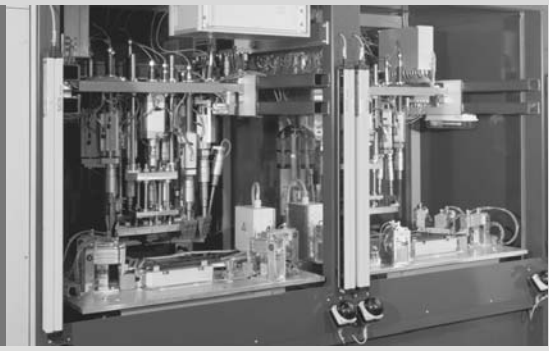
Flexi Soft



TELEGRAMM LISTING

SICK
Sensor Intelligence.

Contents

1 About this document 24
1.1 The function of this document 24
1.2 For whom this document is intended 24
1.3 Depth of information 24
1.4 Scope of validity 24

2 System description 25
2.1 System structure 25
2.1.1 Access via RS232 25
2.1.2 Access via Gateway 25
2.2 Electrical interface 26
2.2.1 RS232 Pin assignment 26
2.2.2 RS232 Transmission and data format 26
2.3 Telegram structure 27
2.3.1 Introduction 27
2.3.2 The command telegram 27
2.3.3 The reply telegram 29
2.3.4 Data order 31
2.3.5 CRC calculation 31
2.4 Application interface 32
2.4.1 Introduction 32
2.4.2 Authorization 32
2.4.3 Visualization read block 32
2.4.4 Process data write block 34
2.4.5 Error history 34

3 Glossary 35

4 Appendix 36
4.1 CRC calculation sample code 36

1 About this document

Please read this chapter carefully before you work with the documentation and the Flexi Soft.

1.1 The function of this document

This Telegram Listing Standard describes the mechanisms to communicate with the Flexi Soft system facilitating the SICK specific serial protocol. Also this document describes the data objects of the Flexi Soft, which can be used for diagnosis and for data exchange. It is to be considered as a supplement to the Flexi Soft Operating Instructions.

WARNING

Please see the Flexi Soft Operating Instructions, and read them carefully, for general information on, for example, mounting, installing and commissioning the safety controller.
Pay attention on the safety instructions before you operate the system for the first time!

The available data communication may not be used for safety purposes!

1.2 For whom this document is intended

This Telegram Listing Standard is intended for system specialists in hardware and software development who want to integrate and evaluate the Flexi Soft safety controller within their application, e.g. a PLC or a HMI.

1.3 Depth of information

This Telegram Listing Standard contains information on the following topics:

- Description of the RK512 protocol used
- Description of Flexi Soft safety controller special specific functions for data exchange of process data and diagnosis data

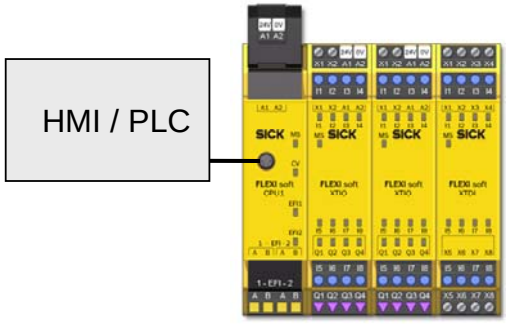
1.4 Scope of validity

This Telegram Listing Standard is applicable for the Flexi Soft safety controller with the following type label: FX3-CPU0 V1.00.0, FX3-CPU1 V1.00.0**.**

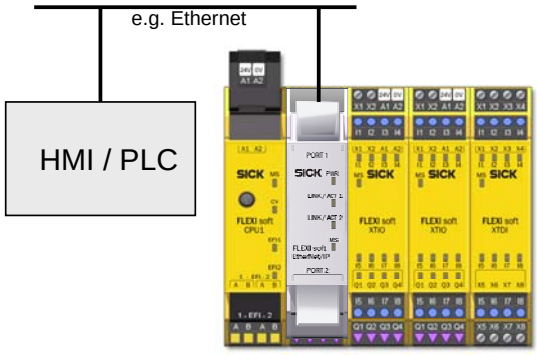
2 System description

2.1 System structure

2.1.1 Access via RS232



2.1.2 Access via Gateway

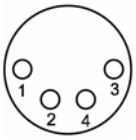


2.2 Electrical interface

The electrical interface is implemented according to the RS232 Standard.

Electrical connection is described in the “Electrical installation” chapter of the Flexi Soft Operating Instructions.

2.2.1 RS232 Pin assignment

Socket	Pin	Color	Assignment
	1	brown	Reserved (leave unconnected)
	2	white	RxD
	3	blue	GND
	4	black	TxD

Note The GND of the RS232 connector is internal electrical connected with terminal A2 of the CPU module. To avoid ground loops for permanent connection to the RS232 interface either leave the GND at the RS232 unconnected or use galvanic isolation elements for the interface, e.g. optocouplers.

2.2.2 RS232 Transmission and data format

The baud rate of RS232 interface is fixed to 115200 baud.

One data byte consists of 1 start bit, 8 data bits, 1 stop bit, no parity bit.

2.3 Telegram structure

2.3.1 Introduction

For the communication a protocol is used that is derived from the Siemens RK512. The original protocol is extended to meet the SICK requirements. As the original RK512 standard does not offer any mechanism for checking the integrity of the data received, such a mechanism is supplemented in the data of the RK512 telegram. For this the original RK512 data section is filled with the SICK RK512 data and also a exact repetition of bytes 5 to 10 from the header of a command telegram and a CRC.

In this document the SICK specific protocol is still named RK12, even it is extended.

Communication according to the RK512 standard is based on “command” and “reply” telegrams. A command telegram is either a “send” or a “fetch” telegram.

The client (e.g. host computer, HMI, PLC) transmits send telegrams with the data to be transferred after the telegram header; the recipient answers with a reply telegram without any further data. The host computer transmits fetch telegrams with the header of a fetch telegram without any subsequent data, and the sensor responds with a reply telegram which contains the requested data after the telegram header.

The client is also always the active participant. The server (Flexi Soft safety controller) does not transmit any RK512 telegrams on its own.

2.3.2 The command telegram

2.3.2.1 Fetch command telegram

Byte	Telegram fields	Content	Meaning
	--- Header ---		
0	Telegram identifier	0x00	
1		0x00	
2	Command telegram type	0x45	Fetch telegram
3	Command data type	0x44	Access to register interface
4	Destination address/ source address	0x00 .. 0xFF	Data block number
5		0x00	Reserved
6	Size (MSB)	0x0000 .. 0xFFFF	Size [words] = register interface block size [words] + 4 (for repeat telegram part and CRC)
7	Size (LSB)		
8	Coordination flag	0xFF	
9	Device address	0x00 .. 0xFF	0x4F Flexi Soft (local device)

2.3.2.2 Send command telegram

Byte	Telegram fields	Content	Meaning
	--- Header ---		
0	Telegram identifier	0x00	
1		0x00	
2	Command telegram type	0x41	Fetch telegram
3	Command data type	0x44	Access to register interface
4	Destination address/ source address	0x00 .. 0xFF	Data block number
5		0x00	Reserved
6	Size (MSB)	0x0000 .. 0xFFFF	Size [words] = register interface block size [words] + 4 (for repeat telegram part and CRC)
7	Size (LSB)		
8	Coordination flag	0xFF	
9	Device address	0x00 .. 0xFF	0x4F Flexi Soft (local device)
	--- Repeat ---		
10	Destination address/ source address	same as byte 4	Data block number
11		same as byte 5	Reserved
12	Size (MSB)	same as byte 6	Size [words] = register interface block size [words] + 4 (for repeat telegram part and CRC)
13	Size (LSB)	same as byte 7	
14	Coordination flag	same as byte 8	
15	Device address	same as byte 9	0x4F Flexi Soft (local device)
	--- Data ---		
16	Data byte 0		
..	..		
m	Data byte n		
	--- CRC ---		
m+1	CRC low byte		Calculated for byte 10 to byte m
m+2	CRC high byte		

2.3.3 The reply telegram

2.3.3.1 Fetch reply telegram

Byte	Telegram fields	Content	Meaning
	--- Header ---		
0	Telegram identifier	0x00	
1		0x00	
2	Reply telegram type	0x00	
3	Error number	0x00 .. 0xFF	0x00 : No error 0x01 .. 0xFF : See error table
	--- Repeat ---		Omitted if error
4	Destination address/ source address	0x00 .. 0xFF	Data block number
5		0x00	Reserved
6	Size (MSB)	0x0000 .. 0xFFFF	Size [words] = register interface block size [words] + 4 (for repeat telegram part and CRC)
7	Size (LSB)		
8	Coordination flag	0xFF	
9	Device address	0x00 .. 0xFF	0x4F Flexi Soft (local device)
	--- Data ---		Omitted if error
10	Data byte 0		
..	..		
m	Data byte n		
	--- CRC ---		Omitted if error
m+1	CRC low byte		Calculated for byte 4 to byte m
m+2	CRC high byte		

2.3.3.2 Send reply telegram

Byte	Telegram fields	Content	Meaning
	--- Header ---		
0	Telegram identifier	0x00	
1		0x00	
2	Reply telegram type	0x00	
3	Error number	0x00 .. 0xFF	0x00 : No error 0x01 .. 0xFF : See error table

2.3.3.3 Reply error codes

If the Flexi Soft detects an error it shows this in the error number byte of the reply telegram. The error telegram consists of 4 bytes only in any case.

Error code in reply telegram	RK512 protocol communication error
0x00	No error
0x01	RI access not allowed at current state
0x02	RI access denied at current access level
0x03	Invalid password
0x04	Device token not available
0x05	Parameter invalid, RK512 header or content of RI
0x08	RK512 handler is busy, RK512 request cannot be processed
0x0A	Source/destination parameter invalid or timeout occurred
0x0C	Coordination flag invalid or CPU number invalid
0x10	RK512 telegram ID invalid
0x14	Invalid block number
0x16	Invalid command type
0x34	RK512 block size incorrect, or limit of block size exceeded, or error in repeated header
0x36	Follow-on command telegram not supported
0x00	No error
0x01	RI access not allowed at current state
0x02	RI access denied at current access level
0x03	Invalid password
0x04	Device token not available
0x05	Parameter invalid, RK512 header or content of RI
0x08	RK512 handler is busy, RK512 request cannot be processed
0x0A	Source/destination parameter invalid or timeout occurred
0x0C	Coordination flag invalid or device number invalid
0x10	RK512 telegram ID invalid
0x14	Invalid block number
0x16	Invalid command type
0x34	RK512 block size incorrect, or limit of block size exceeded, or error in repeated header
0x36	Follow-on command telegram not supported

Table: Reply telegram error codes

2.3.4 Data order

The value in the size field always describes the number of 16-bit words; individual bytes cannot be accessed.

In the telegram header, 16-bit words are transmitted with the most significant byte (MSB: bit 8..15) first.

In the data bytes, 16-bit words are transmitted with the least significant byte (LSB: bit 0..7) first.

2.3.5 CRC calculation

CRC width: 16 bit

Polynomial: $x^{16}+x^{12}+x^5+x^0$, 0x1021 (CCITT-CRC)

Start value: 0xFFFF.

Byte sequence: Data from lowest address to highest address

For a sample code see 4.1 CRC calculation sample code.

2.4 Application interface

2.4.1 Introduction

The application interface of the Flexi Soft is called register interface (RI) and is organized in blocks (= objects), which can be accessed with the RK512 communication protocol.

2.4.2 Authorization

Depending on the block that is accessed a login for authorization is necessary and a system token needs to be requested. For the here listed register interface blocks no authorization and no system token is necessary. Therefore the blocks can be accessed without needing authorization to the system.

2.4.3 Visualization read block

The blocks allows to read a predefined selection of process data and status data from the system. The predefinition is done by configuration with the Flexi Soft Designer. This releases from collection the data from various blocks. Instead, all desired data can be achieved from this single block. There is a default configuration.

A customized configuration of the data content for the visualization block is currently not supported by Flexi Soft Designer.

Block number	118
Block size [words]	50
Access level	Operator (no token required)

Word	Bit	Description
0		Visualization data. Content according to configuration with Flexi Soft Designer
..		
49		

The data content of the visualization block contains the following default configuration:

Word	Bit	Description
0	0..7	Flex Bus+ Module 1 Input I1.. I8 (result of IO pre-evaluation, e.g. result of dual channel evaluation)
0	8..15	Flex Bus+ Module 2 Input I1.. I8
1	0..7	Flex Bus+ Module 3 Input I1.. I8
1	8..15	Flex Bus+ Module 4 Input I1.. I8
2	0..7	Flex Bus+ Module 5 Input I1.. I8
2	8..15	Flex Bus+ Module 6 Input I1.. I8
3	0..7	Flex Bus+ Module 7 Input I1.. I8
3	8..15	Flex Bus+ Module 8 Input I1.. I8
4	0..7	Flex Bus+ Module 9 Input I1.. I8
4	8..15	Flex Bus+ Module 10 Input I1.. I8
5	0..7	Flex Bus+ Module 11 Input I1.. I8
5	8..15	Flex Bus+ Module 12 Input I1.. I8
6	0..7	Flex Bus+ Module 1 Output Q1.. Q8 (for XTIO: Q5..Q8 are not used)
6	8..15	Flex Bus+ Module 2 Output Q1.. Q8
7	0..7	Flex Bus+ Module 3 Output Q1.. Q8
7	8..15	Flex Bus+ Module 4 Output Q1.. Q8
8	0..7	Flex Bus+ Module 5 Output Q1.. Q8
8	8..15	Flex Bus+ Module 6 Output Q1.. Q8

Word	Bit	Description
9	0..7	Flex Bus+ Module 7 Output Q1.. Q8
9	8..15	Flex Bus+ Module 8 Output Q1.. Q8
10	0..7	Flex Bus+ Module 9 Output Q1.. Q8
10	8..15	Flex Bus+ Module 10 Output Q1.. Q8
11	0..7	Flex Bus+ Module 11 Output Q1.. Q8
11	8..15	Flex Bus+ Module 12 Output Q1.. Q8
12	0..15	Logic result markers Result 0.0 .. Result 0.7, Result 1.0 .. Result 1.7
13	0..15	Logic result markers Result 2.0 .. Result 2.7, Result 3.0 .. Result 3.7
14	0..15	EFI 1 device 1 Input status message bit 0..15
15	0..15	EFI 1 device 1 Input status message bit 16..31
16	0..15	EFI 1 device 2 Input status message bit 0..15
17	0..15	EFI 1 device 2 Input status message bit 16..31
18	0..15	EFI 1 device 3 Input status message bit 0..15
19	0..15	EFI 1 device 3 Input status message bit 16..31
20	0..15	EFI 2 device 1 Input status message bit 0..15
21	0..15	EFI 2 device 1 Input status message bit 16..31
22	0..15	EFI 2 device 2 Input status message bit 0..15
23	0..15	EFI 2 device 2 Input status message bit 16..31
24	0..15	EFI 2 device 3 Input status message bit 0..15
25	0..15	EFI 2 device 3 Input status message bit 16..31
26	0..15	EFI 1 CPU module Output status message bit 0..15
27	0..15	EFI 1 CPU module Output status message bit 16..31
28	0..15	EFI 2 CPU module Output status message bit 0..15
29	0..15	EFI 2 CPU module Output status message bit 16..31
30	0..7	System status bits: bit 0: EFI 1 communication status bit 1: EFI 2 communication status bit 2: Configuration verify status bit 3..4: Reserved bit 5: Logic first cycle bit 6..7: Reserved
30	8..15	Not used
31	0..15	Location status bits (0 = bad, 1 = good): bit 0: CPU module bit 1: Flex Bus+ extension module 1 .. bit 12: Flex Bus+ extension module 12 bit 13: Flex Bus+ gateway module 1 bit 14: Flex Bus+ gateway module 2 bit 15: Reserved
32	0..15	Flex Bus+ extension module IO status bits (0 = bad, 1 = good): bit 0: Input data status module 5 bit 1: Input data status module 6 .. bit 7: Input data status module 12 bit 8: Output data status module 9 .. bit 11: Output data status module 12 bit 12: Input data status module 1 .. bit 15: Input data status module 4
33	0..7	Flex Bus+ extension module IO status bits (0 = bad, 1 = good): bit 0: Output data status module 1 .. bit 7: Output data status module 8
33	8..15	Not used
34	0..15	Not used
..		
49	0..15	Not used

2.4.4 Process data write block

The blocks allows to set the value of dedicated process data inputs for the logic.

Attention: This data is not safe related.

If no send access to this block has occurred for 60s the process data are set to safe value (logic 0).

The dedicated process data inputs are yet not available in the current Flexi Soft Designer version.

Block number	66
Block size [words]	2
Access level	Operator (no token required)

Word	Bit	Description
0	0..15	Logic RK512 input bit 0..15
1	0..15	Logic RK512 input bit 16..31

2.4.5 Error history

To be described.

3 Glossary

Designer	Configuration and diagnosis PC tool for Flexi Soft.
EFI	Enhanced function interface. Communication interface for connection intelligent sensors like C4000, M4000, S3000, S300 to the CPU Module.
Flex Bus+	Backplane of Flexi Soft system. Interface to connect extension modules, e.g. IO modules and gateways.
RI	Register Interface: Data application interface.

4 Appendix

4.1 CRC calculation sample code

```
CCITT, x16+x12+x5+x0 Polynome 0x1021, Start–Init-Value 0xFFFF */

const WORD crc_table[256] = {
 0x0000, 0x1021, 0x2042, 0x3063, 0x4084, 0x50a5, 0x60c6, 0x70e7,
 0x8108, 0x9129, 0xa14a, 0xb16b, 0xc18c, 0xd1ad, 0xe1ce, 0xf1ef,
 0x1231, 0x0210, 0x3273, 0x2252, 0x52b5, 0x4294, 0x72f7, 0x62d6,
 0x9339, 0x8318, 0xb37b, 0xa35a, 0xd3bd, 0xc39c, 0xf3ff, 0xe3de,
 0x2462, 0x3443, 0x0420, 0x1401, 0x64e6, 0x74c7, 0x44a4, 0x5485,
 0xa56a, 0xb54b, 0x8528, 0x9509, 0xe5ee, 0xf5cf, 0xc5ac, 0xd58d,
 0x3653, 0x2672, 0x1611, 0x0630, 0x76d7, 0x66f6, 0x5695, 0x46b4,
 0xb75b, 0xa77a, 0x9719, 0x8738, 0xf7df, 0xe7fe, 0xd79d, 0xc7bc,
 0x48c4, 0x58e5, 0x6886, 0x78a7, 0x0840, 0x1861, 0x2802, 0x3823,
 0xc9cc, 0xd9ed, 0xe98e, 0xf9af, 0x8948, 0x9969, 0xa90a, 0xb92b,
 0x5af5, 0x4ad4, 0x7ab7, 0x6a96, 0x1a71, 0x0a50, 0x3a33, 0x2a12,
 0xdbfd, 0xcbdc, 0xfbbf, 0xeb9e, 0x9b79, 0x8b58, 0xbb3b, 0xab1a,
 0x6ca6, 0x7c87, 0x4ce4, 0x5cc5, 0x2c22, 0x3c03, 0x0c60, 0x1c41,
 0xedae, 0xfd8f, 0xcdec, 0xddcd, 0xad2a, 0xbd0b, 0x8d68, 0x9d49,
 0x7e97, 0x6eb6, 0x5ed5, 0x4ef4, 0x3e13, 0x2e32, 0x1e51, 0x0e70,
 0xff9f, 0xefbe, 0xdfdd, 0xcffc, 0xbf1b, 0xaf3a, 0x9f59, 0x8f78,
 0x9188, 0x81a9, 0xb1ca, 0xa1eb, 0xd10c, 0xc12d, 0xf14e, 0xe16f,
 0x1080, 0x00a1, 0x30c2, 0x20e3, 0x5004, 0x4025, 0x7046, 0x6067,
 0x83b9, 0x9398, 0xa3fb, 0xb3da, 0xc33d, 0xd31c, 0xe37f, 0xf35e,
 0x02b1, 0x1290, 0x22f3, 0x32d2, 0x4235, 0x5214, 0x6277, 0x7256,
 0xb5ea, 0xa5cb, 0x95a8, 0x8589, 0xf56e, 0xe54f, 0xd52c, 0xc50d,
 0x34e2, 0x24c3, 0x14a0, 0x0481, 0x7466, 0x6447, 0x5424, 0x4405,
 0xa7db, 0xb7fa, 0x8799, 0x97b8, 0xe75f, 0xf77e, 0xc71d, 0xd73c,
 0x26d3, 0x36f2, 0x0691, 0x16b0, 0x6657, 0x7676, 0x4615, 0x5634,
 0xd94c, 0xc96d, 0xf90e, 0xe92f, 0x99c8, 0x89e9, 0xb98a, 0xa9ab,
 0x5844, 0x4865, 0x7806, 0x6827, 0x18c0, 0x08e1, 0x3882, 0x28a3,
 0xcb7d, 0xdb5c, 0xeb3f, 0xfb1e, 0x8bf9, 0x9bd8, 0xabbb, 0xbb9a,
 0x4a75, 0x5a54, 0x6a37, 0x7a16, 0x0af1, 0x1ad0, 0x2ab3, 0x3a92,
 0xfd2e, 0xed0f, 0xdd6c, 0xcd4d, 0xbdaa, 0xad8b, 0x9de8, 0x8dc9,
 0x7c26, 0x6c07, 0x5c64, 0x4c45, 0x3ca2, 0x2c83, 0x1ce0, 0x0cc1,
 0xef1f, 0xff3e, 0xcf5d, 0xdf7c, 0xaf9b, 0xbfba, 0x8fd9, 0x9ff8,
 0x6e17, 0x7e36, 0x4e55, 0x5e74, 0x2e93, 0x3eb2, 0x0ed1, 0x1ef0
};

/*****************************************************************
*Function: crc16:
*This function receives the current 16 bit crc value and
*a new byte which is to be added to the crc value. The
*new crc value is returned from this function.
*
*Parameters:
*CRC_Data - byte to be included in the CRC
*CRC_16 - present value of the CRC
*
*Returns:
*CRC_16 - new CRC value
*
*****************************************************************/

WORD crc16 (BYTE CRC_Data, WORD CRC_16)
{
CRC_16 = (CRC_16 << 8) ^ (crc_table[(CRC_16 >> 8) ^ (CRC_Data)]);

    RETURN CRC_16;
}
```