ONLINEHELP

Flexi Soft – RK512

RK512 Telegram-Listing

GB

SICK
Sensor Intelligence.

Content

1 About this document

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

1.1 Function of this document

This RK512 Telegram Listing online help 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.

- Flexi Soft Designer Software / Operating instructions 8012480
- Flexi Soft Modular Safety Controller / Operating instruction 8012478
- Flexi Soft Gateways / operating instruction 8012664

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 here described RK512 communication protocol does not meet the requirements for safety related data exchange. Therefore the data exchanged by RK512 must not be used for safety functions!

1.2 Target group

This RK512 Telegram Listing online help 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 RK512 Telegram Listing online help contains information on the following topics:

- Description of the RK512 protocol used
- Description of specific functions for data exchange of process data and diagnosis data (status bits and error history) for the Flexi Soft safety controller

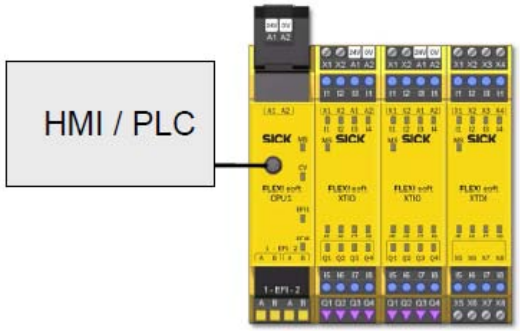
1.4 Scope

Unless otherwise stated, this RK512 Telegram Listing online help is applicable for the Flexi Soft safety controller with the following type label: FX3-CPU0, FX3- CPU1, FX3-CPU2 and FX3-CPU3.

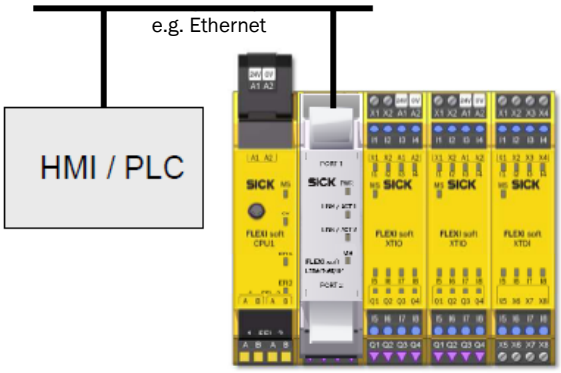
2 System description

2.1 System structure

2.1.1 Access via RS-232



2.1.2 Access via Gateway

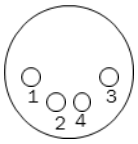


2.2 Electrical interface

2.2.1 RS-232

The Flexi Soft main modules (e.g. FX3-CPU0, FX3-CPU1, FX3-CPU2 and FX3-CPU3) support the RS232 interface.
The electrical interface is implemented according to the RS232 Standard.
The electrical connection is described in the “Electrical installation” chapter of the Flexi Soft Modular Safety Controller / Operating instruction (8012478).

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 internally connected with the terminal A2 of the Flexi Soft main module. For permanent connection to the RS232 interface either leave the GND at the RS232 interface unconnected or use galvanic isolation elements for the interface (e.g. optocouplers) to avoid ground loops.

The parameters for the RS232 are as following:

Baud rate	115200
Start bits	1
Data bits	8
Stop bits	1
Parity	No

This means that the data transmission time for each byte is 0.087ms.

2.2.2 TCP/IP

The TCP/IP interface is supported by certain Flexi Soft Ethernet gateways (e.g. ProfiNet, Modbus TCP, Ethernet/IP).
The electrical interface is a RJ45 connector. For electrical connection see Flexi Soft Gateways operating instruction (8012664).

The parameters for the TCP/IP are as following:

Ethernet Protocol	TCP/IP
IP address	see IP-Address in Gateway configuration in Flexi Soft Designer
Port	9000

The telegram streams for TCP / IP and RS232 are exactly the same, with identical header, etc.

2.2.3 USB

The Flexi Soft main module FX3-CPU3 supports an USB interface. However, this interface is not supported for usage with the RK512 Telegram-Listing, because that would require a special driver DLL at the client (PC, HMI, PLC).

2.3 Telegram structure

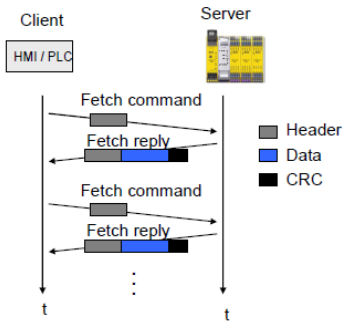
2.3.1 Introduction

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 address information, 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 with an exact repetition of bytes 5 to 10 from the header of a command telegram and also a checksum (CRC) for the data.

In this document the SICK specific protocol is still named RK512, 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.

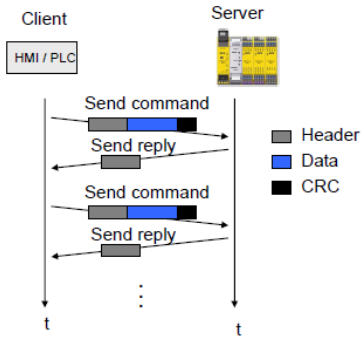
The client (e.g. host computer, HMI, PLC) is always the active participant. The server (Flexi Soft safety controller) does not transmit any RK512 telegrams on its own. If data has to be updated frequently, the client has to initiate for every update a new command telegram.

a) Reading data from Flexi Soft:



The client 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.

b) Writing data to Flexi Soft:



The client transmits send telegrams with the data to be transferred after the telegram header; the recipient answers with a reply telegram without any further data.

2.3.2 Fetch telegram

2.3.2.1 Fetch command telegram

Fetch command telegram: Send from client to Flexi Soft to read data:

Byte	Telegram fields	Content	Meaning
	Header		
0	Telegram identifier	0x00	
1		0x00	
2	Telegram type	0x45	Command fetch telegram
3	Command data type	0x44	Access to register interface
4	Destination address/ source address	0x00 ... 0xFF	Data block number
5		0x00 ... 0xFF	Structure index
6	RK512 size (MSB)	0x0000 ... 0xFFFF	= Size register interface block [words] + 4 (for repeat telegram part and CRC), e.g. Block size : 32 hex → Rk512 size : 0036 hex
7	RK512 size (LSB)		
8	Coordination flag	0xFF	
9	Device address	0x00 ... 0xFF	0x4F: Flexi Link/Flexi Soft local connected device, regardless configured EFI address 0x4E: Flexi Link Station A/Flexi Soft EFI address 14 0x4D: Flexi Link Station B/Flexi Soft EFI address 13 0x4C: Flexi Link Station C 0x4B: Flexi Link Station D

2.3.2.2 Fetch reply telegram

Fetch reply telegram: Send from Flexi Soft to client for delivering the read data:

Byte	Telegram fields	Content	Meaning
	Header		
0	Telegram identifier	0x00	
1		0x00	
2	Telegram type	0x00	Reply telegram
3	Error number	0x00 ... 0xFF	0x00: No error 0x01 ... 0xFF: See error table
	Repeat		
4	Destination address/ source address	0x00 ... 0xFF	Data block number
5		0x00 ... 0xFF	Structure index
6	RK512 size (MSB)	0x0000 ... 0xFFFF	= Size register interface block [words] + 4 (for repeat telegram part and CRC), e.g. Block size : 32 hex → Rk512 size : 0036 hex
7	RK512 size (LSB)		
8	Coordination flag	0xFF	
9	Device address	0x00 ... 0xFF	0x4F: Flexi Link/Flexi Soft local connected device, regardless configured EFI address 0x4E: Flexi Link Station A/Flexi Soft EFI address 14 0x4D: Flexi Link Station B/Flexi Soft EFI address 13 0x4C: Flexi Link Station C 0x4B: Flexi Link Station D
	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		

If Error number (byte 3) is unequal 0, then the repeat data and CRC parts (bytes 4 ... m+2) are not transmitted. The error telegram consists of only 4 bytes in any case.

2.3.3 Send telegram

2.3.3.1 Send command telegram

Send command telegram: Send from client to Flexi Soft to write data:

Byte	Telegram fields	Content	Meaning
	Header		
0	Telegram identifier	0x00	
1		0x00	
2	Telegram type	0x41	Command send telegram
3	Command data type	0x44	Access to register interface
4	Destination address/ source address	0x00 ... 0xFF	Data block number
5		0x00 ... 0xFF	Structure index
6	RK512 size (MSB)	0x0000 ... 0xFFFF	= Size register interface block [words] + 4 (for repeat telegram part and CRC), e.g. Block size : 32 hex → Rk512 size : 0036 hex
7	RK512 size (LSB)		
8	Coordination flag	0xFF	
9	Device address	0x00 ... 0xFF	0x4F: Flexi Link/Flexi Soft local connected device, regardless configured EFI address 0x4E: Flexi Link Station A/Flexi Soft EFI address 14 0x4D: Flexi Link Station B/Flexi Soft EFI address 13 0x4C: Flexi Link Station C 0x4B: Flexi Link Station D
	Repeat		
10	Destination address/ source address	same as byte 4	
11		same as byte 5	
12	RK512 size (MSB)	same as byte 6	
13	RK512 size (LSB)	same as byte 7	
14	Coordination flag	same as byte 8	
15	Device address	same as byte 9	
	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

Send reply telegram: Send from Flexi Soft to client to acknowledge the writing of data:

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

2.3.4 Device Address

The meaning of the device address is as following:

0x4F: Flexi Soft local connected device, regardless configured EFI address
0x4E: Flexi Link Station A / Flexi Soft EFI address 14
0x4D: Flexi Link Station B / Flexi Soft EFI address 13
0x4C: Flexi Link Station C
0x4B: Flexi Link Station D

NOTE Addressing of individual Flexi Link stations within a Flexi Link system requires that Flexi Link is used with EFI 1 and EFI 2 or it is ensured that Flexi Soft Designer is not used in the same power cycle!

Background: There is an EFI communication path selector in the main module to select between EFI 1 and EFI 2. This EFI communication path selector is used also by the Flexi Soft Designer, so that the actual selection may be for EFI 2. If EFI 2 is not connected, the communication to the other stations is not possible in this case.

The device address is by default 0x4E = Flexi Soft EFI address 14. The device address is saved in the system plug. This means that after a module replacement the same address applies.

2.3.5 Exception handling

Timeout in client:

In the client the time out for the reply of Flexi Soft should be according the following formula, measured from complete transmission of the command telegram:

Timeout = 1 sec + n * 8 ms = 1.088 sec + block size in words * 8ms

n = length in words of command and reply telegrams
= block size + 11 words for header and CRC

Retry after busy:

If the client receives an error code 'busy' (0x08), it has to wait at least 200 ms before it send the next command telegram.
This is to ensure that internal clients get the chance to be served.

2.3.6 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 only 4 bytes in any case.

Table: Reply telegram error codes

Error-Code in reply telegram	RK512 protocol communication error
0x00	No error
0x01	Register Interface access not allowed at current state
0x02	Register Interface access denied at current access level
0x03	Invalid password
0x04	Device token not available
0x05	Parameter invalid, RK512 header or content of the Register Interface
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 telegram field “Repeat”
0x36	Follow-on command telegram not supported

2.3.7 Order of data

The telegram streams are transmitted starting from byte 0.

2.3.8 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

Examples to verify your CRC calculation:
Byte 0 ... 15 in hex: 41 00 00 08 FF 4F 00 00 00 00 00 00 00 00
- CRC = B3 F1 (low byte, high byte)

Byte 0 ... 9 in hex: 00 01 02 03 04 05 06 07 08 09
- CRC = 41 C2 (low byte, high byte)

For a sample code how to calculate the CRC please refer to chapter 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. This means, that RK512 is just the vehicle to transport the data between the Flexi Soft and a client. The actual functionality is a dependent on the register interface blocks supported by Flexi Soft. In this document the related register interface blocks are described for:

- Reading process and status data (Visualization Read Block: RS232 routing from main module to network)
- Writing process data for use in Flexi Soft Logic program (Visualization Write Block: RS232 routing from network to main module)
- Reading module status bits
- Reading type key data
- Reading configuration CRCs
- Reading application name
- Reading Flexi Loop status data, if applicable

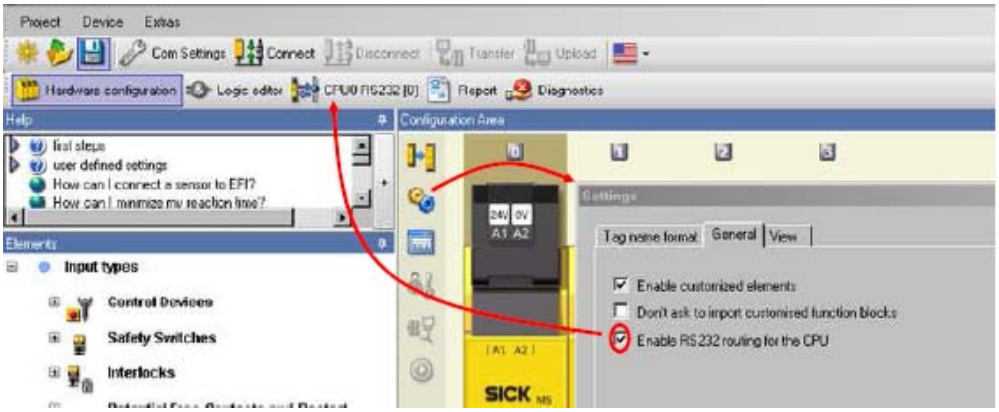
NOTE Even if it is called RS232 routing within the Flexi Soft Designer the register interface blocks can be also accessed via Flexi Soft gateways, provided the gateway and the network is capable to tunnel the RK512 protocol, e.g. via TCP/IP.

2.4.2 Enable RS232 routing in Flexi Soft Designer

To use the data of Visualization Read Block and Visualization Write Block, you need first to enable the configuration dialogues in the Flexi Soft Designer:

Go to Icon *Settings* in the Hardware configuration view, select the page *General* and enable check box *Enable RS232 routing for the CPU*.

Activation RS-232 routing



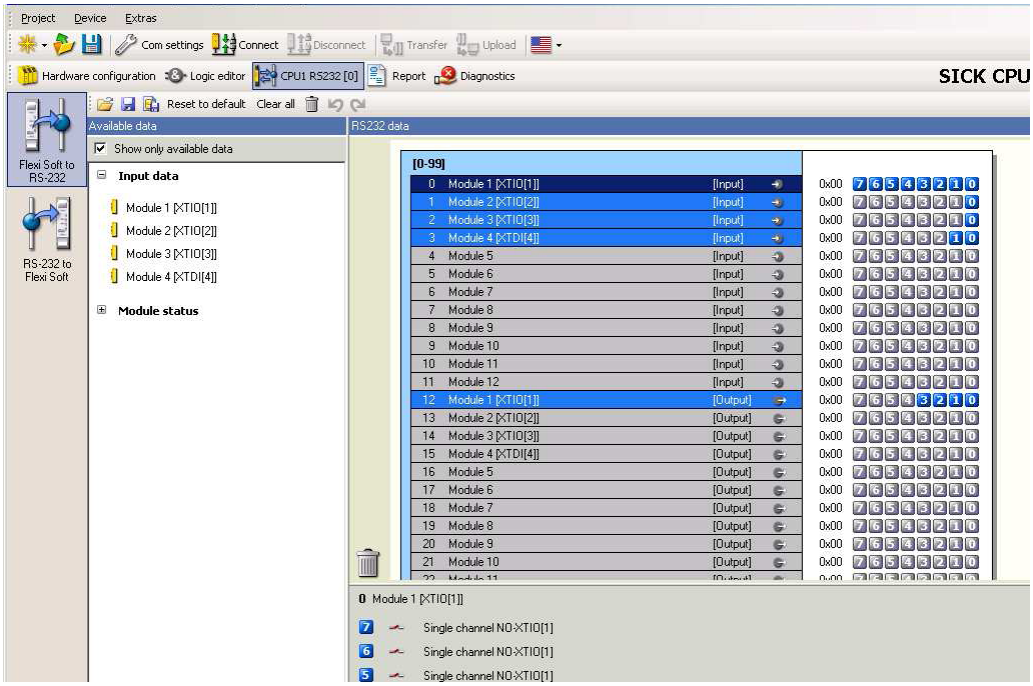
An additional icon in the view bar selection will appear. If gateways are used, the Icon for RS232 will be combined with the gateway routing icon.

If RS232 routing is enabled, the related configuration part will also be included in the report.

2.4.3 Visualization Read Block (RS232 routing from Flexi Soft to network)

The block is a data collector allowing reading up to 100 byte process data and status data from the Flexi Soft. The data that is collected by the Flexi Soft and provided in this block can be configured with the Flexi Soft Designer. In the dialog you can select the desired data and the desired position, individually for every byte. The handling in this dialog is the same as for definition of gateway data, except that here the data container has a size of 100 bytes.

Configuration of RS232 routing data from Flexi Soft to network



The content of the data as defined by this configuration will be included also in the report. This can be a helpful documentation for the evaluation of the data in the client HMI or PLC.

The block has a variable size, meaning that by the fetch request it can be determined how many bytes out of the maximum 100 bytes shall be returned, always starting from byte 0.

NOTE Using variable block size to read less than 100 bytes requires firmware V2.00.0 or higher.

Data sheet of Visualization Read Block:

Data sheet of Visualization Read Block

Block name	Visualization Read Block
Block number	118 (76 hex)
access level	Operator (no login with password required)
Block size [words]	Variable size: 1 ... 50 (1 ... 32 hex, → RK512 size = 0005 ... 0036 hex)
Data	Content according the configuration with Flexi Soft Designer for RS232 routing from Flexi Soft to network.

Examples:

Telegram streams of Visualization Read Block

The RK512 telegrams applied for the Visualization Read Block to read 100 bytes are:

Fetch command telegram (from client)	Telegram byte 0 ... 9 in hex to read 100 bytes: 00 00 45 44 76 00 00 36 FF 4F
Fetch reply telegram (from Flexi Soft)	Telegram byte 0 ... 9 in hex: 00 00 00 00 76 00 00 36 FF xx (xx = actual device address) telegram byte 10 ... 109: block data (Process data byte 0 ... 99) telegram byte 110 ... 111: CRC for byte 4 ... 109

The RK512 telegrams applied for the Visualization Read Block to read 10 bytes are:

Fetch command telegram (from client)	Telegram byte 0 ... 9 in hex to read 10 bytes: 00 00 45 44 76 00 00 09 FF 4F
Fetch reply telegram (from Flexi Soft)	Telegram byte 0 ... 9 in hex: 00 00 00 00 76 00 00 09 FF xx (xx = actual device address) telegram byte 10 ... 19: block data (Process data byte 0 ... 9) telegram byte 20 ... 21: CRC for byte 4 ... 19

When polling the Visualization Read Block via RS232, a polling rate of typically 24 ms can be achieved (for transmitting the fetch command, processing it and transmitting the reply telegram).

2.4.4 Visualiziation Write Block (RS-232-Routing from the network to Flexi Soft)

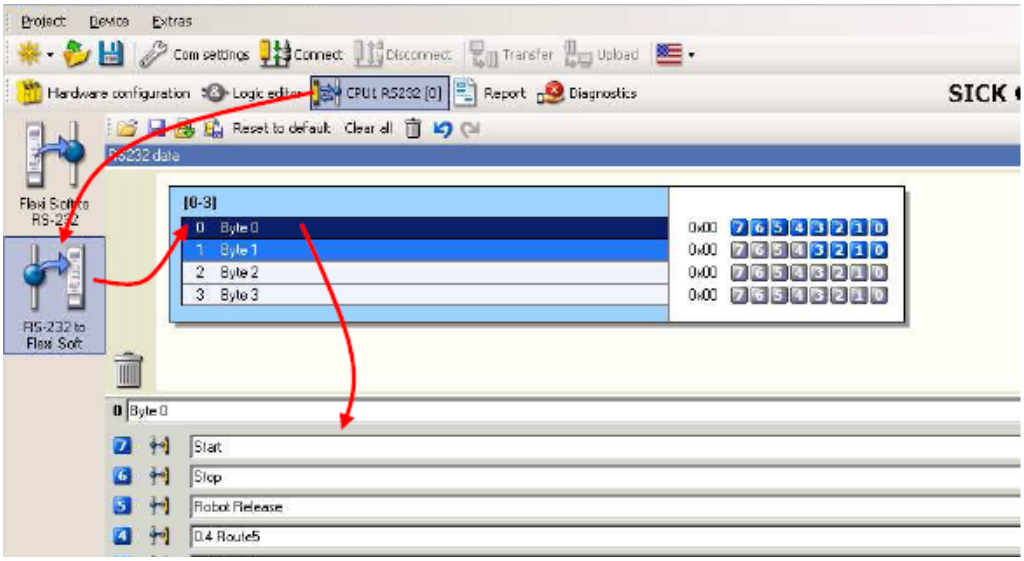
The block is to write 4 bytes process data to the Flexi Soft for use in the logic. If no send access to this block has occurred for 60s the process data are set to safe value (logic 0).

WARNING

The RK512 communication protocol does not meet the requirements for safety related data exchange. Therefore the data exchanged by RK512 must not be used for safety functions!

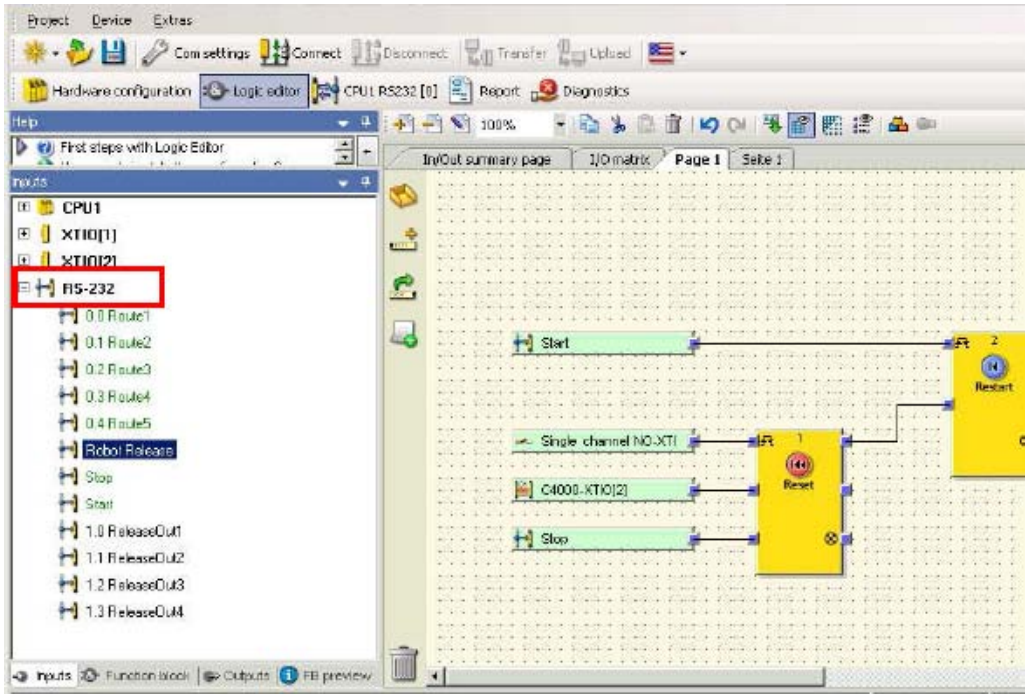
Before the process data of the Visualization Block can be used in the Logic program, the used bits need to be declared in the Flexi Soft Designer in the RS232 routing dialog for network to Flexi Soft:

Configuration of RS232 routing data from network to Flexi Soft



The declared routing data are listed then in the logic editor, in the section **Inputs** in the branch **RS-232**.

Configuration of RS232 routing data from network to CPU in the logic



Data sheet of Visualization Write Block:

Data sheet of Visualization Write Block

Block name	Visualization Write Block
Block number	66 (42 hex)
access level	Operator (no login with password required). Visualization Write Token handling rules have to be obeyed.
Block size [words]	2 (02 hex, → RK512 size = 0006 hex)
Data	Process data according configuration with Flexi Soft Designer for RS232 routing from network to CPU.

RK512-Telegram

The RK512 telegrams applied to write the Visualization Write Block are:

Telegram streams of Visualization Write Block for writing

Send command telegram (from Client)	telegram byte 0 ... 15 in hex: 00 00 41 44 42 00 00 06 FF 4F 42 00 00 06 FF 4F telegram byte 16 ... 19: block data (Process data byte 0 ... 3) telegram byte 20 ... 21: CRC for byte 10 to 19 Example data byte 0 ... 3 = 0 00 00 41 44 42 00 00 06 FF 4F 42 00 00 06 FF 4F 00 00 00 00 A2 CC Example data byte 0 = 1, data byte 1 ... 3 = 0 00 00 41 44 42 00 00 06 FF 4F 42 00 00 06 FF 4F 01 00 00 00 16 BA
Send reply telegram (from Flexi Soft)	telegram byte 0 ... 3 in hex: 00 00 00 00

If required, the Visualization Write Block can also be read to get its current values.
The RK512 telegrams applied to read the Visualization Write Block are as following:

Telegram streams of Visualization Write Block for reading

Fetch command telegram (from client)	telegram byte 0 ... 9: 00 00 45 44 42 00 00 06 FF 4F
Fetch reply telegram (from Flexi Soft)	telegram byte 0 ... 9 in hex: 00 00 00 00 42 00 00 06 FF xx (xx = actual device address) telegram byte 10 ... 13: block data telegram byte 14 ... 15: CRC for byte 4 to 13

NOTE

To avoid that multiple clients use the Visualization Write Data Block at the same time and thereby causing corruption of data, the Flexi Soft offers a specific token handling to allow the clients to manage access to this block. The token is a specific block where every client leaves a mark indicating that there is already a client using the Visualization Write Block. If the token block is already occupied by another client, the write request (RK512 send command) will be replied by an error ' Device token not available' (0x04). It is in the responsibility of the client, to wait until the token is available. The Flexi Soft does not regulate the access to the Visualization Write Block. However, it offers a token handling to manage the access by the clients itself.

The token has also a field of 6 bytes, which can be used to add an ID, which identifies the client. If the ID is unique within the system, it is possible to identify which client has taken the token already.

To return the token the client has to write 0 to the token block. Before the token is returned, the client should check by reading the Visualization Write Token Block and comparing the value of the returned ID with the won ID, whether it actually owned the token.

Data sheet of Visualization Write Token Block

Data sheet of Visualization Write Token Block:

Block name	Visualization Write Token Block
Block number	65 (41 hex)
access level	Operator (no login with password required).
Block size [words]	4 (04 hex, → RK512 size = 0008 hex)
Data	data byte 0 ... 1: 0F 0F (= ID for HMI, PLC, ...) data byte 2 ... 7: Client ID, unique within system

Fetch telegram streams of Visualization Write Token Block

The RK512 telegrams applied for the Visualization Read Token Block are:

Fetch command telegram (from client)	telegram byte 0 ... 9 in hex: 00 00 41 44 41 00 00 08 FF 4F
Fetch reply telegram (from Flexi Soft)	telegram byte 0 ... 9 in hex: 00 00 00 00 41 00 00 08 FF xx (xx = actual device address) telegram byte 10 ... 17: block data (token, client ID) telegram byte 18 ... 19: CRC for byte 4 to 17

Send telegram streams of Visualization Write Token Block

Send command telegram (from Client)	For token request: telegram byte 0 ... 15 in hex: 00 00 41 44 41 00 00 08 FF 4F 41 00 00 08 FF 4F byte 16 ... 17 in hex: 0F 0F (ID for HMI, PLC, ...) telegram byte 18 ... 23: Client ID, unique within the system telegram byte 24 ... 25: CRC for byte 10 to 23 For token release: telegram byte 0 ... 15 in hex: 00 00 41 44 41 00 00 08 FF 4F 41 00 00 08 FF 4F telegram byte 16 ... 23 in hex: 00 00 00 00 00 00 00 00 telegram byte 24 ... 25: CRC for byte 10 to 23: B3 F1
Send reply telegram (from Flexi Soft)	For token acknowledge: telegram byte 0 ... 3 in hex: 00 00 00 00 For token deny: telegram byte 0 ... 3 in hex: 00 00 00 04

2.4.5 Module status bits

This block corresponds with data set 3 of the Flexi Soft gateways.

It offers 60 byte data containing status bits, 32 bit per module, for up to 15 modules (1 main module, max. 12 extension modules, max. 2 gateways). The meaning of the status bits is individual for every module type. For the meaning of the module status bits please refer to the Flexi Soft Gateways operating instructions (8012664), chapter “Error and status information of the modules” (data set 3). Which types of modules currently exist in the system can be found out by the main module type key (see chapter 2.4.6) and Extension modules type key array block (see chapter 2.4.7).

Data sheet of Module Status Bits Block

Data sheet of Module Status Bits Block:

Block name	Module Status Bits Block
Block number	126 (7E hex)
Access level	Operator (no login with password required).
Block size [words]	30 (1E hex, → RK512 size = 0022 hex)
Data	Module status bits of ... data byte 0 ... 3: ... main module. data byte 4 ... 7: ... extension module 1. data byte 8 ... 11: ... extension module 2. ... data byte 48 ... 51: ... extension module 12. data byte 52 ... 55: ... gateway 1. data byte 56 ... 59: ... gateway 2. Each item for every module contain common bits: bit 0: Module operating state (0=stop, 1=Run) bit 1: Internal status (0=internal error, 1=no error) bit 2: External status (0=external error, 1=no error) bit 4: Configuration status (0=invalid, 1=valid/unknown)

Telegram streams of Module Status Bits Block

The RK512 telegrams applied for the Module Status Bits Block are:

Fetch command telegram (from client)	telegram byte 0 ... 9 in hex: 00 00 45 44 7E 00 00 22 FF 4F
Fetch reply telegram (from Flexi Soft)	telegram byte 0 ... 9 in hex: 00 00 00 00 7E 00 00 22 FF xx (xx = actual device address) telegram byte 10 ... 69: block data telegram byte 70 ... 71: CRC for byte 10 to 69

2.4.6 T ype key of main module

This block contains the type keys of the currently connected main module in the system.

Data sheet of Main Module Type Key Block:

Data sheet of Main Module Type Key Array Block

Block name	Main Module Type Key Block
Block number	7 (07 hex)
Access level	Operator (no login with password required).
Block size [words]	9 (09 hex, → RK512 size = 000D hex)
Data	byte 0, bit 0 ... 3: 0x7 = Flexi Soft family byte 0, bit 4 ... 7: Safety level (0=standard, 3=safety) byte 1, bit 0 ... 7: Module type (0x00=CPU0, 0x01=CPU1, 0x02=CPU2, 0x03=CPU3, ...) byte 7, bit 0 ... 7: module diagnosis ID (individual number for every unique meaning of module status bits (0x01=CPU0 / CPU1 / CPU2 / CPU3, ...) byte 14: serial number: year - 2000 (e.g. 10=2010) byte 15: serial number: calendar week number byte 16 ... 17: serial number: consecutive number within week

The RK512 telegrams applied for the Main Module Type Key Array Block is:

Telegram streams of Main Module Type Key Array Block

Fetch command telegram (from client)	telegram byte 0 ... 9 in hex: 00 00 45 44 07 00 00 0D FF 4F
Fetch reply telegram (from Flexi Soft)	telegram byte 0 ... 9 in hex: 00 00 00 00 07 00 00 0D FF xx (xx = actual device address) telegram byte 10 ... 17: block data telegram byte 18 ... 19: CRC for byte 10 to 17

2.4.7 Type key of extension modules

This block offers a list of the type keys of the currently connected modules (extension modules and gateways) in the system.

Data sheet of Extension Modules Type Key Array Block

Data sheet of Extension Modules Type Key Array Block:

Block name	Extension Modules Type Key Array Block
Block number	124 (7C hex)
Access level	Operator (no login with password required).
Block size [words]	168 (A8 hex, → RK512 size = 00AC hex)
Data	Type key of ... byte 0 ... 23: extension module 1 byte 24 ... 47: extension module 2 ... byte 164 ... 287: extension module 12 byte 288 ... 311: gateway 1 byte 312 ... 335: gateway 2 Each type key item contains: byte 0, bit 0 ... 3: 0x7 = Flexi Soft family, 0x0 = No module byte 0, bit 4 ... 7: Safety level (0=standard, 3=safety) byte 1, bit 0 ... 7: Module type (see table further down)) byte 7, bit 0... 7: module diagnosis ID, individual number for every unique meaning of module status bits (see table further down) byte 8: serial number: year - 2000 (e.g. 10 = 2010) byte 9: serial number: calendar week number byte 10 ... 11: serial number: consecutive number within week

Telegram streams of Extension Module Type Key Array Block

The RK512 telegrams applied for the Extension Module Type Key Array Block is:

Fetch command telegram (from client)	telegram byte 0 ... 9 in hex: 00 00 45 44 7C 00 00 AC FF 4F
Fetch reply telegram (from Flexi Soft)	telegram byte 0 ... 9 in hex: 00 00 00 00 7C 00 00 AC FF xx (xx = actual device address) telegram byte 10 ... 177: block data telegram byte 178 ... 179: CRC for byte 10 to 177

Modultyp-Codes und Modudiagnose-IDs

Modultyp	Bezeichnung	Modultyp code	Modul-diagnose ID
CPU0	Main module	0x01	0x01
CPU1	Main module 2 EFI connection	0x02	0x01
CPU2	Main module 2 EFI connection	0x03	0x01
CPU3	Main module 2 EFI connection, 1 Flexi Line connection	0x04	0x01
XTIO	Input/output extension 8 safe inputs/4 safe outputs	0x06	0x02
XTDI	Input/output extension 8 safe inputs	0x04	0x02
XTDS	Input/output extension 8 safe inputs, 4 or 6 non-safe outputs	0x05	0x22
STIO	Input/output extension 6 or 8 non-safe inputs, 8 or 6 non-safe	0x20	0x20
MOC0	Drive monitor Speed	0x24	0x21
MOC1	Drive monitor Speed and position	0x21	0x21
GENT	Ethernet gateway for EthernetNet/IP	0x0A	0x07
GMOD	Ethernet gateway for Modbus TCP	0x0B	0x06
GPNT	Ethernet gateway for PROFINET I/O	0x0C	0x08
GETC	Ethernet gateway for EtherCAT	0x16	0x11
GPRO	PROFIBUS gateway	0x07	0x03
GDEV	DeviceNet gateway	0x08	0x05
GCAN	CANopen gateway	0x09	0x04
GCC1	CC-Link gateway, Mitsubishi	0x14	0x0C
GS3S	Sercos III gateway, Bosch Rexroth	0x15	0x0F

2.4.8 Operating Data Block

The current operating time can be read from the Operating Data Block, e.g. for calculation of absolute occurrence time.

Data sheet of Operating Data Block

Data sheet of Operating Data Block:

Block name	Operating Data Block
Block number	11 (0B hex)
Access level	Operator (no login with password required).
Block size [words]	5 (05 hex, → RK512 size = 0009 hex)
Data	data byte 0 ... 3: operating time (Current time in seconds of total operating life time). data byte 4 ... 7: Power On time (Current time in seconds since last power up). data byte 8 ... 9: Power Cycles (The number of power up cycles in life time).

Telegram streams of Operating Data Block

The RK512 telegrams applied for the Operating Data Block are:

Fetch command telegram (from client)	telegram byte 0 ... 9 in hex: 00 00 45 44 0B 00 00 09 FF 4F
Fetch reply telegram (from Flexi Soft)	telegram byte 0 ... 9 in hex: 00 00 45 44 0B 00 00 09 FF xx (xx = actual device address) telegram byte 10 ... 19: block data telegram byte 20 ... 21: CRC for byte 4 to 19

2.4.9 Configuration CRCs

This block corresponds with data set 2 of the Flexi Soft gateways. For more information please refer to the Flexi Soft Gateway operating instruction (8012664).

The configuration CRCs can be read from the Configuration CRC Block:

- 1 CRC that covers all configuration parts (Overall CRC).
  The overall CRC is the checksum you can see in the Flexi Soft Designer report.
  If ACR is not used, the Overall-CRC and the SCID have the same value.
- 1 CRC that covers the Flexi Soft system (Main module, extension modules, gateways) (SCID).
- 1 CRC for Automatic Configuration Recovery (ACR CRC) This requires main module FX3-CPU2 or FX3-CPU3 firmware V3.00.0 or higher.
- 1 CRC for Configuration Verification Status. If value is equal to SCID, then the configuration is verified. This requires main module firmware V2.00.0 or higher.

Data sheet of Configuration CRC Block:

Data sheet of none Configuration CRC Block

Block name	Configuration CRC Block
Block number	241 (F1 hex)
Access level	Operator (no login with password required).
Block size [words]	10 (0A hex, → RK512 size = 000E hex)
Data	data byte 0 ... 3: Overall CRC. data byte 4 ... 7: SCID (System checksum). data byte 8 ... 11: ACR CRC. data byte 12 ... 15: Configuration Verification Status. data byte 16 ... 19: Reserved = 0.

The RK512 telegrams applied for the Configuration CRC Block are:

Telegram streams of Configuration CRC Block

Fetch command telegram (from client)	telegram byte 0 ... 9 in hex: 00 00 45 44 F1 00 00 0E FF 4F
Fetch reply telegram (from Flexi Soft)	telegram byte 0 ... 9 in hex: 00 00 45 44 F1 00 00 0E FF xx (xx = actual device address) telegram byte 10 ... 29: block data telegram byte 30 ... 31: CRC for byte 10 to 29

2.4.10 Application name

With the Application Name block the application name can be read, which can be defined by the user for each project in the Flexi Soft Designer.
To define the name in the Flexi Soft Designer, disconnect (Offline), make a right click on the main module in the *Hardware configuration area* and select *Edit*. In the section BOM info you find the edit field for the application name.

Data sheet of Application Name Block

Data sheet of Application Name Block:

Block name	Application Name
Block number	23 (17 hex)
Access level	Operator (no login with password required).
Block size [words]	11 (0B hex, → RK512 size = 000F hex)
Data	data byte 0 ... 21: ASCII codes of application name, byte 0 is first character, total 22 characters.

Telegram streams of Application Name Block

The RK512 telegrams applied for the Application Name Block are:

Fetch command telegram (from client)	telegram byte 0 ... 9 in hex: 00 00 45 44 17 00 00 0F FF 4F
Fetch reply telegram (from Flexi Soft)	telegram byte 0 ... 9 in hex: 00 00 45 44 17 00 00 0F FF xx (xx = actual device address) telegram byte 10 ... 31: block data telegram byte 32 ... 33: CRC for byte 10 to 31

2.4.11 Flexi Loop-Status Bits

This block offers 184 Byte Flexi Loop status bits for one Flexi Loop cascade. There are up to 8 Flexi Loop cascades possible for each Flexi Soft station. The relevant cascade is selected by the telegram field „Structure index“.

For further information to the here mentioned status bits please see the Flexi Loop operating instruction in the chapters for diagnostic information.

This requires main module firmware V3.00.0 or higher.

Datasheet of Flexi Loop-Status Bits-Block

Data sheet of Flexi Loop-Status Bits Block:

Block-name	Flexi Loop-Status Bits-Block
Block number	137 (89 hex)
Access level	Operator (no login with password required).
Block size [words]	92 (5C hex, → RK512-Größe = 0060 hex)
Data	Status bits of Flexi Loop cascade Data byte 132, Bit 0: Status safety path Data byte 132, Bit 1: Sensor test Data byte 132, Bit 2: Online Status bits of Loop knodes: Here applies lowest significant bit: node 1 ... highest significant bit: node 32 Data byte 136 ... 139: AUX_IN. Data byte 140 ... 143: AUX_OUT. Data byte 144 ... 147: Status of sensor / switch. Data byte 148 ... 151: Dynamic testing. Data byte 152 ... 155: Static testing. Data byte 156 ... 159: Online. All other data are not documented features within the context of this RK512 telegram listing.

Telegram streams of Flexi Loop Status Bits Block

The RK512 telegrams applied for the Flexi Loop-Status Bits-Block are:

Fetch command telegram (from client)	telegram byte 0 ... 9 in hex: 00 00 45 44 89 ii 00 60 FF 4F (ii = Index for selection of Flexi Loop cascade, 0 : Cascade 1 ... 31: Cascade 32)
Fetch reply telegram (from Flexi Soft)	telegram byte e 0 ... 9 in hex: 00 00 00 00 89 ii 00 60 FF xx (xx = actual device address) telegram byte 10 ... 193: Block data telegram byte 194 ... 195: CRC for Byte 10 to 193

3 Glossary

Flexi Soft Designer	Configuration and diagnosis PC tool for Flexi Soft
EFI	Enhanced function interface. Communication interface to connect to intelligent sensors like C4000, M4000, S3000, S300 to the Flexi Soft main module.
RI	Register Interface: Data application interface of Flexi Soft module

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;
}
```