SPEED7 Library

OPL_SP7-LIB | SW90GS0MA V10.011 | Manual

HB00 | OPL_SP7-LIB | SW90GS0MA V10.011 | en | 24-02

Block library - Serial Communication

Table of contents

1 General

1.1 Copyright © YASKAWA Europe GmbH

All Rights Reserved

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

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

For permission to reproduce or distribute, please contact: YASKAWA Europe GmbH, European Headquarters, Philipp-Reis-Str. 6, 65795 Hattersheim, Germany

Tel.: +49 6196 569 300
Fax.: +49 6196 569 398
Email: info@yaskawa.eu
Internet: www.yaskawa.eu.com

EC conformity declaration

Hereby, YASKAWA Europe 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 representative of YASKAWA Europe GmbH.

Trademarks

SLIO, System 300S and SPEED7 are registered trademarks of YASKAWA Europe GmbH.

CAN is a registered trademark of CAN in Automation e. V. (CiA).

DeviceNet is a registered trademark of ODVA, Inc.

EtherCAT is a registered trademark of Beckhoff Automation GmbH.

Modbus is a registered trademark of Schneider Electric.

PROFINET and PROFIBUS are registered trademarks of PROFIBUS and PROFINET International (PI).

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

All Microsoft Windows, Office and Server products mentioned are registered trademarks of Microsoft Inc., USA.

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

General terms of use

Every effort has been made to ensure that the information contained in this document was complete and accurate at the time of publishing. We cannot guarantee that the information is free of errors, and we reserve the right to change the information at any time. There is no obligation to inform the customer about any changes. The customer is requested to actively keep his documents up to date. The customer is always responsible for the deployment of the products with the associated documentation, taking into account the applicable directives and standards.
This documentation describes all hardware and software units and functions known today. It is possible that units are described that do not exist at the customer. The exact scope of delivery is described in the respective purchase contract.

Document support   Contact your local representative of YASKAWA Europe GmbH if you have errors or questions regarding the content of this document. You can reach YASKAWA Europe GmbH via the following contact:

Email: Documentation.HER@yaskawa.eu

Technical support   Contact your local representative of YASKAWA Europe GmbH if you encounter problems or have questions regarding the product. If such a location is not available, you can reach the Yaskawa customer service via the following contact:

YASKAWA Europe GmbH,
European Headquarters, Philipp-Reis-Str. 6, 65795 Hattersheim, Germany
Tel.: +49 6196 569 500 (hotline)
Email: support@yaskawa.eu

1.2   About this manual

Objective and contents   The manual describes the block library *'Serial Communication'*:

- The manual is targeted at users who have a background in automation technology.
- 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

> ℹ️ *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

2.3   No optimized block access

> ⓘ *Please note that the blocks do not support optimized block access for use in S7-1500 CPUs from Siemens! When using instance and data blocks, the optimized block access must be deactivated!*

Set block access

1. ▸ Open in the Siemens TIA Portal *Project navigation* the *'Program blocks'*.

2. ▸ Select the block for which you want to change the block access and select *'Context menu → Properties'*.

   ➡ The "Properties" dialog of the block opens.

3. ▸ Select *'Attributes'*.

4. ▸ Deactivate the parameter *'Optimized block access'*.

5. ▸ Confirm with [OK].

More information can be found in the manual of the Siemens TIA Portal.

2.4   Declaration types

Please note that the spellings of the declaration types in Siemens STEP7 and TIA Portal differ. This documentation uses the notation for Siemens STEP7. A comparison of the spellings can be found in the following table.

Siemens TIA Portal	Siemens STEP7
Input	IN
Output	OUT
InOut	IN_OUT
Static	STAT
Temp	TEMP

3 Include library

Block library Serial Communication

The block library can be found for download in the *'Download Center'* of www.yaskawa.eu.com under *'Controls Library'* 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.

> *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
SerialCom_S7_V0005.zip	■ Block library for Siemens SIMATIC Manager. ■ For use in Yaskawa CPUs or S7-300 CPUs from Siemens.
SerialCom_TIA_V0006.zip	■ Block library for Siemens TIA Portal V14/V15. ■ For use in Yaskawa CPUs or S7-300 CPUs from Siemens.
SerialCom_TIA_1500_V0005.zip	■ Block library for Siemens TIA Portal V14/V15. ■ For use in S7-1500 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 blocks via your user application.

> *Are FCs used instead of SFCs, so they are supported by the System 300S Yaskawa 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.

   ➡
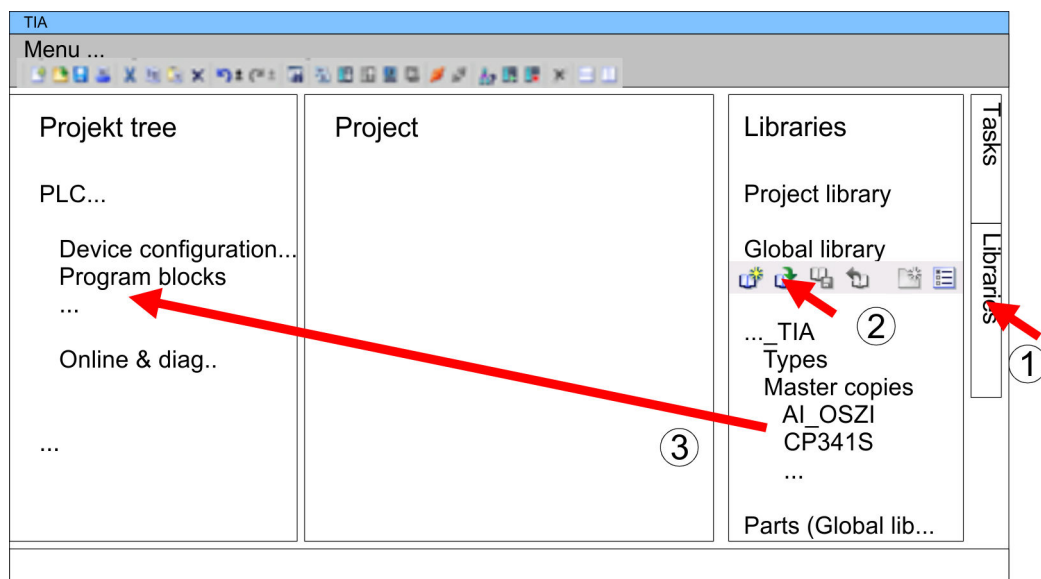
7. ▸ Copy the necessary blocks from the library into the "Program blocks" of the *Project tree* of your project. Now you have access to the blocks via your user application.

4   Block parameters

4.1   HW identifier - HW_ID

HW identifier

■ The parameter *HW_ID* to preset the *HW identifier* is only available in S7-1500 CPUs from Siemens.

■ When configuring a hardware component, a hardware identifier is automatically assigned as *HW identifier* for each object of the hardware configuration.

■ The *HW identifier* is for modules, ports, interfaces and I/O areas of bus systems.

■ The *HW identifier* is a decimal integer constant of data type HW_IO.

■ The *HW identifier* does not distinguish between input and output ranges.

■ You can use the *HW identifier* to address the corresponding hardware components.

Determine *HW identifier*

You can determine the *HW identifier* for the respective component using the following procedure:

1. ▸ Open in the *Project tree* the *'Device configuration'*.

2. ▸ Select the desired hardware component whose *HW identifier* you want to determine.

3. ▸ Click in the *Inspector* window at *'General'*.

   ➨ The *'HW identifier'* is shown. These can be used for the parameter *HW_ID* when the blocks are connected.

***HW identifier* and system constants**

You can also determine the *HW identifier* using the *'System constants'*. Via the *'System constants'* in the *Inspector* window all the HW identifiers of an object, which is selected in the device view, are listed with *Name* and *Type*. *Name* and *Type* are automatically generated when assigning the HW identifier. Here *Name* has a hierarchical structure with a maximum of 4 hierarchical levels, with each level separated by a "~". The name of the component of the corresponding hierarchy level can be changed at any time via the properties.

***HW identifier* in the user program**

■ When creating your user program, you can assign the corresponding hardware component from a list of all possible hardware components by double-clicking on the corresponding input or output parameter.

■ In the case of a hardware interrupt, you can use the start information to determine the *HW identifier* as the *'ID'* of the hardware component that triggers the interrupt.

4.2 RET_VAL and BUSY for asynchronously operating blocks

Output parameters RET_VAL and BUSY

For blocks that work asynchronously, the function execution extends over several block calls. Here, the output parameters RET_VAL and BUSY indicate the status of the job execution:

- First call with REQ = 1
  - With free system resources and correct input parameters, BUSY is set to 1 and RET_VAL returns W#16#7001.
  - If the system resources are occupied or the input parameters are faulty, BUSY is set to 0 and the corresponding error code is returned in RET_VAL.
- Interim call
  - BUSY remains set to 1 and RET_VAL returns W#16#7002. This signals that the job is still being processed.
- Last call
  - After error-free execution, BUSY is set to 0 and RET_VAL returns 0. Please note that some blocks report the number of transmitted data via RET_VAL. For more information, please refer to the corresponding block descriptions.
  - In the event of an error, BUSY is set to 0 and the RET_VAL returns the corresponding error code.

Input parameter REQ

The REQ input parameter is used exclusively to initiate a job:

- REQ = 1 in the corresponding block executes a job that is not yet active.
- REQ is not evaluated with each subsequent call of the block.

Correlation of the parameters

Call number	Call type	REQ	RET_VAL	BUSY
1	First call	1	W#16#7001	1
			Error code in the event of an error	0
2 ... n-1	Interim call	not relevant	W#16#7002	1
n	Last call	not relevant	W#16#0000[1]	0
			Error code in the event of an error	

1) For some blocks, the number of transmitted data - see corresponding block description.

4.3 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).
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   Serial Communication

5.1   Serial communication

5.1.1   SFC 207 - SER_CTRL - Modem functionality PtP

Description

> $\bigcirc$
> *Please note that this block is not supported by SPEED7 CPUs!*
>
> *The use in S7-1500 CPUs from Siemens is also not supported!*

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
FLOWCONTROL=0:	DTR, RTS, DSR, RI, CTS, CD	DTR, RTS
FLOWCONTROL>0:	DTR, RTS, DSR, RI, CTS, CD	not possible

Parameters

Name	Declaration	Type	Description
WRITE	IN	BYTE	▪ Bit 0: New state DTR ▪ Bit 1: New state RTS
MASKWRITE	IN	BYTE	▪ 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: ■ 1: Error at *WRITE* ■ 2: Error at *MASKWRITE* ■ 3: Error at *READ* ■ 4: Error at *READDELTA*
809Ah	Interface missing
809Bh	Interface not configured (SFC 216)

5.1.2 FC/SFC 216 - SER_CFG - Parametrization PtP

Description

Please note that this block can only be used in Yaskawa CPUs or in S7-300 CPUs from Siemens!

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.
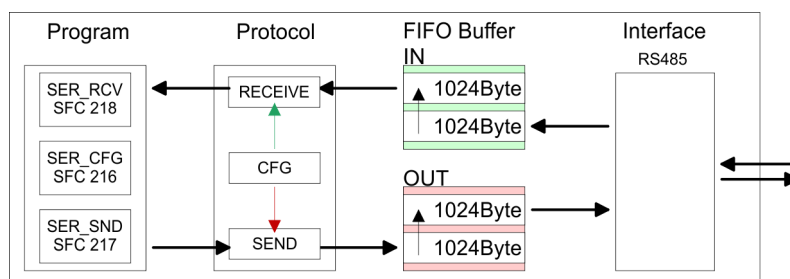
Parameters

Parameter	Declaration	Data type	Description
PROTOCOL	IN	BYTE	1=ASCII, 2=STX/ETX, 3=3964R

Parameter	Declaration	Data type	Description
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: 19200bit/s x 0.008s ≈ 154bit → (9Ah)
- 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[1]	3: 2bit

1) 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 2.4.4 with a System MICRO CPU you can switch between RS422 and RS485 communication.*
>
> *0: RS422 communication*
>
> *1: RS485 communication*

RETVAL 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 *PROTOCOL* 2: Error at *PARAMETER* 3: Error at *BAUDRATE* 4: Error at *CHARLENGTH* 5: Error at *PARITY* 6: Error at *STOPBITS* 7: Error at *FLOWCONTROL*
809xh	Error in FC/SFC parameter value x, where x: 1: Error at *PROTOCOL* 3: Error at *BAUDRATE* 4: Error at *CHARLENGTH* 5: Error at *PARITY* 6: Error at *STOPBITS* 7: Error at *FLOWCONTROL* (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 ...

5.1.3   FC/SFC 217 - SER_SND - Send to PtP

Description

> ⓘ *Please note that this block can only be used in Yaskawa CPUs or in S7-300 CPUs from Siemens!*

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)

Error code	Description
8x24h	Error in FC/SFC parameter x, where x:
	1: Error in *DATAPTR*
	2: Error in *DATALEN*
8122h	Error in parameter *DATAPTR* (e.g. DB too short)
807Fh	Internal error
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

Error code	Description
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

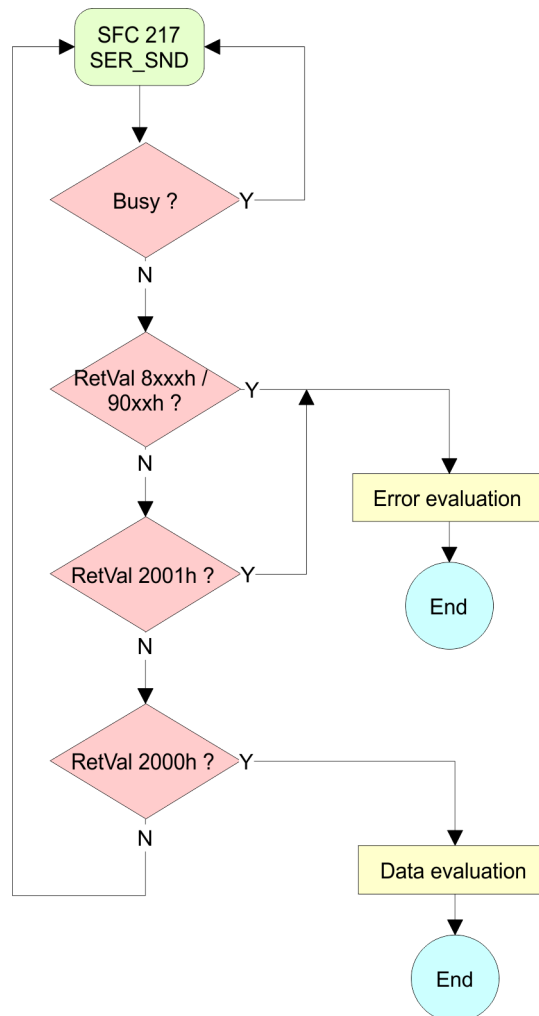| Error code	Description
2000h	Send ready (positive slave respond)
2001h	Send ready (negative slave respond)
8080h	Receive buffer overflow (no space for receipt)
8090h	Acknowledgement delay time exceeded
80F0h	Wrong checksum in respond
80FDh	Length of respond too long
80FEh	Wrong function code in respond
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

```
                    ┌──────────────┐
              ┌────►│   SFC 217    │◄────┐
              │     │   SER_SND    │     │
              │     └──────┬───────┘     │
              │            ▼             │
              │        ◇ Busy ? ◇────Y───┘
              │            │
              │            N
              │            ▼
              │    ◇ RetVal 8xxxh / ◇──Y──────────────┐
              │    ◇   90xxh ?     ◇                   │
              │            │                           │
              │            N                           │
              │            ▼                      ┌──────────────┐
              │    ◇ RetVal 2001h ? ◇──Y──────────│Error evaluation│
              │            │                      └──────┬───────┘
              │            N                             ▼
              │            ▼                          (  End  )
              │    ◇ RetVal 2000h ? ◇──Y──────────┐
              │            │                       │
              │            N                       ▼
              └────────────┘               ┌──────────────┐
                                           │Data evaluation│
                                           └──────┬───────┘
                                                  ▼
                                               (  End  )
```

USS / Modbus

```
                    ┌──────────────┐
              ┌────▶│   SFC 217    │◀────┐
              │     │   SER_SND    │     │
              │     └──────┬───────┘     │
              │            ▼             │
              │          ╱   ╲           │
              │        ╱ Busy ? ╲──Y─────┘
              │        ╲       ╱
              │          ╲   ╱
              │            │ N
              │            ▼
              │          ╱        ╲                  ┌──────────────┐
              │        ╱ RetVal 8xxxh/ ╲──Y────────▶│   SFC 218    │
              │        ╲   90xxh ?    ╱              │   SER_RCV    │
              │          ╲          ╱                └──────┬───────┘
              │            │ N                              ▼
              │            ▼                        ┌──────────────────┐
              │          ╱        ╲                 │ Error evaluation │
              │        ╱ RetVal 2001h?╲──Y──┘       └────────┬─────────┘
              │        ╲            ╱                         ▼
              │          ╲        ╱                        ( End )
              │            │ N
              │            ▼
              │          ╱        ╲                  ┌──────────────┐
              │        ╱ RetVal 2000h?╲──Y────────▶ │   SFC 218    │
              │        ╲            ╱               │   SER_RCV    │
              │          ╲        ╱                 └──────┬───────┘
              │            │ N                             ▼
              └────────────┘                      ┌──────────────────┐
                                                  │  Data evaluation │
                                                  └────────┬─────────┘
                                                           ▼
                                                        ( End )
```

ASCII / STX/ETX

```
              ┌──────────────┐
              │   SFC 217    │
              │   SER_SND    │
              └──────┬───────┘
                     ▼
                   ╱        ╲
                 ╱ RetVal 900xh ╲──Y──┐
                 ╲            ╱        │
                   ╲        ╱          ▼
                     │ N      ┌──────────────────┐
                     │        │ Error evaluation │
                     │        └────────┬─────────┘
                     ▼◀───────────────┘
                  ( End )
```
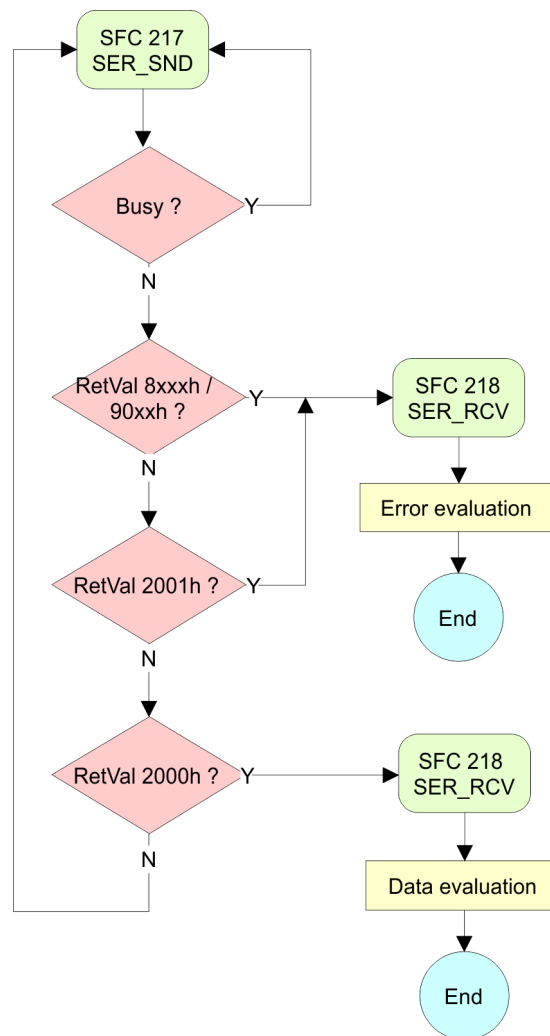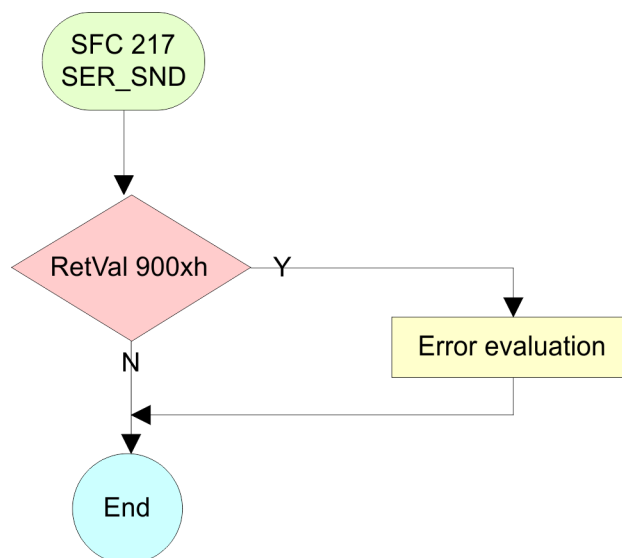
5.1.4 FC/SFC 218 - SER_RCV - Receive from PtP

Description

> ⓘ *Please note that this block can only be used in Yaskawa CPUs or in S7-300 CPUs from Siemens!*

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	overrun	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.

Bit	Error	Description
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 *DATAPTR* 2: Error at *DATALEN* 3: Error at *ERROR*
8122h	Error in parameter *DATAPTR* (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.
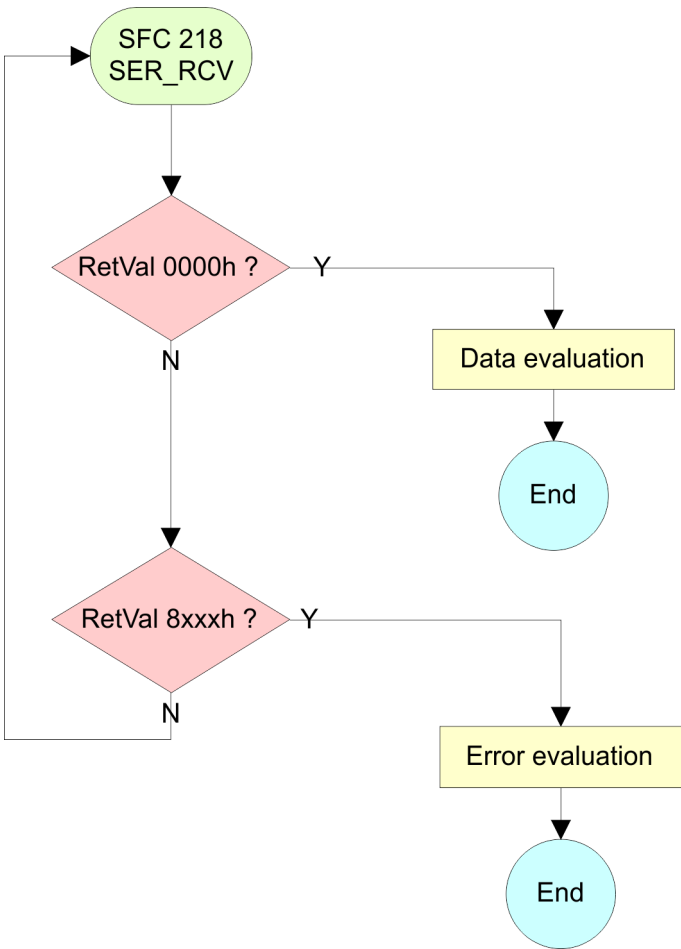
5.1.5   FB 1 - RECEIVE_ASCII - Receiving with defined length from PtP

Description

> ⓘ *Please note that this block can only be used in Yaskawa CPUs or in*
> *S7-300 CPUs from Siemens!*

This FB collects the data, which are received via the internal serial interface in PtP opera-
tion 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 *EMPF_PUFFER* is faulty or does not exist.
9001h	DB setting in *EMPF_PUFFER* is faulty or does not exist.
9002h	Length setting in *EMPF_PUFFER* is faulty or does not exist.

5.1.6 FB 7 - P_RCV_RK - Receive from CP 341

Description

> *Please note that this block can only be used in Yaskawa CPUs or in S7-300 CPUs from Siemens!*

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.
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®). But they may be used by loadable protocols.
NDR[1]	OUT	BOOL	Request complete without errors, data received Parameter *STATUS* = 00h
ERROR[1]	OUT	BOOL	Request complete with error Parameter *STATUS* contains error details
LEN[1]	OUT	BOOL	Length of the received telegram in byte $1 \leq LEN \leq 1024$

Parameter	Declaration	Data type	Description
STATUS[1]	OUT	WORD	Specification of the error on *ERROR* = 1

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 me cancelled 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 Instance 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.

5.1.7 FB 8 - P_SND_RK - Send to CP 341

Description

> *Please note that this block can only be used in Yaskawa CPUs or in S7-300 CPUs from Siemens!*

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 ≤ *DBB_NO* ≤ 8190
LEN	IN	INT	Length of message frame to be sent in byte 1 ≤ *LEN* ≤ 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[1]	OUT	BOOL	Request complete without errors, data sent Parameter *STATUS* = 00h
ERROR[1]	OUT	BOOL	Request complete with error Parameter *STATUS* contains error details
STATUS[1]	OUT	WORD	Specification of the error on *ERROR* = 1

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 me cancelled 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 Instance 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.

5.2   CP040

5.2.1   Overview

Communication principle
- By a cyclic call of FB 60 SEND and FB 61 RECEIVE or FB 65 CP040_COM data may be cyclically sent and received by the CP.
- On the CP the transmission of the communication protocols to the communication partner takes place, which may be configured by the hardware configuration.
- A telegram to be sent is divided into blocks in the CPU depending on the IO size and transferred via the data channel to the CP In the CP these blocks are assembled in the send buffer, and when the telegram is complete, the telegram is sent by the serial interface.
- The exchange of received telegrams via the backplane bus is asynchronous.
- If a complete telegram was received via the serial interface, it is stored in a 1024byte ring buffer. From the length of the still free ring buffer the maximum length of a telegram results.
- Depending upon the parametrization up to 250 telegrams can be buffered, whereby their overall length may not exceed 1024.
- If the buffer is full, arriving telegrams are rejected.
- A complete telegram is divided into blocks, depending on the parametrized IO size, and transferred to the backplane bus.
- The data blocks must be assembled in the CPU.
- Since the data exchange via the backplane bus runs asynchronously, a software handshake is used between the CP and the CPU. For this, both handling blocks have the common CONTROL parameter. The same flag byte is to be used for this parameter.
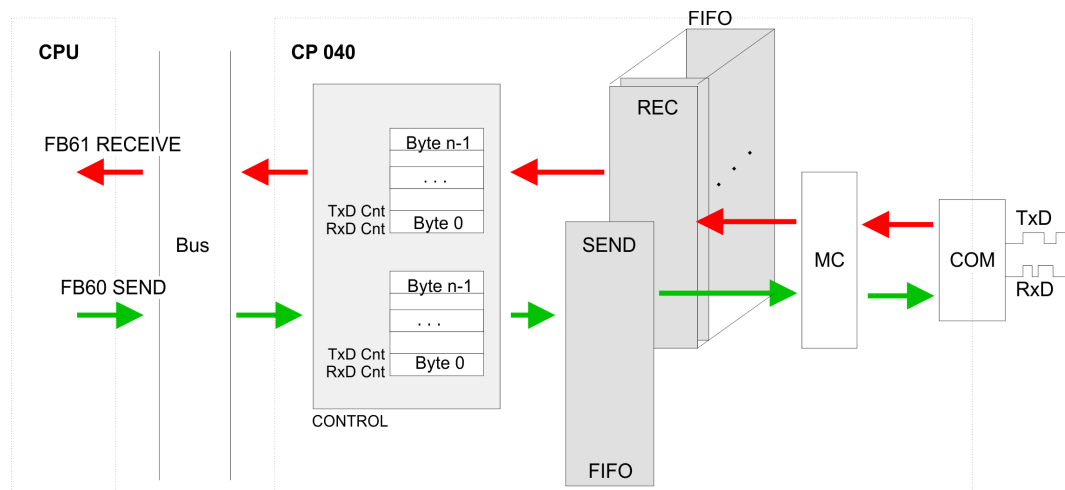
FIFO   Ring buffer max. 250 telegrams 1024byte
CONTROL   Software handshake via CONTROL block

> *For recognizing a signal change a minimum pulse time is necessary. The decisive factors are CPU cycle time, the refresh time on the CP and the response time of the communication partner.*

5.2.2   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 / HW_ID	IN	INT / HW_IO	▪ LADDR
			  – Logical base address of the CP.
			  – When used in Yaskawa CPUs or S7-300 CPUs from Siemens.
			▪ HW_ID
			  – *HW identifier* to address the CP.
			  – When used in S7-1500 CPUs from Siemens.

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[1]	OUT	BOOL	Send order finished without errors.
ERROR[1]	OUT	BOOL	Send order finished with errors. Parameter *STATUS* contains the error information.
STATUS[1]	OUT	WORD	Specification of the error with *ERROR* = 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).

1) 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:

- This parameter is only available in Yaskawa CPUs or in S7-300 CPUs from Siemens.
- 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.

HW_ID

HW identifier:

- This parameter is only available in S7-1500 CPUs of Siemens.
- Enter at *HW_ID* the *HW identifier*, with which your module can be addressed accordingly. ➥ *'HW identifier - HW_ID'...page 10*

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	There is no error.
0202h	Possible sources of error: ■ Handling block and CP are not synchronous (remedy: Trigger synchronous reset) ■ *IO_SIZE* is not valid (*IO_SIZE* = 0 or *IO_SIZE* > 60).
0301h	DB is not valid.
0517h	*LEN* is not valid (*LEN* = 0 or *LEN* > 1024).
070Ah	Transfer failed, there is no response of the partner or the job was negative acknowledged.
8090h	*HW_ID* is unknown.
80A0h	When accessing the periphery, an access error was detected.
80A1h	
8181h	Job is running (status and no error message).
8323h	Send DB available, but too short.
833Ah	Send DB not readable (DB not available or DB in optimized block access). → *'No optimized block access'...page 7*

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.

5.2.3 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.
LADDR / HW_ID	IN	INT / HW_IO	▪ LADDR – Logical base address of the CP. – When used in Yaskawa CPUs or S7-300 CPUs from Siemens. ▪ HW_ID – *HW identifier* to address the CP. – When used in S7-1500 CPUs from Siemens.
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[1]	OUT	BOOL	Receive order finished without errors.
ERROR[1]	OUT	BOOL	Receive order finished with errors. Parameter *STATUS* contains the error information.
STATUS[1]	OUT	WORD	Specification of the error with *ERROR* = 1.
CONTROL	IN_OUT	BYTE	Divided byte with RECEIVE handling block: SEND (bit 0 … 3), RECEIVE (bit 4 … 7).

1) 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:

- This parameter is only available in Yaskawa CPUs or in S7-300 CPUs from Siemens.
- 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.

HW_ID

HW identifier:

- This parameter is only available in S7-1500 CPUs of Siemens.
- Enter at *HW_ID* the *HW identifier*, with which your module can be addressed accordingly. ➥ *'HW identifier - HW_ID'...page 10*

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 ≤ *LEN* ≤ 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	There is no error.
0202h	Possible sources of error: ■ Handling block and CP are not synchronous (remedy: Trigger synchronous reset) ■ *IO_SIZE* is not valid (*IO_SIZE* = 0 or *IO_SIZE* > 60).
0301h	DB is not valid.
070Ah	Transfer failed, there is no response of the partner or the job was negative acknowledged.
8090h	*HW_ID* is unknown.
80A0h	When accessing the periphery, an access error was detected.
80A1h	
8181h	Job is running (status and no error message).
8323h	Receive DB available, but too short.
833Ah	Receive DB not readable (DB not available or DB in optimized block access). ➡ *'No optimized block access'...page 7*

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.

5.2.4   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 when used in Yaskawa CPUs or S7-300 CPUs from Siemens

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.

Name	Declaration	Type	Description
ADDR_OUT	IN	INT	Logical output base address of the CP from the Hardware configuration.
ADDR_IN	IN	INT	Logical input base address of the CP from the Hardware configuration.
IO_SIZE	IN	WORD	Configured IO size of the module.
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.
DONE_SEND[1]	OUT	BOOL	Send order finished without errors. Data sent: Parameter *STATUS_SEND* = 0000h.
ERROR_SEND[1]	OUT	BOOL	Send order finished with errors. Here Parameter *STATUS_SEND* contains the corresponding error message.
NDR_RCV[1]	OUT	BOOL	Receive order finished without errors. Data sent: Parameter *STATUS_RCV* = 0000h. The Parameter is available until a cycle.
ERROR_RCV[1]	OUT	BOOL	Receive order finished with errors. Parameter *STATUS_RCV* contains the error information.
STATUS_SEND[1]	OUT	WORD	Specification of the error with *ERROR_SEND* = 1
LEN_RCV	OUT	INT	Length of received telegram in byte $1 \leq LEN\_RCV \leq 1024$
STATUS_RCV[1]	OUT	WORD	Specification of the error with *ERROR_RCV* = 1

1) Parameter is available until the FB is called.

Parameters when used in S7-1500 CPUs from Siemens

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.
HW_ID	IN	HW_IO	*HW identifier* to address the CP.
IO_SIZE	IN	WORD	Configured IO size of the module.
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.

Name	Declaration	Type	Description
LEN_SEND	IN	INT	Length of telegram in byte, to be sent.
			$1 \le LEN\_SEND \le 1024$
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.
DONE_SEND[1]	OUT	BOOL	Send order finished without errors.
			Data sent: Parameter *STATUS_SEND* = 0000h.
ERROR_SEND[1]	OUT	BOOL	Send order finished with errors.
			Here Parameter *STATUS_SEND* contains the corresponding error message.
NDR_RCV[1]	OUT	BOOL	Receive order finished without errors.
			Data sent: Parameter *STATUS_RCV* = 0000h.
			The Parameter is available until a cycle.
ERROR_RCV[1]	OUT	BOOL	Receive order finished with errors.
			Parameter *STATUS_RCV* contains the error information.
STATUS_SEND[1]	OUT	WORD	Specification of the error with *ERROR_SEND* = 1
LEN_RCV	OUT	INT	Length of received telegram in byte
			$1 \le LEN\_RCV \le 1024$
STATUS_RCV[1]	OUT	WORD	Specification of the error with *ERROR_RCV* = 1

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:

- This parameter is only available in Yaskawa CPUs or S7-300 CPUs from Siemens.
- 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:

- This parameter is only available in Yaskawa CPUs or S7-300 CPUs from Siemens.
- 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.

HW_ID

HW identifier:

- This parameter is only available in S7-1500 CPUs of Siemens.
- Enter at *HW_ID* the *HW identifier*, with which your module can be addressed accordingly. ➥ *'HW identifier - HW_ID'...page 10*

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	There is no error.
0202h	*IO_SIZE* is not valid (*IO_SIZE* = 0 or *IO_SIZE* > 60).
0301h	DB is not valid.
070Ah	Transfer failed, there is no response of the partner or the job was negative acknowledged.
0517h	*LEN* is not valid (*LEN* = 0 or *LEN* > 1024).
8090h	*HW_ID* is unknown.
80A0h	When accessing the periphery, an access error was detected.
80A1h	
8181h	Job is running (status and no error message).
8323h	Send DB available, but too short.
833Ah	Send DB not readable (DB not available or DB in optimized block access. ➥ *'No optimized block access'...page 7*)

LEN_RCV

Length Receive:

- Length of the received telegram in byte.
- 1 ≤ *LEN_RCV* ≤ 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	There is no error.
0202h	*IO_SIZE* is not valid (*IO_SIZE* = 0 or *IO_SIZE* > 60).
0301h	DB is not valid.
070Ah	Transfer failed, there is no response of the partner or the job was negative acknowledged.
080Ah	A free receive buffer is not available.

STATUS	Description
080Ch	Wrong character received (Character frame or parity error)
8090h	*HW_ID* is unknown.
80A0h	When accessing the periphery, an access error was detected.
80A1h	
8181h	Job is running (status and no error message).
8323h	Receive DB available, but too short.
833Ah	Receive DB not readable (DB not available or DB in optimized block access). ➡ *'No optimized block access'...page 7*

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.

5.3   CP240

5.3.1   FC 0 - SEND_ASCII_STX_3964 - 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.

Parameter

Name	Declaration	Type	Description
ADR	IN	INT	Periphery address
_DB	IN	BLOCK_DB / DB_ANY	_DB ■ DB number of *Type* BLOCK_DB containing data to send, used in Yaskawa CPUs or in S7-300 CPUs from Siemens. ■ Pointer of *Typ* DB_ANY to the DB containing data to send, used in S7-1500 CPUs from Siemens.
ABD	IN	WORD	Number of the 1. data word.
ANZ	IN	WORD	Number of bytes.

Name	Declaration	Type	Description
PAFE	OUT / IN_OUT	BYTE	PAFE (0 = OK) ■ Parametrization error code of *Type* BYTE as OUT parameter, used in Yaskawa CPUs or in S7-300 CPUs from Siemens. ■ Parametrization error code of *Type* BYTE as IN_OUT parameter, used in S7-1500 CPUs from Siemens.
FRG	IN_OUT	BOOL	Start bit of the function.
GESE	IN_OUT	WORD	Is internally used.
ANZ_INT	IN_OUT	WORD	Is internally used.
ENDE_KOM	IN_OUT	BOOL	Is internally used.
LETZTER_BLOCK	IN_OUT	BOOL	Is internally used.
SENDEN_LAEUFT	IN_OUT	BOOL	Status of the function.
FEHLER_KOM	IN_OUT	BOOL	Is internally used.

ADR

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

_DB

■ Number of *Type* BLOCK_DB of the data block, which contains the data to send for the CP, used in Yaskawa CPUs or S7-300 CPUs from Siemens.

■ Pointer of *Type* DB_ANY to the data block, which contains the data to send for the CP, used in S7-1500 CPUs from Siemens.

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.

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

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!

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 FRG, ENDE_KOM, LETZTER _BLOCK, SENDEN_LAEUFT and FEHLER_KOM must always be stored in a bit memory byte.

5.3.2 FC 1 - RECEIVE_ASCII_STX_3964 - 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.

Parameter

Name	Declaration	Type	Description
ADR	IN	INT	Periphery address
_DB	IN	BLOCK_DB / DB_ANY	_DB ■ DB number of *Type* BLOCK_DB containing data received, used in Yaskawa CPUs or in S7-300 CPUs from Siemens. ■ Pointer of *Type* DB_ANY to the DB containing data received, used in S7-1500 CPUs from Siemens.
ABD	IN	WORD	Number of the 1. data word.
ANZ	OUT / IN_OUT	WORD	ANZ ■ Number of received bytes of *Type* WORD as OUT parameter, used in Yaskawa CPUs or in S7-300 CPUs from Siemens. ■ Number of received bytes of *Type* WORD as IN_OUT parameter, used in S7-1500 CPUs from Siemens.
PAFE	OUT / IN_OUT	BYTE	PAFE (0 = OK) ■ Parametrization error code of *Type* BYTE as OUT parameter, used in Yaskawa CPUs or in S7-300 CPUs from Siemens. ■ Parametrization error code of *Type* BYTE as IN_OUT parameter, used in S7-1500 CPUs from Siemens.
EMFR	IN_OUT	BOOL	Acknowledgment of receipt
GEEM	IN_OUT	WORD	Is internally used.
ANZ_INT	IN_OUT	WORD	Is internally used.
EMPF_LAEUFT	IN_OUT	BOOL	Status of the function.
LETZTER_BLOCK	IN_OUT	BOOL	Is internally used.
FEHLER_EMPF	IN_OUT	BOOL	Is internally used.
OFFSET	IN_OUT	WORD	Is internally used.

ADR

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

_DB

■ DB number of *Type* BLOCK_DB containing data received from the CP, used in Yaskawa CPUs or in S7-300 CPUs from Siemens.

■ Pointer of *Type* DB_ANY to the data block, containing data received from the CP, used in S7-1500 CPUs from Siemens.

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.
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
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.
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 EMFR, LETZTER_BLOCK, EMPF_LAEUFT and FEHLER_EMPF must always be stored in a bit memory byte.

5.3.3   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	■ 0: do nothing ■ 1: set state RTS
MASKE_DTR	IN	BOOL	■ 0: do nothing ■ 1: set state DTR
RET_VAL	OUT	WORD	Return value (0 = OK)
STATUS	OUT / IN_OUT	BYTE	STATUS ■ Status flags of *Type* BYTE as OUT parameter, used in Yaskawa CPUs or in S7-300 CPUs from Siemens. ■ Status flags of *Type* BYTE as IN_OUT parameter, used in S7-1500 CPUs from Siemens.

Name	Declaration	Type	Comment
DELTA_STATUS	OUT / IN_OUT	BYTE	DELTA_STATUS ■ Status flags of change between 2 accesses of *Type* BYTE as OUT parameter, used in Yaskawa CPUs or in S7-300 CPUs from Siemens. ■ Status flags of change between 2 accesses of *Type* BYTE as IN_OUT parameter, used in S7-1500 CPUs from Siemens.
START	IN_OUT	BOOL	Start bit of the function
AUFTRAG_LAEUFT	IN_OUT	BOOL	Status of function

> *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.

RET_VAL

At this time, this parameter always returns 00h and is reserved for future error messages.

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_LAEUFT

As long as the function is executed, this bit remains set.

5.3.4 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	TIMER	Timer
ANL	IN_OUT	BOOL	CPU restart progressed
ZERO	IN_OUT	BOOL	Internal use
RESET	IN_OUT	BOOL	Reset the CP
TIME_AN	IN_OUT	BOOL	Internal use
STEUERB_S	IN_OUT	BYTE	Internal use
STEUERB_R	IN_OUT	BYTE	Internal use

ADR

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

TIMER_NR

Timer for the delay time.

ANL

With *ANL* = 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, *ANL* is automatically set back.

ZERO

Parameter is used internally.

RESET

RESET = 1 allows you to set back the CP out of your user application.

TIME_AN

Parameter is used internally.

STEUERB_S

Here you have to set the bit memory byte where the control bits FRG, 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 EMFR, LETZTER_BLOCK, EMPF_LAEUFT and FEHLER_EMPF for the RECEIVE-FC are stored.

5.3.5 FC 11 - ASCII_FRAGMENT - Receive fragmented from CP 240

Description

> ⓘ *Please note that this block can only be used in Yaskawa CPUs or in S7-300 CPUs from Siemens!*

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 number of DB containing data received.
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 be received.

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 *EMFR*, the handling block announces that data has been received. Only by 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

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.