Inhalt

S/IP protocol

.S/IP protocol

.General information

.Areas of application

S/IP facilitates easy access to drive parameters.

Typical fields of application are:

- Communication with the drive for Engineering via ctrlX DRIVE Engineering
- Replacing cyclic bus communication to control simple axis applications without real-time requirements, e.g. in positioning block mode or with drive-controlled positioning

.Features

- TCP/IP-based protocol
- The protocol focuses on the exchange of data and requires minimum administration overhead.
- User-defined busy timeout (time until the drive sends a defined response)
- User-defined lease timeout (time as of which the connection is enabled again if no new requests are made)
- Proprietary service for reading all parameter information in a request
- Up to 2 connections are possible simultaneously.

.Diagnostics involved

The following error messages have been defined, and the drive directly returns the error messages via an exception response:

.S/IP error classes

.S/IP error classes

Name	Value in the telegram	Significance
CONNECTION_ERROR	1	Error when establishing the connection
ABORTED	2	Process was aborted
UNKNOWN_MESSAGE_TYPE	3	Message type in the header is unknown

Name	Value in the telegram	Significance
SERVICE_SPECIFIC	4	Service-specific error, e.g. "Data is write protected"

.Additional information and details

.General information

S/IP stands for Sercos Internet Protocol.

In the following, an overview over the services provided in the S/IP protocol and the individual services is explained. Sample requests and responses are presented in the "Notes on commissioning".

A service always contains a request and a response. The request is always transmitted from the client, the server (drive) can respond to a request with different responses. If the request can be processed and answered immediately, then the drive transmits the associated response to the request directly (generally, Service number response = Service number request + 1). If the drive is busy with another request, a busy response (service number: 68) can also be transmitted. If the telegram was formatted incorrectly, the service is not supported or if another error occurred, an exception response (service number: 67) is returned.

Each telegram is composed of a static part and a service-specific part. In the static part, a package number can be assigned to assign request and response. The service executed by this telegram is specified.

Fig. 60: Basic structure of an S/IP telegram

Please note: Indexing using index "0" takes place in bit arrays and is based on the programming language representation. In other value arrays (byte, etc.), indexing starts at "1" and thus corresponds to the regular counting method (first byte, second byte, etc.).

The header consists of 8 bytes:

- Byte 1-4 specify the "TransactionID", an identification number defining this request. This number is specified again in a

response.
- Byte 5-8 define the "MessageType". The MessageType is the service number (for list of services supported by the drive, see below). As currently only services with a length of 1 byte are used, the low byte is relevant.

.Variable definitions

In addition to the data, the "ReadEverything-Response" and "ReadDescription-Response" services send a 2-byte value that specifies which elements are valid in the response. This "ValidElements" value contains a bit mask and is structured as follows:

.Valid elements

Bit mask	Description
0x01	Data status is valid
0x02	Description (Name) & description length (DescriptionLength) are valid
0x04	Attribute is valid
0x08	Unit and UnitLength are valid
0x10	Minimum value is valid
0x20	Maximum value is valid
0x40	Value and ValueLength are valid

The "ReadEverything request", "ReadOnlyData request", "ReadOnlyDescription request", "ReadDataStatus request", "WriteData request" and "WriteDataBits request" services need information on for which parameter the action is to be performed.

A 4-byte value has been transmitted to the drive. The structure corresponds to the following scheme.

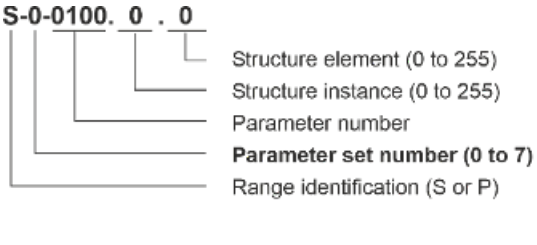

Fig. 61: Exemplary representation of a 4-byte IDN

.Structure of the 4-byte value

Bit No.	Description		Value range	Notes
31-24	Parameter instance/structure instance (SI)		0-255	
23-16	Parameter element/structure element (SE)	Sercos-defined SE	0-127	specified by Sercos (bit 15 = 0)
		product-specific SE	128-255	not in use
15	Sercos-specific or product-specific IDN (S or P)	Standard IDN (S-0-nnnn)	0	SE (0-127), SI and parameter number are specified via Sercos
		Product-specific IDN (P-0-nnnn)	1	product-specific parameter
14-12	Parameter set		0-7	see also "Parameter set switching"
11-0	Parameter number		0-4095	

.Supported services

The following table provides an overview of the services supported by the drive and the relevant "MessageType" for the request and the response.

.Supported S/IP services of ctrlX DRIVE

Service name	Service number	
	Request	Response
Connect	63	64

Service name	Service number	
	Request	Response
Ping	65	66
Exception	-	67
Busy	-	68
ReadEverything	69	70
ReadOnlyData	71	72
ReadDescription	73	74
ReadDataStatus	87	88
WriteData	83	84
WriteDataBits	85	86
Nameplate	89	90

.Connect

.Request

Structure:

Via the Connect service, the client connects to the drive. The MessageType "63" is sent. Moreover, the so-called busy timeout and lease timeout times are suggested in this telegram.

- **Busy timeout:** Specifies how much time can lapse until a response telegram to a query is received (unit: milliseconds).
- **Lease timeout:** Specifies after which time the drive automatically closes the connection if no new queries are received (unit: milliseconds).

Moreover, it is specified which protocol version is used by S/IP to the communication. The drive supports specification 1.0 and accordingly the value "1" is to be transmitted in the "Version" field.

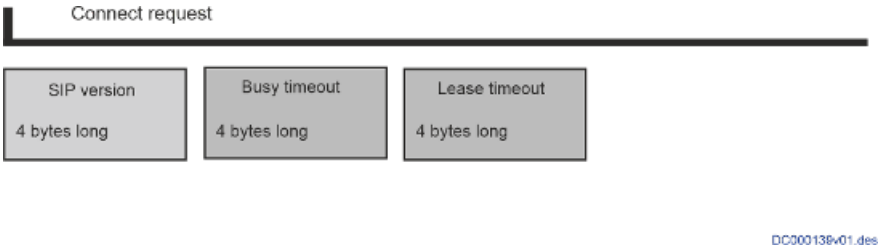

Fig. 62: Representation of a Connect request

.Response

The Connect response is transmitted as a response to a Connect request, if the connection was established successfully. The service is identified with service number "64" and provides information on the connection:

- which busy timeout is actually used
- which lease timeout is actually used
- which protocol version is actually used
- an array of all supported requests of the drive

.Connect response structure

Byte offset	Length (Byte)	Name	Significance	Example
0	4	Version	used protocol version of the drive	1
4	4	BusyTimeout	used busy timeout of the drive in milliseconds	500
8	4	LeaseTimeout	used lease timeout of the drive in milliseconds	15,000
12	4	NumberMessageTypes	Number of services supported by the drive	12
16	NumberMessageTypes * 4	MessageTypes	Array of 4-byte values containing the available services of the drive	-

.Ping

.Ping request

The Ping request is identified by service number "65". This is an empty telegram, i.e. the telegram contains 0 bytes of payload data. The service can be used to:

- maintain the connection, since the drive ends the connection after the lease timeout has lapsed
- Measure the minimum time that a response from the drive requires

.Ping response

The ping response is sent as response to a ping request via service number 66. The telegram does not contain any payload data.

.Busy & Exception

The "Busy" and "Exception" services are special services. These services can be send as response to any request.

.Busy response

The busy response is send if the response telegram has not yet been send once the busy timeout has lapsed. The busy response does not contain any data. Setting the busy timeout to higher values can suppress the response. However, higher values do not ensure that no busy response is send.

.Exception response

The exception response is sent if a general error occurred in the communication. There are various possibilities to explain why an exception response is transmitted, see ↘ “S/IP error classes”.

.ReadEverything

.ReadEverything request

Use the ReadEverything request to query all parameter data collectively. The service is identified via the service number "69". The SlaveIndex and the SlaveExtension is required. The parameter to be read is transmitted.

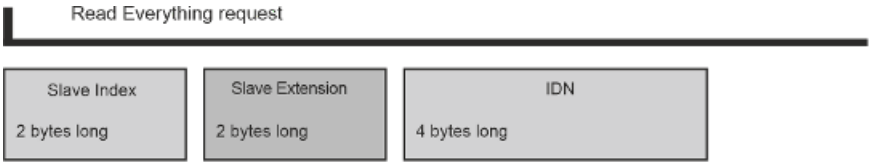

Fig. 63: Representation of a ReadEverything request

.Structure of ReadEverything request

Byte offset	Length (Byte)	Name	Significance	Example

Byte offset	Length (Byte)	Name	Significance	Example
0	2	SlaveIndex	only relevant for multi-axis devices (XMD). 0..n: Local axes	0
2	2	SlaveExtension	reserved - always 0	0
4	4	IDN	parameter to be read	0x30

.ReadEverything response

The ReadEverything response contains data for the "ReadEverything request". The service is identified via the service number "70".

The following values are transmitted:

- Data status
- Attribute
- Minimum value and maximum value
- Maximum list length in bytes (in case of list parameters)
- Current data length in the parameter in bytes
- Current parameter unit length in bytes
- Current name length in bytes
- Parameter data in binary format
- Parameter unit as string
- Parameter name as string

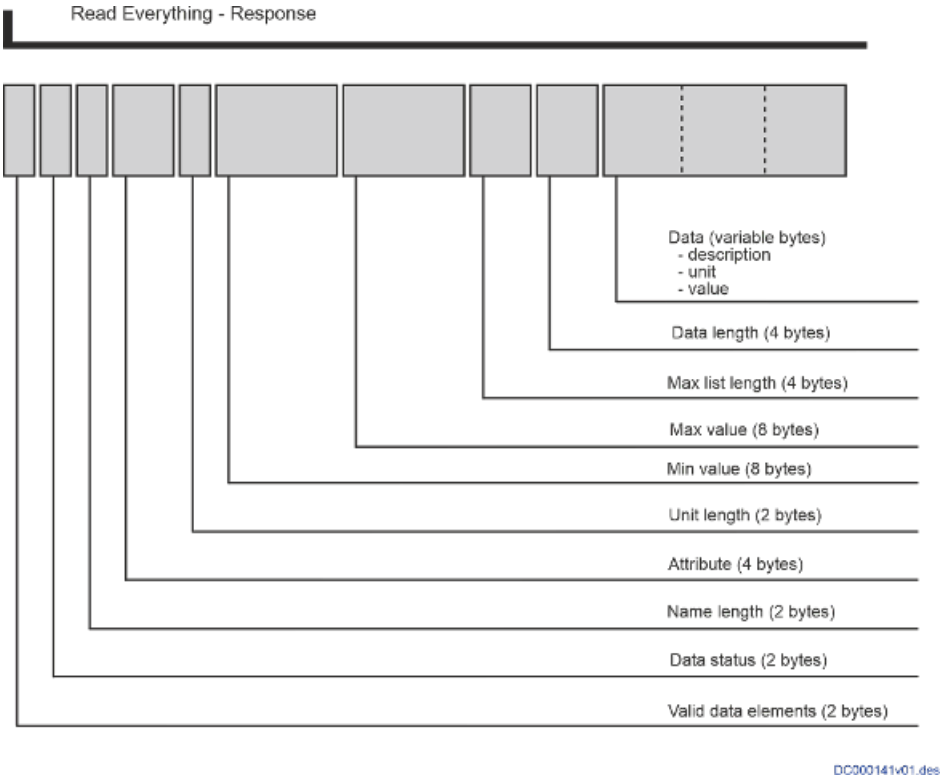

Fig. 64: Representation of the ReadEverything response

.Structure of ReadEverything response

.Structure of ReadEverything response

Byte offset	Length (Byte)	Name	Significance	Example
0	2	ValidElements	Bit string in which the valid elements are encoded	0b0000.0000.0100.11111
2	2	DataStatus	Data status of the parameter (valid/invalid) or command status in command parameters	1

Byte offset	Length (Byte)	Name	Significance	Example
4	2	NameLength	Length of the string that contains the parameter name in the language set in S-0-0265	0x20
6	4	Attribute	The attribute of the parameter contains information on decimal places, display format, maximum length, write protection and whether the parameter is a command parameter	0x77560000
10	2	UnitLength	String length containing the parameter unit (depending on the scaling)	6
12	8	Min	Lower input limit of this parameter	-
20	8	Max	Upper input limit of this parameter	-
28	4	MaxListLength	Maximum length of list parameter in bytes (only valid in case of list parameters)	8196
32	4	DataLength	Current parameter length	38

Byte offset	Length (Byte)	Name	Significance	Example
36	NameLength	Name	Parameter name	"Firmware version"
36 + NameLength	UnitLength	Unit	Parameter unit	""
36 + NameLength + UnitLength	DataLength	Data	The value of the parameter	"FWA-INDRV****"

.ReadOnlyData

.ReadOnlyData request

The operating data is read out via the "ReadOnlyData request". The query of "ReadOnlyData" is identical to the "ReadEverything request"; service number "71" is sent in the header.

.ReadOnlyData response

The "ReadOnlyData response" is identified via service number "72". The following data is transmitted:

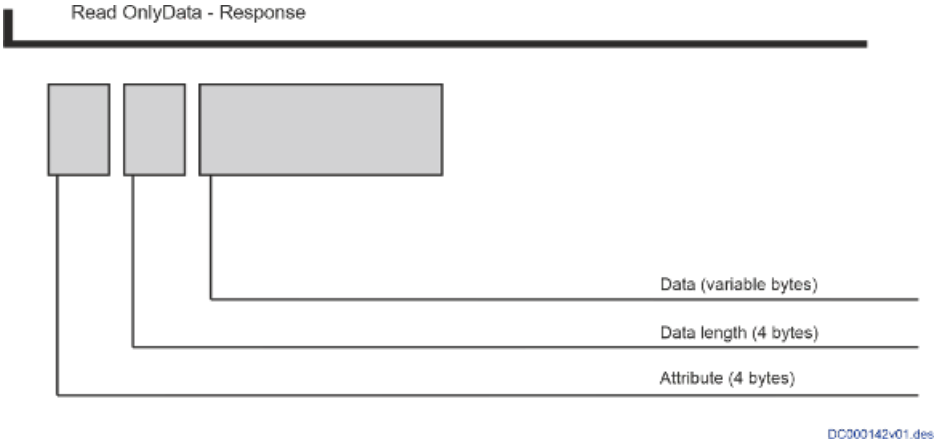

Fig. 65: Representation of the ReadOnlyData response

.Structure of ReadOnlyData response

Byte offset	Length (Byte)	Name	Significance	Example

Byte offset	Length (Byte)	Name	Significance	Example
0	4	Attribute	Attribute of the parameter	0x77560000
4	4	Length	Data array length in bytes	2
4	Length	Data	The data of the parameter	0x1345

.ReadDescription

.ReadDescription request

The "ReadDescription request" is used to request all information on the parameter, with the exception of the operating data per se. The query of "ReadDescription" is completely identical to the "ReadEverything request", but the service number "73" is used here.

.ReadDescription response

The "ReadDescription response" is identified via service number "74". The following data is transmitted:

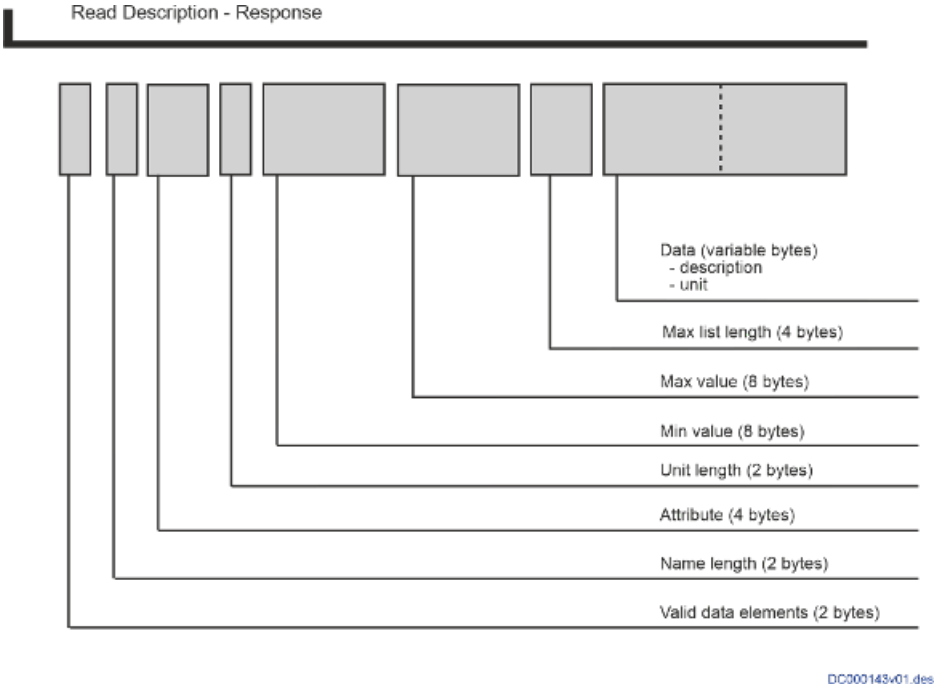

Fig. 66: Representation of the ReadDescription response

.Structure of ReadDescription response

Byte offset	Length (Byte)	Name	Significance	Example
0	2	ValidElements	Bit string in which the valid elements are encoded	0b0000.0000.0100.11111
2	2	NameLength	Length of the string that contains the parameter name in the language set in S-0-0265	0x20
4	4	Attribute	The attribute of the parameter contains information on decimal places, display format, maximum length, write protection and whether the parameter is a command parameter	0x77560000
8	2	UnitLength	String length containing the parameter unit (depending on the scaling)	6
10	8	Min	Lower input limit of this parameter	-
18	8	Max	Upper input limit of this parameter	-

Byte offset	Length (Byte)	Name	Significance	Example
22	4	MaxListLength	Maximum length of list parameter in bytes (only valid in case of list parameters)	8196
26	4	DataLength	Current parameter length	38
30	NameLength	Name	Parameter name	"Firmware version"
30 + NameLength	UnitLength	Unit	Parameter unit	""

.ReadDataStatus

.ReadDataStatus request

The data status can be read out directly via the "ReadDataStatus request". The query of "ReadDataStatus" is identical to the "ReadEverything request"; service number "87" is used here.

.ReadDataStatus response

The "ReadDataStatus response" is identified via service number "88". The following data is transmitted:

.Structure of ReadDataStatus response

Byte offset	Length (Byte)	Name	Significance	Example
0	2	DataStatus	Data status of the requested parameter	1

.WriteData

.WriteData request

The operating data is read out via "WriteData". Service number "83" is used here.

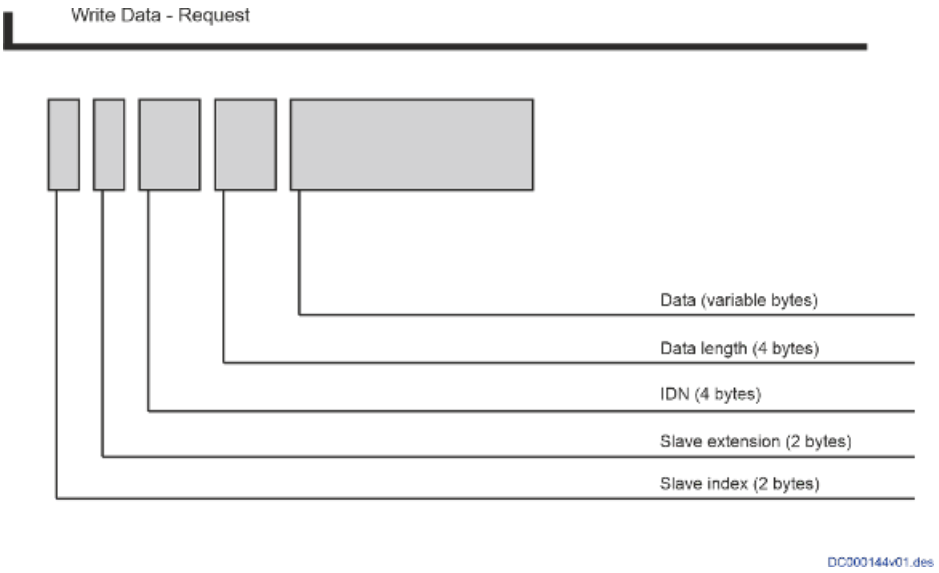

Fig. 67: Representation of the WriteData request

.Structure of WriteData request

Byte offset	Length (Byte)	Name	Significance	Example
0	2	SlaveIndex	only relevant for multi-axis devices (XMD). 0..n: Local axes	0
2	2	SlaveExtension	reserved - always 0	0
4	4	IDN	parameter to be read	0x30
4	4	DataLength	Length of transmitted data	2
8	DataLength	Data	the operating data to be transmitted	0x1234

.WriteData response

The "WriteData response" is identified via service number "84". No data is transmitted.

.WriteDataBits

.WriteDataBits request

Using the "WriteDataBits request", individual parameter bits can be written individually. The "WriteDataBits" request is identified via service number "85".

Any number of bits of a parameter value can be written in a request. Which bits are written depends on the DataMask which is part of the request. Thus, it is possible to change individual bits at any position in the parameter or to write all bits. The bits that are written have to be selected at the corresponding position in the "DataMask". If bit 7 is to be written, bit 7 has to be set to 1 in the "DataMask".

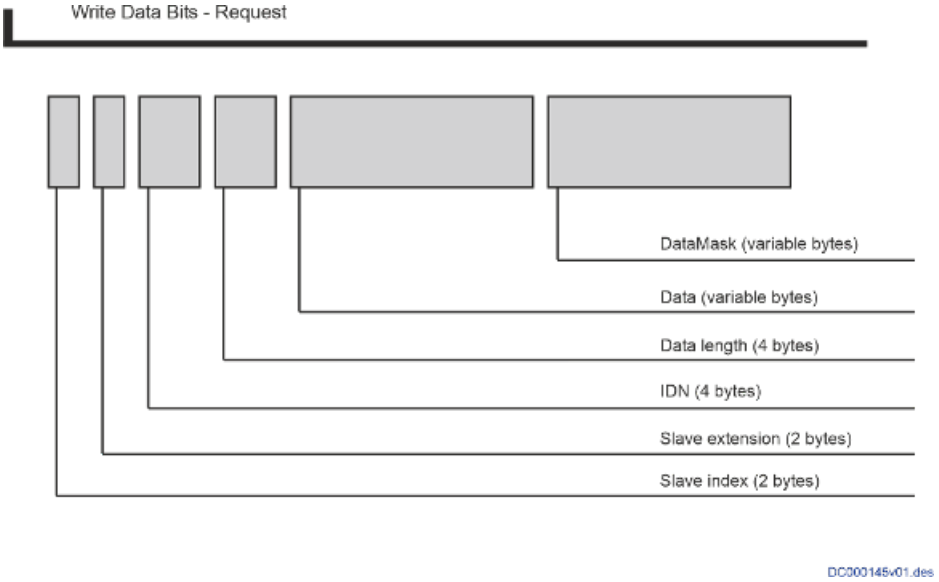

Fig. 68: Representation of the WriteDataBits request

.Structure of WriteDataBits request

Byte offset	Length (Byte)	Name	Significance	Example
0	2	SlaveIndex	reserved - always 0	0
2	2	SlaveExtension	reserved - always 0	0
4	4	IDN	parameter to be read	0x30
4	4	DataLength	Length of transmitted data	2
8	DataLength	Data	the operating data to be transmitted	0x1234

Byte offset	Length (Byte)	Name	Significance	Example
8 + DataLength	DataLength	DataMask	the bits to be updated	

.WriteDataBits response

The "WriteDataBits response" is identified via service number "86". No payload data are transmitted. In case of an error, an "Exception response" is returned instead of the WriteData response.

.Nameplate

.Nameplate request

In preparation

.Nameplate response

In preparation