ABSOLUTE IXARC ROTARY ENCODER WITH ETHERNET/IP INTERFACE

USER MANUAL

ABSOLUTE IXARC ROTARY ENCODER WITH ETHERNET/IP INTERFACE

USER MANUAL

**Imprint**
POSITAL GmbH
Zeppelinstraße 2,
D-50667 Köln
Telefon +49 (0) 221 96213-0
Fax +49 (0) 221 96213-20
Internet http://www.posital.com
E-Mail info@posital.eu



**Alteration of Specifications reserved**
Technical specifications, which are described in this manual, are subject to change due to our permanent strive to improve our products.

**Disclaimer of Warranty**
POSITAL GmbH makes no representations or warranties, either express or implied, by or with respect to anything in this manual. Furthermore, POSITAL shall not be liable for any implied warranties of merchantability and fitness for a particular purpose or for any indirect, special, or consequential damages.

**Document information**
File name: UME-xCx-EEC
Date: Version 2020-12-02
Version number: 1.2
Author: HBA

**Phone Service**
For technical support, questions, and suggestions for improving our products and documentations call our telephone line +49 (0) 221-96213-0.

ABSOLUTE IXARC ROTARY ENCODER WITH ETHERNET/IP INTERFACE

USER MANUAL


***IXARC Absolute ROTARY ENCODER WITH ETHERNET/IP Interface*** *Error! Bookmark not defined.*

**1. Introduction 5**
1.1 Control and Information Protocol (CIP) 5
1.2 Object model 6

**2. Data Transmission 6**

**2.1 Implicit Messaging I/O Connection 7**
2.1.1 I/O Assembly Instances 8
2.1.1.1 Data Attribute Format 8
2.1.2 Data Mapping 9
2.1.3 Data Mapping (Parameter) 10
2.1.3.1 Data Offset 10
2.1.4 Connection Path 11

**2.2 Explicit Messaging 12**
2.2.1 CIP Common Services for Position Sensor Objects (Class 0x23hex) 12
2.2.2 Position Sensor Objects 13

**2.3 TCP/IP Interface Object 14**
2.3.1 Status Instance Attribute ($01_{hex}$) 15
2.3.2 Configuration Instance Attribute ($02_{hex}$) 16
2.3.3 Configuration Control Inst. Attribute ($04_{hex}$) 16
2.3.4 Physical Link Object ($05_{hex}$) 17
2.3.5 Interface Configuration ($06_{hex}$) 17
2.3.6 Host Name 17

**2.4 Ethernet Link Object 18**
2.4.0 Instance Attributes 19
Control Bits 22
Example 22
Use on Transmit data size double (4 bytes) 00000064 for Auto-negotiation = disable on 100 MBaud 22
2.4.1 Interface Flags 23
2.4.2 Common Services 23
2.4.3 Link Object Instances 24

**2.5 Setting parameters with scanners 24**
2.5.1 Read out position value 25
2.5.2 Set preset value 25
2.5.3 Get preset value 26

**3 Diagnostics 27**

**4 Programmable Parameters 29**
4.1 Direction Counting 29
4.2 Scaling Function Control 29
4.3 Resolution per Revolution 29
4.4 Total Resolution 30
4.5 Preset Value 30
4.6 Velocity Format 31
4.7 Velocity Filter 31
4.8 Endless Shaft 31
4.9 Parameter Control Priority 32
4.10 Available Alarms and Warnings 32
4.11 Device Level Ring 33

**5. Installation 34**
5.1 Electrical connection 34
5.2 Ethernet cables 34

**6 Power On 35**

**7 Installation 35**

**7.1 Rockwell configuration tools 35**
7.1.1 Setting IP-Address (BOOTP) 35
7.1.2 Configuration RSLinx Classic™ 38
7.1.3 RSNetWorx™ 40
7.1.4 Configuration RSLogix 5000 42
7.1.4.1. Generic Module Configuration 51

**7.2 VLAN Configuration over a Layer 3 Switch 51**

**8 FAQ 53**

**9 Glossary 55**

**10 Revision index 56**

**List of Tables**

Table 1: Communication Messages 7
Table 2: I/O Assembly Instances 8
Table 3: Data Attribute Format 9
Table 4: Data Mapping 9
Table 5: Assembly Instance Configuration: 7, size 14 Bytes 10
Table 6: Data Offset 10
Table 7: Connection Path 11
Table 8: CIP Position Sensor Objects – Save / Restore 12
Table 9: CIP Position Sensor Objects – Instance Attributes 13
Table 10: TCP/IP Interface Objects Overview 14
Table 11: TCP/IP Interface Objects – Attribute 01hex 15
Table 12: TCP/IP Interface Objects – Attribute 02hex 16
Table 13: TCP/IP Interface Objects – Attribute 04hex 16
Table 14: TCP/IP Interface Objects – Attribute 05hex 17
Table 15: TCP/IP Interface Objects – Attribute 06hex 17
Table 16: TCP/IP Interface Objects – Host Name 17
Table 17: Ethernet Link Objects Overview 18
Table 18: Ethernet Link Objects – Instance Attributes 19
Table 19: Ethernet Link Objects – Control Bits 22
Table 20: Ethernet Link Objects – Interface Flags 23
Table 21: Ethernet Link Objects – Common Services 23
Table 22: Link Object Instances 24
Table 23: Module Status Indicator Stat1/Stat2 27
Table 24: Network Status Indicator Stat2 28
Table 25: Direction Counting 29
Table 26: Direction Counting Bits 29
Table 27: Scaling Function Control 29
Table 28: Scaling Function Control Bits 29
Table 29: Resolution per Revolution 29
Table 30: Total Resolution 30
Table 31: Preset Value 30
Table 32: Velocity Format 31
Table 33: Velocity Filter 31
Table 34: Endless Shaft 32
Table 35: Reserved **Error! Bookmark not defined.**32
Table 36: Parameter Control Priority 32
Table 37: Alarms 33
Table 38: Warnings 33
Table 39: Data Pin Assignment 34
Table 40: Power Supply Pin Assignment 34
Table 41: Ethernet cable RJ45-M12 crossed 34
Table 42: Ethernet cable RJ45-M12 straight 34
Table 43: Ethernet cable M12-M12 crossed 34
Table 44: Rotary Switches Allowed IP Addresses 37
Table 45: Example of VLAN IP Addresses Allocation 52
Table 46: Glossary 56
Table 47: Revision Index 56

**List of Figures**

Figure 1: Setup of an Ethernet Data Package on Layer 2 .....5
Figure 2: CIP OSI Layers .....5
Figure 3: Ethernet/IP Object Model .....6
Figure 4: Data Transmission in Ethernet/IP.....6
Figure 5: Generic Module Communication Parameters .....7
Figure 6: Diagnostic LEDs.....27
Figure 7: Endless Shaft.....32
Figure 8: Device level Ring .....33
Figure 9: Ethernet Connector.....34
Figure 10: Power Supply Connector .....34
Figure 11: IP Address Rotary Switches.....37
Figure 12: VLAN Setup .....52

## 1. Introduction

Absolute rotary encoders provide a definite value for every possible position. All these values are reflected on either code discs, or Hall-effect sensors, depending on the used technology. In the case of an optical encoder, the beams of infrared LEDs are sent through code discs and detected by Opto-Arrays. For a magnetic one, a rotating magnet attached to the shaft creates a field whose direction is measured by the Hall-effect sensor. The output signals are electronically amplified, and the resulting value is transferred to the interface.

The absolute rotary encoder has a maximum resolution of 65,536 steps per revolution (16 bits). The multiturn version can detect up to 16,384 revolutions (14 bits). Therefore, the largest resulting resolution is 30 bits = 1,073,741,824 steps. The standard singleturn version is 13 bits, the standard multiturn version 25 bits.

The integrated Ethernet interface of the absolute rotary encoder supports all necessary EtherNet/IP functions.

The protocol supports the programming of the following additional functions in several ways:

1. Code sequence (complement)
2. Resolution per revolution
3. Total resolution
4. Preset value
5. IP-Address
6. DLR

The functioning of the absolute rotary encoders under general aspects of the EtherNet/IP interface is guaranteed. The data will be transmitted in a standard Ethernet frame in the data section (see at the bottom of this side the pink field with the blue frame).

The MAC address of each encoder is available on the type of label.

The IP address can be programmed with BOOTP via the configuration tools of the PLC.

The physical interface supports auto negotiation and auto crossing.

General information's about EtherNet/IP are available at:
www.ethernetip.de (German)
https://www.odva.org/technology-standards/key-technologies/ethernet-ip/ (English)

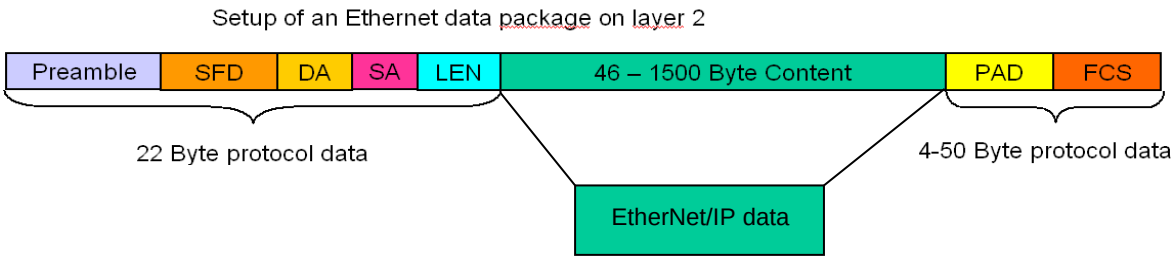



*Figure 1: Setup of an Ethernet Data Package on Layer 2*

### 1.1 Control and Information Protocol (CIP)

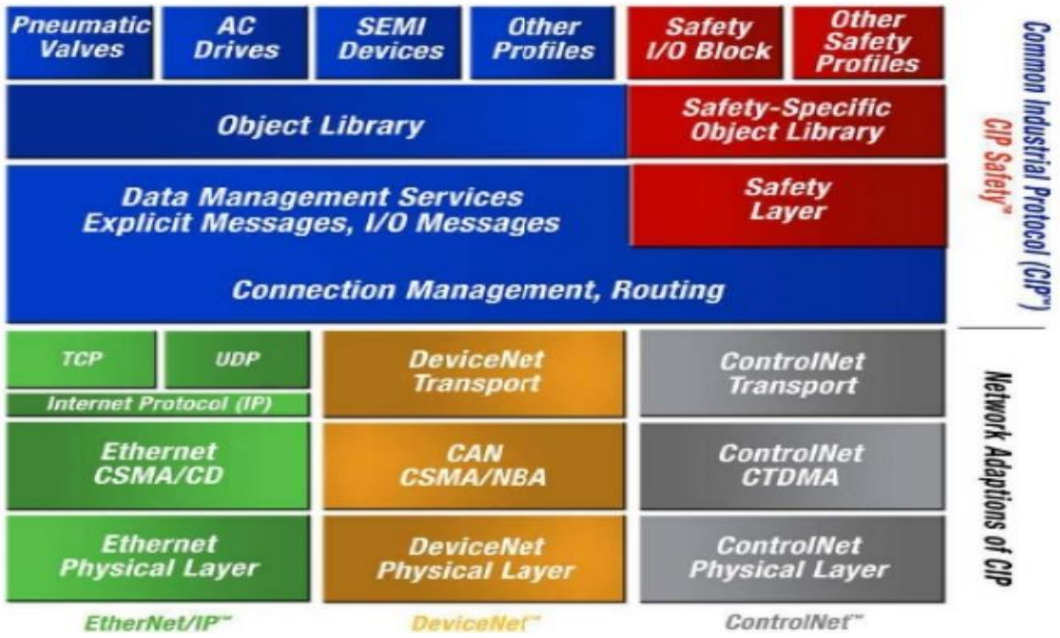



*Figure 2: CIP OSI Layers*

The EtherNet/IP specification defines the Application Layer and the Physical Layer. The Data Link layer is based on the CAN-specification. For the optimal industrial control will be defined two different messaging types. I/O messaging (Implicit Messaging) and explicit messaging. With Implicit Messaging, I/O data are exchanged in real time whereas Explicit Messaging is mostly used for device configuration.

CIP (Common Industrial Protocol) make for the user available four essential functions:

- Unique control service
- Unique communication service
- Unique allocation of messaging
- Common knowledge base

### 1.2 Object model

EtherNet/IP describes all data and functions of a device considering an object model. By means of that object-oriented description, a device can be completely defined with single objects. An object is defined across the centralization by associated attributes (e.g. process data), its functions (read or write access of a single attribute) as well as by its defined behaviors. The absolute rotary encoder supports the Encoder Device Type: $22_{hex}$. This is programmable, see chapter 4.1. All parameters will be used with Big Endian notation.

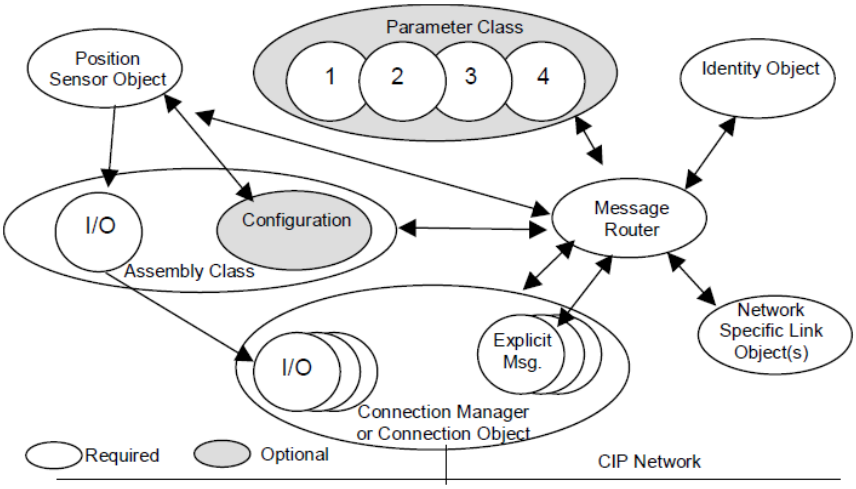



*Figure 3: Ethernet/IP Object Model*

## 2. Data Transmission

The data transmission within the EtherNet/IP network is realized by implicit or explicit messaging. Explicit messages are split in unconnected and connection-based versions. Unconnected messages will be used by EtherNet/IP scanners.

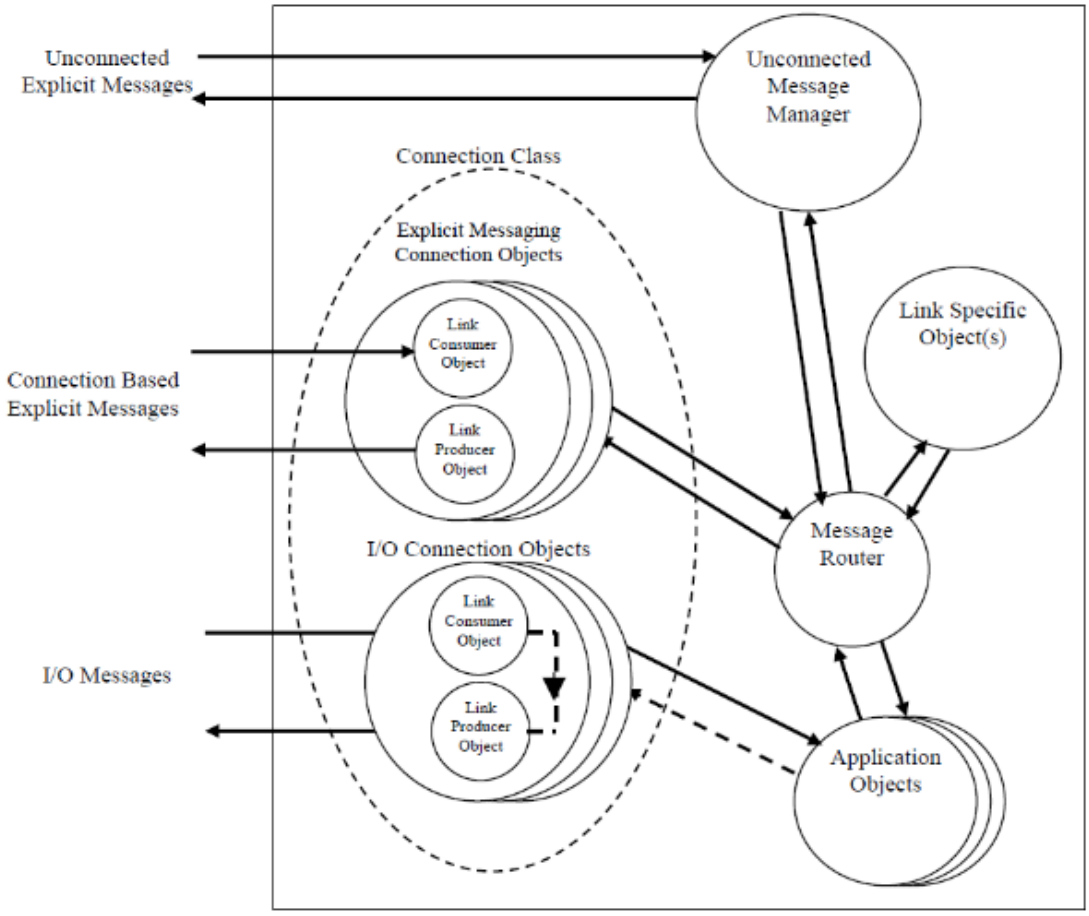



*Figure 4: Data Transmission in Ethernet/IP*

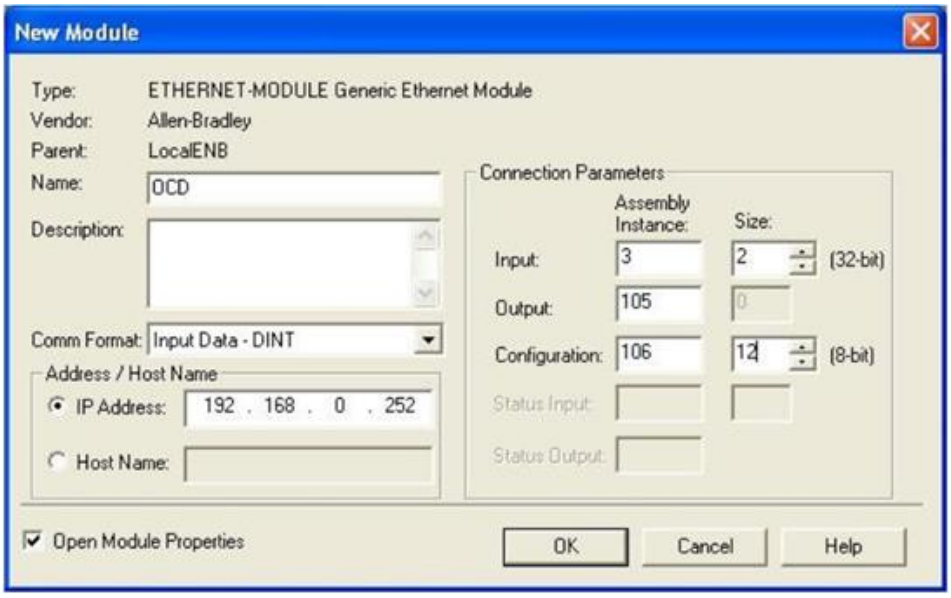
**Input Only**

It is possible to open 5 implicit and 15 explicit connections with the encoder. Input Only connections can only work if all of the parameters are set in the PLC accordingly to the encoder parameters.

| Assembly | Config | Output Instance | Input Instance |
|---|---|---|---|
| Connection Manager | Config | Connection Point 1 | Connection Point 2 |
| Input Only | $0x7A_{hex}$ (122) | $0x69_{hex}$ (105) | 0x01 Position value |
| | | | 0x03 Position value + velocity |

*Table 1: Communication Messages*

*Figure 5: Generic Module Communication Parameters*



## 2.1 Implicit Messaging I/O Connection

Provide dedicated, special-purpose communication paths between a producing application and one or more consuming applications for the purpose of moving application-specific data. This is often referred to as implicit messaging. Class 0 and 1 are supported.

### 2.1.1 I/O Assembly Instances

| Instance | Type | Name |
|---|---|---|
| 1 | Input | Position Value 32 bits unscaled |
| 2 | Input | Position Value 32 bits scaled and Warning Flag 8 bits |
| 3 | Input | Position Value 32 bits and Velocity 32 bits |
| 120 | Input | Position Value 32 bits, Velocity Value 32 bits, Acceleration Value 32 bits and Warning Flag 8 bits |
| 121 | Input | Position Value 32 bits, Velocity Value 32 bits, Acceleration Value 32 bits, Temperature Value 32 bits and Warning Flag 8 bits |
| 122 | Output | Direction Counting Toggle 8 bits, Scaling Function Control 8 bits, Measuring Units per Revolution 32 bits, Total Measuring Range 32 bits, Velocity Format 16 bits, Velocity Filter 8 bits, Parameter Control Priority 8 bits |

*Table 2: I/O Assembly Instances*

#### 2.1.1.1 Data Attribute Format

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | Position Value (low Byte) | | | | | | | |
| | 1 | | | | | | | | |
| | 2 | | | | | | | | |
| | 3 | Position Value (high Byte) | | | | | | | |
| 2 | 0 | Position Value (low Byte) | | | | | | | |
| | 1 | | | | | | | | |
| | 2 | | | | | | | | |
| | 3 | Position Value (high Byte) | | | | | | | |
| | 4 | | | | | | | Warning | Alarm |
| 3 | 0 | Position Value (low Byte) | | | | | | | |
| | 1 | | | | | | | | |
| | 2 | | | | | | | | |
| | 3 | Position Value (high Byte) | | | | | | | |
| | 4 | Velocity Value (low Byte) | | | | | | | |
| | 5 | | | | | | | | |
| | 6 | | | | | | | | |
| | 7 | Velocity Value (high Byte) | | | | | | | |

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 120 | 0 | Position Value (low Byte) | | | | | | | |
| | 1 | | | | | | | | |
| | 2 | | | | | | | | |
| | 3 | Position Value (high Byte) | | | | | | | |
| | 4 | Velocity Value (low Byte) | | | | | | | |
| | 5 | | | | | | | | |
| | 6 | | | | | | | | |
| | 7 | Velocity Value (high Byte) | | | | | | | |
| | 8 | Acceleration Value (low Byte) | | | | | | | |
| | 9 | | | | | | | | |
| | 10 | | | | | | | | |
| | 11 | Acceleration Value (high Byte) | | | | | | | |
| | 12 | | | | | | | Warning | Alarm |
| 121 | 0 | Position Value (low Byte) | | | | | | | |
| | 1 | | | | | | | | |
| | 2 | | | | | | | | |
| | 3 | Position Value (high Byte) | | | | | | | |
| | 4 | Velocity Value (low Byte) | | | | | | | |
| | 5 | | | | | | | | |
| | 6 | | | | | | | | |
| | 7 | Velocity Value (high Byte) | | | | | | | |
| | 8 | Acceleration Value (low Byte) | | | | | | | |
| | 9 | | | | | | | | |
| | 10 | | | | | | | | |
| | 11 | Acceleration Value (high Byte) | | | | | | | |
| | 12 | Temperature Value (low Byte) | | | | | | | |
| | 13 | | | | | | | | |
| | 14 | | | | | | | | |
| | 15 | Temperature Value (high Byte) | | | | | | | |
| | 16 | | | | | | | Warning | Alarm |

*Table 3: Data Attribute Format*

**2.1.2 Data Mapping**

| Data Component Name | Class | | Instance Number | Attribute | |
|---|---|---|---|---|---|
| | Name | Number | | Name | Number |
| Position Value Unscaled | Position Sensor | $23_{hex}$ | 1 | Position Value unscaled | $03_{hex}$ |
| Velocity | Position Sensor | $23_{hex}$ | 1 | Velocity | $18_{hex}$ |
| Acceleration | Position Sensor | $23_{hex}$ | 1 | Acceleration | $1D_{hex}$ |
| Warning flag | Position Sensor | $23_{hex}$ | 1 | Warning flag | $7C_{hex}$ |
| Temperature Value | Position Sensor | $23_{hex}$ | 1 | Temperature Value flag | $7D_{hex}$ |

*Table 4: Data Mapping*

### 2.1.3 Data Mapping (Parameter)

On every Forward Open Request, the following parameters will be sent from the controller to the encoder.

| Configuration Parameter Name | Class | | Instance Number | Attribute | |
|---|---|---|---|---|---|
| | Name | Number | | Name | Number |
| Direct Counting Toggle | Position Sensor | $23_{hex}$ | 1 | Direct Counting Toggle | $0C_{hex}$ |
| Scaling Function Control | Position Sensor | $23_{hex}$ | 1 | Scaling Function Control | $0E_{hex}$ |
| Measuring units per Revolution | Position Sensor | $23_{hex}$ | 1 | Measuring Units per Span | $10_{hex}$ |
| Total Measuring Range in measuring units | Position Sensor | $23_{hex}$ | 1 | Total Measuring Range in measuring units | $11_{hex}$ |
| Velocity Format | Position Sensor | $23_{hex}$ | 1 | Velocity Format | $19_{hex}$ |
| Velocity Filter | Position Sensor | $23_{hex}$ | 1 | Velocity Filter | $70_{hex}$ |
| Parameter Control Priority | Position Sensor | $23_{hex}$ | 1 | Parameter Control Priority | $71_{hex}$ |

*Table 5: Assembly Instance Configuration: 7, size 14 Bytes*

#### 2.1.3.1 Data Offset

| Instance | Byte Offset | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 122 | 0 | Direction Counting Toggle | | | | | | | |
| | 1 | Scaling Function Control | | | | | | | |
| | 2 | Measuring Units per Revolution (low Byte) | | | | | | | |
| | 3 | | | | | | | | |
| | 4 | | | | | | | | |
| | 5 | Measuring Units per Revolution (high Byte) | | | | | | | |
| | 6 | Total Measuring Range in Measuring Units (low Byte) | | | | | | | |
| | 7 | | | | | | | | |
| | 8 | | | | | | | | |
| | 9 | Total Measuring Range in Measuring Units (high Byte) | | | | | | | |
| | 10 | Velocity Format (low Byte) | | | | | | | |
| | 11 | Velocity Format (high Byte) | | | | | | | |
| | 12 | Velocity Filter | | | | | | | |
| | 13 | Parameter Control Priority | | | | | | | |

*Table 6: Data Offset*

### 2.1.4 Connection Path

Is made of a byte stream that defines the application object to which a connection instance applies. This path will be created in the configuration tools and is available in the EDS file too.

This path will be sent to the encoder during power up. For some tools it is necessary to use the connection path as a parameter:

[20] [04] [24 7A] [24 69] [24 01] [80 07 00 01 00100000 00200000 1F04 01 00]

| Segment Groups | Segment | Description |
|---|---|---|
| Application Path | 20 04 | Assembly object class |
| | 24 7A | Instance segment type with Assembly Instance $0x7A_{hex}$ (122) (Configuration) |
| | 24 69 | Assembly Instance $0x69_{hex}$ (105) (Output controller to encoder) |
| | 24 01 | I/O Assembly Instance 1 (Position value) |
| | 80 07 | Data segment with lenght of 6 Bytes |
| | 00 01 00100000 00200000 1F04 01 00 | Configuration Data, see chapter 2.1.3.1 for details |

*Table 7: Connection Path*

## 2.2 Explicit Messaging

Provide generic, multi-purpose communication paths between two devices. These connections often are referred to as just Messaging Connections.

Explicit Messages provide the typical request/response-oriented network communications. Class 2 and 3 are supported.

### 2.2.1 CIP Common Services for Position Sensor Objects (Class 0x23hex)

Save / Restore

| Supported Service Code | Service Name | Comment |
|---|---|---|
| $05_{hex}$ | Reset | Boot up of the encoder, the programmed parameter from the customer will use again |
| $0E_{hex}$ | Get_Attribute_Single | Read out attribute from the encoder |
| $10_{hex}$ | Set_Attribute_Single | Write attribute to the encoder |
| $15_{hex}$ | Restore | Restore the saved parameters. Use instance 0 of position sensor class to restore all configuration parameters at once. To restore a single parameter, use instance 1 of position sensor class with attribute number as argument (see next table). |
| $16_{hex}$ | Save | Save the parameters from chapter 2.1.3 in the nonvolatile memory. Use instance 0 of position sensor class to save all configuration parameters at once. |

*Table 8: CIP Position Sensor Objects – Save / Restore*

### 2.2.2 Position Sensor Objects

Instance Attributes (Get: read, Set: write + read) // Class Code: $23_{hex}$

| Attrib. ID | Access | Name | Data Type | Description |
|---|---|---|---|---|
| $0A_{hex}$ | Get | Position Value Unsigned | DINT | Unscaled position (32 bits) |
| $0C_{hex}$ | Set | Direction Counting Toggle | Boolean | Controls the code sequence clockwise or counterclockwise |
| $0E_{hex}$ | Set | Scaling Function Control | Boolean | Scaling function on/off |
| $10_{hex}$ | Set | Measuring units per Span | UDINT | Resolution for one revolution |
| $11_{hex}$ | Set | Total Measuring Range in Measuring Units | UDINT | Total resolution |
| $18_{hex}$ | Get | Velocity Value | DINT | Current speed in format of attribute $19_{hex}$ and $2A_{hex}$ |
| $19_{hex}$ | Set | Velocity Format | ENGUINT | Format of the velocity attributes |
| $1D_{hex}$ | Get | Acceleration Value | DINT | Current acceleration in rad/s$^2$ |
| $70_{hex}$ | Set | Velocity Filter | DINT | Fine = 0, Normal = 1, Coarse = 2 |
| $71_{hex}$ | Set | Parameter Control Priority | USINT | Choice between PLC or NVM configuration |
| $7C_{hex}$ | Get | Warning Flags | WORD | Internal parameters exceeded |
| $7D_{hex}$ | Get | Temperature Value | DINT | Value of the encoder's temperature un °C |

*Table 9: CIP Position Sensor Objects – Instance Attributes*

## 2.3 TCP/IP Interface Object

The TCP/IP Interface Object provides the mechanism to configure a device's TCP/IP network interface. With these parameters it is possible for example to read or write the device's IP Address and Network Mask.

Class Code: $F5_{hex}$

| Attribute ID | Access | Name | Data Type | Description |
|---|---|---|---|---|
| $01_{hex}$ | Get | Status | DWORD | Interface status, details in chapter 2.3.1 |
| $02_{hex}$ | Get | Configuration Capability | DWORD | Interface capability flags, details in chapter 2.3.2 |
| $03_{hex}$ | Set | Configuration Control | DWORD | Interface control flags, details in chapter 2.3.3 |
| $04_{hex}$ | Get | Physical Link Object | STRUCT of: | Path to physical link object |
| | | Path size | UINT | Size of path |
| | | Path | Padded EPATH | Logical segments identifying the physical link object |
| $05_{hex}$ | Set | Interface Configuration | STRUCT of: | TCP/IP network interface configuration |
| | | IP Address | UDINT | The device's IP address |
| | | Network Mask | UDINT | The device's network mask |
| | | Gateway | UDINT | Default gateway address |
| $06_{hex}$ | Set | Host Name | STRING | |
| $10_{hex}$ | Set | Select ACD (ACSI) | BOOLEAN | Not Supported |
| $11_{hex}$ | Set | Last Conflict Detected (ACSI) | STRUCT of: | Not Supported |
| | | Add Activity | USINT | |
| | | Remote MAC | Array of 6 USINT | |
| | | ArpPdu | Array of 28 USINT | |
| $13_{hex}$ | Set | Encapsulation Inactivity Timeout | UINT | Number of seconds of inactivity before connection is closed |

*Table 10: TCP/IP Interface Objects Overview*

### 2.3.1 Status Instance Attribute ($01_{hex}$)

| Bit(s) | Called | Definition | |
|---|---|---|---|
| 0-3 | Interface Configuration Status | Indicates the status of the Interface Configuration attribute. | 0 = The Interface Configuration attribute has not been configured.<br>1 = The Interface Configuration attribute contains valid configuration obtained from BOOTP or nonvolatile storage.<br>2 = The Interface Configuration attribute contains valid configuration, obtained from hardware settings (e.g.: push wheel, thumbwheel, etc.)<br>3-15 = Reserved for future use. |
| 4 | Mcast Pending | Indicates a pending configuration change in the TTL Value and/or Mcast Config attributes. This bit shall be set when either the TTL Value or Mcast Config attribute is set and shall be cleared the next time the device starts. | |
| 5 | Interface Configuration Pending | Indicates a pending configuration change in the Interface Configuration attribute. This bit shall be 1 (TRUE) when Interface Configuration attribute are set and the device requires a reset in order for the configuration change to take effect (as indicated in the Configuration Capability attribute). The intent of the Interface Config Pending bit is to allow client software to detect that a device's IP configuration has changed, but will not take effect until the device is reset. | |
| 6-31 | Reserved | Reserved for future use and shall be set to zero. | |

*Table 11: TCP/IP Interface Objects – Attribute 01hex*

### 2.3.2 Configuration Instance Attribute ($02_{hex}$)

| Bit(s) | Called | Definition |
|---|---|---|
| 0 | BOOTP Client | 1 (TRUE) indicates that the device is capable of obtaining its network configuration via BOOTP. |
| 1 | DNS Client | Not supported |
| 2 | DHCP Client | Not supported |
| 3 | DHCP – DNS Update | Not Supported |
| 4 | Configuration Settable | 1 (TRUE) indicates that the Interface Configuration attribute is settable. Some devices, for example a PC or workstation, may not allow the Interface Configuration to be set via the TCP/IP Interface Object. |
| 5 | Hardware Configurable | 1 (TRUE) shall indicate the IP Address member of the Interface Configuration attribute can be obtained from hardware settings (e.g., pushwheel, thumbwheel, etc.). If this bit is FALSE the Status Instance Attribute (1), Interface Configuration Status field value shall never be 2 (The Interface Configuration attribute contains valid configuration, obtained from hardware settings). |
| 6 | Interface Configuration Change Requires Reset | 1 (TRUE) shall indicate that the device requires a restart in order for a change to the Interface Configuration attribute to take effect. If this bit is FALSE a change in the Interface Configuration attribute will take effect immediately. |
| 7-31 | Reserved | Reserved for future use and shall be set to zero. |

*Table 12: TCP/IP Interface Objects – Attribute 02hex*

### 2.3.3 Configuration Control Inst. Attribute ($04_{hex}$)

| Bit(s) | Called | Definition | |
|---|---|---|---|
| 0-3 | Startup Configuration | Determines how the device shall obtain its initial configuration at start up. | 0 = The device shall use the interface configuration values previously stored (for example, in non-volatile memory or via hardware switches, etc).<br>1 = The device shall obtain its interface configuration values via BOOTP.<br>2 = DHCP (Not supported)<br>3-15 = Reserved for future use. |

*Table 13: TCP/IP Interface Objects – Attribute 04hex*

### 2.3.4 Physical Link Object ($05_{hex}$)

This attribute identifies the object associated with the underlying physical communication interface (e.g., an 802.3 interface). There are two components to the attribute: a path size (in UINTs) and a path. The path contains a logical segment type class, and a logical segment type instance that identifies the physical link object. The maximum path size is 6 (in the case of a 32-bit logical segment for each of the class and instance).

The physical link object itself typically maintains link-specific counters as well as any link specific configuration attributes. If the CIP port associated with the TCP/IP Interface Object has an Ethernet physical layer, this attribute shall point to an instance of the Ethernet Link Object (class code = $F6_{hex}$). When there are multiple physical interfaces that correspond to the TCP/IP interface, this attribute shall either contain a Path Size of 0 or contain a path to the object representing an internal communications interface (often used in the case of an embedded switch).

For example, the path could be as follows:

| Path | Meaning |
|---|---|
| 0-3 | [20] = 8-bit class segment type; [F6] = Ethernet Link Object class.<br>[24] = 8-bit instance segment type; [01] = instance 1. |

*Table 14: TCP/IP Interface Objects – Attribute 05hex*

### 2.3.5 Interface Configuration ($06_{hex}$)

| Name | Meaning |
|---|---|
| IP Address | The IP address of the device. Value of 0 indicates that no IP address has been configured. Otherwise, the IP address shall be set to a valid Class A, B, or C address and shall not be set to the loopback address (127.0.0.1). |
| Network mask | The network mask of the device. The network mask is used when the IP network has been partitioned into subnets. The network mask is used to determine whether an IP address is located on another subnet. Value of 0 indicates no network mask address has been configured. |

*Table 15: TCP/IP Interface Objects – Attribute 06hex*

### 2.3.6 Host Name

| Name | Meaning |
|---|---|
| Host Name | ASCII characters. Maximum length is 64 characters. Shall be padded to an even number of characters (pad not included in length). A length of 0 shall indicate no Host Name is configured. |

*Table 16: TCP/IP Interface Objects – Host Name*

## 2.4 Ethernet Link Object

Class Code: $F6_{hex}$

| Attribute ID | Access | Name | Data Type | Description | Semantics of Values |
|---|---|---|---|---|---|
| $01_{hex}$ | Get | Revision | UINT | Revision of this object | The minimum value is 1. Shall be 2 or greater if instance attribute 6 is implemented. Shall be 3 if any instance attributes 7-10 are implemented. The maximum value is 3. |
| $02_{hex}$ | Get | Max Instance | UINT | Maximum instance number of an object currently created in this class level of the device | The largest instance number of a created object at this class hierarchy level |
| $03_{hex}$ | Get | Number of Instances | UINT | Number of object instances currently created at this class level of the device | The number of object instances at this class hierarchy level |

*Table 17: Ethernet Link Objects Overview*

**2.4.0 Instance Attributes**

| ID | Access | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|
| 1 | Get | Interface Speed | UINT | Interface speed currently in use | Speed in Mbps (e.g., 10, 100 |
| 2 | Get | Interface Flags | DWORD | Interface status flags | See chapter 2.4.1 |
| 3 | Get | Physical Address | ARRAY of 6 USINTs | MAC layer address | Displayed format “XX-XX-XX-XX-XX-XX” |
| 4 | Get | Interface Counters | STRUCT of: | Contains counters relevant to the receipt of packets in the interface. | |
| | | In Octets | UDINT | Octets received on the interface | |
| | | In Ucast Packets | UDINT | Unicast packets received on the interface | |
| | | In NUcast Packets | UDINT | Non-unicast packets received on the interface | |
| | | In Discards | UDINT | Inbound packets received on the interface but discarded | |
| | | In Errors | UDINT | Inbound packets that contain errors (does not include In Discards) | |
| | | In Unknown | UDINT | Inbound packets with unknown protocol | |
| | | Out Octets | UDINT | Octets sent on the interface | |
| | | Out Ucast Packets | UDINT | Unicast packets sent on the interface | |
| | | Out Uncast Packets | UDINT | Non-unicast packets sent on the interface | |
| | | Out Discards | UDINT | Outbound packets discarded | |
| | | Out Errors | UDINT | Outbound packets that contain errors | |

*Table 18: Ethernet Link Objects – Instance Attributes*

| ID | Access | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|
| 5 | Get | Media Counters | STRUCT of: | Media-specific counters | |
| | | Alignment Errors | UDINT | Frames received that are not an integral number of octets in length | |
| | | FCS Errors | UDINT | Frames received that do not pass the FCS check | |
| | | Single Collisions | UDINT | Successfully transmitted frames which experienced exactly one collision | |
| | | Multiple Collisions | UDINT | Successfully transmitted frames which experienced more than one collision | |
| | | SQE Test Errors | UDINT | Number of times SQE test error message is generated | |
| | | Deferred Transmissions | UDINT | Frames for which first transmission attempt is delayed because the medium is busy | |
| | | Late Collisions | UDINT | Number of times a collision is detected later than 512 bit-times into the transmission of a packet | |
| | | Excessive Collisions | UDINT | Frames for which transmission fails due to excessive collisions | |
| | | MAC Transmit Errors | UDINT | Frames for which transmission fails due to an internal MAC sub layer transmit error | |
| | | Carrier Sense Errors | UDINT | Times that the carrier sense condition was lost or never asserted when attempting to transmit a frame | |
| | | Frame Too Long | UDINT | Frames received that exceed the maximum permitted frame size | |
| | | MAC Receive Errors | UDINT | Frames for which reception on an interface fails due to an internal MAC sub layer receive error | |
| 6 | Set | Interface Control | STRUCT of: | Configuration for physical interface | |
| | | Control Bits | WORD | Interface Control Bits | See table below |
| | | Forced Interface Speed | UINT | Speed at which the interface shall be forced to operate | Speed in Mbps (10 or 100) |

| ID | Access | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|
| 7 | Get | Interface Type | USINT | Type of interface | 1 = The interface is internal to the device, i.e. in the case of an embedded switch<br>2 = Twisted-pair (e.g. 100Base-TX) |
| 8 | Get | Interface State | USINT | Current state of the interface | 0 = No link<br>1 = The interface is enabled and is ready to send and receive data |
| 10 | Get | Interface Label | SHORT_STRING | Human readable identification | „Internal switch“ or „External Port 1“ or „External Port 2“ |
| 11 | Get | Interface Capability | STRUCT of: | Indication of capabilities of the interface | |
| | | Capability Bits | WORD | Interface capabilities, other than speed/duplex | Bit map |
| | | Speed/Duplex Options | STRUCT of: | Indicates speed/duplex pairs supported in the Interface Control attribute | |
| | | | USINT | Speed/Duplex Array Count | Number of elements |
| | | | ARRAY of STRUCT of | Speed/Duplex Array | |
| | | | UINT | Interface Speed | Semantics are the same as the Forced Interface Speed in the Interface Control attribute: speed in Mbps |
| | | | USINT | Interface Duplex Mode | 0=half duplex<br>1=full duplex<br>2-255=Reserved |
| 14 | Get | Ethernet Error | UDINT | Sum of certain error counts that are part of attributes 4, 5 and 13 | |
| 15 | Get | Link Down Counter | UDINT | Counts the number of times a Link Down transition event was detected on this port | |

**Control Bits**

| Bit(s) | Called | Definition |
|---|---|---|
| 0 | Auto-negotiate | 802.3 link Auto-negotiation: 0 = disabled, 1 = enabled (standard)<br>If Auto-negotiation is disabled, then the device shall use the settings indicated by the Forced Duplex Mode and Forced Interface Speed bits. |
| 1 | Forced Duplex Mode | If Auto-negotiation bit = 0 the Forced Duplex Mode bit indicates whether the interface shall operate in full or half duplex mode.<br>0 = Half Duplex, 1 = Full Duplex |
| 2-15 | Reserved | Shall be set to zero |

*Table 19: Ethernet Link Objects – Control Bits*

**Example**

Use on Transmit data size double (4 bytes) 00000064 for Auto-negotiation = disable on 100 MBaud

### 2.4.1 Interface Flags

| Bit(s) | Called | Definition |
|---|---|---|
| 0 | Link Status | Indicates whether the Ethernet 802.3 communications interface is connected to an active network. 0 indicates an inactive link; 1 indicates an active link. The determination of link status is specific to the implementation. In some cases, devices can tell whether the link is active via hardware/driver support. In other cases, the device may only be able to tell whether the link is active if it receives incoming packets. |
| 1 | Half/Full Duplex | Indicates whether the duplex mode is currently in use. 0 indicates the interface is running half duplex; 1 indicates full duplex. Note that if the Link Status flag is 0, then the value of the Half/Full Duplex flag is not determinate. |
| 2-4 | Negotiation Status | Indicates the status of link auto-negotiation:<br>0 = Auto-negotiation in progress.<br>1 = Auto-negotiation and speed detection failed. Using default values for speed and duplex. Default values are product-dependent; recommended defaults are 10Mbps and half duplex.<br>2 = Auto negotiation failed but detected speed. The duplex was defaulted. Default value is product-dependent; recommended default is half duplex.<br>3 = Successfully negotiated speed and duplex.<br>4 = Auto-negotiation not attempted. Forced speed and duplex. |
| 5 | Manual Setting Requires Reset | 0 indicates that the interface can perform changes to link parameters (auto-negotiate, duplex mode, interface speed) automatically. 1 indicates that the device requires a Reset service to be issued to its Identity Object for the changes to be effective. |
| 6 | Local Hardware Fault | 0 indicates the interface detects no local hardware fault; 1 indicates a local hardware fault is detected. The meaning of this is product specific. For example, an AUI/MII interface can detect that no transceiver is plugged, or a radio modem can detect that no antennae plugged. In contrast to the soft, possible self-correcting nature of the Link Status being inactive, this concerns major issues requiring the intervention of the user. |
| 7 | Reserved | Shall be set to zero |

*Table 20: Ethernet Link Objects – Interface Flags*

### 2.4.2 Common Services

| Service Code | Class | Instance* | Service Name | Description of Service |
|---|---|---|---|---|
| $0E_{hex}$ | Conditional | Required | Get_Attribute_Single | Returns the contents of the specified attribute |
| $10_{hex}$ | n/a | Conditional | Set_Attribute_Single | Modifies a single attribute |

*Table 21: Ethernet Link Objects – Common Services*

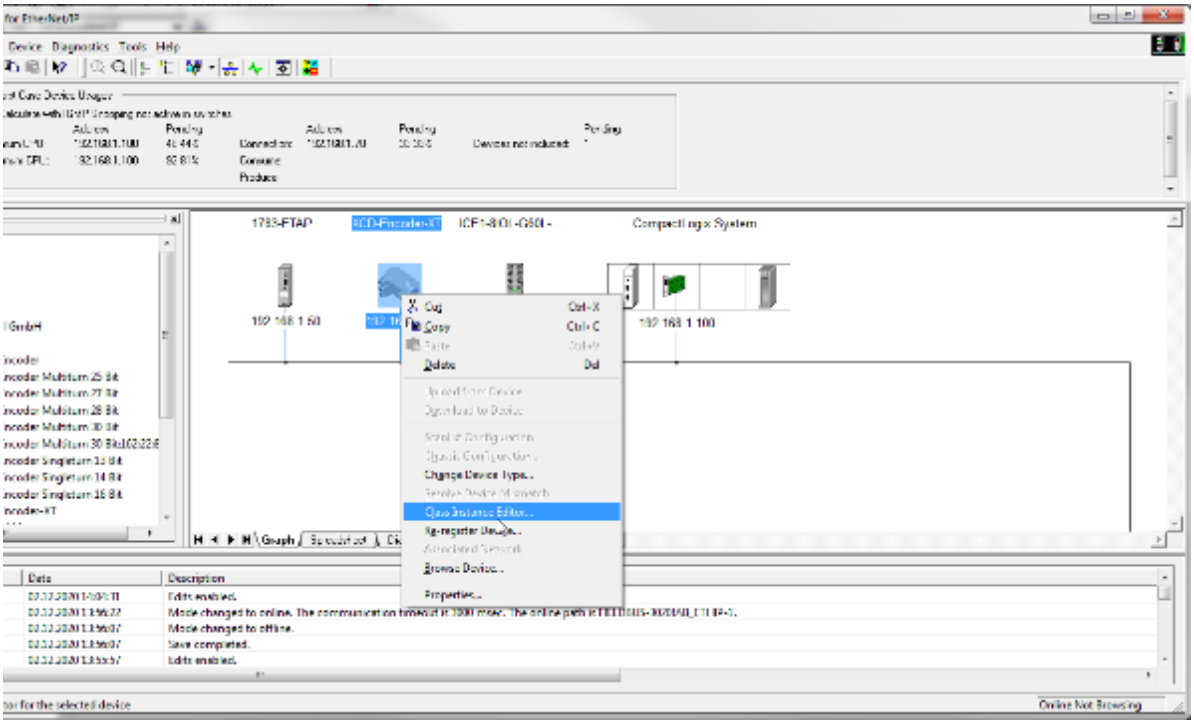
### 2.4.3 Link Object Instances

| Instance | Description |
|---|---|
| 1 | Internal interface |
| 2 | Intern switch Port 1 |
| 3 | Intern switch Port 2 |

*Table 22: Link Object Instances*

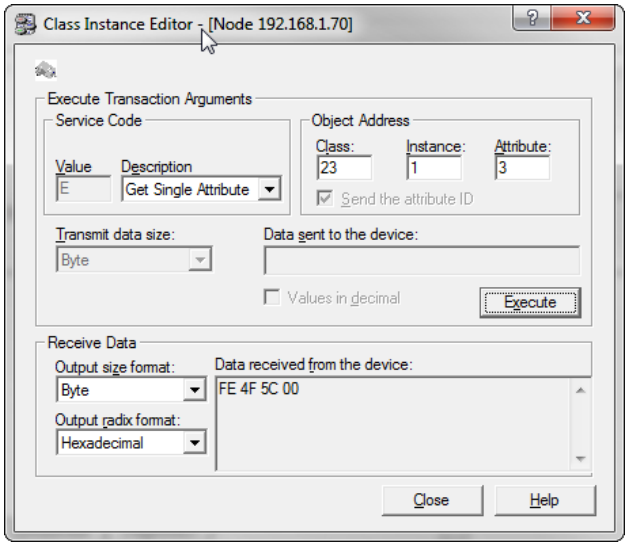
## 2.5 Setting parameters with scanners

Several external scanners exist for EtherNet/IP available. RS-NetWorx™ has one such scanner. In the figure you can see an example where the IP address (192.168.1.70) is read out of the encoder.



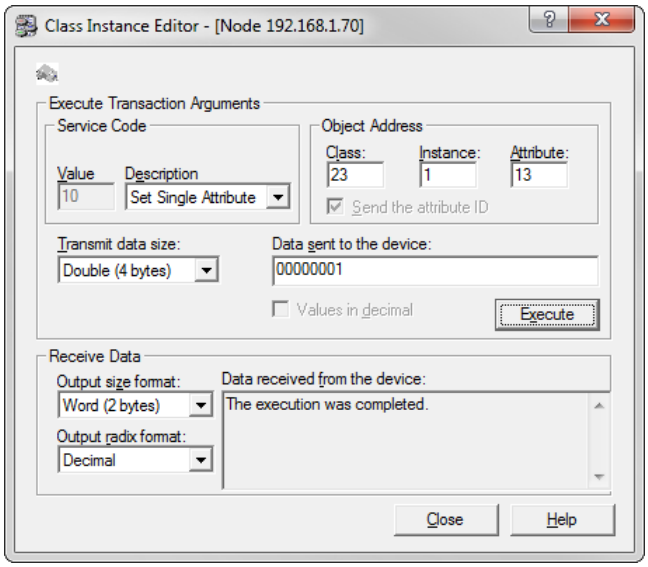
### 2.5.1 Read out position value

Get Single Attribute Position sensor value:
Class: 0x23 (Position sensor object)
Instance: 0x01
Attribute: 0x03 (Position Value)



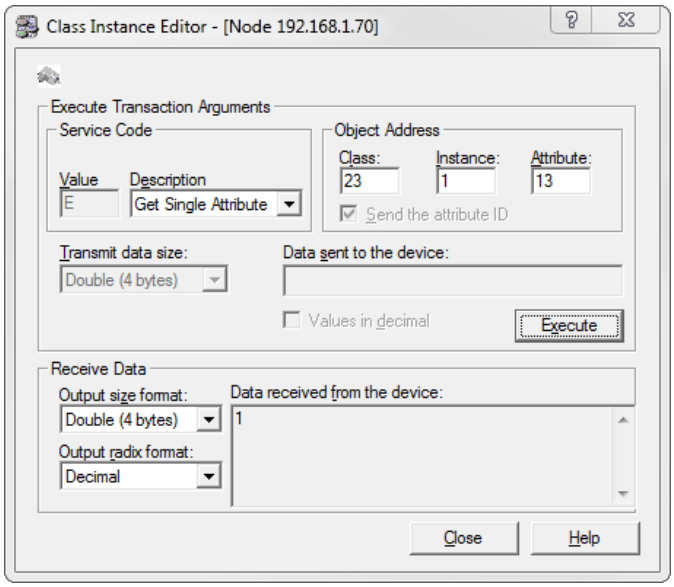

### 2.5.2 Set preset value

Set Single Attribute Position Preset Value to 1
Class: 0x23 (Position sensor object)
Instance: 0x01
Attribute: 0x13 (Preset Value)



**2.5.3 Get preset value**

Get Single Attribute Position Value
Class: 0x23 (Position sensor object)
Instance: 0x01
Attribute: 0x13 (Preset Value)



## 3 Diagnostics

| LED | Color | EtherNet/IP name | Description |
|---|---|---|---|
| Active1 | Yellow | Network Status Indicator 1 | Details in Table 24 |
| Link1 | Green | | |
| Active2 | Yellow | Network Status Indicator 2 | Details in Table 24 |
| Link2 | Green | | |
| Stat1 | Green | Module Status Indicator | Details in Table 23 |
| Stat2 | Red | | |

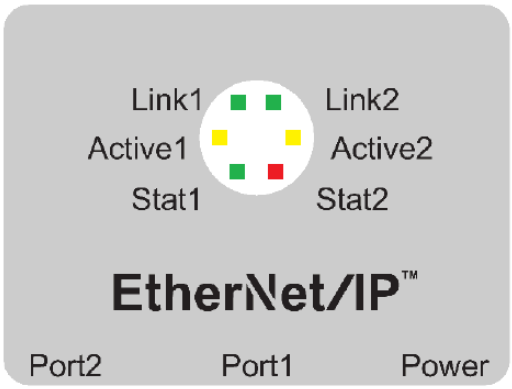



*Figure 6: Diagnostic LEDs*

| LED | Summary | Requirements |
|---|---|---|
| Steady Off | No power | |
| Steady On Green | Device operational | If the device is operating correctly, the module status indicator shall be steady green |
| Flashing Green 1 | Standby | If the device has not been configured (e.g. the IP-Address) the module status indicator shall be flashing green with 1 Hz |
| Flashing Green 2 | Missing IP | If the device does not have an IP-Address, the module status indicator shall be flashing green with 4 Hz |
| Flashing Red | Minor fault | If the device has detected a recoverable minor fault e.g. an incorrect or inconsistent configuration, the module will be flashing red |
| Steady On Red | Major fault | Happens when the device has detected a non-recoverable major fault |
| Flashing Red + Green | Self-test | While the device is performing its power up testing, the Stat1 and Stat2 LED shall be flashing red / green |

*Table 23: Module Status Indicator Stat1/Stat2*

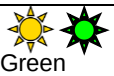
| LED | Summary | Requirements |
|---|---|---|
| Steady Off  | No power, no IP address | If the device does not have an IP address or is powered off |
| Steady Green  | Connected | If the device has at least one established connection (even to the Message router) |
| Flashing Green  | No connection | If the device has no established connections, but has obtained an IP address |
| Flashing Yellow  | Connection timeout | If the target of one or more of the connections in which this device is has timed out. This shall be left only if all timed-out connections are reestablished or if the device is reset |
| Steady Yellow  | Duplicate IP | If the device has detected that the IP address is already in use |
| Flashing Yellow / Green  | Self-test | While the device is performing its power up testing, the Stat1 and Stat2 LED shall be flashing yellow / green |

*Table 24: Network Status Indicator Stat2*

# 4 Programmable Parameters

### 4.1 Direction Counting

This operating parameter can be used to select the code sequence. The parameter can be set via Configuration Assembly and Explicit Messaging.

| Attribute ID | Default value | Value range | Data Type |
|---|---|---|---|
| $0C_{hex}$ | $0_{hex}$ | $0_{hex}$ - $1_{hex}$ | Boolean |

*Table 25: Direction Counting*

The parameter code sequence (complement) defines the counting direction of the process value **as seen from the shaft** (clockwise or counterclockwise). The counting direction is defined in the attribute $0C_{hex}$:

| Bit 0 | Counting direction | Position values |
|---|---|---|
| 0 | CW | Increase |
| 1 | CCW | Decrease |

*Table 26: Direction Counting Bits*

### 4.2 Scaling Function Control

If the Scaling function control is deactivated, then the output value is the physical resolution.

| Attribute ID | Default value | Value range | Data Type |
|---|---|---|---|
| $0E_{hex}$ | $0_{hex}$ | $0_{hex}$ - $1_{hex}$ | Boolean |

*Table 27: Scaling Function Control*

This parameter can be set via Configuration Assembly and Explicit Messaging

| Bit 0 | Scaling function on/off |
|---|---|
| 0 | OFF |
| 1 | ON |

*Table 28: Scaling Function Control Bits*

### 4.3 Resolution per Revolution

The parameter resolution per revolution is used to program the encoder to a desired number of steps per revolution. Each value between 1 and the maximum (see type label) can be realized. The parameter can be set via Configuration Assembly and Explicit Messaging. Scaling function control **must be switched on**!

| Attribute ID | Default value | Value range | Data Type |
|---|---|---|---|
| $10_{hex}$ | (*) | $0_{hex}$ - $10000_{hex}$ | Double Integer 32 |

*Table 29: Resolution per Revolution*

(*) see type label, maximum resolution:
16Bit Encoder: $10{,}000_{hex}$ (65,536)
If the value is set to a value larger than 8,192 for a 13-bits encoder, it will lead to values being skipped while rotating the shaft. Therefore, it is recommended to keep the measuring steps per revolution below 8,192 measuring steps.

### 4.4 Total Resolution

This value is used to program the desired number of measuring steps over the total measuring range. This value must not exceed the total resolution of the encoder.
E.g. with 25-bits = 33,554,432 steps.

Please note the value written on the type label. The parameter can be set via Configuration Assembly and Explicit Messaging.
Scaling function control **must be switched on**!

| Attribute ID | Default value | Value range | Data Type |
|---|---|---|---|
| $11_{hex}$ | (*) | $0_{hex}$ - $40{,}000{,}000_{hex}$ | Unsigned Integer 32 |

*Table 30: Total Resolution*

(*) see type label
Maximum total resolution
30 bits encoder: $40{,}000{,}000_{hex}$ (1,073,741,824)

If the Resolution per Revolution has been changed (see 4.1.3), the Total Resolution must be adapted accordingly.
The total resolution must follow the equation:

***Total Resolution = Number of Turns x Resolution per Revolution***

### 4.5 Preset Value

The preset value is the desired position value, which should be reached at a certain physical position of the axis. The position value of the encoder is set to the desired value by the preset parameter. The preset value must not exceed the total measuring range. The parameter can be set via Explicit Messaging.
**Set the preset value only in standstill!**

| Attribute ID | Default value | Value range | Data Type |
|---|---|---|---|
| $13_{hex}$ | $0_{hex}$ | $0_{hex}$ - total measuring range | Unsigned Integer 32 |

*Table 31: Preset Value*

### 4.6 Velocity Format

The default value for Velocity Format is steps per second. This parameter can be set via Configuration Assembly or Explicit Messaging.

| Attribute ID | Default value | Value range | Data length |
|---|---|---|---|
| $19_{hex}$ | $1F04_{hex}$ | $0800_{hex}$ | Steps per 10 milliseconds |
| | | $0801_{hex}$ | Steps per 100 milliseconds |
| | | $1F04_{hex}$ | Steps per 1 second |
| | | $1F05_{hex}$ | Steps per 1 millisecond |
| | | $1F06_{hex}$ | Steps per 1 microsecond |
| | | $1F07_{hex}$ | Steps per minute |
| | | $1F0E_{hex}$ | RPS (revolutions per second) |
| | | $1F0F_{hex}$ | RPM (revolutions per minute) |

*Table 32: Velocity Format*

### 4.7 Velocity Filter

To increase the accuracy of the velocity in the application, it is possible to switch between three different filtering types of the exponential moving average. Default: Fine.

| Attribute ID | Default value | Value range | Description | Data Type |
|---|---|---|---|---|
| $70_{hex}$ | $0_{hex}$ | $0_{hex}$ / $1_{hex}$ / $2_{hex}$ | 0 = Fine, 1 = Normal, 2 = Coarse | Double Integer |

*Table 33: Velocity Filter*

Ratio between old and current speed:
Fine: 7:3, Normal: 96:4, Rough: 996:4

### 4.8 Endless Shaft

It is important that the ratio “Total resolution” / “measuring units per revolution” remains a multiple of the total number of turns. For example, for a 12-bit revolutions encoder, the ratio must remain a multiple of 4,096:

***Total Resolution / Measuring Units per Revolution = n x 4,096***
Where *n* is an integer

However, the Endless Shaft functionality of POSITAL’s EtherNet/IP encoders can solve the problem directly. It will check whether *n* is an integer (see equation here above).
If not, it will run calculous to make sure we do not see any drops in the position (see pictures here after).
The Endless Shaft functionality must however be enabled.
Default value: Auto.
Can be set only via Explicit Messaging.

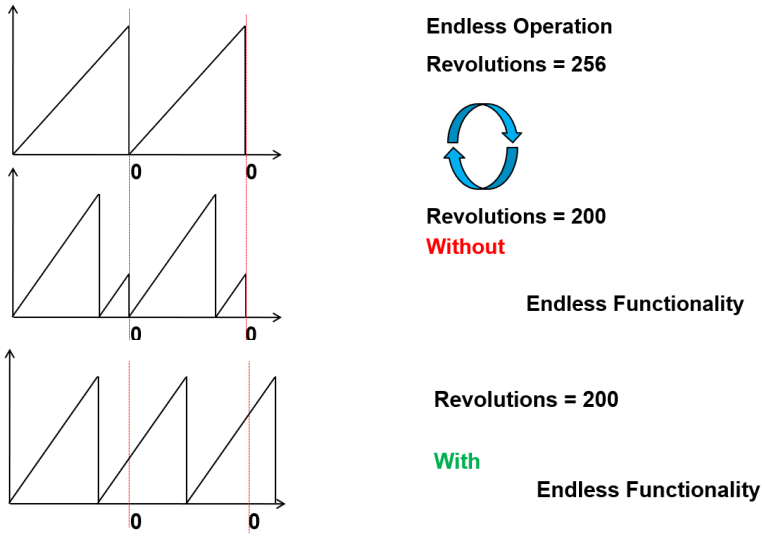



*Figure 7: Endless Shaft*

**Note:** The internal software routine only works if the encoder is in operation. If it is necessary to turn the encoder shaft more than 1,024 revolutions without power supply this can lead to problems (the internal routine will not work without power supply). Therefore, the rule ahead should be observed for new devices.

| Attribute ID | Default value | Value range | Description | Data Type |
|---|---|---|---|---|
| $65_{hex}$ | $2_{hex}$ | $0_{hex}$ / $1_{hex}$ / $2_{hex}$ | 0 = Off, 1 = On, 2 = Auto | Double Integer |

*Table 34: Endless Shaft*

### 4.9 Parameter Control Priority

This parameter determines if the configuration parameters should be loaded from the Configuration Assembly or from the Non-Volatile Data.

| Attribute ID | Default value | Value range | Description | Data Type |
|---|---|---|---|---|
| $71_{hex}$ | $0_{hex}$ | $0_{hex}$ / $1_{hex}$ | 0 = Configuration Assembly, 1 = Non-Volatile Memory | Double Integer |

*Table 35: Parameter Control Priority*

### 4.10 Available Alarms and Warnings

Attributes 44 ($2C_{hex}$) and 47 ($2F_{hex}$) respectively indicate alarms and warnings for the following parameters:

- Position error (alarm attribute): indicates that the internal position self-test failed
- Diagnostic error (alarm attribute): indicates that a malfunction has occurred that can lead to an incorrect position
- Temperature exceeded (warning attribute): indicates that the temperature has exceeded the defined threshold (85°C)

| Attribute ID | Default value | Value range | Description | Data Type |
|---|---|---|---|---|
| $2C_{hex}$ | $0_{hex}$ | $0_{hex}$ / $1_{hex}$ | Bit 0: Position error<br>Bit 1: Diagnostic error<br>Bit 2-15: Reserved<br>0 = No, 1 = Yes | WORD |

*Table 36: Alarms*

| Attribute ID | Default value | Value range | Description | Data Type |
|---|---|---|---|---|
| $2F_{hex}$ | $0_{hex}$ | $0_{hex}$ / $1_{hex}$ | Bit 0-12: Reserved<br>Bit 13: Temperature warning<br>0 = No, 1 = Yes | WORD |

*Table 37: Warnings*

### 4.11 Device Level Ring

The DLR protocol is intended to support simple ring topologies requiring fast recovery from network failures.

A DLR network consists of an active Ring Supervisor and any number of Ring Nodes. Ring nodes incorporate embedded switch technology with at least two external ports. This allows the data to be re-routed backwards to the other devices on the ring, should a failure occur.

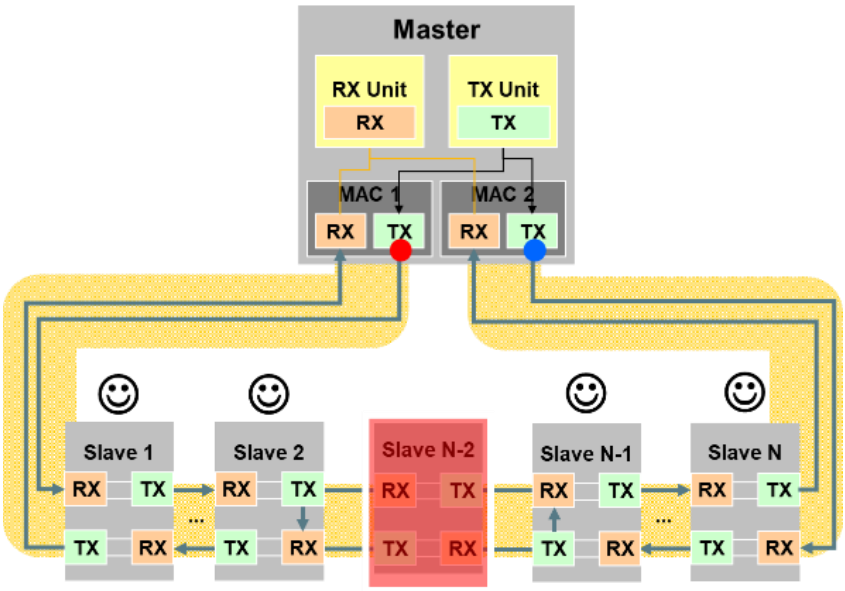



*Figure 8: Device level Ring*

# 5. Installation

## 5.1 Electrical connection

The rotary encoder is connected by a 4 pin M12 connector for the power supply and two 4 pins, D-coded M12 connectors for Ethernet communication.
The encoder uses a second D-coded connector and provides integrated switch functionality.

**Connector Ethernet**
4 pin female, D-coded

| Pin Number | Signal |
|---|---|
| 1 | Tx + |
| 2 | Rx + |
| 3 | Tx - |
| 4 | Rx - |

*Table 38: Data Pin Assignment*

**Connector power supply**
4 pin male, A-coded

| Pin Number | Signal |
|---|---|
| 1 | US (10 - 30 V DC) |
| 2 | N.C. |
| 3 | GND (0V) |
| 4 | N.C. |

*Table 39: Power Supply Pin Assignment*

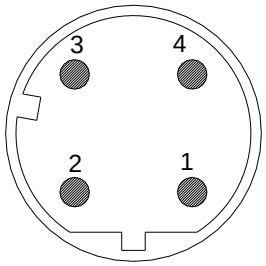



*Figure 9: Ethernet Connector*

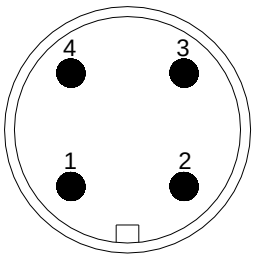



*Figure 10: Power Supply Connector*

## 5.2 Ethernet cables

### 5.2.1 RJ45 – M12 crossed

| Signal | RJ45 Pin | M12 Pin |
|---|---|---|
| Tx+ | 1 | 2 |
| Tx- | 2 | 4 |
| Rx+ | 3 | 1 |
| Rx- | 6 | 3 |

*Table 40: Ethernet cable RJ45-M12 crossed*

### 5.2.2 RJ45 – M12 straight

| Signal | RJ45 Pin | M12 Pin |
|---|---|---|
| Tx+ | 1 | 1 |
| Tx- | 2 | 3 |
| Rx+ | 3 | 2 |
| Rx- | 6 | 4 |

*Table 42: Ethernet cable RJ45-M12 straight*

### 5.2.3 M12 – M12 straight

| Signal | M12 Pin | M12 Pin |
|---|---|---|
| Tx+ | 1 | 1 |
| Tx- | 2 | 2 |
| Rx+ | 3 | 3 |
| Rx- | 4 | 4 |

*Table 41: Ethernet cable M12-M12 crossed*

# 6 Power On

After power on the LEDs on the absolute rotary encoder will flash between green and red or yellow.

# 7 Installation

## 7.1 Rockwell configuration tools

### 7.1.1 Setting IP-Address (BOOTP)

To set the IP Address there are special tools available. I.e. the BOOTP Server is installed with the software package from RSNetWorx™. The server scans the network for the MAC Addresses of all products with active BOOTP. If one MAC address is selected in the Request History then the IP Address can be set by the “Add to Relation List” button. The MAC Address of each EtherNet/IP encoder is available on the type label. Note: After a power up the encoder send the BOOTP request again. But after several times with no answer the frequency of requests decreases. A power up after a long pause could solve the missing requests.

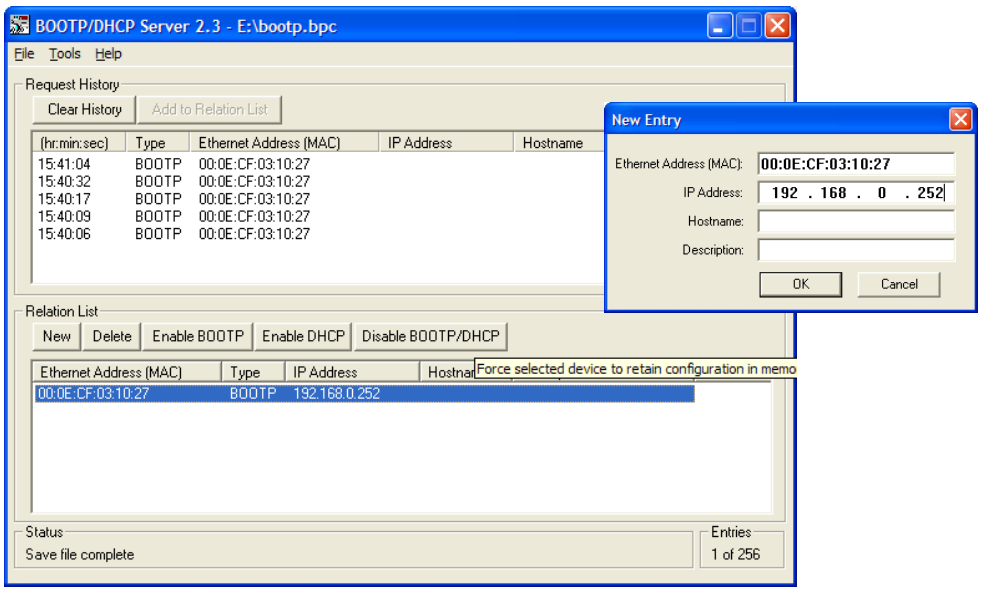
If you cannot find all the encoders within the BOOTP Server list then check the following points:

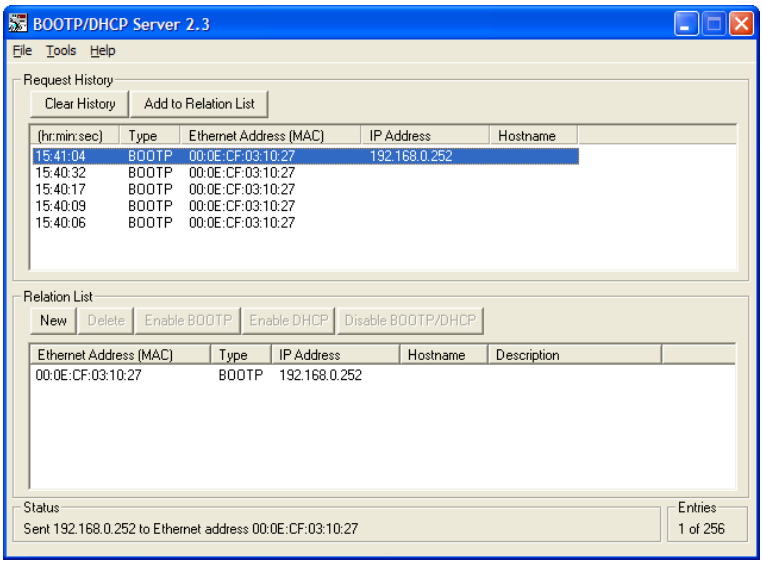
- LED status of the encoder OK?
- Is the Network correctly set?
- Is the BOOTP enabled?

If the encoder already has an IP-Address, the BOOTP must be disabled. Otherwise, the encoder will ask for a new IP-Address again. After setting the IP-Address the Status LED is flashing with 1 Hz. In this case, save the configuration in the File menu, because the products cannot be found by the BOOTP Server. After loading this file, the MAC Address and IP-Address are available and BOOTP can be deactivated by the corresponding button. Possible IP-Range:

Class A-C (0.0.0.0 – 223.255.255.255) without Loopback range (127.x.x.x)

Referenced IP-Address range: 192.168.0.x






After setting the IP-Address with this tool the IP-Address will be available only after the next BOOTP request.

If the IP-Address is not known and BOOTP is deactivated it is possible with the tool named Wireshark to find the IP-Address.

**NOTE:**
Alternatively, for the encoders with 3 rotary switches, the IP address can be set directly via the rotary switches present on the connection cap.

| IP address | Meaning |
|---|---|
| 888 | - The encoder gets its IP address from a BOOTP Server and saves the TCP parameters in non-volatile memory.<br>- BOOTP is always activated and cannot deactivate via server.<br>- The encoder is not allowed to send confirmation that BOOTP is deactivated!<br>- After each Power-up the encoder needs the IP configuration again!<br>- This is the emergency configuration, if the IP address is not known for activating BOOTP. |
| 0, 255-887, 889-999 | - On activated BOOTP the encoder gets its IP address and the other TCP parameters from a BOOTP server.<br>- On deactivated BOOTP the encoder uses the IP address from non-volatile memory.<br>- The encoder sends a confirmation that BOOTP is deactivated |
| 001 – 254 | Allowed ranged of configurable IP addresses for xxx |
| 192.168.1.xxx | Fixed address range* |
| 255.255.255.0 | Fixed subnet mask* |
| 0.0.0.0 | Fixed gateway address* |

*Table 43: Rotary Switches Allowed IP Addresses*

* can be modified

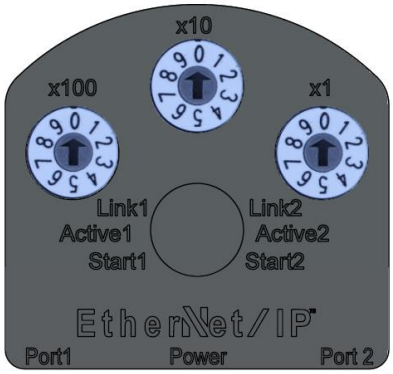
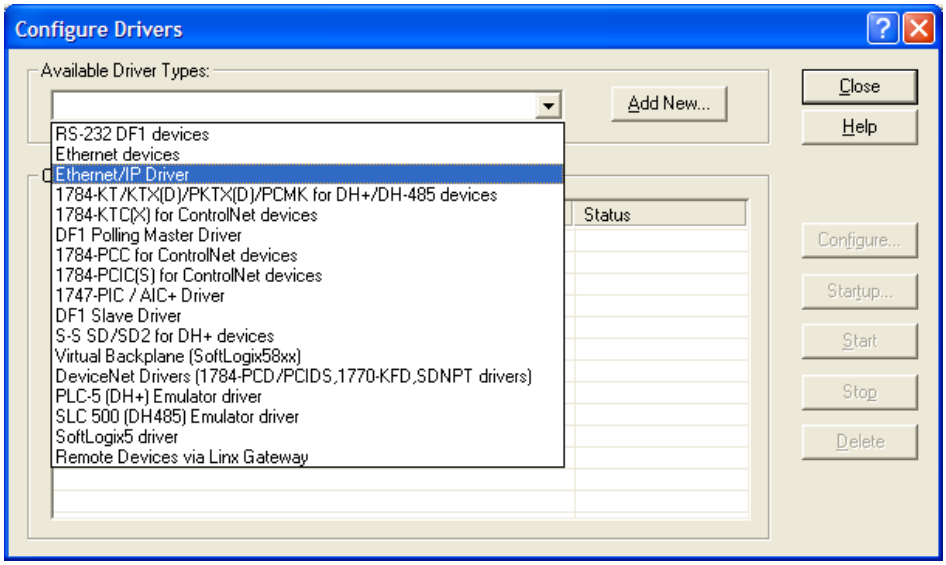


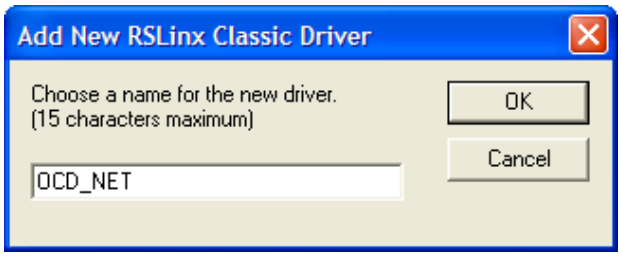

*Figure 11: IP Address Rotary Switches*

### 7.1.2 Configuration RSLinx Classic™

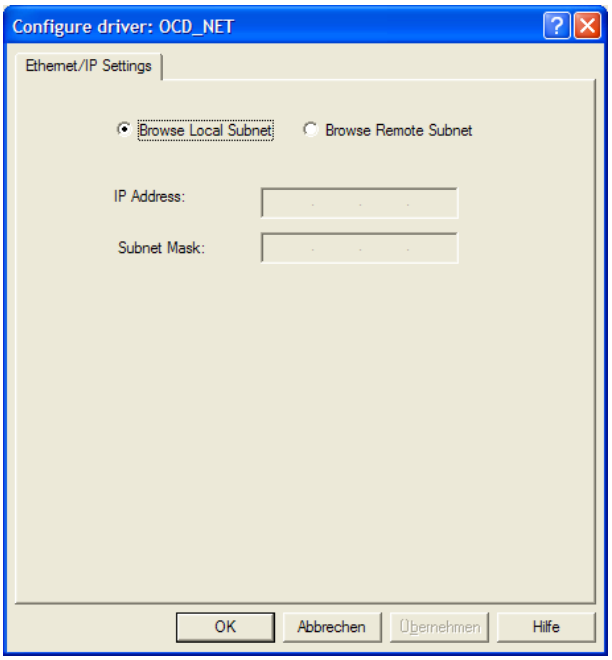
RSLinx™ is a complete communication server providing plant-floor device connectivity for a wide variety of Rockwell Software applications such as RSLogix™, RSNetWorx™,…

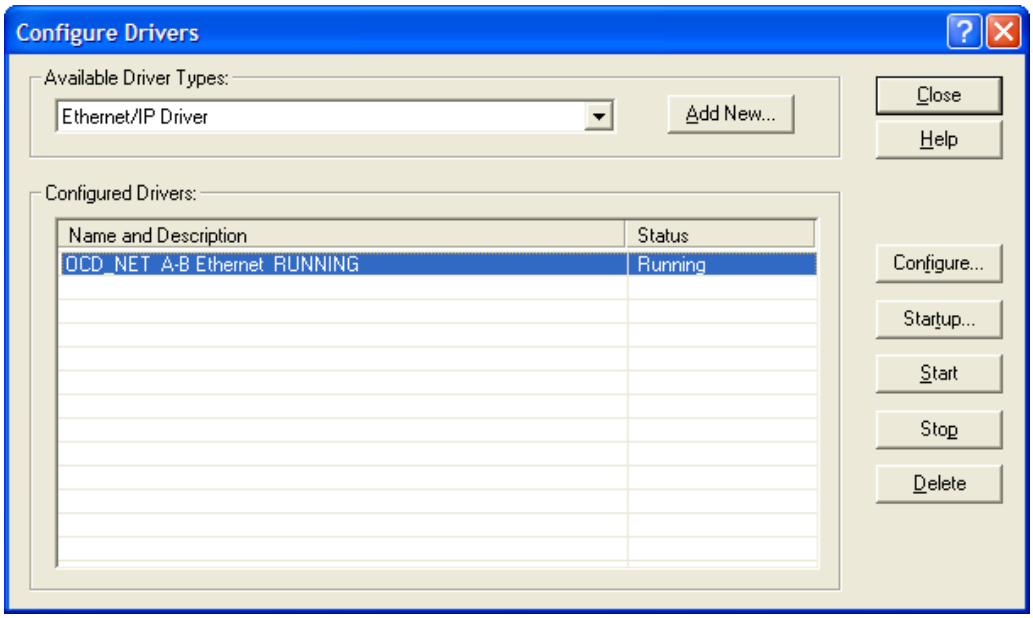
To start a new project add first a new RSLinx Classic™ Driver for EtherNet/IP under *Communications Configuration Drivers* and input the name.



Use *Browse Local Subnet* to find the EtherNet/IP components in the network. The status should be "Running". Then push the *Close* button to finish this configuration.



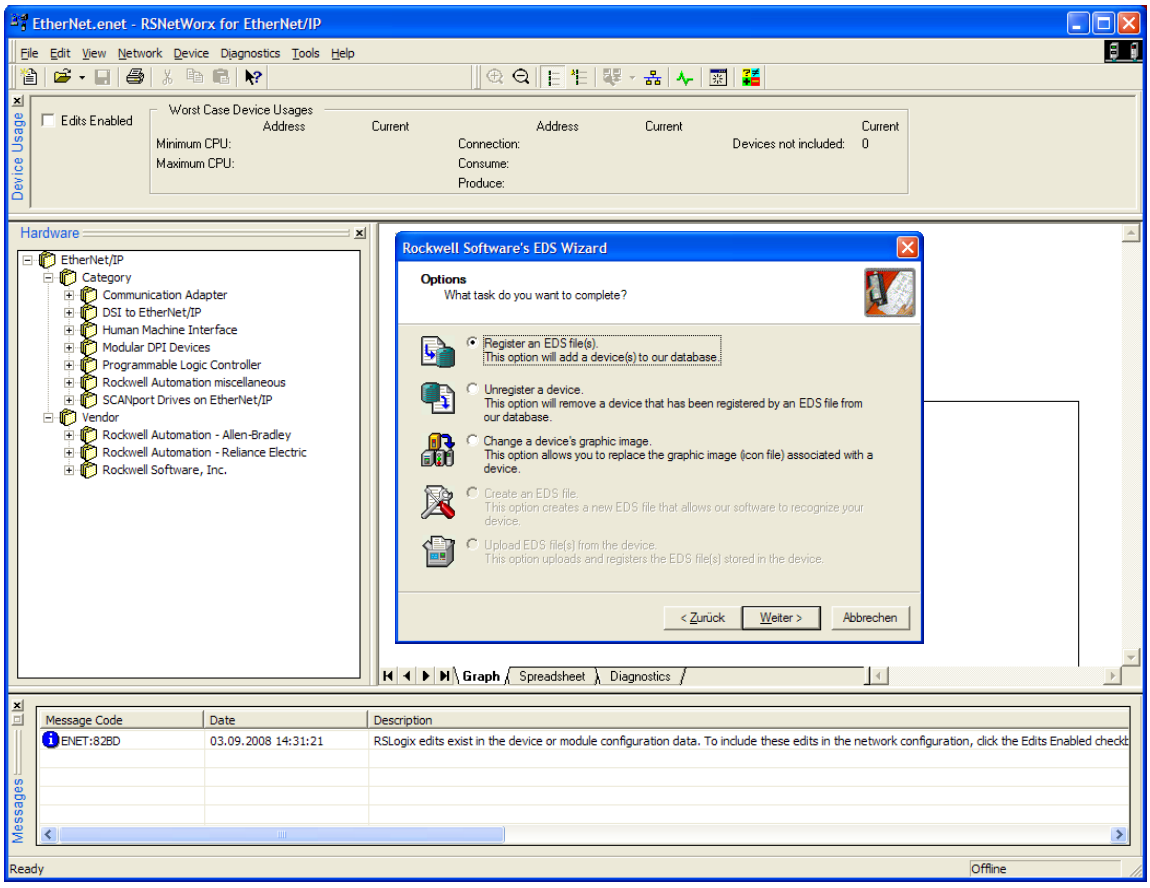



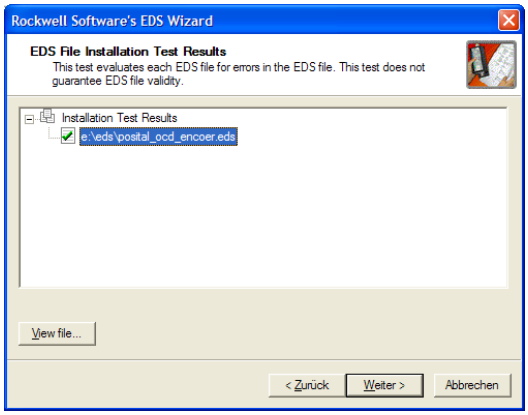
### 7.1.3 RSNetWorx™

RSNetWorx™ products provide design and configuration management services for EtherNet/IP. The program defines and configures the devices on the network quickly through a simple software interface. This definition can take place offline using drag and drop operations or online by using RSLinx® to browse an EtherNet/IP network.

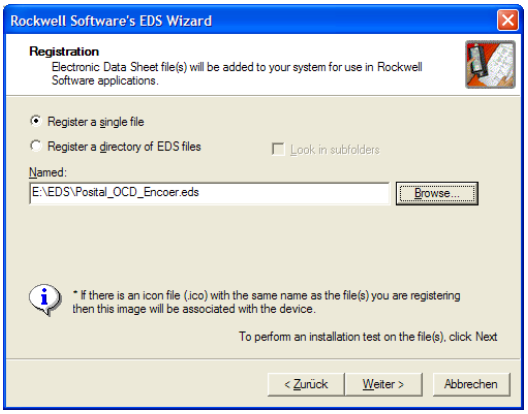
**EDS Wizard**

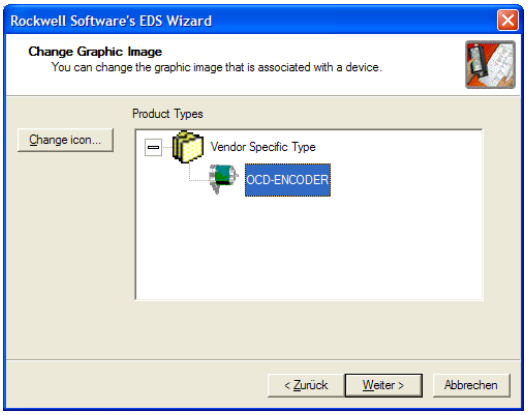
The EDS File contains information about device specific parameters as well as possible operating modes of the encoder. With this file you have a data sheet in an electronic format, which can be used to configure the device in the network, for example with RSNetWorx™ from Rockwell. In this sample the PLC uses address 192.168.0.100 and the encoder 192.100.0.252.
To install the EDS file the EDS Wizard has to be started, that can be done in the menu *Tools/EDS Wizard*. If the EDS Wizard is activated successfully the *Register an EDS File(s)* has to be chosen and after that the button *weiter*. In the next step the *Register a directory of EDS files* has to be chosen and with *Browse* the path of the EDS file(s). That is indicated in the next pictures.

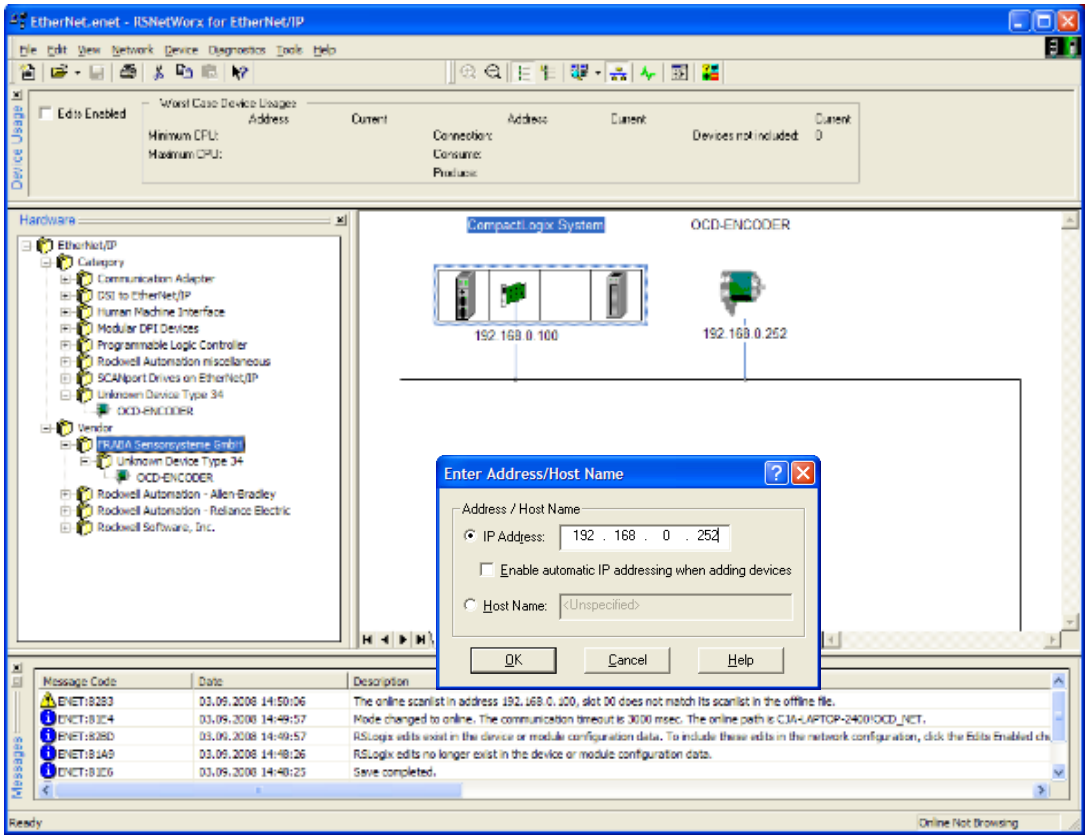





The Wizard finds all EDS files that are discarded in the choosing path and operates a test to check the EDS files on errors. In the next step pictures can be selected for the using nodes. With the button *waiter* the installation can be continued and finished.

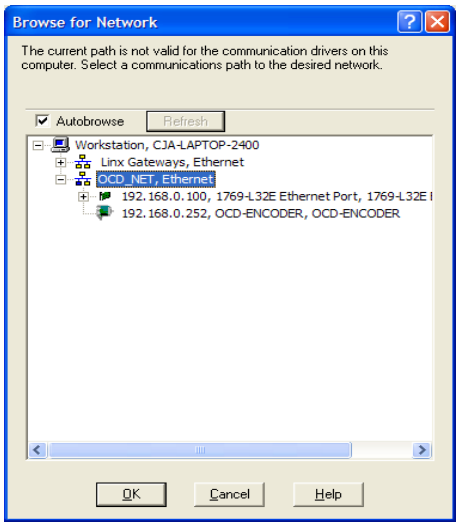





Load a saved *.enet file or start a new project.
Add the devices per Drag and Drop to the network
Line and set the IP-Address

Optionally, browse the network with all devices with Button  or *Upload from Network*. It is not necessary to set the IP-Address manually. To use this configuration in RSLogix save the *.enet file.

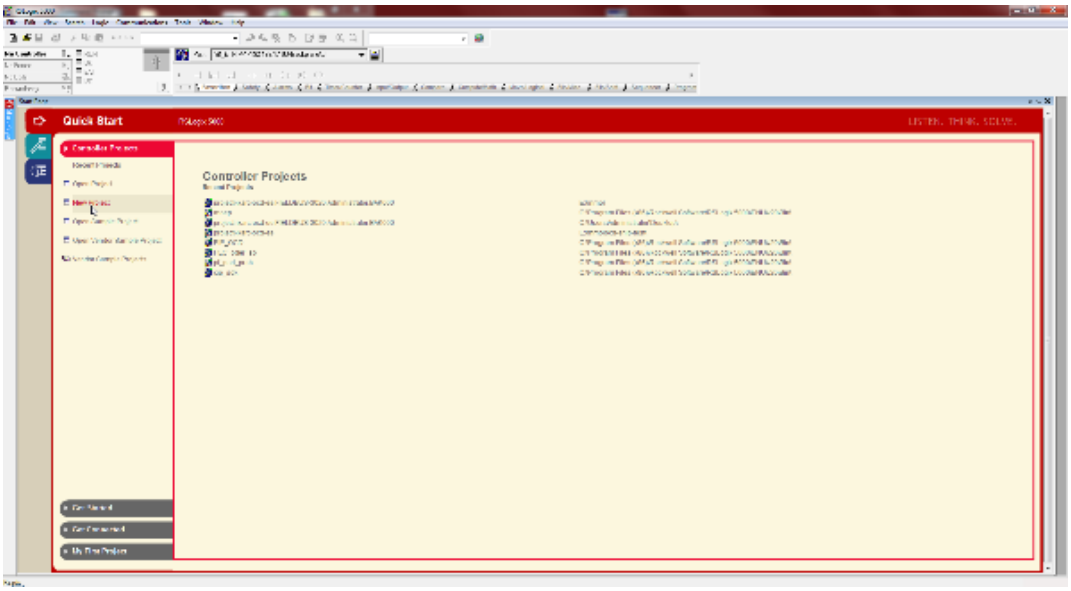


### 7.1.4. Configuration RSLogix 5000

The RSLogix 5000 Series environment offers an easy-to-use, IEC61131-3 compliant interface, symbolic programming with structures and arrays, and a comprehensive instruction set that serves many types of applications. It supports relay ladder, structured text, function block diagram, and sequential function chart editors for you to develop application programs.

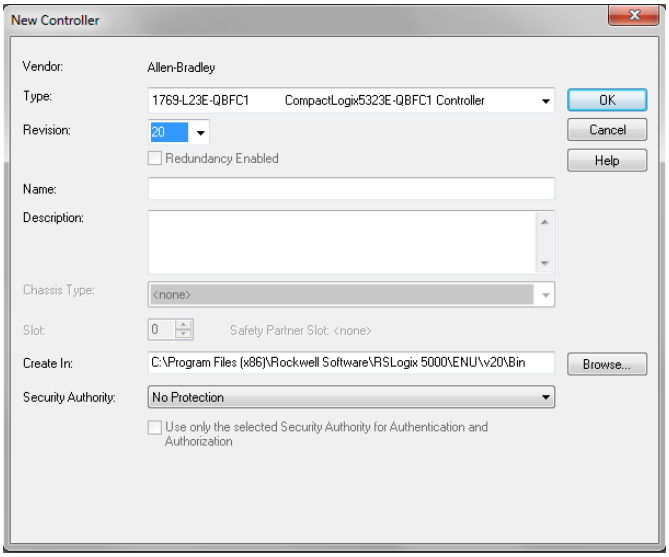
---

1. Create a New Project

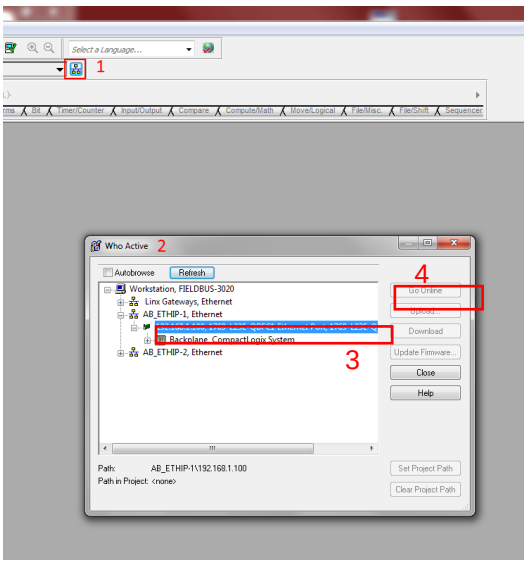



---

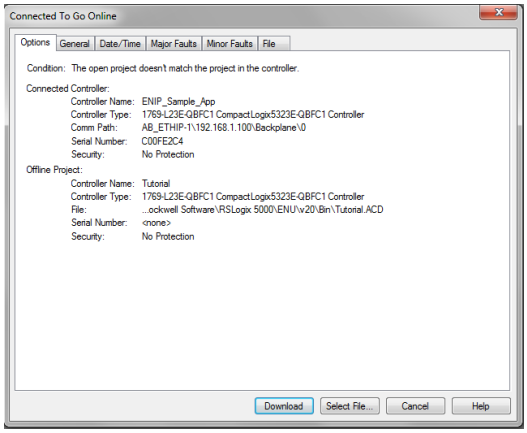
2. Choose your PLC in the available list



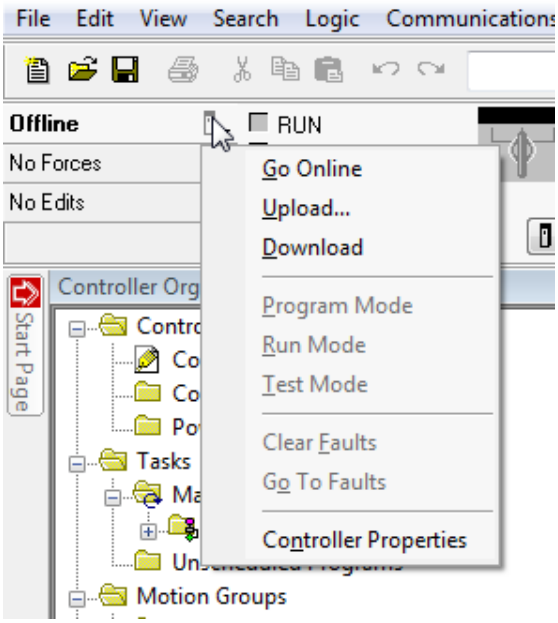
3. Look for the connection path to the encoder (1) In the pop-up window (2) Select the encoder (3) and Click on **Go Online (4)**.




4. Click on **Download**
The empty program will be loaded to the PLC.
A connection is established.



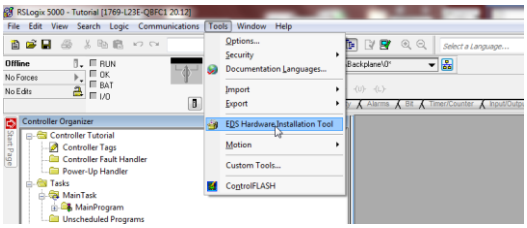

5. Go **Offline** (necessary to perform changes in the configuration)



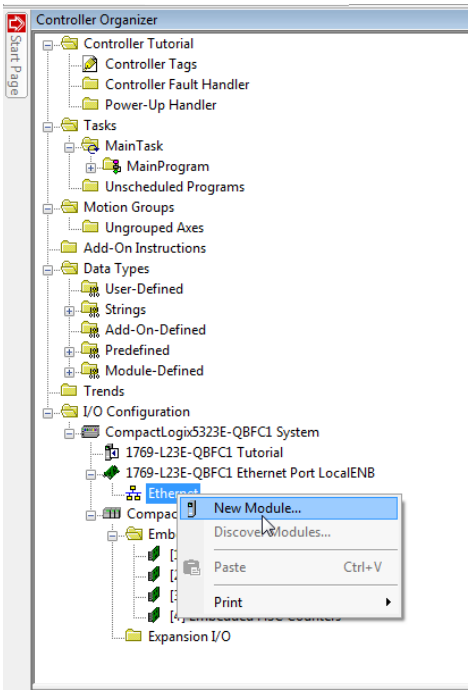
6. Add the EDS file

   Click on **EDS Hardware Installation Tool**

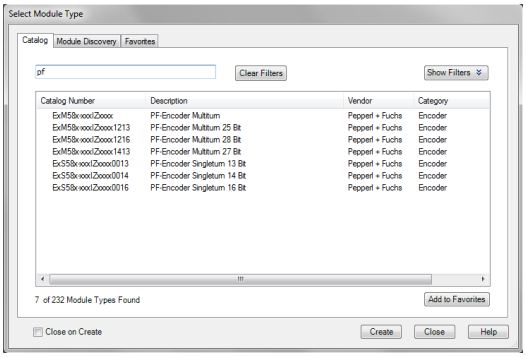
   **Next** -> **Register an EDS** -> **Next** -> **Browse** to the EDS file location -> **Next** -> **OK**



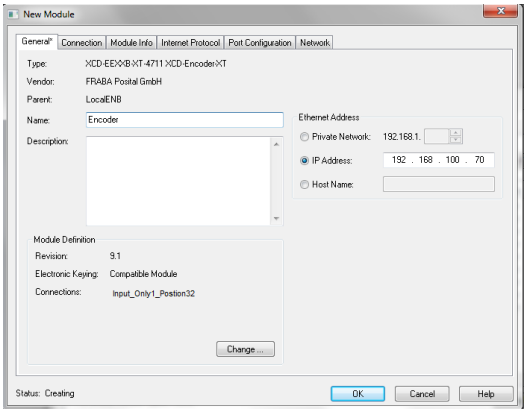

7. Add a new module



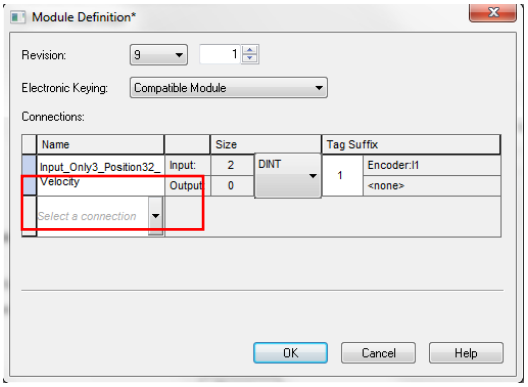

8. Look for your encoder



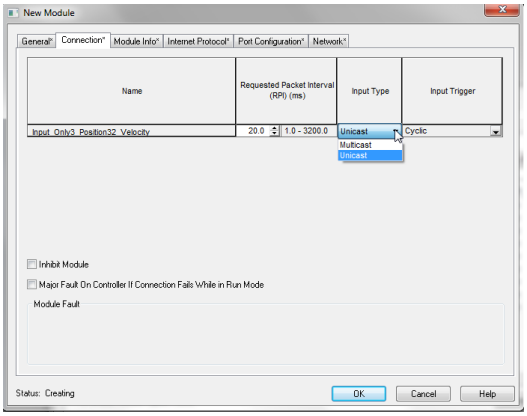
9. Set Name and IP address



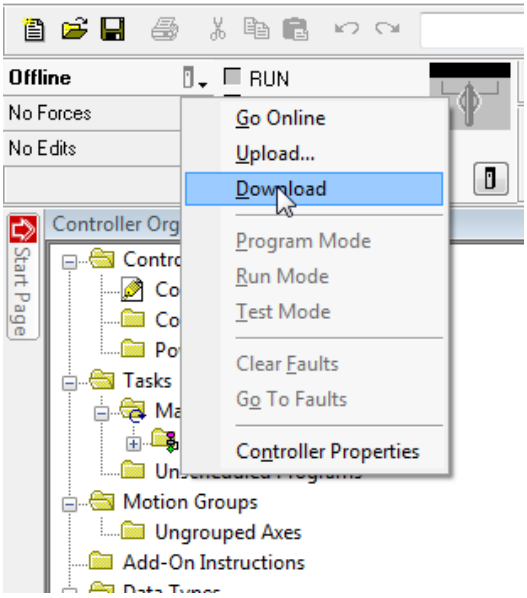

10. Select the relevant input assembly
Do not forget to select the right size and format(1)




11. Configure Cycle Time, Multicast/Unicast, Input Trigger

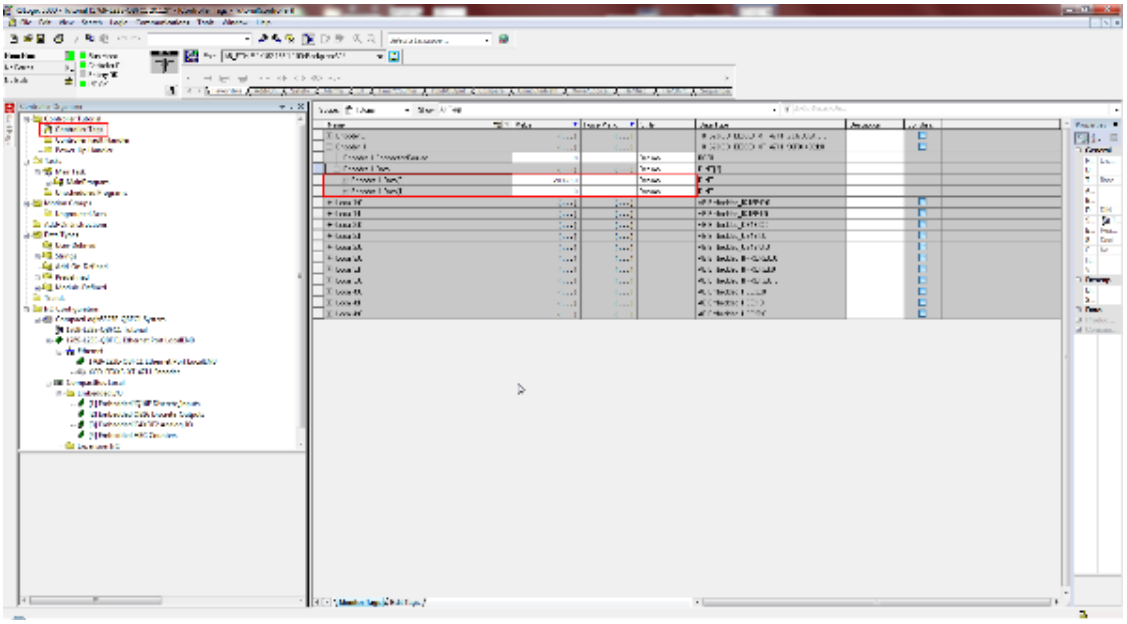



12. Download the configuration on the PLC

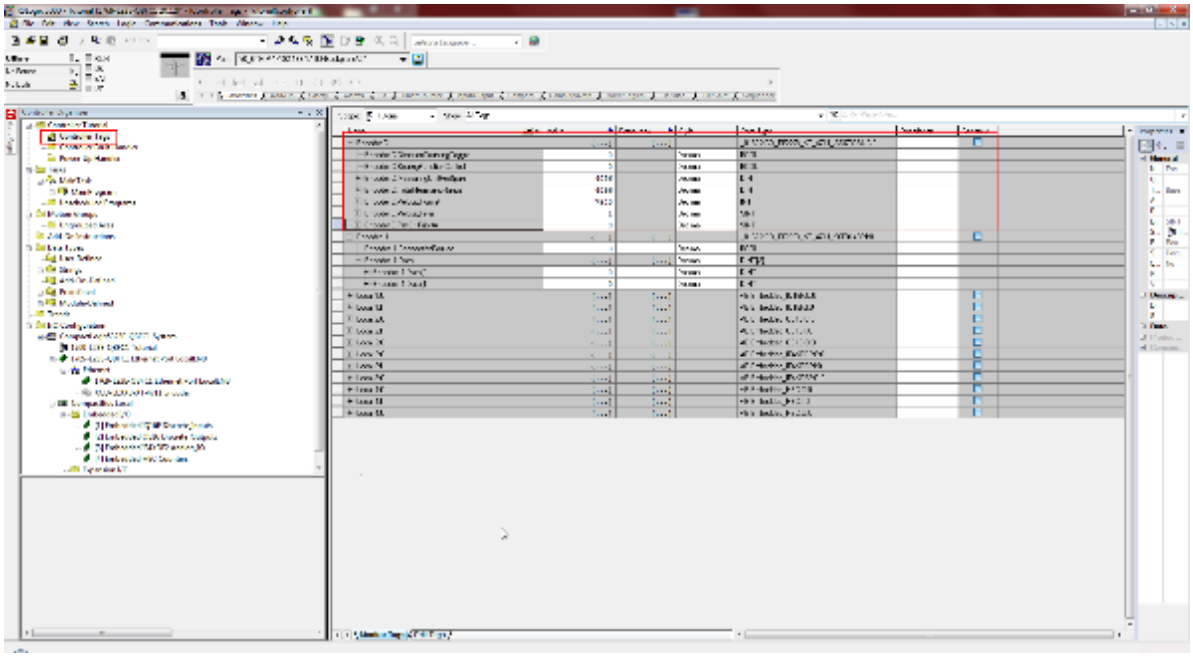



(1) You can change the Input Assembly later by Going **Offline** -> Changing the **Input Assembly** (in the same window) -> **Download** on the PLC

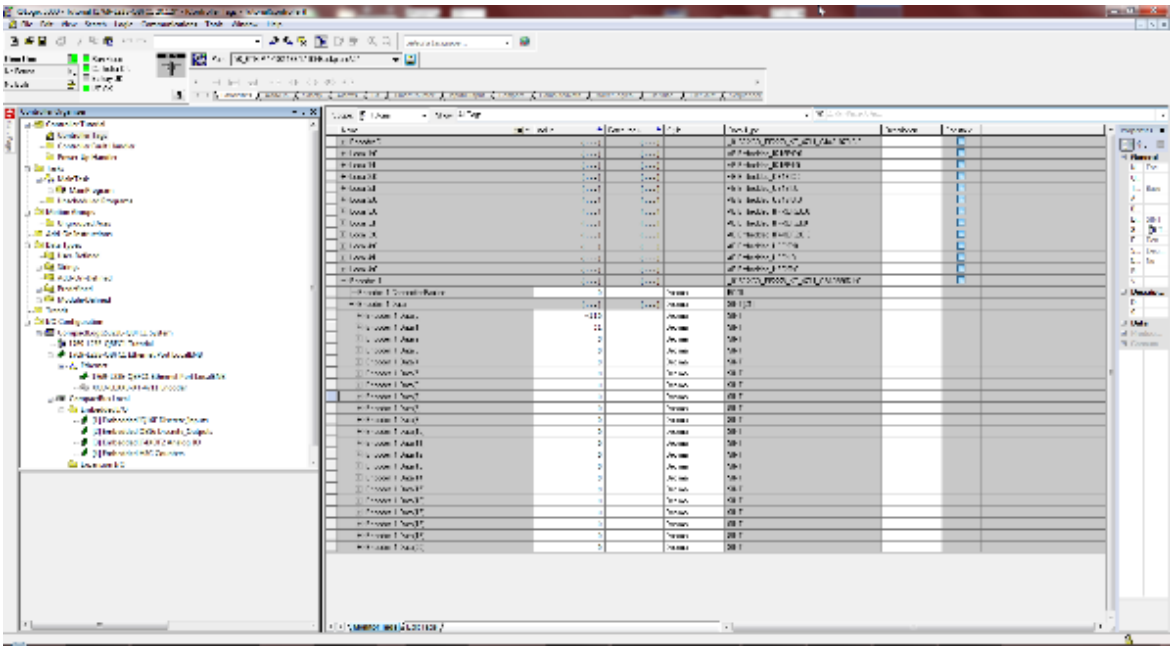
13. In the **Controller Tags** window, you can see *Position* and *Velocity*



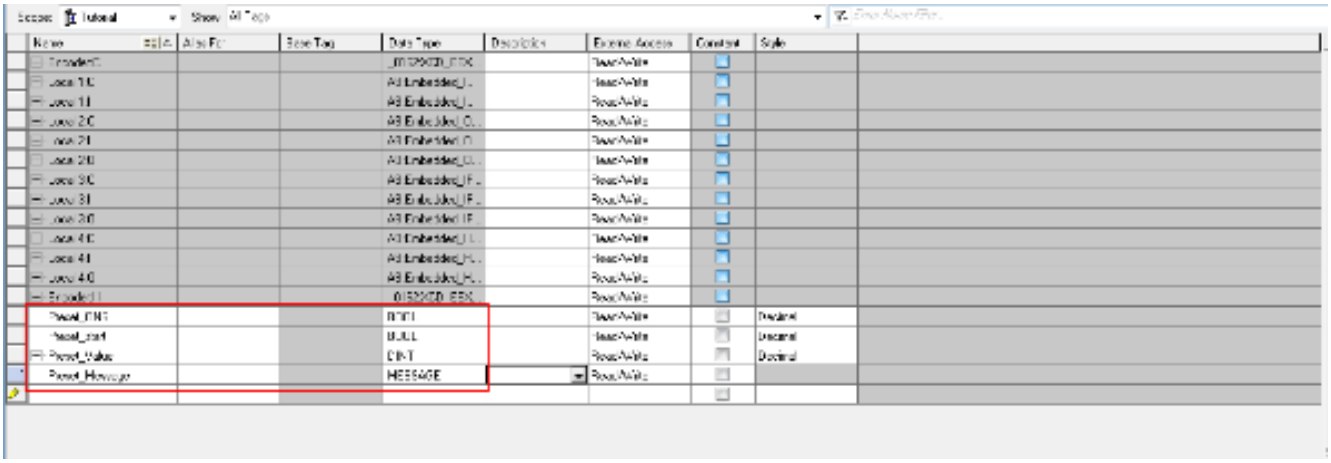
14. In the **Controller Tags** Window, you can set the following parameters:
    *DirectionCountingToogle, ScalingFunctionControl, MeasuringUnitPerSpan,TotalMeasuringRange,VelocityFormat, VelocityFilter,PrmCtrlPriority*



15. You can see Acceleration, Temperature and Warning Flags by selecting the Input Assembly 121 as instance (see pa
10 and note (1))
The values of the different Bytes are then visible in the **Controller Tags** window

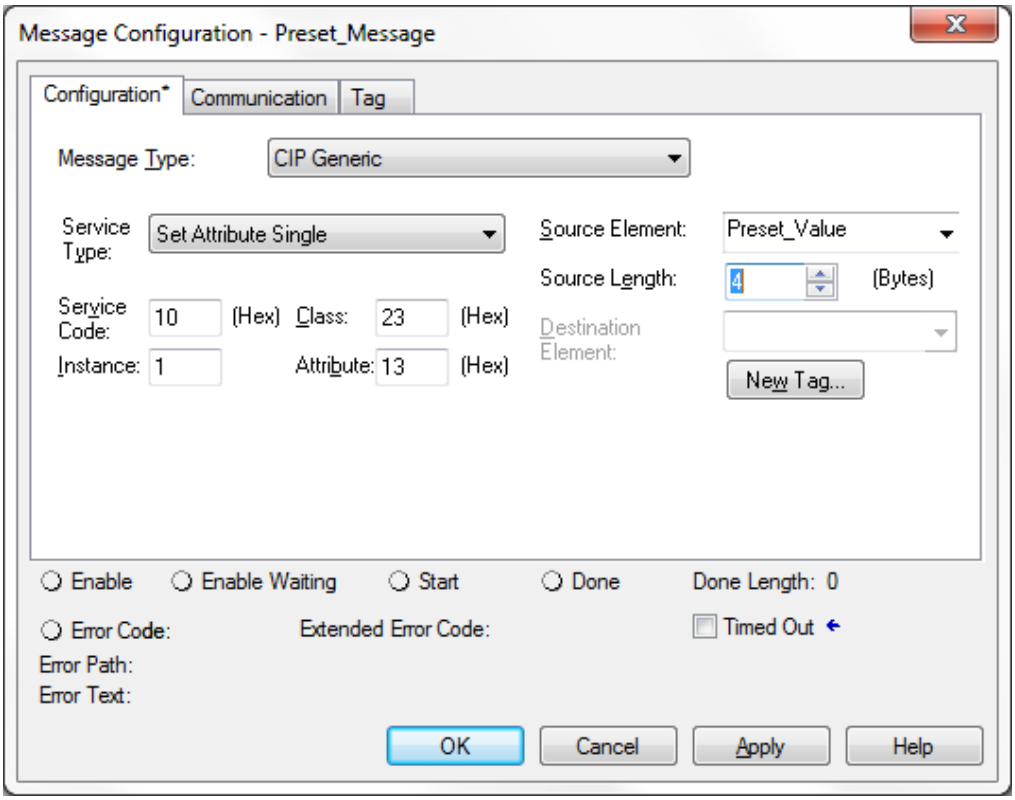


16. Set Preset
In the **Controller Tags** wndow, Create the following tags:

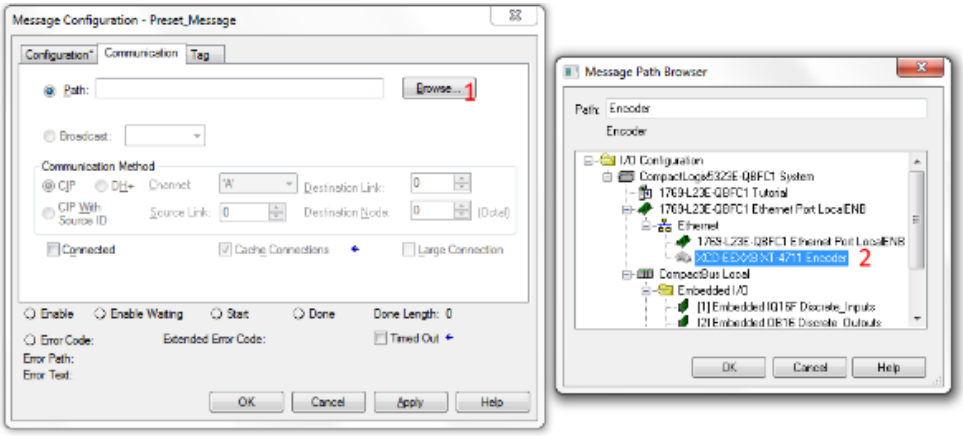


Create the following Main Routine
Click on **View Configuration**

In the **Configuration Tab**, set *Service Type, Code, Class, Attribute, Instance, Source Element* and *Length* as below:

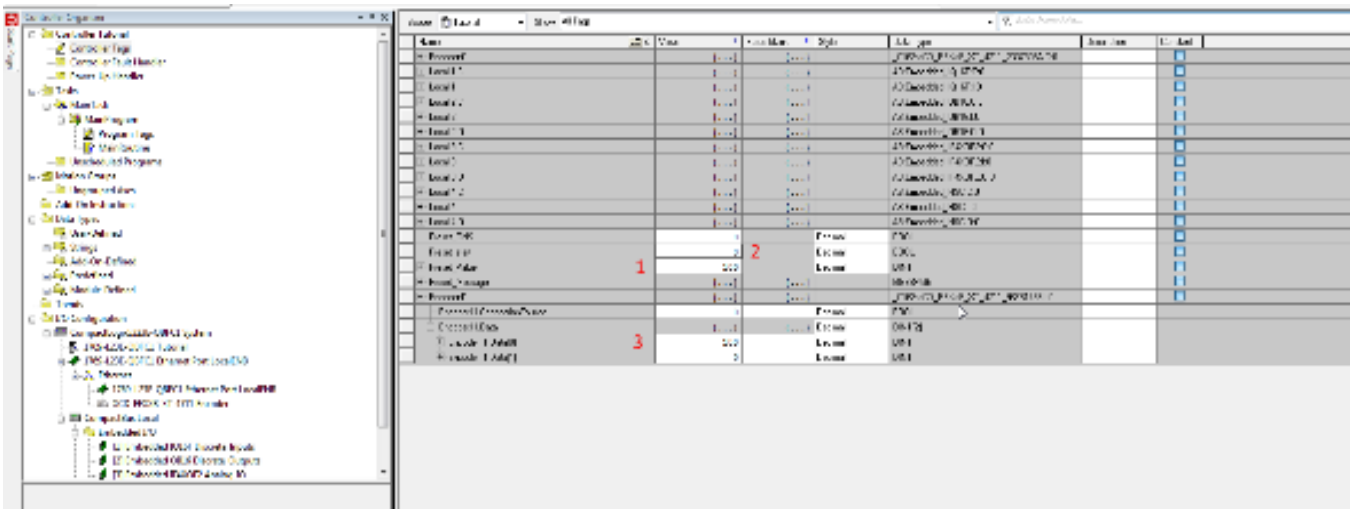



In the **Communication** tab, **Browse** (1) for the encoder (2)
Click on **Apply** and **OK**
**Download** the configuration



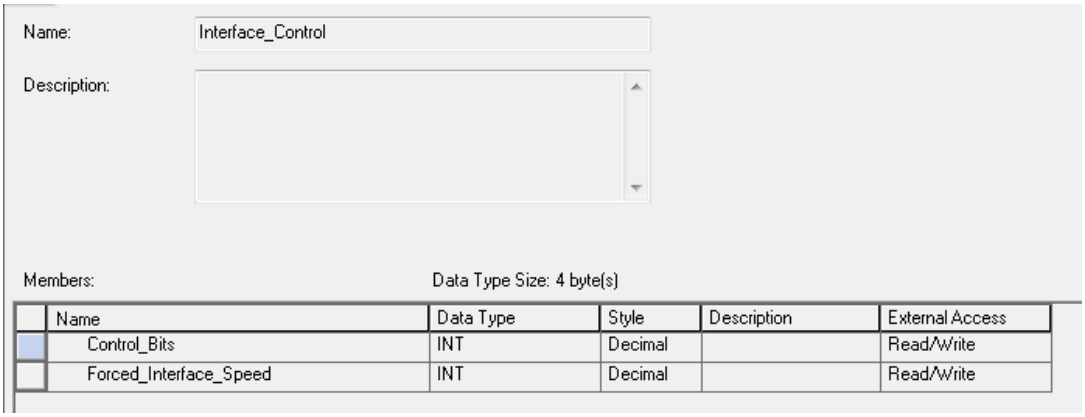

In the Controller Tags window:

1. Set the *Preset_value* as desired
2. Set *Preset_start* to “1”
3. See value changed. Please note that Preset_start is back to “0”

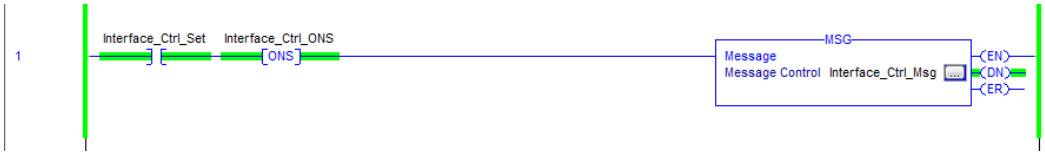



---

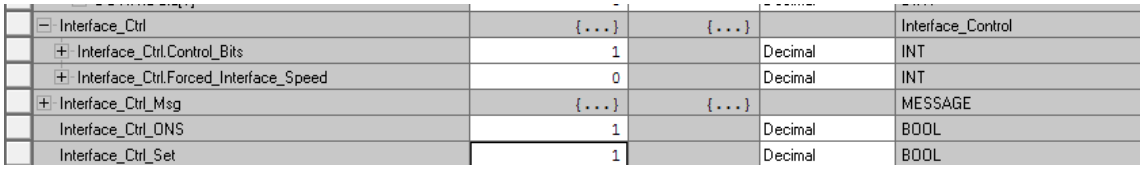
17. Set Auto-negotiate and Half/Full Duplex Mode
    Define **User Type** as follows:




Create the following **Ladder Program**:

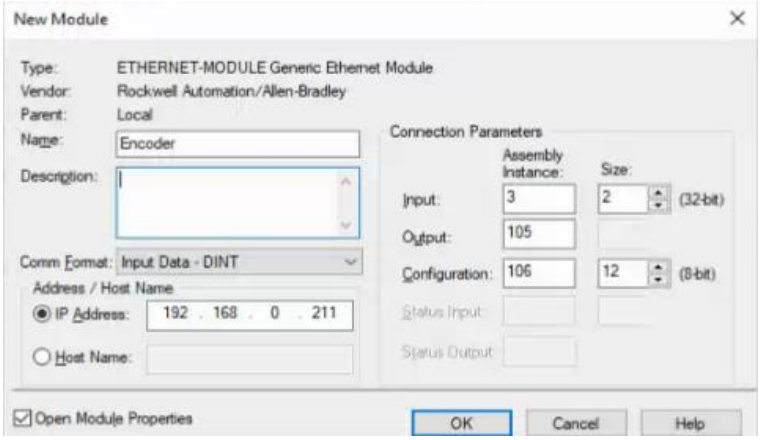



You can now change the parameters in the Controller Tags window




## 7.1.4.1. Generic Module Configuration

1. Add the Generic Module in the project



Additional settings are needed when configuring the Generic Module.
You need to make the settings like in image 2 in Controller Tags -> Monitor Tags window, inside the module you created. In this test it is Encoder:C.
The values for the parameters Encoder:C.Data[0] … Encoder:C.Data[11] need to be entered manually in „Value“ column.

| Name | Value | Force Mask | Style | Data Type | Description |
|---|---|---|---|---|---|
| ◢ Encoder:C | {...} | {...} | | AB:ETHERN... | |
| ◢ Encoder:C.Data | {...} | {...} | Hex | SINT[400] | |
| ▸ Encoder:C.Data[0] | 16#00 | | Hex | SINT | Direction Counting Toggle |
| ▸ Encoder:C.Data[1] | 16#01 | | Hex | SINT | Scaling Function Control |
| ▸ Encoder:C.Data[2] | 16#e8 | | Hex | SINT | Measuring Units per Span byte 0 (LSB) |
| ▸ Encoder:C.Data[3] | 16#03 | | Hex | SINT | Measuring Units oer Span byte 1 |
| ▸ Encoder:C.Data[4] | 16#00 | | Hex | SINT | Measuring Units per Span byte 2 |
| ▸ Encoder:C.Data[5] | 16#00 | | Hex | SINT | Measuring Units per Span byte 3 (MSB) |
| ▸ Encoder:C.Data[6] | 16#1d | | Hex | SINT | Total Measuring byte 0 (LSB) |
| ▸ Encoder:C.Data[7] | 16#09 | | Hex | SINT | Total Measuring byte 1 |
| ▸ Encoder:C.Data[8] | 16#00 | | Hex | SINT | Total Measuing byte 2 |
| ▸ Encoder:C.Data[9] | 16#00 | | Hex | SINT | Total Measuring byte 3 (MSB) |
| ▸ Encoder:C.Data[10] | 16#0f | | Hex | SINT | Velocity 0 (LSB) |
| ▸ Encoder:C.Data[11] | 16#1f | | Hex | SINT | Velocity 1 (MSB) |

In the “Description” column, there is the description of the generic data values added.
This setting imitates the parameters that are contained in a regular EDS fil

## 7.2 VLAN Configuration over a Layer 3 Switch

Some applications require the use of a Layer 3 Switch. When this switch is placed between the encoder and the PLC, these ones will be in different subnets. POSITAL’s Ethernet/IP encoders support the VLAN functionality. In order to be able to establich a connection between the PLC and the encoder, you have to make sure that the encoder and PLC are set as VLAN-Members of the same VLAN group (see setup below).
Note: only available in Unicast.

**Connections:**

| | | |
|---|---|---|
| Port 1 | PC with RSLinx Classic and RSLogix 5000 | IP 192.168.1.99 |
| Port 3 | PLC | IP 192.168.1.100 |
| Port 5 | ETAP | IP 192.168.100.50 |
| Port 6 | Encoder | IP 192.168.100.70 |

*Table 44: Example of VLAN IP Addresses Allocation*

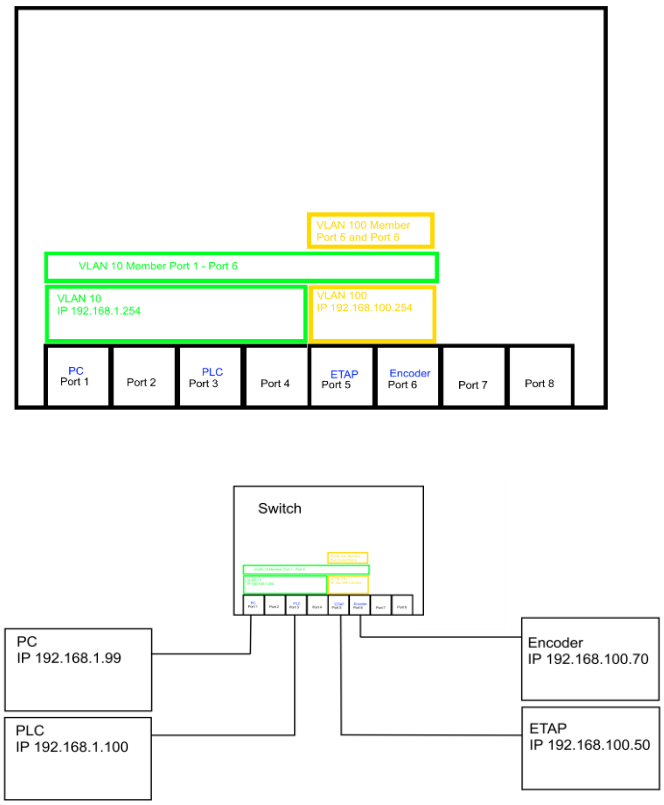



*Figure 12: Example of VLAN Setup*

# 8 FAQ

**8.1** **Problem:** I replaced a rotary encoder in the machine and now the controller cannot start the application. Additionally, the Stat LED is flashing at 4 Hz (4 flashes per second).
**Solution:** Start the BOOTP server to set the IP-Address and deactivate BOOTP. See chapter 7.1.1

**8.2** **Problem:** Cannot deactivate BOOTP.
**Background:** Firewall and/or WLAN block communication.
**Solutions:**
1. Firewall must not block Port 67 and 68
2. Deactivate WLAN and all other network cards.

**8.3** **Problem:** Error message "Invalid identifier"
**Solution:** Check that the major revision of the encoder uses the same number as the EDS file. Check the website.

**8.4 Problem:** Stat LED is flashing with 4 Hz

**Background:** After replacing a rotary encoder in the machine, the controller cannot start the application. Additionally, the Stat LED is flashing with 4 Hz

**Solution:** Start the BOOTP server to set the IP address and deactivate BOOTP.

**8.5 Problem:** After Power-up the programmed parameters are lost.

**Solution**: Use the save command to save all programmed parameters in the non-volatile memory (NVM). Only Preset is saved automatically in the NVM.

**8.6 Problem:** Parameters from Configuration tool like RSLogix overwrite the saved values of the encoder

**Answer:** Yes, that is how it is supposed to work. Please refer to FAQ 8.6 to have the parameters saved on the configuration tool.

**8.7 Problem:** How many encoders can work with one Rockwell PLC?

**Answer:** One encoder represents a traffic of only 100 Ethernet packets per second. The PLCs can manage 6,000-20,000 Ethernet packets/s. Rockwell has got two tools:

- EtherNet/IP Capacity Tool
- Integrated Architecture Builder (overdressed for analysis)

**8.8 Problem**: Rockwell PLC get error code 515 from our encoder

**Answer:**

1. Check if its cable installation was done correctly
2. Check if the network load is not too big. If so, you can use a Unicast instead of Multicast configuration.
3. If the PLC has not worked before with one of our encoders,

You can download and test our sample project from our website.

**8.9 Problem**: What is the fastest possible cycle time (RPI) on our EtherNet/IP encoder?

**Answer**: 1ms.

**8.10 Problem**: How can one set the preset value from the Control Tag Table?

**Answer**: You can download our quick start manual (under the Tool.zip folder of the product page) from our website and investigate the slides for detailed information.

**8.11 Question**: What is the default IP address of the encoder?

**Answer**: 192.168.0.250

**8.12 Question**: Is the new encoder version OCD-EEC1B backward compatible with the old version OCD-EEA1B?

**Answer:** We can only guarantee that the encoder is 100% backward compatible when an EDS file is used. When application engineers are using their own direct connections to the attributes, they can in some cases run into the problem that some attributes aren't one-to-one set the same.

## 9 Glossary

| Term | Explanation |
|---|---|
| 10Base-T | Transmission line with 10 Mbit data transmission rate |
| 100Base-T | Transmission line with 100 Mbit data transmission rate |
| Auto-crossing | Allow to use straight or crossover wiring |
| Auto-negotiation | Is an Ethernet procedure by which two connected devices choose common transmission parameters, such as speed and duplex mode |
| Baudrate | Transmission rate: it displays the transmission bits per second |
| Big Endian | Variables will use Byte 0 as Low and last Byte as High |
| Binary | Numeric system with value 0 or 1. |
| BOOTP | A UDP network protocol used by a network client to obtain its IP address automatically |
| CAT5 | Terminations for transmission rates up to 100 Mbit. |
| CIP | **C**ontrol and **I**nformation **P**rotocol |
| DHCP | **D**ynamic **H**ost **C**onfiguration Protocol is a protocol used by network devices (clients) to obtain the parameters necessary to operate in an Internet Protocol network. This protocol reduces system administration workload, allowing devices to be added to the network with little or no manual configuration. |
| DLR | **D**evice **L**evel **R**ing |
| EIP | **E**theNet**/IP** |
| EMC | **E**lectro**m**agnetic **c**ompatibility, there are rules to verifying devices. |
| ENIP | **EtherNet/IP** |
| Ethernet | Ethernet is a computer network technology based on frames. |
| Explicit Messages | Communication between an Ethernet scanner and an encoder (in our case) |
| Fast Ethernet | Transmission technology with 100 Mbit transmission rate. |
| Flash | Internal memory, saved data will be available after a power cycle. |
| Implicit Messaging | IO Connection: communication between a controller and the device |
| IP Address | Allow a logic addressing of devices in a network. |
| IP Protocol | The **I**nternet **P**rotocol is widespread in computer networks. It is the implementation of the internet layer of the TCP/IP model |
| MAC Address | Worldwide explicit address of a device. The encoder uses three MAC Addresses: one for internal interface and two for the ports. |
| Mbit | Transmission rate or baud rate, million bits per second |
| OCD | Acronym: **O**PTO**C**O**D**E, name of an encoder series manufactured by FRABA POSITAL. |
| OSI Model | The **O**pen **S**ystem **I**nterconnection reference model is an open layer model for the organization of a communication. |
| Scanner | Program to send Explicit Messages to the encoder |
| Switch | A switch is an electronic device to connect computers e.g. network segments in a local network. Unlike a hub, a switch uses stacks to avoid network collisions. |

| | |
|---|---|
| TCP | The **T**ransmission **C**ontrol **P**rotocol is a connection orientated transmission protocol, in a network. |
| UDP | **U**ser **D**atagram **P**rotocol is utilized to send data that does not need to be transferred in a reliable way. |

*Table 45: Glossary*

We do not assume responsibility for technical inaccuracies or omissions. Specifications are subject to change without notice.

## 10 Revision index

| Revision | Date | Revision |
|---|---|---|
| First release | 2020-08-09 | 1.0 |
| General small updates | 2020-09-18 | 1.1 |
| Tutorial RSLogix; Tables and Figures Indexes; 7.4. VLAN | 2020-12-02 | 1.2 |
| DHCP Information | 2024-07-25 | 1.3 |

*Table 46: Revision Index*