TR ELECTRONIC
. . . Passionate about Sensors.

TR Linear Encoder

Module and AOI Configuration Guide

email: customercare@trelectronic.com | web: www.trelectronic.com

US: Head Office: P.O. Box 4448, Troy, Michigan, 48099 | Tel: (248)-244-2280 | Fax: (248)-244-2283 | Toll Free: 1-800-709-3300
Canada: Head Office: P.O. Box 2543, Station B, London, Ontario, N6A 4G9 | Tel: (519)-452-1999 | Fax: (519)-452-1177 | Toll Free: 1-800-265-9483

Contents

Introduction

This guide is intended to help install and use a TR Linear with switch Ethernet IP encoder quickly and successfully. It goes through all of the critical implementation steps in detail including: EDS (Electronic Data Sheets) installation, RSLogix module configuration, and AOI (Add-On-Instruction) implementation.

It is important to note that this guide does not need to be followed to have success with a TR Linear encoder. There are plenty of ways to successfully integrate the device into an automation environment. However, this guide is recommended as an efficient way to ensure the encoder is configured as expected, and position values are properly understood.

Prerequisites

Before beginning this guide, it is recommended you have the following files downloaded and stored in a known location:

- Required:
  - Device user manual: “Magnetostriction LA, LP, LMP”.
    - http://www.tr-electronic.com/service/downloads/operating-manuals/encoder-and-linear-transducer.html#c17692
  - Device EDS file: “04710022_TR_Lxxx_3M_010101.eds”.
    - http://www.tr-electronic.com/service/downloads/file-download.html
- Required if you plan on using the AOI:
  - AOI file: “TR_Linear_AOI.L5X”

Add Module and Import AOI

The first step to successfully using the AOI is adding the TR_Linear module and corresponding AOI to your project RSLogix project.

Add module

Ethernet IP communications require that a module for the encoder is added to the IO tree in your RSLogix project. There are two possible modules that could be used: TR-Lxxx_3M or a Generic Ethernet Module. If you plan on using the AOI provided, the TR-Lxxx_3M must be used. It is still recommended to use the TR-Lxxx_3M even if you do not plan to use the AOI, but if you'd like to use the Generic Ethernet Module then this document will not be helpful. Please refer to "Magnetostriction LA, LP, LMP" for the IO and Configuration Assembly Object information.

The AOI was designed to work with EDS version 1.1 of the TR-Lxxx_3M module. It is possible you've previously installed version 1.1

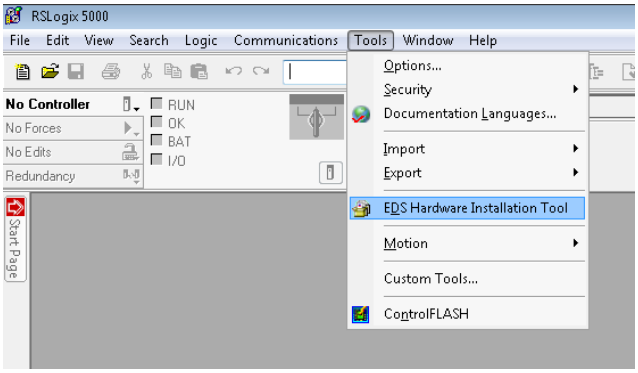
If you've already installed version 1.1 of the TR-Lxxx_3M, this step can be skipped.

If version 1.1 must be installed, perform the following step.

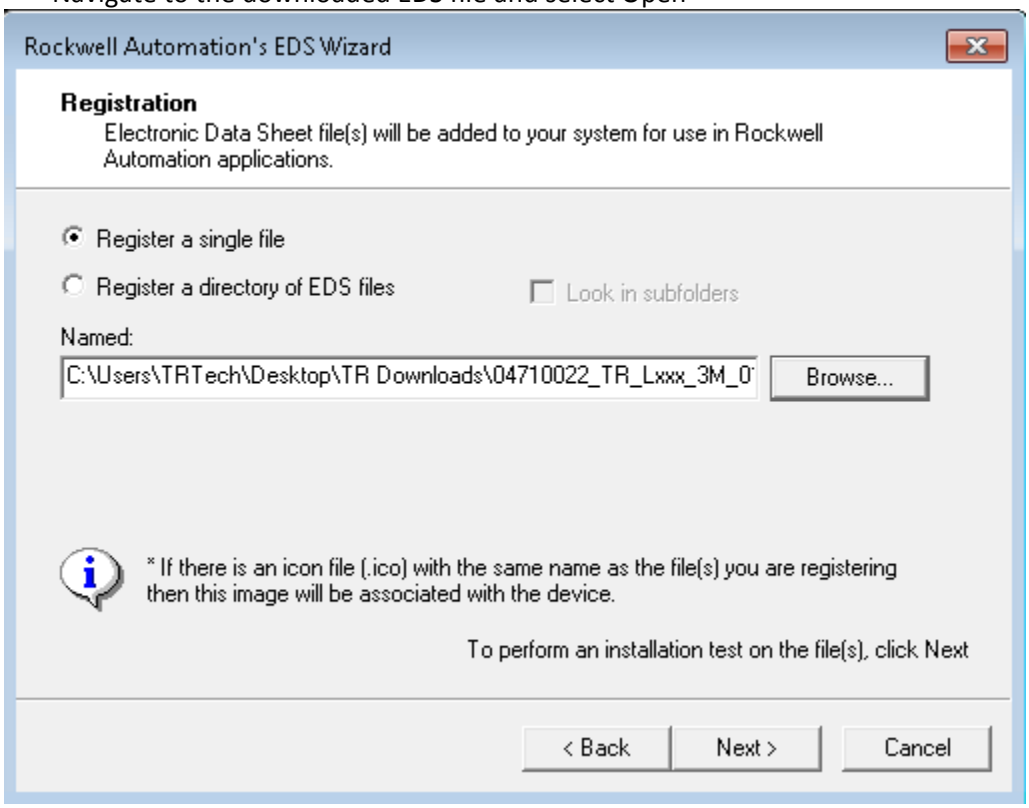
Install EDS

To install the EDS file, open RSLogix and select:

- Tools
  - EDS Hardware Installation Tool



- Within the Rockwell Automation's EDS Wizard select:
  - Next
  - Next – with Register and EDS file(s) selected
  - Browse:
    - Navigate to the downloaded EDS file and select Open

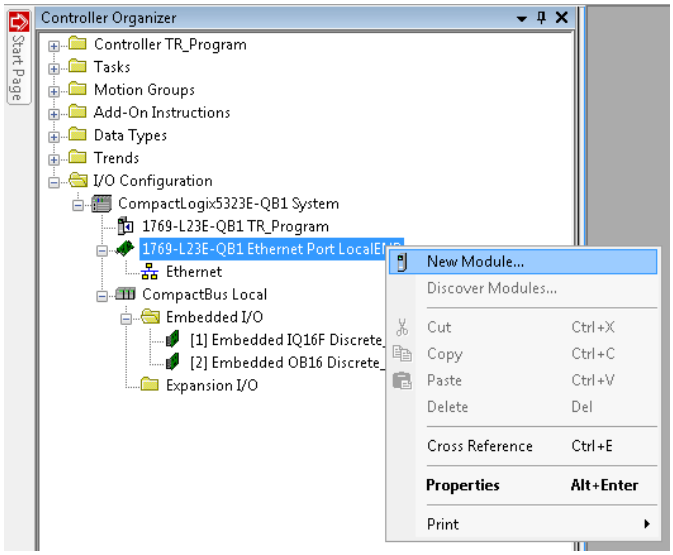


  - Next, Next, Next, Next, Finish

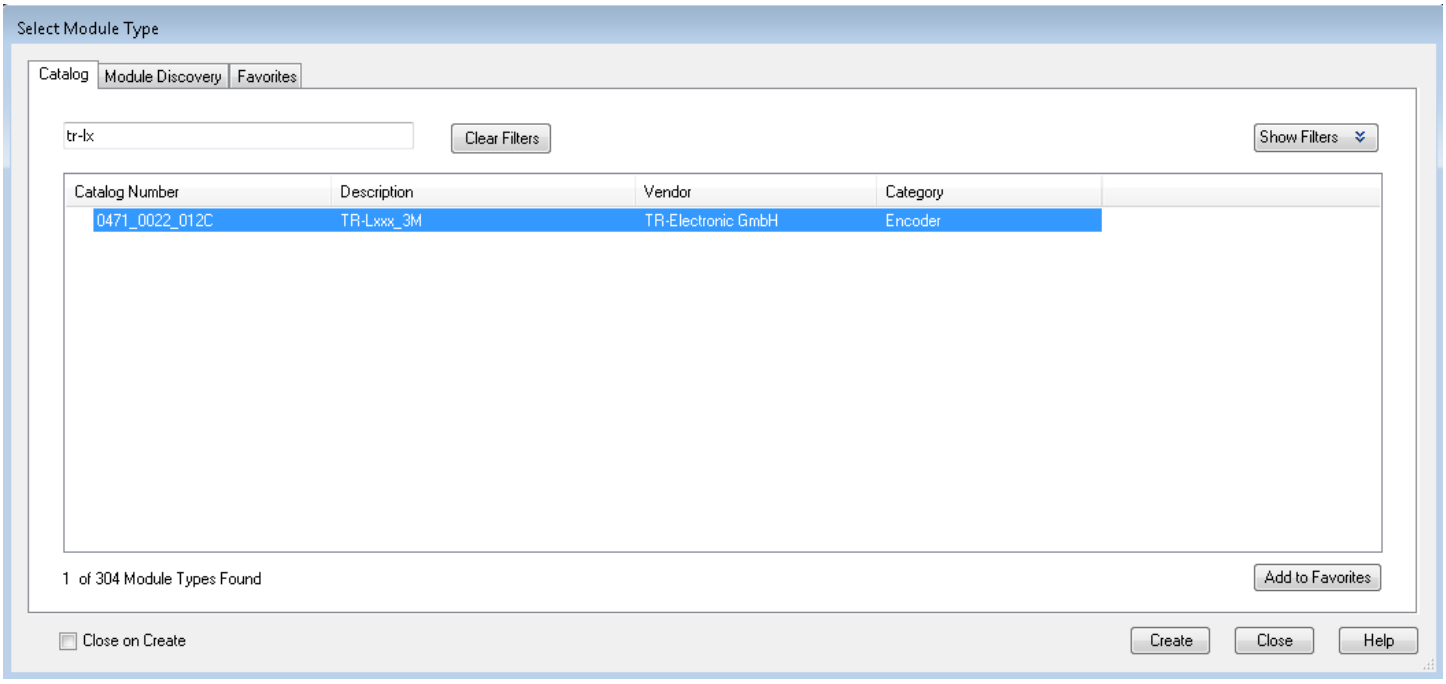
Add Module

Once the correct EDS file has been installed, a module needs to be added to the project. Start by opening your working project.

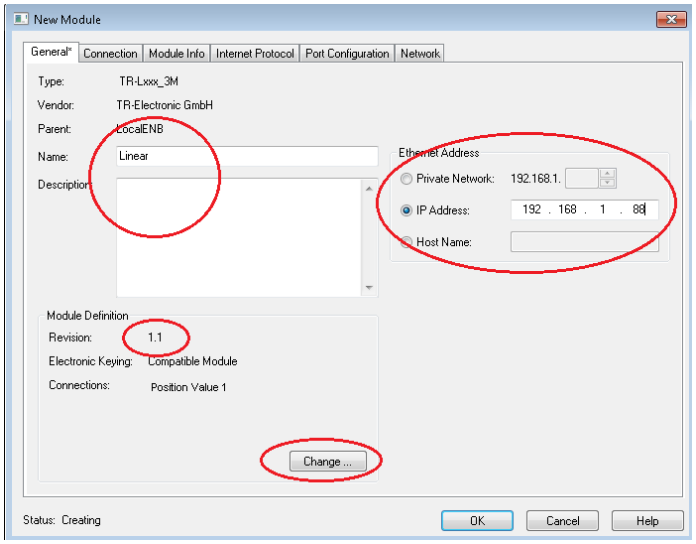
- Right click on your Ethernet module in the IO tree and select new module



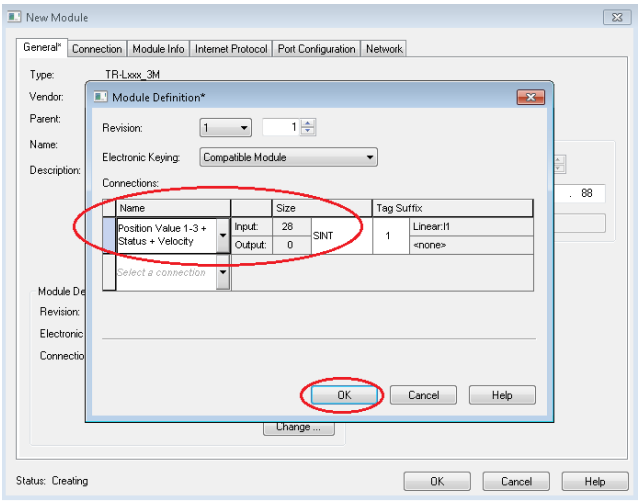
- Search for “tr-lx”, select TR-Lxxx_3M and then select Create.



- Give the module a name, description (optional) and IP address.
    - This guide will continue using the name Linear. However you may choose a name of your preference. Whenever Linear is referenced in this document, replace it with your module name.
- Select Change



- Within the Module Definition ensure the Connections Name and Size match below. This is a requirement for the AOI to function.



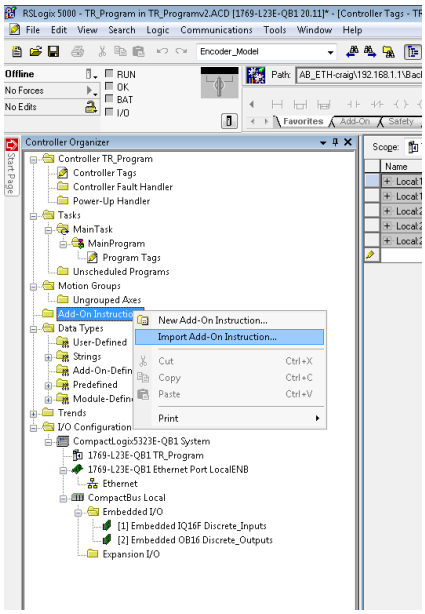
If a popup appears, click Yes. Click OK again to exit Module Properties.

If you do not plan on using the AOI, then please skip to the “Configure Encoder” section. It is the final section of the manual that will be helpful for module configuration only.

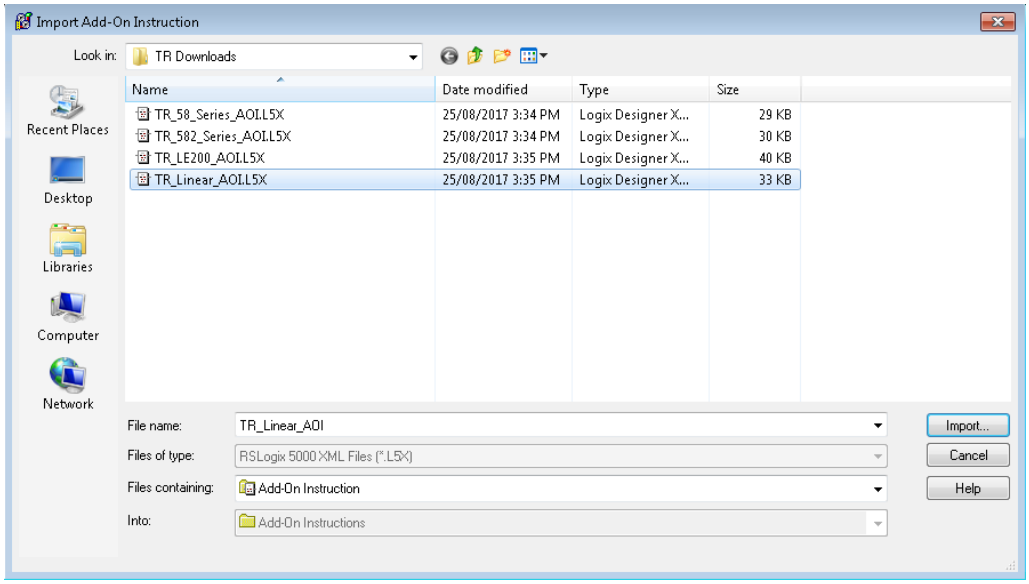
Import AOI

Now that a module has been added to the program, we need to add the logic that will interact with the module. This piece of logic is an AOI (Add-On Instruction) that must be imported into the project.

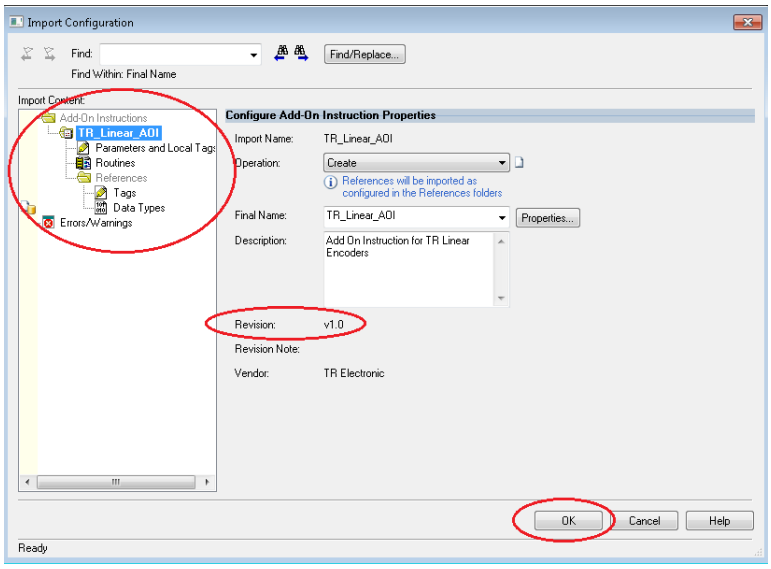
- Right click on Add-On Instructions in RSLogix under the Controller Organizer and select Import Add-On Instruction.



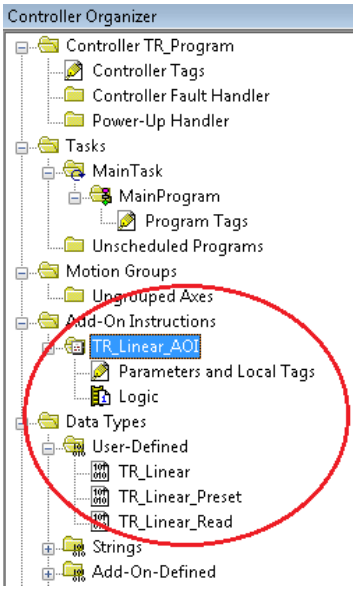
- Navigate to the AOI file that was downloaded and select Import:



- An Import Configuration dialog box will open with information regarding changes that come along with importing this AOI (Routines, Tags, Data Types, etc). Navigate around this dialog box for more information, then select OK.
    - Note the revision. If you've previously installed and are using an earlier revision, please review the consequences of overwriting. It might be best to change the "final name" of the AOI and imported Data Types to something else, so the new AOIs and UDT can be installed without impacting the old one.



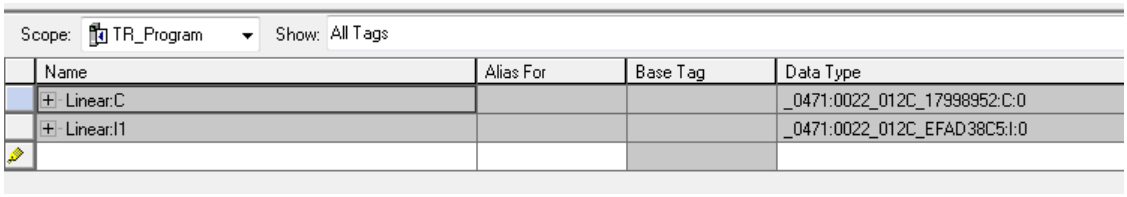
- When the AOI is imported, you'll see it under Add-On-Instruction in Controller Properties. In addition, under Data Types -> User Defined, you should see 3 new datatypes. These are the new datatypes introduced when the AOI was imported. They will be discussed in further detail in later sections.



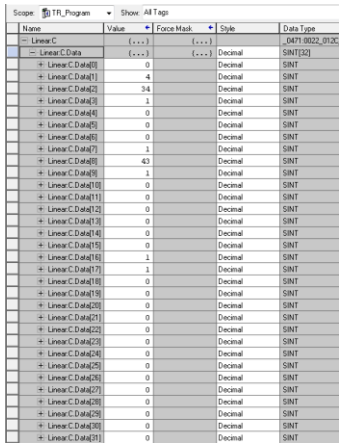
Configuring Encoder and AOI

Configure Encoder

After adding the module, you'll see two new tags in your tag database. The tag name will depend on what you named the new module, which was "Linear" in this example.



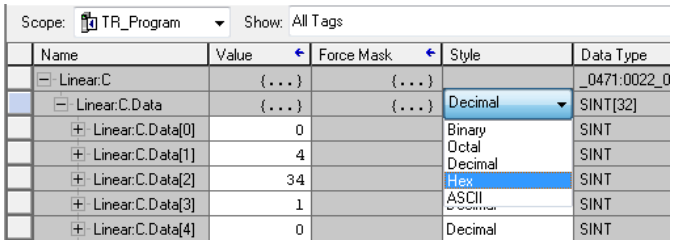
Linear:I1 is the IO Assembly and contains the feedback/readings directly from the encoder. Linear:C is the Configuration Assembly and is used to quickly configure the encoder upon successful connection with the PLC. Expanding the configuration assembly looks like below. The initial values have been populated as defined in the EDS file:



Refer to Appendix **Fehler! Verweisquelle konnte nicht gefunden werden.** to edit these values as required.

It is important to note that the configuration assembly values must be populated byte by byte. Recommended procedure for updated these values is:

- Set Linear:C.Data style to Hex.



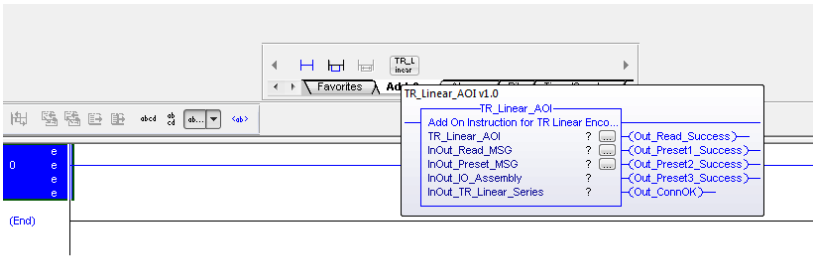
- Convert desired value into Hex
- Break converted Hex number into bytes (2 Hex digits)
- Enter value byte by byte into Linear:C.Data array.

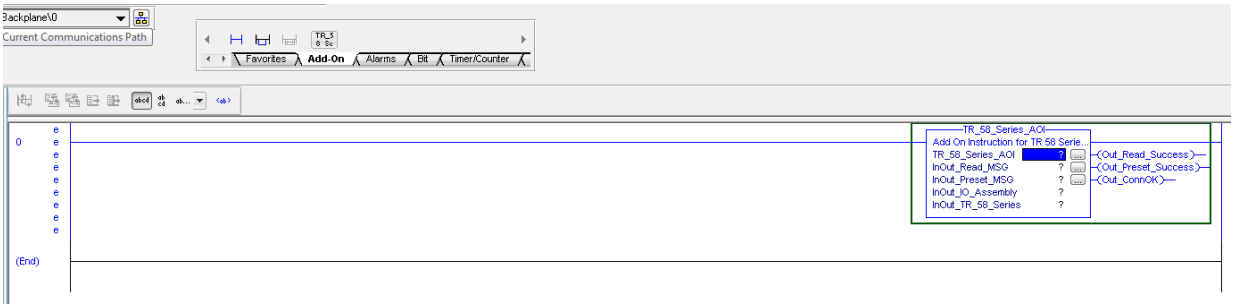
Refer to Appendix **Fehler! Verweisquelle konnte nicht gefunden werden.** for a conversion example.

Configure the AOI

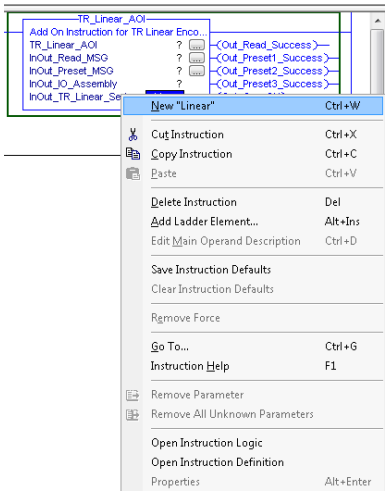
Now that the AOI has been imported, it has to be implemented in ladder logic.

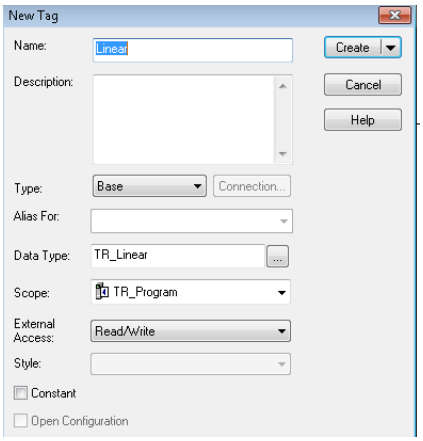
- The new instruction can be called by navigating to the desired routine and rung by selecting it on the standard instruction toolbar:





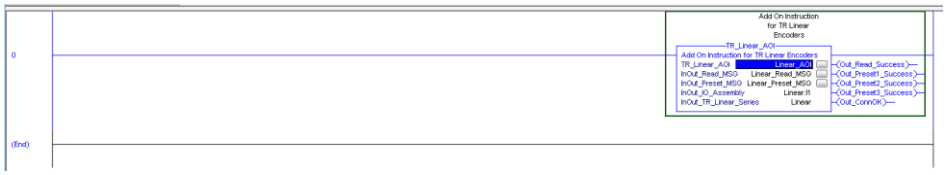
- Now we have "called" the AOI. However in it requires certain variables to be passed to it to function properly. Variable fields can be edited by double clicking.
    - InOut_TR_Linear field:
        - Enter desired encoder name, you can re-use the module name as I have done here. You'll have to create a new variable (of type TR_Linear) to use the AOI as discussed later. Note the scope (adjust if necessary) and add a description to the tag if you wish.



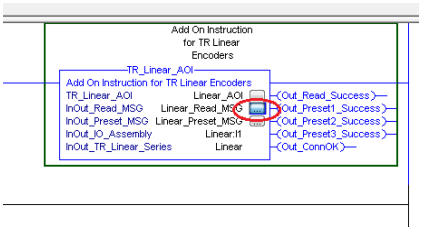


- InOut_IO_Assembly field:
  - Enter Linear:I1
    - NOTE: The AOI requires this variable to be of a certain type as defined in the module properties (Position Value 1-3 + Status + Velocity, SINTs). If you'd like to change the module properties to something else, you'll have to change the AOI parameters accordingly.
- TR_Linear_AOI field:
  - Enter Linear_AOI and create a new tag.
- InOut_Read_MSG field:
  - Enter Linear_Read_MSG and create a new tag.
- InOut_Preset_MSG field:
  - Enter Linear_Preset_MSG and create a new tag.

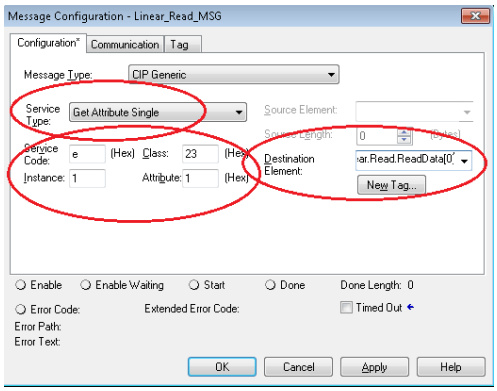
- The rung should now look something like this:



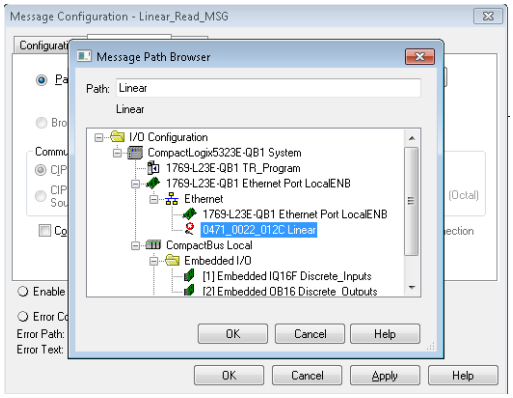
- Now the MSG instruction must be configured. Start with the Linear_Read_MSG. Select the "…" button:



  - Enter the following information (Destination Element reads Linear.Read.ReadData[0])
    - Destination element is critical. If a different value is entered the MSG instruction may still yield a successful result, which will set the AOI's success output even though data could be misallocated.

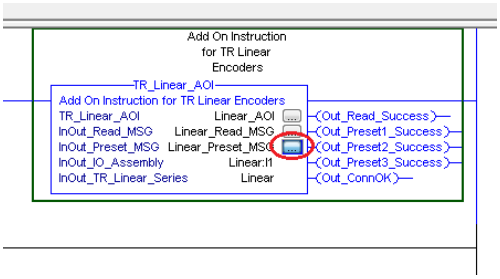


- Then select the Communication tab to configure the path. Click Browse and navigate to the new module:

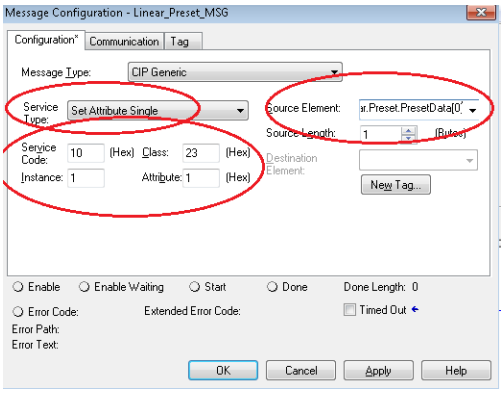


  - Select OK on the Message Path Browser to close the dialog box.
  - Select OK to close the Message Configuration dialog box.

- Then the Linear_Preset_MSG message must be configured. Click on the "…" button:



  - Enter the following information (Source Element reads Linear.Preset.PresetData[0]).
    - Source element is critical. See notes regarding destination element for the read message above.



  - Configure the communication path as shown for the read message.
  - Click OK to close the Message Configuration dialog box.

The module and AOI are now configured and ready to use. The program can be downloaded to the controller and set to run mode if appropriate.

Using the AOI

Importing the AOI also introduced a new UDT (User-Defined Data Type) called TR_Linear. Refer to Appendix TR_Linear UDT for detailed information about the data types. The UDT and AOI work together to perform 3 main tasks:

1. Parse encoder feedback into an understandable and usable format. This parsing takes place every scan.
    a. Status: Position status of the encoder.
    b. Position1, Position2, Position3: The signed position in steps of each magnet.
        i. Any non-configured magnets (number of magnets equal to 1 or 2) will have position value 0.
    c. Velocity1, Velocity2, Velocity3: The signed velocity of each magnet.
        i. Any non-configured magnets (number of magnets equal to 1 or 2) will have position value 0.
2. Read important encoder parameters: Linear.Read.XXXXXXXX. **This is done so that a user can validate the settings read from the encoder against their intended configuration.**
    a. Automatic:
        i. Two seconds after a successful re-connection between PLC and encoder.
    b. On Demand: Linear.Read.Command
        i. Read begins upon positive transition of trigger bit. This bit will be automatically cleared when the read is successful or a read failure is detected.
    c. Outputs: Linear_AOI.Out_Read_Success
        i. Cleared when read initiated or connection between encoder and PLC is lost.
        ii. Set upon successful read.
        iii. Once the AOI has been setup and a successful read has occurred, there should be no read failures in the future (assuming the encoder is powered and connected). However if you want to monitor explicitly for read failures you may use Linear_Read_MSG.ER.
3. Perform a preset: Linear.Preset.Preset1, Preset2, Preset3.
    a. On Demand: Linear.Preset.Preset1Command, Preset2Command, Preset3Command.
        i. There is a different preset value and trigger for each magnet
        ii. Preset operation begins upon positive transition of a trigger bit, so it is recommended that this bit is set using a one-shot (ONS instruction). This bit will be automatically cleared when the preset operation is successful or an error is detected for that each magnet.
        iii. Once the AOI has been setup and a successful preset has occurred, there should be no preset failures in the future (assuming the encoder is powered and connected). However if you want to monitor explicitly for preset failures you may use Linear_Preset_MSG.ER.
    b. Outputs: Linear_AOI.Out_Preset1_Success, Out_Preset2_Success, Out_Preset3_Success
        i. Each bit is cleared on first scan or when preset operation is initiated for the each magnet.
        ii. Set upon successful preset operation for the each magnet.
        Note: Preset operation for magnet 2 and 3 can still be successful even if the number of magnets equals 1 or 2.

The AOI has an additional output called LINEAR_AOI.ConnOK. This bit is set to true when the PLC and encoder have been connected successfully for two seconds. It is cleared when a communication fault is detected.
AOI output usage is at the programmer discretion. They can be used for in logic, but do not need to be.

Appendix

TR_Linear UDT

Member Variable		Data Type	Usage	Description
Status		DINT (32 bits)	Read Only (Linear.Status)	IO Assembly from Device – Position Status
Position1		DINT (32 bits)	Read Only (Linear.Position1)	IO Assembly from Device – Magnet 1 Signed Position
Velocity1		DINT (32 bits)	Read Only (Linear.Velocity1)	IO Assembly from Device – Magnet 1 Signed Velocity
Position2		DINT (32 bits)	Read Only (Linear. Position2)	IO Assembly from Device – Magnet 2 Signed Position
Velocity2		DINT (32 bits)	Read Only (Linear. Velocit2)	IO Assembly from Device – Magnet 2 Signed Velocity
Position3		DINT (32 bits)	Read Only (Linear. Position3)	IO Assembly from Device – Magnet 3 Signed Position
Velocity3		DINT (32 bits)	Read Only (Linear. Velocity3)	IO Assembly from Device – Magnet 3 Signed Velocity
Read	PositionSensorType	SINT (8 bits)	Read Only (Linear.Read.PositionSensorType)	Read From Device - Attribute: 11
	Direction	BOOL (1 bit)	Read Only (Linear.Read.Direction)	Read From Device - Attribute: 12
	PositionFormat	INT (16 bits)	Read Only (Linear.Read.PositionFormat)	Read From Device - Attribute: 15
	RawResolution	DINT (32 bits)	Read Only (Linear.Read.RawResolution)	Read From Device - Attribute: 16
	PositionMeasuringIncrement	DINT (32 bits)	Read Only (Linear.Read.PositionMeasuringIncrement)	Read From Device - Attribute: 18
	VelocityFormat	DINT (32 bits)	Read Only (Linear.Read. VelocityFormat)	Read From Device - Attribute: 25
	VelocityObserver	SINT (8 bits)	Read Only (Linear.Read. VelocityObserver)	Read From Device - Attribute: 108
	PositionFilter	SINT (8 bits)	Read Only (Linear.Read. PositionFilter)	Read From Device - Attribute: 109
	NumberOfMagnets	SINT (8 bits)	Read Only (Linear.Read. NumberOfMagnets)	Read From Device - Attribute: 110
	Alarms	INT (16 bits)	Read Only (Linear.Read.Alarms)	Read From Device - Attribute: 44
	Warnings	INT (16 bits)	Read Only (Linear.Read.Warnings)	Read From Device - Attribute: 47
	EncoderFirmwareNumber	STRING	Read Only (Linear.Read.EncoderFirmwareNumber)	Read From Device - Attribute: 119
	ReadData	SINT[4] (4x8 bits)	N/A	Internal working variables - do not use.
	Command	BOOL (1 bit)	Read/Write (Linear.Read.Command)	Read Params Trigger
Preset	Preset1	DINT (32 bits)	Write Only (Linear.Preset.Preset1)	Write To Device - Attribute: 103, Signed Value
	Preset2	DINT (32 bits)	Write Only (LIN Linear EAR.Preset.Preset2)	Write To Device - Attribute: 104, Signed Value
	Preset3	DINT (32 bits)	Write Only (Linear.Preset.Preset3)	Write To Device - Attribute: 105, Signed Value
	PresetData	SINT[4] (4x8 bits)	N/A	Internal working variables - do not use.
	Preset1Command	BOOL (1 bit)	Read/Write (Linear.Preset. Preset1Command)	Magnet 1 Preset Operation Trigger
	Preset2Command	BOOL (1 bit)	Read/Write (Linear.Preset. Preset2Command)	Magnet 2 Preset Operation Trigger
	Preset3Command	BOOL (1 bit)	Read/Write (Linear.Preset. Preset3Command)	Magnet 3 Preset Operation Trigger

Note: Attribute numbers refer to Object 0x23 Position Sensor Object. Definitions can be found in the "Absolute Linear Encoders LA, LP, LMP (+MultiSensor) – User Manual".

Configuration Assembly

	Default (Hex)	Default (Decimal)	Default (Decimal)	Possibilities	Description
Linear:C.Data[0]	00	0	0	0 or 1	Direction Counting Toggle - Attribute: 12
Linear:C.Data[1]	04	4	8708	8706 (cm), 8707 (mm), 8708 (µm), 8709 (nm)	Position Format Attribute: 15
Linear:C.Data[2]	22	34			
Linear:C.Data[3]	01	1	1	1 – 1 000 000	Position Measuring Increment Attribute: 18
Linear:C.Data[4]	00	0			
Linear:C.Data[5]	00	0			
Linear:C.Data[6]	00	0			
Linear:C.Data[7]	01	1	11009	7940 (stp/s) – 11009 (cm/s)	Velocity Format Attribute: 25
Linear:C.Data[8]	2B	43			
Linear:C.Data[9]	01	1	1	1	Velocity Resolution Attribute: 26
Linear:C.Data[10]	00	0			
Linear:C.Data[11]	00	0			
Linear:C.Data[12]	00	0			
Linear:C.Data[13]	00	0	0	N/A	Reserved
Linear:C.Data[14]	00	0			
Linear:C.Data[15]	00	0	0	0, 1, 4, 7	Velocity Observer - Attribute: 108
Linear:C.Data[16]	01	1	1	1-16	Position Filter - Attribute: 109
Linear:C.Data[17]	01	1	1	1, 2, 3	Number of Magnets – Attribute: 110
Linear:C.Data[18]	00	0	0	N/A	Reserved
Linear:C.Data[19]	00	0			
Linear:C.Data[20]	00	0			
Linear:C.Data[21]	00	0			
Linear:C.Data[22]	00	0	0	N/A	Reserved
Linear:C.Data[23]	00	0			
Linear:C.Data[24]	00	0			
Linear:C.Data[25]	00	0			
Linear:C.Data[26]	00	0	0	N/A	Reserved
Linear:C.Data[27]	00	0			
Linear:C.Data[28]	00	0			
Linear:C.Data[29]	00	0			
Linear:C.Data[30]	00	0	0	N/A	Reserved
Linear:C.Data[31]	00	0			

NOTE: The offline PLC file should be updated with the new values to ensure old values aren't re-loaded upon re-download. This can be done by updating the offline file (recommended) or uploading tag values upon saving the program.

Configuration Assembly Mapping Example

	Desired Value (Decimal)	Desired Value (Hex)	Desired Value (Hex)	Desired Value (Signed Decimal)*
Linear:C.Data[0]	1	0	01	1
Linear:C.Data[1]	8709	22 05	05	5
Linear:C.Data[2]			22	34
Linear:C.Data[3]	5	05	05	5
Linear:C.Data[4]			00	0
Linear:C.Data[5]			00	0
Linear:C.Data[6]			00	0
Linear:C.Data[7]	7940	1F04	04	4
Linear:C.Data[8]			1F	31
Linear:C.Data[9]	Left as default - 1	Left as default - 01	01	1
Linear:C.Data[10]			00	0
Linear:C.Data[11]			00	0
Linear:C.Data[12]			00	0
Linear:C.Data[13]	0	00	00	0
Linear:C.Data[14]			00	0
Linear:C.Data[15]	4	04	04	4
Linear:C.Data[16]	5	05	05	5
Linear:C.Data[17]	3	03	03	3
Linear:C.Data[18]	0	00	00	0
Linear:C.Data[19]			00	0
Linear:C.Data[20]			00	0
Linear:C.Data[21]			00	0
Linear:C.Data[22]	0	00	00	0
Linear:C.Data[23]			00	0
Linear:C.Data[24]			00	0
Linear:C.Data[25]			00	0
Linear:C.Data[26]	0	00	00	0
Linear:C.Data[27]			00	0
Linear:C.Data[28]			00	0
Linear:C.Data[29]			00	0
Linear:C.Data[30]	0	00	00	0
Linear:C.Data[31]			00	0

***Note:** Conversion from Hex bytes to decimal bytes is not required if the style was changed to Hex, as recommended.

Note: In cases where bit 7 of a byte is high, the 2's complement must be taken. See Appendix **Fehler! Verweisquelle konnte nicht gefunden werden.** for more information.

Resolutions and Distance Sample Calculations

Two attributes (16 and 18) are used to define the resolution of the Linear and are configurable from the configuration assembly. In short, attribute 15 defines the unit of the position value and resolution value, and attribute 18 sets the resolution in those units. Then attribute 16 (read-only) calculates the raw resolution that has been configured, in nanometers.

Further explanation is below:

Attribute	Access	Description
15 – Position Format	Get/Set	Unit for Position Output and Attribute 18
16 – Raw Resolution	Get	Resolution in nm
18 – Position Measuring Increment	Get/Set	Resolution in Units from Position Format

Note: Get Access refers to the Linear UDT, Linear.Read.XXXXXXXX

Possible position formats (from User Manual) are:

Attribute Value	Unit
0x2202	[cm]
0x2203	[mm]
0x2204	[µm]
0x2205	[nm]

The resolution calculation is as follows:

$$Position\ Measuring\ Increment[PositionFormatUnit] = Resolution$$

Some examples can be found below:

Attribute 18	Attribute 15		Scaled Resolution		Raw Resolution		Attribute 16
1	0x2204	→	1 µm	→	1 000 nm	→	1000
2	0x2202	→	2 cm	→	20 000 000 nm	→	20 000 000
1	0x2205	→	1 nm	→	1 nm	→	1
5	0x2205	→	5 nm	→	5 nm	→	5
3	0x2203	→	3 mm	→	3 000 000 nm	→	3 000 000

Note: The first example uses the default values.

Position value is reported from the device in steps. This value is read each scan and stored under Linear.Position1 (or 2 or 3). To calculate magnet distance from steps to distance in the Position Format Units or nanometers, a simple multiplication is required:

$$Position\ [PositionFormatUnit] = Linear.Position1\ x\ ScaledResolution$$

OR

$$Position\ [nm] = Linear.Position1\ x\ RawResolution$$

Unsigned Vs Signed

RSLogix uses signed integers (SINTs, DINTs) which can catch some users off guard. This is general RSLogix behaviour and can't be influenced by an AOI, so this appendix has been included to help the user's understanding.

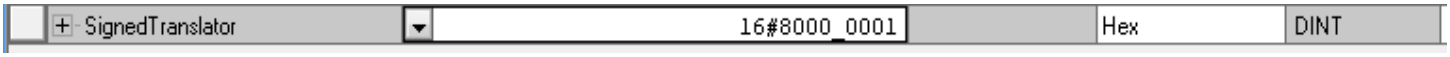
When the most significant bit of the integer is high, RSLogix will interpret and display this number as a negative. Using SINT as an example:

Binary	RSLogix Value
00000001	1
00000011	3
01111111	127
10000000	-128 (2's complement of 128)
10000001	-127 (2's complement of 129)
10000011	-125 (2's complement of 131)
11111111	-1 (2's complement of 255)

Because the liner encoder returns a signed DINT position value, this is not a concern for the position. However, it does apply to other SINTs, INTs, and DINTs,

An easy was to calculate the 2's complement of a number in RSLogix using an available tag. Below is a DINT example, but the same method applies for a SINT:
Translate the number (2, 147, 483, 649) to Hex using a calculator.

- Change the Style of a DINT tag in RSLogix to Hex.
- Enter the number into the value column of that tag, in Hex (80 00 00 01).



- Then change the Style back to Decimal to reveal the 2's complement.

Logic Examples