Manual

Protocols

Send/Receive TCP

Contact

HIMA Paul Hildebrandt GmbH

P.O. Box 1261

68777 Brühl

Phone: +49 6202 709-0

Fax: +49 6202 709-107

E-mail: info@hima.com

Document designation	Description
HI 801 516 D, Rev. 11.00 (1927)	German original document
HI 801 524 E, Rev. 11.00.00 (1934)	English translation of the German original document

Table of Contents

1 Introduction 5

1.1 Structure and Use of This Manual 5

1.2 Target Audience 6

1.3 Writing Conventions 6

1.3.1 Safety Notices 6

1.3.2 Operating Tips 7

1.4 Safety Lifecycle Services 7

2 Safety 8

2.1 Intended Use 8

2.2 Residual Risk 8

2.3 Safety Precautions 8

2.4 Emergency Information 8

2.5 Automation Security for HIMA Systems 8

3 Send/Receive TCP 9

3.1 System Requirements 9

3.1.1 Creating a S/R TCP Protocol 9

3.2 Example: S/R TCP Configuration 10

3.2.1 S/R TCP Configuration of the Siemens Controller SIMATIC 300 12

3.2.2 S/R TCP Configuration of the HIMax Controller 16

3.3 TCP S/R Protocols Menu Functions 18

3.3.1 Edit 18

3.3.2 Properties 18

3.3.3 CPU/COM 19

3.4 Menu Functions for TCP Connection 20

3.4.1 Edit 20

3.4.2 System Variables 20

3.4.3 Properties 21

3.5 Data Exchange 23

3.5.1 TCP Connections 23

3.5.2 Cyclic Data Exchange 24

3.5.3 Acyclic Data Exchange with Function Blocks 24

3.5.4 Simultaneous Cyclic and Acyclic Data Exchange 24

3.5.5 Flow Control 25

3.6 Third-Party Systems with Pad Bytes 25

3.7 S/R TCP Function Blocks 25

3.7.1 TCP_Reset 27

3.7.2 TCP_Send 30

3.7.3 TCP_RECEIVE 33

3.7.4 TCP_ReceiveLine 37

3.7.5 TCP_ReceiveVar 41

3.8 Control Panel (Send/Receive over TCP) 46

3.8.1 View Box for General Parameters 46

3.8.2 View Box for TCP Connections 46

3.8.3 Error Code of the TCP Connection 47

3.8.4 Additional Error Code Table for the Function Blocks 48

3.8.5 Connection State 48
3.8.6 Partner Connection State 48

4 General 49

4.1 Maximum Communication Time Slice 49
4.1.1 Determining the Maximum Duration of the Communication Time Slice 49
4.2 Load Limitation 50
4.3 Configuring the Function Blocks 50
4.3.1 Purchasing Function Block Libraries 50
4.3.2 Configuring the Function Blocks in the User Program 51
4.3.3 Configuring the Function Blocks in the SILworX Structure Tree 52

Appendix 53

Glossary 53
Index of Figures 54
Index of Tables 54

1 Introduction

The Send & Receive TCP manual describes the properties of the Send/Receive TCP protocol and its configuration in SILworX for the safety-related HIMA controller systems.

Knowledge of regulations and the proper technical implementation of the instructions detailed in this manual performed by qualified personnel are prerequisites for planning, engineering, programming, installing, starting up, operating and maintaining the HIMA controllers.

HIMA shall not be held liable for severe personal injuries, damage to property or the environment caused by any of the following: unqualified personnel working on or with the devices, de-activation or bypassing of safety functions, or failure to comply with the instructions detailed in this manual (resulting in faults or impaired safety functionality).

HIMA automation devices have been developed, manufactured and tested in compliance with the pertinent safety standards and regulations. They may only be used for the intended applications under the specified environmental requirements.

1.1 Structure and Use of This Manual

The manual contains the following chapters:

- Introduction
- Safety
- Product description
- Send/Receive TCP

Additionally, the following documents must be taken into account:

Name	Content	Document no.
HIMax system manual	Hardware description HIMax system	HI 801 001 E
HIMax safety manual	Safety function HIMax systems	HI 801 003 E
HIMatrix safety manual	Safety function HIMatrix systems	HI 800 023 E
HIMatrix compact system manual	Hardware description HIMatrix compact system	HI 800 141 E
HIMatrix modular system manual	Hardware description HIMatrix modular F 60 system	HI 800 191 E
HIQuad X system manual	Hardware description HIQuad X system	HI 803 211 E
HIQuad X safety manual	Safety function HIQuad X system	HI 803 209 E
Automation security manual	Description of automation security aspects related to the HIMA systems	HI 801 373 E
SILworX first steps manual	Introduction to SILworX.	HI 801 103 E

Table 1: Additional Applicable Manuals

The current manuals can be obtained upon request by sending an e-mail to: documentation@hima.com. The documentation is available for registered HIMA customers in the download area https://www.hima.com/en/downloads/.

1.2 Target Audience

This document is aimed at the planners, design engineers, programmers and the persons authorized to start up, operate and maintain the automation systems. Specialized knowledge of safety-related automation systems is required.

1.3 Writing Conventions

To ensure improved readability and comprehensibility, the following writing conventions are used in this document:

Bold	To highlight important parts. Names of buttons, menu functions and tabs that can be clicked and used in the programming tool.
Italics	Parameters and system variables, references.
`Courier`	Literal user inputs.
RUN	Operating states are designated by capitals.
Chapter 1.2.3	Cross-references are hyperlinks even if they are not specially marked. In the electronic document (PDF): When the mouse pointer hovers over a hyperlink, it changes its shape. Click the hyperlink to jump to the corresponding position.

Safety notices and operating tips are specially marked.

1.3.1 Safety Notices

Safety notices must be strictly observed to ensure the lowest possible risk.

The safety notices are represented as described below.

- Signal word: warning, caution, notice.
- Type and source of risk.
- Consequences arising from non-observance.
- Risk prevention.

The signal words have the following meanings:

- Warning indicates hazardous situations which, if not avoided, could result in death or serious injury.
- Caution indicates hazardous situation which, if not avoided, could result in minor or moderate injury.
- Notice indicates a hazardous situation which, if not avoided, could result in property damage.

⚠ SIGNAL WORD

Type and source of risk!

Consequences arising from non-observance.

Risk prevention.

NOTICE

Type and source of damage!

Damage prevention.

1.3.2 Operating Tips

Additional information is structured as presented in the following example:

i The text giving additional information is located here.

Useful tips and tricks appear as follows:

TIP The tip text is located here.

1.4 Safety Lifecycle Services

HIMA provides support throughout all the phases of the plant's safety lifecycle, from planning and engineering through commissioning to maintenance of safety and security.

HIMA's technical support experts are available for providing information and answering questions about our products, functional safety and automation security.

To achieve the qualification required by the safety standards, HIMA offers product or customer-specific seminars at HIMA's training center or on site at the customer's premises. The current seminar program for functional safety, automation security and HIMA products can be found on HIMA's website.

Safety Lifecycle Services:	
Onsite+ / On-Site Engineering	In close cooperation with the customer, HIMA performs changes or extensions on site.
Startup+ / Preventive Maintenance	HIMA is responsible for planning and executing preventive maintenance measures. Maintenance actions are carried out in accordance with the manufacturer's specifications and are documented for the customer.
Lifecycle+ / Lifecycle Management	As part of its lifecycle management processes, HIMA analyzes the current status of all installed systems and develops specific recommendations for maintenance, upgrading and migration.
Hotline+ / 24 h Hotline	HIMA's safety engineers are available by telephone around the clock to help solve problems.
Standby+ / 24 h Call-Out Service	Faults that cannot be resolved over the phone are processed by HIMA's specialists within the time frame specified in the contract.
Logistics+/ 24 h Spare Parts Service	HIMA maintains an inventory of necessary spare parts and guarantees quick, long-term availability.

Contact details:	
Safety Lifecycle Services	https://www.hima.com/en/about-hima/contacts-worldwide/
Technical Support	https://www.hima.com/en/products-services/support/
Seminar Program	https://www.hima.com/en/products-services/seminars//

2 Safety

All safety information, notes and instructions specified in this document must be strictly observed. The product may only be used if all guidelines and safety instructions are adhered to.

The product is operated with SELV or PELV. No imminent risk results from the product itself. Use in the Ex zone is only permitted if additional measures are taken.

2.1 Intended Use

To use the HIMA controllers, all pertinent requirements must be met, see the relevant manuals.

2.2 Residual Risk

No imminent risk results from a HIMA system itself.

Residual risk may result from:

- Faults related to engineering.
- Faults in the user program.
- Faults related to the wiring.

2.3 Safety Precautions

Observe all local safety requirements and use the protective equipment required on site.

2.4 Emergency Information

A HIMA system is a part of the safety equipment of an overall system. If the controller fails, the system enters the safe state.

In case of emergency, no action that may prevent the HIMA system from operating safely is permitted.

2.5 Automation Security for HIMA Systems

The objectives of automation security are data confidentiality, integrity and availability. With respect to automation security, targeted attacks are to be expected. In particular, interfaces such as described in this manual, are potential target of cyber attacks.

⚠ WARNING

Physical injury possible due to unauthorized manipulation of the controller!

Protect the controller against unauthorized access!

The user is responsible for implementing the necessary measures in a way suitable for the plant!

Careful planning should identify the measures to implement. The required measures are to be implemented after the risk analysis is completed. Such measures can include:

- Meaningful allocation of user groups.
- Maintained network maps help to ensure that secure networks are permanently separated from public networks and, if required, only a well-defined connection exists (e.g., via a firewall or a DMZ).
- Use of appropriate passwords.

A periodical review of the security measures is recommended, e.g., every year.

For further details, refer to the HIMA automation security manual (HI 801 373 E).

3 Send/Receive TCP

Send/Receive TCP (S/R TCP) is a manufacturer-independent, standard protocol for cyclic and acyclic data exchange and does not use any specific protocols other than TCP/IP.

With the S/R TCP protocol, the HIMA controller supports almost every third-party system as well as PCs with implemented socket interface to TCP/IP (for example, Winsock.dll).

S/R TCP is compatible with the Siemens SEND/RECEIVE interface and ensures communication with Siemens controllers via TCP/IP. Data is exchanged using the S7 function blocks AG_SEND (FC5) and AG_RECV (FC6).

3.1 System Requirements

Equipment and system requirements

Element	Description
Controller	HIMax with COM module HIMatrix
CPU module	The Ethernet interfaces on the processor module may not be used for S/R TCP.
COM module	Ethernet 10/100BaseT One S/R TCP protocol can be configured for each COM module.
Activation	A software activation code is required for activation, refer to the communication manual (HI 801 001 E) for details.

Table 2: Equipment and System Requirements for S/R TCP

Properties of the S/R TCP protocol

Element	Description
Safety-related	No
Data Exchange	Cyclic and acyclic data exchange over TCP/IP.
Function Blocks	The S/R TCP function blocks must be used when acyclically exchanging data.
TCP connections	Up to 32 TCP connections can be configured in one controller, provided that the maximum size of send or received data is not exceeded.
Max. Size of Send Data	Refer to the communication manual (HI 801 101 E). i To determine the maximum amount of reference data, the value for all status variables of the configured TCP connections and TCP/SR function blocks must be subtracted from the value for the maximum amount of send data. The data can be freely allocated among multiple TCP connections.
Max. Size of Receive Data	

Table 3: S/R TCP Properties

3.1.1 Creating a S/R TCP Protocol

To create a new S/R TCP protocol:

1. In the structure tree, open **Configuration**, **Resource**, **Protocols**.
2. Select **New, Send/Receive over TCP** from the context menu to add a new S/R TCP protocol.
3. Right-click the Send/Receive over TCP, click **Properties**, **General**, and then select the **COM Module**.

3.2 Example: S/R TCP Configuration

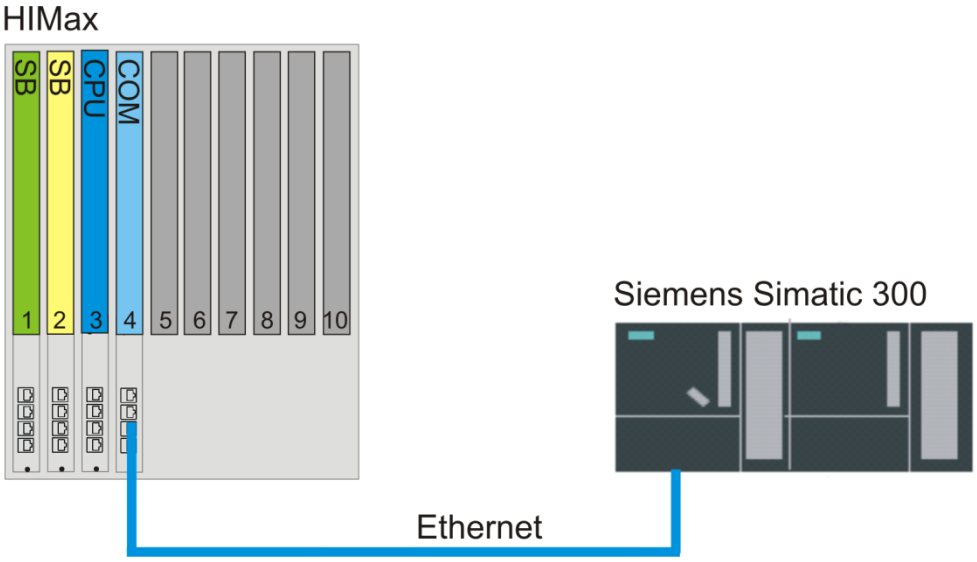


Figure 1: Connecting a HIMax to a Siemens Controller

In this example, the protocol Send/Receive over TCP is installed in a HIMax controller. The HIMax is intended to cyclically communicate with a Siemens controller (e.g., SIMATIC 300) via S/R TCP.

The HIMax (client) is the active station that establishes the TCP connection to the passive Siemens SIMATIC 300 (server). Once the connection has been established, both stations are equal and can send and receive data at any point in time.

When connecting the HIMax to the Siemens SIMATIC 300, take the following points into account:

- The requirements applying to the HIMax are described in Chapter 3.1.
- HIMax and Siemens SIMATIC 300 are connected to one another via Ethernet interfaces.
- HIMax and Siemens SIMATIC 300 must be located in the same subnet or must have the corresponding routing settings if a router is used.

In this example, the HIMax controller is intended to send two BYTES and one WORD to the Siemens SIMATIC 300. The variables are received in the Siemens SIMATIC 300 by the function block AG_RECV (FC 6) and are internally transmitted to the function block AG_SEND (FC 5). Siemens SIMATIC 300 sends the variables (unchanged) back to the controller using the function block AG_SEND (FC 5).HIMax

Once the configuration is completed, proper transmission of variables can be verified using the HIMA Force Editor.

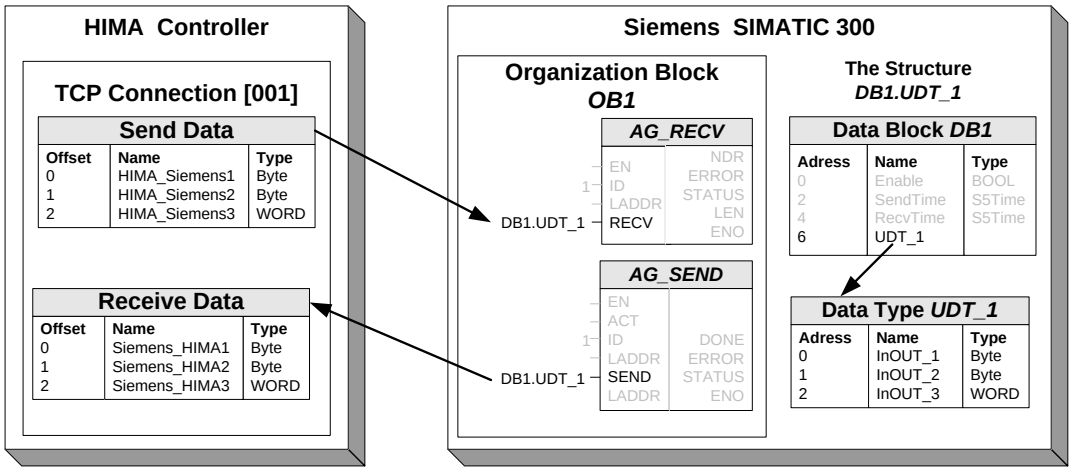


Figure 2: Data Transfer between a HIMax and a Siemens Controller

Description of the HIMax controller configuration

Element	Description
TCP Connection [001]	This dialog box contains all the parameters required for communicating with the communication partner (Siemens SIMATIC 300).
Send Data	The variable offsets and types in the controller must be identical with the variable addresses and types with data type *UDT_1* in the SIMATIC 300.
Receive Data	The variable offsets and types in the HIMax controller must be identical with the variable addresses and types with data type *UDT_1* in the SIMATIC 300.

Table 4: HIMax Controller Configuration

Description of the Siemens SIMATIC 300 Configuration

Element	Description
Organization Block *OB1*	Function blocks *AG_RECV (FC6)* and *AG_SEND (FC 5)* must be created and configured in the *OB1* organization block.
AG_RECV (FC 6)	The function block *AG_RECV (FC 6)* accepts the data received from the communication partner into data type *DB1.UDT_1*. Inputs *ID* and *LADDR* must be appropriately configured for communication with the communication partner.
AG_SEND (FC 5)	Function block *AG_SEND (FC 5)* transfers the data from data type *DB1.UDT_1* to the communication partner. Inputs *ID* and *LADDR* must be appropriately configured for communication with the communication partner.
Data Block *DB1*	Data type *UDT_1* is defined in the data block *DB1*.
Data Type *UDT_1*	The addresses and types of the variables in SIMATIC 300 must be identical with the offsets and types of the controller. Data type *UDT_1* accepts the received user data and stores them until they are transmitted to the communication partner.

Table 5: Siemens SIMATIC 300 Configuration

3.2.1 S/R TCP Configuration of the Siemens Controller SIMATIC 300

i The following step by step instructions for configuring the Siemens controller are not to be considered exhaustive.

No responsibility is taken for the correctness of this information; refer to the Siemens documentation when developing projects with Siemens controllers.

To create the S/R TCP server in the SIMATIC 300 project

1. Start the SIMATIC manager.
2. In the SIMATIC manager, open the project associated with the SIMATIC 300 controller.
3. In this project, create and configure the *Industrial Ethernet* and the *MPI* connections.

To create the UDT1 data type with the following variables

1. Open the *Blocks* folder in the Siemens SIMATIC manager.
2. Select **Add**, **S7 Block**, **Data Type** from the main menu and create a data type.
3. Name the data type **UDT1**.
4. Enter the symbolic name **UDT_1** for the data type.
5. In the *UDT_1* data type, create the three variables **InOut_x** as shown in the figure below.

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	InOut_1	BYTE	B#16#0	
+1.0	InOut_2	BYTE	B#16#0	
+2.0	InOut_3	WORD	W#16#0	
=4.0		END_STRUCT		

Figure 3: List of Variables in the Siemens UDT1 Block

i During cyclic and acyclic data exchange, note that some controllers (e.g., SIMATIC 300) add so-called *pad bytes*. Pad bytes ensure that all data types exceeding 1 byte in size always begin at an even offset and that the total size of all defined variables is also always even.

In these cases, dummy bytes must be added in the HIMax controller at the corresponding positions, refer to Chapter 3.6 for details.

To create the DB1 data block for function blocks FC 5 and FC 6

1. Select **Add**, **S7 Block**, **Data Block** from the main menu and create a data block.
2. Name the data block **DB1**.
3. Enter the symbolic name **DB1** for the data block.
4. Assign the *UDT_1* data type to the *DB1* data block.
5. In the *DB1* data block, configure the data types such as described in the figure below.

Address	Name	Type	Initial val	Comment
0.0		STRUCT		
+0.0	Enable	BOOL	TRUE	
+2.0	SendTime	S5TIME	S5T#100MS	
+4.0	RecvTime	S5TIME	S5T#10MS	
+6.0	UDT_1	"UDT_1"		
=10.0		END_STRUCT		

Figure 4: List of Variables in the Siemens DB1 Function Block

To create the following symbols in the Symbol Editor

1. Double-click the **OB1** organization block to open the *KOP/AWL/FUP* dialog box.
2. Select **Extras**, **Symbol Table** from the main menu to open the Symbol Editor.
3. Extend the *Symbol Editor* by adding the variables **M 1.0…MW 5** such as depicted in the figure below.

S7-Programm(1) (Symbole) -- S7_COMTEST\SIMATIC 300-

	Status	Symbol	Addres	Data type
1		DB1	DB 1	DB 1
2		RecDone	M 1.0	BOOL
3		RecError	M 1.1	BOOL
4		SendDone	M 1.2	BOOL
5		SendError	M 1.3	BOOL
6		RecStatus	MW 1	WORD
7		RecLen	MW 3	INT
8		SendStatus	MW 5	WORD
9		Cycle Execution	OB 1	OB 1
10		UDT_1	UDT 1	UDT 1
11				

Figure 5: SIMATIC Symbol Editor

To create the FC function block AG_RECV (FC 6)

1. Open the *KOP/AWL/FUP* dialog box.
2. From the structure tree located on the left side of the Symatic Manager, select the following function blocks in the given order:
   1 *OR gate*
   1 *S_VIMP*
   1 *AG_RECV (FC 6).*
3. Drag these function blocks onto the *OB1* organization block.
4. Connect and configure the function blocks as described in the figure below.
5. Right-click the **AG_RECV (FC 6)** function block, and then select **Properties**.
6. Deactivate **Active Connection Setup** and configure the ports.
7. Note down the *LADDR* function block parameter and enter it in the function chart on the *AG_RECV (FC 6)* function block.

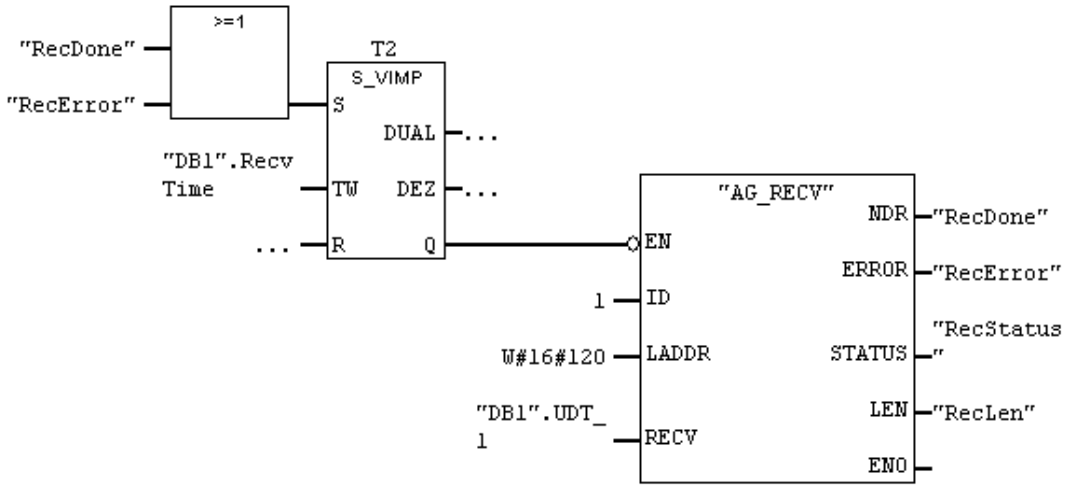


Figure 6: Receive Function Chart

To create the AG_SEND (FC 5) function block

1. Open the *KOP/AWL/FUP* dialog box.
2. From the structure tree located on the left side of the Symatic Manager, select the following function blocks in the given order:
   1 *OR gate*
   1 *S_VIMP*
   1 *AG_SEND (FC 5)*
3. Drag these function blocks onto the *OB1* organization block.
4. Connect and configure the function blocks as described in the figure below.
5. Right-click the **AG_SEND (FC 5)** function block, and select **Properties**.
6. Deactivate **Active Connection Setup** and configure the ports.
7. Note down the *LADDR* function block parameter and enter it in the function chart on the *AG_SEND (FC 5)* function block.

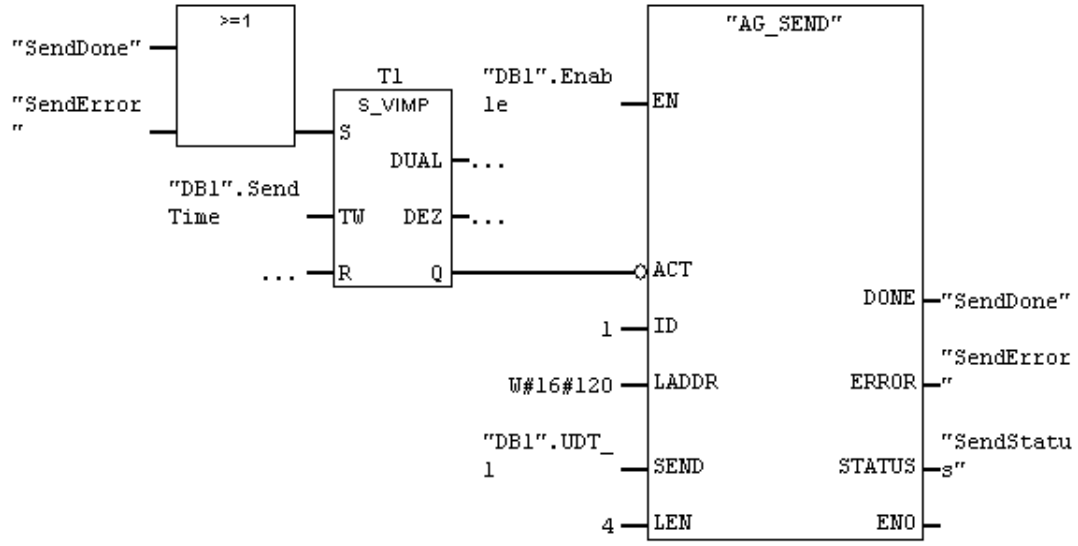


Figure 7: Send Function Chart

To load the code into the SIMATIC 300 controller

1. Start the **Code Generator** for the program.
2. Make sure that the code was generated without error.
3. To load the code into the SIMATIC 300 controller.

3.2.2 S/R TCP Configuration of the HIMax Controller

For further details on how to configure the HIMax controllers and use the SILworX programming tool, refer to the SILworX first steps manual (HI 801 103 E).

To create the following global variables in the Variable Editor

1. In the structure tree, select **Configuration, Global Variables.**
2. Right-click **Global Variables**, and select **Edit**.
3. Create the global variables as described in Table 6.

Name	Type
Siemens_HIMA1	Byte
Siemens_HIMA2	Byte
Siemens_HIMA3	WORD
HIMA_Siemens1	Byte
HIMA_Siemens2	Byte
HIMA_Siemens3	WORD

Table 6: Global Variables

To create the S/R TCP protocol in the resource

1. In the structure tree, open **Configuration, Resource**.
2. Right-click **Protocols** and select **New** from the context menu.
3. Select **Send/Receive over TCP** and enter a name for the protocol.
4. Click OK to confirm and create a new protocol.
5. Right-click **Send/Receive over TCP** and select **Properties** from the context menu.
6. Select **COM Module**. The remaining parameters retain the default values.

To create the TCP connection

1. Right-click **Send/Receive over TCP**, and select **New** from the context menu.
2. Right-click **TCP Connection**, and select **Properties** from the context menu.
3. Configure the properties as indicated in the figure.

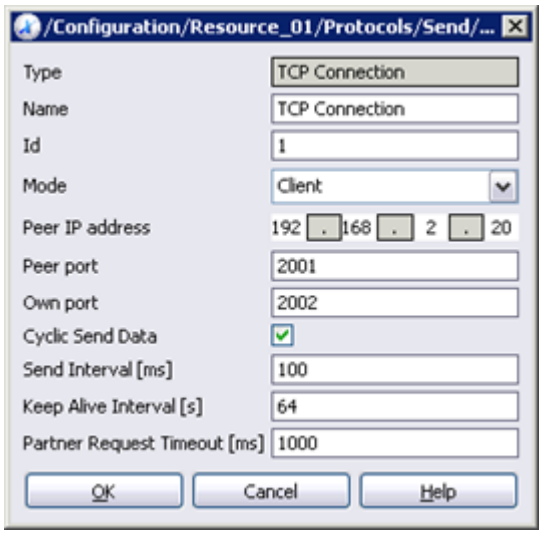


Figure 8: TCP Connection Properties in SILworX

i If a cyclic data exchange should be set up between two controllers, the option *Cyclic Data Transfer* must be activated in the *Properties* dialog box for the TCP connection.

To configure the receive data of the HIMax controller

1. Right-click **TCP Connection** and select **Edit** from the context menu.
2. Select the **Process Variables** tab.
3. Select the following global variables from the **Object Panel** and drag them onto the **Input Signals** area.

Global variable	Type
Siemens_HIMA1	Byte
Siemens_HIMA2	Byte
Siemens_HIMA3	WORD

Table 7: Variables for Receive Data

4. Right-click a free space in the **Register Inputs** area to open the context menu, and select **New Offsets** to renumber the variable offsets.

i Note that the variable offsets in the HIMax controller must be identical with the variable addresses in the *UDT_1* data type within the SIMATIC 300.

To configure the send data of the HIMax controller

1. Right-click **TCP Connection** and select **Edit** from the context menu.
2. Select the **Process Variables** tab.
3. Select the following global variables from the **Object Panel** and drag them onto the **Input Signals** area.

Global variable	Type
HIMA_Siemens1	Byte
HIMA_Siemens2	Byte
HIMA_Siemens3	WORD

Table 8: Variables for Send Data

4. Right-click a free space in the **Register Inputs** area to open the context menu, and select **New Offsets** to renumber the variable offsets.

i Note that the variable offsets in the HIMax controller must be identical with the variable addresses in the *UDT_1* data type within the SIMATIC 300.

To verify the S/R TCP configuration

1. In the structure tree, open **Configuration**, **Resource**, **Protocols**, **Send/Receive Protocol over TCP**.
2. Right-click and select **Verification** from the context menu.
3. Thoroughly verify the messages displayed in the logbook and correct potential errors.

3.3 TCP S/R Protocols Menu Functions

3.3.1 Edit

The **Edit** dialog box for the S/R TCP protocol contains the **System Variables** tab

System variables are used to evaluate the state of the S/R TCP protocol from within the user program.

Element	Description
Active Connection Count	System variable specifying the number of active (not disturbed) connections.
Counter for disturbed connections	System variable specifying the number of disturbed connections. Disturbed means that the TCP connection was interrupted due to a timeout or a fault.
Status	No function.

Table 9: S/R TCP System Variables

3.3.2 Properties

Over a TCP connection, data is exchanged cyclically or acyclically. S/R TCP function blocks are required for acyclic data exchange.

On a connection, data cannot be simultaneously exchanged cyclically and acyclically.

3.3.2.1 General

Name	Description
Type	Send/Receive over TCP.
Name	Name for the current Send/Receive over TCP Protocol.
Module	Selection of the COM module within which the protocol is processed.
Activate Max. µP Budget	Activated: Use the µP budget limit from the *Max. µP Budget in [%]* field. Deactivated: Do not use the µP budget limit for this protocol.
Max. µP Budget in [%]	Maximum µP budget of the module that can be used for processing the protocols. Range of values: 1…100% Default value: 30%
Behavior on CPU/COM Connection Loss	If the connection of the processor module to the communication module is lost, the input variables are either initialized or are still used unchanged in the process module, depending on this parameter. For instance, if the communication module is removed when communication is running. **If a project created with a SILworX prior to V3 should be converted, this value must be set to Retain Last Value to ensure that the CRC does not change.** **For HIMatrix controllers with CPU OS prior to V8, this value must always be set to Retain Last Value.** Adopt Initial Data — Input variables are reset to their initial values. Retain Last Value — The input variables retain the last value.

Table 10: General Properties of the S/R TCP

3.3.3 CPU/COM

The specified parameters provide the fastest possible exchange of S/R TCP data between the COM module (COM) and the processor module (CPU) within the controller. These parameters should only be changed if it is necessary to reduce the COM and CPU loads for an application, and the process allows this change.

i Only experienced programmers should modify the parameters. Increasing the COM and CPU refresh rate means that the effective refresh rate of the S/R TCP data is also increased. The system time requirements must be verified.

Name	Description
Process Data Refresh Rate [ms]	Refresh rate in milliseconds at which the COM and CPU exchange S/R TCP protocol data. If the Refresh Rate is zero or less than the cycle time for the controller, data is exchanged as fast as possible. Range of values: 1…(2^{31}-1) Default value: 0.
Force Process Data Consistency	Activated: Transfer of the S/R TCP data from the CPU to the COM within a CPU cycle. Deactivated: Transfer of the S/R TCP data (a maximum of 1100 bytes per data direction) from the CPU to the COM distributed over multiple CPU cycles.

Table 11: COM/CPU Parameters

3.4 Menu Functions for TCP Connection

3.4.1 Edit

The **Edit** menu function is used to open the tabs **Process Variables** and **System Variables**.

3.4.1.1 Process Variables

Input Signals

The variables for cyclic data exchange received by this controller are entered in the *Input Signals* area.

Any variables can be created in the *Input Signals* area. The variables' offsets and types must be identical with those of the communication partner (send data).

Output Signals

The variables for cyclic data exchange sent by this controller are entered in the *Output Signals* area.

Any variables can be created in the *Output Signals* tab. The variables' offsets and types must be identical with those of the communication partner (receive data).

3.4.2 System Variables

The variables in the **System Variables** tab can be used to evaluate the state of the TCP connection from within the user program.

Name	Data type	R/W	Description
Bytes Received	UDINT	W	Number of bytes received so far.
Bytes Sent	UDINT	W	Number of bytes sent so far.
Error Code	UDINT	W	Error code of the TCP connection, see Chapter 3.8.3.
Error Code Timestamp [ms]	UDINT	W	Millisecond fraction of the timestamp. Point in time when the error occurred.
Error Code Timestamp [s]	UDINT	W	Second fraction of the timestamp. Point in time when the error occurred.
Partner Request Timeout	UDINT	W	For cyclic data transfer: Timeout within which the communication partner must receive at least one time data after data sending.acyclic data transfer: Timeout within which the communication partner must receive at least one time data after data sending. 0 = Off 1…(2^{31}-1) [ms]
Partner Connection State	BYTE	W	If no data is received within the timeout, Partner Connection State is set to *Not Connected* and the connection is restarted. 0 = No connection. 1 = Connection OK.
Status	DWORD	W	TCP connection status, see Chapter 3.8.5.

Table 12: System Variables

3.4.3 Properties

Over a TCP connection, data is exchanged cyclically or acyclically. S/R TCP function blocks are required for acyclic data exchange. The S/R TCP function blocks cannot be used for cyclic data exchange.

Name	Description
Type	TCP connection.
Name	Any unique name for one TCP connection.
ID	Any unique identification number (ID) for each TCP connection. The ID is also required as a reference in the S/R TCP function blocks. Range of values: 0…255 Default value: 0
Mode	Server (default value): This station operates as a server (passive mode). The connection is established by the communication partner (client). Once the connection has been established, both communication partners are equal and can send data at any time. The own port must be specified.
	Server with defined partner: This station operates as a server (passive mode). The connection is established by the communication partner (client). Once the connection has been established, both communication partners are equal and can send data at any time. If the IP address and/or port of the communication partner are defined here, only the specified communication partner can connect to the server. All other stations are ignored. If one of the parameters (IP address or port) is set to zero, the parameter is not verified.
	Client: This station operates as a client, i.e., the station initializes the connection to the communication partner. IP address and port of the communication partner must be specified. Also an own port can optionally be defined.
Partner IP Address	IP address of the communication partner. 0.0.0.0: any IP address is permitted. Valid range: 1.0.0.0…223.255.255.255 Except for: 127.x.x.x Default value: 0
Partner Port	Port of the communication partner. 0: Any port. Ports that are reserved or already used (1…1024), are rejected by the COM operating system. Range of values: 0…65535 Default value: 0
Own Port	Own port. 0: Any port. Ports that are reserved or already used (1...1024), are rejected by the COM operating system. Range of values: 0…65535 Default value: 0

Name	Description
Cyclic Data Transfer	Deactivated (default value): Cyclic data transfer is deactivated. Function blocks must be used to program the data exchange via this TCP connection. No cyclic E/A data may be defined on this connection.
	Activated: Cyclic data transfer is active. Data is defined in the Process Variable dialog box for the TCP connection. Receive data must be defined. No function blocks can be used on this connection.
Send Interval [ms]	Only editable with cyclic data transfer. The send interval can be set here. Range of values 10…2 147 483 647 ms (lower values are rounded to 10 ms). Default value: 0
Keep Alive Interval [s]	Time period until the connection monitoring provided by the TCP is activated. Zero deactivates the connection monitoring. If no data is exchanged within the specified KeepAlive interval, the KeepAlive samples are sent to the communication partner. If the connection still exists, the KeepAlive samples are acknowledged by the communication partner. If no data is exchanged between the partners within a period of > 10 KeepAlive interval, the connection is closed. If no response is received after a data packet transmission, the data packet is resent in predefined intervals. The connection is aborted after 12 unsuccessful resends (approx. 7 min). Ranger 1…65535 s Default value: 0 = deactivated
Partner Request Timeout [ms]	For cyclic data transfer: Timeout within which the communication partner must receive at least one time data after data sending.acyclic data transfer: Timeout within which the communication partner must receive at least one time data after data sending. If no data is received within the timeout, Partner Connection State is set to *Not Connected* and the connection is restarted. After a the connection was closed due to a timeout or another error, the active side re-establishes the connection with a delay of 10 x *Partner Request Timeout* or a delay of 10 s if *Partner Request Timeout* is equal to 0. The passive side already opens the port within half of this time. 0 = Off Range of values: 1…(2^{31}-1) [ms] Default value: 0

Table 13: S/R TCP Connection Properties

3.5 Data Exchange

S/R TCP operates according to the client/server principle. The connection must be initialized by the communication partner, which is configured as a client. After the connection has been established, both communication partners are equal and can send data at any point in time.

S/R TCP has not its own data protection protocol, but uses TCP/IP directly. Because TCP sends data in a data stream, ensure that the offsets and types of the variables to be exchanged are identical on both the receiving and sending sides.

S/R TCP is compatible with the Siemens SEND/RECEIVE interface and allows cyclical data exchange with the Siemens S7 function blocks AG_SEND (FC5) and AG_RECV (FC6), see the example of S/R TCP configuration in Chapter 3.2).

Additionally, HIMA provides 5 S/R TCP function blocks for controlling and customizing communication using the user program. The S/R TCP function blocks allow any arbitrary protocols (e.g., Modbus) that are transferred over TCP to be sent and received.

3.5.1 TCP Connections

For each connection over S/R TCP to a communication partner, at least one TCP connection must exist in the HIMax controller.

The identification number of the TCP connection, the addresses/ports of the own controller and those of the communication partner must be entered in the *Properties* of the TCP connection.

A maximum of 32 TCP connections can be established in a HIMax controller.

These TCP connections must have different identification numbers and different addresses/ports.

To create the TCP connection

1. In the structure tree, open **Configuration, Resource**.
2. Right-click **Protocols** and select **New** from the context menu.
3. Select **Send/Receive over TCP** and enter a name for the protocol.
4. Click OK to confirm and create a new protocol.
5. Right-click **Send/Receive over TCP** and select **Properties** from the context menu.
6. Select **COM Module**. The remaining parameters retain the default values.

TIP The HIMA controller and the third-party system must be located in the same subnet or must have the corresponding routing settings if a router is used.

3.5.2 Cyclic Data Exchange

If data is exchanged cyclically, a send interval must be defined in the HIMA controller and in the communication partner.

The send interval defines the cyclic time period within which the sending communication partner sends the variables to the receiving communication partner.

- To ensure a continuous data exchange, both communication partners should define almost the same send interval (see Chapter 3.5.5).
- For cyclic data exchange, the *Cyclic Data Transfer* option must be activated in the TCP connection used.
- If the *Cyclic Data Transfer* option is activated in a TCP connection, no function blocks may be used.
- The variables to be sent and received are assigned in the *Process Variable* dialog box for the TCP connection. Receive data **must** be available, send data is optional.

i The same variables (same offsets and types) that are defined as send data in a station, must be defined as receive data in the other station.

3.5.3 Acyclic Data Exchange with Function Blocks

In HIMA controllers, the acyclic data exchange is controlled by the user program via the S/R TCP function blocks.

Data exchange can thus be controlled using a timer or a mechanical switch connected to a physical input on the HIMA controller.

- The *Cyclic Data Transfer* option must be deactivated in the TCP connection used.
- Only one S/R TCP function block may send at any given time.
- The variables to be sent or received are assigned in the *Process Variables* dialog box for the S/R TCP function blocks (all except for *Reset*).

i The same variables (same offsets and types) that are defined as send data in a station, must be defined as receive data in the other station.

3.5.4 Simultaneous Cyclic and Acyclic Data Exchange

For this purpose, one TCP connection must be configured for cyclic data and one TCP connection for acyclic data. The two TCP connections must use different *partner IP addresses* and *partner ports*.

A single TCP connection cannot be simultaneously used for cyclic and acyclic data exchange.

3.5.5 Flow Control

The flow control is a component of the TCP and monitors the continuous data traffic between two communication partners.

With cyclic data transmission, at least one packet must be received after a maximum of 3 to 5 packets have been sent; otherwise, transmission is blocked until a packet is received or the connection monitoring process terminates the connection.

The number of packets that may be sent (3…5) before a packet has been received depends on the size of the packets to be sent.

Number = 5 for small packets < 4 kB.

Number = 3 for large packets ≥ 4 kB.

- While planning the project, it must be ensured that no station sends more data than the other station can simultaneously process.
- To ensure cyclical data exchange, both communication partners should define almost the same send interval

3.6 Third-Party Systems with Pad Bytes

During cyclic and acyclic data exchange, note that some controllers (e.g., SIMATIC 300) add so-called *pad bytes*. Pad bytes ensure that all data types exceeding one byte always begin at an even offset and that the total size of the packets (in bytes) is also even.

In the HIMA controller, *dummy bytes* must be added in place of *pad bytes* at the corresponding positions.

Address	Name	Type	Initial value
0.0		STRUCT	
+0.0	InOut_1	BYTE	B#16#0
+2.0	InOut_3	WORD	W#16#0
=4.0		END_STRUCT	

Figure 9: Siemens List of Variables

In the Siemens controller, a *pad byte* is added (not visible) such that the *InOut_3* variable begins at an even offset.

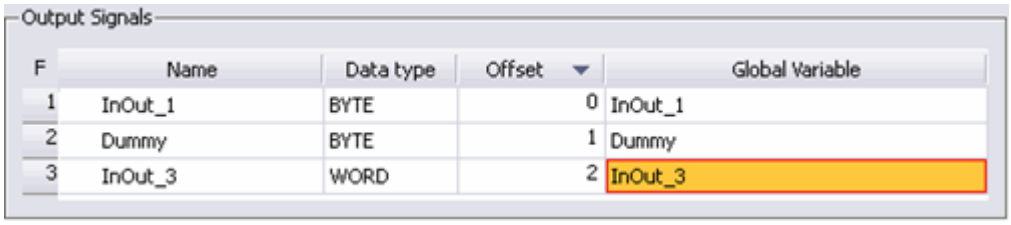


Figure 10: HIMA List of Variables

In the HIMA controller, a *dummy byte* must be added such that the *InOut_3* variable has the same offset as in the Siemens controller.

3.7 S/R TCP Function Blocks

If the cyclic data transmission is not sufficiently flexible, data can also be sent and received using the S/R TCP function blocks. The *Cyclic Data Transfer* option must be deactivated in the TCP connection in use.

The S/R TCP function blocks is used to tailor the data transmission over TCP/IP to best meet the project requirements.

The function blocks are configured in the user program. The functions (Send, Receive, Reset) of the HIMA controller can thus be set and evaluated in the user program.

S/R TCP function blocks are only required for acyclic data exchange. These function blocks are not required for the cyclic data exchange between server and client!

i The configuration of the S/R TCP function blocks is described in Chapter 4.3.

The following function blocks are available:

Function Block	Function description
TCP_RESET (see Chapter 3.7.1)	TCP connection reset.
TCP_Send (see Chapter 3.7.2)	Sending of data.
TCP_RECEIVE (see Chapter 3.7.3)	Reception of data packets with fixed length.
TCP_ReceiveLine (see Chapter 3.7.4)	Reception of an ASCII line.
TCP_ReceiveVar (see Chapter 3.7.5)	Reception of data packets with variable length (with length field).
LATCH	Only used within other function blocks.
PIG	Only used within other function blocks.
PIGII	Only used within other function blocks.

Table 14: Function Blocks for S/R TCP Connections

3.7.1 TCP_Reset

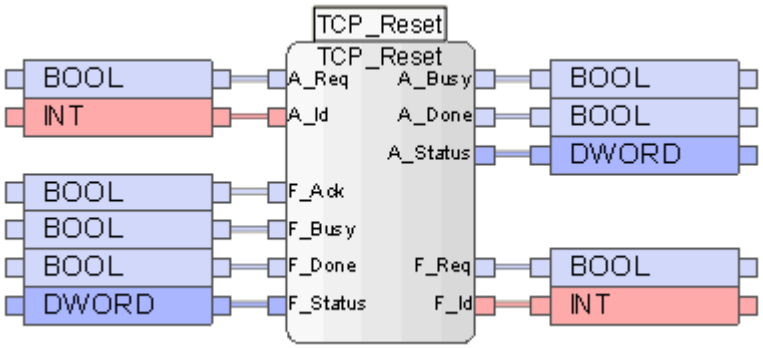


Figure 11: TCP_Reset Function Block

The **TCP Reset** function block is used to re-establish a disturbed connection if a send or receive function block reports a timeout error (16#8A).

i To configure the function block, drag it from the function block library onto the user program (see also chapter 4.3).

Inputs and Outputs of the Function Block with Prefix A

These inputs and outputs can be used to control and evaluate the function block using the user program. The prefix A means Application.

A_Inputs	Description	Type
A_Req	The rising edge starts the function block.	BOOL
A_Id	Identification number *ID* of the disturbed TCP connection to be reset.	INT

Table 15: A-Inputs for the TCP_Reset Function Block

A_Outputs	Description	Type
A_Busy	TRUE: The TCP connection is still being reset.	BOOL
A_Done	TRUE: The sending process was completed without errors.	BOOL
A_Status	The status and error code of the function block and of the TCP connection are output to *A_Status*.	DWORD

Table 16: A-Outputs for the TCP_Reset Function Block

Inputs and Outputs of the Function Block with Prefix F:

These inputs and outputs of the function block establish the connection to the **Reset** function block in structure tree. The prefix F means Field.

i Common variables are used to connect the **Reset** function block (in the Blocks structure tree folder) to the **TCP_Reset** function block (in the user program). These must have been previously created in the Global Variable Editor.

Connect the *F-Inputs* of the **TCP_Reset** function block in the user program to the same variables that will be connected to the outputs of the **Reset** function block in the structure tree.

F-Inputs	Type
F_Ack	BOOL
F_Busy	BOOL
F_Done	BOOL
F_Status	DWORD

Table 17: F-Inputs for the TCP_Reset Function Block

Connect the *F-Outputs* of the **TCP_Reset** function block in the user program to the same variables that will be connected to the outputs of the **Reset** function block in the structure tree.

F-Outputs	Type
F_Req	BOOL
F_ID	DWORD

Table 18: F-Outputs for the TCP_Reset Function Block

To create the Reset function block in the structure tree

1. In the structure tree, open **Configuration**, **Resource**, **Protocols**, **Send/Receive over TCP**, **Blocks, New**.
2. Select the **Reset** function block.
3. Right-click the **Reset** function block and select **Edit**.
   ☑ The window for assigning variables to the function blocks appears.

Connect the inputs of the **Reset** function block in the structure tree to the same variables that have been previously connected to the *F-Outputs* of the **TCP_Reset** function block in the user program.

Inputs	Type
ID	DWORD
REQ	BOOL

Table 19: Input System Variables

Connect the outputs of the **Reset** function block in the structure tree to the same variables that have been previously connected to the *F-Inputs* of the **TCP_Reset** function block in the user program.

Outputs	Type
ACK	BOOL
Busy	BOOL
DONE	BOOL
STATUS	DWORD

Table 20: Output System Variables

The following steps are essential to operate the TCP_Reset function block:

1. In the user program, set the identification number for the faulty TCP connection at the *A_Id* input.
2. In the user program, set the *A_Req* input to TRUE.

i The function block responds to a rising edge at *A_Req*.

The *A_Busy* output is set to TRUE until a reset is sent to the specified TCP connection. Afterwards, *A_Busy* is set to FALSE and *A_Done* is set to TRUE.

3.7.2 TCP_Send

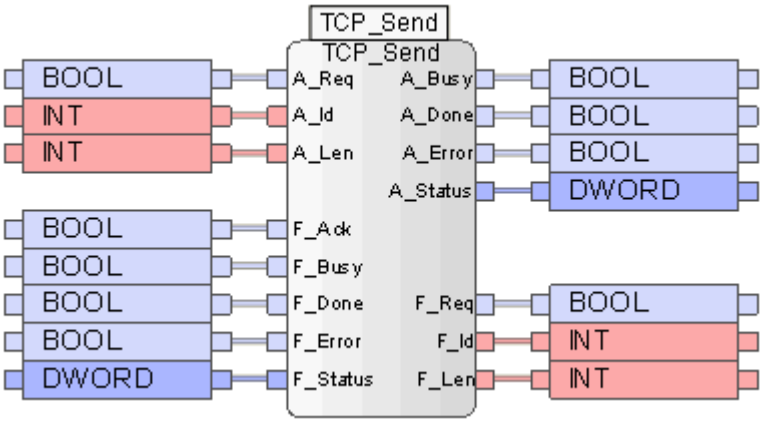


Figure 12: TCP_Send Function Block

The **TCP_Send** function block is used for acyclic transmission of variables to a communication partner. A function block with the same variables and offsets, e.g., *Receive*, must be configured in the communication partner.

i To configure the function block, drag it from the function block library onto the user program (see also chapter 4.3).

Inputs and Outputs of the Function Block with Prefix A

These inputs and outputs can be used to control and evaluate the function block using the user program. The prefix A means Application.

A_Inputs	Description	Type
A_Req	The rising edge starts the function block.	BOOL
A_Id	Identification number of the configured TCP connection to the communication partner to which data should be sent.	INT
A_Len	Number of transmitted variables, expressed in bytes. A_Len must be greater than zero and must not end within a variable.	INT

Table 21: A-Inputs for the TCP_Send Function Block

A_Outputs	Description	Type
A_Busy	TRUE: Data is still being transmitted.	BOOL
A_Done	TRUE: The sending process was completed without errors.	BOOL
A_Error	TRUE: An error occurred FALSE: No error.	BOOL
A_Status	The status and error code of the function block and of the TCP connection are output to *A_Status*.	DWORD

Table 22: A-Outputs for the TCP_Send Function Block

Inputs and Outputs of the Function Block with Prefix F:

These inputs and outputs of the function block establish the connection to the **Send** function block in structure tree. The prefix F means Field.

i Common variables are used to connect the **Send** function block (in the Blocks structure tree folder) to the **TCP_Send** function block (in the user program). These must have been previously created in the Global Variable Editor.

Connect the *F-Inputs* of the **TCP_Send** function block in the user program to the same variables that will be connected to the outputs of the **Send** function block in the structure tree.

F-Inputs	Type
F_Ack	BOOL
F_Busy	BOOL
F_Done	BOOL
F_Error	BOOL
F_Status	DWORD

Table 23: F-Inputs for the TCP_Send Function Block

Connect the *F-Outputs* of the **TCP_Send** function block in the user program to the same variables that will be connected to the inputs of the **Send** function block in the structure tree.

F-Outputs	Type
F_ID	DWORD
F_Len	INT
F_Req	BOOL

Table 24: F-Outputs for the TCP_Send Function Block

To create the Send function block in the structure tree

1. In the structure tree, open **Configuration**, **Resource**, **Protocols**, **Send/Receive over TCP**, **Blocks, New**.
2. Select the **Send** function block.
3. Right-click the **Send** function block and select **Edit**.
   ☑ The window for assigning variables to the function blocks appears.

Connect the outputs of the **Send** function block in the structure tree to the same variables that have been previously connected to the *F-Outputs* of the **TCP_Send** function block in the user program.

Inputs	Type
ID	DWORD
LEN	INT
REQ	BOOL

Table 25: Input System Variables

Connect the outputs of the **Send** function block in the structure tree to the same variables that have been previously connected to the *F-Inputs* of the **TCP_Send** function block in the user program.

Outputs	Type
Ack	BOOL
Busy	BOOL
Done	BOOL
Error	BOOL
STATUS	DWORD

Table 26: Output System Variables

Data	Description
Send data	Any variables can be created in the *Process Variables* tab. Offsets and types of the variables must be identical with offsets and types of the variables of the communication partner.

Table 27: Send Data

The following steps are essential to operate the TCP_Send function block:

i The send variables must be created in the *Process Variables* tab of the *Send* dialog box. Offsets and types of the received variables must be identical with offsets and types of the transmitted variables of the communication partner.

1. In the user program, set the TCP connection ID at the *A_Id* input.
2. In the user program, set the expected length (in bytes) of the variables to be sent at the *A_Len* input.
3. In the user program, set the *A_Req* input to TRUE.

i The function block responds to a rising edge at *A_Req*.

The A_Busy output is set to TRUE until the variables have been sent. Afterwards, A_Busy is set to FALSE and A_Done is set to TRUE.

If the sending process was not successful, the A_Error output is set to TRUE and an error code is output to A_Status.

3.7.3 TCP_RECEIVE

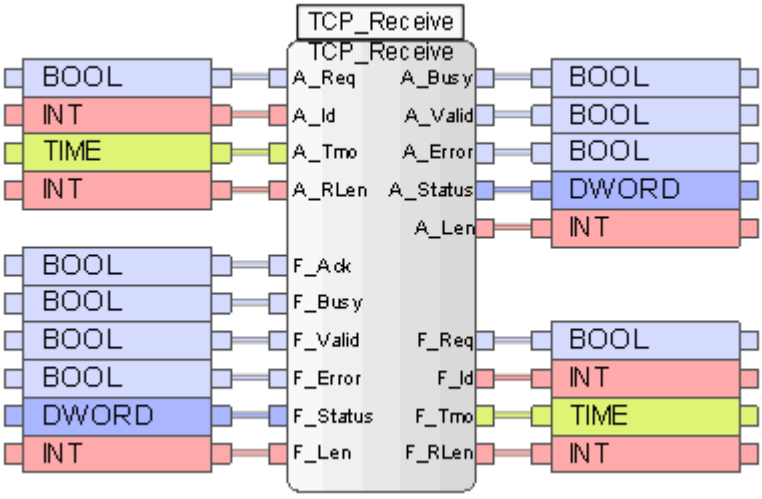


Figure 13: TCP_Receive Function Block

The **TCP_Receive** function block is used to receive predefined variables from the communication partner.

A function block with the same variables and offsets, e.g., TCP_Send, must be configured in the communication partner.

i To configure the function block, drag it from the function block library onto the user program (see also chapter 4.3).

Inputs and Outputs of the Function Block with Prefix A

These inputs and outputs can be used to control and evaluate the function block using the user program. The prefix A means Application.

A_Inputs	Description	Type
A_Req	The rising edge triggers the function block.	BOOL
A_Id	Identification number of the configured TCP connection to the communication partner from which data should be received.	INT
A_Tmo	Receive timeout If no data are received within the timeout, the function block stops and an error message appears. If the *A_Tmo* input is not used or set to zero, the timeout is deactivated.	TIME
A_RLen	*A_RLen* is the expected length of the variables to be received, expressed in bytes. *A_RLen* must be greater than zero and must not end within a variable.	INT

Table 28: A-Inputs for the TCP_Receive Function Block

A_Outputs	Description	Type
A_Busy	TRUE: Data is still being received.	BOOL
A_Valid	TRUE: Data reception ended without error.	BOOL
A_Error	TRUE: An error occurred FALSE: No error.	BOOL
A_Status	The status and error code of the function block and of the TCP connection are output to *A_Status*.	DWORD
A_Len	Number of received bytes.	INT

Table 29: A-Outputs for the TCP_Receive Function Block

Inputs and Outputs of the Function Block with Prefix F:

These inputs and outputs of the function block establish the connection to the **Receive** function block in structure tree. The prefix F means Field.

i Common variables are used to connect the **Receive** function block (in the Blocks structure tree folder) to the **TCP_Receive** function block (in the user program). These must have been previously created in the Global Variable Editor.

Connect the *F-Inputs* of the **TCP_Receive** function block in the user program to the same variables that will be connected to the outputs of the **Receive** function block in the structure tree.

F-Inputs	Type
F_Ack	BOOL
F_Busy	BOOL
F_Valid	BOOL
F_Error	BOOL
F_Status	DWORD
F_Len	INT

Table 30: A-Inputs for the TCP_Receive Function Block

Connect the *F-Outputs* of the **TCP_Receive** function block in the user program to the same variables that will be connected to the inputs of the **Receive** function block in the structure tree.

F-Outputs	Type
F_Req	BOOL
F_ID	DWORD
F_Tmo	INT
F_RLen	INT

Table 31: F-Outputs for the TCP_Receive Function Block

To create the corresponding Receive function block in the structure tree:

1. In the structure tree, open **Configuration**, **Resource**, **Protocols**, **Send/Receive over TCP**, **Blocks, New**.
2. Select the **Receive** function block.
3. Right-click the **Receive** function block and select **Edit**.
   ☑ The window for assigning variables to the function blocks appears.

Connect the inputs of the **Receive** function block in the structure tree to the same variables that have been previously connected to the *F-Outputs* of the **TCP_Receive** function block in the user program.

Inputs	Type
ID	INT
REQ	BOOL
RLEN	INT
TIMEOUT	TIME

Table 32: Input System Variables

Connect the outputs of the **Receive** function block in the structure tree to the same variables that have been previously connected to the *F-Inputs* of the **TCP_Receive** function block in the user program.

Outputs	Type
Ack	BOOL
Busy	BOOL
Error	BOOL
LEN	INT
STATUS	DWORD
VALID	BOOL

Table 33: Output System Variables

Data	Description
Receive variables	Any variables can be created in the *Process Variables* tab. Offsets and types of the variables must be identical with offsets and types of the variables of the communication partner.

Table 34: Receive Variables

To operate the TCP_Receive function block, the following steps are essential:

i The receive variables must be created in the *Process Variables* tab of the *Receive* dialog box. Offsets and types of the receive variables must be identical with offsets and types of the send variables of the communication partner.

1. In the user program, set the identification number for the TCP connection at the *A_Id* input.
2. In the user program, set the receive timeout at the *A_Tmo* input.
3. In the user program, set the expected length of the variables to be received at the *A_RLen* input.
4. In the user program, set the *A_Req* input to TRUE.

i The function block starts with a rising edge at *A_Req*.

The *A_Busy* output is set to TRUE until the variables have been received or the receive timeout has expired. Afterwards, *A_Busy* is set to FALSE and *A_Valid* or *A_Error* to TRUE.

If no error occurred during the reception of the variables, the *A_Valid* output is set to TRUE. The variables defined in the *Data* tab can be evaluated.

If an error occurred during the variable reception, the *A_Error* output is set to TRUE and an error code is output to *A_Status*.

3.7.4 TCP_ReceiveLine

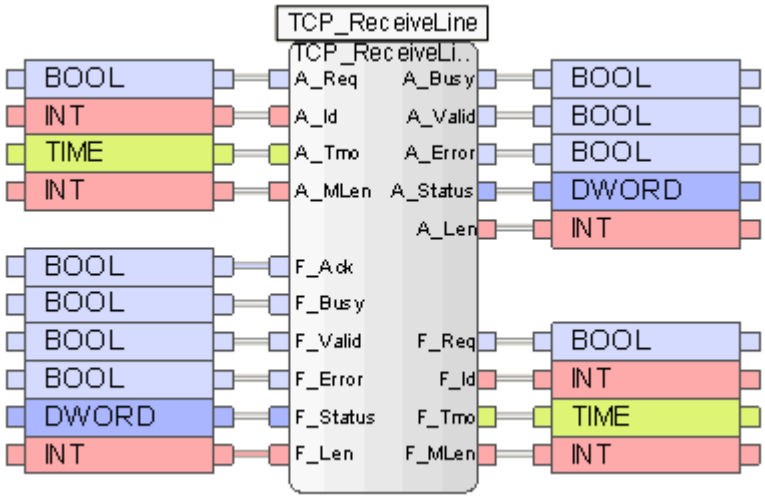


Figure 14: TCP_ReceiveLine Function Block

The **TCP_ReceiveLine** function block is used to receive an ASCII character string incl. LineFeed (16#0A) from a communication partner.

i To configure the function block, drag it from the function block library onto the user program (see also chapter 4.3).

Inputs and Outputs of the Function Block with Prefix A

These inputs and outputs can be used to control and evaluate the function block using the user program. The prefix A means Application.

A_Inputs	Description	Type
A_Req	Rising edge triggers the function block.	BOOL
A_Id	Identification number of the configured TCP connection to the communication partner from which data should be received.	INT
A_Tmo	Receive timeout If no data are received within the timeout, the function block stops and an error message appears. If the input is not used or set to zero, the timeout is deactivated.	TIME
A_MLen	*A_Mlen* is the maximum length of a line to be received, expressed in bytes. The receive variables must created in the *Data* tab located in the COM function block. Transmitted bytes = Min (A_MLen, line length, length of the data range).	INT

Table 35: A-Inputs for the TCP_ReceiveLine Function Block

A_Outputs	Description	Type
A_Busy	TRUE: Data is still being received.	BOOL
A_Valid	TRUE: Data reception ended without error.	BOOL
A_Error	TRUE: An error occurred. FALSE: No error.	BOOL
A_Status	The status and error code of the function block and of the TCP connection are output to A_Status.	DWORD
A_Len	Number of received bytes.	INT

Table 36: A-Outputs for the TCP_ReceiveLine Function Block

Inputs and Outputs of the Function Block with Prefix F:

These inputs and outputs of the function block establish the connection to the **ReceiveLine** function block in structure tree. The prefix F means Field.

i Common variables are used to connect the **Receive Line** function block (in the Blocks structure tree folder) to the **TCP_ReceiveLine** function block (in the user program). These must have been previously created in the Global Variable Editor.

Connect the *F-Inputs* of the **TCP_ReceiveLine** function block in the user program to the same variables that will be connected to the outputs of the **ReceiveLine** function block in the structure tree.

F-Inputs	Type
F_Ack	BOOL
F_Busy	BOOL
F_Valid	BOOL
F_Error	BOOL
F_Status	DWORD
F_Len	INT

Table 37: F-Inputs for the TCP_ReceiveLine Function Block

Connect the *F-Outputs* of the **TCP_ReceiveLine** function block in the user program to the same variables that will be connected to the inputs of the **ReceiveLine** function block in the structure tree.

F-Outputs	Type
A_Req	BOOL
A_Id	INT
A_Tmo	TIME
A_MLen	INT

Table 38: F-Outputs for the TCP_ReceiveLine Function Block

To create the corresponding ReceiveLine function block in the structure tree

1. In the structure tree, open **Configuration**, **Resource**, **Protocols**, **Send/Receive over TCP**, **Blocks, New**.
2. Select the **ReceiveLine** function block.
3. Right-click the **ReceiveLine** function block and select **Edit**.
   ☑ The window for assigning variables to the function blocks appears.

Connect the inputs of the **ReceiveLine** function block in the structure tree to the same variables that have been previously connected to the *F-Outputs* of the **TCP_ReceiveLine** function block in the user program.

Inputs	Type
ID	INT
MLEN	INT
REQ	BOOL
TIMEOUT	TIME

Table 39: Input System Variables

Connect the outputs of the **ReceiveLine** function block in the structure tree to the same variables that have been previously connected to the *F-Inputs* of the **TCP_ReceiveLine** function block in the user program.

Outputs	Type
ACK	BOOL
Busy	BOOL
Error	BOOL
LEN	INT
STATUS	DWORD
VALID	BOOL

Table 40: Output System Variables

Data	Description
Receive variables	The *Process Variables* tab should only contain variables of type BYTE. Offsets of the variables must be identical with offsets of the variables of the communication partner.

Table 41: Receive Variables

To operate the TCP_ReceiveLine function block, the following steps are essential:

i The receive variables of type BYTE must be created in the *Process Variables* tab of the *ReceiveLine* dialog box. Offsets of the receive variables must be identical with offsets of the send variables of the communication partner.

1. In the user program, set the identification number for the TCP connection at the *A_Id* input.
2. In the user program, set the receive timeout at the *A_Tmo* input.
3. In the user program, set the maximum length of the line to be received at the *A_MLen* input.

i *A_Mlen* must be greater than zero and determines the size of the receive buffer in bytes.

If the receive buffer is full and a line end has not yet occurred, the reading process ends and no error message appears.

The number of received bytes is output to *A_Len*:

Received bytes = Min (A_MLen, line length, length of the data range).

4. In the user program, set the *A_Req* input to TRUE.

i The function block responds to a rising edge at *A_Req*.

The *A_Busy* output is set to TRUE until the receive buffer is full, or the end of line *LineFeed* has been received or the receive timeout has expired. Afterwards, *A_Busy* is set to FALSE and *A_Valid* or *A_Error* to TRUE.

If no error occurred during the line reception, the *A_Valid* output is set to TRUE. The variables defined in the *Data* tab can be evaluated.

If an error occurred during the line reception, the *A_Error* output is set to TRUE and an error is output to *A_Status*.

3.7.5 TCP_ReceiveVar

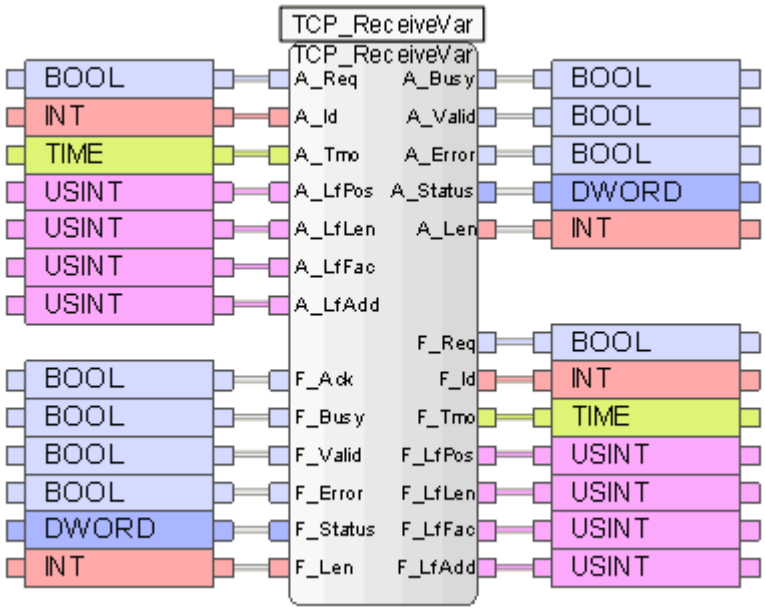


Figure 15: Function Block TCP_ReceiveVar

The **TCP_ReceiveVar** function block is used to evaluate data packets with variable length and containing a length field.

i To configure the function block, drag it from the function block library onto the user program (see also chapter 4.3).

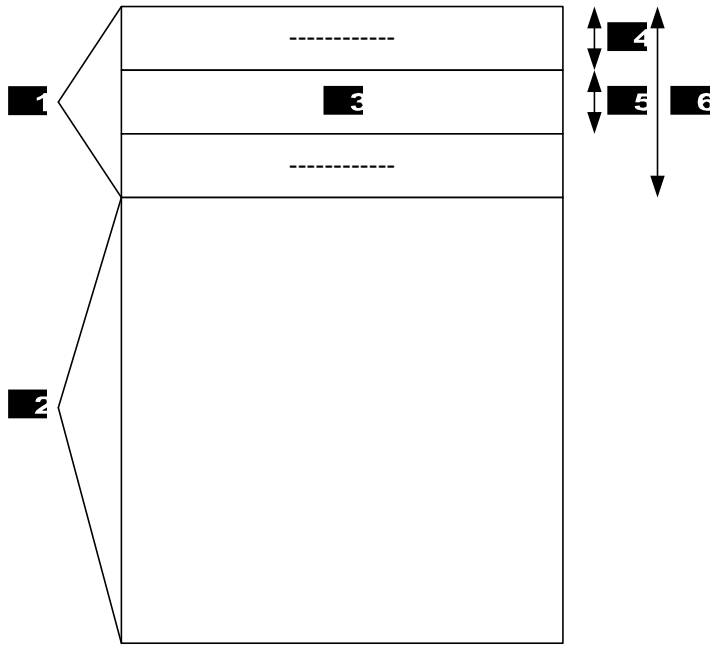
Functional Description

The received data packets must have the structure represented in the figure below (e.g., Modbus protocol). Modifying the input parameters *A_LfPos*, *A_LfLen*, *A_LfFac*, *A_LfLen*, the received data packets can be adapt to any protocol format.

The received data packet is composed of a header and a data range. The header contains data such as subscriber address, telegram function, length field etc. required for establishing communication. To evaluate the user data range, separate the header and read out the length field.

The size of the header is entered in the *A_LfAdd* parameter.

The length of the data range must be read from the length field of the data packet currently read. The position of the length field is entered in *A_LfPos*. The size of the length field expressed in bytes is entered in *LfLen*. If the length is not expressed in bytes, the corresponding conversion factor must be entered in *A_LfFac* (e.g., 2 for WORD or 4 for DOUBLE WORD).



1 Header
2 Data range
3 Length field
4 A_LfPos
5 A_LfLen
6 A_LfAdd

Figure 16: Data Packet Structure

Inputs and Outputs of the Function Block with Prefix A

These inputs and outputs can be used to control and evaluate the function block using the user program. The prefix A means Application.

A_Inputs	Description	Type
A_Req	Rising edge triggers the CPU function block.	BOOL
A_Id	Identification number *ID* of the configured TCP connection to the communication partner from which the data should be received.	DWORD
A_Tmo	Receive timeout If no data are received within the timeout, the function block stops and an error message appears. If the input is not used or set to zero, the timeout is deactivated.	INT
A_LfPos	Start position of the length field in the data packet; the numbering begins with zero (measured in bytes).	USINT
A_LfLen	Size of the *A_LfLen* length field in bytes. Permitted: 1, 2 or 4 bytes.	USINT
A_LfFac	Conversion factor in bytes if the value set in the length field is not expressed in bytes. If the input is not used or set to zero, **1** is used as default value.	USINT
A_LfAdd	Size of the header in bytes.	USINT

Table 42: A-Inputs for the TCP_ReceiveVar Function Block

A_Outputs	Description	Type
A_Busy	TRUE: Data is still being received.	BOOL
A_Valid	TRUE: Data reception ended without error.	BOOL
A_Error	TRUE: An error occurred during the reading process. FALSE: No error.	BOOL
A_Status	The status and error code of the function block and of the TCP connection are output to *A_Status*.	DWORD
A_Len	Number of received bytes.	INT

Table 43: A-Outputs for the TCP_ReceiveVar Function Block

Inputs and Outputs of the Function Block with Prefix F:

These inputs and outputs of the function block establish the connection to the **ReceiveVar** function block in structure tree. The prefix F means Field.

i Common variables are used to connect the **ReceiveVar** function block (in the Blocks structure tree folder) to the **TCP_ReceiveVar** function block (in the user program). These must have been previously created in the Global Variable Editor.

Connect the *F-Inputs* of the **TCP_ReceiveVar** function block in the user program to the same variables that will be connected to the outputs of the **ReceiveVar** function block in the structure tree.

F-Inputs	Type
F_Ack	BOOL
F_Busy	BOOL
F_Valid	BOOL
F_Error	BOOL
A_Status	DWORD
A_Len	INT

Table 44: F-Inputs for the TCP_ReceiveVar Function Block

Connect the *F-Outputs* of the **TCP_ReceiveVar** function block in the user program to the same variables that will be connected to the inputs of the **ReceiveVar** function block in the structure tree.

F-Outputs	Type
F_Req	BOOL
F_ID	INT
F_Tmo	TIME
F_LfPos	USINT
A_LfLen	USINT
A_LfFac	USINT
A_LfAdd	USINT

Table 45: F-Outputs for the TCP_ReceiveVar Function Block

To create the ReceiveVar function block in the structure tree

1. In the structure tree, open **Configuration**, **Resource**, **Protocols**, **Send/Receive over TCP**, **Blocks, New**.
2. Select the **ReceiveVar** function block.
3. Right-click the **ReceiveVar** function block and select **Edit**.
   ☑ The window for assigning variables to the function blocks appears.

Connect the inputs of the **ReceiveVar** function block in the structure tree to the same variables that have been previously connected to the *F-Outputs* of the **TCP_ReceiveVar** function block in the user program.

Inputs	Type
ID	INT
Lf Add	USINT
Lf Fac	USINT
Lf Len	USINT
Lf Pos	USINT
REQ	BOOL
TIMEOUT	TIME

Table 46: Input System Variables

Connect the outputs of the **ReceiveVar** function block in the structure tree to the same variables that have been previously connected to the *F-Inputs* of the **TCP_ReceiveVar** function block in the user program.

Outputs	Type
ACK	BOOL
Busy	BOOL
Error	BOOL
LEN	INT
STATUS	DWORD
VALID	BOOL

Table 47: Output System Variables

Data	Description
Receive variables	Any variables can be created in the *Process Variables* tab. Offsets and types of the variables must be identical with offsets and types of the variables of the communication partner.

Table 48: Receive Variables

To operate the TCP_ReceiveVar function block, the following steps are essential:

i The receive variables must be created in the *Process Variables* tab of the *ReceiveVar* dialog box. Offsets and types of the receive variables must be identical with offsets and types of the send variables of the communication partner.

1. In the user program, set the identification number for the TCP connection at the *A_Id* input.
2. In the user program, set the receive timeout at the *A_Tmo* input.
3. In the user program, set the parameters A_LfPos, A_LfLen, A_LfFac and *A_LfAdd*.
4. In the user program, set the *A_Req* input to TRUE.

i The function block starts with a rising edge at *A_Req*.

The *A_Busy* output is set to TRUE until the variables have been received or the receive timeout has expired. Afterwards, *A_Busy* is set to FALSE and *A_Valid* or *A_Error* to TRUE.

If no error occurred during the reception of the variables, the *A_Valid* output is set to TRUE. The variables defined in the Data tab can be evaluated. The *A_Len* output contains the number of bytes that were actually read out.

If an error occurred during the variable reception, the *A_Error* output is set to TRUE and an error code is output to *A_Status*.

3.8 Control Panel (Send/Receive over TCP)

The Control Panel can be used to verify and control the settings for the Send/Receive protocol. Details about the current status of the Send/Receive protocol (e.g., disturbed connections) are displayed.

To open the Control Panel for monitoring the Send/Receive protocol

1. In the structure tree, select **Resource**.
2. Right-click and select **Online** from the context menu.
3. In the **System Log-in** window, enter the access data to open the Control Panel for the resource.
4. In the structure tree associated with the Control Panel, select **Send/Receive Protocol**.

3.8.1 View Box for General Parameters

The view box displays the following values of the Send/Receive protocol.

Element	Description
Name	TCP SR protocol.
Planned µP Budget [%]	Value displayed for the planned maximum µP budget of the COM module, which may be produced during the protocol's processing.
Current µP Budget [%]	Value displayed for the current µP budget of the COM module, which is being produced during the protocol's processing.
Undisturbed connections	Number of undisturbed connections.
Disturbed Connections	Number of disturbed connections.

Table 49: View Box of the S/R Protocol

3.8.2 View Box for TCP Connections

The view box displays the following values of the selected TCP connections.

Element	Description
Name	TCP connection.
Partner timeout	Yes: Partner request timeout expired. No: Partner request timeout not expired.
Connection State	Current state of this connection: 0x00: Connection OK. 0x01: Connection closed. 0x02: Server waits for the connection to be established. 0x04: Client is attempting to establish a connection. 0x08: Connection is blocked.
Peer Address	IP address of the communication partner.
Peer Port	Port of the communication partner.
Own Port	Port of this controller.
Watchdog Time [ms]	It is the actual partner request timeout within which the communication partner received data at least one time after data has been sent.
Error code	Error code, see Chapter 3.8.3.
Timestamp Error Code [ms]	Timestamp for the last reported fault. Range of values: Seconds since 1/1/1970 in milliseconds.
Received Bytes [Bytes]	Number of bytes received in this TCP connection.
Transmitted Bytes [Bytes]	Number of bytes sent in this TCP connection.

Table 50: View Box of the Modbus Slaves

3.8.3 Error Code of the TCP Connection

The error codes can be read out from the *Error Code* variable.

For each configured connection: The connection state is composed of the connection state and error code of the last operation.

Error code Decimal	Error code Hexadecimal	Description
0	16#00	OK.
4	16#04	Interrupted system call.
5	16#05	I/O Error
6	16#06	Device unknown.
9	16#09	Invalid socket descriptor.
12	16#0C	No memory available.
13	16#0D	Access denied.
14	16#0E	Invalid address.
16	16#10	Device occupied.
22	16#16	Invalid value (e.g., in the length field).
23	16#17	Descriptor table is full.
32	16#20	Connection aborted.
35	16#23	Operation is blocked.
36	16#24	Operation currently in process.
37	16#25	Operation already in process.
38	16#27	Target address required.
39	16#28	Message to long.
40	16#29	Incorrect protocol type for the socket.
42	16#2A	Protocol not available.
43	16#2B	Protocol not supported.
45	16#2D	Operation on socket not supported.
47	16#2F	The address is not supported by the protocol.
48	16#30	Address already in use.
49	16#31	The address cannot be assigned.
50	16#32	Network is down.
53	16#35	Software caused connection abort.
54	16#36	Connection reset by peer.
55	16#37	No buffer space available.
56	16#38	Socket already connected.
57	16#39	Socket not connected.
58	16#3A	Socket closed.
60	16#3C	Operation time expired.
61	16#3D	Connection refused (from peer).
65	16#41	No route to peer host.
78	16#4E	Function not available.
254	16#FE	Timeout occurred.
255	16#FF	Connection closed by peer.

Table 51: Error Codes of the TCP Connection

3.8.4 Additional Error Code Table for the Function Blocks

The error codes for the function blocks are only output to A_Status of the S/R TCP function blocks.

Error code Decimal	Error code Hexadecimal	Description
129	16#81	No connection exists with this identifier.
130	16#82	Length is greater than or equal to null.
131	16#83	Only cyclic data is permitted for this connection.
132	16#84	Invalid state.
133	16#85	Timeout value too large.
134	16#86	Internal program error.
135	16#87	Configuration error.
136	16#88	Transferred data does not match data area.
137	16#89	Function block stopped.
138	16#8A	Timeout occurred or transmission blocked.
139	16#8B	Another function block of this type is already active on this connection.

Table 52: Additional Error Codes

3.8.5 Connection State

The partner's connection state is displayed via the following error codes.

Error code Hexadecimal	Description
16#00	Connection OK.
16#01	Connection closed.
16#02	Server waits for the connection to be established.
16#04	Client attempts to establish connection.
16#08	Connection is blocked.

Table 53: Connection State

3.8.6 Partner Connection State

The partner's connection state is displayed as follows.

Protocol State Decimal	Description
0	No connection.
1	Connection OK.

Table 54: Partner's Connection State

4 General

This chapter describes parameters that are relevant for all communication protocols.

4.1 Maximum Communication Time Slice

The maximum communication time slice is the time period in milliseconds (ms) per CPU cycle assigned to the processor module for processing the communication tasks. Even if the protocol processing could not be completed within one communication time slice, the CPU still executes the safety-relevant monitoring for all protocols within one CPU cycle.

i If not all upcoming communication tasks can be processed within one CPU cycle, the whole communication data is transferred over multiple CPU cycles. The number of communication time slices is then greater than 1.

For calculating the maximum response time, the number of communication time slices must be equal to 1.

4.1.1 Determining the Maximum Duration of the Communication Time Slice

For a first estimate of the maximum duration of the communication time slice, the sum of the following times must be entered in the *Max. Com. Time Slice [ms]* system parameter located in the properties of the resource.

- For each COM module: 3 ms.
- For each redundant safe**ethernet** connection: 1 ms.
- For non-redundant safe**ethernet** connection: 0.5 ms.
- For each kilobyte user data of non-safety-related protocols, e.g., Modbus: 1 ms.

HIMA recommends comparing the value estimated for *Max. Com. Time Slice [ms]* with the value displayed in the Control Panel and, if necessary, correcting it in the properties of the resource. This can be done during an FAT (factory acceptance test) or SAT (site acceptance test).

To determine the actual duration of the maximum communication time slice

1. Operate the HIMA system under full load (FAT, SAT):
   All communication protocols are in operation (safe**ethernet** and standard protocols).
2. Open the **Control Panel** and select the **Com. Time Slice** structure tree folder.
3. Read the vale displayed for *Maximum Com. Time Slice Duration per Cycle [ms].*
4. Read the value displayed for *Maximum Number of Required Com. Time Slice Cycles*.

The duration of the communication time slice must be set so that, when using the communication time slice, the CPU cycle cannot exceed the watchdog time specified by the process.

4.2 Load Limitation

A computing time budget expressed in % *(µP budget)* can be defined for each communication protocol. It allows the available computing time to be distributed among the configured protocols. The sum of the computing time budgets configured for all communication protocols on a CPU or COM module may not exceed 100%.

The defined computing time budgets of the individual communication protocols are monitored. If a communication protocol has already achieved or exceeded its budget and no reserve computing time is available, the communication protocol cannot be processed.

If sufficient additional computing time is available, it is used to process the communication protocol that has already achieved or exceeded its budget. It can therefore happen that a communication protocol uses more computing time budget than has been allocated to it.

It is possible that more than 100% computing time budget is displayed online. This is not a fault; the computing time budget exceeding 100% indicates the additional computing time used.

i The additional computing time budget is not a guarantee for a certain communication protocol and can be revoked from the system at any time.

4.3 Configuring the Function Blocks

The fieldbus protocols and the corresponding function blocks operate on the COM module of HIMA controllers. In the SILworX structure tree, these function blocks must therefore be created as child element of **Configuration, Resource, Protocols**.

To control the function blocks on the COM module, function blocks can be created in the SILworX user program (see Chapter 4.3.1). These can be used as standard function blocks.

Shared variables are used to connect the function blocks in the SILworX user program to the corresponding function blocks in the SILworX structure tree. These must have been previously created in the Global Variable Editor.

4.3.1 Purchasing Function Block Libraries

The function block libraries for PROFIBUS DP and TCP Send/Receive must be added to the project using the *Restore* function (context menu of the project).

The function block library is available from HIMA support upon request, see Chapter 1.4.

4.3.2 Configuring the Function Blocks in the User Program

Drag the required function blocks onto the user program. Configure the inputs and outputs as described for the individual function block.

Upper part of the function block

The upper part of the function block corresponds to the user interface used by the user program to control it.

The variables used in the user program are connected at this level. The prefix A means Application.

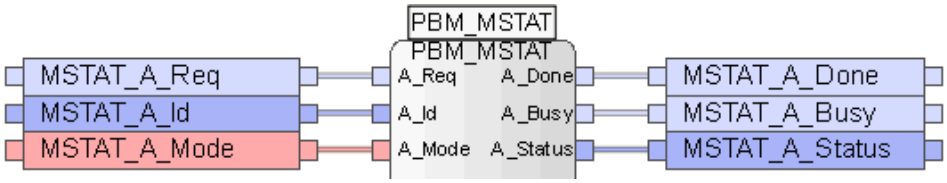


Figure 17: PNM_MSTST Function Block (Upper Part)

Lower part of the function block

The lower part of the function block represents the connection to the function block (in the SILworX structure tree).

The variables that must be connected to the function block located in SILworX structure tree are connected here. The prefix F means Field.

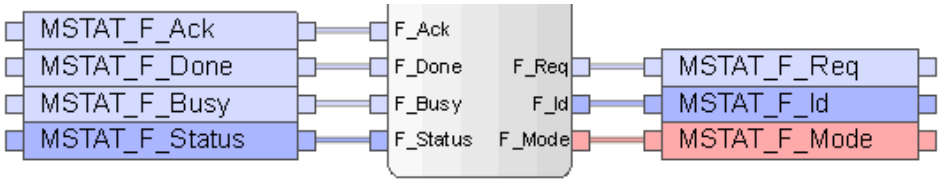


Figure 18: PNM_MSTST Function Block (Lower Part)

4.3.3 Configuring the Function Blocks in the SILworX Structure Tree

To configure the function block in the SILworX structure tree

1. In the structure tree, open **Configuration**, **Resource**, **Protocols**, e.g., **PROFIBUS Master**.
2. Right-click **Function Blocks** and select **New**.
3. In the SILworX structure tree, select the suitable function block.

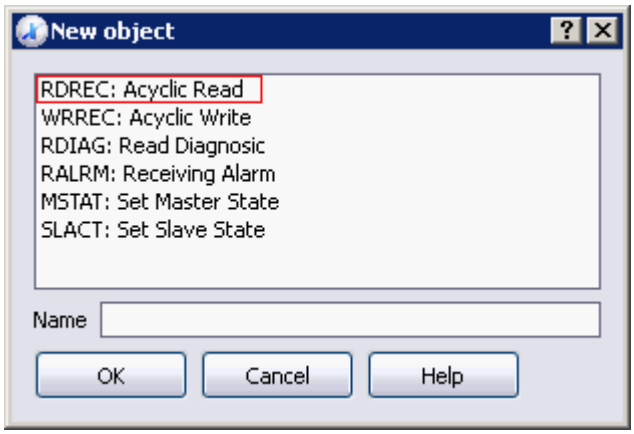


Figure 19: Selecting Function Blocks

The inputs of the function block (checkmark in the Input Variables column) must be connected to the same variables that are connected in the user program to the *F_Outputs* of the function block.

The outputs of the function block (no checkmark in the Input Variables column) must be connected to the same variables that are connected in the user program to the *F_Inputs* of the function block.

System Variables

F	Name	Data type	Input variable	Global Variable
1	ACK	BOOL	☑	MSTAT_F_Ack
2	BUSY	BOOL	☑	MSTAT_F_Busy
3	DONE	BOOL	☑	MSTAT_F_Done
4	M_ID	DWORD	☐	MSTAT_F_Id
5	MODE	INT	☐	MSTAT_F_Mode
6	REQ	BOOL	☐	MSTAT_F_Req
7	STATUS	DWORD	☑	MSTAT_F_Status

Figure 20: System Variables of the MSTAT Function Block

Appendix

Glossary

Term	Description
ARP	Address resolution protocol, network protocol for assigning the network addresses to hardware addresses.
Bit variable	Variable that is addressed bit by bit.
CENELEC	Comité Européen de Normalisation Électrotechnique (European Commitee for Electrotechnical Standardization).
COM	Communication module.
Connector board	Connector board for the HIMax module.
CPU	Processor module.
CRC	Cyclic redundancy check.
Data view	The global variables for output and output data are assigned to a data view to allow access to Modbus sources.
EN	European standard.
Export area	The export area is the process data volume that is written to by the system (a user program, hardware input or another protocol) and is read by the Modbus master.
FB	Fieldbus.
FBD	Function block diagrams.
ICMP	Internet control message protocol, network protocol for status or error messages.
IEC	International electrotechnical commission.
Import area	Process data volume that is written to by the Modbus master and can be used as input data for the system (in a user program, hardware output or another protocol).
Interference-free	Supposing that two input circuits are connected to the same source (e.g., a transmitter). An input circuit is termed "interference-free" if it does not distort the signals of the other input circuit.
KE	Communication end point.
MAC address	Media access control address, hardware address of one network connection.
NSIP	Not safety-related protocol.
PADT	Programming and debugging tool (acc. to IEC 61131-3), PC with SILworX.
PE	Protective ground.
PELV	Protective extra low voltage.
PES	Programmable electronic system.
R	Read.
R/W	Read/Write.
Rack ID	Base rack identification (number).
Register variable	Variable that is addressed word by word.
SB	System bus.
SFF	Safe failure fraction, i.e., portion of faults that can be safely controlled.
SIF	Safety-instrumented function.
SIL	Safety integrity level (in accordance with IEC 61508).
SILworX	Programming tool for HIMax, HIQuad X und HIMatrix.
SIP	Safety-instrumented protocol.
SNTP	Simple network time protocol (RFC 1769).
SRS	System.Rack.Slot.
SW	Software.
TMO	Timeout.
W	Write.
WD	Watchdog.
WDT	Watchdog time.

Index of Figures

Figure 1: Connecting a HIMax to a Siemens Controller 10
Figure 2: Data Transfer between a HIMax and a Siemens Controller 11
Figure 3: List of Variables in the Siemens UDT1 Block 12
Figure 4: List of Variables in the Siemens DB1 Function Block 13
Figure 5: SIMATIC Symbol Editor 13
Figure 6: Receive Function Chart 14
Figure 7: Send Function Chart 15
Figure 8: TCP Connection Properties in SILworX 16
Figure 9: Siemens List of Variables 25
Figure 10: HIMA List of Variables 25
Figure 11: TCP_Reset Function Block 27
Figure 12: TCP_Send Function Block 30
Figure 13: TCP_Receive Function Block 33
Figure 14: TCP_ReceiveLine Function Block 37
Figure 15: Function Block TCP_ReceiveVar 41
Figure 16: Data Packet Structure 42
Figure 17: PNM_MSTST Function Block (Upper Part) 51
Figure 18: PNM_MSTST Function Block (Lower Part) 51
Figure 19: Selecting Function Blocks 52
Figure 20: System Variables of the MSTAT Function Block 52

Index of Tables

Table 1: Additional Applicable Manuals 5
Table 2: Equipment and System Requirements for S/R TCP 9
Table 3: S/R TCP Properties 9
Table 4: HIMax Controller Configuration 11
Table 5: Siemens SIMATIC 300 Configuration 11
Table 6: Global Variables 16
Table 7: Variables for Receive Data 17
Table 8: Variables for Send Data 17
Table 9: S/R TCP System Variables 18
Table 10: General Properties of the S/R TCP 18
Table 11: COM/CPU Parameters 19
Table 12: System Variables 20
Table 13: S/R TCP Connection Properties 22
Table 14: Function Blocks for S/R TCP Connections 26
Table 15: A-Inputs for the TCP_Reset Function Block 27
Table 16: A-Outputs for the TCP_Reset Function Block 27

Table 17: F-Inputs for the TCP_Reset Function Block 28
Table 18: F-Outputs for the TCP_Reset Function Block 28
Table 19: Input System Variables 28
Table 20: Output System Variables 29
Table 21: A-Inputs for the TCP_Send Function Block 30
Table 22: A-Outputs for the TCP_Send Function Block 30
Table 23: F-Inputs for the TCP_Send Function Block 31
Table 24: F-Outputs for the TCP_Send Function Block 31
Table 25: Input System Variables 31
Table 26: Output System Variables 32
Table 27: Send Data 32
Table 28: A-Inputs for the TCP_Receive Function Block 33
Table 29: A-Outputs for the TCP_Receive Function Block 34
Table 30: A-Inputs for the TCP_Receive Function Block 34
Table 31: F-Outputs for the TCP_Receive Function Block 34
Table 32: Input System Variables 35
Table 33: Output System Variables 35
Table 34: Receive Variables 35
Table 35: A-Inputs for the TCP_ReceiveLine Function Block 37
Table 36: A-Outputs for the TCP_ReceiveLine Function Block 38
Table 37: F-Inputs for the TCP_ReceiveLine Function Block 38
Table 38: F-Outputs for the TCP_ReceiveLine Function Block 38
Table 39: Input System Variables 39
Table 40: Output System Variables 39
Table 41: Receive Variables 39
Table 42: A-Inputs for the TCP_ReceiveVar Function Block 42
Table 43: A-Outputs for the TCP_ReceiveVar Function Block 43
Table 44: F-Inputs for the TCP_ReceiveVar Function Block 43
Table 45: F-Outputs for the TCP_ReceiveVar Function Block 43
Table 46: Input System Variables 44
Table 47: Output System Variables 44
Table 48: Receive Variables 44
Table 49: View Box of the S/R Protocol 46
Table 50: View Box of the Modbus Slaves 46
Table 51: Error Codes of the TCP Connection 47
Table 52: Additional Error Codes 48
Table 53: Connection State 48
Table 54: Partner's Connection State 48

MANUAL

Send/Receive TCP

HI 801 524 E

For further information, please contact:

HIMA Paul Hildebrandt GmbH
Albert-Bassermann-Str. 28
68782 Brühl, Germany

Phone +49 6202 709-0
Fax +49 6202 709-107
E-mail info@hima.com

Learn more about HIMA solutions online:

www.hima.com/en/

www.hima.com