

Industrial-Automation **System *HIMatrix***

TCP S/R **Manual**



HIMA Paul Hildebrandt GmbH
Industrial Automation

HI 800 117 CEA

Important Notes

All HIMA products mentioned in this manual are protected under the HIMA trademark. Unless not explicitly noted, this may apply for other referenced manufacturers and their respective products.

All technical statements and data in this manual have been written with great care and effective quality measures have been taken to ensure their validity; however this manual may contain flaws or typesetting errors.

For this reason, HIMA does not offer any warranties nor assume legal responsibility nor any liability for possible consequences of any errors in this manual. HIMA appreciates any correspondence noting potential errors.

Technical modifications reserved.

For more information see the documentation on CD-ROM and on our web site www.hima.com .

More information can be requested from:

HIMA Paul Hildebrandt GmbH
Postfach 1261
68777 Brühl

Tel: +49(6202)709 0
Fax: +49(6202)709 107

e-mail: info@hima.com

About this manual

The objective of this manual is to familiarize the user with the HIMA "Send/Receive over TCP" protocol (TCP S/R protocol) and to support the installation, configuration and operation of the TCP S/R protocol.

To set up the TCP S/R protocol, the user needs the programming tool **ELOP II Factory** which must be installed on a PC running Microsoft Windows NT[®] or Windows 2000[®].

The user should be familiar with the programming tool **ELOP of II Factory** and the HIMA *HIMatrix* controllers. For self-study, HIMA recommends the manual "First Steps **ELOP II Factory**" and the **ELOP II Factory**'s Online Help. Additionally, HIMA offers client specific training.

This manual is organized in four parts:

- The first part "Introduction" gives an overview of the properties and the usage of the TCP S/R protocol.
- The second part "Program description" explains the menu functions and the dialogs within **ELOP II Factory** used to configure the TCP S/R protocol.
- The third part "TCP S/R function blocks" describes the function and configuration of the TCP S/R function blocks.
- In the fourth part "Application" an application of the TCP S/R protocol is described, which the user can reproduce in a step instruction.

We would like to wish You good speed in converting Your TCP S/R projects. For questions, please contact HIMA directly.

All rights and technical reserved changes.

© HIMA Paul Hildebrandt GmbH
Postfach 1261
D - 68777 Brühl bei Mannheim

Table of Contents	Page
1 Introduction	5
1.1 Required equipment and system requests.....	5
1.2 Properties of the TCP S/R protocol	5
1.3 Communication via TCP S/R.....	7
1.3.1 TCP Connections	7
1.3.2 Cyclical Data Exchange	8
1.3.3 Acyclic Data Exchange with Function Blocks	8
1.3.4 Simultaneous Cyclical and Acyclic Data Exchange.....	9
1.3.5 Flow Control	9
1.4 Third-party systems with “pad bytes”	10
2 Program description	11
2.1 Context menu "Send/Receive over TCP"	12
2.1.1 Menu function "Connect signals"	12
2.1.2 Menu function "Validate"	12
2.1.3 Menu function "New"	12
2.1.4 Menu function "Copy, Insert, Delete, Print"	13
2.2 Context menu "TCP Connection"	13
2.2.1 Menu function "Connect signals"	13
2.2.2 Menu function "Validate"	14
2.2.3 Menu function "Copy, Insert, Delete, Print"	14
2.2.4 Menu function "Properties"	15
2.3 Context menu "Function blocks"	18
2.3.1 Menu function "New"	18
2.3.2 Menu function "Print"	18
2.3.3 Context Menu of the COM Function Blocks	19
2.3.4 Menu function "Connect signals".....	19
2.3.5 Menu function "Copy, Insert, Delete, Print"	19
2.3.6 Menu function "Properties"	19
2.4 Status and error codes.....	20
3 TCP S/R Function Block	23
3.1 Mode of Operation of the Function Blocks.....	24
3.2 Function Block "TCP_Receive"	26
3.2.1 CPU Function Block "TCP_Receive"	26
3.2.2 Sequence of operation	28
3.2.3 COM Function Block "Receive"	29
3.3 Function Block "TCP_Receive_Line"	31
3.3.1 CPU Function Block "TCP_Receive_Line"	31
3.3.2 Sequence of operations	33
3.3.3 COM Function Block "Receive_Line"	34

3.4	Function Block "TCP_Receive_Var"	36
3.4.1	CPU Function Block "TCP_Receive_Var"	37
3.4.2	Sequence of operations	39
3.4.3	COM Function Block "Variable_Receive"	40
3.5	Function Block "TCP_Reset"	42
3.5.1	CPU Function Block "TCP_Reset"	42
3.5.2	Sequence of operations	43
3.5.3	COM Function Block "Reset"	44
3.6	Function Block "TCP_Send"	46
3.6.1	CPU Function Block "TCP_SEND"	46
3.6.2	Sequence of operations	48
3.6.3	COM Function Block "Send"	49
3.7	Auxiliary Function Blocks	51
4	Application	53
4.1	Cyclic data exchange between Siemens and HIMA	53
4.1.1	Configuration of the data exchange	54
4.1.2	Configuration the Siemens SIMATIC 300	56
4.1.3	Configuration of the TCP S/R protocol in a <i>HIMatrix</i> F60	60

1 Introduction

TCP S/R is a manufacturer-independent, not safety-related protocol for cyclic and acyclic data exchange and does not require a specific protocol except for TCP/IP. With the TCP S/R protocol the HIMA *HIMatrix* support almost every third-party system and also PC's with an available socket interface (for example Winsock.dll) to TCP/IP.

Note	In the first place the not safety-related TCP S/R protocol is an additional interface to communicate with systems from other manufacturers. For the communication between HIMA <i>HIMatrix</i> control devices via Ethernet, the safety-related HIMA peer to peer protocol should be used.
-------------	---

1.1 Required equipment and system requests

HIMA <i>ELOP II Factory</i>	from version 5.xx
<i>HIMatrix</i> control devices	F20, F30, F35 and F60 from hardware revision: 00
Operating system versions of <i>HIMatrix</i> control devices	-COM OS from version 8.4 -CPU OS from version 4.xx
License number	For unlocking the TCP S/R protocol

1.2 Properties of the TCP S/R protocol

Safety-related	No
Interface	Ethernet 10/100BaseT
Data exchange	Cyclic and acyclic data exchange via TCP/IP.
Function blocks	The TCP S/R function blocks must be used for the acyclic data exchange (see chapter 3).
TCP connection	Up to 32 TCP connections can be configured in one control device, unless the maximum size of send data or receive data is exceeded.
Send data	A maximum of 8192 Bytes of data can be transmitted in total (The status signals of the configured TCP connections and TCP/SR Functionblocks must be subtracted from the maximum of send data). The distribution onto several TCP connections is arbitrary.

Receive data

A maximum of 8192 Bytes receive data can be received in total (The status signals of the configured TCP connections and TCP/SR Functionblocks must be subtracted from the maximum of send data). The distribution onto several TCP connections is arbitrary.

Note

In addition to the TCP S/R protocol, other protocols (e.g. Profibus-DP, Modbus...) can operate at the same time on a *HIMatrix* control device. In sum, 16284 bytes of data can be transmitted and 16284 bytes of data can be received per *HIMatrix* control device. The 16284 bytes can be arbitrarily separated between the protocols, but not more than 8192 bytes for one protocol and one direction.

1.3 Communication via TCP S/R

TCP S/R works according to the client/server principle. The connection must be initiated by the communication partner, which is configured as the client. After the first connection, both communication partners are equal and can send data at any time. TCP S/R has no protocol for data protection of its own, but uses the TCP/IP protocol. Since TCP sends the data in a "data stream", it must be guaranteed that the offsets and types of the to be exchanged signals are identical on the receiver side and on the transmitter side.

TCP S/R is compatible to the Siemens SEND/RECEIVE interface and allows the cyclical data exchange with the Siemens S7 function blocks of AG_SEND (FC5) and AG_RECV (FC6) (see also 4.1).

In addition, HIMA provides five TCP S/R function blocks by which the communication can be controlled via the user program and can be configured individually. Using the TCP S/R function blocks, arbitrary protocols (e.g. Modbus), which are transmitted via TCP can be send or receive.

1.3.1 TCP Connections

For each connection via TCP S/R with a communication partner, at least one TCP-connection must be created in the HIMA *HIMatrix* control device.

In the "properties" of the TCP connection, the identification number of the TCP connection and the addresses/ports of the own control device and of the communication partner must be entered.

- A new TCP connection is created via the context menu *New->TCP-Connection* of "Send/Receive over TCP".
- The configuration of the TCP connection must be done in the dialog window "properties" of the TCP connection (see 2.2.4).
- A maximum of 32 TCP connections can be created in one HIMA *HIMatrix* control device.
- The created TCP connections must have different identification numbers and different addresses/ports.

Note	The HIMA <i>HIMatrix</i> and the third-party system must be located in the same subnet. Or if you use a router, the corresponding routing settings are required (see ELOP II Factory Online Help "IP Settings").
-------------	---

1.3.2 Cyclical Data Exchange

When using cyclical data exchange, a send interval must be defined in the HIMA *HIMatrix* control device and in the communication partner.

The send interval defines the cyclical interval, within which the sending communication partner send the signals to the receiving communication partner.

- In order to ensure a continuous data exchange, both communication partners should approximately define the same sending interval (see 1.3.5).
- The option "Cyclical Send Data" must be activated in the used TCP connection for the cyclical data exchange.
- If "Cyclical Send Data" is activated in a TCP connection, no function blocks can be used.
- The signals to be send and received are allocated in the dialog window "Signal connections" of the TCP connection. Receive signals must exist, send signals are optional.

Note	The same signals (same offsets and types) which are defined as send signals by one communication partner must be defined as receive signals by the other communication partner.
-------------	---

1.3.3 Acyclic Data Exchange with Function Blocks

The acyclic data exchange is controlled by the user program via the TCP S/R function blocks in the HIMA *HIMatrix* control device.

So it is possible to control the data exchange with a timer or a mechanical switch at a physical input of the HIMA *HIMatrix* control device.

- The option "Cyclical Send Data" must be disabled in the used TCP connection.
- Only one TCP S/R function block can send at a certain time. The function and the configuration of the TCP S/R function blocks is described in chapter 3.
- The send signals or receive signals are allocated in the tab "Data" in the dialog window "Signal connections" of the TCP S/R function blocks (all except for "Reset").

Note	The same signals (same offsets and types) which are defined as send signals by one communication partner must be defined as receive signals by the other communication partner.
-------------	---

1.3.4 Simultaneous Cyclical and Acyclic Data Exchange

A HIMA *HIMatrix* control device can simultaneously exchange cyclical and acyclic data with one communication partner. For this, one TCP connection must be configured for cyclical data and a second TCP connection for the acyclic data. One single TCP connection cannot be used simultaneously for cyclical and acyclic data exchange.

1.3.5 Flow Control

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

If no data packet is received after five transmitted data packets, then the sending is blocked. If no data packet is received within the Send Timeout (45 seconds), the TCP connection check, closes this TCP connection.

- During the project planning it has to be ensured, that neither of the two Stations sends more data than the other Station can be process synchronously.
- In order to guarantee a continuous data exchange, the flow control must be considered for the cyclical and acyclic data exchange.
 - For the cyclical data exchange the same send interval must be adjusted near at both communication partners.
 - For the acyclic data exchange the user must configure the function block TCP S/R, that only one communication partner can send at a certain time.

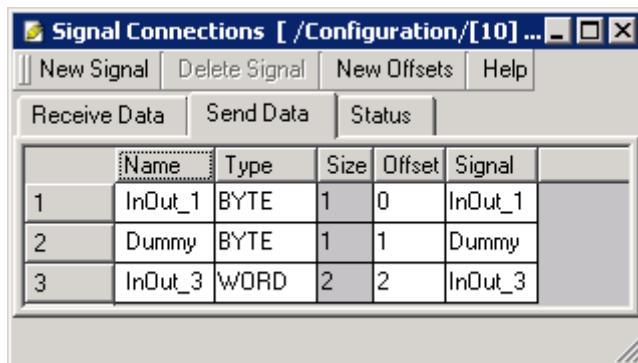
1.4 Third-party systems with “pad bytes”

For the cyclical and acyclic data exchange it has to be considered, that some control devices (e.g. SIMATIC 300) insert "pad bytes". The “pad bytes” guarantee, that all data types which are larger than a byte, always begin at an even offset and that the overall length of the data packets is also even.

In the HIMA control device “dummy bytes” must be inserted for the “pad bytes” at the corresponding places.

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 1: In the Siemens control device a "pad byte" is inserted (not visible), so that the variable “InOut_3” begins at an even offset



Signal Connections [/Configuration/[10] ...					
New Signal Delete Signal New Offsets Help					
Receive Data Send Data Status					
	Name	Type	Size	Offset	Signal
1	InOut_1	BYTE	1	0	InOut_1
2	Dummy	BYTE	1	1	Dummy
3	InOut_3	WORD	2	2	InOut_3

Figure2: In the HIMA control device a “dummy byte” must be inserted, so that the signal “InOut_3” has the same offset as in the Siemens control device.

2 Program description

The program description explains the menu options and dialogs in **ELOP II Factory** which are needed for the configuration of the TCP S/R protocol.

Note The TCP S/R protocol can be configured in the *HIMatrix* control devices F20, F30, F35 and F60 from hardware revision “00”.

Start **ELOP II Factory** and create a new project, or load an existing project. Then switch to the Hardware-Management and choose *New->Send/Receive over TCP* from the context menu for Protocols to create a new TCP S/R protocol in the resource.

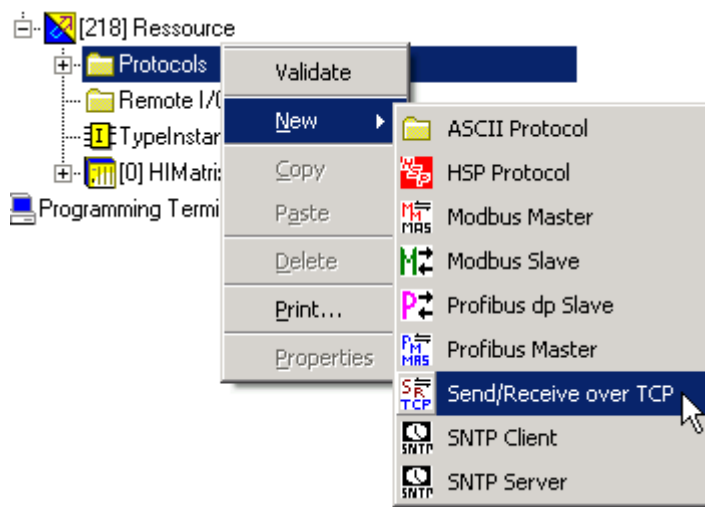


Figure3: New TCP S/R protocol

2.1 Context menu "Send/Receive over TCP"

The context menu of the TCP S/R protocol contains the following functions.

TCP S/R
Connect signals
Validate
New
Copy
Insert
Delete
Print
Properties

2.1.1 Menu function "Connect signals"

The menu function *Connect Signals* from the context menu of the TCP S/R protocol opens the dialog window "Signal-Connections".

Signal	Description	Type
Status	No function! (To evaluate status information of the user program see chapter 2.4)	DWORD

Table 1: Register "status" in the dialog window "Connect signals"

2.1.2 Menu function "Validate"

Before the code generation, the parameter settings of the TCP S/R protocol can be tested. In the structure view, the "Send/Receive over TCP" protocol must be selected and *Validate* must be chosen in the context menu. Errors and warnings will be displayed in the error-state viewer.

The validation is also executed automatically before each code generation. If an error is found by the validation, the code generation will be aborted.

2.1.3 Menu function "New"

With *New->TCP-Connection* from the context menu of the TCP S/R protocol a new TCP Connection is added to the TCP S/R protocol.

2.1.4 Menu function "Copy, Insert, Delete, Print"

Copy: Copy the TCP S/R protocol into the clipboard, including configuration.
Insert: Add a TCP S/R protocol to a protocol from the clipboard.
Delete: Delete the chosen TCP S/R protocol from the project.
Print: Print all signals and configurations of the TCP S/R protocol from this resource.

2.2 Context menu "TCP Connection"

The context menu of the TCP Connection contains the following functions.

TCP Connection
Connect signals
Validate
New
Copy
Insert
Delete
Print
Properties

2.2.1 Menu function "Connect signals"

The menu function *Connect Signals* from the context menu of the TCP connection opens the dialog "Signal Connections".

The dialog "Signal Connections" provides the three tabs:

- Receive data,
- Send data and
- Status

The signals for the cyclic data exchange, which are received by this control device, must be registered in the tab "Receive data".

Receive data	Description
Signals for the cyclic data exchange	In the tab "Receive data" arbitrary signals can be created. The offsets and types of the signals, however, must be identical with the offsets and types of the signals (send data) of the communication partner.

Table 2: Tab "Receive data" in the dialog window "Connect signals"

The signals for cyclic data exchange, which are sent by this control device, must be registered in the tab "Send data".

Send data	Description
Signals for the cyclic data exchange	In the tab "Send data" arbitrary signals can be created. The offsets and types of the signals must be, however, identical with the offsets and types of the signals (receive data) of the communication partner.

Table 3: Tab "Send data" in the dialog window "Connect signals"

The signals in the tab "status" allow to evaluate the state of the TCP connection in the user program.

Status	Description	Type
Bytes received	Number of bytes that were received up to now	UDINT
Bytes sent	Number of bytes that were sent up to now	UDINT
Status	Connection status and error code of the TCP Connection (see 2.4).	DWORD

Table 4: Tab "status" in the dialog window "Connect signals"

2.2.2 Menu function "Validate"

Before the code generation, the parameter settings of the TCP connection can be tested. In the structure view, the TCP connection must be selected and *Validate* must be chosen in the context menu. Errors and warnings will be displayed in the error-state viewer.

The validation is also executed automatically before every code generation. If an error is found by the validation, the code generation will be aborted.

2.2.3 Menu function "Copy, Insert, Delete, Print"

Copy: Copy the TCP connection into the clipboard including configuration.
Insert: Add a TCP connection to a TCP S/R protocol from the clipboard.
Delete: Delete the chosen TCP connection from the TCP S/R protocol.
Print: Print all signals and configurations of this TCP connection.

2.2.4 Menu function "Properties"

With *Properties* in the context menu of the TCP S/R protocol, the Dialog "Properties" is opened.

Note	The data exchange via a TCP connection is executed either cyclic or acyclic. The TCP S/R function blocks are required for the acyclic data exchange. In case of the cyclic data exchange the operation of TCP S/R function blocks is not possible.
-------------	--

Status	Description	Type
Type	TCP connection	Display only
Name	Arbitrary, unique name, for one TCP connection.	A maximum of. 32 chars
Id	Arbitrary, but unique identification number "Id" for every TCP connection. The "Id" is required also as reference for the TCP S/R function blocks.	0..255 Default: 0
Mode	Server: This station works as a server (passive mode). The connection must be initiated through the communication partner (Client). After the first connection, both communication partners are equal and can send data at any time. Indication of the own port is required.	Default: Server
	Server with defined peer: This station works as a server (passive mode). The connection must be initiated through the communication partner (Client). After the first connection, both communication partner are equal and can send data at any time. If the IP address and/or port of the communication partner are defined here, only this defined 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, no check up occurs for this parameter.	

Status	Description	Type
	<p>Client:</p> <p>This station works as a client, i.e. the station initiates the connection with the communication partner.</p> <p>Information of Ip address and port of the communication partner is required.</p> <p>Optionally can be defined an own port.</p>	
Peer IP address	<p>IP address of the communication partner.</p> <p>0.0.0.0 mean arbitrary IP address is allowed.</p> <p>Valid Interval: 1.0.0.0 up to 223.255.255.255, except for: 127.x.x.x</p>	<p>Valid IP address</p> <p>Default: 0</p>
Peer port	<p>Port of the communication partner.</p> <p>Zero means an arbitrary port.</p> <p>Reserved or already occupied ports (1 up to 1024) are deprecated from the COM BS.</p>	<p>0..65535</p> <p>Default: 0</p>
Own port	<p>Own port.</p> <p>Zero means an arbitrary port.</p> <p>Reserved or already occupied ports (1 up to 1024) are deprecated from the COM BS.</p>	<p>0..65535</p> <p>Default: 0</p>
Cyclic send data	<p>Disabled:</p> <p>Cyclic sent data is disabled.</p> <p>The data exchange via this TCP connection must be programmed with function blocks.</p> <p>No cyclic E/A data may be defined.</p> <p>Activated:</p> <p>Cyclic sent data is active.</p> <p>The data for the data exchange is defined in the dialog "Signal connections" of the TCP connection.</p> <p>Receive data must be defined.</p> <p>No function blocks can operated.</p>	<p>Default: No</p>
Send Interval [ms]	<p>Only editable in case of cyclic sent data.</p> <p>Here is adjusted the send interval.</p> <p>Only steps in a grid of 10 ms are possible.</p>	<p>0..2147483647 ms</p> <p>Default: 0</p>

Status	Description	Type
Keep Alive Interval [s]	<p>The time until the connection check (provided by TCP) is activated.</p> <p>Zero deactivates the connection check.</p> <p>Within no data is exchanged in the adjusted KeepAlive Interval, KeepAlive-samples are sent to the communication partner. If the connection still exists, the KeepAlive-samples are confirmed by the communication partner.</p> <p>After ten transmitted KeepAlive-samples without a confirmation of the communication partners, the connection is deactivated.</p> <p>The shortest possible time interval for the KeepAlive-samples is 64 seconds.</p> <p>Therefore it takes at least 640 seconds until the connection is closed.</p> <p>Faster works the send timeout (can not be configured).</p> <p>The send timeout terminates a connection, when the communication partner does not confirm a transmitted data packet within 45 seconds.</p>	<p>0, 64.. 65535s</p> <p>Default: 0</p>

Table 5: Settings of the TCP connection

2.3 Context menu "Function blocks"

The context menu of the folder "Function blocks" contains the following functions:

Function blocks
New
Copy
Insert
Delete
Print
Properties

2.3.1 Menu function "New"

With the menu option *New*, a new function block is added to the TCP S/R protocol. The following function blocks are available to the user:

- Receive
- Reset
- Send
- Variable Receive
- Receive Line

2.3.2 Menu function "Print"

After selection of *Print* a standard Dialog window for printing opens.

If the customer activate the command button *OK*, all function blocks with the current connected signals are printed.

2.3.3 Context Menu of the COM Function Blocks

The Com function blocks are executed on the communication processor (COM). For further information to the COM function blocks see chapter 3.

The context menu of the COM function blocks contains the following functions:

COM Function Blocks
Connect signals
Copy
Insert
Delete
Print
Properties

2.3.4 Menu function "Connect signals"

With *Connect Signals* the dialog window "Signal Connections" opens.

In the dialog window "Signal Connections" are the three tabs "FB Inputs" "FB Outputs" and "Output Record".

The tabs "FB Input" and "FB Output" contain predefined system variables, that the customer must allocate by signals.

The tab "data" is used in all functional blocks, except "TCP_Reset". In the tab "data" must registered the signals, which are transmitted/received via the function block.

2.3.5 Menu function "Copy, Insert, Delete, Print"

Copy: Copy the function block into the clipboard including configuration.

Insert: Add a function block to a TCP S/R protocol from the clipboard.

Delete: Delete the chosen function block from the project.

Print: Print all signals and configurations of the function block.

2.3.6 Menu function "Properties"

Select *Properties* to open the dialog "Properties". The name of the function Block can be changed here.

2.4 Status and error codes

The status and the error codes can be readout from the signal "status" (see Table 4) or from the output "A_State" of a function block (e.g. Table 12). The error codes of the function blocks (e.g. „16xx8x“) can only readout in “A_State” of the TCP S/R function blocks.

In the two low bytes, the error code of the last operation is indicated.

In the two high bytes, the current protocol state is indicated.

Error code	Description
16#xx00	OK
16#xx23	Operation is blocked
16#xx30	Address already in use
16#xx32	Network is down
16#xx35	Software caused connection abort
16#xx36	Connection reset by peer
16#xx37	No buffer space available
16#xx3C	Operation time out
16#xx3D	Connection refused
16#xx41	No route to peer host
16#xxFF	Connection closed by peer

Table 6: Error code of the TCP connection

Error code	Description
16#xx81	Unknown connection Id
16#xx82	Illegal length
16#xx83	Only cyclic data are allowed by this connection
16#xx84	Connection is not available
16#xx85	Timeout-Value is oversized
16#xx86	Fatal program error
16#xx87	Internal configuration error
16#xx88	Data do not match to the configured data structure
16#xx89	Program is stopped
16#xx8A	Timeout is lapsed
16#xx8B	An other function block is already active

Table 7: Error code table of the function blocks

Protocol state	Description
16#00xx	Connection OK
16#01xx	Connection closed
16#02xx	Server waits for connection set up
16#04xx	Client try to setup a connection
16#08xx	Connection is blocked

Table 8: Protocol state of the TCP connection

3 TCP S/R Function Block

If the cyclical data transfer is too inflexible, data can also be transmitted and received with the TCP S/R function blocks. The option "Cyclic send data" must be disabled in the used TCP connection.

With the TCP S/R function blocks, the customer can match the data transfer via TCP/IP optimally to the requirements of his project.

The function blocks are parameterized in the user program. So that the functions (Send, Receive, Reset) of the *HiMatrix* control device can be set and evaluated in the user program.

Note	TCP S/R Function blocks are needed only for the acyclic data exchange. For the cyclic data exchange between server and client these function blocks are not necessary!
-------------	--

The following function blocks are available:

Function Block	Description of the function
TCP_Receive	Receiving of data packets with fixed length
TCP_ReceiveLine	Receiving of an ASCII-line
TCP_ReceiveVar	Receiving of data packets with variable length (with length field)
TCP_Reset	Reset of a TCP connection
TCP_Send	Sending of data

Table 9: Description of the function blocks

3.1 Mode of Operation of the Function Blocks

TCP S/R function blocks are executed as well by the communication processor (COM) as by the CPU of a *HIMatrix* control device and therefore must be created twice.

COM function blocks are created in the hardware management of ***ELOP II Factory*** in the structure tree.

CPU function blocks are created in the project management of ***ELOP II Factory*** in the application program. CPU function blocks are used to access COM function blocks.

The data exchange between COM- and CPU function blocks is performed via signals, that are defined by the user in the signal editor and have to be connected to the inputs/outputs of the function blocks via *Connect Signals* by drag&drop.

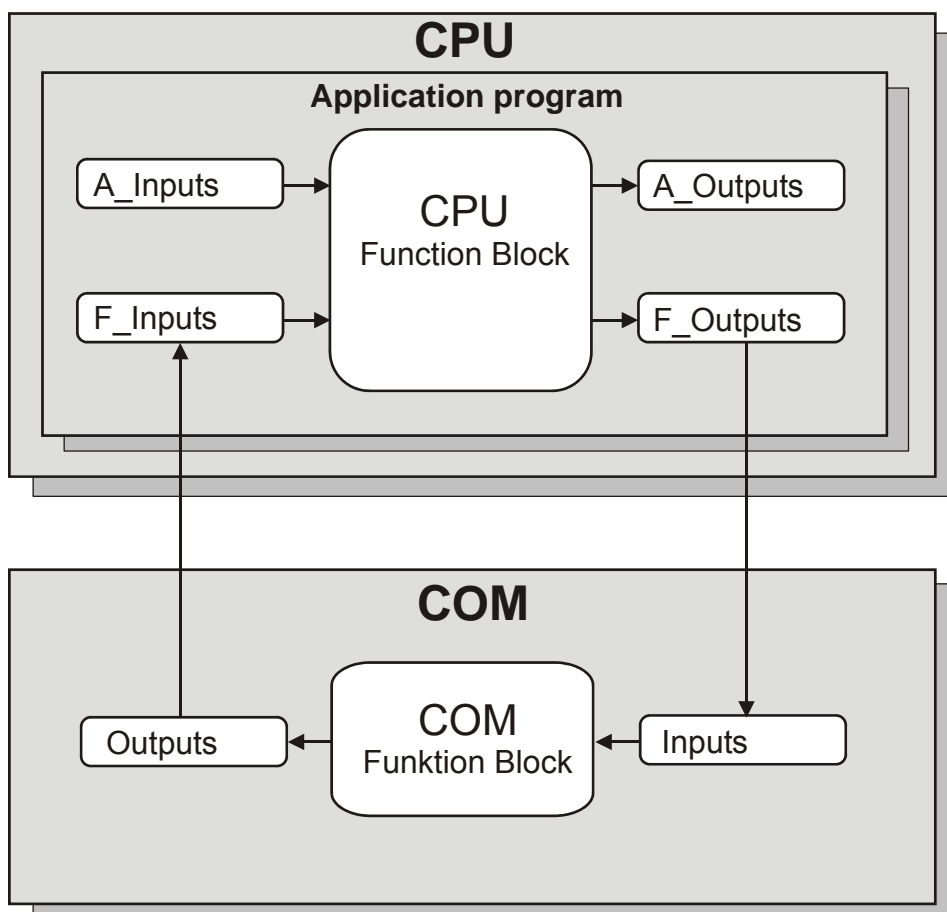


Figure4: Communication between the CPU-FB and the COM-FB

CPU Function Blocks

The CPU function blocks can be found within the project management in the folder "TCPLib" and are copied into the application program the same way as standard function blocks by drag&drop.

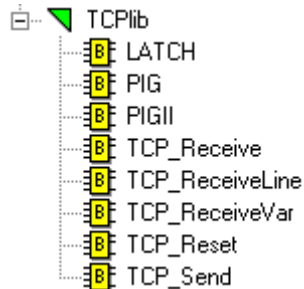


Figure5: CPU Function Blocks

COM Function Blocks

All function blocks of the TCP S/R communication are shown by selection of *Function Blocks->New* in the structure tree of the hardware-management.

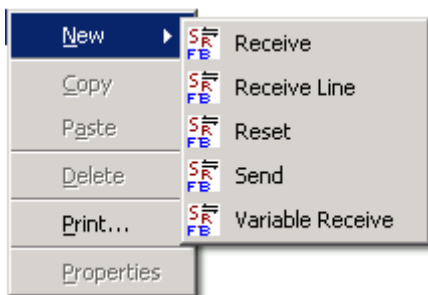


Figure6: COM Function Blocks

The CPU and COM function blocks are assigned to each other as follow:

CPU Function Block	-> COM Function Block
TCP_Receive	-> Receive
TCP_ReceiveLine	-> Receive Line
TCP_ReceiveVar	-> Variable Receive
TCP_Reset	-> Reset
TCP_Send	-> Send

3.2 Function Block "TCP_Receive"

With the function block "TCP_Receive", defined signals can be received from the communication partner.

Note All signals for the CPU and COM function block "TCP_Receive" must be created in the signal editor of the hardware management. The signals are then inserted into the application program by drag&drop. It is recommended to name the signals suiting to the inputs/outputs of the function block "TCP_Receive".

3.2.1 CPU Function Block "TCP_Receive"

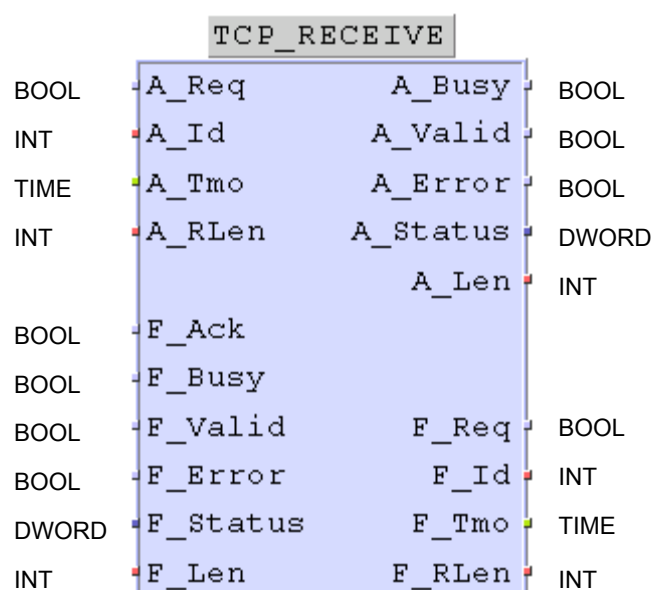


Figure7: CPU Function Block "TCP_Receive"

"A_xxx"-Inp.	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 from which should be received the data.	INT
A_Tmo	Receive timeout. If within the timeout no data is received, the function block terminates with an error report. When the input "A_Tmo" is not used, or set to zero, the timeout is deactivated.	TIME
A_RLen	A_RLen is the expected length of the signals to be received in bytes. A_RLen must be larger than zero and may not end inside of a signal.	INT

Table 10: "A_xxx"-Inputs of the CPU Function Block "TCP_Receive"

"F_xxx"-Inp.	Description	Type
F_Ack	These inputs must be connected with the corresponding output signals of the COM function block (see Table 15).	BOOL
F_Busy		BOOL
F_Valid		BOOL
F_Error		BOOL
F_Status		DWORD
F_Len		INT

Table 11: "F_xxx"-Inputs of the CPU Function Block "TCP_Receive"

"A_xxx"-Outp.	Description	Type
A_Busy	TRUE: The receiving of the data is still not ended.	BOOL
A_Valid	TRUE: The receiving of the data ended without error.	BOOL
A_Error	TRUE: An error occurred FALSE: No error	BOOL
A_Status	At the output "A_Status" are issued the status and error code of the function block and the TCP-connection. (See 2.4 Status and error codes)	DWORD
A_Len	Number of received bytes.	INT

Table 12: "A_xxx"-Output of the CPU Function Block "TCP_Receive".

"F_xxx"-Outp.	Description	Type
F_Req	These outputs must be connected with the corresponding input signals of the COM function block (see Table 14).	BOOL
F_Id		DWORD
F_Tmo		INT
F_RLen		INT

Table 13: "F_xxx"-Outputs of the CPU Function Block "TCP_Receive"

3.2.2 Sequence of operation

For the operation of the CPU function block "TCP_Receive" the following steps are required:

Note	The receive signals must be created in the tab "Record" from COM function block "Receive". The offsets and types of the receive signals must be identical with the offsets and types of the sent signals of the communication partner.
-------------	--

1. Set the identification number of the TCP connection at the input "A_Id", in the user program.
2. Set the receive timeout at the input "A_Tmo" in the user program.
3. Set the expected length of the signals to be received at the input "A_RLen", in the user program.
4. Set the input "A_Req" on TRUE in the user program.

Note	The rising edge at "A_Req" starts the function block. The function block "TCP_Receive" is now ready to receive.
-------------	---

5. The output "A_Busy" is TRUE until the signals were received, or the receive timeout lapsed. Subsequently the outputs "A_Busy" change on FALSE and "A_Valid" or "A_Error" on TRUE.
6. If the receive of the signals is proper, the output "A_Valid" changes to TRUE. The signals which were defined in the register "Record" can be evaluated.
7. If the receive of the signals is fault, the output "A_Error" changes to TRUE, and at output "A_Status" an error code is issued.

3.2.3 COM Function Block "Receive"

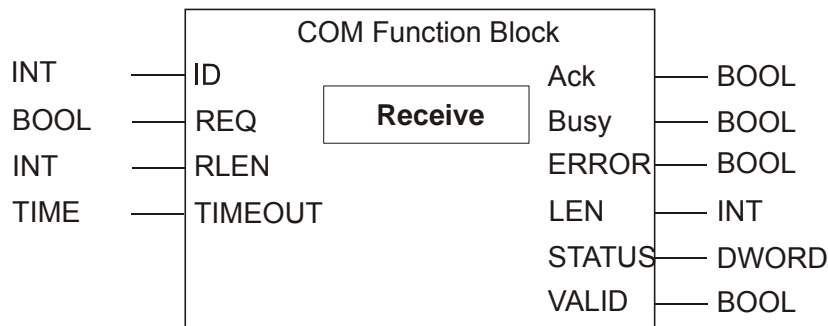


Figure8: Signal allocation of the COM Function Block "Receive"

The COM function block "Receive" is the counterpart to the CPU function block "TCP_Receive" and communicates with this via signals.

The signals for the COM function block "Receive" are created while configuring the CPU function block "TCP_Receive" for the "F_Inputs" and "F_Outputs" in the signal editor.

Note The representation of the COM function block "Receive" in the Figure8 is not visible in the hardware management and is only used for illustrating the function.

The inputs of the COM function block must be connected with the "F_Outputs" of the CPU function block using signals.

Inputs	Description	Type
ID	Identification number of the configured TCP connection to the communication partner, from which should be received the data.	INT
REQ	The rising edge starts the function block.	BOOL
RLEN	A_RLen is the expected length of the signals to be received in bytes. A_RLen must be larger than zero and may not end inside of a signal.	INT
TIMEOUT	Receive timeout Within the timeout, no data received, the function block terminated with an error report. When the input "A_Tmo" is not used, or set to zero, the timeout is deactivated.	TIME

Table 14: Inputs of the COM Function Block "Receive"

The outputs of the COM function block must be connected to the "F_Inputs" of the CPU function block using signals.

Outputs	Description	Type
Ack	TRUE: Output signals valid FALSE: Output signals invalid	BOOL
Busy	TRUE: The receiving of the data is still not ended.	BOOL
ERROR	TRUE: An error occurred FALSE: No error	BOOL
LEN	Number of received bytes.	INT
STATUS	At the output "A_Status" are issued the status and error code of the function block and the TCP-connection. (See 2.4 Status and error codes)	DWORD
VALID	TRUE: The receiving of the data ended without error.	BOOL

Table 15: Outputs of the COM Function Block "Receive"

Record	Description
Receive signals	In the tab "Record" arbitrary signals can be created. The offsets and types of the receive signals must be identical with the offsets and types of the send signals of the communication partner.

Table 16: The receive signals must created In the tab "Record".

3.3 Function Block "TCP_Receive_Line"

The function block "TCP_Receive_Line" is used for the receive of an ASCII character string including LineFeed (16#0A) from a communication partner.

Note All signals for the CPU and COM function block "TCP_Receive_Line" must be created in the signal editor of the hardware management. The signals are then inserted into the application program by drag&drop. It is recommended to name the signals suiting to the inputs/outputs of the function block "Receive_Line".

3.3.1 CPU Function Block "TCP_Receive_Line"

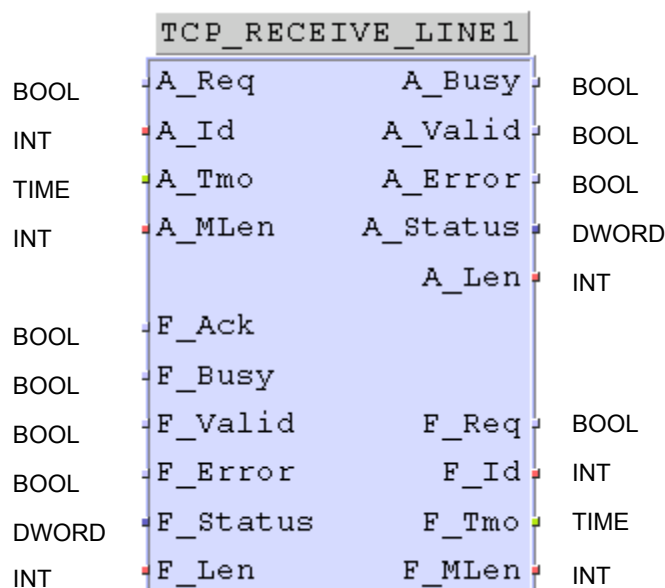


Figure9: CPU Function Block "TCP_Receive_Line"

"A_xxx"-Inp.	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, from which should be received the data.	INT
A_Tmo	Receive timeout Within the timeout, no data received, the function block terminated with an error report. When the input "A_Tmo" is not used, or set to zero, the timeout is deactivated.	TIME

"A_xxx"-Inp.	Description	Type
A_MLen	<p>"A_MLen" is the maximum length of a received line in bytes.</p> <p>The received signals must created in the tab "Record" in the COM function block. Or actual received from the communication partner.</p> <p>Transmitted bytes = Min (A_MLen, line length, length of the data area).</p>	INT

Table 17: "A_xxx"-Inputs of the CPU Function Block "TCP_Receive_Line"

"F_xxx"-Inp.	Description	Type
F_Ack	These inputs must be connected with the corresponding output signals of the COM function block (see Table 22)	BOOL
F_Busy		BOOL
F_Valid		BOOL
F_Error		BOOL
F_Status		DWORD
F_Len		INT

Table 18: "F_xxx"-Inputs of the CPU Function Block "TCP_Receive_Line"

"A_xxx"-Outp.	Description	Type
A_Busy	TRUE: The reception of the data is still not ended.	BOOL
A_Valid	TRUE: The receiving of the data ended without error.	BOOL
A_Error	TRUE: An error occurred FALSE: No error	BOOL
A_Status	At the output "A_Status" are issued the status and error code of the function block and the TCP-connection. (See 2.4 Status and error codes)	DWORD
A_Len	Number of received bytes.	INT

Table 19: "A_xxx"-Outputs of the CPU Function Block "TCP_Receive_Line"

"F_xxx"-Outp.	Description	Type
A_Req	These outputs must be connected with the corresponding input signals of the COM function block (see Table 21).	BOOL
A_Id		INT
A_Tmo		TIME
A_MLen		INT

Table 20: "F_xxx"-Outputs of the CPU Function Block "TCP_Receive_Line"

3.3.2 Sequence of operations

For the operation of the CPU function block "TCP_Receive_Line" the following steps are required:

Note	The receive signals must be created in the tab "Record" from COM function block "Receive_Line". The offsets and types of the receive signals must be identical with the offsets and types of the send signals of the communication partner.
-------------	---

1. Set the identification number of the TCP connection at the input "A_Id", in the user program.
2. Set the receive timeout at the input "A_Tmo", in the user program.
3. Set the expected maximum length of the line to be received at the input "A_MLen", in the user program.

Note	<p>A_MLen must be larger than zero and determined the size of the receive buffer in byte.</p> <p>If the receive buffer is filled, and no end of line has occurred, the read operation is ended without a error report.</p> <p>At the output "A_Len" the value of the received bytes is available:</p> <p>Received bytes = Min (A_MLen, line length, length of the data area)</p>
-------------	---

4. Set the input "A_Req" on TRUE, in the user program.

Note	The rising edge at "A_Req" starts the function block. The function block "Receive_Line" is now ready to receive.
-------------	--

5. The output "A_Busy" is TRUE until

- the signals are received, or
- Linefeed is received, or
- the receive timeout is lapsed.

Subsequently the outputs "A_Busy" change on FALSE and "A_Valid" or "A_Error" change on TRUE.

6. If the receive of the line is proper, the output "A_Valid" changes to TRUE. The signals which were defined in the tab "Record" can be evaluated.
7. If the receive of the line is fault, the output "A_Error" changes to TRUE, and at the output "A_Status" an error code is issued.

3.3.3 COM Function Block "Receive_Line"

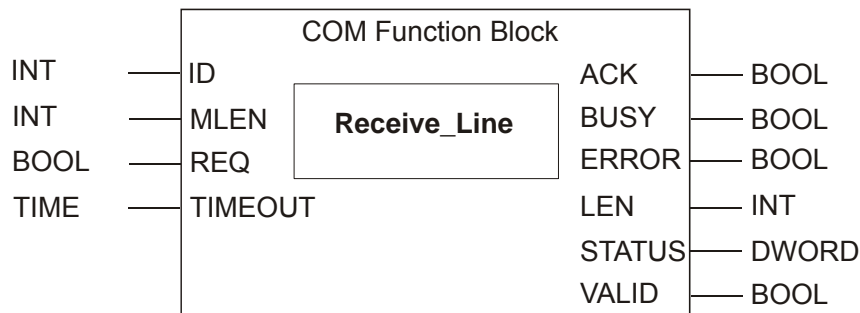


Figure10: COM Function Block "Receive_Line"

The COM function block "Receive Line" is the counterpart to the CPU function block "TCP_Receive_Line" and communicates with this via signals.

The signals for the COM function block "Receive_Line" are created while configuring the CPU function block "TCP_Receive_Line" for the "F_Inputs" and "F_Outputs" in the signal editor.

Note The representation of the COM function block "Receive Line" in the Figure above is not visible in the hardware management and is only used for illustrating the function.

The inputs of the COM function block must be connected with the "F_Outputs" of the CPU function block using signals.

Inputs	Description	Types
ID	Identification number of the configured TCP connection to the communication partner, from which should be received the data.	INT
MLEN	Maximum length of a receive line in bytes. "MLEN" must be larger than zero and may not end right in a signal.	INT
REQ	The rising edge starts the function block.	BOOL
TIMEOUT	Receive timeout If within the timeout no data is received, the function block terminates with an error report. When the input "A_Tmo" is not used, or set to zero, the timeout is deactivated.	TIME

Table 21: Inputs of the COM Function Block "Receive_Line"

The outputs of the COM function block must be connected to the "F_Inputs" of the CPU function block using signals.

Outputs	Description	Type
ACK	TRUE: Output signals valid FALSE: Output signals invalid	BOOL
BUSY	TRUE: The receiving of the data is still not ended.	BOOL
ERROR	TRUE: An error occurred FALSE: No error	BOOL
LEN	Number of received bytes.	INT
STATUS	At the output "A_Status" are issued the status and error code of the function block and the TCP-connection. (See 2.4 Status and error codes)	DWORD
VALID	TRUE: The receiving of the data ended without error.	BOOL

Table 22: Outputs of the COM Function Block "Receive_Line"

Data	Description
Receive signals	In the tab "Record" arbitrary signals can be created. The offsets and types of the receive signals must be identical with the offsets and types of the send signals of the communication partner.

Table 23: The receive signals must created In the tab "Record".

3.4 Function Block "TCP_Receive_Var"

With this function block, data packets of variable length that are equipped with a length field can be evaluated.

The received data packets must possess the structure (e.g. Modbus protocol), represented in Figure11. The adaptation to an arbitrary protocol must be done by the adjustments of the input parameters "A_LfPos, A_LfLen, A_LfFac, A_LfLen".

The received data packets consist of a header and a data field. The header consists of data like address of the communication partner, function of the telegram, length field, etc., which are required for the connection. To evaluate the data field, the header must be separated and the length field must be selected.

The size of the header must be registered in the parameter "A_LfAdd".

The length of the data field must be read out from the length field of the current data packet. The position of the length field is registered in the parameter "A_LfPos". The size in byte of the length field must be registered in "LfLen".

If the length is not registered in byte, then a conversion factor must be registered in "A_LfFac" (e.g. "2" for Word or "4" for Double Word).

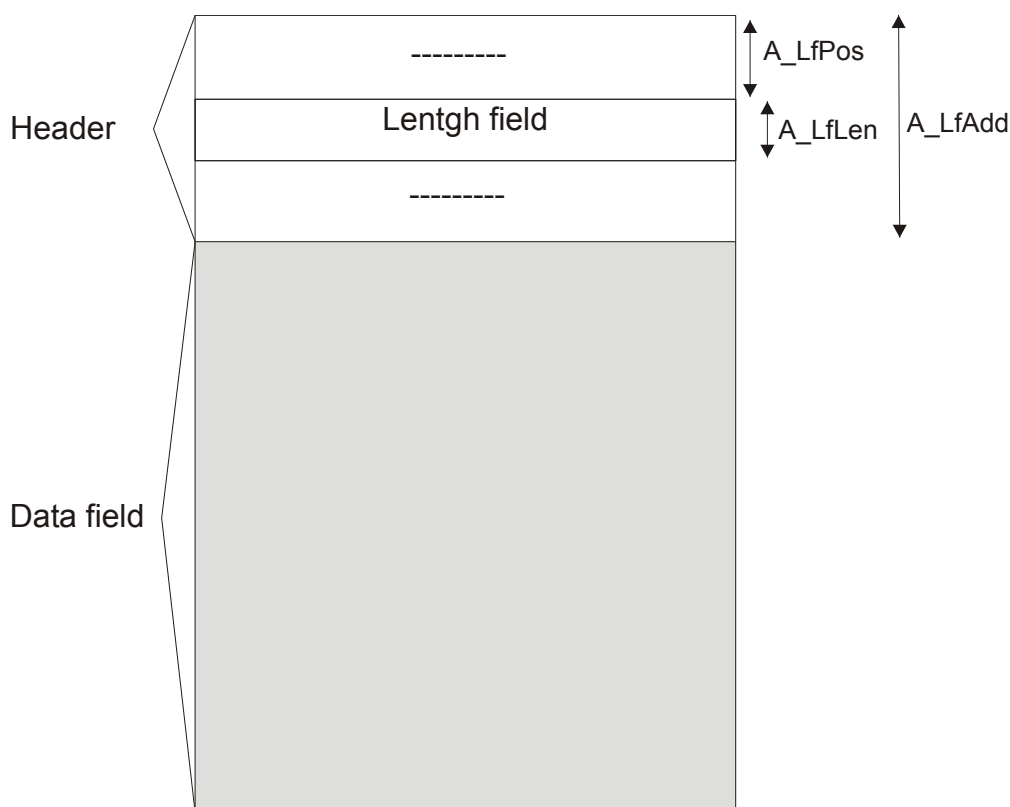


Figure11: Construction of the record

Note All signals for the CPU and COM function block "TCP_Receive_Var" must be created in the signal editor of the hardware management. The signals are then inserted into the application program by drag&drop. It is recommended to name the signals suiting to the inputs/outputs of the function block "Variable Receive".

3.4.1 CPU Function Block "TCP_Receive_Var"

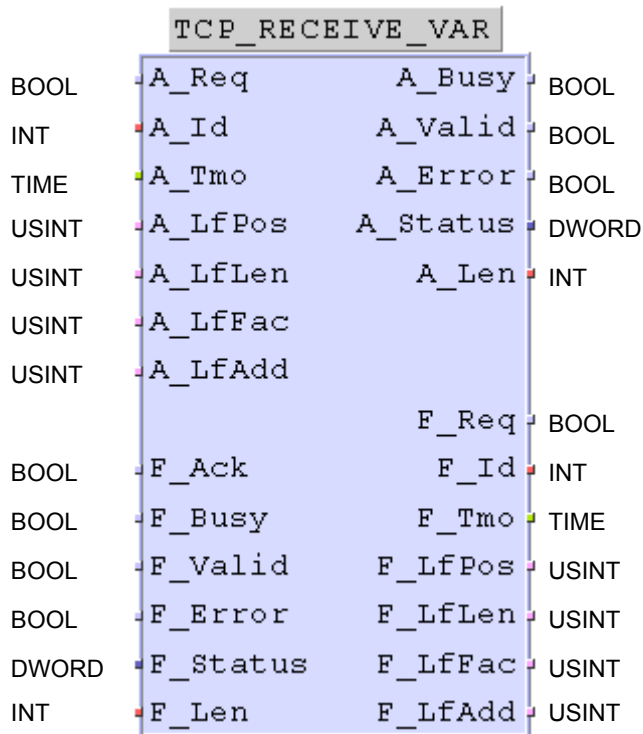


Figure12: CPU Function Block "TCP_Receive_Var"

"A_xxx"-Inp.	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, from which should be received the data packet.	DWORD
A_Tmo	Receive timeout If within the timeout, no data is received, the function block terminates with an error report. When the input "A_Tmo" 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 length field in bytes. Allowed are 1, 2 or 4 bytes.	USINT
A_LfFac	Conversion factor to bytes, if the setting in the length field is not in bytes. If the Input is not used, or assigned with zero, "1" is taken as default value.	USINT
A_LfAdd	Size of the header field in bytes.	USINT

Table 24: "A_xxx"-Inputs of the CPU Function Block "TCP_Receive_Var"

"F_xxx"-Inp.	Description	Type
F_Ack	These inputs must be connected with the corresponding output signals of the COM function block (see Table 29).	BOOL
F_Busy		BOOL
F_Valid		BOOL
F_Error		BOOL
F_Status		DWORD
F_Len		INT

Table 25: "F_xxx"-Inputs of the CPU Function Block "TCP_Receive_Var"

"F_xxx"-Outp.	Description	Type
F_Req	These outputs must be connected with the corresponding input signals of the COM function block (see Table 28).	BOOL
F_Id		INT
F_Tmo		TIME
F_LfPos		USINT
A_LfLen		USINT
A_LfFac		USINT
A_LfAdd		USINT

Table 26: "F_xxx"-Outputs of the CPU Function Block "TCP_Receive_Var"

"A_xxx"-Outp.	Description	Type
A_Busy	TRUE: The receiving of the data is still not ended.	BOOL
A_Valid	TRUE: The receiving of the data ended without error.	BOOL
A_Error	TRUE: An error occurred FALSE: No error	BOOL
A_Status	At the output "A_Status" are issued the status and error code of the function block and the TCP-connection. (See 2.4 Status and error codes)	DWORD
A_Len	Number of received bytes	INT

Table 27: "A_xxx"-Outputs of the CPU Function Block "TCP_Receive_Var"

3.4.2 Sequence of operations

For the operation of the CPU function block "TCP_Receive_Var" the following steps are required:

Note	The receive signals must be created in the tab "Record" from COM function block "Variable Receive". The offsets and types of the receive signals must be identical with the offsets and types of the sent signals of the communication partner.
-------------	---

1. Set the identification number of the TCP connection at the input "A_Id", in the user program.
2. Set the receive timeout at the input "A_Tmo" in the user program.
3. Set the parameters A_LfPos, A_LfLen, A_LfFac and A_LfAdd in the user program.
4. Set the input "A_Req" in the user program to TRUE.

Note	The rising edge at "A_Req" starts the function block. The function block "TCP_Receive_Var" is now ready to receive.
-------------	---

5. The output "A_Busy" goes for so long onto TRUE until the signals were obtained. Subsequently the outputs "A_Busy" change on FALSE and "A_Valid" or "A_Error" change on TRUE.
6. If the receive of the line is proper, the output "A_Valid" change on TRUE. The signals which were defined in the tab "Record" can be evaluated.
The output "A_Len" contains the number of the bytes which were readout.
7. If the receive of the line is fault, the output "A_Error" changes to TRUE, and at the output "A_Status" an error code is issued.

3.4.3 COM Function Block "Variable_Receive"

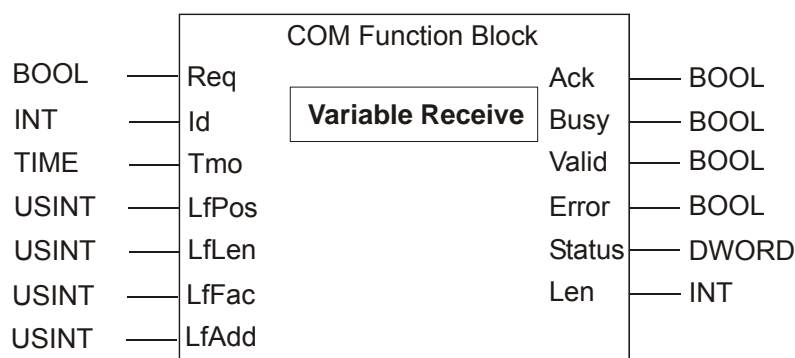


Figure13: COM Function Block "Variable_Receive"

The COM function block "Variable Receive" is the counterpart to the CPU function block "TCP_Receive_Var" and communicates with this via signals.

The signals for the COM function block "Variable Receive" are created while configuring the CPU function block "TCP_Receive_Var" for the "F_Inputs" and "F_Outputs" in the signal editor.

Note The representation of the COM function block "Variable Receive" in the Figure above is not visible in the hardware management and is only used for illustrating the function.

The inputs of the COM function block must be connected with the "F_Outputs" of the CPU function block using signals.

Inputs	Description	Type
ID	Identification number of the configured TCP connection to the communication partner, from which the data should be received.	INT
LfAdd	Size of the header field in bytes.	USINT
LfFac	Conversion factor to bytes, if the setting in the length field is not in bytes. If the Input is not used, or assigned with zero, "1" is taken as default value.	USINT
LfLen	Size of the length field in bytes. Allowed are 1, 2 or 4 bytes.	USINT
LfPos	Start position of the length field in the data packet. The numbering begins with zero (measured in bytes).	USINT
REQ	The rising edge starts the function block.	BOOL

Inputs	Description	Type
TIMEOUT	Receive timeout If within the timeout no data is received, the function block terminates with an error report. When the input "A_Tmo" is not used, or set to zero, the timeout is deactivated.	TIME

Table 28: Inputs of the COM Function Block "TCP_Receive_Var"

The outputs of the COM function block must be connected to the "F_Inputs" of the CPU function block using signals.

Outputs	Description	Type
ACK	TRUE: Output signals valid FALSE: Output signals invalid	BOOL
BUSY	TRUE: The receiving of the data is still not ended.	BOOL
ERROR	TRUE: An error occurred FALSE: No error	BOOL
LEN	Number of received bytes.	INT
STATUS	At the output "A_Status" are issued the status and error code of the function block and the TCP-connection. (See 2.4 Status and error codes)	DWORD
VALID	TRUE: The receiving of the data ended without error.	BOOL

Table 29: Outputs of the COM Function Block "TCP_Receive_Var"

Data	Description
Receive signal	In the tab "Record" arbitrary signals can be created. The offsets and types of the receive signals must be identical with the offsets and types of the send signals of the communication partner.

Table 30: The receive signals must created In the tab "Record".

3.5 Function Block "TCP_Reset"

With this function block a faulty connection can reestablished, when a send or receive function block reports itself with a timeout error (16#8A).

Note All signals for the CPU and COM function block "TCP_Reset" must be created in the signal editor of the hardware management. The signals are then inserted into the application program by drag&drop.

It is recommended to name the signals suiting to the inputs/outputs of the function block "Reset".

3.5.1 CPU Function Block "TCP_Reset"

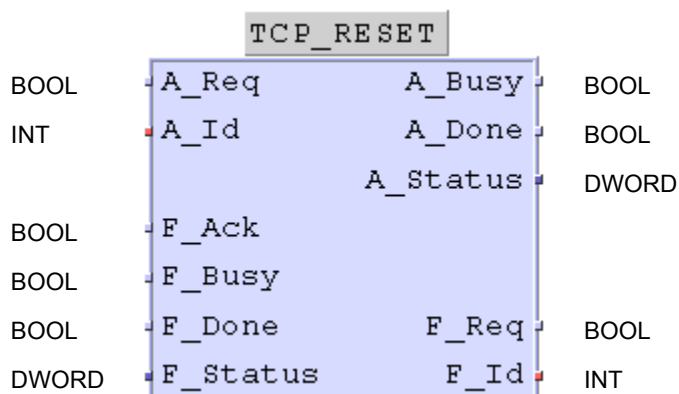


Figure14: CPU Function Block "TCP_Reset"

"A_xxx"-Inp.	Description	Type
A_Req	The rising edge starts the function block.	BOOL
A_Id	Identification number of the faulty TCP connection, which should be reset.	INT

Table 31: "A_xxx"-Inputs of the CPU Function Blocks "TCP_Reset"

"F_xxx"-Inp.	Description	Type
F_Ack	These inputs must be connected with the corresponding output signals of the COM function block (see Table 36)	BOOL
F_Busy		BOOL
F_Done		BOOL
F_Status		DWORD

Table 32: "F_xxx"-Inputs of the CPU Function Block "TCP_Reset"

"A_xxx"-Outp.	Description	Type
A_Busy	TRUE: The reset of the TCP connection is still not ended.	BOOL
A_Done	TRUE: The sending ended without error.	BOOL
A_Status	At the output "A_Status" are issued the status and error code of the function block and the TCP-connection. (See 2.4 Status and error codes)	DWORD

Table 33: "A_xxx" Output of the CPU Function Block "TCP_Reset"

"F_xxx"-Outp.	Description	Type
F_Req	These outputs must be connected with the corresponding input signals of the COM function block (see Table 35).	BOOL
F_Id		DWORD

Table 34: "F_xxx"-Outputs of the CPU Function Block "TCP_Reset"

3.5.2 Sequence of operations

For the operation of the CPU function block "TCP_Reset" the following steps are required:

1. Set the identification number of the TCP connection at the input "A_Id", in the user program.
2. Set the input "A_Req" on TRUE in the user program.

Note The function block starts with a rising edge at "A_Req".

3. The output "A_Busy" changes to TRUE until a reset is transmitted to the defined TCP connection. After that, the outputs "A_Busy" changes to FALSE and "A_Done" to TRUE.

3.5.3 COM Function Block "Reset"

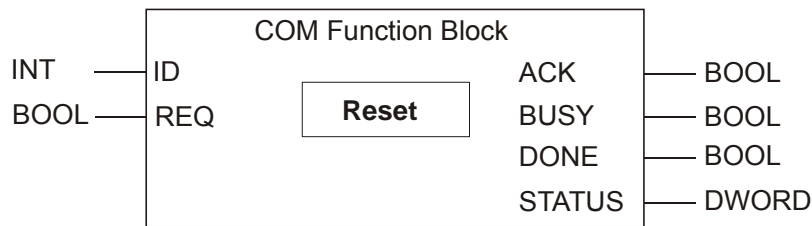


Figure15: COM Function Block "Reset"

The COM function block "Reset" is the counterpart to the CPU function block "TCP_Reset" and communicates with this via signals.

The signals for the COM function block "Receive" are created while configuring the CPU function block "Reset" for the "F_Inputs" and "F_Outputs" in the signal editor.

Note The representation of the COM function block "Reset" in the Figure above is not visible in the hardware management and is only used for illustrating the function.

The inputs of the COM function block must be connected with the "F_Outputs" of the CPU function block using signals.

Inputs	Description	Type
ID	Identification number of the configured TCP connection to the communication partner, from which the data should be received.	DWORD
REQ	The rising edge starts the function block.	BOOL

Table 35: Inputs of the COM Function Block "Reset"

The outputs of the COM function block must be connected to the "F_Inputs" of the CPU function block using signals.

Outputs	Description	Type
ACK	TRUE: Output signals valid FALSE: Output signals invalid	BOOL
BUSY	TRUE: CPU function block waits for the feedback from the COM functional function block that the RESET was transmitted.	BOOL
DONE	TRUE: The reset signal is transmitted.	BOOL
STATUS	At the output "A_Status" are issued the status and error code of the function block and the TCP-connection. (See 2.4 Status and error codes)	DWORD

Table 36: Outputs of the COM Function Block "TCP_Reset"

3.6 Function Block "TCP_Send"

The function block "TCP_Send" is used for the acyclic transmitting of signals to a communication partner. Inside the communication partner must be configured a function block e.g. "Receive" with the same signals and the same offsets.

Note All signals for the CPU and COM function block "TCP_Send" must be created in the signal editor of the hardware management. The signals are then inserted into the application program by drag&drop.

It is recommended to name the signals suiting to the inputs/outputs of the function block "Send".

3.6.1 CPU Function Block "TCP_SEND"

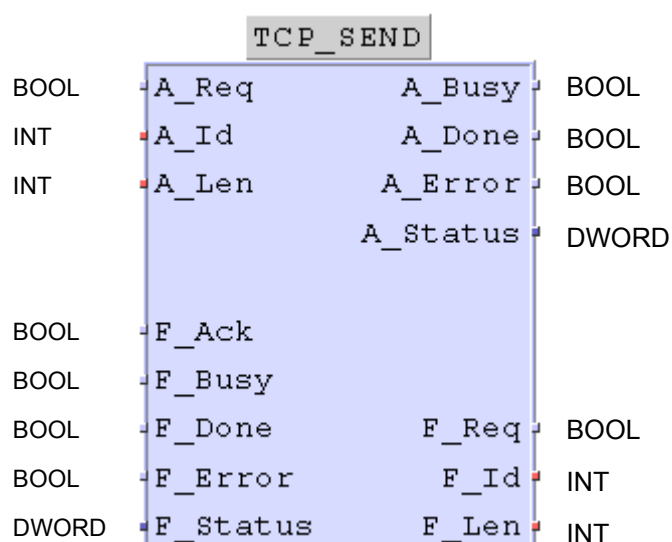


Figure16: The CPU Function Block "TCP_Send" in the user program

"A_xxx"-Inp.	Description	Typ
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 the data should be send.	INT
A_Len	Number of transmitted bytes. A_Len must be larger than zero and may not end inside of a signal.	INT

Table 37: "A_xxx"-Inputs of the CPU Function Block "TCP_Send"

"F_xxx"-Inp.	Description	Type
F_Ack	These inputs must be connected with the corresponding output signals of the COM function block (see Table 42).	BOOL
F_Busy		BOOL
F_Done		BOOL
F_Error		BOOL
F_Status		DWORD

Table 38: "F_xxx"-Inputs of the CPU Function Block "TCP_Send"

"A_xxx"-Outp.	Description	Type
A_Busy	TRUE: The transmitting of the data is still not ended.	BOOL
A_Done	TRUE: The transmitting of the data ended without error.	BOOL
A_Error	TRUE: An error occurred FALSE: No error	BOOL
A_Status	At the output "A_Status" are issued the status and error code of the function block and the TCP connection. (See 2.4 Status and error codes)	DWORD

Table 39: "A_xxx"-Outputs of the CPU Function Block "TCP_Send"

"F_xxx"-Outp.	Description	Type
F_Req	These outputs must be connected with the corresponding input signals of the COM function block (see Table 41).	BOOL
F_Id		DWORD
F_Len		INT

Table 40: "F_xxx"-Outputs of the CPU Function Block "TCP_SEND"

3.6.2 Sequence of operations

For the operation of the CPU function block "TCP_Send" the following steps are required:

Note	The transmit signals must be created in the tab "Record" from COM function block "Send". The offsets and types of the transmit signals must be identical with the offsets and types of the receive signals of the communication partner.
-------------	--

1. Set the identification number of the TCP connection at the input "A_Id", in the user program.
2. Set the length of the send signals at the input "A_Len" in the user program.
3. Set the input "A_Req" in the user program to TRUE.

Note	The rising edge at "A_Req" starts the function block. The function block "TCP_Send" is now ready to transmit.
-------------	---

4. The output "A_Busy" is TRUE until the signals were transmitted. Subsequently the outputs "A_Busy" change on FALSE and "A_Done" on TRUE.
5. If the transmission of the signals is fault, the output "A_Error" change on TRUE, and at output "A_Status" an error code is issued.

3.6.3 COM Function Block "Send"

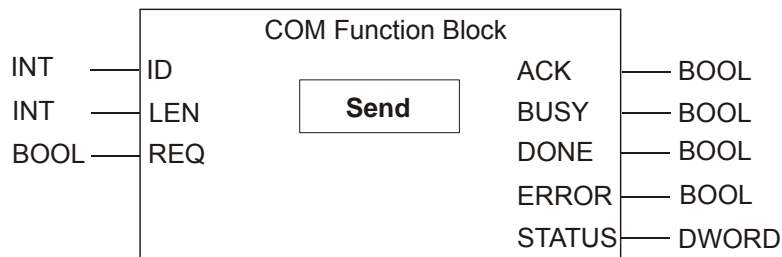


Figure17: COM Function Block "Send"

The COM function block "Send" is the counterpart to the CPU function block "TCP_Send" and communicates with this via signals.

The signals for the COM function block "Send" are created while configuring the CPU function block "TCP_Send" for the "F_Inputs" and "F_Outputs" in the signal editor.

Note The representation of the COM function block "Send" in Figure17 is not visible in the hardware management and is only used for illustrating the function.

The inputs of the COM function block must be connected with the "F_Outputs" of the CPU function block using signals.

Inputs	Description	Type
ID	Identification number of the configured TCP connection to the communication partner, to which should be send the data.	DWORD
LEN	Number of transmitted bytes "LEN" must be larger than zero and may not end inside of a signal.	INT
REQ	The rising edge starts the function block.	BOOL

Table 41: Inputs of the COM Function Block "Send"

The outputs of the COM function block must be connected to the "F_Inputs" of the CPU function block using signals.

Outputs	Description	Type
ACK	TRUE: Output signals valid FALSE: Output signals invalid	BOOL
BUSY	TRUE: The transmit of the data is still not ended.	BOOL
Done	TRUE: The transmission process is done.	BOOL
ERROR	TRUE: An error occurred FALSE: No error	BOOL
STATUS	At the output "A_Status" are issued the status and error code of the function block and the TCP-connection. (See 2.4 Status and error codes)	DWORD

Table 42: Outputs of the COM Function Block "Send"

Data	Description
Send signals	In the tab "Output Record" arbitrary signals can be created. The offsets and types of the send signals must be identical with the offsets and types of the receive signals of the communication partner.

Table 43: The send signals must created In the tab "Output Record".

3.7 Auxiliary Function Blocks

The following auxiliary function blocks operate completely in the application program on the CPU.

The auxiliary function blocks are in the project tree in the library "TCPLib". The auxiliary function blocks are used only inside the other function blocks.

The following auxiliary function blocks are in the library "TCPLib".

Auxiliary Function Blocks	Description
LATCH	The function ID generates an identifier from four bytes.
PIG	Create with a slot-number a SLOT identification number.
PIGII	Create a up continuous identification number for the slots.

Table 44: The auxiliary function blocks and there function

Note	The signals for the auxiliary function blocks must be created in the signal editor of the hardware management. The signals must be inserted by drag&drop into the application program.
-------------	--

4 Application

4.1 Cyclic data exchange between Siemens and HIMA

In this example the protocol "Send/Receive over TCP" is installed in a HIMA *HIMatrix* F60 control device. The HIMA *HIMatrix* F60 is supposed to communicate cyclic via TCP S/R with a Siemens control device (for example SIMATIC 300).

In this example the HIMA *HIMatrix* F60 (Client) is the active station, which sets up the TCP connection to the passive Siemens SIMATIC 300 (Server). After the connection setup, both stations are equal and can send and receive at any time.

When connecting the *HIMatrix* F60 with the Siemens SIMATIC 300 following is to consider:

1. The orders described in the chapter 1.1 are required for the HIMA *HIMatrix* F60.
2. The HIMA *HIMatrix* F60 and the Siemens SIMATIC 300 are connected with each other via their Ethernet interfaces.
3. The HIMA *HIMatrix* F60 and the Siemens SIMATIC 300 must be in the same subnet, or must own the corresponding routing settings, if a router is used.

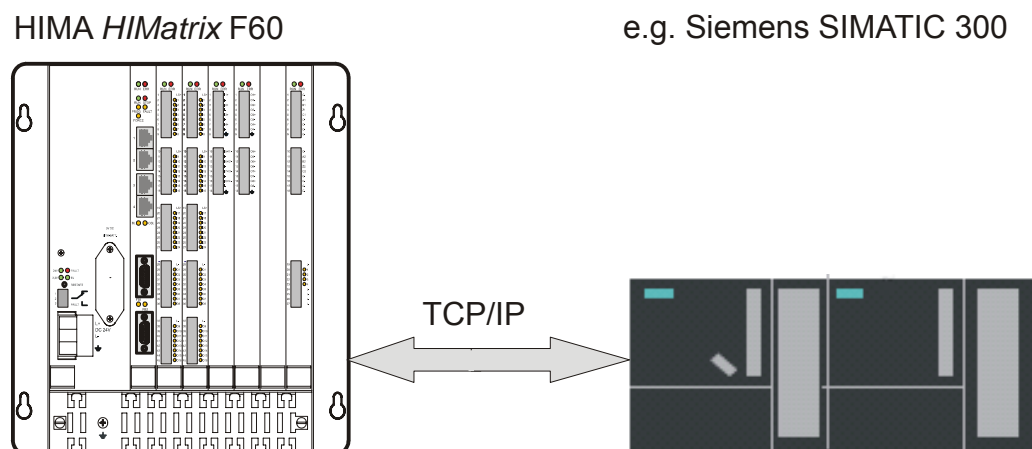


Figure18: The HIMA *HIMatrix* F60 and the Siemens SIMATIC 300 are connected via TCP/IP.

4.1.1 Configuration of the data exchange

In this application two BYTES and one WORD should send from the HIMA *HIMatrix* F60 to the Siemens SIMATIC 300. The signals are received in the Siemens SIMATIC 300 by the function block "AG_RECV" (FC 6) and transfer internal to the function block "AG_SEND" (FC 5). The Siemens SIMATIC 300 transmits the unchanged signals via the function block "AG_SEND" (FC 5) back to the HIMA *HIMatrix* F60. The user can test the transfer of the signals after the configuration with the HIMA Forceeditor.

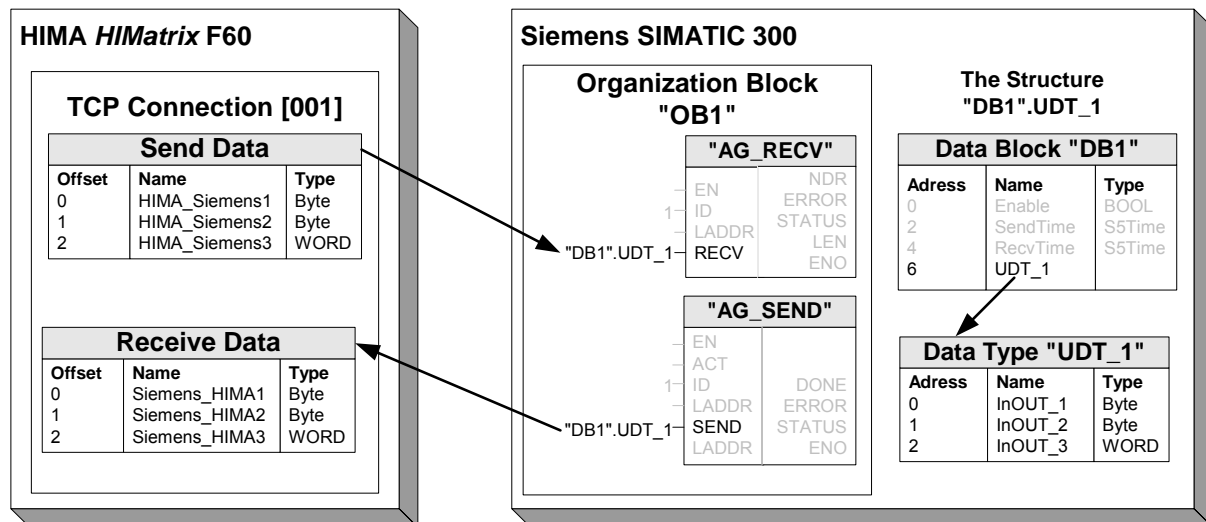


Figure19: Data exchange between HIMA and Siemens via TCP S/R

Description to the Figure19.	
HIMA <i>HIMatrix</i> F60	
TCP connection [001]	This dialog includes all settings, which are necessary for the communication with the communication partner (Siemens SIMATIC 300).
Send data	The offsets and types of the signals in the <i>HIMatrix</i> F60 must agree with the address and types of the variables in the data type „UDT_1“ of the SIMATIC 300.
Receive data	The offsets and types of the signals in the <i>HIMatrix</i> F60 must agree with the address and types of the variables in the data type „UDT_1“ of the SIMATIC 300.
Siemens SIMATIC 300	
Organization block „OB1“	The function blocks "AG_RECV" (FC6) and "AG_SEND" (FC 5) must created and configured in the organization block "OB1".
"AG_RECV" (FC 6)	The function block „AG_RECV“ (FC 6) transfer the received data from the communication partner into the data type „DB1“.UDT_1. The inputs "ID" and "LADDR" must be configured for the communication with the communication partner.

"AG_SEND" (FC 5)	<p>The function block "AG_SEND" (FC 5) transfers the data from the data type „DB1“.UDT_1 to the communication partner.</p> <p>The inputs "ID" and "LADDR" must be configured for the communication with the communication partner.</p>
Data block „DB1"	<p>The data type „UDT_1" is defined in the Data block "DB1".</p>
Data type „UDT_1"	<p>The addresses and types of the variables in the SIMATIC 300 must agree with the offsets and types of the <i>HIMatrix</i> F60. The data type „UDT_1" transfer the received user data and stores them, until the transmission to the communication partner.</p>

Table 45: Description to the Figure19.

4.1.2 Configuration the Siemens SIMATIC 300

Caution The following step instruction for the configuration of the Siemens control device does not claim to be complete.
All information are without warranty, for the project planning of the Siemens control device the documentation of Siemens is mandatory.

Step 1: Create a new Siemens SIMATIC 300 project:

- ❑ Start the Siemens SIMATIC manager.
- ❑ Select in the main menu *File->"New project" wizard* to create a new project.
- ❑ Follow the statements of the Siemens SIMATIC *wizard*.
- ❑ Configure the "Industrial Ethernet" and "MPI" Connections.

Step 2: Create the data type „UDT1” with the following variables:

- ❑ Change into the folder "function blocks" in the Siemens SIMATIC manager.
- ❑ Select in the main menu *Insert->S7 Function block->Datatype* and create a data type.
- ❑ Choose the name „UDT1" for the data type.
- ❑ Choose the Symbolic Name „UDT_1” for the data type.
- ❑ Create the following variables in the data type „UDT_1" as in Figure20.

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		

Figure20: In the data type „UDT_1" the "InOut_x" variables are created.

Note At the cyclic and acyclic data exchange is to be considered, that some control devices (e.g. SIMATIC 300) insert "Pad bytes". The "Pad bytes" guarantee, that all data types which are larger than one byte, always begin at an even offset and the overall length of the defined variables is also always even.
In such cases “dummy bytes” must be inserted at the corresponding places, in the HIMA control device (see 1.4).

Step 3:

Create the data block „DB1” for the function blocks “FC 5” and “FC 6”:

- ❑ Select in the main menu *Insert->S7 block->data block* and create a data block.
 - ❑ Choose the name „DB1” for the data block.
 - ❑ Choose the symbolic name „DB1 for the data block”.
- ❑ Allocate the datatype “UDT_1” to the data block “DB1”.
- ❑ Create the data types in the data block „DB1” as in Figure21.

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		

Figure21: The variables for the Function Blocks „AG_RECV” (FC 6) and “AG_SEND” (FC 5) are created in the data block „DB1”.

Step 4:

Create the following symbols in the symbol editor:

- ❑ Open the dialog window “KOP/AWL/FUP” with a double click onto the organization block „OB1”.
- ❑ Select in the main menu *Options->Symbol Table* and open the symbol editor.
- ❑ Complete the symbol editor with the variables as in Figure22.

S7-Programm(1) (Symbole) -- S7_COMTEST\SIMATIC 300-				
	Status	Symbol	Adresse	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				

Figure22: Create the variables M1.0 up to MW 5.

Step 5:

Create the FC function block "AG_RECV" (FC 6):

- ❑ Change to the dialog window "KOP/AWL/FUP".
- ❑ Choose one after the other:
 - ❑ one "OR gate"
 - ❑ one "S_VIMP"
 - ❑ one "AG_RECV" (FC 6)from the structure in the left part of the dialog window and move the function blocks by Drag&Drop into the organization element „OB1".
- ❑ Connect and configure the function blocks as in Figure23.
- ❑ Open the context menu with a right mouse click into the function block "AG_RECV" (FC 6).
 - ❑ Select *Properties* to open the dialog window "Properties" of the TCP connection.
 - ❑ Deactivate *active connection setup*.
 - ❑ Configure the *ports*.
 - ❑ Note the function block parameter "LADDR" and set this in the functional diagram at the function block "AG_RECV" (FC 6).

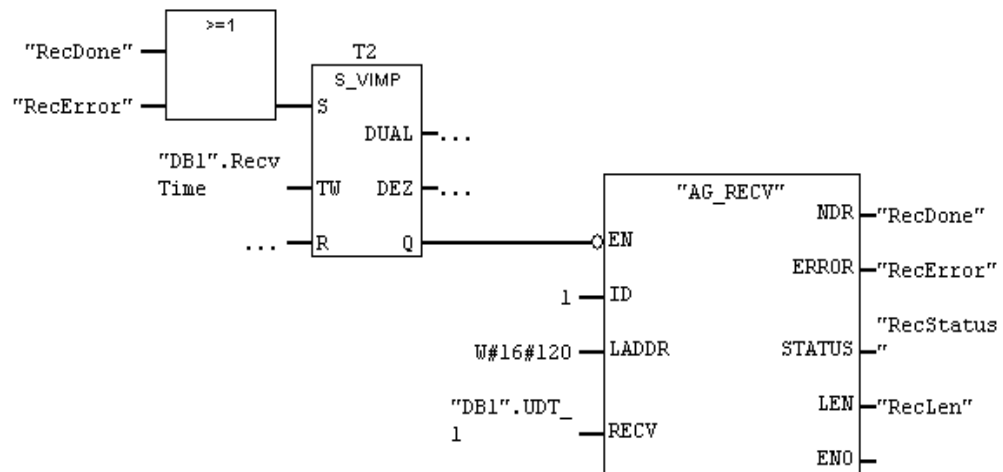


Figure23: The configuration of "AG_RECV" (FC 6)

Step 6:

Create the FC function block "AG_SEND" (FC 5):

- ❑ Change to the dialog window "KOP/AWL/FUP".
- ❑ Choose one after the other:
 - ❑ one "OR gate"
 - ❑ one "S_VIMP"
 - ❑ one "AG_SEND" (FC 5)
 from the structure in the left part of the dialog window and move the function blocks by Drag&Drop into the organization element „OB1".
- ❑ Connect and configure the function blocks as in Figure24.
- ❑ Open the context menu with a right mouse click into the function block "AG_SEND" (FC 5).
 - ❑ Select *Properties*, to open the dialog window "Properties" of the TCP connection.
 - ❑ Deactivate *active connection setup*.
 - ❑ Configure the *ports*.
 - ❑ Note the function block parameter "LADDR" and set this in the functional diagram at the function block "AG_SEND" (FC 5).

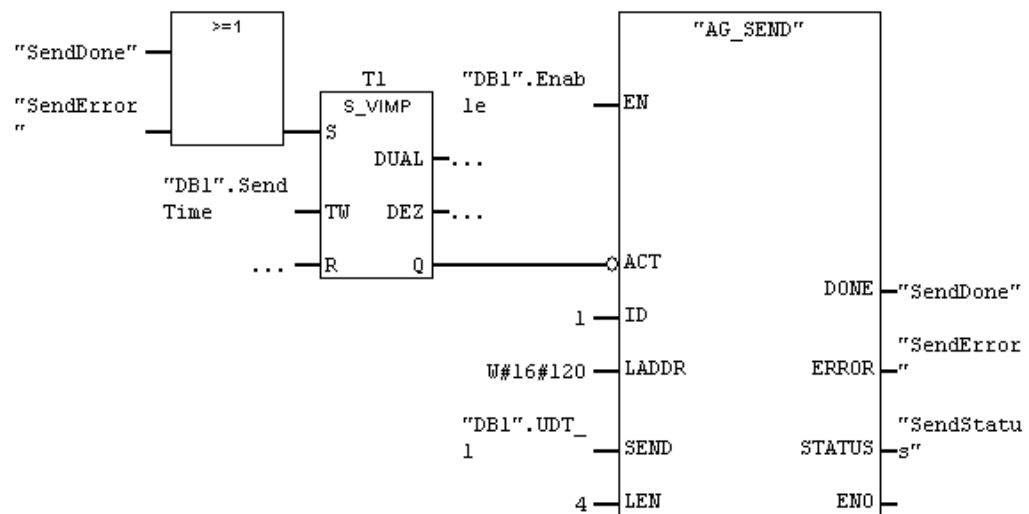


Figure24: The configuration of "AG_SEND" (FC 5)

Step 7:

Download the code into the Siemens SIMATIC 300 control device:

- ❑ Start the code generator for the program.
- ❑ Make sure that the code was generated without any error.
- ❑ Load the program into the Siemens SIMATIC 300 control device.

4.1.3 Configuration of the TCP S/R protocol in a *HIMatrix* F60

Step 1: Create a new HIMA *HIMatrix* F60 project:

- ❑ Start **ELOP II Factory**.
- ❑ Start in the project management via the main menu *Project->Project wizard...* the project wizard.
- ❑ Follow the instructions of the project wizard to create a new project.
 - ❑ Choose a name for the project (e.g. "SR_Test").
 - ❑ Choose a name for the resource (e.g. „F60"), which is created automatically with a new project.

Note For the configuration of the HIMA *HIMatrix* control devices and the programming tool **ELOP II Factory**, the manual "First steps in **ELOP II Factory**" and the online help from **ELOP II Factory** are recommended.

Step 2: Configure the resource F60:

- ❑ Change to the hardware management.
- ❑ Open the context menu of the resource F60 and change the following parameters:
 - ❑ Assign type „F60"
 - ❑ Assign "System Id [SRS]"
 - ❑ Activate "Forcing allowed".

Step 3: Create the TCP S/R protocol in the resource F60:

- ❑ Open the folder of the F60 in the structure tree.
- ❑ Click with the right mouse button on "Protocols".
- ❑ Select *New->Send/Receive over TCP*.

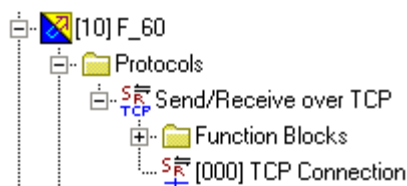


Figure25: Structure view of a new created TCP S/R protocol

Step 4:

Configure the TCP/IP connection of the resource F60:

- ❑ Select the TCP connection in the structure.
- ❑ Open the context menu of the TCP connection.
- ❑ Select *properties* to open the dialog window "Properties".
- ❑ Configure the properties as in Figure26: .
- ❑ To create more TCP connections, select *New->TCP Connection* in the context menu of "*Send/Receive over TCP*".

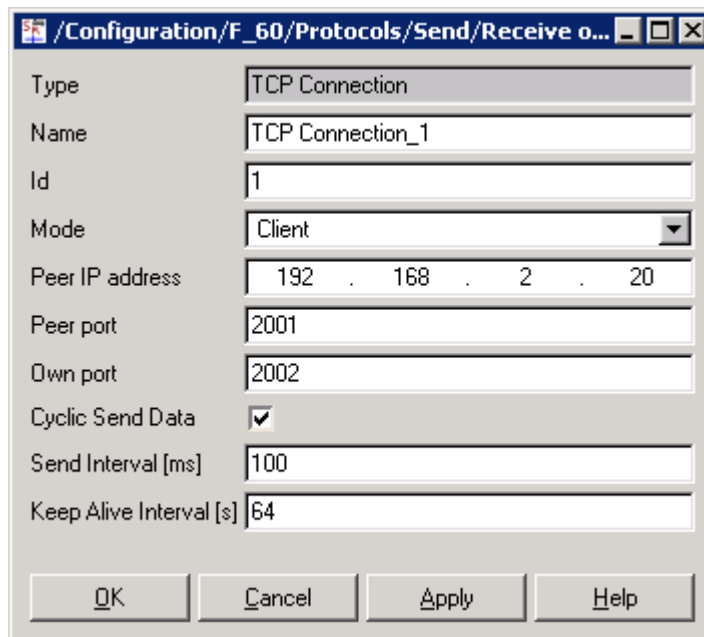


Figure26: The HIMatrix F60 is the active communication partner (Client)

Note

If cyclic data exchange should be parameterized between two control devices, the option "Cyclic data dispatch" must be activated in the dialog "Properties" of the TCP Connection.

Step 5:

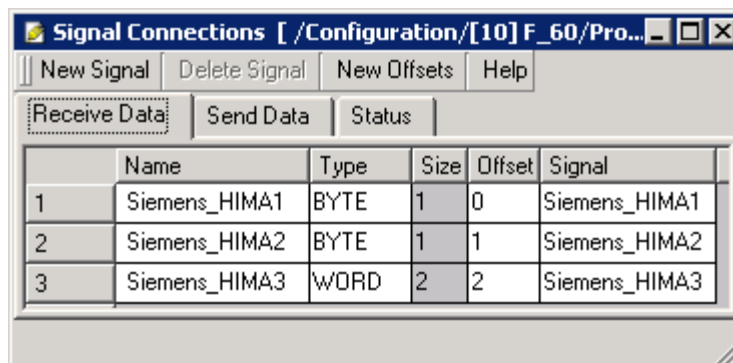
Create the following signals in the signal editor of the F60:

- ❑ Open with *Signal->Editor* the signal editor in the hardware-management.
- ❑ Create the following signals which are used for the F60 as received data:
 - ❑ „Siemens_HIMA1" from type "Byte"
 - ❑ „Siemens_HIMA2" from type "Byte"
 - ❑ „Siemens_HIMA3" from type "WORD"
- ❑ Create the following signals which are used for the F60 as transmitted data:
 - ❑ „HIMA_Siemens1" from type "Byte"
 - ❑ „HIMA_Siemens2" from type "Byte"
 - ❑ „HIMA_Siemens3" from type "WORD"

Step 6:

Connect the "Receive data" of the F60 with the SIMATIC 300 variables:

- ❑ Select the TCP connection configured in step 4 with the Id „001".
- ❑ Open the context menu and select *connect signals*.
- ❑ Open the tab *Receive data* in the dialog window "Signal Connections".
- ❑ Open the signal editor with *Signale->Editor* in the hardware management.
- ❑ In the signal editor, click on the "Name" of the signal „Siemens_HIMA1" and move the signal by Drag&Drop into the tab "Receive data" of the dialog window "Signal Connections".
- ❑ Repeat the previous step, if you have defined more than one input signal for this TCP connection.
- ❑ Click on the command button *New Offsets* in the dialog window "Signal Connections".
- ❑ In the popup window "Renumber Offsets" click on the command button *Renumber*.
- ❑ Close the dialog window.



	Name	Type	Size	Offset	Signal
1	Siemens_HIMA1	BYTE	1	0	Siemens_HIMA1
2	Siemens_HIMA2	BYTE	1	1	Siemens_HIMA2
3	Siemens_HIMA3	WORD	2	2	Siemens_HIMA3

Figure27: Receive data of the *HIMatrix* F60

Note

Consider that the offsets of the signals in the *HIMatrix* F60 have to agree with the addresses of the variables in the data type „UDT_1" from the SIMATIC 300.

Step 7:

Connect the "Send Data" of the F60 with the SIMATIC 300 variables:

- ❑ Select the TCP connection configured in step 4 with the Id „001".
- ❑ Open the context menu and select *connect signals*.
- ❑ Open the tab "Send Data" in the dialog window "Signal Connections".
- ❑ Open the signal editor with *Signal->Editor* in the hardware management.
- ❑ In the signal editor, click on the "Name" of the signal „Siemens_HIMA1" and move the signal by Drag&Drop into the tab "Send Data" of the dialog window "Signal Connections".
- ❑ Repeat the previous step, if you have defined more than one input signal for this TCP connection.
- ❑ Click on the command button *New Offsets* in the dialog window "Signal Connections".
- ❑ In the popup window "Renumber Offsets" click on the command button *Renumber*.
- ❑ Close the dialog window.

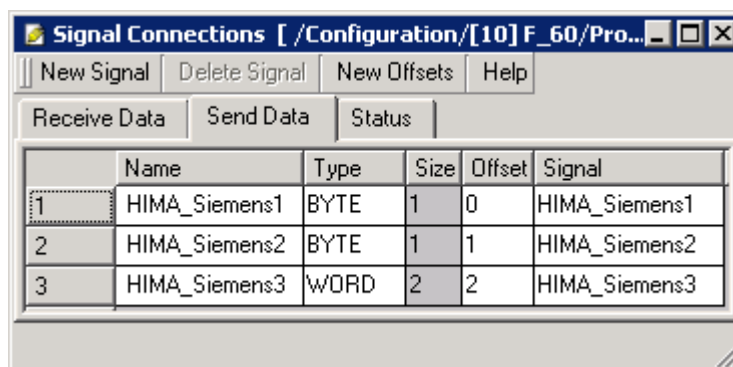


Figure28: Send data of the *HIMatrix* F60

Note

Consider that the offsets of the signals in the *HIMatrix* F60 have to agree with the addresses of the variables in the data type „UDT_1" from the SIMATIC 300.

Step 8:

Load the code into the resource F60:

- ❑ Start the code generator for the resource F60.
- ❑ Make sure that the code was generated without any error (see Error state viewer).
- ❑ Download the code into the resource F60.

Step 9:

Check the communication with the HIMA Force Editor:

- ❑ Open the context menu of the resource F60 and open the Force editor via *Online->Force editor*.
- ❑ Select all the signals which you have created in step 5.
- ❑ Force the signals from the register “Send Data”.

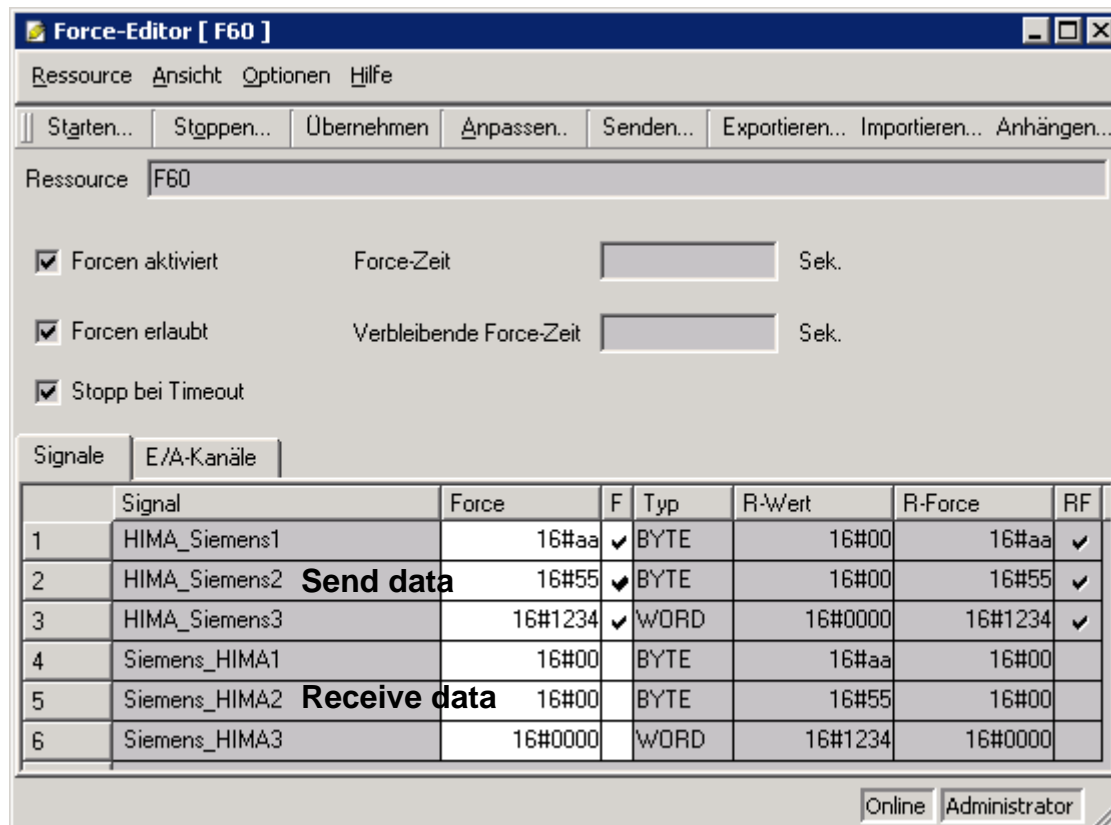


Figure29: The “Send Data” is written back to the HIMA control device into the “Receive Data”, by the Siemens control device.

HIMA
...the safe decision.



HIMA Paul Hildebrandt GmbH
Industrial Automation

Postfach 1261 • D - 68777 Bruehl

Phone: (+49) 6202 709-0 • Fax: (+49) 6202 709-107

E-mail: info@hima.com • Internet: www.hima.com

(0921)