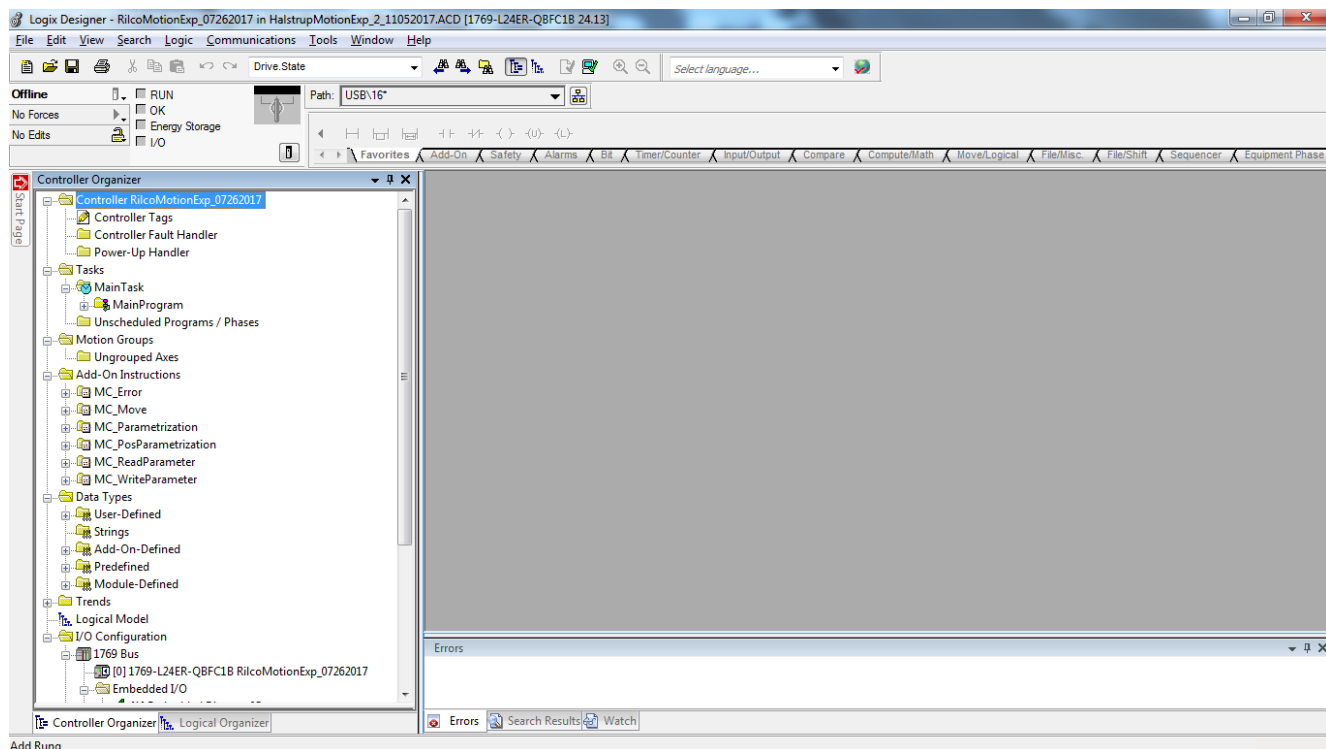
(The default settings might be sufficient for most applications.)

- As result, the AOI is listed in the Add-On Instructions section in the Controller Organizer:



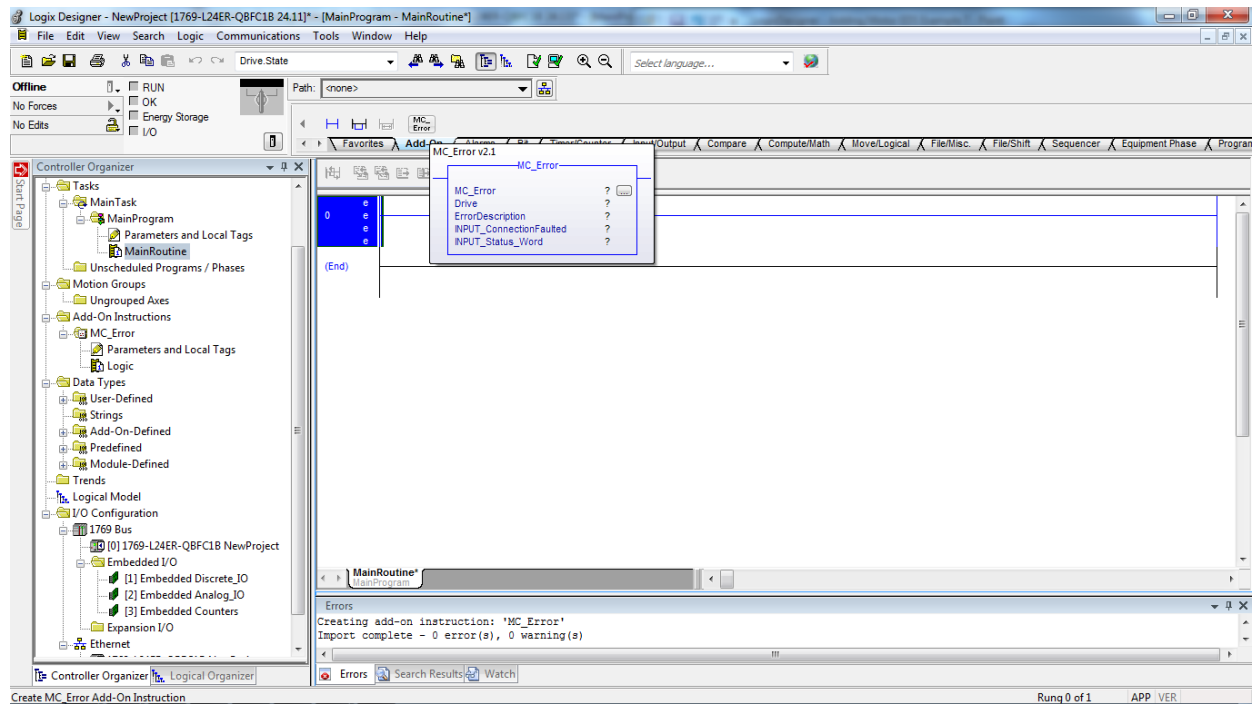
6.2 Representation of drives and AOIs in the Controller Organizer

Altogether, now each desired drive and each desired AOI is listed in the Controller Organizer of the main screen:



7 How to Add an AOI and Controller Tags to a Ladder

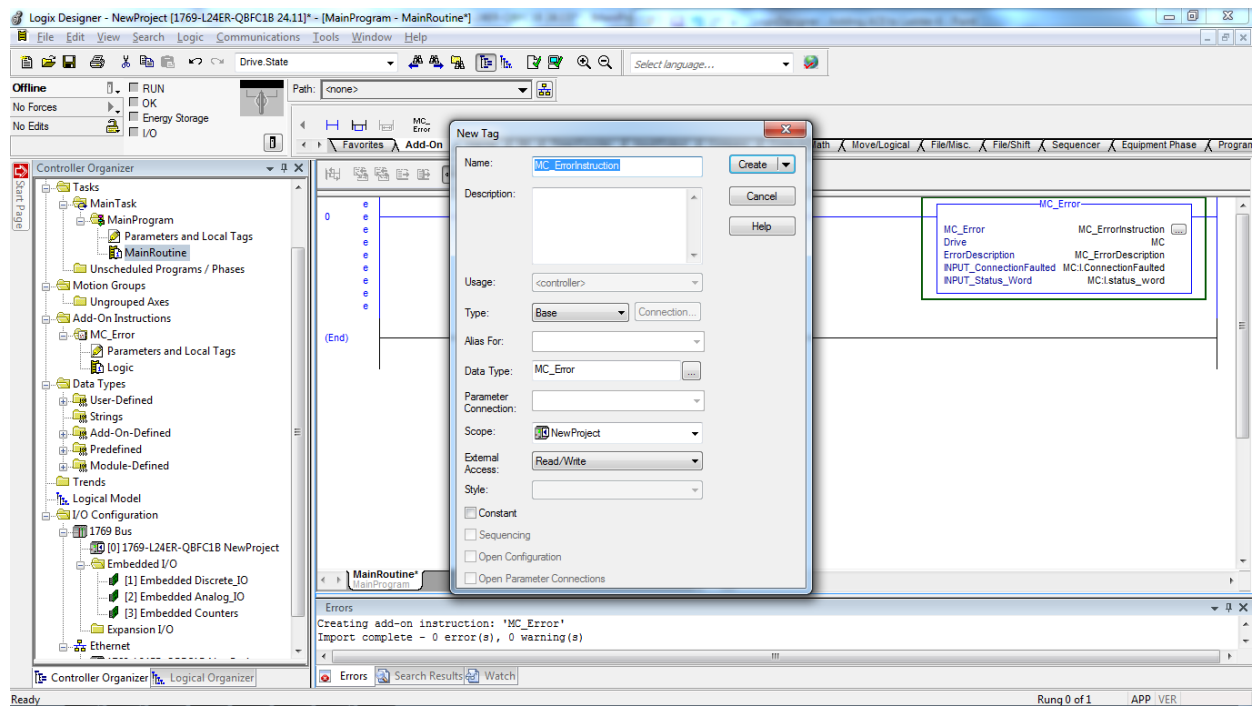
- In the desired routine, click on “Add-On” and select the desired Add-On (in this example the Add-On “MC_Error” which reports the state of the drive and the AOI as error bit, error code “ErrorID” and as text):



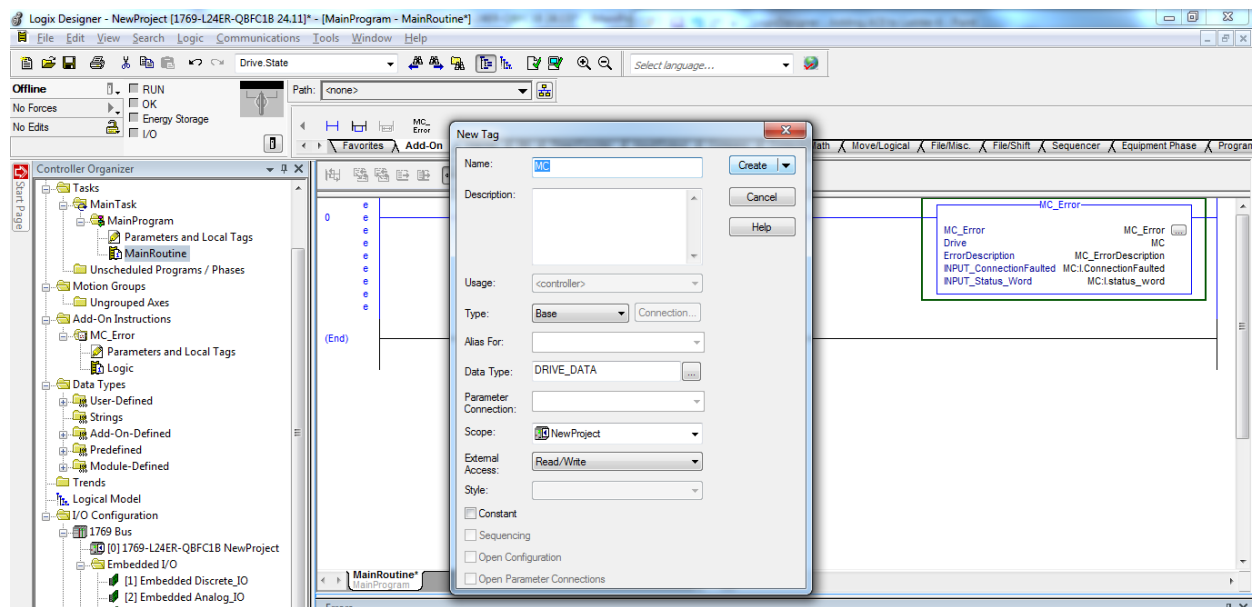
-
- The screenshot displays the Logix Designer software interface. The top menu bar includes File, Edit, View, Search, Logic, Communications, Tools, Window, and Help. Below the menu is a toolbar with various icons for file operations and a 'Drive State' dropdown menu. The main workspace is divided into three panes:
- Controller Organizer (Left Pane):** A tree view showing the project structure. The 'MainRoutine' is selected under 'MainProgram'. Other visible items include 'MainTask', 'Parameters and Local Tags', 'MainRoutine', 'Unscheduled Programs / Phases', 'Motion Groups', 'Unscheduled Axes', 'Add-On Instructions', 'MC_Error', 'Parameters and Local Tags', 'Logic', 'Data Types', 'User-Defined', 'Strings', 'Add-On-Defined', 'Predefined', 'Module-Defined', 'Trends', 'Logical Model', 'I/O Configuration', '1769 Bus', '1769-L24ER-QBFC1B NewProject', 'Embedded I/O', '[1] Embedded Discrete_IO', '[2] Embedded Analog_IO', '[3] Embedded Counters', 'Expansion I/O', and 'Ethernet'.
 - MainRoutine Ladder Logic (Center Pane):** A ladder logic editor showing a single rungs. The rung is labeled 'MC_Error' and contains a network with the following components:
 - MC_Error (coil)
 - Drive (normally open contact)
 - ErrorDescription (normally open contact)
 - MC_ErrorDescription (normally open contact)
 - INPUT_ConnectionFaulted (normally open contact)
 - MC_ConnectionFaulted (normally open contact)
 - INPUT_Status_Word (normally open contact)
 - MC_Status_Word (normally open contact)
 - Errors (Bottom Pane):** A message box indicating the creation of the add-on instruction 'MC_Error' and the completion of the import with 0 errors and 0 warnings.
- The status bar at the bottom shows 'Run 0 of 1' and 'APP_VER'.

-
- The screenshot displays the Logix Designer software interface for a project named [1769-L24ER-Q8FC1B 24.11]. The main window shows the 'MainRoutine' ladder logic. A context menu is open over the 'MC_Error' instruction, listing options such as 'New 'MC_Error'', 'Cut Instruction', 'Copy Instruction', 'Paste', 'Delete Instruction', 'Add Ladder Element...', 'Edit Main Operand Description', 'Save Instruction Defaults', 'Clear Instruction Defaults', 'Remove Force', 'Go To...', 'Instruction Help', 'Remove Parameter', 'Remove All Unknown Parameters', 'Open Instruction Logic', 'Open Instruction Definition', and 'Properties'. The 'Controller Organizer' on the left shows the project structure, including 'MainTask', 'MainProgram', 'Parameters and Local Tags', 'MainRoutine', 'Unscheduled Programs / Phases', 'Motion Groups', 'Ungrouped Axes', 'Add-On Instructions', 'MC_Error', 'Parameters and Local Tags', 'Logic', 'Data Types', 'User-Defined', 'Strings', 'Add-On-Defined', 'Predefined', 'Module-Defined', 'Trends', 'Logical Model', 'I/O Configuration', '1769 Bus', '1769-L24ER-Q8FC1B NewProject', 'Embedded I/O', '[1] Embedded Discrete_I/O', '[2] Embedded Analog_I/O', '[3] Embedded Counters', 'Expansion I/O', and 'Ethernet'. The status bar at the bottom indicates 'Run 0 of 1' and 'APP VER'.

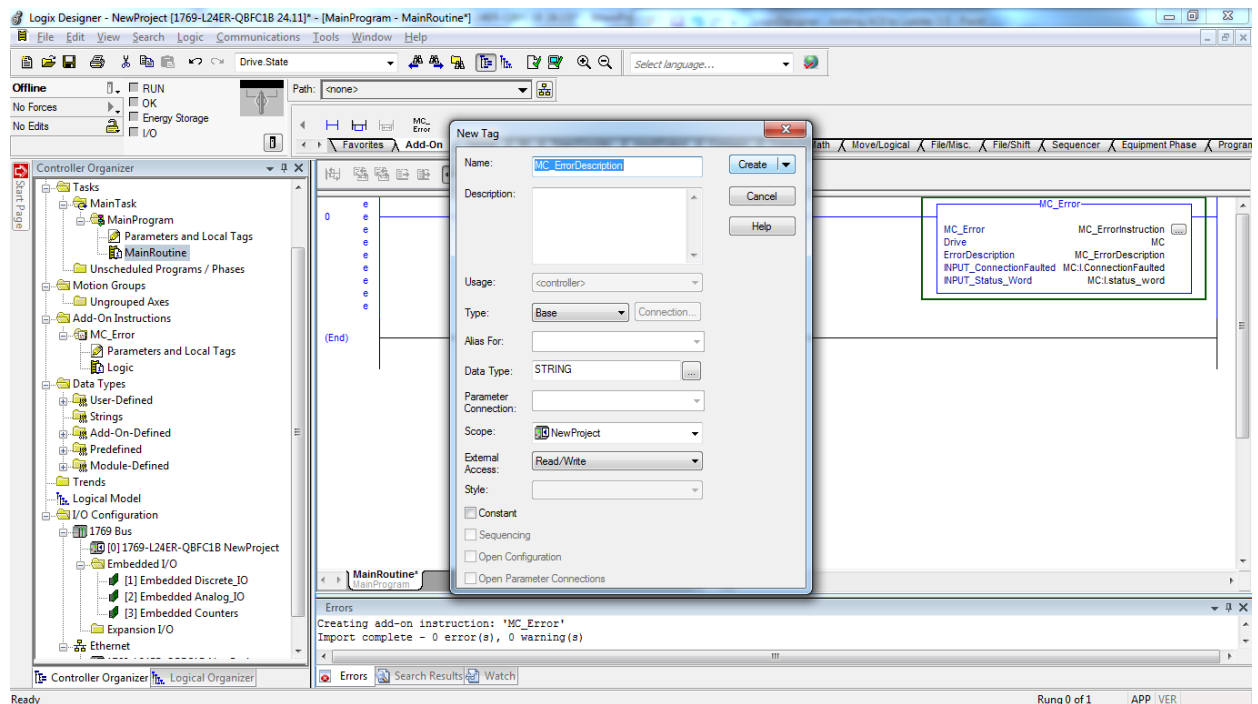
- Then create a new tag of the type “MC_Error”:



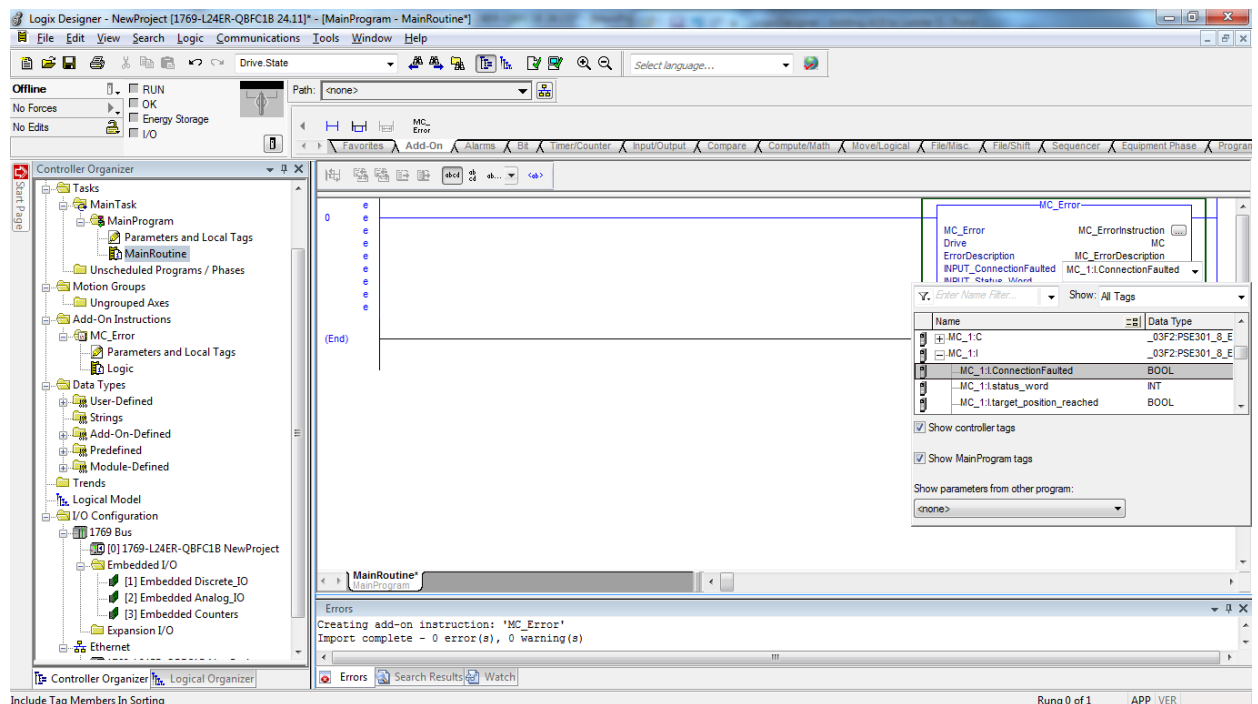
- Then select the desired drive for which this AOI shall be used (type “DRIVE_DATA”):



- Then select the desired string variable for the error description:



- Then select the connected boolean variable for the "Connection Faulted" tag of the corresponding drive:



- Then select the connected integer variable for the “status_word” tag of the corresponding drive:

