OPERATING INSTRUCTIONS

AFS60 S01/S02 EtherNet/IP
AFM60 S01/S02 EtherNet/IP
incl. WEB and FTP functionality

Absolute Encoder

GB

SICK
Sensor Intelligence.

8018909/1EH9/2021-12-16

Subject to change without notice

Contents

1 About this document

Please read this chapter carefully before working with this documentation and the AFS60/AFM60 EtherNet/IP Absolute Encoder.

1.1 Function of this document

These operating instructions are designed to address *the technical personnel of the machine manufacturer* or *the machine operator* in regards to correct configuration, electrical installation, commissioning, operation and maintenance of the AFS60/AFM60 EtherNet/IP Absolute Encoder.

1.2 Target group

The operating instructions are addressed at the *planners*, *developers* and *operators* of systems in which one or more AFS60/AFM60 EtherNet/IP Absolute Encoders are to be integrated. They also address people who initialize the use of the AFS60/AFM60 EtherNet/IP or who are in charge of servicing and maintaining the device.

These instructions are written for trained personnel who are responsible for the installation, mounting and operation of the AFS60/AFM60 EtherNet/IP in an industrial environment.

1.3 Information depth

These operating instructions contain information on the AFS60/AFM60 EtherNet/IP Absolute Encoder on the following subjects:

- product features
- electrical installation
- commissioning and configuration
- fault diagnosis and troubleshooting
- conformity

The operating instructions do not contain any information on the mounting of the AFS60/AFM60 EtherNet/IP. You will find this information in the mounting instructions included with the device.

They also do not contain any information on technical specifications, dimensional drawings, ordering information or accessories. You will find this information in the data sheet for the AFS60/AFM60 EtherNet/IP.

Planning and using measurement systems such as the AFS60/AFM60 EtherNet/IP also requires specific technical skills beyond the information in the operating instructions and mounting instructions. The information required to acquire these specific skills is not contained in this document.

When operating the AFS60/AFM60 EtherNet/IP, the national, local and statutory codes and regulations must be observed.

Further information

- www.odva.org
- www.ethernetip.de

1.4 Scope

These operating instructions are original operating instructions.

Note These operating instructions apply to the AFS60/AFM60 EtherNet/IP Absolute Encoder with the following type codes:

- Singleturn Encoder Basic = AFS60B-xxIx032768
- Multiturn Encoder Basic = AFM60B-xxIx015x12
- Singleturn Encoder Advanced = AFS60A-xxIx262144
- Multiturn Encoder Advanced = AFM60A-xxIx018x12

1.5 Abbreviations used

CIP Common Industrial Protocol

CMR Counts per Measuring Range

CNR_D Customized Number of Revolutions, Divisor = divisor of the customized number of revolutions

CNR_N Customized Number of Revolutions, Nominator = nominator of the customized number of revolutions

CPR Counts Per Range

DHCP Dynamic Host Control Protocol

DLR Device Level Ring

EADK EtherNet/IP Adapter Developers Kit = development environment for EtherNet/IP devices

EDS Electronic Data Sheet

EEPROM Electrically Erasable Programmable Read-only Memory

FPGA Field Programmable Gate Array = electronic component that can be programmed to provide an application-specific circuit

I/O Input and Output Data (from the point of view of the master)

IP in EtherNet/IP Industrial Protocol

IP in TCP/IP Internet Protocol

MAC Media Access Control

ODVA Open DeviceNet Vendor Association

PLC Programmable Logic Controller

TCP Transmission Control Protocol

UDP User Datagram Protocol

1.6 Symbols used

Note Refer to notes for special features of the device.

● **Red**, ☼ **Yellow**, ○ **Green** LED symbols describe the state of a diagnostics LED. Examples:

- ● **Red** The red LED is illuminated constantly.
- ☼ **Yellow** The yellow LED is flashing.
- ○ **Green** The green LED is off.

➢ Take action ... Instructions for taking action are shown by an arrow. Read carefully and follow the instructions for action.

WARNING

Warning!

A warning notice indicates an actual or potential risk or health hazard. They are designed to help you to prevent accidents.

Read carefully and follow the warning notices.

2 On safety

This chapter deals with your own safety and the safety of the equipment operators.

➢ Please read this chapter carefully before working with the AFS60/AFM60 EtherNet/IP or the machine or system in which the AFS60/AFM60 EtherNet/IP is used.

2.1 Authorized personnel

The AFS60/AFM60 EtherNet/IP Absolute Encoder must only be installed, commissioned and serviced by authorized personnel.

Note Repairs to the AFS60/AFM60 EtherNet/IP are only allowed to be undertaken by trained and authorized service personnel from SICK AG.

The following qualifications are necessary for the various tasks:

Tab. 1: Authorized personnel

Activity	Qualification
Mounting	• Basic technical training • Knowledge of the current safety regulations in the workplace
Electrical installation and replacement	• Practical electrical training • Knowledge of current electrical safety regulations • Knowledge on the use and operation of devices in the related application (e.g. industrial robots, storage and conveyor technology)
Commissioning, operation and configuration	• Knowledge on the current safety regulations and the use and operation of devices in the related application • Knowledge of automation systems (e.g. Rockwell ControlLogix Controller) • Knowledge of EtherNet/IP • Knowledge of the usage of automation software (e.g. Rockwell RSLogix)

2.2 Correct use

The AFS60/AFM60 EtherNet/IP Absolute Encoder is an instrument that is manufactured in accordance with recognized industrial regulations and meets the quality requirements as per ISO 9001:2008 as well as those of an environment management system as per ISO 14 001:2009.

An encoder is a device for mounting that cannot be used independent of its foreseen function. For this reason an encoder is not equipped with immediate safe devices.

Considerations for the safety of personnel and systems must be provided by the constructor of the system as per statutory regulations.

Due to its design, the AFS60/AFM60 EtherNet/IP can only be operated within an EtherNet/IP network. It is necessary to comply with the EtherNet/IP specifications and guidelines for setting up a EtherNet/IP network.

In case of any other usage or modifications to the AFS60/AFM60 EtherNet/IP, e.g. opening the housing during mounting and electrical installation, or in case of modifications to the SICK software, any claims against SICK AG under warranty will be rendered void.

2.3 General safety notes and protective measures

WARNING

Please observe the following procedures in order to ensure the correct and safe use of the AFS60/AFM60 EtherNet/IP!

The encoder is to be installed and maintained by trained and qualified personnel with knowledge of electronics, precision mechanics and control system programming. It is necessary to comply with the related standards covering the technical safety stipulations.

All safety regulations are to be met by all persons who are installing, operating or maintaining the device:

- The operating instructions must always be available and must always be followed.
- Unqualified personnel are not allowed to be present in the vicinity of the system during installation.
- The system is to be installed in accordance with all applicable safety regulations and the mounting instructions.
- All work safety regulations of the applicable countries are to be followed during installation.
- Failure to follow all applicable health and safety regulations may result in injury or damage to the system.
- The current and voltage sources in the encoder are designed in accordance with all applicable technical regulations.

2.4 Environmental protection

Please note the following information on disposal.

Tab. 2: Disposal of the assemblies

Assembly	Material	Disposal
Packaging	Cardboard	Waste paper
Shaft	Stainless steel	Scrap metal
Flange	Aluminium	Scrap metal
Housing	Aluminium die-cast	Scrap metal
Electronic assemblies	Various	Electronic waste

3 Product description

This chapter provides information on the special features and properties of the AFS60/AFM60 EtherNet/IP Absolute Encoder. It describes the construction and the operating principle of the device.

➢ Please read this chapter before mounting, installing or commissioning the device.

3.1 Special features

Tab. 3: Special features of the encoder variants

Properties	Singleturn Encoder Basic	Singleturn Encoder Advanced	Multiturn Encoder Basic	Multiturn Encoder Advanced
Absolute Encoder in 60 mm design	■	■	■	■
Robust nickel coded disk for harsh environments	■	■	■	■
High precision and reliability	■	■	■	■
Large ball bearing spacing of 30 mm	■	■	■	■
High level of resistance to vibration	■	■	■	■
Optimal rotational accuracy	■	■	■	■
Compact design	■	■	■	■
Face mount flange, servo flange and blind hollow shaft	■	■	■	■
15 bit singleturn resolution (1 to 32,767 steps)	■		■	
18 bit singleturn resolution (1 to 262,144 steps)		■		■
27 bit total resolution			■	
30 bit total resolution				■
12 bit multiturn resolution (1 to 4,096 revolutions)			■	■
Round axis functionality	■	■	■	■
EtherNet/IP interface (as per IEC 61784-1)	■	■	■	■
Supports the encoder profile 22h defined in the CIP (Common Industrial Protocol)	■	■	■	■
Device Level Ring (DLR)		■		■

3.2 Operating principle of the encoder

The AFS60/AFM60 EtherNet/IP acquires the position of rotating axes and outputs the position in the form of a unique digital numeric value. Optical acquisition of the rotary position value is from an internal coded disk.

The AFS60 EtherNet/IP is a singleturn encoder

Singleturn encoders are used if the absolute position of the shaft for one revolution is required.

The AFM60 EtherNet/IP is a multiturn encoder

Multiturn encoders are used if the absolute position is required for more than one shaft revolution.

3.2.1 Scaleable resolution

The steps per revolution and the total resolution can be scaled and adapted to the related application.

The steps per revolution can be scaled in integers from 1 ... 32,767 (Basic) or from 1 ... 262,144 (Advanced). The total resolution of the AFM60 EtherNet/IP must be 2^n times the steps per revolution. This restriction is not relevant if the round axis functionality is activated.

3.2.2 Round axis functionality

The encoder supports the function for round axes. During this process, the steps per revolution are set as a fraction (see section 3.5.9 on page 36). As a result, the total resolution does not have to be configured to 2^n times the steps per revolution and can also be a decimal number (e.g. 12.5)

Note The output position value is adjusted with the zero point correction, the counting direction set and the gearbox parameters entered.

Example:

A rotary table for a filling system is to be controlled. The steps per revolution are pre-defined by the number of filling stations. There are nine filling stations. For the precise measurement of the distance between two filling stations, 1000 steps are required.

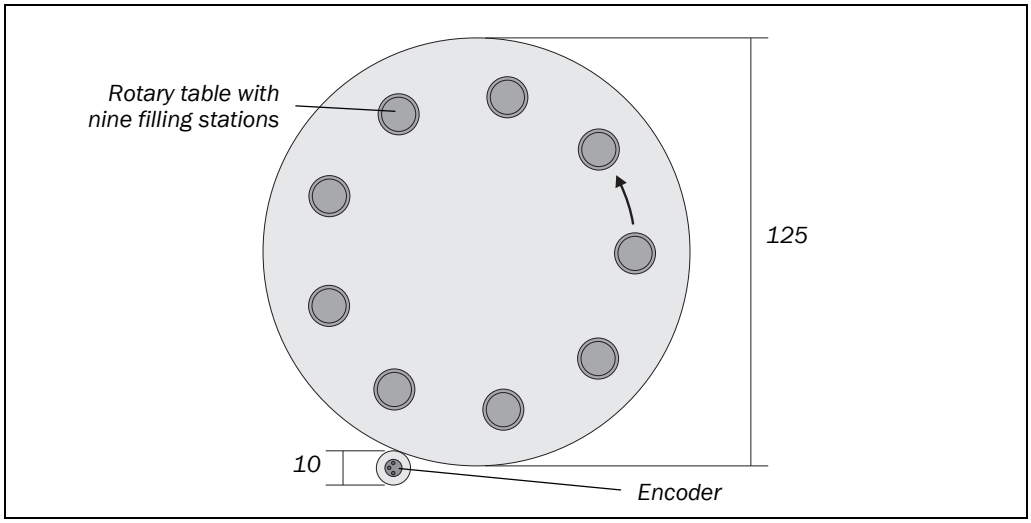


Fig. 1: Example round axis functionality for position measurement on a rotary table

The number of revolutions is pre-defined by the transmission ratio = 12.5 of the rotary table gearing.

The total resolution is then 9 × 1000 = 9000 steps, to be realized in 12.5 revolutions of the encoder. This ratio cannot be realized via the steps per revolution and the total resolution, as the total resolution is not 2^n times the steps per revolution.

The application problem can be solved using the round axis functionality. Here the steps per revolution are ignored. The total resolution as well as the nominator and divisor for the number of revolutions are configured.

9000 steps are configured as the total resolution.

For the nominator for the number of revolutions 125 is configured, 10 as the divisor ($^{125}/_{10}$ = 12.5).

After 12.5 revolutions (that is after one complete revolution of the rotary table) the encoder reaches the total resolution of 9000.

3.3 Integration in EtherNet/IP

3.3.1 EtherNet/IP architecture

EtherNet/IP and therefore also the AFS60/AFM60 EtherNet/IP use Ethernet for the transmission technology.

The network components are generally integrated into a **star topology**.

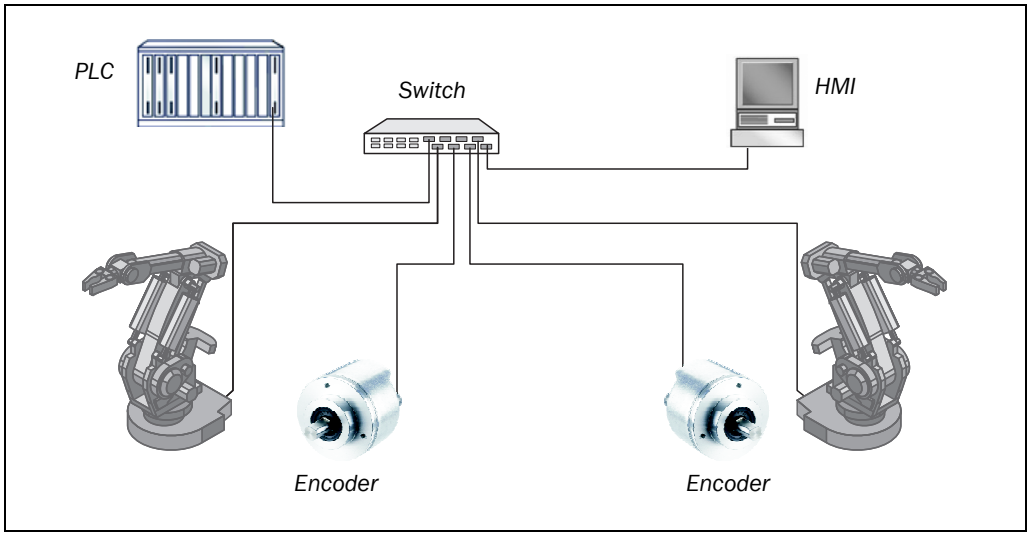


Fig. 2: Example of an EtherNet/IP network in a star topology.

The system can also be integrated in a **Device Level Ring (DLR)** in order to achieve a higher reliability and less wiring effort.

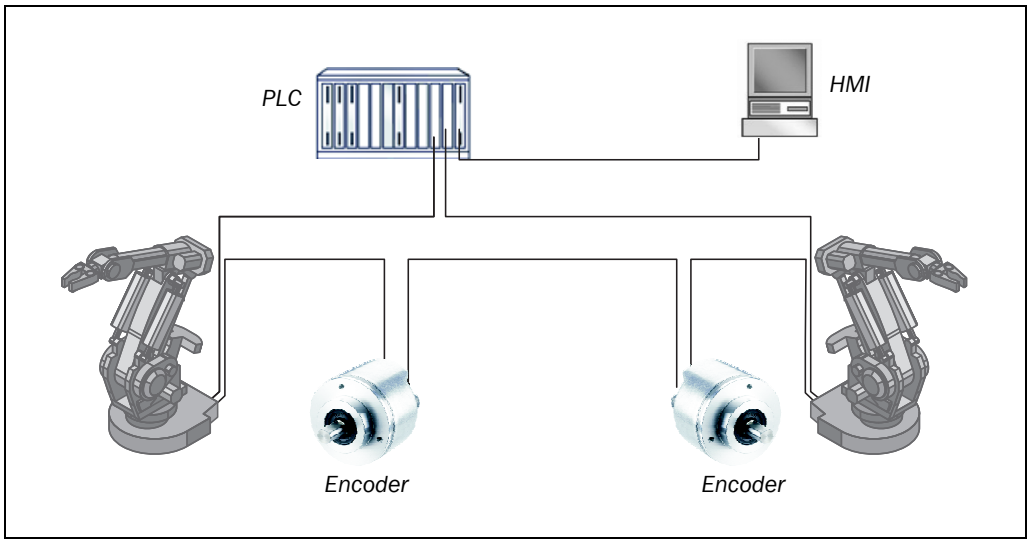


Fig. 3: Example of an EtherNet/IP network in a Device Level Ring

The AFS60/AFM60 EtherNet/IP supports Device Level Ring.

3.3.2 EtherNet/IP communication

MAC address

Each AFS60/AFM60 EtherNet/IP has a factory-assigned worldwide unique MAC address for device identification. It is used for the identification of the Ethernet node. This 6 byte device identification can not be changed and comprises the following components:

- 3 bytes manufacturer ID
- 3 bytes device ID

TCP/IP and UDP/IP

EtherNet/IP uses TCP/IP or UDP/IP for the communication.

For identification the IP address is required. A fixed address is assigned to the encoder using the decade switches or the address is obtained from a DHCP server.

If the IP address is configured fix, only the least significant byte can be configured. 192.168.1.xxx is preset permanently.

Additionally the subnet mask (default = 255.255.255.0) and if required a gateway must be configured in the network.

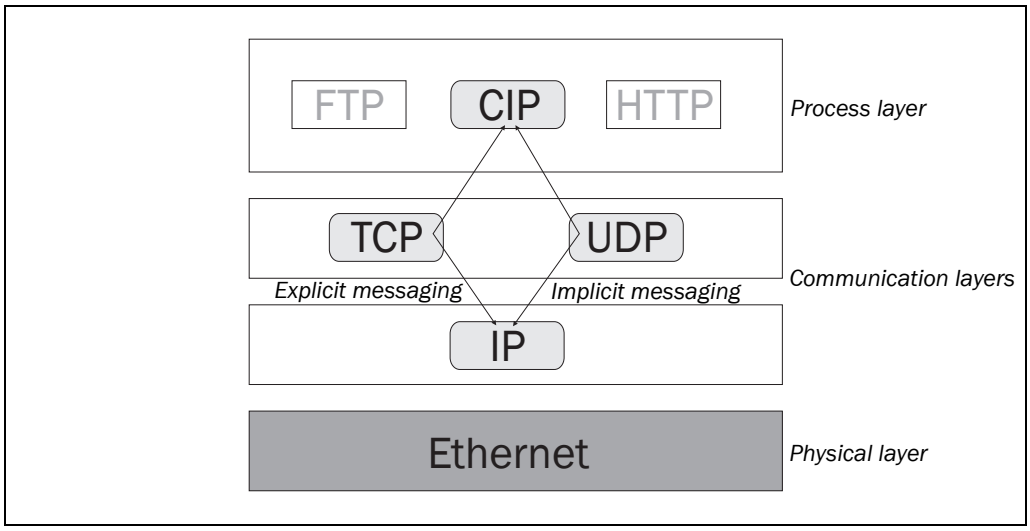
For real-time communication between the controller and the encoder in EtherNet/IP **Implicit messaging** is used. With implicit messaging, a connection is established between two devices within the CIP to transfer, e.g., I/O data such as position, velocity etc. from the encoder to the controller (see also section 3.4.3 “Position Sensor Object” on page 27). Implicit messaging uses **UDP/IP** via port 2222. As a result a fast data rate is used.

Explicit messaging is used in EtherNet/IP for communication that does **not** need to take place in real time. Explicit messaging uses **TCP/IP**, it is used e.g. to transfer parameters from the controller to the encoder (see also section 3.4.2 “Assembly Object” on page 22).

Common Industrial Protocol (CIP)

EtherNet/IP uses the CIP on the process layer. Similarly as e.g. FTP is used for the transfer of files, this protocol is used for process control.

Fig. 4: CIP and other services



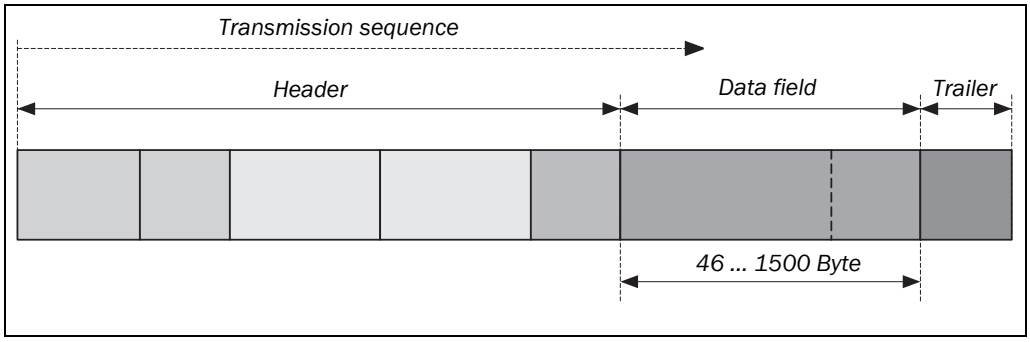
The AFS60/AFM60 EtherNet/IP meets the requirements of the EtherNet/IP protocol according to IEC 61784-1 and those of the encoder profile 22h.

The encoder is an I/O adapter in the EtherNet/IP. It receives and sends explicit messages and implicit messages either cyclic or on request (polled).

EtherNet/IP communication

EtherNet/IP is based on the standard Ethernet FRAME. This contains the Ethernet header, the Ethernet data and the Ethernet trailer. The MAC addresses of the receiver (destination address) and of the source (source address) are contained in the Ethernet header.

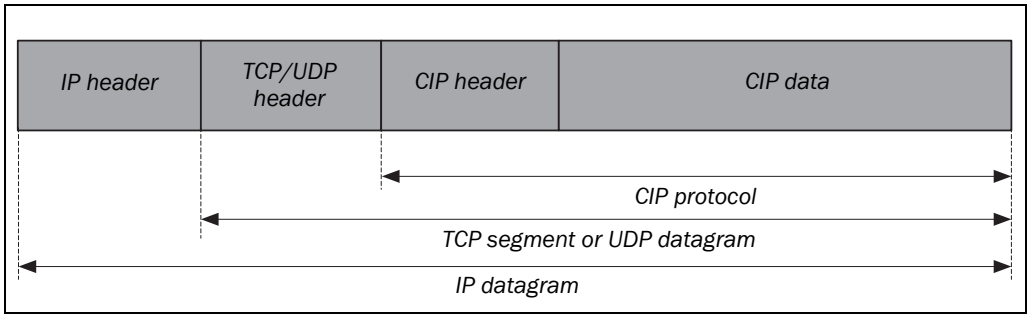
Fig. 5: Ethernet FRAME



The Ethernet data field consists of several nested protocols:

- The IP datagram is transported in the user data of the Ethernet data field.
- The TCP segment or the UDP datagram are transported in the user data of the IP datagram.
- The CIP protocol is transported in the user data of the TCP segment or of the UDP datagram.

Fig. 6: Ethernet data field



3.4 CIP object model

EtherNet/IP uses a so-called object model for network communication wherein all functions and data of a device are defined.

The most important terms are as follows:

Class A class contains related objects of a device, organized in instances.

Instance An instance consists of different attributes that describe the properties of this instance. Different instances of a class have the same services, the same behavior and the same attributes. They can, however, have different attribute values.

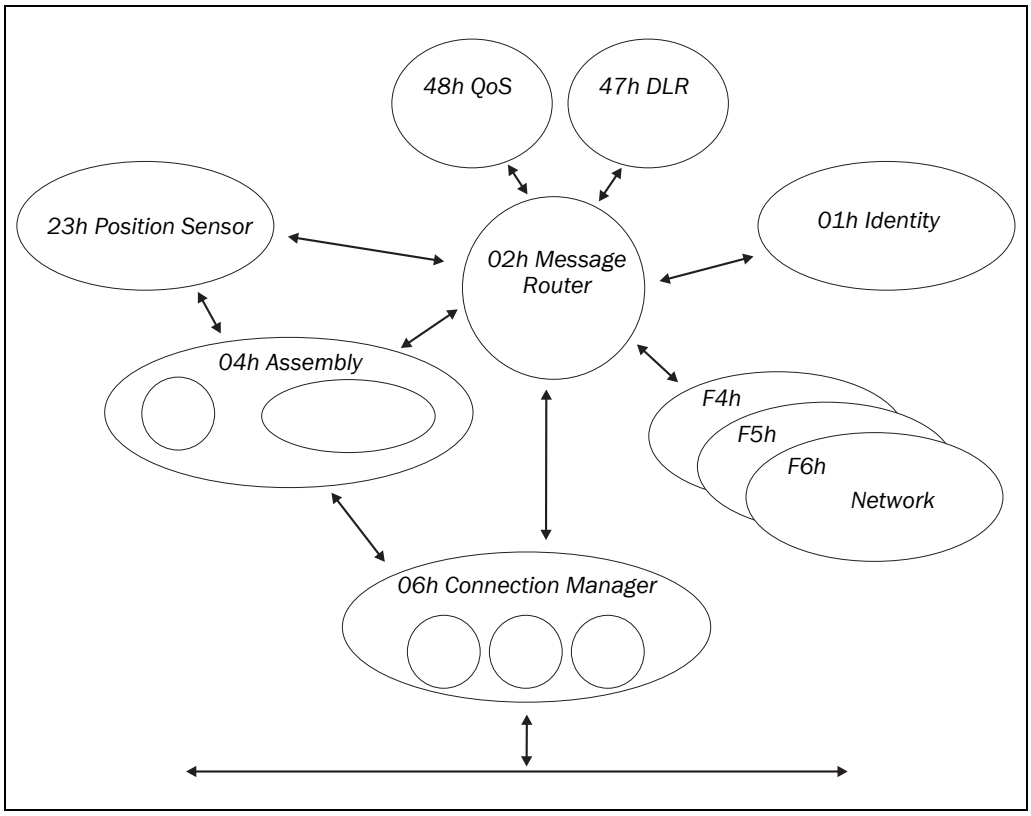
Attribute The attributes represent the data a device provides over EtherNet/IP. These include the current values of, for example, a configuration or an input. Typical attributes are configuration or status information.

Behavior The behavior defines how a device reacts as a result of external events such as changed process data or internal events such as lapsing timers.

Service Services are used to access classes or the attributes of a class or to generate specific events. These services execute defined actions such as the reading of attributes.

The AFS60/AFM60 EtherNet/IP supports the following classes of the 22h encoder profile:

Fig. 7: Supported classes



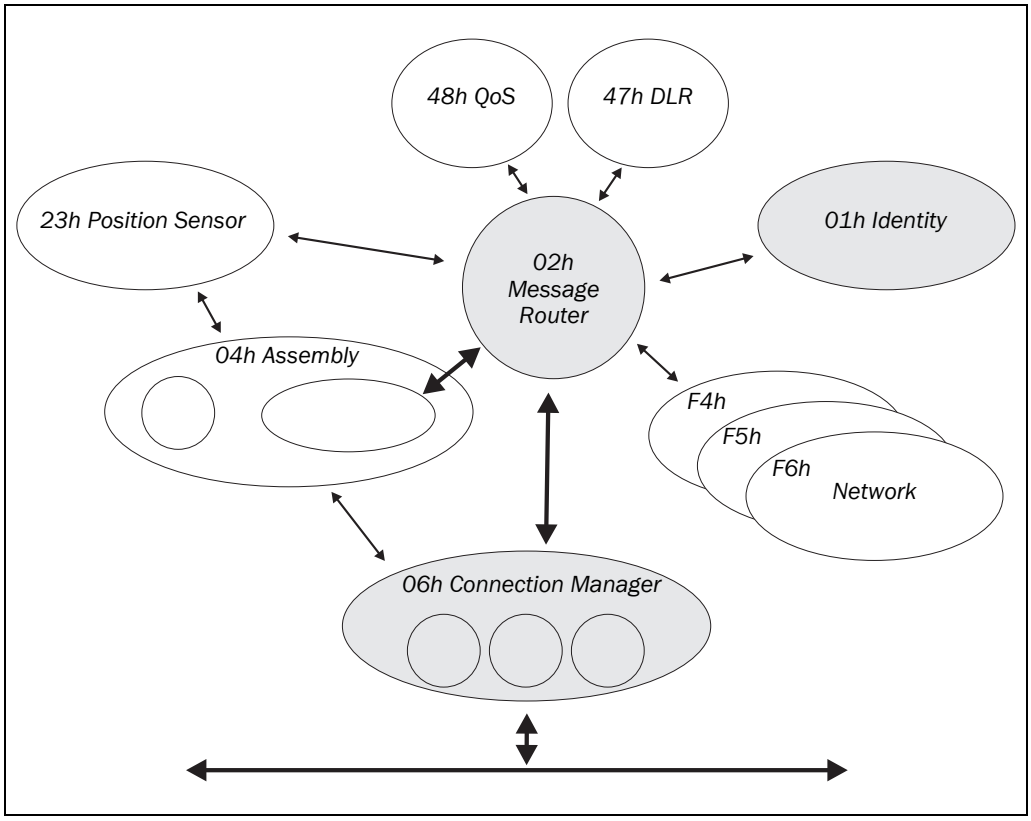
Tab. 4: Supported classes

Code	Class	Description	Access	Instances
01h	Identity Object	Includes all device specific data (e.g. ID, device type, device status etc.)	Get	1
02h	Message Router Object	Includes all supported class codes of the encoder and the maximum number of connections	Get	1
04h	Assembly Object	Assembles the data of several objects to one single object. Supplies (for example) the position value of the encoder	Get	7
06h	Connection Manager Object	Includes connection specific attributes for triggering, transport, connection type etc.	Get	1
23h	Position Sensor Object	Includes all attributes for the programming of the encoder parameters such as the scaling	Set/Get	1
F4h	Port Object	Includes the available ports, port name and node address	Get	1
F5h	TCP/IP Interface Object	Includes the attributes for TCP/IP such as IP address, subnet mask and gateway or acquisition of the IP address via DHCP or hardware switches	Set/Get	1
F6h	Ethernet Link Object	Includes connection specific attributes such as transmission speed, interface status and the MAC address	Get	3
47h	Device Level Ring (DLR) Object	Includes status attributes and configuration attributes of the DLR protocol	Get	1
48h	Quality of Service (QoS) Object	Contains mechanisms for processing data streams with different priorities	Get	1

3.4.1 Identity Object

The device information and device parameters are opened via the instances.

Fig. 8: Connections for the Identity Object



Tab. 5: Class services of the Identity Object

Instance	Service	Description
01h	Get_Attribute_All	Returns the values of all attributes
0Eh	Get_Attribute_Single	Returns the values of one attribute

Tab. 6: Class attributes of the Identity Object

Number	Access	Description	Data type	Default value
1	Get	Object revision index	UINT	0001h
2	Get	Maximum number of object instances in this class	UINT	0001h
3	Get	Number of object instances in this class	UINT	0001h
4	Get	Optional attribute list	STRUCT	–
6	Get	Highest existing class attribute ID	UINT	0007h
7	Get	Highest implemented instance attribute	UINT	0067h

Note Class attribute 5 is not implemented.

Tab. 7: Instance services of the Identity Object

Instance	Service	Description
01h	Get_Attribute_All	Returns the values of all attributes
0Eh	Get_Attribute_Single	Returns the values of one attribute
05h	Reset	Resets the device: 0 = The device is re-initialized (power on). 1 = The device is re-initialized (power on) and reset to the factory settings.

Tab. 8: Instance attributes of the Identity Object

ID	Access	Name	Description	Data type	Default value
1 **01h**	Get	Vendor ID	Vendor ID 0328h = SICK AG	UINT	0328h
2 **02h**	Get	Device Type	Device profile 22h = Encoder	UINT	0022h
3 **03h**	Get	Product Code	Vendor specific product code 03h: Singleturn 04h: Multiturn	UINT	
4 **04h**	Get	Revision	Contains the firmware revision number in the format XX.XX	STRUCT	
	Get	Major Revision	First part of the revision number, e.g. 01 (depending on the release)	UINT	0001h
	Get	Minor Revision	Last part of the revision number, e.g. 02 (depending on the release)	UINT	0002h
5 **05h**	Get	Status	Device status flags	WORD	See Tab. 9
6 **06h**	Get	Serial Number	Serial number in the format YY.WW.xxxx Y = Year W = Week x = Sequential number E.g. 12.25.1234 (depending on the release)	UDINT	12251234
7 **07h**	Get	Product Name	Product name	Short_ String	1 Channel EtherNet/IP Encoder

ID	Access	Name	Description	Data type	Default value
100 64h	Get	Vendor	EADK version (EtherNet/IP Adapter Developers Kit) (e.g. V4.1.0)	UDINT	00040100h
101 65h	Get	Vendor	Firmware version in the FPGA (e.g. 1.1.0)	UDINT	00010100h
102 66h	Get	Vendor	Supported ports of the FPGA 9 = 2 ports	UDINT	00000009h
103 67h	Get	Vendor	Hardware version	UDINT	00000101h

Tab. 9: Bits of the instance attribute "Status"

Bit	Name	Description	Default value
0	Owned	0 = No connection to the master 1 = Connection to the master established	0
1	–	Reserved	0
2	Configured	0 = Device with standard configuration 1 = No standard configuration	0
3	–	Reserved	0
4 ... 7	Extended Device Status Field	Vendor specific status bits	See Tab. 10
8	Minor Recoverable Status	0 = No error 1 = Recoverable error (device not in error status)	0
9	Minor Unrecoverable Status	0 = No error 1 = Recoverable error (device not in error status)	0
10	Major Recoverable Status	0 = No serious error 1 = Recoverable serious error (device in error status)	0
11	Major Unrecoverable Status	0 = No serious error 1 = Unrecoverable serious error (device in error status)	0
12 ... 15	–	Reserved	0000

Tab. 10: Bits 4 to 7 of the instance attribute “Status”

Possible combinations Bit 4 ... 7	Description
0000	Device in self test
0001	Firmware update in progress
0010	At least one connection error
0011	No I/O connection established
0100	Configuration in non-volatile memory (EEPROM) failed
0101	Serious error, bit 10 or bit 11 = 1
0110	At least one connection in the “Run” operating mode
0111	At least one connection exists, all in “Idle” operating mode
1000 ... 1111	Reserved

3.4.2 Assembly Object

The Assembly Object allows assembling of data attributes of other objects in one single object. The AFS60/AFM60 EtherNet/IP supports only static assemblies of attributes. For this reason the number of instances is fixed.

Tab. 11: Class services of the Assembly Object

Instance	Service	Description
01h	Get_Attribute_All	Returns the values of all attributes
0Eh	Get_Attribute_Single	Returns the values of one attribute

Tab. 12: Class attributes of the Assembly Object

Number	Access	Description	Data type	Default value
1	Get	Object revision index	UINT	0002h
2	Get	Maximum number of object instances in this class	UINT	0067h
3	Get	Number of object instances in this class	UINT	0007h
6	Get	Highest existing class attribute ID	UINT	0007h
7	Get	Highest implemented instance attribute	UINT	0067h

Note Class attributes 4 and 5 are not implemented.

The encoder supports only “Input” and “Listen Only” connections.

Tab. 13: Instance attributes of the Assembly Object

Number	Connection	Description	Bits	Bytes
1	Input	Position value	32	4
2	Input	Position value Warning and alarm flags	32 8	5
3	Input	Position value Velocity	32 32	8
4 ... 5	–	–	–	–
101	Input	Error Position value	32 32	8
102	Input	Error Position value Warning and alarm flags	32 32 8	9
103	Input	Error Position value Velocity	32 32 32	12
100	Configuration “Input-only”	Configuration data	192/0	24/0
110	Configuration “Listen-only”	Dummy instance for the configuration data of a “Listen-only” connection	0	0

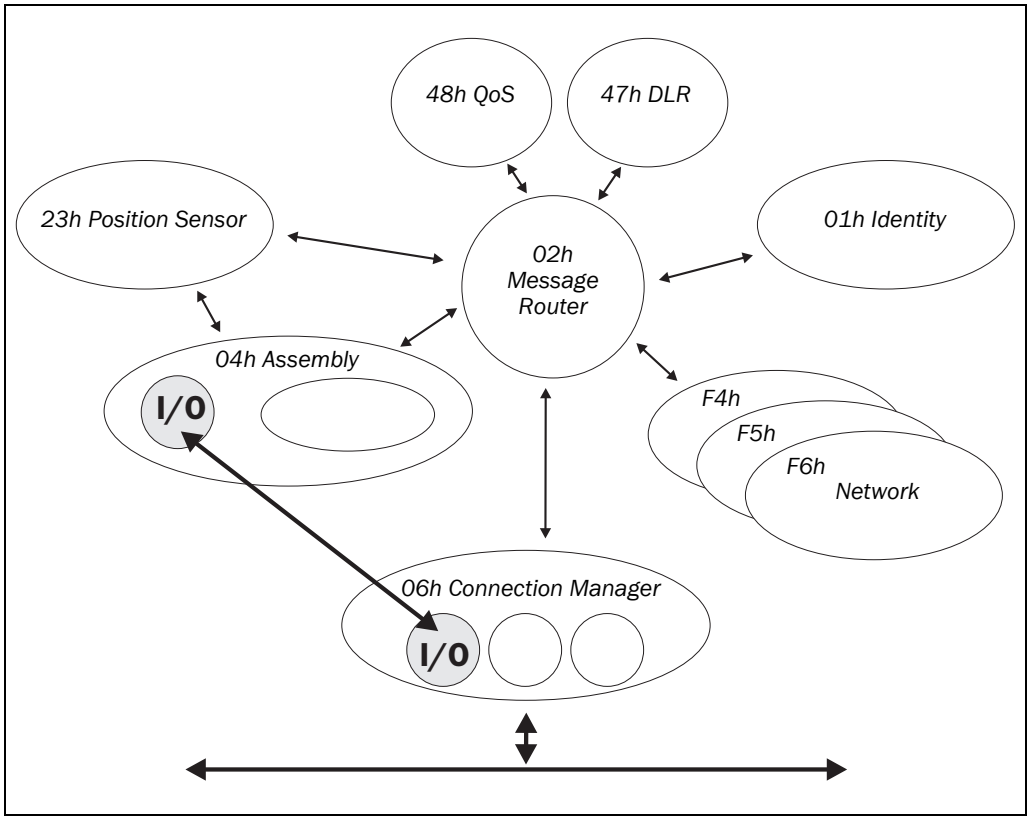
Notes

- Instance attributes 4 and 5 from the encoder profile 22h are not implemented.
- Instance attributes 100 to 110 are vendor specific attributes.

I/O assembly

The I/O data are retrieved/output via instances.

Fig. 9: Connections for the I/O assembly



Tab. 14: Data format of the attributes of the I/O assembly

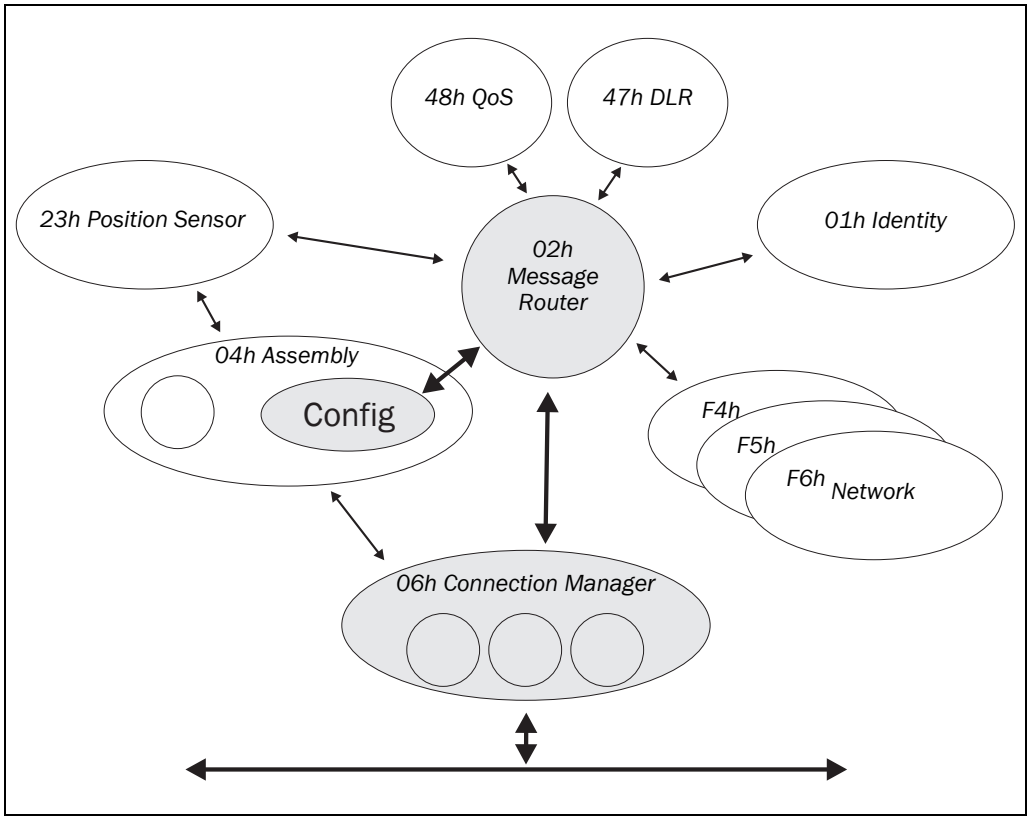
Instance	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	0	Position value (least significant byte)							
	1	Position value							
	2	Position value							
	3	Position value (most significant byte)							
2	0	Position value (least significant byte)							
	1	Position value							
	2	Position value							
	3	Position value (most significant byte)							
	4							Warning	Alarm
3	0	Position value (least significant byte)							
	1	Position value							
	2	Position value							
	3	Position value (most significant byte)							
	4	Velocity value (least significant byte)							
	5	Velocity value							
	6	Velocity value							
	7	Velocity value (most significant byte)							

Instance	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
101	0	Fault header (least significant byte, see Tab. 25 on page 67)							
	1	Fault header							
	2	Fault header							
	3	Fault header (most significant byte)							
	4	Position value (least significant byte)							
	5	Position value							
	6	Position value							
	7	Position value (most significant byte)							
102	0	Fault header (least significant byte)							
	1	Fault header							
	2	Fault header							
	3	Fault header (most significant byte)							
	4	Position value (least significant byte)							
	5	Position value							
	6	Position value							
	7	Position value (most significant byte)							
	8							Warning	Alarm
103	0	Fault header (least significant byte, see Tab. 25 on page 67)							
	1	Fault header							
	2	Fault header							
	3	Fault header (most significant byte)							
	4	Position value (least significant byte)							
	5	Position value							
	6	Position value							
	7	Position value (most significant byte)							
	8	Velocity value (least significant byte)							
	9	Velocity value							
	10	Velocity value							
	11	Velocity value (most significant byte)							

Configuration assembly

The encoder can be configured via the configuration assembly.

Fig. 10: Connections for the configuration assembly



Tab. 15: Data format for the attributes for the configuration assembly

Instance	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
100	0	Not used							
	1	Not used							
	2	Not used							
	3	Not used							
	4	Counts (steps) per revolution CPR (least significant byte)							
	5	CPR							
	6	CPR							
	7	CPR (most significant byte)							
	8	Total resolution CMR (least significant byte)							
	9	CMR							
	10	CMR							
	11	CMR (most significant byte)							
	12	Not used							cw/ ccw[1]
	13	Not used							scf[2]
	14	Not used							raf[3]
	15	Not used							
	16	Nominator for the number of revolutions CNR_N (least significant byte)							
	17	CNR_N							
	18	CNR_N							
	19	CNR_N (most significant byte)							
	20	Divisor for the number of revolutions CNR_D (least significant byte)							
	21	CNR_D							
	22	CNR_D							
	23	CNR_D (most significant byte)							
	24	Velocity measuring unit (least significant byte)							
	25	Velocity measuring unit (most significant byte)							
	26	Not used							
	27	Not used							

Notes

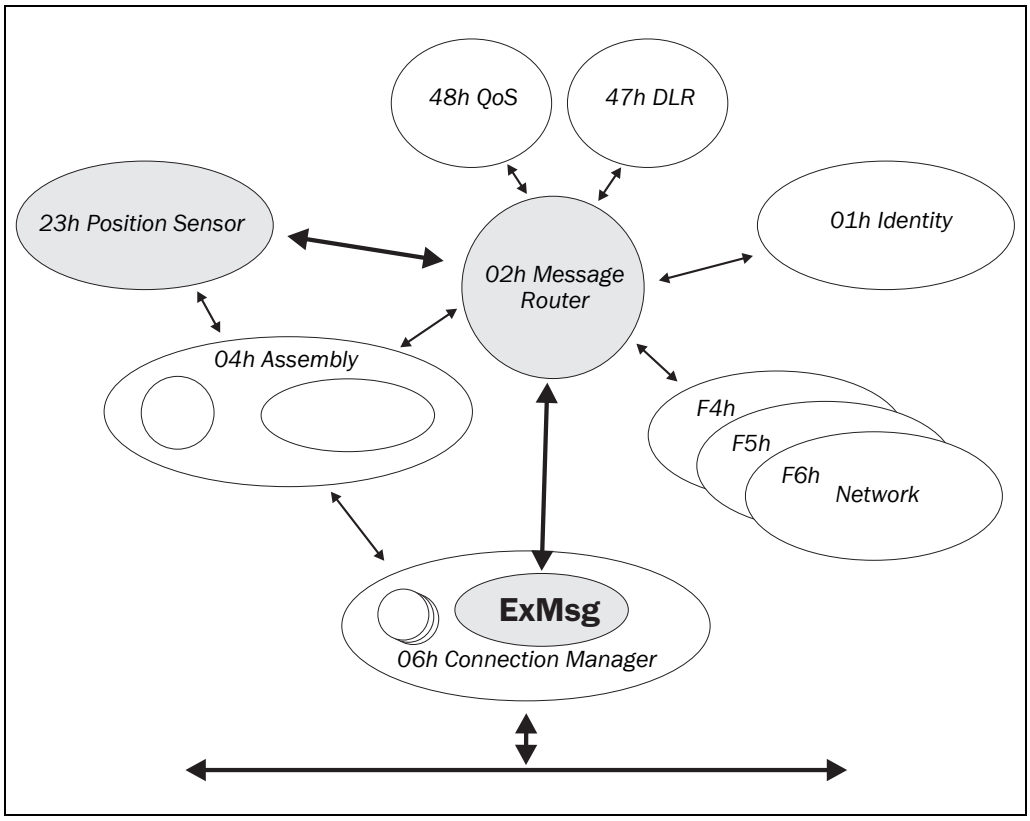
- The structure of the configuration assembly is fixed.
- During the initialization of the encoder, it reads the data from the control system.
- The “Heartbeat connection point” for input connections of the PLC, i.e. for the encoder output, must be set to 198 (see Fig. 28 on page 45).
- The “Heartbeat connection point” for listen-only connections must be set to 199.

[1] cw = clockwise.
ccw = counterclockwise.
[2] scf = scaling function.
[3] raf = round axis functionality.

3.4.3 Position Sensor Object

The Position Sensor Object contains all the attributes of the encoder. All parameters can be retrieved or set using explicit messages.

Fig. 11: Connections for explicit messages to the Position Sensor Object



Tab. 16: Class services of the Position Sensor Object

Instance	Service	Description
05h	Reset	Resets the encoder
0Eh	Get_Attribute_Single	Returns the values of one attribute
10h	Set_Attribute_Single	Sets the values of an attribute
15h	Restore	Restores all parameters last saved in non-volatile memory
16h	Save	Saves all parameters in the non-volatile memory

Tab. 17: Class attributes of the Position Sensor Object

Number	Access	Description	Data type	Default value
1	Get	Object revision index	UINT	0002h
2	Get	Maximum number of object instances in this class	UINT	0001h
3	Get	Number of object instances in this class	UINT	0001h
4	Get	Optional attribute list	STRUCT	–
5	Get	Optional services list	STRUCT	–
6	Get	Highest existing class attribute ID	UINT	0064h
7	Get	Highest implemented instance attribute	UINT	007Ah
100	Get	Serial number	Array	AFM_aa.bb. dd.mm.yy

Tab. 18: Instance services of the Position Sensor Object

Instance	Service	Description
01h	Get_Attribute_All	Returns the values of all attributes
0Eh	Get_Attribute_Single	Returns the values of one attribute

Tab. 19: Instance attributes of the Position Sensor Object

ID	Access	V/NV[4)]	Name	Description	Data type	Min. Max. (Default value)
1 **01h**	Get	V	Number of Attributes	Number of attributes in this class	UINT	0000h 003Eh
2 **02h**	Get	V	Attribute list	List of the supported attributes	Array of Bytes	–
10 **0Ah**	Get	V	Position Value Signed	Current position value	DINT	–
11 **0Bh**	Get	NV	Position Sensor Type	01h = Singleturn 02h = Multiturn	UINT	0001h 0002h (0002h)
12 **0Ch**	Set	NV	Direction Counting	Code sequence 0 = Clockwise 1 = Counterclockwise	BOOL	(0)
13 **0Dh**	Set	NV	Commissioning Diagnostic Control	Encoder self-test 0 = Off 1 = On	BOOL	(0)
14 **0Eh**	Set	NV	Scaling Function Control	Scaling 0 = Off 1 = On	BOOL	(0)

[4)] V = volatile, NV = non-volatile.

ID	Access	V/NV[4]	Name	Description	Data type	Min. Max. (Default value)
15 **0Fh**	Set	NV	Position Format	Format of the position measurement 1001h = Steps	ENG UINT	(1001h)
16 **10h**	Set	NV	Counts per Range	Number of steps per revolution	UDINT	00000001h 00040000h (00040000h)
17 **11h**	Set	NV	Total Measuring Range	Total resolution	UDINT	00000001h (2^n × ID16)
18 **12h**	Set	NV	Position Measuring Increment	Minimum resolution (always 1)	UDINT	00000001h 00000001h
19 **13h**	Set	NV	Preset Value	Preset value	DINT	00000000h 2^n × ID17 –1 (00000000h)
21 **15h**	Get	NV	Position Status Register	Indicates whether and how the limit set by ID22 and 23 is dropped below/exceeded. Bit 0 = Out of range Bit 1 = Over range Bit 2 = Under range Bit 3 ... 7 = Reserved	Byte	(00h)
22 **16h**	Set	NV	Position Low Limit	Lower limit for the position	DINT	00000000h 3FFFFFFFh (00000000h)
23 **17h**	Set	NV	Position High Limit	Upper limit for the position	DINT	00000000h 3FFFFFFFh (3FFFFFFFh)
24 **18h**	Get	V	Velocity Value	Current velocity. The format is determined by ID25 and 26.	DINT	00000000h XXXXXXXXh[5]
25 **19h**	Set	NV	Velocity Format	Velocity unit 1F04h = counts/s 1F05h = counts/ms 1F0Eh = turns/s 1F0Fh = turns/min 1F10h = turns/h	ENG UINT	(1F04h)
26 **1Ah**	Set	NV	Velocity Resolution	Minimum resolution of the velocity measurement	DUINT	(00000001h)

[5] The maximum velocity is dependent on the mechanical interface used, “solid shaft” or “blind hollow shaft” (see data sheet).

ID	Access	V/NV[4)]	Name	Description	Data type	Min. Max. (Default value)
27 **1Bh**	Set	NV	Minimum Velocity Setpoint	Minimum/maximum velocity. If the velocity drops below/exceeds this value, the warning flag (ID47) is set.	DINT	(00000000h)
28 **1Ch**	Set	NV	Maximum Velocity Setpoint		DINT	(3FFFFFFFh)
29 **1Dh**	Get	V	Acceleration value	Current acceleration. The format is determined by ID30 and 31.	DINT	00000000h FFFFFFFFh
30 **1Eh**	Set	NV	Acceleration format	Acceleration unit 0810h = counts/ms^2 0811h = counts/s^2 0813h = turns/s^2	ENG UINT	(0810h)
31 **1Fh**	Set	NV	Acceleration resolution	Minimum resolution of the acceleration measurement	DUINT	(1)
32 **20h**	Set	NV	Minimum Acceleration Setpoint	Minimum/maximum acceleration. If the acceleration drops below/exceeds this value, the warning flag (ID47) is set.	DINT	(0)
33 **21h**	Set	NV	Maximum Acceleration Setpoint		DINT	(3FFFFFFFh)
41 **29h**	Get	V	Operating Status	Operating status of the encoder Bit 0: Direction 0 = Upward counting 1 = Downward counting Bit 1: Scaling 0 = Off 1 = On Bit 2 ... 4: Reserved Bit 5: Diagnostics on/off 0 = Off 1 = On Bit 6, 7: Reserved	Byte	
42 **2Ah**	Get	NV	Physical Resolution Span	Physical resolution per revolution Basic = 15 Bit Advanced = 18 Bit	UDINT	00000000h 0003FFFFh (8000h) (40000h)

ID	Access	V/NV[4)]	Name	Description	Data type	Min. Max. (Default value)
43 2Bh	Get	NV	Physical Resolution Number of Span	Physical number of revolutions Singleturn = 0001h Multiturn = 1000h	UINT	(0001h) or (1000h)
44 2Ch	Get	V	Alarms	Bit field with flags for alarms and errors (see Tab. 26 on page 68)	WORD	–
45 2Dh	Get	NV	Supported Alarms	Supported alarms and errors	WORD	3003h
46 2Eh	Get	V	Alarm flag	0 = No alarm/error 1 = Alarm/error	BOOL	–
47 2Fh	Get	V	Warnings	Bit field with flags for warnings (see Tab. 27 on page 69)	WORD	–
48 30h	Get	NV	Supported Warnings	Supported warnings	WORD	67C3h
49 31h	Get	V	Warning flag	0 = No warning 1 = Warning	BOOL	–
50 32h	Get	NV	Operating Time	Saved operating time in 0.1h = 6 min	UDINT	0
51 33h	Get	NV	Offset Value	Offset value is calculated on the initialization of the preset function	DINT	00000000h
100 64h	Get	V	Temperature Value	Actual temperature with ±5° accuracy –40 to +100 °C or –40 to +212 °F	INT	F060h 2710h
101 65h	Set	NV	Temperature Value Format	1200h = °C (Celsius) 1201h = °F (Fahrenheit)	ENG UINT	(1200h)
102 66h	Set	NV	Temperature Resolution	Lowest resolution for the temperature (°C/100 or °F/100)	UDINT	(00000064h)
103 67h	Set	NV	Minimum Temperature Setpoint	Minimum/maximum temperature. If the temperature drops below/exceeds this value, the warning flag (ID47) is set.	INT	F060h – (F060h = –4,000)
104 68h	Set	NV	Maximum Temperature Setpoint		INT	– 2710h (2710h = +10,000) or (52D0h = +21,200)

ID	Access	V/NV[4)]	Name	Description	Data type	Min. Max. (Default value)
105 69h	Get	V	Fault header	See Tab. 25 on page 67	DWORD	(00000000h)
106 6Ah	Set	V	Special Encoder Function-alities	Bit field with flags for special encoder functions Bit 0: Slave Sign of Life (on/off) Bit 1 ... 7: Not used Bit 8 ... 15: Update factor (1 ... 127) Bit 16 ... 31: Not used	DWORD	(00000500h)
107 6Bh	Get	NV	Encoder Motion Time	Saved motion time in seconds (is increased in case of movement)	UDINT	–
108 6Ch	Get	NV	Encoder Operating Time	Saved operating time in seconds (is increased as soon as the encoder is in operation)	UDINT	–
109 6Dh	Get	NV	Max. Velocity	Highest velocity that the encoder has reached since start-up in counts/ms	UDINT	–
110 6Eh	Get	NV	Max. Acce-leration	Highest acceleration that the encoder has reached since start-up in counts/ms^2	UDINT	–
111 6Fh	Get	NV	Max. Temp	Highest operating temperature saved in C°/100	UDINT	–4,000
112 70h	Get	NV	Min. Temp	Lowest operating temperature saved in C°/100	UDINT	10,000
113 71h	Get	NV	Number of Start-ups	Number of times the encoder has been commissioned (powered on)	UDINT	–
114 72h	Get	V	LED Current Value	Actual internal LED current of the sensor in µA	UINT	200 25,000 (0)
115 73h	Get	NV	Max. Current Value	Maximum internal LED current for the sensor in µA	UINT	200
116 74h	Get	NV	Min. Current Value	Minimum internal LED current for the sensor in µA	UINT	25,000

ID	Access	V/NV[4)]	Name	Description	Data type	Min. Max. (Default value)
117 **75h**	Get	V	Direction change counter	The counter increments if the encoder changes direction of rotation.	UDINT	0
118 **76h**	Get	V	Rotation counter forward	The counter is increased if the encoder moves clockwise	UDINT	0
119 **77h**	Get	V	Rotation counter backwards	The counter is increased if the encoder moves counterclockwise	UDINT	0
120 **78h**	Get	V	Power supply voltage	Current operating voltage in mV	UINT	9,500 30,500 (24,000)
121 **70h**	Get	V	Max. power supply voltage	Maximum operating voltage in V (saved in EEPROM)	UINT	0 33 (24,000)
122 **7Ah**	Get	V	Preset Offset Value	Offset value calculated from the preset value[6)]	DINT	(00000000)
125 **7Dh**	Set	NV	Endless Shaft Functionality = round axis functionality	Activates round axis functionality 0 = Off 1 = On	BOOL	(0)
126 **7Eh**	Set	NV	Number of Rotations, nominator	Nominator for the number of revolutions	UDINT	1 2,048 (2,048)
127 **7Fh**	Set	NV	Number of Rotations, divisor	Divisor for the number of revolutions	UDINT	1 65,535 (1)

[6)] With normal scaling = physical position; with round axis functionality = physical position + Range Offset.

3.5 Configurable functions

The AFS60/AFM60 EtherNet/IP is configured with the aid of automation software (e.g. Rockwell RSLogix) via explicit messaging. Explicit messaging is used in EtherNet/IP for communication that does **not** need to take place in real time. Explicit messaging uses TCP.

EDS file

The EDS file (electronic data sheet) contains all the information related to the parameters as well as the operating modes of the AFS60/AFM60 EtherNet/IP. The EDS file is integrated using the EtherNet/IP network configuration tool to be able to configure and place in operation the AFS60/AFM60 EtherNet/IP.

3.5.1 IP address

For identification of the AFS60/AFM60 EtherNet/IP in the EtherNet/IP, the IP address is required. This address is acquired for the encoder from a DHCP server or a fixed entry made using the decade switches.

Acquiring the IP address from a DHCP server

The AFS60/AFM60 EtherNet/IP is set from the factory to obtain the IP address from a DHCP server. For this purpose the three decade switches under the screw cover on the AFS60/AFM60 EtherNet/IP (see Fig. 13 on page 37) must be set to 255 ... 887 or to 889 ... 999.

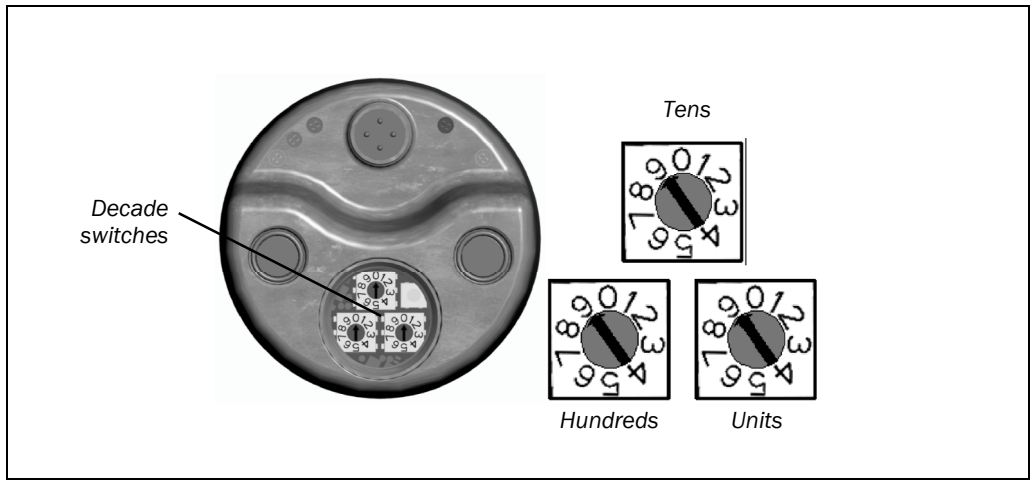
Setting the IP address

If the IP address is configured fix, only the least significant byte (1 ... 254) can be configured. 192.168.1.xxx is preset permanently. The subnet mask is then also fixed at 255.255.255.0, the gateway address 0.0.0.0.

To set a fixed address use the three decade switches (see Fig. 12). The decade switches are located under the screw cover of the AFS60/AFM60 EtherNet/IP (see Fig. 13 on page 37).

- Set the hundreds for the address using the left decade switch.
- Set the decades for the address using the center decade switch.
- Set the units for the address using the right decade switch.

Fig. 12: Decade switches



Note Do not set the address to 888 or 000, this setting will reset the encoder to the factory settings or the last settings saved in the non-volatile memory (see section 3.5.12 on page 37).

3.5.2 Slave Sign of Life

The AFS60/AFM60 EtherNet/IP supports Slave Sign of Life functionality.

It is transferred in bit 30 of the fault header. It is used so that the control system can determine whether the encoder is in operation, even if it is not providing any position data (standstill).

The bit changes its value at the update cycle configured.

The update cycle is formed from the Requested Packed Interval (RPI) and an update factor. The RPI can be between 2 and 750 ms:

update cycle = RPI × update factor (default value = 5)

The update factor is defined using attribute 106 in the Position Sensor Object (see Tab. 19 on page 28).

The value supported is dependent on the RPI time for the encoder connection. The minimum update cycle should be twice as long as the RPI (with RPI = 750 ms therefore 1500 ms).

3.5.3 Code sequence

The code sequence defines the direction of rotation, viewed on the shaft, in which the position value increases.

- clockwise = increasing position value on clockwise revolution of the shaft
- counterclockwise = increasing position value on counterclockwise revolution of the shaft

3.5.4 Scaling

Scaling makes it possible to scale the steps per revolution and the total resolution.

Note Only if the parameter **Scaling** is configured to **Enable** are the values entered for the steps per revolution and the total resolution applied.

3.5.5 Steps per revolution x

The resolution of the AFS60/AFM60 EtherNet/IP Basic is max. 32,768 steps per revolution. Its resolution can be scaled from 1 ... 32,768 as an integer.

The resolution of the AFS60/AFM60 EtherNet/IP Advanced is max. 262,144 steps per revolution. The resolution can be scaled from 1 ... 262,144 as an integer.

Note The parameter is not used if the round axis functionality (see 3.5.9 on page 36) is activated.

3.5.6 Total resolution/measuring range x

The total resolution, that is the measuring range of the AFS60/AFM60 EtherNet/IP, is max. 134,217,728 (Basic) or 1,073,741,824 (Advanced) steps. The total resolution must be 2^n times the steps per revolution.

Tab. 20: Examples for total resolution

Steps per revolution	n	Total resolution
1,000	3	8,000
8,179	5	261,728
2,048	11	4,194,304

Note This restriction is not relevant if the round axis functionality (see 3.5.9 on page 36) is activated.

3.5.7 Preset

The preset function is used to set the encoder to a predefined start position. With the aid of a preset value the encoder can be set to any position within the measuring range.

The preset value can be set in the following manner:

- using the preset push-button
- using an acyclic explicit message
  During this process the preset value is transferred as an attribute (ID19) of the Position Sensor Object.

Note

➢ Only set a preset value when the encoder is at standstill.

WARNING

Immediately after triggering the preset function, check whether there is a hazard from the machine or system in which the encoder is integrated!

The preset function results in an immediate change in the position value output by the encoder. This change could cause an unexpected movement that may result in a hazard for persons or damage to the system or other items.

3.5.8 Velocity measuring unit

Using this parameter you can define the units in which the velocity is transmitted.

Possible units are:

- counts/s
- counts/ms
- turns/s
- turns/min
- turns/h

The factory setting is **turns/min**.

3.5.9 Round axis functionality

The round axis functionality removes the restriction that the total resolution must be 2^n times the steps per revolution. The shaft is considered as an **endless shaft**.

The steps per revolution are not configured directly, instead the nominator and divisor for the number of revolutions are defined.

The total measuring range can be scaled from 1 ... 1,073,741,824 as an integer.

3.5.10 Number of revolutions, nominator for the round axis functionality

The nominator can be scaled from 1 ... 2,048 as an integer. The default factory setting for the nominator is 2,048.

3.5.11 Number of revolutions, divisor for the round axis functionality

The divisor can be scaled from 1 ... 65,535 as an integer. The default factory setting for the divisor is 1.

3.5.12 Resetting the configuration

There are two possibilities to reset the configuration.

You can reset the configuration of the AFS60/AFM60 EtherNet/IP to the factory settings.

➢ Set the IP address to xxx.xxx.xxx.888 (see section 3.5.1 “IP address” on page 34).

The next time the encoder is started up, all parameters are reset to the factory settings.

You can set the configuration of the AFS60/AFM60 EtherNet/IP to the settings last saved in the non-volatile memory.

➢ Set the IP address to xxx.xxx.xxx.000.

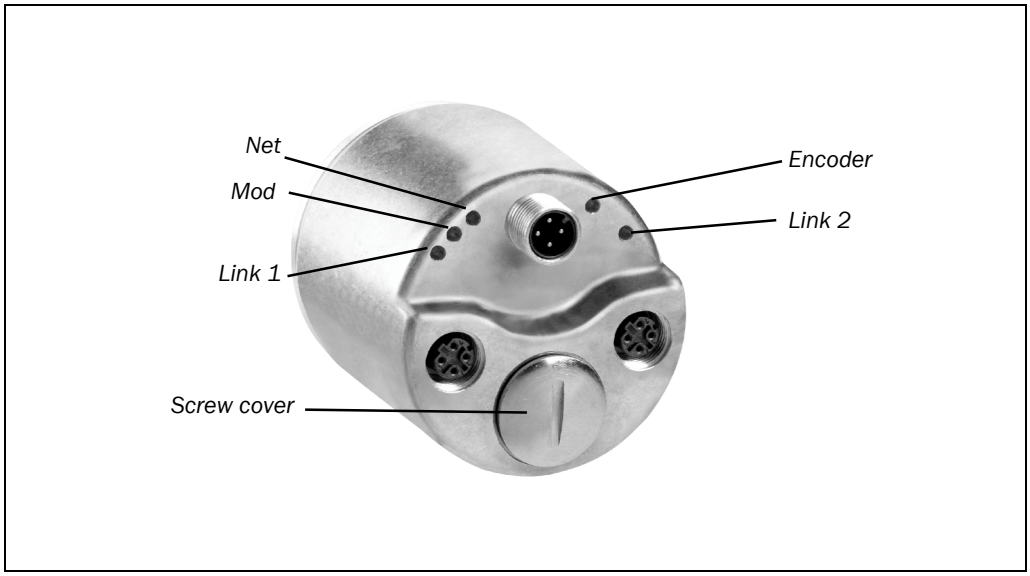
The next time the encoder is started up, all connection parameters (such as IP address, DHCP etc.) are set to the settings last saved in the non-volatile memory.

3.6 Controls and status indicators

The AFS60/AFM60 EtherNet/IP Absolute Encoder has five LEDs.

Three of the LEDs indicate the operating status (Net, Mod and Encoder), two the status of the Ethernet interface (Link 1 and Link 2).

Fig. 13: Position of the LEDs, the decade switches and the preset push-button



The LEDs are multi-colored. Tab. 23 on page 65 and Tab. 24 on page 66 show the meaning of the signals.

There are the following controls under the screw cover:

- decade switches for the address setting
- preset push-button

4 Commissioning

This chapter provides information on the electrical installation, configuration and commissioning of the AFS60/AFM60 EtherNet/IP Absolute Encoder.

➢ Please read this chapter before mounting, installing and commissioning the device.

4.1 Electrical installation

WARNING

Switch the power supply off!

The machine/system could unintentionally start up while you are connecting the devices.

➢ Ensure that the entire machine/system is disconnected during the electrical installation.

For the electrical installation you will need connection plugs and sockets (see the data sheet of the AFS60/AFM60 EtherNet/IP).

4.1.1 Connections of the AFS60/AFM60 EtherNet/IP

The connections of the AFS60/AFM60 EtherNet/IP are on the back.

Fig. 14: Position of the connections of the AFS60/AFM60 EtherNet/IP

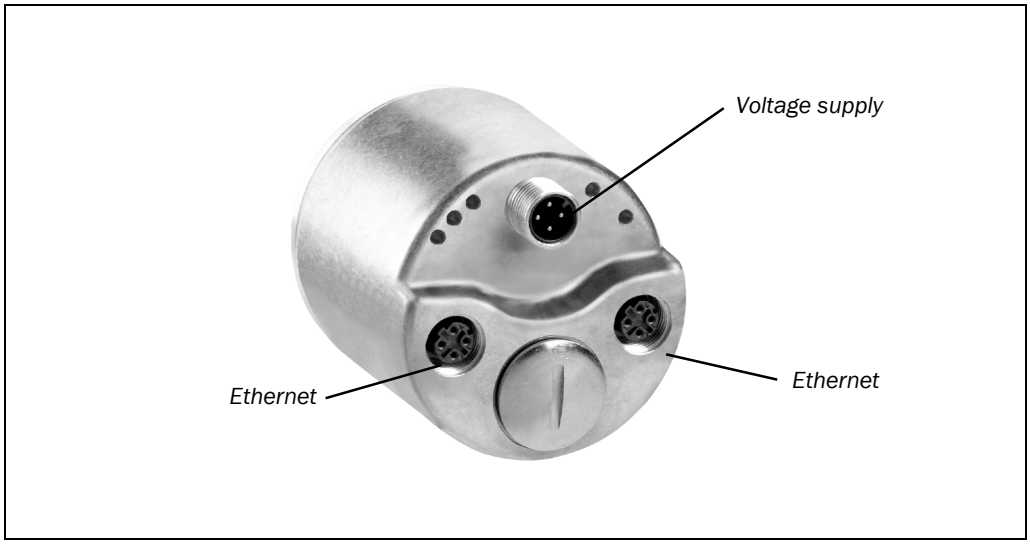
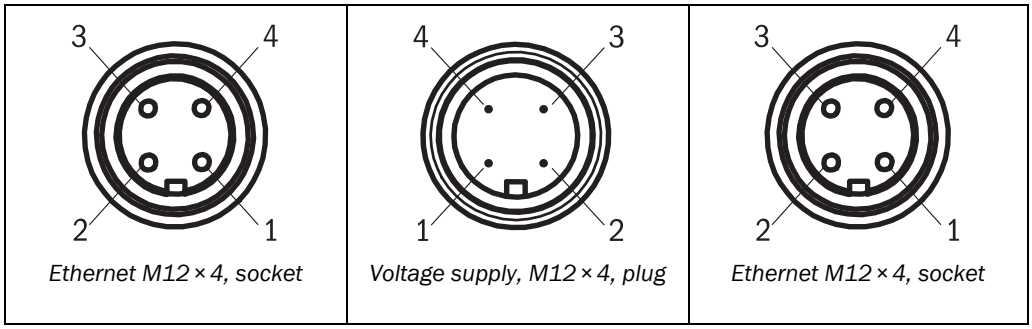


Fig. 15: Connections of the AFS60/AFM60 EtherNet/IP



Note Two Ethernet connections are used if the AFS60/AFM60 EtherNet/IP is integrated in a DLR or a line topology (see Fig. 3 on page 13).

Tab. 21: Pin assignment for the connection of the voltage supply

Pin	Signal	Core color[7)]	Function
1	V_S	Brown	Supply voltage 10 ... 30 V DC
2	–	White	Do not use
3	GND	Blue	0 V DC (Ground)
4	–	Black	Do not use

Note Pin 2 and 4 are **not allowed to be assigned**, otherwise irreparable damage could be caused to the AFS60/AFM60 EtherNet/IP.

Tab. 22: Pin assignment for the Ethernet connection

Pin	Signal	Core color[7)]	Function
1	TxD+	White/orange	Ethernet
2	RxD+	White/gray	Ethernet
3	TxD–	Orange	Ethernet
4	RxD–	Green	Ethernet

Notes

- **Connect the shield to the encoder housing!**
- Pay attention to the maximum cable lengths.
- Mount all cables with strain relief.

7) On the usage of pre-wired cables.

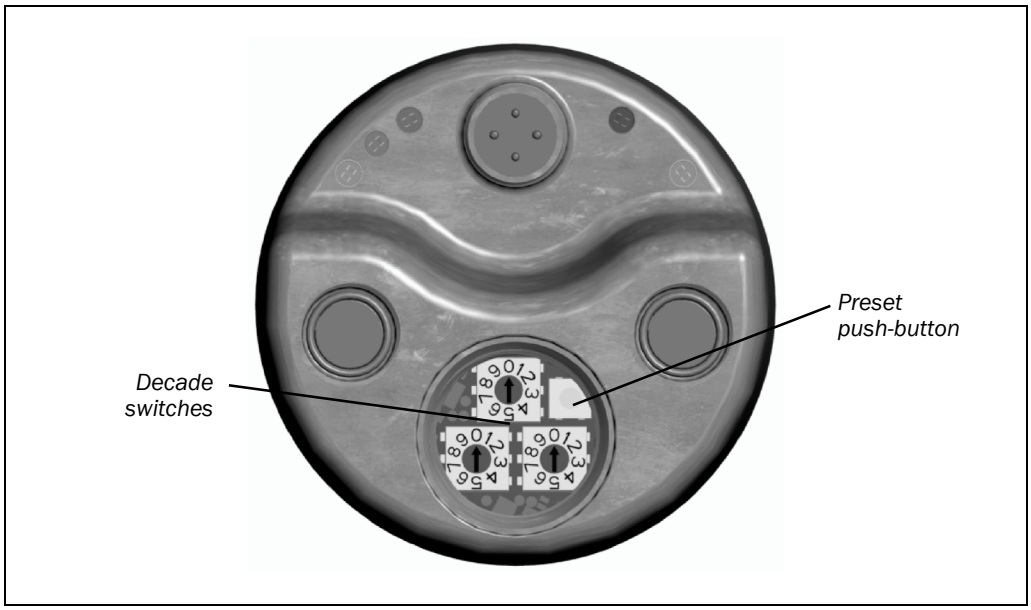
4.2 Hardware settings

There are the following controls for making settings under the screw cover:

- three decade switches for the address setting
- preset push-button

➢ Open the screw cover using a screwdriver for slot-head screws with a blade width of min. 15.0 mm.

Fig. 16: Position of the controls



The AFS60/AFM60 EtherNet/IP is supplied with the following default settings:

- Acquiring the IP address in the EtherNet/IP from a DHCP server (see 3.5.1 “IP address” on page 34).

Note The AFS60/AFM60 EtherNet/IP can also be delivered with a customer specific default setting.

4.3 Configuration

The AFS60/AFM60 EtherNet/IP can be integrated into both a Rockwell control system and into a system with a Schneider control system.

Within the Rockwell system the encoder is configured with the aid of a configuration assembly. In the case of a Schneider control system an EDS file can be loaded to integrate the encoder in the system.

Notes

- All software notes are displayed in English.
- All software notes are related to Rockwell RSLinx software.
  For the following example project the Allen Bradley control system “ControlLogix Controller 1756-L61” with “RSLogix 5000” V18 is used. It is a prerequisite that the hardware has already been installed.

4.3.1 Default delivery status

The AFS60/AFM60 EtherNet/IP is supplied with the following parameters:

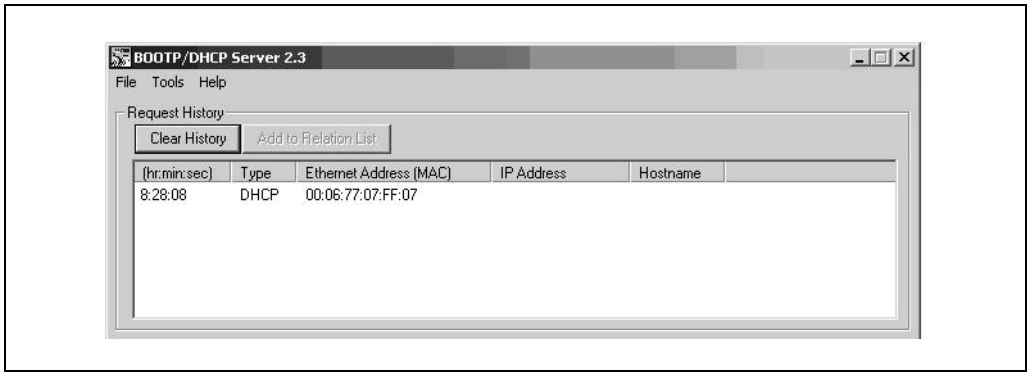
- code sequence = clockwise
- scaling = none
- steps per revolution = 32,767 (Basic), 262,144 (Advanced)
- total resolution = 134,217,727 (Basic), 1,073,741,823 (Advanced)
- preset = 0
- velocity measuring unit = turns/min
- round axis functionality = not activated
- nominator for round axis functionality = 2,048
- divisor for round axis functionality = 1

4.3.2 IP address assignment via DHCP

The AFS60/AFM60 EtherNet/IP is set from the factory to obtain the IP address from a DHCP server.

➢ Start the **BOOTP/DHCP server** (as a rule on the Start menu on your PC/notebook in **Rockwell Software**, **BOOTP-DHCP Server**, **BOOTP-DHCP Server**).

Fig. 17: MAC address in the BOOTP/DHCP server

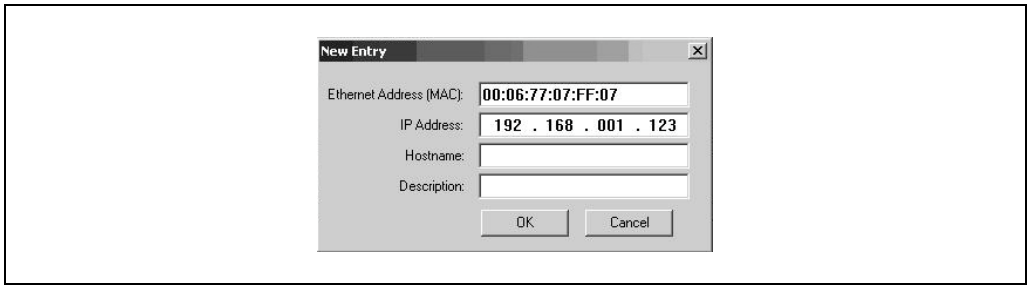


In the program window for the BOOTP/DHCP server the AFS60/AFM60 EtherNet/IP appears as a bus user with its MAC address, however without an IP address assigned.

Green The Mod LED on the AFS60/AFM60 EtherNet/IP flashes green (the encoder does not yet have an IP address).

➢ Open the encoder in the BOOTP/DHCP server by double-clicking.

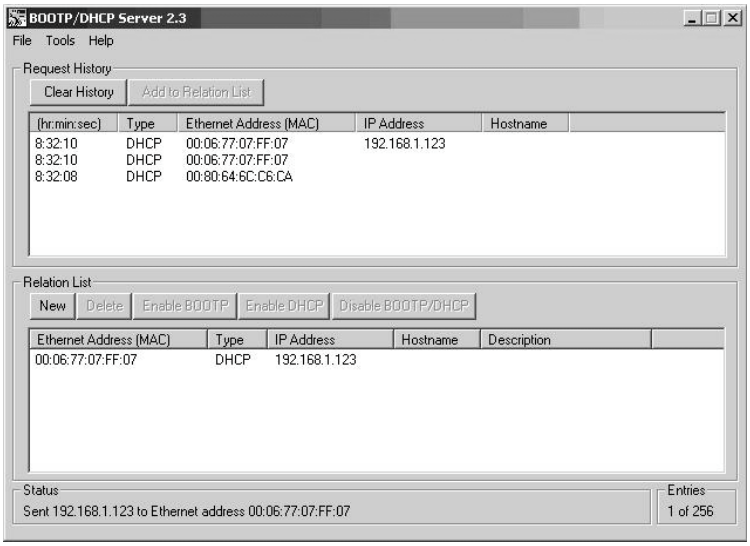
Fig. 18: Entry of the IP address in the BOOTP/DHCP server



➢ In the **IP Address** field type a valid, spare address and click **OK**.

➢ Click on **Clear History**.

Fig. 19: Integration of the IP address in the BOOTP/DHCP server



After a delay the encoder appears both in **Request History** and in **Relation List** with the IP address entered. After one minute or four unsuccessful requests the default IP address 192.168.1.123 is set.

● **Green** The Mod LED on the AFS60/AFM60 EtherNet/IP illuminates green continuously (the encoder now has a valid IP address).

Checking the integration in EtherNet/IP via RSLinx Classic

With the aid of the tool **RSLinx Classic** you can again check whether the IP address set is detected by the control system.

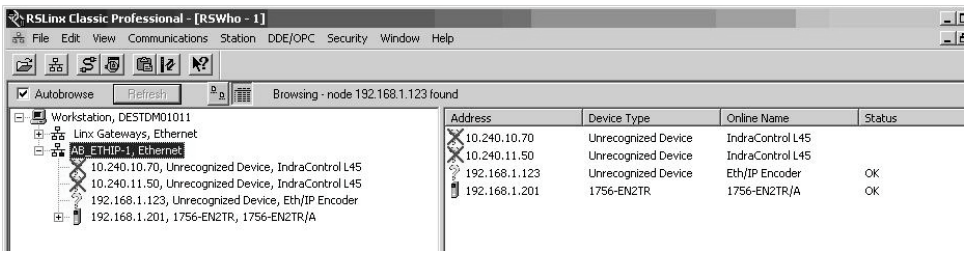
➢ Start **RSLinx Classic** (as a rule on the Start menu on your PC/notebook in **Rockwell Software**, **RSLinx**, **RSLinx Classic**).

➢ Click on the **RSWho** button in the program.

Fig. 20: RSWho button in RSLinx Classic

➢ Then open the path AB_ETHIP-1, Ethernet.
The encoder can be seen with its IP address.

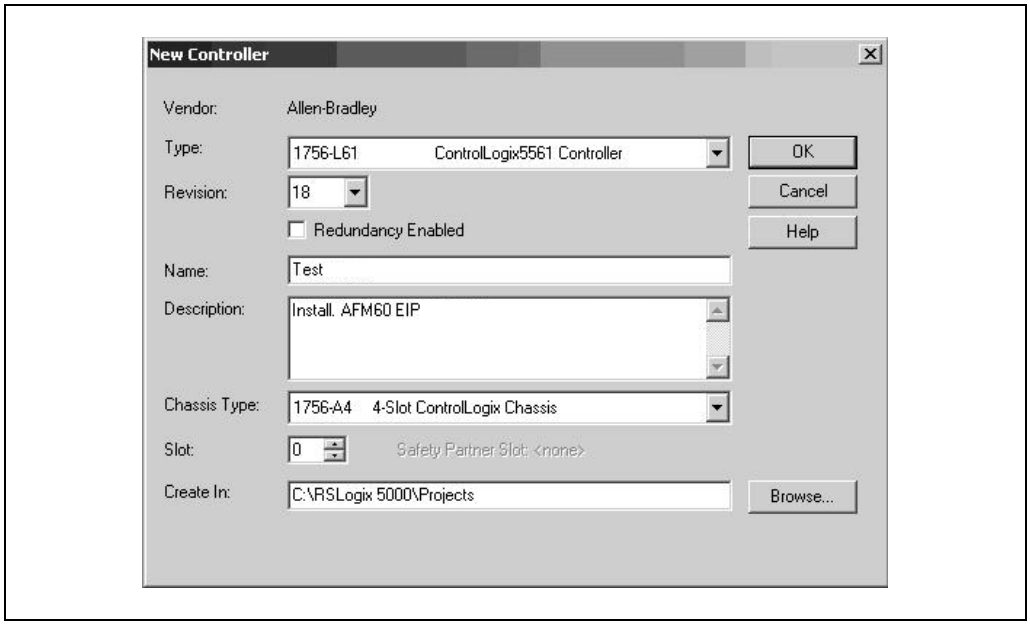
Fig. 21: Encoder on the path AB_ETHIP-1 in RSLinx Classic



4.3.3 Creating a project in the controller software

- Start the controller software **RSLogix 5000** (as a rule on the Start menu on your PC/ notebook in **Rockwell Software**, **RSLogix 5000 Enterprise Series**, **RSLogix 5000**).
- On the **File** menu open a new project using the **New...** command.
- Configure the hardware.

Fig. 22: Configuring the hardware



Example:

- **Type**: 1756-L61 ControlLogix5561 Controller (dependent on the controller)
- **Name**: Test name (name can be selected as required)
- **Description**: Commissioning AFM60 EIP (can be selected as required)
- **Chassis Type**: 1756-A4 4-Slot ControlLogix Chassis (dependent on the chassis)
- **Create In**: storage location (can be selected as required)

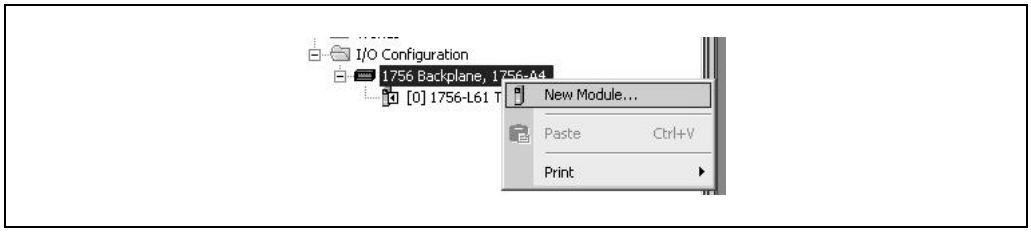
- Click **OK**.
  The **RSLogix 5000 [Name]** window is opened.

Note Type and Chassis Type must match your control system.

Adding communication interface

➢ In the **Controller Organizer** click **1756 Backplane, 1756-A4** using the right mouse button and select **New Module...**.

Fig. 23: Adding communication interface

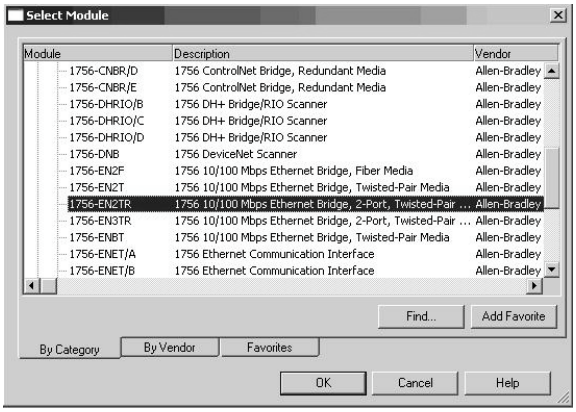


The **Select Module** dialog box opens.

➢ In the **Select Module** dialog box select the **By Category** tab.

➢ In the tree in **Communications** select the module **1756-EN2TR**.

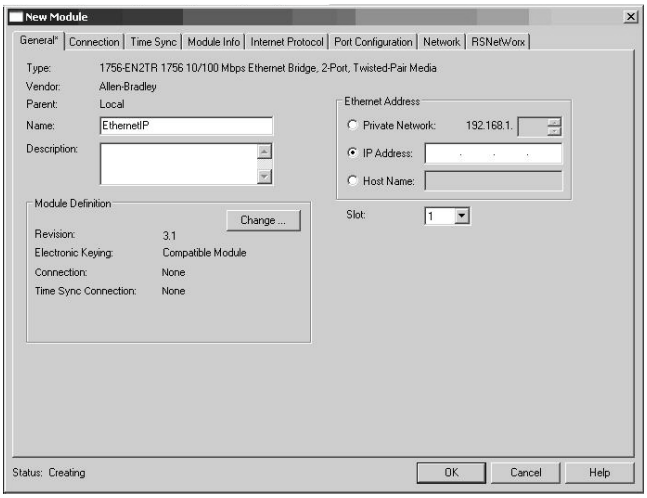
Fig. 24: Selecting communication interface



➢ Click **OK**.
The **New Module** dialog box will open.

➢ On the **General** tab assign a name in the **Name** field, in the **IP Address** field the IP address, and select the **Slot**.

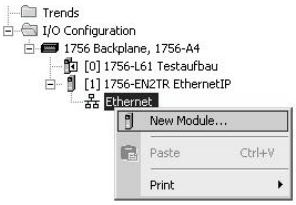
Fig. 25: Name of the communication interface



➢ Click **OK**.
In **Controller Organizer** in **1756 Backplane, 1756-A4** the selected module **1756-EN2TR** [with name] appears along with the symbol for **Ethernet**.

➢ Using the right mouse button click the **Ethernet** symbol and select the **New Module...** command.

Fig. 26: Integrating encoder



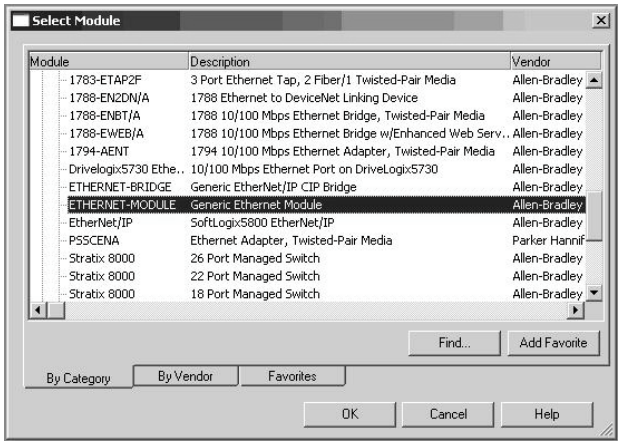
The **Select Module** dialog box opens.

➢ In the **Select Module** dialog box select the **By Category** tab.

➢ Open the **Communication** tree.

➢ In the **Communication** tree select the module **ETHERNET-MODULE (Generic Ethernet Module)**.

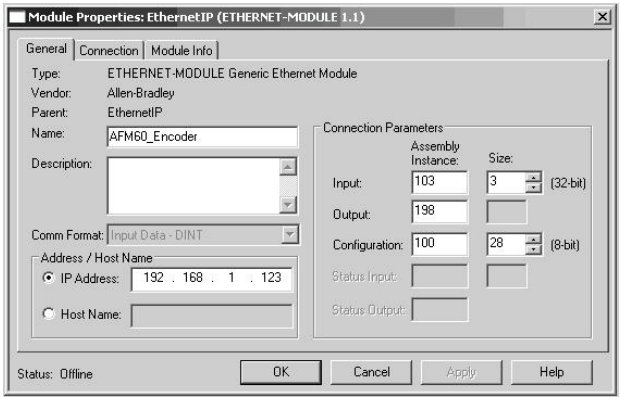
Fig. 27: Selecting module



➢ Click **OK**.
The **Module Properties [module name]** dialog box is opened.

➢ In the **Module Properties [module name]** dialog box enter the settings for **Input**, **Output**, as well as **Configuration**.

Fig. 28: Entering module properties



Example:

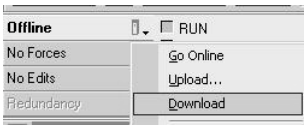
- **Name**: AFM60_Encoder (name can be selected as required)
- **Comm Format**: Input Data — DINT
- **IP Address**: 192.168.1.123
- **Input**: Assembly Instance: 103; Size: 3
- **Output**: Assembly Instance: 198
- **Configuration**: Assembly Instance:100; Size: 28

➢ Click **OK**.

Download the configuration to the control system

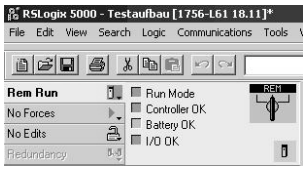
➢ Load the configuration to the control system.

Fig. 29: Loading configuration



The status indicators for **Run Mode**, **Controller OK** and **I/O OK** change to green.

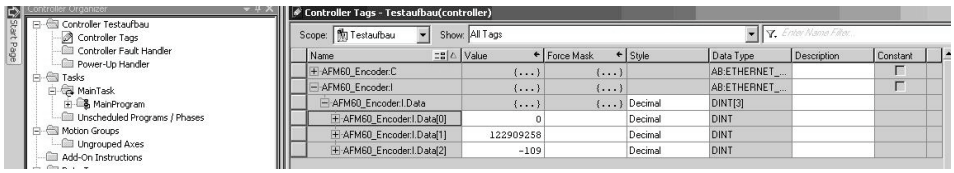
Fig. 30: Communication status



Checking the communication

To check the communication between control system and encoder, the data the control system receives from the encoder can be displayed.

Fig. 31: Checking the communication



➢ In the **Controller Organizer** open the **Controller Testaufbau** folder, **Controller Tags**.

➢ In the **Controller Tags** in the **Name** column open the **AFM60_Encoder:I**, **AFM60_Encoder:I.Data** item.

Displayed data in the example in Fig. 31:

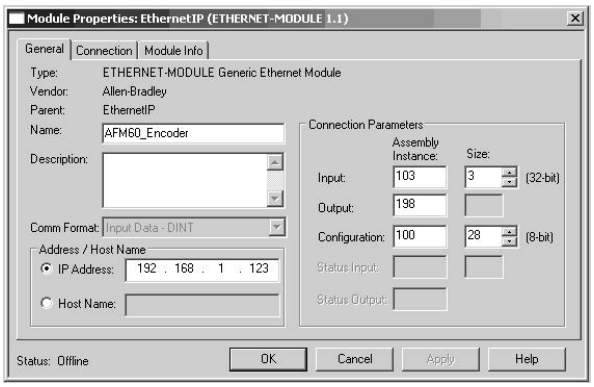
- **AFM60_Encoder:I.Data[0]**: fault header: 0
- **AFM60_Encoder:I.Data[1]**: position: 122909258
- **AFM60_Encoder:I.Data[2]**: velocity: –109 turns/min

4.3.4 Configuration via the configuration assembly

With the aid of the Assembly Object the encoder can be configured via a configuration assembly.

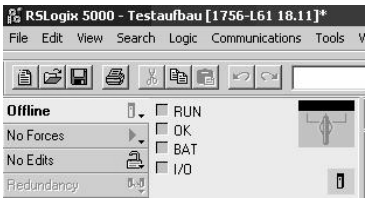
To also send a configuration assembly during the start process, configure in the **Module Properties** in the **Connection Parameters** the **Assembly Instance** for **Configuration** as **100** and its size (**Size**) as **28** bytes.

Fig. 32: Settings for the configuration assembly



➢ To set the parameters for the configuration assembly place the control system in the **Offline mode**.

Fig. 33: Mode for the configuration assembly



Example data for a configuration assembly

The data for the configuration assembly are transferred in the 28 bytes of instance 100 configured previously (see Tab. 15 on page 26).

You can see these data in **Controller Tags** in the **Name** column in the **AFM60_Encoder:C**, **AFM60_Encoder:C.Data** item.

Note The low byte is displayed before the high byte.

Fig. 34: Example data for a configuration assembly

Name	Value	Force Mask	Style
AFM60_Encoder:C	{...}	{...}	
AFM60_Encoder:C.Data	{...}	{...}	Hex
AFM60_Encoder:C.Data[0]	16#00		Hex
AFM60_Encoder:C.Data[1]	16#00		Hex
AFM60_Encoder:C.Data[2]	16#00		Hex
AFM60_Encoder:C.Data[3]	16#00		Hex
AFM60_Encoder:C.Data[4]	16#00		Hex
AFM60_Encoder:C.Data[5]	16#10		Hex
AFM60_Encoder:C.Data[6]	16#00		Hex
AFM60_Encoder:C.Data[7]	16#00		Hex
AFM60_Encoder:C.Data[8]	16#00		Hex
AFM60_Encoder:C.Data[9]	16#80		Hex
AFM60_Encoder:C.Data[10]	16#00		Hex
AFM60_Encoder:C.Data[11]	16#00		Hex
AFM60_Encoder:C.Data[12]	16#00		Hex
AFM60_Encoder:C.Data[13]	16#01		Hex
AFM60_Encoder:C.Data[14]	16#00		Hex
AFM60_Encoder:C.Data[15]	16#00		Hex
AFM60_Encoder:C.Data[16]	16#00		Hex
AFM60_Encoder:C.Data[17]	16#00		Hex
AFM60_Encoder:C.Data[18]	16#00		Hex
AFM60_Encoder:C.Data[19]	16#00		Hex
AFM60_Encoder:C.Data[20]	16#00		Hex
AFM60_Encoder:C.Data[21]	16#00		Hex
AFM60_Encoder:C.Data[22]	16#00		Hex
AFM60_Encoder:C.Data[23]	16#00		Hex
AFM60_Encoder:C.Data[24]	16#0f		Hex
AFM60_Encoder:C.Data[25]	16#1f		Hex
AFM60_Encoder:C.Data[26]	16#00		Hex
AFM60_Encoder:C.Data[27]	16#00		Hex

- Counts (steps) per revolution CPR = 4,096 = 1000h
  C.Data[4] 00h and **C.Data[5] 10h**
- Total resolution CMR = 32,768 = 8000h
  C.Data[8] 00h and **C.Data[9] 80h**
- Direction of revolution cw = 0
  C.Data[12] 00h
- Scaling on = 1h
  C.Data[13] 01h
- Velocity format = 1F0Fh
  C.Data[24] 0Fh and **C.Data[25] 1Fh**

4.4 Configuration examples

The following examples show the configuration of two programs that read and write acyclic data (temperature) (Preset). For this purpose the programs are written in ladder logic with the aid of the software RSLogix 5000 from Rockwell Automation.

Note During programming the control system must be in the offline mode.

Fig. 35: Control system in the offline mode

- First you must define and declare the variables for the program.
- Then add the program blocks to the ladder logic and assign the variables as appropriate.
- After that you must download the program to the control system.
- Finally, you can test the program.

4.4.1 Reading temperature

In the first example the temperature of the encoder is to be read with the aid of the parameter 64h, Temperature Value.

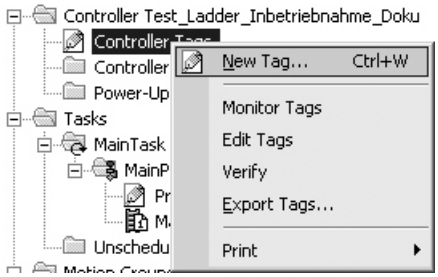
Defining and declaring variables

As the initial step the variables TEMP_Trigger, TEMP_OneShot, TEMP_Value and TEMP_Message must be defined and declared for the program.

First the variable TEMP_Trigger, which controls the reading process, is added.

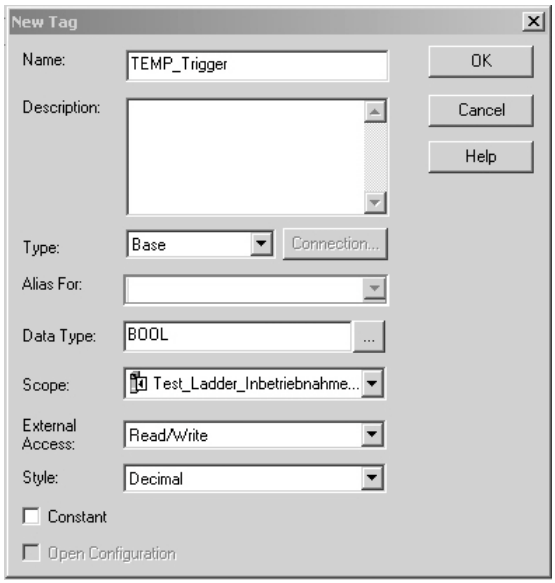
- In the **Controller Organizer**, using the right mouse button click **Controller Tags** and select **New Tag**.

Fig. 36: Adding a new variable



The **New Tag** dialog box opens.

Fig. 37: Definition of the variable TEMP_Trigger

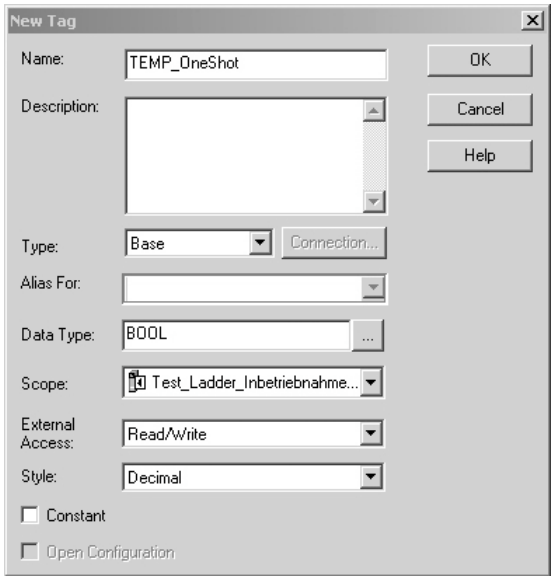


➢ In the **Name** field enter TEMP_Trigger, in the **Data Type** field select the data type BOOL and click **OK**.

To only trigger the action once, a further element, in this case an edge-sensitive element, must be defined and declared. This element ensures that the action is only triggered if an edge change from 0 to 1 occurs in the variable TEMP_Trigger.

➢ Select again **New Tag**.

Fig. 38: Definition of the variable TEMP_OneShot

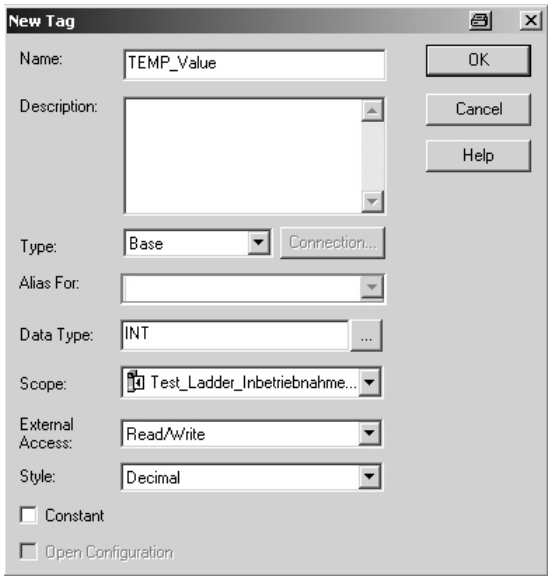


➢ In the **New Tag** dialog box enter TEMP_OneShot in the **Name** field, in the **Data Type** select the data type BOOL and click **OK**.

A further variable must be added that will then contain the temperature value later (see Tab. 19 on page 28, ID100/64h, Temperature Value).

➢ Select again **New Tag**.

Fig. 39: Definition of the variable TEMP_Value

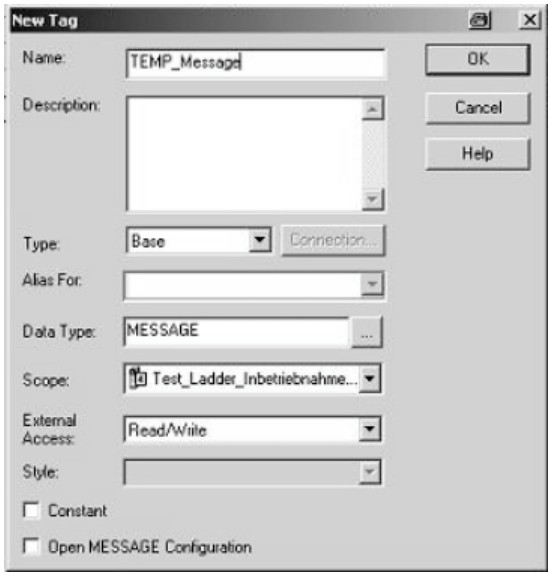


➢ In the **New Tag** dialog box enter TEMP_Value in the **Name** field, select in the **Data Type** field the data type INT and click **OK**.

Finally a further variable must be defined and declared that obtains the temperature value from the control system.

➢ Select again **New Tag**.

Fig. 40: Definition of the variable TEMP_Message



➢ In the **New Tag** dialog box enter TEMP_Message in the **Name** field, select in the **Data Type** field the data type MESSAGE and click **OK**.

Fig. 41 shows the resulting variable structure for reading the temperature acyclically.

Fig. 41: Variable structure for reading the temperature

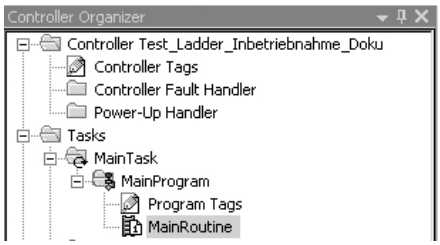
Name	Value	Force Mask	Style	Data Type	Description	Constant
AFM60_EIP:C	{...}	{...}		AB:ETHERNET_...		
AFM60_EIP:I	{...}	{...}		AB:ETHERNET_...		
TEMP_OneShot	0		Decimal	BOOL		
TEMP_Trigger	0		Decimal	BOOL		
TEMP_Value	0		Decimal	INT		
TEMP_Message	{...}	{...}		MESSAGE		

Defining process sequence

After you have defined and declared the variables, the program blocks must be inserted in the ladder logic and the variables assigned as appropriate.

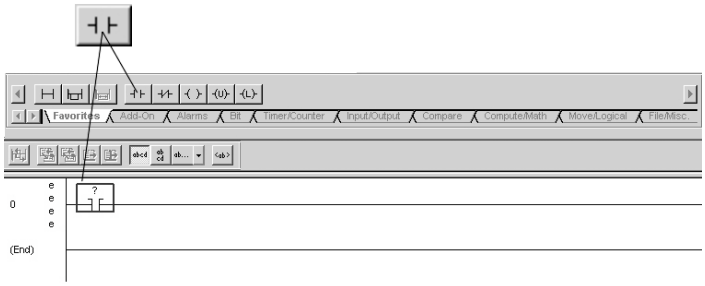
In **Tasks**, **Main Task**, **MainProgramm** open the **MainRoutine** window.

Fig. 42: Opening MainRoutine



For the first block an input is added that is to trigger the “read temperature” process.

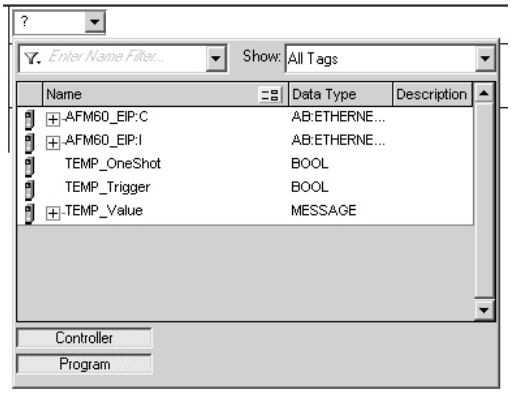
Fig. 43: Adding ExamineOn block



➢ On the **Favorites** tab select the **ExamineOn** block and add it to the **MainRoutine**.

The related variable must be assigned to this input, in our example the variable TEMP_Trigger.

Fig. 44: Allocation of the variable TEMP_Trigger to ExamineOn

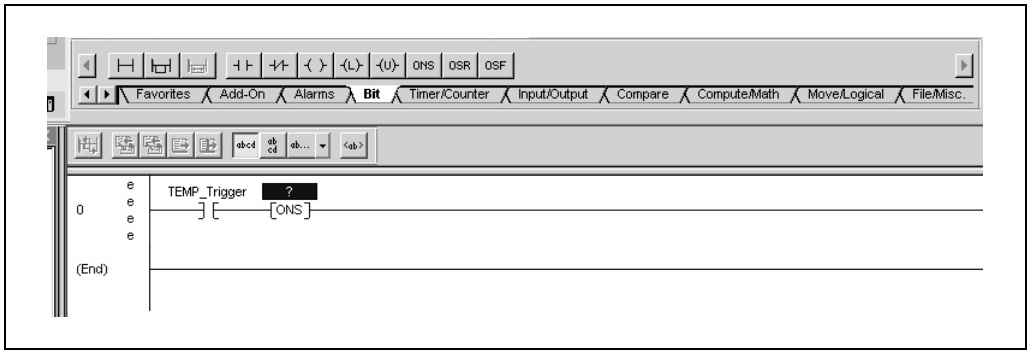


➢ Click on the **question mark**.
A drop-down menu is opened.

➢ Select the variable TEMP_Trigger.

The ONS block must be added for the edge sensitivity of the process sequence.

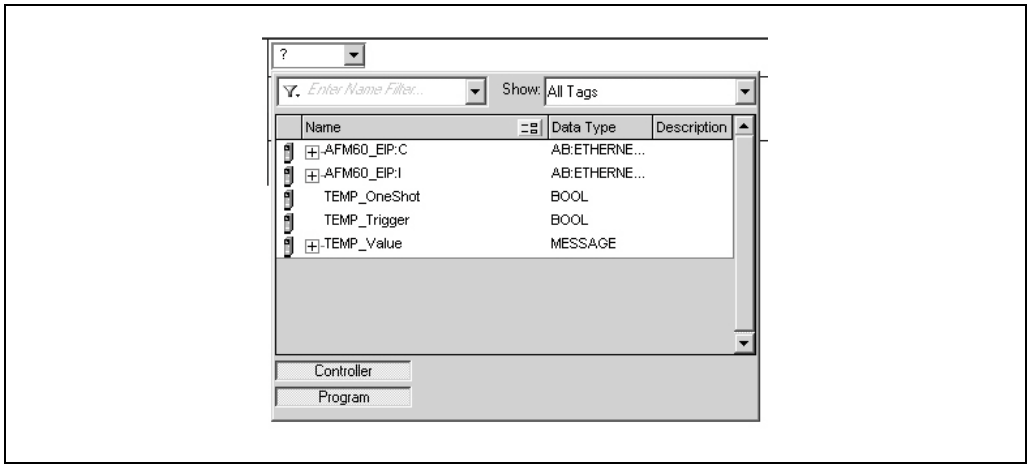
Fig. 45: Adding ONS block



➢ On the **Bit** tab select the **ONS** block and add it to the **MainRoutine**.

A variable must also be assigned to this block.

Fig. 46: Allocation of the variable TEMP_OneShot to ONS

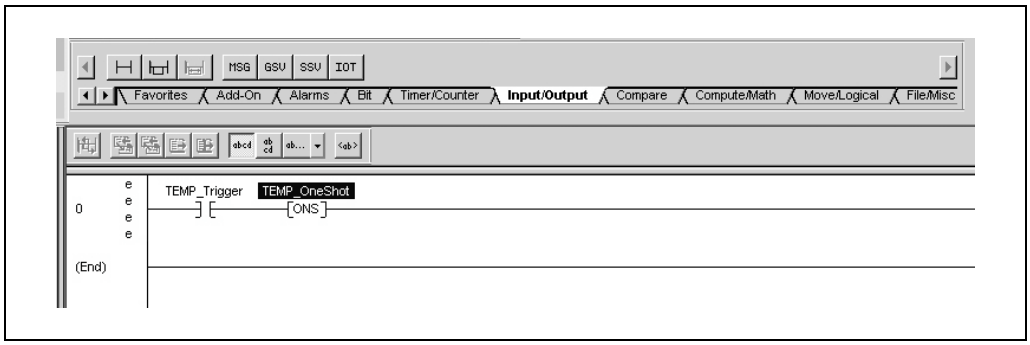


➢ Click on the **question mark**.
A drop-down menu is opened.

➢ Select the variable TEMP_OneShot.

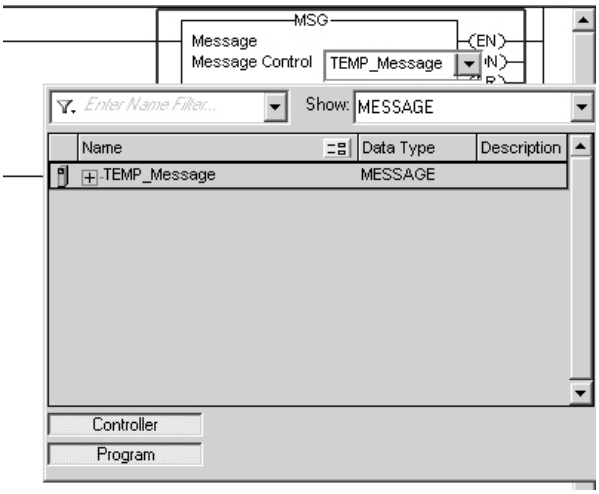
In the next step the message must be configured to read the temperature value from the encoder.

Fig. 47: Adding MSG block



➢ On the **Input/Output** tab select the **MSG** block and add it to the **MainRoutine**.

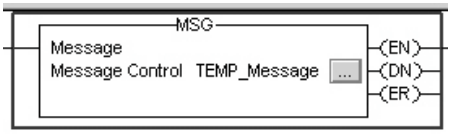
Fig. 48: Allocation of the variable TEMP_Message to MSG



➢ In the **Message Control** field select the variable TEMP_Message.

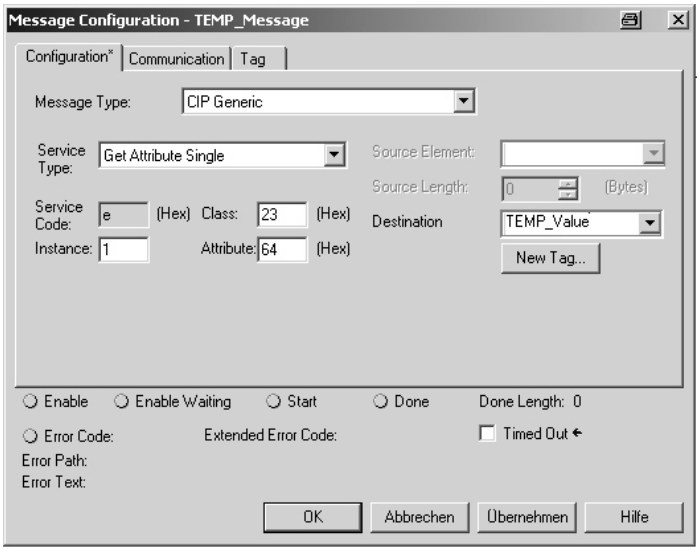
The MSG block must then be configured.

Fig. 49: Opening configuration dialog box for the MSG block



➢ For this purpose click the button with the three dots.
The **Message Configuration** dialog box will open.

Fig. 50: Configuration dialog box for the MSG block



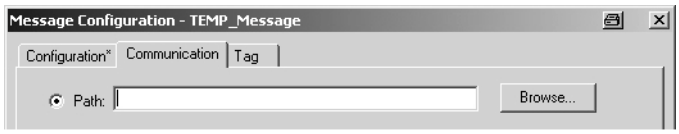
➢ Configure the following parameters on the **Configuration** tab:

- **Service Type:** Get Attribute Single (see Tab. 16 on page 27)
- **Instance:** 1 (as only one device is connected to the control system)
- **Class:** 23(h) (Position Sensor Object, see Tab. 4 on page 17)
- **Attribute:** 64(h) (Temperature Value, see Tab. 19 on page 28)
- **Destination:** TEMP_Value

Note TEMP_Value is the fourth variable added. The value for the temperature is written to this variable on executing the example program.

➢ Open the **Communication** tab.

Fig. 51: Communication tab



➢ Beside the **Path** field click the **Browse...** button.
The **Message Path Browser** dialog box is opened.

➢ Select the encoder connected.

Fig. 52: Selecting encoder

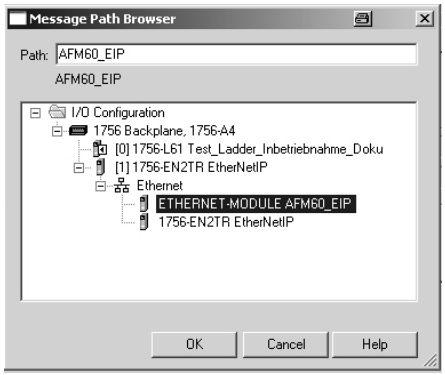


Fig. 53: Selected encoder

The encoder is applied in the **Path** field.

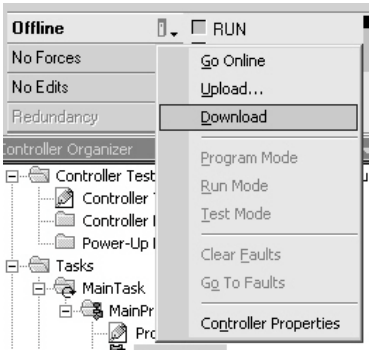
➢ Close the **Message Path Browser** dialog box using **OK**.

Transferring program to the control system

Finally the program is transferred to the control system.

➢ From the **Offline** menu select the **Download** command.

Fig. 54: Transferring the program to the control system

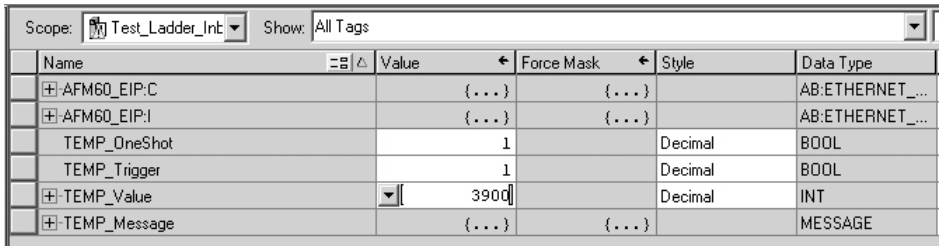


➢ Accept the next message.

Testing program

If the variable TEMP_Trigger is changed from 0 to 1 in the **Controller Organizer**, the temperature value is displayed in the variable TEMP_Value (here: 39.00 °C).

Fig. 55: Display of the temperature value in TEMP_Value



4.4.2 Setting preset value

In the following example a preset value is to be set.

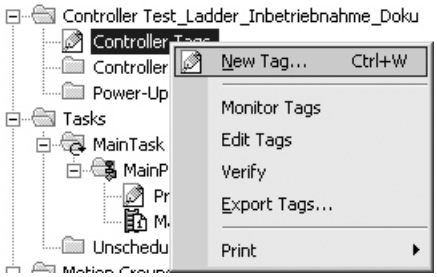
Defining and declaring variables

As the initial step the variables PRESET_Trigger, PRESET_OneShot, PRESET_Value and PRESET_Message must be defined and declared for the program.

First the variable PRESET_Trigger is added, this variable controls the process.

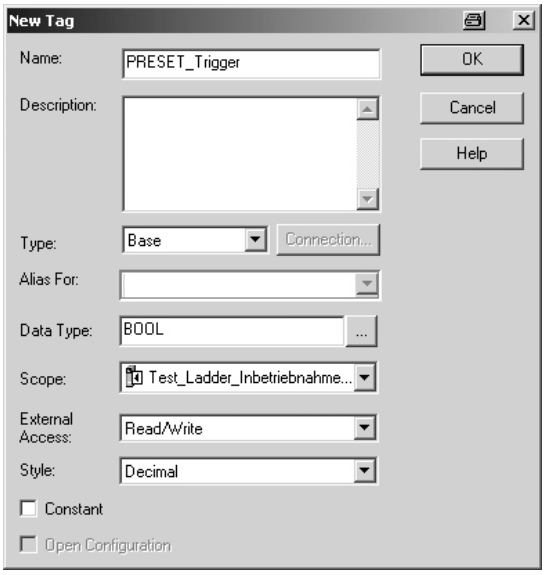
➢ In the **Controller Organizer**, using the right mouse button click **Controller Tags** and select **New Tag**.

Fig. 56: Adding a new variable



The **New Tag** dialog box opens.

Fig. 57: Definition of the variable PRESET_Trigger

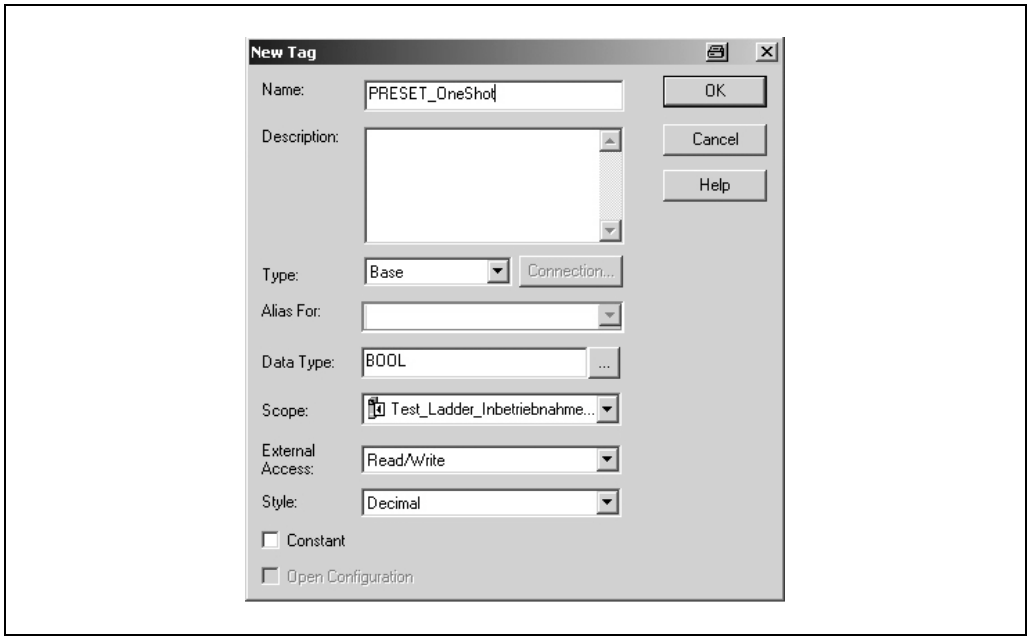


➢ In the **Name** field enter PRESET_Trigger, in the **Data Type** select the data type BOOL and click **OK**.

To only trigger the action once, a further element, in this case an edge-sensitive element, must be defined and declared. This element ensures that the action is only triggered if an edge change from 0 to 1 occurs in the variable PRESET_Trigger.

➢ Select again **New Tag**.

Fig. 58: Definition of the variable PRESET_OneShot

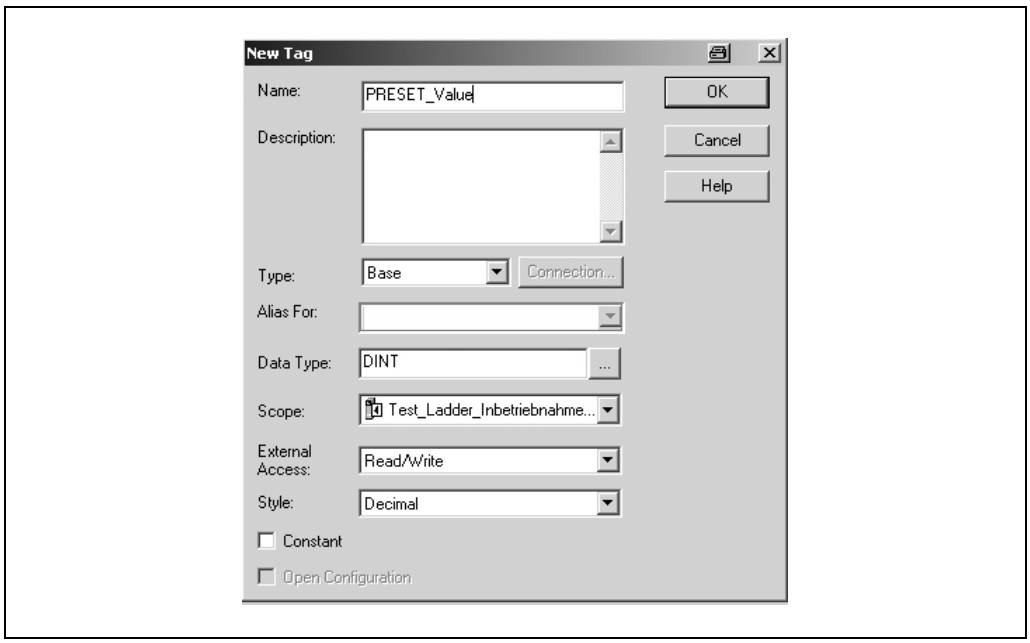


➢ In the **New Tag** dialog box enter PRESET_OneShot in the **Name** field, select in the **Data Type** field the data type BOOL and click **OK**.

A further variable must be added that will then contain the preset value later (see Tab. 19 on page 28, ID19/13h, Preset Value).

➢ Select again **New Tag**.

Fig. 59: Definition of the variable PRESET_Value

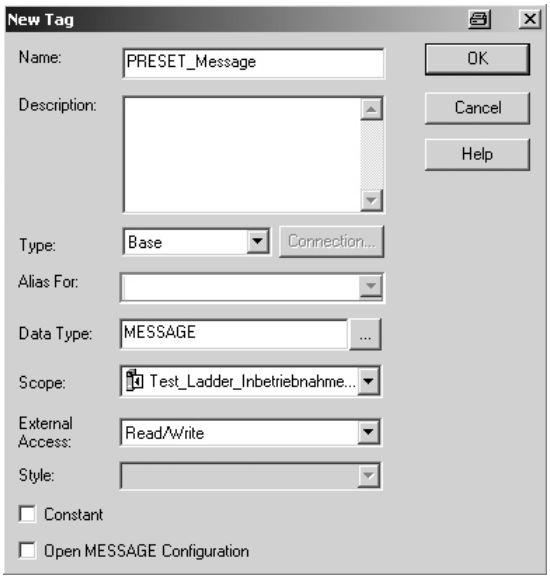


➢ In the **New Tag** dialog box enter PRESET_Value in the **Name** field, select in the **Data Type** field the data type DINT and click **OK**.

Finally a further variable must be defined and declared that obtains the preset value from the control system.

➢ Select again **New Tag**.

Fig. 60: Definition of the variable PRESET_Message



➢ In the **New Tag** dialog box enter PRESET_Message in the **Name** field, select in the **Data Type** field the data type MESSAGE and click **OK**.

Fig. 61 shows the resulting variable structure for setting a preset value.

Fig. 61: Variable structure for setting a preset value

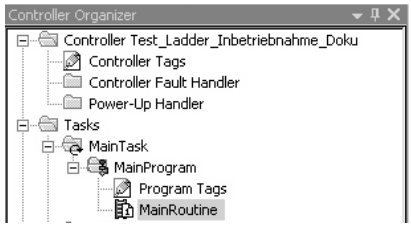
Name	Value	Force Mask	Style	Data Type
⊞ AFM60_EIP:C	{...}	{...}		AB:ETHERNET_...
⊞ AFM60_EIP:I	{...}	{...}		AB:ETHERNET_...
PRESET_Trigger	0		Decimal	BOOL
PRESET_OneShot	0		Decimal	BOOL
⊞ PRESET_Value	0		Decimal	DINT
⊞ PRESET_Message	{...}	{...}		MESSAGE

Defining process sequence

After you have defined and declared the variables, the program blocks must be inserted in the ladder logic and the variables assigned as appropriate.

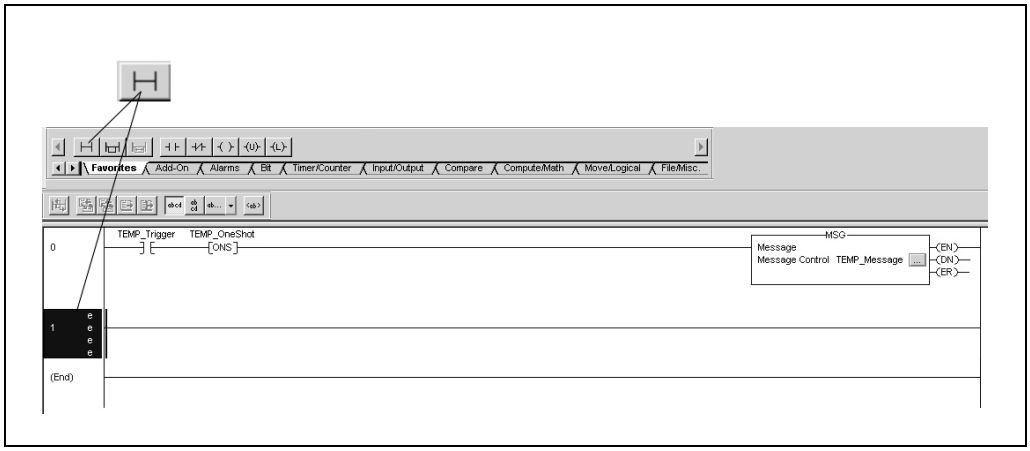
In **Tasks**, **Main Task**, **MainProgramm** open the **MainRoutine** window.

Fig. 62: Opening MainRoutine



If the process sequence for writing a preset value is to run in parallel with the previous example, then a new thread must be added.

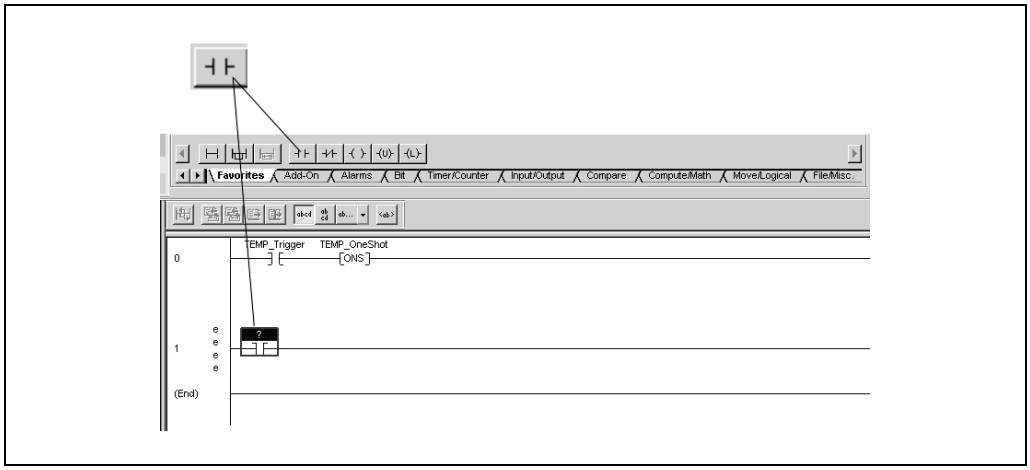
Fig. 63: Adding Rung block



➢ On the **Favorites** tab select the **Rung** block and add it to the **MainRoutine**.

For the first block an input is added that is to trigger the “set preset value” process.

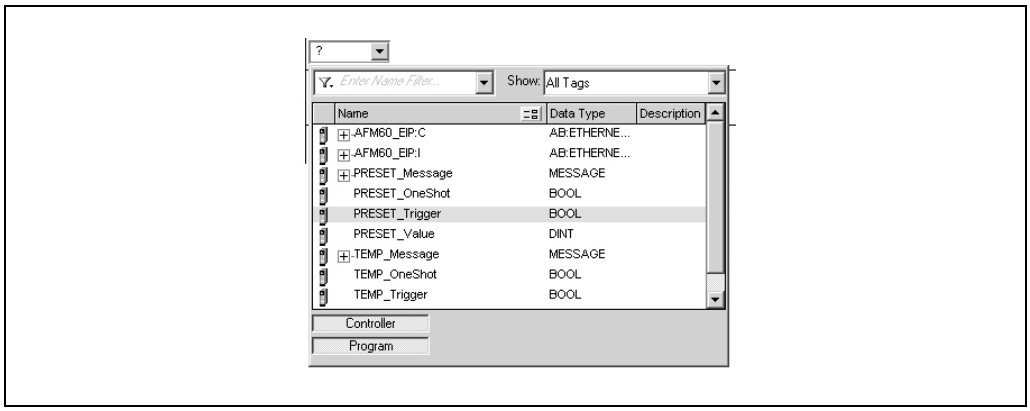
Fig. 64: Adding ExamineOn block



➢ On the **Favorites** tab select the **ExamineOn** block and add it to the **MainRoutine**.

The related variable must be assigned to this input, in our example the variable PRESET_Trigger.

Fig. 65: Allocation of the variable PRESET_Trigger to ExamineOn

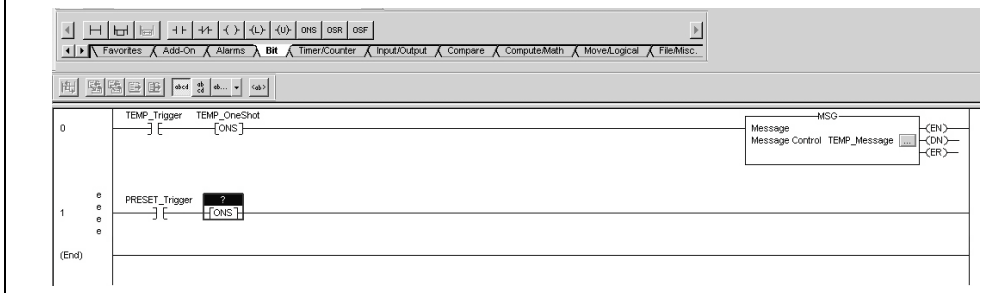


➢ Click on the **question mark**.
A drop-down menu is opened.

➢ Select the variable PRESET_Trigger.

The ONS block must be added for the edge sensitivity of the process sequence.

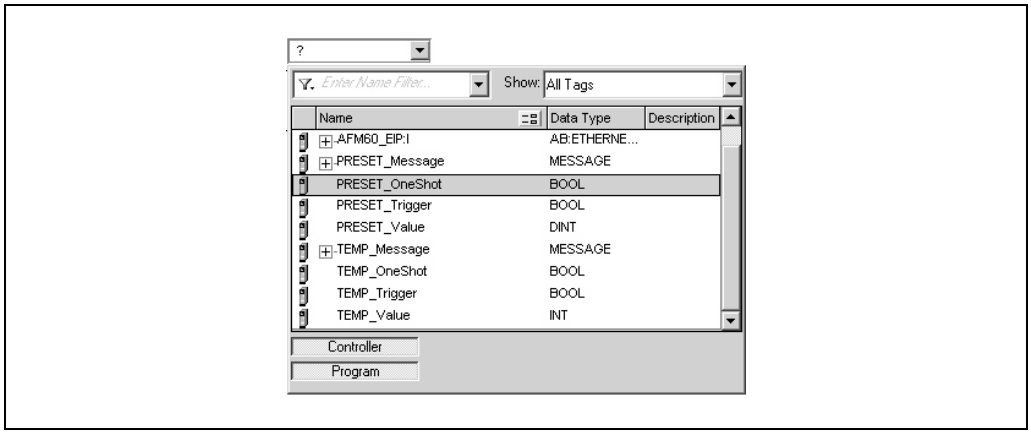
Fig. 66: Adding ONS block



➢ On the **Bit** tab select the **ONS** block and add it to the **MainRoutine**.

A variable must also be assigned to this block.

Fig. 67: Allocation of the variable PRESET_OneShot to ONS

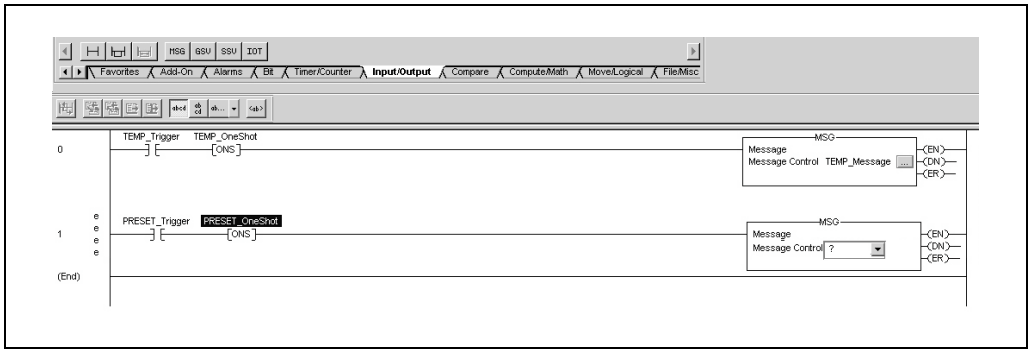


➢ Click on the **question mark**.
A drop-down menu is opened.

➢ Select the variable PRESET_OneShot.

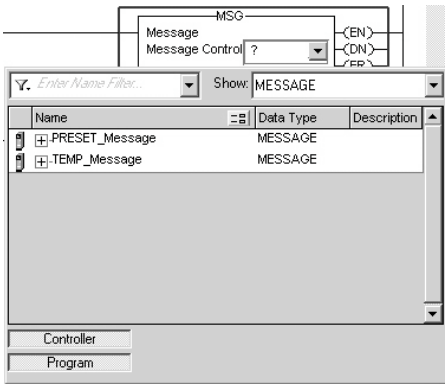
In the next step the message must be configured to write the preset value to the encoder.

Fig. 68: Adding MSG block



➢ On the **Input/Output** tab select the **MSG** block and add it to the **MainRoutine**.

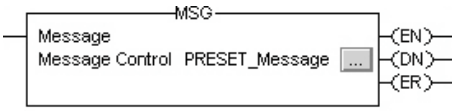
Fig. 69: Allocation of the variable PRESET_Message to MSG



➢ In the **Message Control** field select the variable PRESET_Message.

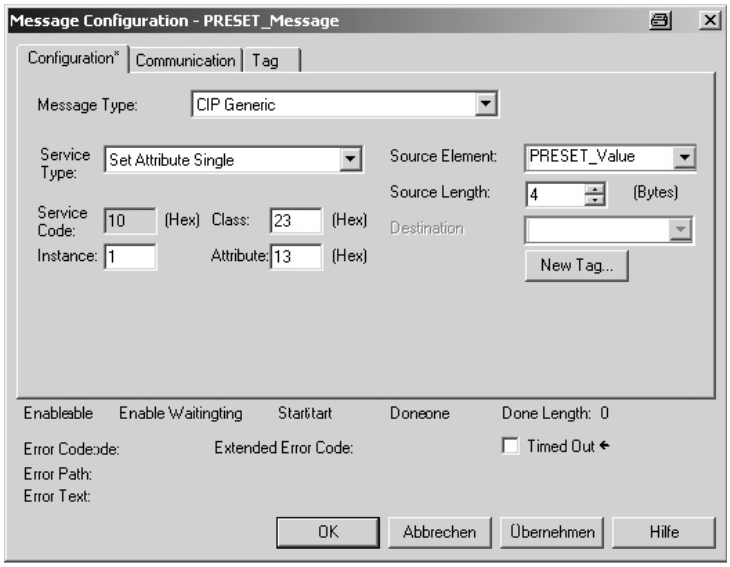
The MSG block must then be configured.

Fig. 70: Opening configuration dialog box for the MSG block



➢ For this purpose click the button with the three dots.
The **Message Configuration** dialog box will open.

Fig. 71: Configuration dialog box for the MSG block



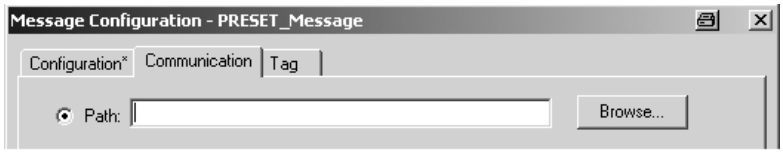
➢ Configure the following parameters on the **Configuration** tab:

- **Service Type:** Set Attribute Single (see Tab. 16 on page 27)
- **Instance:** 1 (as only one device is connected to the control system)
- **Class:** 23(h) (Position Sensor Object, see Tab. 4 on page 17)
- **Attribute:** 13(h) (Preset Value, see Tab. 19 on page 28)
- **Source Element:** PRESET_Value
- **Source Length:** 4

Note PRESET_Value is the fourth variable added. On executing the example program the preset value is taken from this variable and written to the attribute 13h of the Position Sensor Object.

➢ Open the **Communication** tab.

Fig. 72: Communication tab



➢ Beside the **Path** field click the **Browse...** button.
The **Message Path Browser** dialog box is opened.

➢ Select the encoder connected.

Fig. 73: Selecting encoder

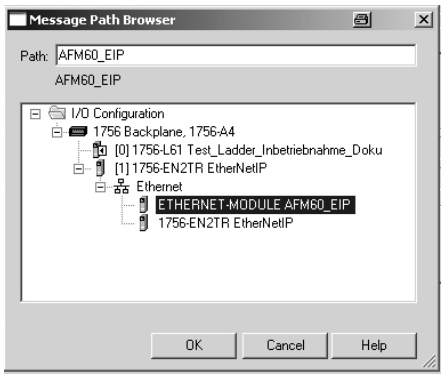


Fig. 74: Selected encoder

The encoder is applied in the **Path** field.

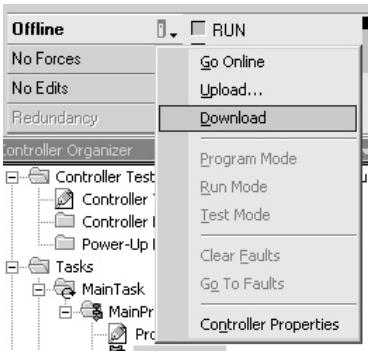
➢ Close the **Message Path Browser** dialog box using **OK**.

Transferring program to the control system

Finally the program is transferred to the control system.

➢ From the **Offline** menu select the **Download** command.

Fig. 75: Transferring the program to the control system



➢ Accept the next message.

Testing program

Fig. 76: Display of the preset value in PRESET_Value

Name	Value	Force Mask	Style	Data Type
AFM60_EIP:C	{...}	{...}		AB:ETHERNET_...
AFM60_EIP:I	{...}	{...}		AB:ETHERNET_...
AFM60_EIP:I.Data	{...}	{...}	Decimal	DINT[3]
AFM60_EIP:I.Data[0]	0		Decimal	DINT
AFM60_EIP:I.Data[1]	500		Decimal	DINT
AFM60_EIP:I.Data[2]	0		Decimal	DINT
PRESET_Trigger	1		Decimal	BOOL
PRESET_OneShot	1		Decimal	BOOL
PRESET_Value	500		Decimal	DINT
PRESET_Message	{...}	{...}		MESSAGE

- To test the example program, in the **Controller Organizer** enter a value (500 in the example) in the variable **PRESET_Value**.
- Change the variable **PRESET_Trigger** from 0 to 1.

  In the position data **AFM60_EIP:I.Data[1]** the value now changes to 500.

4.5 Test notes

WARNING

Commissioning requires a thorough check by authorized personnel!

Before you operate a system equipped with the AFS60/AFM60 EtherNet/IP for the first time, make sure that the system is first checked and released by authorized personnel. Please read the notes in chapter 2 “On safety” on page 8.

5 Fault diagnosis

This chapter describes how to identify and rectify errors and malfunctions of the AFS60/AFM60 EtherNet/IP Absolute Encoder.

5.1 In the event of faults or errors

WARNING

Cease operation if the cause of the malfunction has not been identified!

Stop the machine if you cannot clearly identify the error and/or if you cannot safely rectify the malfunction.

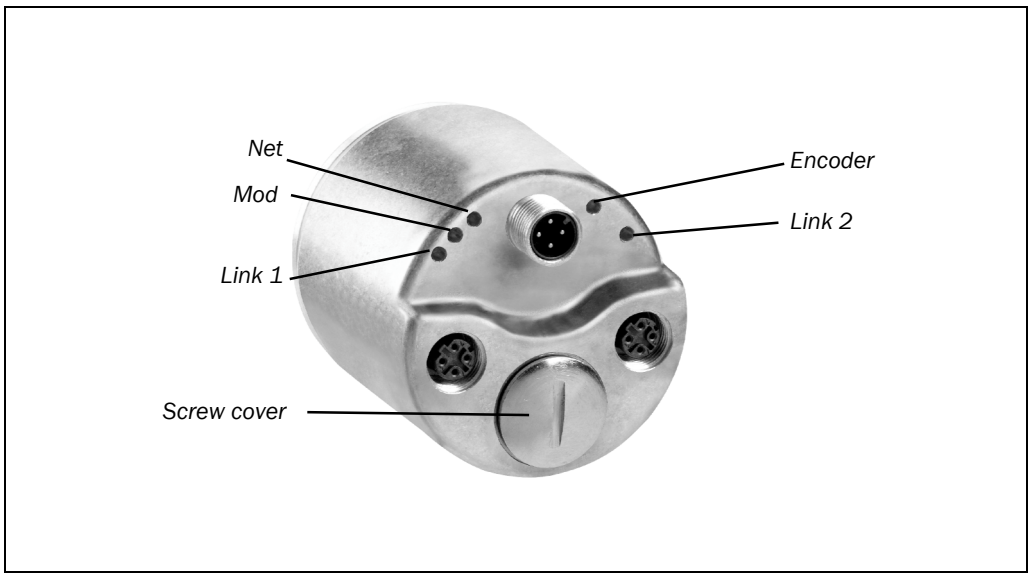
5.2 Support

If you cannot remedy an error with the help of the information provided in this chapter, please contact your local SICK representative.

5.3 Diagnostics

5.3.1 Error and status indications on the LEDs

Fig. 77: Position of the LEDs



Status of the Mod, Net and Encoder LEDs

LED Mod shows the device status, LED Net shows the status of the CIP connection and LED Encoder shows the status of the internal measuring device in the AFS60/AFM60 EtherNet/IP.

Tab. 23: Meaning of the status LEDs Mod, Net and Encoder

Display	Description
LED Mod	
○ Off	No operating voltage
● Green	Device in operation
☀ Green	Stand-by/device not configured, no IP address assigned
☀ Red	Warning, but device still operational or Firmware update in progress
● Red	Error, device not operational
☀ Red/green	Self-test at power-on
LED Net	
○ Off	No operating voltage or No IP address
☀ Green	No connection The device has an IP address but no CIP connection.
● Green	The device has an IP address and a CIP connection.
☀ Red	Warning, connection timeout Cleared by reset or a new connection
● Red	Error IP address has been assigned to another device already.
☀ Red/green	Self-test at power-on
LED Encoder	
○ Off	No operating voltage or No IP address
☀ Green	Warning Wrong parameter
● Green	Device in operation
☀ Red	Warning, but device still operational or Firmware update in progress
● Red	Error Encoder error
☀ Red/green	Self-test at power-on

Ethernet Link LEDs Link 1 and 2

The Ethernet Link LEDs Link 1 and Link 2 display the status of the physical connection on the Ethernet interface.

Tab. 24: Meaning of the LEDs Link 1 and Link 2

Display		Description
○	Off	No operating voltage or No Ethernet connection
●	Green	Ethernet connection established
●	Yellow	Interface port locked
●⁘	Green	Data transmission TxD/RxD
●⁘	Yellow	Data collisions

5.3.2 Self-test via EtherNet/IP

A self-test is available for testing the sensor and the most important functions of the encoder.

Note The self-test is only allowed to be performed while the encoder is at a standstill.

The self-test can be triggered using the diagnostics bit of attribute 13 in the Position Sensor Object (see Tab. 19 on page 28). If an error occurs, bit 27 in the fault header is set (see Tab. 25 on page 67).

On completion of the self-test the diagnostics bit of attribute 13 is reset to 0 automatically.

5.3.3 Warnings, alarms and errors via EtherNet/IP

Within EtherNet/IP warnings, alarms and errors can be retrieved using implicit messages and also explicit messages.

If connections are established via the I/O assembly, the fault header can be read via the instances 101, 102 and 103 (see Tab. 14 on page 23).

Alarms and warnings for the encoder can be read via the Position Sensor Object (see Tab. 19 on page 28) with the aid of the attributes.

For errors, alarms and warnings the following applies:
Bit status = 0: no error, alarm or warning
Bit status = 1: error, alarm or warning present

Fault header

Tab. 25: Fault header

Byte	Bit	Description
0	0	Reserved
	1	Operating temperature of the encoder outside the permissible range
	2	Permissible internal LED current in the sensors exceeded
	3	Supply voltage outside the permissible range
	4	Frequency error, maximum velocity has been exceeded
	5	The velocity has dropped below/exceeded the minimum/maximum velocity configured with attribute 27 or 28 (see Tab. 19 on page 28).
	6	The acceleration has dropped below/exceeded the minimum/ maximum acceleration configured with attribute 32 or 33 (see Tab. 19 on page 28).
	7	The position has dropped below/exceeded the minimum/maximum position configured with attribute 22 or 23 (see Tab. 19 on page 28).
1	8	Position error (amplitude error of the singleturn measurement)
	9	Position error (amplitude error of the multiturn measurement)
	10	Position error (vector error $Sin^2 + Cos^2$ of the singleturn measurement)
	11	Position error (vector error $Sin^2 + Cos^2$ of the multiturn measurement)
	12 ... 15	Reserved
2	16	Singleturn position error (error in the sensor)
	17	Multiturn position error (synchronization MA single)
	18	Multiturn position error (synchronization quad single)
	19	Multiturn position error (internal interface)
	20	Multiturn position error (FRAM)
	21 ... 23	Reserved
3	24	Memory error (EEPROM checksum)
	25	Memory error (EEPROM IRQ)
	26	Error on start-up
	27	Error during self-test
	28 ... 29	Reserved
	30	LifeSign. Active if attribute 13 is set (see Tab. 19 on page 28)
	31	Reserved

Alarms

If, for example, the internal self-test detects that the position value has been incorrectly calculated or an incorrect configuration value has been transferred to the encoder, the alarm flag is set (attribute 46, see Tab. 19 on page 28).

WARNING

It is imperative to evaluate the alarms in your application!

In case of a serious error, incorrect position values may be output. This change could cause an unexpected movement that may result in a hazard for persons or damage to the system or other objects.

● **Red** In addition the S3 LED illuminates red continuously.

The alarm type is coded in a bit field of attributes 44 and 45.

Tab. 26: Alarms

Bit	Description
0	Position error
1	Diagnostics error during self-test
2 ... 11	Reserved
12	Incorrect checksum (vendor specific)
4	Error on system start-up (vendor specific)
14 ... 15	Reserved

Warnings

If, for example, the velocity or temperature drop below/exceed the limit values, the warning flag is set (attribute 49, see Tab. 19 on page 28).

Red In addition the S3 LED flashes red.

The warning type is coded in a bit field of attributes 47 and 48.

Note The position value will continue to be correctly calculated, the encoder is therefore still ready for operation.

Tab. 27: Warnings

Bit	Description
0	Maximum velocity exceeded
1	Permissible internal LED current in the sensors exceeded
2 ... 5	Not supported
6	The velocity has dropped below the minimum velocity configured with attribute 27.
7	The velocity has exceeded the maximum velocity configured with attribute 28.
8	The acceleration has dropped below the minimum acceleration configured with attribute 32.
9	The acceleration has exceeded the maximum acceleration configured with attribute 33.
10	The position has dropped below/exceeded the minimum/maximum position configured with attribute 22 and 23.
11 ... 12	Reserved
13[8)]	The temperature has dropped below/exceeded the minimum/maximum temperature configured with attribute 103 and 104
14[8)]	The operating voltage has dropped below/exceeded the minimum/maximum operating voltage.

8) Vendor specific warning.

6 Annex

6.1 Conformities and certificates

You can obtain declarations of conformity, certificates, and the current operating instructions for the product at www.sick.com. To do so, enter the product part number in the search field (part number: see the entry in the “P/N” or “Ident. no.” field on the type label).

6.1.1 Compliance with EU directives

EU declaration of conformity (extract)

The undersigned, representing the manufacturer, herewith declares that the product is in conformity with the provisions of the following EU directive(s) (including all applicable amendments), and that the standards and/or technical specifications stated in the EU declaration of conformity have been used as a basis for this.

6.1.2 Compliance with UK statutory instruments

UK declaration of conformity (extract)

The undersigned, representing the following manufacturer herewith declares that this declaration of conformity is issued under the sole responsibility of the manufacturer. The product of this declaration is in conformity with the provisions of the following relevant UK Statutory Instruments (including all applicable amendments), and the respective standards and/or technical specifications have been used as a basis.

6.2 List of tables

6.3 List of illustrations

Add ITIo NAI I NFor MATIo N

AFS60 / AFM60 EtherNet / IP WEB

WEB and FTP functionality for EtherNet / IP Encoder

GB

SICK
Sensor Intelligence.

This documentation applies to the WEB and FTP functionality for EtherNet/IP Encoder, release version 0.02, release date September XX, 2013 and is an additional document to the AFS60/AFM60 EtherNet/IP Operating Instruction, part no. 8018909

Subject to change without notice.

SICK AG accepts no responsibility for the non-infringement of patent rights, e. g. in the case of recommendations for circuit designs or processes.

Data integrity: SICK AG uses standardized data interfaces,
such as standard IP technology, in its products.
The emphasis here is on the availability of products and their features. SICK AG always assumes that the integrity and the confidentiality of the data and rights which are affected by the use of these products will be ensured by the customer. In all cases, appropriate security measures, such as network separation, firewalls, virus protection and patch management, must be taken by the customer on the basis of the situation in question.

The trade names listed are the property of the relevant companies.

SICK AG
Erwin-Sick-Str. 1
79183 Waldkirch
Deutschland
www.sick.com info@sick.de

Made in Germany, 2013.

Table of contents

1. Assembly object

q. v. Operating instruction chapter 3.4.2

Note:
The encoder support in addition to “input” and “listen-only”,
the “exclusive owner” connection.

Table 1. Instance-attribute of input assembly object

Number	Connection	Description	Bits	Bytes
104	Input	Fault	32	4
		Position value	32	4
		Velocity	32	4
		Serial number	32	4
		CPR value	32	4
		CMR value	32	4
		cw / ccw	32	4
		scf	32	4
		raf	32	4
		CNR_N	32	4
		CNR_D	32	4
		Velocity format	32	4
		Preset Value	32	4

Table 2. Instance-attribute of output assembly object

Number	Connection	Description	Bits	Bytes
106	Output	Preset Value	32	4
		Sync Preset Value	32	4

Note:
The attribute 104 and 106 are vendor specific.

1.1. I/O Assembly

Table 3. Format of input assembly 104

Instance	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
104	0	Fault header (least significant byte)							
	1	Fault header							
	2	Fault header							
	3	Fault header (most significant byte)							
	4	Position value (least significant byte)							
	5	Position value							
	6	Position value							
	7	Position value (most significant byte)							
	8	Velocity value (least significant byte)							
	9	Velocity value							
	10	Velocity value							
	11	Velocity value (most significant byte)							
	12	Serial number value (least significant byte)							
	13	Serial number value							
	14	Serial number value							
	15	Serial number value (most significant byte)							
	16	CPR value (least significant byte)							
	17	CPR value							
	18	CPR value							
	19	CPR value (most significant byte)							
	20	CMR value (least significant byte)							
	21	CMR value							
	22	CMR value							
	23	CMR value (most significant byte)							
	24	cw-ccw value (least significant byte)							cw/ccw [1]
	25	cw-ccw value							
	26	cw-ccw value							
	27	cw-ccw value (most significant byte)							
	28	Scaling function value (least significant byte)							scf [2]
	29	Scaling function value							
	30	Scaling function value							
	31	Scaling function value (most significant byte)							
	32	Round axis function value (least significant byte)							raf [2]
	33	Round axis function value							
	34	Round axis function value							
	35	Round axis function value (most significant byte)							
	36	CNR_N value (least significant byte)							
	37	CNR_N value							
	38	CNR_N value							
	39	CNR_N value (most significant byte)							
	40	CNR_D value (least significant byte)							
	41	CNR_D value							

Instance	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
104	42	CNR_D value							
	43	CNR_D value (most significant byte)							
	44	Velocity format value (least significant byte)							
	45	Velocity format value							
	46	Velocity format value							
	47	Velocity format value (most significant byte)							
	48	Preset Value (least significant byte)							
	49	Preset Value							
	50	Preset Value							
	51	Preset Value (most significant byte)							

[1] cw = clockwise
ccw = counterclockwise
[2] scf = scaling function
[3] raf = round axis functionality

The input assembly 104 contain the transmission of the serial number from the encoder. This can be used in case of exchange the encoders, due to e. g. a fault/defect, to compare wether it is a new device or the still existing because of different parameter settings. If the encoders get the same IP-addressing, the unique serial number could protect an start up with wrong parameters – and so prevent crashs or damages, also save cost and time.

1.2. Output I/O assembly attribute data format

Table 4. Format of output assembly 106

Instance	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
106	0	Preset Value (low byte)							
	1	Preset Value							
	2	Preset Value							
	3	Preset Value (high byte)							
	4	Sync Preset Value (low byte)							
	5	Sync Preset Value							
	6	Sync Preset Value							
	7	Sync Preset Value (high byte)							

Table 5. Description of output assembly 106

Byte	Name	Description	Standard
0-3	Preset Value	The Preset Value is transferred to the encoder and saved.	0
4-7	Sync Preset Value	Sync Preset Value	1

Function of sync Preset Value

The setting is done not before this value changes from “0” to “1”. The Preset Value can be reset.
(Only a change in the Preset Value is not accepted, even if the network connection disconnects and reconnects.)

Preset

The preset function is used to set the encoder to a predefined start position. With the aid of a Preset Value the encoder can be set to any position within the measuring range.
The Preset Value can be set in the following manner:

- Using the preset push-button (encoder)
- Using an acyclic explicit message (PLC)
  (During this process the Preset Value is transferred as an attribute (ID19) of the position sensor object)
- Using the WebServer preset (WebBrowser)
- Using the PLC preset (output assembly 106)

Warning!
Only set a Preset Value when the encoder is at standstill

Note:
By using the preset function via output assembly 106, the sync Preset Value (Byte 4–7) has to be set from “0” to “1”. This rising edge is the trigger for Preset Value acceptance.

The preset function results in an immediate change in the position value output by the encoder. This change could cause an unexpected movement that may result in a hazard for persons or damage to the system or other items. See AFS60/AFM60 EtherNet/IP Operating instruction, chapter 3.5.7 page 36.

2. Embedded WebServer

The encoder is equipped by embedded WebServer implementation with dynamic HTML sites. This web interface allows the programming of the sensor without the need of special skills of the programming interface.

2.1. Implementation

User interface integrated. This allows diagnostics and programming as part of the AFM-EIP web implementation.

2.2. Implementation details

There are 4 websites and a password dialogue.

Names of the websites and the password dialogue:

- Diagnosis (home)
- Parameterization
- Set Preset Value
- Set to Factory Settings
- Password

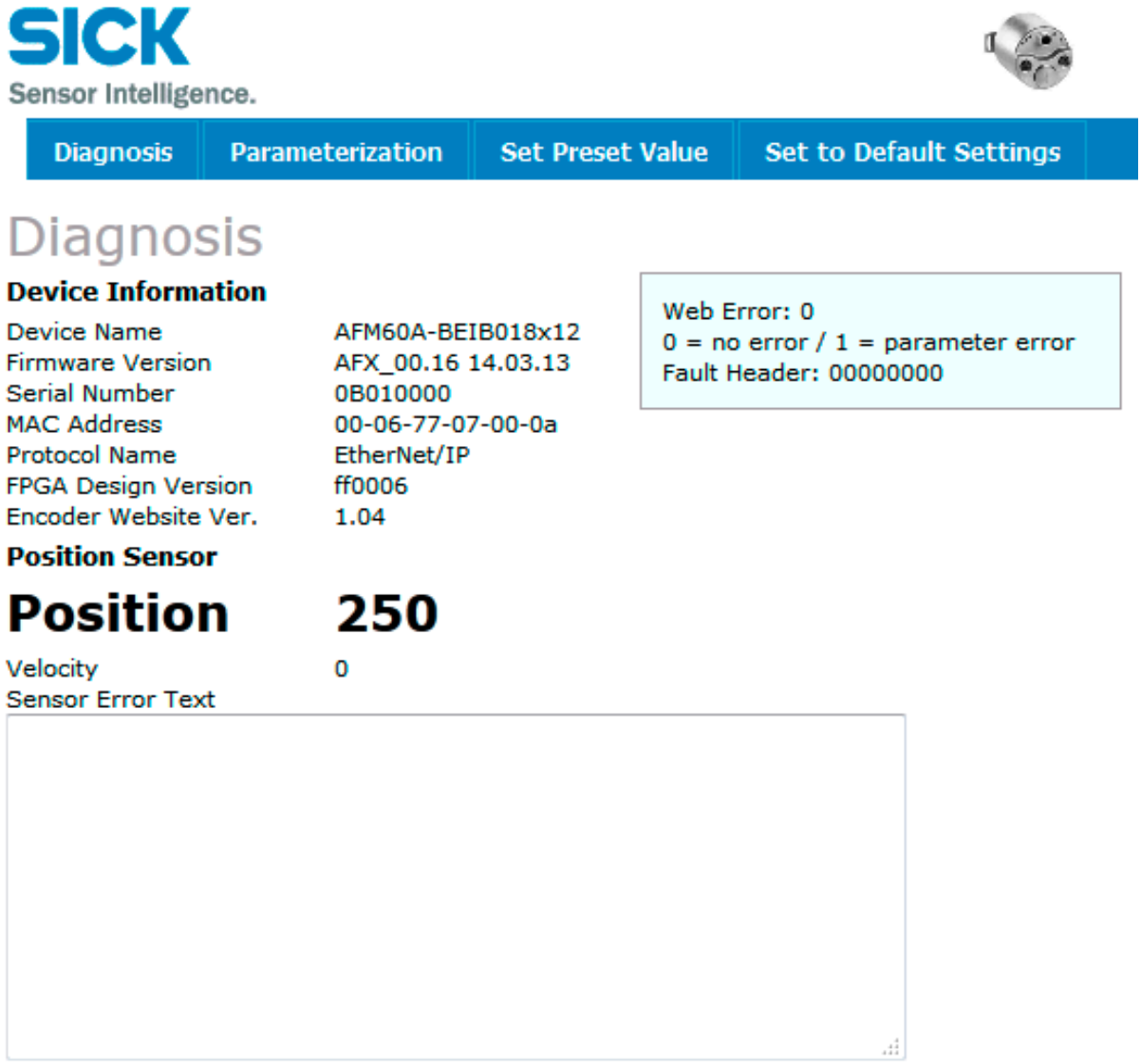
Attention!
To display the websites of the encoder correctly,
please activate Java Script in your webbrowser and maybe adapt your security settings.

Enter the IP-address of the encoder in the webbrowser e. g. “http://192.168.1.123”.
The home page “Diagnosis” need no password.

Attention!
To get access to other web sites, in a dialog, the password must be entered.

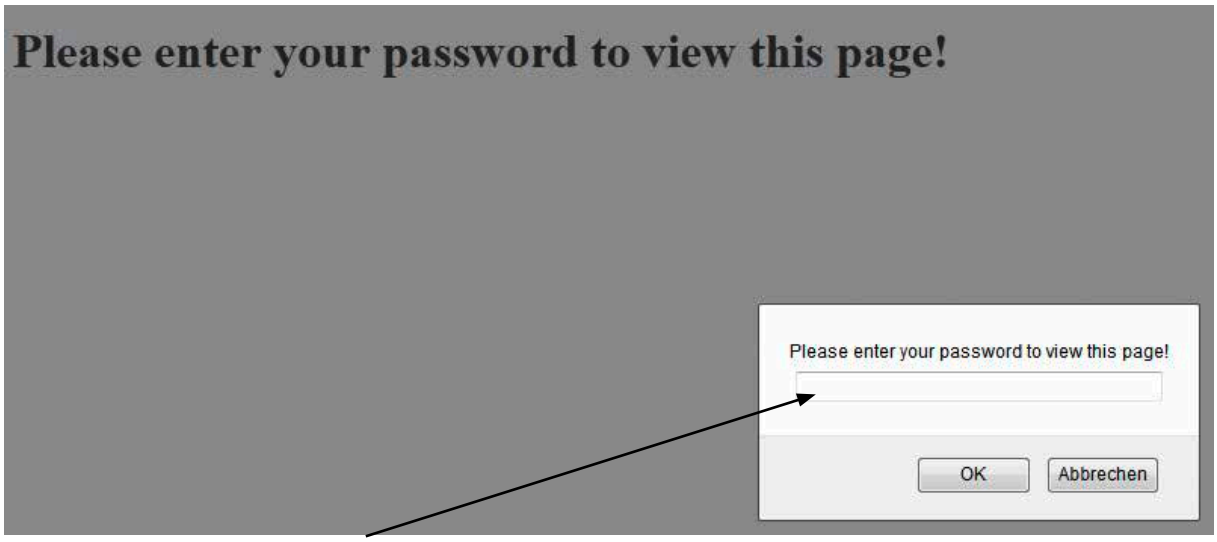
2.3. Diagnosis

The Diagnosis page is going to be updated every 2 seconds by HTML meta refresh.



In the text box <Sensor Error Text> shows the last three faults from the fault header analysis.

2.4. Password for webpages

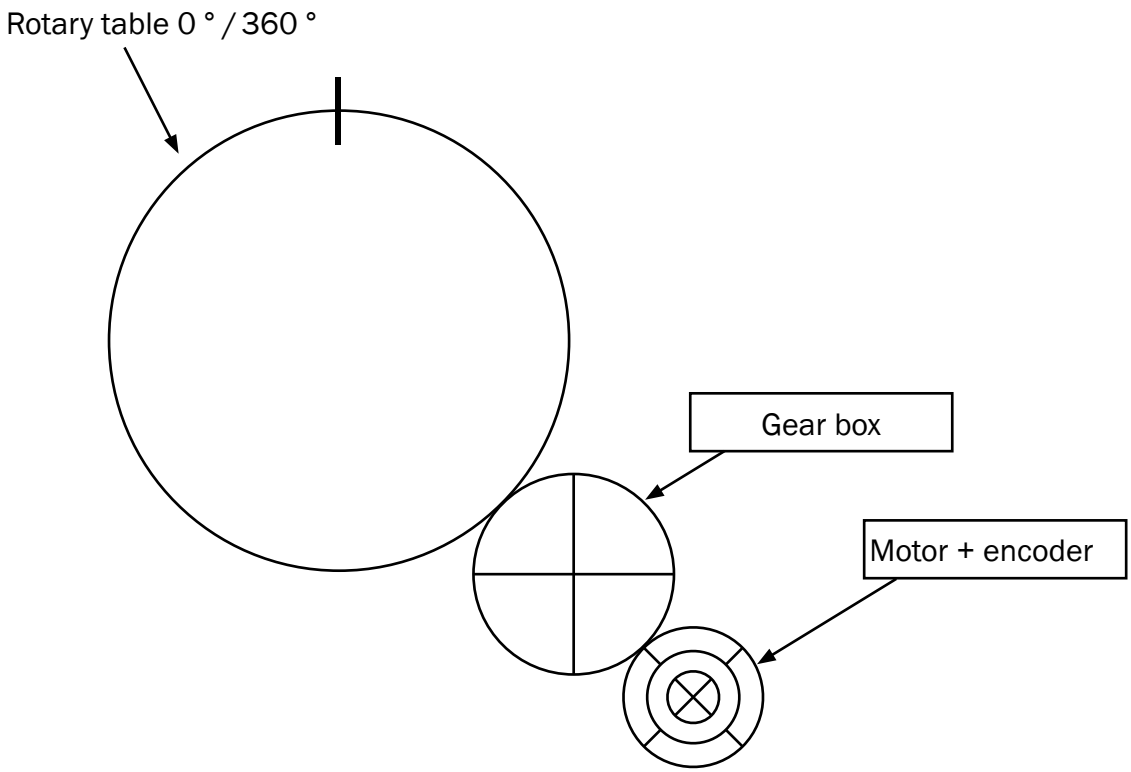


Enter the password: **sickP**
Press OK.

2.5. Device Parameterization

On this page the values can be changed and programmed.
The new configuration is permanent stored in an EEPROM, please press the <Save Changes> button, or use the <ENTER> key to save the parameters.

Application example: Round axis functionality (endless shaft)



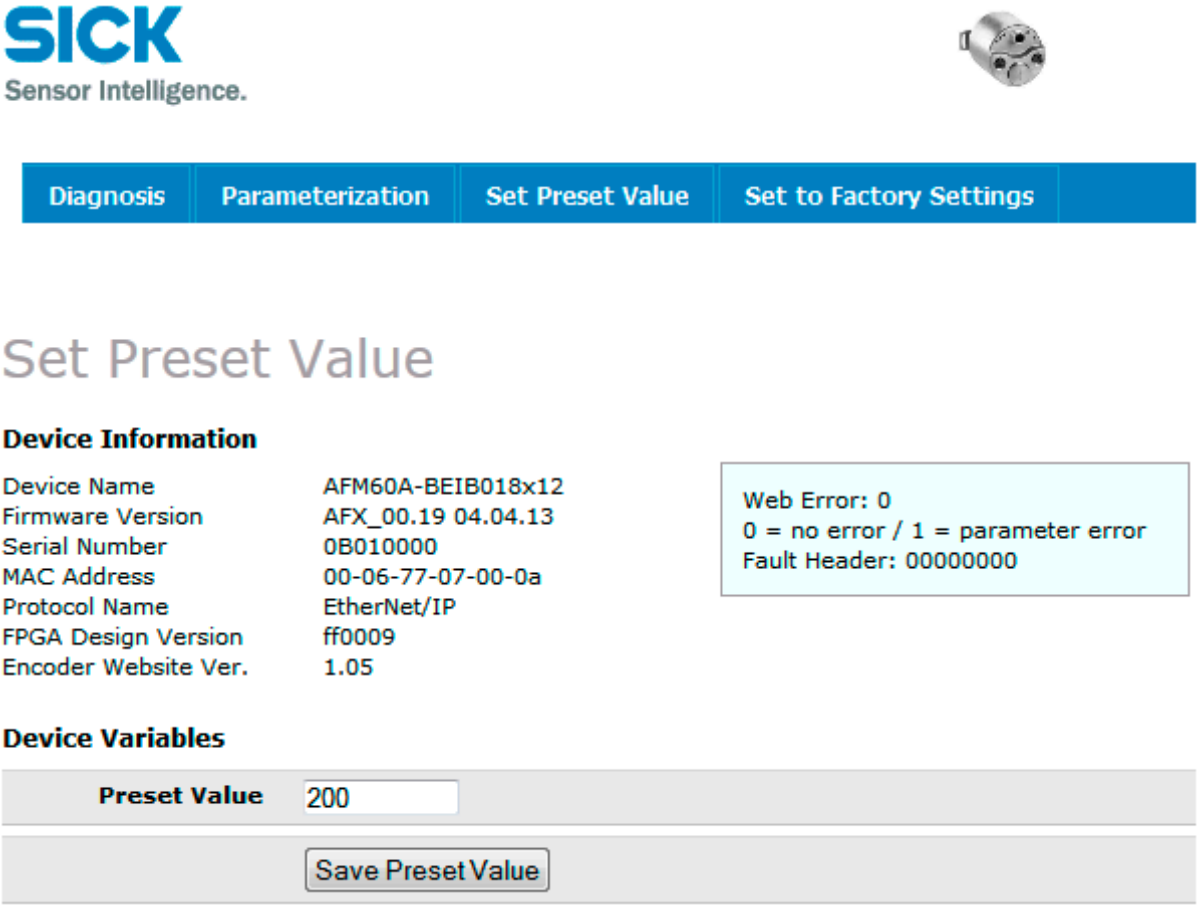
Given:
13.7 turns of the encoder = 1 turn of rotary table
Target: 0.1 ° resolution
Total resolution (measuring range) = 3600
Calculation path: 137 (Nominator)
10 (Divisor)

Screenshot of the “Parameterization” page

Device Variables

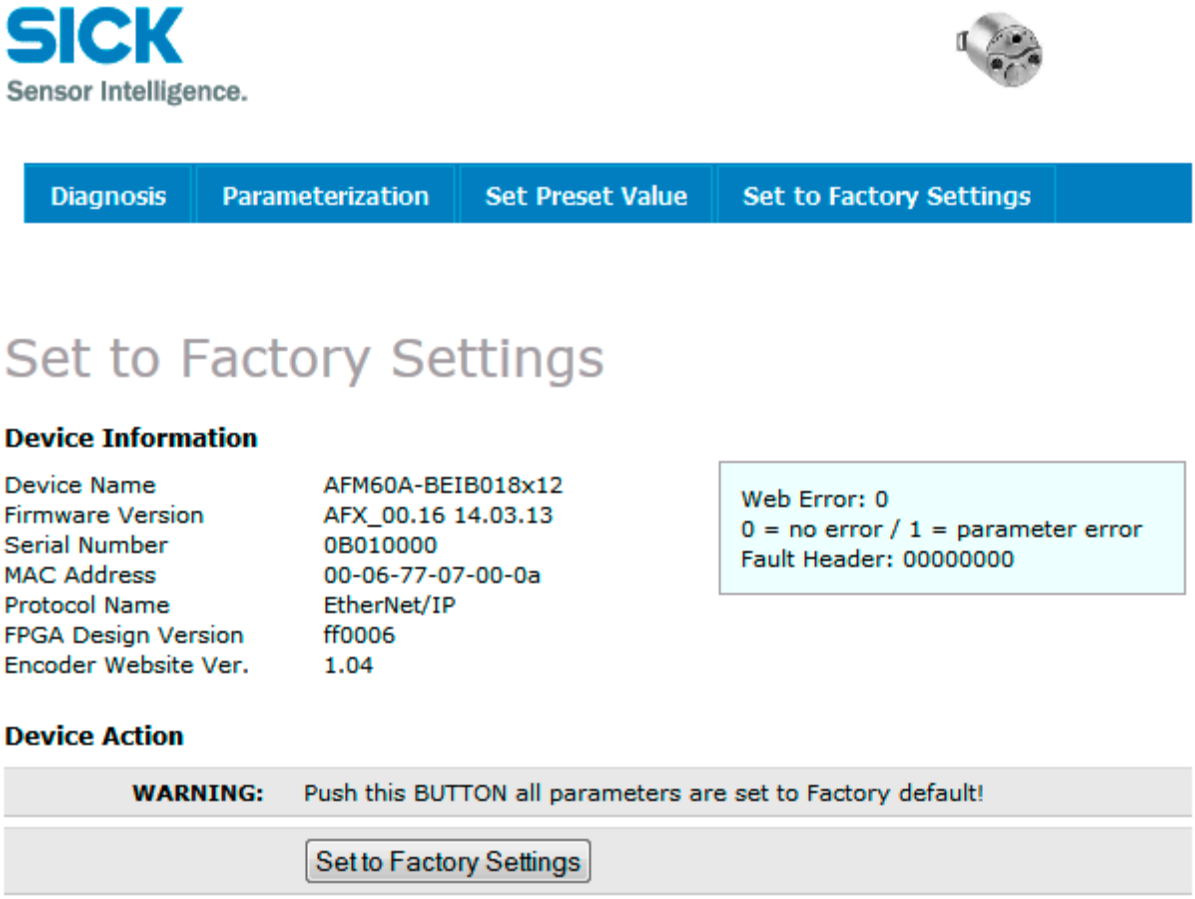
Count Direction	(•) cw ( ) ccw
Steps per Revolution	262
Total Resolution / Measuring Range x	3600
Scaling Function	( ) off (•) on
Endless Shaft Functionality	( ) off (•) on
Nominator / Divisor	137 / 10 = 13.700
Velocity Format	turns/min

2.6. Set Preset Value



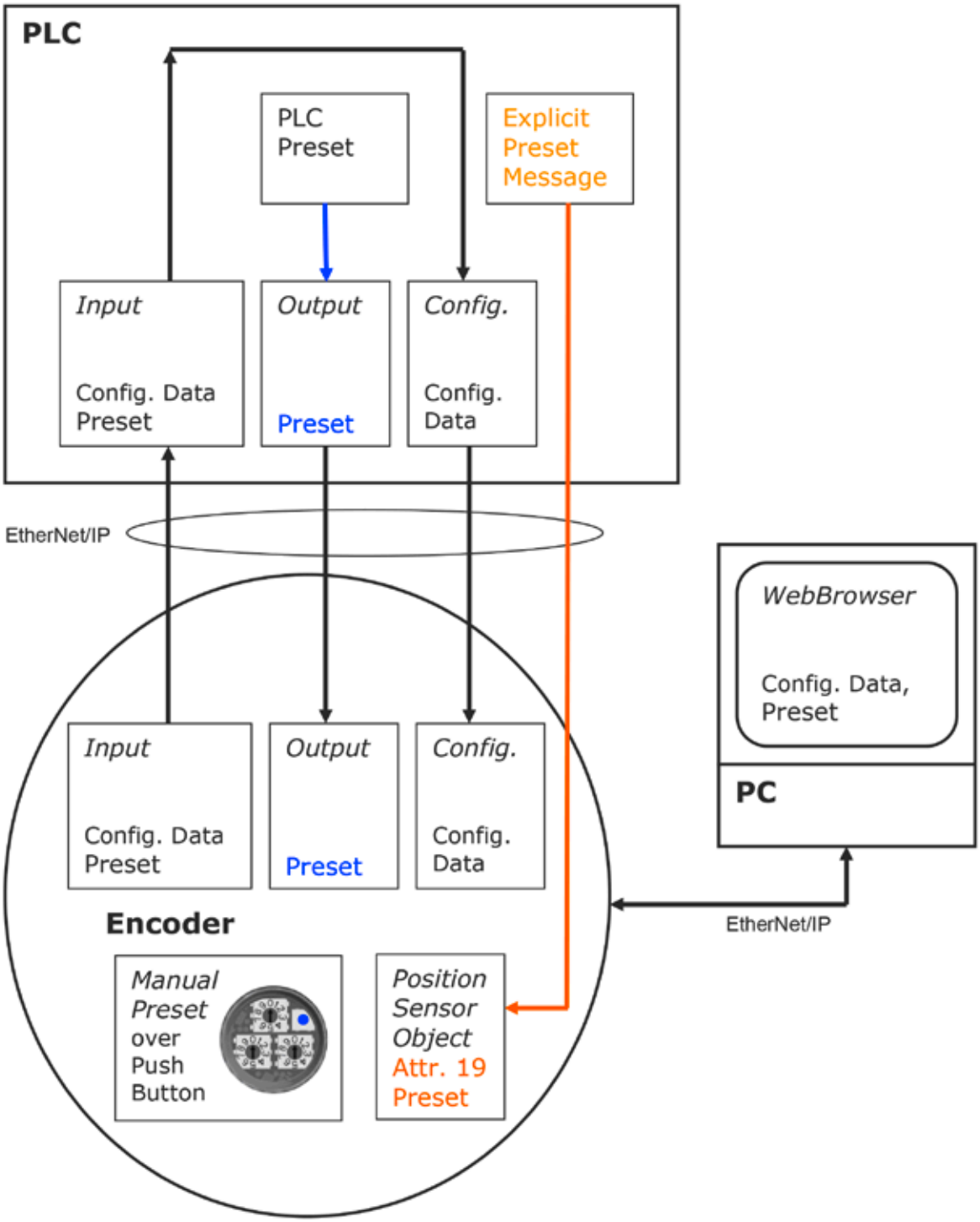
On this page you can change the Preset Value. Click on the button <Save Preset Value> or <ENTER> and the Preset Value is set into the encoder. The controller stores the value in his configuration (by programmed data mapping, see program-sample).

2.7. Set to Factory Settings



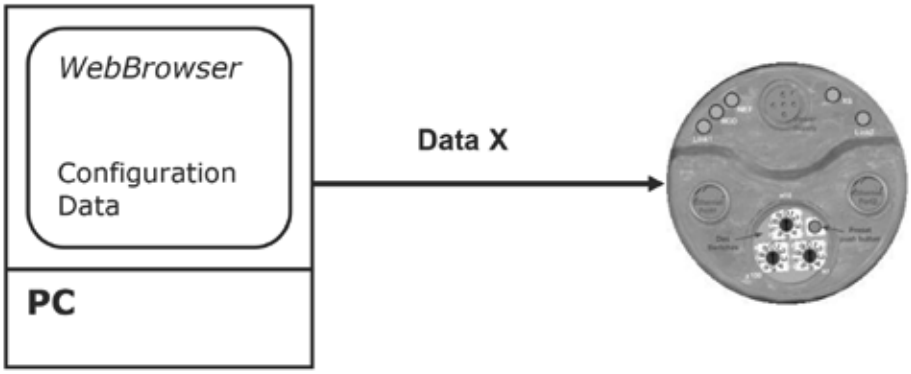
On this page you can switch back to the Factory default Settings.

3. AFx60 EtherNet / IP WEB – functionality overview

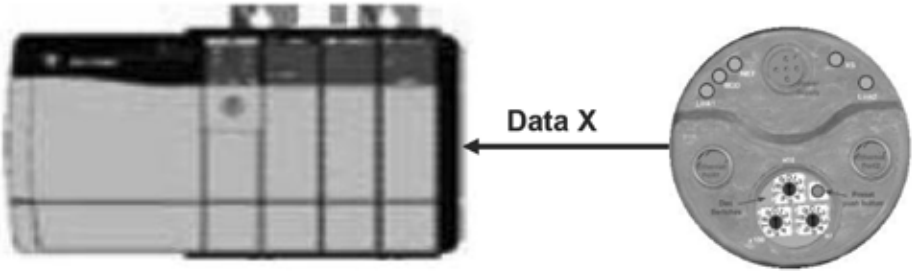


3.1. AFx60 EtherNet / IP WEB – configuration overview

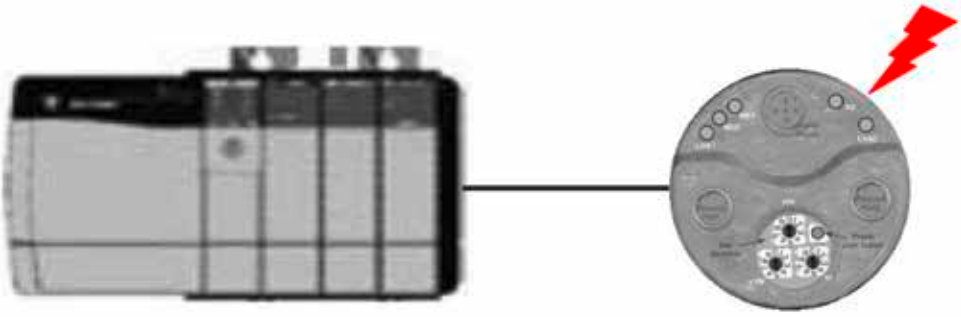
3.1.1. Initial configuration over WebServer



3.1.2. Get Configuration Data to PLC (configuration assembly length = 0)

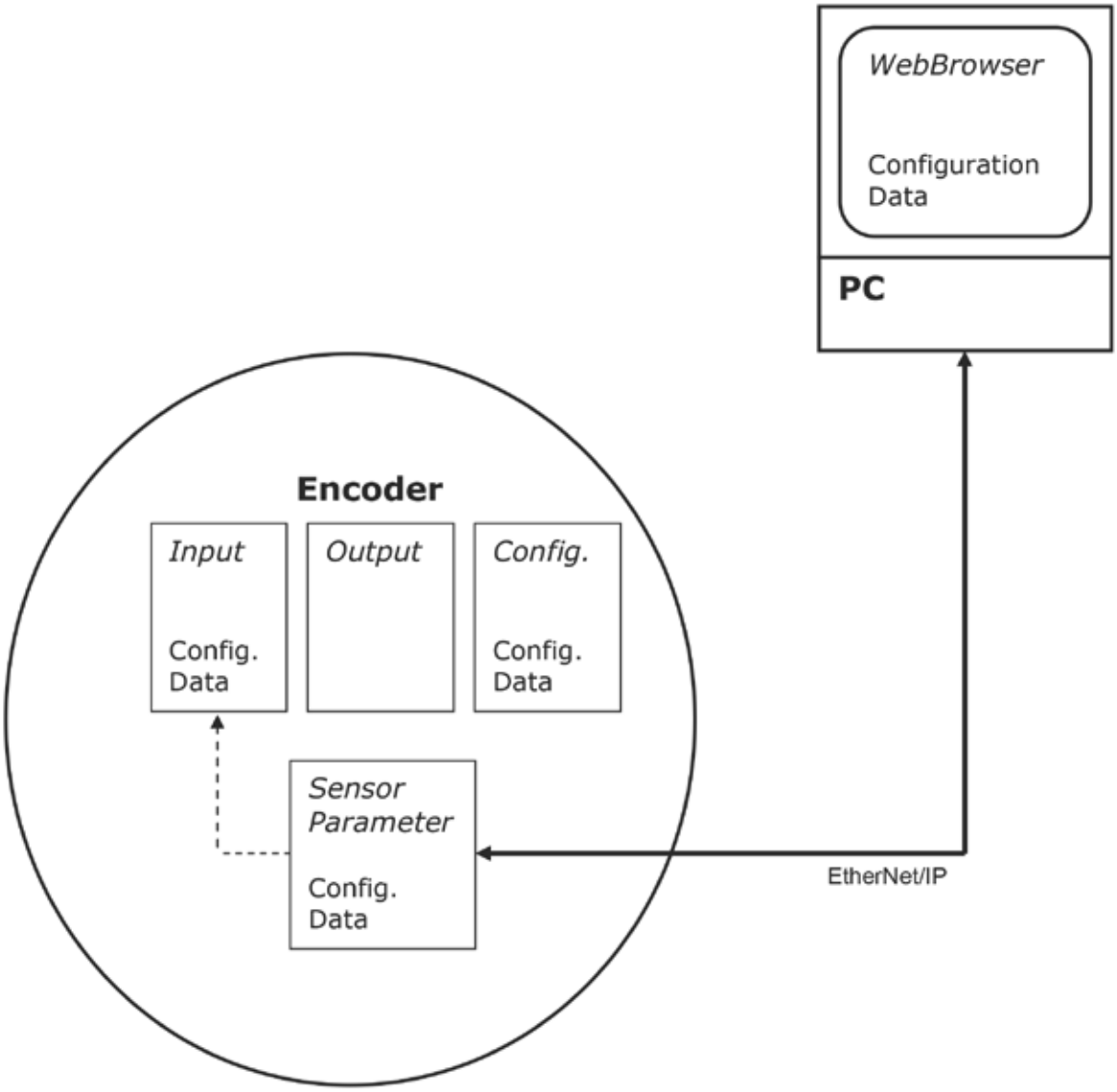


3.1.3. Encoder damage / blackout

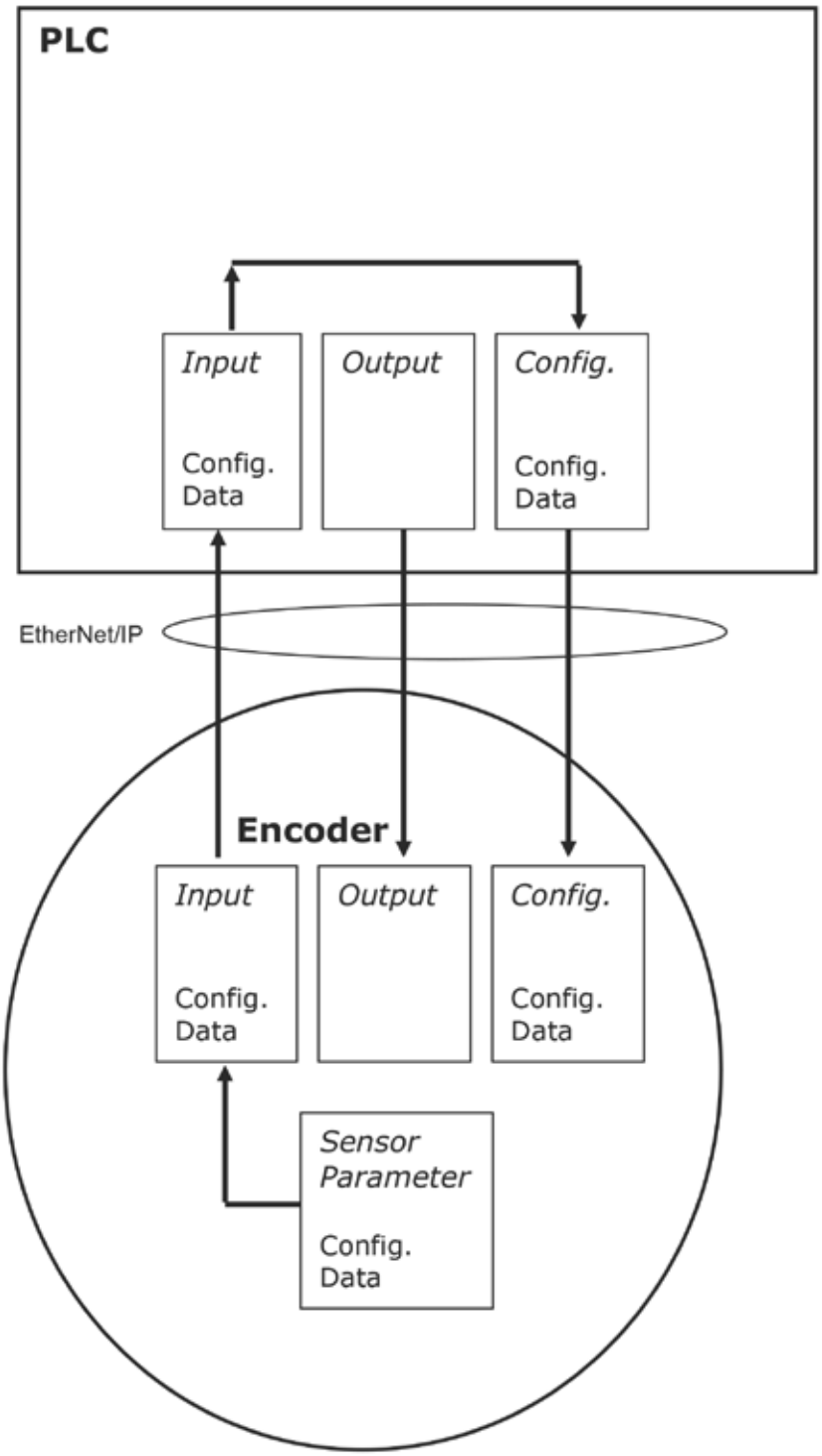


3.1.4. Set Configuration Data to new encoder (configuration assembly length = 28)

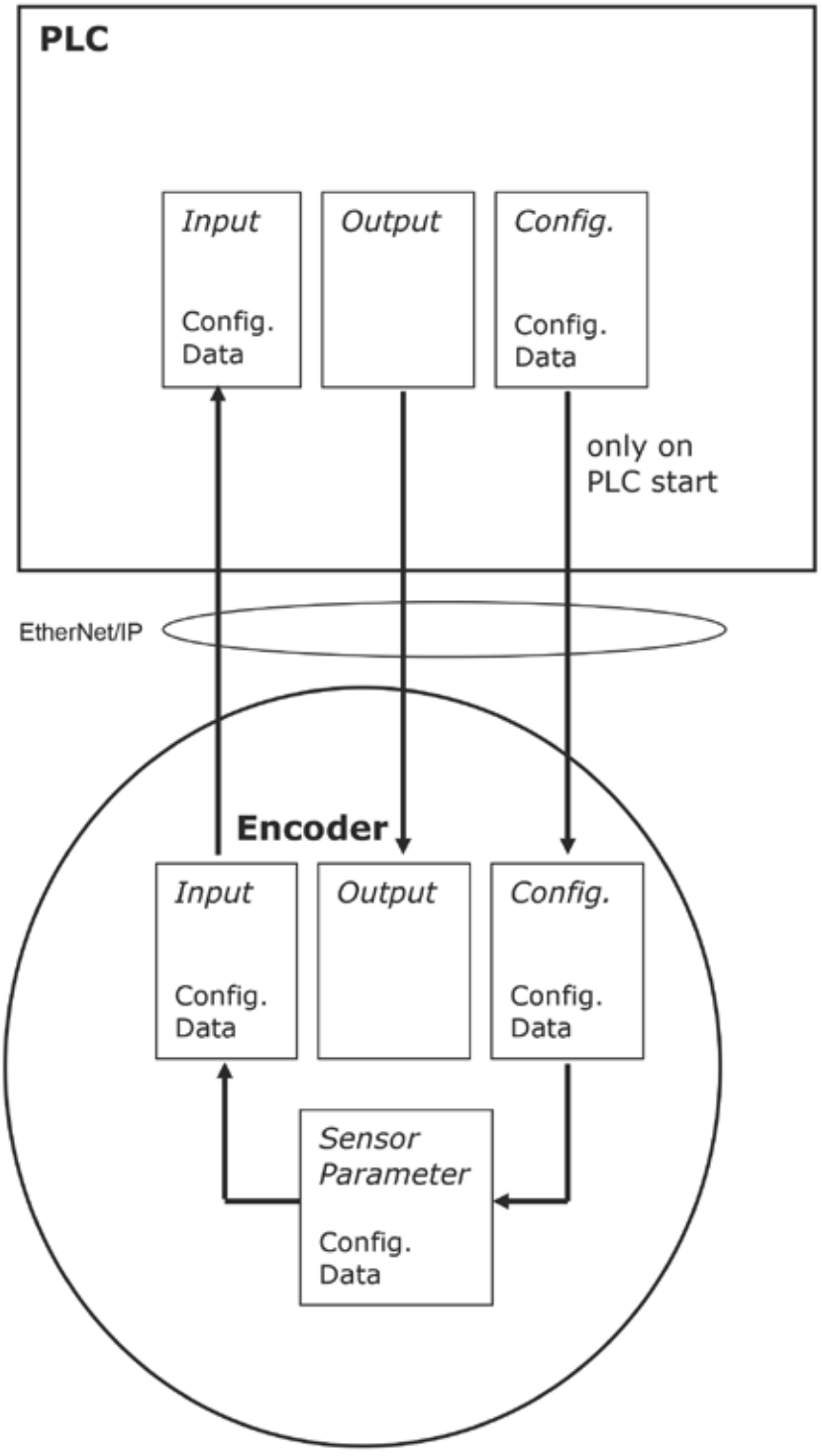
3.1.1. AFx60 EtherNet/IP WEB – configuration only over WebServer



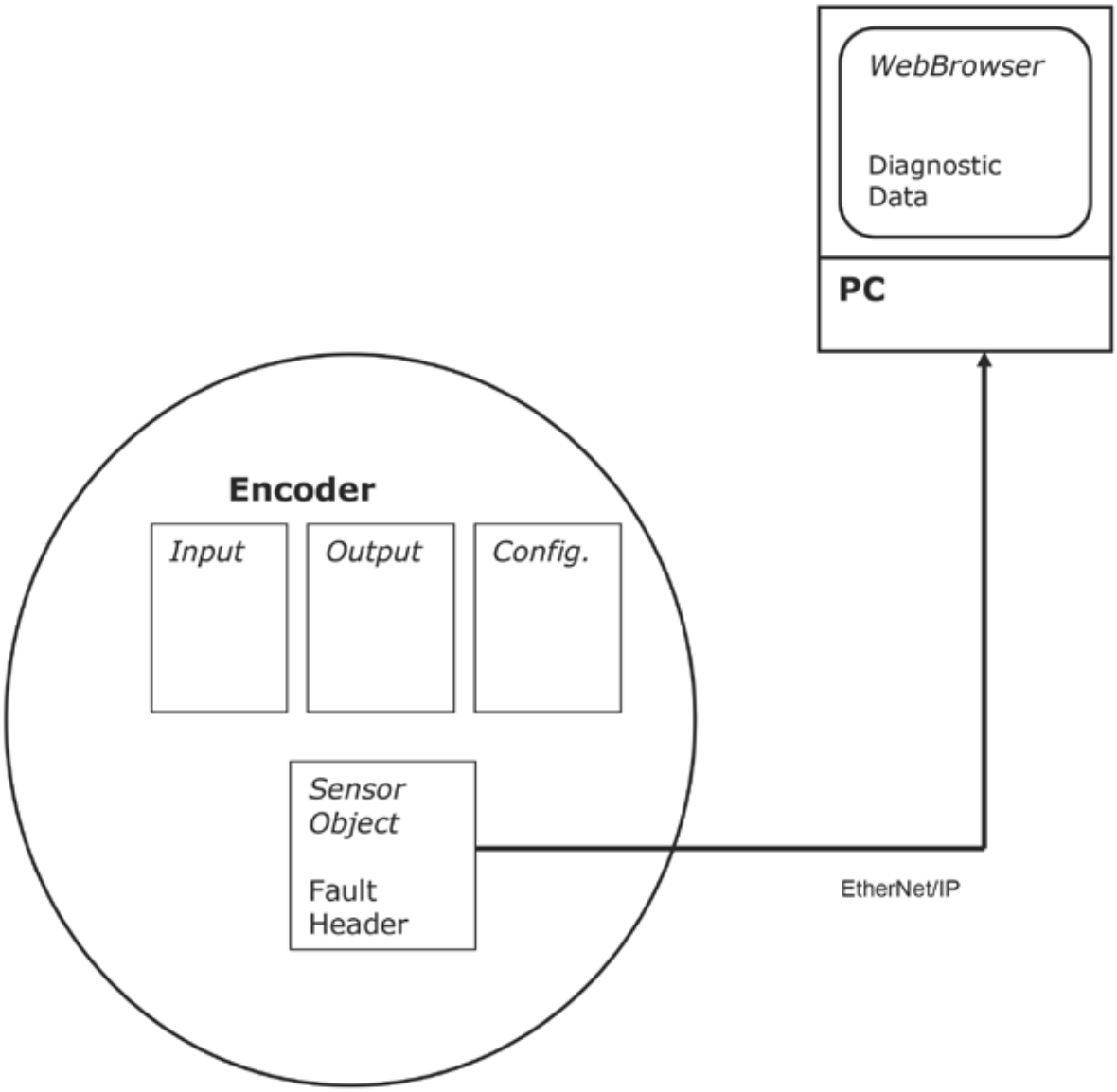
3.1.2. AFx60 EtherNet/IP WEB – get Configuration Data to PLC (configuration assembly length = 0)



3.1.4. AFx60 EtherNet/IP WEB – set Configuration Data to encoder (configuration assembly length = 28)



3.2. AFx60 EtherNet/IP WEB – Diagnostic Data



Note:
Works with or without connected PLC

3.3. Preset warning

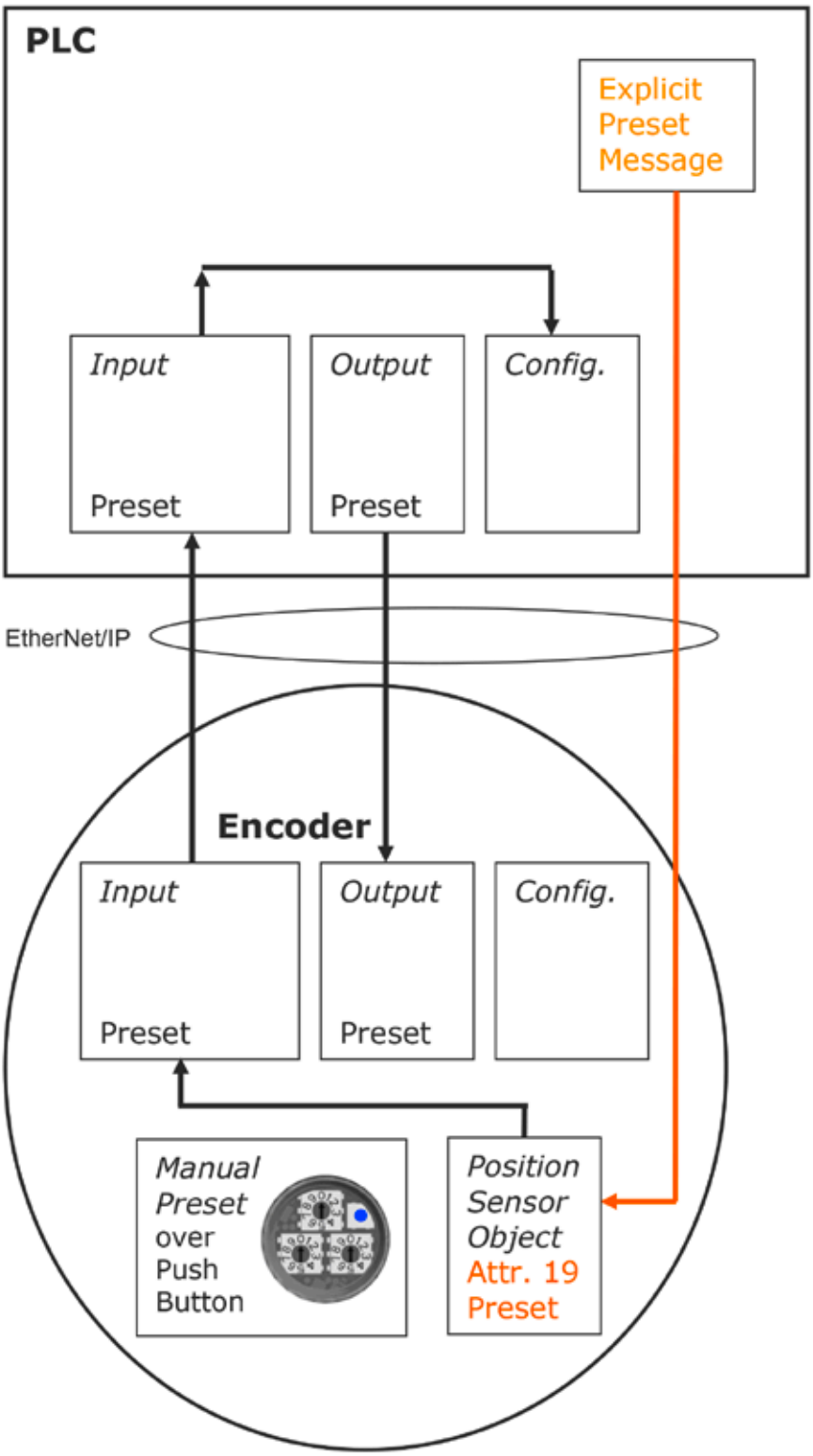
The preset function is used to set the encoder to a predefined start position. With the aid of a Preset Value the endcoder can be set to any position within the measuring range. The Preset Value can be set in the following manner:

- Using the preset push-button
- Using an acyclic explicit message. During this process the Preset Value is transferred as an attribute (ID19) of the position sensor object.
- Using the WebServer preset (output assembly 106)
- Using the PLC preset (output assembly 106)

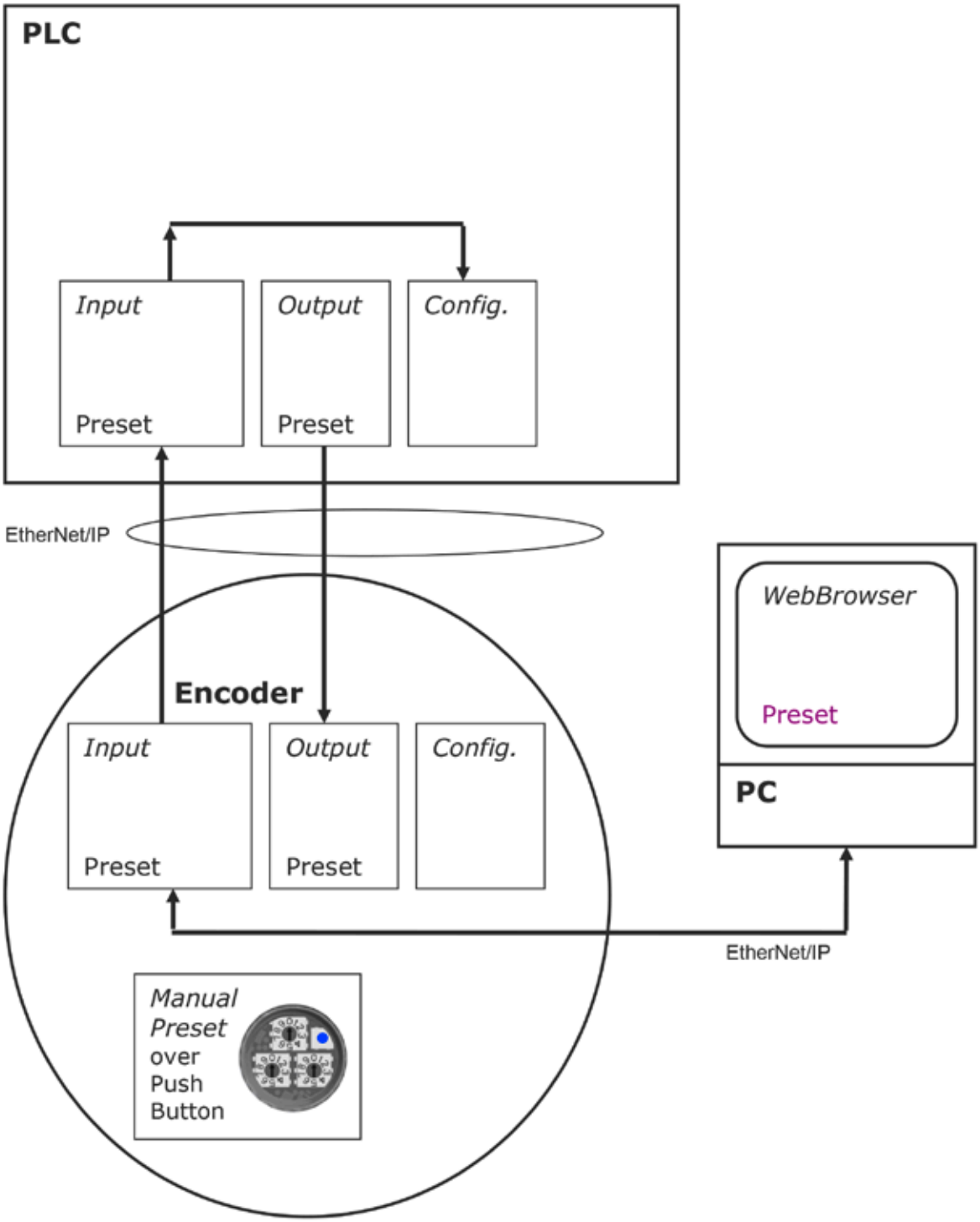
Note:
Only set a Preset Value when the encoder is at standstill

Warning!
The preset function results in an immediate change in the position value output by the encoder. This change could cause an unexpected movement that may result in a hazard for persons or damage to the system or other items.

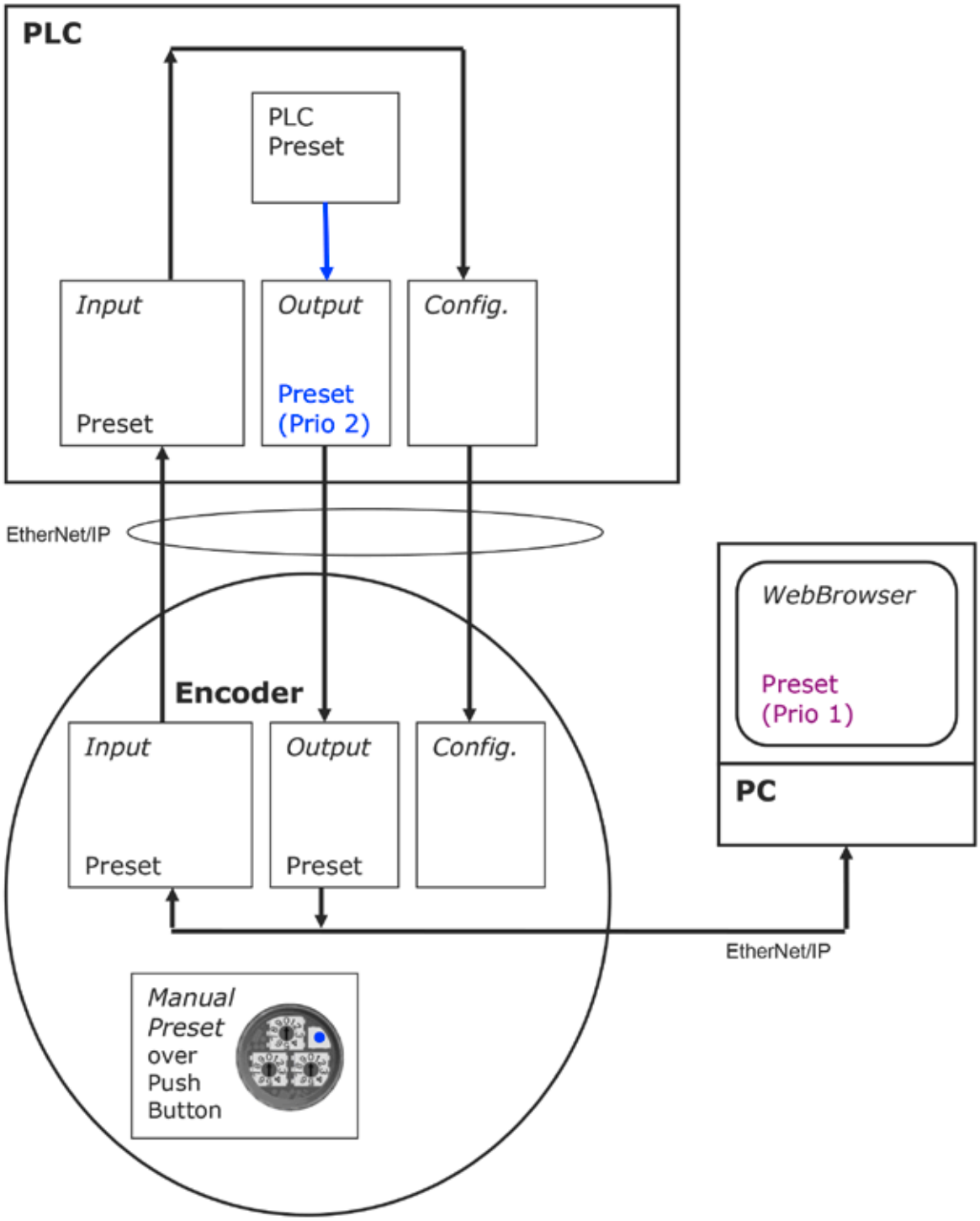
3.3.1. AFx60 EtherNet/IP WEB – Preset Message



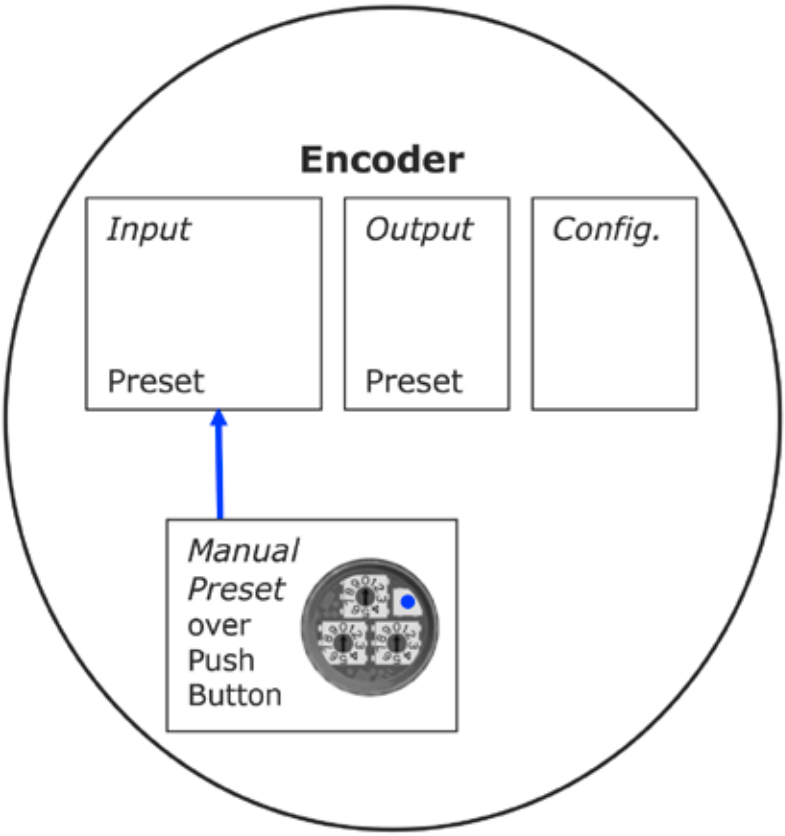
3.3.2. AFx60 EtherNet/IP WEB – WebServer preset



3.3.3. AFx60 EtherNet/IP WEB – manual PLC Preset

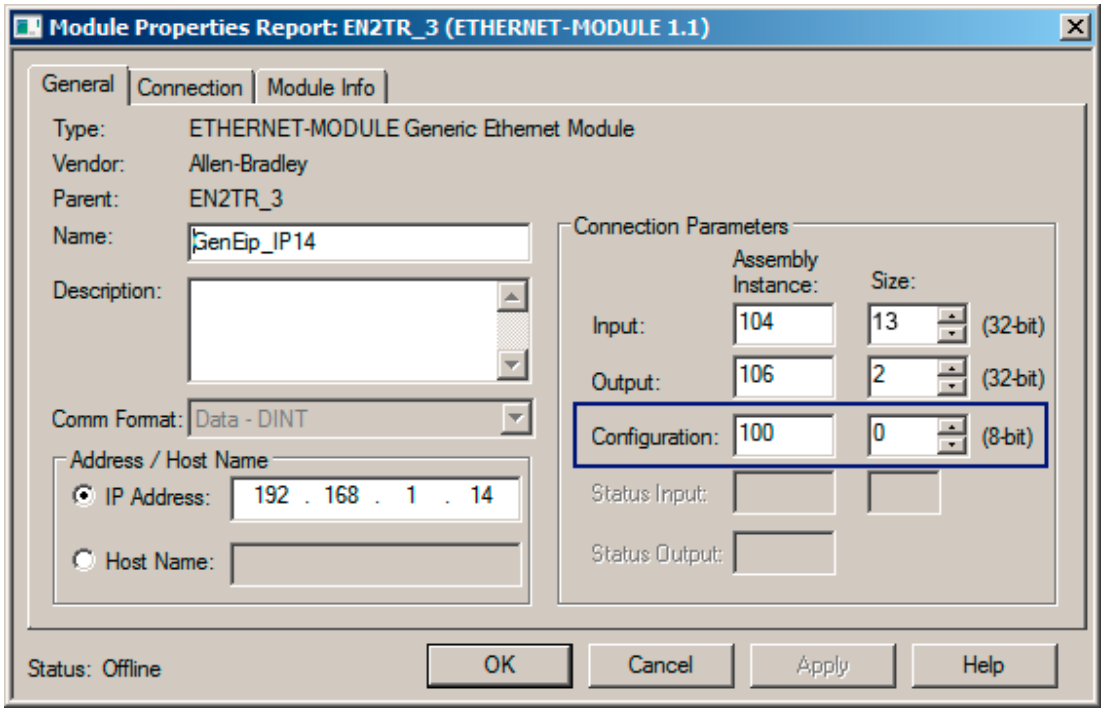


3.3.4. AFx60 EtherNet/IP WEB – Button Preset



4. Integration of AFx60 EtherNet/IP WEB encoder to the RS logix project

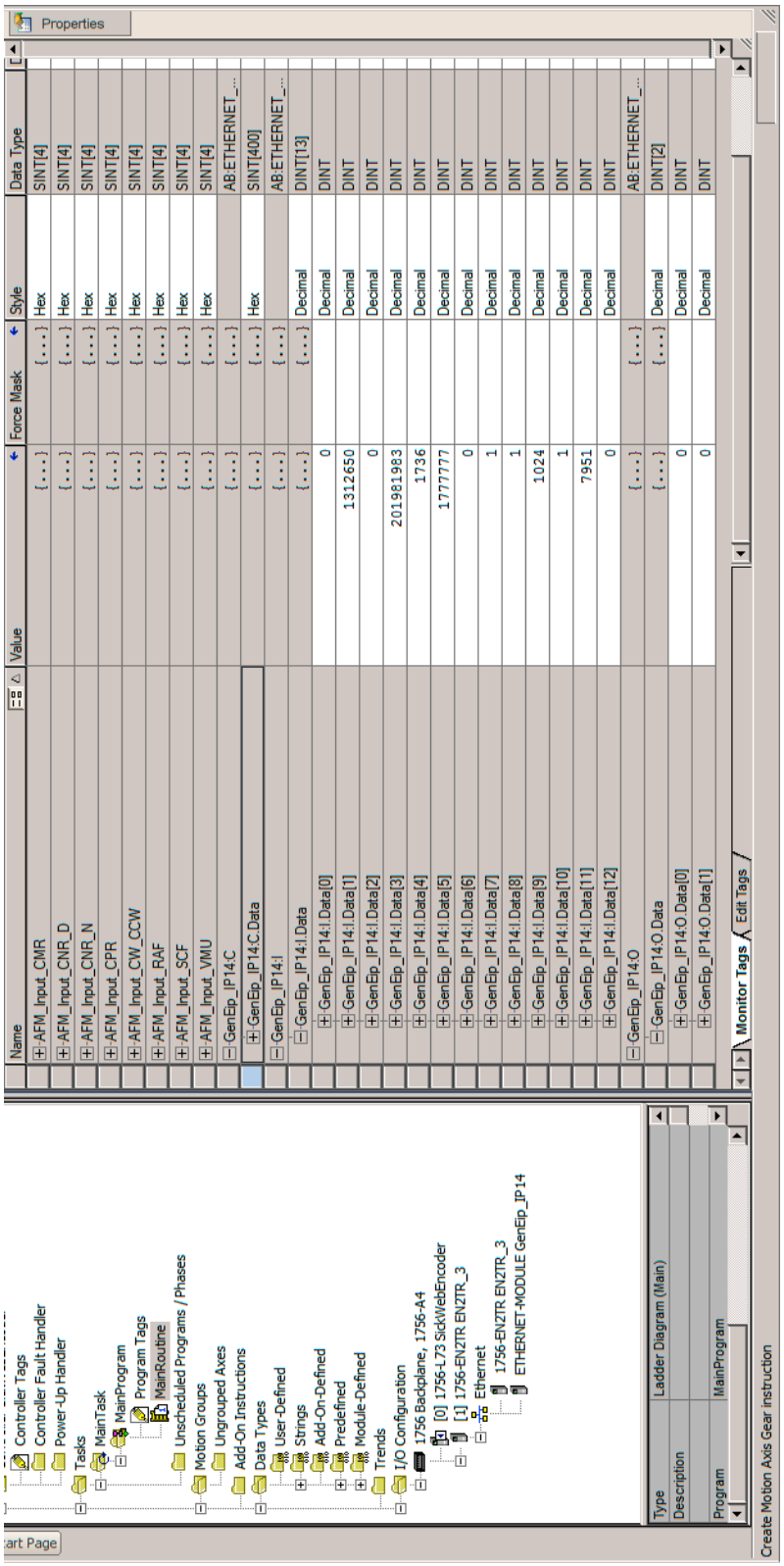
Setup the RS logix project and integrate the AFx60 EtherNet/IP WEB encoder as described in the AFS60/AFM60 EtherNet/IP Operating Instruction, part no. 8018909, chapter “4.3 configuration”.
In this example the length of configuration assembly is set to 0 byte.



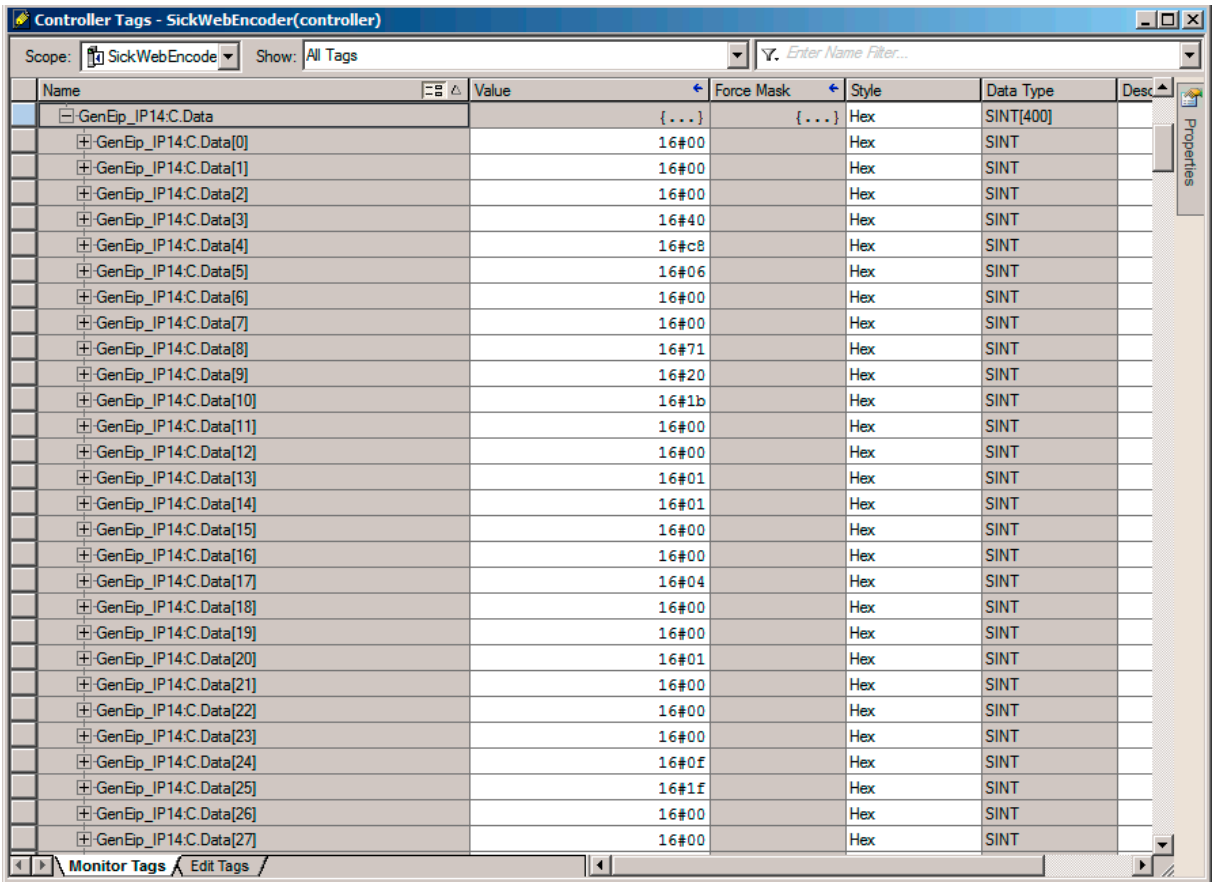
On the following pages the automatically generated module assemblies are displayed:

- Input assembly (104): I.Data [0 ... 13],
- Output assembly (106): O.data [0 ... 2] and
- Configuration assembly (100): C.Data [0 ... 27].

4.1. PLC Controller Input-/Output-assembly tags – generic module

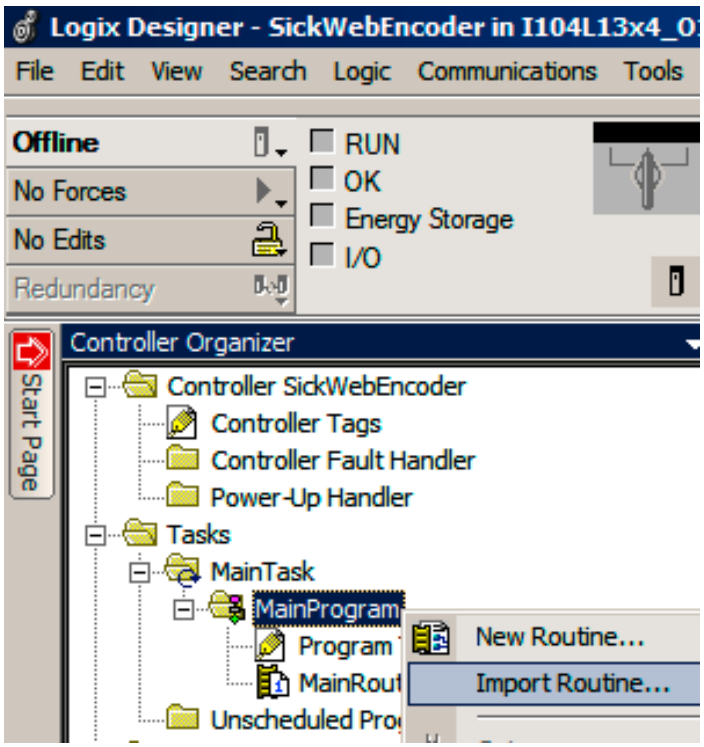


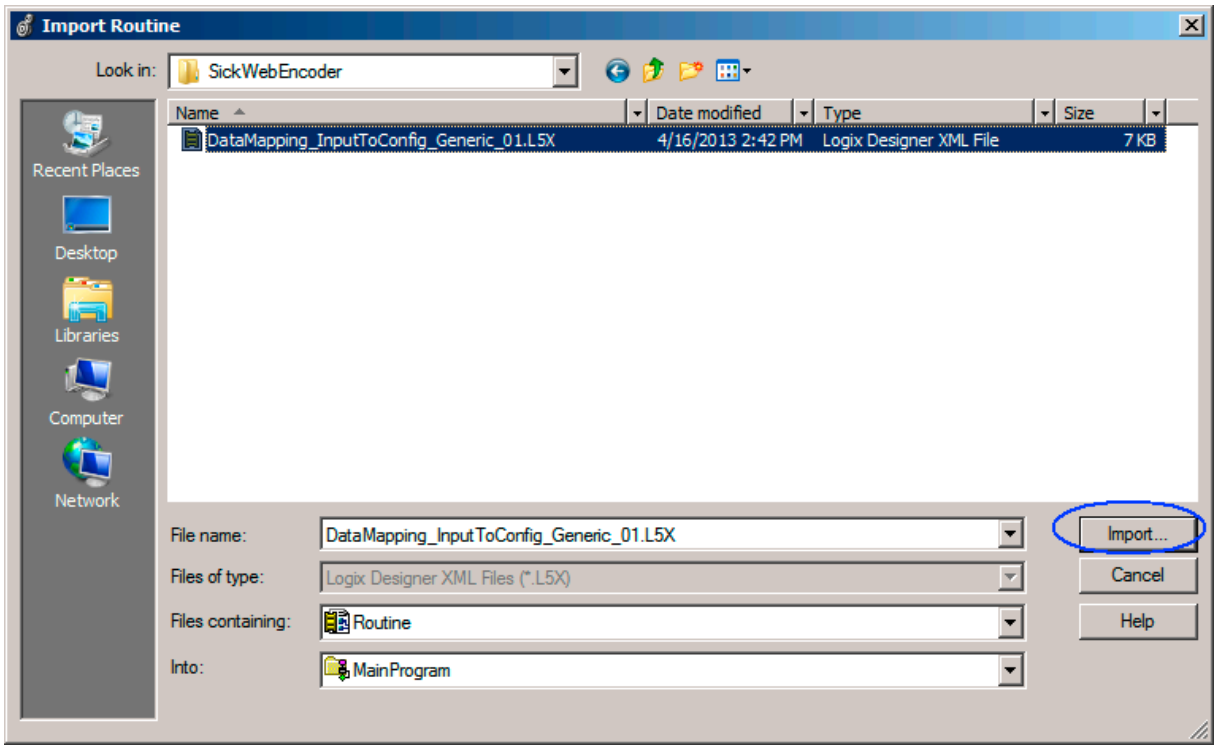
4.2. PLC controller configuration-assembly tags – generic module



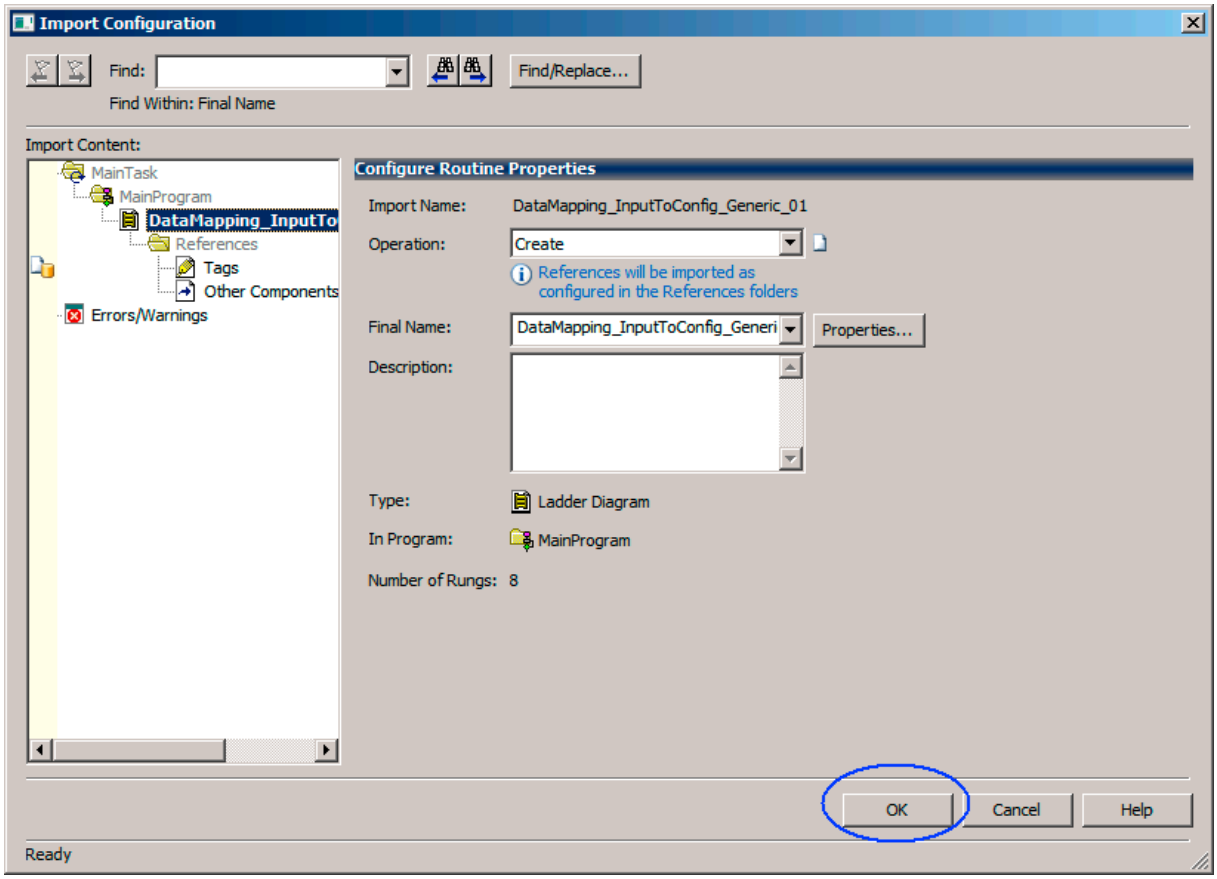
4.3. Import of RS logix Ladder Routine DataMapping_InputToConfig_Generic_01.L5X

Rightclick to the “MainProgram” symbol and select “Import Routine.”

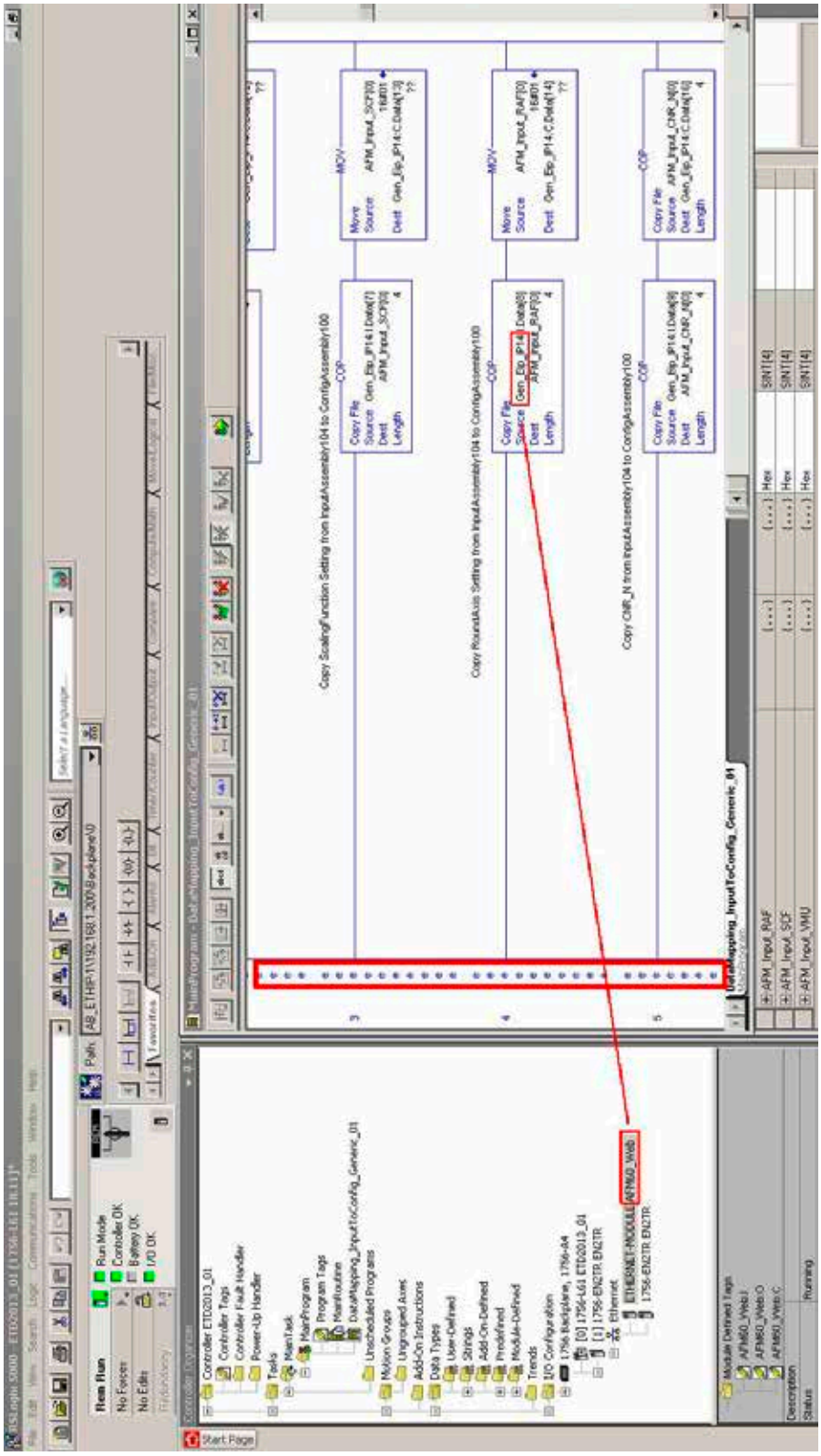




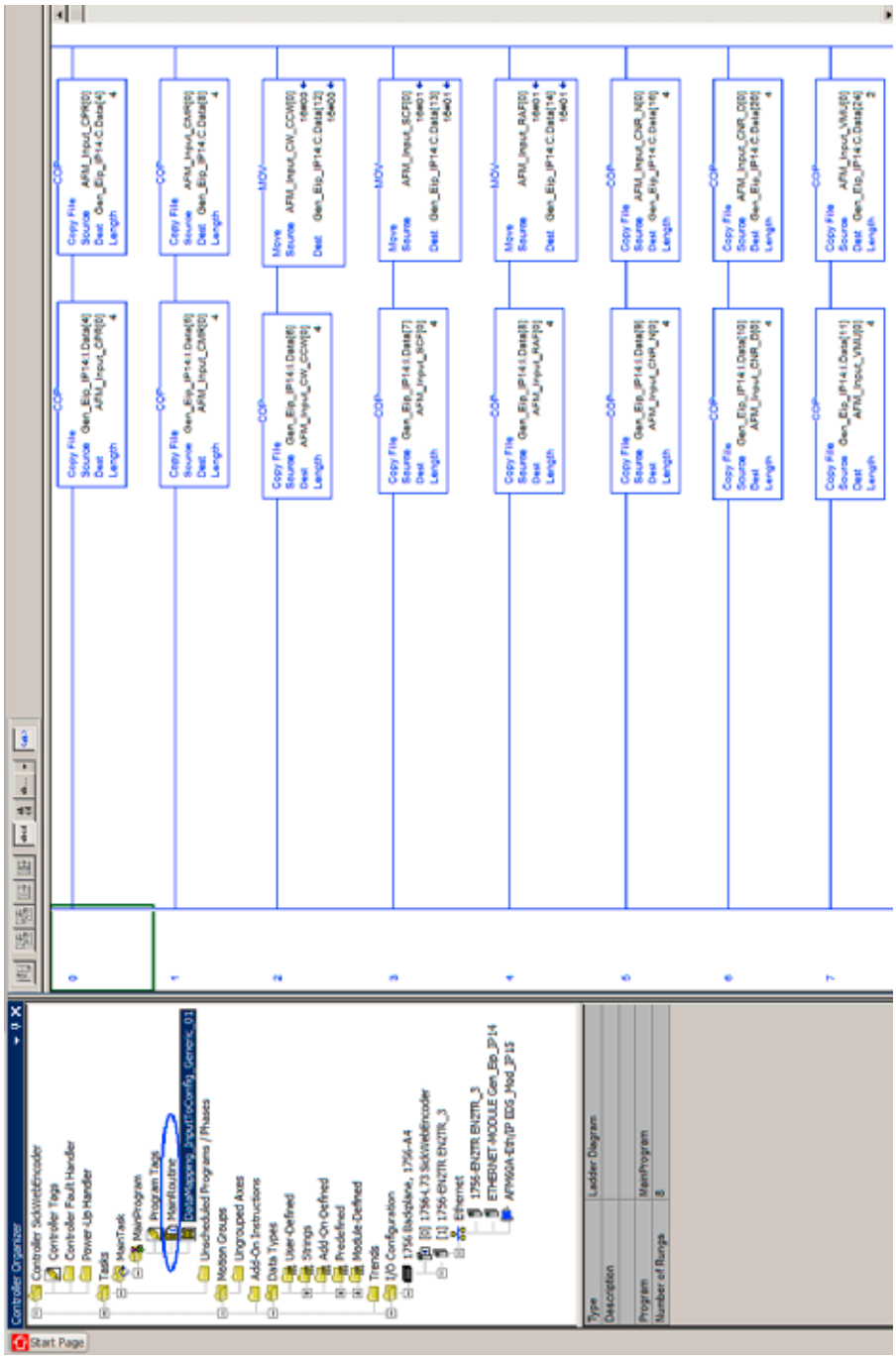
4.4. Import of RS Logix ladder routine / 2



Notes:
Same naming required.



4.5. Import of RS logix Ladder Routine / 3



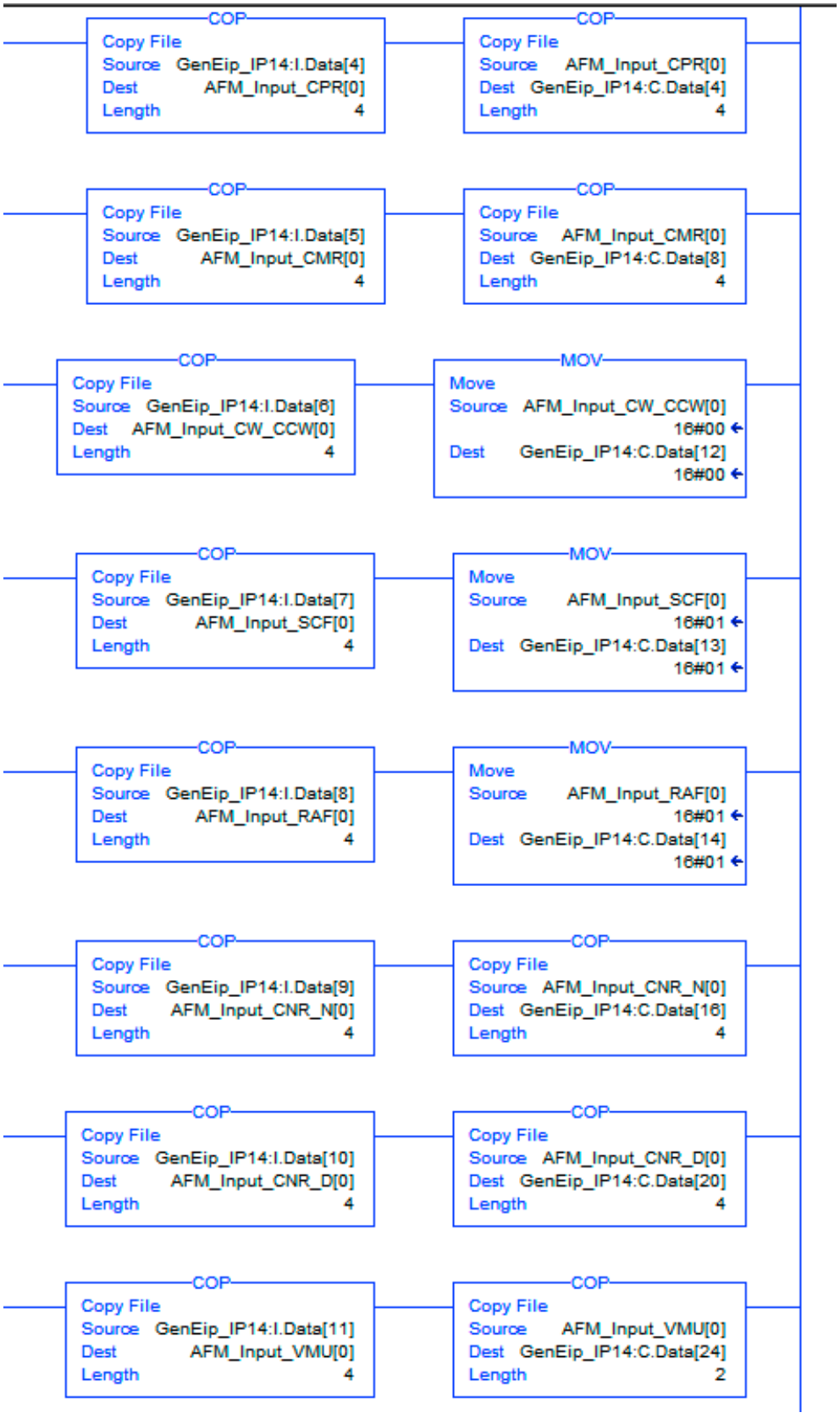
This implementation provides copying of used input data to configuration assembly. The used parameter are listed on the data mapping overview.

Implementation details see on the next page.

4.6. Configuration over PLC – data mapping table

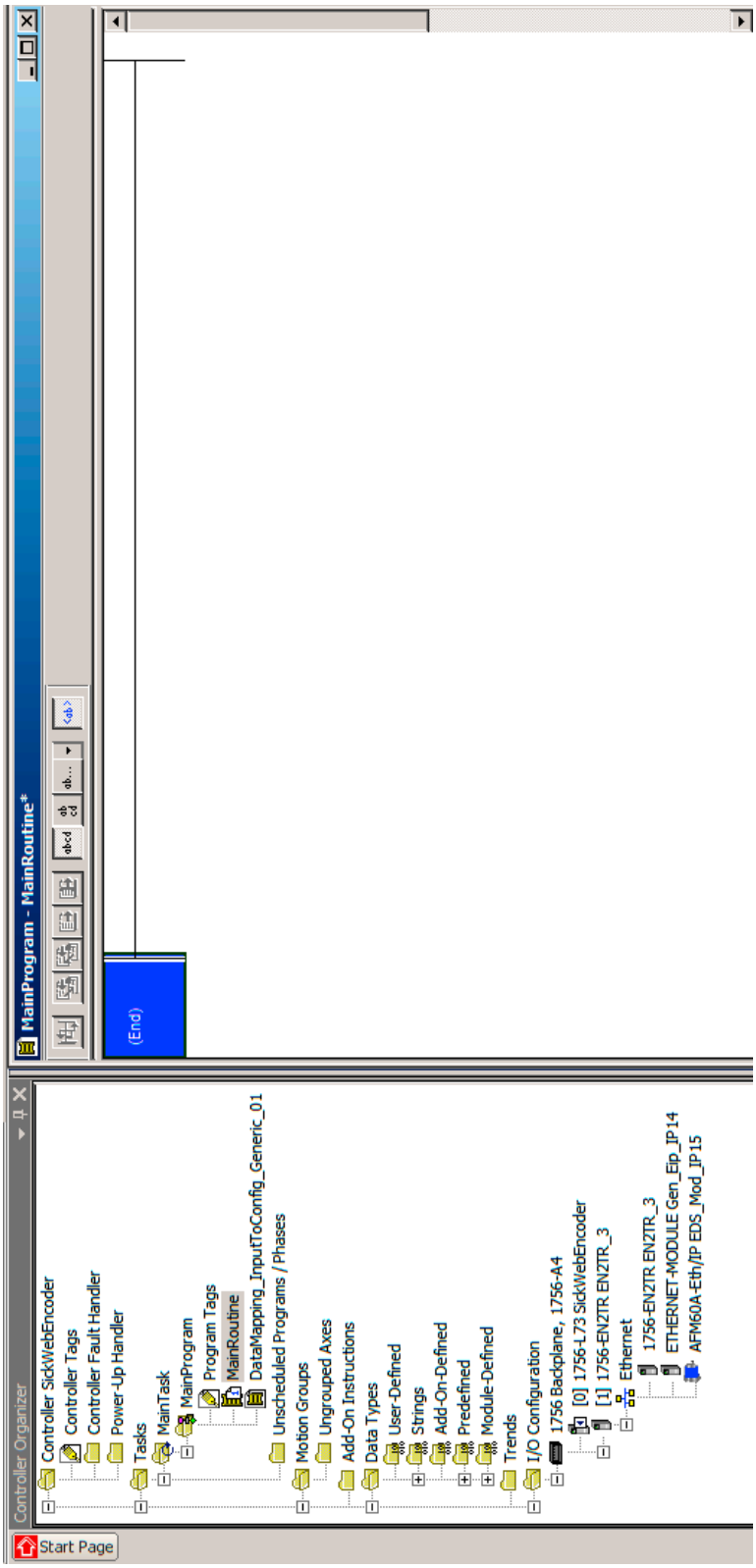
Instance	Input 104								
Element	Byte		Instance	Element	Byte				Data Attributes
0	0								Fault Header (low byte)
	1								Fault Header
	2								Fault Header
	3								Fault Header (high byte)
1	4								Position value (low byte)
	5								Position value
	6								Position value
	7								Position value (high byte)
2	8								Velocity value (low byte)
	9								Velocity value
	10								Velocity value
	11								Velocity value (high byte)
3	12								Serial number value (low byte)
	13								Serial number value
	14								Serial number value
	15								Serial number value (high byte)
4	16	→	100 Config.		4				CPR value (low byte)
	17				5				CPR value
	18				6				CPR value
	19				7				CPR value (high byte)
5	20	→			8				CMR value (low byte)
	21				9				CMR value
	22				10				CMR value
	23				11				CMR value (high byte)
6	24	→	100		12				cw-ccw value (low byte)
	25								cw-ccw value
	26								cw-ccw value
	27								cw-ccw value (high byte)
7	28	→	100		13				Scaling function value (low byte)
	29								Scaling function value
	30								Scaling function value
	31								Scaling function value (high byte)
8	32	→	100		14				Round axis function value (low byte)
	33								Round axis function value
	34								Round axis function value
	35								Round axis function value (high byte)
9	36	→	100		16				CNR_N value (low byte)
	37				17				CNR_N value
	38				18				CNR_N value
	39				19				CNR_N value (high byte)
10	40	→			20				CNR_D value (low byte)
	41				21				CNR_D value
	42				22				CNR_D value
	43				23				CNR_D value (high byte)
11	44	→	100		24				Velocity format value (low byte)
	45				25				Velocity format value
	46								Velocity format value
	47								Velocity format value (high byte)
12	48		106 Output	0	0	←	PLC		Preset Value (low byte)
	49				1				Preset value
	50				2				Preset value
	51				3				Preset value (high byte)
				1	4	←	PLC		Sync Preset Value (low byte)
					5				Sync Preset Value
					6				Sync Preset Value
					7				Sync Preset Value (high byte)

4.7. Data mapping implementation



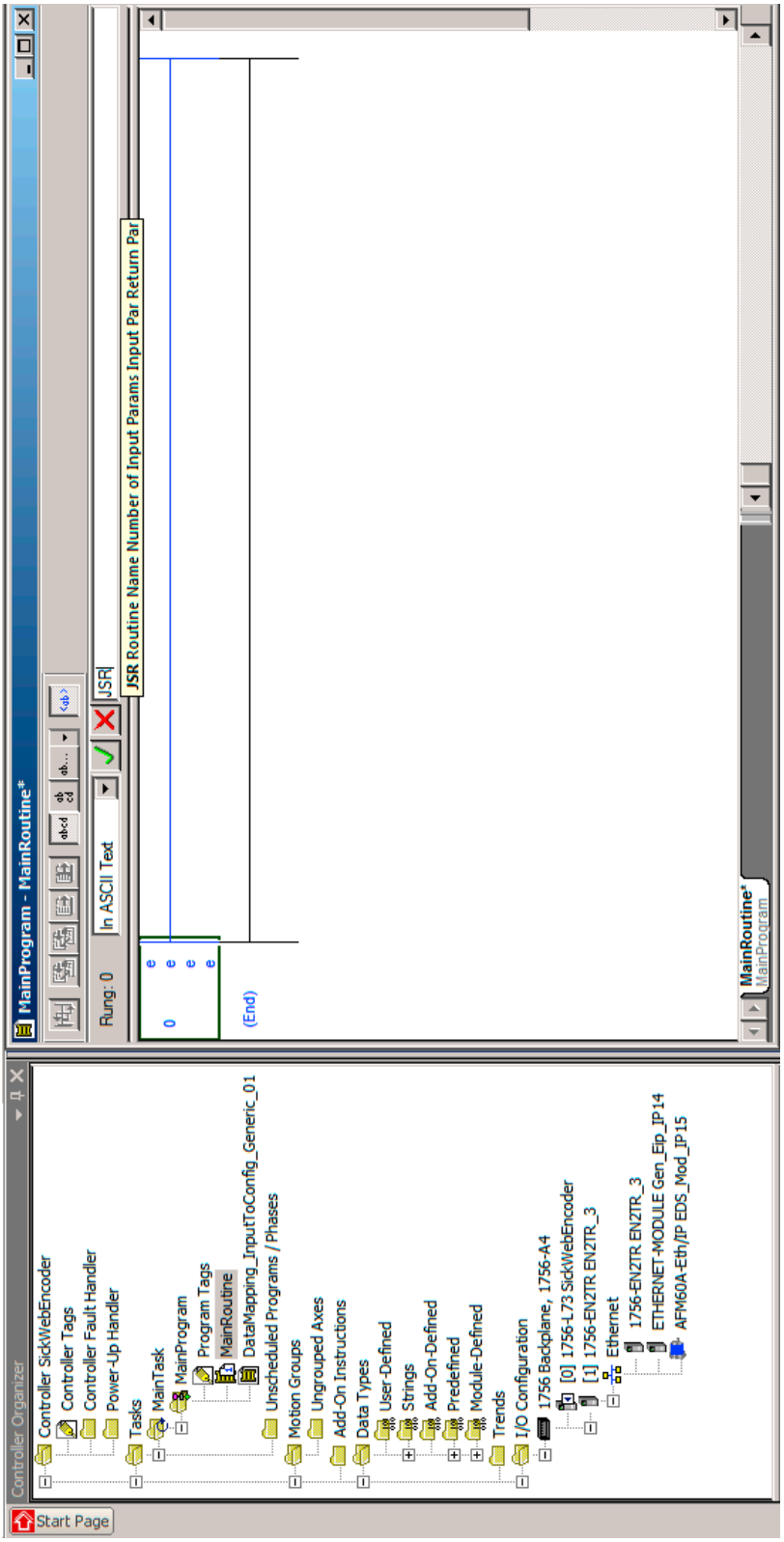
This routine needs to be included in the MainRoutine. Open MainRoutine.

4.8. Configuration over PLC – ladder implementation MainRoutine



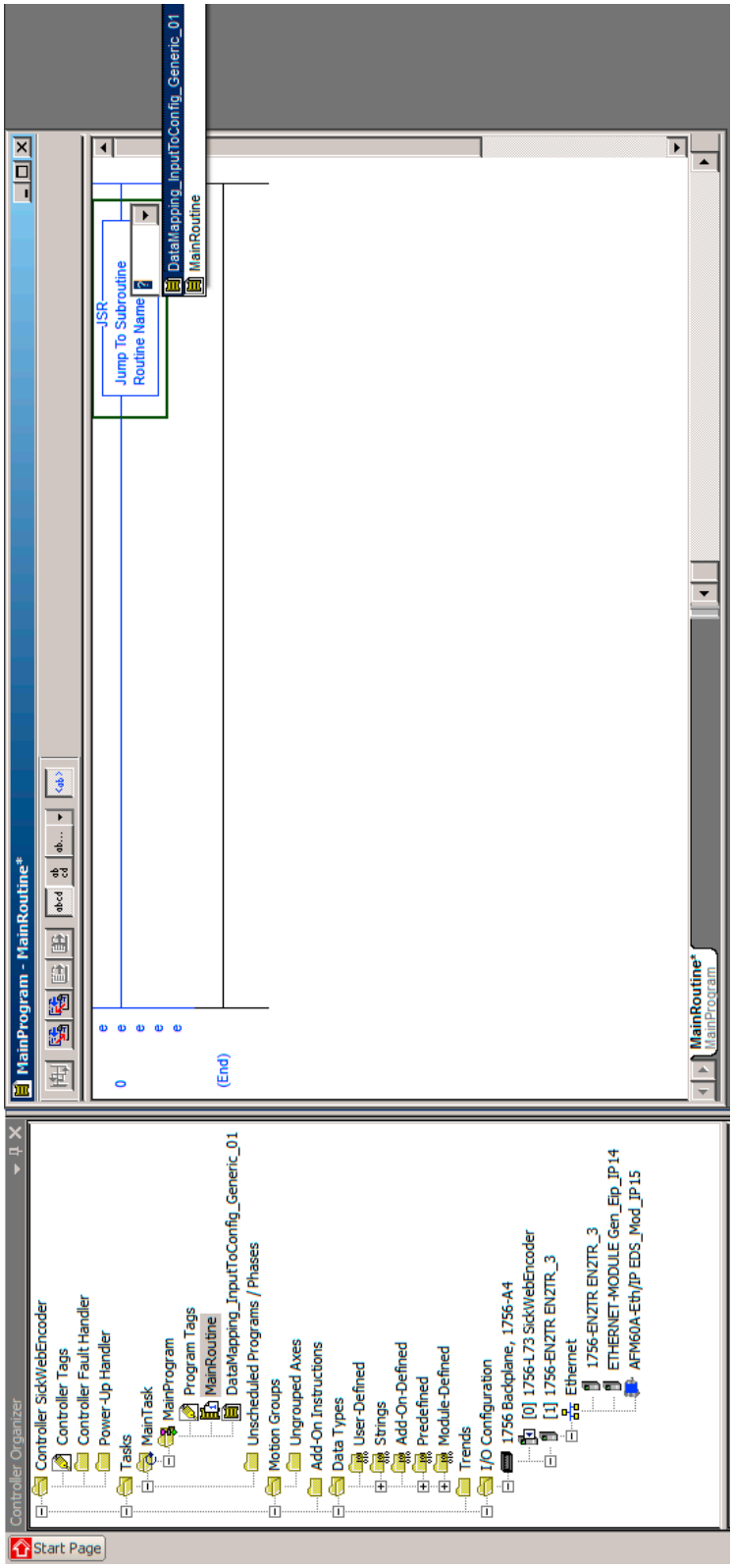
4.9. Configuration over PLC – ladder implementation – JSR command

Implementing command “jump to sub routine”:
Doubleclick on the first rung and insert “JSR” to the opening edit field.
Press enter.

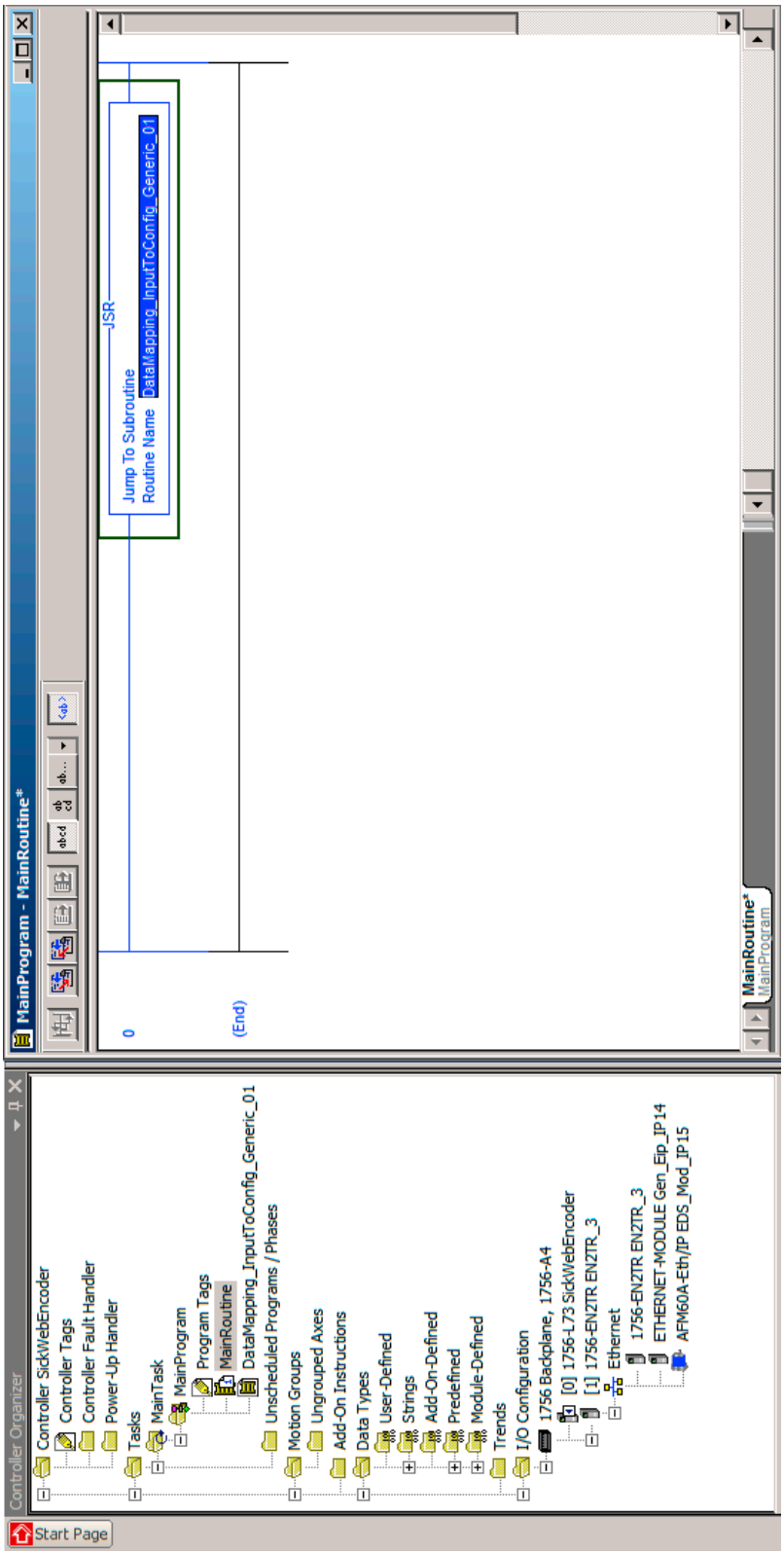


4.10. Configuration over PLC – ladder implementation – select sub routine

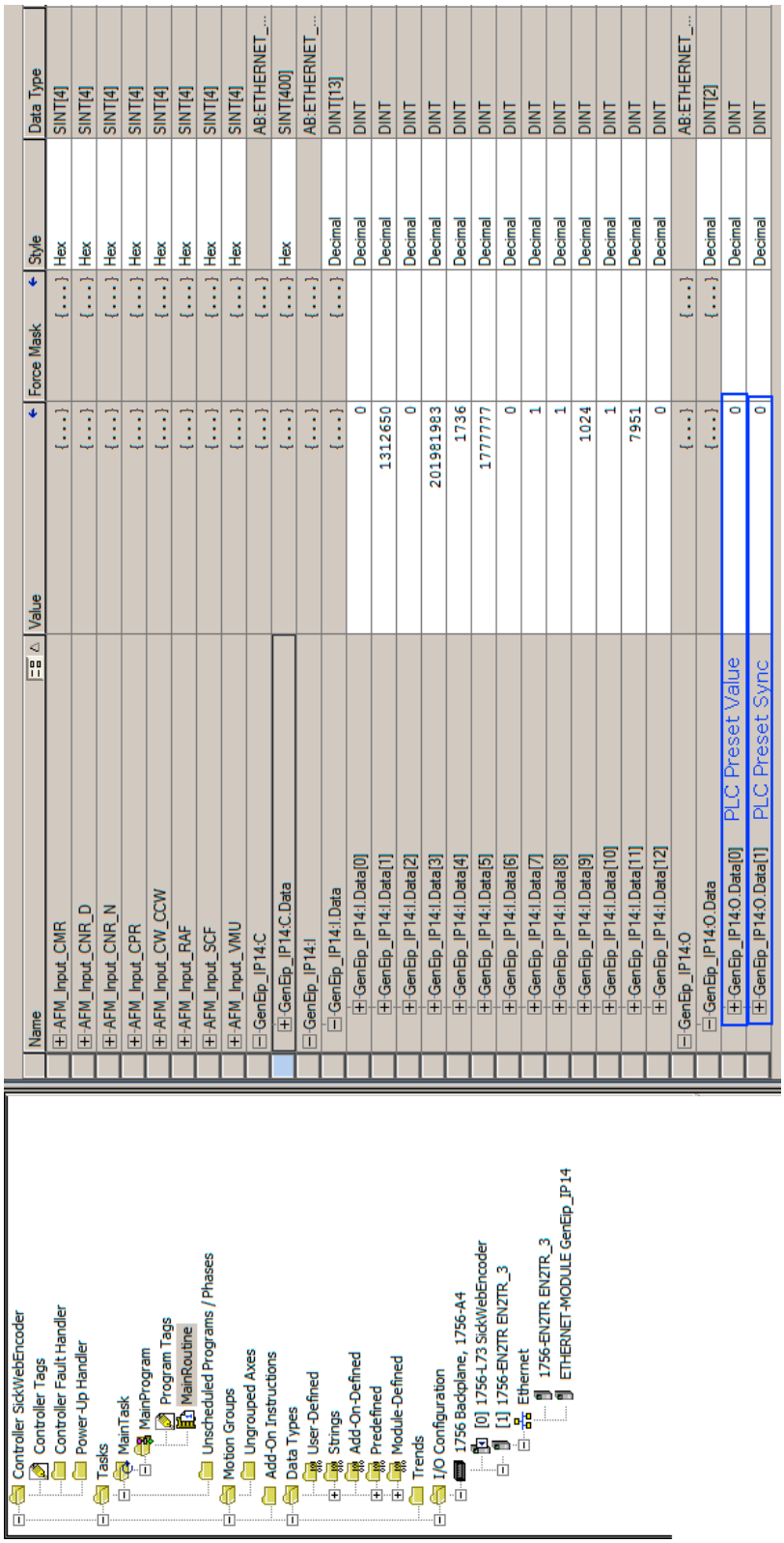
Select sub routine “DataMapping_InputToConfig_Generic_01”.



4.11. Configuration over PLC – ladder implementation complete



4.12. PLC Preset – manual preset over controller tags



5. FTP bootloader information

5.1. FTP update

Please use e. g. the freeware tool “FileZilla” to update the encoder.
If the tool is not installed on your system, enter the search term **FileZilla download** in Google. Install the software.

5.2. Description

A requirement for all further steps is a valid IP address for the encoder, e. g. 192.168.1.14 Launch FileZilla.

- Transfer “FileZilla” to the server manager.

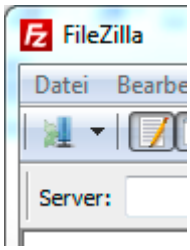
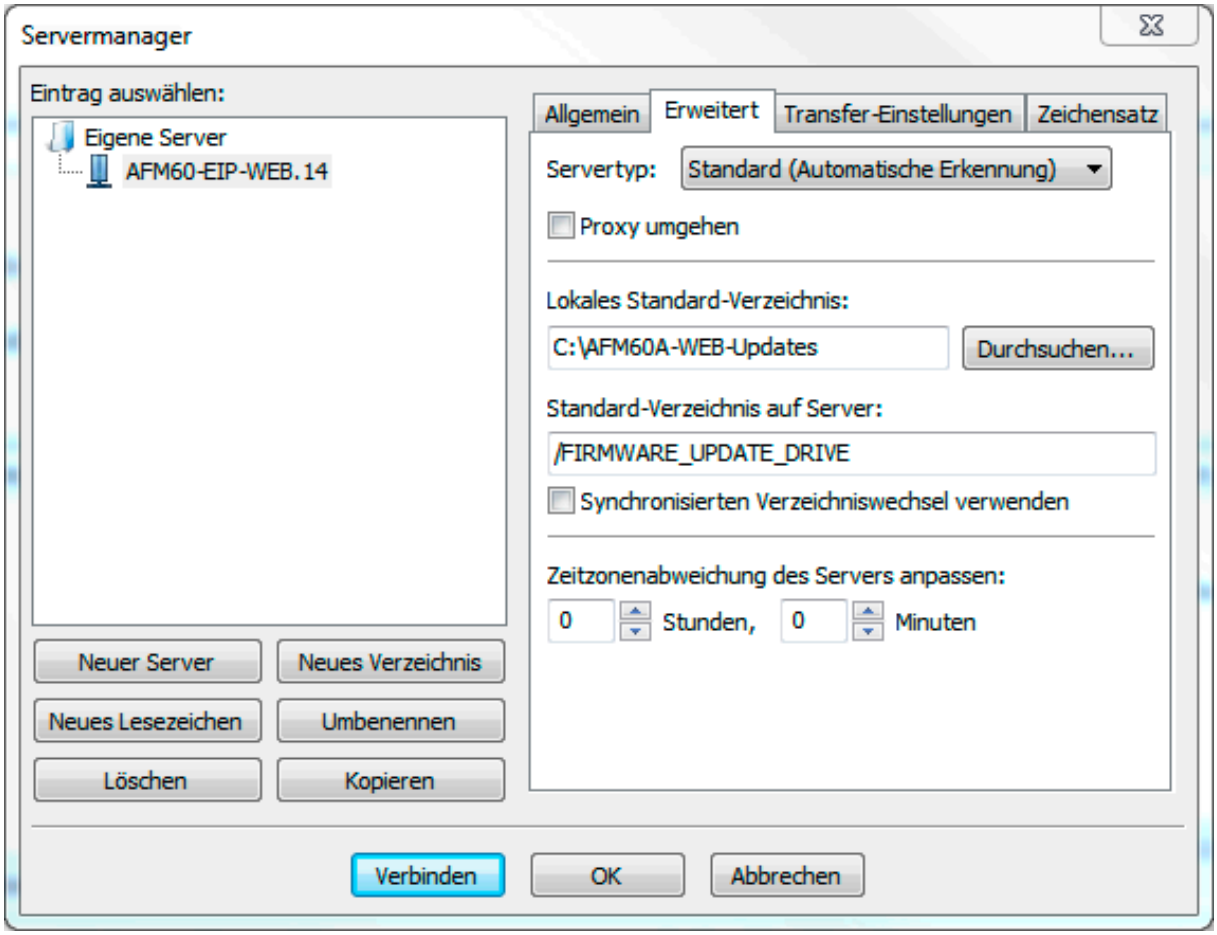


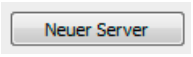
Fig. 1

- Click the button for the server manager.

The server manager dialog opens.

Fig. 2. Server manager – general

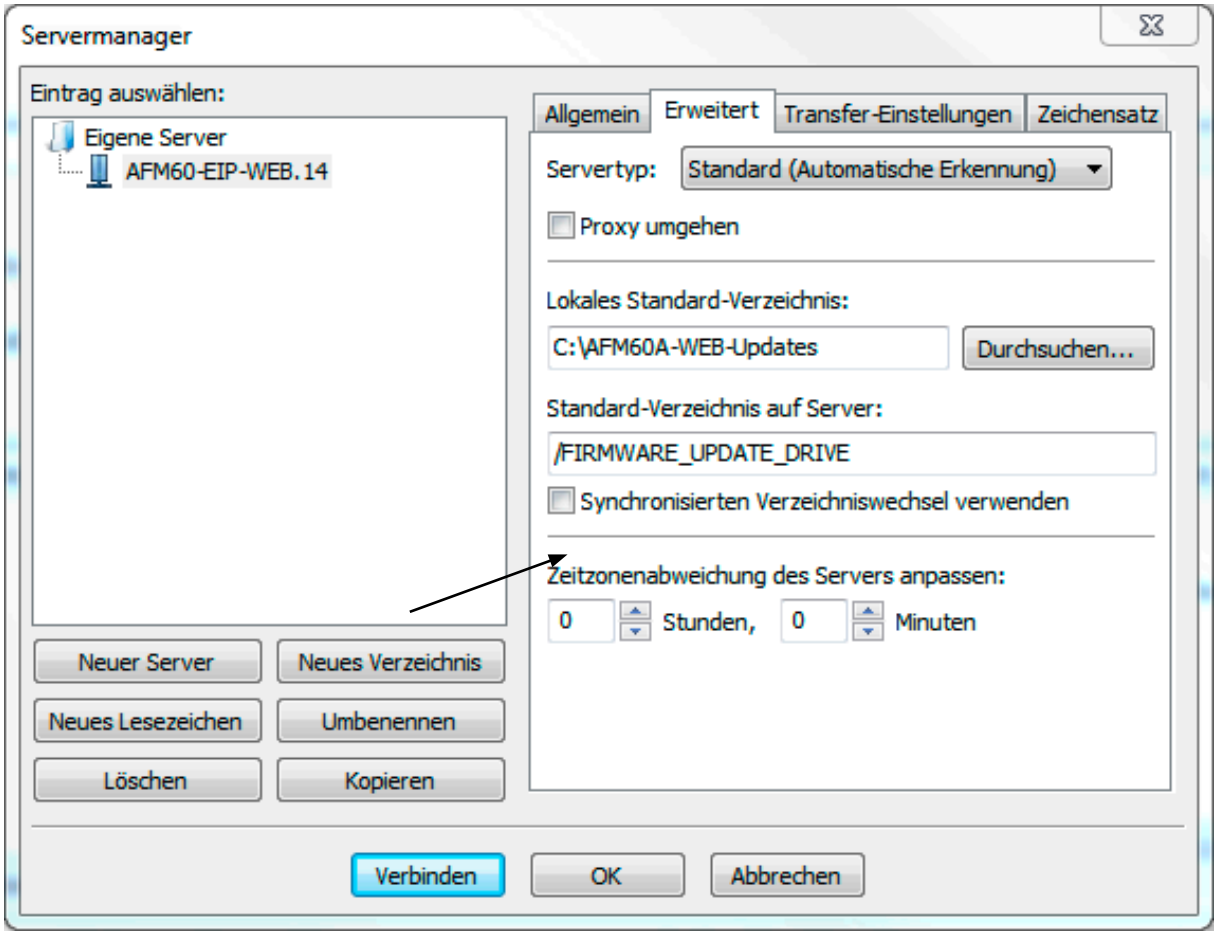


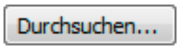
a. Click the  button.

b. Enter a name, e. g. **AFM60-EIP-WEB.14.**
c. Enter the IP address in the “Server” field, e. g. **192.168.1.14.**
d. The “port” field requires no entry. 21 is the default setting.
e. For the “Connection Type”, please select **normal.**
f. Enter **host** in the “User” field.
g. Always enter **enc123** for the “Password”.

Once all these details have been entered, click the **advanced** button.

Fig. 3. Server manager – advanced



h. Under “default local directory” select the required directory by clicking the  button.

i. Under “default directory on server”, enter “FIRMWARE_UPDATE_DRIVE”.

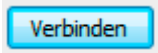
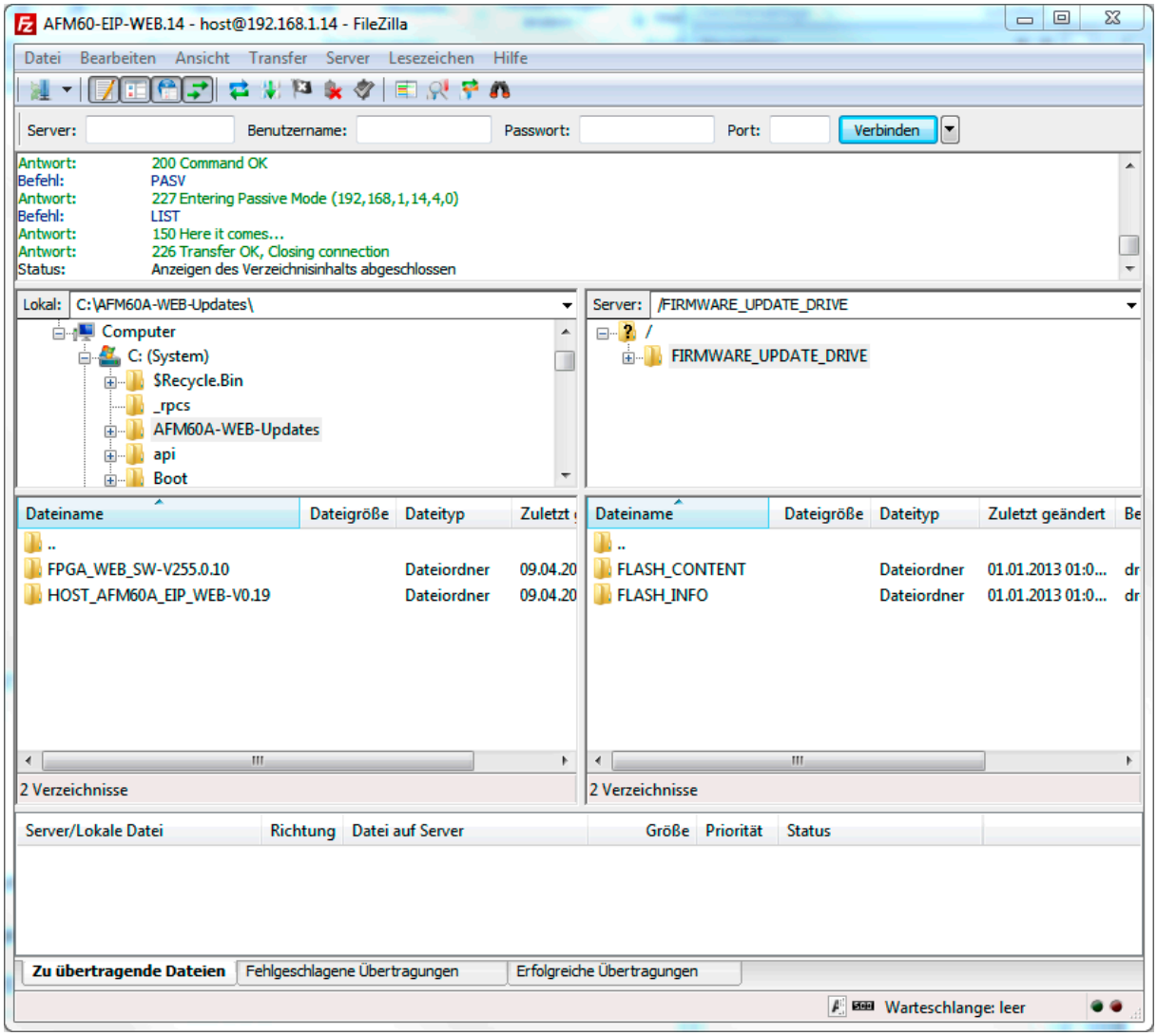
If the encoder is already attached, click the  button to log into the sensor. The following then appears on the monitor (see Fig. 4 on the next page).

Fig. 4. Connect to the sensor with FileZilla

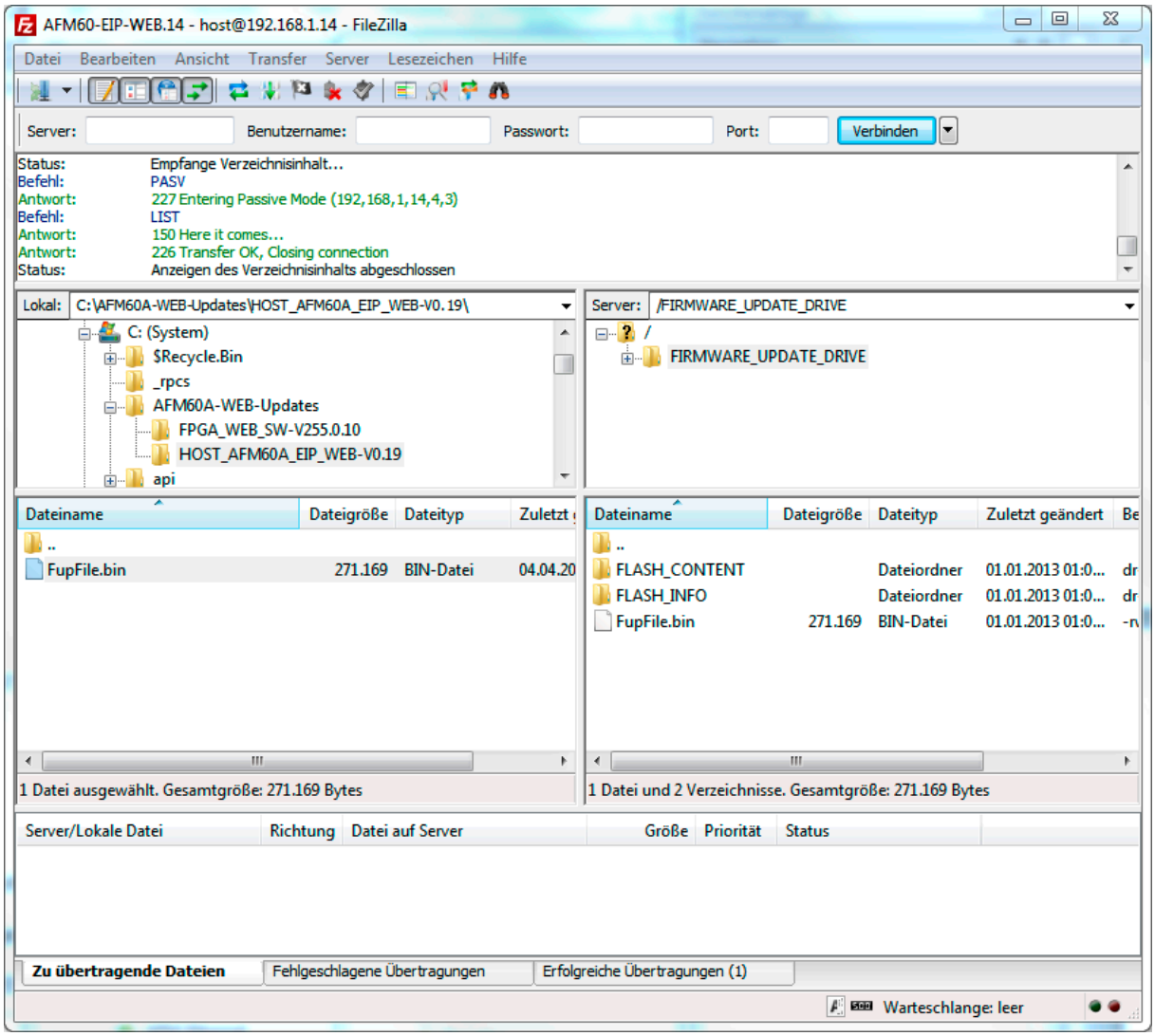


As preconfigured, you are automatically connected to the relevant directories on the PC and encoder.

- To now update the encoder, double-click the folder “HOST_AFM60A_EIP_WEB-V-0.19” (see Fig. 4).
- Then drag the file FupFile.bin to the folder “FIRMWARE_UPDATE_DRIVE” (see Fig. 5, next side).

j. If the firmware update file has been correctly integrated, the file is uploaded to the encoder (see Fig. 5).

Fig. 5. “FupFile.bin” is copied



- While the Firmware is updated, the encoder status LED flashes red, the module and network status LED flash green.
- After the firmware has been successfully updated, the encoder resets and the new application starts.

Warning!
Do not switch the encoder off before the flash process has completed.

The process is identical when updating the WebServer. Just select the **FPGA_WEB_ SW-V255.0.10** directory. The file name is also **FupFile.bin.** The flash process may take longer because the data is approx. 6 times more.

6. Annex

6.1 Conformities and certificates

You can obtain declarations of conformity, certificates, and the current operating instructions for the product at www.sick.com. To do so, enter the product part number in the search field (part number: see the entry in the “P/N” or “Ident. no.” field on the type label).

6.1.1 Compliance with EU directives

EU declaration of conformity (extract)

The undersigned, representing the manufacturer, herewith declares that the product is in conformity with the provisions of the following EU directive(s) (including all applicable amendments), and that the standards and/or technical specifications stated in the EU declaration of conformity have been used as a basis for this.

6.1.2 Compliance with UK statutory instruments

UK declaration of conformity (extract)

The undersigned, representing the following manufacturer herewith declares that this declaration of conformity is issued under the sole responsibility of the manufacturer. The product of this declaration is in conformity with the provisions of the following relevant UK Statutory Instruments (including all applicable amendments), and the respective standards and/or technical specifications have been used as a basis.

8018909/1EH9/2021-12-16 · LF_07 · A4 4c int49

Australia
Phone +61 (3) 9457 0600
1800 33 48 02 – tollfree
E-Mail sales@sick.com.au

Austria
Phone +43 (0) 2236 62288-0
E-Mail office@sick.at

Belgium/Luxembourg
Phone +32 (0) 2 466 55 66
E-Mail info@sick.be

Brazil
Phone +55 11 3215-4900
E-Mail comercial@sick.com.br

Canada
Phone +1 905.771.1444
E-Mail cs.canada@sick.com

Czech Republic
Phone +420 234 719 500
E-Mail sick@sick.cz

Chile
Phone +56 (2) 2274 7430
E-Mail chile@sick.com

China
Phone +86 20 2882 3600
E-Mail info.china@sick.net.cn

Denmark
Phone +45 45 82 64 00
E-Mail sick@sick.dk

Finland
Phone +358-9-25 15 800
E-Mail sick@sick.fi

France
Phone +33 1 64 62 35 00
E-Mail info@sick.fr

Germany
Phone +49 (0) 2 11 53 010
E-Mail info@sick.de

Greece
Phone +30 210 6825100
E-Mail office@sick.com.gr

Hong Kong
Phone +852 2153 6300
E-Mail ghk@sick.com.hk

Hungary
Phone +36 1 371 2680
E-Mail ertekesites@sick.hu

India
Phone +91-22-6119 8900
E-Mail info@sick-india.com

Israel
Phone +972 97110 11
E-Mail info@sick-sensors.com

Italy
Phone +39 02 27 43 41
E-Mail info@sick.it

Japan
Phone +81 3 5309 2112
E-Mail support@sick.jp

Malaysia
Phone +603-8080 7425
E-Mail enquiry.my@sick.com

Mexico
Phone +52 (472) 748 9451
E-Mail mexico@sick.com

Netherlands
Phone +31 (0) 30 229 25 44
E-Mail info@sick.nl

New Zealand
Phone +64 9 415 0459
0800 222 278 – tollfree
E-Mail sales@sick.co.nz

Norway
Phone +47 67 81 50 00
E-Mail sick@sick.no

Poland
Phone +48 22 539 41 00
E-Mail info@sick.pl

Romania
Phone +40 356-17 11 20
E-Mail office@sick.ro

Russia
Phone +7 495 283 09 90
E-Mail info@sick.ru

Singapore
Phone +65 6744 3732
E-Mail sales.gsg@sick.com

Slovakia
Phone +421 482 901 201
E-Mail mail@sick-sk.sk

Slovenia
Phone +386 591 78849
E-Mail office@sick.si

South Africa
Phone +27 10 060 0550
E-Mail info@sickautomation.co.za

South Korea
Phone +82 2 786 6321/4
E-Mail infokorea@sick.com

Spain
Phone +34 93 480 31 00
E-Mail info@sick.es

Sweden
Phone +46 10 110 10 00
E-Mail info@sick.se

Switzerland
Phone +41 41 619 29 39
E-Mail contact@sick.ch

Taiwan
Phone +886-2-2375-6288
E-Mail sales@sick.com.tw

Thailand
Phone +66 2 645 0009
E-Mail marcom.th@sick.com

Turkey
Phone +90 (216) 528 50 00
E-Mail info@sick.com.tr

United Arab Emirates
Phone +971 (0) 4 88 65 878
E-Mail contact@sick.ae

United Kingdom
Phone +44 (0)17278 31121
E-Mail info@sick.co.uk

USA
Phone +1 800.325.7425
E-Mail info@sick.com

Vietnam
Phone +65 6744 3732
E-Mail sales.gsg@sick.com

Detailed addresses and further locations at **www.sick.com**

SICK
Sensor Intelligence.