YASKAWA

VIPA SPEED7 Library

OPL_SP7-LIB | SW90FS0MA V10.002 | Manual

Block library - Network Communication

VIPA CONTROLS

Table of contents

1 General

1.1 Copyright © VIPA GmbH

All Rights Reserved
This document contains proprietary information of VIPA and is not to be disclosed or used except in accordance with applicable agreements.

This material is protected by the copyright laws. It may not be reproduced, distributed, or altered in any fashion by any entity (either internal or external to VIPA), except in accordance with applicable agreements, contracts or licensing, without the express written consent of VIPA and the business management owner of the material.

For permission to reproduce or distribute, please contact: VIPA, Gesellschaft für Visualisierung und Prozessautomatisierung mbH Ohmstraße 4, D-91074 Herzogenaurach, Germany

Tel.: +49 9132 744 -0

Fax.: +49 9132 744-1864

EMail: info@vipa.de

http://www.vipa.com

> *Every effort has been made to ensure that the information contained in this document was complete and accurate at the time of publishing. Nevertheless, the authors retain the right to modify the information.*
>
> *This customer document describes all the hardware units and functions known at the present time. Descriptions may be included for units which are not present at the customer site. The exact scope of delivery is described in the respective purchase contract.*

CE Conformity Declaration
Hereby, VIPA GmbH declares that the products and systems are in compliance with the essential requirements and other relevant provisions. Conformity is indicated by the CE marking affixed to the product.

Conformity Information
For more information regarding CE marking and Declaration of Conformity (DoC), please contact your local VIPA customer service organization.

Trademarks
VIPA, SLIO, System 100V, System 200V, System 300V, System 300S, System 400V, System 500S and Commander Compact are registered trademarks of VIPA Gesellschaft für Visualisierung und Prozessautomatisierung mbH.

SPEED7 is a registered trademark of profichip GmbH.

SIMATIC, STEP, SINEC, TIA Portal, S7-300, S7-400 and S7-1500 are registered trademarks of Siemens AG.

Microsoft and Windows are registered trademarks of Microsoft Inc., USA.

Portable Document Format (PDF) and Postscript are registered trademarks of Adobe Systems, Inc.

All other trademarks, logos and service or product marks specified herein are owned by their respective companies.

Information product support

Contact your local VIPA Customer Service Organization representative if you wish to report errors or questions regarding the contents of this document. If you are unable to locate a customer service centre, contact VIPA as follows:

VIPA GmbH, Ohmstraße 4, 91074 Herzogenaurach, Germany

Telefax: +49 9132 744-1204

EMail: documentation@vipa.de

Technical support

Contact your local VIPA Customer Service Organization representative if you encounter problems with the product or have questions regarding the product. If you are unable to locate a customer service centre, contact VIPA as follows:

VIPA GmbH, Ohmstraße 4, 91074 Herzogenaurach, Germany

Tel.: +49 9132 744-1150 (Hotline)

EMail: support@vipa.de

1.2 About this manual

Objective and contents

The manual describes the block library *'Network Communication'* from VIPA:

- It contains a description of the structure, project implementation and usage in several programming systems.
- The manual is targeted at users who have a background in automation technology.
- The manual is available in electronic form as PDF file. This requires Adobe Acrobat Reader.
- The manual consists of chapters. Every chapter provides a self-contained description of a specific topic.
- The following guides are available in the manual:
  – An overall table of contents at the beginning of the manual
  – References with pages numbers

Icons Headings

Important passages in the text are highlighted by following icons and headings:

> **DANGER!**
> Immediate or likely danger. Personal injury is possible.

> **CAUTION!**
> Damages to property is likely if these warnings are not heeded.

> *Supplementary information and useful tips.*

2   Important notes

2.1   General

> In the following, you will find important notes, which must always be observed when using the blocks.

2.2   Internally used blocks

> **CAUTION!**
> The following blocks are used internally and must not be overwritten! The direct call of an internal block leads to errors in the corresponding instance DB! Please always use the corresponding function for the call.

FC/SFC	Designation	Description
FC/SFC 131	TSEND_	is used internally for FB 63
FC/SFC 132	TRECV_	is used internally for FB 64
FC/SFC 133	TCON_	is used internally for FB 65
FC/SFC 134	TDISCON_	is used internally for FB 66
FC/SFC 135	TUSEND_	is used internally for FB 67
FC/SFC 136	TURECV_	is used internally for FB 68
FC/SFC 192	CP_S_R	is used internally for FB 7 and FB 8
FC/SFC 196	AG_CNTRL	is used internally for FC 10
FC/SFC 198	USEND_	is used internally for FB 8
FC/SFC 198	URCV_	is used internally for FB 9
FC/SFC 200	AG_GET	is used internally for FB/SFB 14
FC/SFC 201	AG_PUT	is used internally for FB/SFB 15
FC/SFC 202	AG_BSEND	is used internally for FB/SFB 12
FC/SFC 203	AG_BRCV	is used internally for FB/SFB 13
FC/SFC 204	IP_CONF	is used internally for FB 55 IP_CONF
FC/SFC 205	AG_SEND	is used internally for FC 5 AG_SEND
FC/SFC 206	AG_RECV	is used internally for FC 6 AG_RECV
FC/SFC 253	IBS_ACCESS	is used internally for SPEED bus INTERBUS masters
SFB 238	EC_RWOD	is used internally for EtherCAT Communication
SFB 239	FUNC	is used internally for FB 240, FB 241

3 Include library

Block library *'Network Communication'*

The block library can be found for download in the *'Service/Support'* area of www.vipa.com at *'Downloads* ➔ *VIPA Lib'* as *'Block library Network Communication - SW90FS0MA'*. The library is available as packed zip file. As soon as you want to use these blocks you have to import them into your project.

> *Please always use the manual associated with your library. As long as there are no description-relevant changes, the version information in the manual can differ from those of the library and its files.*

The following block libraries are available

File	Description
NetworkCom_S7_V0001.zip	◼ Block library for Siemens SIMATIC Manager. ◼ For use in CPUs from VIPA or S7-300 CPUs from Siemens.
NetworkCom_TIA_V0002.zip	◼ Block library for Siemens TIA Portal V14 and V15. ◼ For use in CPUs from VIPA or S7-300 CPUs from Siemens.

3.1 Integration into Siemens SIMATIC Manager

Overview

The integration into the Siemens SIMATIC Manager requires the following steps:

1. Load ZIP file
2. "Retrieve" the library
3. Open library and transfer blocks into the project

Load ZIP file

Navigate on the web page to the desired ZIP file, load and store it in your work directory.

Retrieve library

1. Start the Siemens SIMATIC Manager with your project.
2. Open the dialog window for ZIP file selection via *'File* ➔ *Retrieve'*.
3. Select the according ZIP file and click at [Open].
4. Select a destination folder where the blocks are to be stored.
5. Start the extraction with [OK].

Open library and transfer blocks into the project

1. Open the library after the extraction.
2. Open your project and copy the necessary blocks from the library into the directory "blocks" of your project.

   ⇨ Now you have access to the VIPA specific blocks via your user application.

> *Are FCs used instead of SFCs, so they are supported by the VIPA CPUs starting from firmware 3.6.0.*

3.2 Integration into Siemens TIA Portal

Overview

The integration into the Siemens TIA Portal requires the following steps:

1. ▶ Load ZIP file
2. ▶ Unzip the Zip file
3. ▶ "Retrieve" the library
4. ▶ Open library and transfer blocks into the project

Load ZIP file

1. ▶ Navigate on the web page to the ZIP file, that matches your version of the program.
2. ▶ Load and store it in your work directory.

Unzip the Zip file

▶ Unzip the zip file to a work directory of the Siemens TIA Portal with your unzip application.

Open library and transfer blocks into the project

1. ▶ Start the Siemens TIA Portal with your project.
2. ▶ Switch to the *Project view*.
3. ▶ Choose "Libraries" from the task cards on the right side.
4. ▶ Click at "Global libraries".
5. ▶ Click at "Open global libraries".
6. ▶ Navigate to your work directory and load the file ..._TIA.al1x.
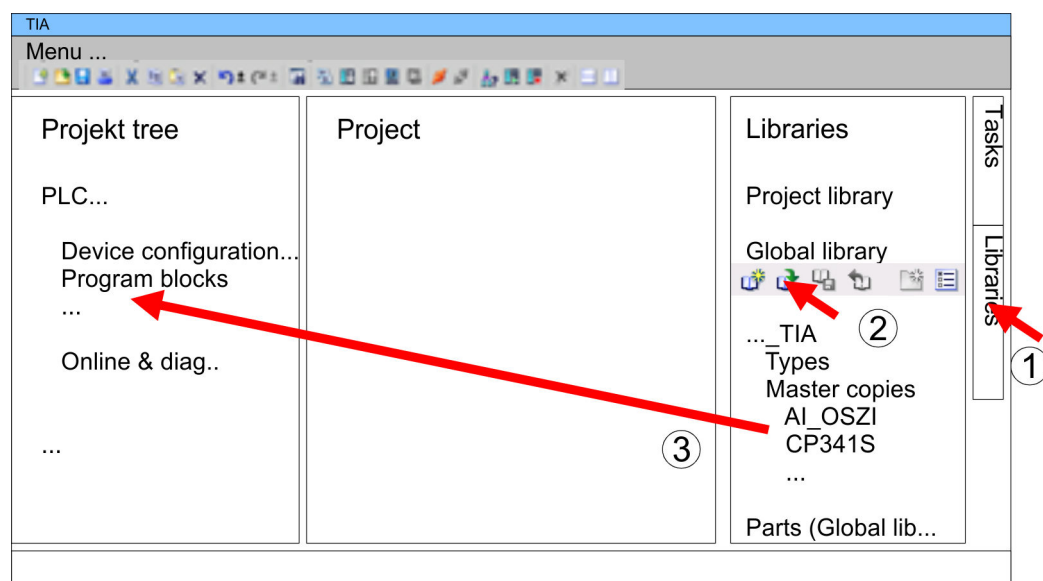
7. ▶ Copy the necessary blocks from the library into the "Program blocks" of the *Project tree* of your project. Now you have access to the VIPA specific blocks via your user application.

4 Block parameters

4.1 General and Specific Error Information RET_VAL

Overview

The return value *RET_VAL* of a system function provides one of the following types of error codes:

- A *general error code*, that relates to errors that can occur in anyone SFC.
- A *specific error code*, that relates only to the particular SFC.

Although the data type of the output parameter *RET_VAL* is integer (INT), the error codes for system functions are grouped according to hexadecimal values.

If you want to examine a return value and compare the value with the error codes, then display the error code in hexadecimal format.

RET_VAL (Return value)

The table below shows the structure of a system function error code:

Bit	Description
7 ... 0	Event number or error class and single error
14 ... 8	Bit 14 ... 8 = "0": **Specific error code**
	The specific error codes are listed in the descriptions of the individual SFCs.
	Bit 14 ... 8 > "0": **General error code**
	The possible general error codes are shown
15	Bit 15 = "1": indicates that an error has occurred.

Specific error code

This error code indicates that an error pertaining to a particular system function occurred during execution of the function.

A specific error code consists of the following two numbers:

- Error class between 0 and 7
- Error number between 0 and 15

Bit	Description
3 ... 0	Error number
6 ... 4	Error class
7	Bit 7 = "1"
14 ... 8	Bit 14 ... 8 = "0"
15	Bit 15 = "1": indicates that an error has occurred.

**General error codes
RET_VAL**

The parameter *RET_VAL* of some SFCs only returns general error information. No specific error information is available.

The general error code contains error information that can result from any system function. The general error code consists of the following two numbers:

- A parameter number between 1 and 111, where 1 indicates the first parameter of the SFC that was called, 2 the second etc.
- An event number between 0 and 127. The event number indicates that a synchronous fault has occurred.

Bit	Description
7 ... 0	Event number
14 ... 8	Parameter number
15	Bit 15 = "1": indicates that an error has occurred.

General error codes

The following table explains the general error codes associated with a return value. Error codes are shown as hexadecimal numbers. The x in the code number is only used as a placeholder. The number represents the parameter of the system function that has caused the error.

Error code	Description
8x7Fh	Internal Error. This error code indicates an internal error at parameter x. This error did not result from the actions if the user and he/she can therefore not resolve the error.
8x01h	Illegal syntax detection for an ANY parameter.
8x22h	Area size error when a parameter is being read.
8x23h	Area size error when a parameter is being written. This error code indicates that parameter x is located either partially or fully outside of the operand area or that the length of the bit-field for an ANY-parameter is not divisible by 8.
8x24h	Area size error when a parameter is being read.
8x25h	Area size error when a parameter is being written. This error code indicates that parameter x is located in an area that is illegal for the system function. The description of the respective function specifies the areas that are not permitted for the function.
8x26h	The parameter contains a number that is too high for a time cell. This error code indicates that the time cell specified in parameter x does not exist.
8x27h	The parameter contains a number that is too high for a counter cell (numeric fields of the counter). This error code indicates that the counter cell specified in parameter x does not exist.
8x28h	Orientation error when reading a parameter.
8x29h	Orientation error when writing a parameter. This error code indicates that the reference to parameter x consists of an operand with a bit address that is not equal to 0.
8x30h	The parameter is located in the write-protected global-DB.
8x31h	The parameter is located in the write-protected instance-DB. This error code indicates that parameter x is located in a write-protected data block. If the data block was opened by the system function itself, then the system function will always return a value 8x30h.
8x32h	The parameter contains a DB-number that is too high (number error of the DB).
8x34h	The parameter contains a FC-number that is too high (number error of the FC).

Error code	Description
8x35h	The parameter contains a FB-number that is too high (number error of the FB). This error code indicates that parameter x contains a block number that exceeds the maximum number permitted for block numbers.
8x3Ah	The parameter contains the number of a DB that was not loaded.
8x3Ch	The parameter contains the number of a FC that was not loaded.
8x3Eh	The parameter contains the number of a FB that was not loaded.
8x42h	An access error occurred while the system was busy reading a parameter from the peripheral area of the inputs.
8x43h	An access error occurred while the system was busy writing a parameter into den peripheral area of the outputs.
8x44h	Error during the n-th (n > 1) read access after an error has occurred.
8x45h	Error during the n-th (n > 1) write access after an error has occurred. This error code indicates that access was denied to the requested parameter.

5   Network Communication

5.1   Open Communication

5.1.1   Connection-oriented protocols

■ Connection-oriented protocols establish a (logical) connection to the communication partner before data transmission is started. And if necessary they terminate the connection after the data transfer was finished.

■ Connection-oriented protocols are used for data transmission when reliable, guaranteed delivery is of particular importance. Also the correct order of the received packets is ensured.

■ In general, many logical connections can exist on one physical line.

■ The following connection-oriented protocols are supported with FBs for open communication via industrial Ethernet:
  – TCP/IP native according to RFC 793 (connection types 01h and 11h)
  – ISO on TCP according to RFC 1006 connection type 12h)

TCP native

■ During data transmission, no information about the length or about the start and end of a message is transmitted. However, the receiver has no means of detecting where one message ends in the data stream and the next one begins.

■ The transfer is stream-oriented. For this reason, it is recommended that the data length of the FBs is identical for the sending and receiving station.

■ If the number of received data does not fit to the preset length you either will get not the whole data, or you will get data of the following job.

■ The receive block copies as many bytes into the receive area as you have specified as length. After this, it will set NDR to TRUE and write RCVD_LEN with the value of LEN. With each additional call, you will thus receive another block of sent data.

ISO on TCP

■ During data transmission, information on the length and the end of the message is also transmitted. The transfer is block-oriented

■ If you have specified the length of the data to be received greater than the length of the data to be sent, the receive block will copy the received data completely into the receive range. After this, it will set NDR to TRUE and write RCVD_LEN with the length of the sent data.

■ If you have specified the length of the data to be received less than the length of the sent data, the receive block will not copy any data into the receive range but instead will supply the following error information: ERROR = 1, STATUS = 8088h.

5.1.2   Connection-less protocols

There is thus no establishment and termination of a connection with a remote partner. Connection-less protocols transmit data with no acknowledge and with no reliable guaranteed delivery to the remote partner. The following connection-oriented protocol is supported with FBs for open communication via Industrial Ethernet:

■ UDP according to RFC 768 (with connection type 13h)

UDP

■ In this case, when calling the sending block you have to specify the address parameters of the receiver (IP address and port number). During data transmission, information on the length and the end of the message is also transmitted.

■ Analog after finishing the receive block you get a reference to the address parameter of the sender (IP address and port no.)

■ In order to be able to use the sending and receiving blocks first you have to configure the local communications access point at both sides.

- With each new call of the sending block, you re-reference the remote partner by specifying its IP address and its port number.
- If you have specified the length of the data to be received greater than the length of the data to be sent, the receive block will copy the received data completely into the receive range. After this, it will set NDR to TRUE and write RCVD_LEN with the length of the sent data.
- If you have specified the length of the data to be received less than the length of the sent data, the receive block will not copy any data into the receive range but instead will supply the following error information: ERROR = 1, STATUS = 8088h.

5.1.3 FB 63 - TSEND - Sending data - TCP native and ISO on TCP

Description

- FB 63 TSEND Sends data over an editing communications connection. FB 63 TSEND is an asynchronously functioning FB, which means that its processing extends over several FB calls.
- To start sending data, call FB 63 with *REQ* = 1.
- The job status is indicated at the output parameters *BUSY* and *STATUS*. *STATUS* corresponds to the *RET_VAL* output parameter of asynchronously functioning SFCs (see also Meaning of the Parameters *REQ*, *RET_VAL* and *BUSY* with Asynchronous SFCs).
- The following table shows the relationships between *BUSY*, *DONE* and *ERROR*. Using this table, you can determine the current status of FB 63 or when the establishment of the connection is complete.

BUSY	DONE	ERROR	Description
TRUE	irrelevant	irrelevant	The job is being processed.
FALSE	TRUE	FALSE	The job was completed successfully.
FALSE	FALSE	TRUE	The job was ended with an error. The cause of the error can be found in the STATUS parameter.
FALSE	FALSE	FALSE	The FB was not assigned a (new) job.

> *Due to the asynchronous function of FB 63 TSEND, you must keep the data in the sender area consistent until the DONE parameter or the ERROR parameter assumes the value TRUE.*

Parameters

Parameter	Declaration	Data type	Memory area	Description
REQ	INPUT	BOOL	I, Q, M, D, L	Control parameter *REQ*, initiates terminating the connection specified by the *ID*. Initiation occurs at rising edge. At the first call with *REQ* = 1, data are transmitted from the area specified by the *DATA* parameter.
ID	INPUT	WORD	M, D, constant	Reference to the connection to determinated. *ID* must be identical to the associated parameter *ID* in the local connection description. Range of values: 0001h ... 0FFFh

Open Communication > FB 63 - TSEND - Sending data - TCP native and ISO on TCP

Parameter	Declaration	Data type	Memory area	Description
LEN	INPUT	INT	I, Q, M, D, L	Number of bytes to be sent with the job Range of values: ■ 1 ... 1460, if connection type = 01h ■ 1 ... 8192, if connection type = 11h ■ 1 ... 1452, if connection type = 12h and a CP is being used ■ 1 ... 8192, if connection type = 12h and no CP is being used
DONE	OUTPUT	BOOL	I, Q, M, D, L	*DONE* status parameter: ■ 0: Job not yet started or still running. ■ 1: Job executed without error.
BUSY	OUTPUT	BOOL	I, Q, M, D, L	■ *BUSY* = 1: Job is not yet completed. A new job cannot be triggered. ■ *BUSY* = 0: Job is completed.
ERROR	OUTPUT	BOOL	I, Q, M, D, L	*ERROR* status parameter: ■ *ERROR* = 1: Error occurred during processing. *STATUS* provides detailed information on the type of error.
STATUS	OUTPUT	WORD	M, D	*STATUS* parameter: Status information
DATA	IN_OUT	ANY	I, Q, M, D	Send area, contains address and length. The address refers to: ■ The process image input ■ The process image output ■ A bit memory ■ A data block Allowed referenced data types: BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TIME_OF_DAY, TIME, S5TIME, DATE_AND_TIME, STRING

Status information

ERROR	STATUS	Description
0	0000h	Send job completed without error.
0	7000h	First call with *REQ* = 0, sending not initiated.
0	7001h	First call with *REQ* = 1, sending initiated.
0	7002h	Follow-on call (REQ irrelevant), job being processed **Note**: during this processing the operating system accesses the data in the *DATA* send buffer.
1	8085h	*LEN* parameter has the value 0 or is greater than the largest permitted value.
1	8086h	The *ID* parameter is not in the permitted address range.
0	8088h	LEN parameter is larger than the memory area specified in *DATA*.
1	80A1h	Communications error: ■ FB 65 TCON was not yet called for the specified *ID* ■ The specified connection is currently being terminated. Transmission over this connection is not possible. ■ The interface is being reinitialized.
1	80B3h	The parameter for the connection type (*connection_type* parameter in the connection description) is set to UDP. Please use the FB 67 TUSEND.
1	80C3h	The resources (memory) of the CPU are temporarily occupied.
1	80C4h	Temporary communications error: ■ The connection to the communications partner cannot be established at this time. ■ The interface is receiving new parameters.
1	8822h	*DATA* parameter: Source area invalid: area does not exist in DB.
1	8824h	*DATA* parameter: Range error in ANY pointer.
1	8832h	*DATA* parameter: DB number too large.
1	883Ah	*DATA* parameter: Access to send buffer not possible (e.g. due to deleted DB).
1	887Fh	*DATA* parameter: Internal error, such as an invalid ANY reference.
1	8F7Fh	Internal Error (VIPA specific)
1	8xyyh	General error information ⬲ *Chap. 4.1 'General and Specific Error Information RET_VAL' page 9*

5.1.4 FB 64 - TRCV - Receiving Data - TCP native and ISO on TCP

Description

FB 64 TRCV receives data over an existing communication connection. The are two variants available for receiving and processing the data:

■ Variant 1: Received data block is processed immediately.
■ Variant 2: Received data block is stored in a receive buffer and is only processed when the buffer is full.

The following table shows the relationships between the connection type is shown in the following table:

Connection type	Variant
01h and 11h	The user can specify the variant.
12h	Variant 2 (fix)

The two variants are more described in the following table.

Received Data ...	Range Values for *LEN*	Range Values for *RCVD_LEN*	Description
are available immediately.	0	1 ... x	The data go into a buffer whose length x is specified in the ANY pointer of the receive buffer (*DATA* parameter). After being received, a data block is immediately available in the receive buffer. The amount of data received (*RCVD_LEN* parameter) can be no greater than the size specified in the *DATA* parameter. Receiving is indicated by *NDR* = 1.
are stored in the receive buffer. The data are available as soon as the configured length is reached.	1 ... 1460, if the connection type = 01h 1 ... 8192, if the connection type = 11h 1 ... 1452, if the connection type = 12h and a CP is being used 1 ... 8192, if the connection type = 12h and no CP is being used	Same value as in the LEN parameter	The data go into a buffer whose length is specified by the LEN parameter. If this specified length is reached, the received data are made available in the *DATA* parameter (*NDR* = 1).

Function	■ FB 64 TRCV is an asynchronously functioning FB, which means that its processing extends over several FB calls. To start receiving data, call FB 64 with *REQ* = 1.
	■ The job status is indicated at the output parameters *BUSY* and *STATUS*. *STATUS* corresponds to the *RET_VAL* output parameter of asynchronously functioning SFCs (see also Meaning of the Parameters *REQ*, *RET_VAL* and *BUSY* with Asynchronous SFCs).
	■ The following table shows the relationships between *BUSY*, *DONE* and *ERROR*. Using this table, you can determine the current status of FB 64 or when the receiving process is complete.

BUSY	DONE	ERROR	Description
TRUE	irrelevant	irrelevant	The job is being processed.
FALSE	TRUE	FALSE	The job was completed successfully.
FALSE	FALSE	TRUE	The job was ended with an error. The cause of the error can be found in the *STATUS* parameter.
FALSE	FALSE	FALSE	The FB was not assigned a (new) job.

> *Due to the asynchronous function of FB 64 TRCV, the data in the receiver area are only consistent when the NDR parameter assumes the value TRUE.*

Parameters

Parameter	Declaration	Data type	Memory area	Description
EN_R	INPUT	BOOL	I, Q, M, D, L	With *EN_R* = 1, FB 64 TRCV is ready to receive (Control parameter). The receive job is processed.
ID	INPUT	WORD	M, D, constant	Reference to the connection to be terminated. *ID* must be identical to the associated parameter *id* in the local connection description. Range of values: 0001h ... 0FFFh
LEN	INPUT	INT	I, Q, M, D, L	■ *LEN* = 0 (ad hoc mode): use implied length specified in the ANY pointer for *DATA*. The received data are made available immediately when the block is called. The amount of data received is available in *RCVD_LEN*. ■ 1 ≤ LEN ≤ max: number of bytes to be received. The amount of data actually received is available in *RCVD_LEN*. The data are available after they have been completely received. "max" depends on the connection type: – max = 1460 with connection type 01h – max = 8192 with connection type 11h – max = 1452 with connection type 12h with a CP – max = 8192 with connection type 12h without a CP

Open Communication > FB 64 - TRCV - Receiving Data - TCP native and ISO on TCP

Parameter	Declaration	Data type	Memory area	Description
NDR	OUTPUT	BOOL	I, Q, M, D, L	*NDR* status parameter: ■ *NDR* = 0: Job not yet started or still running. ■ *NDR* = 1: Job successfully completed
ERROR	OUTPUT	BOOL	I, Q, M, D, L	*ERROR* status parameter: ■ *ERROR* = 1: Error occurred during processing. *STATUS* provides detailed information on the type of error
BUSY	OUTPUT	BOOL	I, Q, M, D, L	■ *BUSY* = 1: Job is not yet completed. A new job cannot be triggered. ■ *BUSY* = 0: Job is completed.
STATUS	OUTPUT	WORD	M, D	*STATUS* parameter: Status information
RCVD_LEN	OUTPUT	INT	I, Q, M, D, L	Amount of data actually received, in bytes
DATA	IN_OUT	ANY	I, Q, M, D	Receiving area (address and length)The address refers to: ■ The process image input ■ The process image output ■ A bit memory ■ A data block Allowed referenced data types: BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TIME_OF_DAY, TIME, S5TIME, DATE_AND_TIME, STRING

Status information

ERROR	STATUS	Description
0	0000h	New data were accepted. The current length of the received data is shown in *RCVD_LEN*.
0	7000h	First call with *REQ* = 0, receiving not initiated
0	7001h	Block is ready to receive. Receiving job has been activated.
0	7002h	Follow-on call, job being processed **Note**: during this processing the operating system writes the operating system data to the *DATA* receive buffer. For this reason, an error could result in inconsistent data being in the receive buffer.
1	8085h	*LEN* parameter is greater than the largest permitted value, or you changed the value of *LEN* from the one that existed during the first call
1	8086h	The *ID* parameter is not in the permitted address range
1	8088h	■ Target buffer (*DATA*) is too small value LEN is greater than the predetermined by *DATA*. Troubleshooting if the connection type = 12h: Increase the destination buffer *DATA*.

ERROR	STATUS	Description
1	80A1h	Communications error: ■ FB 65 TCON was not yet called for the specified *ID* ■ The specified connection is currently being terminated. Receiving over this connection is not possible. ■ The interface is receiving new parameters.
1	80B3h	The parameter for the connection type (*connection_type* parameter in the connection description) is set to UDP. Please use the FB 68 TRCV.
1	80C3h	The operating resources (memory) in the CPU are temporarily occupied.
1	80C4h	Temporary communications error: The connection is currently being terminated.
1	8922h	*DATA* parameter: Target area invalid: area does not exist in DB.
1	8924h	*DATA* parameter: Range error in ANY pointer
1	8932h	*DATA* parameter: DB number too large.
1	893Ah	*DATA* parameter: Access to receive buffer not possible (e.g. due to deleted DB)
1	897Fh	*DATA* parameter: Internal error, such as an invalid ANY reference
1	8F7Fh	Internal Error (VIPA specific)
1	8xyyh	General error information ⬉ *Chap. 4.1 'General and Specific Error Information RET_VAL' page 9*

5.1.5 FB 65 - TCON - Establishing a connection

Use with TCP native and ISO on TCP

Both communications partners call FB 65 TCON to establish the communications connection. In the parameters you specify which partner is the active communications transmission point and which is the passive one. For information on the number of possible connections, please refer to the technical data for your CPU. After the connection is established, it is automatically monitored and maintained by the CPU. If the connection is interrupted, such as due a line break or due to the remote communications partner, the active partner attempts to reestablish the connection. In this case, you do not have to call FB 65 TCON again. An existing connection is terminated when FB 66 TDISCON is called or when the CPU has gone into STOP mode. To reestablish the connection, you will have to call FB 65 TCON again.

Use with UDP

Both communications partner call FB 65 TCON in order to configure their local communications access point. A connection is configured between the user program and the communications level of the operating system. No connection is established to the remote partner. The local access point is used to send and receive UDP message frames.

Description

FB 65 TCON is an asynchronously functioning FB, which means that its processing extends over several FB calls. To start establishing a connection, call FB 65 with REQ = 1. The job status is indicated at the output parameters *RET_VAL* and *BUSY*. *STATUS* corresponds to the *RET_VAL* output parameter of asynchronously functioning SFCs (see also Meaning of the Parameters *REQ*, *RET_VAL* and *BUSY* with asynchronous SFCs). The following table shows the relationships between BUSY, DONE and ERROR. Using this table, you can determine the current status of FB 65 or when the establishment of the connection is complete.

BUSY	DONE	ERROR	Description
TRUE	irrelevant	irrelevant	The job is being processed.
FALSE	TRUE	FALSE	The job was completed successfully.
FALSE	FALSE	TRUE	The job was ended with an error. The cause of the error can be found in the *STATUS* parameter.
FALSE	FALSE	FALSE	The FB was not assigned a (new) job.

Parameters

Parameter	Declaration	Data type	Memory area	Description
REQ	INPUT	BOOL	I, Q, M, D, L	Control parameter *REQ*, initiates establishing the connection at rising edge.
ID	INPUT	WORD	M, D, constant	Reference to the connection to be established to the remote partner or between the user program and the communications level of the operating system. ID must be identical to the associated parameter ID in the local connection description. Range of values: 0001h ... 0FFFh
DONE	OUTPUT	BOOL	I, Q, M, D, L	*DONE* status parameter: ■ 0: Job not yet started or still running. ■ 1: Job executed without error.
BUSY	OUTPUT	BOOL	I, Q, M, D, L	■ *BUSY* = 1: Job is not yet completed. ■ *BUSY* = 0: Job is completed.

Parameter	Declaration	Data type	Memory area	Description
ERROR	OUTPUT	BOOL	I, Q, M, D, L	*ERROR* status parameter: ■ *ERROR* = 1: Error occurred during processing. *STATUS* provides detailed information on the type of error.
STATUS	OUTPUT	WORD	M, D	*STATUS* status parameter: Error information
CONNECT	IN_OUT	ANY	D	Pointer to the associated connection description. Ä *Chap. 5.1.6 'UDT 65 - TCON_PAR Data structure for FB 65' page 22*

Status information

ERROR	STATUS	Description
0	0000h	Connection is able to be established
0	7000h	Call with *REQ* = 0, establishment of connection not initiated
0	7001h	First call with *REQ* = 1, connection being established
0	7002h	Follow-on call (*REQ* irrelevant), connection being established
1	8086h	The ID parameter must not have value of zero.
0	8087h	Maximal number of connections reached; no additional connection possible
1	8089h	The *CONNECT* parameter does not point to a data block.
1	809Ah	The *CONNECT* parameter points to a field that does not have the length of the data structure for assigning connection (UDT 65).
1	809Bh	The communication interface specified via *local_device_id* and *next_staddr* is not supported by the CPU.
1	80A1h	Connection or port is already occupied by the user.
1	80A2h	Local or remote port is occupied by the system.
1	80A3h	Attempt being made to re-establish an existing connection.
1	80A4h	IP address of the remote connection endpoint is invalid.
1	80A7h	Communications error: you have called TDISCON before TCON was complete. TDISCON must first complexly terminate the connection referenced by the ID.
1	80B4h	In the ISO on TCP protocol, one or more of the following conditions have been violated during passive connection setup: ■ *local_tsap_id_len* $\geq$ 02h ■ *local_tsap_id*[1] = E0h at *local_tsap_id_len* = 02h ■ *local_tsap_id*[1] an ASCII character *local_tsap_id_len* $\geq$ 03h ■ *local_tsap_id*[1] is an ASCII character and *local_tsap_id_len* $\geq$ 03h
1	80B5h	Parameter active_est (UDT 65) is TRUE with the protocol variant UDP.
1	80B6h	Parameters connection_type is invalid (UDT 65).

ERROR	STATUS	Description
1	80B7h	Error in one of the following parameters of UDT 65: ■ block_length ■ local_tsap_id_len ■ rem_subnet_id_len ■ rem_staddr_len ■ rem_tsap_id_len ■ next_staddr_len
1	80B8h	Parameters *id* in the local connection description (UDT 65) and parameter ID are different.
1	80C3h	Temporary lack of resources in the CPU.
1	80C4h	Temporary communications error: ■ The connection cannot be established at this time. ■ The interface is receiving new parameters.
1	8F7Fh	Internal Error (VIPA specific)
1	8xyyh	General error information ⬧ *Chap. 4.1 'General and Specific Error Information RET_VAL' page 9*

5.1.6 UDT 65 - TCON_PAR Data structure for FB 65

5.1.6.1 Data structure for assigning connection

In the TCP Connection parameterization of native or ISO on TCP, you define which communication partners enabled the connection and which to a request through the communication partner performs a passive connection. If both communication partners have launched their connection, the operating system can restore the communication link. To communicate a DB is needed. Facility whereby the DB's data structure from the UDT 65 TCON_PAR. For each connection such a data structure is needed that can be summarized in a global DB. The CONNECT connection parameter address of FB 65 TCON contains a reference to the associated connection description (e.g. P#DB10.DBX0.0 byte 64).

Data structure

Byte	Parameter	Data type	Start value	Description
0 ... 1	block_length	WORD	40h	Length of UDT 65: 64 bytes (fixed)
2 ... 3	id	WORD	0000h	■ Reference to the connection (range of values: 0001h ... 0FFFh) ■ You must specify the value of the parameter in the respective block with *ID*.
4	connection_type	BYTE	01h	Connection type: ■ 11h: TCP/IP native ■ 12h: ISO on TCP ■ 13h: UDP ■ 01h: TCP/IP native (Compatibility mode)

Byte	Parameter	Data type	Start value	Description
5	active_est	BOOL	FALSE	ID for the way the connection is established: TCP, TCP, IoT: ■ FALSE: passive establishment ■ TRUE: active establishment UDP: ■ FALSE
6	local_device_id	BYTE	02h	Communication device ■ 00h: Ethernet PG/OP channel of the CPU ■ 02h: Ethernet CP of the CPU
7	local_tsap_id_len	BYTE	02h	Length of parameter *local_tsap_id* used; possible values: TCP ■ Active side: 0 (dynamic port) or 2 ■ Passive side: 2 ISO on TCP ■ 2 ... 16 UDP ■ 2 TCP ■ Active side: 0 ■ Passive side: 2
8	rem_subnet_id_len	BYTE	00h	This parameter is currently not used. You must assign 00h to it.
9	rem_staddr_len	BYTE	00h	Length of address for the remote connection transmission point: TCP/ISO on TCP/TCP (Comp.) ■ 0: unspecified, i.e. parameter *rem_staddr* is irrelevant. 4: valid IP address in the parameter *rem_staddr* UDP ■ 0*

Open Communication > UDT 65 - TCON_PAR Data structure for FB 65

Byte	Parameter	Data type	Start value	Description
10	rem_tsap_id_len	BYTE	00h	Length of parameter *rem_tsap_id* used; possible values: TCP ■ Active side: 2 (The port must be specified.) ■ Passive side: 0 or 2 ISO on TCP ■ 0 or 2 ... 16 UDP ■ This parameter is not used. Assign parameter to 00h. TCP (Comp.) ■ Active side: 2 (The port must be specified.) For the passive side, only the value 00h permitted.
11	next_staddr_len	BYTE	00h	Length of parameter *next_staddr* used ■ 00h: Ethernet CP of the CPU ■ 01h: Ethernet PG/OP channel of the CPU
12 ... 27	local_tsap_id	ARRAY [1..16] of BYTE	00h ...	With *connection_type* TCP, UDP ■ local_tsap_id[1] = high byte of port number in hexadecimal representation ■ local_tsap_id[2] = low byte of port number in hexadecimal representation ■ local_tsap_id[3-16] = 00h ISO on TCP ■ local TSAP-ID (possible values: 2000 ... 5000) – local_tsap_id[1] = E0h (connection type T-connection) – local_tsap_id[2] = Rack and slot in own CPU (bits 0 ... 4 slot, bits 5 ... 7: rack number) – local_tsap_id[3-16] = TSAP extension TCP (Comp.) ■ local_tsap_id[1] = low byte of port number in hexadecimal representation ■ local_tsap_id[2] = high byte of port number in hexadecimal representation ■ local_tsap_id[3-16] = 00h **Note:** Make sure that each value of *local_tsap_id* that you use in your CPU is unique.
28 ... 33	rem_subnet_id	ARRAY [1..6] of BYTE	00h ...	This parameter is currently not used. You must assign 00h to it.

Byte	Parameter	Data type	Start value	Description
34 ... 39	rem_staddr	ARRAY [1..6] of BYTE	00h ...	IP address for the remote connection transmission point: e.g. 192.168.002.003: With *connection_type* ■ TCP / ISO on TCP – rem_staddr[1] = C0h (192) – rem_staddr[2] = A8h (168) – rem_staddr[3] = 02h (002) – rem_staddr[4] = 03h (003) – rem_staddr[5-6] = irrelevant ■ UDP – This parameter is not used. Assign parameter to 00h. ■ TCP (Comp.) – rem_staddr[1] = 03h (003) – rem_staddr[2] = 02h (002) – rem_staddr[3] = A8h (168) – rem_staddr[4] = C0h (192) – rem_staddr[5-6] = irrelevant
40 ... 55	rem_tsap_id	ARRAY [1..16] of BYTE	00h ...	With *connection_type* ■ TCP: remote port number (possible values: 2000 ... 5000) – rem_tsap_id[1] = high byte of port no in hexadecimal representation – rem_tsap_id[2] = low byte of port no in hexadecimal representation – rem_tsap_id[3-16] = 00h ■ ISO on TCP: remote TSAP-ID: – rem_tsap_id[1] = E0h (connection type T-connection) – rem_tsap_id[2] = Rack and slot for the remote connection transmission point CPU (bits 0 ... 4: slot, bits 5 ... 7: rack number), – rem_tsap_id[3-16] = TSAP extension ■ UDP This parameter is not used. Assign parameter to 00h ■ 01h: remote port number (possible values: 2000 ... 5000) – local_tsap_id[1] = low byte of port number in hexadecimal representation – local_tsap_id[2] = high byte of port number in hexadecimal representation – local_tsap_id[3-16] = 00h
56 ... 61	next_staddr	ARRAY [1..6] of BYTE	00h ...	Rack and slot of the configured CP for the PG/OP interface ■ 00h (Ethernet P/OP channel) – next_staddr[1]: 04h – next_staddr[2-6]: 00h ■ 02h (Ethernet CP) – next_staddr[1-6]: 00h

Byte	Parameter	Data type	Start value	Description
62 ... 63	spare	WORD	0000h	irrelevant

*) The partner IP address is specified by calling the TUSEND/TURECV parameter via the *ADDR* parameter.

5.1.6.2 Data structure for communications access point

A communications access point provides the link between application of the communication layer of the operating system. Defined for communication over UDP, each communication partner a communication access point using a DB. Facility whereby the DB's data structure from the UDT 65 "TCON_PAR".

Data structure

Byte	Parameter	Data type	Start value	Description
0 ... 1	block_length	WORD	40h	Length of UDT 65: 64 Bytes (fixed)
2 ... 3	id	WORD	0000h	■ Reference to this connection between the user program and the communications level of the operating system (range of values: 0001h ... 0FFFh) ■ You must specify the value of the parameter in the respective block with the ID.
4	connection_type	BYTE	01h	Connection type: ■ 13h: UDP
5	active_est	BOOL	FALSE	ID for the way the connection is established: You must assign FALSE to this parameter since the communications access point can be used to both send and receive data.
6	local_device_id	BYTE	02h	Communication device ■ 00h: Ethernet PG/OP channel of the CPU ■ 02h: Ethernet CP of the CPU
7	local_tsap_id_len	BYTE	02h	Length of parameter local_tsap_id used; possible value: 2
8	rem_subnet_id_len	BYTE	00h	This parameter is currently not used. Value 00h (fix).
9	rem_staddr_len	BYTE	00h	This parameter is currently not used. Value 00h (fix).
10	rem_tsap_id_len	BYTE	00h	This parameter is currently not used. Value 00h (fix).
11	next_staddr_len	BYTE	00h	This parameter is currently not used. Value 00h (fix).

Byte	Parameter	Data type	Start value	Description
12 ... 27	local_tsap_id	ARRAY [1..16] of BYTE	00h ...	■ Remote port number (possible values: 2000 ... 5000), local_tsap_id[1] = high byte of port no in hexadecimal representation, local_tsap_id[2] = low byte of port no in hexadecimal representation, local_tsap_id[3-16] = irrelevant **Note:** Make sure that each value of *local_tsap_id* that you use in your CPU is unique.
28 ... 33	rem_subnet_id	ARRAY [1..6] of BYTE	00h ...	This parameter is currently not used. Value 00h (fix).
34 ... 39	rem_staddr	ARRAY [1..6] of BYTE	00h ...	This parameter is currently not used. Value 00h (fix).
40 ... 55	rem_tsap_id	ARRAY [1..16] of BYTE	00h ...	This parameter is currently not used. Value 00h (fix).
56 ... 61	next_staddr	ARRAY [1..6] of BYTE	00h ...	This parameter is currently not used. Value 00h (fix).
62 ... 63	spare	WORD	0000h	irrelevant

5.1.7 FB 66 - TDISCON - Terminating a connection

Use with TCP native and ISO on TCP

FB 66 TDISCON terminates a communications connection from the CPU to a communications partner.

Use with UDP

The FB 66 TDISCON closes the local communications access point. The connection between the user program and the communications level of the operating system is terminated.

Description

FB 66 TDISCON is an asynchronously functioning FB, which means that its processing extends over several FB calls. To start terminating a connection, call FB 66 with REQ = 1.

After FB 66 TDISCON has been successfully called, the ID specified for FB 65 TCON is no longer valid and thus cannot be used for sending or receiving.

The job status is indicated at the output parameters *RET_VAL* and *BUSY*. *STATUS* corresponds to the *RET_VAL* output parameter of asynchronously functioning SFCs (see also Meaning of the Parameters REQ, RET_VAL and *BUSY* with asynchronous SFCs).

The following table shows the relationships between *BUSY*, *DONE* and *ERROR*. Using this table, you can determine the current status of FB 66 or when the establishment of the connection is complete.

BUSY	DONE	ERROR	Description
TRUE	irrelevant	irrelevant	The job is being processed.
FALSE	TRUE	FALSE	The job was completed successfully.

BUSY	DONE	ERROR	Description
FALSE	FALSE	TRUE	The job was ended with an error. The cause of the error can be found in the *STATUS* parameter.
FALSE	FALSE	FALSE	The FB was not assigned a (new) job.

Parameters

Parameter	Declaration	Data type	Memory area	Description
REQ	INPUT	BOOL	I, Q, M, D, L	Control parameter *REQ*, initiates terminating the connection specified by the *ID*. Initiation occurs at rising edge.
ID	INPUT	WORD	M, D, constant	Reference to the connection to be terminated to the remote partner or between the user program and the communications level of the operating system. *ID* must be identical to the associated parameter *ID* in the local connection description. Range of values: 0001h ... 0FFFh
DONE	OUTPUT	BOOL	I, Q, M, D, L	*DONE* status parameter: ■ 0: Job not yet started or still running. ■ 1: Job executed without error.
BUSY	OUTPUT	BOOL	I, Q, M, D, L	■ *BUSY* = 1: Job is not yet completed ■ *BUSY* = 0: Job is completed.
ERROR	OUTPUT	BOOL	I, Q, M, D, L	ERROR status parameter: ■ *ERROR* = 1: Error occurred during processing. *STATUS* provides detailed information on the type of error.
STATUS	OUTPUT	WORD	M, D	*STATUS* parameter: Status information

Status information

ERROR	STATUS	Description
0	0000h	Connection is terminated
0	7000h	First call with *REQ* = 0, establishment of connection not initiated
0	7001h	First call with *REQ* = 1, start of the processing, connection being terminated
0	7002h	Follow-on call (*REQ* irrelevant), connection being terminated
1	8086h	The ID parameter is not in the permitted address range
1	80A3h	Attempt being made to terminate a non-existent connection
1	80C4h	Temporary communications error: The interface is receiving new parameters.
1	8F7Fh	Internal Error (VIPA specific)
1	8xyyh	General error information ⅌ *Chap. 4.1 'General and Specific Error Information RET_VAL' page 9*

5.1.8 FB 67 - TUSEND - Sending data - UDP

Description

FB 67 TUSEND sends data via UDP to the remote partner specified by the parameter *ADDR*.

> *When sending separate data in sequence to different partners, you only need to adjust the parameter ADDR when calling FB 67 TUSEND. It is not necessary to call FB 65 TCON and FB 66 TDISCON again.*

Function

- FB 67 TUSEND is an asynchronously functioning FB, which means that its processing extends over several FB calls. To start sending data, call FB 67 with *REQ* = 1.
- The job status is indicated at the output parameters *BUSY* and *STATUS*. *STATUS* corresponds to the *RET_VAL* output parameter of asynchronously functioning SFCs (see also Meaning of the Parameters *REQ*, *RET_VAL* and *BUSY* with asynchronous SFCs).
- The following table shows the relationships between *BUSY*, *DONE* and *ERROR*. Using this table, you can determine the current status of FB 67 or when the sending process (transmission) is complete.

BUSY	DONE	ERROR	Description
TRUE	irrelevant	irrelevant	The job is being processed.
FALSE	TRUE	FALSE	The job was completed successfully.
FALSE	FALSE	TRUE	The job was ended with an error. The cause of the error can be found in the *STATUS* parameter.
FALSE	FALSE	FALSE	The FB was not assigned a (new) job.

> *Due to the asynchronous function of FB 67 TUSEND, you must keep the data in the sender area consistent until the DONE parameter or the ERROR parameter assumes the value TRUE.*

Parameters

Parameter	Declaration	Data type	Memory area	Description
REQ	INPUT	BOOL	I, Q, M, D, L	Control parameter *REQ*, initiates the transmission at rising edge. At the first call with *REQ* = 1, bytes are transmitted from the area specified by the *DATA* parameter.
ID	INPUT	WORD	M, D, constant	Reference to the associated connection between the user program and the communication level of the operating system. *ID* must be identical to the associated parameter *ID* in the local connection description. Range of values: 0001h ... 0FFFh

Parameter	Declaration	Data type	Memory area	Description
LEN	INPUT	INT	I, Q, M, D, L	Number of bytes to be sent with the job: Range of values: 1 ... 1460
DONE	OUTPUT	BOOL	I, Q, M, D, L	*DONE* status parameter: ■ 0: Job not yet started or still running ■ 1: Job executed without error.
BUSY	OUTPUT	BOOL	I, Q, M, D, L	■ *BUSY* = 1: Job is not yet completed. A new job cannot be triggered. ■ *BUSY* = 0: Job is completed.
ERROR	OUTPUT	BOOL	I, Q, M, D, L	ERROR status parameter: ■ *ERROR* = 1: Error occurred during processing. *STATUS* provides detailed information on the type of error
STATUS	OUTPUT	WORD	M, D	*STATUS* parameter: Error information
DATA	IN_OUT	ANY	I, Q, M, D	Sender area, contains address and length The address refers to: ■ The process image input table ■ The process image output table ■ A bit memory ■ A data block Allowed referenced data types: BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TIME_OF_DAY, TIME, S5TIME, DATE_AND_TIME, STRING
ADDR	IN_OUT	ANY	D	Pointer to the address of the receiver (e.g. P#DB100. DBX0.0 byte 8), see Structure of the Address Information for the Remote Partner with UDP.

Error information

ERROR	STATUS	Description
0	0000h	Send job completed without error.
0	7000h	First call with *REQ* = 1, sending not initiated.
0	7001h	First call with *REQ* = 1, sending initiated.
0	7002h	Follow-on call (*REQ* irrelevant), job being processed **Note**: during this processing the operating system accesses the data in the *DATA* send buffer.
1	8085h	*LEN* parameter has the value 0 or is greater than the largest permitted value.
1	8086h	The *ID* parameter is not in the permitted address range.
0	8088h	*LEN* parameter is larger than the memory area specified in *DATA*.
1	8089h	Parameter ADDR does not point to a data block.

ERROR	STATUS	Description
1	80A1h	Communications error: ■ FB 65 TCON was not yet called for the specified *ID* ■ The specified connection between the user program and the communication level of the operating system is currently being terminated. Transmission over this connection is not possible. ■ The interface is being reinitialized (receiving new parameters).
1	80A4h	The IP address of the communication partner is not valid.
1	80B3h	■ The parameter for the connection type (connection_type parameter in the connection description) is not set to UDP. Please use the FB 63 TSEND. ■ Parameter *ADDR*: invalid port number or IP address.
1	80B7h	Length error: The parameter *ADDR* is the length specification < 8byte.
1	80C4h	Temporary communications error: ■ The communication partner is currently not available. ■ The connection is currently being configured (or TCON is still running).
1	8822h	*DATA* parameter: Source area invalid: area does not exist in DB.
1	8824h	*DATA* parameter: Range error in ANY pointer.
1	8832h	*DATA* parameter: DB number too large.
1	883Ah	*DATA* parameter: Access to send buffer not possible (e.g. due to deleted DB).
1	887Fh	*DATA* parameter: Internal error, e.g. an invalid ANY reference.
1	8F7Fh	Internal Error (VIPA specific)
1	8xyyh	General error information ⬩ *Chap. 4.1 'General and Specific Error Information RET_VAL' page 9*

5.1.9 FB 68 - TURCV - Receiving data - UDP

Description

■ FB 68 TURCV receives data via UDP. After successful completion of FB 68 TURCV the parameter ADDR will show you the address of the remote partner (the sender).
■ FB 68 TURCV is an asynchronously functioning FB, which means that its processing extends over several FB calls. To start sending data, call FB 68 with *REQ* = 1.
■ The job status is indicated at the output parameters *RET_VAL* and *BUSY*. *STATUS* corresponds to the *RET_VAL* output parameter of asynchronously functioning SFCs (see also Meaning of the Parameters *REQ*, *RET_VAL* and *BUSY* with asynchronous SFCs).
■ The following table shows the relationships between *BUSY*, *NDR* and *ERROR*. Using this table, you can determine the current status of FB 68 or when the receiving process is complete.

BUSY	NDR	ERROR	Description
TRUE	irrelevant	irrelevant	The job is being processed.
FALSE	TRUE	FALSE	The job was completed successfully.

BUSY	NDR	ERROR	Description
FALSE	FALSE	TRUE	The job was ended with an error. The cause of the error can be found in the *STATUS* parameter.
FALSE	FALSE	FALSE	The FB was not assigned a (new) job.

> *Due to the asynchronous function of FB 68 TURCV, the data in the receiver area are only consistent when the NDR parameter assumes the value TRUE.*

Parameters

Parameter	Declaration	Data type	Memory area	Description
EN_R	INPUT	BOOL	I, Q, M, D, L	Control parameter enabled to receive: when *EN_R* = 1, FB 68 TURCV is ready to receive.
ID	INPUT	WORD	M, D, constant	Reference to the associated connection between the user program and the communication level of the operating system. *ID* must be identical to the associated parameter *ID* in the local connection description. Range of values: 0001h ... 0FFFh
LEN	INPUT	INT	I, Q, M, D, L	$1 \le LEN \le 1472$: number of bytes to be received. The received data are immediately available when the block is called. The amount of data received is available in *RCVD_LEN*.
NDR	OUTPUT	BOOL	I, Q, M, D, L	NDR status parameter: ■ *NDR* = 0: Job not yet started or still running. ■ *NDR* = 1: Job successfully completed
ERROR	OUTPUT	BOOL	I, Q, M, D, L	*ERROR* status parameter: ■ *ERROR* = 1: Error occurred during processing. *STATUS* provides detailed information on the type of error
BUSY	OUTPUT	BOOL	I, Q, M, D, L	■ *BUSY* = 1: Job is not yet completed. A new job cannot be triggered. ■ *BUSY* = 0: Job is completed.
STATUS	OUTPUT	WORD	M, D	Status parameter: Error information
RCVD_LEN	OUTPUT	INT	I, Q, M, D, L	Amount of data actually received, in bytes

Parameter	Declaration	Data type	Memory area	Description
DATA	IN_OUT	ANY	I, Q, M, D	Receiver area, contains address and length The address refers to: ■ The process image input table ■ The process image output table ■ A bit memory ■ A data block Allowed referenced data types: BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TIME_OF_DAY, TIME, S5TIME, DATE_AND_TIME, STRING
ADDR	IN_OUT	ANY	D	Pointer to the address of the sender (e.g. P#DB100.DBX0.0 byte 8), see Structure of the Address Information for the Remote Partner with UDP

Error information

ERROR	STATUS	Description
0	0000h	New data were accepted. The current length of the received data is shown in *RCVD_LEN*.
0	7000h	First call with *REQ* = 0, receiving not initiated
0	7001h	Block is ready to receive.
0	7002h	Follow-on call, job being processed **Note**: during this processing the operating system writes the operating system data to the *DATA* receive buffer. For this reason, an error could result in inconsistent data being in the receive buffer.
1	8085h	*LEN* parameter is greater than the largest permitted value, or you changed the value of *LEN* from the one that existed during the first call
1	8086h	The *ID* parameter is not in the permitted address range
1	8088h	■ Target buffer (*DATA*) is too small. ■ The value in *LEN* is greater than the receiver area specified by *DATA*.
1	8089h	Parameter ADDR does not point to a data block.
1	80A1h	Communications error: ■ FB 65 TCON was not yet called for the specified *ID* ■ The specified connection between the user program and the communication level of the operating system is currently being terminated. Receiving over this connection is not possible. ■ The interface is being reinitialized (receiving new parameters).
1	80B3h	The parameter for the connection type (connection_type parameter in the connection description) is not set to UDP. Please use the FB 64 TRCV.
1	80B7h	Length error: The parameter *ADDR* is the length specification < 8byte.
1	80C4h	Temporary communications error: ■ The connection is currently being configured (or TCON is still running).

ERROR	STATUS	Description
1	8922h	*DATA* parameter: Target area invalid: area does not exist in DB.
1	8924h	*DATA* parameter: Range error in ANY pointer
1	8932h	*DATA* parameter: DB number too large.
1	893Ah	*DATA* parameter: Access to receive buffer not possible (e.g. deleted DB)
1	897Fh	*DATA* parameter: Internal error, such as an invalid ANY reference
1	8F7Fh	Internal Error (VIPA specific)
1	8xyyh	General error information ⅋ *Chap. 4.1 'General and Specific Error Information RET_VAL' page 9*

5.1.10   UDT 66 - TADDR_PAR Data structure

5.1.10.1   Data structure for assigning connection

Description

- With FB 67 TUSEND, at the parameter *ADDR* you transfer the address of the receiver. This address information must have structure specified below.
- With FB 68 TURCV, in the parameter *ADDR* you get the address of the sender of the data that were received. This address information must have structure specified below.

Data block

You have to create an DB that contains one or more data structures as per UDT 66 TADDR_PAR.

In parameter *ADDR* of FB 67 TUSEND you transfer and in parameter *ADDR* of FB 68 TURCV you receive a pointer to the address of the associated remote partner (e.g. P#DB10.DBX0.0 byte 8).

Structure of the address information for the remote partner

Byte	Parameter	Data type	Start value	Description
0 ... 3	rem_ip_addr	ARRAY [1..4] of BYTE	00h ...	IP address of the remote partner, e.g. 192.168.002.003: - rem_ip_addr[1] = C0h (192) - rem_ip_addr[2] = A8h (168) - rem_ip_addr[3] = 02h (002) - rem_ip_addr[4] = 03h (003)
4 ... 5	rem_port_nr	ARRAY [1..2] of BYTE	00h ...	remote port number (possible values: 2000 ... 5000) - rem_port_nr[1] = high byte of port number in hexadecimal representation - rem_port_nr[2] = low byte of port number in hexadecimal representation
6 ... 7	spare	ARRAY [1..2] of BYTE	00h ...	reserved (00h)

5.2 Ethernet Communication

5.2.1 Communication - FC 5...6 for CP 343

The two blocks are used to process connection requests on the PLC side of an Ethernet CP 343. Through integration of these blocks in the cycle block OB1 you may cyclically send and receive data. Within these blocks, the SFCs 205 and 206 are called that are stored as special function blocks in the CPU.

> *Please regard that you may only use the SEND/RECV-FCs from VIPA in your user application for the communication with VIPA CPs. At a change to VIPA CPs in an already existing project, the present AG_SEND / AG_LSEND res. AG_RECV / AG_LRECV may be replaced by AG_SEND res. AG_RECV from VIPA without adaptation. Due to the fact that the CP automatically adjusts itself to the length of the data to transfer, the L variant of SEND res. RECV is not required for VIPA CPs.*

Communication blocks

For the communication between CPU and Ethernet-CP 343, the following FCs are available:

- AG_SEND (FC 5)
  - This block transfers the user data from the data area given in *SEND* to the CP specified via *ID* and *LADDR*. As data area you may set a PI, bit memory or data block area. When the data area has been transferred without errors, "job ready without error" is returned.
- AG_RECV (FC 6)
  - The block transfers the user data from the CP into a data area defined via *RECV*. As data area you may set a PI, bit memory or data block area. When the data area has been transferred without errors, "job ready without error" is returned.

Status displays

The CP processes send and receive commands independently from the CPU cycle and needs for this transfer time. The interface with the FC blocks to the user application is here synchronized by means of acknowledgements/receipts. For status evaluation the communication blocks return parameters that may be evaluated directly in the user application. These status displays are updated at every block call.

Deployment at high communication load

Do not use cyclic calls of the communication blocks in OB 1. This causes a permanent communication between CPU and CP. Program instead the communication blocks within a time OB where the cycle time is higher OB 1 res. event controlled.

FC call is faster than CP transfer time

If a block is called a second time in the user application before the data of the last time is already completely send res. received, the FC block interface reacts like this:

- AG_SEND
  - No command is accepted until the data transfer has been acknowledged from the partner via the connection. Until this you receive the message "Order running" before the CP is able to receive a new command for this connection.
- AG_RECV
  - The job is acknowledged with the message "No data available yet" as long as the CP has not received the receive data completely.

AG_SEND, AG_RECV in user application

The following illustration shows a possible sequence for the FC blocks together with the organizations and program blocks in the CPU cycle:
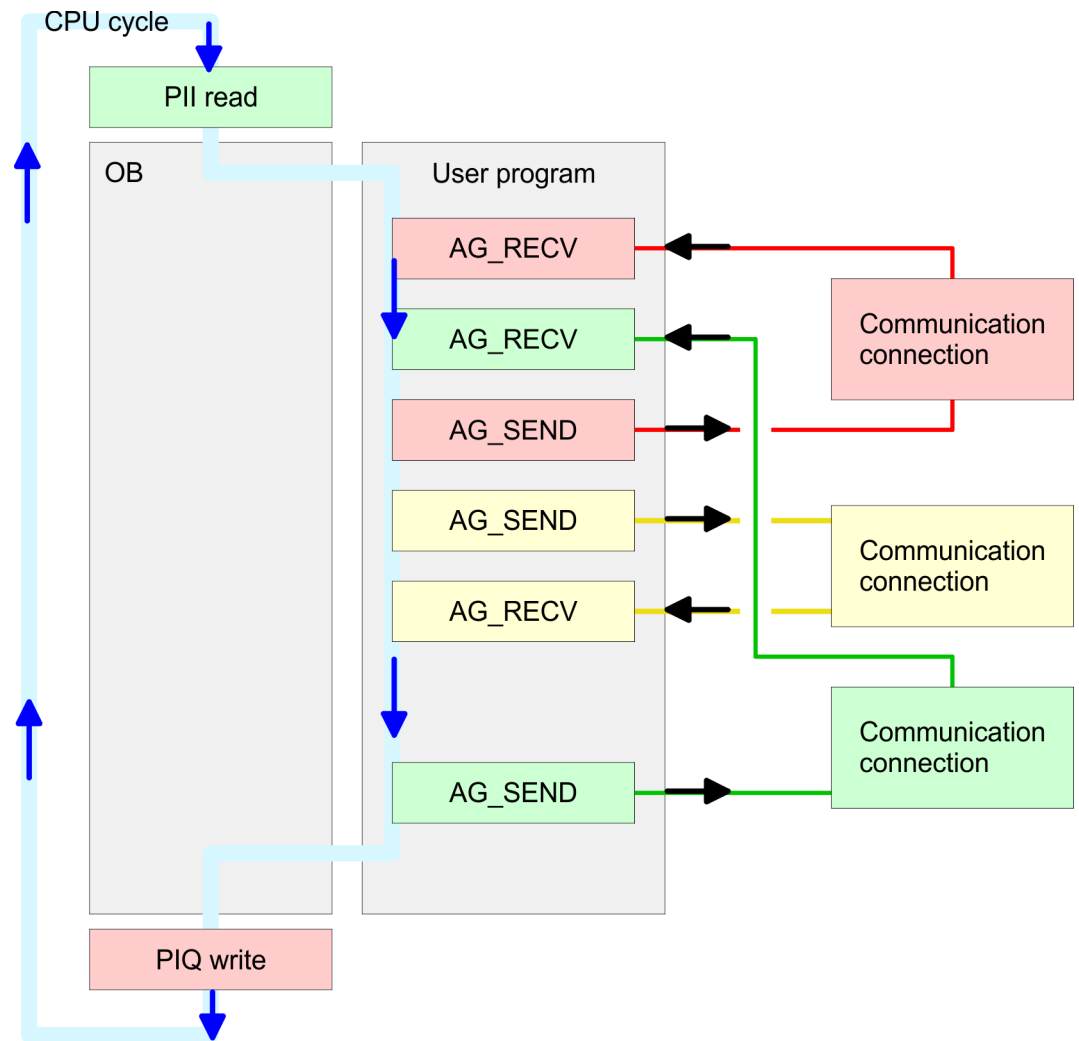
The FC blocks with concerning communication connection are grouped by color. Here you may also see that your user application may consist of any number of blocks. This allows you to send or receive data (with AG_SEND res. AG_RECV) event or program driven at any wanted point within the CPU cycle. You may also call the blocks for **one** communication connection several times within one cycle.

5.2.2 FC 5 - AG_SEND - Send to CP 343

By means of AG_SEND the data to send are transferred from the CPU to an Ethernet CP.

> ℹ️ *Please note that this block calls the FC or SFC 205 AG_SEND internally. These must not be overwritten! The direct call of an internal block leads to errors in the corresponding instance DB!*

Parameter

Parameter	Declaration	Data type	Description
ACT	INPUT	BOOL	Activation of the sender 0: Updates *DONE, ERROR* and *STATUS* 1: The data area defined in *SEND* with the length *LEN* is send
ID	INPUT	INT	Connection number 1 ... 16 (identical with *ID* of NetPro)
LADDR	INPUT	WORD	Logical basic address of the CP (identical with *LADDR* of NetPro)
SEND	INPUT	ANY	Data area
LEN	INPUT	INT	Number of bytes from data area to transfer
DONE	OUTPUT	BOOL	Status parameter for the job 0: Job running 1: Job finished without error.
ERROR	OUTPUT	BOOL	Error message 0: Job running (at *DONE* = 0) 0: Job ready without error (at *DONE* = 1) 1: Job ready with error
STATUS	OUTPUT	WORD	Status message returned with *DONE* and *ERROR*. More details are to be found in the following table.

DONE, ERROR, STATUS   The following table shows all messages that can be returned by the Ethernet CP after a SEND res. RECV job. A "-" means that this message is not available for the concerning SEND res. RECV command.

DONE (SEND)	NDR (RECV)	ERROR	STATUS	Description
1	-	0	0000h	Job finished without error.
-	1	0	0000h	New data taken without error.
0	-	0	0000h	There is no job being executed
-	0	0	8180h	No data available yet.
0	0	0	8181h	Job running
0	0	1	8183h	No CP project engineering for this job.

DONE (SEND)	NDR (RECV)	ERROR	STATUS	Description
0	-	1	8184h	System error occurred
-	0	1	8184h	System error occurred (source data area failure).
0	-	1	8185h	Parameter *LEN* exceeds source area *SEND*.
	0	1	8185h	Destination buffer (RECV) too small.
0	0	1	8186h	Parameter *ID* invalid (not within 1 ...16).
0	-	1	8302h	No receive resources at destination station, receive station is not able to process received data fast enough res. has no receive resources reserved.
0	-	1	8304h	The connection is not established. The send command shouldn't be sent again before a delay time of > 100ms.
-	0	1	8304h	The connection is not established. The receive command shouldn't be sent again after a delay time of > 100ms.
0	-	1	8311h	Destination station not available under the defined Ethernet address.
0	-	1	8312h	Ethernet error in the CP.
0		1	8F22h	Source area invalid, e.g. when area in DB not present Parameter *LEN* < 0
-	0	1	8F23h	Source area invalid, e.g. when area in DB not present Parameter *LEN* < 0
0	-	1	8F24h	Range error at reading a parameter.
-	0	1	8F25h	Range error at writing a parameter.
0	-	1	8F28h	Orientation error at reading a parameter.
-	0	1	8F29h	Orientation error at writing a parameter.
-	0	1	8F30h	Parameter is within write protected 1. recent data block
-	0	1	8F31h	Parameter is within write protected 2. recent data block Data block
0	0	1	8F32h	Parameter contains oversized DB number.
0	0	1	8F33h	DB number error
0	0	1	8F3Ah	Area not loaded (DB)
0	-	1	8F42h	Acknowledgement delay at reading a parameter from peripheral area.
-	0	1	8F43h	Acknowledgement delay at writing a parameter from peripheral area.
0	-	1	8F44h	Address of the parameter to read locked in access track
-	0	1	8F45h	Address of the parameter to write locked in access track
0	0	1	8F7Fh	Internal error e.g. invalid ANY reference e.g. parameter *LEN* = 0.
0	0	1	8090h	Module with this module start address not present or CPU in STOP.

DONE (SEND)	NDR (RECV)	ERROR	STATUS	Description
0	0	1	8091h	Module start address not within double word grid.
0	0	1	8092h	ANY reference contains type setting unequal BYTE.
-	0	1	80A0h	Negative acknowledgement at reading from module.
0	0	1	80A4h	reserved
0	0	1	80B0h	Module doesn't recognize the record set.
0	0	1	80B1h	The length setting (in parameter *LEN*) is invalid.
0	0	1	80B2h	reserved
0	0	1	80C0h	Record set not readable.
0	0	1	80C1h	The set record set is still in process.
0	0	1	80C2h	There is a job jam.
0	0	1	80C3h	The operating sources (memory) of the CPU are temporarily occupied.
0	0	1	80C4h	Communication error (occurs temporarily; a repetition in the user application is reasonable).
0	0	1	80D2h	Module start address is wrong.

Status parameter at reboot   At a reboot of the CP, the output parameters are set as follows:

■   DONE = 0
■   NDR = 0
■   ERROR = 0
■   STATUS = 8180h (at AG_RECV)
■   STATUS = 8181h (at AG_SEND)

5.2.3 FC 6 - AG_RECV - Receive from CP 343

With the 1. call of AG_RECV a receive buffer for the communication between CPU and an Ethernet CP 343 is established. From now on received data are automatically stored in this buffer. As soon as after calling AG_RECV the return value of *NDR* = 1 is returned, valid data are present. Since with a further call of AG_RECV the receive buffer is established again for the receipt of new data, you have to save the previous received data.

> Please note that this block calls the FC or SFC 206 AG_RECV internally. These must not be overwritten! The direct call of an internal block leads to errors in the corresponding instance DB!

Parameter

Parameter	Declaration	Data type	Description
ID	INPUT	INT	Connection number 1 ... 16 (identical with *ID* of NetPro)
LADDR	INPUT	WORD	Logical base address of the CP (identical with *LADDR* of NetPro)
RECV	INPUT	ANY	Data area for the received data.
NDR	OUTPUT	BOOL	Status parameter for the order 0: Order running 1: Order ready data received without error
ERROR	OUTPUT	BOOL	Error message 0: Order running (at *NDR* = 0) 0: Order ready without error (at *NDR* = 1) 1: Order ready with error
STATUS	OUTPUT	WORD	Status message returned with *NDR* and *ERROR*. More details are to be found in the following table.
LEN	OUTPUT	INT	Number of bytes that have been received

DONE, ERROR, STATUS   The following table shows all messages that can be returned by the Ethernet CP after a SEND res. RECV job. A "-" means that this message is not available for the concerning SEND res. RECV command.

DONE (SEND)	NDR (RECV)	ERROR	STATUS	Description
1	-	0	0000h	Job finished without error.
-	1	0	0000h	New data taken without error.
0	-	0	0000h	There is no job being executed
-	0	0	8180h	No data available yet.
0	0	0	8181h	Job running
0	0	1	8183h	No CP project engineering for this job.

DONE (SEND)	NDR (RECV)	ERROR	STATUS	Description
0	-	1	8184h	System error occurred
-	0	1	8184h	System error occurred (source data area failure).
0	-	1	8185h	Parameter *LEN* exceeds source area *SEND*.
	0	1	8185h	Destination buffer (RECV) too small.
0	0	1	8186h	Parameter *ID* invalid (not within 1 ...16).
0	-	1	8302h	No receive resources at destination station, receive station is not able to process received data fast enough res. has no receive resources reserved.
0	-	1	8304h	The connection is not established. The send command shouldn't be sent again before a delay time of > 100ms.
-	0	1	8304h	The connection is not established. The receive command shouldn't be sent again after a delay time of > 100ms.
0	-	1	8311h	Destination station not available under the defined Ethernet address.
0	-	1	8312h	Ethernet error in the CP.
0		1	8F22h	Source area invalid, e.g. when area in DB not present Parameter *LEN* < 0
-	0	1	8F23h	Source area invalid, e.g. when area in DB not present Parameter *LEN* < 0
0	-	1	8F24h	Range error at reading a parameter.
-	0	1	8F25h	Range error at writing a parameter.
0	-	1	8F28h	Orientation error at reading a parameter.
-	0	1	8F29h	Orientation error at writing a parameter.
-	0	1	8F30h	Parameter is within write protected 1. recent data block
-	0	1	8F31h	Parameter is within write protected 2. recent data block Data block
0	0	1	8F32h	Parameter contains oversized DB number.
0	0	1	8F33h	DB number error
0	0	1	8F3Ah	Area not loaded (DB)
0	-	1	8F42h	Acknowledgement delay at reading a parameter from peripheral area.
-	0	1	8F43h	Acknowledgement delay at writing a parameter from peripheral area.
0	-	1	8F44h	Address of the parameter to read locked in access track
-	0	1	8F45h	Address of the parameter to write locked in access track
0	0	1	8F7Fh	Internal error e.g. invalid ANY reference e.g. parameter *LEN* = 0.
0	0	1	8090h	Module with this module start address not present or CPU in STOP.

DONE (SEND)	NDR (RECV)	ERROR	STATUS	Description
0	0	1	8091h	Module start address not within double word grid.
0	0	1	8092h	ANY reference contains type setting unequal BYTE.
-	0	1	80A0h	Negative acknowledgement at reading from module.
0	0	1	80A4h	reserved
0	0	1	80B0h	Module doesn't recognize the record set.
0	0	1	80B1h	The length setting (in parameter *LEN*) is invalid.
0	0	1	80B2h	reserved
0	0	1	80C0h	Record set not readable.
0	0	1	80C1h	The set record set is still in process.
0	0	1	80C2h	There is a job jam.
0	0	1	80C3h	The operating sources (memory) of the CPU are temporarily occupied.
0	0	1	80C4h	Communication error (occurs temporarily; a repetition in the user application is reasonable).
0	0	1	80D2h	Module start address is wrong.

Status parameter at reboot At a reboot of the CP, the output parameters are set as follows:

- DONE = 0
- NDR = 0
- ERROR = 0
- STATUS = 8180h (at AG_RECV)
- STATUS = 8181h (at AG_SEND)

5.2.4 FC 10 - AG_CNTRL - Control CP 343

Description

The connections of the Ethernet CP 343 may be diagnosed and initialized by means of the VIPA FC 10.

The following jobs may be executed by parameterizable commands:

- Reading connection information
- Resetting configured connections

The commands of this block are permitted only for SEND/RECV connections based on the ISO/RFC/TCP and UDP protocols.

> *Please note that this block calls the FC or SFC 196 AG_CNTRL internally. These must not be overwritten! The direct call of an internal block leads to errors in the corresponding instance DB!*

FC 10 in the user program

The following diagram shows a typical sequence of AG_CNTRL. Here it is shown how the connection status is initially queried and then, in a second job, how the connection termination is triggered with the rest command.

User program
(CPU cycle)

Ethernet-CP

Abort and reestablish
a connection

Status query with AG_CNTRL:
ACT: 0-1; CMD = 1, ID = 1 (connection)

AG_CNTRL (status query)

Status query for connection 1
is acknowledged

Status connection ID 1 [1]

Reset with AG_CNTRL:
ACT: 0-1; CMD = 2, ID = 1 (connection)

AG_CNTRL (Reset)

Reset request active:
(Bit 15 in RESULT1 is set)

Acknowledgement of job
acceptance:
RESULT1: "Connection
establishment triggered" [1]

Entry in diagnostic buffer

Connection is terminated
and reinitialized:
(Bit 15 in RESULT1
remains set)

Status query with AG_CNTRL:
ACT: 0-1; CMD = 1, ID = 1 (connection)

AG_CNTRL (status query)

Status query is acknow-
ledged then the reset ID
(Bit 15 in Result 1) is cleared

Status connection ID1 [1]
Acknowledgement: RESULT1:
B095h "Reset was executed"

Time

Time

[1] Parameter transfer *DONE*, *ERROR*, *STATUS* and *RESULT1/2*

Parameters

Parameter	Declaration	Data type	Description
ACT	INPUT	BOOL	Job triggered by edge change 0-1 of the memory bit *ACT*
ID	INPUT	INT	Connection ID according to configuration
LADDR	INPUT	WORD	Base address of CP in hardware configuration
CMD	INPUT	INT	Job ID
DONE	OUTPUT	BOOL	Execution code
ERROR	OUTPUT	BOOL	Error code

Parameter	Declaration	Data type	Description
STATUS	OUTPUT	WORD	Status code
RESULT1	OUTPUT	DWORD	Job result 1 under command
RESULT2	OUTPUT	DWORD	Job result 2 under command

ACT

Possible values: 0, 1

The FC is to be called with edge change 0-1 of *ACT*.

If it is called with *ACT* = 0, there is no function call and the block is exited immediately.

ID

Possible values: 1, 2 ... n, or 0

The number of the connection is specified in the parameter *ID*. The connection number may be found in the configuration. n is the maximum number of connections.

If the call addresses every connection as *ID* 0 is to be specified (_ALL-function with *CMD* 3 respectively *CMD* 4).

LADDR

Module base address

At CP configuration with the hardware configurator the module base address is displayed in the configuration table.

Specify this address here.

CMD

Command to the FC AG_CNTRL

DONE

0: Job is still being processed or not yet triggered

1: Job executed

This parameter indicates whether or not the job was completed without errors.

If *DONE* = 1 *RESULT* may be evaluated.

ERROR

0: No error

1: Error indication

STATUS

Status indication

RESULT1/2

Information returned according to the command sent to the FC AG_CNTRL

DONE, ERROR, STATUS

The following table shows the messages that may be returned by the Ethernet-CP 343 after an AG_CNTRL call.

Additional the command results in the parameters *RESULT1* and *RESULT2* are to be evaluated.

DONE	ERROR	STATUS	Description
1	0	0000h	Job executed without error
0	0	0000h	No job executing
0	0	8181h	Job active, the block call is to be repeated with the same parameters until *DONE* or *ERROR* is returned.
0	1	8183h	There is no CP configuration for this job or the service has not yet started in the Ethernet-CP 343.
0	1	8186h	Parameter *ID* is invalid. The permitted *ID* depends on the selected command.
0	1	8187h	Parameter *CMD* is invalid
0	1	8188h	Sequence error in the *ACT* control
0	1	8090h	Module with this address does not exist or CPU in STOP.
0	1	8091h	The module base address is not on a double-word boundary.
0	1	80B0h	The module does not recognize the record set.
0	1	80C0h	The record set cannot be read.
0	1	80C1h	The specified record set is currently being processed.
0	1	80C2h	There are too many jobs pending.
0	1	80C3h	CPU resources (memory) occupied.
0	1	80C4h	Communication error (error occurs temporarily; it is usually best to repeat the job in the user program).
0	1	80D2h	The module base address is incorrect.

Status parameter at cold restart

The output parameters are set to the following values during a restart of the CP:

- *DONE* = 0
- *NDR* = 0
- *ERROR* = 8180h (at AG_RECV)
- *ERROR* = 8181h (at AG_SEND)

> *Please consider the block may only be called with new parameters if a job started before was just ended with DONE = 1.*

Commands and evaluating the job results

The following table shows the possible commands and the results that may be evaluated in the parameters *RESULT1* and *RESULT2*.

CMD 0

NOP - no operation

The block is executed without a job being sent to the CP.

RESULT	Hex value/range	Description
RESULT 1	0000 0001h	Executed without error
RESULT 2	0000 0000h	Default

CMD 1	CN_STATUS - connection status

This command returns the status of the connection selected with the *ID* of the CP addressed by *LADDR*. If bit 15 (reset ID) is set, this is automatically reset (this action corresponds to the CMD 5 - CN_CLEAR_RESET).

RESULT	Hex value/range	Description
RESULT 1	0000 000xh	Bit 3 ... 0: Codes for the send direction (excluded: 0010_b)
		Bit 0: Connection reserved for send and receive jobs
		Bit 1: Send job being executed
		Bit 3, 2: Previous job
		00: No information
		01: Send job completed successful
		10: Send job not completed successfully
	0000 00x0h	Bit 7 ... 4: Codes for receive direction (excluded: 0010_b)
		Bit 4: Connection reserved for send and receive jobs
		Bit 5: Receive job being executed
		Bit 7, 6: Previous job
		00: No information
		01: Receive job completed successfully
		10: Receive job not completed successfully
	0000 0x00h	Bit 11 ... 8: Codes for FETCH/WRITE
		(excluded: 0011_b, 0111_b, 1000_b, 1011_b, 0010_b)
		Bit 8: Connection type
		0: No FETCH connection
		1: Connection reserved for FETCH jobs
		Bit 9: Connection type
		0: No WRITE connection
		1: Connection reserved for WRITE jobs
		Bit 10: Job status (FETCH/ WRITE)
		0: Job status OK
		1: Job status not OK
		This ID is set in the following situations:
		- The job was acknowledged negatively by the CPU
		- The job could not be forwarded to the CPU because the connection was in the "LOCKED" status.
		- The job was rejected because the FETCH/WRITE header did not have the correct structure.
		Bit 11: Status of FETCH/WRITE job
		0: No job active
		1: Job from LAN active

RESULT	Hex value/range	Description
	0000 x000h	Bit 15 ... 12: General CP information
		(excluded: 0011_b, 1011_b)
		Bit 13, 12: Connection status
		(only available for SEND/RECV connections based on the ISO/RFC/TCP protocols; with UDP, the corresponding internal information is output)
		00: Connection is terminated
		01: Connection establishment active
		10: Connection termination active
		11: Connection is established
		Bit 14: CP information
		0: CP in STOP
		1: CP in RUN
		Bit 15: Reset ID
		0: FC 10 has not yet reset a connection or the reset ID was cleared.
		1: The FC 10 has executed a connection reset
	xxxx 0000h	Bit 31 ... 16: Reserved for later expansions
RESULT 2	0000 0000h	Reserved for later expansions

CMD 2 CN_RESET - connection reset

This command resets the connection selected with the *ID* of the CP addressed by *LADDR*.

Resetting the connection means that a connection is aborted and established again (active ore passive depending on the configuration).

An entry is also generated in the diagnostic buffer in which the job result may be found.

RESULT	Hex value/range	Description
RESULT 1	0000 0001h	The reset job was transferred to the CP successfully.
		The connection abort and subsequent connection establishment were triggered.
	0000 0002h	The reset job could not be transferred to the CP because the service was not started on the CP (for example CP in STOP).
RESULT 2	0000 0000h	Default

CMD 3 CN_STATUS_ALL - all connections status

This command returns the connection status of all connections (established/terminated) in the *RESULT1/2* parameters (at total of 8byte of group information) of the CP addressed by *LADDR*.

The *ID* parameter must be set to "0" (checked for "0").

When necessary, you may obtain detailed information about a terminated or not configured connection using a further connection status call with *CMD* = 1.

RESULT	Hex value/range	Description
RESULT 1	xxxx xxxxh	32 Bit: Connection 1 ... 32 0: Connection terminated / not configured 1: Connection established
RESULT 2	xxxx xxxxh	32 Bit: Connection 33 ... 64 0: Connection terminated / not configured 1: Connection established

CMD 4

CN_RESET_ALL - all connections reset

This command resets all connection of the CP addressed by *LADDR*.

The *ID* parameter must be set to "0" (checked for "0").

Resetting the connection means that a connection is aborted and established again (active ore passive depending on the configuration).

An entry is also generated in the diagnostic buffer in which the job result may be found.

RESULT	Hex value/ range	Description
RESULT 1	0000 0001h	The reset job was transferred to the CP successfully. The connection abort and subsequent connection establishment of every connection were triggered.
	0000 0002h	The reset job could not be transferred to the CP because the service was not started on the CP (for example CP in STOP).
RESULT 2	0000 0000h	Default

CMD 5

CN_CLEAR_RESET - Clear the reset ID

This command resets the reset ID (bit 15 in RESULT1) for the connection selected with the ID of the CP addressed by *LADDR*.

This job executes automatically when the connection status is read (*CMD* = 1); the separate job described here is therefore only required in special situations.

RESULT	Hex value/range	Description
RESULT 1	0000 0001h	The clear job was transferred to the CP successfully.
	0000 0002h	The clear job could not be transferred to the CP because the service was not started on the CP (for example CP in STOP).
RESULT 2	0000 0000h	Default

CMD 6

CN_DISCON - connection disconnect

This command resets the connection, which was selected by *ID* and *LADDR*. The reset is executed by means of aborting the connection.

Possibly in the stack stored data are lost without any instructions. After that no further connection is automatically established. The connection may again be established by the control job CN_STARTCON. An entry is also generated in the diagnostic buffer in which the job result may be found.

RESULT	Hex value/ range	Description
RESULT 1	0000 0001h	The job was transferred to the CP successfully. The connection abort was triggered.
	0000 0002h	This job could not be transferred to the CP because the service was not started on the CP (for example CP in STOP).
RESULT 2	0000 0000h	Default

CMD 7

CN_STARTCON - start connection

This command establishes a connection, which was selected by *ID* and *LADDR* and aborted by the control job CN_DISCON before. An entry is also generated in the diagnostic buffer in which the job result may be found.

RESULT	Hex value/ range	Description
RESULT 1	0000 0001h	The job was transferred to the CP successfully. The connection abort was triggered.
	0000 0002h	This job could not be transferred to the CP because the service was not started on the CP (for example CP in STOP).
RESULT 2	0000 0000h	Default

5.2.5 FC 62 - C_CNTR - Querying the Connection Status

Description

Query a connection status with FC 62. The current status of the communication that has been determined via ID is queried after the system function has been called with value 1 at the control input *EN_R*.

Parameters

Parameter	Declaration	Data Type	Memory Area	Description
EN_R	INPUT	BOOL	I, Q, M, D, constant	Control parameter enabled to receive, signals ready to receive if the input is set.
ID	INPUT	WORD	M, D, constant	Addressing parameter *ID*,
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Error information
ERROR	OUTPUT	BOOL	I, Q, M, D, L	Status parameter *ERROR* and *STATUS*

Parameter	Declaration	Data Type	Memory Area	Description
STATUS	OUTPUT	WORD	I, Q, M, D, L	■ *ERROR* = 0 and *STATUS* have the values: – 0000h: Neither warning nor error – <> 0000h: Warning, *STATUS* supplies detailed information. ■ *ERROR* = 1 – There is an error. *STATUS* supplies detailed information on the type of error.
C_CONN	OUTPUT	BOOL	I, Q, M, D, L	Status of the corresponding connection. Possible values: ■ 0: The connection was dropped or it is not up. ■ 1: Connection is established.
C_STATUS	OUTPUT	WORD	I, Q, M, D, L	Connection status: ■ W#16#0000: Connection is not established ■ W#16#0001: Connection is being established ■ W#16#0002: Connection is established ■ W#16#000F: No data on connection status available (such as at CP startup) ■ W#16#00FF: Connection is not configured

Error Information

The output parameter *RET_VAL* can assume the following values at FC 62 C_CNTRL:

■ 0000h: No error when FC was executed.
■ 8000h: Error when FC was executed.

> *The output parameters ERROR and STATUS are to be evaluated regardless of the output parameter RET_VAL showing the value 0000h.*

ERROR	STATUS (decimal)	Description
1	10	CP access error. Another job is currently running. Repeat job later.
1	27	There is no function code in the CPU for this block.

5.2.6 FB/SFB 8 - FB 55 - Overview

With the Siemens S7 connection large data sets may be transferred between via Ethernet connected PLC systems based on Siemens STEP7®. The communication connections are static i.e. they are to be configured in a connection table.

Possibilities of communication functions

- Siemens S7-300 communication functions
  - By including the VIPA specific function blocks FB 8 ... FB 55 you get access to the Siemens S7-300 communication functions. ✋ *Chap. 3 'Include library' page 7*
- Siemens S7-400 communication functions
  - To deploy the Siemens S7-400 communication functions the in the operating system of the CPU integrated system function blocks SFB 8 ... SFB 23 should be used. Here copy the interface description of the SFBs from the standard library at system function block to the directory container, generate an instance data block for each call and call the SFB with the associated instance data block.

Project engineering

Precondition for the Siemens S7 communication is a configured connection table, which contains the defined connections for communication. For this e.g. WinPLC7 from VIPA or NetPro from Siemens can be used. A communication connection is specified by a connection ID for each connection partner. Use the local ID to initialize the FB/SFB in the PLC from which the connection is regarded and the partner ID to configure the FB/SFB in the partner PLC.

Function blocks

FB/SFB	Designation	Description
FB/SFB 8	USEND	Uncoordinated data transmission
FB/SFB 9	URCV	Uncoordinated data reception
FB/SFB 12	BSEND	Sending data in blocks
FB/SFB 13	BRCV	Receiving data in blocks
FB/SFB 14	GET	Remote CPU read
FB/SFB 15	PUT	Remote CPU write
FB 55	IP_CONF	Programmed communication connections

Please use for the Siemens S7 communication exclusively the FB/SFBs listed here. The direct call of the associated internal SFCs leads to errors in the corresponding instance DB!

5.2.7 FB/SFB 8 - USEND - Uncoordinated data transmission

Description

FB/SFB 8 USEND may be used to transmit data to a remote partner FB/SFB of the type URCV (FB/SFB 9). You must ensure that parameter *R_ID* of both FB/SFBs is identical. The transmission is started by a positive edge at control input *REQ* and proceeds without coordination with the partner FB/SFB.

Depending upon communication function the following behavior is present:

- Siemens S7-300 Communication (FB 8)
  - The data is sent on a rising edge at *REQ*. The parameters *R_ID, ID* and *SD_1* are transferred on each rising edge at *REQ*. After a job has been completed, you can assign new values to the *R_ID, ID* and *SD_1* parameters.
- *Siemens S7-400 Communication (SFB 8)*
  - The data is sent on a rising edge at *REQ*. The data to be sent is referenced by the parameters *SD_1 ... SD_4* but not all four send parameters need to be used.

Parameters

Parameter	Declaration	Data type	Memory block	Description
REQ	INPUT	BOOL	I, Q, M, D, L	Control parameter request, activates the exchange of data when a rising edge is applied (with respect to the most recent FB/SFB-call)
ID	INPUT	WORD	I, Q, M, D, constant	Connection reference. The *ID* must be specified in the form wxyzh.
R_ID	INPUT	DWORD	I, Q, M, D, constant	Addressing parameter *R_ID*. Format DW#16#wxyzWXYZ.
DONE	OUTPUT	BOOL	I, Q, M, D, L	Status parameter *DONE*: 0: task has not been started or it is still being executed. 1: task was executed without error.
ERROR	OUTPUT	BOOL	I, Q, M, D, L	Status parameter *ERROR*: *ERROR* = 0 + *STATUS* = 0000h - No warnings or errors *ERROR* = 0 + *STATUS* unequal to 0000h - A Warning has occurred. *STATUS* contains detailed information. *ERROR* = 1 - An error has occurred.
STATUS	OUTPUT	WORD	I, Q, M, D, L	Status parameter *STATUS*, returns detailed information about the type of error.
SD_i, $1 \leq i \leq 4$	IN_OUT	ANY	I, Q, M, D, T, C	Pointer to transmit buffer i.. Only data type BOOL is valid (Bit field not permitted), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER.

> *You must, however, make sure that the areas defined by the parameters SD_1/SD_1...SD_4 and RD_1/RD_1...RD_4 (at the corresponding partner FB/SFB URCV) agree in Number, Length and Data type.*
>
> *The parameter R_ID must be identical at both FB/SFBs. Successful completion of the transmission is indicated by the status parameter DONE having the logical value 1.*

Error information

ERROR	STATUS (decimal)	Description
0	11	Warning: the new task is not active, since the previous task has not completed.
0	25	Communications initiated. The task is being processed.
1	1	Communication failures, e.g. ■ Connection parameters not loaded (local or remote) ■ Connection interrupted (e.g. cable, CPU turned off, CP in STOP)
1	4	Error in transmission range pointers SD_i with respect to the length or the data type.
1	10	Access to local application memory not possible (e.g. access to deleted DB).
1	12	The call to the FB/SFB ■ contains an instance DB that does not belong to the FB/SFB 8 ■ contains a global DB instead of an instance DB ■ could not locate an instance DB (load a new instance DB from the PG)
1	18	*R_ID* already exists in the connection *ID*.
1	20	Not enough memory.

Data consistency

To ensure the data consistency is not compromised, can the currently used transmission ranges SD_i be described again only if the current job is completed. This requires that the DONE parameter is evaluated. This is the case when the value of the status parameter *DONE* changes to 1.

5.2.8 FB/SFB 9 - URCV - Uncoordinated data reception

Description

FB/SFB 9 URCV can be used to receive data asynchronously from a remote partner FB/SFB of the type USEND (FB/SFB 8). You must ensure that parameter *R_ID* of both FB/SFBs is identical. The block is ready to receive then there is a logical 1 at the *EN_R* input. An active job can be cancelled with *EN_R=0*.

Depending upon communication function the following behavior is present:

■ Siemens S7-300 Communication (FB 9)
  – The parameters *R_ID, ID* and *RD_1* are applied with every positive edge on *EN_R*. After a job has been completed, you can assign new values to the *R_ID, ID* and *RD_1* parameters.
■ Siemens S7-400 Communication (SFB 9)
  – The receive data areas are referenced by the parameters *RD_1...RD_4*.

Parameters

Parameter	Declaration	Data type	Memory block	Description
EN_R	INPUT	BOOL	I, Q, M, D, L	Control parameter enabled to receive, indicates that the partner is ready for reception
ID	INPUT	WORD	I, Q, M, D, constant	A reference for the connection. Format wxyzh
R_ID	INPUT	DWORD	I, Q, M, D, constant	Address parameter *R_ID*. Format DW#16#wxyzWXYZ.
NDR	OUTPUT	BOOL	I, Q, M, D, L	Status parameter *NDR*: new data transferred.
ERROR	OUTPUT	BOOL	I, Q, M, D, L	Status parameter *ERROR*: ■ *ERROR* = 0 + *STATUS* = 0000h – No warnings or errors ■ *ERROR* = 0 + *STATUS* unequal to 0000h – A Warning has occured. *STATUS* contains detailed information. ■ *ERROR* = 1 – An error has occurred.
STATUS	OUTPUT	WORD	I, Q, M, D, L	Status parameter *STATUS*, returns detailed information about the type of error.
RD_i,1≤ i ≤4	IN_OUT	ANY	I, Q, M, D, T, C	Pointer to receive buffer i. Only data type BOOL is valid (Bit field not permitted), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER.

> *The quantity, length and data type of the buffer areas defined by parameters SD_i and RD_i, 1 ≤ i ≤ 4 must be identical (RD_i is the receive buffer of the respective partner FB/SFB, see FB/SFB 8). The initial call to FB/SFB 9 creates the "receive box". The receive data available during any subsequent calls must fit into this receive box. When a data transfer completes successfully parameter NDR is set to 1.*

Error information

ERROR	STATUS (decimal)	Description
0	9	Overrun warning: old receive data was overwritten by new receive data.
0	11	Warning: the new task is not active since the previous task has not completed.
0	25	Communications initiated. The task is being processed.

ERROR	STATUS (decimal)	Description
1	1	Communication failures, e.g. ■ Connection parameters not loaded (local or remote) ■ Connection interrupted (e.g. cable, CPU turned off, CP in STOP)
1	4	Error in receive buffer pointer *RD_i* with respect to the length or the data type.
1	10	Access to local application memory not possible (e.g. access to deleted DB).
1	12	The call to the FB/SFB ■ contains an instance DB that does not belong to the FB/SFB 9 ■ contains a global DB instead of an instance DB ■ could not locate an instance DB (load a new instance DB from the PG)
1	18	*R_ID* already exists in the connection ID.
1	19	The respective FB/SFB USEND transmits data quicker than FB/SFB URCV can copy the data into the receive buffers.
1	20	Not enough memory.

Data consistency

The data are received consistently if you remember the following points:

■ Siemens S7-300 Communication:
  – After the status parameter *NDR* has changed to the value 1, you must immediately call FB 9 URCV again with the value 0 at *EN_R*. This ensures that the receive area is not overwritten before you have evaluated it. Evaluate the receive area (RD_1) completely before you call the block with the value 1 at control input *EN_R*).
■ Siemens S7-400 Communication:
  – After the status parameter *NDR* has changed to the value 1, there are new receive data in your receive areas (*RD_i*). A new block call may cause these data to be overwritten with new receive data. If you want to prevent this, you must call SFB 9 URCV (such as with cyclic block processing) with the value 0 at *EN_R* until you have finished processing the receive data.

5.2.9 FB/SFB 12 - BSEND - Sending data in blocks

Description

FB/SFB 12 BSEND sends data to a remote partner FB/SFB of the type BRCV (FB/SFB 13). The data area to be transmitted is segmented. Each segment is sent individually to the partner. The last segment is acknowledged by the partner as it is received, independently of the calling up of the corresponding FB/SFB/FB BRCV. With this type of data transfer, more data can be transported between the communications partners than is possible with all other communication FBs/SFBs for configured S7 connections, namely 65534 bytes.

> *Please note that this block calls the FC or SFC 202 AG_BSEND internally. These must not be overwritten! The direct call of an internal block leads to errors in the corresponding instance DB!*

Depending upon communication function the following behavior is present:

- *Siemens S7-300 Communication (FB 12)*
  - The send job is activated on a rising edge at *REQ*. The parameters *R_ID*, I*D, SD_1* and LEN are transferred on each positive edge at *REQ*. After a job has been completed, you can assign new values to the *R_ID*, *ID*, *SD_1* and *LEN* parameters. For the transmission of segmented data the block must be called periodically in the user program. The start address and the maximum length of the data to be sent are specified by *SD_1*. You can determine the job-specific length of the data field with *LEN*.
- *Siemens S7-400 Communication (SFB 12)*
  - The send job is activated after calling the block and when there is a rising edge at *REQ*. Sending the data from the user memory is carried out asynchronously to the processing of the user program. The start address and the maximum length of the data to be sent are specified by *SD_1*. You can determine the job-specific length of the data field with *LEN*. In this case, *LEN* replaces the length section of *SD_1*.

Function

- If there is a rising edge at control input *R*, the current data transfer is cancelled.
- Successful completion of the transfer is indicated by the status parameter *DONE* having the value 1.
- A new send job cannot be processed until the previous send process has been completed if the status parameter *DONE* or *ERROR* have the value 1.
- Due to the asynchronous data transmission, a new transmission can only be initiated if the previous data have been retrieved by the call of the partner FB/SFB. Until the data are retrieved, the status value 7 will be given when the FB/SFB BSEND is called.

> *The parameter R_ID must be identical at the two corresponding FBs/ SFBs.*

Parameters

Parameter	Declaration	Data type	Memory block	Description
REQ	INPUT	BOOL	I, Q, M, D, L	Control parameter request, a rising edge activates the data exchange (with respect to the most recent FB/SFB call)
R	INPUT	BOOL	I, Q, M, D, L, constant	control parameter reset: terminates the active task

Parameter	Declaration	Data type	Memory block	Description
ID	INPUT	WORD	I, Q, M, D, constant	A reference for the connection. Format W#16#xxxx
R_ID	INPUT	DWORD	I, Q, M, D, L, constant	Address parameter *R_ID*. Format DW#16#wxyzWXYZ.
DONE	OUTPUT	BOOL	I, Q, M, D, L	Status parameter *DONE*: 0: task has not been started or is still being executed. 1: task was executed without error.
ERROR	OUTPUT	BOOL	I, Q, M, D, L	Status parameter *ERROR*: ■ *ERROR* = 0 + *STATUS* = 0000h – No warnings or errors. ■ *ERROR* = 0 + *STATUS* unequal to 0000h – A Warning has occurred. *STATUS* contains detailed information. ■ *ERROR* = 1 – An error has occurred.
STATUS	OUTPUT	WORD	I, Q, M, D, L	Status parameter *STATUS*, returns detailed information about the type of error.
SD_1	IN_OUT	ANY	I, Q, M, D, T, C	Pointer to the send data buffer. The length parameter is only utilized when the block is called for the first time after a start. It specifies the maximum length of the send buffer. Only data type BOOL is valid (Bit field not permitted), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER.
LEN	IN_OUT	WORD	I, Q, M, D, L	The length of the send data block in bytes.

Error information

ERROR	STATUS (decimal)	Description
0	11	Warning: the new task is not active since the previous task has not completed.
0	25	The communication process was initiated. The task is being processed.
1	1	Communication failures, e.g.: ■ Connection parameters not loaded (local or remote) ■ Connection interrupted (e.g. cable, CPU turned off, CP in STOP)
1	2	Negative acknowledgment received from the partner FB/SFB. The function cannot be executed.
1	3	*R_ID* is not available to the communication link specified by ID or the receive block has never been called.
1	4	Error in send buffer pointer *SD_1* with respect to the length or the data type, or parameter *LEN* was set to 0 or an error has occurred in the receive data buffer pointer *RD_1* of the respective FB/SFB 13 BRCV

ERROR	STATUS (dec-imal)	Description
1	5	Reset request was executed.
1	6	The status of the partner FB/SFB is DISABLED (*EN_R* has a value of 0)
1	7	The status of the partner FB/SFB is not correct (the receive block has not been called after the most recent data transfer).
1	8	Access to the remote object in application memory was rejected.
1	10	Access to local application memory not possible (e.g. access to deleted DB).
1	12	The call to the FB/SFB ■ contains an instance DB that does not belong to the FB/SFB 12 ■ contains a global DB instead of an instance DB ■ could not locate an instance DB (load a new instance DB from the PG)
1	18	*R_ID* already exists in the connection ID.
1	20	Not enough memory.

Data consistency

To guarantee consistent data the segment of send buffer *SD_1* that is currently being used can only be overwritten when current send process has been completed. For this purpose the program can test parameter *DONE*.

5.2.10   FB/SFB 13 - BRCV - Receiving data in blocks

Description

The FB/SFB 13 BRCV can receive data from a remote partner FB/SFB of the type BSEND (FB/SFB 12). The parameter *R_ID* of both FB/SFBs must be identical. After each received data segment an acknowledgment is sent to the partner FB/SFB and the *LEN* parameter is updated.

> *Please note that this block calls the FC or SFC 203 AG_BRCV internally. These must not be overwritten! The direct call of an internal block leads to errors in the corresponding instance DB!*

Depending upon communication function the following behavior is present:

- *Siemens S7-300 Communication (FB 13)*
  - The parameters *R_ID*, *ID* and *RD_1* are applied with every positive edge on *EN_R*. After a job has been completed, you can assign new values to the *R_ID*, *ID* and *RD_1* parameters. For the transmission of segmented data the block must be called periodically in the user program.
- *Siemens S7-400 Communication (SFB 13)*
  - Receipt of the data from the user memory is carried out asynchronously to the processing of the user program.

Parameters

Parameter	Declaration	Data type	Memory block	Description
EN_R	INPUT	BOOL	I, Q, M, D, L, constant	control parameter enabled to receive, indicates that the partner is ready for reception
ID	INPUT	WORD	I, Q, M, D, constant	A reference for the connection. Format: W#16#xxxx
R_ID	INPUT	DWORD	I, Q ,M, D, L, constant	Address parameter *R_ID*. Format: DW#16#wxyzWXYZ
NDR	OUTPUT	BOOL	I, Q, M, D, L	Status parameter NDR: new data accepted.
ERROR	OUTPUT	BOOL	I, Q, M, D, L	Status parameter *ERROR*: ■ *ERROR* = 0 + *STATUS* = 0000h – No warnings or errors. ■ *ERROR* = 0 + *STATUS* unequal to 0000h – A Warning has occurred. *STATUS* contains detailed information. ■ *ERROR* = 1 – An error has occurred.
STATUS	OUTPUT	WORD	I, Q, M, D ,T, C	Status parameter *STATUS*, returns detailed information about the type of error.
RD_1	IN_OUT	ANY	I, Q, M, D ,T, C	Pointer to the receive data buffer. The length specifies the maximum length for the block that must be received. Only data type BOOL is valid (Bit field not permitted), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER.
LEN	IN_OUT	WORD	I, Q, M, D, L	Length of the data that has already been received.

Function

■ The FB/SFB 13 is ready for reception when control input *EN_R* is set to 1. Parameter *RD_1* specifies the start address of the receive data buffer. An acknowledgment is returned to the partner FB/SFB after reception of each data segment and parameter *LEN* of the FB/SFB 13 is updated accordingly. If the block is called during the asynchronous reception process a warning is issued via the status parameter *STATUS*.

■ Should this call be received with control input *EN_R* set to 0 then the receive process is terminated and the FB/SFB is reset to its initial state. When all data segments have been received without error parameter *NDR* is set to 1. The received data remains unaltered until FB/SFB 13 is called again with parameter *EN_R* = 1.

Error information

ERROR	STATUS (decimal)	Description
0	11	Warning: the new task is not active since the previous task has not completed.
0	17	Warning: block is receiving asynchronous data.
0	25	Communications has been initiated. The task is being processed.
1	1	Communication failures, e.g. ■ Connection parameters not loaded (local or remote) ■ Connection interrupted (e.g. cable, CPU turned off, CP in STOP)
1	2	Function cannot be executed.
1	4	Error in the receive data block pointer *RD_1* with respect to the length or the data type (the send data block is larger than the receive data block).
1	5	Reset request received, incomplete data transfer.
1	8	Access to the remote object in application memory was rejected.
1	10	Access to local application memory not possible (e.g. access to deleted DB).
1	12	The call to the FB/SFB ■ contains an instance DB that does not belong to the FB/SFB 13 ■ contains a global DB instead of an instance DB ■ could not locate an instance DB (load a new instance DB from the PG)
1	18	*R_ID* already exists in the connection *ID*.
1	20	Not enough memory.

Data consistency

To guarantee data consistency during reception the following points must be met:

■ When copying has been completed (parameter *NDR* is set to 1) FB/SFB 13 must again be called with parameter *EN_R* set to 0 in order to ensure that the receive data block is not overwritten before it has bee evaluated.

■ The most recently used receive data block *RD_1* must have been evaluated completely before the block is denoted as being ready to receive (calls with parameter *EN_R* set to 1).

Receiving Data S7-400

- If a receiving CPU with a BRCV block ready to accept data (that is, a call with *EN_R* = 1 has already been made) goes into STOP mode before the corresponding send block has sent the first data segment for the job, the following will occur:
- The data in the first job after the receiving CPU has gone into STOP mode are fully entered in the receive area.
- The partner SFB BSEND receives a positive acknowledgment.
- Any additional BSEND jobs can no longer be accepted by a receiving CPU in STOP mode.
- As long as the CPU remains in STOP mode, both *NDR* and *LEN* have the value 0.
- To prevent information about the received data from being lost, you must perform a hot restart of the receiving CPU and call SFB 13 BRCV with *EN_R* = 1.

5.2.11 FB/SFB 14 - GET - Remote CPU read

Description

The FB/SFB 14 GET can be used to read data from a remote CPU. The respective CPU must be in RUN mode or in STOP mode.

> *Please note that this block calls the FC or SFC 200 AG_GET internally. These must not be overwritten! The direct call of an internal block leads to errors in the corresponding instance DB!*

Depending upon communication function the following behavior is present:

- *Siemens S7-300 Communication (FB 14)*
  - The data is read on a rising edge at *REQ*. The parameters *ID*, *ADDR_1* and *RD_1* are transferred on each rising edge at *REQ*. After a job has been completed, you can assign new values to the *ID*, *ADDR_1* and *RD_1* parameters.
- *Siemens S7-400 Communication (SFB 14)*
  - The SFB is started with a rising edge at *REQ*. In the process the relevant pointers to the areas to be read out (*ADDR_i*) are sent to the partner CPU.

Parameters

Parameter	Declaration	Data type	Memory block	Description
REQ	INPUT	BOOL	I, Q, M, D, L	control parameter request, a rising edge activates the data exchange (with respect to the most recent FB/SFB-call)
ID	INPUT	WORD	I, Q, M, D, constant	A reference for the connection. Format: W#16#xxxx
NDR	OUTPUT	BOOL	I, Q, M, D, L	Status parameter *NDR*: data from partner CPU has been accepted.
ERROR	OUTPUT	BOOL	I, Q, M, D, L	Status parameter *ERROR*: ■ *ERROR* = 0 + *STATUS* = 0000h – No warnings or errors. ■ *ERROR* = 0 + *STATUS* unequal to 0000h – A Warning has occurred. *STATUS* contains detailed information. ■ *ERROR* = 1 – An error has occurred.

Parameter	Declaration	Data type	Memory block	Description
STATUS	OUTPUT	WORD	I, Q, M, D, L	Status parameter *STATUS*, returns detailed information about the type of error.
ADDR_1	IN_OUT	ANY	e.g. I, Q, M, D	Pointer indicating the buffers in the partner CPU that must be read
ADDR_2	IN_OUT	ANY	e.g. I, Q, M, D	Pointer indicating the buffers in the partner CPU that must be read
ADDR_3	IN_OUT	ANY	e.g. I, Q, M, D	Pointer indicating the buffers in the partner CPU that must be read
ADDR_4	IN_OUT	ANY	e.g. I, Q, M, D	Pointer indicating the buffers in the partner CPU that must be read
RD_i,1≤ I ≤4	IN_OUT	ANY	I, Q, M, D, T, C	Pointers to the area of the local CPU in which the read data are entered. Only data type BOOL is valid (bit field not permitted), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER.

Function

- The remote CPU returns the data and the answer is checked for access problems during the read process for the data. The data type is checked in addition.
- When a data transfer error is detected the received data are copied into the configured receive data buffer (*RD_i*) with the next call to FB/SFB 14 and parameter *NDR* is set to 1.
- It is only possible to activate a new read process when the previous read process has been completed. You must ensure that the defined parameters on the *ADDR_i* and *RD_i* areas and the number that fit in quantity, length and data type of data to each other.

Error information

ERROR	STATUS (decimal)	Description
0	11	Warning: the new task is not active since the previous task has not completed.
0	25	The communication process was initiated. The task is being processed.
1	1	Communication failures, e.g. ■ Connection parameters not loaded (local or remote) ■ Connection interrupted (e.g.: cable, CPU turned off, CP in STOP)
1	2	Negative acknowledgment from partner device. The function cannot be executed.
1	4	Error in receive data buffer pointer *RD_i* with respect to the length or the data type.
1	8	Partner CPU access error
1	10	Access to local application memory not possible (e.g. access to deleted DB).

ERROR	STATUS (decimal)	Description
1	12	The call to the FB/SFB ■ contains an instance DB that does not belong to the FB/SFB 14 ■ contains a global DB instead of an instance DB ■ could not locate an instance DB (load a new instance DB from the PG)
1	20	Not enough memory.

Data consistency

The data are received consistently if you evaluate the current use of range *RD_i* completely before initiating another job.

5.2.12 FB/SFB 15 - PUT - Remote CPU write

Description

The FB/SFB 15 PUT can be used to write data to a remote CPU. The respective CPU may be in RUN mode or in STOP mode.

> *Please note that this block calls the FC or SFC 201 AG_PUT internally. These must not be overwritten! The direct call of an internal block leads to errors in the corresponding instance DB!*

Depending upon communication function the following behavior is present:

■ *Siemens S7-300 Communication (FB 15)*
   – The data is sent on a rising edge at *REQ*. The parameters *ID*, *ADDR_1* and *SD_1* are transferred on each rising edge at *REQ*. After a job has been completed, you can assign new values to the *ID*, *ADDR_1* and *SD_1* parameters.
■ *Siemens S7-400 Communication (SFB 15)*
   – The SFB is started on a rising edge at *REQ*. In the process the pointers to the areas to be written (*ADDR_i*) and the data (*SD_i*) are sent to the partner CPU.

Parameters

Parameter	Declaration	Data type	Memory block	Description
REQ	INPUT	BOOL	I, Q, M, D, L	control parameter request, a rising edge activates the data exchange (with respect to the most recent FB/SFB-call)
ID	INPUT	WORD	I, Q, M, D, constant	A reference for the connection. Format W#16#xxxx
DONE	OUTPUT	BOOL	I, Q, M, D, L	Status parameter *DONE*: function completed.

Parameter	Declaration	Data type	Memory block	Description
ERROR	OUTPUT	BOOL	I, Q, M, D, L	Status parameter *ERROR*: ■ *ERROR* = 0 + *STATUS* = 0000h – No warnings or errors. ■ *ERROR* = 0 + *STATUS* unequal to 0000h – A Warning has occurred. *STATUS* contains detailed information. ■ *ERROR* = 1 – An error has occurred.
STATUS	OUTPUT	WORD	I, Q, M, D, L	Status parameter *STATUS*, returns detailed information about the type of error.
ADDR_1	IN_OUT	ANY	e.g. I, Q, M, D	Pointer indicating the buffers in the partner CPU into which data is written
ADDR_2	IN_OUT	ANY	e.g. I, Q, M, D	Pointer indicating the buffers in the partner CPU into which data is written
ADDR_3	IN_OUT	ANY	e.g. I, Q, M, D	Pointer indicating the buffers in the partner CPU into which data is written
ADDR_4	IN_OUT	ANY	e.g. I, Q, M, D	Pointer indicating the buffers in the partner CPU into which data is written
SD_i, 1 ≤ I ≤ 4	IN_OUT	ANY	I, Q, M, D, T, C	Pointer to the data buffers in the local CPU that contains the data that must be sent. Only data type BOOL is valid (Bit field not permitted), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER.

Function

■ The partner CPU stores the data at the respective address and returns an acknowledgment.

■ This acknowledgment is tested and when an error is detected in the data transfer parameter *DONE* is set to 1 with the next call of FB/SFB 15.

■ The write process can only be activated again when the most recent write process has been completed. The amount, length and data type of the buffer areas that were defined by means of parameters *ADDR_i* and *SD_i*, 1 ≤ I ≤ 4 must be identical.

Error information

ERROR	STATUS (decimal)	Description
0	11	Warning: the new task is not active since the previous task has not completed.
0	25	The communication process was initiated. The task is being processed.
1	1	Communication failures, e.g. ■ Connection parameters not loaded (local or remote) ■ Connection interrupted (e.g.: cable, CPU turned off, CP in STOP)
1	2	Negative acknowledgment from partner device. The function cannot be executed.

VIPA SPEED7 Library

Network Communication

Ethernet Communication > FB 55 - IP_CONF - Progr. Communication Connections

ERROR	STATUS (decimal)	Description
1	4	Error in transmission range pointers SD_i with respect to the length or the data type
1	8	Partner CPU access error
1	10	Access to local application memory not possible (e.g. access to deleted DB).
1	12	The call to the FB/SFB contains an instance DB that does not belong to the FB/SFB 15. contains a global DB instead of an instance DB. could not locate an instance DB (load a new instance DB from the PG).
1	20	Not enough memory.

Data consistency

■ *Siemens S7-300 Communication*
  – In order to ensure data consistency, send area SD_1 may not be used again for writing until the current send process has been completed. This is the case when the state parameter *DONE* has the value "1".
■ *Siemens S7-400 Communication*
  – When a send operation is activated (rising edge at *REQ*) the data to be sent from the send area SD_i are copied from the user program. After the block call, you can write to these areas without corrupting the current send data.

5.2.13 FB 55 - IP_CONF - Progr. Communication Connections

5.2.13.1 Overview

To configure flexible communication connections, the FB 55 - IP_CONF allows the program controlled transfer of data blocks with configuration data for a CP.

> *Please note that this block calls the FC or SFC 204 IP_CONF internally. These must not be overwritten! The direct call of an internal block leads to errors in the corresponding instance DB!*

Principle

Configuration data for communication connections may be transferred to the CPU by the FB 55 called in the user program. The configuration DB may be loaded into the CP at any time.

Network Communication

VIPA SPEED7 Library

Ethernet Communication > FB 55 - IP_CONF - Progr. Communication Connections
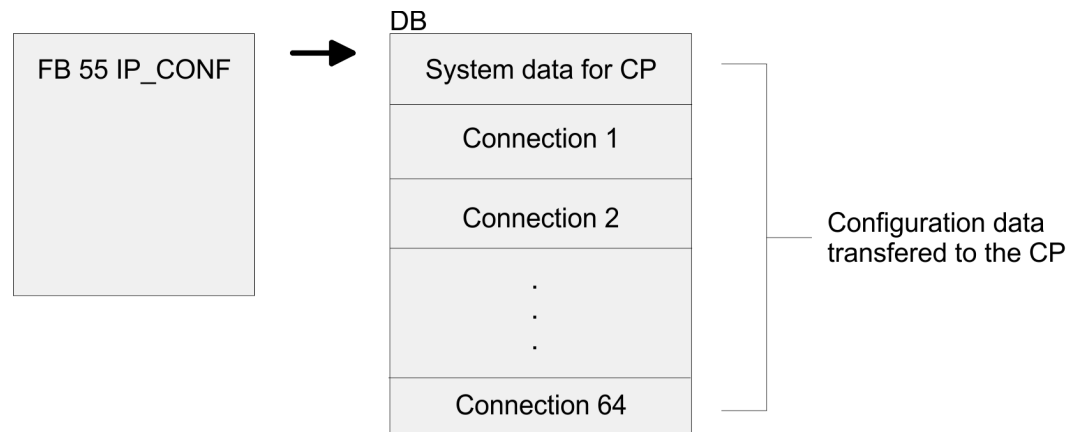
> **CAUTION!**
>
> As soon as the user program transfers the connection data via FB 55 IP_CONF, the CPU switches the CP briefly to STOP. The CP accepts the system data (including IP address) and the new connection data and processes it during startup (RUN).

5.2.13.2 FB 55 - IP_CONF

Depending on the size of the configuration DB, the data may be transferred to the CP in several segments. This means that the FB must as long be called as the FB signals complete transfer by setting the *DONE* bit to 1.

The Job is started with *ACT* = 1.

Parameters

Parameter	Declaration	Data type	Memory block	Description
ACT	INPUT	BOOL	I, Q, M, D, L	■ When the FB is called with *ACT* = 1, the DBxx is transmitted to the CP. ■ If the FB is called with *ACT* = 0, only the status codes *DONE*, *ERROR* and *STATUS* are updated.
LADDR	INPUT	WORD	I, Q, M, D, constant	Module base address When the CP is configured by the hardware configuration, the module base address is displayed in the configuration table. Enter this address here.
CONF_DB	INPUT	ANY	I, Q, M, D	The parameter points to the start address of the configuration data area in a DB.
LEN	INPUT	INT	I, Q, M, D, constant	Length information in bytes for the configuration data area.
DONE	OUTPUT	BOOL	I, Q, M, D, L	The parameter indicates whether the configuration data areas was completely transferred. Remember that it may be necessary to call the FB several times depending on the size of the configuration data area (in several cycles) until the *DONE* parameter is set to 1 to signal completion of the transfer.
ERROR	OUTPUT	BOOL	I, Q, M, D, L	Error code

VIPA SPEED7 Library

Network Communication

Ethernet Communication > FB 55 - IP_CONF - Progr. Communication Connections

Parameter	Declaration	Data type	Memory block	Description
STATUS	OUTPUT	WORD	I, Q, M, D	Status code
EXT_STATUS	OUTPUT	WORD	I, Q, M, D	If an error occurs during the execution of a job, the parameter indicates, which parameter was detected as the cause of the error in the configuration DB. ■ High byte: Index of the parameter block ■ Low byte: Index of the subfield within the parameter block

Error information

ERROR	STATUS	Description
0	0000h	Job completed without errors
0	8181h	Job active
1	80B1h	The amount of data to be sent exceeds the upper limit permitted for this service.
1	80C4h	Communication error The error can occur temporarily; it is usually best to repeat the job in the user program.
1	80D2h	Configuration error, the module you are using does not support this service.
1	8183h	The CP rejects the requested record set number.
1	8184h	System error or illegal parameter type.
1	8185h	The value of the *LEN* parameter is larger than the *CONF_DB* less the reserved header (4bytes) or the length information is incorrect.
1	8186h	Illegal parameter detected. The ANY pointer *CONF_DB* does not point to data block.
1	8187h	Illegal status of the FB. Data in the header of *CONF_DB* was possibly overwritten.
1	8A01h	The status code in the record set is invalid (value is >=3).
1	8A02h	There is no job running on the CP; however the FB has expected an acknowledgment for a competed job.
1	8A03h	There is no job running on the CP and the CP is not ready; the FB triggered the first job to read a record set.
1	8A04h	There is no job running on the CP and the CP is not ready; the FB nevertheless expected an acknowledgment for a completed job.
1	8A05h	There is a job running, but there was no acknowledgment; the FB nevertheless triggered the first job for a read record set job.
1	8A06h	A job is complete but the FB nevertheless triggered the first job for a read record sets job.
1	8B01h	Communication error, the DB could not be transferred.
1	8B02h	Parameter error, double parameter field
1	8B03h	Parameter error, the subfield in the parameter field is not permitted.
1	8B04h	Parameter error, the length specified in the FB does not match the length of the parameter fields/subfields.
1	8B05h	Parameter error, double parameter field.
1	8B06h	Parameter error, the subfield in the parameter field is not permitted.

Network Communication   VIPA SPEED7 Library

Ethernet Communication > FB 55 - IP_CONF - Progr. Communication Connections

ERROR	STATUS	Description
1	8B07h	Parameter error, the length of the parameter field is invalid.
1	8B08h	Parameter error, the ID of the subfield is invalid.
1	8B09h	System error, the connection does not exist.
1	8B0Ah	Data error, the content of the subfield is not correct.
1	8B0Bh	Structure error, a subfield exists twice.
1	8B0Ch	Data error, the parameter does not contain all the necessary parameters.
1	8B0Dh	Data error, the *CONF_DB* does not contain a parameter field for system data.
1	8B0Eh	Data error/structure error, the *CONF_DB* type is invalid.
1	8B0Fh	System error, the CP does not have enough resources to process *CONF_DB* completely.
1	8B10	Data error, configuration by the user program is not set.
1	8B11	Data error, the specified type of parameter field is invalid.
1	8B12	Data error, too many connections were specified.
1	8B13	CP internal error
1	8F22h	Area length error reading a parameter.
1	8F23h	Area length error writing a parameter.
1	8F24h	Area error reading a parameter.
1	8F25h	Area error writing a parameter.
1	8F28h	Alignment error reading a parameter.
1	8F29h	Alignment error writing a parameter.
1	8F30h	The parameter is in the write-protected first current data block.
1	8F31h	The parameter is in the write-protected second current data block.
1	8F32h	The parameter contains a DB number that is too high.
1	8F33h	DB number error
1	8F3Ah	The target area was not loaded (DB).
1	8F42h	Timeout reading a parameter from the I/O area.
1	8F43h	Timeout writing a parameter from the I/O area.
1	8F44h	Address of the parameter to be read is disabled in the accessed rack.
1	8F45h	Address of the parameter to be written is disabled in the accessed rack.
1	8F7Fh	Internal error

5.2.13.3   Configuration Data Block

The configuration data block (*CONF_DB*) contains all the connection data and configuration data (IP address, subnet mask, default router, NTP time server and other parameters) for an Ethernet CP. The configuration data block is transferred to the CP with function block FB 55.

VIPA SPEED7 Library   **Network Communication**

Ethernet Communication > FB 55 - IP_CONF - Progr. Communication Connections

Structure

The *CONF_DB* can start at any point within a data block as specified by an offset range. The connections and specific system data are described by an identically structured parameter field.
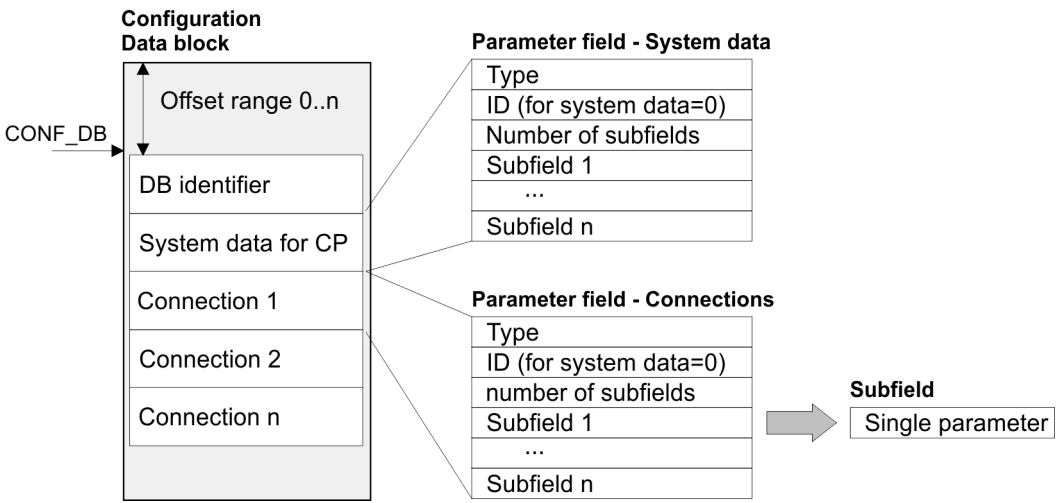
Parameter field for system data for CP

Below, there are the subfields that are relevant for networking the CP. These must be specified in the parameter field for system data. Some applications do not require all the subfield types.

Structure

Type = 0
ID = 0
Number of subfields = n
Subfield 1
Subfield 2
Subfield n

Subfield				Parameter	
ID	Type	Length (byte)	Description	Special features	Use
1	SUB_IP_V4	4 + 4	IP address of the local station according to IPv4		mandatory
2	SUB_NETMASK	4 + 4	Subnet mask of the local station		mandatory
4	SUB_DNS_SERV_ADDR	4 + 4	DNS Server Address	This subfield can occur to 4 times. The first entry is the primary DNS server.	optional
8	SUB_DEF_ROUTER	4 + 4	IP address of the default router		optional
14	SUB_DHCP_ENABLE	4 + 1	Obtain an IP address from a DHCP	0: no DHCP 1: DHCP	optional

Network Communication   VIPA SPEED7 Library

Ethernet Communication > FB 55 - IP_CONF - Progr. Communication Connections

Subfield				Parameter	
ID	Type	Length (byte)	Description	Special features	Use
15	SUB_CLIENT_ID	Length Client-ID + 4	-	-	optional
51	MAC-ADR	4 + 6	MAC address local node		optional

Parameter fields for Connections

There is shown below which values are needed to be entered in the parameter fields and which subfields are to be used for the various connection types. Some applications do not require all the subfield types. The ID parameter that precedes each connection parameter field beside the type ID is particularly important. On programmed connections this ID may freely be assigned within the permitted range of values. For identification of the connection this ID is to be used on the call interface of the FCs for the SEND/RECV.

Range of values for the connection ID: 1, 2 ... 64

TCP connection

Type = 1
ID = Connection ID
Number of subfields = n
Subfield 1
Subfield 2
Subfield n

Subfield				Parameter	
ID	Type	Length (byte)	Description	Special features	Use
1	SUB_IP_V4	4 + 4	IP address of the remote station according to IPv4		mandatory[1]
9	SUB_LOC_PORT	4 + 2	Port of the local station		mandatory
10	SUB_REM_PORT	4 + 2	Port of the remote station		mandatory[1]
18	SUB_CONNECT_NAME	Length Name + 4	Name of the connection		optional
19	SUB_LOC_MODE	4 + 1	Local mode of the connection, Possible values: 0x00 = SEND/REC 0x10 = S5-addressing mode for FETCH/WRITE [2] 0x80 = FETCH[2] 0x40 = WRITE [2] If you do not set the parameter, the default setting is SEND/RECV. For FETCH/WRITE a passive connection setup is necessary.		optional
21	SUB_KBUS_ADR	-	-	Value: fix 2	optional

VIPA SPEED7 Library

Network Communication

Ethernet Communication > FB 55 - IP_CONF - Progr. Communication Connections

Subfield					Parameter	
ID	Type	Length (byte)	Description		Special features	Use
22	SUB_CON_ESTABL	4 + 1	Type of connection establishment. With this option, you specify whether the connection is established by this station. Possible values: 0 = passive 1 = active			mandatory

1) Option using passive connection

2) the coding may be combined with OR operations

UDP connection

Type = 2
ID = Connection ID
Number of subfields = n
Subfield 1
Subfield 2
Subfield n

Subfield				Parameter	
ID	Type	Length (byte)	Description	Special features	Use
1	SUB_IP_V4	4 + 4	IP address of the remote station according to IPv4		mandatory
9	SUB_LOC_PORT	4 + 2	Port of the local station		mandatory
10	SUB_REM_PORT	4 + 2	Port of the remote station		mandatory
18	SUB_CON-NECT_NAME	Length Name + 4	Name of the connection		optional
19	SUB_LOC_MODE	4 + 1	Local mode of the connection Possible values: 0x00 = SEND/REC0x10 = S5-addressing mode for FETCH/WRITE [1] 0x80 = FETCH [1] 0x40 = WRITE [1] If you do not set the parameter, the default setting is SEND/RECV. For FETCH/WRITE a passive connection setup is necessary		optional
21	SUB_KBUS_ADR	-	-	Value: fix 2	optional

Network Communication

VIPA SPEED7 Library

Ethernet Communication > FB 55 - IP_CONF - Progr. Communication Connections

Subfield				Parameter	
ID	**Type**	**Length (byte)**	**Description**	**Special features**	**Use**
23	SUB_ADDR_IN_DATA_ BLOCK	4 + 1	Select free UDP connection. The remote node is entered in the job header of the job buffer by the user program when it calls AG_SEND. This allows any node on Ethernet/LAN/WAN to be reached. Possible values: 1 = free UDP connection 0 = otherwise		optional

1) the coding may be combined with OR operations

ISO-on-TCP connection

Type = 3
ID = Connection ID
Number of subfields = n
Subfield 1
Subfield 2
Subfield n

Subfield				Parameter	
ID	**Type**	**Length (byte)**	**Description**	**Special features**	**Use**
1	SUB_IP_V4	4 + 4	IP address of the remote station according to IPv4		mandatory[1]
11	SUB_LOC_PORT	Length TSAP + 4	TSAP of the local station		mandatory
12	SUB_REM_PORT	Length TSAP + 4	TSAP of the remote station		mandatory[1]
18	SUB_CONNECT_NAME	Length Name + 4	Name of the connection		optional
19	SUB_LOC_MODE	4 + 1	Local mode of the connection Possible values: 0x00 = SEND/RECV 0x10 = S5-addressing mode for FETCH/ WRITE [2] 0x80 = FETCH [2] 0x40 = WRITE [2] If you do not set the parameter, the default setting is SEND/RECV. For FETCH/WRITE a passive connection setup is necessary		optional
21	SUB_KBUS_ADR	-	-	Value: fix 2	optional

VIPA SPEED7 Library   **Network Communication**

Ethernet Communication > FB 55 - IP_CONF - Progr. Communication Connections

Subfield			Parameter		
ID	**Type**	**Length (byte)**	**Description**	**Special features**	**Use**
22	SUB_CON_ESTABL	4 + 1	Type of connection establishment With this option, you specify whether the connection is established by this station. Possible values: 0 = passive 1 = active		mandatory

1) option using passive connection

2) the coding may be combined with OR operation

H1 connection (ISO)

Type = 10
ID = Connection ID
Number of subfields = n
Subfield 1
Subfield 2
Subfield n

Subfield			Parameter		
ID	**Type**	**Length (byte)**	**Description**	**Special features**	**Use**
51	SUB_MAC	4 + 6	MAC address of the remote station		mandatory
11	SUB_LOC_TSAP	Length TASP + 4	TSAP of the local station		mandatory
12	SUB_REM_TSAP	Length TASP + 4	TSAP of the remote station		mandatory[1]
18	SUB_CONNECT_NAME	Length Name + 4	Name of the connection		optional
19	SUB_LOC_MODE	4 + 1	Local mode of the connection Possible values: 0x00 = SEND/RECV 0x10 = S5-addressing mode for FETCH/WRITE [2] 0x80 = FETCH [2] 0x40 = WRITE [2] If you do not set the parameter, the default setting is SEND/RECV.For FETCH/WRITE a passive connection setup is necessary		optional

Network Communication   VIPA SPEED7 Library

Ethernet Communication > FB 55 - IP_CONF - Progr. Communication Connections

Subfield			Parameter		
ID	**Type**	**Length (byte)**	**Description**	**Special features**	**Use**
22	SUB_CON_ESTABL	4 + 1	Type of connection establishment With this option, you specify whether the connection is established by this station. Possible values: 0 = passive; 1 = active		mandatory
52	SUB_TIME_CON_RETRAN	4 + 2	Time interval after which a failed connection is established again. (1...60s, default: 5s)	irrelevant with passive connection establishment	optional
53	SUB_TIME_DAT_RETRAN	4 + 2	Time interval after which a failed send is triggered again. (100...30000ms, default: 1000ms)		optional
54		4 + 2	Number of send attempts, incl 1. attempt(1...100, Default: 5)		optional
55		4 + 2	Time interval after which a connection is released, if there is no responds of the partner station.(6...160s, default: 30s)		optional

1) option using passive connection

2) the coding may be combined with OR operation

Siemens S7 connection

Type = 11
ID = Connection ID
Number of subfields = n
Subfield 1
Subfield 2
Subfield n

Subfield			Parameter		
ID	**Type**	**Length (byte)**	**Description**	**Special features**	**Use**
56	SUB_S/_C_DETAIL	4 + 14	Connection specific parameter		mandatory
18	SUB_CONNECT_NAME	LengthName + 4	Name of the connection		optional
1	SUB_IP_V4	4 + 4	IP address according to IPv4	IP address of the remote partner	mandatory[1]
51	SUB_MAC	4 + 6	MAC address of the remote station		mandatory

VIPA SPEED7 Library

Network Communication

Ethernet Communication > FB 55 - IP_CONF - Progr. Communication Connections

Subfield				Parameter	
ID	Type	Length (byte)	Description	Special features	Use
22	SUB_CON_ESTABL	4 + 1	Type of connection establishment. With this option, you specify whether the connection is established by this station. Possible values: 0 = passive 1 = active	mandatory	

1) option using passive connection

SUB_S/_C_DETAIL

Parameter	Declaration	Data type	Description
SubBlockID	IN	WORD	ID
SubBlockLen	IN	WORD	Length
TcpIpActive	IN	INT	Connection via MAC or IP address (MAC=0, IP=1)
LocalResource	IN	WORD	Local resource 0001h ... 00DFh (1=PG, 2=OP, 0010h ... 00DFh=not specified)
LocalRack	IN	WORD	Number local rack 0000h ... 0002h
LocalSlot	IN	WORD	Number local slot 0002h ... 000Fh (2=CPU, 4=VIPA-PG/OP, 5=CP int., 6=CP ext.)
RemoteResource	IN	WORD	Remote resource 0001h ... 00DFh (1=PG, 2=OP, 0010h ... 00DFh=not specified)
RemoteRack	IN	WORD	Number remote rack 0000h ... 0002h
RemoteSlot	IN	WORD	Number remote slot 0002h ... 000Fh (2=CPU, 4=VIPA-PG/OP, 5=CP int., 6=CP ext.)

The "local TSAP" is created with *LocalResource*, *LocalRack* and *LocalSlot*.

The "remote TSAP" is created with *RemoteResource*, *RemoteRack* and *RemoteSlot*.

Example for configuring a Siemens S7 connection

The configuration of a dynamic Siemens S7 connection via IP_CONF takes place analog to the configuration of a fix Siemens S7 connection with Siemens NetPro. Based on Siemens NetPro there are the following parameters corresponding to the following subfields:

Properties - Siemens S7- Connection	
Siemens NetPro	FB55 - IP_CONFIG
establish an active connection	SUB_CON_ESATBL.CON_ESTABL
TCP/IP	SUB_S7_C_DETAILS.TcpIpActive

Network Communication

VIPA SPEED7 Library

Ethernet Communication > FB 55 - IP_CONF - Progr. Communication Connections

Properties - Siemens S7- Connection	
Siemens NetPro	**FB55 - IP_CONFIG**
IP respectively MAC address remote station	SUB_IP_V4.rem_IP.IP_0...IP_3 resp. SUB_MAC.rem_MAC.MAC_0...MAC5
Local ID	Connection ID

Address details	
Siemens NetPro	**FB55 - IP_CONFIG**
Local rack	SUB_S7_C_DETAILS.LocalRack
Local slot	SUB_S7_C_DETAILS.LocalSlot
Local resource	SUB_S7_C_DETAILS.LocalResource
Remote rack	SUB_S7_C_DETAILS.RemoteRack
Remote slot	SUB_S7_C_DETAILS.RemoteSlot
Remote resource	SUB_S7_C_DETAILS.RemoteResource

Additional Parameter fields

Block_VIPA_HWK

As soon as the Block_VIPA_HWK (special identification 99) is contained in the DB, all connections, which were parameterized in the NETPRO, are still remain. Now it is possible to change with IP_CONFIG only the system data (IP, Netmask etc.). If the special identification Block_VIPA_HWK were found, no other connecting data may be parameterized in the DB, otherwise error is announced in the RETVAL. If the Block_VIPA_HWK is not in the DB, then all connections are removed from NETPRO (as with Siemens) and the connections from this DB are only configured.

Type = 99
ID = 0
Number of subfields = 0

Block_VIPA_ BACNET

As soon as the Block_VIPA_BACNET (special identification 100) is contained in the DB, a BACNET configuration is derived from the DB and no further blocks are evaluated thereafter.

Type = 100
Number of subfields = 0

Block_VIPA_IPK

Type = 101
ID = Connection ID
Number of subfields = n
Subfield 1
Subfield 2
Subfield n

VIPA SPEED7 Library

Network Communication

Ethernet Communication > FB 55 - IP_CONF - Progr. Communication Connections

Subfield				Parameter	
ID	Type	Length (byte)	Description	Special features	Use
1	VIPA_IPK_CYCLE	4 + 4	IPK cycle time for connection ID	VIPA specific	optional

Example DB

Address	Name	Type	Initial value	Actual	Comment
0.0	DB_Ident	WORD	W#16#1	W#16#1	
2.0	Systemdaten.Typ	INT	0	0	System data
4.0	Systemdaten.VerbId	INT	0	0	fix 0
6.0	Systemdaten.SubBlock_Anzahl	INT	3	3	
8.0	Systemdaten.ip.SUB_IP_V4	WORD	W#16#1	W#16#1	
10.0	Systemdaten.ip.SUB_IP_V4_LEN	WORD	W#16#8	W#16#8	
12.0	Systemdaten.ip.IP_0	BYTE	B#16#0	B#16#AC	
13.0	Systemdaten.ip.IP_1	BYTE	B#16#0	B#16#14	
14.0	Systemdaten.ip.IP_2	BYTE	B#16#0	B#16#8B	
15.0	Systemdaten.ip.IP_3	BYTE	B#16#0	B#16#61	
16.0	Systemdaten.netmask.SUB_NETMASK	WORD	W#16#2	W#16#2	
18.0	Systemdaten.netmask.SUB_NETMASK_LEN	WORD	W#16#8	W#16#8	
20.0	Systemdaten.netmask.NETMASK_0	BYTE	B#16#0	B#16#FF	
21.0	Systemdaten.netmask.NETMASK_1	BYTE	B#16#0	B#16#FF	
22.0	Systemdaten.netmask.NETMASK_2	BYTE	B#16#0	B#16#FF	
23.0	Systemdaten.netmask.NETMASK_3	BYTE	B#16#0	B#16#0	
24.0	Systemdaten.router.SUB_DEF_ROUTER	WORD	W#16#8	W#16#8	
26.0	Systemdaten.router.SUB_DEF_ROUTER_LEN	WORD	W#16#8	W#16#8	
28.0	Systemdaten.router.ROUTER_0	BYTE	B#16#0	B#16#AC	
29.0	Systemdaten.router.ROUTER_1	BYTE	B#16#0	B#16#14	
30.0	Systemdaten.router.ROUTER_2	BYTE	B#16#0	B#16#8B	
31.0	Systemdaten.router.ROUTER_3	BYTE	B#16#0	B#16#61	
32.0	Con_TCP_ID1.Typ	INT	1	1	TCP connection
34.0	Con_TCP_ID1.VerbId	INT	0	1	Connection ID
36.0	Con_TCP_ID1.SubBlock_Anzahl	INT	4	4	
38.0	Con_TCP_ID1.ip1.SUB_IP_V4	WORD	W#16#1	W#16#1	
40.0	Con_TCP_ID1.ip1. SUB_IP_V4_LEN	WORD	W#16#8	W#16#8	
42.0	Con_TCP_ID1.ip1.IP_0	BYTE	B#16#0	B#16#AC	
43.0	Con_TCP_ID1.ip1.IP_1	BYTE	B#16#0	B#16#14	
44.0	Con_TCP_ID1.ip1.IP_2	BYTE	B#16#0	B#16#8B	
45.0	Con_TCP_ID1.ip1.IP_3	BYTE	B#16#0	B#16#62	
46.0	Con_TCP_ID1.locport.SUB_LOC_PORT	WORD	W#16#9	W#16#9	
48.0	Con_TCP_ID1.locport.SUB_LOC_PORT_LEN	WORD	W#16#6	W#16#6	
50.0	Con_TCP_ID1.locport.LOC_PORT	WORD	W#16#0	W#16#3E9	

Ethernet Communication > FB 55 - IP_CONF - Progr. Communication Connections

Address	Name	Type	Initial value	Actual	Comment
52.0	Con_TCP_ID1.remport.SUB_REM_PORT	WORD	W#16#A	W#16#A	
54.0	Con_TCP_ID1.remport.SUB_REM_PORT_LEN	WORD	W#16#6	W#16#6	
56.0	Con_TCP_ID1.remport.REM_PORT	WORD	W#16#0	W#16#3E9	
58.0	Con_TCP_ID1.con_est.SUB_CON_ESTABL	WORD	W#16#16	W#16#16	
60.0	Con_TCP_ID1.con_est.SUB_CON_ESTABL_LEN	WORD	W#16#6	W#16#6	
62.0	Con_TCP_ID1.con_est.CON_ESTABL	BYTE	B#16#0	B#16#1	
64.0	Con_ISO_ID3.Typ	INT	3	3	ISO-on-TCP connection
66.0	Con_ISO_ID3.VerbId	INT	0	3	Connection ID
68.0	Con_ISO_ID3.SubBlock_Anzahl	INT	4	4	
70.0	Con_ISO_ID3.ip1. SUB_IP_V4	WORD	W#16#1	W#16#1	
72.0	Con_ISO_ID3.ip1. SUB_IP_V4_LEN	WORD	W#16#8	W#16#8	
74.0	Con_ISO_ID3.ip1.IP_0	BYTE	B#16#0	B#16#AC	
75.0	Con_ISO_ID3.ip1.IP_1	BYTE	B#16#0	B#16#10	
76.0	Con_ISO_ID3.ip1.IP_2	BYTE	B#16#0	B#16#8B	
77.0	Con_ISO_ID3.ip1.IP_3	BYTE	B#16#0	B#16#62	
78.0	Con_ISO_ID3.loc_TSAP.SUB_LOC_PORT	WORD	W#16#B	W#16#B	
80.0	Con_ISO_ID3.loc_TSAP.SUB_LOC_PORT_LEN	WORD	W#16#A	W#16#A	
82.0	Con_ISO_ID3.loc_TSAP.LOC_TSAP[0]	BYTE	B#16#0	B#16#54	
83.0	Con_ISO_ID3.loc_TSAP.LOC_TSAP[1]	BYTE	B#16#0	B#16#53	
84.0	Con_ISO_ID3.loc_TSAP.LOC_TSAP[2]	BYTE	B#16#0	B#16#41	
85.0	Con_ISO_ID3.loc_TSAP.LOC_TSAP[3]	BYTE	B#16#0	B#16#50	
86.0	Con_ISO_ID3.loc_TSAP.LOC_TSAP[4]	BYTE	B#16#0	B#16#30	
87.0	Con_ISO_ID3.loc_TSAP.LOC_TSAP[5]	BYTE	B#16#0	B#16#31	
88.0	Con_ISO_ID3.rem_TSAP.SUB_REM_PORT	WORD	W#16#C	W#16#C	
90.0	Con_ISO_ID3.rem_TSAP.SUB_REM_PORT_LEN	WORD	W#16#A	W#16#A	
92.0	Con_ISO_ID3.rem_TSAP.REM_TSAP[0]	BYTE	B#16#0	B#16#54	
93.0	Con_ISO_ID3.rem_TSAP.REM_TSAP[1]	BYTE	B#16#0	B#16#53	
94.0	Con_ISO_ID3.rem_TSAP.REM_TSAP[2]	BYTE	B#16#0	B#16#41	
95.0	Con_ISO_ID3.rem_TSAP.REM_TSAP[3]	BYTE	B#16#0	B#16#50	
96.0	Con_ISO_ID3.rem_TSAP.REM_TSAP[4]	BYTE	B#16#0	B#16#30	
97.0	Con_ISO_ID3.rem_TSAP.REM_TSAP[5]	BYTE	B#16#0	B#16#31	
98.0	Con_ISO_ID3.con_est.SUB_CON_ESTABL	WORD	W#16#16	W#16#16	
100.0	Con_ISO_ID3.con_est.SUB_CON_ESTABL_LEN SUB_CON_ESTABL SUB_CON_ESTABL_LEN	WORD	W#16#6	W#16#6	
102.0	Con_ISO_ID3.con_est.CON_ESTABL	BYTE	B#16#0	B#16#1	
104.0	S7_Verb.Typ	INT	11	11	S7 connection
106.0	S7_Verb.Verb_ID	INT	0	0	Connection ID
108.0	S7_Verb.SubBlock_Anzahl	INT	5	5	
110.0	S7_Verb.Verb_Parameter.SUB_S7_C_DETAIL	INT	56	56	
112.0	S7_Verb.Verb_Parameter. SUB_S7_C_DETAIL_LEN	INT	18	18	
114.0	S7_Verb.Verb_Parameter.TcpIpActive	INT	0	1	

VIPA SPEED7 Library

Network Communication

Ethernet Communication > FB 55 - IP_CONF - Progr. Communication Connections

Address	Name	Type	Initial value	Actual	Comment
116.0	S7_Verb.Verb_Parameter.LocalResource	INT	0	2	
118.0	S7_Verb.Verb_Parameter.LocalRack	INT	0	0	
120.0	S7_Verb.Verb_Parameter.LocalsSlot	INT	0	2	
122.0	S7_Verb.Verb_Parameter.RemoteResource	INT	0	2	
124.0	S7_Verb.Verb_Parameter.RemoteRack	INT	0	0	
126.0	S7_Verb.Verb_Parameter.RemoteSlot	INT	0	2	
128.0	S7_Verb.ipl.SUB_IP_V4	WORD	W#16#1	W#16#1	
130.0	S7_Verb.ipl. SUB_IP_V4_LEN	WORD	W#16#8	W#16#8	
132.0	S7_Verb.ipl.IP_0	BYTE	B#16#0	B#16#AC	
133.0	S7_Verb.ipl.IP_1	BYTE	B#16#0	B#16#10	
134.0	S7_Verb.ipl.IP_2	BYTE	B#16#0	B#16#8B	
135.0	S7_Verb.ipl.IP_3	BYTE	B#16#0	B#16#62	
136.0	S7_Verb.Mac.SUB_MAC	INT	51	51	
138.0	S7_Verb.Mac.SUB_MAC_LEN	INT	10	10	
140.0	S7_Verb.Mac.MAC_0	BYTE	B#16#0	B#16#0	
141.0	S7_Verb.Mac.MAC_1	BYTE	B#16#0	B#16#20	
142.0	S7_Verb.Mac.MAC_2	BYTE	B#16#0	B#16#D5	
143.0	S7_Verb.Mac.MAC_3	BYTE	B#16#0	B#16#77	
144.0	S7_Verb.Mac.MAC_4	BYTE	B#16#0	B#16#53	
145.0	S7_Verb.Mac.MAC_5	BYTE	B#16#0	B#16#9B	
146.0	S7_Verb.con_est .SUB_CON_ESTABL	WORD	W#16#16	W#16#16	
148.0	S7_Verb.con_est.SUB_CON_ESTABL_LEN	WORD	W#16#6	W#16#6	
150.0	S7_Verb.con_est.CON_ESTABL	BYTE	B#16#0	B#16#1	
152.0	S7_Verb.name_verb.SUB_CONNECT_NAME	WORD	W#16#12	W#16#12	
154.0	S7_Verb.name_verb.SUB_CONNECT_NAME_LEN	WORD	W#16#23	W#16#23	
156.0	S7_Verb.name_verb.CONNECT_NAME[0]	CHAR	' '	'V'	Connection S7 with IP-Config 1
157.0	S7_Verb.name_verb.CONNECT_NAME[1]	CHAR	' '	'e'	
158.0	S7_Verb.name_verb.CONNECT_NAME[2]	CHAR	' '	'r'	
159.0	S7_Verb.name_verb.CONNECT_NAME[3]	CHAR	' '	'b'	
160.0	S7_Verb.name_verb.CONNECT_NAME[4]	CHAR	' '	'l'	
161.0	S7_Verb.name_verb.CONNECT_NAME[5]	CHAR	' '	'n'	
162.0	S7_Verb.name_verb.CONNECT_NAME[6]	CHAR	' '	'd'	
163.0	S7_Verb.name_verb.CONNECT_NAME[7]	CHAR	' '	'u'	
164.0	S7_Verb.name_verb.CONNECT_NAME[8]	CHAR	' '	'n'	
165.0	S7_Verb.name_verb.CONNECT_NAME[9]	CHAR	' '	'g'	
166.0	S7_Verb.name_verb.CONNECT_NAME[10]	CHAR	' '	' '	
167.0	S7_Verb.name_verb.CONNECT_NAME[11]	CHAR	' '	'S'	
168.0	S7_Verb.name_verb.CONNECT_NAME[12]	CHAR	' '	'7'	
169.0	S7_Verb.name_verb.CONNECT_NAME[13]	CHAR	' '	' '	
170.0	S7_Verb.name_verb.CONNECT_NAME[14]	CHAR	' '	'm'	
171.0	S7_Verb.name_verb.CONNECT_NAME[15]	CHAR	' '	'l'	

Ethernet Communication > FB 55 - IP_CONF - Progr. Communication Connections

Address	Name	Type	Initial value	Actual	Comment
172.0	S7_Verb.name_verb.CONNECT_NAME[16]	CHAR	' '	't'	
173.0	S7_Verb.name_verb.CONNECT_NAME[17]	CHAR	' '	' '	
174.0	S7_Verb.name_verb.CONNECT_NAME[18]	CHAR	' '	'I'	
175.0	S7_Verb.name_verb.CONNECT_NAME[19]	CHAR	' '	'P'	
176.0	S7_Verb.name_verb.CONNECT_NAME[20]	CHAR	' '	'-'	
177.0	S7_Verb.name_verb.CONNECT_NAME[21]	CHAR	' '	'C'	
178.0	S7_Verb.name_verb.CONNECT_NAME[22]	CHAR	' '	'o'	
179.0	S7_Verb.name_verb.CONNECT_NAME[23]	CHAR	' '	'n'	
180.0	S7_Verb.name_verb.CONNECT_NAME[24]	CHAR	' '	'f'	
181.0	S7_Verb.name_verb.CONNECT_NAME[25]	CHAR	' '	'I'	
182.0	S7_Verb.name_verb.CONNECT_NAME[26]	CHAR	' '	'g'	
183.0	S7_Verb.name_verb.CONNECT_NAME[27]	CHAR	' '	' '	
184.0	S7_Verb.name_verb.CONNECT_NAME[28]	CHAR	' '	'1'	
185.0	S7_Verb.name_verb.CONNECT_NAME[29]	CHAR	' '	' '	
186.0	S7_Verb.name_verb.CONNECT_NAME[30]	CHAR	' '	' '	