

VIPA SPEED7 Library

OPL_SP7-LIB | SW90GS0MA V10.002 | Manual

HB00 | OPL_SP7-LIB | SW90GS0MA V10.002 | en | 18-04

Block library - Serial Communication



VIPA GmbH
Ohmstr. 4
91074 Herzogenaurach
Telephone: +49 9132 744-0
Fax: +49 9132 744-1864
Email: info@vipa.com
Internet: www.vipa.com

Table of contents

1	General	4
1.1	Copyright © VIPA GmbH	4
1.2	About this manual.....	5
2	Important notes	6
2.1	Internally used blocks.....	6
3	Include library	7
3.1	Integration into Siemens SIMATIC Manager.....	7
3.2	Integration into Siemens TIA Portal.....	8
4	Serial Communication	9
4.1	Serial communication.....	9
4.1.1	SFC 207 - SER_CTRL - Modem functionality PtP.....	9
4.1.2	FC/SFC 216 - SER_CFG - Parametrization PtP.....	10
4.1.3	FC/SFC 217 - SER_SND - Send to PtP.....	14
4.1.4	FC/SFC 218 - SER_RCV - Receive from PtP.....	18
4.1.5	FB 1 - RECEIVE_ASCII - Receiving with defined length from PtP.....	20
4.1.6	FB 7 - P_RCV_RK - Receive from CP 341.....	20
4.1.7	FB 8 - P_SND_RK - Send to CP 341.....	22
4.2	CP040.....	23
4.2.1	FB 60 - SEND - Send to System SLIO CP 040.....	23
4.2.2	FB 61 - RECEIVE - Receive from System SLIO CP 040.....	25
4.2.3	FB 65 - CP040_COM - Communication SLIO CP 040.....	28
4.3	CP240.....	32
4.3.1	FC 0 - SEND - Send to CP 240.....	32
4.3.2	FC 1 - RECEIVE - Receive from CP 240.....	33
4.3.3	FC 8 - STEUERBIT - Modem functionality CP 240.....	34
4.3.4	FC 9 - SYNCHRON_RESET - Synchronization CPU and CP 240.....	35
4.3.5	FC 11 - ASCII_FRAGMENT - Receive fragmented from CP 240.....	36

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

E-Mail: 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 and S7-400 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

E-Mail: 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)

E-Mail: support@vipa.de

1.2 About this manual

Objective and contents

The manual describes the block library 'Serial 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



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

2.1 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 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 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 'Serial 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 Serial Communication - SW90GS0MA'. 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.

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

1. 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. Open library and transfer blocks into the project

Load ZIP file

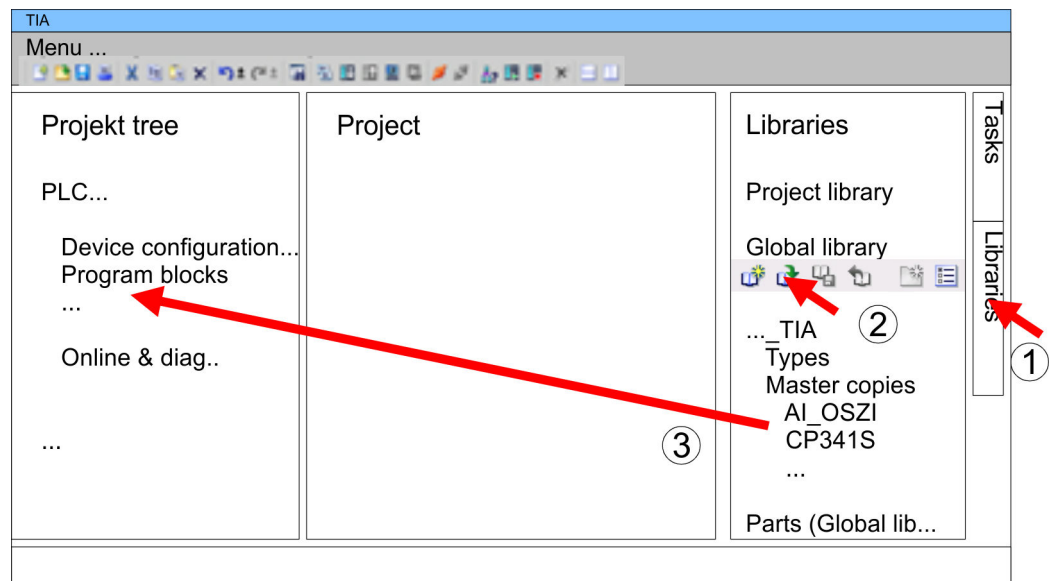
1. Navigate on the web page to the ZIP file, that matches your version of the program.
2. Load and store it in your work directory.

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 Serial Communication

4.1 Serial communication

4.1.1 SFC 207 - SER_CTRL - Modem functionality PtP

Description



Please note that this block is not supported by SPEED7 CPUs!

Using the RS232 interface by means of ASCII protocol the serial modem lines can be accessed with this SFC during operation. Depending on the parameter *FLOWCONTROL*, which is set by *SFC 216 (SER_CFG)*, this SFC has the following functionality:

	Read	Write
<i>FLOWCONTROL=0:</i>	DTR, RTS, DSR, RI, CTS, CD	DTR, RTS
<i>FLOWCONTROL>0:</i>	DTR, RTS, DSR, RI, CTS, CD	not possible

Parameters

Name	Declaration	Type	Description
WRITE	IN	BYTE	<ul style="list-style-type: none"> ■ Bit 0: New state DTR ■ Bit 1: New state RTS
MASKWRITE	IN	BYTE	<ul style="list-style-type: none"> ■ Bit 0: Set state DTR ■ Bit 1: Set state RTS
READ	OUT	BYTE	Status flags (CTS, DSR, RI, CD, DTR, RTS)
READDELTA	OUT	BYTE	Status flags of change between 2 accesses
RETVAL	OUT	WORD	Return value (0 = OK)

WRITE

With this parameter the status of DTR and RTS is set and activated by *MASKWRITE*. The byte has the following allocation:

- Bit 0 = DTR
- Bit 1 = RTS
- Bit 7 ... Bit 2: reserved

MASKWRITE

Here with "1" the status of the appropriate parameter is activated. The byte has the following allocation:

- Bit 0 = DTR
- Bit 1 = RTS
- Bit 7 ... Bit 2: reserved

READ

You get the current status by *READ*. The current status changed since the last access is returned by *READDELTA*. The bytes have the following structure:

Bit No.	7	6	5	4	3	2	1	0
Read	x	x	RTS	DTR	CD	RI	DSR	CTS
ReadDelta	x	x	x	x	CD	RI	DSR	CTS

RETVAL (Return value)

Value	Description
0000h	no error
8x24h	Error SFC parameter x, with x: <ul style="list-style-type: none"> ■ 1: Error at <i>WRITE</i> ■ 2: Error at <i>MASKWRITE</i> ■ 3: Error at <i>READ</i> ■ 4: Error at <i>READDELTA</i>
809Ah	Interface missing
809Bh	Interface not configured (SFC 216)

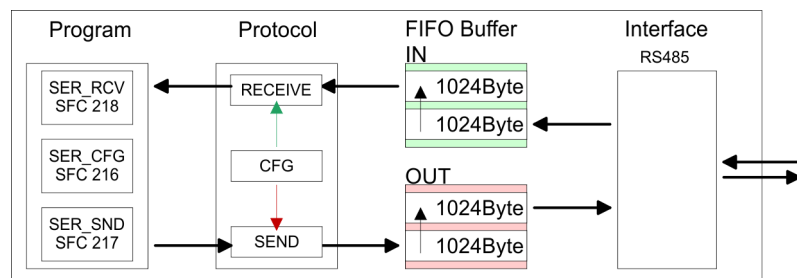
4.1.2 FC/SFC 216 - SER_CFG - Parametrization PtP

Description

You may de-activate the DP master integrated in the SPEED7-CPU via a hardware configuration using object properties and the parameter "Function RS485". Thus release the RS485 interface for PtP (point-to-point) communication. The RS485 interface supports in PtP operation the serial process connection to different source res. destination systems. The parametrization happens during runtime deploying the FC/SFC 216 (SER_CFG). For this you have to store the parameters in a DB for all protocols except ASCII.

Communication

- Data, which are written into the according data channel by the PLC, is stored in a FIFO send buffer (first in first out) with a size of 2x1024byte and then put out via the interface.
- When the interface receives data, this is stored in a FIFO receive buffer with a size of 2x1024byte and can there be read by the PLC.
- If the data is transferred via a protocol, the adoption of the data to the according protocol happens automatically. In opposite to ASCII and STX/ETX, the protocols 3964R, USS and Modbus require the acknowledgement of the partner.
- An additional call of the FC/SFC 217 SER_SND causes a return value in RETVAL that includes among others recent information about the acknowledgement of the partner. Further on for USS and Modbus after a SER_SND the acknowledgement telegram must be evaluated by call of the FC/SFC 218 SER_RCV.



Parameters

Parameter	Declaration	Data type	Description
PROTOCOL	IN	BYTE	1=ASCII, 2=STX/ETX, 3=3964R
PARAMETER	IN	ANY	Pointer to protocol-parameters
BAUDRATE	IN	BYTE	Number of baudrate
CHARLEN	IN	BYTE	0=5bit, 1=6bit, 2=7bit, 3=8bit
PARITY	IN	BYTE	0=Non, 1=Odd, 2=Even
STOPBITS	IN	BYTE	1=1bit, 2=1.5bit, 3=2bit
FLOWCONTROL	IN	BYTE	1 - see note
RETVAL	OUT	WORD	Return value (0 = OK)

All time settings for timeouts must be set as hexadecimal value. Find the Hex value by multiply the wanted time in seconds with the baudrate.

Example:

- Wanted time 8ms at a baudrate of 19200baud
- Calculation: $19200\text{bit/s} \times 0.008\text{s} \approx 154\text{bit} \rightarrow (9\text{Ah})$
- Hex value is 9Ah.

PROTOCOL

Here you fix the protocol to be used. You may choose between:

- 1: ASCII
- 2: STX/ETX
- 3: 3964R
- 4: USS Master
- 5: Modbus RTU Master
- 6: Modbus ASCII Master

PARAMETER (as DB)

At ASCII protocol, this parameter is ignored. At STX/ETX, 3964R, USS and Modbus you fix here a DB that contains the communication parameters and has the following structure for the according protocols:

Data block at STX/ETX			
DBB0:	STX1	BYTE	(1. Start-ID in hexadecimal)
DBB1:	STX2	BYTE	(2. Start-ID in hexadecimal)
DBB2:	ETX1	BYTE	(1. End-ID in hexadecimal)
DBB3:	ETX2	BYTE	(2. End-ID in hexadecimal)
DBW4:	TIMEOUT	WORD	(max. delay time between 2 telegrams)



The start res. end sign should always be a value <20, otherwise the sign is ignored!

With not used IDs please always enter FFh!

Data block at 3964R			
DBB0:	Prio	BYTE	(The priority of both partners must be different)
DBB1:	ConnAttmptNr	BYTE	(Number of connection trials)
DBB2:	SendAttmptNr	BYTE	(Number of telegram retries)
DBB4:	CharTimeout	WORD	(Char. delay time)
DBW6:	ConfTimeout	WORD	(Acknowledgement delay time)

Data block at USS			
DBW0:	Timeout	WORD	(Delay time)

Data block at Modbus master			
DBW0:	Timeout	WORD	(Respond delay time)

BAUDRATE

Velocity of data transfer in bit/s (baud)							
04h:	1200baud	05h:	1800baud	06h:	2400baud	07h:	4800baud
08h:	7200baud	09h:	9600baud	0Ah:	14400baud	0Bh:	19200baud
0Ch:	38400baud	0Dh:	57600baud	0Eh:	115200baud		

CHARLEN

Number of data bits where a character is mapped to.			
0: 5bit	1: 6bit	2: 7bit	3: 8bit

PARITY

The parity is -depending on the value- even or odd. For parity control, the information bits are extended with the parity bit, that amends via its value ("0" or "1") the value of all bits to a defined status. If no parity is set, the parity bit is set to "1", but not evaluated.

0: NONE	1: ODD	2: EVEN
---------	--------	---------

STOPBITS

The stop bits are set at the end of each transferred character and mark the end of a character.

1: 1bit	2: 1.5bit*	3: 2bit
---------	------------	---------

*) Only permitted when *CHARLEN* = 0 (5bit)

FLOWCONTROL

The parameter *FLOWCONTROL* is ignored. When sending RTS=1, when receiving RTS=0.

**Special function in System MICRO CPU**

From firmware version 2.8.1 with a System MICRO CPU you can switch between RS422 and RS485 communication.

0: RS422 communication

1: RS485 communication

**RETVL FC/SFC 216
(Return values)**

Return values send by the block:

Error code	Description
0000h	no error
809Ah	Interface not found e. g. interface is used by PROFIBUS
8x24h	Error at FC/SFC-Parameter x, with x: 1: Error at <i>PROTOCOL</i> 2: Error at <i>PARAMETER</i> 3: Error at <i>BAUDRATE</i> 4: Error at <i>CHARLENGTH</i> 5: Error at <i>PARITY</i> 6: Error at <i>STOPBITS</i> 7: Error at <i>FLOWCONTROL</i>
809xh	Error in FC/SFC parameter value x, where x: 1: Error at <i>PROTOCOL</i> 3: Error at <i>BAUDRATE</i> 4: Error at <i>CHARLENGTH</i> 5: Error at <i>PARITY</i> 6: Error at <i>STOPBITS</i> 7: Error at <i>FLOWCONTROL</i> (parameter is missing)
8092h	Access error in parameter DB (DB too short)
828xh	Error in parameter x of DB parameter, where x: 1: Error 1. parameter 2: Error 2. parameter ...

4.1.3 FC/SFC 217 - SER_SND - Send to PtP

Description

This block sends data via the serial interface. The repeated call of the FC/SFC 217 SER_SND delivers a return value for 3964R, USS and Modbus via RETVAL that contains, among other things, recent information about the acknowledgement of the partner station. The protocols USS and Modbus require to evaluate the receipt telegram by calling the FC/SFC 218 SER_RCV after SER_SND.

Parameters

Parameter	Declaration	Data type	Description
DATAPTR	IN	ANY	Pointer to Data Buffer for sending data
DATALEN	OUT	WORD	Length of data sent
RETVAL	OUT	WORD	Return value (0 = OK)

DATAPTR

Here you define a range of the type Pointer for the send buffer where the data to be sent are stored. You have to set type, start and length.

Example:

- Data is stored in DB5 starting at 0.0 with a length of 124byte.
- DataPtr:=P#DB5.DBX0.0 BYTE 124

DATALEN

- Word where the number of the sent Bytes is stored.
- At **ASCII** if data were sent by means of FC/SFC 217 faster to the serial interface than the interface sends, the length of data to send could differ from the DATALEN due to a buffer overflow. This should be considered by the user program.
- With **STX/ETX**, **3964R**, **Modbus** and **USS** always the length set in *DATAPTR* is stored or 0.

RETVAL FC/SFC 217 (Return values)

Return values of the block:

Error code	Description
0000h	Send data - ready
1000h	Nothing sent (data length 0)
20xxh	Protocol executed error free with xx bit pattern for diagnosis
7001h	Data is stored in internal buffer - active (busy)
7002h	Transfer - active
80xxh	Protocol executed with errors with xx bit pattern for diagnosis (no acknowledgement by partner)
90xxh	Protocol not executed with xx bit pattern for diagnosis (no acknowledgement by partner)
8x24h	Error in FC/SFC parameter x, where x: 1: Error in <i>DATAPTR</i> 2: Error in <i>DATALEN</i>
8122h	Error in parameter <i>DATAPTR</i> (e.g. DB too short)
807Fh	Internal error

Error code	Description
809Ah	interface not found e.g. interface is used by PROFIBUS
809Bh	interface not configured

Protocol specific RETVAL values

ASCII

Value	Description
9000h	Buffer overflow (no data send)
9002h	Data too short (0byte)

STX/ETX

Value	Description
9000h	Buffer overflow (no data send)
9001h	Data too long (>1024byte)
9002h	Data too short (0byte)
9004h	Character not allowed

3964R

Value	Description
2000h	Send ready without error
80FFh	NAK received - error in communication
80FEh	Data transfer without acknowledgement of partner or error at acknowledgement
9000h	Buffer overflow (no data send)
9001h	Data too long (>1024byte)
9002h	Data too short (0byte)

USS

Error code	Description
2000h	Send ready without error
8080h	Receive buffer overflow (no space for receipt)
8090h	Acknowledgement delay time exceeded
80F0h	Wrong checksum in respond
80FEh	Wrong start sign in respond
80FFh	Wrong slave address in respond
9000h	Buffer overflow (no data send)
9001h	Data too long (>1024byte)
9002h	Data too short (<2byte)

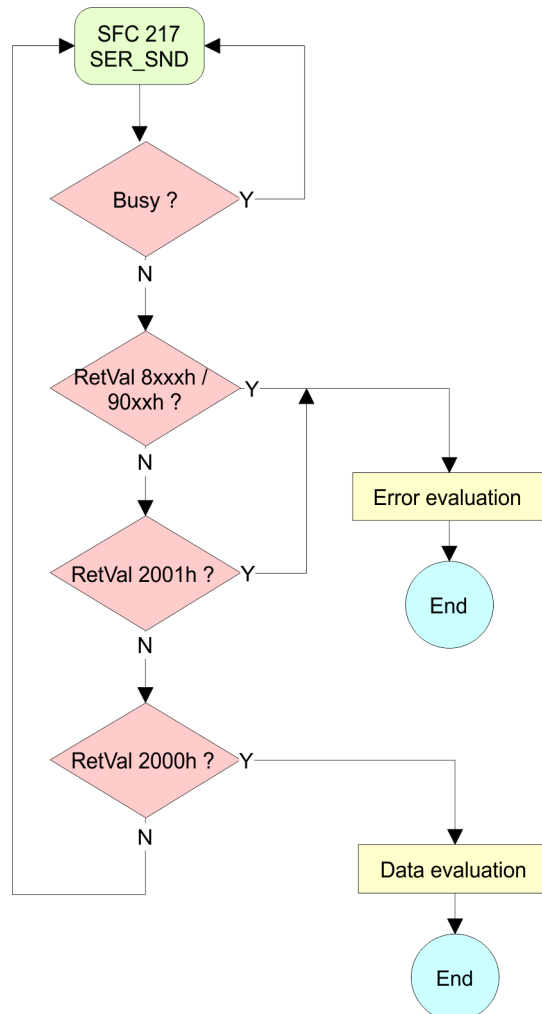
Modbus RTU/ASCII Master

Error code	Description
2000h	Send ready (positive slave respond)
2001h	Send ready (negative slave respond)
8080h	Receive buffer overflow (no space for receipt)
8090h	Acknowledgement delay time exceeded
80F0h	Wrong checksum in respond
80FDh	Length of respond too long
80FEh	Wrong function code in respond
80FFh	Wrong slave address in respond
9000h	Buffer overflow (no data send)
9001h	Data too long (>1024byte)
9002h	Data too short (<2byte)

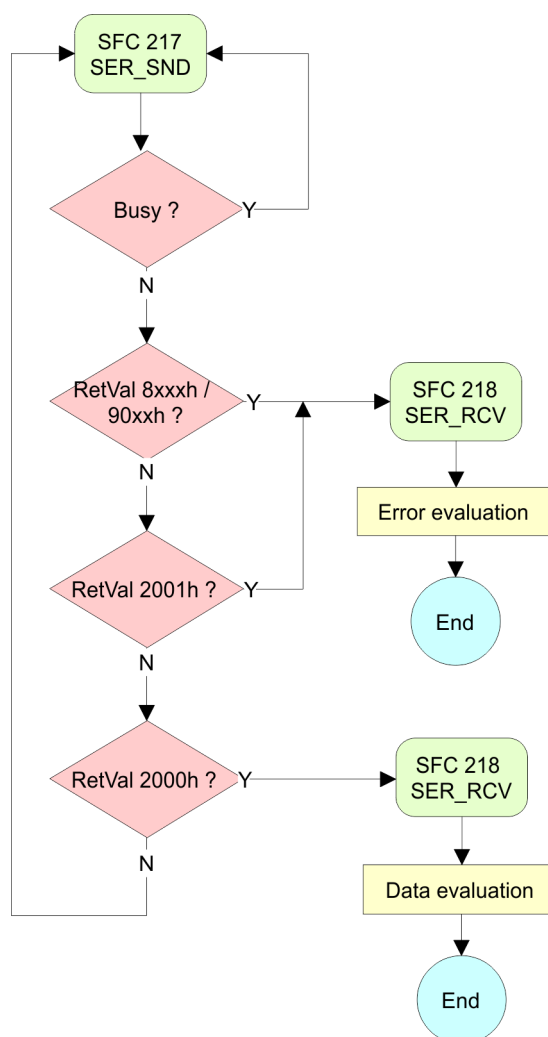
Principles of programming

The following text shortly illustrates the structure of programming a send command for the different protocols.

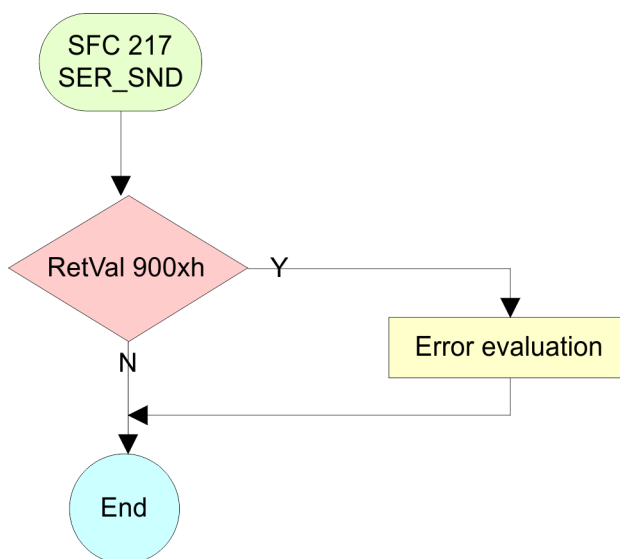
3964R



USS / Modbus



ASCII / STX/ETX



4.1.4 FC/SFC 218 - SER_RCV - Receive from PtP

Description

This block receives data via the serial interface. Using the FC/SFC 218 SER_RCV after SER_SND with the protocols USS and Modbus the acknowledgement telegram can be read.

Parameters

Parameter	Declaration	Data type	Description
DATAPTR	IN	ANY	Pointer to Data Buffer for received data
DATALEN	OUT	WORD	Length of received data
ERROR	OUT	WORD	Error Number
RETVAL	OUT	WORD	Return value (0 = OK)

DATAPTR

Here you set a range of the type Pointer for the receive buffer where the reception data is stored. You have to set type, start and length.

Example:

- Data is stored in DB5 starting at 0.0 with a length of 124byte.
- DataPtr:=P#DB5.DBX0.0 BYTE 124

DATALEN

- Word where the number of received Bytes is stored.
- At **STX/ETX** and **3964R**, the length of the received user data or 0 is entered.
- At **ASCII**, the number of read characters is entered. This value may be different from the read telegram length.

ERROR

This word gets an entry in case of an error. The following error messages may be created depending on the protocol:

ASCII

Bit	Error	Description
0	overflow	Overflow, a sign couldn't be read fast enough from the interface
1	framing error	Error that shows that a defined bit frame is not coincident, exceeds the allowed length or contains an additional bit sequence (Stop bit error)
2	parity	Parity error
3	overflow	Buffer is full

STX/ETX

Bit	Error	Description
0	overflow	The received telegram exceeds the size of the receive buffer.
1	char	A sign outside the range 20h ... 7Fh has been received.
3	overflow	Buffer is full.

3964R / Modbus RTU/ASCII Master

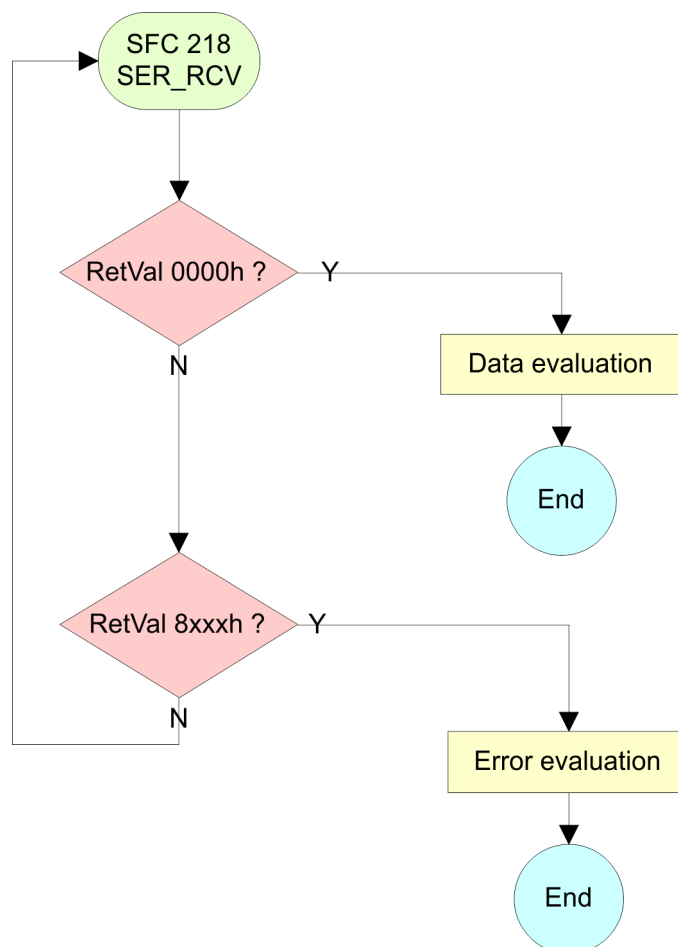
Bit	Error	Description
0	overflow	The received telegram exceeds the size of the receive buffer.

**RETVAL FC/SFC 218
(Return value)**

Error code	Description
0000h	no error
1000h	Receive buffer too small (data loss)
8x24h	Error at FC/SFC-Parameter x, with x: 1: Error at <i>DATAPTR</i> 2: Error at <i>DATALEN</i> 3: Error at <i>ERROR</i>
8122h	Error in parameter <i>DATAPTR</i> (e.g. DB too short)
809Ah	Serial interface not found res. interface is used by PROFIBUS
809Bh	Serial interface not configured

Principles of programming

The following picture shows the basic structure for programming a receive command. This structure can be used for all protocols.



4.1.5 FB 1 - RECEIVE_ASCII - Receiving with defined length from PtP

Description

This FB collects the data, which are received via the internal serial interface in PtP operation and copies them into the telegram buffer specified by *EMPF_PUFFER*. If the entire telegram was received *EMPF_FERTIG* is set and the FB is left. The reading of the data may require several FB calls. The next telegram is only be read, if the bit *EMPF_FERTIG* was reset by the user. With this FB only telegrams with fix length can be received.

Parameter

Parameter	Declaration	Data type	Description
EMPF_PUFFER	IN	ANY	Pointer to DB in which the received telegram is transmitted.
ER_BYTE	OUT	WORD	Error code
EMPF_FERTIG	IN_OUT	BOOL	Status

EMPF_PUFFER

Specify here an area of type pointer, in which the received data are to be copied. Specify type, start and length.

Example:

- Data are to be stored in DB5 starting from 0.0 with length 124byte
 - DataPtr:=P#DB5.DBX0.0 BYTE 124

ER_BYTE

This word gets an entry in case of error.

Error code	Description
0003h	DB with telegram buffer does not exist.
0004h	DB with telegram buffer is too short.
7000h	Receive buffer is too small - data have been deleted!
8000h	Pointer setting in <i>EMPF_PUFFER</i> is faulty or does not exist.
9001h	DB setting in <i>EMPF_PUFFER</i> is faulty or does not exist.
9002h	Length setting in <i>EMPF_PUFFER</i> is faulty or does not exist.

4.1.6 FB 7 - P_RCV_RK - Receive from CP 341

Description

The FB 7 P_RCV_RK transfers data from the CP to a data area of the CPU specified by the parameter *DB_NO*, *DBB_NO* and *LEN*. For data transfer the FB is to be called either cyclically or statically by a timer-driven program. Please note that this block calls the FC or SFC 192 CP_S_R 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
EN_R	IN	BOOL	Enables data read
R	IN	BOOL	Aborts request - current request is aborted and receiving is blocked.

Parameter	Declaration	Data type	Description
LADDR	IN	INT	Logical basic address of the CP - corresponds to the address of the hardware configuration of the CP.
DB_NO	IN	INT	Data block number - number of the receive DB, zero is not allowed.
DBB_NO	IN	INT	Data byte number - received data as of data byte $0 \leq DBB_NO \leq 8190$
L_...	OUT	-	These parameters are not relevant for ASCII and 3964(R). But they may be used by loadable protocols.
NDR*	OUT	BOOL	Request complete without errors, data received Parameter <i>STATUS</i> = 00h
ERROR*	OUT	BOOL	Request complete with error Parameter <i>STATUS</i> contains error details
LEN*	OUT	BOOL	Length of the received telegram in byte $1 \leq LEN \leq 1024$
STATUS*	OUT	WORD	Specification of the error on <i>ERROR</i> = 1

*) Parameter is available until the next call of the FB.

Release and cancel a request

- With the signal state "1" at parameter *EN_R*, the software checks whether data can be read by the CP. A data transmission operation can run over several program cycles, depending on the amount of data involved.
- An active transmission can be aborted with signal state "0" at the *EN_R* parameter. The aborted receive request is terminated with an error message (*STATUS*).
- Receiving is deactivated as long as the *EN_R* parameter shows the signal state "0". A running request may be canceled with *R* = "1" then the FB is reset to the basic state. Receiving is deactivated as long as the *R* parameter shows the signal state "1".

Mechanism for startup synchronization

The FB 7 has a mechanism for startup-synchronization between CPU and CP, which is automatically executed at the first call of the FB. Before the CP can process an activated request after the CPU has changed from STOP to RUN mode, the CP CPU start-up mechanism must be completed. Any requests initiated in the meantime are transmitted once the start-up coordination with the CP is finished.



A minimum pulse time is necessary for a signal change to be identified. Significant time periods are the CPU cycle time, the updating time on the CP and the response time of the communication partner.

Error indication

- The *NDR* output shows "request completed without errors/data accepted". If there was an *ERROR*, the corresponding event number is displayed in the *STATUS*. If no error occurs the value of *STATUS* is "0".
- *NDR* and *ERROR/STATUS* are also output in response to a *RESET* of the FB. In the event of an error, the binary result BR is reset. If the block is terminated without errors, the binary result has the status "1".
- Please regard the parameter *NDR*, *ERROR* and *STATUS* are only available at one block call. For further evaluation these should be copied to a free data area.

Addressing

With *LADDR* the address of the corresponding CP is specified. This is the address, which was specified by the hardware configuration of the CP. Please regard that the base address for input and output of the CP are identical.

Data area The FB 7 - P_RCV_RK deals with an Instanz-DB I_RCV_RK. This has a length from 60byte. The DB no. is transmitted with the call. It is not allowed to access the data of an instance DB.

4.1.7 FB 8 - P_SND_RK - Send to CP 341

Description The FB 8 - P_SND_RK transfers a data block of a DB to the CP, specified by the parameters *DB_NO*, *DBB_NO* and *LEN*. For data transfer the FB is to be called either cyclically or statically by a timer-driven program. Please note that this block calls the FC or SFC 192 CP_S_R 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
SF	IN	CHAR	S = Send, F = Fetch. At ASCII and 3964R the default value "S" for Send may be used
REQ	IN	BOOL	Initiates request with positive edge
R	IN	BOOL	Aborts request - current request is aborted and sending is blocked.
LADDR	IN	INT	Logical basic address of the CP - corresponds to the address of the hardware configuration of the CP.
DB_NO	IN	INT	Data block number - number of the send DB, zero is not allowed.
DBB_NO	IN	INT	Data byte number - transmitted data as of data byte $0 \leq DBB_NO \leq 8190$
LEN	IN	INT	Length of message frame to be sent in byte $1 \leq LEN \leq 1024$
R_...	IN	-	These parameters are not relevant for ASCII and 3964(R). But they may be used by loadable protocols. With Modbus enter here "X".
DONE*	OUT	BOOL	Request complete without errors, data sent Parameter <i>STATUS</i> = 00h
ERROR*	OUT	BOOL	Request complete with error Parameter <i>STATUS</i> contains error details
STATUS*	OUT	WORD	Specification of the error on <i>ERROR</i> = 1

*) Parameter is available until the next call of the FB.

Release and cancel a request

- The data transmission is initiated by a positive edge at the *REQ* input of FB 8 - P_SND_RK. A data transmission operation can run over several program cycles, depending on the amount of data involved.
- A running request may be canceled at any time with *R* = "1" then the FB is reset to the basic state. Please regard that data, which the CP still has received from the CPU, were sent to the communication partner.
- If the *R* input is statically showing the signal state "1", this means that sending is deactivated.

Mechanism for startup synchronization

The FB 8 has a mechanism for startup-synchronization between CPU and CP, which is automatically executed at the first call of the FB. Before the CP can process an activated request after the CPU has changed from STOP to RUN mode, the CP CPU start-up mechanism must be completed. Any requests initiated in the meantime are transmitted once the start-up coordination with the CP is finished.



A minimum pulse time is necessary for a signal change to be identified. Significant time periods are the CPU cycle time, the updating time on the CP and the response time of the communication partner.

Error indication

- The *DONE* output shows "request completed without errors". If there was an *ERROR*, the corresponding event number is displayed in the *STATUS*. If no error occurs the value of *STATUS* is "0".
- *DONE* and *ERROR/STATUS* are also output in response to a *RESET* of the FB. In the event of an error, the binary result BR is reset. If the block is terminated without errors, the binary result has the status "1".
- Please regard the parameter *DONE*, *ERROR* and *STATUS* are only available at one block call. For further evaluation these should be copied to a free data area.

Addressing

With *LADDR* the address of the corresponding CP is specified. This is the address, which was specified by the hardware configuration of the CP. Please regard that the base address for input and output of the CP are identical.

Data area

The FB 8 - P_SND_RK deals with an Instanz-DB I_SND_RK. This has a length from 62byte. The DB no. is transmitted with the call. It is not allowed to access the data of an instance DB.

4.2 CP040

4.2.1 FB 60 - SEND - Send to System SLIO CP 040

Description

This FB serves for the data output from the CPU to the System SLIO CP 040. Here you define the send range via the identifiers *DB_NO*, *DBB_NO* and *LEN*. A rising edge at *REQ* a transmission is initiated and the data is sent.

Parameters

Name	Declaration	Type	Description
REQ	IN	BOOL	Release SEND with positive edge.
R	IN	BOOL	Release synchronous reset.
LADDR	IN	INT	Logical base address of the CP.
DB_NO	IN	INT	Number of DB containing data to send.
DBB_NO	IN	INT	Data byte number - send data starting from data byte.
LEN	IN	INT	Length of telegram in byte, to be sent.
IO_SIZE	IN	WORD	Configured IO size of the module.
DONE*	OUT	BOOL	Send order finished without errors.
ERROR*	OUT	BOOL	Send order finished with errors. Parameter <i>STATUS</i> contains the error information.
STATUS*	OUT	WORD	Specification of the error with <i>ERROR</i> = 1.

CP040 > FB 60 - SEND - Send to System SLIO CP 040

Name	Declaration	Type	Description
CONTROL	IN_OUT	BYTE	Divided byte with RECEIVE handling block: SEND (bit 0 ... 3), RECEIVE (bit 4 ... 7).
*) Parameter is available until the FB is called.			

- REQ** Request - Send release:
- With a positive edge on input *REQ* the transfer of the data is triggered.
 - Depending on the number of data, a data transfer can run over several program cycles.
- R** Synchronous reset:
- For the initialization SEND is once to be called in the start-up OB with every parameter and set *R*.
 - At any time a current order may be cancelled and the FB may be set to initial state with signal state "1" of *R*. Please regard that the data, which the CP has already received, are still sent to the communication partner.
 - The Send function is deactivated as long as *R* is statically set to "1".
- LADDR** Peripheral address:
- With *LADDR* the address of the corresponding CP may be determined. This is the address, which you have assigned via the hardware configuration for the CP.
- DB_NO** Data block number:
- Number of the data block, which contains the data to send.
 - Zero is not permitted.
- DBB_NO** Data byte number:
- Number of data byte in the data block, starting from which the transmit data are stored.
- LEN** Length:
- Length of the user data to be sent.
 - It is: $1 \leq LEN \leq 1024$.
- IO_SIZE** Size I/O area:
- Enter the size of the I/O area. Depending on the host system the CP occupies for input and output the following bytes in the I/O areas:
 - PROFIBUS: 8byte, 20byte or 60byte selectable
 - PROFINET: 20byte or 60byte selectable
 - CANopen: 8byte
 - EtherCAT: 60byte
 - DeviceNET: 60byte
 - ModbusTCP: 60byte

DONE**DONE:**

- is set at order ready without errors and *STATUS* = 0000h.

ERROR**ERROR:**

- is set at order ready with error. Here *STATUS* contains the corresponding error message.

STATUS

If there is no error, *STATUS* = 0000h or 8181h. With an error here the corresponding error code may be found. As long as *ERROR* is set, the value of *STATUS* is available. The following status messages are possible:

STATUS	Description
0000h	No error found
0202h	Handling block and CP are not synchronous (Remedy: Start synchronous reset)
0301h	DB not valid
070Ah	Transfer failed, there is no response of the partner or the telegram was negative acknowledged.
0816h	Parameter <i>LEN</i> is not valid (<i>LEN</i> = 0 or <i>LEN</i> > 1024)
8181h	Order running (Status and no error message)

CONTROL

The handling blocks SEND and RECEIVE use the common parameter *CONTROL* for the handshake. Assign to this parameter a common flag byte.

Error indication

- The *DONE* output shows "order ready without error". If there was an *ERROR*, the corresponding event number is displayed in the *STATUS*. If no error occurs the value of *STATUS* is "0".
- *DONE*, *ERROR* and *STATUS* are also output in response to a reset of the FB. In the event of an error, the binary result *BR* is reset. If the block is terminated without errors, the binary result has the status "1".
- Please regard the parameter *DONE*, *ERROR* and *STATUS* are only available at one block call. For further evaluation these should be copied to a free data area.

4.2.2 FB 61 - RECEIVE - Receive from System SLIO CP 040**Description**

This FB serves for the data reception from the System SLIO CP 040. Here you set the reception range via the identifiers *DB_NO* and *DBB_NO*. The length of the telegram is stored in *LEN*.

Parameters

Parameter	Declaration	Data type	Description
EN_R	IN	BOOL	Release RECEIVE data.
R	IN	BOOL	Release synchronous reset.

CP040 > FB 61 - RECEIVE - Receive from System SLIO CP 040

Parameter	Declaration	Data type	Description
LADDR	IN	INT	Logical base address of the CP.
DB_NO	IN	INT	Number of DB containing received data.
DBB_NO	IN	INT	Data byte number - receive data starting from data byte.
IO_SIZE	IN	WORD	Configured IO size of the module.
LEN	OUT	INT	Length of received telegram in byte
NDR*	OUT	BOOL	Receive order finished without errors.
ERROR*	OUT	BOOL	Receive order finished with errors. Parameter <i>STATUS</i> contains the error information.
STATUS*	OUT	WORD	Specification of the error with <i>ERROR</i> = 1.
CONTROL	IN_OUT	BYTE	Divided byte with RECEIVE handling block: SEND (bit 0 ... 3), RECEIVE (bit 4 ... 7).

*) Parameter is available until the FB is called.

- EN_R** Enable Receive - Release to read:
- With signal status "1" at *EN_R* the examination, whether data from the CP are read, is released. Depending upon the number of data, a data transfer can run over several program cycles.
 - At any time a current order may be cancelled with signal state "0" of *EN_R*. Here the cancelled receipt order is finished with an error message (*STATUS*).
 - The Receive function is deactivated as long as *EN_R* is statically set to "0".
- R** Synchronous reset:
- For the initialization RECEIVE is once to be called in the start-up OB with every parameter and set *R*.
 - At any time a current order may be cancelled and the FB may be set to initial state with signal state "1" of *R*.
 - The Receive function is deactivated as long as *R* is statically set to "1".
- LADDR** Peripheral address:
- With *LADDR* the address of the corresponding CP may be determined. This is the address, which you have assigned via the hardware configuration for the CP.
- DB_NO** Data block number:
- Number of the data block, which contains the data are read.
 - Zero is not permitted.
- DBB_NO** Data byte number:
- Number of data byte in the data block, starting from which the received data are stored.

IO_SIZE

Size I/O area:

- Enter the size of the I/O area. Depending on the host system the CP occupies for input and output the following bytes in the I/O areas:
 - PROFIBUS: 8byte, 20byte or 60byte selectable
 - PROFINET: 20byte or 60byte selectable
 - CANopen: 8byte
 - EtherCAT: 60byte
 - DeviceNET: 60byte
 - ModbusTCP: 60byte

LEN

Length:

- Length of the user data to be sent.
- It is: $1 \leq LEN \leq 1024$.

NDR

- New received data are ready for the CPU in the CP.

ERROR

ERROR:

- is set at order ready with error. Here *STATUS* contains the corresponding error message.

STATUS

If there is no error, *STATUS* = 0000h or 8181h. With an error here the corresponding error code may be found. As long as *ERROR* is set, the value of *STATUS* is available. The following status messages are possible:

STATUS	Description
0000h	No error found
0202h	Handling block and CP are not synchronous (Remedy: Start synchronous reset)
0301h	DB not valid
070Ah	Transfer failed, there is no response of the partner or the telegram was negative acknowledged.
0816h	Parameter <i>LEN</i> is not valid ($LEN = 0$ or $LEN > 1024$)
080Ah	A free receive buffer is not available
080Ch	Wrong character received (Character frame or parity error)
8181h	Order running (Status and no error message)

CONTROL

- The handling blocks SEND and RECEIVE use the common parameter *CONTROL* for the handshake.
- Assign to this parameter a common flag byte.

Error indication

- The *NDR* output shows "order ready without error / data kept". If there was an *ERROR*, the corresponding event number is displayed in the *STATUS*. If no error occurs the value of *STATUS* is "0".
- *NDR*, *ERROR* and *STATUS* are also output in response to a reset of the FB. In the event of an error, the binary result BR is reset. If the block is terminated without errors, the binary result has the status "1".
- Please regard the parameter *NDR*, *ERROR* and *STATUS* are only available at one block call. For further evaluation these should be copied to a free data area.

4.2.3 FB 65 - CP040_COM - Communication SLIO CP 040**Description**

The FB 65 serves the data in-/output from the System SLIO CPU to the CP 040. Here you define the send/receive range via the identifiers *DB_NO_SEND* and *DB_NO_RECV*. A rising edge at *REQ_SEND* a transmission is initiated and the data are sent. Via *EN_RECV* the received data are enabled.

Parameters

Name	Declaration	Type	Description
REQ_SEND	IN	BOOL	Release SEND with positive edge.
EN_RECV	IN	BOOL	Enable receive data.
RESET	IN	BOOL	Release synchronous reset.
ADDR_IN	IN	INT	Logical input base address of the CP from the Hardware configuration.
ADDR_OUT	IN	INT	Logical output base address of the CP from the Hardware configuration.
DB_NO_SEND	IN	INT	Number of DB containing data to send. Zero is not permitted.
DBB_NO_SEND	IN	INT	Data byte number - send data starting from data byte.
LEN_SEND	IN	INT	Length of telegram in byte, to be sent. $1 \leq LEN_SEND \leq 1024$
DB_NO_RECV	IN	INT	Number of DB containing data to receive. Zero is not permitted.
DBB_NO_RECV	IN	INT	Data byte number - receive data starting from data byte.
IO_SIZE	IN	WORD	Configured IO size of the module.
DONE_SEND*	OUT	BOOL	Send order finished without errors. Data sent: Parameter <i>STATUS_SEND</i> = 0000h.
ERROR_SEND*	OUT	BOOL	Send order finished with errors. Here Parameter <i>STATUS_SEND</i> contains the corresponding error message.
STATUS_SEND*	OUT	WORD	Specification of the error with <i>ERROR_SEND</i> = 1
LEN_RCV	OUT	INT	Length of received telegram in byte $1 \leq LEN_RCV \leq 1024$

Name	Declaration	Type	Description
NDR_RCV*	OUT	BOOL	Receive order finished without errors. Data sent: Parameter <i>STATUS_RCV</i> = 0000h. The Parameter is available until a cycle.
ERROR_RCV*	OUT	BOOL	Receive order finished with errors. Parameter <i>STATUS_RCV</i> contains the error information.
STATUS_RCV*	OUT	WORD	Specification of the error with <i>ERROR_RCV</i> = 1

*) Parameter is available until the FB is called.

REQ_SEND

Request - Send release:

- With a positive edge on input *REQ_SEND* the transfer of the data is triggered. Depending on the number of data, a data transfer can run over several program cycles.

EN_RECV

Enable receive data.

RESET

Synchron Reset:

- For the initialization the FB 65 is once to be called in the start-up OB with every parameter and set *RESET*.
- At any time a current order may be cancelled and the FB may be set to initial state with signal state "1" of *RESET*.
- Please regard that the data, which the CP has already received, are still sent to the communication partner. The Send function is deactivated as long as *RESET* is statically set to "1".

ADDR_IN

Peripheral input address:

- With *ADDR_IN* the input address of the corresponding CP may be determined. This is the address, which you have assigned via the hardware configuration for the CP.

ADDR_OUT

Peripheral output address:

- With *ADDR_OUT* the output address of the corresponding CP may be determined. This is the address, which you have assigned via the hardware configuration for the CP.

DB_NO_SEND

Number of the DB SEND:

- Number of the data block, which contains the data to send.
- Zero is not permitted.

DBB_NO_SEND

Data byte number SEND:

- Number of data byte in the data block, starting from which the transmit data are stored.

- LEN_SEND** Length SEND:
- Length of the user data to be sent.
 - It is: $1 \leq LEN_SEND \leq 1024$.
- DB_NO_RECV** Number of the DB RECV:
- Number of the data block, which contains the receive data.
 - Zero is not permitted.
- DBB_NO_RECV** Data byte number RECV:
- Number of data byte in the data block, starting from which the received data are stored.
- IO_SIZE** Size I/O area:
- Enter the size of the I/O area. Depending on the host system the CP occupies for input and output the following bytes in the I/O areas:
 - SLIO CPU: 8byte, 20byte or 60byte selectable
 - PROFIBUS: 8byte, 20byte or 60byte selectable
 - PROFINET: 20byte or 60byte selectable
 - CANopen: 8byte
 - EtherCAT: 60byte
 - DeviceNET: 60byte
 - ModbusTCP: 60byte
- DONE_SEND** *DONE_SEND* is set at order ready without errors and *STATUS_SEND* = 0000h.
- ERROR_SEND** *ERROR_SEND* is set at order ready with error. Here *STATUS_SEND* contains the corresponding error message.
- STATUS_SEND** If there is no error, *STATUS_SEND* = 0000h or 8181h. With an error here the corresponding error code may be found. As long as *ERROR_SEND* is set, the value of *STATUS_SEND* is available. Following status messages are possible:

STATUS	Description
0000h	No error found
0202h	<i>IO_SIZE</i> = 0 or <i>IO_SIZE</i> > 60
0301h	DB not valid
070Ah	Transfer failed, there is no response of the partner or the telegram was negative acknowledged
0517h	Parameter <i>LEN_SEND</i> is not valid (<i>LEN_SEND</i> = 0 or <i>LEN_SEND</i> > 1024)
8181h	Order running (Status and no error message)

LEN_RCV

Length Receive:

- Length of the received telegram in byte.
- $1 \leq LEN_RCV \leq 1024$

NDR_RCV

New data ready:

- New received data are ready in receive DB. Signal stays for one cycle.
- Received data without error: Parameter *STATUS_RCV* = 0000h.

ERROR_RCV

ERROR_RCV is set at order ready with error. Here *STATUS_RCV* contains the corresponding error message.

STATUS_RCV

If there is no error, *STATUS_RCV* = 0000h or 8181h. With an error here the corresponding error code may be found. As long as *ERROR_RCV* is set, the value of *STATUS_RCV* is available. The following status messages are possible:

STATUS	Description
0000h	No error found
0202h	<i>IO_SIZE</i> = 0 or <i>IO_SIZE</i> > 60
0301h	DB not valid
070Ah	Transfer failed, there is no response of the partner or the telegram was negative acknowledged
0816h	Parameter <i>LEN_RCV</i> is not valid (<i>LEN_RCV</i> = 0 or <i>LEN_RCV</i> > 1024)
080Ah	A free receive buffer is not available
080Ch	Wrong character received (Character frame or parity error)
8181h	Order running (Status and no error message)

Error indication

- The *DONE_SEND* output shows "send order finished without error / data kept".
- The *NDR_RCV* output shows "receive order finished without error".
- If there was *ERROR_SEND* or *ERROR_RCV*, the corresponding event number is displayed in the *STATUS_SEND*, *STATUS_RCV*. If no error occurs the value of *STATUS_SEND* and *STATUS_RCV* is 0000h.
- *DONE_SEND*, *NDR_RCV*, *ERROR_SEND*, *ERROR_RCV* and *STATUS_SEND*, *STATUS_RCV* are also output in response to a reset of the FB. In the event of an error, the binary result BR is reset. If the block is terminated without errors, the binary result has the status "1".
- Please regard the parameter *DONE_SEND*, *NDR_RCV*, *ERROR_SEND*, *ERROR_RCV* and *STATUS_SEND*, *STATUS_RCV* are only available at one block call. For further evaluation these should be copied to a free data area.

4.3 CP240

4.3.1 FC 0 - SEND - Send to CP 240

Description

This FC serves the data output from the CPU to the CP 240. Here you define the send range via the identifiers *_DB*, *ABD* and *ANZ*. Via the bit *FRG* the send initialization is set and the data is send. After the data transfer the handling block sets the bit *FRG* back again.

Parameters

Name	Declaration	Type	Comment
ADR	IN	INT	Logical Address
_DB	IN	BLOCK_DB	DB No. of DB containing data to send
ABD	IN	WORD	Number of 1. data word to send
ANZ	IN	WORD	No of bytes to send
FRG	IN_OUT	BOOL	Start bit of the function
GESE	IN_OUT	WORD	internal use
ANZ_INT	IN_OUT	WORD	internal use
ENDE_KOMM	IN_OUT	BOOL	internal use
LETZTER_BLOCK	IN_OUT	BOOL	internal use
SENDEN_LAEUFT	IN_OUT	BOOL	Status of function
FEHLER_KOM	IN_OUT	BOOL	internal use
PAFE	OUT	BYTE	Parameterization error (0 = OK)

ADR Periphery address with which you may call the CP 240. Via the hardware configuration you may set the periphery address.

_DB Number of the data block, which contains the data to send.

ABD Word variable that contains the number of the data word from where on the characters for output are stored.

ANZ Number of the bytes that are to be transferred.

FRG enable send At *FRG* = "1" the data defined via *_DB*, *ADB* and *ANZ* are transferred once to the CP addresses by *ADR*. After the transmission the *FRG* is set back again. When *FRG* = "0" at call of the block, it is left immediately!

PAFE At proper function, all bits of this bit memory byte are "0". At errors an error code is entered. The error setting is self-acknowledging, i.e. after elimination of the error cause, the byte is set back to "0" again. The following errors may occur:

- 1 = Data block not present
- 2 = Data block too short
- 3 = Data block number outside valid range

**GESE, ANZ_INT
ENDE_KOM
LETZTER_BLOCK
SENDEN_LAEUFT
FEHLER_KOM**

These parameters are internally used. They serve the information exchange between the handling blocks. For the deployment of the SYNCHRON_RESET (FC9) the control bits ENDE_KOM, LETZTER_BLOCK, SENDEN_LAEUFT and FEHLER_KOM must always be stored in a bit memory byte.

4.3.2 FC 1 - RECEIVE - Receive from CP 240

Description

This FC serves the data reception of the CP 240. Here you set the reception range via the identifiers *_DB* and *ABD*. When the output *EMFR* is set, a new telegram has been read completely. The length of the telegram is stored in *ANZ*. After the evaluation of the telegram this bit has to be set back by the user, otherwise no further telegram may be taken over by the CPU.

Parameters

Name	Declaration	Type	Comment
ADR	IN	INT	Logical Address
_DB	IN	BLOCK_DB	DB No. of DB containing received data
ABD	IN	WORD	No. of 1st data word received
ANZ	OUT	WORD	No of bytes received
EMFR	OUT	BOOL	1=data received, reset by user
GEEM	IN_OUT	WORD	internal use
ANZ_INT	IN_OUT	WORD	internal use
EMPF_LAEUFT	IN_OUT	BOOL	Status of function
LETZTER_BLOCK	IN_OUT	BOOL	internal use
FEHLER_EMPF	IN_OUT	BOOL	internal use
PAFE	OUT	BYTE	Parameterization error (0 = OK)
OFFSET	IN_OUT	WORD	internal use

ADR Periphery address for calling the CP 240. You define the periphery address via the hardware configuration.

_DB Number of the data block, which contains the data.

ABD Word variable that contains the number of the data word from where on the received characters are stored.

ANZ Word variable that contains the amount of received bytes.

EMFR By setting of *EMFR* the handling block shows that data has been received. Not until setting back *EMFR* in the user application new data can be received.

PAFE At proper function, all bits of this bit memory byte are "0". At errors an error code is entered. The error setting is self-acknowledging, i.e. after elimination of the error cause, the byte is set back to "0" again. The following errors may occur:

- 1 = Data block not present
- 2 = Data block too short
- 3 = Data block number outside valid range

**GEEM, ANZ_INT
LETZTER_BLOCK
EMPF_LAEUFT
FEHLER_EMPF OFFSET** These parameters are internally used. They serve the information exchange between the handling blocks. For the deployment of the SYNCHRON_RESET (FC9) the control bits LETZTER_BLOCK, EMPF_LAEUFT and FEHLER_EMPF must always be stored in a bit memory byte.

4.3.3 FC 8 - STEUERBIT - Modem functionality CP 240

Description This block allows you the following access to the serial modem lines:

Read: DTR, RTS, DSR, RI, CTS, CD
Write: DTR, RTS

Parameters

Name	Declaration	Type	Comment
ADR	IN	INT	Logical Address
RTS	IN	BOOL	New state RTS
DTR	IN	BOOL	New state DTR
MASKE_RTS	IN	BOOL	<ul style="list-style-type: none"> ■ 0: do nothing ■ 1: set state RTS
MASKE_DTR	IN	BOOL	<ul style="list-style-type: none"> ■ 0: do nothing ■ 1: set state DTR
STATUS	OUT	BYTE	Status flags
DELTA_STATUS	OUT	BYTE	Status flags of change between 2 accesses
START	IN_OUT	BOOL	Start bit of the function
AUFTRAG_LAEU	IN_OUT	BOOL	Status of function
RET_VAL	OUT	WORD	Return value (0 = OK)



This block must not be called as long as a transmit command is running otherwise you risk a data loss.

ADR Periphery address with which you may call the CP 240. Via the hardware configuration you may set the periphery address.

- RTS, DTR** This parameter presets the status of RTS res. *DTR*, which you may activate via *MASK_RTS* res. *MASK_DTR*.
- MASK_RTS, MASK_DTR** With 1, the status of the according parameter is taken over when you set *START* to 1.
- STATUS, DELTA_STATUS** *STATUS* returns the actual status of the modem lines. *DELTA_STATUS* returns the state of the modem lines that have changed since the last access. The bytes have the following structure:

Bit no.	7	6	5	4	3	2	1	0
STATUS	x	x	RTS	DTR	CD	RI	DSR	CTS
DELTA_STATUS	x	x	x	x	CD	RI	DSR	CTS

- START** By setting of *START*, the state, which has been activated via the mask, is taken over.
- AUFTRAG_LAEU** As long as the function is executed, this bit remains set.
- RET_VAL** At this time, this parameter always returns 00h and is reserved for future error messages.

4.3.4 FC 9 - SYNCHRON_RESET - Synchronization CPU and CP 240

Description

The block must be called within the cyclic program section. This function is used to acknowledge the start-up ID of the CP 240 and thus the synchronization between CPU and CP. Furthermore it allows to set back the CP in case of a communication interruption to enable a synchronous start-up.



A communication with SEND and RECEIVE blocks is only possible when the parameter ANL of the SYNCHRON block has been set in the start-up OB before.

Parameters

Name	Declaration	Type	Comment
ADR	IN	INT	Logical address
TIMER_NR	IN	WORD	Timer number
ANL	IN_OUT	BOOL	CPU restart progressed
NULL	IN_OUT	BOOL	Internal use
RESET	IN_OUT	BOOL	Reset the CP
STEUERB_S	IN_OUT	BYTE	Internal use
STEUERB_R	IN_OUT	BYTE	Internal use

CP240 > FC 11 - ASCII_FRAGMENT - Receive fragmented from CP 240

ADR	Periphery address with which you may call the CP 240. Via the hardware configuration you may set the periphery address.
TIMER_NR	Number of the timer for the delay time.
ANL	With <i>ANL</i> = 1 the handling block is informed that a STOP/START res. NETZ-AUS/NETZ-EIN has been executed at the CPU and now a synchronization is required. After the synchronization, <i>ANL</i> is automatically set back.
NULL	Parameter is used internally.
RESET	<i>RESET</i> = 1 allows you to set back the CP out of your user application.
STEUERB_S	Here you have to set the bit memory byte where the control bits ENDE_KOM, LETZTER_BLOCK, SENDEN_LAEUFT and FEHLER_KOM for the SEND-FC are stored.
STEUERB_R	Here you have to set the bit memory byte where the control bits LETZTER_BLOCK, EMPF_LAEUFT and FEHLER_EMPF for the RECEIVE-FC are stored.

4.3.5 FC 11 - ASCII_FRAGMENT - Receive fragmented from CP 240

Description This FC serves the fragmented ASCII data reception. This allows you to handle on large telegrams in 12byte blocks to the CPU directly after the reception. Here the CP does not wait until the complete telegram has been received. The usage of the FC 11 presumes that you've parameterized "ASCII-fragmented" at the receiver. In the FC 11, you define the reception range via the identifiers *_DB* and *ABD*. When the output *EMFR* is set, a new telegram has been read completely. The length of the read telegram is stored in *ANZ*. After the evaluation of the telegram this bit has to be set back by the user, otherwise no further telegram may be taken over by the CPU.

Parameters

Name	Declaration	Type	Comment
ADR	IN	INT	Logical Address
_DB	IN	BLOCK_DB	DB No. of DB containing received data
ABD	IN	WORD	No. of 1st data word received
ANZ	OUT	WORD	No of bytes received
EMFR	IN_OUT	BOOL	Receipt confirmation
GEEM	IN_OUT	WORD	Internal use
ANZ_INT	IN_OUT	WORD	Internal use
EMPF_LAEUFT	IN_OUT	BOOL	Internal use
LETZTER_BLOCK	IN_OUT	BOOL	Internal use
FEHLER_EMPF	IN_OUT	BOOL	Internal use
PAFE	OUT	BYTE	Parameterization error (0 = OK)

ADR	Periphery address with which you may call the CP 240. Via the hardware configuration you may set the periphery address.
_DB	Number of the data block, which contains the data to receive.
ABD	Word variable that contains the number of the data word from where on the received characters are stored.
ANZ	Word variable that contains the amount of bytes that have been received.
EMFR	By setting of <i>EMFR</i> , the handling block announces that data has been received. Only by setting back <i>EMFR</i> in the user application new data can be received.
PAFE	<p>At proper function, all bits of this bit memory byte are "0". At errors an error code is entered. The error setting is self-acknowledging, i.e. after elimination of the error cause, the byte is set back to "0" again. The following errors may occur:</p> <ul style="list-style-type: none">■ 1 = Data block not present■ 2 = Data block too short■ 3 = Data block number outside valid range
GEEM, ANZ_INT LETZTER_BLOCK EMPF_LAEUFT FEHLER_EMPF	These parameters are internally used. They serve the information exchange between the handling blocks. For the deployment of the SYNCHRON_RESET (FC 9) the control bits LETZTER_BLOCK, EMPF_LAEUFT and FEHLER_EMPF must always be stored in a bit memory byte.