Network-specific properties of the
Axioline E EtherNet/IP™ IO-Link devices

User manual

UM EN AXL E EIP IOL

PHOENIX CONTACT
INSPIRING INNOVATIONS

User manual

Network-specific properties of the Axioline E EtherNet/IP™ IO-Link devices

Revision 01 2019-03-26

This user manual is valid for Axioline E EtherNet/IP™ IO-Link devices.

AXL E EIP IOL8 DI4 M12 6M, Order No. 2701491
AXL E EIP IOL8 DI4 M12 6P, Order No. 2701496

108007_en_01

PHOENIX CONTACT GmbH & Co. KG • Flachsmarktstraße 8 • 32825 Blomberg • Germany
phoenixcontact.com

Table of contents

1 For your safety

Read this user manual carefully and keep it for future reference.

1.1 Labeling of warning notes

This symbol indicates hazards that could lead to personal injury.
There are three signal words indicating the severity of a potential injury.

DANGER
Indicates a hazard with a high risk level. If this hazardous situation is not avoided, it will result in death or serious injury.

WARNING
Indicates a hazard with a medium risk level. If this hazardous situation is not avoided, it could result in death or serious injury.

CAUTION
Indicates a hazard with a low risk level. If this hazardous situation is not avoided, it could result in minor or moderate injury.

This symbol together with the **NOTE** signal word alerts the reader to a situation which may cause damage or malfunction to the device, hardware/software, or surrounding property.

Here you will find additional information or detailed sources of information.

1.2 Qualification of users

The use of products described in this user manual is oriented exclusively to:

- Qualified electricians or persons instructed by them. The users must be familiar with the relevant safety concepts of automation technology as well as applicable standards and other regulations.
- Qualified application programmers and software engineers. The users must be familiar with the relevant safety concepts of automation technology as well as applicable standards and other regulations.

1.3 Product changes

Changes or modifications to hardware and software of the device are not permitted.

Incorrect operation or modifications to the device can endanger your safety or damage the device. Do not repair the device yourself. If the device is defective, please contact Phoenix Contact.

1.4 Security in the network

NOTE: : Risk of unauthorized network access

Connecting devices to a network via Ethernet always entails the risk of unauthorized access to the network.

Therefore, please check your application for any option of deactivating active communication channels . Setting passwords to prevent third parties from accessing the controller without authorization and modifying the system.

Because of the controller's communication interfaces, we advise against using the controller in safety-critical applications without additional security appliances.
Please take additional protective measures according to the IT security requirements and the standards applicable to your application (for instance virtual networks (VPN) for remote maintenance access, firewalls, etc.) for protection against unauthorized network access.

On first request, you shall release Phoenix Contact and the companies associated with Phoenix Contact GmbH & Co. KG, Flachsmarktstraße 8, 32825 Blomberg in accordance with §§ 15 ff. AktG or German Stock Corporation Act (hereinafter collectively referred to as "Phoenix Contact") from all third-party claims that are made due to improper use.

For the protection of networks for remote maintenance via VPN, Phoenix Contact offers the mGuard product series security appliances which you can find described in the latest Phoenix Contact catalog (phoenixcontact.net/products).

Additional measures for protection from unauthorized network access can be found in the AH EN INDUSTRIAL SECURITY application note. The application note can be downloaded at phoenixcontact.net/products.

2 EtherNet/IP™ object classes, messages, and services

The device supports the Common Industrial Protocol (CIP) according to the ODVA secification. EtherNet/IP™ uses the Common Industrial Protocol as the application layer. IP and TCP or UDP are used for the network and transport layers. CIP and EtherNet/IP™ are standardized by the ODVA on a manufacturer-neutral basis. The Common Industrial Protocol is an object-oriented protocol with two different types of communication between a controller and termination devices.

The following table describes the two communication types.

Connection type	Description
Explicit messaging	This means that a controller or an engineering system sends a request and the termination device responds. For example, explicit messaging can be used for configuration and/or diagnostics.
Implicit messaging	Implicit messaging is used for the cyclic transmission of I/O data. That means, for example, that a termination device sends an analog value which is present at a termination device input. The time for a transmission cycle can be set via the requested packet interval (RPI).

2.1 CIP class services

The device supports the following class services and instance services:

Service Code		Service Name
dec	hex	
01	01	Get_Attribute_All
02	02	Set_Attribute_All
05	05	Reset
09	09	Delete
14	0E	Get_Attribute_Single
16	10	Set_Attribute_Single
75	4B	Read_ISDU
78	4C	Write_ISDU

2.2 CIP object classes

The device supports the following CIP object classes:

Class code		Object type
dec	hex	
01	01	Identity object
02	02	Message router object
04	04	Assembly object
06	06	Connection manager object
09	09	Digital output point object
71	47	Device level ring object
72	48	Quality of service object
128	80	IO-Link device parameter (Vendor specific)
245	F5	TCP/IP interface object
246	F6	Ethernet link object

2.3 Identity object (class code 01_{hex})

The Identity object is required by all devices and provides the device ID and general information about the device.

Class attributes

Attribute	Name	Access	Data type	Value
1	Version	Get	UINT	1
2	Max instance	Get	UINT	1
6	Max class attribute	Get	UINT	7
7	Max instance attribute	Get	UINT	9

2.4 Instance attributes

Attribute	Name	Access	Data type	Value	
1	Vendor ID	Get	UINT	562 ("Phoenix Contact")	
2	Product type	Get	UINT	12 ("Communications adapter")	
3	Product code	Get	UINT	8179	
4	Version	Get	STRUCT of	1	
	- Major revision		- USINT	1	
	- Minor revision		- USINT	1	
5	Status	Get	WORD	Bit 0	Self-Testing or unknown
				Bit 1	Reserved
				Bit 2	At least one fault I/O connection
				Bit 3	No I/O connections established
				Bit 4	Reserved
				Bit 5	Major Fault
				Bit 6	At least one I/O connection in run mode.
				Bit 7	At least one I/O connection established, all in idle mode.
				Bit 8	Reserved
				Bit 9	Reserved
				Bit 10 ... bit 15	Vendor specific
6	Serial number	Get	UDINT	Is fixed in production process	
7	Product name	Get	STRING	AXL E EIP IOL8 DI4 M12 6M/AXL E EIP IOL8 DI4 M12 6P	
8	State	Get	USINT	0 = Nonexistent	
				1 = Device self testing	
				2 = Standby	
				3 = Operational	
				4 = Major recoverable fault	
				5 = Major unrecoverable fault	
				6 ... 254 = Reserved	
				255 = Default for Get_ Attribute_All service	
9	Configuration consistency value	Get	USINT	CRC checksum	

Common services

Service code		Class	Instance	Service name
dec	hex			
01	01	Yes	Yes	Get_Attribute_All
05	05	No	Yes	Reset
14	0E	Yes	Yes	Get_Attribute_Single
16	10	No	Yes	Set_Attribute_Single

2.5 Message router object (class code 02_{hex})

The Message router object provides a messaging connection point through which a client may address a service to any object class or instance residing in the physical device.

The device does not support any object attributes.

2.6 Assembly object (class code 4_{hex})

The Assembly object combines attributes of multiple objects to allow data to or from each object to be sent or received via a single connection.

Class attributes

Attribute	Name	Access	Data type	Value
1	Version	Get	UINT	2
2	Max instance	Get	UINT	107

Instance attributes

Attribute	Name	Access	Data type	Value
3	Data	Get, Set	ARRAY of Byte	Current Process data of the correspondent Assembly instance
4	Size	Get	UINT	Number of bytes in attribute 3

Common services

Service code		Class	Instance	Service name
dec	hex			
14	0E	Yes	Yes	Get_Attribute_Single
16	10	No	Yes	Set_Attribute_Single

2.7 Connection manager object (class code 06_{hex})

The Connection manager object allocates and manages the internal resources that are used for I/O and Explicit messaging connections.

Class attributes

Attribute	Name	Access	Data type	Value
1	Version	Get	UINT	1
2	Max instance	Get	UINT	1

Class attributes

The device does not support any instance attributes.

Common services

Service code		Class	Instance	Service name
dec	hex			
14	0E	Yes	Yes	Get_Attribute_Single

2.8 Device level ring object (class code 47_{hex})

The Device level ring object (DLR) is the interface for configuration and status information for the DLR protocol.

Class attributes

Attribute	Name	Access	Data type	Value
1	Version	Get	UINT	3

Instance attributes

Attribute	Name	Access	Data type	Value/description	
1	Network topology	Get	USINT	0 = Linear	
				1 = Ring	
2	Network status	Get	USINT	0 = Normal	
				1 = Ring fault	
				2 = Unexpected loop detected	
				3 = Partial network fault	
				4 = Rapid fault/Restore cycle	
10	Active supervisor address	Get	STRUCT of:	IP and/or MAC address of the active ring supervisor	
			UDINT	Supervisor IP address	
			Array of 6 USINTs	Supervisor MAC address	
12	Capability flags	Get	DWORD	Bit 0	Announced-based ring node
				Bit 1	Beacon-based ring note
				Bit 2 ... bit 4	Reserved
				Bit 5	Supervisor capable
				Bit 6 ... bit 31	Reserved

Common services

Service code		Class	Instance	Service name
dec	hex			
1	01	Yes	Yes	Get_Attribute_All
14	0E	Yes	Yes	Get_Attribute_Single

2.9 Quality of service object (class code 48_{hex})

Quality of service (QoS) affects the forwarding and handling of data streams and results in individual data streams being given differential treatment (usually preferential). QoS can be used, e.g., to guarantee a transmission bandwidth for individual data streams. The device uses QoS in connection with prioritization.

Class attributes

Attribute	Name	Access	Data type	Value
1	Version	Get	UINT	1
2	Max instance	Get	UINT	1

Instance attributes

Attribute	Name	Access	Data type	Value/description
1	802.1Q tag enable	Get, Set	UINT	0 = off; 1 = on
4	DSCP urgent	Get, Set	USINT	DSCP value for CIP transport class 0/1 Urgent priority message (default 55)
5	DSCP scheduled	Get, Set	USINT	DSCP value for CIP transport class 0/1 Scheduled priority message (default 47)
6	DSCP high	Get, Set	USINT	DSCP value for CIP transport class 0/1 High priority message (default 43)
7	DSCP low	Get, Set	USINT	DSCP value for CIP transport class 0/1 Low priority message (default 31)
8	DSCP explicit	Get, Set	USINT	DSCP value for CIP explicit messages (transport class 2/3 and UCMM) and all other EtherNet/IP™ encapsulation messages (default 27)

Common services

Service code		Class	Instance	Service name
dec	hex			
14	0E	Yes	Yes	Get_Attribute_Single
16	10	No	Yes	Set_Attribute_Single

2.10 IO-Link parameter object (class code 80_{hex})

The IO-Link parameter object is a vendor-specific implementation to access IO-Link device parameters. The object has the class instance 0 and the instance 1. The class instance 1 addresses the IO-Link master.

Class attributes

Attribute	Name	Access	Data type	Value
1	Version	Get	UINT	1
2	Max instance	Get	UINT	1
3	Max class attribute	Get	UINT	8 (Number of the IO-Link ports)

Instance attributes

The IO-Link ports are addressed via the corresponding instance attributes number, see section "Read_ ISDU" or "Write_ISDU".

Common services

Service code		Class	Instance	Service name
dec	hex			
14	0E	Yes	No	Get_Attribute_Single
75	4B	No	Yes	Read_ISDU
78	4C	No	Yes	Write_ISDU

Read_ISDU

This service is used to read IO-Link device parameters.

Request

	MSG Config	Description	
Class	80_{hex}	IO-Link parameter object	
Instance	1	Addressing of the IO-Link masters	
Instance attributes	1 ... 8	IO-Link port number	
Service code	$4B_{hex}$	Read_ISDU	
Data	Read ISDU Service request parameter		
	Name	Data type	Description
	Index	UINT	IO-Link ISDU object index
	Sub-Index	USINT	IO-Link ISDU object Sub-Index

EtherNet/IP™ uses the little endian format (Intel), IO-Link the big endian format (Motorola).
For the IO-Link process data, no byte swapping is performed.

Response

The response to the request can either be positive or negative.

- Positive Response

If the request service was successful ("General Status" of the CIP service response is = 0), the response has the following structure:

Name	Data type	Description
ISDU Data	Array of Byte	Maximal 232 bytes

- Negative Response

If the request service was not successful ("General Status" of the CIP Service Response unequal 0), the response has the following structure:

Name	Data type	Value/Description	
IO-Link master Error	UINT	1	Service not available
		2	Port blocked
		3	Timeout
		4	Invalid index
		5	Invalid sub index
		6	Wrong port
		7	Wrong port function
		8	Invalid length
		9	ISDU not supported
IO-Link Device Error	USINT	Please refer to the specific IO-Link device data sheet	
IO-Link Device Additional Error	USINT	Please refer to the specific IO-Link device data sheet	

Example: Reading an IO-Link device object

The following table shows the examplary structure of a message (MSG) for reading the VendorName of an IO-Link device, which is connected to port 7.

Request

	MSG Config	Description
Class	80_{hex}	IO-Link parameter object
Instance	1	Addressing of the IO-Link master
Instance attributes	7	IO-Link port 7
Service code	$4B_{hex}$	Read ISDU
Data	0010_{hex}	IO-Link ISDU object index: 0010_{hex} (VendorName)
	00_{hex}	IO-Link ISDU objekt Sub-Index: 00_{hex} (no Sub-Index)

Response

Depending on wheter the response is positive or negative, the data structure contains either the VendorName or additional error codes.

Write_ISDU

This service is used to write IO-Link device parameters.

Request

	MSG Config	Description	
Class	80_{hex}	IO-Link parameter object	
Instance	1	Addressing of the IO-Link masters	
Instance attributes	1 ... 8	IO-Link port number	
Service codes	$4C_{hex}$	Write ISDU	
Data	Write ISDU Service request parameter		
	Name	Data type	Description
	Index	UINT	IO-Link ISDU object index
	Sub-Index	USINT	IO-Link ISDU object Sub-Index
	Data	Array of Byte	IO-Link ISDU data (max. 232 bytes)

EtherNet/IP™ uses the little endian format (Intel), IO-Link the big endian format (Motorola).
For the IO-Link process data, no byte swapping is performed.

Response

The response to the request can either be positive or negative.

- Positive Response

If the request service was successful ("General Status" of the CIP service response is = 0), the response does not contain any further data.

- Negative Response

If the request service was not successful ("General Status" of the CIP service response unequal 0), the response has the following structure:

Name	Data type	Value/Description	
IO-Link Master Error	UINT	1	Service not available
		2	Port blocked
		3	Timeout
		4	Invalid index
		5	Invalid sub index
		6	Wrong port
		7	Wrong port function
		8	Invalid length
		9	ISDU not supported
IO-Link Device Error	USINT	Please refer to the specific IO-Link device data sheet	
IO-Link Device Additional Error	USINT	Please refer to the specific IO-Link device data sheet	

Example: Writing to an IO-Link device object

The following table shows the exemplary structure of a message (MSG) for writing the **Application Specific Name Object (18_{hex})** of the Phoenix Contact Device PSK AFS5000IOL (Order No 2700705), which is connected to port 5.

Request

	MSG Config	Description
Class	80_{hex}	IO-Link parameter object
Instance	1	Addressing of the IO-Link master
Instance attributes	5	IO-Link port 7
Service code	$4C_{hex}$	Write ISDU
Data	0018_{hex}	IO-Link ISDU objekt index: 0018_{hex} (Application specific name)
	00_{hex}	IO-Link ISDU object Sub-Index: 00_{hex} (no Sub-Index)
	[0] 4F [1] 75 ... [13] 5F [14] 31 [15] 00	IO-Link ISDU data: The object has 16 bytesof data. - ASCII: Outside_Meter_1 (= 15 byte) - Hex: 4f 75 74 73 69 64 65 5f 4d 65 74 65 72 5f 31 - The last byte is filled with zeros.

Response

If the request service was successful, the general status of the CIP service response is = 0. In the event of an error the general status is unequal 0.

2.11 TCP/IP object (Class code $F5_{hex}$)

The TCP/IP interface object makes it is possible to configure the physical network interface. These include setting for example the IP address, subnet mask and gateway address.

Class attributes

Attribute	Name	Access	Data type	Value
1	Revision	Get	UINT	3
2	Max instance	Get	UINT	1

Instance attributes.

Attribute	Name	Access	Data type	Description	
1	Status	Get	DWORD	Bit 0 ... bit 3	Interface configuration status
				Bit 4	Mcast pending (always 0)
				Bit 5	Interface configuration pending
				Bit 6	AcdStatus
				Bit 7	AcdFault
				Bit 8 ... bit 31	Reserved
2	Configuration capability	Get	DWORD	Bit 0	BOOTP client
				Bit 1	Not supported
				Bit 2	DHCP client
				Bit 3	Not supported
				Bit 4	TCP/IP config setable via ETH/IP
				Bit 5	Not supported
				Bit 6	Not supported
				Bit 7	AcdCapable
				Bit 8 ... bit 31	Reserved
3	Configuration control	Get, Set	DWORD	Bit 0 ... bit 3	0 = The device use static IP configuration.
					1 = The device use BOOTP.
					2 = The device use DHCP.
				Bit 4	Not supported
				Bit 5 ... bit 31	Reserved
4	Physical link object	Get	STRUCT of:	Path to Physical link object	
	- Path size		- UINT	4	
	- Path		- Padded EPATH	20_{hex}, $F6_{hex}$, 25_{hex}, 01_{hex}	

Attribute	Name	Access	Data type	Description
5	Interface configuration	Get, Set	STRUCT of:	TCP/IP network interface configuration
	- IP address		- UDINT	The device's IP address
	- Network mask		- UDINT	The device's network mask
	- Gateway address		- UDINT	Default gateway address
	- Name server		- UDINT	Primary name server (always 0.0.0.0)
	- Name server 2		- UDINT	Secondary name server (always 0.0.0.0)
	- Domain name		- STRING	Default domain name (always empty)
6	Host name	Get, Set	STRING	The device's host name
10	SelectAcd	Get, Set	BOOL	0 = disable; 1 = enable (default)
11	Last conflict detected	Set	STRUCT of:	Structure containing information related to the last conflict detected
	- AcdActivity		- USINT	State of ACD activity when last conflict detected
				0 = Noconflictdetected (default)
				1 = ProbeIpv4Address
				2 = OngoingDetection
				3 = SemiActiveProbe
	- Remote MAC		- ARRAY of 6 USINTs	MAC address of remote node from the ARP PDU in which a conflict was detected
	- ArpPdu		- Array of 28 USINTs	Copy of the raw ARP PDU in which a conflict was detected.
12	Ethernet/IP quick connect	Get, Set	BOOL	0 = disable (default); 1 = enable

Common services

Service code		Class	Instance	Service name
dec	hex			
1	01	Yes	Yes	Get_Attribute_All
14	0E	Yes	Yes	Get_Attribute_Single
16	10	No	Yes	Set_Attribute_Single

0.1 Ethernet link object (class code $F6_{hex}$)

The Ethernet link object contains specific status information of the Ethernet (IEEE 802.3).

Class attributes

Attribute	Name	Access	Data type	Value
1	Version	Get	UINT	3
2	Max instance	Get	UINT	2
3	Number of instances	Get	UINT	2

Instance attributes

Attribute	Name	Access	Data type	Description	
1	Interface speed	Get	UDINT	Interface speed currently in use. Speed in bps (e.g. 10, 100)	
2	Interface flags	Get	DWORD	Bit 0	Link status
				Bit 1	Half/Full duplex status
					0 = Half duplex
					1 = Full duplex
				Bit 2 ... bit 4	Autonegotiation status
					0 = Autonegotiation in progress
					1 = Autonegotiation and speed detection failed
					2 = Autonegotiation failed but detected speed
					3 = Successfully negotiated
					4 = Autonegotiation not attempted
				Bit 5	Not supported
				Bit 6	Not supported
				Bit 7 ... bit 31	Reserved
3	Physical address	Get	ARRAY of 6 USINTs	MAC layer adress	

Attribute	Name	Access	Data type	Description	
6	Interface control	Get, Set	STRUC of:	Configuration for physical interface	
	- Control bits		- WORD	Interface control bits	
				Bit 0	Auto negotiate (set)
					0 = Autonegotiation disabled
					1 = Autonegotiation enabled
				Bit 1	Forced duplex mode (set)
					0 = Half duplex
					1 = Full duplex
					Set only if Autonegotiation is disabled
				Bit 2 ... bit 15	Reserved, shall be set to 0
	- Forced interface speed		- UINT	The Forced interface speed bits indicate the speed at which the interface shall operate.	
				10 = 10 Mbps	
				100 = 100 Mbps	
				Set only if Autonegotiation is disabled.	
7	Interface type	Get		Indicates the type of the physical interface.	
				0 = Unknown interface type	
				1 = The interface is internal to the device	
				2 = Twisted-pair	
				3 = Optical fiber	
				4 … 255 = Reserved	
8	Interface state	Get		Indicate the current operational state of the interface.	
				0 = Unknown interface state	
				1 = The interface is enabled and is ready to send and receive data.	
				2 = The interface is disabled.	
				3 = The interface is testing.	
				4 … 255 = Reserved	
9	Interface lable	Get, Set		Allows administrative setting of the interface state	
				0 = Reserved	
				1 = Enable the interface	
				2 = Disable the interface	
				3 … 255 = Reserved	
10	Interface lable	Get	STRING	Interface name	

Common services

Service code		Class	Instance	Service name
dez	hex			
14	0E	Yes	Yes	Get_Attribute_Single
16	10	No	Yes	Set_Attribute_Single

3 EtherNet/IP™ Assembly instances

The AXL E EIP IO-Link master has several Assembly instances for input and output data, which differ mainly in the data length.

The configuration assembly is always the same.

	Input Assembly	Data length		Output Assembly	Data length		Config Assembly	Data length
Exclusive Owner								
Without IO-Link	101	4		100	4		106	214
With IO-Link input	103	260		100	4		106	214
With IO-Link output	101	4		102	260		106	214
With IO-Link I/O	103	260		102	260		106	214
With IO-Link I/O + status	105	316		102	260		106	214
With IO-Link I/O + status + events	107	388		102	260		106	214
Input Only								
Without IO-Link	101	4		193	0		106	214
With IO-Link	103	260		193	0		106	214
With IO-Link input + status	105	316		193	0		106	214
With IO-Link input + status + events	107	388		193	0		106	214
Listen Only								
Without IO-Link	101	4		192	0		-	-
With IO-Link	103	260		192	0		-	-
With IO-Link input + status	105	316		192	0		-	-
With IO-Link input + status + events	107	388		192	0		-	-

3.1 Input process data

Assembly instance 101

Byte	Function	Description
0 ... 3	Status bytes	IO-Link master status bytes

Assembly instance 103

Byte	Function	Description
0 ... 3	Status bytes	IO-Link master status bytes
4 ... 35	IO-Link port 1 IN	32 byte IO-Link input process data
36 ... 67	IO-Link port 2 IN	32 byte IO-Link input process data
68 ... 99	IO-Link port 3 IN	32 byte IO-Link input process data
100 ... 131	IO-Link port 4 IN	32 byte IO-Link input process data
132 ... 163	IO-Link port 5 IN	32 byte IO-Link input process data
164 ... 295	IO-Link port 6 IN	32 byte IO-Link input process data
296 ... 227	IO-Link port 7 IN	32 byte IO-Link input process data
228 ... 259	IO-Link port 8 IN	32 byte IO-Link input process data

Assembly instance 105

Byte	Function	Description
0 ... 3	Status bytes	IO-Link master status bytes
4 ... 35	IO-Link port 1 IN	32 byte IO-Link input process data
36 ... 42		Extended IO-Link port status
43 ... 74	IO-Link port 2 IN	32 byte IO-Link input process data
75 ... 81		Extended IO-Link port status
82 ... 113	IO-Link port 3 IN	32 byte IO-Link input process data
112 ... 120		Extended IO-Link port status
121 ... 152	IO-Link port 4 IN	32 byte IO-Link input process data
153 ... 159		Extended IO-Link port status
160 ... 191	IO-Link port 5 IN	32 byte IO-Link input process data
192 ... 198		Extended IO-Link port status
199 ... 230	IO-Link port 6 IN	32 byte IO-Link input process data
231 ... 237		Extended IO-Link port status
238 ... 276	IO-Link port 7 IN	32 byte IO-Link input process data
270 ... 276		Extended IO-Link port status
277 ... 308	IO-Link port 8 IN	32 byte IO-Link input process data
309 ... 315		Extended IO-Link port status

Assembly instance 107

Byte	Function	Description
0 ... 3	Status bytes	O-Link master status bytes
4 ... 35	IO-Link port 1 IN	32 byte IO-Link input process data
36 ... 42		Extended IO-Link port status
43 ... 51		IO-Link event
52 ... 83	IO-Link port 2 IN	32 byte IO-Link input process data
84 ... 90		Extended IO-Link port status
91 ... 99		IO-Link event
100 ... 131	IO-Link port 3 IN	32 byte IO-Link input process data
132 ... 138		Extended IO-Link port status
139 ... 147		IO-Link event
148 ... 179	IO-Link port 4 IN	32 byte IO-Link input process data
180 ... 186		Extended IO-Link port status
187 ... 195		IO-Link event
196... 227	IO-Link port 5 IN	32 byte IO-Link input process data
228 ... 234		Extended IO-Link port status
235 ... 243		IO-Link event
244 ... 275	IO-Link port 6 IN	32 byte IO-Link input process data
276 ... 282		Extended IO-Link port status
283 ... 291		IO-Link event
292 ... 323	IO-Link port 7 IN	32 byte IO-Link input process data
324 ... 330		Extended IO-Link port status
331 ...339		IO-Link event
340 ... 371	IO-Link port 8 IN	32 byte IO-Link input process data
372 ... 378		Extended IO-Link port status
379 ... 388		IO-Link event

IO-Link master status bytes

The first two status bytes provide information for the IO-Link ports about the communication status to the IO-Link device (COM State) and whether the IO-Link process data is valid or not (PD Valid State).

Byte	Byte 0								Byte 1							
Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Port	8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1
Function	COM State								PD Valid State							
	0 = Communication is not active								0 = Invalid data							
	1 = Communication is active								1 = Valid data							

Byte 2 shows the status of the IO-Link port in the configured digital input mode. Byte 3 shows the status of the hardwired digital input of the type A ports.

Byte	Byte 2								Byte 3							
Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Port	8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1
Function	DI state of Pin 4 (C/Q) (Type A and type B ports)								Reserved				DI state of Pin 2 (Type A ports)			
	0 = Input is not active												0 = Input is not active			
	1 = Input is active												1 = Input is active			

32 Byte IO-Link input process data

EtherNet/IP™ uses the little endian format (Intel), IO-Link the big endian format (Motorola). For the IO-Link process data, no byte swapping is performed.

Input data from the IO-Link device is received as is and copied directly into the input assembly.

Byte	Function	Description
4	IO-Link port 1 IN	IOLD input process data byte 0
5		IOLD input process data byte 1
...		...
34		IOLD input process data byte 30
35		IOLD input process data byte 31

The structure for other IO-Link ports is the same.

Extended IO-Link port status

Byte	Function	Description	
0	IO-Link Status	Bit 0	IO-Link operating mode for ports is active.
		Bit 1	IO-Link-Device connected and communication is active.
		Bit 2 ... bit 7	Reserved
1	IO-Link-Error	Bit 0	Incompatible device found (wrong Vendor ID and/or Device ID)
		Bit 1	Data storage error
		Bit 2	Device detected, which has invalid parameters in the Direct Parameter Page 2.
		Bit 3 ... bit 7	Reserved
2	Vendor ID of the connected device	Vendor ID byte 1 (LSB)	
3		Vendor ID byte 2 (MSB)	
4	Device ID of the connected device	Device ID byte 1 (LSB)	
5		Device ID byte 2	
6		Device ID byte 3 (MSB)	

IO-Link events

Byte	Function	Description	
0	Event Qualifier	Bit 0 ... bit 2	**Event Instance**
			0 = Unknown
			1 ... 3 = Reserved
			4 = Application
			5 ... 7 = Reserved
		Bit 3	**Event Source**
			0 = Device application (Remote)
			1 = Master application (Local)
		Bit 4 ... bit 5	**Event Type**
			0 = Reserved
			1 = Notification
			2 = Warning
			3 = Error
		Bit 6 ... bit 7	**Event Mode**
			0 = Reserved
			1 = Event single shot
			2 = Event disappears
			3 = Event appears
1	Event Code 1 (LSB)	The reported Event Codes of the IO-Link device are mapped 1:1 to the these bytes. Please refer to the specific IO-Link device data sheet for the definition of the event codes.	
2	Event Code 2 (MSB)		
3	Event Qualifier	Data of the IO-Link events 2 (see byte 0 … byte 2)	
4	Event Code 1 (LSB)		
5	Event Code 2 (MSB)		
6	Event Qualifier	Data of the IO-Link events 3 (see byte 0 … byte 2	
7	Event Code 1 (LSB)		
8	Event Code 2 (MSB)		

3.2 Output process data

Assembly instance 100

Byte	Function	Description
0 ... 3	Control bytes	IO-Link master control bytes

Assembly instance 102

Byte	Function	Description
0 ... 3	Control bytes	IO-Link master control bytes
4 ... 35	IO-Link port 1 OUT	32 byte IO-Link output process data
36 ... 67	IO-Link port 2 OUT	32 byte IO-Link output process data
68 ... 99	IO-Link port 3 OUT	32 byte IO-Link output process data
100 ... 131	IO-Link port 4 OUT	32 byte IO-Link output process data
132 ... 163	IO-Link port 5 OUT	32 byte IO-Link output process data
164 ... 295	IO Link port 6 OUT	32 byte IO-Link output process data
296 ... 227	IO-Link port 7 OUT	32 byte IO-Link output process data
228 ... 259	IO-Link port 8 OUT	32 byte IO-Link output process data

IO-Link master control bytes

With byte 0 of the IO-Link master control bytes it is possible to configure the IO-Link ports into IO-Link mode (as long as the COM control bit is set), if they are previously working in the digital input mode (DI, this is the default mode).

This allows a re-parameterization of the IO-Link ports during actual operation.

Byte	**Byte 0**								**Byte 1**							
Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Port	8	7	6	5	4	3	2	1	Whole device							
Function	**COM Control (Port configuration)**								**Restart (Reboot)**							
	0 = Use configuration of Assembly instance106															
	1 = Activate IO-Link mode for port **Only possible for ports that are running in DI mode**															

Byte 2 (DO state of pin 4 (C/Q) allows the control (Set or reset) of IO-Link ports in digitale output mode (DO).

Byte	**Byte 2**								**Byte 3**							
Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Port	8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1
Function	**DO state of Pin 4 (C/Q)**								**Reserved**							
	0 = Reset the output								0 = No function							
	1 = Set the output								1 = Not allowed							

32 byte IO-Link output process data

EtherNet/IP™ uses the little endian format (Intel), IO-Link the big endian format (Motorola). For the IO-Link process data, no byte swapping is performed.

Output data are written to the IO-Link device as received in the output assembly.

Byte	Function	Description
4	IO-Link port 1 OUT	IOLD output process data byte 0
5		IOLD output process data byte 1
...		...
34		IOLD output process data byte 30
35		IOLD output process data byte 31

The structure for other IO-Link ports is the same.

0.2 Configuration data

Assembly instance 106

Byte	Data type	Function
0	UINT8	Reserved
1	UINT8	Reserved
2	UINT8	Fail Safe Mode
3	UINT8	Fail Safe Pattern DO pin 4
4	UINT8	Reserved
5	UINT8	Reserved
6	UINT8	Port Mode port 1
7 ... 8	UINT16	Vendor ID port 1
9 ... 12	UINT32	Device ID port 1
13	UINT8	Data Storage port 1
14	UINT8	IO-Link Fail Safe Mode port 1
15	UINT8	Enable Direct Parameter port 1
16 ... 31	Array of 16 x UINT8	Direct Parameter Page 2 port 1
32 ... 213	...	26 parameter bytes per port

Fail Safe Mode

This parameter is used for setting the substitute value behavior of IO-Link ports in "Digital Output (DO)" mode.

Adjustable values are:

Code (hex)	Description
00	Reset, DO -> 0
01	Set, DO -> 1
02	Hold last value
03	Substitute value pattern Fail safe pattern DO Pin 4 is active.

If the IO-Link port is working in IO-Link mode than the port specific behavior is active (please refer to byte 14).

Fail Safe Pattern DO Pin 4

The substitute value can be adjusted individually for every digital output with this parameter.

Example:

The outputs 1, 4, 5 and 8 shall be reset to "0" and the outputs 2,3, 6 and 7 shall be set to "1".

Bit	7	6	5	4	3	2	1	0
DO	8	7	6	5	4	3	2	1
Muster	0	1	1	0	0	1	1	0
hex	= 66							

The pattern is only active if 03_{hex} is used in the parameter **Fail Safe Mode**.

Port Mode

With this parameter you can set the runtime performance of the individual IO-Link port.

Code (hex)	Description	
00	Deactivated	The IO-Link port (Pin 4) is deactivated. In this operating mode, the sensor supply voltage is also switched off.
01	Digital input (DI)	In this operating mode, the IO-Link port functions are a standard digital input. **The process data can be found in byte 2 of the IO-Link master status bytes.**
02	Digital output (DO)	In this operating mode, the IO-Link port functions are a standard digital output. **The process data can be found in byte 2 of the IO-Link master control bytes**.
03	DI with IO-Link	The operating mode can be used if an IO-Link device is to be acquired as quickly as possible. The device can be parameterized via IO-Link (C/Q state is IO-Link). After parameterization, the IO-Link master switches the C/Q cable to the DI mode (C/Q state is DI). The switching state of the device is now acquired as a digital signal rather than via IO-Link communication. **The process data can be found in byte 2 of the IO-Link master status bytes.**
04	IO-Link (IOL)	Use this operating mode to communicate with IO-Link sensors and actuators. You can set the individual process data length of the IO-Link port , Vendor ID and Device ID. **The process data can be found in the IO-Link input/output process data area.**

Vendor ID

The vendor ID of the connected IO-Link device for the respective port can be parameterized via this. The vendor ID can be found in the data sheet of the IO-Link device.

If the Vendor ID is set, the Device ID must be set as well, because both are always checked together.

As soon as the content of the Vendor ID and/or Device ID is not equal to "0", the IO-Link "Type Compatible" inspection level is activated. It is only if the Vendor ID parameterized as well as the Device ID correspond with the information read out (in the device) that communication to the device is established (COM State bit of the corresponding port = 1), otherwise this is rejected (IO-Link LED is red).

Example:

The Vendor ID for port 1 shall be set.

– The Vendor ID is a 16 bit value (2 byte): exemplary 2211_{hex}
– The mapping is done according to the Intel format, i.e. LSB/MSB.

Byte	Data (hex)
7	11
8	12

Device ID

The Device ID of the connected IO-Link device for the respective port can be parameterized via this. The Device ID can be found in the data sheet of the IO-Link device. To check the Device ID, a Vendor ID check must be carried out first of all.

Example:

The Device ID for port 1 shall be set.

– The Device ID is a 24 Bit value (3 byte): exemplary $CCBBAA_{hex}$
– The mapping is done according to the Intel format, i.e. LSB/MSB.

Byte	Data (hex)
9	AA
10	BB
11	CC
12	00

Data Storage

The data storage mechanism allows the exchange of parameters between the master and device. To use this mechanism connected devices must at least support the IO-Link specification v1.1.

Code (hex)	Description	
0000	**Deactivated (Default)**	The Data Storage mechanism is deactivated.
0001	Download only	The parameter data are sent to the device from the IO-Link master. In the event of an inconsistency between the parameter data of the IO-Link device and the master, the data from the IO-Link master is taken as the default. As such it is possible to exchange the IO-Link device.
0002	Upload only	The parameter data are sent from the IO-Link device to the master. In the event of an inconsistency between the parameter data of the IO-Link device and the master, the data from the device is taken as the default. As such it is possible to exchange the IO-Link master.
0003	Download/upload allowed	The parameter data are saved in both the IO-Link master as well as the device. In the event of an inconsistency between the parameter data of the IO-Link device or the master, the data from both is used as the default. As such it is possible to exchange the IO-Link device or the IO-Link master.
0004	Cleared	The Data Storage mechanism is deactivated and the master deletes all stored parameters for the respective port.

IO-Link Fail Safe Mode

Substitute behavior for the OUT process data of the IO-Link port in the IO-Link operating mode. Adjustable values are:

Code (hex)	Description
00	Clear, DO -> 0
01	Set, DO -> 1
02	Hold last value
03	Reserved
04	**IO-Link Master Command (Default)**

The IO-Link master command option enables the use of IO-Link-specific mechanisms for valid/invalid OUT process data.

Enable Direct Parameter

This parameter determines whether the data of the Direct Parameter Page 2 should be transferred to the IO-Link device or not.

Code (hex)	Description
00	**Do not transfer data (Default)**
01	Transfer data

Direct Parameter Page 2 (DPP2)

The Device Parameters Page 2 (DPP2) describes the area between the IO-Link objects 10_{hex} … $1F_{hex}$. This is the manufacturer specific range of IO-Link device data.

The following example for port 1 shows the actual mapping of the DPP2 objects, the mapping is done transparently:

Byte	DPP2 object (hex)
16	10
17	11
...	...
30	1D
31	1F

4 Startup

4.1 Delivery state/default settings

By default upon delivery, the following functions and features are available:

IP settings

IP parameters:	0.0.0.0
Subnet mask:	0.0.0.0
Default Gateway:	0.0.0.0
BootP:	Activated

Firmware update

Firmware update on next restart:	Deactivated
TFTP server IP address:	192.168.210.211
Firmware file name:	FIRMWARE.NXF

System identification

Device name:	No name assigned
Description:	No description assigned
Device location:	No location assigned
Contact:	No contact assigned

Web-based Management (WBM)

User name:	admin
Password:	private

4.2 Starting the firmware

Once you have connected the power, the firmware is started. After completion of the firmware boot process, the NET LED either lights up or flashes green.

5 IO-Link master

IO-Link is an internationally standardized I/O technology (IEC 61131-9) for communicating with sensors and actuators.

The device supports IO-Link specification v1.1.

An IO-Link master is integrated in the device. The IO-Link master establishes the connection between the IO-Link devices and the automation system.

6 Connection monitoring/Substitute behavior

The device executes connection monitoring, which is configured for the device using a Time Out Multiplier scanner.

In the event of a connection timeout, the device switches its outputs to the previously configured safe state.

For further information, please refer to the following chapters:

– Fail Safe Mode
– Fail Safe Pattern DO Pin 4
– IO-Link Fail Safe Mode

7 Diagnostic messages

The following diagnostic messages are indicated by the EtherNet/IP™ device:

Diagnostic messages	Status bit process data	Network-specific diagnostic mapping
Overtemperature device	-	Yes
Undervoltage U_S	-	Yes
Overload U_S	-	Yes
Overvoltage U_S	-	Yes
Undervoltage/Overload/ Short-circuit U_A	-	Yes
Overvoltage U_A	-	Yes
Overload/Short-circuit of an output	-	Yes
Cable break at IO-Link port	Port status/ Status module*	Yes

***** Port status = 1; the IO-Link connection is active.
***** Port status = 0; the IO-Link connection is interrupted.
The LED status can be found in the data sheet.

8 Firmware update

In order to update the firmware of the device, the device must be provided with a firmware container via a TFTP server or it must be loaded onto the device via FTP. Any FTP client or TFTP server can be used for this. The update must always be initiated by the web-based management. When carrying out the firmware update, the RDY LED flashes yellow.

9 Restarting the device (Reboot)

The device can be restarted (rebooted) via process data. Therefore you have to follow this sequence:

Output control bytes (hex)				Sequence steps (Meaning)	Input status bytes (hex)			
0	1	2	3		0	1	2	3
xx	xx	xx	xx	Normal data communication	xx	xx	xx	xx
xx	52	xx	xx	**Sequence step 1 (Write)** – Reset-Sequences is initiated (ASCII: R) – Control bytes und status bytes go into "Restart Mode"	xx ...	xx ...	xx ...	xx ...
xx	xx	xx	xx	**Sequence step 2 (Read)** – When 52_{hex} in byte 1 of the control bytes is received, all four status bytes show 52 20 59 $3F_{hex}$ (ASCII: R Y?)	52	20	59	3F
00	59	00	00	**Sequence step 3 (Write)** – Now the outputs can be put into a defined state – 00 59 00 00_{hex} has to bee written to all four control bytes (ASCII: Y). – IO-Link ports in IO-Link operating mode stop IO-Link communication (connected IO-Link devices switch to the defined substitute behavior) – IO-Link ports in digital output operating mode switch their defined substitute behavior	xx ...	xx ...	xx ...	xx ...
xx	xx	xx	xx	**Sequence step 4 (Read)** Status bytes show 00 00 00 00_{hex}, if all outputs and IO-Link ports have transitioned into a defined state.	00	00	00	00
52	20	59	21	**Sequence step 5 (Write)** – To restart the device write 52 20 59 21_{hex} into all four control bytes (ASCII: R Y!). – Device restarts immediately after receiving this sequence.	xx	xx	xx	xx

Any deviation from the above sequence will abort the reboot sequence.

10 Restoring the device (Factory default setting)

The following options are available to restore the factory default settings:

Rotary encoding switches

Switch position 0F, please refer to chapter **Configuration via rotary encoding switch**

Web-based Management

Navigate to "Administration > Factory Defaults" and follow the instructions.

Identity object

Send a Request service with the reset type 1 to the Identity object.

11 WBM - Web-based management

The device has a web server, which generates the required pages for web-based management and, depending on the requirements of the user, sends them to a standard web browser. Web-based management can be used to access static information (e.g., technical data, MAC address) or dynamic information (e.g., IP address, status information).

Calling web-based management

The device web server can be addressed using the IP address if configured accordingly. The homepage (web page) of the device is accessed by entering the URL “http://ip-address”.
Example: http://172.16.113.38
The default user name is “admin”, the default password is “private”.

If you cannot access the WBM pages, check the connection settings in your browser and deactivate the proxy, if set.

12 SNMP - Simple Network Management Protocol

The device supports SNMP v1.

Management Information Base - MIB

The corresponding latest MIBs are available on the Internet at phoenixcontact.net/products.

For the object descriptions, please refer to the ASN1 descriptions for this product.

The password for read access is “public” and cannot be changed.
By default upon delivery, the password for write/read access is “private” and can be modified at any time.

13 DLR - Device Level Ring

The device supports DLR.

DLR is a protocol that works on Layer 2 for multi-port-EtherNet/IP™ devices.

The use of DLR is transparent for the protocols which work on higher levels such as TCP/IP.

DLR increases the network availability, thanks to a redundant ring topology with a switch-over time of less than 3 ms for error detection and reconfiguration.

The DLR protocol supports the 1-ring topology; multiple rings or overlapping rings are not possible.

14 Quick Connect

The device supports the Quick Connect Class A. The device is ready to process a connection request after power on within 350 ms. Cyclic process data communication is possible within 500 ms.

When operating as an IO-Link master, the actual ready up time depends on the IO-Link port configuration and the connected IO-Link devices.

15 Connection types

The device supports the connection types Exclusive Owner, Input Only and Listen Only.

– Exclusive Owner and Input Only can be configured as multicast or point-to-point connections in "Target to Originator" direction from the scanner.
– Listen Only connections can only be used as multicast.

Axioline E EtherNet/IP™-devices transmit the data always with only one connection to the scanner.

16 Device Description File (EDS)

EtherNet/IP™ uses a device description file for various engineering tools. This file is called Electronic Data Sheet, short EDS.

For each Axioline E EtherNet/IP™ device one specific file is provided.

Use the EDS file in the RSLogix ™ 5000 software from version 1.20.00 (CPR 9 SR 5).

17 Data format

EtherNet/IP™ uses the Little Endian Format (Intel), IO-Link uses the Big Endian Format (Motorola). No byte swapping is performed for the IO-Link process data.

Input data from the IO-Link device is received as is and copied directly into the input assembly. Output data is written to the IO-Link device as received in the output assembly.

Please observe the following notes

General terms and conditions of use for technical documentation

Phoenix Contact reserves the right to alter, correct, and/or improve the technical documentation and the products described in the technical documentation at its own discretion and without giving prior notice, insofar as this is reasonable for the user. The same applies to any technical changes that serve the purpose of technical progress.

The receipt of technical documentation (in particular user documentation) does not constitute any further duty on the part of Phoenix Contact to furnish information on modifications to products and/or technical documentation. You are responsible to verify the suitability and intended use of the products in your specific application, in particular with regard to observing the applicable standards and regulations. All information made available in the technical data is supplied without any accompanying guarantee, whether expressly mentioned, implied or tacitly assumed.

In general, the provisions of the current standard Terms and Conditions of Phoenix Contact apply exclusively, in particular as concerns any warranty liability.

Phoenix Contact reserves the right to register its own intellectual property rights for the product identifications of Phoenix Contact products that are used here. Registration of such intellectual property rights by third parties is prohibited.

Other product identifications may be afforded legal protection, even where they may not be indicated as such.

How to contact us

Internet

Up-to-date information on Phoenix Contact products and our Terms and Conditions can be found on the Internet at:
phoenixcontact.com

Make sure you always use the latest documentation.
It can be downloaded at:
phoenixcontact.net/products

Subsidiaries

If there are any problems that cannot be solved using the documentation, please contact your Phoenix Contact subsidiary.
Subsidiary contact information is available at phoenixcontact.com.

Published by

PHOENIX CONTACT GmbH & Co. KG
Flachsmarktstraße 8
32825 Blomberg
GERMANY

Should you have any suggestions or recommendations for improvement of the contents and layout of our manuals, please send your comments to:
tecdoc@phoenixcontact.com

SCATTERGOOD & JOHNSON LTD
ELECTRICAL ENGINEERING & FLUID CONTROL DISTRIBUTORS
Est.1899

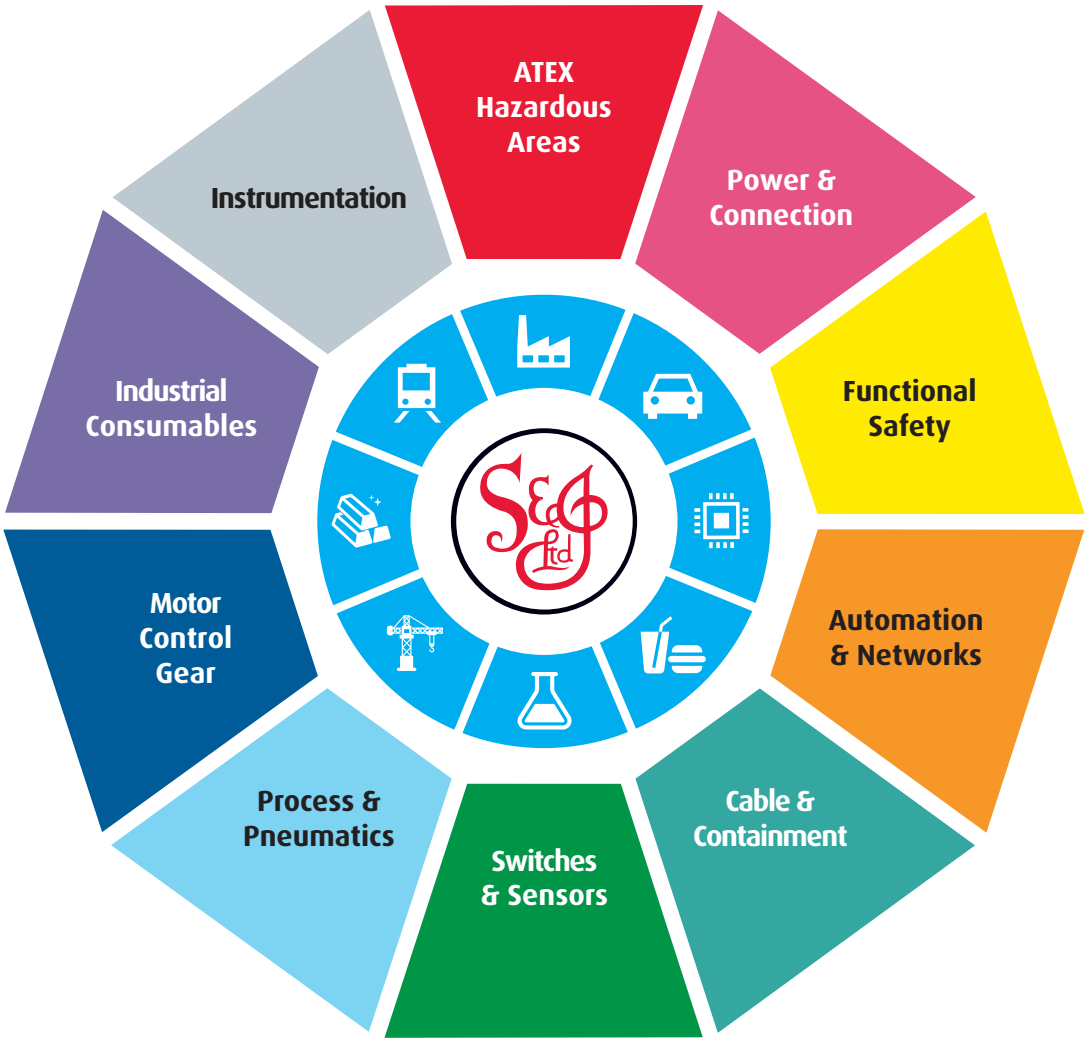
At Scattergood & Johnson Ltd, we pride ourselves on being a technical distributor to specialist industries.

Working with a range of quality product suppliers across a number of specialist markets, we are not your average 'box shifter' - we are your technical and supply chain partner.

We fully support every product we sell - for free! Our internal team and external sales engineers can answer any product or application question, no matter the complexity.

Backing up this technical ability is a range of 50,000+ products available from stock for nationwide next day delivery (same day if required!), or you can collect what you need from any of our trade counters around the UK.

Select your specialist interest below to learn more about how we can help.



Online, In Branch and On the Road - Scattergood & Johnson Ltd, there when you need us.

www.scatts.co.uk