SIEMENS

SIMATIC NET

Industrial Remote Communication - TeleControl Telecontrol ST7

Configuration Manual

V3.0 SP2

06/2020
C79000-G8976-C578-02

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠ DANGER

indicates that death or severe personal injury **will** result if proper precautions are not taken.

⚠ WARNING

indicates that death or severe personal injury **may** result if proper precautions are not taken.

⚠ CAUTION

indicates that minor personal injury can result if proper precautions are not taken.

NOTICE

indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

⚠ WARNING

Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Siemens AG
Digital Industries
Postfach 48 48
90026 NÜRNBERG
GERMANY

C79000-G8976-C578-02
℗ 06/2020 Subject to change

Preface

Validity of this manual

This configuration manual is valid for the "Telecontrol ST7" block library version V3.0 SP2

The block library is to be used in STEP 7 Professional. See the applicable versions in the next section.

New in this version

New functions:

- New data point typicals for CPU 1500:
  - Command typicals: Cmd01B_FS / Cmd01B_FR
  - Setpoint typical: Set01W_FS
  - Parameter typical: Par12D_FS

  Send command, setpoint and parameter frames without 1-out-of-n check (FC Safe not required).

  Can be used as of STEP 7 V15
- Similar to TD7onTIM, an Xcelerated general request (XGR) can also be performed via the blocks of the CPU 1500.

  Can be used as of STEP 7 V15
- New blocks for TD7onCPU for S7-300: This allows a CPU 300 to communicate via a local TIM 1531 IRC.

  Can be used as of STEP 7 V17

Enhanced functions:

The ListGenerator1500 and ListGenerator400 blocks can now receive more than 5461 objects without a target object number (max. 10922 receive objects).

Can be used as of STEP 7 V15

Replaced manual edition

Edition 01/2019

Current manual edition on the Internet

You will also find the current version of this manual on the Internet pages of Siemens Industry Online Support:

Link: (https://support.industry.siemens.com/cs/ww/en/view/109764356)

Cross references

In this manual there are often cross references to other sections.

To be able to return to the initial page after jumping to a cross reference, some PDF readers support the command <Alt>+<left arrow>.

Table of contents

Block library TeleControl ST7

1

1.1 Creating a library

Delivery form of the block library

The program blocks of the TD7onCPU library are used as global library for STEP 7 in the TIA Portal.

The library is archived and compressed on the Internet at the following address:
Link: (https://support.industry.siemens.com/cs/ww/en/view/109755374)

The file name has the following structure: Telecontrol_ST7_<Version>.exe

Creating the library

To use the library in STEP 7, follow these steps:

1. Create a suitable directory in the STEP 7 installation directory of your engineering station, for example, one named "Telecontrol ST7".
2. Save the library in this directory.
3. Run the "*.exe" file.

   The self-extracting library is stored as a zip archive in a selectable directory.
4. Unzip the file.

   The library is now available as an archived global library under the name "Telecontrol ST7.zal<version>" together with the readme files.
5. Open the STEP 7 project.
6. In the "Options" menu, select the command "Global libraries > Retrieve library".

   The "Retrieve archived global library" dialog opens.
7. Select the folder and the library archive.

   Recommendation: Leave the default "Open read-only" option activated to download the global library read-only.
8. Click Open.

   The "Select target directory" window opens.
9. Select the target directory (your project) to which the archived global library is to be unpacked (select folder).

   The library is unpacked.

   If you are working with a newer STEP 7 version, you will be asked whether you want to upgrade the library. Confirm this using the "Upgrade" button.

The library is opened after unpacking.

You can find the library in the STEP 7 project in the library view (task card) in the "Global libraries" pane under the name "Telecontrol ST7_<Version>".

1.2 Structure of the library

Structure of the global library

The opened library contains the following directories:

- **Types**
  - PLC data types (UDT)

    Here you will find the PLC data types (UDTs) used by the libraries.
  - System data types (SDT)

    Here you will find the system data types used by the libraries.
- **Master copies**

  You can find the following program blocks here:
  - **CPU300/400**

    Program blocks for the S7-300/400 CPU
  - **CPU1500**

    Program blocks for the S7-1500 CPU

Master copies

The two libraries under "Master copies" for S7-300/400 and S7-1500 are divided into the following folders:

- Data point typicals

  Data point typicals (blocks) for transferring data
- Optional blocks

  Optional blocks that you can use when necessary.

- System blocks

  Blocks that are used for basic communication.

  Depending on the SIMATIC product series used, you can find the following auxiliary blocks for communication between TIM and CPU in separate folders:

  – BCom

    Blocks for processing the communication mailbox of the S7-400 and S7-1500

    Blocks for processing the communication compartment of the S7-300 (from version V3.0 SP2 of the block library)

  – PCom

    Blocks for processing the communication mailbox of an S7-300 with P bus (CPU without partyline)

  – XCom

    Blocks for processing the communication mailbox of an S7-300 with X connections

For information on the "party line", see Glossary (Page 175).

1.3 Using, calling and generating the blocks

Generating TD7onCPU blocks

Once you have opened the TD7 library in the pane "Global libraries" under the name "Telecontrol ST7", you can generate the blocks for basic communication for one or more CPUs.

Requirements

The following requirements must be met before generating the blocks:

- The communications modules must be networked with each other.
- Valid telecontrol connections must be created between the communication partners involved.
- A TIM 1531 IRC must be networked with a CPU via Ethernet.
- A TIM 1531 IRC cannot be assigned to a CPU.

  The "Assigned CPU" field in the "Subscriber numbers" parameter group must be empty.
- No data point must be created in the data point editor of a TIM.
- No message must be created in the message editor of a TIM.
- The "Telecontrol ST7" library must be open (visible) in the "Global libraries" task card.

Different requirements for modules with imported configuration data

Note the other requirements for communications modules whose SINAUT configuration was imported from a STEP 7 V5 project. With these modules, the option "Telecontrol configuration" is set to "Import" in the parameter group "Basic settings > Configuration".

These modules have different requirements for generating the blocks:

- No ST7 telecontrol connections need to be created between the communication partners involved.

  The relevant information is stored in the imported configuration data.

- The CPU of the communications module must have the same IP address as in the STEP 7 V5 project. The CPU is assigned to the communications module via its IP address.

Procedure for generating

Generate the blocks for basic communication as follows:

1. Select the desired TIM.
2. Select the shortcut menu command (right mouse button) "Generate TD7onCPU blocks".
3. In the dialog that follows, select one or more CPUs and click "Generate..."

The TD7onCPU blocks for the SINAUT basic communication are created and compiled in the selected CPUs under the program blocks in a new directory "TD7onCPU".

Using blocks

- **System blocks**

  The required blocks for basic communication are created automatically during the generation in the CPU.

  You do not need to use or call these blocks manually in the CPU.

  The system blocks handle central tasks such as startup, monitoring of the connections, reachability of the connection partners, entering data in the send mailbox or taking data from the receive mailbox of the communication DBs, general request, time keeping, handling communication etc.

  Communication DBs:

  – S7-300/400

    For each connection between the TIM and the CPU, a separate communication DB is created automatically: It contains the send and receive mailbox and all the data required for controlling and monitoring this connection.

  – S7-1500

    For the CPU 1500, there is only one communication DB "BConnectData". A connection instance is stored in this DB in the "BConnection" array for each local connection with the TIM.

- **Data point typicals**

  You need to use data point typicals on the relevant CPU according to the communications tasks.

  Drag the individual blocks with the mouse from the global library to the user program of the CPU. The corresponding instance data block is created automatically when using a data point typical.

  The number of the instance data block is also the source object number, which you need to specify at the corresponding partner object of the communication partner under "PartnerObjectNo". For this reason, each typical needs to use its own instance data block. You can also assign the instance DB number manually.

- **PLC data types (UDT)**

  – Dat256D_S / Dat256D_R

    Copy also the UDT "TransmitBlock" for these typicals from the library (Types > PLC data types (UDT)) into the "PLC data types"directory of the CPU. The UDT is not automatically referenced by the typical.

- **Optional blocks**

  You should also use the optional blocks on the CPU depending on the communications tasks of the station.

  Drag the individual blocks with the mouse from the global library to the user program of the CPU.

Call in the user program

- **System blocks**

  Only the two following FCs need to be called in the user program:

  – StartUp

    Call the FC in the startup OB. The block has no parameters.

    To be able to call the FC StartUp, you must first create the startup OB (OB100).

  – BasicTask

    Call the FC in the cyclic OB or user program.

    The DB BasicData is created automatically. The DB contains all the centrally required data including the records of all communication partners and the connections to be managed.

    For information on the various "BasicTask" FCs, see section Note on the "BasicTask_*" FCs (Page 12).

- **Data point typicals**

  Depending on the communication tasks, call up all required data point typicals in the user program.

- **Optional blocks**

  Depending on the communication tasks, call up all required optional blocks in the user program.

See also section The cyclic SINAUT program (Page 13) for information on calling the blocks.

Note

Error display for library conformance

In the "Compilation" parameter group of blocks, an error can be displayed under the entry "Library conformance" (call of single instances). This is caused by the call of global data blocks, for example, DB BasicData.

Since the "Telecontrol ST7" library has no "know-how protection", you can ignore these error notifications.

1.4 Note on the "BasicTask_*" FCs

Versions of the "BasicTask" FC

There are different versions of the "BasicTask" FC in the TD7 block library "Telecontrol ST7" for STEP 7 Professional.

Depending on the SIMATIC product series and the backplane bus, the FCs have a different suffix at the end of the block name:

- **BasicTask_B**

  For S7-1500 CPU

- **BasicTask_B**

  For S7-300 and S7-400 CPU

- **BasicTask_X**

  For S7-300 PU with partyline

- **BasicTask_P**

  For S7-300 CPU without partyline

For information on "Partyline", see Glossary.

For the block description, see section BasicTask_* FC (Page 167).

"BasicTask" label for three FCs

References to the respective FC are made in many places in the following sections. Since the description of the different blocks usually applies to several SIMATIC product series, the three FCs are referred to hereafter as "BasicTask". The FC version for the relevant CPU type is meant in each context.

1.5 Structure of user program for TD7onCPU

1.5.1 The cyclic SINAUT program

Introduction

The structure of the TD7 blocks in the user program (OB1) is explained below.

The TD7 blocks must be proceeded in every OB1 cycle. Keep to the call sequence of the blocks unless instructed otherwise.

Other user-specific parts of the program can be linked in before or after the TD7 blocks in OB1 or, if practical, within the TD7 blocks.

You can structure the SINAUT-specific part in OB1 by calling it in lower-level FCs.

All data point typicals are FBs. An instance DB must be specified when an FB is called. The number of this instance DB is identical to the object number of the data point object.

The structure of the TD7 blocks in OB1

Cyclic OB1	
BasicTask	The respective FC BasicTask must always be called at the start. It handles basic SINAUT tasks that are always required.
Optional SINAUT basic functions	After FC BasicTask, call the following optional blocks (FCs) if you need them: • ListGenerator Creation of address lists for received frames with incomplete destination address. • TimeTask Provides the time of day.
	If necessary, call additional optional FCs in the subsequent program execution, for example: • Trigger Scheduled start of user programs and data frames • PartnerStatus Display the status of partners • PartnerMonitor Extended subscriber-specific display and control options. • PulseCounter Count pulse acquisition • PathStatus Display the status of the connection path to a partner

Cyclic OB1	
	Note: The following two blocks may not be called explicitly, since they are activated via an internal interface. • TestCopy • TestcopyDB
Data point typicals	In the subsequent program execution, call the data point typicals for sending and receiving data. The sequence of the individual typicals is unimportant. The number of typicals to call and the required types depend on the amount and type of data to be sent and received. **Note:** The instance DBs of these FBs are referenced using the partner object number of the communication partner.

Example of a TD7 program for a station

Cyclic OB1	
BasicTask	The respective FC BasicTask must always be called at the start of the cyclic program. Generally, data is ready for further processing after it has been received. The FC has only one parameter, "UserFC". Normally 0 can be specified. If you require user-specific processing for received data, specify here the number of an FC containing the program for this processing. You can also copy received data chronologically to an archive memory via this interface.
TimeTask	As an option, you can call the FC TimeTask immediately after FC BasicTask. The FC has no parameters. FC TimeTask TimeTask must be included if you need the time of day. This allows data frames to be time-stamped. However, you can also use the time of day to start program components at a specific point in time or to schedule the transmission of data frames. FC Trigger, described below, is then required. For this FC Time Task to be used, the CPU must be provided with the time from a local TIM module. You specify this in the configuration.
Trigger	FC Trigger can be included as an option. The FC sets its output for the duration of one OB1 cycle when the point in time or the time interval set for the FC has been reached. The FC can be inserted several times if several times or various time intervals are required. Using the FC requires that FC TimeTask must be called first in the OB1 program (see above) and the CPU time must have already been set once.
PartnerStatus	FC PartnerStatus can be included as an option. The FC shows the reachability for a maximum of 8 communications partners.

Cyclic OB1	
ListGenerator	• ListGenerator300 for S7-300 CPU • ListGenerator400 for S7-400 CPU • ListGenerator1500 for S7-1500 CPU The FC can be installed as an option. The FC is required if the station receives frames containing no destination address or an incomplete destination address. This happens if the configuration of the destination address has been completely or partially omitted in a partner for data point typicals. ("PartnerNo" and "PartnerObjectNo" were not specified, so transmission to all known destination subscribers takes place.)
PulseCounter **PathStatus** **PartnerMonitor**	When necessary
Data point typicals	Following the FCs for SINAUT basic tasks, data point typicals for sending and receiving data are called. The sequence of the individual typicals is unimportant. The number of typicals to call and the required types depend on the amount and type of data to be sent and received. For one station applies: • The following are typically sent: – Binary information, such as status messages and alarms – Analog values – Counted values – Additional data if necessary • The following are typically received: – Commands – Setpoints, parameters
Bin04B_S	One or more FBs for the acquisition and transfer of binary information, such as messages, alarms, etc.
Ana04W_S	One or more FBs for the acquisition and transfer of analog values
Cnt01D_S / Cnt04D_S	One or more FBs for the acquisition and transfer of counted values A requirement for the use of the FBs is that FC PulseCounter is included in a cyclic interrupt OB, e.g. OB35. This FC is responsible for the time-driven acquisition of counted pulses in the background.
Cmd01B_R	One or more FBs for receipt and output of commands.
Set01W_R / Par12D_R	One or more FBs for receipt and output of setpoints or parameters
Dat12D_S / Dat256D_S	One or more FBs for the acquisition and transmission of 1 to 12 or up to 256 data double words with any informational content There is no data-specific processing and change control for these typicals. The user program is responsible for this. As an option change control can be activated that triggers a transfer with each bit change.

TD7 programs for master stations and node stations

In principle, the TD7 programs of master stations and node stations look the same as for a station. The corresponding receive typicals are used in a central station for the send typicals of a station, the corresponding send typicals are used in a central station for the receive typicals of the station.

In a node station, both send typicals and the corresponding receive typicals are used according to the transmission direction.

In the master station it is practical to structure the OB1 program according to stations, in other words, all data typicals belonging to the same station are called in one FC. The best overview is provided when the number of the FC is identical to the subscriber number of the station.

1.5.2 Cyclic interrupt OB

Introduction

Only include time-controlled TD7 blocks in a CPU if fast count pulses must be detected in this CPU which could not be reliably detected within an OB1 cycle due to an excessively long cycle time.

The count pulses are acquired via any digital input module. To acquire the pulses reliably, the digital inputs use must be queried for change at a fixed time interval. The time interval is based on the duration of the shortest count pulse. The minimum count pulse duration may be 50 ms. The same applies to the duration of the pause. This results in a maximum count frequency of 10 Hz.

The time interval in which the count pulse acquisition is performed must be approximately half the count pulse duration, i.e. at 50 ms at an interval of approximately 25 ms.

For this time-controlled count pulse acquisition, OB35 needs to be configured for an S7-300 CPU, one of the available cyclic interrupt OBs OB30 to OB38 for an S7-400 CPU, and a cyclic interrupt OB with a number from 30..38 for an S7-1500 CPU.

All cyclic interrupt OBs have a preset time interval, for OB35 for example 100 ms. This can be changed in increments of 1 ms. This makes it possible to set a cyclic interrupt OB, for example, to 25 ms.

You change the time interval for a cyclic interrupt OB in the Properties dialog (S7-400/1500), or in the Properties dialog of the CPU with S7-300.

In highly time-critical applications, count pulses can also be recorded in a process alarm OB of an alarm module, e.g. in OB40.

Table 1- 1 Calling FC PulseCounter in the alarm OB

OB	
PulseCounter	One or more FC PulseCounters can be integrated in an alarm OB for the acquisition of count pulses if several count pulses can occur during an OB1 cycle and are to be acquired. FC PulseCounter processes up to 8 pulse inputs of any digital input. The acquired count pulses are added together in programmable SIMATIC counters. These access the function blocks that put together the count value frames (FB-Cnt0x_S).

1.5.3 Programming error OB

Validity

Only for S7-300/400

Function

When a block that does not exist is called in an S7-300/400 CPU, the CPU normally changes to STOP. The diagnostics buffer indicates which FB, FC or DB was missing. You can then reload the missing block and restart the CPU.

If, however, you do not want the CPU to change to STOP if there is a missing block or only changes to STOP when certain block types or block numbers are missing, you can specify the reaction you require in OB121.

Even if you have loaded OB121 as an empty block on the CPU, this is enough to have the CPU continue running if a block is missing. If you want to decide more selectively when the CPU should continue running or change to STOP, include OB121 in the user program.

In conjunction with SINAUT ST7 it is possible that a CPU changes to STOP when it receives data from another CPU that it does not (or not yet) know. This is, for example, the case when you add a data point typical to a station and give it a complete destination address (destination subscriber no. plus destination object no.). The specified destination object no. can lead to a stop on the destination subscriber in the following situation:

- As soon as you have installed a new data point typical in a station, the data is transferred to the destination.
- If there is no corresponding receive typical installed on the destination CPU, the destination object no. (= instance DB of the received typical) does not exist.

The result is that the CPU changes to STOP as soon as the data is received.

To avoid this it is advisable to call FC ST7ObjectTest in OB121.

Table 1- 2 Call of FC ST7ObjectTest in the programming error OB

OB121	
ST7ObjectTest	Calling FC ST7ObjectTest in OB121 prevents a CPU STOP, if the CPU receives data with an unknown destination object no. Other calls can be included at any point in OB121 regardless of the FC ST7ObjectTest call. FC ST7ObjectTest has a parameter "StopInOtherCases". Here you can specify what happens in other situations (STOP or continue running) if OB121 was called because another data block or an FB or FC is missing.

Types

2

2.1 PLC data types (UDT)

UDTs used

The following PLC data type (UDT) is used by the block library:

- TransmitBlock

  UDT for the data transfer between the CPU and TIM

  When using the "Dat256D_S / Dat256D_R" typical, copy the UDT into the "PLC data types" directory of the respective CPU. The UDT is not automatically referenced by the typical.

  The UDT does not require any parameter assignment.

2.2 System data types (SDT)

SDTs used

The following system data types (SDTs) are used by the DB BasicData and stored automatically in the directory "PLC data types" of the CPU when generating/compiling the TD7 blocks:

- ConnectionDescription

  SDT is part of the "BasicData" DB and has an "Array[1..1] of ConnectionDescription" data type. It contains the following details for the connection description:

  – Number of the configuration DB

  – Connection type

  – CFB number in the case of an S7 connection

  – Subscriber number of the local TIM.

- SubscriberObject

  SDT is part of the "BasicData" DB and has an "Array[1..1] of SubscriberObject" data type. It contains the following information about the communication partner:

  – Subscriber number of the partner

  – Array index of the connection description

  – Subscriber type

  – Information whether the partner is local or remote.

Master copies

3

Validity

The following program blocks are located in the master copies of the global library:

- **CPU300 / CPU400**

  Program blocks for the S7-300 and S7-400 CPU

- **CPU1500**

  Program blocks for the S7-1500 CPU

The blocks are largely identical. Unless otherwise stated, the description applies equally to the blocks of the S7-300/400 and S7-1500.

If individual blocks cannot be used for all SIMATIC product series mentioned or if functions for the SIMATIC product series differ, this is expressly indicated.

3.1 Data point typicals

3.1.1 Types and overview of the data point typicals

Types of the data point typicals

Data point typicals process one or more data points of the same information type, e.g. 4 bytes of binary information, 4 analog values or 1 byte commands, etc.

When a data point typical (FB) is called the corresponding instance DB needs to be specified in which the data will be written or from which the data to be transferred will be read.

Transfer direction of the typicals

A data point typical for a specific type or amount of information always come in two versions:

- A typical for acquiring and sending
- A typical for receiving and outputting

When using data point typicals a distinction is therefore made according to the transfer direction:

- **Sending typicals**

  Sending typicals process data and send it to the remote partner.

  They have the ending "_S" Examples: Bin04B_S, Ana04W_S

- **Receiving typicals**

  Receiving typicals receive the data from their remote partner.

  They have the ending "_R" Examples: Bin04B_R, Ana04W_R

When transferring data on the two communicating partners there is always a corresponding pair of typicals involved.

Examples:

- Monitoring direction
  - The station sends binary data with the typical Bin04B_S.
  - The central station receives the data with the typical Bin04B_R.
- Control direction
  - The central station sends a command with the typical Cmd01B_S.
  - The station receives the command with the typical Cmd01B_R.

Structure of typical names

Note

No SMS blocks

Unlike the block library for STEP 7 V5, there are no blocks for sending SMS in the global library "Telecontrol ST7".

You configure the sending of e-mails and/or SMS in STEP 7 Professional in the TIA Portal via the data point and message editor.

The names of the data point typicals are assigned according to the following scheme:

Table 3- 1 Structure of typical names

Data point type	Number of values	Data format	Under-score	Function
Bin = Binary information	Maximum number of values sent per data frame	**X** = bit **B** = byte **W** = word **D** = double word **R** = real		**S** Send **R** Receive **FS** Sending without 1-out-of-n check **FR** Receiving without 1-out-of-n check
Ana = Analog value				
Cnt = Count value				
Cmd = Command				
Set = Setpoint, parameter				
Par = Parameter				
Dat = Data (any mixture of information types)				

Overview of the data point typicals

The following table provides an overview of the data point typicals.

Table 3- 2 Overview of the data point typicals

Data format	Symbolic block name	Function
Binary value typicals		
Byte	Bin04B_S	Send 4 byte binary values
Byte	Bin04B_R	Receive 4 byte binary values
Analog value typicals		
Int	Ana04W_S	Send 4 analog values
Int	Ana04W_R	Receive 4 analog values
Real	Ana04R_S	Send 4 analog values as floating point number (CPU 1500 only)
Real	Ana04R_R	Receive 4 analog values as floating point number (CPU 1500 only)
Counted value typicals		
UInt, Word *	Cnt01D_S	Send 1 counted value
UInt, Word *	Cnt01D_R	Receive 1 counted value
UInt, Word *	Cnt04D_S	Send 4 counted values
UInt, Word *	Cnt04D_R	Receive 4 counted values
Command typicals		
Byte, USInt *	Cmd01B_S	Send 1 byte commands (1-out-of-8)
Byte, USInt *	Cmd01B_FS	Send 1 byte commands without (1-out-of-n) control
Byte, USInt *	Cmd01B_R	Receive 1 byte commands (1-out-of-8)
Byte, USInt *	Cmd01B_FR	1 byte commands received without 1-out-of-8 control
Bit	Cmd08X_S	Send 8 commands for specific bits (only CPU 1500)
Bit	Cmd08X_R	Receive 8 commands for specific bits (only CPU 1500)
Setpoint and parameter typicals		
Word	Set01W_S	Send 1 setpoint and receive current local setpoint. Transmission is subject to 1-out-of-n control (via FC Safe).
Word	Set01W_FS	Send 1 setpoint (without 1-out-of-n control) and receive current local setpoint.
Word	Set01W_R	Receive 1 setpoint and send current local setpoint
ARRAY [1...12] of DInt / UDInt / DWord / Real *	Par12D_S	Send max. 12 double words with parameters and receive current local parameters Transmission is subject to 1-out-of-n control (via FC Safe).
ARRAY [1...12] of DInt / UDInt / DWord / Real *	Par12D_FS	Send max. 12 double words (without 1-out-of-n control) with parameters and receive current local parameters
ARRAY [1...12] of DInt / UDInt / DWord / Real *	Par12D_R	Receive max. 12 double words with parameters and send current local parameters

Data format	Symbolic block name	Function
Typicals for variable data types and amounts		
ARRAY [1...12] of DInt / UDInt / DWord / Real *	Dat12D_S	Send maximum of 12 double words with any data
ARRAY [1...12] of DInt / UDInt / DWord / Real *	Dat12D_R	Receive maximum of 12 double words with any data
Div. data types **	Dat12x1D_S	Send max. 12 channels with data (only CPU 1500)
Div. data types **	Dat12x1D_R	Receive max. 12 channels with data (only CPU 1500)
DWord	Dat256D_S	Send maximum of 256 double words with any data
DWord	Dat256D_R	Receive maximum of 256 double words with any data

***** USInt, UInt and UDInt are not supported by S7-300/400.

****** For the supported data types, refer to the block description.

Parameters not required

Parameters not required in data point typicals can be left open.

3.1.2 Reoccurring parameters

Description of the typical parameters

In the following description data point typicals that are identical except for the number of data points to be processed are described together.

In the tables of the data point typicals you will find the following information about the individual parameters:

- Parameter

  Name of the parameter
- Declaration

  Parameter type
  - INPUT

    Input parameter
  - OUTPUT

    Output parameter
  - IN_OUT

    In-out parameter
- Data type

  Data type supported for this parameter
- Range of values

- Default

  Preset value of the parameter

  If you do not configure individual parameters of a data point typical, the default value is used.

- Explanation

  Description of the function and the typical-specific properties of the parameter

Parameters used by many typicals

The following parameters are used by many of the typicals of the TD7onCPU library. These parameters are described once here and the description is not repeated in the following sections for the individual data point typicals.

Depending on the use of the typicals, some parameters are configured differently. Note the typical use described below.

PartnerNo

Declaration:	INPUT
Data type:	INT
Range of values:	0 / 1 ... 32000
Default:	0
Explanation:	Subscriber number of the partner The subscriber number of the partner with which the block communicates must be specified.. • In a station typical, this is normally the subscriber number of the central station or the application of the control station (e.g. ST7cc). • For a typical that is used in a central station, this is normally the subscriber number of a station.

Effects of the value 0 (zero) on various typical classes

- **Sending typicals (1)**

  (Bin04B_S, Ana04W_S, Cnt01D_S/Cnt04D_S, Dat12D_S, Dat256D_S)

  With the value 0 the data is sent to all subscribers to which an ST7 connection is configured. In this case, the parameter "PartnerObjectNo" is automatically sent with the value zero by the process typical.

  If PartnerNo is not found in the administration (DB BasicData), an entry to this effect is made in the diagnostics buffer (event ID B101). The CPU does not change to STOP. The block is then no longer processed, however, until the error has been corrected.

  Notes
  - If the "PartnerObjectNo" is missing, there must be a list on the partner CPU from which the missing object number can be recognized (see ListGenerator1500/300/400 FC (Page 136)).
  - Use of the block in a node station

    If the CPU of the node station maintains both connections to higher-level subscribers as well as to lower-level stations, a data frame with PartnerNo = 0 is transferred to all subscribers in the direction master station and in the direction stations.

- **Sending typicals (2)**

  (Cmd01B_S, Set01W_S, Par12D_S)

  The value 0 is not permitted!

  If "PartnerNo" is < 1 or > 32000, an error message is entered in the diagnostic buffer (event ID B100).

  If the configured value is permitted and correct but "PartnerNo" is not found in the administration (DB BasicData), an entry is also made in the diagnostics buffer (event ID B101). The CPU does not change to STOP. The block is then no longer processed, however, until the error has been corrected.

- **Receiving typicals (1)**

  (Bin04B_R, Ana04W_R, Cnt01D_R/Cnt04D_R, Dat12D_R, Dat256D_R)

  The value 0 is not permitted!

  If PartnerNo is < 1 or > 32000, an error message is entered in the diagnostic buffer (event ID B100).

  If the configured value is permitted and correct but "PartnerNo" is not found in the administration (DB BasicData), an entry is also made in the diagnostics buffer (event ID B101). The CPU does not change to STOP. The block is then no longer processed, however, until the error has been corrected.

  If the CPU receives a data frame for this typical, a check is made to establish whether the source subscriber number in the data frame is identical to the "PartnerNo" configured here. If the two subscriber numbers are different, the received information is discarded and an error message is entered in the diagnostic buffer (Event ID B130).

- **Receiving typicals (2)**

  (Cmd01B_R, Set01W_R, Par12D_R)

  Set the value 0 if the typical is to receive data from more than one partner, e.g. if it is to receive data from several control centers.

  If the CPU receives data for this typical and "PartnerNo" is > 0, a check is made to establish whether the source subscriber number in the data frame is identical to the "PartnerNo" configured here. If the two are different, the received information is discarded and an error message is entered in the diagnostic buffer (Event ID B130).

  This check is not made if "PartnerNo" is = 0. Regardless of the sender, each data frame addressed to the typical is passed on to the typical.

  If "PartnerNo" is > 0 and this number is not found in the administration (in DB-BasicData), an entry is made in the diagnostic buffer (event ID B101). The CPU does not change to STOP. The block is then no longer processed, however, until the error has been corrected.

Notes

- If "PartnerNo = 0" is set, make sure that each partner sends the data with a complete destination address (target subscriber no. and target object no.).
- Use of the block in a node station

  If the CPU of the node station maintains both connections to higher-level subscribers as well as to lower-level stations, a data frame with PartnerNo = 0 is transferred to all subscribers in the direction master station and in the direction stations.

PartnerObjectNo

Declaration:	INPUT
Data type:	INT
Range of values:	0 / 1 ... 32000
Default:	0
Explanation:	Object number of the partner The number of the object (= DB number) on the partner with which the block communicates.

Effects of the value 0 (zero) on various typical classes

- **Sending typicals (1)**

  (Bin04B_S, Ana04W_S, Cnt01D_S/Cnt04D_S, Dat12D_S, Dat256D_S)

  Setting the parameter to 0 makes sense if PartnerNo = 0 was set for the preceding parameter. If the"PartnerObjectNo" is missing, there must be a list on the partner CPU from which the missing object number can be recognized (see FC-ListGenerator).

  If the partner is an ST7cc control center specifying the "PartnerObjectNo" for this block can be omitted because in ST7cc there are no DBs as target objects. ST7cc decodes its data solely based on the source address in the data frame.

- **Sending typicals (2)**

  (Cmd01B_S, Set01W_S, Par12D_S)

  The value 0 is not permitted!

  If the parameter assignment is incorrect (< 1 or > 32000), an error message is entered in the diagnostic buffer (event ID B102). The CPU does not change to STOP. The block is then no longer processed, however, until the error has been corrected.

- **Receiving typicals (1)**

  (Bin04B_R, Ana04W_R, Cnt01D_R/Cnt04D_R, Dat12D_R, Dat256D_R)

  The value 0 is not permitted!

  If the parameter assignment is incorrect (< 1 or > 32000), an error message is entered in the diagnostic buffer (event ID B102). The CPU does not change to STOP. The block is then no longer processed, however, until the error has been corrected.

  If the CPU receives data for the object configured here, there is a check to establish whether the source object number in the data frame is identical to the "PartnerObjectNo" configured here. If they are different, the received information is discarded. An error message is entered in the diagnostic buffer (event ID B131).

- **Receiving typicals (2)**

  (Cmd01B_R, Set01W_R, Par12D_R)

  0 must be set in the following situations:
  – The partner is not an S7 CPU, i.e. there is no DB number as object. This is, for example, the case when the partner is an ST7cc control center.
  – There is more than one partner (PartnerNo = 0) from which the typical is to receive data. The corresponding objects of these partners will then generally have different numbers; in other words, no unique number can be specified here.

  If the CPU receives data for the object configured here and "PartnerObjectNo" is > 0, there is a check to establish whether the source object number in the data frame is identical to the "PartnerObjectNo" configured here. If they are different, the received information is discarded. An error message is entered in the diagnostic buffer (event ID B131).

  This check is not made if "PartnerObjectNo" is = 0. Regardless of the sender object, each data frame addressed to the object is also passed on to the receiving object.

Enabled

Declaration:	INPUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	TRUE	
Address range:	Input	I 0.0 ... I n.7
	Memory	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7

Explanation: Enable block processing

- If processing is enabled, all the functions of the block execute.
- The response is different if processing has not been enabled:

Processing not enabled

- **Sending typicals**

  (Bin04B_S, Ana04W_S, Cnt01D_S, Cnt04D_S, Dat12D_S, Dat256D_S)

  Also applies to the mirror-image value of: Set01W_R, Par12D_R

  If processing is not enabled, the block can only communicate at the organizational level; in other words, Org frames can be sent and received.

  Please note: A general query is answered, but the response frame contains the data valid at the time of the lock.

  Note
  The response described here does not apply to Cmd01B_R; see Command typical Cmd01B_R (Page 54).

- **Receiving typicals**

  (Bin04B_R, Ana04W_R, Cnt01D_R, Cnt04D_R, Dat12D_R, Dat256D_R)

  Also applies to the mirror-image value of: Set01W_S, Par12D_S

  If processing is not enabled, the block can only communicate at the organizational level; in other words, Org frames can be sent and received.

  Please note: A request can still be sent and the answer received, the received information is, however, not output to the outputs. You will find the relevant outputs in the description of the appropriate data point typicals.

  Note
  The response described here does not apply to Cmd01B_S; see Command typical Cmd01B_S (Page 51).

ImageMemory

Declaration:	INPUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	TRUE	
Address range:	Input	I 0.0 ... I n.7
	Memory	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7
Using in typicals	Bin04B_S, Ana04W_S, Cnt01D_S/Cnt04D_S, Set01W_R, Par12D_R, Dat12D_S	

Explanation: Image memory principle for spontaneous data transmission

- TRUE

  The data is transferred according to the image memory principle.

  The image memory principle reduces the memory required for storing data frames and produces as little data traffic on the WAN as possible. The default TRUE is the correct choice in most cases.
- FALSE

  The data is transferred according to the send buffer principle.

  The send buffer principle is only required with data points whose individual data changes will be saved and transferred to the partner, for example alarms with time stamps.

Conditional

Declaration:	INPUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	TRUE	
Address range:	Memory	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7
Using in typicals	Bin04B_S, Ana04W_S, Cnt01D_S/Cnt04D_S, Set01W_R, Par12D_R, Dat12D_S	
Explanation:	Conditional spontaneous data transfer You will information on this below with the parameter "Unconditional".	

Unconditional

Declaration:	INPUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	FALSE	
Address range:	Memory	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7
Using in typicals	Bin04B_S, Ana04W_S, Cnt01D_S/Cnt04D_S, Set01W_R, Par12D_R, Dat12D_S, Dat256D_S	

Explanation: Unconditional spontaneous data transfer

With the two parameters "Conditional" and "Unconditional" you decide whether a data frame is transferred immediately by the module if the value changes (unconditional spontaneous) or at a later point in time (conditional spontaneous).

The two parameters are configured as follows:

- **Conditional spontaneous transfer (not necessarily immediately)**
  - Conditional = TRUE
  - Unconditional = FALSE
- **Unconditional spontaneous transfer (immediately)**
  - Conditional = FALSE
  - Unconditional = TRUE

The default of the two parameters was chosen so that a data frame is not transmitted immediately.

The decision whether a data frame is transferred immediately or later only relates to dial-up networks.

In a dial-up network you must decide from case to case whether a change to the value of the data point requires immediate transfer and therefore the immediate establishment of a connection. This can, for example, be required for data points with alarms.

On a dedicated line, the transmission is always immediate even if the combination of "Conditional" and "Unconditional" is set to "not immediately". On dedicated lines, you do not need to make changes to the two parameter settings.

Permanent

Note

Parameter irrelevant

The parameter (INPUT, Bool) only appears in blocks for S7-300/400.

It is no longer supported functionally with S7-300/400, the value is always = FALSE.

TimeStamp

Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Default:	FALSE
Using in typicals	Bin04B_S, Ana04W_S, Cnt01D_S/Cnt04D_S, Dat12D_S, Dat256D_S, Set01W_R, Par12D_R

Explanation: Time stamp

- TRUE

  The data frame will be transferred with a time stamp.

  The prerequisite is that the time provided by the local TIM is available in the CPU. For more detailed information, refer to the description of FC TimeTask.

  Note
  With Ana04W_S, remember the dependency of the time stamp on the parameter "MeanValueGeneration"; see Analog value typical Ana04W_S (Page 40).
- FALSE

  The data frame is transferred without a time stamp.

For information on the format of the time stamp, refer to the section Time stamp (Page 34).

Note

Re-initialization

This parameter requires re-initialization of the CPU.

Recommendation:

Do not change the parameter in runtime or after the restart.

NewData

Declaration:	OUTPUT	
Data type:	BOOL	
	DWORD with 1 status per segment for Dat256D_R and Dat12x1D_R	
Range of values:	TRUE / FALSE	
Default:	FALSE	
Address range:	Output	Q 0.0 ... Q n.7
	Memory	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7
Using in typicals	Bin04B_R, Ana04W_R, Cnt01D_R/Cnt04D_R, Cmd01B_R, Set01W_S, Set01W_R, Par12D_S, Par12D_R, Dat12D_R, Dat12x1D_R, Dat256D_R	

Explanation: Receive new data

The NewData output is intended for user-specific further processing, for example to react in a specific way to receipt of new data.

Whenever the block has received new data and has output it to the outputs for the specific typicals, the parameter is set to TRUE for one OB1 cycle.

You will find the specific outputs in the description of the individual data point typicals.

With the data point typicals Set01W_R and Par12D_R, "NewData" is also set to TRUE for one OB1 cycle if there is a new local value entered in the Local = 1 state.

If you do not require the parameter, simply leave it open.

3.1.3 Time stamp

Format of the SINAUT time stamp

For many data point typicals you can use the TimeStamp parameter to instruct that the data of the object should be transferred with a time stamp.

However for the receiving data point typicals there is no output parameter with which to output the received time stamp. The time stamp is only saved in the instance DB which you have specified when calling the respective receive typical. To further process the time stamp, the data must be read out of the DB by the user program.

The time stamp is saved in two data double words that have the same name in all object DBs:

Name of the double word	Contents
RecTimeStamp_1	Year, month, day and hour
RecTimeStamp_2	Minute, second, millisecond and time status

With the exception of the half byte with the time status, the date and time are coded in BCD format.

Table 3- 3 Assignment of the structure of the time stamp

Name of the double word	Byte no.	Contents	
		High nibble	Low nibble
RecTimeStamp_1	0	Year * 10	Year * 1
	1	Month * 10	Month * 1
	2	Day * 10	Day * 1
	3	Hour * 10	Hour * 1
RecTimeStamp_2	0	Minute * 10	Minute * 1
	1	Second * 10	Second * 1
	2	Millisecond * 100	Millisecond * 10
	3	Millisecond * 1	Time status

Table 3- 4 The bit assignment of the half byte "time status" (low nibble of byte 3 of Rec-TimeStamp_2)

Bit no.	Value	Meaning
0	0	Time is invalid
	1	Time is valid
1	0	Standard time
	1	Daylight saving time
2		Not used
3		Not used

The time double words occupy different addresses depending on the typical. Look in the instance DB or in the declaration header of the FB to find the absolute address of both double words.

It is more convenient to give the instance DBs symbolic names. You can then use the symbolic addresses to read out the information. In this case, you do not need to worry about the actual absolute addresses. These are used automatically by STEP 7. The following example clarifies this procedure.

Example

Symbolic name of instance DB: `ObjectDB27`

The STEP 7 program for reading the date and time of day and for saving in DB20 beginning with data byte 100 may appear as follows programmed in STL:

```
L "ObjectDB27".RecTimeStamp_1

T DB 20.DBD 100

L "ObjectDB27".RecTimeStamp_2

T DB 20.DBD 104
```

3.1.4 Analog value typical Ana04R_S

Validity

S7-1500

Function

Send 4 analog values as 32-bit floating-point number

Ana04R_S transfers the 4 analog values as instantaneous values. At the time of the transfer, the currently pending analog value is acquired and transferred to the partner.

Note

Common processing of the four analog values

The processing parameters such as threshold, smoothing factor etc. exist only once in a typical. These parameters apply to all 4 analog values in common; in other words, it is not possible to set the parameters for the individual analog values. For this reason, each typical should only acquire analog values that should be processed in the same way.

Parameters

General parameters: **PartnerNo**
PartnerObjectNo
Enabled
ImageMemory
Conditional
Unconditional
TimeStamp

For a description, see section Reoccurring parameters (Page 24).

Name:	**ThresholdIntegration**
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Default:	FALSE
Explanation:	Threshold processing according to the integration principle With this parameter, you can specify whether the integration principle is used in threshold value processing. With the default FALSE the threshold value is calculated without integration. In this case, you can expect less data traffic on the telecontrol line and locally between CPU and TIM.

Name:	**ZeroLimitation**
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Default:	TRUE
Explanation:	Zero limitation If this parameter is activated negative values are be suppressed and replaced with the value 0.

Name:	**TriggerInput**	
Declaration:	INPUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	FALSE	
Address	Input	I 0.0 ... I n.7

range:	Memory	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7

Explanation: Trigger input

With the edge change 0 → 1 of the "TriggerInput" input the transfer of the analog value frame can be triggered at a required time.

Example:
Time-driven analog value transfer with time stamp for supplying an analog value archive in the control center.

Make sure that you set the "ImageMemory" parameter to FALSE to prevent these data frames with time stamps from being overwritten when saving on the station TIM.

The FC Trigger block can be used for time-driven triggering of a transmission over "TriggerInput".

If you do not require the parameter, simply leave it open. The transfer should then be triggered based on the "ThresholdValue" and "ThresholdIntegration" threshold parameters.

"TriggerInput" actually only triggers transmission indirectly. With a 0 → 1 edge change at "TriggerInput", the data frame is put together with its current values and transferred to the local TIM. The TIM is responsible for the actual transmission to the partner. With dedicated lines or wireless networks the transfer is immediate. With a dial-up connection, it is possible that the data frame is saved first on the TIM and sent at a later point in time. The reason can, for example, be that the data frame is marked as "Conditional spontaneous", see parameter "Conditional".

Name:	**AnalogInput_1 ... _4**	
Declaration:	INPUT	
Data type:	REAL	
Range of values:	See address range	
Default:	0.0	
Address range:	Bit memory address	MD0 ... MDn
	Data address	LD0 ... LDn DBm.DBD0 ... DBDn

Explanation: Analog input word

For each analog value to be transmitted in the data frame, you can specify from where the FB will take the analog information. Any combination of a data block and the bit memory address area can be used.

If you do not require parameters, simply leave them open. The value 0 is transferred for these analog inputs in the data frame.

Name:	**SamplingPeriod**
Declaration:	INPUT
Data type:	INT
Range of values:	0 ... 32767 [ms]
Default:	500
Explanation:	Acquisition interval for analog inputs in milliseconds The acquisition interval is required for the following parameters: • Threshold formation according to the integration principle (ThresholdIntegration) • Smoothing the analog input value (SmoothingFactor) The value must be selected high enough so that it is certain that a new value was acquired within the encryption time of the analog input. The interval to be specified has to be at least as long as the encoding time of the analog input module being used at the selected resolution (8 ... 15 bits). The value must also be selected generously so that analog values are acquired even with the highest resolution and with analog modules with the highest number of inputs. Specifying a "SamplingPeriod" that is too short may lead to an overflow of the internal accumulation counter.

Name:	**ThresholdValue**
Declaration:	INPUT
Data type:	REAL
Range of values:	+1.175495e-38 ... +3.402823e+38
Default:	1.0
Explanation:	Threshold value Without configuration, the default value 1.0 is used. Point to note with "ThresholdValue" = 0: Changes are not checked based on the threshold value. The analog value frame will only be sent in the following situations: • When there is a trigger via the "TriggerInput" input, typically a time-driven or event-driven trigger. • When there is a general request to the station or a single request for the data frame.

Name:	**SmoothingFactor**
Declaration:	INPUT
Data type:	REAL

Range of values:	1.0 (no smoothing) 4.0 (weak smoothing) 32.0 (medium smoothing) 64.0 (strong smoothing)
Default:	1.0
Explanation:	Smoothing factor Using the smoothing factor, quickly fluctuating analog values can be smoothed to a greater or lesser extent depending on the factor. This may allow a narrower threshold band to be selected (ThresholdValue). The smoothing factors are identical to the smoothing factors that are configured for some S7 analog input modules. The smoothing is calculated using the same formula as on the input module:

$$y_n = \frac{x_n + (k - 1)\ y_{n-1}}{k}$$

y_n = smoothed value in the current cycle n

y_{n-1} = smoothed value in previous cycle n-1

x_n = acquired value in the current cycle n

k = smoothing factor

3.1.5 Analog value typical Ana04R_R

Validity

S7-1500

Function

Receive 4 analog values as 32-bit floating-point number

Parameters

General parameters:	**PartnerNo** **PartnerObjectNo** **Enabled** For a description, see section Reoccurring parameters (Page 24).
Name:	**AnalogOutput_1 ... _4**
Declaration:	OUTPUT
Data type	REAL

Range of values:	See address range	
Default:	0.0	
Address range:	Bit memory address	MD0 ... MDn
	Data address	LD0 ... LDn DBm.DBD0 ... DBDn
Explanation:	You can select where the individual analog values received by the FB are output. Addresses of a data block and in the bit memory address area can be mixed as required. If you do not require parameters, simply leave them open.	

Name:	**NewData** For the declaration, data type, range of values, default and function, refer to the section Reoccurring parameters (Page 24).
Typical-specific response:	Whenever the FB has received new data and has output it to the outputs "AnalogOutput_1" to "AnalogOutput_4", the "NewData" output is set to TRUE for one OB1 cycle.

3.1.6 Analog value typical Ana04W_S

Function

Send 4 analog values as 16 bit values

Ana04W_S transfers the 4 analog values alternatively:

- As instantaneous values

  At the time of the transfer, the currently pending analog value is acquired and transferred to the partner.

- As mean values

  The pending analog value is accumulated at selectable intervals. At the time of the transmission, a mean value is formed from the total value and transferred to the partner.

Note

Common processing of the four analog values

The processing parameters such as threshold, smoothing factor etc. exist only once in a typical. These parameters apply to all 4 analog values in common; in other words, it is not possible to set the parameters for the individual analog values. For this reason, each typical should only acquire analog values that should be processed in the same way.

Parameters

General parameters: **PartnerNo**
PartnerObjectNo
Enabled
ImageMemory
Conditional
Unconditional

For a description, see section Reoccurring parameters (Page 24).

Name: **TimeStamp**

For the declaration, data type, range of values, default and function, refer to the section Reoccurring parameters (Page 24).

Typical-specific functions: With the setting TRUE the time stamp depends on the setting of the parameter "MeanValueGeneration":

- MeanValueGeneration = FALSE

  Instantaneous values are transferred in the data frame.

  The time stamp in the data frame is identical to the time of acquisition of the instantaneous values contained in the data frame.
- MeanValueGeneration = TRUE

  The data frame contains mean values.

  The time stamp is identical to the time at which the mean value calculation period was completed.

  The start of the mean value calculation period is not included in the data frame. This is, however, identical to the time stamp of the previously transferred mean value frame.

Name: **ThresholdIntegration**
Declaration: INPUT
Data type: BOOL
Range of values: TRUE / FALSE
Default: FALSE
Explanation: Threshold processing according to the integration principle

With this parameter, you can specify whether the integration principle is used in threshold value processing.

With the default FALSE the threshold value is calculated without integration. In this case, you can expect less data traffic on the telecontrol line and locally between CPU and TIM.

When MeanValueGeneration = TRUE, (analog values are sent as mean values), the "ThresholdIntegration" parameter has no meaning.

Name:	**ZeroLimitation**	
Declaration:	INPUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	TRUE	
Explanation:	Zero limitation If this parameter is activated negative values are be suppressed and replaced with the value 0.	

Name:	**TriggerInput**	
Declaration:	INPUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	FALSE	
Address range:	Input	I 0.0 ... I n.7
	Memory	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7
Explanation:	Trigger input With the edge change 0 → 1 of the "TriggerInput" input the transfer of the analog value frame can be triggered at a required time. Example: Time-driven analog value transfer with time stamp for supplying an analog value archive in the control center. Make sure that you set the "ImageMemory" parameter to FALSE to prevent these data frames with time stamps from being overwritten when saving on the station TIM. If the block calculates mean values, the duration of the calculation period is decided by the "TriggerInput" input. The current period is ended and a new period begun each time a transmission is triggered by this input. The interval between triggering a data frame twice determines the duration of the mean value calculation period. The FC Trigger block can be used for time-driven triggering of a transmission over "TriggerInput". If you do not require the parameter, simply leave it open. The transfer should then be triggered based on the "ThresholdValue" and "ThresholdIntegration" threshold parameters.	

"TriggerInput" actually only triggers transmission indirectly. With a 0 → 1 edge change at "TriggerInput", the data frame is put together with its current values/mean values and transferred to the local TIM. The TIM is responsible for the actual transmission to the partner. With dedicated lines or wireless networks the transfer is immediate. With a dial-up connection, it is possible that the data frame is saved first on the TIM and sent at a later point in time. The reason can, for example, be that the data frame is marked as "Conditional spontaneous", see parameter "Conditional".

Name:	**MeanValueGeneration**
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Default:	FALSE
Explanation:	Mean value generation

If the parameter is enabled the analog values to be acquired are transferred as mean values.

If you select mean value generation, the currently pending analog value is acquired cyclically and accumulated. The acquisition cycle depends on the "SamplingPeriod" parameter (for example 500 ms, see also the description of this parameter). The mean value is calculated from the accumulated values as soon as a transmission is triggered via the "TriggerInput" input. Following this, the accumulation starts again so that the next mean value can be calculated.

The mean value can also be calculated if the transmission of the analog value frame is triggered by a general or single request. The duration of the mean value calculation period is then the time from the last transmission (for example triggered via TriggerInput) to the time of the general or single request. Once again, the accumulation restarts so that the next mean value can be calculated.

If the acquired analog value is above or below the permitted range (7FFFH bzw. 8000H), this value can either be taken into account immediately in the calculation of the mean value or it can be suppressed for a specific period for the calculation of the mean value. The required response can be decided with the "FaultSuppressionTime" parameter:

- **FaultSuppressionTime = 0**

  Acquisition of a value above or below the over- or underrange results in immediate cancellation of the mean value calculation. The value $7FFF_H$ or 8000_H is saved as an invalid mean value for the current mean value calculation period and sent when the next analog value frame is triggered. The calculation of a new mean value is then started. If the analog value remains in the overshoot or undershoot range, this new value is again saved immediately as an invalid mean value and sent when the next frame is triggered.

- **FaultSuppressionTime > 0**

  If the acquired analog value is in the overshoot or undershoot range, the bad values are excluded from the calculation of the mean value for a maximum duration as defined by the FaultSuppressionTime. If this time is exceeded, the value $7FFF_H$ or 8000_H is saved as an invalid mean value and sent when the next analog value frame is triggered. The procedure is identical in each new mean value calculation period averaging period; in other words, bad values are again suppressed for the duration of the "FaultSuppressionTime".

  The duration of the "FaultSuppressionTime" also indirectly decides the proportion of invalid values per mean value calculation period. For example, if the mean value is calculated every 15 minutes and "FaultSuppressionTime" is set to 5 minutes, the mean value is only sent as invalid when more than 1/3 of the analog values acquired are above or below the overshoot or undershoot range in the current mean value calculation period.

Name:	**AnalogInput_1 ... _4**	
Declaration:	INPUT	
Data type:	WORD	
Range of values:	See address range	
Default:	0 (W#16#0)	
Address range:	I/O words	PIW0 ... PIWn
	Memory words	MW0 ... MWn
	Data words	LW0 ... LWn DBm.DBW0 ... n
Explanation:	Analog input word For each analog value to be transmitted in the data frame, you can specify from where the FB will take the analog information. I/O words from analog input modules, data words from a data block and memory words can be mixed as required. If you do not require parameters, simply leave them open. The value 0 is transferred for these analog inputs in the data frame.	

Name:	**SamplingPeriod**
Declaration:	INPUT
Data type:	INT
Range of values:	0 ... 32767 [ms]
Default:	500

Explanation: Acquisition interval for analog inputs in milliseconds

The acquisition interval is required for the following parameters:

- Threshold formation according to the integration principle (Threshold Integration)
- Smoothing of the analog input value (Smoothing Factor)
- Mean value generation

The value must be selected high enough so that it is certain that a new value was acquired within the encryption time of the analog input. The interval to be specified has to be at least as long as the encoding time of the analog input module being used at the selected resolution (8 ... 15 bits).

The value must also be selected generously so that analog values are acquired even with the highest resolution and with analog modules with the highest number of inputs.

If mean values are calculated, SamplingPeriod should not be less than 500 ms. If mean values are calculated over very long periods, the time must be increased as follows:

- Mean value calculation period 12 h: SamplingPeriod = 1000 [ms]
- Mean value calculation period 24 h: SamplingPeriod = 2000 [ms]

Specifying a "SamplingPeriod" that is too short may lead to an overflow of the internal accumulation counter. The maximum value of 2 147 483 647 of a double integer must not be exceeded. When an overflow is detected, the invalid mean value of 8000_H is transferred for the current mean value calculation period.

Name: **ThresholdValue**

Declaration: INPUT

Data type: INT

Range of values: 0 / 1 ... 32767

Default: 270

Explanation: Threshold value

When specifying the threshold value, take the encryption range of the analog values into account. Raw values from S7 analog inputs are always encoded in the range from 0 ... 27648 (= 0 ... 100 %) or + 27648 (= + 100%). Depending on the resolution of the analog input, the value jumps from 128 (with 8-bit resolution) or 1 (with 15-bit resolution). If the acquired analog values have a different encoding range, specify a threshold value oriented toward this.

If the parameter is not configured, the default value of 270 is used. This corresponds to approximately 1% of the normal S7 analog raw value range.

Point to note with "ThresholdValue" = 0:

Changes are not checked based on the threshold value. The analog value frame will only be sent in the following situations:

- When there is a trigger via the "TriggerInput" input, typically a time-driven or event-driven trigger.
- When there is a general request to the station or a single request for the data frame.
- When the analog value moves into the overshoot or undershoot range ($7FFF_H$ or 8000_H) (possibly after the suppression time set for "FaultSuppressionTime" has elapsed).

When MeanValueGeneration = TRUE, i.e. the analog values are sent as mean values, the "ThresholdValue" parameter has no meaning.

Parameter:	**SmoothingFactor**
Declaration:	INPUT
Data type:	INT
Range of values:	1 (no smoothing) 4 (weak smoothing) 32 (medium smoothing) 64 (strong smoothing)
Default:	1
Explanation:	Smoothing factor When MeanValueGeneration = TRUE, i.e. the analog values are sent as mean values, the "ThresholdValue" parameter has no meaning. Using the smoothing factor, quickly fluctuating analog values can be smoothed to a greater or lesser extent depending on the factor. This may allow a narrower threshold band to be selected (ThresholdValue). The smoothing factors are identical to the smoothing factors that are configured for some S7 analog input modules. The smoothing is calculated using the same formula as on the input module: $y_n = \frac{x_n + (k-1)\ y_{n-1}}{k}$ y_n = smoothed value in the current cycle n y_{n-1} = smoothed value in previous cycle n-1 x_n = acquired value in the current cycle n k = smoothing factor

Name:	**FaultSuppressionTime**
Declaration:	INPUT
Data type:	INT
Range of values:	0 ... 32767

Default: 0

Explanation: Fault suppression time in seconds.

Transmission of an analog value located in the overshoot or undershoot range ($7FFF_H$ or 8000_H) is suppressed for the time period specified here. The value $7FFF_H$ or 8000_H is only sent after this time has elapsed, if it is still pending. If the value returns to below $7FFF_H$ or above 8000_H again before this time elapses, it is immediately sent again as normal. The suppression time is started again for the full duration the next time $7FFF_H$ or 8000_H is acquired.

This is typically used for temporary suppression of current values that may occur when powerful pumps and motors are started. The analog input may exceed the maximum range several times under some circumstances. Suppression prevents these values from being signaled as faults in the control center system.

The suppression is adjusted to analog values that are acquired by the S7 analog input modules as raw values. These modules return the specified values for the overflow or underflow range for all input ranges (also for life-zero inputs). With provided ready values, fault suppression is only possible if these also adopt the values $7FFF_H$ or 8000_H when there is overshoot or undershoot. If this is not the case, the parameter does not need to have a value entered.

The parameter can also be used in combination with the mean value calculation for temporary suppression of the values $7FFF_H$ or 8000_H (see parameter MeanValueGeneration).

If no parameter is specified, the default of 0 seconds applies. An acquired value of $7FFF_H$ or 8000_H is then sent immediately when it is first detected or, with mean value calculation, as an invalid mean value for the current mean value calculation period.

3.1.7 Analog value typical Ana04W_R

Function

Receive 4 analog values as 16-bit values

Parameters

General parameters: **PartnerNo**
PartnerObjectNo
Enabled
For a description, see section Reoccurring parameters (Page 24).

Name: **AnalogOutput_1 … _4**
Declaration: OUTPUT
Data type WORD

Default:	TRUE		
Explanation	0 (W#16#0)		
	Range of values:	I/O words	PQW0 ... PQWn
		Memory words	MW0 ... MWn LW0 ... LWn
		Data words	DBm.DBW0 ... n
	You can select where the individual analog values received by the FB are output. I/O words from analog output modules, data words from a data block and memory words can be mixed as required.		
	If you do not require parameters, simply leave them open.		

Name:	**NewData**
	For the declaration, data type, range of values, default and function, refer to the section Reoccurring parameters (Page 24).
Typical-specific response:	Whenever the FB has received new data and has output it to the outputs "AnalogOutput_1" to "AnalogOutput_4", the "NewData" output is set to TRUE for one OB1 cycle.

3.1.8 Binary value typical Bin04B_S

Function

Send 4 bytes of binary information

Parameters

General parameters:	**PartnerNo** **PartnerObjectNo** **Enabled** **ImageMemory** **Conditional** **Unconditional** **TimeStamp**
	For a description, see section Reoccurring parameters (Page 24).

Name:	**InputByte_1 … _4**
Declaration:	INPUT
Data type:	BYTE
Range of values:	See address range

Default:	0 (B#16#0)	
Address range:	Input bytes	IB0 ... IBn PIB0 ... PIBn
	Memory bytes	MB0 ... MBn LB0 ... LBn
	Data bytes	DBm.DBB0 ... n

Explanation: Input byte

Specify the memory area (1 to 4 bytes) of the binary information to be transferred by the FB.

Input bytes from the process input image, I/O bytes directly from digital input modules, data bytes from a data block and memory bytes can be mixed.

If you do not require parameters, simply leave them open.

For unconfigured bytes, the value 0 (zero) is transferred.

Name: **DisableMask**

Declaration: INPUT

Data type: DWORD

Range of values: 0 ... 2147483647

- As 32 bit binary number

  2#0 ... 2#11111111_11111111_11111111_11111111
- As 32 bit hexadecimal number

  DW#16#0 ... DW#16#FFFF_FFFF

Default: 0 (2#0)

Explanation: Disable mask

- For every input to be blocked enter a 1 at the relevant position in the bit pattern.
- For the other inputs enter a 0.

A disabled input always has the value 0 (zero) during the transfer.

For the assignment of the 32 inputs from "InputByte_1" to "InputByte_4" to the 32 bits of the blocking mask, see the following table.

	InputByte_1								InputByte_2								InputByte_3								InputByte_4							
Bit	.7	.6	.5	.4	.3	.2	.1	.0	.7	.6	.5	.4	.3	.2	.1	.0	.7	.6	.5	.4	.3	.2	.1	.0	.7	.6	.5	.4	.3	.2	.1	.0
2#	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
DW#16#	_				_				_				_				_				_				_				_			

Name:	**InversionMask**
Declaration:	INPUT
Data type:	DWORD
Range of values:	0 ... 2147483647 • As 32 bit binary number 2#0 ... 2#11111111_11111111_11111111_11111111 • As 32 bit hexadecimal number DW#16#0 ... DW#16#FFFF_FFFF
Default:	0 (2#0)
Explanation:	Inversion mask The inversion of input signals can, for example, be useful when using a mixture of sensors operating on the open and closed circuit principle. • For every input to be inverted enter a 1 at the relevant position in the bit pattern. • For the other inputs enter a 0. For the assignment of the 32 inputs from "InputByte_1" to "InputByte_4" to the 32 bits of the inversion mask, see the following table.

	InputByte_1								**InputByte_2**								**InputByte_3**								**InputByte_4**							
Bit	.7	.6	.5	.4	.3	.2	.1	.0	.7	.6	.5	.4	.3	.2	.1	.0	.7	.6	.5	.4	.3	.2	.1	.0	.7	.6	.5	.4	.3	.2	.1	.0
2#	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
DW#16#	_				_				_				_				_				_				_				_			

3.1.9 Binary value typical Bin04B_R

Function

Receive 4 bytes of binary information

Parameters

General parameters:	**PartnerNo** **PartnerObjectNo** **Enabled** For a description, see section Reoccurring parameters (Page 24).

Name:	**OutputByte_1 … _4**	
Declaration:	OUTPUT	
Data type:	BYTE	
Range of values:	See address range	
Default:	0 (B#16#0)	
Address range:	Output bytes	QB0 ... QBn PQB0 ... PQBn
	Memory bytes	MB0 ... MBn LB0 ... LBn
	Data bytes	DBm.DBB0 ... n
Explanation:	Output byte	
	Specify the memory area (1 to 4 bytes) of the binary information to be output in the binary information.	
	Output bytes from the process image output, I/O bytes directly to digital output modules, data bytes from a data block and memory bytes can be mixed.	
	If you do not require parameters, simply leave them open.	
	For information on reading out the time stamp received with the data using the user program, see the section Time stamp (Page 34).	

Name:	**NewData**
Explanation:	For the declaration, data type, range of values, default and function, refer to the section Reoccurring parameters (Page 24).
	The output "NewData" is always set to TRUE for one OB1 cycle when the FB has received new data and output it to the output bytes "OutputByte_1" to "OutputByte_4"

3.1.10 Command typical Cmd01B_S

Function

Send 1 byte commands with 1-out-of-8 check

The 1-out-of-8 check is performed by the data point typical.

The 1-out-of-n check is performed by FC Safe.

Note

FC Safe required

With Cmd01B_S, data can only be transmitted when FC Safe is linked in cyclic program.

Parameters

Parameter:	**PartnerNo** **PartnerObjectNo** For a description, see section Reoccurring parameters (Page 24).
Parameter:	**Enabled** For the declaration, data type, range of values, default and function, refer to the section Reoccurring parameters (Page 24).
Typical-specific response:	Enable block processing If processing is disabled the FB only checks to see if the disabled status has been canceled. The block cannot communicate at the organizational level in this status because it cannot send any Org. frames.

Parameter:	**CommandInputByte_HW**	
Declaration:	INPUT	
Data type:	BYTE	
Range of values:	See address range	
Default:	0 (B#16#0)	
Address range:	Input bytes	IB0 ... IBn PIB0 ... PIBn
	Memory bytes	MB0 ... MBn LB0 ... LBn
	Data bytes	DBm.DBB0 ... n
Explanation:	Command input byte for hardware input. This command input byte is specially designed for entering commands using hardware, i.e. via digital inputs. Input using memory or data bytes is also possible, but you must then make sure that the command pending at the input byte is reset, which occurs during hardware input when the command button is released. When input is detected, if no error is detected during the 1-out-of-8 and 1-out-of-n check, and if the central enable memory bit is set, the command is transferred. This is automatically set by FC Safe following a selected time delay set there (see FC Safe, "InputDelayTime" parameter). If a 1-out-of-8 or 1-out-of-n error is detected, the entered command is no longer processed. A new command is only read in again if previously for the time of one OB1 cycle no hardware command was detected in the CPU at this or another command input block with a hardware input. The FB enters a detected 1- out-of-8- or 1-out-of-n error in the diagnostic buffer (event ID B171 or B172). The error status is also indicated via the "InputError" output of FC Safe (see FC Safe, "InputError" parameter) and continues to be indicated as long as the error remains.	

Name:	**CommandInputByte_SW**	
Declaration:	IN_OUT	
Data type	BYTE	
Range of values:	See address range	
Default:	0 (B#16#0)	
Address range:	Memory bytes	MB0 ... MBn
	Data bytes	DBm.DBB0 ... n
	This is an in/out parameter (declaration IN_OUT). It is difficult to specify local bit memory with this parameter type and this should not be used.	
Explanation:	Command input byte for software input This command input byte is specially designed for entering commands using software, i.e. by the user program or an operator panel (OP). When an input is detected and if no error is detected during the 1-out-of-8 and 1-out-of-n check, the command id reset at the input byte and transferred. The central enable memory bit is ignored here because it is only intended for command input over hardware (see "CommandInputByte_HW"). If a 1-out-of-8 or 1-out-of-n error is detected, the entered command is no longer processed. A new command is only read in again if previously for the time of one OB1 cycle no software command was detected in the CPU at this or another command input block with a software input. The FB enters a detected 1- out-of-8- or 1-out-of-n error in the diagnostic buffer (event ID B171 or B172). Appropriate error bits are also set in the central data block BasicData where they can be queried by the software. For further details, refer to the description of FC Safe. In principle it is possible to create a new command at "CommandInputByte_SW" in every OB1 cycle. However, only one command per OB1 cycle is allowed and this applies to all command input blocks with software input (1-out-of-n check). An 'empty cycle' between two consecutive software commands is therefore not necessary.	

Note

CommandInputByte_HW / CommandInputByte_SW

If individual commands need to be entered over the hardware and over the software, the two command inputs "CommandInputByte_HW" and "CommandInputByte_SW" can also be used at the same time.

If a command is entered at the same time over both input bytes, this is only accepted when the same command is entered over the hardware as well as the software input. The hardware input is then processed further.

In all other cases the input is rejected and an error message is entered in the diagnostic buffer (Event ID B170). The error status is also indicated via the InputError output of FC Safe. Appropriate error bits are set in the central data block BasicData where they can be queried by the software (see FC Safe).

3.1.11 Command typical Cmd01B_R

Function

Receive 1 byte commands (1-out-of-8 format)

The 1-out-of-n check is performed by FC Safe.

Note

FC Safe required

With Cmd01B_R, data can only be received if the FC Safe is integrated at the end of the cyclic program.

Parameters

General parameters: **PartnerNo**
PartnerObjectNo
For a description, see section Reoccurring parameters (Page 24).

Name: **Enabled**
For the declaration, data type, range of values, default and function, refer to the section Reoccurring parameters (Page 24).

Typical-specific response: Enable block processing
If processing is disabled the FB only checks to see if the disabled status has been canceled. Any commands that are still received are not output. The FB cannot communicate at the organizational level in this status because Cmd01B_R cannot send or receive Org. frames.
If the "Enabled" input is operable by a switch, this local disable means received commands are no longer output. Since the block does not send Org. frames, it cannot report this local disable back to the partner itself. This must be done with another typical, e.g. Bin04B_S.

Name: **MultipleOutput**
Declaration: INPUT
Data type: BOOL
Range of values: TRUE / FALSE
Default: FALSE

Explanation: Simultaneous output of multiple commands permitted

With this parameter, you can specify whether or not several (consecutively received) commands can be output simultaneously; in other words, you specify how the block reacts when a new command is received and the previously received command still needs to be output.

Requirement: The command output time has not yet elapsed and the user program has not yet reset this command.

- FALSE

  Multiple output is not permitted. The newly received command overwrites the output byte. Any command still pending is therefore reset to 0 unless the new command is identical to the old one.
- TRUE

  Multiple output is permitted. A newly received command is written to the current output byte. The command output time is restarted and applies to all pending commands.

Name: **CommandOutputTime**

Declaration: INPUT

Data type: INT

Range of values: 0 ... 500

Default: 500

Explanation: Command output time for command outputs in milliseconds

The specified time applies to all command outputs.

If more than one output can be set at the same time (MultipleOutput = TRUE), the output time is restarted with each newly received command. This means retriggering for already pending commands. All the command outputs are reset at the same time onöy when the output time elapses.

With the value 0 a set command output is not reset by the command typical. You need to do this via the user program.

Name: **NewData**

Explanation: For the declaration, data type, range of values, default and function, refer to the section Reoccurring parameters (Page 24).

The output "NewData" is always set to TRUE for one OB1 cycle when the FB has received new data and output it to the output byte "CommandOuputByte".

Name: **CommandOutputByte**

Declaration: IN_OUT

Data type: BYTE

Range of values: See address range

Default:	0 (B#16#0)	
Address range:	(process image) output bytes	QB0 ... QBn
	Memory bytes	MB0 ... MBn
	Data bytes	DBm.DBB0 ... n
	Since the parameter is an IN_OUT parameter, direct I/O output of the command byte to PQB0 ... PQBn is not permitted. It is also difficult to specify local bit memory with this parameter type and this should not be used.	
Explanation:	Command output byte	
	To allow the command outputs to be reset both by the command typical itself as well as by the user program (when output time = 0), the parameter was declared as an IN_OUT parameter.	

3.1.12 Command typical Cmd01B_FS

Validity

S7-1500 only

Function

Send 1 byte commands without 1-out-of-8 test This means that it is not necessary to call the FC Safe in the cyclic program.

There is less transmission security due to the lack of a 1-out-of-8 check.

Parameters

Parameter:	**PartnerNo** **PartnerObjectNo** For a description, see section Reoccurring parameters (Page 24).
Parameter:	**Enabled** For the declaration, data type, range of values, default and function, refer to the section Reoccurring parameters (Page 24).
Typical-specific response:	Enable block processing If processing is disabled the FB only checks to see if the disabled status has been canceled. The block cannot communicate at the organizational level in this status because it cannot send any org. frames.
Parameter:	**HWmode**
Declaration:	INPUT
Data type:	BOOL

Range of values:	TRUE / FALSE
Default:	FALSE
Explanation:	The parameter specifies whether the block works in hardware mode (TRUE) or software mode (FALSE). Depending on the selected mode, the CommandInputByte is available for hardware input or for software input. In hardware mode, sending of commands is edge-controlled (edge transition from 0 to 1), the commands are not automatically reset. If a command is recognized, it is only included in the frame to be sent and sent when it has been continuously pending for the configured delay time (InputDelayTime). A possible delayed second command, for which the input delay time has not yet been reached, is ignored, but is only sent when its input delay time has also expired. At the same time is entered in the send buffer, the „InputOK" is set for 1 cycle. The same command can only be sent again if it has been reset to "0" beforehand. The „InputDelayTime", „MaxInputTime" and „InputError" parameters apply to all 8 commands of the command byte together and are only relevant in hardware mode.

Name:	**InputDelayTime**
Declaration:	INPUT
Data type:	INT
Range of values:	• 0 or open When the parameter is not required • 1 ... 32000 Value range for delay time, only relevant in hardware mode.
Explanation:	Delay time in ms for commands entered via hardware. A delay time of at least 1000 ms is recommended. The parameter applies to all 8 commands of the command byte. The parameter is irrelevant if the parameter HWmode = FALSE is set.

Name:	**MaxInputTime**
Declaration:	INPUT
Data type:	INT
Range of values:	• 0 When the parameter is not required • 1 ... 32000 Value range for monitoring time, only relevant in HW mode.

Explanation:	Monitoring time in s for commands entered via hardware. A monitoring time of at least 30 s is recommended. The parameter applies to all 8 commands of the command byte. The parameter is irrelevant if the parameter HWmode = FALSE is set.	

Name:	**InputOK**	
Declaration:	OUTPUT	
Data type:	BOOL	
Range of values:	Output	Q 0.0 ... Q n.7
	Memory	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7
Explanation:	The "InputOK" display is also set for 1 cycle with the entry in the send buffer. The display is independent of the "HWmode" parameter.	

Name:	**InputError**	
Declaration:	OUTPUT	
Data type:	BOOL	
Range of values:	Output	Q 0.0 ... Q n.7
	Memory	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7
Explanation:	Display of input errors for hardware commands The parameter applies to all 8 commands of the command byte. The output is set to 1 if at least one command is pending longer than the value configured under "MaxInputTime". The output is only reset to 0 if all 8 possible commands are free of errors. If a timeout occurs, you have to reset the respective command manually.	

Name:	**CommandInputByte**	
Declaration:	IN_OUT	
Data type	BYTE	
Range of values:	See address range	
Address range:	Memory bytes	MB0 ... MBn
	Data bytes	DBm.DBB0 ... n
	Address range for hardware inputs:	
	Input bytes	IB0 ... IBn
	This is an in/out parameter (declaration IN_OUT). It is difficult to specify local bit memory with this parameter type and this should not be used.	

Explanation **Command input byte for hardware input**

When the HWmode parameter has the value TRUE, this command input byte is specially designed for entering commands using hardware, i.e. via digital inputs.

The command is only entered in the send buffer if the partner can be reached.

You have to reset the command pending at the input byte yourself.

Command input byte for software input

When the HWmode parameter has the value FALSE, this command input byte is specially designed for entering commands using software, i.e. by the user program or an operator panel (OP).

The command is only entered in the send buffer if the partner can be reached.

Commands that have been entered are automatically reset.

3.1.13 Command typical Cmd01B_FR

Validity

S7-1500 only

Function

1 byte commands received without 1-out-of-8 check. There is less transmission security due to the lack of a 1-out-of-8 check.

Parameters

Parameter: **PartnerNo**

PartnerObjectNo

For a description, see section Reoccurring parameters (Page 24).

Parameter: **Enabled**

For the declaration, data type, range of values, default and function, refer to the section Reoccurring parameters (Page 24).

Typical-specific response: Enable block processing

If processing is disabled the FB only checks to see if the disabled status has been canceled. Any commands that are still received are not output.

If the "Enabled" input is operated via a switch, this local disable has the effect that received commands are no longer output. Since the block is, however, not capable of sending org. frames, it cannot report this local block back to the partner itself. This must be done with another typical, e.g. Bin04B_S.

Parameter:	**CommandOutputTime**
Declaration:	INPUT
Data type:	INT
Range of values:	0 ... 500
Default:	500
Explanation:	Command output time for command outputs in milliseconds The specified time applies to all command outputs. The output time is restarted with each newly received command. This means retriggering for already pending commands. All the command outputs are reset at the same time onöy when the output time elapses. With the value 0 a set command output is not reset by the command typical. You need to do this via the user program.

Parameter:	**NewData**
Explanation:	For the declaration, data type, range of values, default and function, refer to the section Reoccurring parameters (Page 24). The output "NewData" is always set to TRUE for one OB1 cycle when the FB has received new data and output it to the output byte "CommandOuputByte".

Parameter:	**CommandOutputByte**	
Declaration:	IN_OUT	
Data type:	BYTE	
Range of values:	See address range	
Default:	0 (B#16#0)	
Address range:	(process image) output bytes	QB0 ... QBn
	Memory bytes	MB0 ... MBn
	Data bytes	DBm.DBB0 ... n
	Since the parameter is an IN_OUT parameter, direct I/O output of the command byte to PQB0 ... PQBn is not permitted. It is also difficult to specify local bit memory with this parameter type and this should not be used.	
Explanation:	Command output byte To allow the command outputs to be reset both by the command typical itself as well as by the user program (when output time = 0), the parameter was declared as an IN_OUT parameter.	

3.1.14 Command typical Cmd08X_S

Validity

S7-1500

Function

Sending 8-byte commands over 8 channels for specific bits

No 1-out-of-n test is performed because the FC Safe is not required by the typical.

Parameters

General parameters:

PartnerNo

PartnerObjectNo

For a description, see section Reoccurring parameters (Page 24).

Name: **Enabled**

For the declaration, data type, range of values, default and function, refer to the section Reoccurring parameters (Page 24).

Typical-specific response: Enable block processing

If processing is disabled the FB only checks to see if the processing lock has been canceled. The block cannot communicate at the organizational level in this status because it cannot send any Org. frames.

Name: **CommandInput01..08**

Declaration: IN_OUT

Data type: Bool

Range of values: See address range

Default: 0

Address range:

Input	I 0.0 ... I n.7
Memory	M 0.0 ... M n.7
Data bit	DBm.DBX 0.0 ... n.7

Since the parameter is an in/out parameter (IN_OUT declaration), a direct I/O output of command bits to PQ0 ... PQn is not permitted. In addition, the specification of local memories should not be used.

Explanation: Command input bit

The parameter exists eight times, once per channel.

When the input of a command is detected, the command is immediately transferred. Each command is transferred to bit 0 of a byte and transferred in a copy of the byte.

When a command is detected, the entire command byte and its copy are always transferred.

The command inputs are not reset automatically and are not monitored for a permanently pending value 1. Make sure you reset the pending command.

3.1.15 Command typical Cmd08X_R

Validity

S7-1500

Function

Receiving 8-byte commands over 8 channels for specific bits

For 1-out-of-8 check see below.

Parameters

General parameters: **PartnerNo**

PartnerObjectNo

For a description, see section Reoccurring parameters (Page 24).

Name: **Enabled**

For the declaration, data type, range of values, default and function, refer to the section Reoccurring parameters (Page 24).

Typical-specific response: Enable block processing

If processing is disabled the FB only checks to see if the processing lock has been canceled. Any commands that are still received are not output. The FB cannot communicate at the organizational level in this status because it cannot send or receive org. frames.

If the "Enabled" input is operable by a switch, this local disable means received commands are no longer output. Since the block does not send Org. frames, it cannot report this local disable back to the partner itself. This must be done with another typical, e.g. Bin04B_S.

Name:	**CommandOutputTime**
Declaration:	INPUT
Data type:	INT
Range of values:	0 ... 500
Default:	500
Explanation:	Command output time for command outputs in milliseconds The specified time applies to all command outputs. The output time is started separately for each command channel. Only if the same command is received again within the output, is its output time restarted. The output times of the other commands are not affected. With the value 0 a set command output is not reset by the command typical. You need to do this via the user program.

Name:	**NewData**
Explanation:	For the declaration, data type, range of values, default and function, refer to the section Reoccurring parameters (Page 24). The output "NewData" is always set to TRUE for one OB1 cycle when the FB has received new data and output it to the respective "CommandOuputXX" output.

Name:	**CommandOutput01..08**	
Declaration:	IN_OUT	
Data type:	Bool	
Range of values:	See address range	
Default:	0 (B#16#0)	
Address range:	Output bits	Q 0.0 ... Q n.7
	Memory	M 0.0 ... M n.7
	Data bit	DBm.DBX 0.0 ... n.7
	Since the parameter is an in/out parameter (IN_OUT declaration), a direct I/O output of command bits to PQ0 ... PQn is not permitted. In addition, the specification of local memories should not be used.	

Explanation: Command output bit

The parameter exists eight times, once per channel.

To allow the command outputs to be reset both by the command typical itself and by the user program (when output time = 0), the parameter is declared as an in/out parameter (IN_OUT).

An internal 1-out-of-8 check is performed in each byte.

- If more than 1 bit is received in one byte, the following error is output:

  0xB174 - 1-out-of-8 error
- If the copy of the command byte is not identical, the following error is output:

  0xB173 - Command and control byte not identical

The command is deleted in both cases.

3.1.16 Counted value typicals Cnt01D_S / Cnt04D_S

Function

- Cnt01D_S: Send 1 counted value (32 bits).
- Cnt04D_S: Send 4 counted values (32 bits).

  Note that the parameter "DifferenceValue" for forming the difference value can only be activated at the same time for all four counted values.

Parameters

General parameters: **PartnerNo**
PartnerObjectNo
Enabled
ImageMemory
Conditional
Unconditional
TimeStamp
For a description, see section Reoccurring parameters (Page 24).

Name: **GeneralTriggerCommand**
Declaration: INPUT
Data type: BOOL
Range of values: TRUE / FALSE

Default:	FALSE
Explanation:	Restore collective command The restore collective command as a central system memory bit belongs to the organizational SINAUT system commands. Set the parameter to TRUE if the counted value transfer is to be triggered by a restore collective command. • When the destination subscriber no. (PartnerNo) = 0 (transfer to all), the restore collective command is taken into account. When the restore collective command is detected, the currently accumulated counted value is transferred regardless of other triggers for transfer. The restore bit is inverted in this counted value. • If in the typical an explicit destination subscriber no. (PartnerNo > 0) is configured, the restore collective command is evaluated in the corresponding subscriber object in the central administration. You can use the parameters "GeneralTriggerCommand" and "TriggerInput" at the same time. In this case the transfer is triggered both by an edge change 0 → 1 at "TriggerInput" and when a restore collective command is received.

Name:	**TriggerInput**	
Declaration:	INPUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	FALSE	
Address range:	Input	I 0.0 ... I n.7
	Memory	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7
Explanation:	Trigger input With the edge change 0 → 1 of the "TriggerInput" input the triggered transfer can be triggered at a required time regardless of other criteria for a transfer.. The currently accumulated counted value is transmitted. The restore bit (see above) is inverted in this counted value. Example: Time-driven transmission with time stamp for supplying an archive in the control center. You can use the parameters "GeneralTriggerCommand" and "TriggerInput" at the same time. In this case the transfer is triggered both by an edge change 0 → 1 at "TriggerInput" and when a restore collective command is received. With the setting FALSE no restoring and no transfer via the "TriggerInput" input is triggered.	

Name:	**Counter_1 (Cnt01D_S)** **Counter_1 ... _4 (Cnt04D_S)**
Declaration:	INPUT
Data type:	COUNTER
Range of values:	0 ... 32767 • Z0 as placeholder or • Z1 ... Zn n depends on the CPU type.
Default:	-
Explanation:	Number of the SIMATIC counter Here, you specify the SIMATIC counter in which the pulses were counted time-driven. This counting takes place in the background using FC PulseCounter that is called in a cyclic interrupt OB (for example in OB35). See also section FC PulseCounter (Page 149) and Cyclic interrupt OB (Page 16). The COUNTER data type cannot be preassigned a value. If you configure the Z0 placeholder, the corresponding counted value is not processed.

Name:	**DifferenceValue**
Declaration:	INPUT
Data type:	INT
Range of values:	0 ... 31767
Default:	0
Explanation:	Difference value • When a value between 1 and 31767 is configured, the counted value is transferred as soon as the difference between the current and most recently transferred counted value reaches the value specified here. • If the default value 0 is configured, a counted value is transferred only in the following situations: – On an edge change 0 → 1 at the "TriggerInput" input – On receipt of a restore command when "GeneralTriggerCommand" = TRUE. Select the difference value dependent on the maximum pulse rate per second. Do not select a value that is too low so that the counted value is not constantly transferred to the TIM. This would put load on the communications path to the CPU and the send queue of the CPU. **Note on Cnt04D_S** In the typical this processing parameter for forming the difference value only exists once. It applies to all 4 counted values together. It is not possible to set the parameter for the individual counted values. When using this parameter, each typical should therefore only acquire counted values that can be processed identically.

3.1.17 Counted value typicals Cnt01D_R / Cnt04D_R

Function

- Cnt01D_S: Receive 1 counted value (32 bits)
- Cnt04D_S: Receive 4 counted values (32 bits)

Parameters

General parameters: **PartnerNo**
PartnerObjectNo
Enabled
For a description, see section Reoccurring parameters (Page 24).

Name:	**BCD_Format**
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Default:	TRUE
Explanation:	Counted value output in BCD format • If the parameter is activated the received counted value is output as a positive BCD value at the "CountedValueOutput_n" output. • If the parameter is deactivated the counted value is output as a positive 32-bit integer value. For the different value ranges of the two formats, see the parameter "CountedValueOutput_n". If the maximum counted value that can be represented is exceeded, the counted value starts again at 0 and counting continues in the positive numeric range. **Note on Cnt04D_R** The parameter exists only one in the typical and it applies to all 4 counted values together. It is not possible to make an individual setting per counted value. When using this parameter, per typical only counted values with an identical output format should be output.

Name:	**CntValInvalid**
Declaration:	OUTPUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Default:	FALSE

Address range:	Output	Q 0.0 ... Q n.7
	Memory	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7

Explanation: Counted value invalid

When evaluating "CntValInvalid" remember that the bit might only be set for one OB1 cycle.

The CntValInvalid output indicates whether the last received counted value was invalid. . With "Cnt04D_R", this counts as a group display for all 4 counted values, see the note below.

The output shows the validity status of the most recently received counted value in inverted form.

The output serves the following purposes:

- Error display
- Signal for user-specific further processing

  You can, for example, react to the lack of up-to-dateness, by correcting the counted value output at "CountedValueOutput_n" with possibly lost counting pulses.

If you do not require the parameter, simply leave it open.

Note on Cnt04D_R
Although all 4 counted values have their own status bit in the data frame, for the status at the output "CntValInvalid" only the bit of the first counted value in the previously received data frame is evaluated.
This status, however, applies to all 4 counted values.
(All counted values in the data frame always have the same status.)

Name:	**RestoreStatus**	
Declaration:	OUTPUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	FALSE	
Address range:	Output	Q 0.0 ... Q n.7
	Memory	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7

Explanation: Status of the restore bit in the received counted value.

The "RestoreStatus" output indicates the current status of the restore bit from the last received counted value frame.

You can use the output for user-specific further processing.

Example:
You can only access the information at "CountedValueOutput_n" when a change has been detected at the "RestoreStatus" output; in other words, when the counted value has been received due to a restore, such as a local time-driven restore.

If you do not require the parameter, simply leave it open.

Note on Cnt04D_R
Although all 4 counted values have their own restore bit in the data frame, for the status at the "RestoreStatus" output only the restore bit of the first counted value in the previously received data frame is evaluated.
This status, however, applies to all 4 counted values.
(All counted values in the data frame always have the same status.)

Name: **NewData**

Explanation: For the declaration, data type, range of values, default and function, refer to the section Reoccurring parameters (Page 24).

The output "NewData" is always set to TRUE for one OB1 cycle when the FB has received new data and output it to the output / outputs "CountedValueOutput_1" to "CountedValueOutput_4"

Name: **CountedValueOutput_1 (Cnt01D_S)**
CountedValueOutput_1 ... _4 (Cnt04D_S)

Declaration: IN_OUT

Data type: DWord, UDInt

Range of values:
- Integer: 0 ... 2 147 483 647
- BCD: 0 ... 9 999 999

Default: 0

Address range:

Output (DWORD)	QD0 … QDn
Memory (DWORD)	MD0 ... MDn
Data (DWORD)	DBm.DBB0 … n

Since the parameter is an in-out parameter (declaration IN_OUT), direct I/O output of the counted value to PQD0 ...PQDn is not permitted.

It is also difficult to specify local bit memory with this parameter type and this should not be used.

Explanation: Counted value output

The counted value typical always adds the newly formed difference value (difference between the new and last received counted value) to the value currently output at the counted value output.

The counted value output is a double word in which the counted value is stored in BCD format or as a 32-bit integer value (depending on the "BCD_Format" parameter, see above).

The counted value id always output as a positive number. If the maximum counted value that can be represented is exceeded, the counted value starts again at 0 and counting continues in the positive numeric range.

Since the parameter is an in-out parameter (IN_OUT), the value can be reset to 0 or another value at the counted value output by the user program at any time.

3.1.18 Data typical Dat12D_S

Function

Send maximum of 12 double words with any data content.

The content of each double word may be a value in double word (DWORD, DINT, REAL) format, it can also be a mixture of other data types which together form a double word, for example:

- 4 bytes
- 2 words
- 2 bytes + 1 word

Sending the data area can be triggered in two ways:

- By a change check

  The data is transferred as soon as a bit changes ("SendOnChange" = TRUE).

- By the user program

  The transfer can be triggered by an edge change 0 → 1 at the "TriggerInput" input

  For time-driven transfer FC Trigger can be used.

With "SendAll" you can also specify whether the transfer always includes all data or only the data double words that have changed.

Note

Remember double word boundaries

When changed data is transferred and the data area contains values in double word format, make sure that the double word values are actually located in one of the maximum 12 double words of the data area to be acquired.

Distribution over two consecutive data double words could lead to the transfer of only one word of the double word value (high or low word) because a change has occurred in only that particular word. In this case, the missing word can lead to a data error on the receiving partner (applies to ST7cc, not for an S7 CPU).

Note

DB with standard access

The block has parameters of the "ANY" type. Therefore, leave the "Optimized block access" attribute in the properties of the DB disabled.

Parameters

General parameters:

PartnerNo
PartnerObjectNo
Enabled
ImageMemory
Conditional
Unconditional
TimeStamp
For a description, see section Reoccurring parameters (Page 24).

Name:	**SendOnChange**
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Default:	FALSE
Explanation:	Send on change With the setting TRUE, the block runs a change check within the acquired data area "DataInput". The block checks whether at least one bit has changed. If a change is detected, a transfer of the data area is started automatically. You specify whether the entire area is transferred or only the changed part with the "SendAll" parameter. If the setting is FALSE, you need to trigger the transfer via the input parameter "TriggerInput".

Name:	**TriggerInput**	
Declaration:	INPUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	FALSE	
Address range:	Input	I 0.0 ... I n.7
	Memory	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7

Explanation:	Trigger input With the edge change 0 → 1 of the "TriggerInput" input, the transfer of the data frame can be triggered at a required time. Example: Time-driven analog value transfer with time stamp for supplying an analog value archive in the control center. Make sure that you set the "ImageMemory" parameter to FALSE to prevent this data with time stamps from being overwritten when saving on the station TIM. The FC Trigger block can be used for time-driven triggering of a transmission over "TriggerInput". "TriggerInput" actually only triggers transmission indirectly. With a 0 → 1 edge change at "TriggerInput", the data frame is put together with its current values and transferred to the local TIM. The TIM is responsible for the actual transmission to the partner. With dedicated lines or wireless networks the transfer is immediate. With a dial-up connection, it is possible that the data frame is saved first on the TIM and sent at a later point in time. The reason can, for example, be that the data frame is marked as "Conditional spontaneous", see parameter "Conditional". If you do not require the parameter, simply leave it open. You should, however, then set the "SendOnChange" parameter to TRUE so that the data is transmitted automatically at every change. For the triggering you can also select a combination of "SendOnChange" plus "TriggerInput". This means that a transfer is triggered both when a change is detected and at every edge change from 0 to 1 at the "TriggerInput" input. If you use neither "SendOnChange" nor "TriggerInput" to trigger data transfer, the data will only be transferred when there is a single request for this data object or within the framework of a general request.

Name:	**SendAll**
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Default:	TRUE
Explanation:	Send all data with every transfer With the parameter, you specify whether the block will always transfer all data of the area specified with "DataInput" or only changed data. The transfer can be triggered by the activated change check (SendOnChange = TRUE) or by "TriggerInput".

- SendAll = TRUE

  Always send all data
- SendAll = FALSE

  Send only changed data

  Exception:
  If "SendAll" is set to = FALSE, the transfer is triggered by "TriggerInput" and if no data has changed at this time, the complete area will be transferred. For this exception this corresponds to "SendAll" = TRUE.

  When only the changed data area is transferred ("SendAll" = FALSE), this area consists of the first and the last double word in which a change was detected and all words located in between, even if these have not changed.

If there is a single request for this data object or within the framework of a general request, all data words of the area specified by "DataInput" are always transferred.

Name:	**DataInput**
Declaration:	INPUT
Data type:	ANY
Range of values:	See address range
Default:	P#P 0.0 VOID 0 (null pointer)
Address range:	P#DBxx.DBX yy.0 DWORD zz • xx: Data block number 1...32767 • yy: Byte number • zz: Number of double words 1...12 starting at byte number yy Example: `P#DB20.DBX100.0 DWORD 4` Remember the periods and spaces when entering the pointer! Note that the default value (null pointer) is not permitted. A pointer with a real address must be specified.
Explanation:	Data input area The ANY pointer addresses the data area in which the data to be acquired is located. This data area must be within a data block and its length can vary between 1 and 12 data double words. For information on the content and formats, refer to the section "Function" above. If the parameter assignment is incorrect (null pointer, length > 12, data area not a DB), an error message is entered in the diagnostics buffer (event ID B114, [Info2/3] = 11). The CPU does not change to STOP. The block is then no longer processed, however, until the error has been corrected.

3.1.19 Data typical Dat12D_R

Function

Receive maximum of 12 double words with any data content.

The content of each double word may be a value in double word (DWORD, DINT, REAL) format, it can also be a mixture of other data types which together form a double word, for example:

- 4 bytes
- 2 words
- 2 bytes + 1 word

Dat12D_R stores the received data without further processing in the data area specified by "DataOutput". You need to evaluate and process the received data with the user program.

Note

DB with standard access

The block has parameters of the "ANY" type. Therefore, leave the "Optimized block access" attribute in the properties of the DB disabled.

Parameters

General parameters: **PartnerNo**
PartnerObjectNo
Enabled
For a description, see section Reoccurring parameters (Page 24).

Name:	**DataOutput**
Declaration:	INPUT
Data type:	ANY
Range of values:	See address range
Default:	P#P 0.0 VOID 0 (null pointer)
Address range:	P#DBxx.DBX yy.0 DWORD zz • xx: Data block number 1...32767 • yy: Byte number • zz: Number of double words 1...12 starting at byte number yy Example: `P#DB20.DBX100.0 DWORD 4` Note that the default value (null pointer) is not permitted. A pointer with a real address must be specified.

Explanation: Data output area

The ANY pointer addresses the data area in which the received data is saved. This data area must be within a data block and its length can vary between 1 and 12 double words.

For information on the content and formats, refer to the section "Function" above.

Dat12D_R stores the received data without further processing in the data area specified by "DataOutput". You need to evaluate and process the received data with the user program.

When only changed data is sent by the partner object Dat12D_S, it is possible that only part of the data output area is newly written. this is the area in which the changes were detected at the acquisition end

If the parameter assignment is incorrect (null pointer, length > 12, data area not a DB), an error message is entered in the diagnostics buffer (event ID B114, [Info2/3] = 11). The CPU does not change to STOP. The block is then no longer processed, however, until the error has been corrected.

Name: **NewData**

Explanation: Receive new data

For the declaration, data type, range of values, default and function, refer to the section Reoccurring parameters (Page 24).

Whenever the block has received new data values from the partner object and has output them to the output field "DataOutput", the "NewData" output is set to TRUE for one OB1 cycle.

3.1.20 Data typical Dat12x1D_S

Validity

S7-1500

Function

Send data a maximum of 12 times with any data content via 12 channels for specific double words

You define the number of channels of the object used (1..12) in the "ChnCnt" parameter.

One data type is permitted per channel. You specify the data type for the content of a channel in the "DataInputXX" parameter using the respective ANY pointer.

The number of transferred data per channel must not exceed the length of 1 double word (32 bits). If you specify the number of data for one channel with 3 CHAR, for example, the fourth byte is not read and will not be evaluated for the transfer time (SendOnChange).

Associated instance data block

Note

DB with standard access

The typical uses ANY pointers in the "DataInputXX" parameters. Disable the "Optimized block access" attribute in the properties of the DB.

Triggering transfer and transferred data areas

As a maximum, the frame is sent completely with all channels specified in "ChnCnt". Depending on the parameter assignment, not all data of all channels are transferred.

Sending the data can be triggered in the following ways:

- **Time-driven**

  The transfer can be triggered via the user program.

  With an edge change 0 → 1 at the "TriggerInput", all data specified under "DataInputXX" is always transferred.

  For time-driven transfer FC Trigger can be used.

- **On change**

  The transfer is triggered via the change control ("SendOnChange" = TRUE).

  – On change in only one channel:

    Only the data of the changed channel is transferred.

  – On change in multiple channels:

    The transferred data comprises a contiguous area from the first changed channel to the last changed channel.

    Unchanged channels that may be before or after the transferred area are not transferred.

- **Requests**

  If there is a single request for this data object or within the framework of a general request, all data of the typical is always transferred.

To trigger the data transfer, you can also select a combination of "SendOnChange" plus "TriggerInput". This means that a transfer is triggered both when a change is detected and at every edge change from 0 → 1 at the "TriggerInput" input.

If you use neither "SendOnChange" nor "TriggerInput" to trigger data transfer, the data is only transferred when there is a single request for this data object or within the framework of a general request.

Two parameter sets for different prioritization of the channels

The typical provides the possibility to handle channels differently with respect to saving and the transfer time by means of two parameter sets. Both parameter sets contain the following parameters:

Parameter set 1

- ImageMemory01
- Conditional01
- SendOnChange01

Parameter set 2

- ImageMemory02
- Conditional02
- SendOnChange02

Function of the parameters:

- ImageMemory

  Determines whether the data is transferred according to the image memory principle or the send buffer principle.

- Conditional

  Determines whether the data is transferred "Conditional spontaneous" or "Unconditional spontaneous".

- SendOnChange

  Determines whether the data is transferred on change.

You can prioritize the channels as follows, for example, using the different parameter assignment of the respective parameters:

- Channels for parameter set 1

  Important events to be saved individually and transferred immediately.

- Channels for parameter set 2

  Operating data whose values may be overwritten and which does not need to be transferred immediately.

Each channel is assigned to one of the two parameter sets using the "ChnSets" parameter. You will find a parameter assignment example below.

Parameters

General parameters:

PartnerNo
PartnerObjectNo
Enabled
ImageMemory01..02 *
Conditional01..02 *
TimeStamp

For a description, see section Reoccurring parameters (Page 24).

***** Parameter one time each for parameter set 1 and 2

Name:	**SendOnChange01..02 ***	
Declaration:	INPUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	FALSE	
Parameter assignment:	* Parameter once for parameter set 1 and 2	
Explanation:	Send on change The parameter exists two times, once per channel. With the setting TRUE, the block runs a change check within the acquired data area "DataInput". The block checks whether at least one bit has changed. If a change is detected, a transfer of the data area is started automatically. See above for information on the transferred area. If the setting is FALSE, you need to initiate the transfer via the input parameter "TriggerInput".	

Name:	**TriggerInput**	
Declaration:	INPUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	FALSE	
Address range:	Input	I 0.0 ... I n.7
	Memory	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7
Explanation:	Trigger input With the edge change 0 → 1 of this input, the transfer of the data frame can be triggered at a selected time. Example: Time-driven analog value transfer with time stamp for supplying an archive in the control center. Make sure that you set the "ImageMemory" parameter to FALSE to prevent this data with time stamps from being overwritten when saving on the station TIM.	

The FC Trigger block can be used for time-driven triggering of transmission over "TriggerInput".

"TriggerInput" actually only triggers transmission indirectly. With a 0 → 1 edge change at "TriggerInput", the data frame is put together with its current values and transferred to the local TIM.

With dedicated lines or wireless networks, the TIM transfers the data immediately.

With a dial-up connection, the transfer depends on the setting of the parameter "Conditional":

- "Conditional" = TRUE (conditional spontaneous)

  The data frame is first saved on the TIM and transferred on the next connection establishment.
- "Conditional" = FALSE (unconditional spontaneous)

  The data frame is transferred immediately.

Name:	**ChnCnt**
Declaration:	INPUT
Data type:	Int
Range of values:	1..12
Explanation:	Number of transfer channels Of the max. 12 channels of the object, between 1 and 12 channels can be defined.

Name:	**ChnSets**
Declaration:	INPUT
Data type:	WORD
Format:	Bit sequence of 16 bits
Default:	0000000000000000
Explanation:	Assignment of transfer channels to parameter set 1 or 2 Assign each channel via its bit to one of the two parameter sets, beginning with bit 1 for channel 1. Bit 0 always needs to be occupied with 0 (null) (reserved). Coding of the individual bits: • **0 = Assignment of a channel to parameter set 1** • **1 = Assignment of a channel to parameter set 2** With the two parameter sets, you can transfer the channels differently.

Example:

- Parameter set 1

  Important events, possible parameter assignment:
  - ImageMemory01 = FALSE (Send buffer principle)
  - Conditional01 = FALSE (Unconditional spontaneous)
  - SendOnChange01 = TRUE (Transfer to TIM on change)
- Parameter set 2

  Operating data, possible parameter assignment:
  - ImageMemory02 = TRUE (Image memory principle)
  - Conditional02 = TRUE (Conditional spontaneous)
  - SendOnChange02 = TRUE (Transfer to TIM on change)

Note:

"ImageMemory" and "Conditional" apply to the entire frame.

- If sending of the frame is triggered via "TriggerInput", then the entire frame is sent as unconditional spontaneous send buffer frame as soon as "ImageMemory" and "Conditional" are set to FALSE for a channel.
- If sending of the frame is triggered via "SendOnChange" to multiple channels, then the entire frame is sent as unconditional spontaneous send buffer frame as soon as "ImageMemory" and "Conditional" are set to FALSE for a channel.
- If a change is only detected at one channel ("SendOnChange" = TRUE), then the frame is sent in the mode that is specified by the "ImageMemory" and "Conditional" parameters of this channel.

Table 3- 5 Assignment of the bits of the "ChnSets" parameter to the channels

Bit	**.15**	**.14**	**.13**	**.12**	**.11**	**.10**	**.9**	**.8**	**.7**	**.6**	**.5**	**.4**	**.3**	**.2**	**.1**	**.0**
Channel	**-**	**-**	**-**	12	11	10	9	8	7	6	5	4	3	2	1	**- ***
***** Bit 0 is reserved.																

Table 3- 6 Parameter assignment example for "ChnSets": Assignment of seven channels to parameter set 1 or 2

Bit	.15	.14	.13	.12	.11	.10	.9	.8	**.7**	**.6**	**.5**	**.4**	**.3**	**.2**	**.1**	.0
Assignment	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	0 *****

***** Bit 0 is reserved.

Seven channels are assigned in the example. Channels 8..12 are not used.

Coding of the individual bits:

- 0 = Assignment to parameter set 1

  In the example, the channels 2, 3 and 6 are assigned to parameter set 1.
- 1 = Assignment to parameter set 2

  In the example, the channels 1, 4, 5 and 7 are assigned to parameter set 2.

Name:	**DataInput01..12** The parameter exists twelve times, once per channel.
Declaration:	INPUT
Data type:	ANY
Range of values:	See address range
Default:	P#P 0.0 VOID 0 (null pointer)
Address range:	P#DBxx.DBXyy.0 TYPE zz • xx: Data block number • yy: Byte number • TYPE: Data type Permitted data types are: Bool, Byte, Char, SInt, USInt, Int, UInt, Word, DInt, DWord, Real, UDInt • zz: Length as number (specified data type) as of byte no. yy Max. length: 1 DWord Example: `P#DB20.DBX100.0 BYTE 3` Area with 3 bytes in DB20 as of DBB100
Explanation:	Data input area The ANY pointer addresses the data area in which the data to be acquired is located. The occupied null pointer is not permitted. Specify a pointer with a real address. The data area must be in a data block and can have a maximum length of 4 bytes (max. 1 double word). If not all 4 bytes are transferred for a channel or if they are not to be evaluated for triggering of the frame (SendOnChange), make sure you observe the correct parameter assignment. For information on the content and formats, refer to the section "Function" above. If the parameter assignment is incorrect (null pointer, length > 4, data area not a DB), an error message is entered in the diagnostics buffer (event ID B114, [Info2/3] = 11). The CPU does not change to STOP. The block is then no longer processed, however, until the error has been corrected.

3.1.21 Data typical Dat12x1D_R

Validity

S7-1500

Function

Receive data a maximum of 12 times with any data content via 12 channels for specific double words

You define the number of channels of the object used (1..12) in the "ChnCnt" parameter.

One data type is permitted per channel. You specify the data type for the content of a channel in the "DWxx_DataOut" parameter using the respective ANY pointer.

Dat12x1D_R saves the received data without further processing for specific channels in the respective data area specified at "DWxx_DataOut". You need to evaluate and process the received data with the user program.

Note

DB with standard access

The typical uses an ANY pointer in the "DWx_DataOut" parameter. Therefore, disable the "Optimized block access" attribute in the properties of the DB.

Parameters

General parameters: **PartnerNo**
PartnerObjectNo
Enabled
For a description, see section Reoccurring parameters (Page 24).

Name:	**ChnCnt**
Declaration:	INPUT
Data type:	Int
Range of values:	1..12
Explanation:	Number of transfer channels Of the max. 12 channels of the object, between 1 and 12 channels can be defined.

Name:	**DW1_DataOut .. DW12_DataOut**
Declaration:	INPUT
Data type:	ANY

Range of values:	See address range
Default:	P#P 0.0 VOID 0 (null pointer)
Address range:	P#DBxx.DBXyy.0 TYPE zz • xx: Data block number • yy: Byte number • TYPE: Data type Permitted data types are: Bool, Byte, Char, SInt, USInt, Int, UInt, Word, DInt, DWord, Real, UDInt • zz: Length as number (specified data type) as of byte no. yy Max. length: 1 DWord Example: `P#DB20.DBX100.0 BYTE 3` Area with 3 bytes in DB20 as of DBB100
Explanation:	Data output area The parameter is available once for each channel (max. 12). The ANY pointer addresses the data area in which the received data is saved. The occupied null pointer is not permitted. Specify a pointer with a real address. The data area must be in a data block and can have a maximum length of 4 bytes (max. 1 double word). If not all 4 bytes are to be output for a channel, make sure you observe the correct parameter assignment. For information on the content and formats, refer to the section "Function" above. Dat121xD_R stores the received data without further processing in the data area specified at "DataOutput". You need to evaluate and process the received data with the user program. When only changed data is sent by the partner object Dat121xD_S, it is possible that only part of the data output area is re-written. this is the area in which the changes were detected at the acquisition end You can identify changed areas by "NewData". If the parameter assignment is incorrect (null pointer, length > 4, data area not a DB), an error message is entered in the diagnostics buffer (event ID B114, [Info2/3] = 11). The CPU does not change to STOP. The block is then no longer processed, however, until the error has been corrected.

Name:	**NewData**
Declaration:	OUTPUT
Data type:	Array [0..11] of BOOL
Format:	Bit sequence of 16 bits
Range of values:	See address range

Default:	0000000000000000	
Address range:	Output words	QW0 ... QWn PQW0 ... PQWn
	Memory words	MW0 ... MWn LW0 ... LWn
	Data words	DBm.DBW0 ... n
Explanation:	Receive new data	

Whenever the block receives new data segments, the display of status bits 1 to 12 according to the received data segments 1 to 12 appears in "NewData".

If at least one data segment of the received data contains changes, bit 0 of "NewData" is set to TRUE for one OB1 cycle.

When receiving a sequence of several data segments (data frames), the status bits 1 to 12 in the "NewData" parameter are set to TRUE one after the other and remain set to TRUE until the last segment has been received.

The output is intended for user-specific further processing, for example to react in a specific way to receipt of new data.

If you do not require the parameter, simply leave it open.

3.1.22 Data typical Dat256D_S

Function

Send a maximum of 256 double words with any data content

The content of each double word can be a value in double word format (DINT, REAL etc.). A combination of other formats is permitted that together result in a double word again, for example

- 32 Bool
- 4 bytes
- 2 words
- Any combination such as 2 bytes plus 1 word etc.

Note

Remember double word boundaries

When changed data is transferred and the data area contains values in double word format, make sure that the double word values are actually located in one of the maximum 256 double words of the data area to be acquired.

Distribution over two consecutive data double words could lead to the transfer of only one word of the double word value (high or low word) because a change has occurred in only that particular word. In this case, the missing word can lead to a data error on the receiving partner (applies to ST7cc, not for an S7 CPU).

Sending the data area can be triggered in two ways:

- By a change check

  The data is transferred as soon as a bit changes ("SendOnChange" = TRUE).

- By the user program

  The transfer can be triggered by an edge change 0 → 1 at the "TriggerInput" input

  For time-driven transfer FC Trigger can be used.

With "SendAll" you can also specify whether the transfer always includes all data or only the data double words that have changed.

With S7-300 CPUs with X communication, the maximum length of a data frame is 76 bytes. 1024 bytes of net data are transferred using a serial transfer process consisting of a sequence of at least 22 data frames (segments). Each data frame apart from the last contains a segment of 48 bytes of net data of the input data area.

To ensure data consistency when the "SendAll" parameter is activated or during a general or single request, the data is transferred in consecutive segments. During the transfer process, the status is indicated by "SendAllBusy". On the recipient, the status is indicated at the "DataStatus" output.

Note

TriggerInput - SendAllBusy

If "TriggerInput" is triggered when "SendAllBusy" = TRUE, this leads to the "DataLoss" error message (status in the frame header) if the transfer is triggered again.

Only when "SendAllBusy" = FALSE is set is the edge change 0 → 1 triggered at "TriggerInput".

If the transfer is interrupted, "SendAllError" is indicated. An entry is also made in the diagnostics buffer with the event ID B14DTD7_Diagnostics.

If the transfer is incomplete, the data status at the recipient is also "invalid". This is indicated on the recipient in the DataStatus parameter. Apart from this an entry with the event ID B13BTD7_Diagnostics is written to the diagnostics buffer.

Note

Availability of the partner

If the status of the partner changes from "available" to "unavailable", the transfer of all data is stopped immediately. All object data is deleted from the TIM buffer. This can lead to loss of data.

As soon as the partner is available again, the automatic general request ensures that the data of the partner is up-to-date again for the next transfer.

Note

Dat256D_S and Dat256D_R require the UDT "TransmitBlock".

When using the typical, copy the UDT from the global library into the "PLC data types" directory of the CPU. The UDT is automatically referenced by the typical from the block directory of the CPU, but not from the global library.

Note

DB with standard access

The block has parameters of the "ANY" type. Therefore, leave the "Optimized block access" attribute in the properties of the DB disabled.

Parameters

General parameters: **PartnerNo**
PartnerObjectNo
Enabled
Unconditional
TimeStamp
For a description, see section Reoccurring parameters (Page 24).

Name:	**SendOnChange**
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Default:	FALSE
Explanation:	Send on change With the setting TRUE, the block runs a change check within the acquired data area "DataInput". The block checks whether at least one bit has changed. If a change is detected, a transfer of the data area is started automatically. You specify whether the entire area is transferred or only the changed part with the "SendAll" parameter. If the setting is FALSE, you need to trigger the transfer via the input parameter "TriggerInput".

Name:	**TriggerInput**
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Default:	FALSE

Address range:	Input	I 0.0 ... I n.7
	Memory	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7

Explanation: Trigger input

With the edge change 0 → 1 of the "TriggerInput" input, the transfer of the data frame can be triggered at a required time.

Example:
Time-driven analog value transfer with time stamp for supplying an analog value archive in the control center.

The FC Trigger block can be used for time-driven triggering of a transmission over "TriggerInput".

"TriggerInput" actually only triggers transmission indirectly. With a 0 → 1 edge change at "TriggerInput", the data frame is put together with its current values and transferred to the local TIM. The TIM is responsible for the actual transmission to the partner. With dedicated lines or wireless networks the transfer is immediate. With a dial-up connection, it is possible that the data frame is saved first on the TIM and sent at a later point in time. The reason can, for example, be that the data frame is marked as "Conditional spontaneous", see parameter "Conditional".

Select suitable trigger points so that the data on the TIM is not overwritten by buffer overflow (intervals too long).

If you do not require the parameter, simply leave it open. You should, however, then set the "SendOnChange" parameter to TRUE so that the data is transmitted automatically at every change.

For the triggering you can also select a combination of "SendOnChange" plus "TriggerInput". This means that a transfer is triggered both when a change is detected and at every edge change from 0 to 1 at the "TriggerInput" input.

If you use neither "SendOnChange" nor "TriggerInput" to trigger data transfer, the data will only be transferred when there is a single request for this data object or within the framework of a general request.

Do not transfer any analog values for which the "SendOnChange" parameter = TRUE is set without first preprocessing the process data. You will find more detailed information on this with the analog value typical Ana04W_S, parameter "ThresholdValue".

Name:	**SendAll**
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Default:	TRUE

Explanation: Send all data with every transfer

With the parameter, you specify whether the block will always transfer all data of the area specified with "DataInput" or only changed data. The transfer can be triggered by the activated change check (SendOnChange = TRUE) or by "TriggerInput".

- SendAll = TRUE

  Always send all data
- SendAll = FALSE

  Send only changed data

  Exception:
  If "SendAll" is set to = FALSE, the transfer is triggered by "TriggerInput" and if no data has changed at this time, the complete area will be transferred. For this exception this corresponds to "SendAll" = TRUE.

If there is a single request for this data object or within the framework of a general request, all data words of the area specified by "DataInput" are always transferred.

Name:	**DataInput**
Declaration:	INPUT
Data type:	ANY
Range of values:	See address range
Default:	P#P 0.0 VOID 0 (null pointer)
Address range:	P#DBxx.DBX yy.0 DWORD zz • xx: Data block number 1...32767 • yy: Byte number • zz: Number of double words 1...256 starting at byte number yy Example: `P#DB20.DBX100.0 DWORD 200` Note that the default value (null pointer) is not permitted. A pointer with a real address must be specified.

Explanation: Data input area

The ANY pointer addresses the data area in which the data to be acquired is located. This data area must be within a data block and its length can vary between 1 and 256 data double words. For information on the possible double word formats, refer to the section "Function" above.

If the parameter assignment is incorrect (null pointer, length > 256, data area not a DB), an error message is entered in the diagnostics buffer (event ID B114, [Info2/3] = 11). The CPU does not change to STOP. The block is then no longer processed, however, until the error has been corrected.

Data consistency;
If a data segment to be transferred consists of a maximum of 48 bytes, data consistency during the transfer is assured.

Name:	**SendAllBusy**
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Default:	FALSE
Explanation:	Block being processed with "SendAll" = TRUE This output indicates that the block is currently transferring the data specified by "DataInput". The procedure is activated either by a remote single or general request or by a local internal or external trigger. If "SendAll" is set to TRUE, the transfer of all data is triggered either by internal change control (SendOnChange = TRUE) or by the external "TriggerInput" (edge change 0 → 1). The edge change 0 → 1 has no effect whatsoever with an external "TriggerInput" as long as "SendAllBusy" indicates TRUE. The edge change 0 → 1 of "TriggerInput" only takes effect when "SendAllBusy" = FALSE.

Name:	**SendAllError**
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Default:	FALSE
Explanation:	Error when processing "SendAll" "SendAllError" is set to TRUE when the connection is interrupted during processing of "SendAll". Apart from this an entry is written to the diagnostics buffer. "SendAllError" remains set to TRUE until it is reset by the user program or by the next CPU restart.

3.1.23 Data typical Dat256D_R

Function

Receive maximum of 256 double words with any data content.

The content of each double word may be a value in double word format (e.g. DINT, REAL etc.); it can also be a mixture of other formats which together form a double word, for example,

- 32 Bool
- 4 bytes

- 2 words
- Any combination such as 2 bytes plus 1 word etc.

Note

Remember double word boundaries

When changed data is transferred and the data area contains values in double word format, make sure that the double word values are actually located in one of the maximum 256 double words of the data area to be acquired.

Distribution over two consecutive data double words could lead to the transfer of only one word of the double word value (high or low word) because a change has occurred in only that particular word. In this case, the missing word can lead to a data error on the receiving partner (applies to ST7cc, not for an S7 CPU).

Dat256D_R stores the received data without further processing in the data area defined by "DataOutput". Evaluate the received data with the user program.

With S7-300 CPUs with X communication, the maximum length of a data frame is 76 bytes (net 48 bytes). 1024 bytes of net data are transferred using a serial transfer process consisting of a sequence of at least 22 data frames (segments). Each data frame apart from the last contains a segment of 48 bytes of net data of the output data area.

Each time a detected data segment is received, this is indicated by a corresponding status (bit 1 to 22) of the "NewData" output parameter.

If a change was detected in the data segment, the status bit 0 is also set to TRUE in "NewData" for one CPU cycle. This makes it possible to recognize which segment of the output data area has changed.

Note

When receiving a sequence of several data segments (data frames), the status bits 1 to 22 in the "NewData" parameter are set to TRUE one after the other and remain set to TRUE until the last segment has been received.

If a data segment (data frame) is not part of a received sequence (SendAll = FALSE), the status remains set to TRUE for only one CPU cycle.

To ensure data consistency when "SendAll"= TRUE or during a general or single request, the data area is updated in consecutive individual segments.

During receipt, the status is indicated by the "DataStatus" output byte ("SequenceState" status). If the receive sequence was completed successfully, the data output area is up-to-date and the output data is consistent. This is indicated by "DataStatus" ("DataValid" = TRUE status).

Note

The consistency of the data segments or the limit segments cannot be guaranteed if the "SendAll" parameter was set to FALSE on the sender.

Receipt of a sequence can be disrupted by the following causes:

- The receive sequence was interrupted when communication to the partner fails during an active sequence (event ID B13BTD7_Diagnostics).
- The monitoring time was exceeded. Not all segments could be received within the time set for the "MonitoringTime" parameter (event ID B13CTD7_Diagnostics).
- Other receiving errors occur (event ID B13DTD7_Diagnostics), for example:
  - A new receive sequence is registered during an active, error-free sequence.
  - A spontaneous segment (data frame) is received during an active sequence.

Note

Dat256D_S and Dat256D_R require the UDT "TransmitBlock".

When using the typical, copy the UDT from the global library into the "PLC data types" directory of the CPU. The UDT is automatically referenced by the typical from the block directory of the CPU, but not from the global library.

Note

DB with standard access

The block has parameters of the "ANY" type. Therefore, leave the "Optimized block access" attribute in the properties of the DB disabled.

Parameters

General parameters:	**PartnerNo** **PartnerObjectNo** **Enabled** For a description, see section Reoccurring parameters (Page 24).	
Name:	**SingleRequest**	
Declaration:	INPUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	TRUE	
Address range:	Input	I 0.0 … I n.7
	Memory	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7

Explanation: A single request is sent to the partner object.

If the partner is available, you can send a single request to the partner object. If a reply is returned, the information is forwarded to the data area specified in "DataOutput".

In terms of transfer sequences, there are priorities:

- Lowest priority: TriggerInput

  An active transfer triggered, for example, by "TriggerInput" on the sender can be interrupted by a single request or a general request.
- Medium priority: Single request

  An active transfer triggered, for example, by "TriggerInput" on the sender can be interrupted by a single request or a general request.

  The interrupted or restarted request leads to a restart of the active sequences without an error message.
- Highest priority: General request

  A general request can interrupt itself or a single request.

  The interrupted or restarted request leads to a restart of the active sequences without an error message.

The request interrupted or restarted by a single or general request leads to a restart of the active sequences without an error message. If a sequence was completed successfully, the "DataValid" status of the "DataStatus" output byte remains set to TRUE.

The time taken for the response to the single request is evaluated by the "MonitoringTime" parameter.

Note:
Consistency of the data output area across segments is only assured if the receive sequence was completed successfully.

Name:	**MonitoringTime**
Declaration:	INPUT
Data type:	INT
Range of values:	0 (no limit) / 1 … 32000 (seconds)
Default:	0

Explanation: Maximum time for a complete response to a single request

Each time a single request starts (see SingleRequest parameter), the time specified here is started in "SingleRequest".

If a value higher than 0 is entered and the time for the response sequence is exceeded, an error is indicated via the "DataStatus" output byte (status bits "SequenceState"). Apart from this an entry is written to the diagnostics buffer (event ID B13CTD7_Diagnostics).

Each time a single request starts, "MonitoringTime" is reactivated.

Name:	**DataOutput**
Declaration:	INTPUT
Data type:	ANY
Range of values:	See address range
Default:	P#P 0.0 VOID 0 (null pointer)
Address range:	P#DBxx.DBX yy.0 DWORD zz • xx: Data block number 1...32767 • yy: Byte number • zz: Number of double words 1...256 starting at byte number yy Example: `P#DB20.DBX100.0 DWORD 200` Note that the default value (null pointer) is not permitted. A pointer with a real address must be specified.
Explanation:	Data output area The ANY pointer addresses the data area in which the received data is saved. This data area must be within a data block and its length can vary between 1 and 256 data double words. For information on the possible double word formats, refer to the section "Function" above. Dat256D_R stores the received data without further processing in the data area specified by "DataOutput". You need to evaluate and process the received data with the user program. When only changed data is sent by the partner object Dat256D_S, it is possible that only part of the data output area is newly written. this is the area in which the changes were detected at the acquisition end If the parameter assignment is incorrect (null pointer, length > 12, data area not a DB), an error message is entered in the diagnostics buffer (event ID B114, [Info2/3] = 11). The CPU does not change to STOP. The block is then no longer processed, however, until the error has been corrected.

Name:	**NewData**	
Declaration:	OUTPUT	
Data type:	DWORD	
Range of values:	See address range	
Default:	0 (DW#16#0)	
Address range:	Output (DWORD)	QD0 … QDn PQD0 … PQDn
	Memory (DWORD)	MD0 … MDn LD0 … LDn
	Data (DWORD)	DBm.DBB0 … n

Explanation: Receive new data

Whenever the block receives new data segments, the display of status bits 1 to 22 according to the received data segments 1 to 22 appears in "NewData".

If at least one data segment of the received data contains changes, bit 0 of "NewData" is set to TRUE for one OB1 cycle.

When receiving a sequence of several data segments (data frames), the status bits 1 to 22 in the "NewData" parameter are set to TRUE one after the other and remain set to TRUE until the last segment has been received.

If a data segment (data frame) is not part of a received sequence ("SendAll" = FALSE), the status remains set to TRUE for only one CPU cycle.

The output is intended for user-specific further processing, for example to react in a specific way to receipt of new data.

If you do not require the parameter, simply leave it open.

Name:	**DataStatus**	
Declaration:	OUTPUT	
Data type:	BYTE	
Range of values:	See address range	
Default:	0 (B#16#0)	
Address range:	Output	QB0 … QBn PQB0 … PQBn
	Memory	MB0 … MBn LB0 … LBn
	Data	DBm.DBB0 … n

Explanation: Currentness status of a received data segment

During receipt of a sequence, the current status is indicated by the "DataStatus" output byte:

- If the receipt of the sequence was completed successfully, the "DataOutput" data output area is up-to-date. The status bit "DataValid" is set to TRUE.

  If "SendAll" is set to TRUE on the sender, the data is consistent.
- If the receipt of a sequence is disrupted, the "DataOutput" data output area is not up-to-date and "DataStatus" indicates an error.
  - The status bit "DataValid" is set to FALSE and an entry is written to the diagnostics buffer.
  - The "SequenceState" status shows the error (see table).

Table 3- 7 Bit assignment of "DataStatus"

Bit	Name	Value	Meaning
0	**DataValid**	FALSE	Data invalid
		TRUE	Data valid
1 ... 5	Reserved	- (FALSE)	Not used

Bit	Name	Value	Meaning
7, 6	**SequenceState**	0	No data being received or receipt completed without error.
		1	First segment of a sequence received
		2	Second or higher segment of a sequence received
		3	Errors: • Transmission sequence aborted • Monitoring time exceeded • Other error receiving

3.1.24 Parameter typical Par12D_S

Function

Send 1 to 12 parameter values (each 1 double word) and receive the current, locally valid parameter values from the partner.

The block operates with 1-out-of-n test. The 1-out-of-n check is performed by FC Safe.

Note

FC Safe required

With Par12D_S, data can only be transmitted when FC Safe is linked in the end of cyclic program, see Section FC Safe (Page 118).

The content of each double word may be a value in double word (DWORD, DINT, REAL) format, it can also be a mixture of other data types which together form a double word, for example:

- 4 bytes
- 2 words
- 2 bytes + 1 word

See also the note "Note word boundaries" below.

The data area to be transferred is specified for the "ParameterInput" parameter in the form of an Any pointer. This data area must be within a data block and its length can vary between 1 and 12 double words. The data area sent to the partner or the parameter values entered locally at the partner are returned from there and output here at the "ReturnedParameter" parameter. This output area (Any pointer) must also be within a data block and its length must match that specified for "ParameterInput".

Separate data areas are normally specified for "ParameterInput" and "ReturnedParameter". This makes it easy to recognize what was most recently entered and what is locally valid. However, it is also possible to specify the same data area for both parameters. The two areas then overlap 100% and therefore always match. In this case, you can no longer distinguish the difference between what has been entered most recently and what is locally valid.

Even when separate areas are specified for "ParameterInput" and "ReturnedParameter", it is still possible to ensure that the "ParameterInput" input area is always synchronized with the mirrored back values of "ReturnedParameter". This can be done manually from case to case with the "ApplyRemoteParamMan" input or automatically by setting the "ApplyRemoteParamAuto" parameter to TRUE.

A parameter can also be set locally at the partner object that receives the parameter. The partner object must then be set to ’local’ at the "Local" input parameter (see block Par12D_R). The current status of the "Local" input parameter is reported by the partner object and indicated here at the "LocalOperation" output. As long as the partner object is set to 'local', no parameters are accepted there from other nodes.

The sending of the data area defined by "ParameterInput" can be triggered via the following parameters:

- EnterInput

  You should use this input parameter when the data area defined at "ParameterInput" is entered over hardware (digital and analog input modules). "EnterInput" must then be connected to a button on a console or panel via a digital input. The transmission of the entered values is then triggered by pressing this button.

  The entire data area specified by "ParameterInput" is transferred.

- ContinuousEnterFunct

  Set the parameter to TRUE if you enter the parameters using software, for example via an operator panel (OP). There is a constant check for changes. When a change is detected in the data area defined with "ParameterInput", the data double words that have changed since the last transfer are transferred.

  Only the changed data is transferred (see note "Changed data areas" below).

- Release

  Use this input parameter if you enter the parameters using software, for example via an OP. The "Release" input should then be set using a function key on the OP. Changes are checked when a 1 signal is detected at the "Release" input. The data double words from the data area specified with "ParameterInput" that have changed since the last transfer are transferred.

  Only the changed data is transferred (see note "Changed data areas" below).

- RetransmitAll

  Use this input parameter if you enter the parameters using software, for example via an OP. The "RetransmitAll" input should then be set using a function key on the OP. When a 1 signal is detected at the "RetransmitAll" input, the data area configured in "ParameterInput" is transferred without checking for changes.

  The entire data area specified by "ParameterInput" is transferred.

Note

Changed data areas

When only the changed data area is transferred, this area consists of the first and the last double word in which a change was detected and all words located in between, even if these have not changed.

Example:
The area to be read in is 10 double words long. In this case, changes were detected in the second, fifth and eighth double words. The transferred area is therefore from the 2nd to the 8th double word.

Remember word boundaries

When only changed data is transferred and the data area contains values in double word format, make sure that the double word values are actually located in one of the maximum 12 double words of the data area to be acquired.

Distribution over two consecutive data double words could lead to the transfer of only one word of the double word value (high or low word) because a change has occurred in only that particular word. In this case, the missing word can lead to a data error on the receiving partner (applies to ST7cc, not for an S7 CPU).

Note

DB with standard access

The block has parameters of the "ANY" type. Therefore, leave the "Optimized block access" attribute in the properties of the DB disabled.

Parameters

General parameters:	**PartnerNo** **PartnerObjectNo** **Enabled** For a description, see section Reoccurring parameters (Page 24).	
Name:	**EnterInput**	
Declaration:	INPUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	FALSE	
Address range:	Input	I 0.0 ... I n.7
	Memory	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7

Explanation: Enter input

The transfer of the parameter values at the "ParameterInput" input can be triggered vis this input by a signal edge change.

A signal change at "EnterInput" is only taken into account when "ContinuousEnterFunct" = FALSE. If this condition is fulfilled, an edge change 0 → 1 causes the parameter values entered in "ParameterInput" to be adopted and transferred. There is no change check. The entire data area specified in "ParameterInput" is always transferred.

This method of triggering transfer is suitable for input vis hardware, for example via a console or control panel. For further information and related parameters, refer to the section "Function" above.

If you do not require the parameter, simply leave it open.

Data checks:

- The parameters that are read in are then transferred f no error is detected during the 1-out-of-n check, and if the central enable memory bit is set. This is automatically set by FC Safe following a selected time delay set there (see FC Safe, "InputDelayTime" parameter). The input area is then only read in again by the FB when a 0 signal was detected at EnterInput for at least one OB1 cycle.
- When a 1-out-of-n error is detected at the hardware input, the entered parameters are no longer processed. New parameters are read in again only when previously for the length of an OB1 cycle, no hardware input via a command, setpoint or parameter block was detected.

  The FB enters the detected 1-out-of-n error in the diagnostic buffer (event ID B172). As long as the error remains, the error status is indicated via the "InputError" output of FC Safe (see FC Safe, "InputError" parameter).

Name:	**ContinuousEnterFunct**
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Default:	FALSE

Explanation: Continuous change check

With this parameter, you can decide whether the parameter values at the "ParameterInput" input should be continuously read in and checked for changes. The change check is made by comparison with the last values that were transferred. Only changed values are sent. If more than one change is detected, the block sends the data area in which all changed parameter values are located.

The changed data area that is transferred consists of the first and the last double word in which a change was detected and all words located in between, even if these have not changed.

A new transfer of the parameter values can be triggered via the "RetransmitAll" input (see below) even when the parameter entries have not changed.

This method of transmission triggering is suitable when the parameter values are entered in the ParameterInput area by software, but can also be used for entering the parameters from an operator panel (OP).

For further information refer to the section "Function" above.

If you do not require the parameter, simply leave it open.

Data checks:

The parameters read in are only transmitted if no error is detected during the 1-out-of-n check.

- While for hardware input (see EnterInput) an empty cycle must be detected before new parameter values can be transferred from the block.
- For software input new parameter values can be transferred in every OB1 cycle. This assumes that there is no other software entry by another block pending in this cycle. Otherwise a 1-out-of-n error is detected.

  When a 1-out-of-n error is detected during the software input, the entered parameters are no longer processed. The FB enters the detected 1-out-of-n error in the diagnostic buffer (event ID B172).

  New parameters are read in again only when previously for the length of an OB1 cycle in the CPU, no software input via a command, setpoint or parameter block was detected.

Name:	ApplyRemoteParamAuto
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Default:	FALSE

Explanation: Automatic synchronization of the input area with the returned area.

With the parameter all the parameter values from the "ReturnedParameter" mirror back area are then copied to the "ParameterInput" area.

In addition to this the mirrored back parameter values are written the send mailbox of the communications DB.

Automatic synchronization is then always performed when new data is received from the partner object (Par12D_R).

If you do not require the parameter, simply leave it open.

Name: **ParameterInput**

Declaration: INPUT

Data type: ANY

Range of values: See address range

Default: P#P 0.0 VOID 0
(null pointer)

Address range: P#DBxx.DBX yy.0 DWORD zz

- xx: Data block number 1...32767
- yy: Byte number
- zz: Number of double words 1...12 starting at byte number yy

Example:

```
P#DB20.DBX100.0 DWORD 4
```

Note that the default value (null pointer) is not permitted. A pointer with a real address must be specified.

Explanation: Parameter input area.

The ANY pointer addresses the data area in which the parameter values to be acquired are located. This data area must be within a data block and its length can vary between 1 and 12 data double words.

For information on the content and formats, refer to the section "Function" above.

If the parameter assignment is incorrect (null pointer, length > 12, data area not a DB), an error message is entered in the diagnostics buffer (event ID B114, [Info2/3] = 11). The CPU does not change to STOP. The block is then no longer processed, however, until the error has been corrected.

How the parameters at ParameterInput are processed depends on whether they are hardware or software entries and how the transfer of this data area is triggered. You will find more information on this in the section "Function" above.

Name: **ReturnedParameter**

Declaration: OUTPUT

Data type: ANY

Range of values:	See address range
Default:	P#P 0.0 VOID 0 (null pointer)
Address range:	P#DBxx.DBX yy.0 DWORD zz • xx: Data block number 1...32767 • yy: Byte number • zz: Number of double words 1...12 starting at byte number yy Example: `P#DB20.DBX100.0 DWORD 4` Note that the preset value (null pointer) is not permitted. A pointer with a real address must be specified.
Explanation:	Parameter output area The partner object receiving the parameter values reports back the valid parameter values there. These values are displayed at the "ReturnedParameter" output. If the partner object is set to 'local' and a new input is made there, the parameters changed locally are indicated here by "ReturnedParameter". The ANY pointer defines the data area in which the received parameter values are output. This data area must be within a data block and its length can vary between 1 and 12 data double words. The length must be identical with the length set for ParameterInput. After startup of the local or partner CPU, or after restoring a connection, an automatic general request ensures that the current, local, valid parameters are indicated at "ReturnedParameter". If the parameter setting is incorrect (data area not a data block, length greater than 12 or length different from the length set for ParameterInput), an error message to this effect is entered in the diagnostics buffer (event ID B114, [Info2/3] = 11). The CPU does not change to STOP. The FB is then no longer processed, however, until the error has been corrected.

Name:	**LocalOperation**	
Declaration:	OUTPUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	FALSE	
Address range:	Output	Q 0.0 ... Q n.7
	Memory	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7

Explanation: Return message from partner object: Object is set to local operation.

At the partner object that receives the parameters a local setpoint entry can also be made.. The partner object Par12D_R must then be set to 'local' at the "Local" input parameter. The current status of the "Local" input parameter is reported by the partner object and indicated here at the "LocalOperation" output.

After startup of the local or partner CPU, or after restoring a connection, an automatic general request ensures that the local currently valid status is displayed at "LocalOperation".

Name: **NewData**

Explanation: For the declaration, data type, range of values, default and function, refer to the section Reoccurring parameters (Page 24).

Whenever the block has received new data and has output it to the output "ReturnedSetpoint" or "LocalOperation", the "NewData" output is set to TRUE for one OB1 cycle.

Name: **Release**

Declaration: IN_OUT

Data type: BOOL

Range of values: TRUE / FALSE

Default: FALSE

Address range:

Memory	M 0.0 ... M n.7
Data bit	DBm.DBX 0.0 ... n.7

This is an in/out parameter (declaration IN_OUT). It is difficult to specify local bit memory with this parameter type and this should not be used.

Explanation: Trigger input for sending the currently pending parameter values

You can use this input parameter when the parameter is entered by software, for example at an operator panel (OP). "Release" should then be set using a function key on the OP. You can then enter several parameters initially on the OP. The parameters are transferred only when the Release function key is activated.

A change check is performed only with signal 1 at the Release input. The data double words from the data area configured with "ParameterInput" that have changed since the last transfer are transferred.

If you always want to transfer the entire data area specified with "ParameterInput" and not only the changed parameter values, you should use the "RetransmitAll" input parameter instead of "Release".

The "Release" input is reset automatically. You should therefore only specify memory or data inputs as the input. The automatic reset would not work with a digital input.

Data checks: The same safety checks are carried out as with ContinuousEnterFunct, see above.

If you do not require the parameter, simply leave it open.

Name:	**RetransmitAll**	
Declaration:	IN_OUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	TRUE	
Address range:	Memory	M 0.0 ... M n.7
	Data bit	DBm.DBX 0.0 ... n.7
	This is an in/out parameter (declaration IN_OUT). It is difficult to specify local bit memory with this parameter type and this should not be used.	
Explanation:	Trigger input for retransferring the entire data area specified by "ParameterInput".	
	You can use this input parameter when the parameter is entered by software, for example at an operator panel (OP). "RetransmitAll" should then be set using a function key on the OP. When a 1 signal is detected at the "RetransmitAll" input, the entire data area specified by "ParameterInput" is transferred. A change check is not performed	
	The "RetransmitAll" input is reset automatically. You should therefore only specify memory or data inputs as the input. The automatic reset would not work with a digital input. Since there is no change check, this would lead to continuous transfer of all parameter values as long as the input has a 1 signal.	
	"RetransmitAll" can also be used as an option in addition to "Release" or "ContinuousEnterFunct" when new parameter values were entered but could not be transferred to the partner (disrupted connection or the partner object is set to 'local'). In this case you can then trigger repeated transfer of the entire data area specified by "ParameterInput" using "RetransmitAll". All changes that were previously entered but are not yet available at the partner are consistently included.	
	"RetransmitAll" can also be used as an independent transfer trigger when you always want to send all entries and not just those that have changed. In this case you can use "RetransmitAll" instead of "Release" that only sends the changed parameter values.	
	Data checks: The same safety checks are carried out as with "ContinuousEnterFunct", see above.	
	If you do not require the parameter, simply leave it open.	

Name:	**ApplyRemoteParamMan**	
Declaration:	IN_OUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	FALSE	
Address range:	This is an in/out parameter (declaration IN_OUT). It is difficult to specify local bit memory with this parameter type and this should not be used.	

Explanation: Trigger input for synchronization of the input area with the returned area.

The input triggers a one-time synchronization of the ParameterInput input area with the "ReturnedParameter" mirror back area. All the parameter values from the "ReturnedParameter" mirror back area are then copied to the "Parameter-Input" input area.

The send mailbox of the communications DB is also synchronized with the returned parameter values.

"ApplyRemoteParamMan" is reset automatically. You should therefore only specify memory or data inputs as the input. The automatic reset would not work with a digital input. The result would be a constant synchronization as long as the parameter has a 1 signal.

If you do not require the parameter, simply leave it open.

3.1.25 Parameter typical Par12D_FS

Validity

- S7-1500

Function

Send 1 to 12 parameter values (each 1 double word) without a 1-out-of-n check and receive the current, locally valid parameter values from the partner. This means that it is not necessary to call the FC Safe in the cyclic program.

There is less transmission security due to the lacking 1-out-of-n check.

No parameters are sent to subscribers that cannot be reached.

Frames of this block can be received and evaluated with the Par12D_R block.

The content per double word can be a value in double word format (DWORD, DINT, REAL).

Note

Remember word boundaries

When only changed data is transferred and the data area contains values in double word format, make sure that the double word values are actually located in one of the maximum 12 double words of the data area to be acquired.

Distribution over two consecutive data double words could lead to the transfer of only one word of the double word value (high or low word) because a change has occurred in only that particular word. In this case, the missing word can lead to a data error on the receiving partner (applies to ST7cc, not for an S7 CPU).

The data area to be transferred is specified for the "ParameterInput" parameter in the form of an Any pointer. This data area must be within a data block and its length can vary between 1 and 12 double words. The data area sent to the partner or the parameter values entered locally at the partner are returned from there and output here at the "ReturnedParameter" parameter. This output area (Any pointer) must also be within a data block and its length must match that specified for "ParameterInput".

Mirroring

Separate data areas are normally specified for "ParameterInput" and "ReturnedParameter". This makes it easy to recognize what was most recently entered and what is locally valid.

However, the same data area can also be specified for both parameters. Both areas then overlap 100 % and are then always synchronized. In this case, you can no longer distinguish the difference between what has been entered most recently and what is locally valid.

Even when separate areas are specified for "ParameterInput" and "ReturnedParameter", it is still possible to ensure that the "ParameterInput" input area is always synchronized with the mirrored back values of "ReturnedParameter". You can perform this synchronization automatically by setting the "ApplyRemoteParam" parameter to TRUE.

A parameter can also be set locally at the partner object that receives the parameter. The "Local" input parameter must then be set to TRUE at the partner object. The locally valid parameters can be created by the partner object at the "LocalParameterInput" input parameter; they are then output here to "ReturnedParameter". The local status can be reported by the partner object via the "Local" parameter and displayed here at the "LocalOperation" output. The partner object can be set so that it does not accept any parameters from Remote when it is working in local mode (see Par12D_R block).

Triggering sending

Sending the data area defined at "ParameterInput" can be triggered by the following parameters:

- ContinuousEnterFunct

  Automatic transmission in case of a change in the area "ParameterInput" in the software mode of the block (HWmode = FALSE)

- Release

  Single triggered transmission in hardware and software mode of the device

Note

DB with standard access

The block has parameters of the "ANY" type. Therefore, leave the "Optimized block access" attribute in the properties of the DB disabled.

Parameters

Parameter: **PartnerNo**
PartnerObjectNo
Enabled
For a description, see section Reoccurring parameters (Page 24).

Parameter: **HWmode**
Declaration: INPUT
Data type: BOOL

Range of values:	TRUE / FALSE
Default:	FALSE
Explanation:	The parameter specifies whether the block works in hardware mode (TRUE) or software mode (FALSE).

- **Hardware mode**

  Transmission can be triggered by a digital input module connected to the "Release" input.

  The parameter value is applied in hardware mode when the signal at the release input changes from 0 to 1. Changes at the "ParameterInput" input while "InputDelayTime" is running are not sent.
  In hardware mode, the complete data area defined with "ParameterInput" is sent.

  In hardware mode, an empty cycle must be detected at the "Release" input before new parameter values can be sent by the block.

- **Software mode**

  Sending can be triggered via the "Release" input or via the "ContinuousEnterFunct" function in software mode.

  New parameter values can be sent in each OB1 cycle with a software input. The "ContinousEnterFunct" function can be switched on and off as required in the cycle.

  Note: Changed data areas

  In software mode, the block transmits selectively, i.e. it only transmits the area of double words in which a change was detected.
  Exception: If no change is detected when the "Release" input is actuated, it sends the complete parameter area.

  The changed data area consists of the first and the last double word in which a change was detected and all words located in between, even if these have not changed.

  Example:
  The area to be read in is 10 double words long. In this case, changes were detected in the second, fifth and eighth double words. The transferred area is therefore from the 2nd to the 8th double word.

The "**HWmode**" parameter is applied during startup and cannot be changed in the cycle.

Parameter:	**ContinuousEnterFunct**
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE

Default:	FALSE
Explanation:	Continuous change check The parameter is not evaluated if HWmode equals TRUE. Set the parameter to TRUE if you enter the parameters using software or, for example, via an operator panel (OP). There is a constant check for changes. The change check is made by comparison with the last values that were transferred. As soon as a change is detected in the data area defined with "ParameterInput", this results in transmission. The changed values are sent. If several changes are detected, the block sends the data area from the first to the last double word in which a change was detected. A new transfer of the parameter values can be triggered via the "Release" input (see below) even when the parameter entries have not changed. If you do not require the parameter, simply leave it open.

Parameter:	**ApplyRemoteParam**	
Declaration:	INPUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	FALSE	
Address range:	Input	I 0.0 ... I n.7
	Memory bit	M 0.0 ... M n.7
	Data bit	DBm.DBX 0.0 ... n.7
Explanation:	If the parameter has the value "TRUE", an automatic synchronization of the input area with the returned area is performed. With the parameter all the parameter values from the "ReturnedParameter" mirror back area are then copied to the "ParameterInput" area. In addition to this the mirrored back parameter values are written the send mailbox of the communications DB. Automatic synchronization is then always performed when new data is received from the partner object (Par12D_R). Synchronization can also be triggered once with a signal transition from 0 to 1, e.g. manually via a pushbutton. If the parameter has the value "FALSE" or the parameter is not specified, no automatic synchronization of the input area with the returned area is performed.	

Parameter:	**ParameterInput**
Declaration:	INPUT
Data type:	ANY
Range of values:	See address range

Default:	P#P 0.0 VOID 0 (null pointer)
Address range:	P#DBxx.DBX yy.0 DWORD zz • xx: Data block number 1...32767 • yy: Byte number • zz: Number of double words 1...12 starting at byte number yy Example: `P#DB20.DBX 100.0 DWORD 4` Remember the periods and spaces when entering the pointer! Note that the default value (null pointer) is not permitted. A pointer with a real address must be specified.
Explanation:	Parameter input area. The ANY pointer addresses the data area in which the parameter values to be acquired are located. This data area must be within a data block and its length can vary between 1 and 12 data double words. For information on the content and formats, refer to the section "Function" above. If the parameter assignment is incorrect (null pointer, length > 12, data area not a DB), an error message is entered in the diagnostics buffer (event ID B114, [Info2/3] = 7). The CPU does not change to STOP. The block is then no longer processed, however, until the error has been corrected. The way in which the values present at "ParameterInput" are subsequently processed depends on whether it is a hardware or a software input and how the transmission is triggered. You can find more information on this in the "Function" section above and under the "HWmode" parameter.

Parameter:	**ReturnedParameter**
Declaration:	OUTPUT
Data type:	ANY
Range of values:	See address range
Default:	P#P 0.0 VOID 0 (null pointer)
Address range:	P#DBxx.DBX yy.0 DWORD zz • xx: Data block number 1...32767 • yy: Byte number • zz: Number of double words 1...12 starting at byte number yy Example: `P#DB20.DBX 100.0 DWORD 4` Remember the periods and spaces when entering the pointer! Note that the preset value (null pointer) is not permitted. A pointer with a real address must be specified.

Explanation: Parameter output area

The partner object receiving the parameter values reports back the valid parameter values there. These values are displayed at the "ReturnedParameter" output. If the partner object is set to 'local' and a new input is made there, the parameters changed locally are indicated here by "ReturnedParameter".

The ANY pointer defines the data area in which the received parameter values are output. This data area must be within a data block and its length can vary between 1 and 12 data double words. The length must be identical with the length set for ParameterInput.

After startup of the local or partner CPU, or after restoring a connection, an automatic general request ensures that the current, local, valid parameters are indicated at "ReturnedParameter".

If the parameter setting is incorrect (data area not a data block, length greater than 12 or length different from the length set for ParameterInput), an error message to this effect is entered in the diagnostics buffer (event ID B114, [Info2/3] = 8). The CPU does not change to STOP. The FB is then no longer processed, however, until the error has been corrected.

Name: **InputDelayTime**

Declaration: INPUT

Data type: INT

Range of values:
- 0 or open

  When the parameter is not required
- 1 ... 32000

  Value range for delay time (milliseconds) in hardware mode

Explanation: Delay time in milliseconds for the "Release" input when the block is operating in hardware mode.

A delay time of at least 1000 ms is recommended.

Changes to "ParameterInput" while "InputDelayTime" is running are not sent in the current cycle.

Name: **MaxInputTime**

Declaration: INPUT

Data type: INT

Range of values:
- 0 or open

  When the parameter is not required.
- 1 ... 32000

  Value range for monitoring time (seconds) "HWmode" = TRUE.

Explanation: Monitoring time in seconds for the "Release" input if the block is operating in hardware mode.

A monitoring time of at least 30 s is recommended.

If you do not require the parameter, leave it open.

Parameters:	**Release**	
Declaration:	IN_OUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	FALSE	
Address range:	Input	I 0.0 ... I n.7
	Memory	M 0.0 ... M n.7
	Data bit	DBm.DBX 0.0 ... n.7
	This is an in/out parameter (declaration IN_OUT). It is difficult to specify local bit memory with this parameter type and this should not be used.	
Explanation:	Trigger input for sending the currently pending parameter values	
	You can use the parameter in software or hardware mode of the block.	

- **Hardware mode**

  If you use the input via an OR, it should be operated via a function key of the OP. You can then enter several parameters initially on the OP. The parameters are transferred only when the Release function key is activated.

  In hardware mode, the "Release" input is not automatically reset. You should therefore only specify memory or data inputs as the input. The automatic reset would not work with a digital input.

  The "Release" input is subject to the "InputDelayTime" and is monitored via "MaxInputTime". If the input is set longer than "MaxInputTime, the timeout is output at "InputError". The "InputError" output does not change back to "0" until the "Release" input is also = 0. (See also "InputError")

- **Software mode**

  A change check is performed only with signal 1 at the Release input. The data from the "ParameterInput" area that has changed since the last transmission is sent. If no change is detected when activating the "Release" input, the block transmits the complete parameter area.

  In software mode, the "Release" input is automatically reset. You should therefore only specify memory or data inputs as the input.

If you do not require the parameter, simply leave it open.

Parameter:	**LocalOperation**
Declaration:	OUTPUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Default:	FALSE

Address range:	Output	Q 0.0 ... Q n.7
	Memory bit	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7

Explanation: Return message from partner object: Object is set to local operation.

A parameter can also be set locally at the partner object that receives the parameter. The partner object Par12D_R must then be set to 'local' at the "Local" input parameter. The current status of the "Local" input parameter is reported by the partner object and indicated here at the "LocalOperation" output.

After startup of the local or partner CPU, or after restoring a connection, an automatic general request ensures that the local currently valid status is displayed at "LocalOperation".

Parameter: **NewData**

Explanation: For the declaration, data type, range of values, default and function, refer to the section Reoccurring parameters (Page 24).

Whenever the block has received new data and has output it to the output "ReturnedParameter" or "LocalOperation", the "NewData" output is set to TRUE for one OB1 cycle.

Name: **InputOK**

Declaration: OUTPUT

Data type: BOOL

Range of values:	Output	Q 0.0 ... Q n.7
	Memory	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7

Explanation: The display "InputOK" is set for one cycle at the entry of the data frame into the send buffer. The display is independent of the "HWmode" parameter.

If you do not require the parameter, leave it open.

Name: **InputError**

Declaration: OUTPUT

Data type: BOOL

Range of values:	Output	Q 0.0 ... Q n.7
	Memory	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7

Explanation: Display of input errors at the "Release" input, if "HWmode" equals TRUE.

The output has a 1 signal as soon as the "Release" command lasts longer than defined via MaxInputTime. The "InputError" output is only set to 0 again when the release input has a 0 signal. This can be used, for example, to detect that the input is defective.

If you do not require the parameter, leave it open.

3.1.26 Parameter typical Par12D_R

Function

Receive 1 to 12 parameter values (each 1 double word) or enter locally and send back the current, locally valid parameter values to the partner.

The block can receive and output data of the "Par12D_S" and "Par12D_FS" typicals.

The content of each double word may be a value in double word (DWORD, DINT, REAL) format, it can also be a mixture of other data types which together form a double word, for example:

- 4 bytes
- 2 words
- 2 bytes + 1 word

See also the note "Note word boundaries" below.

The data area to be transferred is specified for the "ParameterOutput" parameter in the form of an Any pointer. This data area must be within a data block and its length can vary between 1 and 12 double words.

You can also use the block to enter the parameter values locally. The input area for this is specified as an Any pointer with the "LocalParameterInput" parameter. It must be located within a data block and its length must be identical to the length configured in the "ParameterOutput" parameter.

The block only processes the changed data area. In response to a general or single request, on the other hand, the entire parameter set is transferred or mirrored back..

Bumpless switchover between the "Local" and "Remote" operating modes is guaranteed.

Note

Changed data areas

The changed data area consists of the first and the last double word in which a change was detected and all words located in between, even if these have not changed.

Example:
The area to be read in is 10 double words long. In this case, changes were detected in the second, fifth and eighth double words. The transferred area is therefore from the 2nd to the 8th double word.

Remember word boundaries

When only changed data is transferred and the data area contains values in double word format, make sure that the double word values are actually located in one of the maximum 12 double words of the data area to be acquired.

Distribution over two consecutive data double words could lead to the transfer of only one word of the double word value (high or low word) because a change has occurred in only that particular word. In this case, the missing word can lead to a data error on the receiving partner (applies to ST7cc, not for an S7 CPU).

Note

DB with standard access

The block has parameters of the "ANY" type. Therefore, leave the "Optimized block access" attribute in the properties of the DB disabled.

Parameters

General parameters:	**PartnerNo** **PartnerObjectNo** **Enabled** **ImageMemory** **Conditional** **Unconditional** For a description, see section Reoccurring parameters (Page 24).	
Name:	**Local**	
Declaration:	INPUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	FALSE	
Address	Input	I 0.0 ... I n.7

range:	Memory	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7

Explanation: Local parameter input released

Release of the local parameter input of the data area specified by "LocalParameterInput".

A setpoint sent by the partner (master station) is not accepted by the object as long as "Local" = TRUE.

The current status of the "Local" input is transferred to the partner.

Bumpless switchover:

- When there is a switchover from "Local" = 0 to 1, the last values output at the "ParameterOutput" output are retained until new parameter values are entered via the local input area "LocalParameterInput".
- When there is a switchback from Local = 1 to 0, the last values at the "ParameterOutput" output are retained until the block receives new parameter values from the partner.

Read the note on the "ContinuousEnterFunct" parameter.

Special situation:

You can also enter the parameter values during local input directly in the output area specified by "ParameterOutput". Either you do not specify an input area for "LocalParameterInput" or you specify the same data area both for "LocalParameterInput" and "ParameterOutput".

This type of parameter input cannot be prevented by the "Local" input. Regardless of the status of the "Local" parameter, the values entered in the output area are sent immediately by the block to the partner.

Local parameter entries can therefore be made regardless of the status of the "Local" input. "Local" only influences the acceptance of parameters sent by the partner:

- Local = 0

  The parameters sent by the partner are accepted and output to the "ParameterOutput" data area.
- Local = 1

  The parameters sent by the partner are rejected.

In this special situation, the "Release" and "ContinuousEnterFunct" have no function.

A status change of the "Local" parameter is always transferred by the module according to the send buffer principle, even when the parameter "ImageMemory" = TRUE. This ensures that the optional synchronization of the input and output area on the partner is always performed correctly (see Par12D_S, parameters "ApplyRemoteParamMan" and "ApplyRemoteParamAuto").

Name: **ContinuousEnterFunct**

Declaration: INPUT

Data type: BOOL

Range of values:	TRUE / FALSE
Default:	FALSE
Explanation:	Continuous local parameter acquisition. With this parameter, you can decide whether the values in the "LocalParameterInput" input area should be continuously read in and checked for changes. The change is checked by comparing the current values at the "ParameterOutput" output. Changes in the input area are copied immediately to the output area and transferred to the partner object. Only changed values are sent. If there is more than one change, the block sends the data area in which all changed parameter values are located. The "ContinuousEnterFunct" = TRUE setting only takes effect when the following conditions are met: • An input area is defined by "LocalParameterInput" and this is not identical to the output area set for "ParameterOutput". and • there is a 1 signal at the "Local" input (= TRUE). With the setting "ContinuousEnterFunct" = TRUE, when the signal 1 is detected at the "Local" input, the values pending at "LocalparameterInput" are adopted immediately and output at the "ParameterOutput" output. The condition is that the local input values differ from the currently output values at this time. This method of acquisition of local values is suitable when the values are entered in the "LocalParameterInput" area by software. If you do not require the parameter, simply leave it open.

Name:	**LocalParameterInput**
Declaration:	INPUT
Data type:	ANY
Range of values:	See address range
Default:	P#P 0.0 VOID 0 (null pointer)
Address range:	P#DBxx.DBX yy.0 DWORD zz • xx: Data block number 1...32767 • yy: Byte number • zz: Number of double words 1...12 starting at byte number yy Example: `P#DB20.DBX100.0 DWORD 4` Note that the default value (null pointer) is not permitted. A pointer with a real address must be specified.

Explanation:	Local parameter input area The ANY pointer addresses the data area in which the parameter values to be acquired are located. This data area must be within a data block and its length can vary between 1 and 12 data double words. The length must be identical to the length specified for" ParameterOutput". For information on the content and formats, refer to the section "Function" above. If the parameter assignment is incorrect (null pointer, length > 12, data area not a DB), an error message is entered in the diagnostics buffer (event ID B114, [Info2/3] = 11). The CPU does not change to STOP. The block is then no longer processed, however, until the error has been corrected. If you do not require the parameter, simply leave it open.

Name:	**ParameterOutput**
Declaration:	INPUT
Data type:	ANY
Range of values:	See address range
Default:	P#P 0.0 VOID 0 (null pointer)
Address range:	P#DBxx.DBX yy.0 DWORD zz • xx: Data block number 1...32767 • yy: Byte number • zz: Number of double words 1...12 starting at byte number yy Example: `P#DB20.DBX100.0 DWORD 4` Remember the periods and spaces when entering the pointer! Note that the preset value (null pointer) is not permitted. A pointer with a real address must be specified.
Explanation:	Parameter output area The ANY pointer addresses the data area in which the locally entered parameter values or those received from the partner are output. This data area must be within a data block and its length can vary between 1 and 12 double words. For information on the content and formats, refer to the section "Function" above. Par12D saves the received data without further processing in the data area specified by "ParameterOutput". You need to evaluate and process the received data with the user program.

When only changed data is sent by the partner object Par12D_S, it is possible that only part of the data output area is newly written. this is the area in which the changes were detected at the acquisition end

If the parameter assignment is incorrect (null pointer, length > 12, data area not a DB), an error message is entered in the diagnostics buffer (event ID B114, [Info2/3] = 11). The CPU does not change to STOP. The block is then no longer processed, however, until the error has been corrected.

Name:	**NewData**
Explanation:	For the declaration, data type, range of values, default and function, refer to the section Reoccurring parameters (Page 24). Whenever the block has received new parameter values from the partner object and has output them to the output field "ParameterOutput", the "NewData" output is set to TRUE for one OB1 cycle. This also applies when there is new local input in the status "Local" = 1.

Name:	**Release**	
Declaration:	IN_OUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	FALSE	
Address range:	Input	I 0.0 ... I n.7
	Memory	M 0.0 ... M n.7
	Data bit	DBm.DBX 0.0 ... n.7
	This is an in/out parameter (declaration IN_OUT). It is difficult to specify local bit memory with this parameter type and this should not be used.	
Explanation:	Input for the acceptance of local parameter input. The acceptance of the parameter values at the "LocalParameterInput" parameter input can be triggered via this input by a signal edge change. A change from 0 to 1 at the "Release" input is taken into account only when the following conditions are met: • An input area is specified by the "LocalParameterInput" parameter and this is not identical to the output area specified by "ParameterOutput" and • The "Local" input is set to TRUE.	

You can use "Release" for parameter input via software, e.g. via an operator panel (OP). The "Release" input should then be set using a function key on the OP. You can then enter several parameters initially on the OP. The parameter values are read in and checked for changes only when the Release function key is activated.

The change is checked by comparing the current parameter values at the "ParameterOutput" output. Changes in the input area are then copied immediately to the output area and transferred to the partner.

Only changed values are sent. If there is more than one change, the block sends the data area in which all changed parameter values are located.

The "Release" input is reset automatically. Instead of a memory bit or data bit, a digital input can also be specified as the input. The automatic reset would not work with a digital input. This does not, however, have negative effects. Acquisition via "Release" is edge triggered, in other words only once.

If you do not require the parameter, simply leave it open.

3.1.27 FC Safe

Function

The FC Safe ensures reliable input of commands and setpoints by using the 1-out-of-n check.

The FC Safe checks th einput for the following data point typicals:

- Cmd01B_S
- Set01W_S
- Par12D_S

If an input is pending, the FC checks whether there is only one entry pending in the current OB1 cycle and then enables the block reading in. As soon as two (or more) entries are pending in an OB1 cycle, both entries are ignored.

In every CPU in which commands and/or setpoints are acquired, the FC Safe needs to be called in OB1 to complete of all command and setpoint FBs.

The FC has separate monitoring for entries via different system memory areas. They are divided into hardware entries and software entries:

- Hardware entries
  - Input modules (I)
  - Peripheral inputs (PI)
- Software entries
  - Bit memory (M)
  - Data blocks (DB)
  - Local data (L)
  - Operator panels

The two types of entry "hardware entries" and "Software entries" are checked separately by the block. The FC enables the hardware and software entries separately. For each type of entry, only one command or setpoint entry may be detected.

If a single hardware entry and a single software entry are pending at the same time both entries are enabled.

For hardware entries, the following additional condition applies: The entry must be pending constantly for the duration of "InputDelayTime". Only when an entered command or setpoint is pending unchanged for this time and during this time no other command or setpoint entry is detected is the block processing enabled.

The actual putting together of the command or setpoint frame is handled by the block that read in the command or setpoint.

For the hardware entry the FC Safe provides the following two display bits:

- InputOK

  Display of the enabling of hardware commands and setpoints
- InputError

  Display of entry errors with hardware commands and setpoints

FC-Safe shows a command output error detected in a station via the following output:

- GlobalCmdOutputError

  Group message: Command output error in a station

Parameters

Name:	**InputDelayTime**
Declaration:	INPUT
Data type:	INT

Range of values:	• 0 Value for unrequired parameter • 1 ... 32000 Range of values for delay time
Explanation:	Delay time in ms for commands and setpoints that are input via hardware. A delay time of at least 1000 ms is recommended.

Name:	**MaxInputTime**
Declaration:	INPUT
Data type:	INT
Range of values:	• 0 Value for unrequired parameter • 1 ... 32000 Range of values for monitoring time
Explanation:	Monitoring time in ms for commands and setpoints that are input via hardware. A monitoring time of at least 30 s is recommended. If you do not require the parameter, specify 0 (zero).

Name:	**ResetError**	
Declaration:	INPUT	
Data type:	BOOL	
Range of values:	Input	I 0.0 ... I n.7
	Memory	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7
Explanation:	Input for resetting the output GlobalCmdOutputError. If you do not require the parameter, specify a memory or data bit that always has signal 0.	

Name:	**InputOK**	
Declaration:	• S7-300 • S7-1500	• OUTPUT • IN_OUT
Data type:	BOOL	
Range of values:	Output	Q 0.0 ... Q n.7
	Memory	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7

Explanation:	Display of the enabling of hardware commands and setpoints Has a 1 signal as soon as the current entry was enabled, i.e. when the hardware command or setpoint was entered correctly. The display bit goes off when the entry is reset, in other words when the command key is released or the setpoint entry key "EnterInput" input is released. If you do not require the parameter, specify a memory or data bit in the memory area of the local data.	
Name:	**InputError**	
Declaration:	• S7-300 • S7-1500	• OUTPUT • IN_OUT
Data type:	BOOL	
Range of values:	Output	Q 0.0 ... Q n.7
	Memory	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7
Explanation:	Display of entry errors with hardware commands and setpoints The output has a 1 signal when one of the following hardware entry errors is detected within the monitoring time "MaxInputTime": • Two or more command and/or setpoint entries were detected at the same time. • At one of the inputs a 1 signal was detected over a longer period of time (for example when the input is defective). If you do not require the parameter, specify a memory or data bit in the memory area of the local data.	
Name:	**GlobalCmdOutputError**	
Declaration:	• S7-300 • S7-1500	• OUTPUT • IN_OUT
Data type:	BOOL	
Range of values:	Output	Q 0.0 ... Q n.7
	Memory	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7

Explanation: Group message: In a station a command output error was detected.

A command output error can occur at the receive end only in the following cases:

- The content of the two command bytes in the received frame is not identical.
- More than one bit is set in the command byte (1-out-of-8 error).

If such an error is detected, this is sent by the station with an organizational frame to the subscriber that sent the commands. FC Safe on the sending subscriber then indicates the disruption at the output GlobalCmdOutputError.

When an error is detected, the output remains set to the 1 signal until you request the group signal to be reset via the "ResetError" input.

If you do not require the parameter, specify a memory or data bit in the memory area of the local data.

3.1.28 Setpoint typical Set01W_S

Function

Send 1 setpoint as a 16 bit value and receive local setpoint from partner

The block operates with 1-out-of-n test. The 1-out-of-n check is performed by FC Safe.

Note

FC Safe required

With Set01W_S, data can only be transferred when the block FC Safe is linked in the end of the cyclic program, see section FC Safe (Page 118).

Parameters

General parameters: **PartnerNo**
PartnerObjectNo
Enabled
For a description, see section Reoccurring parameters (Page 24).

Name:	**EnterInput**	
Declaration:	INPUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	FALSE	
Address range:	Input	I 0.0 ... I n.7
	Memory	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7

Explanation:	Enter input for hardware setpoint The adoption of a setpoint at the "SetpointInput" can be triggered by a signal edge change. A signal change at "EnterInput" is only taken into account when "ContinuousEnterFunct" = FALSE. If this condition is fulfilled, an edge change 0 → 1 causes the setpoint entered in "SetpointInput" to be adopted and transferred. This also applies when the newly entered setpoint is identical to the last setpoint transferred. This method of setpoint adoption is suitable for input vis hardware, for example via a console or control panel. It can also be used for entering setpoints at an operator panel (OP). In this the adoption must be triggered by a function key on the OP. If you do not require the parameter, simply leave it open.

Name:	**ContinuousEnterFunct**
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Default:	FALSE
Explanation:	Apply setpoint continuously for software setpoint With this parameter, you specify whether the setpoint at "SetpointInput" should be continuously read in and checked for changes. Changes are checked by comparison with the last setpoint that was transferred. This method of setpoint adoption is suitable for input via suitable software. It can, however, also be used for entering setpoints at an operator panel (OP) if the OP does not have a separate function key that can be used to trigger the input. If you do not require the parameter, simply leave it open.

Name:	**SetpointInput**	
Declaration:	INPUT	
Data type:	WORD	
Range of values:	See address range	
Default:	0 (W#16#0)	
Address range:	Input words	IW0 ... IWn PIW0 ... PIWn
	Memory words	MW0 ... MWn LW0 ... LWn
	Data words	DBm.DBW0 ... n

Explanation: Setpoint input

How a setpoint available at SetpointInput is processed depends on whether it is a hardware or software input. You specify the type of input with the "ContinuousEnterFunct" parameter:

- **ContinuousEnterFunct = FALSE (hardware input)**

  A setpoint at "SetpointInput" is only read in as long as the signal is 1 at "EnterInput". The setpoint that is read in is then transferred if no error is detected during the 1-out-of-n check, and if the central enable memory bit is set. This is automatically set by FC Safe following a selected time delay set there (see FC Safe, "InputDelayTime" parameter).

  A further setpoint is then only read in again by the FB when a 0 signal was detected at EnterInput for at least one OB1 cycle.

  When a 1-out-of-n error is detected at the hardware input, the entered setpoint is no longer processed. A new setpoint is read in again only when previously for the length of an OB1 cycle in the CPU, no hardware input via a command, setpoint or parameter block was detected.

  The FB enters the detected 1-out-of-n error in the diagnostic buffer (event ID B172). As long as the error remains, the error status is indicated via the "InputError" output of FC Safe (see FC Safe, "InputError" parameter).

- **ContinuousEnterFunct = TRUE (software input)**

  A setpoint at SetpointInput is read in continuously and checked for changes. Changes are checked by comparison with the last setpoint that was transferred. The setpoint is sent immediately every time a change occurs unless the 1-out-of-n check detects an error.
  - With hardware input (see EnterInput) an empty cycle must be detected before a new setpoint can be transferred by the block.
  - With software input a new setpoint can be transferred in every OB1 cycle. This assumes that there is no other software entry by another block pending in this cycle. Otherwise a 1-out-of-n error is detected.

    A new transfer of the software setpoint can be triggered via the "SendSoftSetpoint" input even when the software input has not changed (see below).

    When a 1-out-of-n error is detected at the software input, the entered setpoint is no longer processed.

    A new setpoint is read in again only when previously for the length of an OB1 cycle in the CPU, no software input (command or setpoint) was detected. The block enters detected 1-out-of-n errors in the diagnostics buffer (event ID B172). Appropriate error bits are also set in the central data block BasicData where they can be queried by the software. For further details, refer to the description of FC Safe.

Name: **ReturnedSetpoint**
Declaration: OUTPUT
Data type: WORD

Range of values:	See address range	
Default:	0 (W#16#0)	
Address range:	Output words	QW0 ... QWn PQW0 ... PQWn
	Memory words	MW0 ... MWn LW0 ... LWn
	Data words	DBm.DBW0 ... n
Explanation:	Output for a returned setpoint The partner object receiving the setpoint reports back the currently valid setpoint there. This value is displayed at the "ReturnedSetpoint" output. If the partner object is set to 'local' and a new input is made there, the setpoint changed locally is indicated here by "ReturnedSetpoint". After startup of the local or partner CPU, or after restoring a connection, an automatic general request ensures that the locally valid setpoint is indicated by "ReturnedSetpoint". If you do not require the parameter, simply leave it open.	

Name:	**LocalOperation**	
Declaration:	OUTPUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	FALSE	
Address range:	Output	Q 0.0 ... Q n.7
	Memory	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7
Explanation:	Return message from the partner object: The object is set to local operation. A setpoint can also be set locally at the partner object that receives the setpoint. The partner object Set01W_R must then be set to ’local’ at the "Local" input parameter. The current status of the "Local" input parameter is reported by the partner object and indicated here at the "LocalOperation" output. After startup of the local or partner CPU, or after restoring a connection, an automatic general request ensures that the local currently valid status is displayed at "LocalOperation". If you do not require the parameter, simply leave it open.	

Name:	**NewData**	
Explanation:	For the declaration, data type, range of values, default and function, refer to the section Reoccurring parameters (Page 24). Whenever the block has received new data and has output it to the output "ReturnedSetpoint" or "LocalOperation", the "NewData" output is set to TRUE for one OB1 cycle.	

Name:	**SendSoftSetpoint**	
Declaration:	IN_OUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	FALSE	
Address range:	Memory	M 0.0 ... M n.7
	Data bit	DBm.DBX 0.0 ... n.7
	This is an in/out parameter (declaration IN_OUT). It is difficult to specify local bit memory with this parameter type and this should not be used.	
Explanation:	Trigger input for resending the last software setpoint. For further details, refer to the "SetpointInput" parameter. If you do not require the parameter, simply leave it open.	

3.1.29 Setpoint typical Set01W_FS

Function

Send 1 setpoint as a 16 bit value and receive local setpoint from partner

Frames of this block can be received and evaluated with the Set01W_R block.

Sending the setpoint can be triggered by the following parameters:

- ContinuousEnterFunct

  Automatic transmission in case of a change in the area "ParameterInput" in the software mode of the block (HWmode = FALSE)

- Release

  Single triggered transmission in hardware and software mode of the device

No setpoints are sent to stations which cannot be reached.

Parameters

Parameter: **PartnerNo**
PartnerObjectNo
Enabled
For a description, see section Reoccurring parameters (Page 24).

Parameter: **HWmode**
Declaration: INPUT
Data type: BOOL
Range of values: TRUE / FALSE
Default: FALSE
Explanation: The parameter specifies whether the block works in hardware mode (TRUE) or software mode (FALSE).

- **Hardware mode**

  Transmission can be triggered by a digital input module connected to the "Release" input.

  The setpoint is applied in hardware mode when the signal at the release input changes from 0 to 1.

  Changes made to the setpoint input "SetpointInput" while the "InputDelayTime" is running are not sent.

  In hardware mode, an empty cycle must be detected before new parameter values can be sent by the block.
- **Software mode**

  Sending can be triggered via the "Release" input or via the "ContinuousEnterFunct" function in software mode.

  New parameter values can be sent in each OB1 cycle with a software input. The "ContinousEnterFunct" function can be switched on and off as required in the cycle.

The "**HWmode**" parameter is applied during startup and cannot be changed in the cycle.

Parameter: **ContinuousEnterFunct**
Declaration: INPUT
Data type: BOOL
Range of values: TRUE / FALSE

Default:	FALSE
Explanation:	Apply setpoint continuously for software setpoint With this parameter, you specify whether the setpoint at "SetpointInput" should be continuously read in and checked for changes. Changes are checked by comparison with the last setpoint that was transferred. This method of setpoint adoption is suitable for input via suitable software. It can, however, also be used for entering setpoints at an operator panel (OP) if the OP does not have a separate function key that can be used to trigger the input. If you do not require the parameter, simply leave it open.

Parameter:	**SetpointInput**	
Declaration:	INPUT	
Data type:	WORD	
Range of values:	See address range	
Default:	0 (W#16#0)	
Address range:	Input words	IW0 ... IWn PIW0 ... PIWn
	Memory words	MW0 ... MWn LW0 ... LWn
	Data words	DBm.DBW0 ... n
Explanation:	Setpoint input The way in which the value at "SetpointInput" is subsequently processed depends on whether it is a hardware or a software input and how transmission is triggered. You can find more information on this in the "Function" section above and under the "HWmode" parameter.	

Name:	**InputDelayTime**
Declaration:	INPUT
Data type:	INT
Range of values:	• 0 or open Value for unrequired parameter • 1 ... 32000 Value range for delay time (milliseconds) in hardware mode
Explanation:	Delay time in milliseconds for the "Release" input when the block is operating in hardware mode. A delay time of at least 1000 ms is recommended. Changes to "SetpointInput" while "InputDelayTime" is running are not sent in the current cycle.

Name:	**MaxInputTime**
Declaration:	INPUT
Data type:	INT
Range of values:	• 0 or open When the parameter is not required. • 1 ... 32000 Value range for monitoring time (seconds) "HWmode" = TRUE.
Explanation:	Monitoring time in seconds for the "Release" input if the block is operating in hardware mode. A monitoring time of at least 30 s is recommended. If you do not require the parameter, leave it open.

Parameters:	**Release**	
Declaration:	IN_OUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	FALSE	
Address range:	Input	I 0.0 ... I n.7
	Memory	M 0.0 ... M n.7
	Data bit	DBm.DBX 0.0 ... n.7
	This is an in/out parameter (declaration IN_OUT). It is difficult to specify local bit memory with this parameter type and this should not be used.	

Explanation: Trigger input for sending the current setpoint

You can use the parameter in software or hardware mode of the block.

- **Hardware mode**

  If you use the input via an OR, it should be operated via a function key of the OP. You can then first enter the setpoint at the OP. The value is only be sent when you press the release function key.

  In hardware mode, the "Release" input is not automatically reset. You should therefore only specify memory or data inputs as the input. The automatic reset would not work with a digital input.

  The "Release" input is subject to the "InputDelayTime" and is monitored via "MaxInputTime". If the input is set longer than "MaxInputTime, the timeout is output at "InputError". The "InputError" output does not change back to "0" until the "Release" input is also = 0. (See also "InputError")

- **Software mode**

  A change check is performed only with signal 1 at the Release input. Sending occurs if the value at "SetpointInput" has changed since the last transmission.

  In software mode, the "Release" input is automatically reset. You should therefore only specify memory or data inputs as the input.

If you do not require the parameter, simply leave it open.

Parameter:	**ReturnedSetpoint**	
Declaration:	OUTPUT	
Data type:	WORD	
Range of values:	See address range	
Default:	0 (W#16#0)	
Address range:	Output words	QW0 ... QWn PQW0 ... PQWn
	Memory words	MW0 ... MWn LW0 ... LWn
	Data words	DBm.DBW0 ... n

Explanation: Output for a returned setpoint

The partner object receiving the setpoint reports back the currently valid setpoint there. This value is displayed at the "ReturnedSetpoint" output.

If the partner object is set to 'local' and a new input is made there, the setpoint changed locally is indicated here by "ReturnedSetpoint".

After startup of the local or partner CPU, or after restoring a connection, an automatic general request ensures that the locally valid setpoint is indicated by "ReturnedSetpoint".

If you do not require the parameter, simply leave it open.

Parameter:	**LocalOperation**	
Declaration:	OUTPUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	FALSE	
Address range:	Output	Q 0.0 ... Q n.7
	Memory bit	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7
Explanation:	Return message from the partner object: The object is set to local operation. A setpoint can also be set locally at the partner object that receives the setpoint. The partner object Set01W_R must then be set to 'local' at the "Local" input parameter. The current status of the "Local" input parameter is reported by the partner object and indicated here at the "LocalOperation" output. After startup of the local or partner CPU, or after restoring a connection, an automatic general request ensures that the local currently valid status is displayed at "LocalOperation". If you do not require the parameter, simply leave it open.	

Parameter:	**NewData**
Explanation:	For the declaration, data type, range of values, default and function, refer to the section Reoccurring parameters (Page 24). Whenever the block has received new data and has output it to the output "ReturnedSetpoint" or "LocalOperation", the "NewData" output is set to TRUE for one OB1 cycle.

Name:	**InputOK**	
Declaration:	OUTPUT	
Data type:	BOOL	
Range of values:	Output	Q 0.0 ... Q n.7
	Memory	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7
Explanation:	The display "InputOK" is set for one cycle at the entry of the data frame into the send buffer. The display is independent of the "HWmode" parameter. If you do not require the parameter, leave it open.	

Name:	**InputError**
Declaration:	OUTPUT
Data type:	BOOL

Range of values:	Output	Q 0.0 ... Q n.7
	Memory	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7
Explanation:	Display of input errors at the "Release" input, if "HWmode" equals TRUE. The output has a 1 signal as soon as the "Release" command lasts longer than defined via MaxInputTime. The "InputError" output is only set to 0 again when the release input has a 0 signal. This can be used, for example, to detect that the input is defective. If you do not require the parameter, leave it open.	

3.1.30 Setpoint typical Set01W_R

Function

Receive 1 setpoint in the station as a 16 bit value or enter locally and return the local setpoint to the master station

The block can receive and output data of the "Set01W_S" and "Set01W_FS" typicals.

Parameters

General parameters:	**PartnerNo** **PartnerObjectNo** **Enabled** **ImageMemory** **Conditional** **Unconditional** **TimeStamp** For a description, see section Reoccurring parameters (Page 24).	
Name:	**Local**	
Declaration:	INPUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	FALSE	
Address range:	Input	I 0.0 ... I n.7
	Memory	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7

Explanation: Release of the local setpoint input of the data area specified by "LocalSetpointInput".

A setpoint sent by the partner (master station) is not accepted by the object as long as "Local" = TRUE.

The current status of the "Local" input is transmitted to the partner together with a copy of the setpoint which is currently being output at "SetpointOutput" (setpoint mirroring).

Bumpless switchover:

- When there is a switchover from "Local" = 0 to 1, the last values output at the "SetpointOutput" output are retained until a new setpoint is entered via the local input area "LocalSetpointInput".
- When there is a switch back from "Local" = 1 to 0, the last value output at the "SetpointOutput" output is retained until the block receives a new setpoint from the partner.

Read the note on the "ContinuousEnterFunct" parameter.

Name:	**EnterInput**	
Declaration:	INPUT	
Data type:	BOOL	
Range of values:	TRUE / FALSE	
Default:	FALSE	
Address range:	Input	I 0.0 ... I n.7
	Memory	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7

Explanation: Enter input for local setpoint input.

Edge triggered, the parameter triggers adoption of the setpoint at the setpoint input "LocalSetpointInput".

A signal change at "EnterInput" is only taken into account when the value TRUE is set at the "Local" input parameter and "ContinuousEnterFunct" = FALSE. If these conditions are fulfilled, a signal change from 0 to 1 at the causes the setpoint entered at "LocalSetpointInput" to be adopted and output at "SetpointOutput".

This method of setpoint adoption is suitable for hardware input, for example via a console, control panel or an operator panel (OP). With an OP, the adoption must be triggered by a function key.

If you do not require the parameter, simply leave it open.

Name:	**ContinuousEnterFunct**
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE

Default:	FALSE	
Explanation:	Continuous local setpoint acquisition When "Local" = TRUE and "ContinuousEnterFunct" = TRUE the setpoint at "LocalSetpointInput" is continuously read in and checked for changes. The setpoint read in at "LocalSetpointInput" is output at the "SetpointOutput" output if at this time the local input value differs from the last returned setpoint. If an array is detected at the setpoint input "LocalSetpointInput" this is output immediately at "SetpointOutput" (without a change check). This method of setpoint adoption is suitable for software input. It can, however, also be used for entering setpoints at an operator panel (OP) if the OP does not have a function key. If you do not require the parameter, simply leave it open.	
Name:	**LocalSetpointInput**	
Declaration:	INPUT	
Data type:	WORD	
Range of values:	See address range	
Default:	0 (W#16#0)	
Address range:	Input words	IW0 ... IWn PIW0 ... PIWn
	Memory words	MW0 ... MWn LW0 ... LWn
	Data words	DBm.DBW0 ... n
Explanation:	Local setpoint input A value at "LocalSetpointInput" is only adopted if "Local" = TRUE. If this condition is met, how a pending setpoint is processed depends on whether it is a hardware or software input. The type of input is decided by the "ContinuousEnterFunct" parameter: • ContinuousEnterFunct = FALSE Hardware input A setpoint at "LocalSetpointInput" is only read in when a signal change 0 → 1 is detected at "EnterInput". The setpoint entered locally is output via the "SetpointOutput" output and transferred to the partner. A further setpoint is then only read in again by the block when a 0 signal was detected at "EnterInput" for at least one OB1 cycle.	

- ContinuousEnterFunct = TRUE

  Software input

  A setpoint at "LocalSetpointInput" is read in continuously and checked for changes. The change check is achieved by comparison with the last valid setpoint; in other words, the value stored as the returned setpoint. With each change, the setpoint is output via the output set at "SetpointOutput" and transferred to the partner.

  With software input a new setpoint can be entered in every OB1 cycle.

  With hardware input an empty cycle must be detected before a new setpoint can be read in by the block.

If you do not require the parameter, simply leave it open.

Name:	**SetpointOutput**	
Declaration:	OUTPUT	
Data type:	WORD	
Range of values:	See address range	
Default:	0 (W#16#0)	
Address range:	Output words	QW0 ... QWn PQW0 ... PQWn
	Memory words	MW0 ... MWn LW0 ... LWn
	Data words	DBm.DBW0 ... n
Explanation:	Setpoint output word. The setpoint sent by the partner object or entered locally at "LocalSetpointInput" is output at the "SetpointOutput" output.	

Name:	**NewData**
Explanation:	For the declaration, data type, range of values, default and function, refer to the section Reoccurring parameters (Page 24). Whenever the block has received a new setpoint from the partner object and has output it to "SetpointOutput", the "NewData" output is set to TRUE for one OB1 cycle. This also applies when there is a new local setpoint input (Local = 1).

3.2 Optional blocks

3.2.1 ListGenerator1500/300/400 FC

The ListGenerator FC is available in three versions with the following symbolic names:

- For S7-300

  ListGenerator300
- For S7-400

  ListGenerator400
- For S7-1500

  ListGenerator1500

Function

The FC ListGenerator is required in a CPU that receives data containing either an incomplete destination address or no destination address at all. The lack of the destination object number is the most important factor here because this points to the instance DB in which the received information should be stored.

Missing or incomplete destination addresses can occur when no or incomplete parameters are set for them in the station. This is permitted for typicals that send binary information, analog values or counted values. If these typicals send data to more than one destination, no destination address is set for them. Due to the missing destination information, the send frame is automatically transmitted to all destinations for which a connection was set up during configuration. This data is therefore received without a destination address at the various destinations.

Note

Supplement to destination subscriber no.

Data frames to be sent without a destination address have the destination subscriber number added by the sending TIM, and sometimes several if there are several destinations.

The TIM enters 0 in the address field for the destination object number, since the TIM does not have the relevant information. The only destination subscribers it knows are those to which it has a configured connection.

At the receiving end, the data frame therefore contains the destination subscriber number but the destination object number is 0.

If the destination object number is not contained in the received data frame, FC Distribute, which is responsible for distributing the received frames, references an object reference list.

Using the source address (source subscriber no. + source object no.) contained in every data frame, FC Distribute searches through the list for an entry that specifies the missing destination object number for the given source address; in other words, it searches for the number of the local instance DB.

This object reference list is created by FC ListGenerator. The FC has no parameters. It is linked into the cyclic user program (OB1) following the FC BasicTask.

When creating the list, FC ListGenerator uses the addresses set in the parameters for the receiving typicals. Specification of "PartnerNo" and "PartnerObjectNo" is mandatory for these typicals. These parameters are identical to the source address in the corresponding receive frame. Since the typical also knows the number of its instance DB, it therefore knows all the addresses required for an entry in the reference list.

During startup, FC ListGenerator arranges that all receiving typicals enter the addresses from their parameter assignment with the number of the instance DB in the reference list. The object reference list therefore does not require special parameter settings, it is simply created from the existing parameters of the receive typicals and is therefore always consistent.

How it works

FC ListGenerator creates the lists after startup in three consecutive OB1 cycles:

- In the first cycle, it determines how many entries will be required in the first and, if applicable, in the second object reference list. The typicals involved only increment a counter during this run.
- In the second cycle, FC ListGenerator generates the data block for the first and, if applicable, the second object reference list with the required length and enters 0 in all the data words. During the same cycle, all typicals involved enter their addresses and the number of the corresponding instance DB in the list.
- In the third and final cycle, FC ListGenerator sorts all the entries in ascending order. Sorting speeds up the search in the list during actual operation.

When generating the data block, FC ListGenerator does the following:

If no list has yet been created, a search is made for a free DB number.

- S7-1500

  The FC searches in descending order in the range from 60999 to 60000 (reserved area) for a free DB number.

- S7-300/400

  Starting with the number of DB BasicData, the number of the next lowest free DB is taken.

If a list already exists, FC ListGenerator checks to see whether the existing DB is long enough for the currently required number of references. If the length is adequate, 0 is entered as the content and the addresses are written again and sorted.

The object reference list 1 can be used in full length (65532 bytes) for max. 10922 receive objects. The maximum number of objects without a target object number is 10922.

If the existing data block is too short, different procedures are used for the various SIMATIC product series:

- For S7-300

  A new DB is generated. The old DB remains in memory because S7-300 has no delete function for data blocks.

  With the 300 CPU, you have to delete the old DBs with the programming device.

  Note:
  Without generating a new DB, you would need to delete this DB with the programming device. If there is not enough memory on the CPU to be able to generate a new DB, you need to delete the existing DB before restarting.

- For S7-400

  The existing DB is deleted, the memory is compressed and the DB is then generated with the same number in the new length.

  With the 400 CPU, you may be able to manually compress the memory or reload the CPU.

- For S7-1500

  The existing DB is deleted and generated in the new length with the same number.

  With the 1500 CPU, the memory is automatically compressed, just like with the 300 CPU.

If the ListGenerator FC can no longer generate a DB, an error message is written to the diagnostic buffer of the CPU:

- 0xB107 "Error generating the object reference list"
  - DB[Info1] cannot be created.
  - Cause: [Info2].

    In Info2, the return value of the SFC Create_DB function is output, refer to the description there.

3.2.2 FC PartnerMonitor

Function

FC PartnerMonitor has the following functions:

- It displays important status information about a SINAUT subscriber (see "PartnerStatus" parameter).
- The FC can also be used to trigger a general request of the subscriber, except to a control center (e.g. ST7cc/ST7sc).
- It can also be used to establish and terminate a permanent connection to the subscriber.

The FC can be called at any point in the cyclic user program (in OB1).

If you want to monitor and control more than one subscriber, include an appropriate number of FC PartnerMonitor in the user program.

A SINAUT subscriber (partner) can only be an ST7 CPU or an ST7cc to which a connection was configured. TIMs cannot be monitored or controlled by FC PartnerMonitor.

Note

FC PartnerMonitor in a station

FC PartnerMonitor can also be used in a station. However, the control inputs for establishing and terminating a permanent connection can then no longer be used. This only works in the master station when the local TIM is a master station TIM.

Parameter

Name:	**PartnerNo**
Declaration:	INPUT
Data type:	INT
Range of values:	1 ... 32000 [ms]
Explanation	SINAUT subscriber no. of the subscriber to be monitored and controlled. If the set "PartnerNo" is not found in the administration (DB BasicData), then (only during startup) an entry is written to the diagnostics buffer (event ID B101). The CPU does not change to STOP. The status of a subscriber with correct parameter settings is indicated in the "PartnerStatus" output word and the control inputs are processed. An unknown subscriber is not processed until the error is eliminated. The "PartnerStatus" output word remains set to 0 during this time.

Name:	**MaxConnectTime**
Declaration:	INPUT
Data type:	INT
Range of values:	• 0 (no limitation) • 1 … 480 [minutes]
Explanation	Maximum duration of a permanent connection The time specified here (> 0) is activated at the start of a permanent connection (see "PermanentCall_On" parameter). If the time elapses before the permanent connection is reset, it is automatically disconnected. The time is retriggered as long as the signal 1 is present at the "PermanentCall_On" input. The time specified here applies to a permanent connection in a dial-up network as well as to a permanent connection (continuous polling) on a dedicated line.

Name:	**PartnerStatus**	
Declaration:	OUTPUT	
Data type:	WORD	
Range of values:	Output words	QW0 ... QWn PQW0 ... PQWn
	Memory words	MW0 ... MWn LW0 ... LWn
	Data words	DBm.DBW0 ... n
Explanation	Output word to indicate the status of the subscriber to be monitored. If you do not require the parameter, simply leave it open.	

The meaning of the status bits in the "PartnerStatus" output word:

Bit	.0	Status of the subscriber
	0	0 = Subscriber disrupted
	1	1 = Subscriber OK

Bit	.1	Status of the redundant connection
	0	0 = Redundant connection is disrupted
	1	1 = All connections OK.

Bit	.3	.2	Status of the general request (GR)
	0	0	0 = GR complete without error
	0	1	1 = GR started
	1	0	2 = GR start received
	1	1	3 = GR finished with error (GR incomplete or cannot be executed, e.g. due to fault at subscriber)

Bit	.6	.5	.4	Status of the dial-up connection
	0	0	0	0 = No connection
	0	0	1	1 = Outgoing call activated
	0	1	0	2 = Incoming call established
	0	1	1	3 = Outgoing call established
	1	0	0	4 = Permanent connection registered
	1	0	1	5 = Permanent connection established
	1	1	0	6 = Permanent connection disconnected

Bit	.7	Status of the dial-up connection
	0	0 = No dial-up connection check in background
	1	1 = Dial-up connection check in background is activated

Bit	.8	Status of continuous polling (on dedicated line)
	0	0 = No continuous polling
	1	1 = Continuous polling activated

Bit	.9	Status of the WAN connection resources
*)	0	0 = Sufficient resources on partner
	1	1 = Insufficient resources on partner

Bit	.10	Time status
	0	0 = Date/time not available / not OK on partner
	1	1 = Date/time OK on partner

Bit	.11	Time-of-day synchronization
	0	0 = The partner CPU received a plausible time during the last synchronization or no time of day has been received since startup.
	1	1 = The partner CPU has received an implausible time of day; the last valid time will continue to be used. (Display only with TimeMask > V1.6)

Note

In-out parameters

The following parameters are in/out parameters (declaration IN_OUT):

- GeneralRequest
- PermanentCall_On
- PermanentCall_Off

It is difficult to specify local bit memory with this parameter type and this should not be used.

Name: **GeneralRequest**
Declaration: IN_OUT
Data type: BOOL

Range of values:	Input	I 0.0 ... I n.7
	Memory	M 0.0 ... M n.7
	Data bit	DBm.DBX 0.0 ... n.7

Explanation

Input for triggering a general request to the subscriber specified with PartnerNo.

A general request to the subscriber is triggered with a 1 signal at this input if no request is active for this subscriber at this time. The input is then automatically reset by the FC.

If an input of a digital input is specified (I 0.0 ... I n.7), you are responsible for resetting the signal at the input. Reset the signal before ending the currently running general request so that another general request is not triggered immediately.

Name: **PermanentCall_On**

Declaration: IN_OUT

Data type: BOOL

Range of values:	Input	I 0.0 ... I n.7
	Memory	M 0.0 ... M n.7
	Data bit	DBm.DBX 0.0 ... n.7

Explanation

Input for triggering a permanent connection to the subscriber specified with PartnerNo.

A permanent connection to the subscriber is triggered with a 1 signal at this input if there is currently no permanent connection to this subscriber. The input is then automatically reset by the FC. If an input of a digital input is specified (I 0.0 ... I n.7), you are responsible for resetting the signal at the input at the latest before the termination of the existing permanent connection.

A 1 signal at the input "PermanentCall_On" also activates the time specified with "MaxConnectTime" if it is greater than 0.

Depending on whether the subscriber can be reached over a dial-up connection or a dedicated line, the command to establish the permanent connection is processed as follows and indicated at the "PartnerStatus" output:

- **For a dial-up connection:**

  A dial-up connection is established by the master TIM to the appropriate subscriber and, regardless of the data traffic, maintained until the terminate command is sent.
  The current status of the permanent connection is indicated in the PartnerStatus output word with the bits 4 ... 6 (see PartnerStatus parameter).

- **For a dedicated line:**

  In this case the master TIM operates in polling mode with the stations. A permanent connection is implemented in this case by 'continuous polling' of the subscriber. This is actually an intermittent poll to the subscriber; in other words, the other subscribers on the dedicated line network are still polled but the preferred subscriber is polled again after every poll to a 'normal' subscriber .
  The current status of the continuous polling is indicated by bit 8 in the "PartnerStatus" output parameter.

Special feature with stations:

A permanent connection cannot be established from a station. This control input cannot therefore be used when FC PartnerMonitor is used in a station.

Name:	**PermanentCall_Off**	
Declaration:	IN_OUT	
Data type:	BOOL	
Range of values:	Input	I 0.0 ... I n.7
	Memory	M 0.0 ... M n.7
	Data bit	DBm.DBX 0.0 ... n.7

Explanation

Terminating an existing permanent connection

The input serves to trigger termination of an existing permanent connection to the subscriber specified with "PartnerNo".

A permanent connection to the subscriber is terminated with a 1 signal at this input if there is currently a permanent connection to this subscriber. The input is then automatically reset by the FC. If an input of a digital input is specified (I 0.0 ... I n.7), you are responsible for resetting the signal at the input via the user program. This should be done at the latest before establishing a permanent connection again.

Depending on whether the subscriber can be reached over a dial-up connection or a dedicated line, the command to terminate the permanent connection is processed as follows and indicated at the "PartnerStatus" output:

- **For a dial-up connection:**

  The existing dial-up connection is terminated by the master TIM but only after any pending data has been sent.
  The current status of the permanent connection is indicated in the PartnerStatus output word with the bits 4 ... 6 (see PartnerStatus parameter).

- **For a dedicated line:**

  The master TIM deletes the registration for continuous polling of the corresponding subscriber. The polling cycle for all connected subscribers continues in normal mode.
  The current status of the continuous polling is indicated by bit 8 in the "PartnerStatus" output parameter.

  Continuous polling can also be canceled on a dedicated line by instructing the master TIM to start continuous polling of another subscriber. The existing job is then replaced by the new one.

Special feature with stations:

A permanent connection cannot be terminated by a station. This control input cannot therefore be used when FC PartnerMonitor is used in a station.

3.2.3 FC PartnerStatus

Function

The FC PartnerStatus can show the current status ’disrupted’ or ’OK’ for a maximum of 8 SINAUT subscribers.

The FC can be called at any point in the cyclic user program (in OB1).

If you want to monitor more than 8 subscribers, include the appropriate number of FCs PartnerStatus in the user program.

The partner can be an ST7 CPU or an ST7cc to which a connection was configured, or a local TIM.

One bit per subscriber is reserved in the "PartnerStatus" output byte to indicate the status of the respective subscriber:

- FALSE (0):
  - Subscriber disrupted
  - Corresponding input parameter not used (= 0)
  - Subscriber unknown
- TRUE (1): Subscriber OK

Parameters

Name:	**Partner1 ... Partner8**
Declaration:	INPUT
Data type:	INT

Range of values:
- 0

  Value for unrequired parameter
- 1 ... 32000

  Number of the subscriber to be monitored

Explanation: SINAUT subscriber number of the subscriber to be monitored

If a set subscriber number is not found in the administration (DB BasicData), then (only during startup) an entry is written to the diagnostics buffer (event ID B101). The CPU does not change to STOP.

The status of a subscriber with a correct parameter assignment is indicated in the "PartnerStatus" output byte.

An unknown subscriber is not processed until the error is eliminated. Their status bits are set to 0.

Name:	**PartnerStatus**	
Declaration:	OUTPUT	
Data type:	BYTE	
Range of values:	Output bytes	QB0 ... QBn PQB0 ... PQBn
	Memory bytes	MB0 ... MBn LB0 ... LBn
	Data bytes	DBm.DBB0 ... n

Explanation: Indication of the status of the subscriber to be monitored.

Assignment of the status bits in the "PartnerStatus" output byte depending the parameters Partner1 ... "Partner8":

Bit	.7	.6	.5	.4	.3	.2	.1	.0
Partner	8	7	6	5	4	3	2	1

Value:
- 0 = partner disrupted, not set in the parameters or unknown
- 1 = partner OK

3.2.4 FC PathStatus

Function

The block (FC) shows the status of the path to a partner from the perspective of the local TIM.

A maximum of 2 paths (main and substitute path) to a partner can be configured. Both paths must begin or end on a local TIM.

The block shows the following:

- The paths via which the partner can be reached.
- The path currently being used
- The TIM interface via which the main path was configured.
- The TIM interface via which the substitute path was configured.

The path of a connection is specified as a combination of the used interfaces of the TIM and the status of the path.

In the output byte PathStatus the following bits are reserved:

- Two bits for the interface of the main path
- Two bits for the interface of the substitute path
- Two bits for the status of the main path
- Two bits for the status of the substitute path

The FC can be called at any point in the cyclic user program (OB1) after calling the FC BasicTask.

If the status of the path to more than one subscriber is to be shown, a corresponding number of FC calls need to be programmed in the user program program.

Parameters

Name:	**Partner**	
Declaration:	INPUT	
Data type:	INT	
Range of values:	1 ... 32000 (subscriber number)	
Explanation:	SINAUT subscriber no. of the partner	

Name:	**PathStatus**	
Declaration:	OUTPUT	
Data type:	BYTE	
Range of values:	Output bytes	QB0 ... QBn PQB0 ... PQBn
	Memory bytes	MB0 ... MBn LB0 ... LBn
	Data bytes	DBm.DBB0 ... n
Explanation:	Display of the path status of the connection to the partner	

PathStatus - Coding

Table 3- 8 Coding of the status bits in the "PathStatus" output byte

Bits 6 + 7	Bits 4 + 5	Bits 2 + 3	Bits 0 + 1
Configured interface		Path status	
No. for substitute path	No. for main path	Substitute path (2nd path)	Main path (1st path)

Configured interface (bits 4..7)

The TIM interfaces in the block are consecutively numbered decimally from 0 to 3, see "No." column in the table. The table shows the coding of the bits for the different TIM types.

Table 3- 9 Coding of bits 4 + 5 (interface of the main path) or bits 6 + 7 (interface of the substitute path)

Status bit 5 (7)	Status bit 4 (6)	Interfaces			
		No.	TIM 3V-IE	TIM 4R-IE	TIM 1531 IRC
0	0	0	Ethernet "IE1" (X2)	Ethernet "IE1" (X3)	Ethernet "IE1" (X1)
0	1	1	-	Ethernet "IE2" (X4)	Ethernet "IE2" (X2)
1	0	2	Serial "WAN1" (X1)	Serial "WAN1" (X1)	Ethernet "IE3" (X3)
1	1	3	-	Serial "WAN2" (X2)	Serial "WAN" (X4)

Path status (bits 0..3)

- Main path = 1. Path (bits 0 + 1)
- Substitute path = 2nd path (bits 2 + 3)

Table 3- 10 Status table: Coding of bits 0 + 1 or bits 2 + 3

Status bit 1 (3)	Status bit 0 (2)	Meaning bit 1	Meaning bit 0
0	0	Path not current	Subscriber not reachable
0	1	Path not current	Subscriber reachable
1	0	Path current	Subscriber not reachable
1	1	Path current	Subscriber reachable

Coding options - TIM 1531 IRC

Same coding of the configured interface for the main and the substitute path means that there is no path redundancy (only 1 interface configured). The path status is output via the bits of the main path (1st path).

Table 3- 11 Coding options for the output byte "PathStatus"

Configured interface		Path status	
Coding for substitute path	Coding for main path	Substitute path (2nd path)	Main path (1st path)
0 0	0 0 (Coding for IE1)	Irrelevant (not redundant)	Status IE1
0 0	0 1 (Coding for IE2)	Status IE1	Status IE2
0 0	1 0 (Coding for IE3)	Status IE1	Status IE3
0 0	1 1 (Coding for WAN1)	Status IE1	Status WAN1
0 1	0 0	Status IE2	Status IE1
0 1	0 1	Irrelevant (not redundant)	Status IE2
0 1	1 0	Status IE2	Status IE3
0 1	1 1	Status IE2	Status WAN1
1 0	0 0	Status IE3	Status IE1
1 0	0 1	Status IE3	Status IE2
1 0	1 0	Irrelevant (not redundant)	Status IE3
1 0	1 1	Status IE3	Status WAN1
1 1	0 0	Status WAN1	Status IE1
1 1	0 1	Status WAN1	Status IE2
1 1	1 0	Status WAN1	Status IE3
1 1	1 1	Irrelevant (not redundant)	Status WAN1

Coding options - TIM 4R-IE

Same coding of the configured interface for the main and the substitute path means that there is no path redundancy (only 1 interface configured). The path status is output via the bits of the main path (1st path).

Table 3- 12 Coding options for the output byte "PathStatus"

Configured interface		Path status	
No. for substitute path	No. for main path	Substitute path (2nd path)	Main path (1st path)
0 0	0 0 (Coding for IE1)	Irrelevant (not redundant)	Status IE1
0 0	0 1 (Coding for IE2)	Status IE1	Status IE2
0 0	1 0 (Coding for WAN1)	Status IE1	Status WAN1
0 0	1 1 (Coding for WAN2)	Status IE1	Status WAN2

0 1	0 0	Status IE2	Status IE1
0 1	0 1	Irrelevant (not redundant)	Status IE2
0 1	1 0	Status IE2	Status WAN1
0 1	1 1	Status IE2	Status WAN2
1 0	0 0	Status WAN1	Status IE1
1 0	0 1	Status WAN1	Status IE2
1 0	1 0	Irrelevant (not redundant)	Status WAN1
1 0	1 1	Status WAN1	Status WAN2
1 1	0 0	Status WAN2	Status IE1
1 1	0 1	Status WAN2	Status IE2
1 1	1 0	Status WAN2	Status WAN1
1 1	1 1	Irrelevant (not redundant)	Status WAN2

3.2.5 FC PulseCounter

Function

The FC PulseCounter is responsible for count pulse acquisition.

A maximum of 8 pulse strings are detected via digital inputs and fed to the function blocks with the aid of SIMATIC counters that put together the counted value frames (Cnt01D_S, Cnt04D_S).

The acquisition of the count pulses is time-controlled To do this the FC PulseCounter must be included in a cyclic interrupt OB, e.g. OB35. The call interval of the cyclic interrupt OB must be matched to the pulse duration of the count pulses. You will find more information on count pulse acquisition with the cyclic interrupt OB in the section Cyclic interrupt OB (Page 16).

Parameters

Name:	**InByte**	
Declaration:	INPUT	
Data type:	BYTE	
Range of values:	Input bytes	PEB0 ... PEBn
	Memory bytes	MB0 ... MBn LB0 ... LBn
	Data bytes	DBm.DBB0 ... n

If an input byte of a digital input is specified, this must be the address of the I/O byte (PIB) directly from the digital input modules. The current status of the count input can only be detected reliably by direct access.

When reading out from the process image of the inputs (PII) count pulses could remain undetected.

Explanation: Input byte for count pulses.

The parameters for inputs for count pulse acquisition can can be set in bytes.

Name:	**EnableMask**
Declaration:	INPUT
Data type:	BYTE
Range of values:	B#16#00 ... B#16#FF
Explanation	Enable mask for the counting inputs

With this parameter, it is possible to specify in the form of a bit mask at which inputs in the input byte count pulses are actually connected. The following applies to every bit in the bit mask:

- 0 = Input bit blocked for acquisition
- 1 = Input bit enabled for acquisition

The input of the mask is only permitted in hexadecimal format B#16#00 to B#16#FF.

Input as an 8-bit binary number from 2#0 to 2#1111 1111 is not possible with the data type BYTE.

The assignment of the bits in the mask to the inputs in the input byte "InByte":

InByte	.7	.6	.5	.4	.3	.2	.1	.0
EnableMask B#16#	0 … F					0 … F		

Example: EnableMask B#16#83

Enabled are: Inputs .7, .1 and .0

Blocked are: Inputs .6 to .2

Name:	**CntIn_0 ... CntIn_7**
Declaration:	INPUT
Data type:	COUNTER
Range of values:	C0 or C1 ... Cn (n depends on the CPU)

Explanation Pulse counter

For each of the enabled counting inputs a SIMATIC counter needs to be specified for the corresponding parameter CntIn_0 ... . "CntIn_7". With each pulse acquired, the SIMATIC counter is incremented.

The counters set here must be specified as input counters (parameter Counter_1 ... _4) in the actual counted value function blocks "Cnt01D_S" and "Cnt04D_S". These function blocks out read the assigned counter and then reset it.

As the placeholder for parameters that are not required, it is recommended to specify the counter C0.

Example of the parameter assignment of "CntIn_0" ... "CntIn_7" starting at "EnableMask" = B#16#83:

CntIn_0 : = Z10
CntIn_1 : = Z11
CntIn_2 : = Z0
CntIn_3 : = Z0
CntIn_4 : = Z0
CntIn_5 : = Z0
CntIn_6 : = Z0
CntIn_7 : = Z12

3.2.6 FC ST7ObjectTest

Validity

S7-300/400

Function

Calling FC ST7ObjectTest in programming error OB121 prevents a CPU STOP, if the CPU receives data with an unknown destination object no.

FC ST7ObjectTest checks why OB121 was called, i.e. which block type is missing.

- If the missing block is a data block and this data block is an instance DB of a SINAUT object, the CPU does not change to STOP.
- If no SINAUT instance DB is missing but rather another block (DB, FB, FC), you can specify the reaction with the parameter "StopInOtherCases".
  - CPU changes to STOP.
  - CPU continues to run.

For more information on the programming error OB121 and background information relating to the use of FC ST7ObjectTest, see section Programming error OB (Page 17).

Parameters

Name:	**StopInOtherCases**
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Explanation:	The CPU should change to STOP in other error situations. For details of the parameter see above, section "Function". • TRUE: CPU changes to STOP. • FALSE: CPU continues to run.

3.2.7 FC TestCopy

Function

Using FC TestCopy, extracts of the data traffic between ST7 subscribers can be recorded or the entire traffic can be recorded. With search masks to be set in the control field of DB TestCopyData individual frame types can be filtered out and then copied from the send or receive buffer for further evaluation in the DB TestCopyData. For details, see below.

Send and receive frames are all stored in the same data block DB TestCopyData. This makes it simple to track the chronological order of the copied send and receive frames.

Requirements

To use the TestCopy function, the user program must meet the following conditions:

- The FC TestCopy function must be available on the CPU.
- DB TestCopyData must be present on the CPU and have a sufficient length.
  - To achieve this, copy DB TestCopyData (DB99) from the TD7 library to your CPU.
  - If necessary, change the length of the buffer area in the DB by increasing or decreasing the size of the array "TestCopyBuffer" in DW40, which has the default length of [0..240] WORD.
- Make the following entries in the respective communication DB of the CPU 300/400 (BComData / XComData / PComData) whose send and/or receive frames you want to write:
  - In DW32 (TestCopyDBNo) of the communication DB, enter the number of the DB TestCopyData.
  - In DW34 (TestCopyFCNo) of the communication DB, enter the number of the FC TestCopy.

  Proceed in the same way for the communication DB of CPU 1500. The tags in DB BConnectData have the same names.

Linking FC TestCopy into the user program

If the above-mentioned conditions are fulfilled, the test function is processed cyclically by the respective communication FB of the CPU.

FC TestCopy cannot be called in the user program.

Monitoring the written data

Use the ready-made watch table "TestCopyMonitor", which you can copy from the TD7 library into the "Watch and force tables" directory of the CPU.

If the settings are to be retained even after CPU startup, they can also be stored directly in the start values of the individual BConnection instances of the DB BConnectData.

3.2.8 DB TestCopyData

Structure of the TestCopyData DB

Areas of the TestCopyData DB

The DB for the TestCopy function is divided into the following areas (after Offset in the DB):

- **0 ... 27: User interface**

  Interface for setting the TestCopy mode and functions. The area is divided into:

  – 1 ... 13
  Filter settings for RecvCopy function and number of counted received frames

  – 15 ... 25
  Filter settings for SendCopy function and number of counted sent frames

- **28: Error display**
- **31 ... 39: Internal management pointer**
- **40 ... (Default: 523): Buffer range**

  Buffer area for storing frames that match the filter criteria.

  The buffer area must be configured as an array [0...xxxx] of WORD.

The following table shows the structure of the DB TestCopyData:

Data type / (offset)		Tag name	Format	Explanation
User interface				
DBB	0	OperationMode	BYTE	Mode
DBW	12	Recv_TgramCounter	INT	Number of copied receive frames
DBW	26	Send_TgramCounter	INT	Number of copied send frames
RecvCopy function				
DBB	1	Recv_TgrmType	BYTE	Receive filter: Message type (MT)
DBW	2	Recv_DestSubscr	INT	Receive filter: Destination subscriber no.

Data type / (offset)		Tag name	Format	Explanation
DBW	4	Recv_DestObject	INT	Receive filter: Destination object no.
DBW	6	Recv_SourceSubscriber	INT	Receive filter: Source subscriber no.
DBW	8	Recv_SourceObject	INT	Receive filter: Source object no.
DBW	10	Recv_StartIndex	INT	Receive filter: Start index no.
DBB	14	SpareDBB14	BYTE	*Reserve*
SendCopy function				
DBB	15	Send_TgrmType	BYTE	Send filter: Message type (MT)
DBW	16	Send_DestSubscr	INT	Send filter: Destination subscriber no.
DBW	18	Send_DestObject	INT	Send filter: Destination object no.
DBW	20	Send_SourceSubscriber	INT	Send filter: Source subscriber no.
DBW	22	Send_SourceObject	INT	Send filter: Source object no.
DBW	24	Send_StartIndex	INT	Send filter: Start index no.
Error display				
DBB	28	FC_RetVal	BYTE	Error information: 0 = No error 1 = DB TestCopyData too short 10 = Unknown mode
DBB	29	SpareDBB29	BYTE	*Reserve*
DBB	30	SpareDBB30	BYTE	*Reserve*
Internal management pointer				
DBB	31	TestCopyStatus	BYTE	Status byte for TestCopy operation
DBB	32	TestCopyCmdByte	BYTE	Command byte for TestCopy operation
DBB	33	TestCopyDelCount	BYTE	Loop counter for TestCopy delete function
DBW	34	NextFreeCopyByte	INT	Address of the next free TestCopyBuffer byte
DBD	36	StartTimeSFC64	DINT	SFC64 time at the start of the copy procedure
Buffer range				
DBB	40	TestCopyBuffer[0]	BYTE	Copy area, byte 0
DBB	41	TestCopyBuffer[1]	BYTE	Copy area, byte 1
DBB	42	TestCopyBuffer[2]	BYTE	Copy area, byte 2
DBB	43	TestCopyBuffer[3]	BYTE	Copy area, byte 3
DBB	n	TestCopyBuffer[n]	BYTE	Copy area, byte n

Structure of a copied message block

A frame block can contain several frames. The frames are saved in the DB TestCopyData according to the following rules:

1. The first entry indicates the time difference in milliseconds (7 decade BCD plus sign) since the last selection of an operating mode > 0.
2. This is followed by a separation sign AAAA for sent messages, EEEE for received messages.
3. Storage of the first message from the frame block.
4. Separation identifier AAAA, or EEEE:
5. Storage of the last frame from the message block.
6. Block end identifier FFFF.

Example

- All received frames will be stored in DB TestCopyData.
- Communication via X blocks, i.e. max. 76 bytes per receive block.
- The receive buffer of the DB XComData is the source for FC TestCopy.
- The current receive block contains 3 messages.

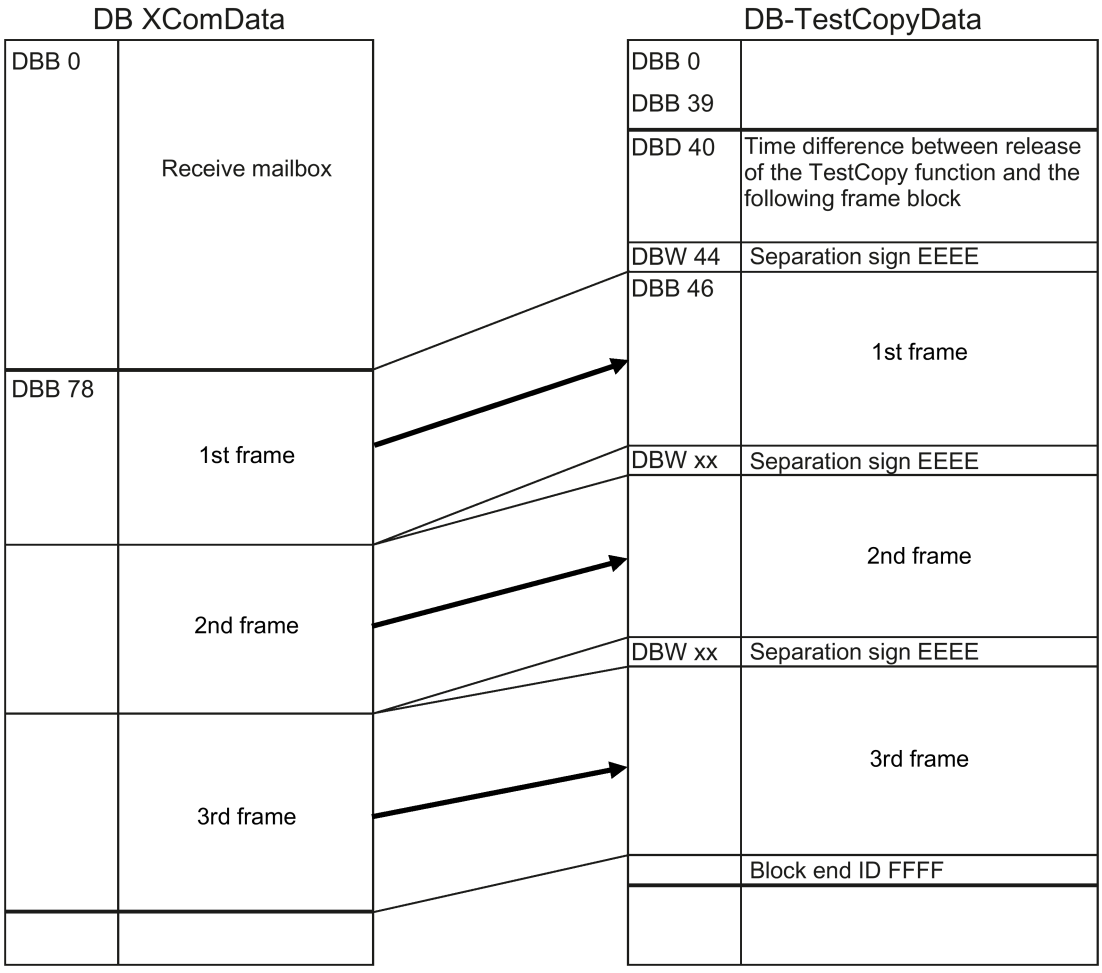

Figure 3-1 Example of filling the DB TestCopyData

Length calculation

FC TestCopy uses the following parameters for determining the minimum length for the DB TestCopyData:

Parameter	Parameter name	Length
Length of communication buffer	$Len_{ComBuffer}$	76 or 202 bytes *
Minimum frame length	$Len_{MinTgrm}$	14 bytes
Offset management area	Offset	40 bytes
Length of the time difference	Len_{dt}	4 bytes
Length of the block separators	$Len_{Trenner}$	2 bytes

* 76 with X communication, 202 with B communication

The formula used for the actual calculation is the same for X communication and B communication. The results differ only due to different lengths for the communication buffer for X and B communication:

- **Length with X communication**

Len_{Min_Xcom}	$=Len_{ComBuffer} + Offset + Len_{dt} + (Len_{ComBuffer} / Len_{MinTgrm} + 1) * Len_{Trenner}$
	= 76 + 40 + 4 + (76 / 14 + 1) * 2
	= 120 + 12 = 132 bytes minimum

- **Length with B communication**

Len_{Min_Bcom}	$=Len_{ComBuffer} + Offset + Len_{dt} + (Len_{ComBuffer} / Len_{MinTgrm} + 1) * Len_{Trenner}$
	= 202 + 40 + 4 + (202 / 14 + 1) * 2
	= 236 + 40 = 276 bytes minimum

If FC TestCopy determines that the DB TestCopyData does not have the calculated minimum length an error message to this effect is entered in data byte DBB28.

3.2.9 Operation of TestCopyMonitor

Operating modes and transfer directions of DB TestCopyData

In byte 0 (OperationMode) of the DB TestCopyData, the mode of the FC is coded. The values 0 ... 3 identify the mode:

- Mode 0

  Function blocked

- Mode 1

  Frame entry as of the start of DB TestCopyData

- Mode 2

  Write to DB TestCopyData endlessly as circulating buffer

- Mode 3

  Fill DB TestCopyData once, then set mode 0.

You will find the detailed coding in the table below.

Along with the mode the coding of DBB0 also includes the transfer direction of the logged data. The following assignment applies for the coding:

- Bit 0...3

  Modes (0, 1, 2, 3) for the direction "RecvCopy" (copy receive frames)

- Bit 4...7

  Modes (0, 1, 2, 3) for the direction "SendCopy" (copy send frames)

Bit	.7	.6	.5	.4	.3	.2	.1	.0
Direction	SendCopy				RecvCopy			

The functions "SendCopy" and "RecvCopy" can be activated individually or at the same time, however only a common mode is possible for both communication directions.

When a mode > 0 is set in the less significant half byte, this always applies to both communication directions.

Only if mode 0 is coded in bit 0...3 (RecvCopy), does the value in bit 4...7 apply (SendCopy).

Exception:
To delete the DB TestCopyData, DBB0 must be written with FF, 0F is not enough.

Examples of coding of DBB0:

- 00_h

  No TestCopy function enabled

- 03_h

  Only RecvCopy function (OperationMode = 3), no SendCopy function.

- 30_h

  Only SendCopy function (OperationMode = 3), no RecvCopy function.

- 33_h

  RecvCopy function and SendCopy function (OperationMode = 3)

- FF_h

  Delete content of DB TestCopyData

Filter settings via TestCopyMonitor

Run FC TestCopy via a watch table which is available in finished form as "TestCopyMonitor" in the TD7 library.

The following settings are possible using the watch table in the DB TestCopy:

Name (symbol)	Permitted values (hex.)	Meaning
OperationMode	00	Mode 0: Function blocked
	11	Mode 1: Frame entry always as of the start of DB TestCopyData
	22	Mode 2: Write to DB TestCopyData endlessly as circulating buffer
	02 20	• Write for SendCopy function • Write for RecvCopy function
	33	Mode 3: Fill DB TestCopyData once, then set mode 0.
	03 30	• Fill for SendCopy function • Fill for RecvCopy function
	FF	Delete the entire DB TestCopyData and reset to default

As the output value, FC TestCopy returns a count value in DBW12 (Recv_TgramCounter) of the DB TestCopyData for the frames received since setting mode 1, 2 or 3 that match the filter criteria.

The number of send frames is entered in DBW26 (Sendcv_TgramCounter).

In DBB28, you receive a return value that indicates the errors that occurred during processing of the FC:

- FC_RetVal = 0

  No error

- FC_RetVal = 1

  The specified DB TestCopyData is too short.

- FC_RetVal = 10d

  The mode entered in DBB0 is not defined.

In addition, the following special filters are available for the RecvCopy or SendCopy function:

Name (symbol) *	Permitted values (hex.)	Meaning
Xxxx_TgrmType	FF	Copy all frame types (TA) to DB TestCopyData
	00	Copy only spontaneous Org. frames (TA = 0)
	11	Copy only queried Org. frames (TA = 1).
	22	Copy only spontaneous data frames (TA = 2).
	33	Copy only queried data frames (TA = 3).
	01	Copy Org. frames with TA = 0 or 1
	23	Copy data frames with TA = 2 or 3
	Any combination	Copy any 0, 1, 2, 3 combinations.
Xxxx_DestSubscr	All permitted subscribers	Filter for the dest. subscriber no. in the frame
	-1	Copy all frames regardless of the dest. subscriber no.
Xxxx_DestObject	All permitted objects	Filter for the dest. object no. in the data frame.
	-1	Copy all frames regardless of the dest. object no.
Xxxx_SourceSubscr	All permitted subscribers	Filter for the source subscriber no. in the frame
	-1	Copy all frames regardless of the source subscriber no.
Xxxx_SourceObject	All permitted objects	Filter for the source object no. in the data frame
	-1	Copy all data frames regardless of the source object no.
Xxxx_StartIndex	All permitted indexes	Filter for the start index no. in the data frame
	-1	Copy all data frames regardless of the start index no.

***** "Xxxx": Placeholder for "Recv" or "Send"

The desired filter tags can be easily added using the tag names of the TestCopy DB.

Notes on operator input

When changing from one mode to the next, the content of DB TestCopyData is not deleted. Only internal pointers and frame counters in the management area of the DB TestCopyData are reset. When there is a mode change it is therefore recommended to use the delete function "FF" to preset the frame buffer area with 0. The makes the copied frame blocks easier to read.

If send and receive frames are to be copied, in the left half byte of the "OperationMode" parameter, the same mode must be entered as in the right half byte.

The following scheme applies to mode 0, 1, 2 and 3:

- If the buffer is deleted, FF_h must always be entered.
- Separate deletion of the receive and send frames is not possible.

3.2.10 FC TimeTask

Function

The FC TimeTask keeps a continuous date and time on a CPU The FC has no parameters.

Link the FC into the cyclic user program (in OB1) following the FC BasicTask.

FC TimeTask can only be used when the CPU is synchronized by a local TIM. Activate the time-of-day synchronization for the relevant TIM modules.

After the CPU has started up, the TIM supplies the date and current time the first time with an Org. frame. After this, time-of-day synchronization is performed at the interval specified in the configuration of the TIM. For time-of-day synchronization on MPI/partyline an interval of one minute is recommended. FC TimeTask sets the clock of the CPU with the time provided by the TIM.

The FC reads out the time in every OB1 cycle. The read time is entered in the first two double words of the DB BasicData and marked as valid or invalid and with an indication whether it is daylight saving or standard time.

From DB BasicData all blocks take the current time if they need it. For example the data point typicals do this to time stamp their data or FC Trigger to check whether a time set for the FC has been reached or a preset interval has elapsed. This time of day is also available to the user program.

Table 3- 13 Assignment of the data words with date, time and time status

CurrentDate	Data byte 0	Year * 10	Year * 1
	Data byte 1	Month * 10	Month * 1
	Data byte 2	Day * 10	Day * 1
	Data byte 3	Hour * 10	Hour * 1
CurrentTime	Data byte 4	Minute * 10	Minute * 1
	Data byte 5	Second * 10	Second * 1
	Data byte 6	Milliseconds * 100	Milliseconds * 10
	Data byte 7	Milliseconds * 1	Time status

Table 3- 14 Assignment of the half byte "time status"

0	0 = date/time invalid 1 = date/time valid
1	0 = standard time 1 = daylight saving time
2	(not used)
3	(not used)

Apart from the time status, whether or not the date/time is valid can also be determined based on data bit 16.1 "CpuClockOk". As soon as the time on the CPU is valid, this bit is set to 1 by FC TimeTask. In the user program this bit can be queried directly under the symbolic name "BasicData.CpuClockOk".

3.2.11 FC Trigger

Function

The FC sets an output (memory bit, data bit or digital output) at a time that can be configured by the user or at a preset time interval.

The FC resets this output after one OB1 cycle.

The FC can be called at any point in the cyclic user program (OB1) also more than once.

If the running of a program section or a software function is to be triggered using FC Trigger we recommend that FC Trigger is called directly before execution of this function. Applications for triggering functions due to the memory bit set by FC Trigger are, for example, as follows:

- Running through a function
- Calling a block
- Triggering transfer of a counted value every 2 hours

If several functions need to be activated at the same time this can be implemented by one FC Trigger block if all functions query the same memory bit set by the FC. This works, however, only when the triggered functions do not reset this memory bit themselves.

Remedy if triggered blocks reset the memory bit:

- You call FC Trigger often, always with the same time but a different output memory bit.
- After calling FC Trigger you reproduce the set output memory bit in an appropriate number of further memory bits.

The FC accesses the SINAUT time of day in the first two data double words of the DB BasicData. These are constantly supplied if an FC TimeTask is included in the user program and this is synchronized at regular intervals by a local TIM. FC Trigger compares the time set for it with the current time only when the time in DB BasicData, data byte 7 (time status byte, bit 0 = 1) is marked as being valid.

The accuracy with which FC Trigger operates depends on the accuracy of the time and on the OB1 cycle time.

If the OB1 cycle time is lower than 1 second (this is normal) the output is set exactly at the programmed second value with the inaccuracy of the OB1 cycle time of less than 1 s.

If the OB1 cycle time is higher than 1 seconds, the FC works with a tolerance of 4 seconds. If the FC is processed too late but still within 4 seconds of the configured time, the output is still set.

The edge memory bit "Flag" configured for the FC is set at the same time as the output and reset 5 seconds after the configured time.

No placeholder parameter may be used for the edge memory bit and it must not be reset by the user program.

You can find examples of the parameter assignment for FC Trigger further below.

Parameters

Name:	**IntervalMode**
Declaration:	INPUT
Data type:	BOOL
Range of values:	TRUE / FALSE
Explanation	Point in time / time interval • FALSE = point in time • TRUE = time interval You will find examples of the parameter assignment for a time or time interval after this explanation of the parameters.

Name:	**Hour_Minute**
Declaration:	INPUT
Data type:	WORD
Explanation	Specifies the values for hours and minutes. Further explanation: Refer to the parameter "Month_Year".

Name:	**Second_Day**
Declaration:	INPUT
Data type:	WORD
Explanation	Specifies the values for seconds and day. Further explanation: Refer to the parameter "Month_Year".

Name:	**Month_Year**
Declaration:	INPUT
Data type:	WORD

Range of values:	• 00 ... 99 • FF	
Explanation	Specifies the values for month and year Each parameter is in two parts. Each of the two values per parameter is specified with two digits as a BCD-coded value. • The first two digits specify the value for hours, seconds or month. • The two other digits specify the value for minutes, day or year. For parts of the parameter that are not needed FF is entered. Which parameters are permitted depends on the particular parameter and on the "IntervalMode" parameter. You will find further information following the explanation of the parameters.	

Name:	**TriggerOutput**	
Declaration:	OUTPUT	
Data type:	BOOL	
Range of values:	Output	Q 0.0 ... I n.7
	Memory	M 0.0 ... M n.7 L 0.0 ... L n.7
	Data bit	DBm.DBX 0.0 ... n.7
Explanation	Trigger output The output is set for the duration of one OB1 cycle when the time or time interval set for "Hour" to "Year" is reached.	

Name:	**Flag**	
Declaration:	IN_OUT	
Data type:	BOOL	
Range of values:	Memory	M 0.0 ... M n.7
	Data bit	DBm.DBX 0.0 ... n.7
	This is an in/out parameter (declaration IN_OUT). It is difficult to specify local bit memory with this parameter and this should not be used.	
Explanation	Edge memory bit for the "TriggerOutput" output. No placeholder memory bit may be specified! The edge memory bit must not be reset by the user program.	

Examples of the parameter assignment of a time or time interval

IntervalMode = FALSE (or 0)

The FC operates according the time principle. When the time set is reached, the output "TriggerOutput" is set for one OB1 cycle.

All time parameters can be used to specify the point in time. Parameters not required should be assigned FF. They are ignored during the check for reaching the time.

Even when "IntervalMode" = 0, time intervals can be set in certain ranges, see the following examples.

Permitted values for the time parameters:

Hours	00-23	Day	01-31
Minutes	00-59	Month	01-12
Seconds	00-59	Year	00-99

Examples:

- IntervalMode = FALSE

  The output "TriggerOutput" is set once on 04.02.91 at 06:45:12:

  – Hour_Minute : W#16#0645
  – Second_Day : W#16#1204
  – Month_Year : W#16#0291

- IntervalMode = FALSE

  The output "TriggerOutput" is set every day at 06:00:00:

  – Hour_Minute : W#16#0600
  – Second_Day : W#16#00FF
  – Month_Year : W#16#FFFF

- IntervalMode = FALSE

  The output "TriggerOutput "is set on the 1st of every month at 06:00:00:

  – Hour_Minute : W#16#0600
  – Second_Day : W#16#0001
  – Month_Year : W#16#FFFF

- IntervalMode = FALSE

  The output "TriggerOutput" is set every year on October 1, at 06:00:00:

  – Hour_Minute : W#16#0600
  – Second_Day : W#16#0001
  – Month_Year : W#16#10FF

IntervalMode = TRUE (or 1)

The FC operates according the time interval principle. When the time value set or a multiple of it is reached, the output "TriggerOutput" is set for one OB1 cycle.

Only the specifications for hours, minutes and seconds are relevant. The date parameters are ignored. A time interval can also only be set in hours or in minutes or in seconds. Time parameters not required should be assigned FF.

The following time intervals are permitted:

- Hours: 01, 02, 03, 04, 06, 08, 12, 24
- Minutes: 01, 02, 03, 04, 05, 06, 10, 12, 15, 20, 30, 60
- Seconds: 10, 12, 15, 20, 30, 60

Examples:

- IntervalMode : TRUE

  The output "TriggerOutput" is set as follows:

  – Hour_Minute : W#16#06FF (every 6 hours)
  – Second_Day : W#16#FFFF (at 00:00:00, 06:00:00, 12:00:00 o'clock and ...)
  – Month_Year : W#16#FFFF (... at 18:00:00 o'clock)

- IntervalMode : TRUE

  The output "TriggerOutput" is set as follows:

  – Hour_Minute : W#16#FF30
  – Second_Day : W#16#FFFF (at 00:00:00, 00:30:00, 01:00:00 o'clock and ...)
  – Month_Year : W#16#FFFF (... at 01:30:00, 02:00:00, 02:30:00 o'clock etc.)

Error message during startup

The FC checks the parameters Hour_Minute, Second_Day und Month_Year in every cycle to ensure that they keep to the permitted range of values. What is permitted is also dependent on the "IntervalMode" parameter.

If the parameter assignment is incorrect, an error message is entered in the diagnostics buffer (event ID B113) only during startup. The CPU does not change to STOP. Afterwards, the FC checks the parameters without outputting error messages until the error has been eliminated.

The diagnostics message provides a precise identification of the incorrect parameter (continuous number of the parameter, i.e. 2, 3 or 4). The causes of the diagnostics message can be depend on the parameter "IntervalMode".

IntervalMode = FALSE (or 0)

The permitted ranges of values for the parameters Hours, Minutes, Seconds, Day, Month and Year were not kept to. Apart from FF, the following can be configured:

Hours	00-23	Day	01-31
Minutes	00-59	Month	01-12
Seconds	00-59	Year	00-99

"IntervalMode" = TRUE (or 1)

In this case the error can have two different causes:

- The permitted ranges of values for the parameters Hours, Minutes and Seconds were not kept to. Apart from FF, the following can be configured:
  – Hours: 01, 02, 03, 04, 06, 08, 12, 24
  – Minutes: 01, 02, 03, 04, 05, 06, 10, 12, 15, 20, 30, 60
  – Seconds: 10, 12, 15, 20, 30, 60
- A time interval can only be set in hours or in minutes or in seconds. The two unused parameters must have FF written to them. Even if FF was entered for all three named parameters, an error exists.

3.3 System blocks

3.3.1 DB BasicData

This data block provides the central data management. It contains information that must be maintained centrally for all blocks. Among other things, the data block contains the subscriber records and the connection descriptions.

DB BasicData is automatically generated in the required length preset with the subscriber and connection specific data and then saved in the block directory of the CPU.

DB BasicData exists once on every CPU.

Note

Number of DB BasicData

In the TD7 library, DB BasicData has the number DB127 and is also saved there under this number when the DB is generated for the various CPUs In principle it would be possible to change the number, but this involves a lot of work and can cause errors in the further creation of the user program.

It is therefore recommended to keep the DB number 127 free for the DB BasicData.

3.3.2 FC Create

Auxiliary block for putting together the data to be sent and its entry in the relevant instances of the send mailbox(es) (SendBuffer array) in the respective communication DB. These are the following DBs:

- **S7-1500**

  DB BConnectData > BConnection[n]

- **S7-400**

  DB BComDataXX

- **S7-300**
  - DB BComDataXX

    or
  - DB XComDataXX

    or
  - DB PComDataXX

XX is a placeholder for the sequential number (01, 02, etc.).

FC Create is required by the data point typicals for data and organizational frames and by FC BasicTask for organizational frames only.

3.3.3 Diagnose / Diagnostics FC

Name of the block:

- For S7-300: FC Diagnose
- For S7-1500: FC Diagnosis

Auxiliary block for entering SINAUT system messages in the diagnostics buffer of the CPU.

3.3.4 FC Distribute

Auxiliary block for distribution of the data located in the receive mailbox to the data point typicals responsible or to the node objects in the subscriber records.

3.3.5 FC Search

Auxiliary block for the following search tasks:

- Search for the initial address of a subscriber object within the subscriber records
- Search for the local object no. (Instance DB) from one of the two object reference lists for a received message with an incomplete destination address

The auxiliary block is required by almost all blocks.

3.3.6 FC Startup

The block is required in every CPU. It must be linked into the startup program OB100.

The block has the task of setting the startup memory bit in the DB BasicData and to reset the corresponding edge memory bit if this is still set.

The block has no parameters.

3.4 BasicTask_* FC

Block versions

Note

Only 1 FC type per CPU

You may only use one FC type in one CPU. A mixture of several communication types with different FCs "BasicTask_*" is not allowed.

The block is available in the following versions for the different CPU types:

- **BasicTask_B**

  For S7-1500 CPU

  The block for the S7-1500 CPU cannot be used for the S7-300 or S7-400 CPU.

- **BasicTask_B**

  For S7-400 CPU

  The block for the S7-400 CPU cannot be used for the S7-300 or S7-1500 CPU.

- **BasicTask_Bnn**

  For S7-300 CPU

  The blocks for the S7-300 CPU cannot be used for the S7-400 or S7-1500 CPU.

  As of version V3.0 + SP2 of the block library, 16 versions of the "BasicTask_B" FC are provided for an S7-300 CPU. These are provided for the respective configured number of local TIM 1531 IRC, via which the CPU communicates. According to the number (nn) of local TIM modules, the FC receives the suffix "nn", e.g. "BasicTask_B03".

  When generating the TD7onCPU in the CPU a suitable "BasicTask_B" FC is generated. The block version, from which the number of local TIM 1531 IRC is determined, can be taken from the "Comment" block property of the "BasicTask_B" FC in the "TD7onCPU" directory of the CPU.

- **BasicTask_X**

  For S7-300 PU with partyline

- **BasicTask_P**

  For S7-300 CPU without partyline

For information on "Partyline", see Glossary.

The tasks of the various "BasicTask_*" FCs for handling basic communication tasks are the same in the respective CPU type. The only parameter of the FC is identical in each case.

When the TD7onCPU is generated, the correct FC is automatically created in the respective CPU.

Function

The block is required in every CPU. It handles the following tasks:

- Central tasks during startup
- The processing of all communication mailboxes
- Central organizational tasks such as starting, monitoring and answering general requests.

Call FC BasicTask as the first block in OB1.

Parameters

Name:	**UserFC**
Declaration:	INPUT
Data type:	INT
Range of values:	• 1 ... 32000 Number of the FC The maximum possible number depends on the CPU. • 0 Substitute value if there is no FC present for the specified purpose.
Explanation:	Number of a user FC for specific further processing of the received data. If an FC is specified, this FC is called automatically by the user program with all received data. At the time of the call, the receive frame is still in the receive mailbox of the communication DB. The program in the user FC can read the receive frame from the receive mailbox and process it further in any way required, e.g. write the received data to an intermediate buffer. Using the user program, the required information on the communication DB can be read from the DB BasicData via the "CurrentComDB" variable in DW60. CurrentComDB contains the following information: • S7-1500 Index of the current BConnection instance in the DB BConnectData • S7-300/400 Number of the current communication DB In the opened communication DB, the start of the current receive frame in the receive mailbox can be found via the "CurrentReceivedMessage" variable in DW10.

3.5 Communication blocks BCom (CPU 1500)

3.5.1 FB BCom

Validity

- S7-1500

The block for the S7-1500 CPU cannot be used for the S7-300/400 CPU and vice versa. The tasks of the two FBs are the same.

Function

Auxiliary block for FC BasicTask for processing a communication mailbox of the DB BComData type (S7-1500).

Via the DB, a configured PBK (programmed block communication) connection is handled using the SFBs "BSEND" and "BRCV".

FB BCom also ensures that received frames are distributed immediately to the receive objects responsible in the CPU. To do this FB BCom calls FC Distribute as an auxiliary block.

3.5.2 BConnect FB

Validity

S7-1500

Function

Auxiliary block for FC BasicTask for processing connections to the communication partners.

3.5.3 DB BConnectData

Validity

S7-1500

Function

Instance data block for the communication block FB BCom. The instance DB represents the communication mailbox and contains the following, among other things:

- The receive mailbox (ReceiveBuffer)
- A send mailbox (SendBuffer)

It also contains central data required to control and manage the PBK connection that runs via this mailbox.

The data block is required in every CPU in which FB BCom is used. If the CPU has several PBK connections the DB is required more than once.

DB BComData is automatically generated in the required length preset with the connection specific data and then saved in the block directory of the CPU.

3.6 Communication blocks BCom (CPU 300/400)

3.6.1 BCom FB

Validity

- S7-400
- S7-300

The blocks for the S7-300/400-CPU cannot be used for the S7-1500-CPU and vice versa. The tasks of the FBs are the same.

Function

Auxiliary block for FC BasicTask for processing a communication mailbox of the type DB BComData (S7-300/S7-400).

A configured PBK connection (programmed block communication) is handled via the DB using the "BSEND" and "BRCV" SFBs of an S7-400 CPU or the function blocks (FBs) of the same name of an S7-300 CPU.

FB BCom also ensures that received frames are distributed immediately to the receive objects responsible in the CPU. To do this FB BCom calls FC Distribute as an auxiliary block.

S7-300

Availability and function

The BCom FB can only be used as of block version V1.5 for an S7-300 CPU, . The block is included in the "Telecontrol ST7" library as of version V3.0 SP2. The block version can be found in the properties dialog of the block in the "TD7onCPU" container of the program blocks of the CPU.

The BCom V1.5 FB can be used as of STEP 7 Professional V17.

The block is required for an S7-300 CPU especially for communication via a local TIM 1531 IRC.

Number of configurable S7 connections

S7-300 CPU supports up to 16 S7 connections for TD7onCPU. The configurable number of S7 connections depends on the respective CPU type and cannot be exceeded even by CPUs with more connection resources.

3.6.2 DB BComData

Validity

S7-300

S7-400

Function

Instance data block for the communication block FB BCom. The instance DB represents the communication mailbox and contains the following, among other things:

- The receive mailbox (ReceiveBuffer)
- A send mailbox (SendBuffer)

It also contains central data required to control and manage the PBK connection that runs via this mailbox.

The data block is required in every CPU in which FB BCom is used. If the CPU has several PBK connections the DB is required more than once.

DB BComData is automatically generated in the required length preset with the connection specific data and then saved in the block directory of the CPU.

3.7 Communication blocks PCom (CPU 300)

3.7.1 FB PCom

Validity

S7-300 CPU without partyline

FB PCom is used only with communication via the P bus. This affects the communication between a TIM and a CPU with P bus.

Function

Auxiliary block for FC BasicTask_P for processing a communication mailbox of the type DB PComData using the SFCs WR_REC and RD_REC.

Received frames are also distributed immediately to the receive objects responsible in the CPU. To do this FB PCom calls FC Distribute as an auxiliary block.

3.7.2 DB PComData

Instance data block for the communication block FB PCom. The instance DB makes the communication mailbox available and contains the following:

- The receive mailbox (ReceiveBuffer)
- A send mailbox (SendBuffer)

It also contains central data required to control and manage the connection that runs via this mailbox.

The data block is required in every CPU in which FB PCom is used. If the CPU has several corresponding connections the DB is required more than once.

3.8 Communication blocks XCom (CPU 300)

3.8.1 FB XCom

Validity

S7-300 CPU with partyline

Function

Auxiliary block for FC BasicTask_X for processing a communication mailbox of the type DB XComData.

Via DB BComData a configured connection (PBK connection) is handled using the SFCs "X_SEND" und "X_RCV".

The FB XCom also ensures that received frames are distributed immediately to the receiving objects responsible in the CPU. To do this the FB XCom calls the FC Distribute as an auxiliary block.

3.8.2 DB XComData

Instance data block for the communication block FB XCom. The instance DB makes the communication mailbox available and contains the following:

- The receive mailbox (ReceiveBuffer)
- A send mailbox (SendBuffer)

It also contains central data required to control and manage the X connection that runs via this mailbox.

The data block is required in every CPU in which FB XCom is used. If the CPU has several X connections the DB is required more than once.

DB XComData is automatically generated in the required length preset with the connection specific data and then saved in the block directory of the CPU.

Glossary

1-out-of-8 check

Mechanism for interlocking multiple simultaneous commands.

When entering commands, there is a check to determine if only one command is pending at the time of acquisition. Transmission of the command byte is only triggered if there is a single modified command bit in the command byte compared to the last cycle. If several bits within the command byte have been changed, errors are detected and the command byte is not sent.

The function is performed by the data point typical "Cmd01B_S" of the TD7onCPU block library. The "FC Safe" block is also required.

1-out-of-n check

Mechanism for interlocking multiple simultaneous commands.

When entering commands, there is a check to determine if only one command is pending at the time of acquisition. Transmission of the command to the communication partner is only triggered when a single command is pending. If several commands are pending at the same time, errors are detected and the command is not sent or not issued by the receiver.

The function is executed by the "FC Safe" block of the TD7onCPU block library. The function is supported by the data point typicals "Cmd01B_S", "Par12D_S" and "Set01W_S".

APN

Access Point Name

DNS host name of the access point for an external network (in this case: access point in the GPRS network to the Internet).

Conditional spontaneous frame

→ *Spontaneous / conditional spontaneous / unconditional spontaneous frame*

CP

Communications processor

Module for expanded communications tasks that provides the CPU with additional interface types or communications options.

CPU

Central Processing Unit

Main processor of a SIMATIC controller

CSD

Circuit Switched Data

Service for transferring data in the GSM network. Possible are dial-in connections of GSM modems to GSM/ISDN/analog modems and other devices with modems. The transmission speed is 14400 bps full duplex for non-secure transmission and 9600 bps for secure transmission.

CTS

Clear to send

Signal in the data flow control

Data frame

Data unit transferred between communication partners. Meaning:

- Data unit transferred on the application layer (OSI layer 7)
- General term for a transferred data unit regardless of the relevant OSI layer.

Data frames contain the data of an ST7 object to be transmitted. Depending on the object type, a frame can contain either all data of an ST7 object or a contiguous subarea of the object data.

See also "Organizational frame"

Direct communication

With direct communication, the S7 stations communicate directly with each other without the frames needing to be forwarded by a master station or station.

See also "Inter-station communication"

Context: Telecontrol / SINAUT

DNP

Distributed Network Protocol

DSL

Digital Subscriber Line

Standards for transmission of telephone and Internet data with transmission speeds up to 1000 Mbps.

EGPRS

Enhanced GPRS

Packet-oriented service for IP-based data transmission in GSM networks. By using an additional modulation procedure (EDGE technology), a higher transmission speed is achieved compared with GPRS.

Engineering station

PC with the STEP 7 Professional project (TIA Portal)

Frame

→ *Data frame*

General request

- **General request (GR)**

  With a general request, a central station requests the current process image from the connected subscribers.

  Depending on the telecontrol protocol used, a GR can be started for a variety of reasons.

  With ST7, a GR is started automatically when a disrupted connection has been restored or when a failed partner reports a restart. Apart from the automatic general request, a GR can also be triggered by the user program or from the control center. TD7onTIM does not support general requests.

- **Xcelerated general request (XGR)**

  In an xcelerated general request, the frames with the requested process image are entered at the start of the send buffer of the TIM; in other words, before any other frames still buffered in the TIM. This enables you to have the current process image available more quickly.

GPRS

General Packet Radio Service

Packet-oriented service for IP-based data transmission in GSM networks. The data is transmitted using the Internet protocols TCP/IP or UDP/IP.

GSM

Global System for Mobile Communication

Worldwide standard for mobile communication (2G)

Image memory

Memory area for the process image in a telecontrol module

Each data frame is saved exactly one time in the image memory. New values of a data point overwrite the existing value in the image memory.

See also send buffer

Context: TeleControl

Image memory / send buffer principle

- **Image memory principle**

  A fixed position is reserved in the image memory for each data frame transferred to the TIM for transmission. Each newly transferred frame always overwrites the old frame in the image memory.

  If a send frame is entered using the image memory principle, only a reference to the location of the frame in the frame image memory is entered. If the TIM has not yet been able to transmit the frame when the same frame is transferred to it again, the frame is not entered in the send buffer a second time, but rather the image is simply updated.

  At the time of transmission, the frame is sent with its up-to-date content from the image memory. Only then can the frame be entered in the send buffer again.

  Transmission using the image memory principle achieves the following:

  – There is less load on transmission link, fewer frames are transmitted.

  – There is less load on the send buffer of the TIM; an image memory frame is entered a maximum of once in the send buffer.

- **Send buffer principle**

  If a data frame is transmitted using the send buffer principle, it is entered completely in the send buffer each time it is transferred to the TIM. If such a frame cannot or should not be transmitted immediately, it may therefore exist more than once in the send buffer.

  When it is sent, the frame is taken completely from the send buffer and transmitted.

Inter-station communication

Communication between two stations, which is mediated by a Telecontrol master station.

In dial-up networks, a direct connection between the two stations is established.

See also "Direct communication".

IRC

Industrial Remote Communication

SIMATIC NET product group for Telecontrol

ISDN

Integrated Services Digital Network

Standard for a digital transmission network for telephone, telefax, telex, teletext and datex-J/L/P services. The data of various services can be transferred simultaneously. Telephone connections normally operate at transmission speeds of 56 to 64 Kbps.

LAN

Local Area Network

Local network, usually "Industrial Ethernet".

Local CPU

CPU assigned to a TIM.

Local TIM

TIM that is connected to a CPU or a PC (ST7cc, ST7sc) via an IP-based network.

LTOP

Line Transformer with Overvoltage Protection

Overvoltage protection module of the SINAUT device program - discontinued

Main cycle / sub-cycle

The sequence of the polling cycle can be structured on the master TIM by assigning individual polling stations to the main cycle or the sub-cycle.

The subcycle is always activated at the end of the main cycle once all stations from the main cycle have been polled. A configurable number of stations is called in a subcycle.

Following this, all the stations in the main cycle are polled again. This is followed by a subcycle in which further stations that are assigned to the subcycle are called.

Master station

Station in the top hierarchy of a telecontrol network. It is connected to the control system and the substations or node stations.

The interfaces of a master module are set to the network node type "Master station".

MCC - Mobile Country Code

→ *PLMN*

Messages

Emails and SMS in the TeleControl context

See also Data frame.

MNC - Mobile Network Code

→ *PLMN*

MPI

Multi Point Interface

MPI is the programming device interface of the SIMATIC S7-300/400. Devices such as the TIM can communicate with each other via the MPI interface.

See also Partyline.

MSC

The MSC transmission protocol is a proprietary protocol on OSI layer 3 for the secure communication via Ethernet, landline or mobile wireless networks in SINAUT ST7. The MSC protocol provides an authentication mechanism and simple encryption of data.

The protocol is available in the MSC and MSCsec versions (with cyclic key exchange).

MSCsec

→ *MSC*

Multi-master polling with time slots

When stations need to communicate with more than one master station in dedicated line or wireless operation, the multimaster polling with time slots mode is used. Each of the connected master stations is assigned one or more defined time slots per minute for polling the stations. The master stations then have their turn to poll in every minute.

Node station

A node station is a station located between the master station and stations in the hierarchy of a telecontrol network. One or more subordinate stations are connected to a node station. The data traffic between these stations and the master station is handled via the node station. Direct data exchange between the node station and the subordinate stations is also possible. Multiple node station levels are possible in a SINAUT network.

Organizational frame

Organizational frames are used to execute organizational system functions, for example:

- General requests
- Time-of-day synchronization
- Counted value storage
- Coordinated connection establishment and termination in a dial-up network
- Message indicating station startup and station failure
- Requests for and transmission of subscriber records

Party line

- Party line CPUs are:
  - CPU 312/313/314/315 to CPU 315-2 DP
  - C7 devices

  The communication bus of the smaller S7-300 CPUs is physically wired through to the MPI interface of the CPU.

  With party line CPUs you can use every type of TIM. You will find details on the Internet at the following address:

  https://support.industry.siemens.com/cs/ww/en/view/24059469
- Non-party line CPUs are:
  - CPU 315-2 PN/DP to CPU 319-3 PN/DP

  With non-party line CPUs, the MPI interface and communications bus are separate.

Permanent call

A permanent call does not interrupt the normal polling cycle; it is always executed alternating with the standard poll from the normal polling cycle.

PG

Programming device

Allows access by the STEP 7 configuration software to the SIMATIC CPU.

PG routing

Using PG routing, it is possible to access programmable modules or modules with diagnostics capability beyond network boundaries from a programming device (PG) or computer (PC).

PLMN

Public Land Mobile Network

Worldwide unique identifier of mobile networks. The PLMN is made up of the three-digit Mobile Country Code (MCC) and the two-or three-digit Mobile Network Code (MNC) of the network provider.

Polled frames

Polled data frames are data frames of a station or node TIM with a special identifier indicating that they were sent in response to a general request from the master station.

Polling

→ *Polling mode*

Polling mode

The polling mode is a method of data transmission in which a central instance controls the data exchange with the communication partners.

Using a polling frame, the master TIM instructs the connected station TIMs one after the other to transmit their stored data frames to the master TIM. If a polled station has no stored data, it responds with an acknowledgment frame and the polling cycle then continues by polling the next station.

A station that has stored data sends a single data frame or, if block transfer was configured, several data frames in a block.

If the TIM has stored additional data, it indicates this in the response frame. In this case, the station is then immediately called up again until the stored data has been transferred.

Polling with time slots

The polling with time slots mode is used in a wireless network in which the use of the radio frequency assigned by the registration authorities must be shared with other users. Each user typically has 6 seconds per minute to exchange data with its stations. The frequency must then be released for other operators. During the allocated time slot, this pooling variant functions like a normal polling system.

Context: SINAUT ST7

Protocol

A protocol is a set of rules for controlled transfer of data. Protocols, for example, specify the data structure, the structure of data packets and the coding. Protocols can also specify a control mechanisms and hardware and software requirements.

RS-232

RS-232 is a standard for serial (bit-by bit) data transmission with +12 V and -12 V signals. RS-232 is a Recommended Standard of the Electronic Industries Association. 9-pin and 25-pin connections with D-sub connectors (subminiature connector with D-shaped surface area) are normal for the RS-232 interface.

RS-485

RS-485 is a standard for data transmission with 5 V differential signals. The RS-485 interface uses only one pair of wires and is operated in half duplex. The connection is multipoint-compliant; in other words, up to 32 subscribers can be connected.

RTS

Request to send

Signal in the data flow control

S0 interface

Basic interface of ISDN for connecting end devices

Send buffer principle

→ *Image memory / send buffer principle*

SIM card

SIM - Subscriber Identity Module

The SIM card is an identification card for a subscriber of a mobile wireless service.

Simple Internet communication

In SINAUT ST7, simple Internet communication means data exchange between TCP/IP-compliant devices in Ethernet, landline or mobile networks using the MSC protocol.

SINAUT

Siemens Network Automation

Station control system or telecontrol system based on SIMATIC S7.

SINAUT ST7 works with the SINAUT ST7 telecontrol protocol.

SINAUT object

A SINAUT object contains the data of one or more process variables such as analog values, commands, calculated values, status information on motors, sliders etc. An ST7 object has type-specific processing functions and change checks assigned to it to minimize the communication traffic in the WAN. Type-specific processing functions include, for example, threshold checks or mean value calculation with the object type for analog values. The change check is designed so that a frame is generated only when the object data has changed compared with the last time it was transferred.

SINAUT ST7

Proprietary telecontrol protocol for SIMATIC NET telecontrol modules

SINAUT ST7cc

Control center system based on SIMATIC WinCC for SINAUT ST7.

SINAUT ST7sc

System for networking SIMATIC stations with a control station via WAN. The control center can also be a SIMATIC station or a PC-based control center, for example, WinCC with the SINAUT ST7cc add-on.

SINAUT TD7 Library

Software for control of ST7 communication of telecontrol modules. The TD7 software in the stations allows change-controlled transmission of process data between the individual CPUs and the control center, for example ST7cc. Failure of connections, CPUs, or the control center are displayed. Once a problem has been corrected or the CPUs or control center have started up, data is updated automatically. Data frames can be given a time stamp, if required.

The following variants of the TD7 software exist:

- **TD7onCPU**

  Program blocks in the CPU user program

  The SINAUT TD7 library consists of program blocks for the CPU. They are available in the following versions:

  – Library for STEP 7 V5

    The blocks are executable on S7-300- and S7-400-CPUs (except CPU 400H). There are only a few blocks intended specifically for the S7-300 or S7-400.

  – Libraries for STEP 7 Professional

    There is a global library with two versions for STEP 7 projects in the TIA Portal:

    - Blocks for S7-1500

    - Blocks for S7-300 and S7-400

  TD7onCPU is not supported when using the DNP3 and IEC 60870-5 protocols.

- **TD7onTIM**

  Configurable part of the firmware of the communications module

  TD7onTIM can be used as an alternative to TD7onCPU for an Ethernet TIM. TD7onTIM runs on the communications module and is configured as follows:

  – STEP 7 V5: In the SINAUT engineering software

  – STEP 7 Professional: Via the data points of the communications module

    With CPs (S7-1200 / ET 200SP), TD7onTIM is the only variant that can be selected.

TD7onCPU and TD7onTIM cannot be used simultaneously in a station.

SMS

Short Message Service

The short message service in the GSM standard is used to transfer short text messages to mobile wireless subscribers.

When the short messages are transferred, they are first transferred to the SMS center (SMSC) using a store-and-forward technique. They are buffered there and then forwarded to the recipient. The sender can query the status of the message in the SMS center or can request acknowledgment of delivery.

SMSC

Short Message Service Center

When sending an SMS message, the message is first sent to the SMSC, buffered there and then forwarded to the recipient.

Spontaneous / conditional spontaneous / unconditional spontaneous frame

- **Spontaneous frame**

  In SINAUT networks, data frames are always transmitted spontaneously; in other words, data are created and transmitted only when changes to process values occur or event-controlled. These frames are known as spontaneous frames.

- **Conditional spontaneous frame**

  In the dial-up network, you can specify whether or not a change causes a "conditional spontaneous" or "unconditional spontaneous" transmission for each individual frame.

  Conditional spontaneous frames are initially only entered in the send buffer of the TIM. They are only transmitted when a connection is established to the partner for whatever reason, for example because an unconditional spontaneous frame needs to be transmitted or because the partner calls.

  Even when using pay by volume transmission in a GPRS network, frame prioritization "conditional spontaneous" can also be used. Such a frame is not transmitted immediately, but is first buffered. In a GPRS network, the TIM stores "conditional spontaneous" frames in the following situations:

  – When the collected frames reach or exceed a size of 202 bytes.
  – An important frame ("unconditional spontaneous" or "spontaneous" priority) should be transmitted immediately.
  – The collected frames have not yet reached a volume of 202 bytes, but the TCP/IP keep-alive interval expires.
  – The fill level of the send buffer has reached 90% of its maximum capacity.

- **Unconditional spontaneous frame**

  In the dial-up network, you can specify whether or not a change causes an "conditional spontaneous" or "unconditional spontaneous" transmission for each individual frame.

  Unconditional spontaneous frames cause the connection to be established immediately. Even with pay-by-volume/time transmission in a GPRS network, you can use the frame prioritization "unconditional spontaneous"; in other words, in contrast to a "conditional spontaneous" frame, a frame is transmitted immediately.

Spontaneous mode

Spontaneous mode is a method of ST7 data transmission in which subscribers can exchange data directly amongst themselves. Here, no central entity is necessary in the form of a master TIM as in polling mode (see "Polling mode"). The spontaneous mode is intended for data transmission in dial-up networks and for communication via IP-based networks.

For transmission in a dial-up network and in IP-based networks (for example GPRS), the data to be sent is assigned different priorities during configuration ("high" / "normal").

When data with high priority is ready for transmission, a connection is established immediately.

If the data has "normal" priority, it is first stored in the communications module. It is sent the next time a connection is established to the partner. This can be the case, for example, when data is to be transferred with high priority or when the partner establishes a connection.

ST7 protocol

→ *SINAUT ST7*

ST7cc

→ *SINAUT ST7cc*

ST7sc

→ *SINAUT ST7sc*

Station

- Hardware

  SIMATIC controller with the required components for acquisition, processing and communication, consisting of: CPU, I/O modules, communications module, modem, etc.

- Network node type

  Setting a WAN interface of the TIM. An interface of the "Station" network node type communicates at the lowest hierarchy level in a SINAUT network.

Subcycle

→ *Main cycle / sub-cycle*

TA

Frame type

- 0 = Spontaneous Org. frames
- 1 = Requested Org. frames
- 2 = Spontaneous data frames
- 3 = Requested data frames

TD7 software

→ *SINAUT TD7 Library*

TIM

Telecontrol Interface Module

Communications module that handles all data transmission functions provided by the SINAUT system independently.

Unconditional spontaneous frame

→ *Spontaneous / conditional spontaneous / unconditional spontaneous frame*

VPN

Virtual Private Network

Technology for secure transportation of confidential data in public IP networks, for example the Internet.

WAN

Wide Area Network

Data network with a large geographical span, such as the Internet. telephone or enterprise networks. We distinguish between the following WAN networks:

- **WAN, classic**

  Includes dedicated lines (private or leased), private wireless networks, analog telephone network, digital ISDN network and mobile networks (without Internet).

  A classic WAN is connected via suitable transmission device (modem) to a serial interface of the TIM.

- **WAN, IP-based**

  Includes IP-based networks with telecontrol communication via wireless or fiber-optic cables, public networks and the Internet using services such as DSL, GPRS, UMTS or LTE or via broadband systems such as OTN and PCM30.

  An IP-based WAN is normally connected to an RJ45 interface of a module via an Ethernet-capable module.

Xcelerated general request

→ *General request*