VIPA

Standard | Operation List | Manual

HB00E_OPL_STD | Rev. 14/22

May 2014

VIPA
A YASKAWA COMPANY

VIPA, Gesellschaft für Visualisierung und Prozessautomatisierung mbH
Ohmstraße 4, D-91074 Herzogenaurach, Germany
Tel.: +49 (91 32) 744 -0
Fax.: +49 9132 744 1864
EMail: info@vipa.de
http://www.vipa.com

Note

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

CE Conformity Declaration

Hereby, VIPA GmbH declares that the products and systems are in compliance with the essential requirements and other relevant provisions.

Conformity is indicated by the CE marking affixed to the product.

Conformity Information

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

Trademarks

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

SPEED7 is a registered trademark of profichip GmbH.

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

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

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

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

Information product support

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

VIPA GmbH, Ohmstraße 4, 91074 Herzogenaurach, Germany

Telefax:+49 9132 744 1204
EMail: documentation@vipa.de

Technical support

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

VIPA GmbH, Ohmstraße 4, 91074 Herzogenaurach, Germany

Telephone: +49 9132 744 1150 (Hotline)
EMail: support@vipa.de

Contents

About this manual

This manual provides you with a comprehensive overview of the blocks integrated to the VIPA Standard CPUs System 100V, 200V, 300V and 500V.

Described are command list, integrated OBs, SFBs, SFCs and the VIPA specific blocks.

Outline

Chapter 1: Instruction list

This chapter lists the available commands of the standard CPUs from VIPA of the Systems 100V, 200V, 300V and 500V. The instruction list intends to give you an overview over the commands and their syntax. The commands are sorted by topics in alphabetical order.

Chapter 2: Organization Blocks

In this Chapter the description of the integrated organization blocks of the VIPA standard CPUs of Systems 100V, 200V, 300V and 500V may be found.

Chapter 3: Integrated SFBs

The description of the integrated function blocks of the VIPA standard CPUs of the systems 100V, 200V, 300V and 500V may be found here.

Chapter 4: Integrated Standard SFCs

The integrated standard SFCs of the VIPA standard CPUs of the systems 100V, 200V, 300V and 500V are described in this chapter.

Chapter 5: VIPA specific blocks

Here the VIPA specific blocks are described that are exclusively may be used with the standard CPUs from VIPA of the Systems 100V, 200V, 300V and 500V. Please note that not every block listed here is integrated in every system CPU. The assignment to the according system may be in the table of the "Overview".

Objective and contents

This manual provides you with the instruction list and the description of the integrated blocks that are exclusively may be used with the VIPA Standard CPUs System 100V, 200V, 300V and 500V.

Target audience

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

Structure of the manual

The manual consists of chapters. Every chapter provides a self-contained description of a specific topic.

Guide to the document

The following guides are available in the manual:

- an overall table of contents at the beginning of the manual
- an overview of the topics for every chapter
- an index at the end of the manual.

Availability

The manual is available in:

- printed form, on paper
- in electronic form as PDF-file (Adobe Acrobat Reader)

Icons Headings

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

Danger!
Immediate or likely danger.
Personal injury is possible.

Attention!
Damages to property is likely if these warnings are not heeded.

Note!
Supplementary information and useful tips.

Chapter 1 Instruction list

Overview

The following chapter lists the available commands of the standard CPUs from VIPA of the Systems 100V, 200V, 300V and 500V. The instruction list intends to give you an overview over the commands and their syntax. The commands are sorted by topics in alphabetical order.

Via the content the different topics are available.

The alphabetical instruction list gives you direct access to the instructions.

For the parameters are integrated in the instruction list, there is no extra parameter list.

Content

Alphabetical instruction list

Abbreviations

Abbreviation	Description
/FC	First check bit
2#	Binary constant
a	Byte address
ACCU	Register for processing bytes, words and double words
AR	Address registers, contain the area-internal or area-crossing addresses for the instructions addressed register-indirect
b	Bit address
B	area-crossing, register-indirect addressed byte
B (b1,b2)	Constant, 2byte
B (b1,b2,b3,b4)	Constant, 4byte
B#16#	Byte hexadecimal
BR	Binary result
c	Operand range
C	Counter
C#	Counter constant (BCD-coded)
CC0	Condition code
CC1	Condition code
D	area-crossing, register-indirect addressed double word
D#	IEC date constant
DB	Data block
DBB	Data byte in the data block
DBD	Data double word in the data block
DBW	Data word in the data block
DBX	Data bit in the data block
DI	Instance data block
DIB	Data byte in the instance DB
DID	Data double word in the instance DB
DIW	Data word in the instance DB
DIX	Data bit in the instance DB
DW#16#	Double word hexadecimal
f	Timer/Counter No.
FB	Function block
FC	Functions
g	Operand range
h	Operand range
I	Input (in the PII)
i	Operand range
i8	Integer (8bit)
i16	Integer (16bit)
i32	Integer (32bit)
IB	Input byte (in the PII)
ID	Input double word (in the PII)
IW	Input word (in the PII)
k8	Constant (8bit)
k16	Constant (16bit)
k32	Constant (32bit)

continued ...

... continue

Abbreviation	Description
L	Local data
L#	Integer constant (32bit)
LABEL	Symbolic jump address (max. 4 characters)
LB	Local data byte
LD	Local data double word
LW	Local data word
m	Pointer constant P#x.y (pointer)
M	Bit memory bit
MB	Bit memory byte
MD	Bit memory double word
MW	Bit memory word
n	Binary constant
OB	Organization block
OR	Or
OS	Stored overflow
OV	Overflow
p	Hexadecimal constant
P#	Pointer constant
PIQ	Process image of the outputs
PII	Process image of the inputs
PIB	Periphery input byte (direct periphery access)
PID	Periphery input double word (direct periphery access)
PIW	Periphery input word (direct periphery access)
PQB	Periphery output byte (direct periphery access)
PQD	Periphery output double word (direct periphery access)
PQW	Periphery output word (direct periphery access)
Q	Output (in the PIQ)
q	Real number (32bit floating-point number)
QB	Output byte (in the PIQ)
QD	Output double word (in the PIQ)
QW	Output word (in the PIQ)
r	Block no.
RLO	Result of (previous) logic instruction
S5T#	S5 time constant (16bit), loads the S5-Timer
SFB	System function block
SFC	System function
STA	Status
T	Timer (times)
T#	Time constant (16/32bit)
TOD#	IEC time constant
W	area-crossing, register-indirect addressed word
W#16#	Word hexadecimal

Registers

ACCU1 and ACCU2 (32bit)

The ACCUs are registers for the processing of byte, words or double words. Therefore the operands are loaded in the ACCUs and combined. The result of the instruction is always in ACCU1.

ACCU	Bit
ACCUx (x=1 to 2)	Bit 0 ... bit 31
ACCUx-L	Bit 0 ... bit 15
ACCUx-H	Bit 16 ... bit 31
ACCUx-LL	Bit 0 ... bit 7
ACCUx-LH	Bit 8 ... bit 15
ACCUx-HL	Bit 16 ... bit 23
ACCUx-HH	Bit 24 ... bit 31

Address register AR1 and AR2 (32bit)

The address registers contain the area-internal or area-crossing addresses for the register-indirect addressed instructions. The address registers are 32bit wide.

The area-internal or area-crossing addresses have the following structure:

area-internal address:

00000000 00000bbb bbbbbbbb bbbbbxxx

area-crossing address:

10000yyy 00000bbb bbbbbbbb bbbbbxxx

Legend:

b	Byte address
x	Bit number
Y	Range ID (see chapter "Addressing examples")

Status word (16bit)

The values are analyzed or set by the instructions.

The status word is 16bit wide.

Bit	Assignment	Description
0	/FC	First check bit*
1	RLO	Result of (previous) logic instruction
2	STA	Status*
3	OR	Or*
4	OS	Stored overflow
5	OV	Overflow
6	CC0	Condition code
7	CC1	Condition code
8	BR	Binary result
9 ... 15	not used	-

* Bit may not be analyzed with the instruction L STW in the user application, for the bit isn't updated during application run-time.

Addressing examples

Addressing example	Description
Immediate addressing	
L +27	Load 16bit integer constant "27" in ACCU1
L L#-1	Load 32bit integer constant "-1" in ACCU1
L 2#1010101010101010	Load binary constant in ACCU1
L DW#16#A0F0_BCFD	Load hexadecimal constant in ACCU1
L 'End'	Load ASCII code in ACCU1
L T#500ms	Load time value in ACCU1
L C#100	Load counter value in ACCU1
L B#(100,12)	Load constant as 2byte
L B#(100,12,50,8)	Load constant as 4byte
L P#10.0	Load area-internal pointer in ACCU1
L P#E20.6	Load area-crossing pointer in ACCU1
L -2.5	Load real number in ACCU1
L D#1995-01-20	Load date
L TOD#13:20:33.125	Load time-of-day
Direct addressing	
A I 0.0	AND operation of input bit 0.0
L IB 1	Load input byte 1 in ACCU1
L IW 0	Load input word 0 in ACCU1
L ID 0	Load input double word 0 in ACCU1
Indirect addressing timer/counter	
SP T [LW 8]	Start timer; timer no. is in local data word 8
CU C [LW 10]	Start counter; counter no. is in local data word 10
Memory-indirect, area-internal addressing	
A I [LD 12] e.g.: LP#22.2 T LD 12 A I [LD 12]	AND instruction; input address is in local data double word 12 as pointer
A I [DBD 1]	AND instruction; input address is in data double word 1 of the DB as pointer
A Q [DID 12]	AND instruction; output address is in data double word 12 of the instance DB as pointer
A Q [MD 12]	AND instruction; output address is in bit memory double word 12 as pointer
Register-indirect, area-internal addressing	
A I [AR1,P#12.2]	AND instruction; input address is calculated "pointer value in address register 1 + pointer P#12.2"

continued ...

... continue

Register-indirect, area-crossing addressing			
For the area-crossing, register indirect addressing the address needs an additional range-ID in the bits 24-26. The address is in the address register.			
Range-ID	**Binary code**	**hex.**	**Area**
P	1000 0**000**	80	Periphery area
I	1000 0**001**	81	Input area
Q	1000 0**010**	82	Output area
M	1000 0**011**	83	Bit memory area
DB	1000 0**100**	84	Data area
DI	1000 0**101**	85	Instance data area
L	1000 0**110**	86	Local data area
VL	1000 0**111**	87	Preceding local data area (access to the local data of the calling block)

L B [AR1,P#8.0]	Load byte in ACCU1; the address is calculated "pointer value in address register 1 + pointer P#8.0"
A [AR1,P#32.3]	AND instruction; operand address is calculated "pointer value in address register 1 + pointer P#32.3"
Addressing via parameters	
A parameter	The operand is addressed via the parameter

Example for pointer calculation

Example when sum of bit addresses ≤ 7:

```
LAR1 P#8.2
A    I [AR1,P#10.2]
```

Result: The input 18.4 is addressed (by adding the byte and bit addresses)

Example when sum of bit addresses > 7:

```
L MD 0          at will calculated pointer, e.g. P#10.5
LAR1
A    I [AR1,P#10.7]
```

Result: Addressed is input 21.4 (by adding the byte and bit addresses with carry).

Command	Operand	Parameter	Status word									Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
												: Instruction depends on	
												: Instruction influences	

Math instructions

Fixed-point arithmetic (16bit)			*Status word*									Math instructions of two 16bit numbers. The result is in ACCU1 res. ACCU1-L.	
+I	-		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Add up two integers (16bit)	1
			-	-	-	-	-	-	-	-	-	(ACCU1-L)=(ACCU1-L)+(ACCU2-L)	
-I	-		-	Y	Y	Y	Y	-	-	-	-	Subtract two integers (16bit)	1
												(ACCU1-L)=(ACCU2-L)-(ACCU1-L)	
*I	-											Multiply two integers (16bit)	1
												(ACCU1-L)=(ACCU2-L)*(ACCU1-L)	
/I	-											Divide two integers (16bit)	1
												(ACCU1-L)=(ACCU2-L):(ACCU1-L)	
												The remainder is in ACCU1-H	
Fixed-point arithmetic (32bit)			*Status word*									Math instructions of two 32bit numbers. The result is in ACCU1.	
+D	-		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Add up two integers (32bit)	1
			-	-	-	-	-	-	-	-	-	(ACCU1)=(ACCU2)+(ACCU1)	
-D	-		-	Y	Y	Y	Y	-	-	-	-	Subtract two integers (32bit)	1
												(ACCU1)=(ACCU2)-(ACCU1)	
*D	-											Multiply two integers (32bit)	1
												(ACCU1)=(ACCU2)*(ACCU1)	
/D	-											Divide two integers (32bit)	1
												(ACCU1)=(ACCU2):(ACCU1)	
MOD	-											Divide two integers (32bit) and load the rest of the	1
												division in ACCU1	
												(ACCU1)=remainder of [(ACCU2):(ACCU1)]	
Floating-point arithmetic (32bit)			*Status word*									The result of the math instructions is in ACCU1. The execution time of the instruction depends on the value to calculate.	
+R	-		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Add up two real numbers (32bit)	1
			-	-	-	-	-	-	-	-	-	(ACCU1)=(ACCU2)+(ACCU1)	
-R	-		-	Y	Y	Y	Y	-	-	-	-	Subtract two real numbers (32bit)	1
												(ACCU1)=(ACCU2)-(ACCU1)	
*R	-											Multiply two real numbers (32bit)	1
												(ACCU1)=(ACCU2)*(ACCU1)	
/R	-											Divide two real numbers (32bit)	1
												(ACCU1)=(ACCU2):(ACCU1)	
NEGR	-		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Negate the real number in ACCU1	1
			-	-	-	-	-	-	-	-	-		
ABS	-		-	-	-	-	-	-	-	-	-	Form the absolute value of the real number in ACCU1	1

Command	Operand	Parameter	Status word BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Function	Length in words
												: Instruction depends on	
												: Instruction influences	

			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
Square root an square instructions (32bit)			*Status word*									The result of the instructions is in ACCU1. The instructions may be interrupted by alarms.	
SQRT	-		-	-	-	-	-	-	-	-	-	Calculate the Square root of a real number in ACCU1	1
SQR	-		-	Y	Y	Y	Y	-	-	-	-	Form the square of a real number in ACCU1	1
Logarithmic function (32bit)			*Status word*									The result of the logarithm function is in ACCU1. The instructions may be interrupted by alarms.	
LN	-		-	-	-	-	-	-	-	-	-	Calculate the natural logarithm of a real number in ACCU1	1
EXP	-		-	Y	Y	Y	Y	-	-	-	-	Calculate the exponential value of a real number in ACCU1 on basis e (=2.71828)	1
Trigonometrical functions (32bit)			*Status word*									The result of the trigonometrical function is in ACCU1. The instructions may be interrupted by alarms.	
SIN[1]	-		-	-	-	-	-	-	-	-	-	Calculate the sine of the real number	1
ASIN[2]	-		-	Y	Y	Y	Y	-	-	-	-	Calculate the arcsine of the real number	1
COS[1]	-											Calculate the cosine of the real number	1
ACOS[2]	-											Calculate the arccosine of the real number	1
TAN[1]	-											Calculate the tangent of the real number	1
ATAN[2]	-											Calculate the arctangent of the real number	1
Addition of constants												Addition of integer constants to ACCU1. The condition code bits are not affected.	
+	i8											Add an 8bit integer constant	1
+	i16											Add a 16bit integer constant	2
+	i32											Add a 32bit integer constant	3
Addition via address register												Adding a 16bit integer to contents of address register. The value is in the instruction or in ACCU1-L. Condition code bits are not affected	
+AR1	-											Add the contents of ACCU1-L to AR1	1
+AR1	m											Add a pointer constant to the contents of AR1	2
+AR2	-											Add the contents of ACCU1-L to those of AR2	1
+AR2	m											Add pointer constant to the contents of AR2	2

1 Specify the angle in radians; the angle must be given as a floating point value in ACCU 1.

2 The result is an angle in radians.

Command	Operand	Parameter	Status word									Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
												: Instruction depends on	
												: Instruction influences	

Block instructions

Block call instructions			*Status word*										
CALL	FB r, DB r		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Unconditional call of a FB, with parameter transfer	1
			-	-	-	-	-	-	-	-	-		
CALL	SFB r, DB r		-	-	-	-	0	0	1	-	0	Unconditional call of a SFB, with parameter transfer	2
CALL	FC r											Unconditional call of a function, with parameter transfer	1
CALL	SFC r											Unconditional call of a SFC, with parameter transfer	2
UC	FB r FC r Parameter											Unconditional call of blocks, without parameter transfer FB/FC call via parameters	1
CC	FB r		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Conditional call of blocks, without parameter transfer	1
	FC r		-	-	-	-	-	-	-	Y	-		
	Parameter		-	-	-	-	0	0	1	-	0	FB/FC call via parameters	
OPN			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
	DB r		-	-	-	-	-	-	-	-	-	Open a data block	1/2
	DI r		-	-	-	-	-	-	-	-	-	Open a instance data block	2
	Parameter											Open a data block via parameter	2
Block end instructions			*Status word*										
BE			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	End block	1
			-	-	-	-	-	-	-	-	-		
BEU			-	-	-	-	0	0	1	-	0	End block unconditionally	1
BEC			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	End block if RLO="1"	
			-	-	-	-	-	-	-	Y	-		
			-	-	-	-	Y	0	1	1	0		
Exchanging shared data block an instance data block												Exchanging the two current data blocks. The current shared data block becomes the current instance data block and vice versa. The condition code bits are not affected.	
CDB												Exchange shared data block and instant data block	1

Command	Operand	Parameter	Status word									Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
												: Instruction depends on	
												: Instruction influences	

Program display and null instruction instructions

Program display and null operation instructions				The status word is not affected.	
BLD	0 ... 255			Program display instruction: is treated by the CPU like a null operation instruction	1
NOP	0 1			Null operation instruction	1

Edge-triggered instructions

Edge-triggered instructions			*Status word*									Detection of an edge change. The current signal state of the RLO is compared with the signal state of the instruction or edge bit memory. FP detects a change in the RLO from "0" to "1". FN detects a change in the RLO from "1" to "0".	
FP	I/Q a.b	0.0 ... 127.7	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Detecting the positive edge in the RLO. The bit addressed	2
	M a.b	0.0 ...1023.7	-	-	-	-	-	-	-	Y	-	in the instruction is the auxiliary edge bit memory	2
	L a.b	0.0 ...1043.7	-	-	-	-	-	0	Y	Y	1		2
	DBX a.b	0.0 ...8191.7											2
	DIX a.b	0.0 ...8191.7											2
	c [AR1,m]												2
	c [AR2,m]												2
	[AR1,m]												2
	[AR2,m]												2
	Parameter												2
FN	I/Q a.b	0.0 ...127.7	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Detecting the negative edge in the RLO. The bit addressed	2
	M a.b	0.0 ...1023.7	-	-	-	-	-	-	-	Y	-	in the instruction is the auxiliary edge bit memory	2
	L a.b	0.0 ...1043.7	-	-	-	-	-	0	Y	Y	1		2
	DBX a.b	0.0 ...8191.7											2
	DIX a.b	0.0 ...8191.7											2
	c [AR1,m]												2
	c [AR2,m]												2
	[AR1,m]												2
	[AR2,m]												2
	Parameter												2

Command	Operand	Parameter	Status word									Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
												: Instruction depends on	
												: Instruction influences	

Load instructions

Load instructions				Loading address identifiers into ACCU1. The contents of ACCU1 and ACCU2 are saved first. The status word is not affected.	
L				Load ...	
	IB a	0 ... 127		input byte	1/2
	QB a	0 ... 127		output byte	1/2
	PIB a	0 ... 1023		periphery input byte	2
	MB a	0 ... 1023		bit memory byte	1/2
	LB a	0 ... 1043		local data byte	2
	DBB a	0 ... 8191		data byte	2
	DIB a	0 ... 8191		instance data byte	2
				... in ACCU1	2
	g [AR1,m]			register-indirect, area-internal (AR1)	2
	g [AR2,m]			register-indirect, area-internal (AR2)	2
	B [AR1,m]			area-crossing (AR1)	2
	B [AR2,m]			area-crossing (AR2)	2
	Parameter			via parameters	2
L				Load ...	
	IW a	0 ... 126		input word	1/2
	QW a	0 ... 126		output word	1/2
	PIW a	0 ... 1022		periphery input word	
	MW a	0 ... 1022		bit memory word	1/2
	LW a	0 ... 1042		local data word	2
	DBW a	0 ... 8190		data word	1/2
	DIW a	0 ... 8190		instance data word	1/2
				... in ACCU1-L	
	h [AR1,m]			register-indirect, area-internal (AR1)	2
	h [AR2,m]			register-indirect, area-internal (AR2)	2
	W [AR1,m]			area-crossing (AR1)	2
	W [AR2,m]			area-crossing (AR2)	2
	Parameter			via parameters	2

Command	Operand	Parameter	Status word									Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
												: Instruction depends on	
												: Instruction influences	

L				Load ...	
	ID a	0 ... 124		input double word	1/2
	QD a	0 ... 124		output double word	1/2
	PID a	0 ... 1020		periphery input double word	2
	MD a	0 ... 1020		bit memory double word	1/2
	LD a	0 ... 1040		local data double word	2
	DBD a	0 ... 8188		data double word	2
	DID a	0 ... 8188		instance data double word	2
				... in ACCU1-L	
	i [AR1,m]			register-indirect, area-internal (AR1)	2
	i [AR2,m]			register-indirect, area-internal (AR2)	2
	D [AR1,m]			area-crossing (AR1)	2
	D [AR2,m]			area-crossing (AR2)	2
	Parameter			via parameters	2
L				Load ...	
	k8			8bit constant in ACCU1-LL	1
	k16			16bit constant in ACCU1-L	2
	k32			32bit constant in ACCU1	3
	Parameter			Load constant in ACCU1 (addressed via parameters)	2
L	2#n			Load 16bit binary constant in ACCU1-L	2
				Load 32bit binary constant in ACCU1	3
L	B#8#p			Load 8bit hexadecimal constant in ACCU1-LL	1
	W#16#p			Load 16bit hexadecimal constant in ACCU1-L	2
	DW#16#p			Load 32bit hexadecimal constant in ACCU1	3
L	x			Load one character	
L	xx			Load two characters	2
L	xxx			Load three characters	
L	xxxx			Load four characters	3
L	D# Date			Load IEC-date (BCD-coded)	3
L	S5T# time value			Load time constant (16bit)	2
L	TOD# time value			Load 32bit time constant (IEC-time-of-day)	3
L	T# time			Load 16bit time constant	2
	value			Load 32bit time constant	3
L	C# counter value			Load 16bit counter constant	2
L	P# bit pointer			Load bit pointer	3
L	L# Integer			Load 32bit integer constant	3
L	Real			Load real number	3

Command	Operand	Parameter	Status word									Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
												: Instruction depends on	
												: Instruction influences	

Load instructions for timer and counter				Load a time or counter value in ACCU1, before the recent content of ACCU1 is saved in ACCU2. The status word is not affected.	
L	T f	0 ... 255		Load time value	1/2
	Timer para.			Load time value (addressed via parameters)	2
L	C f	0 ... 255		Load counter value	1/2
	Counter para.			Load counter value (addressed via parameters)	2
LD	T f	0 ... 255		Load time value BCD-coded	1/2
	Timer para.			Load time value BCD-coded (addressed via parameters)	2
LD	C f	0 ... 255		Load counter value BCD-coded	1/2
	Counter para.			Load counter value BCD-coded (addressed via parameters)	2

Command	Operand	Parameter	Status word									Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
												: Instruction depends on	
												: Instruction influences	

Shift instructions

Shift instructions			*Status word*									Shifting the contents of ACCU1 and ACCU1-L to the left or right by the specified number of places. If no address identifier is specified, shift the number of places into ACCU2-LL. Any positions that become free are padded with zeros or the sign. The last shifted bit is in condition code bit CC1.	
SLW	-		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Shift the contents of ACCU1-L to the left	1
SLW	0 ... 15		-	-	-	-	-	-	-	-	-	Positions that become free are provided with zeros	
SLD	-		-	Y	Y	Y	-	-	-	-	-	Shift the contents of ACCU1 to the left	1
SLD	0 ... 32											Positions that become free are provided with zeros	
SRW	-											Shift the contents of ACCU1-L to the right	1
SRW	0 ... 15											Positions that become free are provided with zeros	
SRD	-											Shift the contents of ACCU1 to the right	1
SRD	0 ... 32											Positions that become free are provided with zeros	
SSI	-											Shift the contents of ACCU1-L to the right with sign	1
SSI	0 ... 15											Positions that become free are provided with the sign (bit 15)	
SSD	-											Shift the contents of ACCU1 to the right with sign	1
SSD	0 ... 32												
Rotation instructions			*Status word*									Rotate the contents of ACCU1 to the left or right by the specified number of places. If no address identifier is specified, rotate the number of places into ACCU2-LL.	
RLD	-		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Rotate the contents of ACCU1 to the left	1
RLD	0 ... 32		-	-	-	-	-	-	-	-	-		
RRD	-		-	Y	Y	Y	-	-	-	-	-	Rotate the contents of ACCU1 to the right	1
RRD	0 ... 32												
RLDA	-		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Rotate the contents of ACCU1 one bit position to the left,	
			-	-	-	-	-	-	-	-	-	via CC1 bit	
RRDA	-		-	Y	0	0	-	-	-	-	-	Rotate the contents of ACCU1 one bit position to the right,	
												via CC1 bit	

Command	Operand	Parameter	Status word									Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
												: Instruction depends on	
												: Instruction influences	

Setting/resetting bit addresses

Set/Reset bit addresses			*Status word*									Assign the value "1" or "0" or the RLO to the addressed instructions.	
S			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Set ...	
	I/Q a.b	0.0 ... 127.7	-	-	-	-	-	-	-	Y	-	input/output to "1"	1/2
	M a.b	0.0 ... 1023.7	-	-	-	-	-	0	Y	-	0	set bit memory to "1"	1/2
	L a.b	0.0 ... 1043.7										local data bit to "1"	2
	DBX a.b	0.0 ... 8191.7										data bit to "1"	2
	DIX a.b	0.0 ... 8191.7										instance data bit to "1"	2
	c [AR1,m]											register-indirect, area-internal (AR1)	2
	c [AR2,m]											register-indirect, area-internal (AR2)	2
	[AR1,m]											area-crossing (AR1)	2
	[AR2,m]											area-crossing (AR2)	2
	Parameter											via parameters	2
R			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Reset ...	
	I/Q a.b	0.0 ... 127.7	-	-	-	-	-	-	-	Y	-	input/output to "0"	1/2
	M a.b	0.0 ... 1023.7	-	-	-	-	-	0	Y	-	0	set bit memory to "0"	1/2
	L a.b	0.0 ... 1043.7										local data bit to "0"	2
	DBX a.b	0.0 ... 8191.7										data bit to "0"	2
	DIX a.b	0.0 ... 8191.7										instance data bit to "0"	2
	c [AR1,m]											register-indirect, area-internal (AR1)	2
	c [AR2,m]											register-indirect, area-internal (AR2)	2
	W [AR1,m]											area-crossing (AR1)	2
	W [AR2,m]											area-crossing (AR2)	2
	Parameter											via parameters	2
=			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Assign ...	
	I/Q a.b	0.0 ... 127.7	-	-	-	-	-	-	-	Y	-	RLO to input/output	1/2
	M a.b	0.0 ... 1023.7	-	-	-	-	-	0	Y	-	0	RLO to bit memory	1/2
	L a.b	0.0 ... 1043.7										RLO to local data bit	2
	DBX a.b	0.0 ... 8191.7										RLO to data bit	2
	DIX a.b	0.0 ... 8191.7										RLO to instance data bit	2
	c [AR1,m]											register-indirect, area-internal (AR1)	2
	c [AR2,m]											register-indirect, area-internal (AR2)	2
	[AR1,m]											area-crossing (AR1)	2
	[AR2,m]											area-crossing (AR2)	2
	Parameter											via parameters	2

Command	Operand	Parameter	Status word									Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
												: Instruction depends on	
												: Instruction influences	

Instructions directly affecting the RLO			*Status word*									The following instructions have a directly effect on the RLO.	
CLR			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Set RLO to "0"	1
			-	-	-	-	-	-	-	-	-		
			-	-	-	-	-	0	0	0	0		
SET			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Set RLO to "1"	1
			-	-	-	-	-	-	-	-	-		
			-	-	-	-	-	0	1	1	0		
NOT			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Negate RLO	1
			-	-	-	-	-	Y	-	Y	-		
			-	-	-	-	-	-	1	Y	-		
SAVE			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Save RLO into BR-Bit	1
			-	-	-	-	-	-	-	Y	-		
			Y	-	-	-	-	-	-	-	-		

Command	Operand	Parameter	Status word									Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
												: Instruction depends on	
												: Instruction influences	

Jump instructions

Jump instructions			*Status word*									Jump, depending on conditions. 8-bit operands have a jump width of (-128...+127), 16-bit operands of (-32768...-129) or (+128...+32767)	
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
JU	LABEL		-	-	-	-	-	-	-	-	-	Jump unconditionally	1/2
JC	LABEL		-	-	-	-	-	-	-	Y	-	Jump if RLO="1"	1/2
JCN	LABEL		-	-	-	-	-	-	-	Y	-	Jump if RLO="0"	2
JCB	LABEL		-	-	-	-	-	-	-	Y	-	Jump if RLO="1" Save the RLO in the BR-bit	2
JNB	LABEL		Y	-	-	-	-	0	1	1	0	Jump if RLO="0" Save the RLO in the BR-bit	2
JBI	LABEL		-	-	-	-	-	-	-	-	Y	Jump if BR="1"	2
JNBI	LABEL		Y	-	-	-	-	-	-	-	-	Jump if BR="0"	2
JO	LABEL		-	-	-	Y	-	-	-	-	-	Jump on stored overflow (OV="1")	1/2
JOS	LABEL		-	-	-	-	Y	-	-	-	-	Jump on stored overflow (OS="1")	2
JUO	LABEL		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Jump if "unordered instruction" (CC1=1 and CC0=1)	
JZ	LABEL		-	Y	Y	-	-	-	-	-	-	Jump if result=0 (CC1=0 and CC0=0)	1/2
JP	LABEL		-	-	-	-	-	-	-	-	-	Jump if result>0 (CC1=1 and CC0=0)	1/2
JM	LABEL											Jump if result<0 (CC1=0 and CC0=1)	1/2
JN	LABEL											Jump if result≠0 (CC1=1 and CC0=0) or (CC1=0) and (CC0=1)	1/2
JMZ	LABEL											Jump if result≤0 (CC1=0 and CC0=1) or (CC1=0 and CC0=0)	2
JPZ	LABEL											Jump if result≥0 (CC1=1 and CC0=0) or (CC1=0 and CC0=0)	2
JL	LABEL		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Jump distributor	2
			-	-	-	-	-	-	-	-	-	This instruction is followed by a list of jump instructions	
			-	-	-	-	-	-	-	-	-	The operand is a jump label to subsequent instructions in this list. ACCU1-L contains the number of the jump instruction to be executed	
LOOP	LABEL											Decrement ACCU1-L and jump if ACCU1-L _ 0 (loop programming)	2

Command	Operand	Parameter	Status word									Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
												: Instruction depends on	
												: Instruction influences	

Transfer instructions

Transfer instructions				Transfer the contents of ACCU1 into the addressed operand. The status word is not affected.	
T				Transfer the contents of ACCU1-LL to...	
	IB a	0 ... 127		input byte	1/2
	QB a	0 ... 127		output byte	1/2
	PQB a	0 ... 1023		periphery output byte	1/2
	MB a	0 ... 1023		bit memory byte	1/2
	LB a	0 ... 1043		local data byte	2
	DBB a	0 ... 8191		data byte	2
	DIB a	0 ... 8191		instance data byte	2
	g [AR1,m]			register-indirect, area-internal (AR1)	2
	g [AR2,m]			register-indirect, area-internal (AR2)	2
	B [AR1,m]			area-crossing (AR1)	2
	B [AR2,m]			area-crossing (AR2)	2
	Parameter			via parameters	2
T				Transfer the contents of ACCU1-L to ...	
	IW	0 ... 126		input word	1/2
	QW	0 ... 126		output word	1/2
	PQW	0 ... 1022		periphery output word	1/2
	MW	0 ... 1022		bit memory word	1/2
	LW	0 ... 1042		local data word	2
	DBW	0 ... 8190		data word	2
	DIW	0 ... 8190		instance data word	2
	h [AR1,m]			register-indirect, area-internal (AR1)	2
	h [AR2,m]			register-indirect, area-internal (AR2)	2
	W [AR1,m]			area-crossing (AR1)	2
	W [AR2,m]			area-crossing (AR2)	2
	Parameter			via parameters	2

Command	Operand	Parameter	Status word									Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
												: Instruction depends on	
												: Instruction influences	

Transfer instructions				Transfer the contents of ACCU1 into the addressed operand. The status word is not affected.	
T				Transfer the contents of ACCU1 to...	
	ID	0 ... 124		input double word	1/2
	QD	0 ... 124		output double word	1/2
	PQD	0 ... 1020		periphery output double word	1/2
	MD	0 ... 1020		bit memory double word	1/2
	LD	0 ... 1040		local data double word	2
	DBD	0 ... 8188		data double word	2
	DID	0 ... 8188		instance data double word	2
	i [AR1,m]			register-indirect, area-internal (AR1)	2
	i [AR2,m]			register-indirect, area-internal (AR2)	2
	D [AR1,m]			area-crossing (AR1)	2
	D [AR2,m]			area-crossing (AR2)	2
	Parameter			via parameters	2
Load and transfer instructions for address register				Load a double word from a memory area or a register into AR1 or AR2.	
LAR1				Load the contents from...	
	-			ACCU1	1
	AR2			address register 2	1
	DBD a	0 ... 8188		data double word	2
	DID a	0 ... 8188		instance data double word	2
	m			32bit constant as pointer	3
	LD a	0 ... 1040		local data double word	2
	MD a	0 ... 1020		bit memory double word	2
				... into AR1	
LAR2				Load the contents from ...	
	-			ACCU1	1
	DBD a	0 ... 8188		data double word	2
	DID a	0 ... 8188		instance data double word	2
	m			32bit constant as pointer	3
	LD a	0 ... 1040		local data double word	2
	MD a	0 ... 1020		bit memory double word	2
				... into AR2	
TAR1				Transfer the contents from AR1 to...	
	-			ACCU1	1
	AR2			address register 2	1
	DBD a	0 ... 8188		data double word	2
	DID a	0 ... 8188		instance data double word	2
	LD a	0 ... 1040		local data double word	2
	MD a	0 ... 1020		bit memory double word	2

Command	Operand	Parameter	Status word BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Function	Length in words
												: Instruction depends on	
												: Instruction influences	

TAR2												Transfer the contents from AR2 to...	
	-											ACCU1	1
	DBD a	0 ... 8188										data double word	2
	DID a	0 ... 8188										instance data double word	2
	LD a	0 ... 1040										local data double word	2
	MD a	0 ... 1020										bit memory double word	2
TAR												Exchange the contents of AR1 and AR2	1
Load and transfer instructions for the status word			*Status word*										
L	STW		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Load status word in ACCU1	
	-		Y	Y	Y	Y	Y	0	0	Y	0		
			-	-	-	-	-	-	-	-	-		
T	STW		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Transfer ACCU1 (Bits 8 ... 0) into status word	
	-		-	-	-	-	-	-	-	-	-		
			Y	Y	Y	Y	Y	-	-	Y	-		
Load instructions for DB number and DB length												Load the number/length of a data block to ACCU1. The old contents of ACCU1 are saved into ACCU2. The condition code bits are not affected	
L	DBNO											Load number of data block	1
L	DINO											Load number of instance data block	1
L	DBLG											Load length of data block into byte	1
L	DILG											Load length of instance data block into byte	1
ACCU transfer instructions, increment, decrement												The status word is not affected.	
CAW	-											Reverse the order of the bytes in ACCU1-L	1
												LL, LH becomes LH, LL	
CAD	-											Reverse the order of the bytes in ACCU1	1
												LL, LH, HL, HH becomes HH, HL, LH, LL	
TAK	-											Swap the contents of ACCU1 and ACCU2	1
PUSH	-											The contents of ACCU1 are transferred to ACCU2	1
POP	-											The contents of ACCU2 are transferred to ACCU1	1
INC	0 ... 255											Increment ACCU1-LL	1
DEC	0 ... 255											Decrement ACCU1-LL	1

Command	Operand	Parameter	Status word									Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
												: Instruction depends on	
												: Instruction influences	

Data type conversion instructions

Data type conversion instructions			*Status word*									The results of the conversion are in ACCU1. When converting real numbers, the execution time depends on the value.	
BTI	-		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Convert contents of ACCU1 from BCD to integer (16bit)	1
			-	-	-	-	-	-	-	-	-	(**B**CD **T**o **I**nt.)	
BTD	-		-	-	-	-	-	-	-	-	-	Convert contents of ACCU1 from BCD to integer (32bit).	1
												(**B**CD **T**o **D**oubleint.)	
DTR	-											Convert cont. of ACCU1 from integer (32bit) to Real number	1
												(32bit) (**D**oubleint. **T**o **R**eal)	
ITD	-											Convert contents of ACCU1 from integer (16bit) to integer	1
												(32bit) (**I**nt. **T**o **D**oubleint)	
ITB	-		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Convert contents of ACCU1 from integer (16bit) to BCD	1
			-	-	-	-	-	-	-	-	-	0 ... +/-999 (**I**nt. **T**o **B**CD)	
DTB	-		-	-	-	Y	Y	-	-	-	-	Convert contents of ACCU1 from integer (32bit) to BCD	1
												0 ... +/-9 999 999 (**D**oubleint. **T**o **B**CD)	
RND	-		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Convert a real number to 32bit integer	1
RND-	-		-	-	-	-	-	-	-	-	-	Convert a real number to 32bit integer	1
			-	-	-	Y	Y	-	-	-	-	The number is rounded next hole number	
RND+	-											Convert real number to 32bit integer	1
												It is rounded up to the next integer	
TRUNC	-											Convert real number to 32bit integer	1
												The places after the decimal point are truncated	
Complement creation			*Status word*										
INVI	-		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Forms the ones complement of ACCU1-L	1
INVD	-		-	-	-	-	-	-	-	-	-	Forms the ones complement of ACCU1	1
			-	-	-	-	-	-	-	-	-		
NEGI	-		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Forms the twos complement of ACCU1-L (integer)	1
NEGD	-		-	-	-	-	-	-	-	-	-	Forms the twos complement of ACCU1 (double integer)	1
			-	Y	Y	Y	Y	-	-	-	-		

Command	Operand	Parameter	Status word									Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
												: Instruction depends on	
												: Instruction influences	

Comparison instructions

Comparison instructions with integer (16bit)			*Status word*									Comparing the integer (16bit) in ACCU1-L and ACCU2-L. RLO=1, if condition is satisfied.	
==I	-		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	ACCU2-L=ACCU1-L	1
<>I	-		-	-	-	-	-	-	-	-	-	ACCU2-L≠ACCU1-L	1
<I	-		-	Y	Y	0	-	0	Y	Y	1	ACCU2-L<ACCU1-L	1
<=I	-											ACCU2-L<=ACCU1-L	1
>I	-											ACCU2-L>ACCU1-L	1
>=I	-											ACCU2-L>=ACCU1-L	1
Comparison instructions with integer (32bit)			*Status word*									Comparing the integer (32bit) in ACCU1 and ACCU2. RLO=1, if condition is satisfied.	
==D	-		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	ACCU2=ACCU1	1
<>D	-		-	-	-	-	-	-	-	-	-	ACCU2≠ACCU1	1
<D	-		-	Y	Y	0	-	0	Y	Y	1	ACCU2<ACCU1	1
<=D	-											ACCU2<=ACCU1	1
>D	-											ACCU2>ACCU1	1
>=D	-											ACCU2>=ACCU1	1
Comparison instructions with 32bit real number			*Status word*									Comparing the 32bit real numbers in ACCU1 and ACCU2. RLO=1, is condition is satisfied. The execution time of the instruction depends on the value to be compared.	
==R	-		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	ACCU2=ACCU1	1
<>R	-		-	-	-	-	-	-	-	-	-	ACCU2≠ACCU1	1
<R	-		-	Y	Y	Y	Y	0	Y	Y	1	ACCU2<ACCU1	1
<=R	-											ACCU2<=ACCU1	1
>R	-											ACCU2>ACCU1	1
>=R	-											ACCU2>=ACCU1	1

Command	Operand	Parameter	Status word									Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
												: Instruction depends on	
												: Instruction influences	

Combination instructions (Bit)

Combination instructions with bit operands			*Status word*									Examining the signal state of the addressed instruction and gating the result with the RLO according to the appropriate logic function.	
A			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	AND operation at signal state "1"	
	I/Q a.b	0.0 ... 127.7	-	-	-	-	-	Y	-	Y	Y	Input/output	1/2
	M a.b	0.0 ... 1023.7	-	-	-	-	-	Y	Y	Y	1	Bit memory	1/2
	L a.b	0.0 ... 1043.7										Local data bit	2
	DBX a.b	0.0 ... 8191.7										Data bit	2
	DIX a.b	0.0 ... 8191.7										Instance data bit	2
	c [AR1,m]											register-indirect, area-internal (AR1)	2
	c [AR2,m]											register-indirect, area-internal (AR2)	2
	[AR1,m]											area-crossing (AR1)	2
	[AR2,m]											area-crossing (AR2)	2
	Parameter											via parameters	2
AN			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	AND operation of signal state "0"	
	I/Q a.b	0.0 ... 127.7	-	-	-	-	-	Y	-	Y	Y	Input/output	1/2
	M a.b	0.0 ... 1023.7	-	-	-	-	-	Y	Y	Y	1	Bit memory	1/2
	L a.b	0.0 ... 1043.7										Local data bit	2
	DBX a.b	0.0 ... 8191.7										Data bit	2
	DIX a.b	0.0 ... 8191.7										Instance data bit	2
	c [AR1,m]											register-indirect, area-internal (AR1)	2
	c [AR2,m]											register-indirect, area-internal (AR2)	2
	[AR1,m]											area-crossing (AR1)	2
	[AR2,m]											area-crossing (AR2)	2
	Parameter											via parameters	2
O			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	OR operation at signal state "1"	
	I/Q a.b	0.0 ... 127.7	-	-	-	-	-	-	-	Y	Y	Input/output	1/2
	M a.b	0.0 ... 1023.7	-	-	-	-	-	0	Y	Y	1	Bit memory	1/2
	L a.b	0.0 ... 1043.7										Local data bit	2
	DBX a.b	0.0 ... 8191.7										Data bit	2
	DIX a.b	0.0 ... 8191.7										Instance data bit	2
	c [AR1,m]											register-indirect, area-internal (AR1)	2
	c [AR2,m]											register-indirect, area-internal (AR2)	2
	[AR1,m]											area-crossing (AR1)	2
	[AR2,m]											area-crossing (AR2)	2
	Parameter											via parameters	2

Command	Operand	Parameter	Status word BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Function	Length in words
												: Instruction depends on	
												: Instruction influences	

			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
ON												OR operation at signal state "0"	
	I/Q a.b	0.0 ... 127.7	-	-	-	-	-	-	-	Y	Y	Input/output	1/2
	M a.b	0.0 ... 1023.7	-	-	-	-	-	0	Y	Y	1	Bit memory	1/2
	L a.b	0.0 ... 1043.7										Local data bit	2
	DBX a.b	0.0 ... 8191.7										Data bit	2
	DIX a.b	0.0 ... 8191.7										Instance data bit	2
	c [AR1,m]											register-indirect, area-internal (AR1)	2
	c [AR2,m]											register-indirect, area-internal (AR2)	2
	[AR1,m]											area-crossing (AR1)	2
	[AR2,m]											area-crossing (AR2)	2
	Parameter											via parameters	2
X												EXCLUSIVE-OR operation at signal state "1"	
	I/Q a.b	0.0 ... 127.7	-	-	-	-	-	-	-	Y	Y	Input/output	2
	M a.b	0.0 ... 1023.7	-	-	-	-	-	0	Y	Y	1	Bit memory	2
	L a.b	0.0 ... 1043.7										Local data bit	2
	DBX a.b	0.0 ... 8191.7										data bit	2
	DIX a.b	0.0 ... 8191.7										Instance data bit	2
	c [AR1,m]											register-indirect, area-internal (AR1)	2
	c [AR2,m]											register-indirect, area-internal (AR2)	2
	[AR1,m]											area-crossing (AR1)	2
	[AR2,m]											area-crossing (AR2)	2
	Parameter											via parameters	2
XN												EXCLUSIVE-OR operation at signal state "0"	
	I/Q a.b	0.0 ... 127.7	-	-	-	-	-	-	-	Y	Y	Input/output	2
	M a.b	0.0 ... 1023.7	-	-	-	-	-	0	Y	Y	1	Bit memory	2
	L a.b	0.0 ... 1043.7										Local data bit	2
	DBX a.b	0.0 ... 8191.7										Data bit	2
	DIX a.b	0.0 ... 8191.7										Instance data bit	2
	c [AR1,m]											register-indirect, area-internal (AR1)	2
	c [AR2,m]											register-indirect, area-internal (AR2)	2
	[AR1,m]											area-crossing (AR1)	2
	[AR2,m]											area-crossing (AR2)	2
	Parameter											via parameters	2

Command	Operand	Parameter	Status word									Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
												: Instruction depends on	
												: Instruction influences	

Combination instructions with parenthetical expressions			*Status word*									Saving the bits BR, RLO, OR and a function ID (A, AN, ...) at the nesting stack. For each block 7 nesting levels are possible.	
A(			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	AND left parenthesis	1
AN(			Y	-	-	-	-	Y	-	Y	Y	AND-NOT left parenthesis	1
O(			-	-	-	-	-	0	1	-	0	OR left parenthesis	1
ON(												OR-NOT left parenthesis	1
X(												EXCLUSIVE-OR left parenthesis	1
XN(												EXCLUSIVE-OR-NOT left parenthesis	1
)			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Right parenthesis, popping an entry off the nesting stack,	1
			-	-	-	-	-	-	-	Y	-	gating RLO with the current RLO in the processor.	
			Y	-	-	-	-	Y	1	Y	1		
ORing of AND operations			*Status word*									The ORing of AND operations is implemented according the rule: AND before OR.	
O			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	OR operations of AND functions according the rule:	1
			-	-	-	-	-	Y	-	Y	Y	AND before OR	
			-	-	-	-	-	Y	1	-	Y		

Command	Operand	Parameter	Status word									Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
												: Instruction depends on	
												: Instruction influences	

Combinations instructions with timer and counters			*Status word*									Examining the signal state of the addressed timer/counter an gating the result with the RLO according to the appropriate logic function.	
A			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	AND operation at signal state	
	T f	0 to 255	-	-	-	-	-	Y	-	Y	Y	Timer	1/2
	C f	0 to 255	-	-	-	-	-	Y	Y	Y	1	Counter	1/2
	Timer para.											Timer addressed via parameters	2
	Counter para.											Counter addressed via parameters	
AN			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	AND operation at signal state	
	T f	0 to 255	-	-	-	-	-	Y	-	Y	Y	Timer	1/2
	C f	0 to 255	-	-	-	-	-	Y	Y	Y	1	Counter	1/2
	Timer para.											Timer addressed via parameters	2
	Counter para.											Counter addressed via parameters	
O			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	OR operation at signal state	
	T f	0 to 255	-	-	-	-	-	-	-	Y	Y	Timer	1/2
	C f	0 to 255	-	-	-	-	-	0	Y	Y	1	Counter	1/2
	Timer para.											Timer addressed via parameters	2
	Counter para.											Counter addressed via parameters	
ON			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	OR operation at signal state	
	T f	0 to 255	-	-	-	-	-	-	-	Y	Y	Timer	1/2
	C f	0 to 255	-	-	-	-	-	0	Y	Y	1	Counter	1/2
	Timer para.											Timer addressed via parameters	2
	Counter para.											Counter addressed via parameters	
X			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	EXCLUSIVE-OR operation at signal state	
	T f	0 to 255	-	-	-	-	-	-	-	Y	Y	Timer	2
	C f	0 to 255	-	-	-	-	-	0	Y	Y	1	Counter	2
	Timer para.											Timer addressed via parameters	2
	Counter para.											Counter addressed via parameters	
XN			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	EXCLUSIVE-OR operation at signal state	
	T f	0 to 255	-	-	-	-	-	-	-	Y	Y	Timer	2
	C f	0 to 255	-	-	-	-	-	0	Y	Y	1	Counter	2
	Timer para.											Timer addressed via parameters	2
	Counter para.											Counter addressed via parameters	

Command	Operand	Parameter	Status word									Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
												: Instruction depends on	
												: Instruction influences	

Combination instructions using AND, OR and EXCLUSIVE OR			*Status word*									Examining the specified conditions for their signal status, and gating the result with the RLO according to the appropriate function.	
A			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	AND operation at signal state "1"	
	==0		Y	Y	Y	Y	Y	Y	-	Y	Y	Result=0 (CC1=0) and (CC0=0)	1
	>0		-	-	-	-	-	Y	Y	Y	1	Result>0 (CC1=1) and (CC0=0)	1
	<0											Result<0 (CC1=0) and (CC0=1)	1
	<>0											Result≠0 ((CC1=0) and (CC0=1)) or ((CC1=1) and (CC0=0))	1
	<=0											Result<=0 ((CC1=0) and (CC0=1)) or ((CC1=0) and (CC0=0))	1
	>=0											Result>=0 ((CC1=1) and (CC0=0)) or ((CC1=1) and (CC0=0))	1
	UO											unordered (CC1=1) and (CC0=1)	1
	OS											OS=1	1
	BR											BR=1	1
	OV											OV=1	1
AN			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	AND operation at signal state "0"	
	==0		Y	Y	Y	Y	Y	Y	-	Y	Y	Result=0 (CC1=0) and (CC0=0)	1
	>0		-	-	-	-	-	Y	Y	Y	1	Result>0 (CC1=1) and (CC0=0)	1
	<0											Result<0 (CC1=0) and (CC0=1)	1
	<>0											Result≠0 ((CC1=0) and (CC0=1)) or ((CC1=1) and (CC0=0))	1
	<=0											Result<=0 ((CC1=0) and (CC0=1)) or ((CC1=0) and (CC0=0))	1
	>=0											Result>=0 ((CC1=1) and (CC0=0)) or ((CC1=1) and (CC0=0))	1
	UO											unordered (CC1=1) and (CC0=1)	1
	OS											OS=0	1
	BR											BR=0	1
	OV											OV=0	1
O			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	OR operation at signal state "1"	
	==0		Y	Y	Y	Y	Y	-	-	Y	Y	Result=0 (CC1=0) and (CC0=0)	1
	>0		-	-	-	-	-	0	Y	Y	1	Result>0 (CC1=1) and (CC0=0)	1
	<0											Result<0 (CC1=0) and (CC0=1)	1
	<>0											Result≠0 ((CC1=0) and (CC0=1)) or ((CC1=1) and (CC0=0))	1
	<=0											Result<=0 ((CC1=0) and (CC0=1)) or ((CC1=0) and (CC0=0))	1
	>=0											Result>=0 ((CC1=1) and (CC0=0)) or ((CC1=1) and (CC0=0))	1
	UO											unordered (CC1=1) and (CC0=1)	1
	OS											OS=1	1
	BR											BR=1	1
	OV											OV=1	1

Command	Operand	Parameter	Status word									Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
												: Instruction depends on	
												: Instruction influences	

ON			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	OR operation at signal state "0"	
	==0		Y	Y	Y	Y	Y	-	-	Y	Y	Result=0 (CC1=0) and (CC0=0)	1
	>0		-	-	-	-	-	0	Y	Y	1	Result>0 (CC1=1) and (CC0=0)	1
	<0											Result<0 (CC1=0) and (CC0=1)	1
	<>0											Result≠0 ((CC1=0) and (CC0=1)) or ((CC1=1) and (CC0=0))	1
	<=0											Result<=0 ((CC1=0) and (CC0=1)) or ((CC1=0) and (CC0=0))	1
	>=0											Result>=0 ((CC1=1) and (CC0=0)) or ((CC1=1) and (CC0=0))	1
	UO											unordered (CC1=1) and (CC0=1)	1
	OS											OS=0	1
	BR											BR=0	1
	OV											OV=0	1
X			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	EXCLUSIVE-OR operation at signal state "1"	
	==0		Y	Y	Y	Y	Y	-	-	Y	Y	Result=0 (CC1=0) and (CC0=0)	1
	>0		-	-	-	-	-	0	Y	Y	1	Result>0 (CC1=1) and (CC0=0)	1
	<0											Result<0 (CC1=0) and (CC0=1)	1
	<>0											Result≠0 ((CC1=0) and (CC0=1)) or ((CC1=1) and (CC0=0))	1
	<=0											Result<=0 ((CC1=0) and (CC0=1)) or ((CC1=0) and (CC0=0))	1
	>=0											Result>=0 ((CC1=1) and (CC0=0)) or ((CC1=1) and (CC0=0))	1
	UO											unordered (CC1=1) and (CC0=1)	1
	OS											OS=1	1
	BR											BR=1	1
	OV											OV=1	1
XN			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	EXCLUSIVE-OR operation at signal state "0"	
	==0		Y	Y	Y	Y	Y	-	-	Y	Y	Result=0 (CC1=0) and (CC0=0)	1
	>0		-	-	-	-	-	0	Y	Y	1	Result>0 (CC1=1) and (CC0=0)	1
	<0											Result<0 (CC1=0) and (CC0=1)	1
	<>0											Result≠0 ((CC1=0) and (CC0=1)) or ((CC1=1) and (CC0=0))	1
	<=0											Result<=0 ((CC1=0) and (CC0=1)) or ((CC1=0) and (CC0=0))	1
	>=0											Result>=0 ((CC1=1) and (CC0=0)) or ((CC1=1) and (CC0=0))	1
	UO											unordered (CC1=1) and (CC0=1)	1
	OS											OS=0	1
	BR											BR=0	1
	OV											OV=0	1

Command	Operand	Parameter	Status word									Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
												: Instruction depends on	
												: Instruction influences	

Combination instructions (Word)

Combination instructions with the contents of ACCU1			*Status word*									Gating the contents of ACCU1 and/or ACCU1-L with a word or double word according to the appropriate function. The word or double word is either a constant in the instruction or in ACCU2. The result is in ACCU1 and/or ACCU1-L.	
AW			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	AND ACCU2-L	1
AW	k16		-	-	-	-	-	-	-	-	-	AND 16bit constant	2
OW			-	Y	0	0	-	-	-	-	-	OR ACCU2-L	1
OW	k16											OR 16bit constant	2
XOW												EXCLUSIVE OR ACCU2-L	1
XOW	k16											EXCLUSIVE OR 16bit constant	2
AD												AND ACCU2	1
AD	k32											AND 32bit constant	3
OD												OR ACCU2	1
OD	k32											OR 32bit constant	3
XOD												EXCLUSIVE OR ACCU2	1
XOD	k32											EXCLUSIVE OR 32bit constant	3

Timer instructions

Time instructions			*Status word*									Starting or resetting a timer (addressed directly or via parameters). The time value must be in ACCU1-L.	
SP	T f	0 ... 255	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Start time as pulse on edge change from "0" to "1"	1/2
	Timer para.		-	-	-	-	-	-	-	Y	-		2
SE	T f	0 ... 255	-	-	-	-	-	0	-	-	0	Start timer as extended pulse on edge change from	1/2
	Timer para.											"0" to "1"	2
SD	T f	0 ... 255										Start timer as ON delay on edge change from "0" to "1"	1/2
	Timer para.												2
SS	T f	0 ... 255										Start timer as saving start delay on edge change from	1/2
	Timer para.											"0" to "1"	2
SA	T f	0 ... 255										Start timer as OFF delay on edge change from "1" to "0"	1/2
	Timer para.												2
FR	T f	0 ... 255										Enable timer for restarting on edge change from "0" to "1"	1/2
	Timer para.											(reset edge bit memory for starting timer)	2
R	T f	0 ... 255										Reset timer	1/2
	Timer para.												2

Command	Operand	Parameter	Status word									Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
												: Instruction depends on	
												: Instruction influences	

Counter instructions

Counter instructions			*Status word*									The counter value is in ACCU1-L res. in the address transferred as parameter.	
S	C f Counter para.	0 ... 255	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Presetting of counter on edge change from "0" to "1"	1/2 2
			-	-	-	-	-	-	-	Y	-		
R	C f Counter para.	0 ... 255	-	-	-	-	-	0	-	-	0	Reset counter to "0"	1/2 2
CU	C f Counter para.	0 ... 255										Increment counter by 1 on edge change from "0" to "1"	1/2 2
CD	C f Counter para.	0 ... 255										Decrement counter by 1 on edge change from "0" to "1"	1/2 2
FR	C f Counter para.	0 ... 255										Enable counter on edge change from "0" to "1" (reset the edge bit memory for up and down counting)	1/2 2

Chapter 2 Organization Blocks

Overview

Here the description of the integrated organization blocks of the VIPA standard CPUs of Systems 100V, 200V, 300V and 500V may be found.

Content

Overview

General

OBs (**O**rganization **b**locks) are the interface between the operating system of the CPU and the user program. For the main program OB 1 is used. There are reserved numbers corresponding to the call event of the other OBs. Organization blocks are executed corresponding to their priority.

OBs are used to execute specific program sections:

- at the startup of the CPU
- in a cyclic or clocked execution
- whenever errors occur
- whenever hardware interrupts occur

Integrated OBs

The following organization blocks (OBs) are available:

OB	Description
OB 1	Free cycle
OB 10	Time-of-day interrupt
OB 20	Time-delay interrupt
OB 35	Watchdog interrupt
OB 40	Hardware interrupt
OB 80	Time error (cycle time exceeded or clock alarm run out)
OB 81	Power supply fault
OB 82	Diagnostics interrupt
OB 85	Program execution error (OB not available or Periphery error at update process image)
OB 86	Slave failure / restart
OB 100	Restart
OB 121	Programming error (synchronous error)
OB 122	Periphery access error

OB 1 - Main program

Description

The operating system of the CPU executes OB 1 cyclically. After STARTUP to RUN the cyclical processing of the OB 1 is started. OB 1 has the lowest priority (priority 1) of each cycle time monitored OB. Within the OB 1 functions and function blocks can be called.

Function

When OB 1 has been executed, the operating system sends global data. Before restarting OB 1, the operating system writes the process-image output table to the output modules, updates the process-image input table and receives any global data for the CPU.

Cycle time

Cycle time is the time required for processing the OB 1. It also includes the scan time for higher priority classes which interrupt the main program respectively communication processes of the operating system. This comprises system control of the cyclic program scanning, process image update and refresh of the time functions.

By means of the Siemens SIMATIC manager the recent cycle time of an online connected CPU may be shown.
With **PLC** > *Module Information* > *Scan cycle time* the min., max. and recent cycle time can be displayed.

Scan cycle monitoring time

The CPU offers a scan cycle watchdog for the *max. cycle time*. The default value for the *max. cycle time* is 150ms as *scan cycle monitoring time*. This value can be reconfigured or restarted by means of the SFC 43 (RE_TRIGR) at every position of your program. If the main program takes longer to scan than the specified *scan cycle monitoring time*, the OB 80 (Timeout) is called by the CPU. If OB 80 has not been programmed, the CPU goes to STOP.

Besides the monitoring of the *max. cycle time* the observance of the *min cycle time* can be guaranteed. Here the restart of a new cycle (writing of process image of the outputs) is delayed by the CPU as long as the *min. cycle time* is reached.

Access to local data

The CPU's operating system forwards start information to OB 1, as it does to every OB, in the first 20 bytes of temporary local data.

The start information can be accessed by means of the system function SFC 6 RD_SINFO. Note that direct reading of the start information for an OB is possible only in that OB because that information consists of temporary local data.
More information can be found at chapter "Integrated standard SFCs".

Local data

The following table describes the start information of the OB 1 with default names of the variables and its data types:

Variable	Type	Description
OB1_EV_CLASS	BYTE	Event class and identifiers: 11h: OB 1 active
OB1_SCAN_1	BYTE	01h: completion of a restart 03h: completion of the main cycle
OB1_PRIORITY	BYTE	Priority class: 1
OB1_OB_NUMBR	BYTE	OB number (01)
OB1_RESERVED_1	BYTE	reserved
OB1_RESERVED_2	BYTE	reserved
OB1_PREV_CYCLE	INT	Run time of previous cycle (ms)
OB1_MIN_CYCLE	INT	Minimum cycle time (ms) since the last startup
OB1_MAX_CYCLE	INT	Maximum cycle time (ms) since the last startup
OB1_DATE_TIME	DATE_AND_TIME	Date and time of day when the OB was called

OB 10 - Time-of-day Interrupt

Description

The time-of-day interrupt is used when you want to run a program at a particular time, either once only or periodically. The time-of-day interrupt can be configured within the hardware configuration or controlled by means of system functions in your main program at run time.

The prerequisite for proper handling of the time-of-day interrupt is a correctly set real-time clock on the CPU.

For execution there are the following intervals:

- once
- every minute
- hourly
- daily
- weekly
- monthly
- once at year
- at the end of each month

Note!

For monthly execution of the time-of-day interrupt OB, only the day 1, 2, ...28 can be used as a starting date.

Function

To start the time-of-day interrupt, you must first set and than activate the interrupt. The three following start possibilities exist:

- The time-of-day interrupt is configured via the hardware configuration. Open the selected CPU with **Edit** > *Object properties* > *Time-of-Day* interrupts. Here the time-of-day interrupt may be adjusted and activated. After transmission to CPU and startup the monitoring of time-of-day interrupt is automatically started.
- Set the time-of-day interrupt within the hardware configuration as shown above and then activate it by calling SFC 30 ACT_TINT in your program.
- You set the time-of-day interrupt by calling SFC 28 SET_TINT and then activate it by calling SFC 30 ACT_TINT.

The time-of-day interrupt can be delayed and enabled with the system functions SFC 41 DIS_AIRT and SFC 42 EN_AIRT.

Behavior on error

If the time-of-day interrupt OB is called but was not programmed, the operating system calls OB 85. If OB 85 was not programmed, the CPU goes to STOP. Is there an error at time-of-day interrupt processing e.g. start time has already passed, the time error OB 80 is called. The time-of-day interrupt OB is then executed precisely once.

Possibilities of activation

The possibilities of activation of the time-of-day interrupt is shown at the following table:

Interval	Description
Not activated	The time-of-day interrupt is not executed, even when loaded in the CPU. It may be activated by calling SFC 30.
Activated once only	The time-of-day OB is cancelled automatically after it runs the one time specified. Your program can use SFC 28 and SFC 30 to reset and reactivate the OB.
Activated periodically	When the time-of-day interrupt occurs, the CPU calculates the next start time for the time-of-day interrupt based on the current time of day and the period.

Local data for time-of-day interrupt OB

The following table describes the start information of the OB 10 with default names of the variables and its data types:

Variable	Type	Description
OB10_EV_CLASS	BYTE	Event class and identifiers: 11h: interrupt is active
OB10_STRT_INFO	BYTE	11h: Start request for OB 10
OB10_PRIORITY	BYTE	Assigned priority class: default 2
OB10_OB_NUMBR	BYTE	OB number (10)
OB10_RESERVED_1	BYTE	reserved
OB10_RESERVED_2	BYTE	reserved
OB10_PERIOD_EXE	WORD	The OB is executed at the specified intervals: 0000h: once 0201h: once every minute 0401h: once hourly 1001h: once daily 1201h: once weekly 1401h: once monthly 1801h: once yearly 2001h: end of month
OB10_RESERVED_3	INT	reserved
OB10_RESERVED_4	INT	reserved
OB10_DATE_TIME	DATE_AND_TIME	Date and time of day when the OB was called

Information to access the local data can be found at the description of the OB 1.

OB 20 - Time-delay Interrupt

Description

The time-delay interrupt allows you to implement a delay timer independently of the standard timers. The time-delay interrupts can be configured within the hardware configuration respectively controlled by means of system functions in your main program at run time.

Activation

For the activation no hardware configuration is necessary. The time-delay interrupt is started by calling SFC 32 SRT_DINT and by transferring the corresponding OB to the CPU. Here the function needs OB no., delay time and a sign. When the delay interval has expired, the respective OB is called by the operating system. The time-delay interrupt that is just not activated can be cancelled with SFC 33 CAN_DINT respectively by means of the SFC 34 QRY_DINT the status can be queried.

More information for using the SFCs can be found at chapter "Integrated standard SFCs".

The priority of the corresponding OBs are changed via the hardware configuration. For this open the selected CPU with **Edit** > *Object properties* > *Interrupts*. Here the corresponding priority can be adjusted.

Behavior on error

If the time-delay interrupt OB is called but was not programmed, the operating system calls OB 85. If OB 85 was not programmed, the CPU goes to STOP. Is there an error at time-delay interrupt processing e.g. delay interval has expired and the associated OB is still executing, the time error OB 80 is called. The time-of-day interrupt OB is then executed. If there is no OB 80 in the user program the CPU goes to STOP

Local data

The following table describes the start information of the OB 20 and 21 with default names of the variables and its data types:

Variable	Type	Description
OB20_EV_CLASS	BYTE	Event class and identifiers: 11h: interrupt is active
OB20_STRT_INF	BYTE	21h: start request for OB 20
OB20_PRIORITY	BYTE	assigned priority class: Default 3 (OB 20)
OB20_OB_NUMBR	BYTE	OB number (20)
OB20_RESERVED_1	BYTE	reserved
OB20_RESERVED_2	BYTE	reserved
OB20_SIGN	WORD	User ID: input parameter SIGN from the call for SFC 32 (SRT_DINT)
OB20_DTIME	TIME	Configured delay time in ms
OB20_DATE_TIME	DATE_AND_TIME	Date and time of day when the OB was called

Information to access the local data can be found at the description of the OB 1.

OB 35 - Watchdog Interrupt

Description

By means of the watchdog interrupt the cyclical processing can be interrupted in equidistant time intervals. The start time of the time interval and the phase offset is the instant of transition from STARTUP to RUN after execution of OB 100.

Watchdog interrupt OB	Default time interval	Default priority class
OB 35	100ms	12

Activation

The watchdog interrupt is activated by programming the OB 35 within the CPU.

The watchdog interrupt can be delayed and enabled with the system functions SFC 41 DIS_AIRT and SFC 42 EN_AIRT.

Function

After STARTUP to RUN the activated watchdog OB is called in the configured equidistant intervals.

So a sub program can be called time controlled by programming the OB 35.

Local data

The following table describes the start information of the OB 35 with default names of the variables and its data types:

Variable	Type	Description
OB35_EV_CLASS	BYTE	Event class and identifiers: 11h: Cyclic interrupt is active
OB35_STRT_INF	BYTE	36h: Start request for OB 35
OB35_PRIORITY	BYTE	Assigned priority class; Defaults: 12 (OB 35)
OB35_OB_NUMBR	BYTE	OB number (35)
OB35_RESERVED_1	BYTE	reserved
OB35_RESERVED_2	BYTE	reserved
OB35_PHASE_OFFSET	WORD	Phase offset in ms
OB35_RESERVED_3	INT	reserved
OB35_EXT_FREQ	INT	Interval in ms
OB35_DATE_TIME	DATE_AND_TIME	Date and time of day when the OB was called

Information to access the local data can be found at the description of the OB 1.

OB 40 - Hardware Interrupt

Description

Within the configuration you specify for each module, which channels release a hardware interrupt during which conditions.

With the system functions SFC 55 WR_PARM, SFC 56 WR_DPARM and SFC 57 PARM_MOD you can (re)parameterize the modules with hardware interrupt capability even in RUN.

Activation

The hardware interrupt processing of the CPU is always active. So that a module can release a hardware interrupt, you have to activate the hardware interrupt on the appropriate module by a hardware configuration.

Here you can specify whether the hardware interrupt should be generated for a coming event, a leaving event or both.

Function

After a hardware interrupt has been triggered by the module, the operating system identifies the slot and the corresponding hardware interrupt OB. If this OB has a higher priority than the currently active priority class, it will be started. The channel-specific acknowledgement is sent after this hardware interrupt OB has been executed.

If another event that triggers a hardware interrupt occurs on the same module during the time between identification and acknowledgement of a hardware interrupt, the following applies:

- If the event occurs on the channel that previously triggered the hardware interrupt, then the new interrupt is lost.
- If the event occurs on another channel of the same module, then no hardware interrupt can currently be triggered. This interrupt, however, is not lost, but is triggered if just active after the acknowledgement of the currently active hardware interrupt. Else it is lost.
- If a hardware interrupt is triggered and its OB is currently active due to a hardware interrupt from another module, the new request can be processed only if it is still active after acknowledgement.

During STARTUP there is no hardware interrupt produced. The treatment of interrupts starts with the transition to operating mode RUN. Hardware interrupts during transition to RUN are lost.

Behavior on error

If a hardware interrupt is generated for which there is no hardware interrupt OB in the user program, OB 85 is called by the operating system. The hardware interrupt is acknowledged. If OB 85 has not been programmed, the CPU goes to STOP.

Diagnostic interrupt

While the treatment of a hardware interrupt a diagnostic interrupt can be released. Is there, during the time of releasing the hardware interrupt up to its acknowledgement, on the same channel a further hardware interrupt, the loss of the hardware interrupt is announced by means of a diagnostic interrupt for system diagnostics.

Local data

The following table describes the start information of the OB 40 with default names of the variables and its data types:

Variable	Type	Description
OB40_EV_CLASS	BYTE	Event class and identifiers: 11h: Interrupt is active
OB40_STRT_INF	BYTE	At CPU 112: The OB start information depends on the line, which the interrupt has triggered. 41h: E0.0 42h: E0.1 43h: E0.2 44h: E0.3 Other CPUs 11x and CC03: 40h: Interrupt inputs E0.0 ... E0.3 41h: Interrupt sources at V-bus (ext. plugged modules) At System 200V and 300V: 41h: Interrupt via interrupt line 1
OB40_PRIORITY	BYTE	Assigned priority class: Default: 16 (OB 40)
OB40_OB_NUMBR	BYTE	OB number (40)
OB40_RESERVED_1	BYTE	reserved
OB40_IO_FLAG	BYTE	54h: Input module 55h: Output module
OB40_MDL_ADDR	WORD	Logical base address of the module that triggers the interrupt
OB40_POINT_ADDR	DWORD	For digital modules: bit field with the statuses of the inputs on the module (Bit 0 corresponds to the first input). For analog modules: bit field, informing which channel has exceeded which limit. For CPs or IMs: Module interrupt status (not user relevant).
OB40_DATE_TIME	DATE_AND_TIME	Date and time of day when the OB was called.

Information to access the local data can be found at the description of the OB 1.

OB 80 - Time Error

Description

The operating system of the CPU calls OB 80 whenever an error occurs like:

- Cycle monitoring time exceeded
- OB request error i.e. the requested OB is still executed or an OB was requested too frequently within a given priority class.
- Time-of-day interrupt error i.e. interrupt time past because clock was set forward or after transition to RUN.

The time error OB can be delayed by SFC 41 and released by SFC 42.

Note!

If OB 80 has not been programmed, the CPU changes to the STOP mode.

If OB 80 is called twice during the same scan cycle due to the scan time being exceeded, the CPU changes to the STOP mode. You can prevent this by calling SFC 43 RE_TRIGR at a suitable point in the program.

Local data

The following table describes the start information of the OB 80 with default names of the variables and its data types:

Variable	Type	Description
OB80_EV_CLASS	BYTE	Event class and identifiers: 35h
OB80_FLT_ID	BYTE	Error code (possible values: 01h, 02h, 05h, 06h, 07h, 08h, 09h, 0Ah)
OB80_PRIORITY	BYTE	Priority class: 26 (RUN mode) 28 (Overflow of the OB request buffer)
OB80_OB_NUMBR	BYTE	OB number (80)
OB80_RESERVED_1	BYTE	reserved
OB80_RESERVED_2	BYTE	reserved
OB80_ERROR_INFO	WORD	Error information: depending on error code
OB80_ERR_EV_CLASS	BYTE	Event class for the start event that caused the error
OB80_ERR_EV_NUM	BYTE	Event number for the start event that caused the error
OB80_OB_PRIORITY	BYTE	Error information: depending on error code
OB80_OB_NUM	BYTE	Error information: depending on error code
OB80_DATE_TIME	DATE_AND_TIME	Date and time of day when the OB was called

Information to access the local data can be found at the description of the OB 1.

Variables depending on error code

The variables dependent on the error code have the following allocation:

Error code	Variable	Bit	Description
01h			*Cycle time exceeded*
	OB80_ERROR_INFO		Run time of last scan cycle (ms)
	OB80_ERR_EV_CLASS		Class of the event that triggered the interrupt
	OB80_ERR_EV_NUM		Number of the event that triggered the interrupt
	OB80_OB_PRIORITY		Priority class of the OB which was being executed when the error occurred
	OB80_OB_NUM		Number of the OB which was being executed when the error occurred
02h			*The called OB is still being executed*
	OB80_ERROR_INFO		The respective temporary variable of the called block which is determined by OB80_ERR_EV_CLASS and OB80_ERR_EV_NUM
	OB80_ERR_EV_CLASS		Class of the event that triggered the interrupt
	OB80_ERR_EV_NUM		Number of the event that triggered the interrupt
	OB80_OB_PRIORITY		Priority class of the OB causing the error
	OB80_OB_NUM		Number of the OB causing the error
05h and 06h			*Elapsed time-of-day interrupt due to moving the clock forward*
			Elapsed time-of-day interrupt on return to RUN after HOLD
	OB80_ERROR_INFO	Bit 0 = "1"	The start time for time-of-day interrupt 0 is in the past
		...	...
		Bit 7 = "1"	The start time for time-of-day interrupt 7 is in the past
		Bit 15 ... 8	Not used
	OB80_ERR_EV_CLASS		Not used
	OB80_ERR_EV_NUM		Not used
	OB80_OB_PRIORITY		Not used
	OB80_OB_NUM		Not used

continued ...

... continue error code

Error code	Variable	Bit	Description
07h	meaning of the parameters see error code 02h		*Overflow of OB request buffer for the current priority class* (Each OB start request for a priority class will be entered in the corresponding OB request buffer; after completion of the OB the entry will be deleted. If there are more OB start requests for a priority class than the maximum permitted number of entries in the corresponding Ob request buffer OB 80 will be called with error code 07h)
08h			*Synchronous-cycle interrupt time error*
09h			*Interrupt loss due to high interrupt load*
0Ah	OB80_ERROR_INFO		*Resume RUN after CiR (**C**onfiguration **i**n **R**UN)* CiR synchronizations time in ms

OB 81 - Power supply Error

Description

The operating system of the CPU calls OB 81 whenever an event occurs that is triggered by an error or fault related to the power supply (when entering and when outgoing event).

The CPU does not change to the STOP mode if OB 81 is not programmed.

You can disable or delay and re-enable the power supply error OB using SFCs 39 ... 42.

Local Data

The following table describes the start information of the OB 81 with default names of the variables and its data types:

Variable	Data type	Description
OB81_EV_CLASS	BYTE	Event class and identifiers: 39h: incoming event
OB81_FLT_ID	BYTE	Error code: 22h: Back-up voltage missing
OB81_PRIORITY	BYTE	Priority class: 28 (mode STARTUP)
OB81_OB_NUMBR	BYTE	OB-NR. (81)
OB81_RESERVED_1	BYTE	reserved
OB81_RESERVED_2	BYTE	reserved
OB81_RACK_CPU	WORD	Bit 2 ... 0: 000 (Rack number) Bit 3: 1 (master CPU) Bit 7 ... 4: 1111 (fix)
OB81_RESERVED_3	BYTE	reserved
OB81_RESERVED_4	BYTE	reserved
OB81_RESERVED_5	BYTE	reserved
OB81_RESERVED_6	BYTE	reserved
OB80_DATE_TIME	DATE_AND_TIME	Date and time of day when the OB was called

Information to access the local data can be found at the description of the OB 1.

OB 82 - Diagnostic Interrupt

Description

The system diagnostic is the detection, evaluation and reporting of messages which occur within a PLC system. Examples of errors for these messages could be errors in the user program, module failures or wire breaks on signaling modules.
If a module with diagnostic capability for which you have enabled the diagnostic interrupt detects an error, it outputs a request for a diagnostic interrupt to the CPU (when entering and outgoing event). The operating system then calls OB82.
The local variables of OB82 contain the logical base address as well as four bytes of diagnostic data of the defective module.
If OB82 has not been programmed, the CPU changes to the STOP mode.

You can delay and re-enable the diagnostic interrupt OB using SFC 41 DIS_AIRT and SFC 42 EN_AIRT.

Diagnostic in ring buffer

All diagnostic events reported to the CPU operating system are entered in the diagnostic buffer in the order in which they occurred, and with date and time stamp. This is a buffered memory area on the CPU that retains its contents even in the event of a memory reset.

The diagnostic buffer is a ring buffer with. VIPA CPUs offer space for 100 entries. When the diagnostic buffer is full, the oldest entry is overwritten by the newest. By use of the *PLC functions* of the Siemens SIMATIC manager the diagnostic buffer can be queried.

Besides of the standard entries in the diagnostic buffer, the VIPA CPUs support some additional specific entries in form of event-IDs. More information may be found at the manual of the CPU at the chapter "Deployment of the CPU ..." at "VIPA specific diagnostic entries".

Configurable Diagnostics

Programmable diagnostic events are reported only when you have set the parameters necessary to enable diagnostics. Non-programmable diagnostics events are always reported, regardless of whether or not diagnostics have been enabled.

Write diagnostics user entry with SFC

A diagnostic entry can be written to the diagnostic buffer by means of the system function SFC 52 WR_USMSG.

More information can be found at chapter "Integrated standard SFCs".

Read diagnostic data with SFC 59

You can use system function SFC 59 RD_REC (read record set) in OB 82 to obtain detailed error information. The diagnostic information are consistent until OB 82 is exited, that is, they remain "frozen". Exiting of OB 82 acknowledges the diagnostic interrupt on the module.

The module's diagnostic data is in record sets DS 0 and DS 1. The record set DS 0 contains 4 byte of diagnostic data describing the current status of the module. The contents of these 4 byte are identical to the contents of byte 8 ... 11 of the OB 82 start information

Record set DS 1 contains the 4 byte from record set DS 0 and, in addition, the module specific diagnostic data.

More information about module specific diagnostic data can be found at the description of the appropriate module.

Local data

The following table describes the start information of the OB 82 with default names of the variables and its data types:

Variable	Data type	Description
OB82_EV_CLASS	BYTE	Event class and identifiers: 38h: outgoing event 39h: incoming event
OB82_FLT_ID	BYTE	Error code (42h)
OB82_PRIORITY	BYTE	Priority class: can be assigned via hardware configuration
OB82_OB_NUMBR	BYTE	OB number (82)
OB82_RESERVED_1	BYTE	reserved
OB82_IO_FLAG	BYTE	Input Module 54h Output Module 55h
OB82_MDL_ADDR	INT	Logical base address of the module where the fault occurred
OB82_MDL_DEFECT	BOOL	Module is defective
OB82_INT_FAULT	BOOL	Internal fault
OB82_EXT_FAULT	BOOL	External fault
OB82_PNT_INFO	BOOL	Channel fault
OB82_EXT_VOLTAGE	BOOL	External voltage failed
OB82_FLD_CONNCTR	BOOL	Front panel connector not plugged in
OB82_NO_CONFIG	BOOL	Module is not configured
OB82_CONFIG_ERR	BOOL	Incorrect parameters on module
OB82_MDL_TYPE	BYTE	Bit 3 ... 0: Module class Bit 4: Channel information exists Bit 5: User information exists Bit 6: Diagnostic interrupt from substitute Bit 7: Reserved
OB82_SUB_MDL_ERR	BOOL	Submodule is missing or has an error
OB82_COMM_FAULT	BOOL	Communication failure
OB82_MDL_STOP	BOOL	Operating mode (0: RUN, 1:STOP)
OB82_WTCH_DOG_FLT	BOOL	Watchdog timer responded
OB82_INT_PS_FLT	BOOL	Internal power supply failed
OB82_PRIM_BATT_FLT	BOOL	Battery exhausted
OB82_BCKUP_BATT_FLT	BOOL	Entire backup failed
OB82_RESERVED_2	BOOL	Reserved
OB82_RACK_FLT	BOOL	Expansion rack failure
OB82_PROC_FLT	BOOL	Processor failure
OB82_EPROM_FLT	BOOL	EPROM fault
OB82_RAM_FLT	BOOL	RAM fault
OB82_ADU_FLT	BOOL	ADC/DAC error
OB82_FUSE_FLT	BOOL	Fuse tripped
OB82_HW_INTR_FLT	BOOL	Hardware interrupt lost
OB82_RESERVED_3	BOOL	Reserved
OB82_DATE_TIME	DATE_AND_TIME	Date and time of day when the OB was called

Information to access the local data can be found at the description of the OB 1.

OB 85 - Program execution Error

Description

The operating system of the CPU calls OB 85 whenever one of the following events occurs:

- Start event for an OB that has not been loaded
- Error when the operating system accesses a block
- I/O access error during update of the process image by the system (if the OB 85 call was not suppressed due to the configuration)

The OB 85 may be delayed by means of the SFC 41 and re-enabled by the SFC 42.

Note!

If OB 85 has not been programmed, the CPU changes to STOP mode when one of these events is detected.

Local data

The following table describes the start information of the OB 85 with default names of the variables and its data types:

Variable	Type	Description
OB85_EV_CLASS	BYTE	Event class and identifiers: 35h 38h (only with error code B3h, B4h) 39h (only with error code B1h, B2h, B3h, B4h)
OB85_FLT_ID	BYTE	Error code (possible values: A1h, A2h, A3h, A4h, B1h, B2h, B3h, B4h)
OB85_PRIORITY	BYTE	Priority class: 26 (Default value mode RUN) 28 (mode ANLAUF)
OB85_OB_NUMBR	BYTE	OB number (85)
OB85_RESERVED_1	BYTE	reserved
OB85_RESERVED_2	BYTE	reserved
OB85_RESERVED_3	INT	reserved
OB85_ERR_EV_CLASS	BYTE	Class of the event that caused the error
OB85_ERR_EV_NUM	BYTE	Number of the event that caused the error
OB85_OB_PRIOR	BYTE	Priority class of the OB that was active when the error occurred
OB85_OB_NUM	BYTE	Number of the OB that was active when the error occurred
OB85_DATE_TIME	DATE_AND_TIME	Date and time of day when the OB was called

Information to access the local data can be found at the description of the OB 1.

OB 85 dependent on error codes

If you want to program OB 85 dependent on the possible error codes, we recommend that you organize the local variables as follows:

Variable	Type
OB85_EV_CLASS	BYTE
OB85_FLT_ID	BYTE
OB85_PRIORITY	BYTE
OB85_OB_NUMBR	BYTE
OB85_DKZ23	BYTE
OB85_RESERVED_2	BYTE
OB85_Z1	WORD
OB85_Z23	DWORD
OB85_DATE_TIME	DATE_AND_TIME

The following table shows the event that started OB 85:

OB85_EV_CLASS	OB85_FLT_ID	Variable	Description
35h	A1h, A2h		As a result of your configuration your program or the operating system creates a start event for an OB that is not loaded on the CPU.
	A1h, A2h	OB85_Z1	The respective local variable of the called OB that is determined by OB85_Z23.
	A1h, A2h	OB85_Z23	high word: Class and number of the event causing the OB call low word, high byte: Program level and OB active at the time of error low word, low byte: Active OB
35h	A3h		Error when the operating system accesses a module
		OB85_Z1	Error ID of the operating system high byte: 1: Integrated function 2: IEC-Timer low byte: 0: no error resolution 1: block not loaded 2: area length error 3: write-protect error

continued ...

... continue

OB85_EV_CLASS	OB85_FLT_ID	Variable	Description
		OB85_Z23	high word: block number low word: Relative address of the MC7 command causing the error. The block type must be taken from OB85_DKZ23. (88h: OB, 8Ch: FC, 8Eh: FB, 8Ah: DB)
35h	A4h		PROFINET DB cannot be addressed
34h	A4h		PROFINET DB can be addressed again
39h	B1h		I/O access error when updating the process image of the inputs
	B2h		I/O access error when transferring the output process image to the output modules
	B1h, B2h	OB85_DKZ23	ID of the type of process image transfer where the I/O access error happened. 10: Byte access 20: Word access 30: DWord access 57h: Transmitting a configured consistency range
	B1h, B2h	OB85_Z1	Reserved for internal use by the CPU: logical base address of the module If OB85_RESERVED_2 has the value 76h OB85_Z1 receives the return value of the affected SFC
	B1h, B2h	OB85_Z23	Byte 0: Part process image number Byte 1: Irrelevant, if OB85_DKZ23=10, 20 or 30 OB85_DKZ23=57: Length of the consistency range in bytes Byte 2, 3 The I/O address causing the PII, if OB85_DKZ23=10, 20 or 30 OB85_DKZ23=57: Logical start address of the consistency range
You obtain the error codes B1h and B2h if you have configured the repeated OB 85 call of I/O access errors for the system process image table update.			

continued ...

... continue

OB85_EV_CLASS	OB85_FLT_ID	Variable	Description
38h, 39h	B3h		I/O access error when updating the process image of the inputs, incoming/outgoing event
38h, 39h	B4h		I/O access error when updating the process image of the outputs, incoming/outgoing event
	B3h, B4h	OB85_DKZ23	ID of the type of process image transfer during which the I/O access error has occurred 10: Byte access 20: Word access 30: DWord access 57h: Transmitting a configured consistency range
	B3h, B4h	OB85_Z1	Reserved for internal use by the CPU: logical base address of the module If OB85_RESERVED_2 has the value 76h OB85_Z1 receives the return value of the affected SFC
	B3h, B4h	OB85_Z23	Byte 0: Part process image number Byte 1: Irrelevant, if OB85_DKZ23=10, 20 or 30 OB85_DKZ23=57: Length of the consistency range in bytes Byte 2, 3 The I/O address causing the PII, if OB85_DKZ23=10, 20 or 30 OB85_DKZ23=57: Logical start address of the consistency range
You obtain the error codes B3h or B4h, if you configured the OB 85 call of I/O access errors entering and outgoing event for process image table updating by the system. After a restart, all access to non-existing inputs and outputs will be reported as I/O access errors during the next process table updating.			

OB 86 - Slave Failure / Restart

Description

The operating system of the CPU calls OB 86 whenever the failure of a slave is detected (both when entering and outgoing event).

Note!
If OB 86 has not been programmed, the CPU changes to the STOP mode when this type of error is detected.

The OB 86 may be delayed by means of the SFC 41 and re-enabled by the SFC 42.

Local data

The following table describes the start information of the OB 86 with default names of the variables and its data types:

Variable	Type	Description
OB86_EV_CLASS	BYTE	Event class and identifiers: 38h: outgoing event 39h: incoming event
OB86_FLT_ID	BYTE	Error code: (possible values: C4h, C5h, C7h, C8h)
OB86_PRIORITY	BYTE	Priority class: may be assigned via hardware configuration
OB86_OB_NUMBR	BYTE	OB number (86)
OB86_RESERVED_1	BYTE	reserved
OB86_RESERVED_2	BYTE	reserved
OB86_MDL_ADDR	WORD	Depends on the error code
OB86_RACKS_FLTD	ARRAY (0 ... 31) OF BOOL	Depends on the error code
OB86_DATE_TIME	DATE_AND_TIME	Date and time of day when the OB was called

Information to access the local data can be found at the description of the OB 1.

OB 86 depending on error codes

If you want to program OB 86 dependent on the possible error codes, we recommend that you organize the local variables as follows:

Variable	Type
OB86_EV_CLASS	BYTE
OB86_FLT_ID	BYTE
OB86_PRIORITY	BYTE
OB86_OB_NUMBR	BYTE
OB86_RESERVED_1	BYTE
OB86_RESERVED_2	BYTE
OB86_MDL_ADDR	WORD
OB86_Z23	DWORD
OB86_DATE_TIME	DATE_AND_TIME

The following table shows the event started OB 86:

EV_CLASS	FLT_ID	Variable	Bit ...	Description
39h, 38h	C4h			Failure of a DP station
	C5h			Fault in a DP station
		OB86_MDL_ADDR		Logical base address of the DP master
		OB86_Z23		Address of the affected DP slave:
			Bit 7 ... 0	Number of the DP station
			Bit 15 ... 8	DP master system ID
			Bit 30 ... 16	Logical base address of the DP slave
			Bit 31	I/O identifier
38h	C7h			Return of a DP station, but error in module parameter assignment
		OB86_MDL_ADDR		Logical base address of the DP master
		OB86_Z23		Address of the DP slaves affected:
			Bit 7 ... 0	Number of the DP station
			Bit 15 ... 8	DP master system ID
			Bit 30 ... 16	Logical base address of the DP slave
			Bit 31	I/O identifier
	C8h			Return of a DP station, however discrepancy in configured and actual configuration
		OB86_MDL_ADDR		Logical base address of the DP master
		OB86_Z23		Address of the DP slaves affected:
			Bit 7 ... 0	Number of the DP station
			Bit 15 ... 8	DP master system ID
			Bit 30 ... 16	Logical base address of the DP slave
			Bit 31	I/O identifier

OB 100 - Reboot

Description

On a restart, the CPU sets both itself and the modules to the programmed initial state, deletes all not-latching data in the system memory, calls OB 100 and then executes the main program in OB 1.

Here the current program and the current data blocks generated by SFC remain in memory.

The VIPA CPU executes a startup with OB 100 as follows:

- after POWER ON and operating switch in RUN
- whenever you switch the mode selector from STOP to RUN
- after a request using a communication function (menu command from the programming device)

Even if no OB 100 is loaded into the CPU, the CPU goes to RUN without an error message.

Local data

The following table describes the start information of the OB 100 with default names of the variables and its data types:

Variable	Type	Description
OB100_EV_CLASS	BYTE	Event class and identifiers: 13h: active
OB100_STRTUP	BYTE	Startup request 81h: Manual restart request
OB100_PRIORITY	BYTE	Priority class: 27
OB100_OB_NUMBR	BYTE	OB number (100)
OB100_RESERVED_1	BYTE	reserved
OB100_RESERVED_2	BYTE	reserved
OB100_STOP	WORD	Number of the event that caused the CPU to STOP
OB100_STRT_INFO	DWORD	Supplementary information about the current startup (see next page)
OB100_DATE_TIME	DATE_AND_TIME	Date and time of day when the OB was called

Information to access the local data can be found at the description of the OB 1.

Allocation OB100_STR_INFO

The following table shows the allocation of OB100_STR_INFO variables:

Bit-No.	Explanation	Possible values (binary)	Description
31 - 24	Startup information	xxxx xxx0	No difference between expected and actual configuration
		xxxx xxx1	Difference between expected and actual configuration
		xxxx 0xxx	Clock for time stamp not battery-backed at last POWER ON
		xxxx 1xxx	Clock for time stamp battery-backed at last POWER ON
23 - 16	Startup just completed	0000 0011	Restart triggered with mode selector
		0000 0100	Restart triggered by command via MPI
		0001 0000	Automatic restart after battery-backed POWER ON
		0001 0011	Restart triggered with mode selector; last POWER ON battery-backed
		0001 0100	Restart triggered by command via MPI; last POWER ON battery-backed
		0010 0000	Automatic restart battery-backed POWER ON (with memory reset by system)
		0010 0011	Restart triggered with mode selector last POWER ON not battery-backed
		0010 0100	Restart triggered by command via MPI last POWER ON not battery-backed
15 - 12	Permissibility of automatic startup	0000	Automatic startup illegal, memory request requested
		0001	Automatic startup illegal, parameter modifications, etc. necessary
		0111	Automatic startup permitted
11 - 8	Permissibility of manual startup	0000	Manual startup illegal, memory request requested
		0001	Manual startup illegal, parameter modifications, etc. necessary
		0111	Manual startup permitted
7 - 0	Last valid intervention or setting of the automatic startup at POWER ON	0000 0000	No startup
		0000 0011	Restart triggered with mode selector
		0000 0100	Restart triggered by command via MPI
		0001 0000	Automatic restart after battery-backed POWER ON
		0001 0011	Restart triggered with mode selector; last POWER ON battery-backed
		0001 0100	Restart triggered by command via MPI; last POWER ON battery-backed
		0010 0000	Automatic restart after battery-backed POWER ON (with memory reset by system)
		0010 0011	Restart triggered with mode selector last POWER ON not battery-backed
		1010 0000	Restart triggered by command via MPI last POWER ON not battery-backed

OB 121 - Programming Error (Synchronous error)

Description

The operating system of the CPU calls OB 121 whenever an event occurs that is caused by an error related to the processing of the program. If OB 121 is not programmed, the CPU changes to STOP. For example, if your program calls a block that has not been loaded on the CPU, OB 121 is called.
OB 121 is executed in the same priority class as the interrupted block. So you have read/write access to the registers of the interrupted block.

Masking of start events

The CPU provides the following SFCs for masking and unmasking start events for OB 121 during the execution of your program:

- SFC 36 MSK_FLT masks specific error codes.
- SFC 37 DMSK_FLT unmasks the error codes that were masked by SFC 36.
- SFC 38 READ_ERR reads the error register.

Local data

The following table describes the start information of the OB 121 with default names of the variables and its data types:

Variable	Data type	Description
OB121_EV_CLASS	BYTE	Event class and identifiers: 25h
OB121_SW_FLT	BYTE	Error code (see next page)
OB121_PRIORITY	BYTE	Priority class: priority class of the OB in which the error occurred.
OB121_OB_NUMBR	BYTE	OB number (121)
OB121_BLK_TYPE	BYTE	Type of block where the error occurred 88h: OB, 8Ah: DB, 8Ch: FC, 8Eh: FB
OB121_RESEVED_1	BYTE	reserved (Data area and access type)
OB121_FLT_REG	WORD	Source of the error (depends on error code). For example: • Register where the conversation error occurred • Incorrect address (read/write error) • Incorrect timer/counter/block number • Incorrect memory area
OB121_BLK_NUM	WORD	Number of the block with command that caused the error.
OB121_PRG_ADDR	WORD	Relative address of the command that caused the error.
OB121_DATE_TIME	DATE_AND_TIME	Date and time of day when the OB was called.

Information to access the local data can be found at the description of the OB 1.

Error codes The variables dependent on the error code have the following meaning:

Error code	Variable	Description
21h		BCD conversion error
	OB121_FLT_REG:	ID for the register concerned (0000h: accumulator 1)
22h		Area length error when reading
23h		Area length error when writing
28h		Read access to a byte, word or double word with a pointer whose bit address is not 0.
29h		Write access to a byte, word or double word with a pointer whose bit address is not 0. Incorrect byte address. The data area and access type can be read from OB121_RESERVED_1.
	OB121_RESERVED_1	Bit 3 ... 0 memory area: 0: I/O area 1: process-image input table 2: process-image output table 3: bit memory 4: global DB 5: instance DB 6: own local data 7: local data of caller Bit 7 ... 4 access type: 0: bit access 1: byte access 2: word access 3: double word access
24h		Range error when reading
25h		Range error when writing
	OB121_FLT_REG	Contains the ID of the illegal area in the low byte (86h of own local data area)
26h		Error for timer number
27h		Error for counter number
	OB121_FLT_REG	Illegal number

continued ...

... continue

Error code	Variable	Description
30h		Write access to a write-protected global DB
31h		Write access to a write-protected instance DB
32h		DB number error accessing a global DB
33h		DB number error accessing an instance DB
	OB121_FLT_REG	Illegal DB number
34h		FC number error in FC call
35h		FB number error in FB call
3Ah		Access to a DB that has not been loaded; the DB number is in the permitted range
3Ch		Access to an FC that has not been loaded; the FC number is in the permitted range
3Dh		Access to an SFC that has not been loaded; the SFC number is in the permitted range
3Eh		Access to an FB that has not been loaded; the FB number is in the permitted range
3Fh		Access to an SFB that has not been loaded; the SFB number is in the permitted range
	OB121_FLT_REG	Illegal DB number

OB 122 - Periphery access Error

Description

The operating system of the CPU calls OB 122 whenever an error occurs while accessing data on a module. For example, if the CPU detects a read error when accessing data on an I/O module, the operating system calls OB 122. If OB 122 is not programmed, the CPU changes from the RUN mode to the STOP mode.
OB 122 is executed in the same priority class as the interrupted block. So you have read/write access to the registers of the interrupted block.

Masking of start events

The CPU provides the following SFCs for masking and unmasking start events for OB 122:

- SFC 36 MASK_FLT masks specific error codes
- SFC 37 DMASK_FLT unmasks the error codes that were masked by SFC 36
- SFC 38 READ_ERR reads the error register

Local data

The following table describes the start information of the OB 122 with default names of the variables and its data types:

Variable	Type	Description
OB122_EV_CLASS	BYTE	Event class and identifiers: 29h
OB122_SW_FLT	BYTE	Error code: 42h: I/O access error - reading 43h: I/O access error - writing
OB122_PRIORITY	BYTE	Priority class: Priority class of the OB where the error occurred
OB122_OB_NUMBR	BYTE	OB number (122)
OB122_BLK_TYPE	BYTE	No valid number is entered here
OB122_MEM_AREA	BYTE	Memory area and access type: Bit 3 ... 0: memory area 0: I/O area; 1: Process image of the inputs 2: Process image of the outputs Bit 7 ... 4: access type: 0: Bit access, 1: Byte access, 2: Word access, 3: Dword access
OB122_MEM_ADDR	WORD	Memory address where the error occurred
OB122_BLK_NUM	WORD	No valid number is entered here
OB122_PGR_ADDR	WORD	No valid number is entered here
OB122_DATE_TIME	DATE_AND_TIME	Date and time of day when the OB was called

Information to access the local data can be found at the description of the OB 1.

Chapter 3 Integrated SFBs

Overview

Here the description of the integrated function blocks of the VIPA standard CPUs of the systems 100V, 200V, 300V and 500V may be found.

Content

Overview

General

The system program of the CPU offers you some additional functions that you may use by calling FBs, FCs or OBs. Those additional functions are part of the system program and don't use any work memory. Although the additional functions may be requested, they cannot be read or altered.

The calling of an additional function via FB, FC or OB is registered as block change and influences the nesting depth for blocks.

Integrated SFBs

The following system function blocks (SFBs) are available:

SFB	Label	Description
SFB 0	CTU	Count forward
SFB 1	CTD	Count backwards
SFB 2	CTUD	Count forward and backwards
SFB 3	TP	Create pulse
SFB 4	TON	Create turn-on delay
SFB 5	TOF	Create turn-off delay
SFB 32	DRUM	Realization of a step sequential circuit with a max. of 16 steps
SFB 52	RDREC	DP-V1-SFB Reading a Data Record from a DP slave
SFB 53	WRREC	DP-V1-SFB Writing a Data Record in a DP slave
SFB 54	RALRM	DP-V1-SFB Receiving an Interrupt from a DP slave

SFB 0 - CTU - Up-counter

Description

The SFB 0 can be used as Up-counter. Here you have the following characteristics:

- If the signal at the up counter input *CU* changes from "0" to "1" (positive edge), the current counter value is incremented by 1 and displayed at output *CV*.
- When called for the first time with *R*="0" the counter value corresponds to the preset value at input *PV*.
- When the upper limit of 32767 is reached the counter will not be incremented any further, i.e. all rising edges at input *CU* are ignored.
- The counter is reset to zero if reset input *R* has signal state "1".
- Output *Q* has signal state "1" if *CV* ≥ *PV*.
- When it is necessary that the instances of this SFB are initialized after a restart, then the respective instances must be initialized in OB 100 with *R* = 1.

Parameters

Parameter	Declaration	Data type	Memory block	Description
CU	INPUT	BOOL	I, Q, M, D, L, constant	Count input
R	INPUT	BOOL	I, Q, M, D, L, constant	Reset input. *R* takes precedence over *CU*.
PV	INPUT	INT	I, Q, M, D, L, constant	Preset value. The effect of *PV* is described under parameter *Q*.
Q	OUTPUT	BOOL	I, Q, M, D, L	Status of the counter
CV	OUTPUT	INT	I, Q, M, D, L	Current count

CU

Count input:

This counter is incremented by 1 when a rising edge (with respect to the most recent SFB call) is applied to input *CU*.

R

Reset input:

The counter is reset to 0 when input *R* is set to "1", irrespective of the status of input *CU*.

PV

Preset value:

This value is the comparison value for the current counter value. Output *Q* indicates whether the current count is greater than or equal to the preset value *PV*.

Q

Status of the counter:

- *Q* is set to "1", if *CV* ≥ *PV* (current count ≥ preset value)
- else Q = "0"

CV

Current count:

- possible values: 0 ... 32767

SFB 1 - CTD - Down-counter

Description

The SFB 1 can be used as Down-counter. Here you have the following characteristics:

- If the signal state at the down counter input *CD* changes from "0" to "1" (positive edge), the current counter value is decremented by 1 and displayed at output *CV*.
- When called for the first time with *LOAD* = "0" the counter value corresponds to the preset value at input *PV*.
- When the lower limit of -32767 is reached the counter will not be decremented any further, i.e. all rising edges at input *CU* are ignored.
- When a "1" is applied to the *LOAD* input then the counter is set to preset value *PV* irrespective of the value applied to input *CD*.
- Output *Q* has signal state "1" if *CV* ≤ 0.
- When it is necessary that the instances of this SFB are initialized after a restart, then the respective instances must be initialized in OB 100 with *LOAD* = 1 and *PV* = required preset value for *CV*.

Parameters

Parameter	Declaration	Data type	Memory block	Description
CD	INPUT	BOOL	I, Q, M, D, L, constant	Count input
LOAD	INPUT	BOOL	I, Q, M, D, L, constant	Load input. *LOAD* takes precedence over *CD*.
PV	INPUT	INT	I, Q, M, D, L, constant	Preset value
Q	OUTPUT	BOOL	I, Q, M, D, L	Status of the counter
CV	OUTPUT	INT	I, Q, M, D, L	Current count

CD

Count input:

This counter is decremented by 1 when a rising edge (with respect to the most recent SFB call) is applied to input *CU*.

LOAD

Load input:

When a 1 is applied to the *LOAD* input then the counter is set to preset value *PV* irrespective of the value applied to input *CD*.

PV

Preset value:

The counter is set to preset value *PV* when the input *LOAD* is “1”.

Q

Status of the counter:

Q is set to

- 1, if 0 ≥ *CV* (Current count value smaller/even 0)
- else Q = "0"

CV

Current count:

- possible values: -32 768 ... 32 767

SFB 2 - CTUD - Up-Down counter

Description

The SFB 2 can be used as an Up-Down counter. Here you have the following characteristics:

- If the signal state at the up count input *CU* changes from "0" to "1" (positive edge), the counter value is incremented by 1 and displayed at output *CV*.
- If the signal state at the down count input *CD* changes from "0" to "1" (positive edge), the counter value is decremented by 1 and displayed at output *CV*.
- If both counter inputs have a positive edge, the current counter value does not change.
- When the count reaches the upper limit of 32767 any further edges are ignored.
- When the count reaches the lower limit of -32768 any further edges are ignored.
- When a "1" is applied to the *LOAD* input then the counter is set to preset value *PV*.
- The counter value is reset to zero if reset input *R* has signal state "1". Positive signal edges at the counter inputs and signal state "1" at the load input remain without effect while input *R* has signal state "1".
- Output *QU* has signal state "1", if *CV* ≥ *PV.*
- Output *QD* has signal state "1", if *CV* ≤ 0.
- When it is necessary that the instances of this SFB are initialized after a restart, then the respective instances must be initialized in OB 100 with:
  - when the counter is used as up-counter with *R* = "1"
  - when the counter is used as down-counter with *R* = 0 and *LOAD* = 1 and *PV* = preset value.

Parameters

Parameter	Declaration	Data type	Memory block	Description
CU	INPUT	BOOL	I, Q, M, D, L, constant	Count up input
CD	INPUT	BOOL	I, Q, M, D, L, constant	Count down input
R	INPUT	BOOL	I, Q, M, D, L, constant	Reset input, *R* takes precedence over *LOAD*.
LOAD	INPUT	BOOL	I, Q, M, D, L, constant	Load input, *LOAD* takes precedence over *CU* and *CD*.
PV	INPUT	INT	I, Q, M, D, L, constant	Preset value
QU	OUTPUT	BOOL	I, Q, M, D, L	Status of the up counter
QD	OUTPUT	BOOL	I, Q, M, D, L	Status of the down counter
CV	OUTPUT	INT	I, Q, M, D, L	Current count

CU Count up input:
A rising edge (with respect to the most recent SFB-call) at input *CU* increments the counter.

CD Count down input:
A rising edge (with respect to the most recent SFB-call) at input *CD* decrements the counter.

R Reset input:
When input *R* is set to "1" the counter is reset to 0, irrespective of the status of inputs *CU*, *CD* and *LOAD*.

LOAD Load input:
When the *LOAD* input is set to "1" the counter is preset to the value applied to *PV*, irrespective of the values of inputs *CU* and *CD*.

PV Preset value:
The counter is preset to the value applied to *PV*, when the *LOAD* input is set to 1.

QU Status of the down counter:
- *QU* is set to "1", if *CV* ≥ *PV* (Current count ≥ Preset value)
- else *QU* is 0.

QD Status of the down counter:
- *QD* is set to 1", if 0 ≥ *CV* (Current count smaller/= 0)
- else *QD* is 0.

CV Current count
- possible values: -32 768 ... 32 767

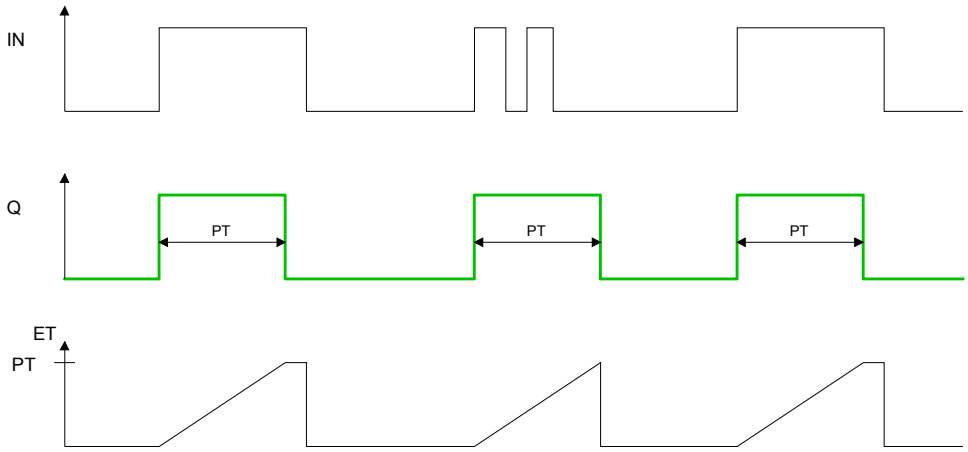
SFB 3 - TP - Create pulse

Description

The SFB 3 can be used to generate a pulse with a pulse duration equal to *PT*. Here you have the following characteristics:

- The pulse duration is only available in the STARTUP and RUN modes.
- The pulse is started with a rising edge at input *IN*.
- During *PT* time the output *Q* is set regardless of the input signal.
- The *ET* output provides the time for which output *Q* has already been set. The maximum value of the *ET* output is the value of the *PT* input. Output *ET* is reset when input *IN* changes to "0", however, not before the time *PT* has expired.
- When it is necessary that the instances of this SFB 3 are initialized after a restart, then the respective instances must be initialized in OB 100 with *PT* = 0 ms.

Time diagram



Parameters

Parameter	Declaration	Data type	Memory block	Description
IN	INPUT	BOOL	I, Q, M, D, L, constant	Start input
PT	INPUT	TIME	I, Q, M, D, L, constant	Pulse duration
Q	OUTPUT	BOOL	I, Q, M, D, L	Status of the time
ET	OUTPUT	TIME	I, Q, M, D, L	Expired time

IN Start input:
The pulse is started by a rising edge at input *IN*.

PT Pulse duration:
PT must be positive.
The range of these values is determined by data type TIME.

Q Output *Q*:
Output *Q* remains active for the pulse duration *PT*, irrespective of the subsequent status of the input signal

ET Expired time:
The duration for which output *Q* has already been active is available at output *ET* where the maximum value of this output can be equal to the value of *PT*. When input *IN* changes to 0 output *ET* is reset, however, this only occurs after *PT* has expired.

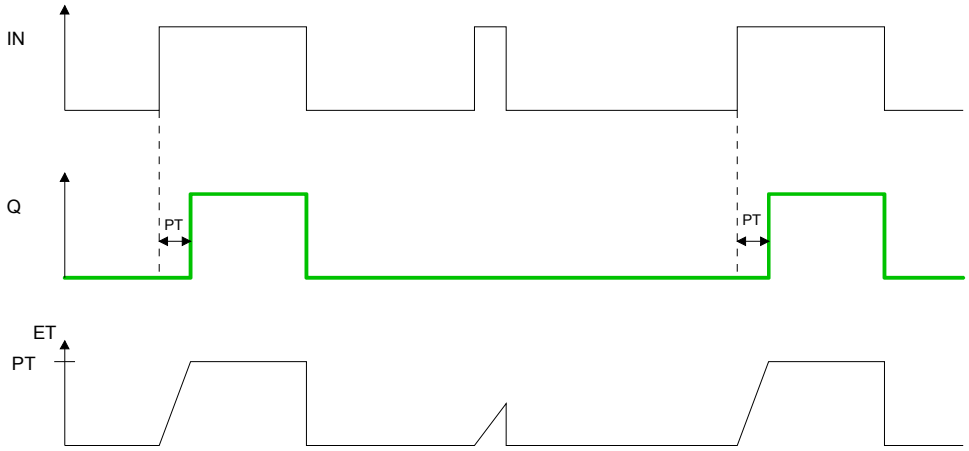
SFB 4 - TON - Create turn-on delay

Description

SFB 4 can be used to delay a rising edge by period *PT*. Here you have the following characteristics:

- The timer runs only in the STARTUP and RUN modes.
- A rising edge at the *IN* input causes a rising edge at output *Q* after the time *PT* has expired. *Q* then remains set until the *IN* input changes to 0 again. If the *IN* input changes to "0" before the time *PT* has expired, output *Q* remains set to "0".
- The *ET* output provides the time that has passed since the last rising edge at the *IN* input. Its maximum value is the value of the *PT* input. *ET* is reset when the *IN* input changes to "0".
- When it is necessary that the instances of this SFB are initialized after a restart, then the respective instances must be initialized in OB 100 with *PT* = 0 ms.

Timing diagram



Parameters

Parameter	Declaration	Type	Memory block	Description
IN	INPUT	BOOL	I, Q, M, D, L, constant	Start input
PT	INPUT	TIME	I, Q, M, D, L, constant	Time delay
Q	OUTPUT	BOOL	I, Q, M, D, L	Status of time
ET	OUTPUT	TIME	I, Q, M, D, L	Expired time

IN Start input:

The time delay is started by a rising edge at input *IN*. Output *Q* also produces a rising edge when time delay *PT* has expired.

PT Time delay:

Time delay applied to the rising edge at input *IN PT* must be. The range of values is defined by the data type TIME.

Q Output Q:

The time delay is started by a rising edge at input *IN*. Output *Q* also produces a rising edge when time delay *PT* has expired and it remains set until the level applied to input *IN* changes back to 0. If input *IN* changes to 0 before time delay *PT* has expired then output *Q* remains at "0".

ET Expired time:

Output *ET* is set to the time duration that has expired since the most recent rising edge has been applied to input *IN*. The highest value that output *ET* can contain is the value of input *PT*. Output *ET* is reset when input *IN* changes to "0".

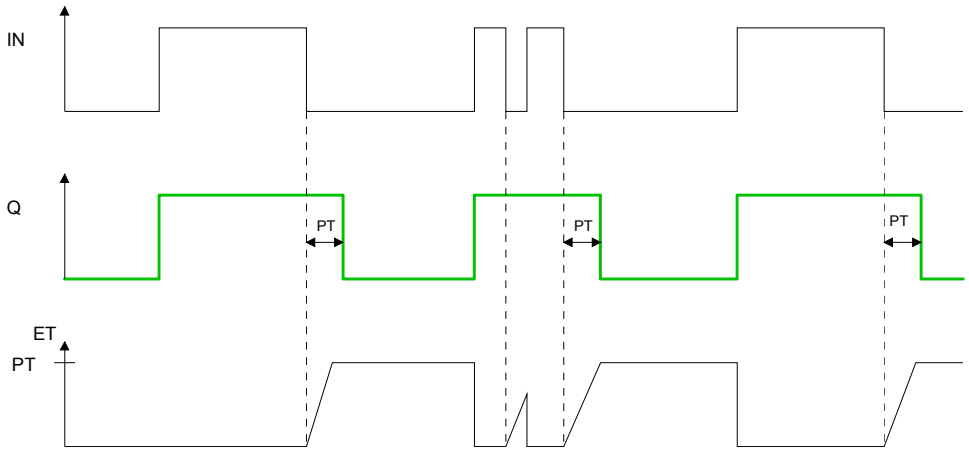
SFB 5 - TOF - Create turn-off delay

Description

SFB 5 can be used to delay a falling edge by period *PT*. Here you have the following characteristics:

- The timer runs only in the STARTUP and RUN modes.
- A rising edge at the *IN* input causes a rising edge at output *Q*. A falling edge at the *IN* input causes a falling edge at output *Q* delayed by the time *PT*. If the *IN* input changes back to "1" before the time *PT* has expired, output *Q* remains set to "1".
- The *ET* output provides the time that has elapsed since the last falling edge at the *IN* input. Its maximum value is, however the value of the *PT* input. *ET* is reset when the *IN* input changes to "1".
- When it is necessary that the instances of this SFB are initialized after a restart, then the respective instances must be initialized in OB 100 with *PT* = 0 ms.

Time diagram



Parameters

Parameter	Declaration	Data type	Memory block	Description
IN	INPUT	BOOL	I, Q, M, D, L, constant	Start input
PT	INPUT	TIME	I, Q, M, D, L, constant	Time delay
Q	OUTPUT	BOOL	I, Q, M, D, L	Status of time
ET	OUTPUT	TIME	I, Q, M, D, L	Expired time

IN Start input:

The time delay is started by a rising edge at input *IN* results in a rising edge at output *Q*. When a falling edge is applied to input *IN* output *Q* will also produce a falling edge when delay *PT* has expired. If the level at input *IN* changes to "1" before time delay *PT* has expired, then the level at output *Q* will remain at "1".

PT Time delay:

Time delay applied to the falling edge at input *IN PT* must be. The range of values is defined by the data type TIME.

Q Status of time:

The time delay is started by a rising edge at input *IN* results in a rising edge at output *Q*. When a falling edge is applied to input *IN* output *Q* will also produce a falling edge when delay *PT* has expired. If the level at input *IN* changes to "1" before time delay *PT* has expired, then the level at output *Q* will remain at "1".

ET Expired time:

The time period that has expired since the most recent falling edge at input *IN* is available from output *ET*. The highest value that output *ET* can reach is the value of input *PT*. Output *ET* is reset when the level at input *IN* changes to "1".

SFB 32 - DRUM - Realize a step-by-step switch

Description

Implementing a 16-state cycle switch using the SFB 32.

Parameter DSP defines the number of the first step, parameter *LST_STEP* defines the number of the last step.
Every step describes the 16 output bits *OUT0* ... *OUT15* and output parameter *OUT_WORD* that summarizes the output bits.
The cycle switch changes to the next step when a positive edge occurs at input *JOG* with respect to the previous SFB-call. If the cycle switch has already reached the last step and a positive edge is applied to *JOG* variables *Q* and *EOD* will be set, *DCC* is set to 0 and SFB 32 remains at the last step until a "1" is applied to the *RESET* input.

Time controlled switching

The switch can also be controlled by a timer. For this purpose parameter *DRUM_EN* must be set to "1". The next step of the cycle switch is activated when:

- the event bit *EVENTi* of the current step is set and
- when the time defined for the current step has expired.

The time is calculated as the product of time base *DTBP* and the timing factor that applies to the current step (from the *S_PRESET* field).

Note!

The remaining processing time *DCC* in the current step will only be decremented if the respective event bit *EVENTi* is set.

If input *RESET* is set to "1" when the call is issued to SFB 32 then the cycle switch changes to the step that you have specified as a number at input *DSP*.

Note!

Special conditions apply if parameter *DRUM_EN* is set to "1":

- timer-controlled cycle switching, if *EVENTi* = "1" with *DSP* = i = *LST_STEP*.
- event-controlled cycle switching by means of event bits *EVENTi*, when *DTBP* = "0".

In addition it is possible to advance the cycle switch at any time (even if *DRUM_EN* = "1") by means of the *JOG* input.

When this module is called for the first time the *RESET* input must be set to "1".
If the cycle switch has reached the last step and the processing time defined for this step has expired, then outputs *Q* and *EOD* will be set and SFB 32 will remain at the last step until the *RESET* input is set to "1".

The SFB 32 is only active in operating modes STARTUP and RUN.

If SFB 32 must be initialized after a restart it must be called from OB 100 with *RESET* = "1".

Parameters

Parameter	Declaration	Data type	Memory block	Description
RESET	INPUT	BOOL	I, Q, M, D, L, constant	Reset
JOG	INPUT	BOOL	I, Q, M, D, L, constant	Switch to the next stage
DRUM_EN	INPUT	BOOL	I, Q, M, D, L, constant	Control parameter
LST_STEP	INPUT	BYTE	I, Q, M, D, L, constant	Number of the last step
EVENTi, $1 \le i \le 16$	INPUT	BOOL	I, Q, M, D, L, constant	Event bit No. i (belongs to step i)
OUTj, $0 \le j \le 15$	OUTPUT	BOOL	I, Q, M, D, L	Output bit No. j
Q	OUTPUT	BOOL	I, Q, M, D, L	Status parameter
OUT_WORD	OUTPUT	WORD	I, Q, M, D, L, P	Output bits
ERR_CODE	OUTPUT	WORD	I, Q, M, D, L, P	*ERR_CODE* contains the error information if an error occurs when the SFB is being processed
JOG_HIS	VAR	BOOL	I, Q, M, D, L, constant	Not relevant to the user
EOD	VAR	BOOL	I, Q, M, D, L, constant	Identical with output parameter *Q*
DSP	VAR	BYTE	I, Q, M, D, L, P constant	Number of the first step
DSC	VAR	BYTE	I, Q, M, D, L, P constant	Number of the current step
DCC	VAR	DWORD	I, Q, M, D, L, P constant	The remaining processing time for the current step in ms
DTBP	VAR	WORD	I, Q, M, D, L, P constant	The time base in ms that applies to all steps
PREV_TIME	VAR	DWORD	I, Q, M, D, L, constant	Not relevant to the user
S_PRESET	VAR	ARRAY of WORD	I, Q, M, D, L, constant	One dimensional field containing the timing factors for every step
OUT_VAL	VAR	ARRAY of BOOL	I, Q, M, D, L, constant	Two-dimensional field containing the output values for every step
S_MASK	VAR	ARRAY of BOOL	I, Q, M, D, L, constant	Two-dimensional field containing the mask bits for every step.

RESET	Reset: The cycle switch is reset if this is set to "1". *RESET* must be set to "1" when the initial call is issued to the block.
JOG	A rising edge (with respect to the last SFB call) increments the cycle switch to the next stage if the cycle switch has not yet reached the last step. This is independent of the value of *DRUM_EN*.
DRUM_EN	Control parameter that determines whether timer-controlled cycle switching to the next step should be enabled or not ("1": enable timer-controlled increments).
LST_STEP	Number of the last step: • possible values: 1 ... 16
EVENTi, **1≤i≤16**	Event bit No. i (belonging to step i)
OUTj **0≤j≤15**	Output bit No. j (identical with bit No. j of *OUT_WORD*)
Q	Status parameter specifying whether the processing time that you have defined for the last step has expired.
OUT_WORD	Output bits summarized in a single variable.
ERR_CODE	*ERR_CODE* contains the error information if an error occurs when the SFB is being processed.
JOG_HIS	Not relevant to the user: input parameter *JOG* of the previous SFB-call.
EOD	Identical with output parameter *Q*
DSP	Number of the first step: • possible values 1 ... 16
DSC	Number of the current step
DCC	The remaining processing time for the current step in ms (only relevant if *DRUM_EN* = "1" and if the respective event bit = "1")
DTBP	The time base in ms that applies to all steps.
PREV_TIME	Not relevant to the user: system time of the previous SFB call.
S_PRESET	One-dimensional field containing the timing factors for every step. • Meaningful indices are: [1 ... 16]. In this case *S_PRESET* [x] contains the timing factor of step x.

OUT_VAL Two-dimensional field containing the output values for every step if you have not masked these by means of *S_MASK*.

- Meaningful indices are: [1 ... 16, 0 ... 15].

In this case *OUT_VAL* [x, y] contains the value that is assigned to output bit *OUTy* in step x.

S_MASK Two-dimensional field containing the mask bits for every step.

- Meaningful indices are: [1 ... 16, 0 ... 15].

In this case *S_MASK* [x, y] contains the mask bit for the value y of step x. Significance of the mask bits:

- 0: the respective value of the previous step is assigned to the output bit
- 1: the respective value of *OUT_VAL* is assigned to the output bit.

Error information *ERR_CODE*

When an error occurs the status of SFB 32 remains at the current value and output *ERR_CODE* contains one of the following error codes:

ERR_CODE	Description
0000h	No error has occurred
8081h	illegal value for *LST_STEP*
8082h	illegal value for *DSC*
8083h	illegal value for *DSP*
8084h	The product *DCC* = *DTBP* x *S_PRESET*[*DSC*] exceeds the value 2^{31-1} (appr. 24.86 Days)

SFB 52 - RDREC - Reading a Data Record from a DP-V1 slave

Note!

The SFB 52 RDREC interface is identical to the FB RDREC defined in the standard "PROFIBUS Guideline PROFIBUS Communication and Proxy Function Blocks according to IEC 61131-3".

Description

With the SFB 52 RDREC (read record) you read a record set with the number *INDEX* from a DP slave component (module or modules) that has been addressed via *ID*.

Specify the maximum number of bytes you want to read in *MLEN*. The selected length of the target area *RECORD* should have at least the length of *MLEN* bytes.

TRUE on output parameter *VALID* verifies that the record set has been successfully transferred into the target area *RECORD*. In this case, the output parameter *LEN* contains the length of the fetched data in bytes.

The output parameter *ERROR* indicates whether a record set transmission error has occurred. In this case, the output parameter *STATUS* contains the error information.

System dependent this block cannot be interrupted!

Note!

If a DP-V1 slave is configured using a GSD file (GSD stating with Rev. 3) and the DP interface of the DP master is set to Siemens "S7 compatible", than record sets must not be read from I/O modules in the user program with SFB 52. The reason is that in this case the DP master addresses the incorrect slot (configured slot +3).

Remedy: Set the interface for DP master to "DP-V1"!

Operating principle

The SFB 52 RDREC operates asynchronously, that is, processing covers multiple SFB calls. Start the job by calling SFB 52 with *REQ* = 1.

The job status is displayed via the output parameter *BUSY* and bytes 2 and 3 of output parameter *STATUS*. Here, the *STATUS* bytes 2 and 3 correspond with the output parameter *RET_VAL* of the asynchronously operating SFCs (see also meaning of *REQ*, *RET_VAL* and *BUSY* with Asynchronously Operating SFCs).

Record set transmission is completed when the output parameter *BUSY* = FALSE.

Parameters

Parameter	Declaration	Data type	Memory block	Description
REQ	INPUT	BOOL	I, Q, M, D ,L, constant	*REQ* = 1: Transfer record set
ID	INPUT	DWORD	I, Q, M, D, L, constant	Logical address of the DP slave component (module) For an output module, bit 15 must be set (e.g. for address 5: *ID*: DW = 8005h). For a combination module, the smaller of the two addresses should be specified.
INDEX	INPUT	INT	I, Q, M, D, L, constant	Record set number
MLEN	INPUT	INT	I, Q, M, D, L, constant	Maximum length in bytes of the record set information to be fetched
VALID	OUTPUT	BOOL	I, Q, M, D, L	New record set was received and valid
BUSY	OUTPUT	BOOL	I, Q, M, D, L	*BUSY* = 1: The read process is not yet terminated.
ERROR	OUTPUT	BOOL	I, Q, M, D, L	*ERROR* = 1: A read error has occurred.
STATUS	OUTPUT	DWORD	I, Q, M, D, L	Call *ID* (bytes 2 and 3) or error code.
LEN	OUTPUT	INT	I, Q, M, D, L	Length of the fetched record set information.
RECORD	IN_OUT	ANY	I, Q, M, D, L	Target area for the fetched record set.

Error information

See Receiving an interrupt from a DP slave with SFB 54 RALRM.

SFB 53 - WRREC - Writing a Data Record in a DP-V1 slave

Note!

The SFB 53 WRREC interface is identical to the FB WRREC defined in the standard "PROFIBUS Guideline PROFIBUS Communication and Proxy Function Blocks according to IEC 61131-3".

Description

With the SFB 53 WRREC (Write record) you transfer a record set with the number *INDEX* to a DP slave component (module) that has been addressed via *ID*.

Specify the byte length of the record set to be transmitted. The selected length of the source area *RECORD* should, therefore, have at least the length of *LEN* bytes.

TRUE on output parameter *DONE* verifies that the record set has been successfully transferred to the DP slave.

The output parameter *ERROR* indicates whether a record set transmission error has occurred. In this case, the output parameter *STATUS* contains the error information.

System dependent this block cannot be interrupted!

Note!

If a DP-V1 slave is configured using a GSD file (GSD stating with Rev. 3) and the DP interface of the DP master is set to Siemens "S7 compatible", than record sets must not be read from I/O modules in the user program with SFB 53. The reason is that in this case the DP master addresses the incorrect slot (configured slot +3).

Remedy: Set the interface for DP master to "DP-V1"!

Operating principle

The SFB 53 WRREC operates asynchronously, that is, processing covers multiple SFB calls. Start the job by calling SFB 52 with *REQ* = 1.

The job status is displayed via the output parameter *BUSY* and bytes 2 and 3 of output parameter *STATUS*. Here, the *STATUS* bytes 2 and 3 correspond with the output parameter *RET_VAL* of the asynchronously operating SFCs (see also meaning of *REQ, RET_VAL* and *BUSY* with Asynchronously Operating SFCs).

Please note that you must assign the same value to the actual parameter of *RECORD* for all SFB 53 calls that belong to one and the same job. The same applies to the *LEN* parameters.

Record set transmission is completed when the output parameter *BUSY* = FALSE.

Parameters

Parameter	Declaration	Data type	Memory block	Description
REQ	INPUT	BOOL	I, Q, M, D, L, constant	*REQ* = 1: Transfer record set
ID	INPUT	DWORD	I, Q, M, D, L, constant	Logical address of the DP slave component (module or submodule). For an output module, bit 15 must be set (e.g. for address 5: *ID*: DW = 8005h). For a combination module, the smaller of the two addresses should be specified.
INDEX	INPUT	INT	I, Q, M, D, L, constant	Record set number.
LEN	INPUT	INT	I, Q, M, D, L, constant	Maximum byte length of the record set to be transferred
DONE	OUTPUT	BOOL	I, Q, M, D, L	Record set was transferred
BUSY	OUTPUT	BOOL	I, Q, M, D, L	*BUSY* = 1: The write process is not yet terminated.
ERROR	OUTPUT	BOOL	I, Q, M, D, L	*ERROR* = 1: A write error has occurred
STATUS	OUTPUT	DWORD	I, Q, M, D, L	Call *ID* (bytes 2 and 3) or error code
LEN	OUTPUT	INT	I, Q, M, D, L	Length of the fetched record set information
RECORD	IN_OUT	ANY	I, Q, M, D, L	Record set

Error information

See Receiving an interrupt from a DP slave with SFB 54 RALRM.

SFB 54 - RALRM - Receiving an interrupt from a DP-V1 slave

Note!

The SFB 54 RALRM interface is identical to the FB RALRM defined in the standard "PROFIBUS Guideline PROFIBUS Communication and Proxy Function Blocks according to IEC 61131-3".

Description

The SFB 54 RALRM receives an interrupt with all corresponding information from a peripheral module (centralized structure) or from a DP slave component. It supplies this information to its output parameters.

The information in the output parameters contains the start information of the called OB as well as information of the interrupt source.

Call the SFB 54 only within the interrupt OB started by the CPU operating system as a result of the peripheral interrupt that is to be examined.

Note!

If you call SFB 54 RALRM in an OB for which the start event was not triggered by peripherals, the SFB supplies correspondingly reduced information on its outputs.
Make sure to use different instance DBs when you call SFB 54 in different OBs. If you want to evaluate data that are the result of an SFB 54 call outside of the associated interrupt OB you should moreover use a separate instance DP per OB start event.

Parameters

Parameter	Declaration	Data type	Memory block	Description
MODE	INPUT	INT	I, Q, M, D, L, constant	Operating mode
F_ID	INPUT	DWORD	I, Q, M, D, L, constant	Logical start address of the Component (module), from which interrupts are to be received.
MLEN	INPUT	INT	I, Q, M, D, L, constant	Maximum length in bytes of the data interrupt information to be received
NEW	OUTPUT	BOOL	I, Q, M, D, L	A new interrupt was received.
STATUS	OUTPUT	DWORD	I, Q, M, D, L	Error code of the SFB or DP master
ID	OUTPUT	DWORD	I, Q, M, D, L	Logical start address of the component (module), from which an interrupt was received. Bit 15 contains the I/O ID: 0: for an input address 1: for an output address
LEN	OUTPUT	INT	I, Q, M, D, L	Length of the received interrupt information
TINFO	IN_OUT	ANY	I, Q, M, D, L	(task information) Target range OB start and management information
AINFO	IN_OUT	ANY	I, Q, M, D, L	(interrupt information) Target area for header information and additional information. For *AINFO* you should provide a length of at least *MLEN* bytes.

MODE

You can call the SFB 54 in three operating modes (*MODE*):

0: shows the component that triggered the interrupt in the output parameter ID and sets the output parameter *NEW* to TRUE.

1: describes all output parameters, independent on the interrupt-triggering component.

2: checks whether the component specified in input parameter *F_ID* has triggered the interrupt.

- if not, *NEW* = FALSE
- if yes, *NEW* = TRUE, and all other outputs parameters are described.

Note!

If you select a target area *TINFO* or *AINFO* that is too short the SFC 54 cannot enter the full information.

TINFO Data structure of the target area (task information):

Byte	Data type	Description				
0 ... 19		Start information of the OB in which SFC 54 was currently called: Byte 0 ... 11: structured like the parameter *TOP_SI* in SFC 6 RD_SINFO Byte 12 ... 19: date and time the OB was requested				
20 ... 27		Management information:				
20	Byte	centralized: 0 decentralized: DP master system ID (possible values: 1 ... 255)				
21	Byte	central: Module rack number (possible values: 0 ... 31) distributed: Number of DP station (possible values: 0 ... 127)				
22	Byte	centralized: 0				
		decentralized:	Bit 3 ... 0	slave type	0000:	DP
					0001:	DPS7
					0010:	DPS7 V1
					0011:	DP-V1
					as of 0100:	reserved
			Bit 7 ... 4	Profile type	0000:	DP
					as of 0001:	reserved
23	Byte	centralized: 0				
		decentralized:	Bit 3 ... 0	Interrupt info type	0000:	Transparent (Interrupt originates from a configured decentralized module)
					0001:	Representative (Interrupt originating from a non-DP-V1 slave or a slot that is not configured)
					0010:	Generated interrupt (generated in the CPU)
					as of 0011:	reserved
			Bit 7 ... 4	Structure version	0000:	Initial
					as of 0001:	reserved

continued ...

... continue TINFO

Byte	Data type	Description	
24	Byte	centralized: 0	
		decentralized: Flags of the DP master interface	
		Bit 0 = 0:	Interrupt originating from an integrated DP interface
		Bit 0 = 1:	Interrupt originating from an external DP interface
		Bit 7 ... 1:	reserved
25	Byte	centralized: 0	
		decentralized: Flags of the DP slave interface	
		Bit 0:	EXT_DIAG_Bit of the diagnostic message frame, or 0 if this bit does not exist in the interrupt
		Bit 7 ... 1:	reserved
26, 27	WORD	centralized: 0	
		decentralized: PROFIBUS ID number	

AINFO Data structure of the target area (interrupt information):

Byte	Data type	Description		
0 ... 3		Header information		
0	Byte	Length of the received interrupt information in bytes		
		centralized: 4 ... 224		
		decentralized: 4 ... 63		
1	Byte	centralized: reserved		
		decentralized:	ID for the interrupt type	
			1:	Diagnostic interrupt
			2:	Hardware interrupt
			3:	Removal interrupt
			4:	Insertion interrupt
			5:	Status interrupt
			6:	Update interrupt
			31:	Failure of an expansion device, DP master system or DP station
			32 ... 126	manufacturer specific interrupt
2	**Byte**	**Slot number of the interrupt triggering component**		
3	Byte	centralized: reserved		
		decentralized:	Identifier	
			Bit 1, 0:	
			00	no further information
			01	incoming event, disrupted slot
			10	going event, slot not disrupted anymore
			11	going event, slot still disrupted
			Bit 2:	Add_Ack
			Bit 7 ... 3	Sequence number
4 ... 223		Additional interrupt information: module specific data for the respective interrupt:		
		centralized:	ARRAY[0] ... ARRAY[220]	
		decentralized:	ARRAY[0] ... ARRAY[59]	

TINFO and AINFO

Target Area:

Depending on the respective OB in which SFB 54 is called, the target areas *TINFO* and *AINFO* are only partially written. Refer to the table below for information on which info is entered respectively.

Interrupt type	OB	*TINFO* OB status information	*TINFO* management information	*AINFO* header information	*AINFO* additional interrupt information
Hardware interrupt	4x	Yes	Yes	Yes	centralized: No
					decentralized: as delivered by the DP slave
Status interrupt	55	Yes	Yes	Yes	Yes
Update interrupt	56	Yes	Yes	Yes	Yes
Manufacturer specific interrupt	57	Yes	Yes	Yes	Yes
Peripheral redundancy error	70	Yes	Yes	No	No
Diagnostic interrupt	82	Yes	Yes	Yes	centralized: Record set 1
					decentralized: as delivered by the DP slave
Removal/ Insertion interrupt	83	Yes	Yes	Yes	centralized: no
					decentralized: as delivered by the DP slave
Module rack/ Station failure	86	Yes	Yes	No	No
...	all other OBs	Yes	No	No	No

Error information The output parameter *STATUS* contains information. It is interpreted as ARRAY[1...4] OF BYTE the error information has the following structure:

Field element	Name	Description
STATUS[1]	Function_Num	00h: if no error Function ID from DP-V1-PDU: in error case 80h is OR linked. If no DP-V1 protocol element is used: C0h
STATUS[2]	Error_Decode	Location of the error ID
STATUS[3]	Error_1	Error ID
STATUS[4]	Error_2	Manufacturer specific error ID expansion: With DP-V1 errors, the DP master passes on *STATUS*[4] to the CPU and to the SFB. Without DP-V1 error, this value is set to 0, with the following exceptions for the SFB 52: • *STATUS*[4] contains the target area length from *RECORD*, if *MLEN* > the target area length from *RECORD* • STATUS[4]=*MLEN*, if the actual record set length < *MLEN* < the target area length from *RECORD*

STATUS[2] (Location of the error ID) can have the following values:

Error_Decode	Source	Description
00 ... 7Fh	CPU	No error no warning
80h	DP-V1	Error according to IEC 61158-6
81h ... 8Fh	CPU	8xh shows an error in the nth call parameter of the SFB.
FEh, FFh	DP Profile	Profile-specific error

STATUS[3] (Error ID) can have the following values:

Error_Decode	Error_Code_1	Explanation according to DP-V1	Description
00h	00h		no error, no warning
70h	00h	reserved, reject	Initial call; no active record set transfer
	01h	reserved, reject	Initial call; record set transfer has started
	02h	reserved, reject	Intermediate call; record set transfer already active
80h	90h	reserved, pass	Invalid logical start address
	92h	reserved, pass	Illegal Type for ANY Pointer
	93h	reserved, pass	The DP component addressed via *ID* or *F_ID* is not configured.
	A0h	read error	Negative acknowledgement while reading the module.
	A1h	write error	Negative acknowledgement while writing the module.
	A2h	module failure	at layer 2
	A3h	reserved, pass	DP protocol error with Direct-Data-Link-Mapper or User-Interface/User
	A4h	reserved, pass	Bus communication disrupted
	A5h	reserved, pass	-
	A7h	reserved, pass	DP slave or module is occupied (temporary error)
	A8h	version conflict	DP slave or module reports non-compatible versions
	A9h	feature not supported	Feature not supported by DP slave or module
	AA ... AFh	user specific	DP slave or module reports a manufacturer specific error in its application. Please check the documentation from the manufacturer of the DP slave or module.
	B0h	invalid index	Record set not known in module illegal record set number ≥256.
	B1h	write length error	Wrong length specified in parameter *RECORD*; with SFB 54: length error in *AINFO*.
	B2h	invalid slot	Configured slot not occupied.
	B3h	type conflict	Actual module type not equal to specified module type
	B4h	invalid area	DP slave or module reports access to an invalid area
	B5h	state conflict	DP slave or module not ready
	B6h	access denied	DP slave or module denies access

continued ...

... continue STATUS[3]

Error_Decode	Error_Code_1	Explanation according to DP-V1	Description
80h	B7h	invalid range	DP slave or module reports an invalid range for a parameter or value
	B8h	invalid parameter	DP slave or module reports an invalid parameter
	B9h	invalid type	DP slave or module reports an invalid type
	BAh ... BFh	user specific	DP slave or module reports a manufacturer specific error when accessing. Please check the documentation from the manufacturer of the DP slave or module.
	C0h	read constrain conflict	The module has the record set, however, there are no read data yet.
	C1h	write constrain conflict	The data of the previous write request to the module for the same record set have not yet been processed by the module.
	C2h	resource busy	The module currently processes the maximum possible jobs for a CPU.
	C3h	resource unavailable	The required operating resources are currently occupied.
	C4h		Internal temporary error. Job could not be carried out. Repeat the job. If this error occurs often, check your plant for sources of electrical interference.
	C5h		DP slave or module not available
	C6h		Record set transfer was canceled due to priority class cancellation
	C7h		Job canceled due to restart of DP masters
	C8h ... CFh		DP slave or module reports a manufacturer specific resource error. Please check the documentation from the manufacturer of the DP slave or module.
	Dxh	user specific	DP slave specific, Refer to the description of the DP slaves.
81h	00h ... FFh		Error in the initial call parameter (with SFB 54: *MODE*)
	00h		Illegal operating mode

continued ...

... continue STATUS[3]

Error_Decode	Error_Code_1	Explanation according to DP-V1	Description
82h	00h ... FFh		Error in the 2. call parameter.
...	...		...
88h	00h ... FFh		Error in the 8. call parameter (with SFB 54: *TINFO*)
	01h		Wrong syntax ID
	23h		Quantity frame exceeded or target area too small
	24h		Wrong range ID
	32h		DB/DI no. out of user range
	3Ah		DB/DI no. is NULL for area ID DB/DI or specified DB/DI does not exist.
89h	00h ... FFh		Error in the 9. call parameter (with SFB 54: *AINFO*)
	01h		Wrong syntax ID
	23h		Quantity frame exceeded or target area too small
	24h		Wrong range ID
	32h		DB/DI no. out of user range
	3Ah		DB/DI no. is NULL for area ID DB/DI or specified DB/DI does not exist
8Ah	00h ... FFh		Error in the 10. call parameter
...	...		...
8Fh	00h ... FFh		Error in the 15. call parameter
FEh, FFh			Profile-specific error

Chapter 4 Integrated Standard SFCs

Overview

Here the description of the integrated standard SFCs of the VIPA standard CPUs of the systems 100V, 200V, 300V and 500V may be found.

The description of the SFCs of the VIPA library may be found at the chapter "VIPA specific blocks".

Content

Overview Integrated standard SFCs

Standard SFCs The following standard system functions (SFCs) are available:

SFC	Label	Description
SFC 0	SET_CLK	Set time
SFC 1	READ_CLK	Read time
SFC 2	SET_RTM	Set operating hour counter
SFC 3	CTRL_RTM	Start/stop operating hour counter
SFC 4	READ_RTM	Read operating hour counter
SFC 5	GADR_LGC	Search logical address of a channel (only modules in rack 0)
SFC 6	RD_SINFO	Read start information of the current OB
SFC 12	D_ACT_DP	Activate or deactivate DP slaves
SFC 13	DPNRM_DG	Read slave diagnostic data
SFC 14	DPRD_DAT	Read consistent user data (also from DP slaves → DP master FW ≥ V3.00)
SFC 15	DPWR_DAT	Write consistent user data (also to DP slaves → DP master FW ≥ V3.00)
SCF 17	ALARM_SQ	Create acknowledgeable block related messages
SFC 18	ALARM_S	Create not acknowledgeable block related messages
SFC 19	ALARM_SC	Acknowledgement state of the last Alarm SQ-arrived-message
SFC 20 [1)]	BLKMOV	Copy variable within work memory
SFC 21 [1)]	FILL	Preset field within work memory
SFC 22	CREAT_DB	Create data block
SFC 23	DEL_DB	Delete data block
SFC 24	TEST_DB	Test data block
SFC 28	SET_TINT	Set time interrupt
SFC 29	CAN_TINT	Cancel time interrupt
SFC 30	ACT_TINT	Activate time interrupt
SFC 31	QRY_TINT	Request time interrupt
SFC 32	SRT_DINT	Start delay interrupt
SFC 33	CAN_DINT	Cancel delay interrupt
SFC 34	QRY_DINT	Request delay interrupt
SFC 36	MSK_FLT	Mask synchronal error event
SFC 37	DMSK_FLT	De-mask synchronal error event
SFC 38	READ_ERR	Read event status register
SFC 39	DIS_IRT	Disabling interrupts
SFC 40	EN_IRT	Enabling interrupts
SFC 41	DIS_AIRT	Delay of interrupt events
SFC 42	EN_AIRT	Abrogate delay of interrupt events
SFC 43	RE_TRIGR	Re-trigger cycle time control
SFC 44	REPL_VAL	Transfer replacement value to AKKU1
SFC 46	STP	Switch CPU in STOP
SFC 47 [1)]	WAIT	Delay program execution additionally to wait time
SFC 49	LGC_GADR	Search plug-in location of a logical address
SFC 50	RD_LGADR	Search all logical addresses of a module

continued ...

... continue Standard SFCs

SFC	Label	Description
SFC 51	RDSYSST	Read information from the system state list
SFC 52	WR_USMSG	Write user entry in diagnostic buffer (send via MPI in preparation)
SFC 54	RD_DPARM	Read predefined parameters
SFC 55	WR_PARM	Write dynamic parameters (only for analog-, digital modules, FMs, CPs and via PROFIBUS DP-V1 possible)
SFC 56	WR_DPARM	Write predefined parameters (only for analog-, digital modules, FMs, CPs and via PROFIBUS DP-V1 possible)
SFC 57	PARM_MOD	Parameterize module (only for analog-, digital modules, FMs, CPs and via PROFIBUS DP-V1 possible)
SFC 58	WR_REC	Write record set (only for analog-, digital modules, FMs, CPs and via PROFIBUS DP-V1 possible)
SFC 59	RD_REC	Read record set (only for analog-, digital modules, FMs, CPs and via PROFIBUS DP-V1 possible)
SFC 64 [1)]	TIME_TCK	Read millisecond timer
SFC 65	X_SEND	Send data to external partner
SFC 66	X_RCV	Receive data from external partner
SFC 67	X_GET	Read data from external partner
SFC 68	X_PUT	Write data to external partner
SFC 69	X_ABORT	Interrupt connection to external partner
SFC 81	UBLKMOV	Copy variable non-interruptible

1) This function block is interruptable and does not affect the interrupt reaction time.

General and Specific Error Information RET_VAL

Overview

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

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

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

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

RET_VAL (Return value)

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

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

Specific error code

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

A specific error code consists of the following two numbers:

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

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

General error codes RET_VAL

The parameter *RET_VAL* of some SFCs only returns general error information. No specific error information is available.
The general error code contains error information that can result from any system function. The general error code consists of the following two numbers:

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

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

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

General error codes

Error code	Description
8x7Fh	Internal Error. This error code indicates an internal error at parameter x. This error did not result from the actions if the user and he/she can therefore not resolve the error.
8x22h	Area size error when a parameter is being read.
8x23h	Area size error when a parameter is being written. This error code indicates that parameter x is located either partially or fully outside of the operand area or that the length of the bit-field for an ANY-parameter is not divisible by 8.
8x24h	Area size error when a parameter is being read.
8x25h	Area size error when a parameter is being written. This error code indicates that parameter x is located in an area that is illegal for the system function. The description of the respective function specifies the areas that are not permitted for the function.
8x26h	The parameter contains a number that is too high for a time cell. This error code indicates that the time cell specified in parameter x does not exist.
8x27h	The parameter contains a number that is too high for a counter cell (numeric fields of the counter). This error code indicates that the counter cell specified in parameter x does not exist.

continued ...

... continue

Error code	Description
8x28h	Orientation error when reading a parameter.
8x29h	Orientation error when writing a parameter. This error code indicates that the reference to parameter x consists of an operand with a bit address that is not equal to 0.
8x30h	The parameter is located in the write-protected global-DB.
8x31h	The parameter is located in the write-protected instance-DB. This error code indicates that parameter x is located in a write-protected data block. If the data block was opened by the system function itself, then the system function will always return a value 8x30h.
8x32h	The parameter contains a DB-number that is too high (number error of the DB).
8x34h	The parameter contains a FC-number that is too high (number error of the FC).
8x35h	The parameter contains a FB-number that is too high (number error of the FB). This error code indicates that parameter x contains a block number that exceeds the maximum number permitted for block numbers.
8x3Ah	The parameter contains the number of a DB that was not loaded.
8x3Ch	The parameter contains the number of a FC that was not loaded.
8x3Eh	The parameter contains the number of a FB that was not loaded.
8x42h	An access error occurred while the system was busy reading a parameter from the peripheral area of the inputs.
8x43h	An access error occurred while the system was busy writing a parameter into den peripheral area of the outputs.
8x44h	Error during the n-th (n > 1) read access after an error has occurred.
8x45h	Error during the n-th (n > 1) write access after an error has occurred. This error code indicates that access was denied to the requested parameter.

SFC 0 - SET_CLK - Set system clock

Description

The SFC 0 SET_CLK (set system clock) sets the time of day and the date of the clock in the CPU. The clock continues running from the new time and date.
If the clock is a master clock then the call to SFC 0 will start a clock synchronization cycle as well. The clock synchronization intervals are defined by hardware settings.

Parameters

Parameter	Declaration	Data type	Memory block	Description
PDT	INPUT	DT	D, L	Enter the new date and time at *PDT*.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	When an error occurs while the function is being processed then the returned value contains the respective error code.

PDT

Date and time are entered as data type DT.

Example:
date: 04.27.2006, time: 14:15:55 → DT#2006-04-27-14:15:55.

The time can only be entered with one-second accuracy. The day of the week is calculated automatically by SFC 0.

Remember that you must first create the data type DT by means of FC 3 D_TOD_DT before you can supply it to the input parameter
(see time functions; FC 3, FC 6, FC 7, FC 8, FC 33, FC 40, FC 1, FC 35, FC 34).

RET_VAL
(Return value)

Value	Description
0000h	no error
8080h	error in the date
8081h	error in the time

SFC 1 - READ_CLK - Read system clock

Description The SFC 1 READ_CLK (read system clock) reads the contents of the CPU clock. This returns the current time and date.

Parameters

Parameter	Declaration	Data type	Memory block	Description
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs when this function is being processed the return value contains the error code.
CDT	OUTPUT	DT	D, L	The current date and time are available at output *CDT*.

RET_VAL (Return value) SFC 1 does not return any specific error information.

CDT The current date and time are available at output *CDT*.

SFC 2 ... 4 - Run-time meter

Description

VIPA CPUs have 8 run-time meters.

You can use:

SFC 2	SET_RTM	set run-time meter
SFC 3	CTRL_RTM	run-time meter starting / stopping
SFC 4	READ_RTM	read run-time meter

You can use a runtime meter for a variety of applications:

- for measuring the runtime of a CPU
- for measuring the runtime of controlled equipment or connected devices.

Characteristics

When it is started, the runtime meter begins to count starting at the last recorded value. If you want it to start at a different initial value, you must explicitly specify this value with the SFC 2.

If the CPU changes to the STOP mode, or you stop the runtime meter, the CPU records the current value of the runtime meter. When a restart of the CPU is executed, the runtime meter must be restarted with the SFC 3.

Range of values

The runtime meter has a range of value from 0 ... 32767 hours.

SFC 2 - SET_RTM - Set run-time meter

Description The SFC 2 SET_RTM (set run-time meter) sets the run-time meter of the CPU to the specified value. VIPA CPUs contain 8 run-time meters.

Parameters

Parameter	Declaration	Data type	Memory block	Description
NR	INPUT	BYTE	I, Q, M, D, L, constant	Input *NR* contains the number of the run-time meter that you wish to set. Range: 0 ... 7
PV	INPUT	INT	I, Q, M, D, L, constant	Input *PV* contains the setting for the run-time meter.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed.

RET_VAL
(Return value)

Value	Description
0000h	no error
8080h	Incorrect number for the run-time meter
8081h	A negative value was supplied to parameter *PV*.

SFC 3 - CTRL_RTM - Control run-time meter

Description The SFC 3 CTRL_RTM (control run-time meter) starts or stops the run-time meter depending on the status of input *S*.

Parameters

Parameter	Declaration	Data type	Memory block	Description
NR	INPUT	BYTE	I, Q, M, D, L, constant	Input *NR* contains the number of the run-time meter that you wish to set. Range: 0 ... 7
S	INPUT	BOOL	I, Q, M, D, L, constant	Input *S* starts or stops the run-time meter. Set this signal to "0" to stop the run-time meter. Set this signal to "1" to start the run-time meter.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed.

RET_VAL
(Return value)

Value	Description
0000h	no error
8080h	Incorrect number for the run-time meter

SFC 4 - READ_RTM - Read run-time meter

Description

The SFC 4 READ_RTM (read run-time meter) reads the contents of the run-time meter. The output data indicates the current run-time and the status of the meter ("stopped" or "started").
When the run-time meter has been active for more than 32767 hours it will stop with this value and return value *RET_VAL* indicates the error message "8081h: overflow".

Parameters

Parameter	Declaration	Data type	Memory block	Description
NR	INPUT	BYTE	I, Q, M, D, L, constant	Input *NR* contains the number of the run-time meter that you wish to read. Range: 0 ... 7
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed.
CQ	OUTPUT	BOOL	I, Q, M, D, L	Output *CQ* indicates whether the run-time meter is started or stopped. • "0": the status of the run-time meter is stopped. • "1": the status of the run-time meter is started.
CV	OUTPUT	INT	I, Q, M, D, L	Output *CV* indicates the up to date value of the run-time meter.

RET_VAL
(Return value)

Value	Description
0000h	no error
8080h	Incorrect number for the run-time meter
8081h	run-time meter overflow

SFC 5 - GADR_LGC - Logical address of a channel

Description The SFC 5 GADR_LGC (convert geographical address to logical address) determines the logical address of the channel of a I/O module.

Parameters

Parameter	Declaration	Data type	Memory block	Description
SUBNETID	INPUT	BYTE	I, Q, M, D, L, constant	area identifier
RACK	INPUT	WORD	I, Q, M, D, L, constant	Rack No.
SLOT	INPUT	WORD	I, Q, M, D, L, constant	Slot-No.
SUBSLOT	INPUT	BYTE	I, Q, M, D, L, constant	Sub-module slot
SUBADDR	INPUT	WORD	I, Q, M, D, L, constant	Offset in user-data address space of the module
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed.
IOID	OUTPUT	BYTE	I, Q, M, D, L	area identifier
LADDR	OUTPUT	WORD	I, Q, M, D, L	Logical base address for the module

SUBNETID area identifier:

- "0": if the module is put locally (including expansion rack).
- DP-master-system-ID of the respective decentralized peripheral system when the slot is located in one of the decentralized peripheral devices.

Rack Rack No., when the address space identification is 0
Station number of the decentralized Peripheral device when falls the area identification >0

SLOT Slot-Number

SUBSLOT Sub-module slot
(when sub-modules cannot be inserted this parameter must be 0)

SUBADDR Offset in user-data address space of the module

RET_VAL (Return value) The return value contains an error code if an error is detected when the function is being processed.

Value	Description
0000h	no error
8094h	No subnet with the specified *SUBNETID* configured.
8095h	Illegal value for parameter *RACK*
8096h	Illegal value for parameter *SLOT*
8097h	Illegal value for parameter SUBSLOT
8098h	Illegal value for parameter *SUBADDR*
8099h	The slot has not been configured.
809Ah	The sub address for the selected slot has not been configured.

IOID Area identifier:

- 54h: peripheral input (PI)
- 55h: peripheral output (PQ)

For hybrid modules the SFC returns the area identification of the lower address. When the addresses are equal the SFC returns identifier 54h.

LADDR Logical base address for the module

SFC 6 - RD_SINFO - Read start information

Description

The SFC 6 RD_SINFO (read start information) retrieves the start information of the last OB accessed and that has not yet been processed completely, as well as the last startup OB. These start information items do not contain a time stamp. Two identical start information items will be returned when the call is issued from OB 100.

Parameters

Parameter	Declaration	Data type	Memory block	Description
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed.
TOP_SI	OUTPUT	STRUCT	D, L	Start information of the current OB
START_UP_SI	OUTPUT	STRUCT	D, L	Start information of the last OB that was started

TOP_SI and START_UP_SI

This refers to two identical structures as shown below.

Structure element	Data type	Description
EV_CLASS	BYTE	Bits 3 ... 0: event identifier Bits 7 ... 4: event class 1: Start events of standard-OBs 2: Start events of synchronous-error OBs 3: Start events of asynchronous-error OBs
EV_NUM	BYTE	event number
PRIORITY	BYTE	Number defining the priority level
NUM	BYTE	Structure element NUM contains the number of the current OB or of the last OB started
TYP2_3	BYTE	Data identifier 2_3: identifies the information entered into ZI2_3
TYP1	BYTE	Data identifier 1: identifies the information entered into ZI1
ZI1	WORD	Additional information 1
ZI2_3	DWORD	Additional information 2_3

Note!

The content of the structure elements shown in the table above corresponds exactly with the temporary variables of an OB. It must be remembered, however, that the name and the data type of the temporary variables in the different OBs might differ. Furthermore, the call interface of the OBs also contains the date and time at which call to the OB was requested.

RET_VAL (Return value)

The SFC 6 only returns general error information. No specific error information is available.

Example

The OB that was called last and that has not yet been completely processed serves as OB 80; the restart OB that was started last serves as OB 100.

The following table shows the assignment of the structure elements of parameter *TOP_SI* of SFC 6 and the respective local variables of OB 80.

TOP_SI Structure element	Data type	Logical Variable	Data type
EV_CLASS	BYTE	OB100_EV_CLASS	BYTE
EV_NUM	BYTE	OB80_FLT_ID	BYTE
PRIORITY	BYTE	OB80_PRIORITY	BYTE
NUM	BYTE	OB80_OB_NUMBR	BYTE
TYP2_3	BYTE	OB80_RESERVED_1	BYTE
TYP1	BYTE	OB80_ RESERVED_2	BYTE
ZI1	WORD	OB80_ERROR_INFO	WORD
ZI2_3	DWORD	OB80_ERR_EV_CLASS OB80_ERR_EV_NUM OB80_OB_PRIORITY OB80_OB_NUM	BYTE BYTE BYTE BYTE

The following table shows the assignment of the structure elements of parameter *START_UP_SI* of SFC 6 and the respective local variables of OB 100.

START_UP_SI Structure element	Data type	Logical Variable	Data type
EV_CLASS	BYTE	OB100_EV_CLASS	BYTE
EV_NUM	BYTE	OB100_STRTUP	BYTE
PRIORITY	BYTE	OB100_PRIORITY	BYTE
NUM	BYTE	OB100_OB_NUMBR	BYTE
TYP2_3	BYTE	OB100_RESERVED_1	BYTE
TYP1	BYTE	OB100_ RESERVED_2	BYTE
ZI1	WORD	OB100_STOP	WORD
ZI2_3	DWORD	OB100_STRT_INFO	DWORD

SFC 12 - D_ACT_DP - Activating and Deactivating of DP slaves

Description

With the SFC 12 D_ACT_DP, you can specifically deactivate and reactivate configured DP slaves. In addition, you can determine whether each assigned DP slave is currently activated or deactivated.

The SFC 12 cannot be used on PROFIBUS PA field devices, which are connected by a DP/PA link to a DP master system.

Note!

As long as any SFC 12 job is busy you cannot download a modified configuration from your PG to the CPU.
The CPU rejects initiation of an SFC 12 request when it receives the download of a modified configuration.

Application

If you configure DP slaves in a CPU, which are not actually present or not currently required, the CPU will nevertheless continue to access these DP slaves at regular intervals. After the slaves are deactivated, further CPU accessing will stop. In this way, the fastest possible DP bus cycle can be achieved and the corresponding error events no longer occur.

Example

Every one of the possible machine options is configured as a DP slave by the manufacturer in order to create and maintain a common user program having all possible options. With the SFC 12, you can deactivate all DP slaves, which are not present at machine startup.

How the SFC operates

The SFC 12 operates asynchronously, in other words, it is executed over several SFC calls. You start the request by calling the SFC 12 with *REQ* = 1.

The status of the job is indicated by the output parameters *RET_VAL* and *BUSY*.

Identifying a job

If you have started a deactivation or activation job and you call the SFC 12 again before the job is completed, the way in which the SFC reacts depends largely on whether the new call involves the same job: if the parameter *LADDR* matches, the SFC call is interpreted as a follow-on call.

Deactivating DP slaves

When you deactivate a DP slave with the SFC 12, its process outputs are set to the configured substitute values or to "0" (secure state).

The assigned DP master does not continue to address this DP slave. Deactivated DP slaves are not identified as fault or missing by the error LEDs on the DP master or CPU.

The process image of the inputs of deactivated DP slaves is updated with 0, that is, it is handled just as for failed DP slaves.

Note!

With VIPA you can not deactivate all DP slaves. At least 1 slave must remain activated at the bus.

If you are using your program to directly access the user data of a previously deactivated DP slave, the I/O access error OB (OB 122) is called, and the corresponding start event is entered in the diagnostic buffer.
If you attempt to access a deactivated DP slave with SFC (i.e. SFC 59 RD_REC), you receive the error information in *RET_VAL* as for an unavailable DP slave.

Deactivating a DP slaves OB 85, even if its inputs or outputs belong to the system-side process image to be updated. No entry is made in the diagnostic buffer.

Deactivating a DP slave does not start the slave failure OB 86, and the operating system also does not make an entry in the diagnostic buffer.

If a DP station fails after you have deactivated it with the SFC 12, the operating system does not detect the failure. As a result, there is no subsequent start of OB 86 or diagnostic buffer entry. The station failure is detected only after the station has been reactivated and indicated in *RET_VAL*.

If you wish to deactivate DP slaves functioning as transmitters in cross communication, we recommend that you first deactivate the receivers (listeners) that detect, which input data the transmitter is transferring to its DP master. Deactivate the transmitter only after you have performed this step.

Activating DP slaves

When you reactivate a DP slave with the SFC 12 it is configured and assigned parameters by the designated DP master (as with the return of a failed station). This activation is completed when the slave is able to transfer user data.

Activating a DP slaves does not start the program error OB 85, even if its inputs or outputs belong to the system-side process image to be updated. An entry in the diagnostic buffer is also not made.

Activating a DP slave does not start the slave failure OB 86, and the operating system also does not make an entry in the diagnostic buffer.

If you attempt to use the SFC 12 to activate a slave, who has been deactivated and is physically separated from the DP bus, a supervision time of 10sec expires. After this monitoring period has expired, the SFC returns the error message 80A2h. The slave remains deactivated. If the slave is reconnected to the DP bus at a later time, it must be reactivated with the SFC 12.

Note!

Activating a DP slave may be time-consuming. Therefore, if you wish to cancel a current activation job, start the SFC 12 again with the same value for *LADDR* and *MODE* = 2. Repeat the call of the SFC 12 until successful cancellation of the activation is indicated by *RET_VAL* = 0.

If you wish to activate DP slaves which take part in the cross communication, we recommend that you first activate the transmitters and then the receivers (listeners).

CPU startup

At a restart the slaves are activated automatically. After the CPU start-up, the CPU cyclically attempts to contact all configured and not deactivated slaves that are either not present or not responding.

Note!

The startup OB 100 does not support the call of the SFC 12.

Parameters

Parameter	Declaration	Data type	Memory block	Description
REQ	INPUT	BOOL	I, Q, M, D, L, constant	Level-triggered control parameter *REQ* = 1: execute activation or deactivation
MODE	INPUT	BYTE	I, Q, M, D, L, constant	Job ID Possible values: 0: request information on whether the addressed DP slave is activated or deactivated. 1: activate the DP slave 2: deactivate the DP slave
LAADR	INPUT	WORD	I, Q, M, D, L, constant	Any logical address of the DP slave
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is processed, the return value contains an error code.
BUSY	OUTPUT	BOOL	I, Q, M, D, L	Active code: *BUSY* = 1: the job is still active. *BUSY* = 0: the job was terminated.

RET_VAL
(Return value)

Value	Description
0000h	The job was completed without errors.
0001h	The DP slave is active (This error code is possible only with *MODE* = 0.)
0002h	The DP slave is deactivated (This error code is possible only with *MODE* = 0.)
7000h	First call with *REQ* = 0. The job specified with *LADDR* is not active; *BUSY* has the value 0.
7001h	First call with *REQ* = 1. The job specified with *LADDR* was triggered; *BUSY* has the value 1.
7002h	Interim call (*REQ* irrelevant). The activated job is still active; *BUSY* has the value 1.
8090h	You have not configured a module with the address specified in *LADDR*. You operate your CPU as I-Slave and you have specified in *LADDR* an address of this slave.

continued ...

... continue

Value	Description
8092h	For the addressed DP slave no activation job is processed at the present. (This error code is possible only with *MODE* = 1.)
8093h	No DP slave is assigned to the address stated in *LADDR* (no projection submitted), or the parameter *MODE* is not known.
80A1h	The addressed DP slave could not be parameterized. (This error code is possible only with *MODE* = 1.) **Note!** The SFC supplies this information only if the activated slave fails again during parameterization. If parameterization of a single module was unsuccessful the SFC returns the error information 0000h.
80A2h	The addressed DP slave does not return an acknowledgement.
80A3h	The DP master concerned does not support this function.
80A4h	The CPU does not support this function for external DP masters.
80A6h	Slot error in the DP slave; user data access not possible. (This error code is possible only with *MODE* = 1.) **Note!** The SFC returns this error information only if the active slave fails after parameterization and before the SFC ends. If only a single module is unavailable the SFC returns the error information 0000h.
80C1h	The SFC 12 was started and continued with another logical address. (This error code is possible only with *MODE* = 1.)
80C3h	• Temporary resource error: the CPU is currently processing the maximum possible activation and deactivation jobs. (this error code is possible only with *MODE* = 1 and *MODE* = 2). • The CPU is busy receiving a modified configuration. Currently you cannot enable/disable DP slaves.
F001h	Not all slaves may be deactivated. At least 1 slave must remain activated.
F002h	Unknown slave address

SFC 13 - DPNRM_DG - Read diagnostic data of a DP slave

Description

The SFC 13 DPNRM_DG (read diagnostic data of a DP slave) reads up-to-date diagnostic data of a DP slave. The diagnostic data of each DP slave is defined by EN 50 170 Volume 2, PROFIBUS.

Input parameter *RECORD* determines the target area where the data read from the slave is saved after it has been transferred without error. The read operation is started when input parameter *REQ* is set to 1.
The following table contains information about the principal structure of the slave diagnosis.
For additional information please refer to the manuals for the DP slaves that you are using.

Byte	description
0	station status 1
1	station status 2
2	station status 3
3	master-station number
4	manufacturer code (high byte)
5	manufacturer code (low byte)
6 ...	additional slave-specific diagnostics

Operation

The SFC 13 is executed as asynchronous SFC, i.e. it can be active for multiple SFC-calls. Output parameters *RET_VAL* and *BUSY* indicate the status of the command as shown by the following table.

Relationship between the call, *REQ*, *RET_VAL* and *BUSY*:

Seq. No. of the call	Type of call	REQ	RET_VAL	BUSY
1	first call	1	7001h or Error code	1 0
2 ... (n-1)	intermediate call	irrelevant	7002h	1
n	last call	irrelevant	If the command was completed without errors, then the number of bytes returned is entered as a positive number or the error code if an error did occur.	0

Parameters

Parameter	Declaration	Data type	Memory block	Description
REQ	INPUT	BOOL	I, Q, M, D, L, constant	*REQ* = 1: read request
LADDR	INPUT	WORD	I, Q, M, D, L, constant	The configured diagnostic address of the DP slave
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed. If no error did occur, then *RET_VAL* contains the length of the data that was transferred.
RECORD	OUTPUT	ANY	I, Q, M, D, L	Target area for the diagnostic data that has been read. Only data type BYTE is valid. The minimum length of the read record or respectively the target area is 6. The maximum length of the read record is 240. When the standard diagnostic data exceeds 240bytes on a norm slave and the maximum is limited to 244bytes, then only the first 240bytes are transferred into the target area and the respective overflow-bit is set in the data.
BUSY	OUTPUT	BOOL	I, Q, M, D, L	*BUSY* = 1: read operation has not been completed.

RECORD

The CPU tests the actual length of the diagnostic data that was read:
When the length of *RECORD*

- is less than the amount of data the data is discarded and the respective error code is entered into *RET_VAL*.
- is larger than or equal to the amount of data then the data is transferred into the target areas and *RET_VAL* is set to the actual length as a positive value.

Note!

It is essential that the matching *RECORD* parameters are be used for all calls that belong to a single task. A task is identified clearly by input parameter *LADDR* and *RECORD*.

Norm slaves

The following conditions apply if the amount of standard diagnostic data of the norm slave lies between 241 and 244bytes:
When the length of *RECORD*

- is less than 240bytes the data is discarded and the respective error code is entered into *RET_VAL*.
- is greater than 240bytes, then the first 240bytes of the standard diagnostic data are transferred into the target area and the respective overflow-bit is set in the data.

RET_VAL (Return value)

The return value contains an error code if an error is detected when the function is being processed.

If no error did occur, then *RET_VAL* contains the length of the data that was transferred.

Note!

The amount of read data for a DP slave depends on the diagnostic status.

Error information

More detailed information about general error information is to be found at the beginning of this chapter.

The SFC 13 specific error information consists of a subset of the error information for SFC 59 RD_REC. More detailed information is available from the help for SFC 59.

SFC 14 - DPRD_DAT - Read consistent data

Description

The SFC 14 DPRD_DAT (read consistent data of a DP norm slave) reads consistent data from a DP norm slave. The length of the consistent data must be three or more than four bytes, while the maximum length is 64Byte. Please refer to the manual of your specific CPU for details. Input parameter *RECORD* defines the target area where the read data is saved when the data transfer has been completed without errors. The length of the respective target area must be the same as the length that you have configured for the selected module.
If the module consists of a DP-norm slave of modular construction or with multiple DP-identifiers, then a single SFC 14 call can only access the data of a single module / DP-identifier at the configured start address.
SFC 14 is used because a load command accessing the periphery or the process image of the inputs can read a maximum of four contiguous bytes.

Definition

consistent data

Consistent data is data, where the contents belongs to the same category and that may not be separated. It is, for instance, important that data returned by analog modules is always processed consistently, i.e. the value returned by analog modules must not be modified incorrectly when it is read at two different times.

Parameters

Parameter	Declaration	Data type	Memory block	Description
LADDR	INPUT	WORD	I, Q, M, D, L, constant	Configured start address of the receive data buffer of the module from which the data must be read
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed
RECORD	OUTPUT	ANY	I, Q, M, D, L	Target area for the user data that was read. The length must be exactly the same as the length that was configured for the selected module. Only data type BYTE is permitted.

RET_VAL
(Return value)

Value	Description
0000h	No error has occurred.
8090h	You have not configured a module for the logical base address that you have specified, or you have ignored the restrictions that apply to the length of the consistent data.
8092h	The ANY-reference contains a type that is not equal to BYTE.
8093h	No DP-module from which consistent data can be read exists at the logical address that was specified under *LADDR*.
80A0h	Incorrect start address for the address range in the transfer I/O buffer.
80B0h	Slave failure at the external DP-interface
80B1h	The length of the specified target area is not equal to the configured user data length.
80B2h	External DP-interface system error
80B3h	External DP-interface system error
80C0h	External DP-interface system error
80C2h	External DP-interface system error
80Fxh	External DP-interface system error
87xyh	External DP-interface system error
808xh	External DP-interface system error

SFC 15 - DPWR_DAT - Write consistent data

Description

The SFC 15 DPWR_DAT (write consistent data to a DP-norm slave) writes consistent data that is located in parameter *RECORD* to the DP-norm slave. The length of the consistent data must be three or more than four bytes, while the maximum length is 64Byte. Please refer to the manual of your specific CPU for details. Data is transferred synchronously, i.e. the write process is completed when the SFC has terminated. The length of the respective source area must be the same as the length that you have configured for the selected module.
If the module consists of a DP-norm slave of modular construction, then you can only access a single module of the DP-slave.
The SFC 15 is used because a transfer command accessing the periphery or the process image of the outputs can write a maximum of four contiguous bytes.

Definition

Consistent data

Consistent data is data, where the contents belongs to the same category and that may not be separated. For instance, it is important that data returned by analog modules is always processed consistently, i.e. the value returned by analog modules must not be modified incorrectly when it is read at two different times.

Parameters

Parameter	Declaration	Data type	Memory block	Description
LADDR	INPUT	WORD	I, Q, M, D, L, constant	Configured start address of the output buffer of the module to which the data must be written
RECORD	INPUT	ANY	I, Q, M, D, L	Source area for the user data that will be written. The length must be exactly the same as the length that was configured for the selected module. Only data type BYTE is permitted.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed.

RET_VAL
(Return value)

Value	Description
0000h	No error has occurred.
8090h	You have not configured a module for the logical base address that you have specified, or you have ignored the restrictions that apply to the length of the consistent data.
8092h	The ANY-reference contains a type that is not equal to BYTE.
8093h	No DP-module to which consistent data can be written exists at the logical address that was specified under *LADDR*.
80A1h	The selected module has failed.
80B0h	Slave failure at the external DP-interface
80B1h	The length of the specified source area is not equal to the configured user data length.
80B2h	External DP-interface system error
80B3h	External DP-interface system error
80C1h	The data of the write command that was previously issued to the module has not yet been processed.
80C2h	External DP-interface system error
80Fxh	External DP-interface system error
85xyh	External DP-interface system error
808xh	External DP-interface system error

SFC 17 - ALARM_SQ and SFC 18 - ALARM_S

Description

Every call to the SFC 17 ALARM_SQ and the SFC 18 ALARM_S generates a message that can have an associated value. This message is sent to all stations that have registered for this purpose. The call to the SFC 17 and the SFC 18 can only be issued if the value of signal *SIG* triggering the message was inverted with respect to the previous call. If this is not true output parameter *RET_VAL* will contain the respective information and the message will not be sent. Input *SIG* must be set to "1" when the call to the SFC 17 and SFC 18 is issued for the first time, else the message will not be sent and *RET_VAL* will return an error code.

Note!

The SFC 17 and the SFC 18 should always be called from a FB after you have assigned the respective system attributes to this FB.

System resources

The SFC 17 and the SFC 18 occupy temporary memory that is also used to save the last two signal statuses with a time stamp and the associated value. When the call to the SFC occurs at a time when the signal statuses of the two most recent "valid" SFC-calls has not been sent (signal overflow), then the current signal status as well as the last signal status are discarded and an overflow-code is entered into temporary memory. The signal that occurred before the last signal will be sent as soon as possible including the overflow-code.

Message acknowledgement

Messages sent by means of the SFC 17 can be acknowledged via a display device. The acknowledgement status for the last "message entering state" and the signal status of the last SFC 17-call may be determined by means of the SFC 19 ALARM_SC.
Messages that are sent by SFC 18 are always acknowledged implicitly. The signal status of the last SFC 18-call may be determined by means of the SFC 19 ALARM_SC.

Temporarily saving

The SFCs 17 and 18 occupy temporary memory that is also used to save the last two signal statuses with a time stamp and the associated value. When the call to the SFC occurs at a time when the signal statuses of the two most recent "valid" SFC-calls has not been sent (signal overflow), then the current signal status as well as the last signal status are discarded and an overflow-code is entered into temporary memory. The signal that occurred before the last signal will be sent as soon as possible including the overflow-code.

Instance overflow

The maximum number of SFC 17- and SFC 18-calls depends on the type of CPU being used.
A resource bottleneck (instance overflow) can occur when the number of SFC-calls exceeds the maximum number of dynamic instances.

This condition is signaled by means of an error condition in *RET_VAL* and via the registered display device.

Parameters

Parameter	Declaration	Data type	Memory block	Description
SIG	INPUT	BOOL	I, Q, M, D, L	The signal that triggered the message.
ID	INPUT	WORD	I, Q, M, D, L	Data channel for messages: EEEEh
EV_ID	INPUT	DWORD	Const. (I, Q, M, D, L)	Message number (0: not permitted)
SD	INPUT	ANY	I, Q, M, D, T, C	Associated value
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Error information

SD

Associated value

Maximum length: 12byte

Valid data types BOOL (bit field not permitted), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME

RET_VAL (Return value) The return value contains an error code if an error is detected when the function is being processed.

Value	Description
0000h	No error has occurred.
0001h	• The associated value exceeds the maximum length, or • application memory cannot be accessed (e.g. access to deleted DB). The message will be transferred. • The associated value points to the local data area
0002h	Warning: the last unused message acknowledgement memory has been allocated**.**
8081h	The specified *EV_ID* lies outside of the valid range.
8082h	Message loss because your CPU suffers from a lack of resources that are required to generate module related messages by means of SFCs.
8083h	Message loss because a signal of the same type is already available but could not be sent (signal overflow).
8084h	The triggering signal *SIG* for messages has the same value for the current and for the preceding SFC 17 / SFC 18 call.
8085h	The specified *EV_ID* has not been registered.
8086h	An SFC call for the specified *EV_ID* is already being processed with a lower priority class.
8087h	The value of the message triggering signal was 0 during the first call to the SFC 17, SFC 18.
8088h	The specified *EV_ID* has already been used by another type of SFC that is currently (still) occupying memory space.
8xyy	General error information

SFC 19 - ALARM_SC - Acknowledgement state last Alarm

Description

The SFC 19 ALARM_SC can be used to:

- determine the acknowledgement status of the last SFC 17-entering-state message and the status of the message triggering signal during the last SFC 17 ALARM_SQ call
- the status of the message triggering signal during the last SFC 18 ALARM_S call.

The predefined message number identifies the message and/or the signal. The SFC 19 accesses temporary memory that was allocated to the SFC 17 or SFC 18.

Parameters

Parameter	Declaration	Data type	Memory block	Description
EV_ID	INPUT	DWORD	I, Q, M, D, L, constant	Message number for which you want to determine the status of the signal during the last SFC call or the acknowledgement status of the last entering-state message (only for SFC 17!)
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Return value
STATE	OUTPUT	BOOL	I, Q, M, D, L	Status of the message triggering signal during the last SFC call.
Q_STATE	OUTPUT	BOOL	I, Q, M, D, L	If the specified parameter *EV_ID* belongs to an SFC 18 call: "1"
				If the specified parameter *EV_ID* belongs to an SFC 17 call: acknowledgement status of the last entering-state message: "0": not acknowledged "1": acknowledged

RET_VAL (Return value)

The return value contains an error code if an error is detected when the function is being processed.

Value	Description
0000h	No error has occurred.
8081h	The specified *EV_ID* lies outside of the valid range.
8082h	No memory is allocated to this *EV_ID* at present (possible cause: the status of the respective signal has never been "1", or it has already changed back to status "0".)
8xyy	General Error information

SFC 20 - BLKMOV - Block move

Description

The SFC 20 BLKMOV (block move) copies the contents of one block of memory (source field) into another block of memory (target field).
Any block of memory may be copied, with the exception of :

- the following blocks: FC, SFC, FB, SFB, OB, SDB
- counters
- timers
- memory blocks of the peripheral area.

It is also possible that the source parameter is located in another data block in load memory that is not relevant to the execution (DB that was compiled with key word UNLINKED).

Interruptability

No limits apply to the nesting depth as long as the source field is not part of a data block that only exists in load memory.
However, when interrupting an SFC 20 that copies blocks from a DB that is not relevant to the current process, then this SFC 20 cannot be nested any longer.

Parameters

Parameter	Declaration	Data type	Memory block	Description
SRCBLK	INPUT	ANY	I, Q, M, D, L	Defines the memory block that must be copied (source field). Arrays of data type STRING are not permitted.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed.
DSTBLK	OUTPUT	ANY	I, Q, M, D, L	Defines the destination memory block to which the data will be copied (target field). Arrays of data type STRING are not permitted.

Note!

Source and target field must not overlap. If the specified target field is larger than the source filed then only the amount of data located in the source field will be copied. When the specified target field should, however, be smaller than the source filed, then only the amount of data that the target field can accommodate will be copied.
If the type of the ANY-pointer (source or target) is BOOL, then the specified length must be divisible by 8, otherwise the SFC cannot be executed.
If the type of the ANY-pointer is STRING, then the specified length must be equal to 1.

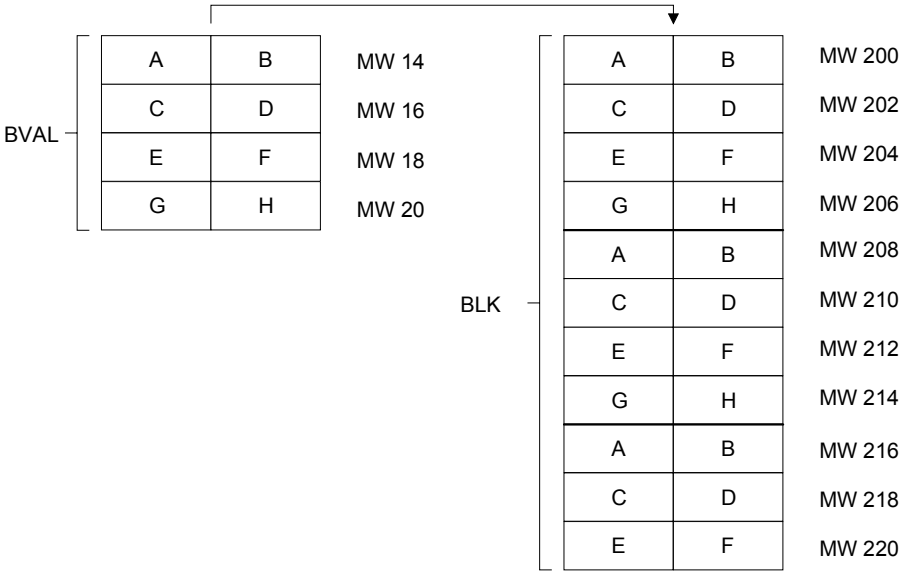
RET_VAL (Return value) The return value contains an error code if an error is detected when the function is being processed.

Value	Description
0000h	No error
8091h	The maximum nesting depth was exceeded

SFC 21 - FILL - Fill a field

Description

The SFC 21 FILL fills one block of memory (target field) with the contents of another block of memory (source field). The SFC 21 copies the contents from the source field into the specified target field until the block of memory has been filled completely.



Note!

Source and target field must not overlap. Even if the specified target field is not an integer multiple of the length of input parameter BVAL, the target field will be filled up to the last byte. If the target field is smaller than the source field, only the amount of data that can be accommodated by the target will be copied.

Values cannot be written with the SFC 21 into:

- the following blocks: FC, SFC, FB, SFB, SDB
- counters
- timers
- memory blocks of the peripheral area

Parameters

Parameter	Declaration	Data type	Memory block	Description
BVAL	INPUT	ANY	I, Q, M, D, L	Contains the value or the description of the source field that should be copied into the target field. Arrays of the data type STRING are not permitted.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed.
BLK	OUTPUT	ANY	I, Q, M, D, L	Contains the description of the target field that must be filled. Arrays of the data type STRING are not permitted.

Parameter is a structure

Pay attention to the following when the input parameter consists of a structure:
the length of a structure is always aligned with an even number of bytes. This means, that if you should declare a structure with an uneven number of bytes, the structure will require one additional byte in memory.

Example:

The structure is declared as follows:

STRUKTUR_7_BYTE: STRUCT
BYTE_1_2 : WORD
BYTE_3_4 : WORD
BYTE_5_6 : WORD
BYTE_7: BYTE
END_STRUCT

Structure "STRUKTUR_7_BYTE" requires 8bytes of memory.

RET_VAL (Return value)

The return value contains an error code if an error is detected when the function is being processed.

The SFC 21 only returns general error information. No specific error information is available.

SFC 22 - CREAT_DB - Create a data block

Description

The SFC 22 CREAT_DB (create data block) allows the application program to create a data block that does not contain any values. A data block is created that has a number in the specified range and with a specific size. The number assigned to the DB will always be the lowest number in the specified range. To create a DB with specific number you must assigned the same number to the upper and the lower limit of the range. If the application program already contains DBs then the respective numbers cannot be assigned any longer. The length of the DB must be an even number.

Interruptability

The SFC 22 may be interrupted by OBs with a higher priority. If a call is issued to an SFC 22 from an OB with a higher priority, then the call is rejected with error code 8091h.

Parameters

Parameter	Declaration	Data type	Memory block	Description
LOW_LIMIT	INPUT	WORD	I, Q, M, D, L, constant	The lower limit is the lowest number in the range of numbers that you may assign to your data block.
UP_LIMIT	INPUT	WORD	I, Q, M, D, L, constant	The upper limit is the highest number in the range of numbers that you may assign to your data block.
COUNT	INPUT	WORD	I, Q, M, D, L, constant	The counter defines the number of data bytes that you wish to reserve for your data block. Here you must specify an even number of bytes (maximum 65534).
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed.
DB_NUMBER	OUTPUT	WORD	I, Q, M, D, L	The data block number is the number of the data block that was created. When an error occurs (bit 15 of *RET_VAL* was set) a value of 0 is entered into *DB_NUMBFC*.

RET_VAL (Return value) The return value contains an error code if an error is detected when the function is being processed.

Value	Description
0000h	no error
8091h	You issued a nested call to the SFC 22.
8092h	The function "Create a DB" cannot be executed at present because • the function "Compress application memory" is active
80A1h	Error in the number of the DB: • the number is 0 • the number exceeds the CPU-specific number of DBs • lower limit > upper limit
80A2h	Error in the length of the DB: • the length is 0 • the length was specified as an uneven number • the length is larger than permitted by the CPU
80B1h	No DB-number available
80B2h	Insufficient memory available
80B3h	Insufficient contiguous memory available (compress the memory!).

SFC 23 - DEL_DB - Deleting a data block

Description

The SFC 23 DEL_DB (delete data block) deletes a data block in application memory and if necessary from the load memory of the CPU. The specified DB must not be open on the current level or on a level with a lower priority, i.e. it must not have been entered into one of the two DB-registers and also not into B-stack. Otherwise the CPU will change to STOP mode when the call to the SFC 23 is issued.

The following table indicates when a DB may be deleted by means of the SFC 23.

When the DB ...	then SFC 23 ...
was created by means of a call to SFC 22 "CREAT_DB",	can be used to delete it.
was not created with the key word UNLINKED,	can be used to delete it.

Interruptability

The SFC 23 may be interrupted by OBs with a higher priority. When another call is issued to the SFC the second call is rejected and *RET_VAL* is set to error code 8091h.

Parameters

Parameter	Declaration	Data type	Memory block	Description
DB_NUMBER	INPUT	WORD	I, Q, M, D, L, constant	Number of the DB that must be deleted.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed.

RET_VAL (Return value)

The return value contains an error code if an error is detected when the function is being processed.

Value	Description
0000h	no error
8091h	The maximum nesting depth of the respective CPU for nested calls to SFC 23 has been exceeded.
8092h	The function "Delete a DB" cannot be executed at present because • the function "Compress application memory" is active • you are copying the DB to be deleted from the CPU to an offline project
80A1h	Error in input parameter *DB_NUMBER*: • has a value of 0 • exceeds the maximum DB number that is possible on the CPU that is being used
80B1h	A DB with the specified number does not exist on the CPU
80B2h	A DB with the specified number was created with the key word UNLINKED
80B3h	The DB is located on the flash memory card

SFC 24 - TEST_DB - Test data block

Description

The SFC 24 TEST_DB (test data block) returns information about a data block that is located in the application memory of the CPU. The SFC determines the number of data bytes and tests whether the selected DB is write protected.

Parameters

Parameter	Declaration	Data type	Memory block	Description
DB_NUMBER	INPUT	WORD	I, Q, M, D, L, constant	Number of the DB that must be tested.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed.
DB_LENGTH	OUTPUT	WORD	I, Q, M, D, L	The number of data bytes that are contained in the selected DB.
WRITE_PROT	OUTPUT	BOOL	I, Q, M, D, L	Information about the write protection code of the selected DB (1 = write protected).

RET_VAL (Return value)

The return value contains an error code if an error is detected when the function is being processed.

Value	Description
0000h	no error
80A1h	Error in input parameter *DB_NUMBER*: the selected actual parameter • has a value of 0 • exceeds the maximum DB number that is possible on the CPU that is being used
80B1h	A DB with the specified number does not exist on the CPU.
80B2h	A DB with the specified number was created with the key word UNLINKED.

SFC 28 ... 31 - Time-of-day interrupt

Conditions

The following conditions must be satisfied before a time-of-day interrupt OB 10 may be called:

- The time-of-day interrupt OB must have been configured by hardware configuration or by means of the SFC 28 (SET_TINT) in the user program.
- The time-of-day interrupt OB must have been activated by hardware configuration or by means of the SFC 30 (ACT_TINT) in the user program.
- The time-of-day interrupt OB must not have been de-selected.
- The time-of-day interrupt OB must exist in the CPU.
- When the SFC 30 is used to set the time-of-day interrupt by a single call to the function the respective start date and time must not have expired when the function is initiated; the periodic execution initiates the time-of-day interrupt OB when the specified period has expired (start time + multiple of the period).

SFCs 28 ... 31

The system function are used as follows

- Set: SFC 28
- Cancel: SFC 29
- Activate: SFC 30
- Query: SFC 31

SFC 28 - SET_TINT

The SFC 28 SET_TINT (set time-of-day interrupt) defines the start date and time for the time-of-day interrupt - organization modules. The start time ignores any seconds and milliseconds that may have been specified, these are set to 0.

Parameters

Parameter	Declaration	Data type	Memory block	Description
OB_NR	INPUT	INT	I, Q, M, D, L, constant	Number of the OB, that is started at a time *SDT* + multiple of *PERIOD* (OB10, OB11).
SDT	INPUT	DT	D, L	Start date and start time
PERIOD	INPUT	WORD	I, Q, M, D, L, constant	Period from the start of *SDT*: 0000h = single 0201h = at minute intervals 0401h = hourly 1001h = daily 1201h = weekly 1401h = monthly 1801h = annually 2001h = at the end of a month
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed.

RET_VAL (Return value)

The return value contains an error code if an error is detected when the function is being processed.

Value	Description
0000h	No error has occurred.
8090h	*OB_NR* parameter error
8091h	*SDT* parameter error
8092h	*PERIOD* parameter error
80A1h	The stated date/time has already expired.

SFC 29 - CAN_TINT - Cancel time-of-day interrupt

The SFC 29 CAN_TINT (cancel time-of-day interrupt) deletes the start date and time of the specified time-of-day interrupt - organization block

Parameters

Parameter	Declaration	Data type	Memory block	Description
OB_NR	INPUT	INT	I, Q, M, D, L, constant	Number of the OB, in which the start date and time will be canceled (OB 10, OB 11).
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed.

RET_VAL (Return value)

Value	Description
0000h	No error has occurred.
8090h	*OB_NR* parameter error
80A0h	No start date/time was defined for the respective time-of-day interrupt OB.

SFC 30 - ACT_TINT - Activate time-of-day interrupt

The SFC 30 ACT_TINT (activate time-of-day interrupt) is used to activate the specified time-of-day interrupt - organization block

Parameters

Parameter	Declaration	Data type	Memory block	Description
OB_NR	INPUT	INT	I, Q, M, D, L, constant	Number of the OB to be activated (OB 10, OB 11)
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed.

RET_VAL (Return value)

Value	Description
0000h	No error has occurred.
8090h	*OB_NR* parameter error
80A0h	No start date/time was defined for the respective time-of.-day interrupt OB
80A1h	The activated time has expired; this error can only occur when the function is executed once only.

SFC 31 - QRY_TINT - Query time-of-day interrupt

The SFC 31 QRY_TINT (query time-of-day interrupt) can be used to make the status of the specified time-of-day interrupt - organization block available via the output parameter *STATUS*.

Parameters

Parameter	Declaration	Data type	Memory block	Description
OB_NR	INPUT	INT	I, Q, M, D, L, constant	Number of the OB, whose status will be queried (OB 10, OB 11).
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed.
STATUS	OUTPUT	WORD	I, Q, M, D, L	Status of the time-of-day interrupt.

RET_VAL
(Return value)

Value	Description
0000h	No error has occurred.
8090h	*OB_NR* parameter error

STATUS

Bit	Value	Description
0	0	The operating system has enabled the time-of-day interrupt.
1	0	New time-of-day interrupts are not discarded.
2	0	Time-of-day interrupt has not been activated and has not expired.
3	-	reserved
4	0	Time-of-day interrupt-OB has not been loaded.
5	0	An active test function disables execution of the time-of-day interrupt-OB.

SFC 32 - SRT_DINT - Start time-delay interrupt

Description

The SFC 32 SRT_DINT (start time-delay interrupt) can be used to start a time-delay interrupt that issues a call to a time-delay interrupt OB after the pre-configured delay time (parameter *DTIME*) has expired. Parameter *SIGN* specifies a user-defined code that identifies the start of the time-delay interrupt. While the function is being executed the values of *DTIME* and *SIGN* appear in the startup event information of the specified OB.

Conditions

The following conditions must be satisfied before a time-delay interrupt OB may be called:

- the time-delay interrupt OB must have been started (using the SFC 32)
- the time-delay interrupt OB must not have been de-selected.
- the time-delay interrupt OB must exist in the CPU.

Parameters

Parameter	Declaration	Data type	Memory block	Description
OB_NR	INPUT	INT	I, Q, M, D, L, constant	Number of the OB, that is started after the time delay (OB 20, OB 21).
DTIME	INPUT	TIME	I, Q, M, D, L, constant	The delay time (1 ... 60 000ms)
SIGN	INPUT	WORD	I, Q, M, D, L, constant	Code that is inserted into the startup event information of the OB when a call is issued to the time-delay interrupt.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed.

Accuracy

The time from the call to the SFC 32 and the start of the time-delay interrupt OB may be less than the configured time by no more than one millisecond, provided that no interrupt events have occurred that delay the call.

RET_VAL (Return value)

Value	Description
0000h	No error has occurred.
8090h	*OB_NR* parameter error
8091h	*DTIME* parameter error

SFC 33 - CAN_DINT - Cancel time-delay interrupt

Description

The SFC 33 CAN_DINT (cancel time-delay interrupt) cancels a time-delay interrupt that has already been started. The call to the respective time-delay interrupt OB will not be issued.

Conditions

The following conditions must be satisfied before a time-delay interrupt OB may be called:

- The time-delay interrupt OB must have been started (using the SFC 32).
- The time-delay interrupt OB must not have been de-selected.
- The time-delay interrupt OB must exist in the CPU.

Parameters

Parameter	Declaration	Data type	Memory block	Description
OB_NR	INPUT	INT	I, Q, M, D, L, constant	Number of the OB, that must be cancelled (OB 20, OB 21).
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed.

RET_VAL
(Return value)

Value	Description
0000h	No error has occurred.
8090h	*OB_NR* parameter error
80A0h	Time-delay interrupt has not been started.

SFC 34 - QRY_DINT - Query time-delay interrupt

Description

The SFC 34 QRY_DINT (query time-delay interrupt) can be used to make the status of the specified time-delay interrupt available via the output parameter *STATUS*.

Conditions

The following conditions must be satisfied before a time-delay interrupt OB may be called:

- The time-delay interrupt OB must have been started (using the SFC 32).
- The time-delay interrupt OB must not have been de-selected.
- The time-delay interrupt OB must exist in the CPU.

Parameters

Parameter	Declaration	Data type	Memory block	Description
OB_NR	INPUT	INT	I, Q, M, D, L, Constant	Number of the OB, that must be cancelled (OB 20, OB 21)
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed.
STATUS	OUTPUT	WORD	I, Q, M, D, L	Status of the time-delay interrupt

RET_VAL
(Return value)

Value	Description
0000h	No error has occurred.
8090h	*OB_NR* parameter error

STATUS

Bit	Value	Description
0	0	The operating system has enabled the time-delay interrupt.
1	0	New time-delay interrupts are not discarded.
2	0	Time-delay interrupt has not been activated and has not expired.
3	-	-
4	0	Time-delay interrupt-OB has not been loaded.
5	0	An active test function disables execution of the time-delay interrupt-OB.

SFC 36 - MSK_FLT - Mask synchronous errors

Description

The SFC 36 MSK_FLT (mask synchronous faults) is used to control the reaction of the CPU to synchronous faults by masking the respective synchronous faults.
The call to the SFC 36 masks the synchronous faults of the current priority class. If you set individual bits of the synchronous fault mask in the input parameters to "1" other bits that have previously been set will remain at "1". This result in new synchronous fault masks that can be retrieved via the output parameters. Masked synchronous faults are entered into an error register and do not issue a call to an OB. The error register is read by means of the SFC 38 READ_ERR.

Parameters

Parameter	Declaration	Data type	Memory block	Description
PRGFLT_SET_MASK	INPUT	DWORD	I, Q, M, D, L, constant	Programming faults that must be masked out
ACCFLT_SET_MASK	INPUT	DWORD	I, Q, M, D, L, constant	Access faults that must be masked out
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed.
PRGFLT_MASKED	OUTPUT	DWORD	I, Q, M, D, L	Masked programming faults
ACCFLT_MASKED	OUTPUT	DWORD	I, Q, M, D, L	Masked access errors

RET_VAL
(Return value)

Value	Description
0000h	None of the faults has previously been masked.
0001h	One or more of the faults has already been masked, however, the other faults will still be masked out.

SFC 37 - DMSK_FLT - Unmask synchronous errors

Description

The SFC 37 DMSK_FLT (unmask synchronous faults) unmasks any masked synchronous faults. A call to the SFC 37 unmasks the synchronous faults of the current priority class. The respective bits in the fault mask of the input parameters are set to "1". This results in new fault masks that you can read via the output parameters. Queried entries are deleted from in the error register.

Parameters

Parameter	Declaration	Data type	Memory block	Description
PRGFLT_RESET_MASK	INPUT	DWORD	I, Q, M, D, L, constant	Programming faults that must be unmasked
ACCFLT_RESET_MASK	INPUT	DWORD	I, Q, M, D, L, constant	Access faults that must be unmasked
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed.
PRGFLT_MASKED	OUTPUT	DWORD	I, Q, M, D, L	Masked programming faults
ACCFLT_MASKED	OUTPUT	DWORD	I, Q, M, D, L	Masked access errors

RET_VAL
(Return value)

Value	Description
0000h	All the specified faults have been unmasked.
0001h	One or more of the faults was not masked, however, the other faults will still be unmasked.

SFC 38 - READ_ERR - Read error register

Description

The SFC 38 READ_ERR (read error registers) reads the contents of the error register. The structure of the error register is identical to the structure of the programming fault and access fault masks that were defined as input parameters by means of the SFC 36 and 37. When you issue a call to the SFC 38 the specified entries are read and simultaneously deleted from the error register. The input parameters define which synchronous faults will be queried in the error register. The function indicates the masked synchronous faults of the current priority class that have occurred once or more than once. When a bit is set it signifies that the respective masked synchronous fault has occurred.

Parameters

Parameter	Declaration	Data type	Memory block	Description
PRGFLT_QUERY	INPUT	DWORD	I, Q, M, D, L, constant	Query programming faults
ACCFLT_QUERY	INPUT	DWORD	I, Q, M, D, L, constant	Query access faults
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed.
PRGFLT_ESR	OUTPUT	DWORD	I, Q, M, D, L	Programming faults that have occurred
ACCFLT_ESR	OUTPUT	DWORD	I, Q, M, D, L	Access faults that have occurred

RET_VAL
(Return value)

Value	Description
0000h	All the specified faults have been masked.
0001h	One or more of the faults that have occurred was not masked.

SFC 39 - DIS_IRT - Disabling interrupts

Description

With the SFC 39 DIS_IRT (disable interrupt) you disable the processing of new interrupts and asynchronous errors.
This means that if an interrupt occurs, the operating system of the CPU reacts as follows:

- if neither calls an interrupt OB asynchronous error OB,
- nor triggers the normal reaction if an interrupt OB or asynchronous error OB is not programmed.

If you disable interrupts and asynchronous errors, this remains in effect for all priority classes. The effects of SFC 39 can only be canceled again by calling the SFC 40 or by a restart.

Whether the operating system writes interrupts and asynchronous errors to the diagnostic buffer when they occur depends on the input parameter setting you select for *MODE*.

Note!

Remember that when you program the use of the SFC 39, all interrupts that occur are lost.

Parameters

Parameter	Declaration	Data type	Memory block	Description
MODE	INPUT	BYTE	I, Q, M, D, L, constant	Specifies which interrupts and asynchronous errors are disabled.
OB_NR	INPUT	INT	I, Q, M, D, L, constant	OB number
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is active, the return value contains an error code.

MODE

MODE	Description
00	All newly occurring interrupts and asynchronous errors are disabled (Synchronous errors are not disabled).
01	All newly occurring events belonging to a specified interrupt class are disabled. Identify the interrupt class by specifying it as follows: • Time-of-day interrupts: 10 • Time-delay interrupts: 20 • Cyclic interrupts: 30 • Hardware interrupts: 40 • Interrupts for DP-V1: 50 • Asynchronous error interrupts: 80 Entries into the diagnostic buffer are continued.
02	All new occurrences of a specified interrupt are disabled. You specify the interrupt using the OB number. Entries into the diagnostic buffer are continued.
80	All new occurrences of a specified interrupt are disabled. You specify the interrupt using the OB number. Entries continue to be made in the diagnostic buffer.
81	All new occurrences belonging to a specified interrupt class are disabled and are no longer entered in the diagnostic buffer. The operating system enters event 5380h in the diagnostic buffer.
82	All new occurrences belonging to a specified interrupt are disabled and are no longer entered in the diagnostic buffer. The operating system enters event 5380h in the diagnostic buffer.

RET_VAL
(Return value)

Value	Description
0000h	No error occurred.
8090h	The input parameter *OB_NR* contains an illegal value.
8091h	The input parameter *MODE* contains an illegal value.
8xyyh	General error information, see Evaluating Errors with the Output parameter *RET_VAL*.

SFC 40 - EN_IRT - Enabling interrupts

Description

With the SFC 40 EN_IRT (enable interrupt) you enable the processing of new interrupts and asynchronous errors that you previously disabled with the SFC 39. This means that if an interrupt event occurs, the operating system of the CPU reacts in one of the follows ways:

- it calls an interrupt OB or asynchronous error OB,
  or
- it triggers the standard reaction if an interrupt OB or asynchronous error OB is not programmed.

Parameters

Parameter	Declaration	Data type	Memory block	Description
MODE	INPUT	BYTE	I, Q, M, D, L, constant	Specifies which interrupts and asynchronous errors will be enabled.
OB_NR	INPUT	INT	I, Q, M, D, L, constant	OB number
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is active, the return value contains an error code.

MODE

MODE	Description
00	All newly occurring interrupts and asynchronous errors are enabled.
01	All newly occurring events belonging to a specified interrupt class are enabled. Identify the interrupt class by specifying it as follows: • Time-of-day interrupts: 10 • Time-delay interrupts: 20 • Cyclic interrupts: 30 • Hardware interrupts: 40 • Interrupts for DP-V1: 50 • Asynchronous error interrupts: 80
02	All newly occurring events of a specified interrupt are enabled. You specify the interrupt using the OB number.

RET_VAL
(Return value)

Value	Description
0000h	No error occurred.
8090h	The input parameter *OB_NR* contains an illegal value.
8091h	The input parameter *MODE* contains an illegal value.
8xyyh	General error information, see Evaluating Errors with the Output parameter *RET_VAL*.

SFC 41 - DIS_AIRT - Delaying interrupts

Description

The SFC 41 DIS_AIRT (disable alarm interrupts) disables processing of interrupt OBs and asynchronous fault OBs with a priority that is higher than the priority of the current OB. You can issue multiple calls to the SFC 41. The operating system will count the number of calls to the SFC 41. Processing of interrupt OBs is disabled until you issue an SFC 42 EN_AIRT to enable all interrupt OBs and asynchronous fault OBs that were disabled by means of SFC 41 or until processing of the current OB has been completed.
Any queued interrupt or asynchronous fault interrupts will be processed as soon as you enable processing by means of the SFC 42 EN_AIRT or when processing of the current OB has been completed.

Parameters

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Number of disable calls (= number of calls to the SFC 41)

RET_VAL (Return value)

When the SFC has been completed the return value *RET_VAL* indicates the number of disables, i.e. the number of calls to the SFC 41 (processing of all alarm interrupts is only enabled again when *RET_VAL* = 0).

SFC 42 - EN_AIRT - Enabling delayed interrupts

Description

The SFC 42 EN_AIRT (enable alarm interrupts) enables processing of high priority interrupt OBs and asynchronous fault OBs.
Every disabled interrupt must be re-enabled by means of the SFC 42. If you have disabled 5 different interrupts by means of 5 SFC 41 calls you must re-enable every alarm interrupt by issuing 5 individual SFC 42 calls.

Parameters

Parameter	Declaration	Data type	Memory block	Description
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Number of disabled interrupts when the SFC 42 has been completed or the error code when an error has occurred while the function was being processed.

RET_VAL (Return value)

When the SFC has been completed the return value *RET_VAL* indicates the number of disables, i.e. the number of calls to the SFC 41 (processing of all alarm interrupts is only enabled again when *RET_VAL* = 0).

Value	Description
8080h	The function was started in spite of the fact that the alarm interrupt had already been enabled.

SFC 43 - RE_TRIGR - Retrigger the watchdog

Description

The SFC 43 RE_TRIGR (retrigger watchdog) restarts the watchdog timer of the CPU.

Parameter and return values

The SFC 43 has neither parameters nor return values.

SFC 44 - REPL_VAL - Replace value to AKKU1

Description

The SFC 44 REPL_VAL (replace value) transfers a value into AKKU1 of the program level that cause the fault. A call to the SFC 44 can only be issued from synchronous fault OBs (OB 121, OB 122).

Application example for the SFC 44:
When an input module malfunctions so that it is not possible to read any values from the respective module then OB 122 will be started after each attempt to access the module. The SFC 44 REPL_VAL can be used in OB 122 to transfer a suitable replacement value into AKKU1 of the program level that was interrupted. The program will be continued with this replacement value. The information required to select a replacement value (e.g. the module where the failure occurred, the respective address) are available from the local variables of OB 122.

Parameters

Parameter	Declaration	Data type	Memory block	Description
VAL	INPUT	DWORD	I, Q, M, D, L, constant	Replacement value
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed.

RET_VAL
(Return value)

Value	Description
0000h	No error has occurred. A replacement value has been entered.
8080h	The call to the SFC 44 was not issued from a synchronous fault OB (OB 121, OB 122).

SFC 46 - STP - STOP the CPU

Description

The SFC 46 STP changes the operation mode of the CPU to STOP.

Parameter and return values

The SFC 46 has neither parameters nor return values.

SFC 47 - WAIT - Delay the application program

Description The SFC 47 WAIT can be used to program time delays or wait times from 1 up to 32767µs in your application program.

Interruptability The SFC 47 may be interrupted by high priority OBs.

Note!

Delay times that were programmed by means of the SFC 47 are minimum times that may be extended by the execution time of the nested priority classes as well as the load on the system!

Parameters

Parameter	Declaration	Data type	Memory block	Description
WT	INPUT	INT	I, Q, M, D, L, constant	Parameter *WT* contains the delay time in µs.

Error information The SFC 47 does not return specific error codes.

SFC 49 - LGC_GADR - Read the slot address

Description The SFC 49 LGC_GADR (convert logical address to geographical address) determines the slot location for a module from the logical address as well as the offset in the user-data address space for the module.

Parameters

Parameter	Declaration	Data type	Memory block	Description
IOID	INPUT	BYTE	I, Q, M, D, L, constant	Identifier for the address space: 54h = peripheral input (PI) 55h = peripheral output (PQ) For hybrid modules the SFC returns the area identifier of the lower address. When the addresses are equal the SFC returns identifier 54h.
LADDR	INPUT	WORD	I, Q, M, D, L, constant	Logical address. For hybrid modules the lower of the two addresses must be specified.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed.
AREA	OUTPUT	BYTE	I, Q, M, D, L	Area identifier: this defines how the remaining output parameters must be interpreted.
RACK	OUTPUT	WORD	I, Q, M, D, L	See next page.
SLOT	OUTPUT	WORD	I, Q, M, D, L	
SUBADDR	OUTPUT	WORD	I, Q, M, D, L	

AREA *AREA* specifies how the output parameters *RACK*, *SLOT* and *SUBADDR* must be interpreted. These dependencies are depicted below.

Value of AREA	System	Significance of RACK, SLOT and SUBADDR
0	-	reserved
1	Siemens S7-300	*RACK*: Rack number *SLOT*: Slot number *SUBADDR* : Address offset to base address
2	Decentra-lized periphery	*RACK* (Low Byte): Station number *RACK* (High Byte): DP master system ID *SLOT*: Slot number at station *SUBADDR* : Address offset to base address
3 ... 6	-	reserved

RET_VAL (Return value) The return value contains an error code if an error is detected when the function is being processed.

Value	Description
0000h	No error has occurred.
8090h	The specified logical address is not valid or an illegal value exists for parameter *IOID*

SFC 50 - RD_LGADR - Read all logical addresses of a module

Description

The SFC 50 RD_LGADR (read module logical addresses) determines all the stipulated logical addresses of a module starting with a logical address of the respective module.
You must have previously configured the relationship between the logical addresses and the modules. The logical addresses that were determined are entered in ascending order into the field *PEADDR* or into field PAADDR.

Parameters

Parameter	Declaration	Data type	Memory block	Description
IOID	INPUT	BYTE	I, Q, M, D, L, constant	Area identification: 54h = peripheral input (PI) 55h = peripheral output (PQ)
LADDR	INPUT	WORD	I, Q, M, D, L, constant	A logical address
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed.
PEADDR	OUTPUT	ANY	I, Q, M, D, L	Field for the PI-addresses, field elements must be of data type WORD.
PECOUNT	OUTPUT	INT	I, Q, M, D, L	Number of returned PI addresses
PAADDR	OUTPUT	ANY	I, Q, M, D, L	Field for PQ addresses, field elements must be of data type WORD.
PACOUNT	OUTPUT	INT	I, Q, M, D, L	Number of returned PQ addresses

RET_VAL (Return value)

The return value contains an error code if an error is detected when the function is being processed.

Value	Description
0000h	No error has occurred.
8090h	The specified logical address is not valid or illegal value for parameter *IOID.*
80A0 h	Error in output parameter *PEADDR*: data type of the field elements is not WORD.
80A1h	Error in output parameter PAADDR: data type of the field elements is not WORD.
80A2h	Error in output parameter *PEADDR*: the specified field could not accommodate all the logical addresses.
80A3h	Error in output parameter PAADDR: the specified field could not accommodate all the logical addresses.

SFC 51 - RDSYSST - Read system status list SSL

Description

With the SFC 51 RDSYSST (read system status) a partial list respectively an extract of a partial list of the SSL (**s**ystem **s**tatus **l**ist) may be requested.

Here with the parameters *SSL_ID* and *INDEX* the objects to be read are defined.

The *INDEX* is not always necessary. It is used to define an object within a partial list.

By setting *REQ* the query is started. As soon as *BUSY* = 0 is reported, the data are located in the target area *DR*.

Information about the SSL may be found in Chapter "System status list SSL".

Parameters

Parameter	Declaration	Data type	Memory block	Description
REQ	INPUT	BOOL	I, Q, M, D, L, constant	*REQ* = 1: start processing
SSL_ID	INPUT	WORD	I, Q, M, D, L, constant	*SSL-ID* of the partial list or the partial list extract
INDEX	INPUT	WORD	I, Q, M, D, L, constant	Type or number of an object in a partial list
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed
BUSY	OUTPUT	BOOL	I, Q, M, D, L	*BUSY* = 1: read operation has not been completed
SSL_HEADER	OUTPUT	STRUCT	D, L	WORD structure with 2 types: LENGTHDR: length record set N_DR: number of existing related records (for access to partial list header information) or number of records transmitted in DR.
DR	OUTPUT	ANY	I, Q, M, D, L	Target area for the SSL partial list or the extraction of the partial list that was read: If you have only read the SSL partial list header info of a SSL partial list, you may not evaluate DR, but only *SSL_HEADER*. Otherwise the product of LENGTHDR and N_DR shows the number of bytes stored in *DR*.

RET_VAL (Return value) The return value contains an error code if an error is detected when the function is being processed.

Value	Description
0000h	no error
0081h	The length of the result field is too low. The function still returns as many records as possible. The SSL header indicates the returned number of records.
7000h	First call with *REQ* = 0: data transfer not active; *BUSY* = 0.
7001h	First call with *REQ* = 1: data transfer initiated; *BUSY* = 1.
7002h	Intermediate call (*REQ* irrelevant): data transfer active; *BUSY* = 1.
8081h	The length of the result field is too low. There is not enough space for one record.
8082h	*SSL_ID* is wrong or unknown to the CPU or the SFC.
8083h	Bad or illegal *INDEX*.
8085h	Information is not available for system-related reasons, e.g. because of a lack of resources.
8086h	Record set may not be read due to a system error.
8087h	Record set may not be read because the module does not exist or it does not return an acknowledgement.
8088h	Record set may not be read because the current type identifier differs from the expected type identifier.
8089h	Record set may not be read because the module does not support diagnostic functions.
80A2h	DP protocol error - Layer-2 error (temporary fault).
80A3h	DP protocol error on user-interface/user (temporary fault)
80A4h	Bus communication failure. This error occurs between the CPU and the external DP interface (temporary fault).
80C5h	Decentralized periphery not available (temporary fault).

SFC 52 - WR_USMSG - Write user entry into diagnostic buffer

Description

The SFC 52 WR_USMSG (write user element in diagnosis buffer) writes a used defined diagnostic element into the diagnostic buffer.

Send diagnostic message

To determine whether it is possible to send user defined diagnostic messages you must issue a call to SFC 51 "RDSYSST" with parameters *SZL_ID* = 0132h and *INDEX* = 0005h. Sending of user defined diagnostic messages is possible if the fourth word of the returned record set is set to "1". If it should contain a value of "0", sending is not possible.

Send buffer full

The diagnostic message can only be entered into the send buffer if this is not full. At a maximum of 50 entries can be stored in the send buffer.

If the send buffer is full

- the diagnostic event is still entered into the diagnostic buffer
- the respective error message (8092h) is entered into parameter *RET_VAL*.

Partner not registered

When a user defined diagnostic message must be sent and no partner has registered, then

- the diagnostic event is still entered into the diagnostic buffer.
- the respective error message (0091h or 8091h) is entered into parameter *RET_VAL*.

The contents of an entry

The structure of the entry in the diagnostic buffer is as follows:

Byte	Contents
1, 2	Event ID
3	Priority class
4	OB number
5, 6	reserved
7, 8	Additional information 1
9, 10, 11, 12	Additional information 2
13 ... 20	Time stamp: The data type of the time stamp is Date_and_Time.

Event ID

Every event is assigned to an event ID.

Additional information

The additional information contains more specific information about the event. This information differs for each event. When a diagnostic event is generated the contents of these entries may be defined by the user.
When a user defined diagnostic message is sent to the partners this additional information may be integrated into the (event-ID specific) message text as an associated value.

Parameters

Parameter	Declaration	Data type	Memory block	Description
SEND	INPUT	BOOL	I, Q, M, D, L, constant	Enable sending of user defined diagnostic messages to all registered partners
EVENTN	INPUT	WORD	I, Q, M, D, L, constant	Event-ID. The user assigns the event-ID. This is not preset by the message server.
INFO1	INPUT	ANY	I, Q, M, D, L	Additional information, length 1 word
INFO2	INPUT	ANY	I, Q, M, D, L	Additional information, length 2 words
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed.

SEND

When SEND is set to 1 the user defined diagnostic message is sent to all partners that have registered for this purpose. Sending is only initiated when one or more partners have registered and the send buffer is not full. Messages are sent asynchronously with respect to the application program.

EVENTN

The event ID of the user event is entered into *EVENTN*. Event IDs must be of the format 8xyzh , 9xyzh, Axyzh and Bxyzh. Here the IDs of format 8xyzh and 9xyzh refer to predefined events and IDs of format Axyzh and Bxyzh refer to user-defined events.
An event being activated is indicated by x = 1, an event being deactivated by x = 0.
For events of the class A and B, yz refers to the message number that was predefined in hexadecimal representation when the messages were configured.

INFO1

INFO1 contains information with a length of one word. The following data types are valid:

- WORD
- INT
- ARRAY [0...1] OF CHAR

INFO1 can be integrated as associated value into the message text, i.e. to add current information to the message.

INFO2

INFO2 contains information with a length of two words. The following data types are valid:

- DWORD
- DINT
- REAL
- TIME
- ARRAY [0...3] OF CHAR

INFO2 can be integrated as associated value into the message text, i.e. to add current information to the message.

RET_VAL (Return value)

The return value contains an error code if an error is detected when the function is being processed.

Value	Description
0000h	no error
0091h	No partner registered (the diagnostic event has been entered into the diagnostic buffer)
8083h	Data type *INFO1* not valid
8084h	Data type *INFO2* not valid
8085h	*EVENTN* not valid
8086h	Length of *INFO1* not valid
8087h	Length of *INFO2* not valid
8091h	Error message identical to error code 0091h
8092h	Send operation currently not possible, send buffer full (the diagnostic event has been entered into the diagnostic buffer)

SFC 54 - RD_DPARM - Read predefined parameter

Description

The SFC 54 RD_DPARM (read defined parameter) reads the record with number *RECNUM* of the selected module from the respective SDB1xy.

Parameter *RECORD* defines the target area where the record will be saved

Parameters

Parameter	Declaration	Data type	Memory block	Description
IOID	INPUT	BYTE	I, Q, M, D, L, constant	Identifier for the address space: 54h = peripheral input (PI) 55h = peripheral output (PQ) For hybrid modules the SFC returns the area identifier of the lower address. When the addresses are equal the SFC returns identifier 54h.
LADDR	INPUT	WORD	I, Q, M, D, L, constant	Logical address. For hybrid modules the lower of the two addresses must be specified.
RECNUM	INPUT	BYTE	I, Q, M, D, L, constant	record number (valid range: 0 ... 240)
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed. Additionally: the length of the record that was read in bytes, provided the size of the record fits into the target area and that no communication errors have occurred.
RECORD	OUTPUT	ANY	I, Q, M, D, L	Target area for the record that was read. Only data type BYTE is valid.

RET_VAL (Return value)

Two distinct cases exist for *RET_VAL* = 8xxxh:

- Temporary error (error codes 80A2h ... 80A4h, 80Cxh):
  For this type of error it is possible that the error corrects itself without intervention. For this reason it is recommended that you re-issue the call to the SFC (once or more than once).
  Example for temporary errors: the required resources are occupied at present (80C3h).
- Permanent error (error codes 809xh, 80A1h, 80Bxh, 80Dxh):
  These errors cannot be corrected without intervention. A repeat of the call to the SFC is only meaningful when the error has been removed.
  Example for permanent errors: incorrect length of the record that must be transferred (80B1h).

Value	Description
7000h	First call with *REQ* = 0: data transfer not active; *BUSY* is set to 0.
7001h	First call with *REQ* = 1: data transfer initiated; *BUSY* is set to 1.
7002h	Intermediate call (*REQ* irrelevant): data transfer active; *BUSY* is set to 1.
8090h	The specified logical base address is invalid: no assignment available in SDB1/SDB2x, or this is not a base address.
8092h	ANY-reference contains a type definition that is not equal to BYTE.
8093h	This SFC is not valid for the module selected by *LADDR* and *IOID*.
80B1h	The length of the target area defined by *RECORD* is too small.
80D0h	The respective SDB does not contain an entry for the module.
80D1h	The record number has not been configured in the respective SDB for the module.
80D2h	According to the type identifier the module cannot be configured.
80D3h	SDB cannot be accessed since it does not exist.
80D4h	Bad SDB structure: the SDB internal pointer points to an element outside of the SDB.

SFC 55 - WR_PARM - Write dynamic parameter

Description

The SFC 55 WR_PARM (write parameter) transfers the record *RECORD* to the target module. Any parameters for this module that exist in the respective SDB will not be replaced by the parameters that are being transferred to the module.

These SFC can be used for digital-, analog modules, FMs, CPs and via PROFIBUS DP-V1.

Conditions

It is important that the record that must be transferred is not static, i.e.:

- do not use record 0 since this record is static for the entire system.
- if the record appears in SDBs 100 ... 129 then the static-bit must not be set.

Parameter

Parameter	Declaration	Data type	Memory block	Description
REQ	INPUT	BOOL	I, Q, M, D, L, constant	*REQ* = 1: write request
IOID	INPUT	BYTE	I, Q, M, D, L, constant	Identifier for the address space: 54h = peripheral input (PI) 55h = peripheral output (PQ) For hybrid modules the SFC returns the area identifier of the lower address. When the addresses are equal the SFC returns identifier 54h.
LADDR	INPUT	WORD	I, Q, M, D, L, constant	Logical base address of the module. For hybrid modules the lower of the two addresses must be specified.
RECNUM	INPUT	BYTE	I, Q, M, D, L, constant	record number (valid values: 0 ... 240)
RECORD	INPUT	ANY	I, Q, M, D, L	Record
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed.
BUSY	OUTPUT	BOOL	I, Q, M, D, L	*BUSY* = 1: the write operation has not been completed.

RECORD

With the first call to the SFC the data that must be transferred is read from the parameter *RECORD*. However, if the transfer of the record should require more than one call duration, the contents of the parameter *RECORD* is no longer valid for subsequent calls to the SFC (of the same job).

RET_VAL (Return value)

Two distinct cases exist for *RET_VAL* = 8xxxh:

- Temporary error (error codes 80A2h ... 80A4h, 80Cxh):
  For this type of error it is possible that the error corrects itself without intervention. For this reason it is recommended that you re-issue the call to the SFC (once or more than once).
  Example for temporary errors: the required resources are occupied at present (80C3h).
- Permanent error (error codes 809xh, 80A1h, 80Bxh, 80Dxh):
  These errors cannot be corrected without intervention. A repeat of the call to the SFC is only meaningful when the error has been removed.
  Example for permanent errors: incorrect length of the record that must be transferred (80B1h).

Value	Description
7000h	First call with *REQ* = 0: data transfer not active; *BUSY* is set to 0.
7001h	First call with *REQ* = 1: data transfer initiated; *BUSY* is set to 1.
7002h	Intermediate call (*REQ* irrelevant): data transfer active; *BUSY* is set to 1.
8090h	The specified logical base address is invalid: no assignment available in SDB1/SDB2x, or this is not a base address.
8092h	ANY-reference contains a type definition that is not equal to BYTE.
8093h	This SFC is not valid for the module selected by *LADDR* and *IOID*.
80A1h	Negative acknowledgement when the record is being transferred to the module (module was removed during the transfer or module failed)
80A2h	DP protocol fault in layer 2, possible hardware-/ interface fault in the DP slave
80A3h	DP protocol fault for user Interface/user
80A4h	Communication failure (this fault occurs between the CPU and the external DP interface)
80B0h	SFC cannot be used with this type of module or the module does not recognize the record.

continued ...

... continue

Value	Description
80B1h	The length of the target area determined by *RECORD* is too small.
80B2h	The slot that was configured has not been populated.
80B3h	The actual type of module is not equal to the required type of module in SDB1
80C1h	The module has not yet completed processing of the data of the preceding write operation for the same record.
80C2h	The module is currently processing the maximum number of jobs for a CPU.
80C3h	Required resources (memory, etc.) are currently occupied.
80C4h	Communication error:
80C5h	Decentralized periphery not available.
80C6h	The transfer of records was aborted due to a priority class abort.
80D0h	The respective SDB does not contain an entry for the module.
80D1h	The record number was not configured in the respective SDB.
80D2h	Based on the type identifier the module cannot be configured.
80D3h	The SDB cannot be accessed since it does not exist.
80D4h	Bad SDB structure: the SDB internal pointer points to an element outside of the SDB.
80D5h	The record is not static.

SFC 56 - WR_DPARM - Write default parameter

Description

The SFC 56 WR_DPARM (write default parameter) transfers the record *RECNUM* from the respective SDB to the target module, irrespective of whether the specific record is static or dynamic.

These SFC can be used for digital-, analog modules, FMs, CPs and via PROFIBUS DP-V1.

Parameters

Parameter	Declaration	Data type	Memory block	Description
REQ	INPUT	BOOL	I, Q, M, D, L, constant	*REQ* = 1: write request
IOID	INPUT	BYTE	I, Q, M, D, L, constant	Identifier for the address space: 54h = peripheral input (PI) 55h = peripheral output (PQ) For hybrid modules the SFC returns the area identifier of the lower address. When the addresses are equal the SFC returns identifier 54h.
LADDR	INPUT	WORD	I, Q, M, D, L, constant	Logical base address of the module. For hybrid modules the lower of the two addresses must be specified.
RECNUM	INPUT	BYTE	I, Q, M, D, L, constant	Record number (valid values: 0 ... 240)
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed.
BUSY	OUTPUT	BOOL	I, Q, M, D, L	*BUSY* = 1: the write operation has not been completed.

RET_VAL (Return value)

Two distinct cases exist for *RET_VAL* = 8xxxh:

- Temporary error (error codes 80A2h ... 80A4h, 80Cxh):
  For this type of error it is possible that the error corrects itself without intervention. For this reason it is recommended that you re-issue the call to the SFC (once or more than once).
  Example for temporary errors: the required resources are occupied at present (80C3h).
- Permanent error (error codes 809xh, 80A1h, 80Bxh, 80Dxh):
  These errors cannot be corrected without intervention. A repeat of the call to the SFC is only meaningful when the error has been removed.
  Example for permanent errors: incorrect length of the record that must be transferred (80B1h).

Value	Description
7000h	First call with *REQ* = 0: data transfer not active; *BUSY* is set to 0.
7001h	First call with *REQ* = 1: data transfer initiated; *BUSY* is set to 1.
7002h	Intermediate call (*REQ* irrelevant): data transfer active; *BUSY* is set to 1.
8090h	The specified logical base address is invalid: no assignment available in SDB1/SDB2x, or this is not a base address.
8093h	This SFC is not valid for the module selected by means of *LADDR* and *IOID*.
80A1h	Negative acknowledgement when the record is being transferred to the module (module was removed during the transfer or module failed)
80A2h	DP protocol fault in layer 2, possible hardware-/ interface fault in the DP slave
80A3h	DP protocol fault for user Interface/user
80A4h	Communication failure (this fault occurs between the CPU and the external DP interface).
80B0h	SFC cannot be used with this type of module or the module does not recognize the record.
80B1h	The length of the target area determined by *RECORD* is too small.
80B2h	The slot that was configured has not been populated.
80B3h	The actual type of module is not equal to the required type of module in SDB1.
80C1h	The module has not yet completed processing of the data of the preceding write operation for the same record.
80C2h	The module is currently processing the maximum number of jobs for a CPU.
80C3h	Required resources (memory, etc.) are currently occupied.
80C4h	Communication error
80C5h	Decentralized periphery not available.
80C6h	The transfer of records was aborted due to a priority class abort.
80D0h	The respective SDB does not contain an entry for the module.
80D1h	The record number was not configured in the respective SDB.
80D2h	Based on the type identifier the module cannot be configured.
80D3h	The SDB cannot be accessed since it does not exist.
80D4h	Bad SDB structure: the SDB internal pointer points to an element outside of the SDB.

SFC 57 - PARM_MOD - Parameterize module

Description

The SFC 57 PARM_MOD (parameterize module) transfers all the records that were configured in the respective SDB into a module, irrespective of whether the specific record is static or dynamic.

These SFC can be used for digital-, analog modules, FMs, CPs and via PROFIBUS DP-V1.

Parameters

Parameter	Declaration	Data type	Memory block	Description
REQ	INPUT	BOOL	I, Q, M, D, L, constant	*REQ* = 1: write request
IOID	INPUT	BYTE	I, Q, M, D, L, constant	Identifier for the address space: 54h = peripheral input (PI) 55h = peripheral output (PQ) For hybrid modules the SFC returns the area identifier of the lower address. When the addresses are equal the SFC returns identifier 54h.
LADDR	INPUT	WORD	I, Q, M, D, L, constant	Logical base address of the module. For hybrid modules the lower of the two addresses must be specified.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed.
BUSY	OUTPUT	BOOL	I, Q, M, D, L	*BUSY* = 1: the write operation has not been completed.

RET_VAL (Return value)

Two distinct cases exist for *RET_VAL* = 8xxxh:

- Temporary error (error codes 80A2h ... 80A4h, 80Cxh):
  For this type of error it is possible that the error corrects itself without intervention. For this reason it is recommended that you re-issue the call to the SFC (once or more than once).
  Example for temporary errors: the required resources are occupied at present (80C3h).
- Permanent error (error codes 809xh, 80A1h, 80Bxh, 80Dxh):
  These errors cannot be corrected without intervention. A repeat of the call to the SFC is only meaningful when the error has been removed.
  Example for permanent errors: incorrect length of the record that must be transferred (80B1h).

Value	Description
7000h	First call with *REQ* = 0: data transfer not active; *BUSY* is set to 0.
7001h	First call with *REQ* = 1: data transfer initiated; *BUSY* is set to 1.
7002h	Intermediate call (*REQ* irrelevant): data transfer active; *BUSY* is set to 1.
8090h	The specified logical base address is invalid: no assignment available in SDB1/SDB2x, or this is not a base address.
8093h	This SFC is not valid for the module selected by means of *LADDR* and *IOID*.
80A1h	Negative acknowledgement when the record is being transferred to the module (module was removed during the transfer or module failed)
80A2h	DP protocol fault in layer 2, possible hardware-/ interface fault in the DP slave
80A3h	DP protocol fault for user Interface/user
80A4h	Communication failure (this fault occurs between the CPU and the external DP interface)
80B0h	SFC cannot be used with this type of module or the module does not recognize the record.
80B1h	The length of the target area determined by *RECORD* is too small.
80B2h	The slot that was configured has not been populated.
80B3h	The actual type of module is not equal to the required type of module in SDB1
80C1h	The module has not yet completed processing of the data of the preceding write operation for the same record.
80C2h	The module is currently processing the maximum number of jobs for a CPU.
80C3h	Required resources (memory, etc.) are currently occupied.
80C4h	Communication error
80C5h	Decentralized periphery not available.
80C6h	The transfer of records was aborted due to a priority class abort.
80D0h	The respective SDB does not contain an entry for the module.
80D1h	The record number was not configured in the respective SDB.
80D2h	Based on the type identifier the module cannot be configured.
80D3h	The SDB cannot be accessed since it does not exist.
80D4h	Bad SDB structure: the SDB internal pointer points to an element outside of the SDB.

SFC 58 - WR_REC - Write record

Description

The SFC 58 WR_REC (write record) transfers the record *RECORD* into the selected module.
The write operation is started when input parameter *REQ* is set to 1 when the call to the SFC 58 is issued. Output parameter *BUSY* returns a value of 0 if the write operation was executed immediately. *BUSY* is set to 1 if the write operation could not be completed.

These SFC can be used for digital-, analog modules, FMs, CPs and via PROFIBUS DP-V1.

System dependent this block cannot be interrupted!

Parameter

Parameter	Declaration	Data type	Memory block	Description
REQ	INPUT	BOOL	I, Q, M, D, L, constant	*REQ* = 1: write request
IOID	INPUT	BYTE	I, Q, M, D, L, constant	Identifier for the address space: 54h = peripheral input (PI) 55h = peripheral output (PQ) For hybrid modules the SFC returns the area identifier of the lower address. When the addresses are equal the SFC returns identifier 54h.
LADDR	INPUT	WORD	I, Q, M, D, L, constant	Logical base address of the module. For hybrid modules the lower of the two addresses must be specified.
RECNUM	INPUT	BYTE	I, Q, M, D, L, constant	Record number (valid range: 2 ... 240)
RECORD	INPUT	ANY	I, Q, M, D, L	Record. Only data type BYTE is valid.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed.
BUSY	OUTPUT	BOOL	I, Q, M, D, L	*BUSY* = 1: the write operation has not been completed.

RECORD

With the first call to the SFC the data that must be transferred is read from the parameter *RECORD*. However, if the transfer of the record should require more than one call duration, the contents of the parameter *RECORD* is no longer valid for subsequent calls to the SFC (of the same job).

RET_VAL (Return value)

Two distinct cases exist for *RET_VAL* = 8xxxh:

- Temporary error (error codes 80A2h ... 80A4h, 80Cxh):
  For this type of error it is possible that the error corrects itself without intervention. For this reason it is recommended that you re-issue the call to the SFC (once or more than once).
  Example for temporary errors: the required resources are occupied at present (80C3h).
- Permanent error (error codes 809xh, 80A0, 80A1h, 80Bxh):
  These errors cannot be corrected without intervention. A repeat of the call to the SFC is only meaningful when the error has been removed.
  Example for permanent errors: incorrect length of the record that must be transferred (80B1h).

Value	Description
7000h	First call with *REQ* = 0: data transfer not active; *BUSY* is set to 0.
7001h	First call with *REQ* = 1: data transfer initiated; *BUSY* is set to 1.
7002h	Intermediate call (*REQ* irrelevant): data transfer active; *BUSY* is set to 1.
8090h	The specified logical base address is invalid: no assignment available in SDB1/SDB2x, or this is not a base address.
8092h	ANY-reference contains a type definition that is not equal to BYTE.
8093h	This SFC is not valid for the module selected by *LADDR* and *IOID*.
80A1h	Negative acknowledgement when the record is being transferred to the module (module was removed during the transfer or module failed)
80A2h	DP protocol fault in layer 2, possible hardware-/ interface fault in the DP slave
80A3h	DP protocol fault for user Interface/user
80A4h	Communication failure (this fault occurs between the CPU and the external DP interface)

continued ...

... continue

Value	Description
80B0h	• SFC not valid for the type of module. • Module does not recognize the record. • Record number ≥ 241 not permitted. • Records 0 and 1 not permitted.
80B1h	The length specified in parameter *RECORD* is wrong.
80B2h	The slot that was configured has not been populated.
80B3h	The actual type of module is not equal to the required type of module in SDB1
80C1h	The module has not yet completed processing of the data of the preceding write operation for the same record.
80C2h	The module is currently processing the maximum number of jobs for a CPU.
80C3h	Required resources (memory, etc.) are currently occupied.
80C4h	Communication error
80C5h	Decentralized periphery not available.
80C6h	The transfer of records was aborted due to a priority class abort.

Note!

A general error 8544h only indicates that access to at least one byte of I/O memory containing the record was disabled. However, the data transfer was continued.

SFC 59 - RD_REC - Read record

Description

The SFC 59 RD_REC (read record) reads the record with the number *RECNUM* from the selected module.

These SFC can be used for digital-, analog modules, FMs, CPs and via PROFIBUS DP-V1.

The read operation is started when input parameter *REQ* is set to 1 when the call to SFC 59 is issued. Output parameter *BUSY* returns a value of 0 if the read operation was executed immediately. *BUSY* is set to 1 if the read operation could not be completed. Parameter *RECORD* determines the target area where the record is saved when it has been transferred successfully.

System dependent this block cannot be interrupted!

Parameter

Parameter	Declaration	Data type	Memory block	Description
REQ	INPUT	BOOL	I, Q, M, D, L, constant	*REQ* = 1: read request
IOID	INPUT	BYTE	I, Q, M, D, L, constant	Identifier for the address space: 54h = peripheral input (PI) 55h = peripheral output (PQ) For hybrid modules the SFC returns the area identifier of the lower address. When the addresses are equal the SFC returns identifier 54h.
LADDR	INPUT	WORD	I, Q, M, D, L, constant	Logical base address of the module. For hybrid modules the lower of the two addresses must be specified.
RECNUM	INPUT	BYTE	I, Q, M, D, L, constant	Record number (valid range: 0 ... 240)
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed. Additionally: the length of the actual record that was read, in bytes (range: +1 ... +240), provided that the target area is greater than the transferred record and that no communication errors have occurred.
BUSY	OUTPUT	BOOL	I, Q, M, D, L	*BUSY* = 1: the write operation has not been completed.

continued ...

... continue

Parameter	Declaration	Data type	Memory block	Description
RECORD	OUTPUT	ANY	I, Q, M, D, L	Target area for the record that was read. When SFC 59 is processed in asynchronous mode you must ensure that the actual parameters of *RECORD* have the same length information for all calls. Only data type BYTE is permitted.

Suitable choice of RECORD

To ensure that an entire record is read you must select a target area with a length of 241bytes. In this case the value in *RET_VAL* indicates the actual length of the data that was transferred successfully.

RET_VAL (Return value)

RET_VAL contains an error code when an error occurs while the function was being processed.
When the transfer was successful *RET_VAL* contains:

- a value of 0 if the entire target area was filled with data from the selected record (the record may, however, be incomplete).
- the length of the record that was transferred, in bytes (valid range: 1 ... 240), provided that the target area is greater than the transferred record.

Error information

Two distinct cases exist for *RET_VAL* = 8xxxh:

- Temporary error (error codes 80A2h ... 80A4h, 80Cxh):
  For this type of error it is possible that the error corrects itself without intervention. For this reason it is recommended that you re-issue the call to the SFC (once or more than once).
  Example for temporary errors: the required resources are occupied at present (80C3h).
- Permanent error (error codes 809xh, 80A0h, 80A1h, 80Bxh):
  These errors cannot be corrected without intervention. A repeat of the call to the SFC is only meaningful when the error has been removed.
  Example for permanent errors: incorrect length of the record that must be transferred (80B1h).

Error information

Value	Description
7000h	First call with *REQ* = 0: data transfer not active; *BUSY* is set to 0.
7001h	First call with *REQ* = 1: data transfer initiated; *BUSY* is set to 1.
7002h	Intermediate call (*REQ* irrelevant): data transfer active; *BUSY* is set to 1.
8090h	The specified logical base address is invalid: no assignment available in SDB1/SDB2x, or this is not a base address.
8092h	ANY-reference contains a type definition that is not equal to BYTE.
8093h	This SFC is not valid for the module selected by *LADDR* and *IOID*.
80A0h	Negative acknowledgement when reading from the module (module was removed during the transfer or module failed)
80A2h	DP protocol fault in layer 2, possible hardware-/ interface fault in the DP slave
80A3h	DP protocol fault for user Interface/user
80A4h	Communication failure (this fault occurs between the CPU and the external DP interface)
80B0h	• SFC not valid for the type of module. • Module does not recognize the record. • Record number ≥ 241 not permitted.
80B1h	The length specified in parameter *RECORD* is wrong.
80B2h	The slot that was configured has not been populated.
80B3h	The actual type of module is not equal to the required type of module in SDB1
80C0h	The module has registered the record but this does not contain any read data as yet.
80C1h	The module has not yet completed processing of the data of the preceding write operation for the same record.
80C2h	The module is currently processing the maximum number of jobs for a CPU.
80C3h	Required resources (memory, etc.) are currently occupied.
80C4h	Communication error
80C5h	Decentralized periphery not available.
80C6h	The transfer of records was aborted due to a priority class abort.

Note!

A general error 8745h only indicates that access to at least one byte of I/O memory containing the record was disabled. However, the data was read successfully from the module and saved to the I/O memory block.

SFC 64 - TIME_TCK - Read system time tick

Description

The SFC 64 TIME_TCK (time tick) retrieves the system time tick from the CPU. This ma be used to assess the time that certain processes require calculating the difference between the values returned by two SFC 64 calls. The system time is a "time counter" that counts from 0 to a max. of 2147483647ms and that restarts from 0 when an overflow occurs. The timing intervals and the accuracy of the system time depend on the CPU. Only the operating modes of the CPU influence the system time.

System time and operating modes

Operating mode	System time ...
Restart RUN	... permanently updated.
STOP	... stopped to retain the last value.
reboot	... is deleted and starts from "0".

Parameters

Parameter	Declaration	Data type	Memory block	Description
RET_VAL	OUTPUT	TIME	I, Q, M, D, L	Parameter *RET_VAL* contains the system time that was retrieved, range from 0 ... 2^{31} -1ms.

RET_VAL (Return value)

The SFC 64 does not return any error information.

SFC 65 - X_SEND - Send data

Description

The SFC 65 X_SEND can be used to send data to an external communication partner outside the local station. The communication partner receives the data by means of the SFC 66 X_RCV. Input parameter *REQ_ID* is used to identify the transmit data. This code is transferred along with the transmit data and it can be analyzed by the communication partner to determine the origin of the data. The transfer is started when input parameter *REQ* is set to 1. The size of the transmit buffer that is defined by parameter *SD* (on the sending CPU) must be less than or equal to the size of the receive buffer (on the communication partner) that was defined by means of parameter *RD*. In addition, the data type of the transmit buffer and the receive buffer must be identical.

Parameters

Parameter	Declaration	Data type	Memory block	Description
REQ	INPUT	BOOL	I, Q, M, D, L, constant	control parameter "request to activate", initiates the operation
CONT	INPUT	BOOL	I, Q, M, D, L, constant	control parameter "continue", defines whether the connection to the communication partner is terminated or not when the operation has been completed
DEST_ID	INPUT	WORD	I, Q, M, D, L, constant	Address parameter "destination ID". Contains the MPI-address of the communication partners.
REQ_ID	INPUT	DWORD	I, Q, M, D, L, constant	Operation code identifying the data on the communication partner.
SD	INPUT	ANY	I, Q, M, D	Reference to the send buffer. The following data types are possible: BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5_TIME, DATE_AND_TIME as well as arrays of the respective data types, with the exception of BOOL.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed.
BUSY	OUTPUT	BOOL	I, Q, M, D, L	*BUSY* = 1: the send operation has not yet been completed. *BUSY* = 0: the send operation has been completed, or no send operation is active.

REQ_ID

Input parameter *REQ_ID* identifies the send data.
Parameter *REQ_ID* is required by the receiver when

- the sending CPU issues multiple calls to SFC 65 with different *REQ_ID* parameters and the data is transferred to a single communication partner.
- more than one sending CPU are transferring data to a communication partner by means of the SFC 65.

Receive data can be saved into different memory blocks by analyzing the *REQ_ID* parameter.

Data consistency

Since send data is copied into an internal buffer of the operating system when the first call is issued to the SFC it is important to ensure that the send buffer is not modified before the first call has been completed successfully. Otherwise an inconsistency could occur in the transferred data.
Any write-access to send data that occurs after the first call is issued does not affect the data consistency.

RET_VAL (Return value)

The return value contains an error code if an error is detected when the function is being processed.

The "real error information" that is contained in the table "specific error information" a. o. may be classified as follows:

Value	Description
809xh	Error on the CPU where the SFC is being executed
80Axh	Permanent communication error.
80Bxh	Error on the communication partner.
80Cxh	Temporary error.

Specific error information:

Value	Description
0000h	Processing completed without errors.
7000h	First call with *REQ* = 0: no data transfer is active; *BUSY* is set to 0.
7001h	First call with *REQ* = 1: data transfer initiated; *BUSY* is set to 1.
7002h	Intermediate call (*REQ* irrelevant): data transfer active; *BUSY* is set to 1.
8090h	The specified target address of the communication partners is not valid, e.g. • bad *IOID* • bad base address exists • bad MPI-address (> 126)
8092h	Error in *SD* or *RD*, e.g.: • illegal length for *SD* • *SD* = NIL is not permitted
8095h	The block is already being processed on a priority class that has a lower priority.
80A0h	Error in received acknowledgement.
80A1h	Communication failures: SFC-call after an existing connection has been terminated.
80B1h	ANY-pointer error. The length of the data buffer that must be transferred is wrong.
80B4h	ANY-pointer data type error, or ARRAY of the specified data type is not permitted.
80B5h	Processing rejected because of an illegal operating mode.
80B6h	The received acknowledgement contains an unknown error code.
80B8h	The SFC 66 "X_RCV" of the communication partner rejected the data transfer (*RD* = NIL).
80B9h	The data block was identified by the communication partner (SFC 66 "X_RCV" was called with *EN_DT* = 0) but it has not yet been accepted into the application program because the operating mode is STOP.
80BAh	The answer of the communication partner does not fit into the communication telegram.
80C0h	The specified connection is already occupied by another operation.
80C1h	Lack of resources on the CPU where the SFC is being executed, e.g.: • The module is already executing the maximum number of different send operations. • Connection resources may be occupied, e.g. by a receive operation.
80C2h	Temporary lack of resources for the communication partner, e.g.: • The communication partner is currently processing the maximum number of operations. • The required resources (memory, etc.) are already occupied. • Not enough memory (initiate compression.)
80C3h	Error when establishing a connection, e.g.: • The local station is connected to the MPI sub-net. • You have addressed the local station on the MPI sub-net. • The communication partner cannot be contacted any longer • Temporary lack of resources for the communication partner.

SFC 66 - X_RCV - Receive data

Description

The SFC 66 X_RCVS can be used to receive data, that was sent by means of SFC 65 X_SEND by one or more external communication partners.
SFC 66 can determine whether the data that was sent is available at the current point in time. The operating system could have stored the respective data in an internal queue. If the data exists in the queue the oldest data block can be copied into the specified receive buffer.

Parameters

Parameter	Declaration	Data type	Memory block	Description
EN_DT	INPUT	BOOL	I, Q, M, D, L, constant	Control parameter "enable data transfer". You can check whether one or more data blocks are available by setting this to 0. A value of 1 results in the oldest data block of the queue being copied into the memory block that was specified by means of *RD*.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed.
REQ_ID	OUTPUT	DWORD	I, Q, M, D, L	Operation code of the SFC 65 "X_SEND" whose send data is located uppermost in the queue, i.e. the oldest data in the queue. If the queue does not contain a data block *REQ_ID* is set to 0.
NDA	OUTPUT	BOOL	I, Q, M, D, L	Status parameter "new data arrived". *NDA* = 0: • The queue does not contain a data block. *NDA* = 1: • The queue does contain one or more data blocks. (call to the SFC 66 with *EN_DT* = 0). • The oldest data block in the queue was copied into the application program. (call to the SFC 66 with *EN_DT* = 1).

continued ...

... continue

Parameter	Declaration	Data type	Memory block	Description
RD	OUTPUT	ANY	I, Q, M, D	Reference to the receive data buffer (receive data area). The following data types are available: BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5_TIME, DATE_AND_TIME as well as arrays of these data types with the exception of BOOL. If you wish to discard the oldest data block in the queue you must assign a value of NIL to *RD*.

Data reception indication

with *EN_DT* = 0

The operating system inserts data received from a communication partner in the sequence in which they are received.
You can test whether at least one data block is ready by issuing a call to the SFC 66 with *EN_DT* = 0 and testing the resulting output parameter *NDA*.

- *NDA* = 0 means that the queue does not contain a data block. *REQ_ID* is irrelevant, *RET_VAL* contains a value of 7000h.
- *NDA* = 1 means that the queue does contain one or more data blocks.

If the queue contains a data block you should also test output parameters *RET_VAL* and *REQ_ID*. *RET_VAL* contains the length of the data block in bytes, *REQ_ID* contains the operation code of the send block. If the queue should contain multiple data blocks parameters *REQ_ID* and *RET_VAL* refer to the oldest data block contained in the queue.

Transferring data into the receive buffer

with *EN_DT* = 1

When input parameter *EN_DT* = 1 then the oldest data block in the queue is copied into the target block defined by *RD*. You must ensure that the size of *RD* is greater than or equal to the size of the transmit buffer of the respective SFC 65 X_SEND defined by parameter *SD* and that that the data types match. If received data should be saved into different areas you can determine the *REQ_ID* in the first call (SFC-call with *EN_DT* = 0) and select a suitable value for *RD* in the subsequent call (with *EN_DT* = 1). If the operation was processed successfully *RET_VAL* contains the length (in bytes) of data block that was copied and a positive acknowledgement is returned to the sending station.

Discarding data

If you do not want to accept the received data assign a value of NIL to *RD*. The respective communication partner receives a negative acknowledgement (the value of *RET_VAL* of the respective SFC 65 X_SEND is 80B8h) and parameter *RET_VAL* is set to 0.

Data consistency

You must make sure that the receive buffer is not read before the operation has been completed since you could otherwise be reading could cause inconsistent data.

Operating mode transition to STOP mode

When the CPU changes to STOP mode,

- all newly received commands receive a negative acknowledgement.
- for commands that have already been received: all commands that have been entered into the in receive queue receive a negative acknowledgement.
- all data blocks are discarded when a new start follows.

Termination of a connection

When the connection is terminated any operation that was entered into the receive queue of this connection is discarded.
Exception: if this is the oldest operation in the queue that has already been recognized by a SFC-call with *EN_DT* = 0 it can be transferred into the receive buffer by means of *EN_DT* = 1.

RET_VAL (Return value)

If no error has occurred, *RET_VAL* contains:

- when *EN_DT* = 0/1 and *NDA* = 0: 7000h. In this case the queue does not contain a data block.
- when *EN_DT* = 0 and *NDA* = 1, *RET_VAL* contains the length (in bytes) of the oldest data block that was entered into the queue as a positive number.
- when *EN_DT* = 1 and *NDA* = 1, *RET_VAL* contains the length (in bytes) of the data block that was copied into the receive buffer *RD* as a positive number.

Error information

The "real error information" that is contained in the table "specific error information" a. o. may be classified as follows:

Value	Description
809xh	Error on the CPU where the SFC is being executed
80Axh	Permanent communication error
80Bxh	Error on the communication partner
80Cxh	Temporary error

Specific Error information:

Value	Description
0000h	Processing completed without errors.
00xyh	When *NDA* = 1 and *RD* <> NIL: *RET_VAL* contains the length of the received data block (when *EN_DT* = 0) or the data block copied into *RD* (when *EN_DT* = 1).
7000h	*EN_DT* = 0/1 and *NDA* = 0
7001h	First call with *REQ* = 1: data transfer initiated; *BUSY* is set to 1.
7002h	Intermediate call (*REQ* irrelevant): data transfer active; *BUSY* is set to 1.
8090h	The specified target address of the communication partners is not valid, e.g. • bad *IOID* • bad base address exists • bad MPI-address (> 126)
8092h	Error in *SD* or *RD*, e.g.: • The amount of data received is too much for the buffer defined by *RD*. • *RD* has data type BOOL but the length of the received data is greater than one byte.
8095h	The block is already being processed on a priority class that has a lower priority.
80A0h	Error in received acknowledgement.
80A1h	Communication failures: SFC-call after an existing connection has been terminated.
80B1h	ANY-pointer error. The length of the data block that must be transferred is wrong.
80B4h	ANY-pointer data type error, or ARRAY of the specified data type is not permitted.
80B6h	The received acknowledgement contains an unknown error code.
80BAh	The answer of the communication partner does not fit into the communication telegram.
80C0h	The specified connection is already occupied by another operation.
80C1h	Lack of resources on the CPU where the SFC is being executed, e.g.: • the module is already executing the maximum number of different send operations. • connection resources may be occupied, e.g. by a receive operation.
80C2h	Temporary lack of resources for the communication partner, e.g.: • The communication partner is currently processing the maximum number of operations. • The required resources (memory, etc.) are already occupied. • Not enough memory (initiate compression).
80C3h	Error when establishing a connection, e.g.: • The local station is connected to the MPI sub-net. • You have addressed the local station on the MPI sub-net. • The communication partner cannot be contacted any longer. • Temporary lack of resources for the communication partner.

SFC 67 - X_GET - Read data

Description

The SFC 67 X_GET can be used to read data from an external communication partner that is located outside the local station. No relevant SFC exists on the communication partner. The operation is started when input parameter *REQ* is set to 1. Thereafter the call to the SFC 67 is repeated until the value of output parameter *BUSY* becomes 0. Output parameter *RET_VAL* contains the length of the received data block in bytes.
The length of the receive buffer defined by parameter *RD* (in the receiving CPU) must be identical or greater than the read buffer defined by parameter *VAR_ADDR* (for the communication partner) and the data types of *RD* and *VAR_ADDR* must be identical.

Parameters

Parameter	Declaration	Data type	Memory block	Description
REQ	INPUT	BOOL	I, Q, M, D, L, constant	Control parameter "request to activate", used to initiate the operation
CONT	INPUT	BOOL	I, Q, M, D, L, constant	Control parameter "continue", determines whether the connection to the communication partner is terminated or not when the operation has been completed
DEST_ID	INPUT	WORD	I, Q, M, D, L, constant	Address parameter "destination ID". Contains the MPI address of the communication partner.
VAR_ADDR	INPUT	ANY	I, Q, M, D	Reference to the buffer in the partner-CPU from where data must be read. You must select a data type that is supported by the communication partner.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed. If no error has occurred, *RET_VAL* contains the length of the data block that was copied into receive buffer *RD* as positive number of bytes.
BUSY	OUTPUT	BOOL	I, Q, M, D, L	*BUSY* = 1: the receive operation has not been completed. *BUSY* = 0: The receive operation has been completed or no receive operation active.
RD	OUTPUT	ANY	I, Q, M, D	Reference to the receive buffer (receive data area). The following data types are permitted: BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5_TIME, DATE_AND_TIME as well as arrays of the above data types, with the exception of BOOL

Data consistency

The following rules must be satisfied to prevent the data consistency from being compromised:

- Active CPU (receiver of data):
  The receive buffer should be read in the OB that issues the call to the respective SFC. If this is not possible the receive buffer should only be read when processing of the respective SFC has been completed.
- Passive CPU (sender of data):
  The maximum amount of data that may be written into the send buffer is determined by the block size of the passive CPU (sender of data).
- Passive CPU (sender of data):
  Send data should be written to the send buffer while interrupts are inhibited.

Operating mode transition to STOP mode

When the CPU changes to STOP mode the connection established by means of the SFC 67 is terminated. The type of start-up that follows determines whether any previously received data located in a buffer of the operating system are discarded or not.

A reboot start means that the data is discarded.

Operating mode transition of the communication partners to STOP mode

A transition to operating mode STOP of the CPU of the communication partner does not affect the data transfer, since it is also possible to read data in operating mode STOP.

RET_VAL (Return value) The "real error information" that is contained in the table "specific error information" a. o. may be classified as follows:

Value	Description
809xh	Error on the CPU where the SFC is being executed
80Axh	Permanent communication error
80Bxh	Error on the communication partner
80Cxh	Temporary error

Specific error information:

Value	Description
0000h	Processing completed without errors.
00xyh	*RET_VAL* contains the length of the received data block.
7000h	Call issued with *REQ* = 0 (call without processing), *BUSY* is set to 0, no data transfer is active.
7001h	First call with *REQ* = 1: data transfer initiated; *BUSY* is set to 1.
7002h	Intermediate call (*REQ* irrelevant): data transfer active; *BUSY* is set to 1.
8090h	The specified target address of the communication partners is not valid, e.g. • bad *IOID* • bad base address exists • bad MPI-address (> 126)
8092h	Error in *SD* or *RD*, e.g.: • illegal length for *RD* • the length or the data type of *RD* does not correspond with the received data. • *RD* = NIL is not permitted.
8095h	The block is already being processed on a priority class that has a lower priority.
80A0h	Error in received acknowledgement.
80A1h	Communication failures: SFC-call after an existing connection has been terminated.
80B0h	Object cannot be found, e.g. DB was not loaded.
80B1h	ANY-pointer error. The length of the data block that must be transferred is wrong.

continued ...

... continue

Value	Description
80B2h	HW-error: module does not exist • The slot that was configured is empty. • Actual module type does not match the required module type. • Decentralized periphery not available. • The respective SDB does not contain an entry for the module.
80B3h	Data may only be read or written, e.g. write protected DB
80B4h	The communication partner does not support the data type specified in *VAR_ADDR*.
80B6h	The received acknowledgement contains an unknown error code.
80BAh	The answer of the communication partner does not fit into the communication telegram.
80C0h	The specified connection is already occupied by another operation.
80C1h	Lack of resources on the CPU where the SFC is being executed, e.g.: • The module is already executing the maximum number of different send operations. • Connection resources may be occupied, e.g. by a receive operation.
80C2h	Temporary lack of resources for the communication partner, e.g.: • The communication partner is currently processing the maximum number of operations. • The required resources (memory, etc.) are already occupied • Not enough memory (initiate compression.)
80C3h	Error when establishing a connection, e.g.: • The local station is connected to the MPI sub-net. • You have addressed the local station on the MPI sub-net. • The communication partner cannot be contacted any longer. • Temporary lack of resources for the communication partner.

SFC 68 - X_PUT - Write data

Description

The SFC 68 X_PUT can be used to write data to an external communication partner that is located outside the local station. No relevant SFC exists on the communication partner. The operation is started when input parameter *REQ* is set to 1. Thereafter the call to SFC 68 is repeated until the value of output parameter *BUSY* becomes 0. The length of the send buffer defined by parameter *SD* (in the sending CPU) must be identical or greater than the receive buffer defined by parameter *VAR_ADDR* (for the communication partner) and the data types of *SD* and *VAR_ADDR* must be identical.

Parameters

Parameter	Declaration	Data type	Memory block	Description
REQ	INPUT	BOOL	I, Q, M, D, L, constant	control parameter "request to activate", used to initiate the operation
CONT	INPUT	BOOL	I, Q, M, D, L, constant	control parameter "continue", determines whether the connection to the communication partner is terminated or not when the operation has been completed
DEST_ID	INPUT	WORD	I, Q, M, D, L, constant	Address parameter "destination ID". Contains the MPI address of the communication partner.
VAR_ADDR	INPUT	ANY	I, Q, M, D	Reference to the buffer in the partner-CPU into which data must be written. You must select a data type that is supported by the communication partner.
SD	INPUT	ANY	I, Q, M, D	Reference to the buffer in the local CPU that contains the send data. The following data types are permitted: BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5_TIME, DATE_AND_TIME as well as arrays of the above data types, with the exception of BOOL.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed.
BUSY	OUTPUT	BOOL	I, Q, M, D, L	*BUSY* = 1: the send operation has not been completed. *BUSY* = 0: The send operation has been completed or no send operation is active.

Data consistency

The following rules must be satisfied to prevent the data consistency from being compromised:

- Active CPU (sender of data):
  The send buffer should be written in the OB that issues the call to the respective SFC. If this is not possible the send buffer should only be written when processing of the first call to the respective SFC has been completed.
- Active CPU (sender of data):
  The maximum amount of data that may be written into the send buffer is determined by the block size of the passive CPU (sender of data).
- Passive CPU (receiver of data):
  Receive data should be read from the receive buffer while interrupts are inhibited.

Operating mode transition to STOP mode

When the CPU changes to STOP mode the connection established by means of the SFC 68 is terminated and data can no longer be sent. If the send data had already been copied into the internal buffer when the transition to STOP mode occurs the contents of the buffer is discarded.

Operating mode transition of the partners to STOP mode

A transition to operating mode STOP of the CPU of the communication partner does not affect the data transfer, since it is also possible to write data in operating mode STOP.

RET_VAL (Return value)

The "real error information" that is contained in the table "specific error information" a. o. may be classified as follows:

Value	Description
809xh	Error on the CPU where the SFC is being executed.
80Axh	Permanent communication error.
80Bxh	Error on the communication partner.
80Cxh	Temporary error.

Specific error information:

Value	Description
0000h	Processing completed without errors.
7000h	Call issued with *REQ* = 0 (call without processing), *BUSY* is set to 0, no data transfer is active.
7001h	First call with *REQ* = 1: data transfer initiated; *BUSY* is set to 1.
7002h	Intermediate call (*REQ* irrelevant): data transfer active; *BUSY* is set to 1.
8090h	The specified target address of the communication partners is not valid, e.g. • bad *IOID* • bad base address exists • bad MPI-address (> 126)
8092h	Error in *SD* or *RD*, e.g.: • illegal length of *SD* • *SD* = NIL is not permitted
8095h	The block is already being processed on a priority class that has a lower priority.
80A0h	The data type specified by *SD* of the sending CPU is not supported by the communication partner.
80A1h	Communication failures: SFC-call after an existing connection has been terminated.
80B0h	Object cannot be found, e.g. DB was not loaded.
80B1h	ANY-pointer error. The length of the data block that must be transferred is wrong.
80B2h	HW-error: module does not exist • the slot that was configured is empty. • Actual module type does not match the required module type. • Decentralized periphery not available. • The respective SDB does not contain an entry for the module.
80B3h	Data can either be read or written, e.g. write protected DB
80B4h	The communication partner does not support the data type specified in *VAR_ADDR*.
80B6h	The received acknowledgement contains an unknown error code.
80B7h	Data type and / or the length of the transferred data does not fit the buffer in the partner CPU where the data must be written.
80BAh	The answer of the communication partner does not fit into the communication telegram.
80C0h	The specified connection is already occupied by another operation.
80C1h	Lack of resources on the CPU where the SFC is being executed, e.g.: • the module is already executing the maximum number of different send operations. • connection resources may be occupied, e.g. by a receive operation.
80C2h	Temporary lack of resources for the communication partner, e.g.: • The communication partner is currently processing the maximum number of operations. • The required resources (memory, etc.) are already occupied • Not enough memory (initiate compression)
80C3h	Error when establishing a connection, e.g.: • The local station is connected to the MPI sub-net. • You have addressed the local station on the MPI sub-net. • The communication partner cannot be contacted any longer. • Temporary lack of resources for the communication partner.

SFC 69 - X_ABORT - Disconnect

Description

The SFC 69 X_ABORT can be used to terminate a connection to a communication partner that is located outside the local station, provided that the connection was established by means one of SFCs 65, 67 or 68. The operation is started when input parameter *REQ* is set to 1.

If the operation belonging to SFCs 65, 67 or 68 has already been completed (*BUSY* = 0) then the connection related resources occupied by both partners are enabled again when the call to the SFC 69 has been issued. However, if the respective operation has not yet been completed (*BUSY* = 1), the call to the respective SFC 65, 67 or 68 must be repeated after the connection has been terminated with *REQ* = 0 and *CONT* = 0. The connection resources are only available again when *BUSY* = 0.
The SFC 69 can only be called on the side where SFC 65, 67 or 68 is being executed.

Parameters

Parameter	Declaration	Data type	Memory block	Description
REQ	INPUT	BOOL	I, Q, M, D, L, constant	Control parameter "request to activate", used to initiate the operation
DEST_ID	INPUT	WORD	I, Q, M, D, L, constant	Address parameter "destination ID". Contains the MPI address of the communication partner.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed.
BUSY	OUTPUT	BOOL	I, Q, M, D, L	*BUSY* = 1: connection termination not yet completed. *BUSY* = 0: connection termination has been completed.

Operating mode transition to STOP mode

The connection termination initiated by means of the SFC 69 is still completed, even if the CPU changes to STOP mode.

Operating mode transition of the partners to STOP mode

A transition to operating mode STOP of the CPU of the communication partner does not affect the connection termination, the connection is terminated in spite of the change of operating mode.

RET_VAL (Return value)

The "real error information" that is contained in the table "specific error information" and others may be classified as follows:

Value	Description
809xh	Error on the CPU where the SFC is being executed
80Axh	Permanent communication error
80Bxh	Error on the communication partner
80Cxh	Temporary error

Specific error information:

Value	Description
0000h	*REQ* = 1 when the specified connection has not been established.
7000h	Call issued with *REQ* = 0 (call without processing), *BUSY* is set to 0, no data transfer is active.
7001h	First call with *REQ* = 1: data transfer initiated; *BUSY* is set to 1.
7002h	Intermediate call with *REQ* = 1.
8090h	The specified target address of the communication partners is not valid, e.g. • bad *IOID* • bad base address exists • bad MPI-address (> 126)
8095h	The block is already being processed on a priority class that has a lower priority.
80A0h	Error in the acknowledgement that was received.
80A1h	Communication failures: SFC-call after an existing connection has been terminated.
80B1h	ANY-pointer error. The length of the data block that must be transferred is wrong.
80B4h	ANY-pointer data type error, or ARRAY of the specified data type is not permitted.
80B6h	The received acknowledgement contains an unknown error code.
80BAh	The answer of the communication partner does not fit into the communication telegram.
80C0h	The specified connection is already occupied by another operation.
80C1h	Lack of resources on the CPU where the SFC is being executed, e.g.: • the module is already executing the maximum number of different send operations. • connection resources may be occupied, e.g. by a receive operation.
80C2h	Temporary lack of resources for the communication partner, e.g.: • The communication partner is currently processing the maximum number of operations. • The required resources (memory, etc.) are already occupied. • Not enough memory (initiate compression).
80C3h	Error when establishing a connection, e.g.: • The local station is connected to the MPI sub-net. • You have addressed the local station on the MPI sub-net. • The communication partner cannot be contacted any longer. • Temporary lack of resources for the communication partner.

SFC 81 - UBLKMOV - Copy data area without gaps

Description

The SFC 81 UBLKMOV (uninterruptible block move) creates a consistent copy of the contents of a memory block (= source field) in another memory block (= target field). The copy procedure cannot be interrupted by other activities of the operating system.
It is possible to copy any memory block, with the exception of:

- the following blocks: FC, SFC, FB, SFB, OB, SDB
- counters
- timers
- memory blocks of the peripheral area
- data blocks those are irrelevant to the execution.

The maximum amount of data that can be copied is 512bytes.

Interruptability

It is not possible to interrupt the copy process. For this reason it is important to note that any use of the SFC 81 will increase the reaction time of your CPU to interrupts.

Parameters

Parameter	Declaration	Data type	Memory block	Description
SRCBLK	INPUT	ANY	I, Q, M, D, L	Specifies the memory block that must be copied (source field). Arrays of data type STRING are not permitted.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	The return value contains an error code if an error is detected when the function is being processed.
DSTBLK	OUTPUT	ANY	I, Q, M, D, L	Specifies the target memory block where the data must be copied (target field). Arrays of data type STRING are not permitted.

Note!

The source and target field must not overlap.

If the specified target field is larger than the source field, only the amount of data located in the source field will be copied into the target field. However, if the size of the specified target field is less than the size of the source field, then only the amount of data that will fit into the target field will be copied.

If the data type of the ANY-pointer (source or target) is BOOL, then the specified length must be divisible by 8, otherwise the SFC will not be executed.

If the data type of the ANY-pointer is STRING the specified length must be 1.

RET_VAL
(Return value)

Value	Description
0000h	no error
8091h	The source area is located in a data block that is not relevant to execution.

Chapter 5 VIPA specific blocks

Overview

Here the VIPA specific blocks are described that are exclusively may be used with the standard CPUs from VIPA of the Systems 100V, 200V, 300V and 500V. Please note that not every block listed here is integrated in every system CPU.

The assignment to the according system may be in the table of the "Overview".

Content

Overview

General

The integrated SFCs are programmed in Assembler and for this they have a fast processing time. They do not occupy space in the internal program memory. The integrated blocks are called in the user application.

The following pages show the VIPA specific blocks that may be called for special functions in the control application.

Assignment table block ←→ CPU

For not every block is integrated in every CPU family the following table shows the assignment between integrated block and CPU. The first number specifies the CPU family, e.g. 31x means CPU 31x from System 300V. For a better overview, in the following block descriptions the corresponding extract of this table is repeated.

Block	Name	Description	11x	21x	31x	51x
FB 55	IP_CONF	Programmed communication connections		•*		
FC 0	SEND	Send data to CP 240		•		
FC 1	RECEIVE	Receive data from CP 240		•		
FC 5	AG_SEND	Send data to Ethernet CP 243		•*		
FC 6	AG_RECEIVE	Receive data from Ethernet CP 243		•*		
FC 8	STEUERBIT	Modem functionality of CP 240		•		
FC 9	SYNCHRON_ RESET	Synchronization between CPU and CP 240		•		
FC 11	ASCII_ FRAGMENT	Receive ASCII fragmented with CP 240		•		
SFC 204	IP_CONF	Internal used for FB 55 IP_CONF		•*		
SCF 205	AG_SEND	Internal used for FC 5 AG_SEND		•*		
SCF 206	AG_RECV	Internal used for FC 6 AG_RECEIVE		•*		
SFC 207	SER_CTRL	RS232 modem functionality	•	•		
SFC 216	SER_CFG	RS232 parameterization	•	•		
SFC 217	SER_SND	RS232 Send	•	•		
SFC 218	SER_RCV	RS232 Receive	•	•		
SFC 219	CAN_TLGR	Send CAN telegram	•	•	•	•
SFC 220	MMC_CR_F	Create file on MMC	•	•	•	•
SFC 221	MMC_RD_F	Read file from MMC	•	•	•	•
SFC 222	MMC_WR_F	Write to file on MMC	•	•	•	•
SFC 223	PWM	Parameterize pulse duration modulation	•			
SFC 224	HSC	Parameterize high-speed counter	•			
SFC 225	HF_PWM	Parameterize HF pulse duration modulation (up to 50kHz)	•			
SFC 227	TD_PRM	Parameterization for TD200 communication	•	•	•	•

*) only in CPU 21x-2BT1x available

continued ...

... continue

Block	Name	Description	11x	21x	31x	51x
SFC 228	RW_KACHEL	Read/Write page frame		•	•	•
SFC 230	SEND	Send via page frame		•	•	•
SFC 231	RECEIVE	Receive via page frame		•	•	•
SFC 232	FETCH	Fetch via page frame		•	•	•
SFC 233	CONTROL	Control for page frame communication		•	•	•
SFC 234	RESET	Reset for page frame communication		•	•	•
SFC 235	SYNCHRON	Synchron for page frame communication		•	•	•
SFC 236	SEND_ALL	Send_All via page frame		•	•	•
SFC 237	RECV_ALL	Receive via page frame		•	•	•
SFC 238	CONTROL1	Control for page frame communication with type ANZW: Pointer and parameter IND.		•	•	•

Include VIPA library

Overview

The VIPA specific blocks may be found at www.vipa.de as downloadable library at the service area with *Downloads* > *VIPA LIB*.

The library is available as packed zip-file.

If you want to use VIPA specific blocks, you have to import the library into your project.

Execute the following steps:

- Extract FX000011_Vxxx.zip
- "Retrieve" the library
- Open library and transfer blocks into the project

Unzip FX000011_Vxxx.zip

Start your un-zip application with a double click on the file FX000011_Vxxx.zip and copy the file VIPA.ZIP to your work directory. It is not necessary to extract this file, too.

Retrieve library

To retrieve your library for the SPEED7-CPUs, start the SIMATIC manager from Siemens. Open the dialog window for archive selection via **File** > *Retrieve*. Navigate to your work directory.

Choose VIPA.ZIP and click at [Open].

Select a destination folder where the blocks are to be stored. [OK] starts the extraction.

Open library and transfer blocks to project

After the extraction open the library.

Open your project and copy the necessary blocks from the library into the directory "blocks" of your project.

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

11x	21x	31x	51x
	✓*		

FB 55 - IP_CONFIG - Programmed Communication Connections

Overview

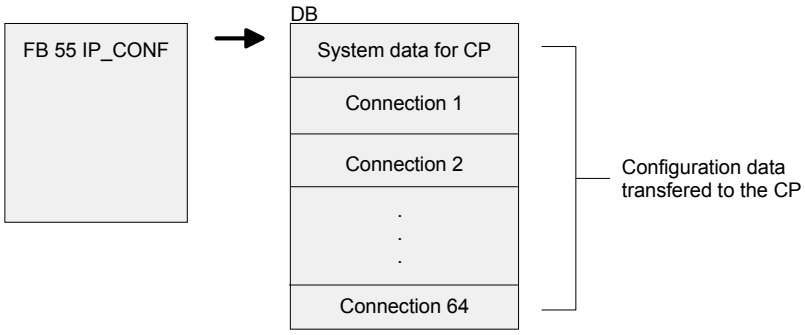
In some situations, it is an advantage to set up communication connections not with Siemens NetPro but program-controlled by a specific application.

A VIPA function block (FB 55) is available for these applications that allows flexible transfer of data blocks with configuration data to a CP.

Within the FB 55 the VIPA special block SFC 204 stored in the CPU is called.

Principle

Configuration data for communication connections may be transferred to the CP by the FB 55 called in the user program.



The configuration DB may be loaded into the CP at any time.

Attention!

As soon as the user program transfers the connection data via FB 55 IP_CONFIG, the CPU switches the CP briefly to STOP. The CP accepts the system data (including IP address) and the new connection data and processes it during startup (RUN).

*) only with CPU 21x-2BT1x available

FB 55 - IP_CONFIG

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

The Job is started with *ACT* = 1.

Parameters

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

Error information

ERROR	STATUS	Description
0	0000h	Job completed without errors
0	8181h	Job active
1	80B1h	The amount of data to be sent exceeds the upper limit permitted for this service.
1	80C4h	Communication error The error can occur temporarily; it is usually best to repeat the job in the user program.
1	80D2h	Configuration error, the module you are using does not support this service.
1	8183h	The CP rejects the requested record set number.
1	8184h	System error or illegal parameter type.
1	8185h	The value of the *LEN* parameter is larger than the *CONF_DB* less the reserved header (4bytes) or the length information is incorrect.
1	8186h	Illegal parameter detected. The ANY pointer *CONF_DB* does not point to data block.
1	8187h	Illegal status of the FB. Data in the header of *CONF_DB* was possibly overwritten.
1	8A01h	The status code in the record set is invalid (value is >=3).
1	8A02h	There is no job running on the CP; however the FB has expected an acknowledgment for a competed job.
1	8A03h	There is no job running on the CP and the CP is not ready; the FB triggered the first job to read a record set.
1	8A04h	There is no job running on the CP and the CP is not ready; the FB nevertheless expected an acknowledgment for a completed job.
1	8A05h	There is a job running, but there was no acknowledgment; the FB nevertheless triggered the first job for a read record set job.
1	8A06h	A job is complete but the FB nevertheless triggered the first job for a read record set job.
1	8B01h	Communication error, the DB could not be transferred.
1	8B02h	Parameter error, double parameter field
1	8B03h	Parameter error, the subfield in the parameter field is not permitted.
1	8B04h	Parameter error, the length specified in the FB does not match the length of the parameter fields/subfields.
1	8B05h	Parameter error, the length of the parameter field is invalid.
1	8B06h	Parameter error, the length of the subfield is invalid.
1	8B07h	Parameter error, the ID of the parameter field is invalid.
1	8B08h	Parameter error, the ID of the subfield is invalid.
1	8B09h	System error, the connection does not exist.
1	8B0Ah	Data error, the content of the subfield is not correct.
1	8B0Bh	Structure error, a subfield exists twice.
1	8B0Ch	Data error, the parameter does not contain all the necessary parameters.
1	8B0Dh	Data error, the *CONF_DB* does not contain a parameter field for system data.

continued...

...continue

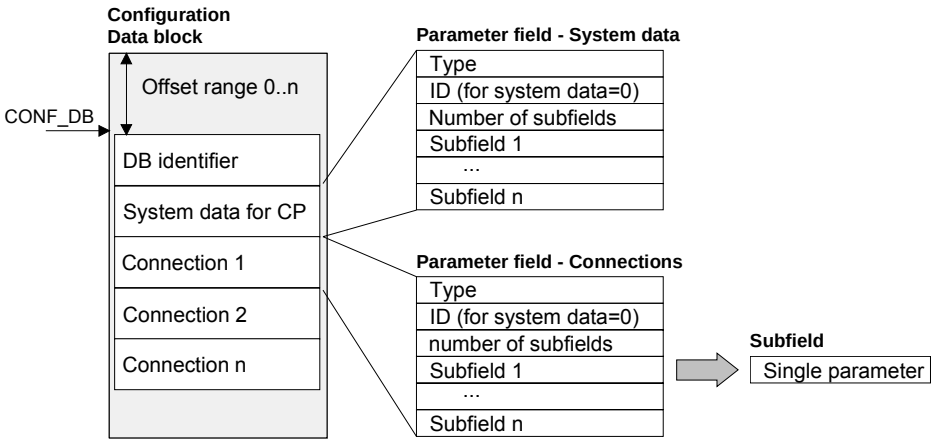
ERROR	STATUS	Description
1	8B0Eh	Data error/structure error, the *CONF_DB* type is invalid.
1	8B0Fh	System error, the CP does not have enough resources to process *CONF_DB* completely.
1	8B10	Data error, configuration by the user program is not set.
1	8B11	Data error, the specified type of parameter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

Configuration Data Block

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

Structure

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



Parameter field for *system data for CP*

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

Structure

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

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

11x	21x	31x	51x
	✓		

FC 0 - SEND - Send to CP 240

Description

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

Parameters

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

ADR

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

_DB

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

ABD

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

ANZ

Number of the bytes that are to be transferred.

FRG enable send

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

PAFE

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

1 = Data block not present

2 = Data block too short

3 = Data block number outside valid range

GESE, ANZ_INT
ENDE_KOM
LETZTER_BLOCK
SENDEN_LAEUFT
FEHLER_KOM

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

11x	21x	31x	51x
	✓		

FC 1 - RECEIVE - Receive from CP 240

Description

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

Parameters

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

ADR

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

_DB

Number of the data block, which contains the data.

ABD

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

ANZ

Word variable that contains the amount of received bytes.

EMFR

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

PAFE

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

1 = Data block not present

2 = Data block too short

3 = Data block number outside valid range

GEEM, ANZ_INT
LETZTER_BLOCK
EMPF_LAEUFT
FEHLER_EMPF
OFFSET

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

11x	21x	31x	51x
	✓*		

FC 5 - AG_SEND / FC 6 - AG_RECV - CP 243 communication

Overview

These two blocks serve for the execution of connection commands at the PLC side of a CP 243. By including these blocks into the cycle block OB 1 you may send and receive data cyclically.

Within these blocks the VIPA special blocks SFC 205 and SFC 206 stored in the CPU are called.

Note!

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

Communication blocks

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

AG_SEND (FC 5)

This block transfers the user data from the data area given in *SEND* to the CP specified via *ID* and *LADDR*. As data area you may set a PA, bit memory or data block area. When the data area has been transferred without errors, "order ready without error" is returned.

AG_RECV (FC 6)

The block transfers the user data from the CP into a data area defined via *RECV*. As data area you may set a PA, bit memory or data block area. When the data area has been transferred without errors, "order ready without error" is returned.

Status displays

The CP processes send and receive commands independently from the CPU cycle and needs for this transfer time. The interface with the FC blocks to the user application is here synchronized by means of acknowledgements/receipts.

For status evaluation the communication blocks return parameters that may be evaluated directly in the user application.

These status displays are updated at every block call.

Deployment at high communication load

Do not use cyclic calls of the communication blocks in OB1. This causes a permanent communication between CPU and CP. Program instead the communication blocks within a time OB where the cycle time is higher res. event controlled.

*) only with CPU 21x-2BT1x available

FC call is faster than CP transfer time

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

AG_SEND

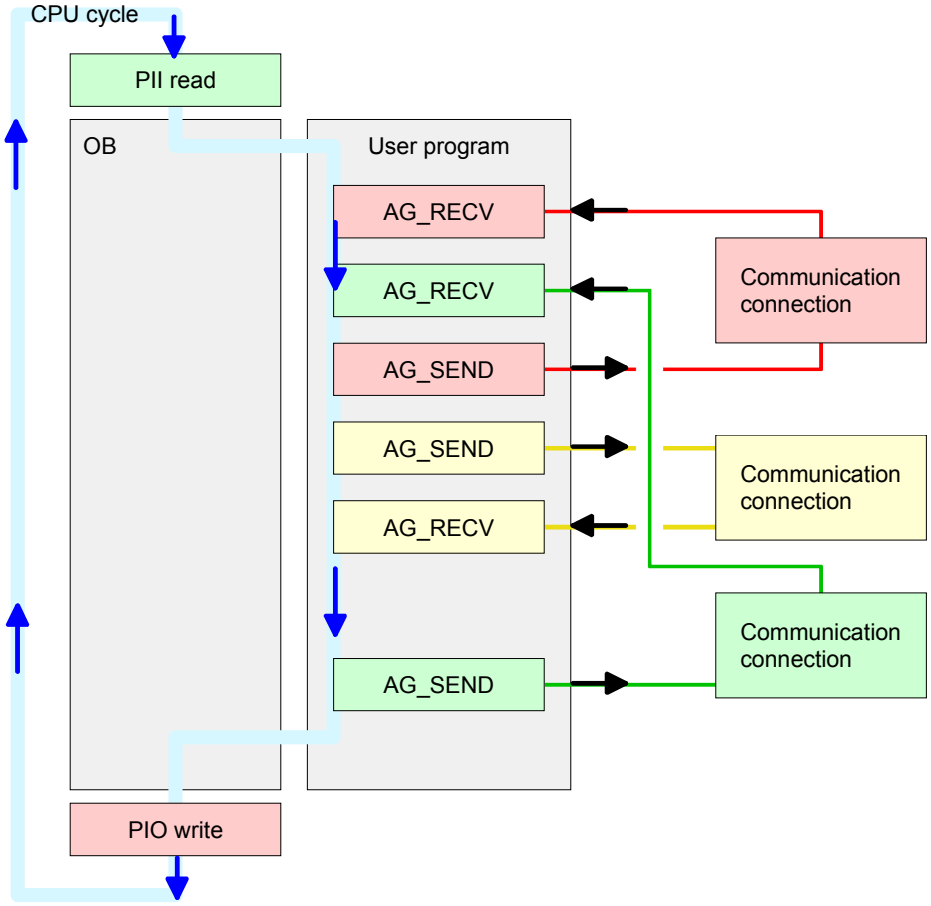
No command is accepted until the data transfer has been acknowledged from the partner via the connection. Until this you receive the message "Order running" before the CP is able to receive a new command for this connection.

AG_RECV

The order is acknowledged with the message "No data available yet" as long as the CP has not received the receive data completely.

AG_SEND, AG_RECV in the user application

The following illustration shows a possible sequence for the FC blocks together with the organizations and program blocks in the CPU cycle:



The FC blocks with concerning communication connection are summed up by color. Here you may also see that your user application may consist of any number of blocks. This allows you to send or receive data (with AG_SEND res. AG_RECV) event or program driven at any wanted point within the CPU cycle.

You may also call the blocks for **one** communication connection several times within one cycle.

AG_SEND (FC 5) By means of AG_SEND the data to send are transferred to the CP 243.

Parameters

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

AG_RECV (FC 6) By means of AG_RECV the data received from the Ethernet CP 243 are transferred to the CPU.

Parameters

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

DONE, ERROR, STATUS

The following table shows all messages that may be returned by the Ethernet CP 243 after a SEND res. RECV command.

A "-" means that this message is not available for the concerning SEND res. RECV command.

DONE (SEND)	NDR (RECV)	ERROR	STATUS	Description
1	-	0	0000h	Order ready without error
-	1	0	0000h	New data received without error
0	-	0	0000h	No order present
-	0	0	8180h	No data available yet
0	0	0	8181h	Order running
0	0	1	8183h	No CP project engineering for this order
0	-	1	8184h	System error
-	0	1	8184h	System error (destination data area failure)
0	-	1	8185h	Parameter LEN exceeds source area SEND
	0	1	8185h	Destination buffer (RECV) too small
0	0	1	8186h	Parameter ID invalid (not within 1 ...16)
0	-	1	8302h	No receive resources at destination station, receive station is not able to process received data fast enough res. has no receive resources reserved.
0	-	1	8304h	The connection is not established. The send command shouldn't be send again before a delay time of >100 ms.
-	0	1	8304h	The connection is not established. The receive command shouldn't be send again after a delay time of >100 ms.
0	-	1	8311h	Destination station not available with the defined Ethernet address.
0	-	1	8312h	Ethernet error in the CP
0		1	8F22h	Source area invalid, e.g. when area in DB not present, Parameter LEN < 0
-	0	1	8F23h	Source area invalid, e.g. when area in DB not present, Parameter LEN < 0
0	-	1	8F24h	Range error at reading a parameter.
-	0	1	8F25h	Range error at writing a parameter.
0	-	1	8F28h	Orientation error at reading a parameter.
-	0	1	8F29h	Orientation error at writing a parameter.
-	0	1	8F30h	Parameter is within write protected 1st recent data block
-	0	1	8F31h	Parameter is within write protected 2nd recent data block
0	0	1	8F32h	Parameter contains oversized DB number.
0	0	1	8F33h	DB number error
0	0	1	8F3Ah	Area not loaded (DB)

continued ...

... continue DONE, ERROR, STATUS

DONE (SEND)	NDR (RECV)	ERROR	STATUS	Description
0	-	1	8F42h	Acknowledgement delay at reading a parameter from peripheral area.
-	0	1	8F43h	Acknowledgement delay at writing a parameter from peripheral area.
0	-	1	8F44h	Address of the parameter to read locked in access track.
-	0	1	8F45h	Address of the parameter to write locked in access track.
0	0	1	8F7Fh	Internal error e.g. invalid ANY reference e.g. parameter LEN = 0.
0	0	1	8090h	Module with this module start address not present or CPU in STOP.
0	0	1	8091h	Module start address not within double word grid.
0	0	1	8092h	ANY reference contains type setting unequal BYTE.
-	0	1	80A0h	Negative acknowledgement at reading the module.
0	0	1	80A4h	reserved
0	0	1	80B0h	Module doesn't recognize record set.
0	0	1	80B1h	The length setting (in parameter LEN) is invalid.
0	0	1	80B2h	reserved
0	0	1	80C0h	Record set not readable.
0	0	1	80C1h	The set record set is still in process.
0	0	1	80C2h	Order accumulation.
0	0	1	80C3h	The operating sources (memory) of the CPU are temporarily occupied.
0	0	1	80C4h	Communication error (occurs temporarily; a repetition in the user application is reasonable.)
0	0	1	80D2h	Module start address is wrong.

Status parameter at reboot

At a reboot of the CP, the output parameter are set back as follows:

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

11x	21x	31x	51x
	✓		

FC 8 - STEUERBIT - Modem functionality CP 240

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

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

Write: DTR, RTS

Parameters

Name	Declaration	Type	Comment
ADR	IN	INT	Logical Address
RTS	IN	BOOL	New state RTS
DTR	IN	BOOL	New state DTR
MASKE_RTS	IN	BOOL	0: do nothing 1: set state RTS
MASKE_DTR	IN	BOOL	0: do nothing 1: set state DTR
STATUS	OUT	BYTE	Status flags
DELTA_STATUS	OUT	BYTE	Status flags of change between 2 accesses
START	IN_OUT	BOOL	Start bit of the function
AUFTRAG_LAEU	IN_OUT	BOOL	Status of function
RET_VAL	OUT	WORD	Return value (0 = OK)

Note!

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

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

RTS, DTR This parameter presets the status of *RTS* res. *DTR*, which you may activate via *MASK_RTS* res. *MASK_DTR*.

MASK_RTS, MASK_DTR With 1, the status of the according parameter is taken over when you set *START* to 1.

STATUS, DELTA_STATUS *STATUS* returns the actual status of the modem lines. *DELTA_STATUS* returns the state of the modem lines that have changed since the last access.

The bytes have the following structure:

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

START By setting of *START*, the state, which has been activated via the mask, is taken over.

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

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

11x	21x	31x	51x
	✓		

FC 9 - SYNCHRON_RESET - Synchronization CPU and CP 240

Description

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

Note!

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

Parameters

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

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

TIMER_NR Number of the timer for the delay time.

ANL With *ANL* = 1 the handling block is informed that a STOP/START res. NETZ-AUS/NETZ-EIN has been executed at the CPU and now a synchronization is required. After the synchronization, *ANL* is automatically set back.

NULL Parameter is used internally.

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

STEUERB_S Here you have to set the bit memory byte where the control bits ENDE_KOM, LETZTER_BLOCK, SENDEN_LAEUFT and FEHLER_KOM for the SEND-FC are stored.

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

11x	21x	31x	51x
	✓		

FC 11 - ASCII_FRAGMENT - Receive fragmented from CP 240

Description

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

Parameters

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

ADR

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

_DB

Number of the data block, which contains the data to receive.

ABD

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

ANZ

Word variable that contains the amount of bytes that have been received.

EMFR

By setting of *EMFR*, the handling block announces that data has been received. Only by setting back *EMFR* in the user application new data can be received.

PAFE

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

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

GEEM, ANZ_INT
LETZTER_BLOCK
EMPF_LAEUFT
FEHLER_EMPF

These parameters are internally used. They serve the information exchange between the handling blocks. For the deployment of the SYNCHRON_RESET (FC 9) the control bits LETZTER_BLOCK, EMPF_LAEUFT and FEHLER_EMPF must always be stored in a bit memory byte.

11x	21x	31x	51x
✓	✓		

Serial communication - SFC 207 and SFC 216...218

General

Some CPUs provide serial interfacing facilities between the processes of different source and destination systems. For the serial communication the CPUs have got a serial interface which may be controlled by these blocks.

Protocols

The protocols ASCII, STX/ETX, 3964R, USS and Modbus are supported.

Parameterization

The parameterization happens during runtime by means of the SFC 216 (SER_CFG). With exception of ASCII the parameters of every protocol is to be stored in a DB.

Communication

The communication is controlled by means of SFCs. Sending is executed with the SFC 217 (SER_SND) and the reception via SFC 218 (SER_RCV).

Another call of the SFC 217 SER_SND, 3964R, USS and Modbus provides you via RetVal with a return value which contains among others recent information about the acknowledgement of the partner.

The protocols USS and Modbus allows you to read the acknowledgement telegram by calling the SFC 218 SER_RCV after a SER_SND.

Overview over the SFCs for the serial communication

The following SFCs are deployed for the serial communication:

SFC		Description
SFC 207	SER_CTRL	Modem functionality
SFC 216	SER_CFG	RS232/RS485 Parameterization
SFC 217	SER_SND	RS232/RS485 Send
SFC 218	SER_RCV	RS232/RS485 Receive

11x	21x	31x	51x
✓	✓		

SFC 207 - SER_CTRL

Modem functionality

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

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

Parameters

Name	Declaration	Type	Description
WRITE	IN	BYTE	Bit 0: New state DTR Bit 1: New state RTS
MASKWRITE	IN	BYTE	Bit 0: Set state DTR Bit 1: Set state RTS
READ	OUT	BYTE	Status flags (CTS, DSR, RI, CD, DTR, RTS)
READDELTA	OUT	BYTE	Status flags of change between 2 accesses
RETVAL	OUT	WORD	Return value (0 = OK)

WRITE

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

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

MASKWRITE

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

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

READ

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

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

RETVAL (Return value)

Value	Description
0000h	no error
8x24h	Error SFC parameter x, with x: 1: Error at *WRITE* 2: Error at *MASKWRITE* 3: Error at *READ* 4: Error at *READDELTA*
809Ah	Interface missing
809Bh	Interface not configured (SFC 216)

11x	21x	31x	51x
✓	✓		

SFC 216 - SER_CFG

Description

The parameterization happens during runtime deploying the SFC 216 SER_CFG. You have to store the parameters for STX/ETX, 3964R, USS and Modbus in a DB.

Please regard that not all protocols support the complete value range of the parameters. More detailed information is to be found in the description of the according parameter.

Note!

Please regard that the SFC 216 is not called again during a communication because as a result of this all buffers are cleared.

If you don't want to alter the communication parameter any more, you should place the call of the SFC 216 in the start-up OB 100.

Parameters

Name	Declaration	Type	Description
PROTOCOL	IN	BYTE	Number of the protocol
PARAMETER	IN	ANY	Pointer to protocol-parameters
BAUDRATE	IN	BYTE	No of baudrate
CHARLEN	IN	BYTE	Number of Data bits
PARITY	IN	BYTE	Parity
STOPBITS	IN	BYTE	Number of stop bits
FLOWCONTROL	IN	BYTE	Flow control (1 fixed)
RETVAL	OUT	WORD	Return value (0 = OK)

PROTOCOL

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

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

PARAMETER (as DB)

At ASCII protocol, this parameter is ignored.

At STX/ETX, 3964R, USS and Modbus you fix here a DB that contains the communication parameters and has the following structure for the according protocols:

Data block at STX/ETX

DBB0:	STX1	BYTE	(1. Start-ID in hexadecimal)
DBB1:	STX2	BYTE	(2. Start-ID in hexadecimal)
DBB2:	ETX1	BYTE	(1. End-ID in hexadecimal)
DBB3:	ETX2	BYTE	(2. End-ID in hexadecimal)
DBW4:	TIMEOUT	WORD	(max. delay time between 2 telegrams in a time window of 10ms)

Note!

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

With not used IDs please always enter FFh!

Data block at 3964R

DBB0:	Prio	BYTE	(The priority of both partners must be different. Prio 0 and 1 are possible)
DBB1:	ConnAttmptNr	BYTE	(Number of connection trials)
DBB2:	SendAttmptNr	BYTE	(Number of telegram retries)
DBW4:	CharTimeout	WORD	(Char. delay time in 10ms time window)
DBW6:	ConfTimeout	WORD	(Ackn. delay time in 10ms time window)

Data block at USS

DBW0:	Timeout	WORD	(Delay time in 10ms time grid)

Data block at Modbus-Master

DBW0:	Timeout	WORD	(Respond delay time in 10ms time grid)

Data block at Modbus-Slave

DBB0:	Address	BYTE	(Address 1...247 in the Modbus network)
DBW2:	Timeout	WORD	(Respond delay time in 10ms time grid)

BAUDRATE

Velocity of data transfer in Bit/s (Baud).

01h: 150 Baud	05h: 1800 Baud	09h: 9600 Baud	0Dh: 57600 Baud
02h: 300 Baud	06h: 2400 Baud	0Ah: 14400 Baud	0Eh: 115200 Baud
03h: 600 Baud	07h: 4800 Baud	0Bh: 19200 Baud	
04h: 1200 Baud	08h: 7200 Baud	0Ch: 38400 Baud	

CHARLEN

Number of data bits where a character is mapped to.

0: 5Bit 1: 6Bit 2: 7Bit 3: 8Bit

Supported values:

Bit	ASCII	STX/ETX	3964R	USS	Modbus RTU	Modbus ASCII
5	x		x			
6	x	x	x			
7	x	x	x			x
8	x	x	x	x	x	x

PARITY

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

0: NONE 1: ODD 2: EVEN

STOPBITS

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

1: 1Bit 2: 1.5Bit 3: 2Bit

1.5Bit can only be used with *CHARLEN* 5 at this number of data 2Bit is not allowed.

FLOWCONTROL

With this bit you affect the behavior from signal **R**equest **t**o **s**end.

0: RTS off

1: RTS is "0" at Receive (AutoRTS)
RTS is "1" at Send (AutoRTS)

2: HW flow (only at ASCII protocols)

Note!

Note: For RS485 *FLOWCONTROL* is not evaluated.
It is set automatically to "1" (AutoRTS)!

RETVAL (Return value)

Value	Description
0000h	no error
809Ah	interface not found
8x24h	Error at SFC-Parameter x, with x: 1: Error at *PROTOCOL* 2: Error at *PARAMETER* 3: Error at *BAUDRATE* 4: Error at *CHARLENGTH* 5: Error at *PARITY* 6: Error at *STOPBITS* 7: Error at *FLOWCONTROL*
809xh	Error in SFC parameter value x, where x: 1: Error at *PROTOCOL* 3: Error at *BAUDRATE* 4: Error at *CHARLENGTH* 5: Error at *PARITY* 6: Error at *STOPBITS* 7: Error at *FLOWCONTROL*
8092h	Access error in parameter DB (DB too short)
828xh	Error in parameter x of DB parameter, where x: 1: Error 1. parameter 2: Error 2. parameter ...

11x	21x	31x	51x
✓	✓		

SFC 217 - SER_SND

Description This block allows to send data via the serial interface.

Parameters

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

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

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

DATALEN Word where the number of sent bytes is stored.

At **STX/ETX** and **3964R**, the length set in *DATAPTR* or 0 is entered.

At **ASCII**, the value may be different from the sent length when the data is sent that fast that not all data can be stored in the send buffer of 256byte.

RETVAL (Return value)

Value	Description
0000h	Send data - ready
1000h	Nothing sent (data length 0)
20xxh	Protocol executed error free with xx bit pattern for diagnosis.
7001h	Data is stored in internal buffer - active (busy).
7002h	Transfer - active
80xxh	Protocol executed with errors with xx bit pattern for diagnosis (no acknowledgement by partner).
90xxh	Protocol not executed with xx bit pattern for diagnosis (no acknowledgement by partner).
8x24h	Error in SFC parameter x, where x: 1: Error in *DATAPTR* 2: Error in *DATALEN.*
8122h	Error in parameter *DATA* of SFC 216 (e.g. no DB).
807Fh	Internal error
809Ah	RS232 interface not found.
809Bh	RS232 interface not configured.

Protocol specific RETVAL values

ASCII

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

STX/ETX

Value	Description
9000h	Buffer overflow (no data sent)
9001h	Data too long (>256byte)
9002h	Data too short (0byte)
9004h	Character not allowed

3964R

Value	Description
2000h	Send ready without error
80FFh	NAK received - error in communication
80FEh	Data transfer without acknowledgement of partner or error at acknowledgement
9000h	Buffer overflow (no data send)
9001h	Data too long (>256byte)
9002h	Data too short (0byte)

USS

Value	Description
2000h	Send ready without error
8080h	Receive buffer overflow (no space for receipt)
8090h	Acknowledgement delay time exceeded
80F0h	Wrong checksum in respond
80FEh	Wrong start sign in respond
80FFh	Wrong slave address in respond
9000h	Buffer overflow (no data send)
9001h	Data too long (>256byte)
9002h	Data too short (<2byte)

Modbus RTU/ASCII Master

Value	Description
2000h	Send ready without error
2001h	Send ready with error
8080h	Receive buffer overflow (no space for receipt)
8090h	Acknowledgement delay time exceeded
80F0h	Wrong checksum in respond
80FDh	Length of respond too long
80FEh	Wrong function code in respond
80FFh	Wrong slave address in respond
9000h	Buffer overflow (no data send)
9001h	Data too long (>256byte)
9002h	Data too short (<2byte)

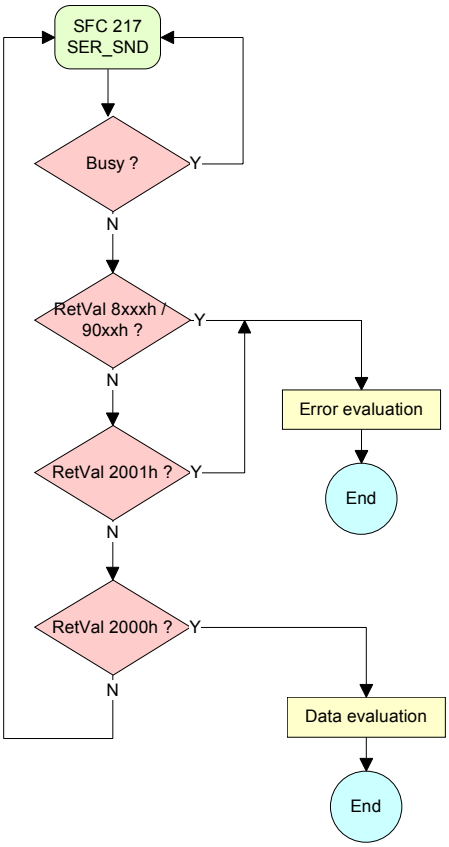
Modbus RTU/ASCII Slave

Value	Description
0000h	Send data - ready
9001h	Data too long (>256byte)

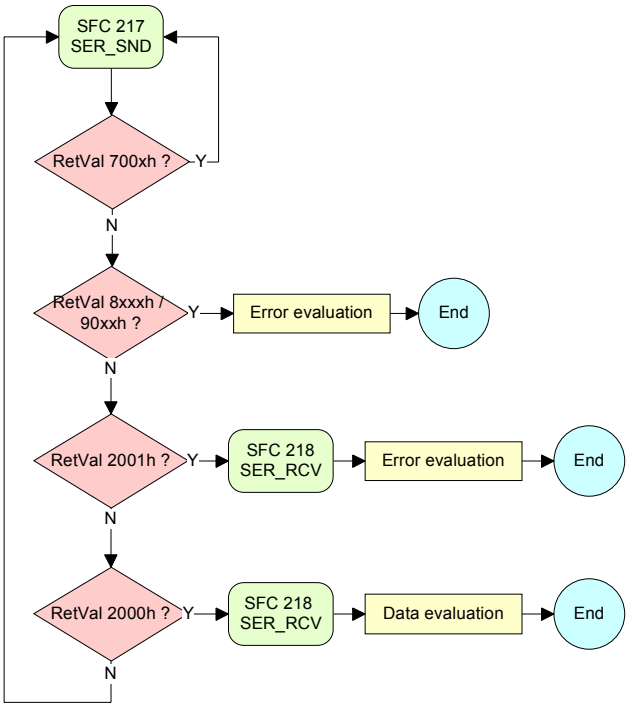
Principles of programming

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

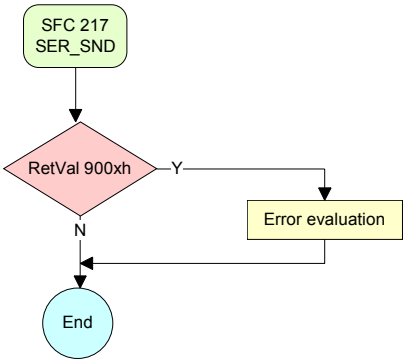
3964R



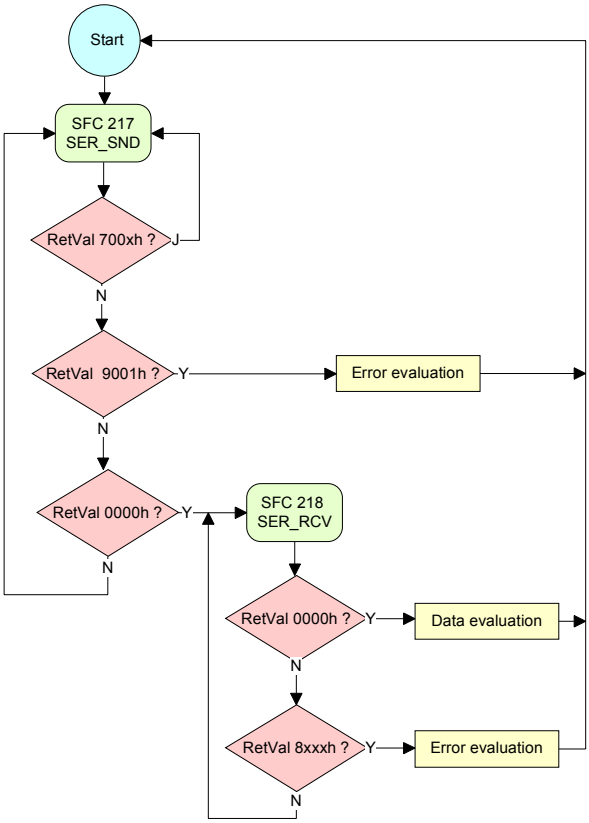
USS / Modbus master



ASCII / STX/ETX



Modbus slave



11x	21x	31x	51x
✓	✓		

SFC 218 - SER_RCV

Description This block receives data via the serial interface.

Parameters

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

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

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

DATALEN Word where the number of received bytes is stored.

At **STX/ETX** and **3964R**, the length of the received user data or 0 is entered.

At **ASCII**, the number of read characters is entered. This value may be different from the read telegram length.

ERROR At ASCII, this word gets an entry in case of an error. The following error messages are possible:

Bit	Error	Description
1	overrun	Overrun when a character could not be read from the interface fast enough.
2	parity	Parity error
3	framing error	Error that shows that a defined bit frame is not met, exceeds the allowed length or contains an additional bit sequence (stop bit error).

RETVAL (Return value)

Value	Description
0000h	no error
1000h	Receive buffer too small (data loss)
8x24h	Error at SFC-Parameter x, with x: 1: Error at *DATAPTR* 2: Error at *DATALEN* 3: Error at *ERROR.*
8122h	Error in parameter *DATA* of SFC 216 (e.g. no DB).
809Ah	serial interface not found.
809Bh	serial interface not configured.

11x	21x	31x	51x
✓	✓	✓	✓

SFC 219 - CAN_TLGR - Send CAN telegram

SFC 219 CAN_TLGR SDO-demand on CAN-master

This block is used by the PLC to cause the CANopen master to execute a SDO read or write access).

Here you address the master via plug-in place number and the destination slave via his CAN address. The process data will be designated of *INDEX* and *SUBINDEX*. It is possible to transfer maximum one data word process data per access via SDO.

Parameters

Name	Declaration	Type	Description
REQUEST	IN	BOOL	Start bit of the job
SLOT_MASTER	IN	BYTE	Slot of the master
NODEID	IN	BYTE	CAN address
TRANSFERTYP	IN	BYTE	Transfer type
INDEX	IN	DWORD	CANopen index
SUBINDEX	IN	DWORD	CANopen subindex
CANOPENERROR	OUT	DWORD	CANopen error
RETVAL	OUT	WORD	Return value
BUSY	OUT	BOOL	Busy flag
DATABUFFER	IN_OUT	ANY	Data area for communication

REQUEST Control parameter: 1: Start order

SLOT_MASTER System 100V: Slot number 0: 21x-2CM02
Slot number 1 … 4: 208-1CA00

System 200V: Slot number 0: 21x-2CM02
Slot number 1 … 32: 208-1CA00

NODEID Address of the CANopen Node (1...127)

TRANSFERTYPE 40h: read SDO
23h: write SDO (1 DWORD)
2Bh: write SDO (1 WORD)
2Fh: write SDO (1 BYTE)

INDEX CANopen Index

SUBINDEX CANopen Subindex

CANOPENERROR If no error occurs *CANOPENERROR* returns value 0.

In case of error the *CANOPENERROR* contains one of the following error messages which are generated in the CAN master:

Code	Description
0x00000000	Successfully service
0x05030000	Toggle bit not alternated
0x05040000	SDO protocol timed out
0x05040001	Client/server command specification not valid or unknown
0x05040002	Invalid block size (block mode only)
0x05040003	Invalid sequence number (block mode only)
0x05040004	CRC error (block mode only)
0x05040005	Out of memory
0x06010000	Unsupported access to an object
0x06010001	Attempt to read a write only object
0x06010002	Attempt to write a read only object
0x06020000	Object does not exist in the object dictionary
0x06040041	Object cannot be mapped to the PDO
0x06040042	The number and length of the objects to be mapped would exceed PDO length
0x06040043	General parameter incompatibility reason
0x06040047	General internal incompatibility in the device
0x06060000	Access failed due to an hardware error
0x06070010	Data type does not match, length of service parameter does not match
0x06070012	Data type does not match, length of service parameter too high
0x06070013	Data type does not match, length of service parameter too low
0x06090011	Sub-index does not exist
0x06090030	Value range of parameter exceeded (only for write access)
0x06090031	Value of parameter written too high
0x06090032	Value of parameter written too low
0x06090036	Maximum value is less than minimum value
0x08000000	general error
0x08000020	Data cannot be transferred or stored to the application
0x08000021	Data cannot be transferred or stored to the application because of local control
0x08000022	Data cannot be transferred or stored to the application because of the present device state
0x08000023	Object dictionary dynamic generation fails or no object dictionary is present (e.g. object dictionary is generated from file and generation fails because of an file error)

RETVAL (return value)

When the function has been executed successfully, the return value contains the valid length of the respond data: 1: BYTE, 2: WORD, 4: DWORD.

If an error occurs during function processing, the return value contains an error code.

Value	Description
0xF021	Invalid slave address (Call parameter equal 0 or above 127).
0xF022	Invalid Transfer type (Value unequal 60h, 61h).
0xF023	Invalid data length (data buffer to small, at SDO read access it should be at least 4byte, at SDO write access 1byte, 2byte or 4byte).
0xF024	The SFC is not supported.
0xF025	Write buffer in the CANopen master full, service cannot be processed at this time.
0xF026	Read buffer in the CANopen master full, service cannot be processed at this time.
0xF027	The SDO read or write access returned wrong answer, see CANopen Error Codes.
0xF028	SDO-Timeout (no CANopen participant with this Node-Id has been found).

BUSY

BUSY = 1: The read/write job is not yet completed.

DATABUFFER

Data area for SFC communication

Read SDO: Destination area for the SDO data that were read.

Write SDO: Source area for the SDO data that were written.

Note

Unless a SDO demand was processed error free, *RETVAL* contains the length of the valid response data in (1, 2 or 4byte) and the *CANOPENERROR* the value 0.

11x	21x	31x	51x
✓	✓	✓	✓

MMC Access - SFC 220...222

Overview

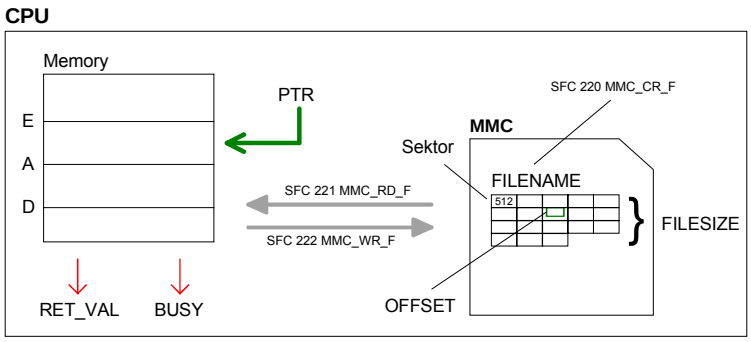
By means of these blocks there is the possibility to integrate MMC access to your application program. Here a new file may be created respectively an existing file may be opened for accessed when a MMC is plugged-in.
As long as you do not open another file, you may access this file via read/write commands.

Restrictions

For deploying the SFCs 220, 221 and 222, you have to regard the following restrictions:

- A read res. write access to the MMC is only possible after creation res. opening of the file via SFC 220.
- The data on MMC must not be fragmentized, for only complete data blocks may be read res. written.
- When transferring data to the MMC from an external reading device, they may be fragmentized, i.e. the data is divided into blocks. This may be avoided by formatting the MMC before the write access.
- At a write access from the CPU to the MMC, the data is always stored not fragmentized.
- When opening an already existing file, you have to use the same *FILENAME* and *FILESIZE* that you used at creation of this file.
- A MMC is structured into sectors. Every sector has a size of 512byte. Sector overlapping writing or reading is not possible. Access to sector overlapping data is only possible by using a write res. read command for every sector. By giving the offset, you define the according sector.

The following picture shows the usage of the single SFCs and their variables:



Note!

For read and write accesses to the MMC, you firstly have to open the file with SFC 220!

11x	21x	31x	51x
✓	✓	✓	✓

SFC 220 - MMC_CR_F

Overview

By means of this SFC a new file may be created respectively an existing file may be opened for accessed when a MMC is plugged-in.
As long as you do not open another file, you may access this file via read/write commands.

Note!
Since calling the SFC from the OB 1 can result in a cycle time-out, instead of this you should call the SFC from the OB 100.

Parameters

Name	Declaration	Type	Description
FILENAME	IN	STRING[254]	Name of file
FILESIZE	IN	DWORD	Size of file
RET_VAL	OUT	WORD	Return value (0 = OK)

FILENAME

Type in the file name used to store the data on the MMC. The name inclusive end ID may not exceed a maximum length of 13 characters:

- 8 characters for name
- 1 character for "."
- 3 characters for file extension
- 1 character 00h as end ID

Note!
For software technical reasons you have to enter 00h into the byte next to the file name (end ID of the file name).

FILESIZE

The *FILESIZE* defines the size of the user data in byte. When accessing an already existing file, it is mandatory to give not only the *FILENAME* but also the *FILESIZE*. The entry of a "Joker" length is not supported at this time.

Structure

Byte 0	Byte 1	Byte 2	Byte 3	...	Byte 255
Max. length	occupied length	ASCII value 1	ASCII value 2	...	ASCII value 254

RET_VAL (Return Value)

Word that returns a diagnostic/error message. 0 means OK.
See the table below for the concerning messages:

Value	Description
Diagnostic messages	
0000h	No errors (appears if new file is generated).
0001h	File already exists, is not fragmentized and the length value is identical or smaller.
8001h	No or unknown type of MMC is plugged-in.
Error messages	
8002h	No FAT on MMC found.
A001h	File name missing. This message appears if file name is inside a not loaded DB.
A002h	File name wrong (not 8.3 or empty)
A003h	File exists but *FILESIZE* too bigger than existing file.
A004h	File exists but is fragmentized and cannot be opened.
A005h	Not enough space on MMC.
A006h	No free entry in root directory. Depending on the used MMC there may be min. 16 up to max. 512 entries in the root directory.
B000h	An internal error occurred.

11x	21x	31x	51x
✓	✓	✓	✓

SFC 221 - MMC_RD_F

Description

Via the SFC 221 you may read data from a MMC. For read and write accesses to the MMC, you firstly have to open the file with SFC 220 and it has to be not fragmentized.

For more detailed information to this and to the restrictions see the description of the SFC 220.

Parameters

Name	Declaration	Type	Description
PTR	IN	ANY	Pointer to area for reading data
OFFSET	IN	DWORD	Offset of data within the file
BUSY	OUT	BOOL	Job state
RET_VAL	OUT	WORD	Return value (0 = OK)

PTR

This variable of the type pointer points to a data area in the CPU where the content of the MMC has to be written to.

OFFSET

Here you define the start address inside the file on the MMC from where on the data has to be transferred to the CPU.

BUSY

During data transfer this Bit remains set. The Bit is reset as soon as the data transfer is complete.

RET_VAL (Return Value)

Word that returns a diagnostic/error message.

0 means OK.

The following messages may be set:

Value	Description
0000h	No errors (data was read)
8001h	No or unknown type of MMC is plugged-in
8002h	No FAT found on MMC
9000h	Bit reading has been tried (Boolean variable). Bit reading is not possible.
9001h	Pointer value is wrong (e.g. points outside DB)
9002h	File length exceeded
9003h	Sector limit of 512 has been tried to overrun. Sector overrun reading is not possible.
B000h	An internal error occurred.

11x	21x	31x	51x
✓	✓	✓	✓

SFC 222 - MMC_WR_F

Description

Via the SFC 222, you may write to the MMC. For read and write accesses to the MMC, you firstly have to open the file with SFC 220 and it has to be not fragmentized.

For more detailed information to this and to the restrictions see the description of the SFC 220.

Parameters

Name	Declaration	Type	Description
PTR	IN	ANY	Pointer to area for writing data
OFFSET	IN	DWORD	Offset of data within the file
BUSY	OUT	BOOL	Job state
RET_VAL	OUT	WORD	Return value (0 = OK)

PTR

This variable of the type pointer points to a data area from where on the data starts that will be written to the MMC.

OFFSET

This defines the beginning of the data inside the file on the MMC where the data is written to.

BUSY

During data transfer this Bit remains set. The Bit is reset as soon as the data transfer is complete.

RET_VAL (Return Value)

Word that returns a diagnostic/error message.

0 means OK.

The following messages may be set:

Value	Description
0000h	No errors
8001h	No or unknown type of MMC is plugged-in.
8002h	No FAT found on MMC.
9000h	Bit writing has been tried (Boolean variable). Bit writing is not possible.
9001h	Pointer value is wrong (e.g. points outside DB).
9002h	File length exceeded
9003h	Sector limit of 512 has been tried to overrun. Sector overrun reading is not possible.
B000h	An internal error occurred.

11x	21x	31x	51x
✓			

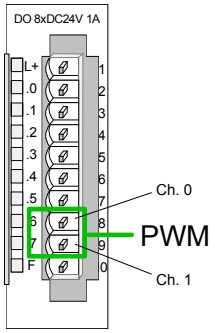
SFC 223 - PWM - Pulse duration modulation

Description

This block serves the parameterization of the pulse duration modulation for the last two output channels of X5.

Parameters

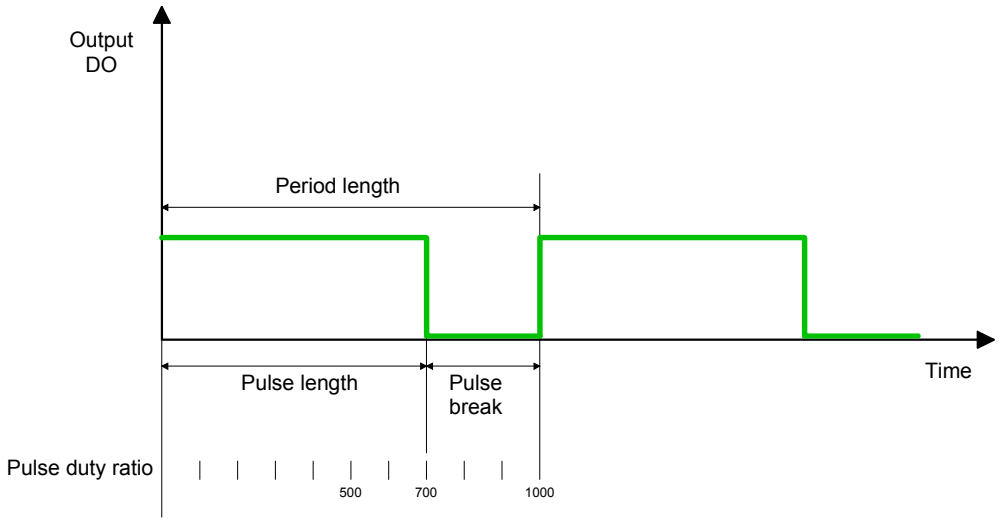
Name	Declaration	Type	Description
CHANNEL	IN	INT	Number of the output channel for PWM
ENABLE	IN	BOOL	Start bit of the job
TIMEBASE	IN	INT	Time base
PERIOD	IN	DINT	Period of the PWM
DUTY	IN	DINT	Output value per mille
MINLEN	IN	DINT	Minimum pulse duration
RET_VAL	OUT	WORD	Return value (0 = OK)



You define a time base, a period, the pulse duty ratio and min. pulse length. The CPU determines a pulse series with an according pulse/break relation and issues this via the according output channel.

The SFC returns a certain error code. You can see the concerning error messages in the table at the following page.

The PWM parameters have the following relationship:



Period length = time base x period

Pulse length = (period length / 1000) x pulse duty ratio

Pulse break = period length - pulse length

The parameters have the following meaning:

Channel

Define the output channel that you want to address.

Value range: 0 ... 1

ENABLE Via this parameter you may activate the PWM function (true) res. deactivate it (false).

Value range: true, false

TIMEBASE *TIMEBASE* defines the resolution and the value range of the pulse, period and minimum pulse length per channel.

You may choose the values 0 for 0.1ms and 1 for 1ms.

Value range: 0 ... 1

PERIOD Through multiplication of the value defined at period with the *TIMEBASE* you get the period length.

Value range: 0 ... 60000

DUTY This parameter shows the pulse duty ratio per mille. Here you define the relationship between pulse length and pulse break, concerned on one period.

1 per mille = 1 *TIMEBASE*

If the calculated pulse duration is no multiplication of the *TIMEBASE*, it is rounded down to the next smaller *TIMEBASE* limit.

Value range: 0 ... 1000

MINLEN Via *MINLEN* you define the minimal pulse length. Switches are only made, if the pulse exceeds the here fixed minimum length.

Value range: 0 ... 60000

RET_VAL (Return Value) Via the parameter *RET_VAL* you get an error number in return. See the table below for the concerning error messages:

Value	Description
0000h	no error
8005h	Parameter *MINLEN* outside the permissible range
8006h	Parameter *DUTY* outside the permissible range
8007h	Parameter *PERIOD* outside the permissible range
8008h	Parameter *TIMEBASE* outside the permissible range
8009h	Parameter *CHANNEL* outside the permissible range
9001h	Internal error: There was no valid address for a parameter.
9002h	Internal hardware error: Please call the VIPA-Service.
9003h	Output is not configured as PWM output respectively there is an error in hardware configuration.
9004h	HF-PWM was configured but SFC 223 was called (please use SFC 225 HF_PWM!).

11x	21x	31x	51x
✓			

SFC 224 - HSC - High-speed counter

Description This SFC serves for parameterization of the counter functions (**h**igh **s**peed **c**ounter) for the first 4 inputs.

Parameters

Name	Declaration	Type	Description
CHANNEL	IN	INT	Number of the input channel for HSC
ENABLE	IN	BOOL	Start bit of the job
DIRECTION	IN	INT	Direction of counting
PRESETVALUE	IN	DINT	Preset value
LIMIT	IN	DINT	Limit for counting
RET_VAL	OUT	WORD	Return value (0 = OK)
SETCOUNTER	IN_OUT	BOOL	Load preset value

CHANNEL Type the input channel that you want to activate as counter.

Value range: 0 ... 3

ENABLE Via this parameter you may activate the counter (true) res. deactivate it (false).

Value range: true, false

DIRECTION Fix the counting direction.

Hereby is: 0: Counter is deactivated, means *ENABLE* = false

1: count up

2: count down

PRESETVALUE Here you may preset a counter content, that is transferred to the according counter via *SETCOUNTER* = true.

Value range: 0 ... FFFFFFFFh

LIMIT Via Limit you fix an upper res. lower limit for the counting direction (up res. down). When the limit has been reached, the according counter is set zero and started new. If necessary an alarm occurs.

Value range: 0 ... FFFFFFFFh

RET_VAL (Return Value)

Via the parameter *RET_VAL* you get an error number in return. See the table below for the concerning error messages:

Value	Description
0000h	No error
8002h	The chosen channel is not configured as counter . (Error in the hardware configuration)
8008h	Parameter *DIRECTION* outside the permissible range
8009h	Parameter *CHANNEL* outside the permissible range
9001h	Internal error There was no valid address for a parameter.
9002h	Internal hardware error Please call the VIPA-Service.

SETCOUNTER

Per *SETCOUNTER* = true the value given by *PRESETVALUE* is transferred into the according counter.

The bit is set back from the SFC.

Value range: true, false

11x	21x	31x	51x
✓			

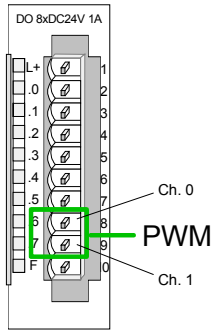
SFC 225 - HF_PWM - HF pulse duration modulation

Description

This block serves the parameterization of the pulse duration modulation for the last two output channels. This block is function identical to SFC 223. Instead of *TIMEBASE* and *PERIOD*, the SFC 225 works with a predefined frequency (up to 50kHz).

Parameters

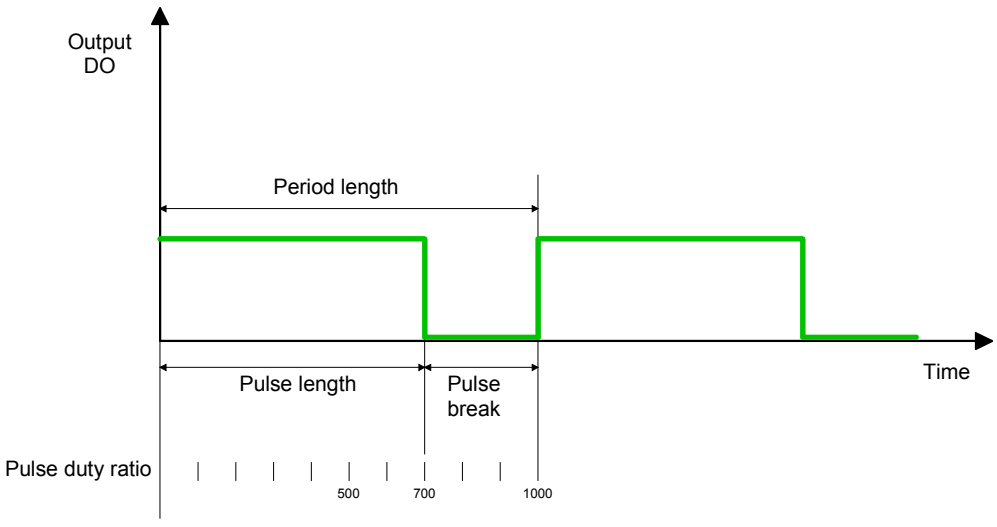
Name	Declaration	Type	Description
CHANNEL	IN	INT	Number of the output channel for HF-PWM
ENABLE	IN	BOOL	Start bit of the job
FREQUENCE	IN	WORD	Frequency of the HF-PWM
DUTY	IN	DINT	Pulse duty ratio per mille
MINLEN	IN	DINT	Minimum pulse duration
RET_VAL	OUT	WORD	Return value (0 = OK)



You define a frequency, the pulse duty ratio and min. pulse length. The CPU determines a pulse series with an according pulse/break relation and issues this via the according output channel.

The SFC returns a certain error code. You can see the concerning error messages in the table at the following page.

The PWM parameters have the following relationship:



Period length = 1 / frequency

Pulse length = (period length / 1000) x pulse duty ratio

Pulse break = period length - pulse length

CHANNEL Define the output channel that you want to address.

Value range: 0 ... 1

ENABLE Via this parameter you may activate the PWM function (true) res. deactivate it (false).

Value range: true, false

FREQUENCE Type in the frequency in Hz as hexadecimal value.

Value range: 09C4h ... C350h (2.5kHz ... 50kHz)

DUTY This parameter shows the pulse duty ratio per mille. Here you define the relationship between pulse length and pulse break, concerned on one period.

1 per mille = 1 Time base

If the calculated pulse duration is no multiplication of the time base, it is rounded down to the next smaller time base limit.

Value range: 0 ... 1000

MINLEN Via *MINLEN* you define the minimal pulse length in µs. Switches are only made, if the pulse exceeds the here fixed minimum length.

Value range: 0 ... 60000

RET_VAL (Return Value) Via the parameter *RET_VAL* you get an error number in return. See the table below for the concerning error messages:

Value	Description
0000h	no error
8005h	Parameter *MINLEN* outside the permissible range
8006h	Parameter *DUTY* outside the permissible range
8007h	Parameter *FREQUENCE* outside the permissible range
8008h	Parameter *TIMEBASE* outside the permissible range
8009h	Parameter *CHANNEL* outside the permissible range
9001h	Internal error: There was no valid address for a parameter
9002h	Internal hardware error: Please call the VIPA-Service.
9003h	Output is not configured as PWM output respectively there is an error in hardware configuration.
9004h	PWM was configured but SFC 225 was called (please use SFC 223 PWM!)

11x	21x	31x	51x
✓	✓	✓	✓

SFC 227 - TD_PRM - TD200 communication

Description

The SFC 227 supports the connection of the TD200 terminal from Siemens to the VIPA CPUs.

Please regard that you have to include a data block with the TD200 configuration data before calling the SFC 227. This data block may be created with the TDWizard from VIPA. This data block contains the general settings like language and display mode and the messages that are comfortably creatable with the TDWizard from VIPA. TDWizard is a part of WinPLC7 and is available from VIPA.

Parameters

The call of the SFC 227 specifies the terminal to communicate with. To call the SFC you have to transfer the following parameters:

Name	Declaration	Type	Description
MPI_ADDR	IN	BYTE	MPI address of TD
TD_STRUCT_PTR	IN	ANY	Pointer to beginning of TD's data (must be in DB)
FUNC_KEY_PTR	IN	ANY	Pointer to area for keys
OFFSET_INPUT	IN	BYTE	Forced values offset in input area
OFFSET_OUTPUT	IN	BYTE	Forced values offset in output area
RET_VAL	OUT	BYTE	Return value (0 = OK)

MPI_ADR

MPI address

Enter the MPI address of the connected TD200 terminal.

Parameter type: Byte

TD_STRUCT_PTR

Terminal Structure Pointer

Points to the start of the data block containing the parameterization and the text blocks of the terminal.

You may create the data block with TDWizard. This tool is a part of WinPLC7, the programming, test, diagnostic and simulation software from VIPA.

Parameter type: Pointer

Convenient range: DB

FUNC_KEY_PTR

Function Key Pointer

Points to the area where the status byte for hidden keys is stored.

Parameter type: Pointer

Convenient range: DB, M

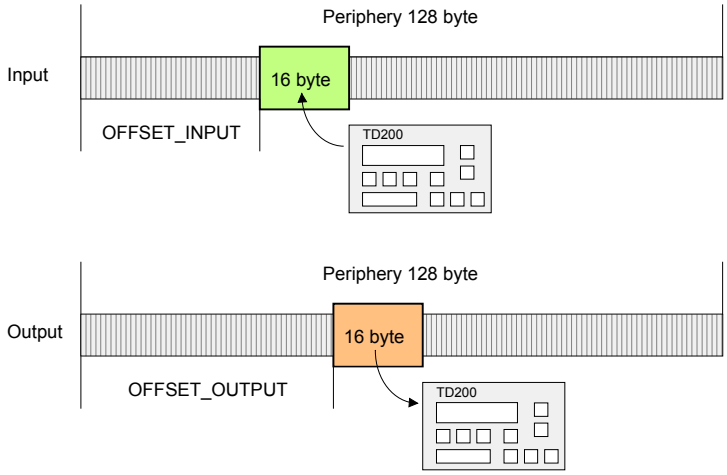
OFFSET_INPUT
OFFSET_OUTPUT

Offset Input, Offset Output

The terminal allows you to set in- and output byte. The TD200 from Siemens supports only a size of 16byte for in- and outputs.

The CPUs from VIPA enables the access to the complete process image (each 128byte for in- and outputs).

By setting an offset, you may overlay the periphery range of 128byte with a window of 16byte for in- and output. The following picture illustrates this:



RET_VAL (Return Value)

Type the bit memory byte (marker byte) where the resulting message should be stored.

For specification of the (error) messages see the table below.

Messages

Value	Description
00h	no error
10h	Error at *MPI_ADDR*
11h	*MPI_ADDR* contains MPI address of the CPU
12h	Value in *MPI_ADDR* exceeds max. MPI address
20h	Error in *OFFSET_INPUT*
21h	Value in *OFFSET_INPUT* is too high
30h	Error in *OFFSET_OUTPUT*
31h	Value in *OFFSET_OUTPUT* is too high
40h	Error in *TD_STRUCT_PTR*
41h	*TD_STRUCT_PTR* points not to a DB
50h	Error in *FUNC_KEY_PTR*
51h	Error in *FUNC_KEY_PTR*
52h	*FUNC_KEY_PTR* points not to an I, Q or M area
60h	An internal error occurred

11x	21x	31x	51x
	✓	✓	✓

SFC 228 - RW_KACHEL - Page frame direct access

Description

This SFC allows you the direct access to the page frame area of the CPU with a size of 4kbyte. The page frame area is divided into four page frames, each with a size of 1kbyte.

Setting the parameters page frame number, -offset and data width, the SFC 228 enables read and write access to an eligible page frame area.

Note!

This SFC has been developed for test purposes and for building-up proprietary communication systems and is completely at the user's disposal. Please regard that a write access to the page frame area influences a communication directly!

Parameters

Name	Declaration	Type	Description
K_NR	IN	INT	Page frame number
OFFSET	IN	INT	Page frame offset
R_W	IN	INT	Access
SIZE	IN	INT	Data width
RET_VAL	OUT	BYTE	Return value (0 = OK)
VALUE	IN_ OUT	ANY	Pointer to area of data transfer

K_NR

Page frame number

Type the page frame no. that you want to access.

Value range: 0 ... 3

OFFSET

Page frame offset

Fix here an offset within the specified page frame.

Value range: 0 ... 1023

R_W

Read/Write

This parameter specifies a read res. write access.

0 = read access, 1 = write access

SIZE

Size

The size defines the width of the data area fixed via *K_NR* and *OFFSET*. You may choose between the values 1, 2 and 4byte.

RET_VAL (Return Value)

Byte where an error message is returned to.

VALUE

In-/output area

This parameter fixes the in- res. output area for the data transfer.

At a read access, this area up to 4byte width contains the data read from the page frame area.

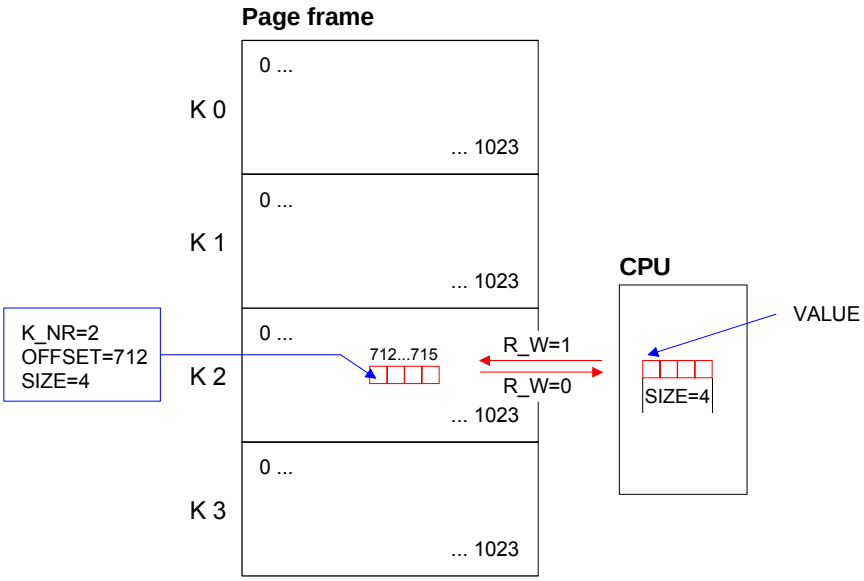
At a write access, the data up to 4byte width is transferred to the page frame area.

Parameter type: Pointer

Example

The following example shows the read access to 4byte starting with byte 712 in page frame 2. The read 4byte are stored in DB10 starting with byte 2. For this the following call is required:

```
CALL SFC 228
K_NR      :=2
OFFSET    :=712
R_W       :=0
SIZE      :=4
RET_VAL   :=MB10
VALUE     :=P#DB10.DBX 2.0 Byte 4
```



Error messages

Value	Description
00h	no error
01h ... 05h	Internal error: No valid address found for a parameter
06h	defined page frame does not exist
07h	parameter SIZE ≠ 1, 2 or 4 at read access
08h	parameter SIZE ≠ 1, 2 or 4 at write access
09h	parameter R_W ≠ 0 or 1

11x	21x	31x	51x
	✓	✓	✓

Page frame communication - SFC 230 ... 238

Overview

The delivered handling blocks allow the deployment of communication processors in the CPUs from VIPA.

The handling blocks control the complete data transfer between CPU and the CPs.

Advantages of the handling blocks:

- you loose only few memory space for user application
- short runtimes of the blocks

The handling blocks don't need:

- bit memory area
- time areas
- counter areas

Parameter description

All handling blocks described in the following use an identical interface to the user application with these parameters:

SSNR:	Interface number
ANR:	Order number
ANZW:	Indicator word (double word)
IND:	Indirect fixing of the relative start address of the data source res. destination
QANF/ZANF:	Relative start address within the type
PAFE:	Parameterization error
BLGR:	Block size

A description of these parameters follows on the next pages.

SSNR Interface number

Number of the logical interface (page frame address) to which the according order refers to.

Parameter type : Integer

Convenient range : 0 ... 255

ANR Job number

The called job number for the logical interface.

Parameter type : Integer

Convenient range : 1 ... 223

ANZW Indicator word (double word)

Address of the indicator double word in the user memory where the processing of the order specified under ANR is shown.

Parameter type : Double word

Convenient range : DW or MW; use either DW and DW+1 or MW and MW+2

The value DW refers to the data block opened before the incoming call or to the directly specified DB.

IND Kind of parameterization (direct, indirect)

This parameter defines the kind of data on which the pointer *QANF* points.

0: *QANF* points directly to the initial data of the source res. destination data.

1: the pointer *QANF/ZANF* points to a memory cell, from where on the source res. destination data are defined (indirect).

2: the pointer *QANF/ZANF* points to a memory area where the source res. destination information lies (indirect).

5: the pointer *QANF/ZANF* points to a memory cell, from where on the source res. destination data and parameters of the indicator word are defined (indirect).

6: the pointer *QANF/ZANF* points to a memory area where the source res. destination data and parameters of the indicator word are laying (indirect).

Parameter type : Integer

Convenient entries : 0, 1, 2, 5, 6

Note!

Please regard, that at *IND* = 5 res. *IND* = 6, the parameter *ANZW* is ignored!

QANF/ZANF

Relative start address of the data source res. destination and at *IND* = 5 res. *IND* = 6 of the indicator word.

This parameter of the type "pointer" (Any-Pointer) allows you fix the relative starting address and the type of the data source (at SEND) res. the data destination (at RECEIVE).

At *IND* = 5 res. *IND* = 6 the parameters of the indicator word are also in the data source.

Parameter type : Pointer

Convenient range : DB, M, Q, I

Example:
```
P#DB10.DBX0.0 BYTE 16
P#M0.0 BYTE 10
P#E 0.0 BYTE 8
P#A 0.0 BYTE 10
```

BLGR

Block size

During the boot process the stations agree about the block size (size of the data blocks) by means of SYNCHRON.

A high block size = high data throughput but longer run-times and higher cycle load.

A small block size = lower data throughput but shorter run-times of the blocks.

These block sizes are available:

Value	*Block size*	*Value*	*Block size*
0	Default (64byte)	4	128byte
1	16byte	5	256byte
2	32byte	6	512byte
3	64byte	255	512byte

Parameter type : Integer

Convenient range : 0 ... 255

PAFE

Error indication at parameterization defects

This "BYTE" (output, marker) is set if the block detects a parameterization error, e.g. interface (plug-in) not detected or a non-valid parameterization of QUANF/ZANF.

Parameter type : Byte

Convenient range : OB 0 ... OB127, MB 0...MB 255

Page frame communication - Parameter transfer

Direct/indirect parameterization

A handling block may be parameterized directly or indirectly. Only the "*PAFE*" parameter must always been set directly.

When using the direct parameterization, the handling block works off the parameters given immediately with the block call.

When using the indirect parameterization, the handling block gets only pointers per block parameters. These are pointing to other parameter fields (data blocks or data words).

The parameters *SSNR*, *ANR*, *IND* and *BLGR* are of the type "integer", so you may parameterize them indirectly.

Example for direct and indirect parameter transfer

Direct parameter transfer

```
CALL  SFC  230
       SSNR:=0
       ANR :=3
       IND :=0
       QANF:=P#A 0.0 BYTE 16
       PAFE:=MB79
       ANZW:=MD44
```

Indirect parameter transfer

Please note that you have to load the bit memory words with the corresponding values before.

```
CALL  SFC  230
       SSNR:=MW10
       ANR :=MW12
       IND :=MW14
       QANF:=P#DB10.DBX0.0 BYTE 16
       PAFE:=MB80
       ANZW:=MD48
```

The direct res. indirect transfer of the source and destination parameters is described in the following section.

Page frame communication - Source res. destination definition

Outline

You have the possibility to set the entries for source, destination and *ANZW* directly or store it indirectly in a block to which the *QANF* / *ZANF* res. *ANZW* pointer points.

The parameter *IND* is the switch criterion between direct and indirect parameterization.

Direct parameterization of source and destination details (*IND* = 0)

With *IND* = 0 you fix that the pointer *QANF* / *ZANF* shows directly to the source res. destination data.

The following table shows the possible *QANF* / *ZANF* parameters at the direct parameterization:

QTYP/ZTYP Description	Data in DB	Data in MB	Data in OB Process image of the outputs	Data in IB Process image of the inputs
Pointer: Example:	P#DB**a**.DBX **b**.0 BYTE **C** P#DB10.DBX 0.0 BYTE 8	P#M **b**.0 BYTE **c** P#M 5.0 BYTE 10	P#O **b**.0 BYTE **c** P#O 0.0 BYTE 2	P#I **b**.0 BYTE **c** P#I 20.0 BYTE 1
DB, MB, OB, IB Definition	P#DBa "a" means the DB-No., from where the source data is fetched or where to the destination data is transferred.	P#M The data is stored in a MB.	P#O The data is stored in the output byte.	P#I The data is stored in the input byte.
Valid range for "a"	0 ... 32767	irrelevant	irrelevant	irrelevant
Data / Marker Byte, OB, IB Definition	DB-No., where data fetch or write starts.	Bit memory byte no., where data fetch or write starts.	Output byte no., where data fetch or write starts.	Input byte no., where data fetch or write starts.
Valid range for "b"	0.0 ... 2047.0	0 ... 255	0 ... 127	0 ... 127
BYTE c Definition	Length of the Source/ Destination data blocks in Words.	Length of the Source/ Destination data blocks in bytes.	Length of the Source/ Destination data blocks in bytes.	Length of the Source/ Destination data blocks in bytes.
Valid range for "c"	1 ... 2048	1 ... 255	1 ... 128	1 ... 128

Indirect parameterization of source and destination details (*IND* = 1 or *IND* = 2)

Indirect addressing means that *QANF* / *ZANF* points to a memory area where the addresses of the source res. destination areas and the indicator word are stored.

In this context you may either define one area for data source, destination and indicator word (*IND* = 1) or each, data source, data destination and the indicator word, get an area of their own (*IND* = 2).

The following table shows the possible *QANF* / *ZANF* parameters for indirect parameterization:

QTYP/ZTYP Description	IND = 1	IND = 2
Definition	Indirect addressing for source **or** destination parameters. The source or destination parameters are stored in a DB. QANF/ZANF DW +0 Data type source +2 DB-Nr. at type "DB", otherwise irrelevant +4 Start address +6 Length in Byte	Indirect addressing for source **and** destination parameters. The source and destination parameters are stored in a DB in a sequential order. QANF/ZANF DW +0 Data type source +2 DB-Nr. at type "DB", otherwise irrelevant (Description data source) +4 Start address +6 Length in Byte +8 Data type destin. +10 DB-Nr. at type "DB", otherwise irrelevant (Description data destination) +12 Start address +14 Length in Byte
valid DB-No.	0 ... 32767	0 ... 32767
Data word Definition	DW-No., where the stored data starts	DW-No., where the stored data starts
Valid range	0.0 ... 2047.0	0.0 ... 2047.0
Length Definition	Length of the DBs in byte	Length of the DBs in byte
Valid range	8 fix	16 fix

Indirect parameterization of source and destination details and ANZW (*IND* = 5 or *IND* = 6)

Indirect addressing means that *QANF* / *ZANF* points to a memory area where the addresses of the source res. destination areas and the indicator word are stored.

In this context you may either define one area for data source, destination and indicator word (*IND* = 5) or each, data source, data destination and the indicator word, get an area of their own (*IND* = 6).

The following table shows the possible *QANF* / *ZANF* parameters for indirect parameterization:

QTYP/ZTYP Description	IND=5	IND=6
Definition	Indirect addressing for source **or** destination parameters and indicator word (ANZW). The source or destination parameters and ANZW are stored in a DB in a sequential order. QANF/ZANF DW +0 Data type source +2 DB-Nr. at type "DB", otherwise irrelevant +4 Start address +6 Length in Byte (Description data source/ destination) +8 Data type destin. +10 DB-Nr. at type "DB", otherwise irrelevant +12 Start address (Description indicator word)	Indirect addressing for source **and** destination parameters and indicator word (ANZW). The source and destination parameters and ANZW are stored in a DB in a sequential order. QANF/ZANF DW +0 Data type source +2 DB-Nr. at type "DB", otherwise irrelevant +4 Start address +6 Length in Byte (Description data source) +8 Data type destin. +10 DB-Nr. at type "DB", otherwise irrelevant +12 Start address +14 Length in Byte (Description data destination) +16 Data type source +18 DB-Nr. at type "DB", otherwise irrelevant +20 Start address (Description indicator word)
valid DB-No.	0 ... 32767	0 ... 32767
Data word Definition	DW-No., where the stored data starts	DW-No., where the stored data starts
Valid range	0.0 ... 2047.0	0.0 ... 2047.0
Length Definition	Length of the DBs in byte	Length of the DBs in byte
Valid range	14 fix	22 fix

Page frame communication - Indicator word ANZW

Status and error reports

Status and error reports are created by the handling blocks:

- by the indicator word *ANZW* (information at order commissioning).
- by the parameter error byte *PAFE* (indication of a wrong order parameterization).

Content and structure of the indicator word *ANZW*

The "Indicator word" shows the status of a certain order on a CP.

In your PLC program you should keep one indicator word for each defined order at hand.

The indicator word has the following structure:

Byte	Bit 7 ... Bit 0
0	Bit 3 ... Bit 0: Error management CPU 0: no error 1 ... 5: CPU-Error 6 ... 15: CP-Error Bit 7 ... Bit 4: reserved
1	*State management CPU* Bit 0: Handshake convenient (data exists) 0: RECEIVE blocked 1: RECEIVE released Bit 1: order commissioning is running 0: SEND/FETCH released 1: SEND/FETCH blocked Bit 2: Order ready without errors Bit 3: Order ready with errors *Data management handling block* Bit 4: Data receive/send is running Bit 5: Data transmission active Bit 6: Data fetch active Bit 7: Disable/Enable data block 0: released 1: blocked
2 ... 3	Length word handling block

In the "length word" the handling blocks (SEND, RECEIVE) store the data that has already been transferred, i.e. received data in case of a Receive order, send data when there is a Send order.

The announcement in the "length word" is always in byte and absolute.

Error management Byte 0, Bit 0 ... Bit 3

Those bits announce the error messages of the order. The error messages are only valid if the bit "Order ready with error" in the status bit is set simultaneously.

The following error messages may occur:

0 **no error**

If the bit "Order ready with error" is set, the CP had to reinitialize the connection, e.g. after a reboot or RESET.

1 **wrong Q/ZTYP at HTB**

The order has been parameterized with the wrong type label.

2 **AG area not found**

The order impulse had a wrong parameterized DB-No.

3 **AG area too small**

Q/ZANF and Q/ZLAE overwrite the range boundaries. Handling with data blocks the range boundary is defined by the block size. With flags, timers, counters etc. the range size depends on the AG.

4 **QVZ-Error in the AG**

This error message means, that you chose a source res. destination parameter of the AG area, where there is either no block plugged in or the memory has a defect. The QVZ error message can only occur with the type Q/ZTYP AS, PB, QB or memory defects.

5 **Error at indicator word**

The parameterized indicator word cannot be handled. This error occurs, if ANZW declared a data word res. double word, that is not (any more) in the specified data block, i.e. DB is too small or doesn't exist.

6 **no valid ORG-Format**

The data destination res. source isn't declared, neither at the handling block (Q/TYP="NN") nor at the coupler block.

7 **Reserved**

8 **no available transfer connections**

The capacity for transfer connections is at limit. Delete unnecessary connections.

9 **Remote error**

There was an error at the communication partner during a READ/WRITE-order.

A **Connection error**

The connection is not (yet) established. The message disappears as soon as the connection is stable. If all connections are interrupted, please check the block itself and the bus cable. Another possibility for the occurrence of this error is a wrong parameterization, like e.g. inconsistent addressing.

B **Handshake error**

This could be a system error or the size of the data blocks has been defined out of range.

C **Initial error**

The wrong handling block tried to initialize the order or the size of the given data block was too large.

D **Cancel after RESET**

This is a normal system message. With PRIO 1 and 2 the connection is interrupted but will be established again, as soon as the communication partner is online. PRIO 3 connections are deleted, but can be initialized again.

E **Order with basic load function**

This is a normal system message. This order is a READ/WRITE-PASSIV and can not be started from the AG.

F **Order not found**

The called order is not parameterized on the CP. This error may occur when the SSNR/A-No. combination in the handling block is wrong or no connection block is entered.

The bits 4 to 7 of byte 2 are reserved for extensions.

Status management Byte 1, Bit 0 ... Bit 3

Here you may see if an order has already been started, if an error occurred or if this order is blocked, e.g. a virtual connection doesn't exist any longer.

Bit 0 **Handshake convenient**

Set:	Per plug-in according to the "delete"-announcement in the order status bit: Handshake convenient (=1) is used at the RECEIVE block (telegram exists at PRIO 1 or RECEIVE impulse is possible at PRIO 2/3)
Analyze:	Per RECEIVE block: The RECEIVE initializes the handshake with the CP only if this bit is set.
	Per application: for RECEIVE request (request a telegram at PRIO 1).

Bit 1 **Order is running**

Set:	Per plug-in: when the CP received the order.
Delete:	Per plug-in: when an order has been commissioned (e.g. receipt received).
Analyze:	Per handling blocks: A new order is only send, when the order before is completely commissioned.
	Per user: when you want to know, if triggering a new order is convenient.

Bit 2 **Order ready without errors**

Set: Per plug-in: when the according order has been commissioned without errors.

Delete: Per plug-in: when the according order is triggered for a second time.

Analyze: Per user: to proof that the order has been commissioned without errors.

Bit 3 **Order ready with errors**

Set: Per plug-in: when the according order has been commissioned with errors. Error causes are to find encrypted in the high-part of the indicator word.

Delete: Per plug-in: when the according order is triggered for a second time.

Analyze: Per user: to proof that the order has been commissioned with errors. If set, the error causes are to find in the high-byte of the indicator word.

Data management Byte 1, Bit 4 ... Bit 7

Here you may check if the data transfer is still running or if the data fetch res. transmission is already finished. By means of the bit "Enable/Disable" you may block the data transfer for this order (Disable = 1; Enable = 0).

Bit 4 **Data fetch / Data transmission is active**

Set: Per handling block SEND or RECEIVE, if the fetch/transmission has been started, e.g. when data is transferred with the ALL-function (DMA-replacement), but the impulse came per SEND-DIRECT.

Delete: Per handling blocks SEND or RECEIVE, if the data transfer of an order is finished (last data block has been transferred).

Analyze: Per user: During the data transfer CP <<->>AG the user must not change the record set of an order. This is uncritical with PRIO 0/1 orders, because here the data transfer is realizable in one block cycle. Larger data amounts however are transferred in blocks during more AG cycles. To ensure data consistency you should proof that the data block isn't in transfer any more before you change the content!

BIT 5 **Data transmission is active**

Set:	Per handling block SEND, when the data transition for an order is ready.
Delete:	Per handling block SEND, when the data transfer for a new order has been started (new trigger). Per user: When analysis is ready (flank creation).
Analyze:	Per user: Here you may ascertain, if the record set of an order has already been transferred to the CP res. at which time a new record set concerning a running order (e.g. cyclic transition) may be started.

Bit 6 **Data fetch active**

Set:	Per RECEIVE, when data fetch for a new order has been finished.
Delete:	Per RECEIVE, when data transfer to AG for a new order (new trigger) has been started. Per user, when analyzing (edge creation).
Analyze:	Per user: Here you may ascertain, if the record set of an order has already been transferred to the CP res. at what time a new record set for the current order has been transferred to the AG.

Bit 7 **Disable/Enable data block**

Set:	Per user: to avoid overwriting an area by the RECEIVE block res. data transition of an area by the SEND block (only for the first data block).
Delete:	Per user: to release the according data area.
Analyze:	Per handling blocks SEND and RECEIVE: if Bit 7 is set, there is no data transfer anymore, but the blocks announce an error to the CP.

Length word Byte 2 and Byte 3

In the length word the handling blocks (SEND, RECEIVE) store the already transferred data of the current order, i.e. the received data amount for receiving orders, the sent data amount for sending orders.

Describe: Per SEND, RECEIVE during the data transfer. The length word is calculated from:
current transfer amount + amount of already transferred data

Delete: Per overwrite res. with every new SEND, RECEIVE, FETCH.

If the bit "order ready without error" res. "Data fetch/data transition ready" is set, the "Length word" contains the current source res. destination length.

If the bit "order ready with error" is set, the length word contains the data amount transferred before the failure occurred.

Status and error reports

Important status and error reports of the CPU

The following section lists important status and error messages that can appear in the "Indicator word". The representation is in "HEX" patterns. The literal X means "not declared" res. "irrelevant"; No. is the error number.

Possible indicator words

Indicator word: X F X A

The error index "F" shows, that the according order is not defined on the CP. The state index "A" causes a block of this order (for SEND/FETCH and RECEIVE).

Indicator word: X A X A

The error index "A" shows that the connection of the communication order is not (yet) established. Together with the state index "A" SEND, RECEIVE and FETCH are blocked.

Indicator word: X 0 X 8

The connection has been established again (e.g. after a CP reboot), the SEND order is released (SEND-communication order).

Indicator word: X 0 X 9

The connection has been established again, the RECEIVE order is released (RECEIVE-communication order).

Indicator word: X 0 2 4

SEND has been worked off without errors, the data was transferred.

Indicator word: X 0 4 5

RECEIVE was successful, the data arrived at the AG.

Indicator word: X 0 X 2

The SEND-, RECEIVE-, READ- res. WRITE order is still running. At SEND the partner is not yet ready for RECEIVE or vice versa.

Important indicator word states

The following table shows the most important indicator word states:

Messages at SEND

State at H1	Prio 0/1	Prio 2	Prio 3/4
State at TCP/IP	Prio 1	Prio 2	Prio 3
after reboot	0 A 0 A	0 A 0 A	0 0 0 8
after connection start	X 0 X 8	X 0 X 8	.....
after initial impulse	X 0 X 2	X 0 X 2	X 0 X 2
ready without error	X 0 2 4	X 0 2 4	X 0 2 4
ready with error	X No X 8	X No X 8	X No X 8
after RESET	X D X A	X D X A	X D X 8

Messages at RECEIVE

State at H1	Prio 0/1	Prio 2	Prio 3/4
State at TCP/IP	Prio 1	Prio 2	Prio 3
after reboot	0 A 0 A	0 A 0 A	0 0 0 1
after connection start	X 0 X 4	X 0 0 9	.....
after initial impulse	X 0 X 2	X 0 X 2	X 0 X 2
Telegram received	X 0 X 1	.....	.....
ready without error	X 0 4 1	X 0 4 5	X 0 4 5
ready with error	X No X 8	X No X 9	X No X 9
after RESET	X D X A	X D X A	X D X 9

Messages at READ/WRITE-ACTIVE

State at H1	Prio 0/1	Prio 2	Prio 3/4
State at TCP/IP	Prio 1	Prio 2	Prio 3
after reboot		0 A 0 A	
after connection start		X 0 0 8	
after initial impulse		X 0 X 2	
READ ready		X 0 4 4	
WRITE ready		X 0 2 4	
ready with error		X No X 8	
after RESET		X D X A	

Page frame communication - Parameterization error PAFE

PAFE

The parameterization error byte *PAFE* is set (output or bit memory), when the block detects a "parameterization error", e.g. there is no interface or there is an invalid parameterization of *QANF* / *ZANF*.

PAFE has the following structure:

Byte	Bit 7 ... Bit 0
0	Bit 0: error 0: no error 1: error, error-No. in Bit 4 to Bit 7 Bit 3 ... Bit 1: reserved Bit 7 ... Bit 4: error number 0: no error 1: wrong ORG-Format 2: area not found (DB not found) 3: area too small 4: QVZ-error 5: wrong indicator word 6: no Source-/Destination parameters at SEND/RECEIVE ALL 7: interface not found 8: interface not specified 9: interface overflow A: reserved B: invalid order-No. C: interface of CP doesn't quit or is negative D: Parameter BLGR not allowed E: reserved F: reserved

11x	21x	31x	51x
	✓	✓	✓

SFC 230 - SEND

Description

The SEND block initializes a send order to a CP.

Normally SEND is called in the cyclic part of the user application program. Although the insertion of this block into the interrupt or the time-alarm program part is possible, the indicator word (*ANZW*), however, may not be updated cyclically. This should be taken over by a CONTROL block.

The connection initialization with the CP for data transmission and for activating a SEND impulse is only started, if:

- the FB RLO (result of operation) received "1".
- the CP released the order. (Bit "order active" in ANZW = 0).

During block stand-by, only the indicator word is updated.

Parameters

Name	Declaration	Type	Description
SSNR	IN	INT	Interface number
ANR	IN	INT	Job number
IND	IN	INT	Mode of addressing
QANF	IN	ANY	Pointer to data source
PAFE	OUT	BYTE	Parameterization error
ANZW	IN_OUT	DWORD	Indicator word

SEND_ALL for data transmission

If the CP is able to take over the data directly, the SEND block transfers the requested data in one session. If the CP requests only the order parameters or the amount of the depending data is too large, the CP only gets the sending parameters res. the parameter with the first data block. The according data res. the assigned serial blocks for this order are requested from the CP by SEND_ALL to the CPU. For this it is necessary that the block SEND_ALL is called minimum one time per cycle.

The user interface is for all initialization types equal, only the transfer time of the data is postponed for minimum one CPU cycle.

11x	21x	31x	51x
	✓	✓	✓

SFC 231 - RECEIVE

Description

The RECEIVE block receives data from a CP.

Normally the RECEIVE block is called in the cyclic part of the user application program. Although the insertion of this block into the interrupt or the waking program part is possible, the indicator word cannot be updated cyclically. This should be taken over by a CONTROL block.

The handshake with the CP (order initialization) and for activating a RECEIVE block is only started, if

- the FB RLO received "1".
- the CP released the order (Bit "Handshake convenient" = 1).

Parameters

Name	Declaration	Type	Description
SSNR	IN	INT	Interface number
ANR	IN	INT	Job number
IND	IN	INT	Mode of addressing
ZANF	IN	ANY	Pointer to data destination
PAFE	OUT	BYTE	Parameterization error
ANZW	IN_OUT	DWORD	Indicator word

If the block runs in stand-by only the indicator word is updated.

The RECEIVE block reacts different depending from the kind of supply and the CP reaction:

- If the CP transmits a set of parameters although the RECEIVE block itself got destination parameters, the parameter set of the block has the priority above those of the CP.
- Large amounts of data can only be transmitted in blocks. Therefore you have to transmit the assigned serial blocks by means of RECEIVE_ALL to the CPU. It is necessary that the block RECEIVE_ALL is called minimum one time per application cycle and CP interface, if you want to transmit larger data amounts. You also have to integrate the RECEIVE_ALL cyclically, if the CP only uses the RECEIVE for releasing a receipt telegram and the data is transmitted via the background communication of the CPU.

11x	21x	31x	51x
	✓	✓	✓

SFC 232 - FETCH

Description

The FETCH block initializes a FETCH order in the partner station.

The FETCH order defines data source and destination and the data source is transmitted to the partner station.

The CPU from VIPA realizes the definition of source and destination via a pointer parameter.

The partner station provides the *Source* data and transmits them via SEND_ALL back to the requesting station. Via RECEIVE_ALL the data is received and is stored in *Destination*.

The update of the indicator word takes place via FETCH res. CONTROL.

The handshake for initializing FETCH is only started, if

- the FB RLO receives "1".
- the function has been released in the according CP indicator word (order active = 0).

Parameters

Name	Declaration	Type	Description
SSNR	IN	INT	Interface number
ANR	IN	INT	Job number
IND	IN	INT	Mode of addressing
ZANF	IN	ANY	Pointer to data destination
PAFE	OUT	BYTE	Parameterization error
ANZW	IN_OUT	DWORD	Indicator word

Note!

More information for indirect parameterization you will find in this chapter in "**Page frame communication - Source res.** ".

11x	21x	31x	51x
	✓	✓	✓

SFC 233 - CONTROL

Description

The purpose of the CONTROL block is the following:

- Update of the indicator word
- Query if a certain order of the CP is currently "active", e.g. request for a receipt telegram
- Query the CP which order is recently in commission

The CONTROL block is not responsible for the handshake with the CP, it just transfers the announcements in the order status to the parameterized indicator word. The block is independent from the RLO and should be called from the cyclic part of the application.

Parameters

Name	Declaration	Type	Description
SSNR	IN	INT	Interface number
ANR	IN	INT	Job number
PAFE	OUT	BYTE	Parameterization error
ANZW	IN_OUT	DWORD	Indicator word

ANR

If *ANR* ≠ 0, the indicator word is built up and handled equal to all other handling blocks.

If the parameter *ANR* gets 0, the CONTROL command transmits the content of the order state cell 0 to the LOW part of the indicator words.

The order state cell 0 contains the number of the order that is in commission, e.g. the order number of a telegram (set by the CP).

11x	21x	31x	51x
	✓	✓	✓

SFC 234 - RESET

Description

The RESET ALL function is called via the order number 0. This resets all orders of this logical interface, e.g. deletes all order data and interrupts all active orders.

With a direct function (*ANR* ≠ 0) only the specified order will be reset on the logical interface.

The block depends on the RLO and may be called from cyclic, time or alarm controlled program parts.

Parameters

Name	Declaration	Type	Description
SSNR	IN	INT	Interface number
ANR	IN	INT	Job number
PAFE	OUT	BYTE	Parameterization error

Operating modes

The block has two different operating modes:

- RESET ALL
- RESET DIRECT

11x	21x	31x	51x
	✓	✓	✓

SFC 235 - SYNCHRON

Description

The SYNCHRON block initializes the synchronization between CPU and CP during the boot process. For this it has to be called from the starting OBs. Simultaneously the transition area of the interface is deleted and predefined and the CP and the CPU agree about the block size.

Parameters

Name	Declaration	Type	Description
SSNR	IN	INT	Interface number
BLGR	IN	INT	Block size
PAFE	OUT	BYTE	Parameterization error

Block size

To avoid long cycle run-times it is convenient to split large data amounts into smaller blocks for transmitting them between CP and CPU. You declare the size of these blocks by means of "block size".

A large block size = high data throughput, but also longer run-times and therefore a high cycle time strain.

A small block size = smaller data throughput, but also shorter run-times of the blocks.

Following block sizes are available:

Value	*Block size*	*Value*	*Block size*
0	Default (64byte)	4	128byte
1	16byte	5	256byte
2	32byte	6	512byte
3	64byte	255	512byte

Parameter type: Integer

Valid range: 0 ... 255

11x	21x	31x	51x
	✓	✓	✓

SFC 236 - SEND_ALL

Description

Via the SEND_ALL block, the data is transmitted from the CPU to the CP by using the declared block size.

Location and size of the data area that is to transmit with SEND_ALL, must be declared before by calling SEND res. FETCH.

In the indicator word that is assigned to the concerned order, the bit "Enable / Disable" is set, "Data transmission starts" and "Data transmission running" is calculated or altered.

Parameters

Name	Declaration	Type	Description
SSNR	IN	INT	Interface number
PAFE	OUT	BYTE	Parameterization error
ANZW	IN_OUT	DWORD	Indicator word

ANZW

In the indicator word of the block, the indicator word, that is parameterized in the SEND_ALL block, the current order number is stored (0 means stand-by).

The amount of the transmitted data for one order is shown in the data word of SEND_ALL which follows the indicator word.

Note!

In the following cases, the SEND_ALL command has to be called for minimum one time per cycle of the block OB 1:

- if the CP is able to request data from the CPU independently.
- if a CP order is initialized via SEND, but the CP still has to request the background communication data of the CPU for this order.
- if the amount of data, that should be transmitted by this SEND to the CP, is higher than the declared block size.

11x	21x	31x	51x
	✓	✓	✓

SFC 237 - RECEIVE_ALL

Description

Via the RECEIVE_ALL block, the data received from the CP is transmitted from the CP to the CPU by using the declared block size.

Location and size of the data area that is to transmit with RECEIVE_ALL, must be declared before by calling RECEIVE.

In the indicator word that is assigned to the concerned order, the bit "ENABLE/DISABLE" is set, "Data transition starts" and "Data transition/fetch running" is analyzed or altered. The receiving amount is shown in the following word.

Parameters

Name	Declaration	Type	Description
SSNR	IN	INT	Interface number
PAFE	OUT	BYTE	Parameterization error
ANZW	IN_OUT	DWORD	Indicator word

ANZW

In the indicator word of the block, the indicator word that is parameterized in the RECEIVE_ALL block, the current order number is stored. In the stand-by running mode of RECEIVE_ALL the block indicator word is deleted.

Note!

In the following cases, the RECEIVE_ALL command has to be called for minimum one time per cycle of the block OB 1:

- if the CP should send data to the CPU independently.
- if a CP order is initialized via RECEIVE, but the CP still has to request the "background communication" data of the CPU for this order.
- if the amount of data that should be transmitted to the CPU by this RECEIVE, is higher than the declared block size.

11x	21x	31x	51x
	✓	✓	✓

SFC 238 - CTRL1

Description

This block is identical to the CONTROL block SFC 233 except that the indicator word is of the type Pointer and that it additionally includes the parameter IND, reserved for further extensions.

The purpose of the CONTROL block is the following:

- Update of the indicator word.
- Query if a certain order of the CP is currently active, e.g. request for a receipt telegram
- Query the CP which order is recently in commission

The CONTROL block is not responsible for the handshake with the CP; it just transfers the announcements in the order status to the parameterized indicator word. The block is independent from the RLO and should be called from the cyclic part of the application.

Parameters

Name	Declaration	Type	Description
SSNR	IN	INT	Interface number
ANR	IN	INT	Job number
IND	IN	INT	Reserved
PAFE	OUT	BYTE	Parameterization error
ANZW	IN_OUT	DWORD	Indicator word

ANR

If *ANR* ≠ 0, the indicator word is built up and handled equal to all other handling blocks.

If the parameter *ANR* gets 0, the CONTROL command transmits the content of the order state cell 0 to the LOW part of the indicator words.

The order state cell 0 contains the number of the order that is in commission, e.g. the order number of a telegram (set by the CP).

IND

The parameter *IND* has no functionality at this time and is reserved for further extensions.

ANZW

The indicator word *ANZW* is of the type Pointer. This allows you to store the indicator word in a data block.