VIPA SPEED7

OPL_SP7 | Operation List | Manual

HB 00 | OPL_SP7 | Operation List | GB | Rev. 16-36

A YASKAWA COMPANY

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

Table of contents

1 IL operations

1.1 Overview

The following canter lists the available commands of the SPEED7 CPUs from VIPA. The instruction list intends to give you an overview over the commands and their syntax. The commands are sorted by topics in alphabetical order. For the parameters are integrated in the instruction list, there is no extra parameter list.

Instruction	Description	Page
)	Combination instructions (Bit)	56
+	Math instructions	27
+AR1	Math instructions	27
+AR2	Math instructions	27
+I	Math instructions	27
+D	Math instructions	27
+R	Math instructions	27
-D	Math instructions	27
-I	Math instructions	27
-R	Math instructions	27
*D	Math instructions	27
*I	Math instructions	27
*R	Math instructions	27
/D	Math instructions	27
/I	Math instructions	27
/R	Math instructions	27
==D	Comparison instructions	54
==I	Comparison instructions	54
==R	Comparison instructions	54
<=D	Comparison instructions	54
<=I	Comparison instructions	54
<=R	Comparison instructions	54
<D	Comparison instructions	54
<I	Comparison instructions	54
<R	Comparison instructions	54
<>D	Comparison instructions	54
<>I	Comparison instructions	54
<>R	Comparison instructions	54

Instruction	Description	Page
>=D	Comparison instructions	54
>=I	Comparison instructions	54
>=R	Comparison instructions	54
>D	Comparison instructions	54
>I	Comparison instructions	54
>R	Comparison instructions	54
A	Combination instructions (Bit)	56
A(	Combination instructions (Bit)	56
ABS	Math instructions	27
ACOS	Math instructions	27
AD	Combination instructions (Word)	64
AN	Combination instructions (Bit)	56
AN(	Combination instructions (Bit)	56
ASIN	Math instructions	27
ATAN	Math instructions	27
AW	Combination instructions (Word)	64
BTD	Data type conversion instructions	52
BTI	Data type conversion instructions	52
BE	Block instructions	33
BEC	Block instructions	33
BEU	Block instructions	33
BLD	Block instructions	33
CAD	Transfer instructions	47
CALL	Block instructions	33
CAR	Transfer instructions	47
CAR1	Transfer instructions	47
CAR2	Transfer instructions	47
CAW	Transfer instructions	47
CC	Block instructions	33
CD	Counter instructions	66
CDB	Block instructions	33
CLR	Setting/resetting bit addresses	41
COS	Math instructions	27
CU	Counter instructions	66

Instruction	Description	Page
DEC	Transfer instructions	*47*
DTB	Data type conversion instructions	*52*
DTR	Data type conversion instructions	*52*
EXP	Math instructions	*27*
FN	Edge-triggered instructions	*35*
FP	Edge-triggered instructions	*35*
FR	Counter instructions Timer instructions	*66* *65*
INC	Transfer instructions	*47*
INVD	Data type conversion instructions	*52*
INVI	Data type conversion instructions	*52*
ITB	Data type conversion instructions	*52*
ITD	Data type conversion instructions	*52*
JBI	Jump instructions	*44*
JC	Jump instructions	*44*
JCB	Jump instructions	*44*
JCN	Jump instructions	*44*
JL	Jump instructions	*44*
JM	Jump instructions	*44*
JMZ	Jump instructions	*44*
JN	Jump instructions	*44*
JNB	Jump instructions	*44*
JNBI	Jump instructions	*44*
JO	Jump instructions	*44*
JOS	Jump instructions	*44*
JP	Jump instructions	*44*
JPZ	Jump instructions	*44*
JU	Jump instructions	*44*
JUO	Jump instructions	*44*
JZ	Jump instructions	*44*
L	Load instructions	*36*
LAR1	Transfer instructions	*47*
LAR2	Transfer instructions	*47*
LD	Load instructions	*36*

Instruction	Description	Page
LN	Math instructions	⇨ *27*
LOOP	Jump instructions	⇨ *44*
MOD	Math instructions	⇨ *27*
NEGD	Data type conversion instructions	⇨ *52*
NEGI	Data type conversion instructions	⇨ *52*
NEGR	Math instructions	⇨ *27*
NOP	Block instructions	⇨ *33*
NOT	Setting/resetting bit addresses	⇨ *41*
O	Combination instructions (Bit)	⇨ *56*
O(	Combination instructions (Bit)	⇨ *56*
OD	Combination instructions (Word)	⇨ *64*
ON	Combination instructions (Bit)	⇨ *56*
ON(	Combination instructions (Bit)	⇨ *56*
OPN	Block instructions	⇨ *33*
OW	Combination instructions (Word)	⇨ *64*
POP	Transfer instructions	⇨ *47*
PUSH	Transfer instructions	⇨ *47*
R	Setting/resetting bit addresses	⇨ *41*
RLD	Shift instructions	⇨ *39*
RLDA	Shift instructions	⇨ *39*
RND	Data type conversion instructions	⇨ *52*
RND+	Data type conversion instructions	⇨ *52*
RND-	Data type conversion instructions	⇨ *52*
RRD	Shift instructions	⇨ *39*
RRDA	Shift instructions	⇨ *39*
S	Setting/resetting bit addresses	⇨ *41*
SA	Timer instructions	⇨ *65*
SAVE	Setting/resetting bit addresses	⇨ *41*
SD	Timer instructions	⇨ *65*
SE	Timer instructions	⇨ *65*
SET	Setting/resetting bit addresses	⇨ *41*
SIN	Math instructions	⇨ *27*
SLD	Shift instructions	⇨ *39*
SLW	Shift instructions	⇨ *39*

Instruction	Description	Page
SP	Timer instructions	➪ *65*
SQR	Math instructions	➪ *27*
SQRT	Math instructions	➪ *27*
SRD	Shift instructions	➪ *39*
SRW	Shift instructions	➪ *39*
SS	Timer instructions	➪ *65*
SSD	Shift instructions	➪ *39*
SSI	Shift instructions	➪ *39*
T	Transfer instructions	➪ *47*
TAK	Transfer instructions	➪ *47*
TAN	Math instructions	➪ *27*
TAR	Transfer instructions	➪ *47*
TAR1	Transfer instructions	➪ *47*
TAR2	Transfer instructions	➪ *47*
TRUNC	Data type conversion instructions	➪ *52*
UC	Block instructions	➪ *33*
X	Combination instructions (Bit)	➪ *56*
X(	Combination instructions (Bit)	➪ *56*
XN	Combination instructions (Bit)	➪ *56*
XN(	Combination instructions (Bit)	➪ *56*
XOD	Combination instructions (Word)	➪ *64*
XOW	Combination instructions (Word)	➪ *64*

1.2 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

Abbreviation	Description
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)

Abbreviation	Description
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)
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)

Abbreviation	Description
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

1.3 Comparison of syntax languages

Comparison

In the following overview, the German and international syntax languages of STL are compared.

Area	German	International
Input	E	I
Output	A	Q
Counter	Z	C
Periphery input byte	PEB	PIB
Periphery input word	PEW	PIW
Periphery input double word	PED	PID
Periphery output byte	PAB	PQB
Periphery output word	PAW	PQW
Periphery output double word	PAD	PQD
Combinations	U	A
	UN	AN
	U(	A(
	UN(	AN(

Area	German	International
	UW	AW
	UD	AD
Time functions	SI	SP
	SV	SE
	SE	SD
	SA	SF
Counter functions	ZV	CU
	ZR	CD
Load and transfer	TAR	CAR
	TAW	CAW
	TAD	CAD
Program control	AUF	OPN
	BEA	BEU
	BEB	BEC
	TDB	CDB
	UW	AW
	UD	AD
Jump functions	SPA	JU
	SPBB	JCB
	SPBIN	JNBI
	SPBNB	JNB
	SPBI	JBI
	SPBN	JCN
	SPB	JC
	SPO	JO
	SPS	JOS
	SPU	JUO
	SPZ	JZ
	SPN	JN
	SPMZ	JMZ
	SPPZ	JPZ
	SPL	JL
	SPM	JM
	SPP	JP

1.4 Differences between SPEED7 and 300V programming

General

The SPEED7-CPUs lean in the command processing against the Siemens S7-400 and differ here to the Siemens S7-300 (VIPA 300V).

These differences are listed below.

In the following, the CPU 318 from Siemens is counted for the S7-400 series from Siemens.

Status register

In opposite to the Siemens S7-300, the VIPA SPEED7-CPUs and Siemens S7-400 (CPU 318) use the status register bits OR, STA, /FC.

If your user application is based upon the circumstance that the mentioned bits in the status register are always zero (like Siemens S7-300), the program is not executable at VIPA SPEED7-CPUs and Siemens S7-400 (CPU 318).

ACCU handling at arithmetic operations

The CPUs of the Siemens S7-300 contain 2 ACCUs. At an arithmetic operation the content of the 2nd ACCU is not altered.

Whereas the SPEED7-CPUs provide 4 ACCUs. After an arithmetic operation (+I, -I, *I, /I, +D, -D, *D, /D, MOD, +R, -R, *R, /R) the content of ACCU 3 and ACCU 4 is loaded into ACCU 3 and 2.

This may cause conflicts in applications that presume an unmodified ACCU2.

RLO at jumps

The missing of the implementation of the start command bit /ER in the Siemens S7-300 may cause, under certain circumstances, deviations in the command execution of bit commands between Siemens S7-300 and VIPA SPEED7-CPUs respectively Siemens S7-400, especially at a jump to a bit conjunction chain.

Examples RLO at jumps

Example A:

```
A I0.0
A M1.1
= M2.0 // RLO =1 Command end
JU =J001 // jumps
.....
A M7.6
A M3.0
A M3.1
→J001:
A Q2.2 // after the jump...
// Siemens S7-300 further combines
// This command is used by VIPA SPEED7,
// Siemens S7-400 and CPU 318 as first request
```

Example B:

```
A I0.0
A M1.1
= M2.0 // RLO =1 command end
A Q3.3 // first request
JU =J001 // jumps
.....
A M3.0
A M3.1
→JO01:
A M3.2 // after jump ...
..... // the CPUs further combine
```

BCD consistency

At setting a timer or counter, a valid BCD value must be present in ACCU 1. The proof of this BCD value is in the Siemens S7-300 only executed when timer or counter are taken over (edge change). The SPEED7-CPUs (like the S7-400 from Siemens) always execute the verification.

Example:

```
......
A I5.4
L MW20
S T30
// Siemens S7-300 only proofs if timer
// is actively executed
// SPEED7, Siemens S7-400 and CPU 318
// always proof (also when no condition is present)
......
```

1.5 Registers

ACCU1 ... ACCU4 (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 ... 4)	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 *Chapter 1.6 'Addressing examples' on page 25*

Status word (16bit)

The values are analysed 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

Bit	Assignment	Description
8	BR	Binary result
9 ... 15	not used	-

1.6 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 time 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	

Addressing example	Description		
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"		
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
U E [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
U E [AR1,P#10.7]
```

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

1.7 Math instructions

Fixed-point arithmetic (16bit)

Math instructions of two 16bit numbers.
The result is in ACCU1 res. ACCU1-L.

Command	Operand	Parameter	Function	Length in words
+I	-		Add up two integers (16bit) (ACCU1-L)=(ACCU1-L)+(ACCU2-L)	1
-I	-		Subtract two integers (16bit) (ACCU1-L)=(ACCU2-L)-(ACCU1-L)	1
*I	-		Multiply two integers (16bit) (ACCU1-L)=(ACCU2-L)*(ACCU1-L)	1
/I	-		Divide two integers (16bit) (ACCU1-L)=(ACCU2-L):(ACCU1-L) The remainder is in ACCU1-H	1

Status word	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	-	-
Instruction influences	-	✓	✓	✓	✓	-	-	-	-

Fixed-point arithmetic (32bit)

Math instructions of two 32bit numbers.

The result is in ACCU1.

Com-mand	Operand	Parameter	Function	Length in words
+D	-		Add up two integers (32bit) (ACCU1)=(ACCU2)+(ACCU1)	1
-D	-		Subtract two integers (32bit) (ACCU1)=(ACCU2)-(ACCU1)	1
*D	-		Multiply two integers (32bit) (ACCU1)=(ACCU2)*(ACCU1)	1
/D	-		Divide two integers (32bit) (ACCU1)=(ACCU2):(ACCU1)	1
MOD	-		Divide two integers (32bit) and load the rest of the division in ACCU1 (ACCU1)=remainder of [(ACCU2):(ACCU1)]	1

Status word	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	-	-
Instruction influences	-	✓	✓	✓	✓	-	-	-	-

Floating-point arithmetic (32bit)

The result of the math instructions is in ACCU1. The execution time of the instruction depends on the value to calculate.

Command	Operand	Parameter	Function	Length in words
+R	-		Add up two real numbers (32bit) (ACCU1)=(ACCU2)+(ACCU1)	1
-R	-		Subtract two real numbers (32bit) (ACCU1)=(ACCU2)-(ACCU1)	1
*R	-		Multiply two real numbers (32bit) (ACCU1)=(ACCU2)*(ACCU1)	1
/R	-		Divide two real numbers (32bit) (ACCU1)=(ACCU2):(ACCU1)	1
NEGR	-		Negate the real number in ACCU1	1
ABS	-		Form the absolute value of the real number in ACCU1	1

Status word for: R	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	-	-
Instruction influences	-	✓	✓	✓	✓	-	-	-	-

Status word for: NEGR, ABS	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	-	-
Instruction influences	-	-	-	-	-	-	-	-	-

Square root an square instructions (32bit)

The result of the instructions is in ACCU1.

The instructions may be interrupted by alarms.

Com-mand	Operand	Parameter	Function	Length in words
SQRT	-		Calculate the Square root of a real number in ACCU1	1
SQR	-		Form the square of a real number in ACCU1	1

Status word	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	-	-
Instruction influences	-	✓	✓	✓	✓	-	-	-	-

Logarithmic function (32bit)

The result of the logarithm function is in ACCU1.

The instructions may be interrupted by alarms.

Com-mand	Operand	Parameter	Function	Length in words
LN	-		Calculate the natural logarithm of a real number in ACCU1	1
EXP	-		Calculate the exponential value of a real number in ACCU1 on basis e (=2.71828)	1

Status word	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	-	-
Instruction influences	-	✓	✓	✓	✓	-	-	-	-

Trigonometrical functions (32bit)

The result of the trigonometrical function is in ACCU1.

The instructions may be interrupted by alarms.

Command	Operand	Parameter	Function	Length in words
SIN[1]	-		Calculate the sine of the real number	1
ASIN[2]	-		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
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.				

Status word	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	-	-
Instruction influences	-	✓	✓	✓	✓	-	-	-	-

Addition of constants

Addition of integer constants to ACCU1.

The condition code bits are not affected.

Command	Operand	Parameter	Function	Length in words
+	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.

Com-mand	Operand	Parameter	Function	Length in words
+AR1	-		Add the contents of ACCU1-L to AR1	1
+AR1	m		Add pointer constant to the contents of AR1	2
+AR2	-		Add the contents of ACCU1-L to AR2	1
+AR2	m		Add pointer constant to the contents of AR2	2

1.8 Block instructions

Block call instructions

Com-mand	Operand	Parameter	Function	Length in words
CALL	FB p DB r	0 ... 8191 0 ... 8191	Unconditional call of a FB, with parameter transfer	
CALL	SFB p DB r	0 ... 8191 0 ... 8191	Unconditional call of a SFB, with parameter transfer	
CALL	FC p		Unconditional call of a function, with parameter transfer	
CALL	SFC p		Unconditional call of a SFC, with parameter transfer	
UC	FB q FC q Parameter	0 ... 8191	Unconditional call of blocks, without parameter transfer FB/FC call via parameters	1/2
CC	FB q FC q Parameter	0 ... 8191	Conditional call of blocks, without parameter transfer FB/FC call via parameters	1/2
OPN	DB p DI p Parameter	0 ... 8191	Open a data block Open a instance data block Open a data block via parameter	1/2 2 2

Status word for: CALL, UC	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	-	-
Instruction influences	-	-	-	-	0	0	1	-	0

Status word for: CC	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	✓	-
Instruction influences	-	-	-	-	0	0	1	-	0

Status word for: OPN	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	-	-
Instruction influences	-	-	-	-	-	-	-	-	-

Block end instructions

Com-mand	Operand	Parameter	Function	Length in words
BE			End block	1
BEU			End block unconditionally	1
BEC			End block if RLO="1"	1

Status word for: BE, BEU	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	-	-
Instruction influences	-	-	-	-	0	0	1	-	0

Status word for: BEC	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	✓	-
Instruction influences	-	-	-	-	✓	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.

Com-mand	Operand	Parameter	Function	Length in words
CDB			Exchange shared data block and instant data block	1

1.9 Program display and Null operation instructions

The status word is not affected.

Com-mand	Operand	Parameter	Function	Length in words
BLD	0 ... 255		Program display instruction: is treated by the CPU like a null operation instruction	1
NOP	0 1		Null operation instruction	1

1.10 Edge-triggered instructions

Edge-triggered instructions

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"

Command	Operand	Parameter	Function	Length in words
FP	I/Q a.b	0.0 ... 2047.7	Detecting the positive edge in the RLO. The bit addressed in the instruction is the auxiliary edge bit memory.	2
	M a.b	0.0 ... 8191.7		2
	L a.b	parameterizable		2
	DBX a.b	0.0 ... 65535.7		2
	DIX a.b	0.0 ... 65535.7		2
	c [AR1,m]			2
	c [AR2,m]			2
	[AR1,m]			2
	[AR2,m]			2
	Parameter			2
FN	I/Q a.b	0.0 ... 2047.7	Detecting the negative edge in the RLO. The bit addressed in the instruction is the auxiliary edge bit memory	2
	M a.b	0.0 ... 8191.7		2
	L a.b	parameterizable		2
	DBX a.b	0.0 ... 65535.7		2
	DIX a.b	0.0 ... 65535.7		2
	c [AR1,m]			2
	c [AR2,m]			2
	[AR1,m]			2
	[AR2,m]			2
	Parameter			2

Status word for: FP, FN	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	✓	-
Instruction influences	-	-	-	-	-	0	✓	✓	1

1.11 Load instructions

Load instructions

Loading address identifiers into ACCU1. The contents of ACCU1 and ACCU2 are saved first.

The status word is not affected.

Command	Operand	Parameter	Function	Length in words
L			Load ...	
	IB a		input byte	1/2
	QB a		output byte	1/2
	PIB a		periphery input byte	2
	MB a	0.0 ... 8191	bit memory byte	1/2
	LB a	parameterizable	local data byte	2
	DBB a	0.0 ... 65535	data byte	2
	DIB a	0.0 ... 65535	instance data byte	2
			... in ACCU1	
	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	0.0 ... 2046	input word	1/2
	QW a	0.0 ... 2046	output word	1/2
	PIW a	0.0 ... 8190	periphery input word	2
	MW a	0.0 ... 8190	bit memory word	1/2
	LW a	parameterizable	local data word	2
	DBW a	0.0 ... 65534	data word	1/2
	DIW a	0.0 ... 65534	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
L			Load ...	

Com-mand	Operand	Parameter	Function	Length in words
	ID a QD a PID a	0.0 ... 2044 0.0 ... 2044 0.0 ... 8188	input double word output double word periphery input double word	1/2 1/2 2
	MD a LD a DBD a DID a	0.0 ... 8188 parameterizable 0.0 ... 65532 0.0 ... 65532	bit memory double word local data double word data double word instance data double word	1/2 2 2 2
			... in ACCU1-L.	
	i [AR1,m] i [AR2,m] D [AR1,m] D [AR2,m] Parameter		register-indirect, area-internal (AR1) register-indirect, area-internal (AR2) area-crossing (AR1) area-crossing (AR2) via parameters	2 2 2 2 2
L			Load ...	
	k8 k16 k32		8bit constant in ACCU1-LL 16bit constant in ACCU1-L 32bit constant in ACCU1	1 2 3
	Parameter		Load constant in ACCU1 (addressed via parameters)	2
L	2#n		Load 16bit binary constant in ACCU1-L Load 32bit binary constant in ACCU1	2 3
L	B#8#p W#16#p DW#16#p		Load 8bit hexadecimal constant in ACCU1-LL Load 16bit hexadecimal constant in ACCU1-L Load 32bit hexadecimal constant in ACCU1	1 2 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

Com-mand	Operand	Parameter	Function	Length in words
L	T# time value		Load 16bit time constant Load 32bit time constant	2 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

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.

Com-mand	Operand	Parameter	Function	Length in words
L	T f Timer para.	0 ... 511	Load time value Load time value (addressed via parameters)	1/2 2
L	Z f Counter para.	0 ... 511	Load counter value Load counter value (addressed via parameters)	1/2 2
LC	T f Timer para.	0 ... 511	Load time value BCD-coded Load time value BCD-coded (addressed via parameters)	1/2 2
LC	Z f Counter para.	0 ... 511	Load counter value BCD-coded Load counter value BCD-coded (addressed via parameters)	1/2 2

1.12 Shift instructions

Shift instructions

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.

Command	Operand	Parameter	Function	Length in words
SLW SLW	- 0 ... 15		Shift the contents of ACCU1-L to the left. Positions that become free are provided with zeros.	1
SLD SLD	- 0 ... 32		Shift the contents of ACCU1 to the left. Positions that become free are provided with zeros.	1
SRW SRW	- 0 ... 15		Shift the contents of ACCU1-L to the right. Positions that become free are provided with zeros	1
SRD SRD	- 0 ... 32		Shift the contents of ACCU1 to the right. Positions that become free are provided with zeros	1
SSI SSI	- 0 ... 15		Shift the contents of ACCU1-L to the right with sign. Positions that become free are provided with the sign (bit 15)	1
SSD SSD	- 0 ... 32		Shift the contents of ACCU1 to the right with sign	1

Status word	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	-	-
Instruction influences	-	✓	✓	✓	-	-	-	-	-

Rotation instructions

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.

Com-mand	Operand	Parameter	Function	Length in words
RLD RLD	- 0 ... 32		Rotate the contents of ACCU1 to the left	1
RRD RRD	- 0 ... 32		Rotate the contents of ACCU1 to the right	1
RLDA	-		Rotate the contents of ACCU1 one bit position to the left, via CC1 bit	
RRDA	-		Rotate the contents of ACCU1 one bit position to the right, via CC1 bit	

Status word for: RLD, RRD	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	-	-
Instruction influences	-	✓	✓	✓	-	-	-	-	-

Status word for: RLDA, RRDA	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	-	-
Instruction influences	-	✓	0	0	-	-	-	-	-

1.13 Setting/resetting bit addresses

Setting/resetting bit addresses

Assign the value "1" or "0" or the RLO to the addressed instructions.

Com-mand	Operand	Parameter	Function	Length in words
S			Set ...	
	I/Q a.b	0.0 ... 2047.7	input/output to "1"	1/2
	M a.b	0.0 ... 8191.7	set bit memory to "1"	1/2
	L a.b	parameterizable	local data bit to "1"	2
	DBX a.b	0.0 ... 65535.7	data bit to "1"	2
	DIX a.b	0.0 ... 65535.7	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			Reset ...	
	I/Q a.b	0.0 ... 2047.7	input/output to "0"	1/2
	M a.b	0.0 ... 8191.7	set bit memory to "0"	1/2
	L a.b	parameterizable	local data bit to "0"	2
	DBX a.b	0.0 ... 65535.7	data bit to "0"	2
	DIX a.b	0.0 ... 65535.7	instance data bit to "0"	2
	c [AR1,m]		register-indirect, area-internal (AR1)	2
	c [AR2,m]		register-indirect, area-internal (AR2)	2
	[AR1,m]		area-crossing (AR1)	2
	[AR2,m]		area-crossing (AR2)	2
	Parameter		via parameters	2
=			Assign ...	

Com-mand	Operand	Parameter	Function	Length in words
	I/Q a.b	0.0 ... 2047.7	RLO to input/output	1/2
	M a.b	0.0 ... 8191.7	RLO to bit memory	1/2
	L a.b	parameterizable	RLO to local data bit	2
	DBX a.b	0.0 ... 65535.7	RLO to data bit	2
	DIX a.b	0.0 ... 65535.7	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

Status word for: S, R, =	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	✓	-
Instruction influences	-	-	-	-	-	0	✓	-	0

Instructions directly affecting the RLO

The following instructions have a directly effect on the RLO.

Com-mand	Operand	Parameter	Function	Length in words
CLR			Set RLO to "0"	1
SET			Set RLO to "1"	1
NOT			Negate RLO	1
SAVE			Save RLO into BR-bit	1

Status word for: CLR	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	-	-
Instruction influences	-	-	-	-	-	0	0	0	0

Status word for: SET	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	-	-
Instruction influences	-	-	-	-	-	0	1	1	0

Status word for: NOT	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	✓	-	✓	-
Instruction influences	-	-	-	-	-	-	1	✓	-

Status word for: SAVE	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	✓	-
Instruction influences	✓	-	-	-	-	-	-	-	-

1.14 Jump instructions

Jump, depending on conditions.

8-bit operands have a jump width of (-128...+127)

16-bit operands of (-32768...-129) or (+128...+32767)

Com-mand	Operand	Parameter	Function	Length in words
JU	LABEL		Jump unconditionally	1/2
JC	LABEL		Jump if RLO="1"	1/2
JCN	LABEL		Jump if RLO="0"	2
JCB	LABEL		Jump if RLO="1" Save the RLO in the BR-bit	2
JNB	LABEL		Jump if RLO="0" Save the RLO in the BR-bit	2
JBI	LABEL		Jump if BR="1"	2
JNBI	LABEL		Jump if BR="0"	2
JO	LABEL		Jump on stored overflow (OV="1")	1/2
JOS	LABEL		Jump on stored overflow (OS="1")	2
JUO	LABEL		Jump if "unordered instruction" (CC1=1 and CC0=1)	2
JZ	LABEL		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

Com-mand	Operand	Parameter	Function	Length in words
JL	LABEL		Jump distributor 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	2
LOOP	LABEL		Decrement ACCU1-L and jump if ACCU1-L ≠ 0 (loop programming)	2

Status word for: JU, JL, LOOP	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	-	-
Instruction influences	-	-	-	-	-	-	-	-	-

Status word for: JC, JCN	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	✓	-
Instruction influences	-	-	-	-	-	0	1	1	0

Status word for: JCB, JNB	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	✓	-
Instruction influences	✓	-	-	-	-	0	1	1	0

Status word for: JBI, JNBI	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	✓	-	-	-	-	-	-	-	-
Instruction influences	-	-	-	-	-	0	1	-	0

Status word for: JO	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	✓	-	-	-	-	-
Instruction influences	-	-	-	-	-	-	-	-	-

Status word for: JOS	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	✓	-	-	-	-
Instruction influences	-	-	-	-	0	-	-	-	-

Status word for: JUO, JZ, JP, JM, JN, JMZ, JPZ	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	✓	✓	-	-	-	-	-	-
Instruction influences	-	-	-	-	-	-	-	-	-

1.15 Transfer instructions

Transfer instructions Transfer the contents of ACCU1 into the addressed operand.

The status word is not affected.

Com-mand	Operand	Parameter	Function	Length in words
T			Transfer the contents of ACCU1-LL to ...	
	IB a	0.0 ... 2047	input byte	1/2
	QB a	0.0 ... 2047	output byte	1/2
	PQB a	0.0 ... 8191	periphery output byte	1/2
	MB a	0.0 ... 8191	bit memory byte	1/2
	LB a	parameterizable	local data byte	2
	DBB a	0.0 ... 65535	data byte	2
	DIB a	0.0 ... 65535	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	0.0 ... 2046	input word	1/2
	QW	0.0 ... 2046	output word	1/2
	PQW	0.0 ... 8190	periphery output word	1/2
	MW	0.0 ... 8190	bit memory word	1/2
	LW	parameterizable	local data word	2
	DBW	0.0 ... 65534	data word	2
	DIW	0.0 ... 65534	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
T			Transfer the contents of ACCU1 to ...	

Com-mand	Operand	Parameter	Function	Length in words
	ID	0.0 ... 2044	input double word	1/2
	QD	0.0 ... 2044	output double word	1/2
	PQD	0.0 ... 8188	periphery output double word	1/2
	MD	0.0 ... 8188	bit memory double word	1/2
	LD	parameterizable	local data double word	2
	DBD	0.0 ... 65532	data double word	2
	DID	0.0 ... 65532	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.

The status word is not affected.

Com-mand	Operand	Parameter	Function	Length in words
LAR1			Load the contents from ...	
	-		ACCU1	1
	AR2		address register 2	1
	DBD a	0 ... 65532	data double word	2
	DID a	0 ... 65532	instance data double word	2
	m		32bit constant as pointer	3
	LD a	parameterizable	local data double word	2
	MD a	0 ... 8188	bit memory double word	2
			... into AR1	
LAR2			Load the contents from ...	
	-		ACCU1	1
	DBD a	0 ... 65532	data double word	2
	DID a	0 ... 65532	instance data double word	2
	m		32bit constant as pointer	3
	LD a	parameterizable	local data double word	2
	MD a	0 ... 8188	bit memory double word.	2
			... into AR2	
TAR1			Transfer the contents from AR1 to ...	
	-		ACCU1	1
	AR2		address register 2	1
	DBD a	0 ... 65532	data double word	2
	DID a	0 ... 65532	instance data double word	2
	LD a	parameterizable	local data double word	2
	MD a	0 ... 8188	bit memory double word	2
TAR2			Transfer the contents from AR2 to...	
	-		ACCU1	1
	DBD a	0 ... 65532	data double word	2
	DID a	0 ... 65532	instance data double word	2
	LD a	parameterizable	local data double word	2

Com-mand	Operand	Parameter	Function	Length in words
	MD a	0 ... 8188	bit memory double word	2
TAR			Exchange the contents of AR1 and AR2	1

Load and transfer instructions for the status word

Com-mand	Operand	Parameter	Function	Length in words
L	STW		Load status word in ACCU1	
T	STW		Transfer ACCU1 (bits 0 ... 8) into status word	

Status word for: L STW	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	✓	✓	✓	✓	✓	✓	✓	✓	0
Instruction influences	-	-	-	-	-	-	-	-	-

Status word for: T STW	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	-	-
Instruction influences	✓	✓	✓	✓	✓	-	-	✓	-

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.

Com-mand	Operand	Parameter	Function	Length in words
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.

Command	Operand	Parameter	Function	Length in words
CAW	-		Reverse the order of the bytes in ACCU1-L LL, LH becomes LH, LL	1
CAD	-		Reverse the order of the bytes in ACCU1 LL, LH, HL, HH becomes HH, HL, LH, LL	1
TAK	-		Swap the contents of ACCU1 and ACCU2	1
ENT	-		The contents of ACCU2 and ACCU3 are transferred to ACCU3 and ACCU4	
LEAVE	-		The contents of ACCU3 and ACCU4 are transferred to ACCU2 and ACCU3	
PUSH	-		The contents of ACCU1, ACCU2 and ACCU3 are transferred to ACCU2, ACCU3 and ACCU4	1
POP	-		The contents of ACCU2, ACCU3 and ACCU4 are transferred to ACCU1, ACCU2 and ACCU3	1
INC	0 ... 255		Increment ACCU1-LL	1
DEC	0 ... 255		Decrement ACCU1-LL	1

1.16 Data type conversion instructions

Data type conversion instructions

The results of the conversion are in ACCU1. When converting real numbers, the execution time depends on the value.

Com-mand	Operand	Parameter	Function	Length in words
BTI	-		Convert contents of ACCU1 from BCD to integer (16bit) (BCD to Int.)	1
BTD	-		Convert contents of ACCU1 from BCD to integer (32bit) (BCD to Doubleint.)	1
DTR	-		Convert cont. of ACCU1 from integer (32bit) to Real number (32bit) (Doubleint. to Real)	1
ITD	-		Convert contents of ACCU1 from integer (16bit) to integer (32bit) (Int. to Doubleint)	1

Status word	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	-	-
Instruction influences	-	-	-	-	-	-	-	-	-

Com-mand	Operand	Parameter	Function	Length in words
ITB	-		Convert contents of ACCU1 from integer (16bit) to BCD 0 ... +/-999 (Int. To BCD)	1
DTB	-		Convert contents of ACCU1 from integer (32bit) to BCD 0 ... +/-9 999 999 (Doubleint. To BCD)	1
RND	-		Convert a real number to 32bit integer	1
RND-	-		Convert a real number to 32bit integer The number is rounded next hole number	1
RND+	-		Convert real number to 32bit integer It is rounded up to the next integer	1
TRUNC	-		Convert real number to 32bit integer The places after the decimal point are truncated	1

Status word	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	-	-
Instruction influences	-	-	-	✓	✓	-	-	-	-

Complement creation

Com-mand	Operand	Parameter	Function	Length in words
INVI	-		Forms the ones complement of ACCU1-L	1
INVD	-		Forms the ones complement of ACCU1	1
NEGI	-		Forms the twos complement of ACCU1-L (integer)	1
NEGD	-		Forms the twos complement of ACCU1 (double integer)	1

Status word for: INVI, INVD	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	-	-
Instruction influences	-	-	-	-	-	-	-	-	-

Status word for: NEGI, NEGD	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	-	-
Instruction influences	-	✓	✓	✓	✓	-	-	-	-

1.17 Comparison instructions

Comparison instructions with integer (16bit)

Comparing the integer (16bit) in ACCU1-L and ACCU2-L.

RLO=1, if condition is satisfied.

Com-mand	Operand	Parameter	Function	Length in words
==I	-		ACCU2-L = ACCU1-L	1
<>I	-		ACCU2-L different to ACCU1-L	1
<I	-		ACCU2-L < ACCU1-L	1
<=I	-		ACCU2-L <= ACCU1-L	1
>I	-		ACCU2-L > ACCU1-L	1
>=I	-		ACCU2-L >= ACCU1-L	1

Status word	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	-	-
Instruction influences	-	✓	✓	0	-	0	✓	✓	1

Comparison instructions with integer (32bit)

Comparing the integer (32bit) in ACCU1 and ACCU2.

RLO=1, if condition is satisfied.

Command	Operand	Parameter	Function	Length in words
==D	-		ACCU2 = ACCU1	1
<>D	-		ACCU2 different to ACCU1	1
<D	-		ACCU2 < ACCU1	1
<=D	-		ACCU2 <= ACCU1	1
>D	-		ACCU2 > ACCU1	1
>=D	-		ACCU2 >= ACCU1	1

Status word	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	-	-
Instruction influences	-	✓	✓	0	-	0	✓	✓	1

Comparison instructions with 32bit real number

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.

Command	Operand	Parameter	Function	Length in words
==R	-		ACCU2 = ACCU1	1
<>R	-		ACCU2 different to ACCU1	1
<R	-		ACCU2 < ACCU1	1
<=R	-		ACCU2 <= ACCU1	1
>R	-		ACCU2 > ACCU1	1
>=R	-		ACCU2 >= ACCU1	1

Status word	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	-	-
Instruction influences	-	✓	✓	✓	✓	0	✓	✓	1

1.18 Combination instructions (Bit)

Combination instructions with bit operands

Examining the signal state of the addressed instruction and gating the result with the RLO according to the appropriate logic function.

Command	Operand	Parameter	Function	Length in words
A			AND operation at signal state "1"	
	I/Q a.b	0.0 ... 2047.7	Input/output	1/2
	M a.b	0.0 ... 8191.7	Bit memory	1/2
	L a.b	parameterizable	Local data bit	2
	DBX a.b	0.0 ... 65535.7	Data bit	2
	DIX a.b	0.0 ... 65535.7	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			AND operation of signal state "0"	
	I/Q a.b	0.0 ... 2047.7	Input/output	1/2
	M a.b	0.0 ... 8191.7	Bit memory	1/2
	L a.b	parameterizable	Local data bit	2
	DBX a.b	0.0 ... 65535.7	Data bit	2
	DIX a.b	0.0 ... 65535.7	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

Status word for: A, AN	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	✓	-	✓	✓
Instruction influences	-	-	-	-	-	✓	✓	✓	1

Com-mand	Operand	Parameter	Function	Length in words
O			OR operation at signal state "1"	
	I/Q a.b	0.0 ... 2047.7	Input/output	1/2
	M a.b	0.0 ... 8191.7	Bit memory	1/2
	L a.b	parameterizable	Local data bit	2
	DBX a.b	0.0 ... 65535.7	Data bit	2
	DIX a.b	0.0 ... 65535.7	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
ON			OR operation at signal state "0"	
	I/Q a.b	0.0 ... 2047.7	Input/output	1/2
	M a.b	0.0 ... 8191.7	Bit memory	1/2
	L a.b	parameterizable	Local data bit	2
	DBX a.b	0.0 ... 65535.7	Data bit	2
	DIX a.b	0.0 ... 65535.7	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

Status word for: O, ON	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	✓	✓
Instruction influences	-	-	-	-	-	0	✓	✓	1

Com-mand	Operand	Parameter	Function	Length in words
X			EXCLUSIVE-OR operation at signal state "1"	
	I/Q a.b	0.0 ... 2047.7	Input/output	1/2
	M a.b	0.0 ... 8191.7	Bit memory	1/2
	L a.b	parameterizable	Local data bit	2
	DBX a.b	0.0 ... 65535.7	Data bit	2
	DIX a.b	0.0 ... 65535.7	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 ... 2047.7	Input/output	1/2
	M a.b	0.0 ... 8191.7	Bit memory	1/2
	L a.b	parameterizable	Local data bit	2
	DBX a.b	0.0 ... 65535.7	Data bit	2
	DIX a.b	0.0 ... 65535.7	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

Status word for: X, XN	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	✓	✓
Instruction influences	-	-	-	-	-	0	✓	✓	1

Combination instructions with parenthetical expressions

Saving the bits BR, RLO, OR and a function ID (A, AN, ...) at the nesting stack.

For each block 7 nesting levels are possible.

Command	Operand	Parameter	Function	Length in words
A(			AND left parenthesis	1
AN(			AND-NOT left parenthesis	1
O(			OR left parenthesis	1
ON(			OR-NOT left parenthesis	1
X(			EXCLUSIVE-OR left parenthesis	1
XN(			EXCLUSIVE-OR-NOT left parenthesis	1
)			Right parenthesis; popping an entry off the nesting stack. Gating RLO with the current RLO in the processor.	1

Status word for: A(, AN(, O(, ON(X(, XN(	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	✓	-	-	-	-	✓	-	✓	✓
Instruction influences	-	-	-	-	-	0	1	-	0

Status word for:)	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	✓	-
Instruction influences	✓	-	-	-	-	✓	1	✓	1

ORing of AND operations

The ORing of AND operations is implemented according the rule: AND before OR.

Command	Operand	Parameter	Function	Length in words
O			OR operations of AND functions according the rule: AND before OR	1

Status word	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	✓	-	✓	✓
Instruction influences	-	-	-	-	-	✓	1	-	✓

Combination instructions with timer and counters

Examining the signal state of the addressed timer/counter an gating the result with the RLO according to the appropriate logic function.

Command	Operand	Parameter	Function	Length in words
A			AND operation at signal state	
	T f	0 ... 511	Timer	1/2
	C f	0 ... 511	Counter	1/2
	Timer para.		Timer addressed via parameters	2
	Counter para.		Counter addressed via parameters	2
AN			AND operation at signal state	
	T f	0 ... 511	Timer	1/2
	C f	0 ... 511	Counter	1/2
	Timer para.		Timer addressed via parameters	2
	Counter para.		Counter addressed via parameters	2

Status word	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	✓	-	✓	✓
Instruction influences	-	-	-	-	-	✓	✓	✓	1

Com-mand	Operand	Parameter	Function	Length in words
O			OR operation at signal state	
	T f	0 ... 511	Timer	1/2
	C f	0 ... 511	Counter	1/2
	Timer para.		Timer addressed via parameters	2
	Counter para.		Counter addressed via parameters	2
ON			OR operation at signal state	
	T f	0 ... 511	Timer	1/2
	C f	0 ... 511	Counter	1/2
	Timer para.		Timer addressed via parameters	2
	Counter para.		Counter addressed via parameters	2

Status word	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	✓	✓
Instruction influences	-	-	-	-	-	0	✓	✓	1

Com-mand	Operand	Parameter	Function	Length in words
X			EXCLUSIVE-OR operation at signal state	
	T f	0 ... 511	Timer	1/2
	C f	0 ... 511	Counter	1/2
	Timer para.		Timer addressed via parameters	2
	Counter para.		Counter addressed via parameters	2
XN			EXCLUSIVE-OR operation at signal state	
	T f	0 ... 511	Timer	1/2
	C f	0 ... 511	Counter	1/2
	Timer para.		Timer addressed via parameters	2
	Counter para.		Counter addressed via parameters	2

Status word	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	✓	✓
Instruction influences	-	-	-	-	-	0	✓	✓	1

Combination instructions

Examining the specified conditions for their signal status, and gating the result with the RLO according to the appropriate function.

Command	Operand	Parameter	Function	Length in words
A, O, X			AND, OR, EXCLUSIVE OR operation at signal state "1"	
	==0		Result = 0 (CC1=0) and (CC0=0)	1
	>0		Result > 0 (CC1=1) and (CC0=0)	1
	<0		Result < 0 (CC1=0) and (CC0=1)	1
	<>0		Result different to 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

Status word for: A	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	✓	✓	✓	✓	✓	✓	-	✓	✓
Instruction influences	-	-	-	-	-	✓	✓	✓	1

Status word for: O, X	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	✓	✓	✓	✓	✓	-	-	✓	✓
Instruction influences	-	-	-	-	-	0	✓	✓	1

Com-mand	Operand	Parameter	Function	Length in words
AN ON XN			AND NOT/OR NOT/EXCLUSIVE OR NOT Operation at signal state "0"	1
	==0		Result = 0 (CC1=0) and (CC0=0)	1
	>0		Result > 0 (CC1=1) and (CC0=0)	1
	<0		Result < 0 (CC1=0) and (CC0=1)	1
	<>0		Result different to 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

Status word for: AN	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	✓	✓	✓	✓	✓	✓	-	✓	✓
Instruction influences	-	-	-	-	-	✓	✓	✓	1

Status word for: ON, XN	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	✓	✓	✓	✓	✓	-	-	✓	✓
Instruction influences	-	-	-	-	-	-	✓	✓	1

1.19 Combination instructions (Word)

Combination instructions with the contents of ACCU1

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.

Command	Operand	Parameter	Function	Length in words
AW	k16		AND ACCU2-L	1
AW			AND 16bit constant	2
OW	k16		OR ACCU2-L	1
OW			OR 16bit constant	2
XOW	k16		EXCLUSIVE OR ACCU2-L	1
XOW			EXCLUSIVE OR 16bit constant	2
AD	k32		AND ACCU2	1
AD			AND 32bit constant	3
OD	k32		OR ACCU2	1
OD			OR 32bit constant	3
XOD	k32		EXCLUSIVE OR ACCU2	1
XOD			EXCLUSIVE OR 32bit constant	3

Status word	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	-	-
Instruction influences	-	✓	0	0	-	-	-	-	-

1.20 Timer instructions

Starting or resetting a timer (addressed directly or via parameters).

The time value must be in ACCU1-L.

Com-mand	Operand	Parameter	Function	Length in words
SP	T f	0 ... 511	Start time as pulse on edge change from "0" to "1"	1/2
	Timer para.			2
SE	T f	0 ... 511	Start timer as extended pulse on edge change from "0" to"1"	1/2
	Timer para.			2
SD	T f	0 ... 511	Start timer as ON delay on edge change from "0" to "1"	1/2
	Timer para.			2
SS	T f	0 ... 511	Start timer as saving start delay on edge change from "0" to "1"	1/2
	Timer para.			2
SA	T f	0 ... 511	Start timer as OFF delay on edge change from "1" to "0"	1/2
	Timer para.			2
FR	T f	0 ... 511	Enable timer for restarting on edge change from "0" to "1" (reset edge bit memory for starting timer)	1/2
	Timer para.			2
R	T f	0 ... 511	Reset timer	1/2
	Timer para.			2

Status word	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	✓	-
Instruction influences	-	-	-	-	-	0	-	-	0

1.21 Counter instructions

The counter value is in ACCU1-L res. in the address transferred as parameter.

Command	Operand	Parameter	Function	Length in words
S	C f	0 ... 511	Presetting of counter on edge change from "0" to "1"	1/2
	Counter para.			2
R	C f	0 ... 511	Reset counter to "0" on edge change from "0" to "1"	1/2
	Counter-para.			2
CU	C f	0 ... 511	Increment counter by 1 on edge change from "0" to "1"	1/2
	Counter-para.			2
CD	C f	0 ... 511	Decrement counter by 1 on edge change from "0" to "1"	1/2
	Counter para.			2
FR	C f	0 ... 511	Enable counter on edge change from "0" to "1" (reset the edge bit memory for up and down counting)	1/2
	Counter para.			2

Status word	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
Instruction depends on	-	-	-	-	-	-	-	✓	-
Instruction influences	-	-	-	-	-	0	-	-	0

2 Block parameters

2.1 General and Specific Error Information RET_VAL

Overview

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

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

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

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

RET_VAL (Return value)

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

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

Specific error code

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

A specific error code consists of the following two numbers:

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

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

General error codes RET_VAL

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

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

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

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

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.
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.

Error code	Description
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.

3 Organization Blocks

3.1 Overview

OBs (Organization blocks) 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:

- On start-up of the CPU
- On cyclic or clocked execution
- On errors
- On hardware interrupts occur

3.2 Main

3.2.1 OB1 - Main - Program Cycle

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.

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

3.3 Startup

3.3.1 OB 100, OB 102 - Complete/Cold Restart - Startup

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 Startup OB and then executes the main program in OB 1. Here the current program and the current data blocks generated by SFC remain in memory.

A distinction is made between the following types of startup:

- OB 100: Complete restart
- OB 102: Cold restart

The CPU executes a startup as follows:

- after PowerON 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 startup OB 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 startup OB with default names of the variables and its data types:

Variable	Type	Description
OB10x_EV_CLASS	BYTE	Event class and identifiers: 13h: active
OB10x_STRTUP	BYTE	Startup request ■ 81h: Manual restart request ■ 82h: Automatic restart request ■ 85h: Request for manual cold restart ■ 86h: Request for automatic cold restart ■ 8Ah: Master: Manual restart request ■ 8Bh: Master: Automatic restart request
OB10x_PRIORITY	BYTE	Priority class: 27
OB10x_OB_NUMBR	BYTE	OB number (100 or 102)
OB10x_RESERVED_1	BYTE	reserved
OB10x_RESERVED_2	BYTE	reserved
OB10x_STOP	WORD	Number of the event that caused the CPU to STOP
OB10x_STRT_INFO	DWORD	Supplementary information about the current startup
OB10x_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 OB10x_STRT_INFO

Bit no.	Explanation	Possible values (binary)	Description
31...24	Startup infor-mation	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 PowerON
		xxxx 1xxx	Clock for time stamp battery-backed at last PowerON
23...16	Startup just completed	0000 0011	Restart triggered with mode selector
		0000 0100	Restart triggered by command via MPI
		0000 0111	Cold restart triggered with mode selector
		0000 1000	Cold restart triggered by command via MPI
		0001 0000	Automatic restart after battery-backed PowerON
		0001 0011	Restart triggered with mode selector; last PowerON battery-backed

Bit no.	Explanation	Possible values (binary)	Description
		0001 0100	Restart triggered by command via MPI; last PowerON battery-backed
		0010 0000	Automatic restart battery-backed PowerON (with memory reset by system)
		0010 0011	Restart triggered with mode selector last PowerON not battery-backed
		0010 0100	Restart triggered by command via MPI last PowerON 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 PowerON	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 PowerON
		0001 0011	Restart triggered with mode selector; last PowerON battery-backed
		0001 0100	Restart triggered by command via MPI; last PowerON battery-backed
		0010 0000	Automatic restart after battery-backed PowerON (with memory reset by system)
		0010 0011	Restart triggered with mode selector last PowerON not battery-backed
		0010 0100	Restart triggered by command via MPI last PowerON not battery-backed

3.4 Communication Interrupts

3.4.1 OB 55 - DP: Status Alarm - Status Interrupt

Description

A status interrupt OB (OB 55) is only available for DP-V1 capable CPUs.

The CPU operating system calls OB 55 if a status interrupt was triggered via the slot of a DP-V1 slave. This might be the case if a component (module) of a DP-V1 slaves changes its operating mode, for example from RUN to STOP. For precise information on events that trigger a status interrupt, refer to the documentation of the DP-V1 slave‘s manufacturer.

Local data

The following table describes the start information of the OB 55 with default names of the variables and its data types:

Variable	Data type	Description
OB55_EV_CLASS	BYTE	Event class and identifiers: 11h: incoming event
OB55_STRT_INF	BYTE	55h: Status interrupt for DP 58h: Status interrupt for PROFINET IO
OB55_PRIORITY	BYTE	Configured priority class: Default value: 2
OB55_OB_NUMBR	BYTE	OB number (55)
OB55_RESERVED_1	BYTE	reserved
OB55_IO_FLAG	BYTE	Input module: 54h Output module: 55h
OB55_MDL_ADDR	WORD	Logical base address of the module that triggers the interrupt
OB55_LEN	BYTE	Data block length supplied by the interrupt
OB55_TYPE	BYTE	ID for the interrupt type "Status interrupt"
OB55_SLOT	BYTE	Slot number of the interrupt triggering component (module)
OB55_SPEC	BYTE	Specifier: ■ Bit 1, 0: Interrupt specifier ■ Bit 2: Add_Ack ■ Bit 7 ... 3: Seq. number
OB55_DATE_TIME	DATE_AND_TIME	Date and time of day when the OB was called

You can obtain the full additional information on the interrupt from the frame by calling SFB 54 "RALRM" in OB 55. *Chapter 11.2.22 ‘SFB 54 - RALRM - Receiving an interrupt from a periphery module’ on page 441*

3.4.2 OB 56 - DP: Update Alarm - Update Interrupt

Description

A update interrupt OB (OB 56) is only available for DP-V1 capable CPUs.

The CPU operating system calls OB 56 if an update interrupt was triggered via the slot of a DP-V1 slave. This can be the case if you have changed the parameters for the slot of a DP-V1 slave. For precise information on events that trigger an update interrupt, refer to the documentation of the DP-V1 slave manufacturer.

Local data

The following table describes the start information of the OB 56 with default names of the variables and its data types:

Variable	Data type	Description
OB56_EV_CLASS	BYTE	Event class and identifiers: 11h: incoming event
OB56_STRT_INF	BYTE	56h: Update interrupt for DP 59h: Update interrupt for PROFINET IO
OB56_PRIORITY	BYTE	Configured priority class: Default value: 2
OB56_OB_NUMBR	BYTE	OB number (56)
OB56_RESERVED_1	BYTE	reserved
OB56_IO_FLAG	BYTE	Input module: 54h Output module: 55h
OB56_MDL_ADDR	WORD	Logical base address of the module that triggers the interrupt
OB56_LEN	BYTE	Data block length supplied by the interrupt
OB56_TYPE	BYTE	ID for the interrupt type "Update interrupt"
OB56_SLOT	BYTE	Slot number of the interrupt triggering component
OB56_SPEC	BYTE	Specifier: ■ Bit 1, 0: Interrupt specifier ■ Bit 2: Add_Ack ■ Bit 7 ... 3: Seq. number
OB56_DATE_TIME	DATE_AND_TIME	Date and time of day when the OB was called

You can obtain the full additional information on the interrupt from the frame by calling SFB 54 "RALRM" in OB 56. *Chapter 11.2.22 'SFB 54 - RALRM - Receiving an interrupt from a periphery module' on page 441*

3.4.3 OB 57 - DP: Manufacture Alarm - Manufacturer Specific Interrupt

Description

The OB 57 is called by the operating system of the CPU if an manufacturer specific interrupt was triggered via the slot of a slave system.

Local data

The following table describes the start information of the OB 57 with default names of the variables and its data types:

Variable	Data type	Description
OB57_EV_CLASS	BYTE	Event class and identifiers: 11h: incoming event
OB57_STRT_INF	BYTE	57h: Start request for OB 57
OB57_PRIORITY	BYTE	Configured priority class: Default value: 2
OB57_OB_NUMBR	BYTE	OB number (57)
OB57_RESERVED_1	BYTE	reserved
OB57_IO_FLAG	BYTE	Input module: 54h Output module: 55h
OB57_MDL_ADDR	WORD	Logical base address of the module that triggers the interrupt
OB57_LEN	BYTE	reserved
OB57_TYPE	BYTE	reserved
OB57_SLOT	BYTE	reserved
OB57_SPEC	BYTE	reserved
OB57_DATE_TIME	DATE_AND_TIME	Date and time of day when the OB was called

You can obtain the full additional information on the interrupt from the frame by calling SFB 54 "RALRM" in OB 57.

3.5 Time delay Interrupts

3.5.1 OB 20, OB 21 - DEL_INTx - Time-delay Interrupt

Description

A 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 number, 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. It can be blocked with SFC 39 DIS_IRT and released with SFC 40 EN_IRT. 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 a 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.

Local data

The following table describes the start information of the OB 20 and OB 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 22h: start request for OB 21
OB20_PRIORITY	BYTE	assigned priority class: Default: 3 (OB 20) ... 6 (OB 23)
OB20_OB_NUMBR	BYTE	OB number (20, 21)
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)

Variable	Type	Description
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.

3.6 Time of day Interrupts

3.6.1 OB 10, OB 11 - TOD_INTx - Time-of-day Interrupt

Description

Time-of-day interrupts are used when you want to run a program at a particular time, either once only or periodically. Time-of-day interrupts 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 time-of-day interrupts 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

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

Function

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

1. The time-of-day interrupts are configured via the hardware configuration. Open the selected CPU with **Edit** > *Object properties* > *Time-of-Day interrupts*. Here the corresponding time-of-day interrupts may be adjusted and activated. After transmission to CPU and startup the monitoring of time-of-day interrupt is automatically started.
2. 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.
3. 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 a 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 time-of-day interrupts 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 ... OB 11 with default names of the variables and its data types. The variable names are the default names of OB 10.

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 12h: Start request for OB 11
OB10_PRIORITY	BYTE	Assigned priority class: default 2
OB10_OB_NUMBR	BYTE	OB number (10 ... 11)
OB10_RESERVED_1	BYTE	reserved
OB10_RESERVED_2	BYTE	reserved

Variable	Type	Description
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.

3.7 Cyclic Interrupts

3.7.1 OB 28, 29, 32, 33, 34, 35 - CYC_INTx - Cyclic Interrupt

Description

By means of a cyclic 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 OB	Default time interval	Default priority class	Option for phase offset
OB 28	250µs	24	no*
OB 29	500µs	24	no*
OB 32	1s	09	yes
OB 33	500ms	10	yes
OB 34	200ms	11	yes
OB 35	100ms	12	yes
*) If both OBs are activated OB 28 is executed first and then OB 29. Due to the very short time intervals and the high priority a simultaneous execution of OB 28 and OB 29 should be avoided.			

Activation

A cyclic interrupt is activated by programming the corresponding OB within the CPU. The cyclic 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 cyclic OBs are called in the configured equidistant intervals with consideration of the phase shift. The equidistant start times of the cyclic OBs result of the respective time frame and the phase shift. So a sub program can be called time controlled by programming a respective OB.

Phase offset

The phase offset can be used to stagger the execution of cyclic interrupt handling routines despite the fact that these routines are timed to a multiple of the same interval. The use of the phase offset achieves a higher interval accuracy. The start time of the time interval and the phase offset is the instant of transition from STARTUP to RUN. The call instant for a cyclic interrupt OB is thus the time interval plus the phase offset.

Parameterization

Time interval, phase offset (not OB 28, 29) and priority may be parameterized by the hardware configurator.

Depending on the OB there are the following possibilities for parameterization:

OB 28, 29, 33:	Parameterizable as VIPA specific parameter by the properties of the CPU.
OB 32, 35:	Parameterizable by Siemens CPU 318-2DP.

You must make sure that the run time of each cyclic interrupt OB is significantly shorter than its interval. The cyclic interrupt that caused the error is executed later.

Local data

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

Variable	Type	Description
OB35_EV_CLASS	BYTE	Event class and identifiers: 11h: Cyclic interrupt is active
OB35_STRT_INF	BYTE	29h: Start request for OB 28 30h: Start request for OB 29 33h: Start request for OB 32 34h: Start request for OB 33 35h: Start request for OB 34 36h: Start request for OB 35

Variable	Type	Description
OB35_PRIORITY	BYTE	Assigned priority class; Default values: 24 (OB 28, 29), 9 (OB 32) ... 12 (OB 35)
OB35_OB_NUMBR	BYTE	OB number (28, 29, 32 ... 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_EXC_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.

Since the blocks SFC58/59 respectively SFB52/53 for reading and writing data blocks cannot be interrupted, in conjunction with OB 28 and OB 29 the CPU may change to STOP state!

3.8 Hardware Interrupts

3.8.1 OB 40, OB 41 - HW_INTx - Hardware Interrupt

Description

Hardware interrupts are used to enable the immediate detection in the user program of events in the controlled process, making it possible to respond with an appropriate interrupt handling routine. Here OB 40 and OB 41 can be used. 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.
Chapter 11.1.43 ‘SFC 55 - WR_PARM - Write dynamic parameter’ on page 347 *Chapter 11.1.44 ‘SFC 56 - WR_DPARM - Write default parameter’ on page 349* *Chapter 11.1.45 ‘SFC 57 - PARM_MOD - Parameterize module’ on page 351*

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 and OB 41 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	41h: Interrupt via Interrupt line 1
OB40_PRIORITY	BYTE	Assigned priority class: Default: 16 (OB 40) Default: 17 (OB 41)
OB40_OB_NUMBR	BYTE	OB number (40, 41)
OB40_RESERVED_1	BYTE	reserved
OB40_IO_FLAG	BYTE	Input Module: 54h Output Module: 55h
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 states of the inputs on the module (bit 0 corresponds to the first input). ■ For analog modules – Bit field with information which channel has exceeded which limit. ■ For CPs or IMs – Informs about the module interrupt status.
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.

3.9 Asynchronous error Interrupts

3.9.1 OB 80 - CYCL_FLT - 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 blocked, respectively delayed and released by means of SFC 39 ... 42.

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

Error code	Variable	Bit	Description
	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

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 **in** **R**UN)* CiR synchronizations time in ms

3.9.2 OB 81 - PS_FLT - 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

Variable	Data type	Description
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
OB81_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.

3.9.3 OB 83 - I/O_FLT2 - Insert / Remove Module

Description

The CPU operating system calls OB 83 in following situations:

- after insertion / removal of a configured module
- after modifications of module parameters and download of changes to the CPU during RUN

If you have not programmed OB 83, the CPU changes to STOP mode. You can disable/delay/enable the insert/remove interrupt OB with the help of SFCs 39 to 42.

Insert/Remove

Each time a configured module is removed or inserted during the RUN, STOP, and STARTUP modes, an insert/remove interrupt is generated (power supply modules, CPUs and Bus coupler must not be removed in these modes). This interrupt causes an entry in the diagnostic buffer and in the system status list for the CPU involved. The insert/remove OB is also started if the CPU is in the RUN mode. If this OB has not been programmed, the CPU changes to the STOP mode. Then system polls modules in seconds intervals to detect insertion or removal. To enable the CPU to detect the removal and insertion of a module, a minimum time interval of two seconds must expire between removal and insertion. If you remove a configured module in the RUN mode, OB 83 is started. Since the existence of modules is only monitored at intervals of one second, an access error may be detected first if the module is accessed directly or when the process image is updated. If you insert a module in a configured slot in the RUN mode, the operating system checks whether the type of the module inserted corresponds to the recorded configuration. OB 83 is then started and parameters are assigned if the module types match.

Reconfiguring modules

You can reassign the parameters to existing modules when you modify your system configuration during runtime. This reassignment of parameters is performed by transferring the required parameter data records to the modules. This is the procedure:

1. OB 83 will be started (Start event: 3367h) after you have assigned new parameters to a module and downloaded this configuration to the CPU in RUN mode. Relevant OB start information is the logical basic address (OB83_MDL_ADDR) and the module type (OB83_MDL_TYPE). Module I/O data may be incorrect as of now, which means that no SFC may be busy sending data records to this module.
2. The module parameters are reassigned after OB 83 was executed.
3. OB 83 will be restarted after the parameters have been assigned
   - Start event: 3267h, provided this parameter assignment was successful, or
   - 3968h, if failed

   The modules I/O data response is identical to their response after an insertion interrupt, that is, currently they may be incorrect. You can now call SFCs again to send data records to the module.

Local Data

The following table describes the start information of the OB 83 with default names of the variables and its data types:

Variable	Data type	Description
OB83_EV_CLASS	BYTE	Event class and identifiers: 32h: End of reassignment of module parameters 33h: Start of reassignment of module parameters 38h: module inserted 39h: module removed or not responding, or end of parameter assignment
OB83_FLT_ID	BYTE	Error code: (possible values: 51h, 54h ... 56h, 58h, 61, 63h ... 68h)
OB83_PRIORITY	BYTE	Priority class: can be assigned via hardware configuration
OB83_OB_NUMBR	BYTE	OB number (83)
OB83_RESERVED_1	BYTE	Identification of module or submodule/ interface module

Variable	Data type	Description
OB83_MDL_ID	BYTE	54h: Peripheral input (PI) 55h: Peripheral output (PQ)
OB83_MDL_ADDR	WORD	■ Central or distributed PROFIBUS DP: – Logical base address of the module affected. If it is a mixed module, it is the smallest logical address used in the module. – If the I and O addresses in the mixed block are equal, the logical base address is the one that receives the event identifier. ■ Distributed PROFINET IO: – Logical base address of the module/submodule
OB83_RACK_NUM	WORD	■ If OB83_RESERVED_1 = A0h: number of submodule/interface sub-module (low byte) ■ If OB83_RESERVED_1 = C4h: – central: rack number – distributed PROFIBUS DP: number of DP station (low byte) and DP master system ID (high byte) – distributed PROFINET IO: physical address: identifier bit (bit 15, 1 = PROFINET IO), IO system ID (bits 11 ... 14) and device number (bits 0 ... 10)

Variable	Data type	Description
OB83_MDL_TYPE	WORD	■ Central or distributed PROFIBUS DP: Module type of affected module (x:irrelevant to the user) – x5xxh: analog module – x8xxh: function module – xCxxh: CP – xFxxh: digital module – 8340h: Replacement type identifier for input module – 9340h: Replacement type identifier for output module – A340h: Replacement type identifier for mixed module (I/O) – F340h: Replacement type identifier for uniquely identifiable module (e.g. with packed addresses) ■ Distributed PROFINET IO: – 8101h: module type of the inserted module is the same as the module type of the removed module – 8102h: module type of the inserted module is not the same as the module type of the removed module
OB83_DATE_TIME	DATE_AND_TIME	DATE_AND_TIME of day when the OB was called

OB83_EV_CLASS The following table shows the event that started OB 83:

OB83_EV_CLASS	OB83_FLT_ID	Description
39h	51h	PROFINET IO module removed
	54h	PROFINET IO submodule removed
38h	54h	PROFINET IO submodule inserted and matches config-ured submodule
	55h	PROFINET IO submodule inserted, but does not match configured submodule
	56h	PROFINET IO submodule inserted, but error with module parameters
	58h	PROFINET IO submodule, access error corrected
39h	61h	Module removed or not responding OB83_MDL_TYPE: Actual module type
38h	61h	Module inserted. Module type OK OB83_MDL_TYPE: Actual module type

OB83_EV_CLASS	OB83_FLT_ID	Description
	63h	Module inserted but incorrect module type OB83_MDL_TYPE: Actual module type
	64h	Module inserted but problem (module ID cannot be read) OB83_MDL_TYPE: Configured module type
	65h	Module inserted but error in module parameter assignment OB83_MDL_TYPE: Actual module type
39h	66h	Module not responding, load voltage error
38h	66h	Module responds again, load voltage error corrected
33h	67h	Start of module reconfiguration
32h	67h	End of module reconfiguration
39h	68h	Module reconfiguration terminated with error

If you are using a DP-V1- or PROFINET-capable CPU you can obtain additional information on the interrupt with the help of SFB 54 "RALRM" which exceeds the start information of the OB.

3.9.4 OB 85 - OBNL_FLT - Priority Class 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.

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 STARTUP)
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
		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

OB85_EV_CLASS	OB85_FLT_ID	Variable	Description
	B1h, B2h	OB85_DKZ23	ID of the type of process image transfer where the I/O access error happened. 10h: Byte access 20h: Word access 30h: 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
			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.			
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: 10h: Byte access 20h: Word access 30h: 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

OB85_EV_CLASS	OB85_FLT_ID	Variable	Description
	B3h, B4h	OB85_Z23	Byte 0: Part process image number 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.			

3.9.5 OB 86 - RACK_FLT - 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).

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)

Variable	Type	Description
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
	C4h, C5h	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

EV_CLASS	FLT_ID	Variable	Bit ...	Description
			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

3.10 Synchronous Interrupts

3.10.1 OB 121 - PROG_ERR - Programming 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
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	OB121_FLT_REG:	BCD conversion error
		ID for the register concerned (0000h: accumulator 1)

Error code	Variable	Description
22h	OB121_RESERVED_1	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.
		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	OB121_FLT_REG	Range error when reading
25h		Range error when writing
		Contains the ID of the illegal area in the low byte (86h of own local data area)
26h	OB121_FLT_REG	Error for timer number
27h		Error for counter number
		Illegal number
30h	OB121_FLT_REG	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
		Illegal DB number
34h	OB121_FLT_REG	FC number error in FC call

Error code	Variable	Description
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
		Illegal DB number

3.10.2 OB 122 - MOD_ERR - 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

Variable	Type	Description
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.

3.11 Cycle synchronous Interrupts

4 Include VIPA library

Libraries

The VIPA specific blocks can be found as library *'...LIB'* for download in the service area of www.vipa.com at *'Downloads'*. The libraries are packed ZIP files. As soon as you want to use VIPA specific blocks you have to import them into your project. The VIPA specific blocks can be found in the libraries according to its applications with the following structure:

- General library
  - *Chapter 5 'Building Control' on page 105*
  - *Chapter 6 'Network Communication' on page 120*
  - *Chapter 8 'Serial Communication' on page 212*
  - *Chapter 9 'EtherCAT Communication' on page 247*
  - *Chapter 11 'Integrated Standard' on page 293*
  - *Chapter 13 'System Blocks' on page 497*
- Modbus library
  - *Chapter 7 'Modbus Communication' on page 188*
- Library for motion, energy and frequency measurement
  This library is only available for the Siemens SIMATIC Manager.
  - *Chapter 10 'Device Specific' on page 255*

4.1 Integration into Siemens SIMATIC Manager

Overview

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

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

Load ZIP file

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

Retrieve library

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

Open library and transfer blocks into the project

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

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

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

4.2 Integration into Siemens TIA Portal

Overview

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

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

Load ZIP file

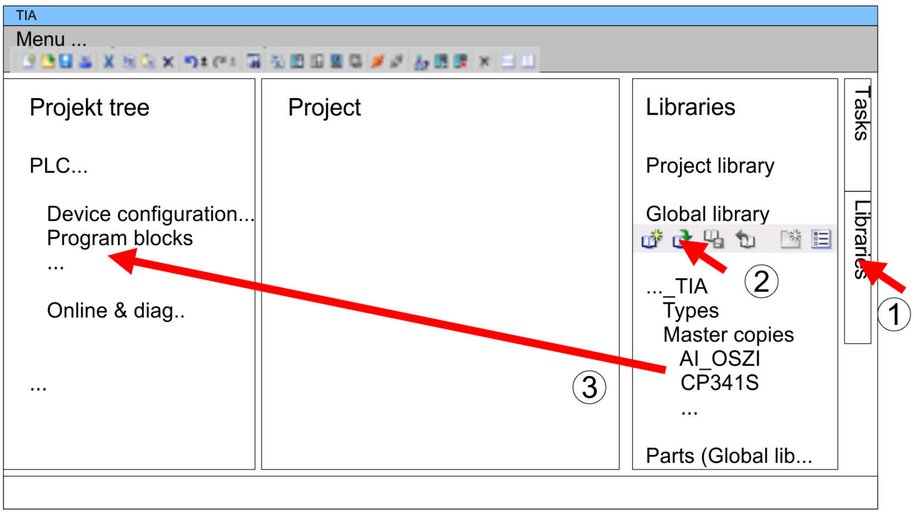
1. Navigate on the web page to the ZIP file, that matches your version of the program.
2. Load and store it in your work directory.

Unzip the Zip file

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

Open library and transfer blocks into the project

1. Start the Siemens TIA Portal with your project.
2. Switch to the *Project view*.
3. Choose "Libraries" from the task cards on the right side.
4. Click at "Global libraries".
5. Click at "Open global libraries".
6. Navigate to your work directory and load the file ..._TIA.al1x.



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

5 Building Control

5.1 Overview

In this chapter the function blocks (FB45 ... FB50) for building control (GLT) can be found. The blocks use the system time of the CPU. There are no S7 timers required. You have the option to use for each block an instance data block or multiple instances. There are the following blocks:

FB		Description
FB 45	LAMP	Controlling a lamp or socket
FB 46	BLIND	Controlling blind
FB 47	DSTRIKE	Controlling an electric door opener
FB 48	ACONTROL	Access control
FB 49	KEYPAD	Requesting a keypad with external power supply
FB 50	KEYPAD2	Requesting a keypad without external power supply

5.1.1 Call example - instance DB

Network 1

```
CALL "Ceiling lamp", DB 1
  ON        :=M20.0
  OFF       :=20.1
  ONOFF     :=20.2
  Duration  :=T#5M
  Output    :=M20.3
  PulseOn   :=
  PulseOff  :=
```

5.1.2 Call example - multi instances DB

Content of: “Environment\Interface\Stat"

In the following there is a STL call example of the usage of multiple lights and a blind with multiple instances.

Name	Data type	Address
Ceiling lamp	LAMP	0.0
Floor lamp	LAMP	46.0

Name	Data type	Address
Mirror lamp	LAMP	92.0
Blind	BLIND	138.0

Network 1

```
CALL #Ceiling lamp
  ON        :=M20.0
  OFF       :=20.1
  ONOFF     :=20.2
  Duration  :=T#5M
  Output    :=M20.3
  PulseOn   :=
  PulseOff  :=
```

Network 2

```
CALL #Blind
  Up              :=M30.0
  Down            :=M30.1
  CentralUp       :=
  CentralDown     :=
  TimeMaxDuration :=T#10S
  TimePause       :=T#1S
  TimeShortLong   :=T#2S
  Endable         :=
  BlindUp         :=M30.6
  BlindDown       :=M30.7
```

5.2 Room

5.2.1 FB 45 - LAMP - Controlling lamp / socket

Description

With this block you can control load relays for lamps and sockets. It can be controlled via On/Off button or via separate On and Off button. Additionally with *Duration* you have the possibility to set a duration for the automatic switching-off. With *TimeDebounce* you can specify a debounce time for the input signals.

- When driving a monostable relay the output remains set as long as the relay is to be activated. With an edge change 0-1 at *OnOff* respectively *On* the static output *Output* is set. It remains set until you reset it with an edge change 0-1 at *OnOff* respectively *Off* or the time of *Duration* has expired.
- When controlling a bistable relay 2 outputs are used. Here *PulseOn* controls the switch on and *PulseOff* the switch off procedure. With *TimePulse* the pulse duration and with *TimePause* the switch time of the two outputs can be specified.

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *⇨ Chapter 4 'Include VIPA library' on page 103*

Parameters

Parameter	Declaration	Data type	Description
OnOff	INPUT	BOOL	With an edge change 0-1 *Output* is activated respectively deactivated and *PulseOn* or *PulseOff* is activated. Default: FALSE
On	INPUT	BOOL	With an edge change 0-1 *Output* is activated respectively deactivated and *PulseOn* is activated. Default: FALSE
Off	INPUT	BOOL	With an edge change 0-1 *Output* is deactivated and *PulseOff* is activated. Default: FALSE
Duration	INPUT	TIME	Time for the duration the *Output* is deactivated respectively *PulseOff* is activated. With 0ms the automatic switch off is deactivated. Default: 0ms
Output	OUTPUT	BOOL	Static output to drive a monostable relay.
PulseOn	OUTPUT	BOOL	Pulse output to control the bistable relay (On signal).
PulseOff	OUTPUT	BOOL	Pulse output to control the bistable relay (Off signal).
TimeDebounce	CONSTANT	TIME	Time for debounce of the inputs. Default: 100ms
TimePulse	CONSTANT	TIME	Time for the pulse duration of *PulseOn* respectively *PulseOff*. Default: 100ms
TimePause	CONSTANT	TIME	Time for the break between resetting and setting of *PulseOn* respectively *PulseOff*. Default: 100ms

5.2.2 FB 46 - BLIND - Controlling blind

Description

With this block a motorized blind can be controlled. For this you have to release the drive with *Enable*.

- The controlling for “lifting” *BlindUp* and “sinking" *BlindDown* happens by 2 buttons (*Up*/*Down* respectively *CentralUp*/*CentralDown*).
  - *CentralUp*/*CentralDown*: Used for central control of all blinds in a building.
  - *Up*/*Down*: Used for local control of a blind. Here a pending *CentralUp*/*CentralDown* signal is ignored.
- If the corresponding button is pressed longer as the specified *TimeShortLong* the blend drive moves to the respective end position. By pressing on of the two buttons (*Up/Down* respectively *CentralUp/CentralDown*) you can stop the movement and reverse, it if necessary.
- With *TimeMaxDuration* you can specify the maximum run time of the motor and with *TimePause* you can specify the pause for the change of direction.
- By jogging the blend drive shortly moves. With this function you can adjust the blind slats fine.
- With *TimeDebounce* you can specify a debounce time for the input signals.
- With Status you can check the position of the blend.
  - 0: Upper limit position
  - 50: Unknown position between the two limit positions
  - 100: Lowest limit position

CAUTION!
The blend drive must have its own limit switches that turn off power automatically!

VIPA specific block
The VIPA specific blocks can be found in the VIPA library. *Chapter 4 ‘Include VIPA library’ on page 103*

Parameters

Parameter	Declaration	Data type	Description
Up	INPUT	BOOL	With an edge change 0-1 the output *BlindUp* is activated. Depending on the input signal the blend drives to the upper limit position or is shortly moved. As long as the signal is pending the signals *CentralUp/CentralDown* are ignored. Default: FALSE
Down	INPUT	BOOL	With an edge change 0-1 the output *BlindDown* is activated. Depending on the input signal the blend drives to the lower limit position or is shortly moved. As long as the signal is pending the signals *CentralUp/CentralDown* are ignored. Default: FALSE
CentralUp	INPUT	BOOL	With an edge change 0-1 the output *BlindUp* is activated. Here the blind moves to the upper limit position. Default: FALSE
CentralDown	INPUT	BOOL	With an edge change 0-1 the output *BlindDown* is activated. Here the blind moves to the lowest limit position. Default: FALSE
TimeMaxDuration	INPUT	TIME	Maximum drive time to reach the respective end position. Default: 30s
TimePause	INPUT	TIME	Break between a direction change. Default: 2s
TimeShortLong	INPUT	TIME	Time for the distinction between jog mode and continuous mode. Default: 1s
Enable	INPUT	BOOL	Release for the drive (static) Default: TRUE
BlindUp	OUTPUT	BOOL	Static output blind “lifting”
BlindDown	OUTPUT	BOOL	Static output blind “sinking”

Parameter	Declaration	Data type	Description
Status	OUTPUT	INT	■ Status - Blind position – 0: Upper limit position – 50: Unknown position between the two limit positions – 100: Lowest limit position
TimeDebounce	CONSTANT	TIME	Time for debounce of the inputs. Default: 100ms

5.2.3 FB 47 - DSTRIKE - Electric door opener

Description

With this block an electric door opener can be controlled, if its not locked with *DoorIsLocked*.

- With an edge change 0-1 at the input *Open* for the duration *'TimeOpening'* *'Output'* is controlled.
- With an edge change 0-1 at the input *EnableAlwaysOpen* respectively *DisableAlwaysOpen Open* is static activated respectively deactivated. Additionally with set *EnableAlwaysOpen* the static output *AlwaysOpen* is set.
- You can connect your door contacts at *DoorIsClosed* and *DoorIsLocked*. *DoorIsClosed* is set, as soon as the door is closed. *DoorIsLocked* is set as soon as the door is locked, i.e. the contact is controlled by the locking mechanism of the door and opening of the door by means of the electric door opener is disabled.

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

Parameters

Parameter	Declaration	Data type	Description
Open	INPUT	BOOL	With an edge change 0-1 *Output* is activated for the duration of *TimeOpening*. Default: FALSE
EnableAlwaysOpen	INPUT	BOOL	With an edge change 0-1 *Output* is static set. Default: FALSE
DisableAlwaysOpen	INPUT	BOOL	With an edge change 0-1 *Output* is static reset. Default: FALSE
TimeOpening	INPUT	TIME	Time for the duration of the activation of *Output*. Default: 3s

Parameter	Declaration	Data type	Description
DoorIsClosed	INPUT	BOOL	■ Optional - Position door TRUE: Door is closed FALSE: Door is open Default: FALSE
DoorIsLocked	INPUT	BOOL	■ Optional - Lock state of the door – TRUE: Door is locked – FALSE: Door is not locked Default: FALSE
Output	OUTPUT	BOOL	Static output to drive a monostable relay.
AlwaysOpen	OUTPUT	BOOL	Static output to indicate "Door is static open".

5.3 Access Control

5.3.1 FB 48 - ACONTROL - Access control

Description

With this block a access control can be implemented. After getting a code from an external keypad, panel or RFID reader, the code is compared with a list. Depending on the result, then the relative outputs are controlled.

- The access codes are to be applied in a data block, which is specified by *ACLBlock*. Here you can also specify which outputs *Access1...6* are to be controlled and how (pulse/static) are they controlled. With the data block up to 16 access codes can be treaded.
- Via *AccessCode1...4* the code of the corresponding input device is specified.
- Via *CheckCode1...4* the code is compared with the code in your data block *ACLBlock*.
  - If the access code in the data block exists, the corresponding outputs are controlled according to the specifications. With configured pulse output you can specify the pulse duration with *TimePulse*.
  - If the access code does not exist in the data block, the output *Error* is set for the duration *TimeError*.
- With an edge change 0-1 of *CentralLock* all the access codes are disabled. Here the output *CentralLocked* is set.
- With an edge change 0-1 of *CentralUnlock* all the access codes are enabled and the output *CentralLocked* is reset.

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *➪ Chapter 4 ‘Include VIPA library’ on page 103*

5.3.1.1 Block parameters

Parameters

Parameter	Declaration	Data type	Description
AccessCode1	INPUT	STRING[16]	Access code, e.g. from keypad.
CheckCode1	INPUT	BOOL	With an edge change 0-1, the *AccessCode1* is compared with the access code in the data block *ACLBlock*. Default: 0
AccessCode2	INPUT	STRING[16]	Access code, e.g. from panel.
CheckCode2	INPUT	BOOL	With an edge change 0-1, the *AccessCode2* is compared with the access code in the data block *ACLBlock*. Default: 0
AccessCode3	INPUT	STRING[16]	Access code, e.g. RFID reader.
CheckCode3	INPUT	BOOL	With an edge change 0-1, the *AccessCode3* is compared with the access code in the data block *ACLBlock*. Default: 0
AccessCode4	INPUT	STRING[16]	Access code, e.g. from an other system
CheckCode4	INPUT	BOOL	With an edge change 0-1, the *AccessCode4* is compared with the access code in the data block *ACLBlock*. Default: 0
CentralLock	INPUT	BOOL	With an edge change 0-1 all the access codes are disabled. Here the output *CentralLocked* is set.
CentralUnlock	INPUT	BOOL	With an edge change 0-1 of *CentralUnlock* all the access codes are enabled and the output *CentralLocked* is reset.
ACLBlock	INPUT	BLOCK	Data block with the access codes. *Chapter 5.3.3 'UDT 4 - ACL - Data structure for FB48' on page 114*
Access1	OUTPUT	BOOL	Output 1, can be controlled as pulse or static.
Access2	OUTPUT	BOOL	Output 2, can be controlled as pulse or static.
Access3	OUTPUT	BOOL	Output 3, can be controlled as pulse or static.
Access4	OUTPUT	BOOL	Output 4, can be controlled as pulse or static.
Access5	OUTPUT	BOOL	Output 5, can be controlled as pulse or static.

Parameter	Declaration	Data type	Description
Access6	OUTPUT	BOOL	Output 6, can be controlled as pulse or static.
Error	OUTPUT	BOOL	If the access code does not exist in the data block, the output *Error* is set for the duration *TimeError*.
CentralLocked	OUTPUT	BOOL	■ Access – TRUE: locked - access not possible – FALSE: not locked - access possible Default: TRUE
TimePulse	CONSTANT	Time	Time for the pulse duration at an output. Default: 3s
TimeError	CONSTANT	Time	Time for the duration of the error signal. Default: 500ms

5.3.2 UDT 3 - ACLREC - Data structure for FB48

Description

Address	Name	Type	Start value	Comment
0.0		**STRUCT**		
+0.0	Code	STRING[16]	' '	Byte 0 ... 17: Access code S7String with max. 16 ASCII characters for access code
+18.0	Enable-Output1	BOOL	FALSE	Byte 18: Signal for the outputs to be controlled TRUE: activate output, FALSE: deactivate output
+18.1	Enable-Output2	BOOL	FALSE	
+18.2	Enable-Output3	BOOL	FALSE	
+18.3	Enable-Output4	BOOL	FALSE	
+18.4	Enable-Output5	BOOL	FALSE	
+18.5	Enable-Output6	BOOL	FALSE	
+18.6	EnableRes7	BOOL	FALSE	
+18.7	EnableRes8	BOOL	FALSE	

Address	Name	Type	Start value	Comment
0.0		**STRUCT**		
+19.0	Signal-Output1	BOOL	FALSE	Byte 19: Signal type FALSE: Pulse, TRUE: static 1, deactivation with additional code
+19.1	Signal-Output2	BOOL	FALSE	
+19.2	Signal-Output3	BOOL	FALSE	
+19.3	Signal-Output4	BOOL	FALSE	
+19.4	Signal-Output5	BOOL	FALSE	
+19.5	Signal-Output6	BOOL	FALSE	
+19.6	SignalRes7	BOOL	FALSE	
+19.7	SignalRes8	BOOL	FALSE	
=20.0				

5.3.3 UDT 4 - ACL - Data structure for FB48

Description

Address	Name	Type	Start value	Comment
0.0		**STRUCT**		
+0.0	Record-Count	INT	16	DBW0: Number valid record sets (0 ... n)
+2.0	RecordLen	INT	20	DBW2: Length of one record set in bytes (20)
+4.0	Record	ARRAY[0... 15]		The first record set starts from DBB4
*20.0		"UDT 3 - ACLREC"		*Chapter 5.3.2 'UDT 3 - ACLREC - Data structure for FB48' on page 113*
=324.0		BOOL		

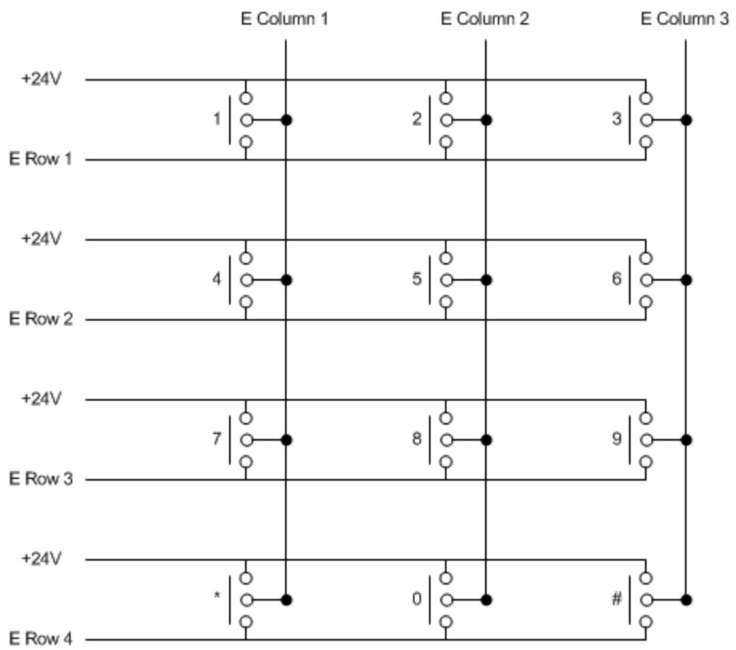
CAUTION!
A code must only occur 1 x in the whole list. Duplicate Codes are not allowed.

5.3.4 FB 49 - KEYPAD - Keyboard

Description

This block is used to connect an external keypad (0...9,*,#) with external DC 24V power supply. Depending on the pressed key, the keyboard provides the row and column signals (24V). The block evaluates the signals internally by means of a bit pattern table and transfers the determined ASCII code into the keyboard buffer. If necessary, or automatically the keyboard buffer is output as max. 16byte character string.

- Via *Row 1...4* the rows 1...4 of the keyboard matrix are connected.
- Via *Column 1...3* the columns 1...3 of the keyboard matrix are connected.
- Via *ClearCode* you can specify a key code to clear the keyboard buffer.
- Via *EnterCode* you can specify a key code to output the keyboard buffer at *Output* for one cycle. During this time the output *Valid* is enabled.
- Via edge change 0-1 of *Clear* the keyboard buffer cleared.
- Via *TimeAutoClear* you specify the max. duration for pressing the keys. Otherwise the keyboard buffer is cleared.
- Via *CountCharAutoEnter* you can specify the number of characters, which are automatically output as keyboard buffer at *Output* for one cycle. During this time the output *Valid* is enabled.
- *Error* is activated for the time *TimeError* when a key is pressed, but the keyboard buffer is full.
- With *TimeDebounce* you can specify a debounce time for the input signals.



VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *↳ Chapter 4 'Include VIPA library' on page 103*

Parameters

Parameter	Declaration	Data type	Description
Row1	INPUT	BOOL	Row 1 of the keyboard matrix. Default: FALSE
Row2	INPUT	BOOL	Row 2 of the keyboard matrix. Default: FALSE
Row3	INPUT	BOOL	Row 3 of the keyboard matrix. Default: FALSE
Row4	INPUT	BOOL	Row 4 of the keyboard matrix. Default: FALSE
Column1	INPUT	BOOL	Column 1 of the keyboard matrix. Default: FALSE
Column2	INPUT	BOOL	Column 2 of the keyboard matrix. Default: FALSE
Column3	INPUT	BOOL	Column 3 of the keyboard matrix. Default: FALSE
ClearCode	INPUT	BYTE	The value at which the keyboard buffer is to be cleared. 0: deactivated Default: 42 = *
EnterCode	INPUT	BYTE	The value at which the keyboard buffer is to be output. 0: deactivated Default: 35 = #
Clear	INPUT	BOOL	Edge change 0-1 clears the keyboard buffer. Default: FALSE
TimeAutoClear	INPUT	TIME	Duration within a further key must be pressed. Otherwise the keyboard buffer is cleared. 0: Buffer is not cleared Default: 10s
CountCharAu-toEnter	INPUT	INT	Number of characters, which are automatically output as keyboard buffer. 0: deactivated Default: 0
Output	OUTPUT	STRING[16]	Contents of the keyboard buffer as max. 16 byte string.

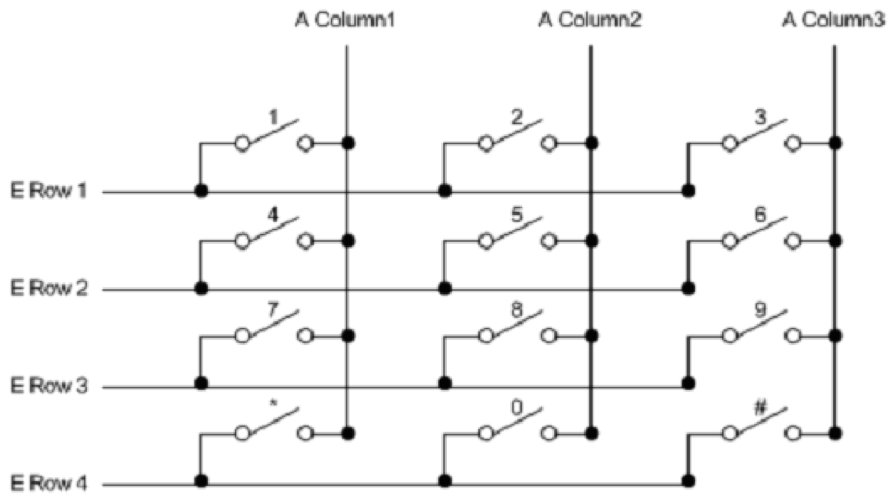
Parameter	Declaration	Data type	Description
Valid	OUTPUT	BOOL	The static output indicates that the string at *Output* is valid. The signal is pending for one cycle.
Error	OUTPUT	BOOL	*Error* is activated for the time *TimeError* when a key is pressed, but the keyboard buffer is full.
TimeDebounce	CONSTANT	TIME	Time for debounce of the inputs. Default: 100ms
TimeError	CONSTANT	TIME	Time for the duration of the error signal. Default: 500ms

5.3.5 FB 50 - KEYPAD2 - Keyboard

Description

This block is used to connect an external keypad (0...9,*,#) without an own power supply. The block provides output column signals. Depending on the pressed key, the keyboard provides the according row signals. The block evaluates the signals internally by means of a bit pattern table and transfers the determined ASCII code into the keyboard buffer. If necessary, or automatically the keyboard buffer is output as max. 16byte character string.

- Via *Row 1...4* the rows 1...4 of the keyboard matrix are connected.
- Via *Column 1...3* the columns 1...3 of the keyboard matrix are connected.
- Via *TimeDelay* you can specify a waiting time after setting the column outputs up to reading the corresponding row inputs. This time must be greater than the delay time of the used module.
- Via *ClearCode* you can specify a key code to clear the keyboard buffer.
- Via *EnterCode* you can specify a key code to output the keyboard buffer at *Output* for one cycle. During this time the output *Valid* is enabled.
- Via edge change 0-1 of *Clear* the keyboard buffer cleared.
- Via *TimeAutoClear* you specify the max. duration for pressing the keys. Otherwise the keyboard buffer is cleared.
- Via *CountCharAutoEnter* you can specify the number of characters, which are automatically output as keyboard buffer at *Output* for one cycle. During this time the output *Valid* is enabled.
- *Error* is activated for the time *TimeError* when a key is pressed, but the keyboard buffer is full.
- With *TimeDebounce* you can specify a debounce time for the input signals.



VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

Parameters

Parameter	Declaration	Data type	Description
Row1	INPUT	BOOL	Row 1 of the keyboard matrix. Default: FALSE
Row2	INPUT	BOOL	Row 2 of the keyboard matrix. Default: FALSE
Row3	INPUT	BOOL	Row 3 of the keyboard matrix. Default: FALSE
Row4	INPUT	BOOL	Row 4 of the keyboard matrix. Default: FALSE
ClearCode	INPUT	BYTE	The value at which the keyboard buffer is to be cleared. 0: deactivated Default: 42 = *
EnterCode	INPUT	BYTE	The value at which the keyboard buffer is to be output. 0: deactivated Default: 35 = #
Clear	INPUT	BOOL	Edge change 0-1 clears the keyboard buffer. Default: FALSE

Parameter	Declaration	Data type	Description
TimeAutoClear	INPUT	TIME	Duration within a further key must be pressed. Otherwise the keyboard buffer is cleared. 0: Buffer is not cleared Default: 10s
CountCharAutoEnter	INPUT	INT	Number of characters, which are automatically output as keyboard buffer. 0: deactivated Default: 0
Column1	OUTPUT	BOOL	Column 1 of the keyboard matrix. Default: FALSE
Column2	OUTPUT	BOOL	Column 2 of the keyboard matrix. Default: FALSE
Column3	OUTPUT	BOOL	Column 3 of the keyboard matrix. Default: FALSE
Output	OUTPUT	BYTE	Contents of the keyboard buffer as max. 16 byte string.
Valid	OUTPUT	BOOL	The static output indicates that the string at *Output* is valid. The signal is pending for one cycle.
Error	OUTPUT	BOOL	*Error* is activated for the time *TimeError* when a key is pressed, but the keyboard buffer is full.
TimeDebounce	CONSTANT	TIME	Time for debounce of the inputs. Default: 100ms
TimeError	CONSTANT	TIME	Time for the duration of the error signal. Default: 500ms
TimeDelay	CONSTANT	TIME	Duration after setting the column outputs up to reading the corresponding row inputs. This time must be greater than the delay time of the used module. Default: 10ms

6 Network Communication

6.1 Open Communication

6.1.1 Connection-oriented protocols

- Connection-oriented protocols establish a (logical) connection to the communication partner before data transmission is started. And if necessary they terminate the connection after the data transfer was finished.
- Connection-oriented protocols are used for data transmission when reliable, guaranteed delivery is of particular importance. Also the correct order of the received packets is ensured.
- In general, many logical connections can exist on one physical line.
- The following connection-oriented protocols are supported with FBs for open communication via industrial Ethernet:
  - TCP/IP native according to RFC 793 (connection types 01h and 11h)
  - ISO on TCP according to RFC 1006 connection type 12h)

TCP native

- During data transmission, no information about the length or about the start and end of a message is transmitted. However, the receiver has no means of detecting where one message ends in the data stream and the next one begins.
- The transfer is stream-oriented. For this reason, it is recommended that the data length of the FBs is identical for the sending and receiving station.
- If the number of received data does not fit to the preset length you either will get not the whole data, or you will get data of the following job.
- The receive block copies as many bytes into the receive area as you have specified as length. After this, it will set NDR to TRUE and write RCVD_LEN with the value of LEN. With each additional call, you will thus receive another block of sent data.

ISO on TCP

- During data transmission, information on the length and the end of the message is also transmitted. The transfer is block-oriented
- If you have specified the length of the data to be received greater than the length of the data to be sent, the receive block will copy the received data completely into the receive range. After this, it will set NDR to TRUE and write RCVD_LEN with the length of the sent data.
- If you have specified the length of the data to be received less than the length of the sent data, the receive block will not copy any data into the receive range but instead will supply the following error information: ERROR = 1, STATUS = 8088h.

6.1.2 Connection-less protocols

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

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

UDP

- In this case, when calling the sending block you have to specify the address parameters of the receiver (IP address and port number). During data transmission, information on the length and the end of the message is also transmitted.
- Analog after finishing the receive block you get a reference to the address parameter of the sender (IP address and port no.)
- In order to be able to use the sending and receiving blocks first you have to configure the local communications access point at both sides.
- With each new call of the sending block, you re-reference the remote partner by specifying its IP address and its port number.
- If you have specified the length of the data to be received greater than the length of the data to be sent, the receive block will copy the received data completely into the receive range. After this, it will set NDR to TRUE and write RCVD_LEN with the length of the sent data.
- If you have specified the length of the data to be received less than the length of the sent data, the receive block will not copy any data into the receive range but instead will supply the following error information: ERROR = 1, STATUS = 8088h.

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

Description

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

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

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

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

Parameters

Parameter	Declaration	Data type	Memory area	Description
REQ	INPUT	BOOL	I, Q, M, D, L	Control parameter REQUEST, initiates the transmission at rising edge. At the first call with REQ = 1, data are transmitted from the area specified by the DATA parameter.
ID	INPUT	WORD	M, D, constant	Reference to the connection to determinated. ID must be identical to the associated parameter ID in the local connection description. Range of values: 0001h ... 0FFFh
LEN	INPUT	INT	I, Q, M, D, L	Number of bytes to be sent with the job Range of values: ■ 1 ... 1460, if connection type = 01h ■ 1 ... 8192, if connection type = 11h ■ 1 ... 1452, if connection type = 12h and a CP is being used ■ 1 ... 8192, if connection type = 12h and no CP is being used
DONE	OUTPUT	BOOL	I, Q, M, D, L	DONE status parameter: ■ 0: Job not yet started or still running. ■ 1: Job executed without error.
BUSY	OUTPUT	BOOL	I, Q, M, D, L	■ BUSY = 1: Job is not yet completed. A new job cannot be triggered. ■ BUSY = 0: Job is completed.

Parameter	Declaration	Data type	Memory area	Description
ERROR	OUTPUT	BOOL	I, Q, M, D, L	ERROR status parameter: ■ ERROR = 1: Error occurred during processing. STATUS provides detailed information on the type of error.
STATUS	OUTPUT	WORD	M, D	STATUS parameter: Error information
DATA	IN_OUT	ANY	I, Q, M, D	Send area, contains address and length. The address refers to: ■ The process image input ■ The process image output ■ A bit memory ■ A data block

Error information

ERROR	STATUS	Description
0	0000h	Send job completed without error.
0	7000h	First call with REQ = 0, sending not initiated.
0	7001h	First call with REQ = 1, sending initiated.
0	7002h	Follow-on call (REQ irrelevant), job being processed **Note**: during this processing the operating system accesses the data in the DATA send buffer.
1	8085h	LEN parameter has the value 0 or is greater than the largest permitted value.
1	8086h	The ID parameter is not in the permitted address range.
0	8088h	LEN parameter is larger than the memory area specified in DATA.
1	80A1h	Communications error:: ■ FB 65 TCON was not yet called for the specified ID ■ The specified connection is currently being terminated. Transmission over this connection is not possible. ■ The interface is being reinitialized.
1	80B3h	The parameter for the connection type (connection_type parameter in the connection description) is set to UDP. Please use the FB 67 TUSEND.
1	80C3h	The resources (memory) of the CPU are temporarily occupied.
1	80C4h	Temporary communications error: ■ The connection to the communications partner cannot be established at this time. ■ The interface is receiving new parameters.
1	8822h	DATA parameter: Source area invalid: area does not exist in DB.
1	8824h	DATA parameter: Range error in ANY pointer.

ERROR	STATUS	Description
1	8832h	DATA parameter: DB number too large.
1	883Ah	DATA parameter: Access to send buffer not possible (e.g. due to deleted DB).
1	887Fh	DATA parameter: Internal error, such as an invalid ANY reference.

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

Description

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

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

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

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

The two variants are more described in the following table.

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

Function

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

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

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

Parameters

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

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

Error information

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

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

6.1.5 FB 65 - TCON - Establishing a connection

Use with TCP native and ISO on TCP

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

Use with UDP

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

Description

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

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

Parameters

Parameter	Declaration	Data type	Memory area	Description
REQ	INPUT	BOOL	I, Q, M, D, L	Control parameter *REQ*, initiates establishing the connection at rising edge.
ID	INPUT	WORD	M, D, constant	Reference to the connection to be established to the remote partner or between the user program and the communications level of the operating system. ID must be identical to the associated parameter ID in the local connection description. Range of values: 0001h ... 0FFFh
DONE	OUTPUT	BOOL	I, Q, M, D, L	*DONE* status parameter: ■ 0: Job not yet started or still running. ■ 1: Job executed without error.
BUSY	OUTPUT	BOOL	I, Q, M, D, L	■ *BUSY* = 1: Job is not yet completed. ■ *BUSY* = 0: Job is completed.
ERROR	OUTPUT	BOOL	I, Q, M, D, L	*ERROR* status parameter: ■ *ERROR* = 1: Error occurred during processing. *STATUS* provides detailed information on the type of error.
STATUS	OUTPUT	WORD	M, D	*STATUS* status parameter: Error information
CONNECT	IN_OUT	ANY	D	Pointer to the associated connection description. *Chapter 6.1.6 'UDT 65 - TCON_PAR Data structure for FB 65' on page 131*

Error information

ERROR	STATUS	Explanation
0	0000h	Connection was able to be established
0	7000h	Call with *REQ* = 0, establishment of connection not initiated
0	7001h	First call with *REQ* = 1, connection being established
0	7002h	Follow-on call (*REQ* irrelevant), connection being established
1	8086h	The ID parameter must not have value of zero.
0	8087h	Maximal number of connections reached; no additional connection possible
1	809Bh	The local_device_id in the connection description does not match the target CPU.
1	80A3h	Attempt being made to re-establish an existing connection.
1	80A7h	Communications error: you have called TDISCON before TCON was complete. TDISCON must first complexly terminate the connection referenced by the ID.
1	80B3h	Inconsistent parameters: ■ Error in the connection description ■ Local port (parameter local_tsap_id) is already present in another connection description ■ ID in the connection description different from the ID specified as parameter
1	80B4h	When using the protocol variant ISO on TCP (connection_type = 12h) for passive establishment of a connection (active_est = FALSE), you violated one or both of the following conditions: "local_tsap_id_len >= 02h" and/or "local_tsap_id[1] = E0h".
1	80C3h	Temporary lack of resources in the CPU.
1	80C4h	Temporary communications error: ■ The connection cannot be established at this time. ■ The interface is receiving new parameters.
1	8722h	*CONNECT* parameter: Source area invalid: area does not exist in DB
1	8732h	*CONNECT* parameter: The DB number lies outside the CPU-specific number range.
1	873Ah	*CONNECT* parameter: Access to connection description not possible (e.g. DB not available)
1	877Fh	*CONNECT* parameter: Access to connection description not possible (e.g. DB not available)

6.1.6 UDT 65 - TCON_PAR Data structure for FB 65

6.1.6.1 Data structure for assigning connection

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

Data structure

Byte	Parameter	Data type	Start value	Description
0 ... 1	block_length	WORD	40h	Length of UDT 65: 64 Bytes (fixed)
2 ... 3	id	WORD	0000h	■ Reference to the connection (range of values: 0001h ... 0FFFh) ■ You must specify the value of the parameter in the respective block with *ID*.
4	connection _type	BYTE	01h	Connection type: ■ 11h: TCP/IP native ■ 12h: ISO on TCP ■ 01h: TCP/IP native (Compatibility mode)
5	active_est	BOOL	FALSE	ID for the way the connection is established: ■ FALSE: passive establishment ■ TRUE: active establishment
6	local_device_id	BYTE	02h	Communicaton device ■ 00h: Ethernet PG/OP channel of the CPU ■ 02h: Ethernet CP of the CPU
7	local_tsap_id_len	BYTE	02h	Length of parameter *local_tsap_id* used; possible values: ■ 0 or 2, if *connection_type* = 01h or 11h. For the active side, only the value 00h is permitted. ■ 2 to 16, if *connection_type* = 12h
8	rem_subnet_id_len	BYTE	00h	This parameter is currently not used. You must assign 00h to it.

Byte	Parameter	Data type	Start value	Description
9	rem_staddr_len	BYTE	00h	Length of address for the remote connection transmission point: ■ 0: unspecified, i.e. parameter *rem_staddr* is irrelevant. 4: valid IP address in the parameter *rem_staddr*
10	rem_tsap_id_len	BYTE	00h	Length of parameter *local_tsap_id* used; possible values: ■ 0 or 2, if *connection_type* = 01h or 11h. For the passive side, only the value 00h permitted. ■ 2 ... 16, if *connection_type* = 12h
11	next_staddr_len	BYTE	00h	Length of parameter *next_staddr* used ■ 00h: Ethernet CP of the CPU ■ 01h: Ethernet PG/OP channel of the CPU
12 ... 27	local_tsap_id	ARRAY [1..16] of BYTE	00h ...	With *connection_type* = ■ 11h: local port no. (possible values: 2000 ... 5000) – local_tsap_id[1] = high byte of port no. in hexadecimal representation local_tsap_id[2] = low byte of port no. in hexadecimal representation local_tsap_id[3-16] = irrelevant ■ 12h: local TSAP-ID: – local_tsap_id[1] = E0h (connection type T-connection) – local_tsap_id[2] = Rack and slot in own CPU (bits 0 to 4 slot, bits 5 to7: rack number) – local_tsap_id[3-16] = TSAP extension ■ 01h: local port no. (possible values: 2000 ... 5000) – local_tsap_id[1] = low byte of Port-Nr. in hexadecimal representation – local_tsap_id[2] = high byte of port no. in hexadecimal representation – local_tsap_id[3-16] = irrelevant **Note:** Make sure that each value of *local_tsap_id* that you use in your CPU is unique.

Byte	Parameter	Data type	Start value	Description
28 ... 33	rem_subnet_id	ARRAY [1..6] of BYTE	00h ...	This parameter is currently not used. You must assign 0 to it.
34 ... 39	rem_staddr	ARRAY [1..6] of BYTE	00h ...	IP address for the remote connection transmission point: e.g. 192.168.002.003: With *connection_type* = ■ 1xh rem_staddr[1] = C0h (192) rem_staddr[2] = A8h (168) rem_staddr[3] = 02h (002) rem_staddr[4] = 03h (003) rem_staddr[5-6] = irrelevant ■ 01h rem_staddr[1] = 03h (003) rem_staddr[2] = 02h (002) rem_staddr[3] = A8h (168) rem_staddr[4] = C0h (192) rem_staddr[5-6] = irrelevant
40 ... 55	rem_tsap_id	ARRAY [1..16] of BYTE	00h ...	With *connection_type* = ■ 11h: remote port no. (possible values: 2000 ... 5000) – rem_tsap_id[1] = high byte of port no in hexadecimal representation – rem_tsap_id[2] = low byte of port no in hexadecimal representation – rem_tsap_id[3-16] = irrelevant – rem_tsap_id[3-16] = irrelevant ■ 12h: remote TSAP-ID: – rem_tsap_id[1] = E0h (connection type T-connection) – rem_tsap_id[2] = Rack and slot for the remote connection transmission point CPU (bits 0 to 4: slot, bits 5 ... 7: rack number), – rem_tsap_id[3-16] = TSAP extension ■ 01h: remote port no. (possible values: 2000 ... 5000) – local_tsap_id[1] = low byte of port no. in hexadecimal representation – local_tsap_id[2] = high byte of port no. in hexadecimal representation – local_tsap_id[3-16] = irrelevant

Byte	Parameter	Data type	Start value	Description
56 ... 61	next_staddr	ARRAY [1..6] of BYTE	00h ...	At *local_device_id* = ■ 00h (Ethernet P/OP channel) – next_staddr[1]: 04h – next_staddr[2-6]: 00h ■ 02h (Ethernet CP) – next_staddr[1-6]: 00h
62 ... 63	spare	WORD	0000h	irrelevant

6.1.6.2 Data structure for communications access point

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

Data structure

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

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

6.1.7 FB 66 - TDISCON - Terminating a connection

Use with TCP native and ISO on TCP

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

Use with UDP

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

Description

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

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

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

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

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

Parameters

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

Error information

ERROR	STATUS	Explanation
0	0000h	Connection was able to be terminated
0	7000h	First call with *REQ* = 0, termination of connection not initiated
0	7001h	First call with *REQ*= 1, connection being terminated
0	7002h	Follow-on call (*REQ* irrelevant), connection being terminated
1	8086h	The ID parameter is not in the permitted address range
1	80A3h	Attempt being made to terminate a non-existent connection
1	80C4h	Temporary communications error: The interface is receiving new parameters.

6.2 Ethernet Communication

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

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

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

Communication blocks

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

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

Status displays

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

Deployment at high communication load

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

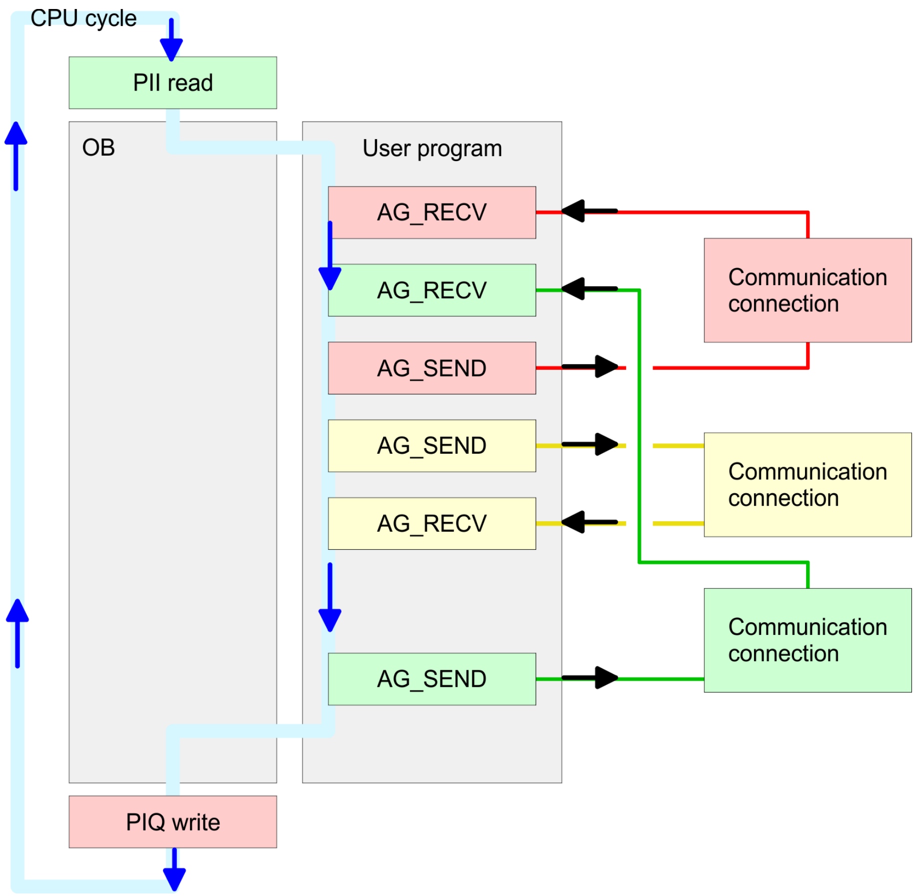
FC call is faster than CP transfer time

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

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

AG_SEND, AG_RECV in user application

The following illustration shows a possible sequence for the FC blocks together with the organizations and program blocks in the CPU cycle:



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

6.2.2 FC 5 - AG_SEND - send to CP 343

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

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

Parameter

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

DONE, ERROR, STATUS

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

DONE (SEND)	NDR (RECV)	ERROR	STATUS	Description
1	-	0	0000h	Job finished without error.
-	1	0	0000h	New data taken without error.
0	-	0	0000h	There is no job being executed
-	0	0	8180h	No data available yet.
0	0	0	8181h	Job running
0	0	1	8183h	No CP project engineering for this job.
0	-	1	8184h	System error occurred
-	0	1	8184h	System error occurred (source data area failure).
0	-	1	8185h	Parameter *LEN* exceeds source area *SEND*.
	0	1	8185h	Destination buffer (RECV) too small.

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

DONE (SEND)	NDR (RECV)	ERROR	STATUS	Description
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 from module.
0	0	1	80A4h	reserved
0	0	1	80B0h	Module doesn't recognize the record set.
0	0	1	80B1h	The length setting (in parameter *LEN*) is invalid.
0	0	1	80B2h	reserved
0	0	1	80C0h	Record set not readable.
0	0	1	80C1h	The set record set is still in process.
0	0	1	80C2h	There is a job jam.
0	0	1	80C3h	The operating sources (memory) of the CPU are temporarily occupied.
0	0	1	80C4h	Communication error (occurs temporarily; a repetition in the user application is reasonable).
0	0	1	80D2h	Module start address is wrong.

Status parameter at reboot

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

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

6.2.3 FC 6 - AG_RECV - receive von CP 343

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

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *↳ Chapter 4 'Include VIPA library' on page 103*

Parameter

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

DONE, ERROR, STATUS

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

DONE (SEND)	NDR (RECV)	ERROR	STATUS	Description
1	-	0	0000h	Job finished without error.
-	1	0	0000h	New data taken without error.
0	-	0	0000h	There is no job being executed
-	0	0	8180h	No data available yet.
0	0	0	8181h	Job running
0	0	1	8183h	No CP project engineering for this job.
0	-	1	8184h	System error occurred
-	0	1	8184h	System error occurred (source data area failure).
0	-	1	8185h	Parameter *LEN* exceeds source area *SEND*.
	0	1	8185h	Destination buffer (RECV) too small.
0	0	1	8186h	Parameter *ID* invalid (not within 1 ...16).

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

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

Status parameter at reboot

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

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

6.2.4 FC 10 - AG_CNTRL - Control CP 343

Description

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

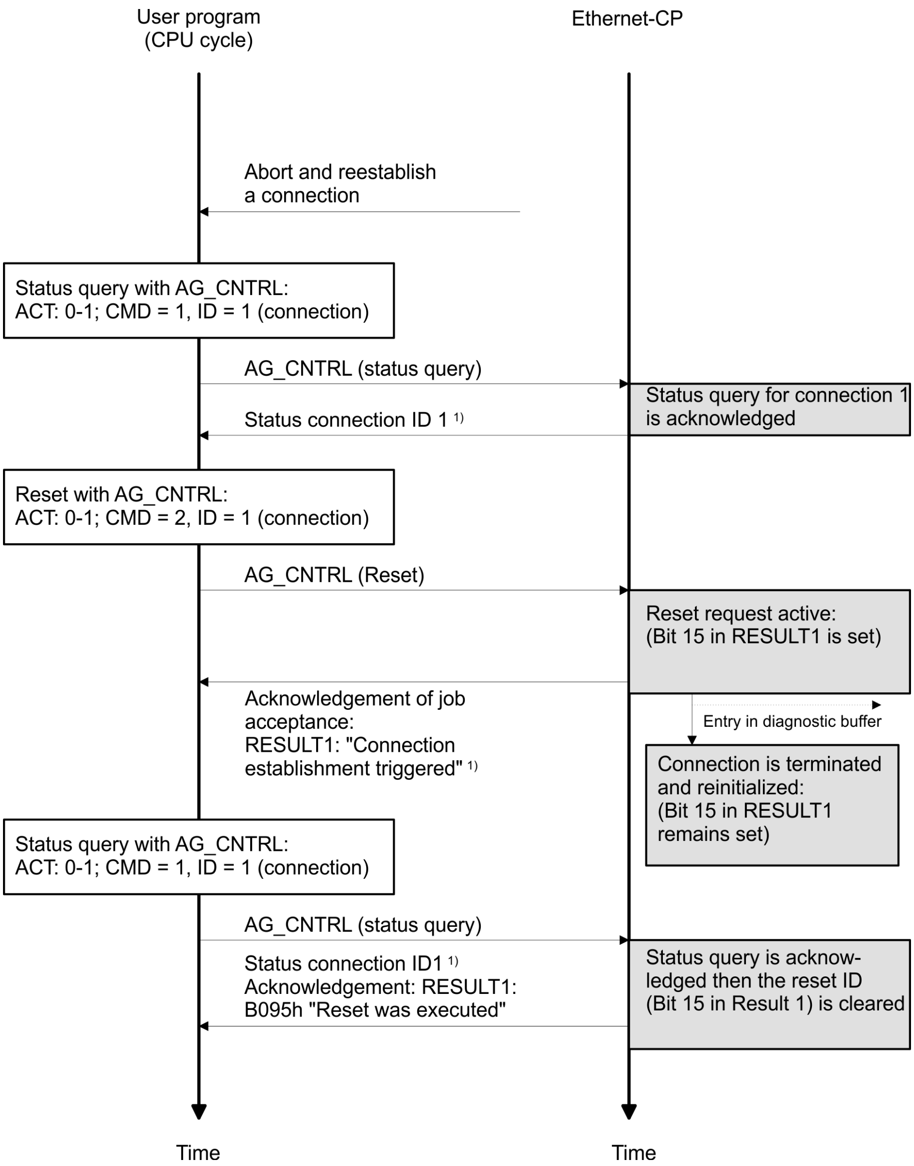
The following jobs may be executed by parameterizable commands:

- Reading connection information
- Resetting configured connections

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

FC 10 in the user program

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



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

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

Parameters

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

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

ACT

Possible values: 0, 1

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

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

ID

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

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

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

LADDR

Module base address

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

Specify this address here.

CMD

Command to the FC AG_CNTRL

DONE

0: Job is still being processed or not yet triggered

1: Job executed

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

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

ERROR

0: No error

1: Error indication

STATUS

Status indication

RESULT1/2

Information returned according to the command sent to the FC AG_CNTRL

DONE, ERROR, STATUS

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

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

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

Status parameter at cold restart

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

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

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

Commands and evaluating the job results

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

CMD 0

NOP - no operation

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

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

CMD 1

CN_STATUS - connection status

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

RESULT	Hex value/range	Description
RESULT 1	0000 000xh	Bit 3 ... 0: Codes for the send direction (excluded: 0010_b) Bit 0: Connection reserved for send and receive jobs Bit 1: Send job being executed Bit 3, 2: Previous job 00: No information 01: Send job completed successful 10: Send job not completed successfully
	0000 00x0h	Bit 7 ... 4: Codes for receive direction (excluded: 0010_b) Bit 4: Connection reserved for send and receive jobs Bit 5: Receive job being executed Bit 7, 6: Previous job 00: No information 01: Receive job completed successfully 10: Receive job not completed successfully

RESULT	Hex value/range	Description
	0000 0x00h	Bit 11 ... 8: Codes for FETCH/WRITE (excluded: 0011_b, 0111_b, 1000_b, 1011_b, 0010_b) Bit 8: Connection type 0: No FETCH connection 1: Connection reserved for FETCH jobs Bit 9: Connection type 0: No WRITE connection 1: Connection reserved for WRITE jobs Bit 10: Job status (FETCH/ WRITE) 0: Job status OK 1: Job status not OK This ID is set in the following situations: - The job was acknowledged negatively by the CPU - The job could not be forwarded to the CPU because the connection was in the "LOCKED" status. - The job was rejected because the FETCH/WRITE header did not have the correct structure. Bit 11: Status of FETCH/WRITE job 0: No job active 1: Job from LAN active
	0000 x000h	Bit 15 ... 12: General CP information (excluded: 0011_b, 1011_b) Bit 13, 12: Connection status (only available for SEND/RECV connections based on the ISO/RFC/TCP protocols; with UDP, the corresponding internal information is output) 00: Connection is terminated 01: Connection establishment active 10: Connection termination active 11: Connection is established Bit 14: CP information 0: CP in STOP 1: CP in RUN Bit 15: Reset ID 0: FC 10 has not yet reset a connection or the reset ID was cleared. 1: The FC 10 has executed a connection reset

RESULT	Hex value/range	Description
	xxxx 0000h	Bit 31 ... 16: Reserved for later expansions
RESULT 2	0000 0000h	Reserved for later expansions

CMD 2

CN_RESET - connection reset

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

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

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

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

CMD 3

CN_STATUS_ALL - all connections status

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

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

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

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

CMD 4

CN_RESET_ALL - all connections reset

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

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

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

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

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

CMD 5

CN_CLEAR_RESET - Clear the reset ID

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

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

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

CMD 6

CN_DISCON - connection disconnect

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

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

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

CMD 7

CN_STARTCON - start connection

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

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

6.2.5 FC 62 - C_CNTR - Querying the Connection Status

Description

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

Parameters

Param-eter	Declara-tion	Data Type	Memory Area	Description
EN_R	INPUT	BOOL	E, A, M, D, L, Konst.	Control parameter enabled to receive, signals ready to receive if the input is set.
ID	INPUT	WORD	M, D, Konst.	Addressing parameter *ID*,
RET_VAL	OUTPUT	INT	E, A, M, D, L	Error information
ERROR	OUTPUT	BOOL	E, A, M, D, L	Status parameter *ERROR* and *STATUS*

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

Error Information

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

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

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

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

6.2.6 FB/SFB 8 - FB 55 - Overview

With the Siemens S7 connection large data sets may be transferred between via Ethernet connected PLC systems based on Siemens STEP®7.® The communication connections are static i.e. they are to be configured in a connection table.

Possibilities of communication functions

- Siemens S7-300 communication functions
  - By including the VIPA specific function blocks FB 8 ... FB 55 you get access to the Siemens S7-300 communication functions. *Chapter 4 'Include VIPA library' on page 103*
- Siemens S7-400 communication functions
  - To deploy the Siemens S7-400 communication functions the in the operating system of the CPU integrated system function blocks SFB 8 ... SFB 23 should be used. Here copy the interface description of the SFBs from the standard library at system function block to the directory container, generate an instance data block for each call and call the SFB with the associated instance data block.

Project engineering

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

Function blocks

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

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

6.2.7 FB/SFB 8 - USEND - Uncoordinated data transmission

Description

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

Depending upon communication function the following behavior is present:

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

Parameters

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

You must, however, make sure that the areas defined by the parameters SD_1/SD_1...SD_4 and RD_1/RD_1...RD_4 (at the corresponding partner FB/SFB URCV) agree in Number, Length and Data type.

The parameter R_ID must be identical at both FB/SFBs. Successful completion of the transmission is indicated by the status parameter DONE having the logical value 1.

Fehlerinformationen

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

Data consistency

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

6.2.8 FB/SFB 9 - URCV - Uncoordinated data reception

Description

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

Depending upon communication function the following behavior is present:

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

Parameters

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

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

Error information

ERROR	STATUS (decimal)	Description
0	9	Overrun warning: old receive data was overwritten by new receive data.
0	11	Warning: the new task is not active since the previous task has not completed.
0	25	Communications initiated. The task is being processed.
1	1	Communication failures, e.g. ■ Connection parameters not loaded (local or remote) ■ Connection interrupted (e.g. cable, CPU turned off, CP in STOP)
1	4	Error in receive buffer pointer *RD_i* with respect to the length or the data type.
1	10	Access to local application memory not possible (e.g. access to deleted DB).
1	12	The call to the FB/SFB ■ contains an instance DB that does not belong to the FB/SFB 9 ■ contains a global DB instead of an instance DB ■ could not locate an instance DB (load a new instance DB from the PG)
1	18	*R_ID* already exists in the connection ID.
1	19	The respective FB/SFB USEND transmits data quicker than FB/SFB URCV can copy the data into the receive buffers.
1	20	Not enough memory.

Data consistency

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

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

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

Description

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

Depending upon communication function the following behavior is present:

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

Function

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

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

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

Parameters

Parameter	Declaration	Data type	Memory block	Description
REQ	INPUT	BOOL	I, Q, M, D, L	Control parameter request, a rising edge activates the data exchange (with respect to the most recent FB/SFB call)
R	INPUT	BOOL	I, Q, M, D, L, constant	control parameter reset: terminates the active task
ID	INPUT	WORD	I, Q, M, D, constant	A reference for the connection. Format W#16#xxxx
R_ID	INPUT	DWORD	I, Q, M, D, L, constant	Address parameter *R_ID*. Format DW#16#wxyzWXYZ.
DONE	OUTPUT	BOOL	I, Q, M, D, L	Status parameter *DONE*: 0: task has not been started or is still being executed. 1: task was executed without error.

Parameter	Declaration	Data type	Memory block	Description
ERROR	OUTPUT	BOOL	I, Q, M, D, L	Status parameter *ERROR*: ■ *ERROR* = 0 + *STATUS* = 0000h – No warnings or errors. ■ *ERROR* = 0 + *STATUS* unequal to 0000h – A Warning has occurred. *STATUS* contains detailed information. ■ *ERROR* = 1 – An error has occurred.
STATUS	OUTPUT	WORD	I, Q, M, D, L	Status parameter *STATUS*, returns detailed information about the type of error.
SD_1	IN_OUT	ANY	I, Q, M, D, T, C	Pointer to the send data buffer. The length parameter is only utilized when the block is called for the first time after a start. It specifies the maximum length of the send buffer. Only data type BOOL is valid (Bit field not permitted), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER.
LEN	IN_OUT	WORD	I, Q, M, D, L	The length of the send data block in bytes.

Error information

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

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

Data consistency

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

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

Description

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

Depending upon communication function the following behavior is present:

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

> ***VIPA specific block***
>
> *The VIPA specific blocks can be found in the VIPA library.* *⇨ Chapter 4 'Include VIPA library' on page 103*

Parameters

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

Function

- The FB/SFB 13 is ready for reception when control input *EN_R* is set to 1. Parameter *RD_1* specifies the start address of the receive data buffer. An acknowledgment is returned to the partner FB/SFB after reception of each data segment and parameter *LEN* of the FB/SFB 13 is updated accordingly. If the block is called during the asynchronous reception process a warning is issued via the status parameter *STATUS*.
- Should this call be received with control input *EN_R* set to 0 then the receive process is terminated and the FB/SFB is reset to its initial state. When all data segments have been received without error parameter *NDR* is set to 1. The received data remains unaltered until FB/SFB 13 is called again with parameter *EN_R* = 1.

Error information

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

ERROR	STATUS (decimal)	Description
1	18	*R_ID* already exists in the connection *ID*.
1	20	Not enough memory.

Data consistency

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

- When copying has been completed (parameter *NDR* is set to 1) FB/SFB 13 must again be called with parameter *EN_R* set to 0 in order to ensure that the receive data block is not overwritten before it has bee evaluated.
- The most recently used receive data block *RD_1* must have been evaluated completely before the block is denoted as being ready to receive (calls with parameter *EN_R* set to 1).

Receiving Data S7-400

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

6.2.11 FB/SFB 14 - GET - Remote CPU read

Description

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

Depending upon communication function the following behavior is present:

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

VIPA specific block
The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

Parameters

Parameter	Declara-tion	Data type	Memory block	Description
REQ	INPUT	BOOL	I, Q, M, D, L	control parameter request, a rising edge activates the data exchange (with respect to the most recent FB/SFB-call)
ID	INPUT	WORD	I, Q, M, D, constant	A reference for the connection. Format: W#16#xxxx
NDR	OUTPUT	BOOL	I, Q, M, D, L	Status parameter *NDR*: data from partner CPU has been accepted.
ERROR	OUTPUT	BOOL	I, Q, M, D, L	Status parameter *ERROR*: ■ *ERROR* = 0 + *STATUS* = 0000h – No warnings or errors. ■ *ERROR* = 0 + *STATUS* unequal to 0000h – A Warning has occurred. *STATUS* contains detailed information. ■ *ERROR* = 1 – An error has occurred.
STATUS	OUTPUT	WORD	I, Q, M, D, L	Status parameter *STATUS*, returns detailed information about the type of error.
ADDR_1	IN_OUT	ANY	e.g. I, Q, M, D	Pointer indicating the buffers in the partner CPU that must be read
ADDR_2	IN_OUT	ANY	e.g. I, Q, M, D	Pointer indicating the buffers in the partner CPU that must be read
ADDR_3	IN_OUT	ANY	e.g. I, Q, M, D	Pointer indicating the buffers in the partner CPU that must be read
ADDR_4	IN_OUT	ANY	e.g. I, Q, M, D	Pointer indicating the buffers in the partner CPU that must be read
RD_i,1≤ I ≤4	IN_OUT	ANY	I, Q, M, D, T, C	Pointers to the area of the local CPU in which the read data are entered. Only data type BOOL is valid (bit field not permitted), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER.

Function

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

Error information

ERROR	STATUS (decimal)	Description
0	11	Warning: the new task is not active since the previous task has not completed.
0	25	The communication process was initiated. The task is being processed.
1	1	Communication failures, e.g. ■ Connection parameters not loaded (local or remote) ■ Connection interrupted (e.g.: cable, CPU turned off, CP in STOP)
1	2	Negative acknowledgment from partner device. The function cannot be executed.
1	4	Error in receive data buffer pointer *RD_i* with respect to the length or the data type.
1	8	Partner CPU access error
1	10	Access to local application memory not possible (e.g. access to deleted DB).
1	12	The call to the FB/SFB ■ contains an instance DB that does not belong to the FB/SFB 14 ■ contains a global DB instead of an instance DB ■ could not locate an instance DB (load a new instance DB from the PG)
1	20	Not enough memory.

Data consistency

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

6.2.12 FB/SFB 15 - PUT - Remote CPU write

Description

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

Depending upon communication function the following behavior is present:

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

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

Parameters

Parameter	Declaration	Data type	Memory block	Description
REQ	INPUT	BOOL	I, Q, M, D, L	control parameter request, a rising edge activates the data exchange (with respect to the most recent FB/SFB-call)
ID	INPUT	WORD	I, Q, M, D, constant	A reference for the connection. Format W#16#xxxx
DONE	OUTPUT	BOOL	I, Q, M, D, L	Status parameter *DONE*: function completed.
ERROR	OUTPUT	BOOL	I, Q, M, D, L	Status parameter *ERROR*: ■ *ERROR* = 0 + *STATUS* = 0000h – No warnings or errors. ■ *ERROR* = 0 + *STATUS* unequal to 0000h – A Warning has occurred. *STATUS* contains detailed information. ■ *ERROR* = 1 – An error has occurred.
STATUS	OUTPUT	WORD	I, Q, M, D, L	Status parameter *STATUS*, returns detailed information about the type of error.
ADDR_1	IN_OUT	ANY	e.g. I, Q, M, D	Pointer indicating the buffers in the partner CPU into which data is written

Parameter	Declaration	Data type	Memory block	Description
ADDR_2	IN_OUT	ANY	e.g. I, Q, M, D	Pointer indicating the buffers in the partner CPU into which data is written
ADDR_3	IN_OUT	ANY	e.g. I, Q, M, D	Pointer indicating the buffers in the partner CPU into which data is written
ADDR_4	IN_OUT	ANY	e.g. I, Q, M, D	Pointer indicating the buffers in the partner CPU into which data is written
SD_i,1≤I ≤4	IN_OUT	ANY	I, Q, M, D, T, C	Pointer to the data buffers in the local CPU that contains the data that must be sent. Only data type BOOL is valid (Bit field not permitted), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER.

Function

- The partner CPU stores the data at the respective address and returns an acknowledgment.
- This acknowledgment is tested and when an error is detected in the data transfer parameter *DONE* is set to 1 with the next call of FB/SFB 15.
- The write process can only be activated again when the most recent write process has been completed. The amount, length and data type of the buffer areas that were defined by means of parameters *ADDR_i* and *SD_i*, 1 ≤ I ≤ 4 must be identical.

Error information

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

ERROR	STATUS (decimal)	Description
1	12	The call to the FB/SFB contains an instance DB that does not belong to the FB/SFB 15. contains a global DB instead of an instance DB. could not locate an instance DB (load a new instance DB from the PG).
1	20	Not enough memory.

Data consistency

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

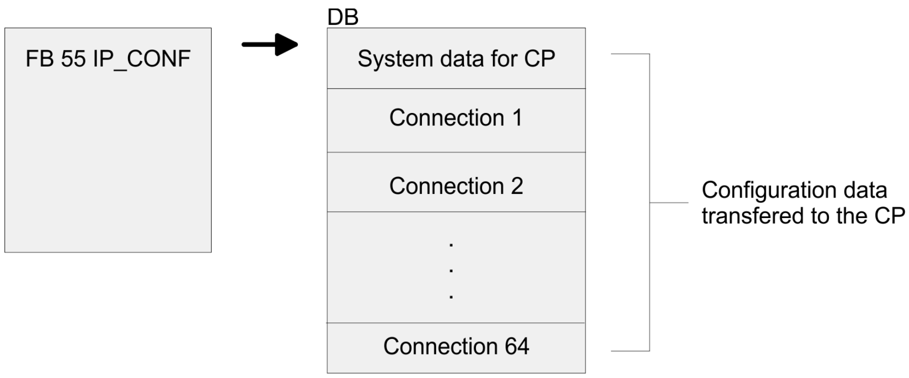
6.2.13 FB 55 - IP_CONF - Progr. Communication Connections

Overview

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

Principle

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



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

6.2.13.1 FB 55 - IP_CONF

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

The Job is started with *ACT* = 1.

VIPA specific block
The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

Parameters

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

Parameter	Declaration	Data type	Memory block	Description
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.

ERROR	STATUS	Description
1	8A06h	A job is complete but the FB nevertheless triggered the first job for a read record sets job.
1	8B01h	Communication error, the DB could not be transferred.
1	8B02h	Parameter error, double parameter field
1	8B03h	Parameter error, the subfield in the parameter field is not permitted.
1	8B04h	Parameter error, the length specified in the FB does not match the length of the parameter fields/subfields.
1	8B05h	Parameter error, double parameter field.
1	8B06h	Parameter error, the subfield in the parameter field is not permitted.
1	8B07h	Parameter error, the length of the parameter field is invalid.
1	8B08h	Parameter error, the ID of the subfield is invalid.
1	8B09h	System error, the connection does not exist.
1	8B0Ah	Data error, the content of the subfield is not correct.
1	8B0Bh	Structure error, a subfield exists twice.
1	8B0Ch	Data error, the parameter does not contain all the necessary parameters.
1	8B0Dh	Data error, the *CONF_DB* does not contain a parameter field for system data.
1	8B0Eh	Data error/structure error, the *CONF_DB* type is invalid.
1	8B0Fh	System error, the CP does not have enough resources to process *CONF_DB* completely.
1	8B10	Data error, configuration by the user program is not set.
1	8B11	Data error, the specified type of parameter field is invalid.
1	8B12	Data error, too many connections were specified.
1	8B13	CP internal error
1	8F22h	Area length error reading a parameter.
1	8F23h	Area length error writing a parameter.
1	8F24h	Area error reading a parameter.
1	8F25h	Area error writing a parameter.
1	8F28h	Alignment error reading a parameter.
1	8F29h	Alignment error writing a parameter.
1	8F30h	The parameter is in the write-protected first current data block.
1	8F31h	The parameter is in the write-protected second current data block.
1	8F32h	The parameter contains a DB number that is too high.
1	8F33h	DB number error
1	8F3Ah	The target area was not loaded (DB).

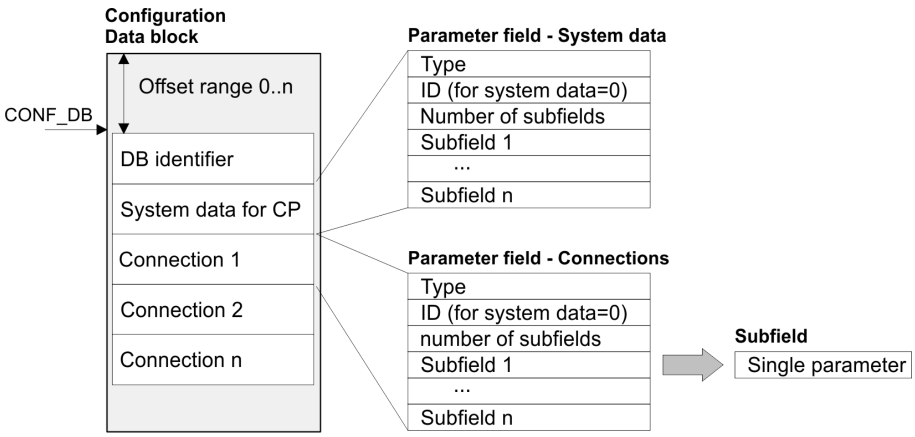
ERROR	STATUS	Description
1	8F42h	Timeout reading a parameter from the I/O area.
1	8F43h	Timeout writing a parameter from the I/O area.
1	8F44h	Address of the parameter to be read is disabled in the accessed rack.
1	8F45h	Address of the parameter to be written is disabled in the accessed rack.
1	8F7Fh	Internal error

6.2.13.2 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
51	MAC-ADR	4 + 6	MAC address local node		optional

Parameter fields for Connections

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

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

TCP connection

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

Subfield				Parameter	
ID	**Type**	**Length (byte)**	**Description**	**Special features**	**Use**
1	SUB_IP_V4	4 + 4	IP address of the remote station according to IPv4		mandatory[1]
9	SUB_LOC_PORT	4 + 2	Port of the local station		mandatory
10	SUB_REM_PORT	4 + 2	Port of the remote station		mandatory[1]
18	SUB_CON-NECT_NAME	Length Name + 4	Name of the connection		optional
19	SUB_LOC_MODE	4 + 1	Local mode of the connection, Possible values: 0x00 = SEND/REC 0x10 = S5-addressing mode for FETCH/WRITE [2] 0x80 = FETCH[2] 0x40 = WRITE [2] If you do not set the parameter, the default setting is SEND/RECV. For FETCH/WRITE a passive connection setup is necessary.		optional
21	SUB_KBUS_ADR	-	-	Value: fix 2	optional
22	SUB_CON_ESTABL	4 + 1	Type of connection establishment. With this option, you specify whether the connection is estab-lished by this station. Possible values: 0 = passive 1 = active		mandatory
1) Option using passive connection 2) the coding may be combined with OR operations					

UDP connection

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

Subfield				Parameter	
ID	**Type**	**Length(byte)**	**Description**	**Special features**	**Use**
1	SUB_IP_V4	4 + 4	IP address of the remote station according to IPv4		mandatory
9	SUB_LOC_PORT	4 + 2	Port of the local station		mandatory
10	SUB_REM_PORT	4 + 2	Port of the remote station		mandatory
18	SUB_CON-NECT_NAME	Length Name + 4	Name of the connection		optional
19	SUB_LOC_MODE	4 + 1	Local mode of the connection Possible values: 0x00 = SEND/REC0x10 = S5-addressing mode for FETCH/WRITE [1] 0x80 = FETCH [1] 0x40 = WRITE [1] If you do not set the parameter, the default setting is SEND/RECV. For FETCH/WRITE a passive connection setup is necessary		optional
21	SUB_KBUS_ADR	-	-	Value: fix 2	optional
23	SUB_ADDR_IN_DATA_ BLOCK	4 + 1	Select free UDP connection. The remote node is entered in the job header of the job buffer by the user program when it calls AG_SEND. This allows any node on Ethernet/LAN/WAN to be reached. Possible values: 1 = free UDP connection 0 = otherwise		optional
1) the coding may be combined with OR operations					

ISO-on-TCP connection

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

Subfield				Parameter	
ID	**Type**	**Length (byte)**	**Description**	**Special features**	**Use**
1	SUB_IP_V4	4 + 4	IP address of the remote station according to IPv4		mandatory[1]
11	SUB_LOC_PORT	Length TSAP + 4	TSAP of the local station		mandatory
12	SUB_REM_PORT	Length TSAP + 4	TSAP of the remote station		mandatory[1]
18	SUB_CON-NECT_NAME	Length Name + 4	Name of the connection		optional
19	SUB_LOC_MODE	4 + 1	Local mode of the connection Possible values: 0x00 = SEND/RECV 0x10 = S5-addressing mode for FETCH/WRITE [2] 0x80 = FETCH [2] 0x40 = WRITE [2] If you do not set the parameter, the default setting is SEND/RECV. For FETCH/WRITE a passive connection setup is necessary		optional
21	SUB_KBUS_ADR	-	-	Value: fix 2	optional
22	SUB_CON_ESTABL	4 + 1	Type of connection establishment With this option, you specify whether the connection is established by this station. Possible values: 0 = passive 1 = active		mandatory
1) option using passive connection 2) the coding may be combined with OR operation					

H1 connection (ISO)

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

Subfield				Parameter	
ID	Type	Length (byte)	Description	Special features	Use
51	SUB_MAC	4 + 6	MAC address of the remote station		mandatory
11	SUB_LOC_TSAP	Length TASP + 4	TSAP of the local station		mandatory
12	SUB_REM_TSAP	Length TASP + 4	TSAP of the remote station		mandatory[1]
18	SUB_CONNECT_NAME	Length Name + 4	Name of the connection		optional
19	SUB_LOC_MODE	4 + 1	Local mode of the connection Possible values: 0x00 = SEND/RECV 0x10 = S5-addressing mode for FETCH/WRITE [2] 0x80 = FETCH [2] 0x40 = WRITE [2] If you do not set the parameter, the default setting is SEND/RECV.For FETCH/WRITE a passive connection setup is necessary		optional
22	SUB_CON_ESTABL	4 + 1	Type of connection establishment With this option, you specify whether the connection is established by this station. Possible values: 0 = passive; 1 = active		mandatory
52	SUB_TIME_CON_RETRAN	4 + 2	Time interval after which a failed connection is established again. (1...60s, default: 5s)	irrelevant with passive connection establishment	optional
53	SUB_TIME_DAT_RETRAN	4 + 2	Time interval after which a failed send is triggered again. (100...30000ms, default: 1000ms)		optional
54		4 + 2	Number of send attempts, incl 1. attempt(1...100, Default: 5)		optional

Subfield			Parameter		
ID	Type	Length (byte)	Description	Special features	Use
55		4 + 2	Time interval after which a connection is released, if there is no responds of the partner station. (6...160s, default: 30s)		optional
1) option using passive connection 2) the coding may be combined with OR operation					

Siemens S7 connection

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

Subfield			Parameter		
ID	Type	Length (byte)	Description	Special features	Use
56	SUB_S/_C_DETAIL	4 + 14	Connection specific parameter		mandatory
18	SUB_CONNECT_NAME	Length-Name + 4	Name of the connection		optional
1	SUB_IP_V4	4 + 4	IP address according to IPv4	IP address of the remote partner	mandatory[1]
51	SUB_MAC	4 + 6	MAC address of the remote station		mandatory
22	SUB_CON_ESTABL	4 + 1	Type of connection establishment. With this option, you specify whether the connection is established by this station. Possible values: 0 = passive 1 = active		mandatory
1) option using passive connection					

SUB_S/_C_DETAIL

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

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

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

Example for configuring a Siemens S7 connection

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

Properties - Siemens S7- Connection	
Siemens NetPro	**FB55 - IP_CONFIG**
establish an active connection	SUB_CON_ESATBL.CON_ESTABL
TCP/IP	SUB_S7_C_DETAILS.TcpIpActive
IP respectively MAC address remote station	SUB_IP_V4.rem_IP.IP_0...IP_3 resp. SUB_MAC.rem_MAC.MAC_0...MAC 5
Local ID	Connection ID

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

Additional Parameter fields

Block_VIPA_HWK

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

Type = 99
ID = 0
Number of subfields = 0

Block_VIPA_ BACNET

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

Type = 100
Number of subfields = 0

Block_VIPA_IPK

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

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

Example DB

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

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

Address	Name	Type	Initial value	Actual	Comment
112.0	S7_Verb.Verb_Parameter. SUB_S7_C_DETAIL_LEN	INT	18	18	
114.0	S7_Verb.Verb_Parameter.TcpIpActive	INT	0	1	
116.0	S7_Verb.Verb_Parameter.LocalResource	INT	0	2	
118.0	S7_Verb.Verb_Parameter.LocalRack	INT	0	0	
120.0	S7_Verb.Verb_Parameter.LocalsSlot	INT	0	2	
122.0	S7_Verb.Verb_Parameter.RemoteResource	INT	0	2	
124.0	S7_Verb.Verb_Parameter.RemoteRack	INT	0	0	
126.0	S7_Verb.Verb_Parameter.RemoteSlot	INT	0	2	
128.0	S7_Verb.ipl.SUB_IP_V4	WORD	W#16#1	W#16#1	
130.0	S7_Verb.ipl. SUB_IP_V4_LEN	WORD	W#16#8	W#16#8	
132.0	S7_Verb.ipl.IP_0	BYTE	B#16#0	B#16#AC	
133.0	S7_Verb.ipl.IP_1	BYTE	B#16#0	B#16#10	
134.0	S7_Verb.ipl.IP_2	BYTE	B#16#0	B#16#8B	
135.0	S7_Verb.ipl.IP_3	BYTE	B#16#0	B#16#62	
136.0	S7_Verb.Mac.SUB_MAC	INT	51	51	
138.0	S7_Verb.Mac.SUB_MAC_LEN	INT	10	10	
140.0	S7_Verb.Mac.MAC_0	BYTE	B#16#0	B#16#0	
141.0	S7_Verb.Mac.MAC_1	BYTE	B#16#0	B#16#20	
142.0	S7_Verb.Mac.MAC_2	BYTE	B#16#0	B#16#D5	
143.0	S7_Verb.Mac.MAC_3	BYTE	B#16#0	B#16#77	
144.0	S7_Verb.Mac.MAC_4	BYTE	B#16#0	B#16#53	
145.0	S7_Verb.Mac.MAC_5	BYTE	B#16#0	B#16#9B	
146.0	S7_Verb.con_est .SUB_CON_ESTABL	WORD	W#16#16	W#16#16	
148.0	S7_Verb.con_est.SUB_CON_ESTABL_LEN	WORD	W#16#6	W#16#6	
150.0	S7_Verb.con_est.CON_ESTABL	BYTE	B#16#0	B#16#1	
152.0	S7_Verb.name_verb.SUB_CONNECT_NAME	WORD	W#16#12	W#16#12	
154.0	S7_Verb.name_verb.SUB_CONNECT_NAME_LEN	WORD	W#16#23	W#16#23	
156.0	S7_Verb.name_verb.CONNECT_NAME[0]	CHAR	' '	'V'	Connection S7 with IP-Config 1
157.0	S7_Verb.name_verb.CONNECT_NAME[1]	CHAR	' '	'e'	
158.0	S7_Verb.name_verb.CONNECT_NAME[2]	CHAR	' '	'r'	
159.0	S7_Verb.name_verb.CONNECT_NAME[3]	CHAR	' '	'b'	
160.0	S7_Verb.name_verb.CONNECT_NAME[4]	CHAR	' '	'I'	
161.0	S7_Verb.name_verb.CONNECT_NAME[5]	CHAR	' '	'n'	
162.0	S7_Verb.name_verb.CONNECT_NAME[6]	CHAR	' '	'd'	
163.0	S7_Verb.name_verb.CONNECT_NAME[7]	CHAR	' '	'u'	
164.0	S7_Verb.name_verb.CONNECT_NAME[8]	CHAR	' '	'n'	
165.0	S7_Verb.name_verb.CONNECT_NAME[9]	CHAR	' '	'g'	
166.0	S7_Verb.name_verb.CONNECT_NAME[10]	CHAR	' '	' '	
167.0	S7_Verb.name_verb.CONNECT_NAME[11]	CHAR	' '	'S'	
168.0	S7_Verb.name_verb.CONNECT_NAME[12]	CHAR	' '	'7'	

Address	Name	Type	Initial value	Actual	Comment
169.0	S7_Verb.name_verb.CONNECT_NAME[13]	CHAR	' '	' '	
170.0	S7_Verb.name_verb.CONNECT_NAME[14]	CHAR	' '	'm'	
171.0	S7_Verb.name_verb.CONNECT_NAME[15]	CHAR	' '	'I'	
172.0	S7_Verb.name_verb.CONNECT_NAME[16]	CHAR	' '	't'	
173.0	S7_Verb.name_verb.CONNECT_NAME[17]	CHAR	' '	' '	
174.0	S7_Verb.name_verb.CONNECT_NAME[18]	CHAR	' '	'I'	
175.0	S7_Verb.name_verb.CONNECT_NAME[19]	CHAR	' '	'P'	
176.0	S7_Verb.name_verb.CONNECT_NAME[20]	CHAR	' '	'-'	
177.0	S7_Verb.name_verb.CONNECT_NAME[21]	CHAR	' '	'C'	
178.0	S7_Verb.name_verb.CONNECT_NAME[22]	CHAR	' '	'o'	
179.0	S7_Verb.name_verb.CONNECT_NAME[23]	CHAR	' '	'n'	
180.0	S7_Verb.name_verb.CONNECT_NAME[24]	CHAR	' '	'f'	
181.0	S7_Verb.name_verb.CONNECT_NAME[25]	CHAR	' '	'I'	
182.0	S7_Verb.name_verb.CONNECT_NAME[26]	CHAR	' '	'g'	
183.0	S7_Verb.name_verb.CONNECT_NAME[27]	CHAR	' '	' '	
184.0	S7_Verb.name_verb.CONNECT_NAME[28]	CHAR	' '	'1'	
185.0	S7_Verb.name_verb.CONNECT_NAME[29]	CHAR	' '	' '	
186.0	S7_Verb.name_verb.CONNECT_NAME[30]	CHAR	' '	' '	

7 Modbus Communication

7.1 TCP

7.1.1 FB 70 - TCP_MB_CLIENT - Modbus/TCP client

Description

This function allows the operation of an Ethernet interface as Modbus/TCP client.

VIPA specific block
The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

Call parameter

Name	Declaration	Type	Description
REQ	IN	BOOL	Start job with edge 0-1.
ID	IN	WORD	ID from TCON.
MB_FUNCTION	IN	BYTE	Modbus: *Function code*.
MB_DATA_ADDR	IN	WORD	Modbus: Start address or *sub function code*.
MB_DATA_LEN	IN	INT	Modbus: Number of register/bits.
MB_DATA_PTR	IN	ANY	Modbus: Data buffer (only flag area or data block of data type byte allowed).
DONE *	OUT	BOOL	Job finished without error.
BUSY	OUT	BOOL	Job is running.
ERROR *	OUT	BOOL	Job is ready with error - parameter STATUS has error information.
STATUS *	OUT	WORD	Extended status and error information.
*) Parameter is available until the next call of the FB			

Parameter in instance DB

Name	Declaration	Type	Description
PROTOCOL_TIMEOUT	STAT	INT	Blocking time before an active job can be cancelled by the user. Default: 3s
RCV_TIMEOUT	STAT	INT	Monitoring time for a job. Default: 2s
MB_TRANS_ID	STAT	WORD	Modbus: Start value for the transaction identifier. Default: 1
MB_UNIT_ID	STAT	BYTE	Modbus: Device identification. Default: 255

The following must be observed:

- The *call parameters* must be specified with the block call. Besides the *call parameters* all parameters are located in the instance DB.
- The communication link must be previously initialized via FB 65 (TCON).
- FB 63 (TSEND) and FB 64 (TRCV) are required for the use of the block.
- During a job processing the instance DB is blocked for other clients.
- During job processing changes to the input parameters are not evaluated.
- With the following conditions a job processing is completed or cancelled:
  – DONE = 1 job without error
  – ERROR = 1 job with error
  – Expiration of RCV_TIMEOUT
  – REQ = FALSE after expiration of PROTOCOL_TIMEOUT
- REQ is reset before DONE or ERROR is set or PROTOCOL_TIMEOUT has expired, STATUS 8200h is reported. Here the current job is still processed.

Status and error indication

The function block reports via STATUS the following status and error information.

STATUS	DONE	BUSY	ERROR	Description
0000h	1	0	0	Operation executed without error.
7000h	0	0	0	No connection established or communication error (TCON).
7004h	0	0	0	Connection established and monitored. No job active.
7005h	0	1	0	Data are sent.
7006h	0	1	0	Data are received.
8210h	0	0	1	The hardware is incompatible with the block library Modbus RTU/TCP.
8380h	0	0	1	Received Modbus frame does not have the correct format or has an invalid length.
8381h	0	0	1	Server returns *Exception code 01h*.
8382h	0	0	1	Server returns *Exception code 03h* or wrong start address.
8383h	0	0	1	Server returns *Exception code 02h*.
8384h	0	0	1	Server returns *Exception code 04h*.
8386h	0	0	1	Server returns wrong *Function code*.
8387h	0	0	1	Connection ID (TCON) does not match the instance or server returns wrong protocol ID.

STATUS	DONE	BUSY	ERROR	Description
8388h	0	0	1	Server returns wrong value or wrong quantity.
80C8h	0	0	1	No answer of the server during the duration (RCV_TIMEOUT).
8188h	0	0	1	MB_FUNCTION not valid.
8189h	0	0	1	MB_DATA_ADDR not valid.
818Ah	0	0	1	MB_DATA_LEN not valid.
818Bh	0	0	1	MB_DATA_PTR not valid.
818Ch	0	0	1	BLOCKED_PROC_TIMEOUT or RCV_TIMEOUT not valid.
818Dh	0	0	1	Server returns wrong transaction ID.
8200h	0	0	1	Another Modbus request is processed at the time via the port (PROTOCOL_TIMEOUT).

7.1.1.1 Example

Task

With *Function code 03h*, starting from address 2000, 100 register are to be read from a Modbus/TCP server and stored in flag area starting from MB200. Errors are to be stored.

OB1

```
 CALL  FB    65 , DB65
       REQ    :=M100.0
       ID     :=W#16#1
       DONE   :=M100.1
       BUSY   :=
       ERROR  :=M100.2
       STATUS :=MW102
       CONNECT:=P#DB255.DBX 0.0 BYTE 64

      UN    M    100.2
      SPB   ERR1
      L     MW   102
      T     MW   104
ERR1: NOP   0
      U     M    100.1
      R     M    100.0

CALL  FB    70 , DB70
       REQ          :=M101.0
       ID           :=W#16#1
       MB_FUNCTION  :=B#16#3
       MB_DATA_ADDR:=W#16#7D0
       MB_DATA_LEN  :=100
       MB_DATA_PTR  :=P#M 200.0 BYTE 200
       DONE         :=M101.1
       BUSY         :=
       ERROR        :=M101.2
       STATUS       :=MW106

      UN    M    101.2
```

```
        SPB   ERR2
        L     MW   106
        T     MW   108
ERR2:   NOP   0
        U     M    101.1
        R     M    101.0
```

OB1 - Description

1. Calling of FB 65 (TCON) to establish a communication connection with the partner station.
2. Calling the handling block of the Modbus/TCP client with the correct parameters.
3. There is no connection to the partner station and MW102 returns 7000h.
4. Set M100.0 in the CPU to TRUE.
   ⇨ If M100.0 is automatically reset, the connection to the partner station is established and MW108 returns 7004h.
5. Set M101.0 in the CPU to TRUE.
   ⇨ The Modbus request is sent and it is waited for a response.

   If M101.0 is automatically reset, the job was finished without errors and the read data are stored in the CPU starting from bit memory byte 200. MW108 returns 7004h and indicates waiting for a new job.

   If M101.0 is not automatically reset and MW108 returns non-zero, an error has occurred. The cause of error can be read by the code of MW108 (e.g. MW108 = 8382h when the start address 2000 in the server is not available). MW108 returns 7004h and indicates waiting for a new job.

7.1.2 FB 71 - TCP_MB_SERVER - Modbus/TCP server

Description

This function allows the operation of an Ethernet interface as Modbus/TCP server.

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *⇨ Chapter 4 ‘Include VIPA library’ on page 103*

Call parameter

Name	Declaration	Type	Description
ENABLE	IN	BOOL	Activation/Deactivation Modbus server.
MB_DATA_PTR	IN	ANY	Modbus: Data buffer (only flag area or data block of type Byte allowed).
ID	IN	WORD	ID from TCON.

Name	Declaration	Type	Description
NDR *	OUT	BOOL	New data were written by the Modbus client.
DR *	OUT	BOOL	Data were read by the Modbus client.
ERROR *	OUT	BOOL	Job is ready with error - parameter STATUS has error information.
STATUS *	OUT	WORD	Extended status and error information.
*) Parameter is available until the next call of the FB			

Parameter in instance DB

Name	Declaration	Type	Description
REQUEST_COUNT	STAT	WORD	Counter for each received frame.
MESSAGE_COUNT	STAT	WORD	Counter for each valid Modbus request.
XMT_RCV_COUNT	STAT	WORD	Counter for each received frame, which contains no valid Modbus request.
EXCEPTION_COUNT	STAT	WORD	Counter for each negatively acknowledged Modbus request.
SUCCESS_COUNT	STAT	WORD	Counter for each positively acknowledged Modbus request.
FC1_ADDR_OUTPUT_START	STAT	WORD	Modbus *Function code 01h* start register for Q0.0 Default: 0
FC1_ADDR_OUTPUT_END	STAT	WORD	Modbus *Function code 01h* end register for Qx.y Default: 19999
FC1_ADDR_MEMORY_START	STAT	WORD	Modbus *Function code 01h* start register for M0.0 Default: 20000
FC1_ADDR_MEMORY_END	STAT	WORD	Modbus *Function code 01h* end register for Mx.y Default: 39999
FC2_ADDR_INPUT_START	STAT	WORD	Modbus *Function code 02h* start register for I0.0 Default: 0
FC2_ADDR_INPUT_END	STAT	WORD	Modbus *Function code 02h* end register for Ix.y Default: 19999

Name	Declara-tion	Type	Description
FC2_ADDR_MEMORY_START	STAT	WORD	Modbus *Function code 02h* start register for M0.0 Default: 20000
FC2_ADDR_MEMORY_END	STAT	WORD	Modbus *Function code 02h* end register for Mx.y Default: 39999
FC4_ADDR_INPUT_START	STAT	WORD	Modbus *Function code 04h* start register for IW0 Default: 0
FC4_ADDR_INPUT_END	STAT	WORD	Modbus *Function code 04h* end register for IWx Default: 19999
FC4_ADDR_MEMORY_START	STAT	WORD	Modbus *Function code 04h* start register for MW0 Default: 20000
FC4_ADDR_MEMORY_END	STAT	WORD	Modbus *Function code 04h* end register for MWx Default: 39999
FC5_ADDR_OUTPUT_START	STAT	WORD	Modbus *Function code 05h* start register for Q0.0 Default: 0
FC5_ADDR_OUTPUT_END	STAT	WORD	Modbus *Function code 05h* end register for Qx.y Default: 19999
FC5_ADDR_MEMORY_START	STAT	WORD	Modbus *Function code 05h* start register for M0.0 Default: 20000
FC5_ADDR_MEMORY_END	STAT	WORD	Modbus *Function code 05h* end register for Mx.y Default: 39999
FC15_ADDR_OUTPUT_START	STAT	WORD	Modbus *Function code 0Fh* start register for Q0.0 Default: 0
FC15_ADDR_OUTPUT_END	STAT	WORD	Modbus *Function code 0Fh* end register for Qx.y Default: 19999

Name	Declaration	Type	Description
FC15_ADDR_MEMORY_START	STAT	WORD	Modbus *Function code 0Fh* start register for Q0.0 Default: 20000
FC15_ADDR_MEMORY_END	STAT	WORD	Modbus *Function code 0Fh* end register for Qx.y Default: 39999

The following must be observed:

- The *call parameters* must be specified with the block call. Besides the *call parameters* all parameters are located in the instance DB.
- The communication link must be previously initialized via FB 65 (TCON).
- FB 63 (TSEND) and FB 64 (TRCV) are required for the use of the block.
- The INPUT/OUTPUT Modbus addresses of a *Function code* must be located in front of the MEMORY Modbus address and thus always be lower.
- Within a *Function code* no Modbus address may be defined multiple times - also not 0!
- The server can only process one job simultaneously. New Modbus requests during job processing are ignored and not answered.

Status and error indication

The function block reports via *STATUS* the following status and error information.

STATUS	NDR	DR	ERROR	Description
0000h	0 or 1 *		0	Operation executed without error.
7000h	0	0	0	No connection established or communication error (TCON).
7005h	0	0	0	Data are sent.
7006h	0	0	0	Data are received.
8210h	0	0	1	The hardware is incompatible with the block library Modbus RTU/TCP.
8380h	0	0	1	Received Modbus frame does not have the correct format or bytes are missing.
8381h	0	0	1	*Exception code 01h*, *Function code* is not supported.
8382h	0	0	1	*Exception code 03h*, data length or data value are not valid.
8383h	0	0	1	*Exception code 02h*, invalid start address or address range.
8384h	0	0	1	*Exception code 04h*, area length error when accessing inputs, outputs or bit memories.
8387h	0	0	1	Connection ID (TCON) does not match the instance or client returns wrong protocol ID.

STATUS	NDR	DR	ERROR	Description
8187h	0	0	1	MB_DATA_PTR not valid.

*) Error free Modbus job with *Function code 05h, 06h, 0Fh* or *10h* returns NDR=1 and DR=0.

Error free Modbus job with *Function code 01h, 02h, 03h, 04h* return DR=1 and NDR=0.

7.1.2.1 Example

Task

The CPU provides 100 byte data in the flag area starting from MB200 for a Modbus client via the Modbus register 0...49. Data can be read from the Modbus client via *Function code 03h* and written with *Function code 06h, 10h*. The CPU output Q1.0 is to be controlled by a Modbus client via *Function code 05h* and the start address 5008. Errors are to be stored.

OB1

```
CALL  FB    65 , DB65
       REQ     :=M100.0
       ID      :=W#16#1
       DONE    :=M100.1
       BUSY    :=
       ERROR   :=M100.2
       STATUS  :=MW102
       CONNECT:=P#DB255.DBX 0.0 BYTE 64

      UN    M    100.2
      SPB   ERR1

      L     MW   102
      T     MW   104

ERR1: NOP   0
      U     M    100.1
      R     M    100.0

      L     5000
      T     DB71.DBW   52

      CALL  FB     71 , DB71
       ENABLE      :=M101.0
       MB_DATA_PTR:=P#M 200.0 BYTE 100
       ID          :=W#16#1
       NDR         :=M101.1
       DR          :=M101.2
       ERROR       :=M101.3
       STATUS      :=MW106

      UN    M    101.3
      SPB   ERR2
      L     MW   106
      T     MW   108
ERR2: NOP   0
```

OB1 - Description

1. Call of FB 65 (TCON) to establish a communication connection with the partner station.
2. Calling the handling block of the Modbus/TCP server with the correct parameters.
3. There is no connection to the partner station and MW102 returns 7000h.
4. Set M100.0 in the CPU to TRUE.
   ⇨ If M100.0 is automatically reset, the connection to the partner station is established and MW108 returns 7006h.

5. The Modbus start register in the process image, which can be reached by *Function code 05h*, may be changed in the example by the parameter FC5_ADDR_OUTPUT_START (word 52 in the instance data block).
6. Set M101.0 in the CPU to TRUE.
   ⇨ The Modbus server now works.
7. The client sends a Modbus request with *Function code 03h* start address 10 and quantity 30.
   ⇨ The server responds with 60 byte starting from MB220. DR is set for one CPU cycle and thus M101.2 is set to "1".
8. The client sends a Modbus request with *Function code 05h* start address 5008 and the value FF00h.
   ⇨ The server acknowledges the request and writes "1" to the output Q1.0. NDR is set for one CPU cycle and thus M101.1 is set to "1".
9. The client sends a Modbus request with *Function code 03h* start address 50 (does not exist) and quantity 1.
   ⇨ The server responds with *Exception code 02h* an sets ERROR/STATUS for one CPU cycle. MW108 returns 8383h.

7.2 RTU

7.2.1 FB 72 - RTU_MB_MASTER - Modbus RTU master

Description

This function block allows the operation of the internal serial RS485 interface of a CPU from VIPA or a System SLIO CP 040 as Modbus RTU master.

> ***VIPA specific block***
> *The VIPA specific blocks can be found in the VIPA library.* *Chapter 4 'Include VIPA library' on page 103*

Call parameter

Name	Declara-tion	Type	Description
REQ	IN	BOOL	Start job with edge 0-1.
HARDWARE	IN	BYTE	1 = System SLIO CP 040 / 2 = VIPA SPEED7 CPU
LADDR	IN	INT	Logical address of the System SLIO CP 040 (parameter is ignored with the VIPA SPEED7 CPU).
MB_UNIT_ID	IN	BYTE	Modbus: Device identification = Address of the slave (0 ... 247).
MB_FUNCTION	IN	BYTE	Modbus: *Function code*.
MB_DATA_ADDR	IN	WORD	Modbus: Start address or *Sub function code*.

Name	Declaration	Type	Description
MB_DATA_LEN	IN	INT	Modbus: Number of register/bits.
MB_DATA_PTR	IN	ANY	Modbus: Data buffer (only flag area or data block of type Byte allowed).
DONE *	OUT	BOOL	Job finished without error.
BUSY	OUT	BOOL	Job is running.
ERROR *	OUT	BOOL	Job is ready with error - parameter *STATUS* has error information.
STATUS *	OUT	WORD	Extended status and error information.
*) Parameter is available until the next call of the FB			

Parameter in instance DB

Name	Declaration	Type	Description
INIT	STAT	BOOL	With an edge 0-1 an synchronous reset is established at the System SLIO CP 040. After a successful reset the bit automatically reset.

The following must be observed:

- The *call parameters* must be specified with the block call. Besides the *call parameters* all parameters are located in the instance DB.
- The interface to be used must be configured before:
  – VIPA System SLIO CP 040: Configuration as "Modbus master RTU" with 60 byte IO-Size in the hardware configuration.
  – Internal serial RS485 interface of a VIPA CPU:
    Configuration via SFC 216 (SER_CFG) with protocol "Modbus master RTU".
- FB 60 SEND and FB 61 RECEIVE (or FB 65 SEND_RECV) are required for the use of the block, even if the internal serial RS485 interface of a CPU from VIPA is used.
- During job processing changes to the input parameters are not evaluated.
- Broadcast request via MB_UNIT_ID = 0 are only accepted for writing functions.
- With the following conditions a job processing is completed or cancelled:
  – *DONE* = 1 job without error
  – *ERROR* = 1 job with error
  – Expiration of time-out (parameterization at the interface)
- If *REQ* is reset before *DONE* or *ERROR* is set, STATUS 8200h is reported. Here the current job is still processed.

Status and error indication

The function block reports via STATUS the following status and error information.

STATUS	DONE	BUSY	ERROR	Description
0000h	1	0	0	Operation executed without error.
7000h	0	0	0	No connection established or communication error.
7004h	0	0	0	Connection established and monitored. No job active.
7005h	0	1	0	Data are sent.
7006h	0	1	0	Data are received.
8210h	0	0	1	The hardware is incompatible with the block library Modbus RTU/TCP.
8381h	0	0	1	Server returns *Exception code 01h*.
8382h	0	0	1	Server returns *Exception code 03h* or wrong start address.
8383h	0	0	1	Server returns *Exception code 02h*.
8384h	0	0	1	Server returns *Exception code 04h*.
8386h	0	0	1	Server returns wrong *Function code*.
8388h	0	0	1	Server returns wrong value or quantity.
80C8h	0	0	1	No answer of the server during the defined duration (time-out parameterizable via interface).
8188h	0	0	1	MB_FUNCTION not valid.
8189h	0	0	1	MB_DATA_ADDR not valid.
818Ah	0	0	1	MB_DATA_LEN not valid.
818Bh	0	0	1	MB_DATA_PTR not valid.
8201h	0	0	1	HARDWARE not valid.
8202h	0	0	1	MB_UNIT_ID not valid.
8200h	0	0	1	Another Modbus request is processed at the time via the port.

7.2.1.1 Example

Task

With *Function code 03h*, starting from address 2000, 100 register are to be read from a Modbus RTU slave with address 99 and stored in flag area starting from MB200. Errors are to be stored. The Modbus RTU master is realized via the internal serial RS485 interface of a VIPA CPU.

OB100

```
CALL  SFC  216
      Protocol :=B#16#5
      Parameter :=DB10
      Baudrate:=B#16#9
      CharLen:=B#16#3
      Parity:=B#16#2
      StopBits:=B#16#1
      FlowControl:=B#16#1
      RetVal:=MW100
```

OB100 - Description

1. Calling of the SFC 216 (SER_CFG) to configure the internal serial interface of the CPU from VIPA.
2. Protocol: "Modbus Master RTU", 9600 baud, 8 data bit, 1 stop bit, even parity, no flow control.
3. DB10 has a variable of type WORD with a Modbus time-out (value in ms).

OB1

```
CALL  FB    72 , DB72
       REQ          :=M101.0
       HARDWARE     :=B#16#2
       LADDR        :=
       MB_UNIT_ID   :=B#16#63
       MB_FUNCTION  :=B#16#3
       MB_DATA_ADDR:=W#16#7D0
       MB_DATA_LEN  :=100
       MB_DATA_PTR  :=P#M 200.0 BYTE 200
       DONE         :=M101.1
       BUSY         :=
       ERROR        :=M101.2
       STATUS       :=MW102

      UN    M    101.2
      SPB   ERR1
      L     MW   102
      T     MW   104
ERR1: NOP   0
      U     M    101.1
      R     M    101.0
```

OB1 - Description

1. Calling the handling block of the Modbus RTU master with the correct parameters.
2. If the interface was correctly initialized in the OB 100, the master can be used and MW102 returns 7004h.
3. Set M101.0 in the CPU to TRUE.
   ⇨ The Modbus request is sent and it is waited for a response.

   If M101.0 is automatically reset, the job was finished without errors and the read data are stored in the CPU starting from bit memory byte 200. MW104 returns 7004h and indicates waiting for a new job.

   If M101.0 is not automatically reset and MW104 returns non-zero, an error has occurred. The cause of error can be read by the code of MW104 (e.g. MW104 = 8382h when the start address 2000 in the server is not available). MW102 returns 7004h and indicates waiting for a new job.

7.2.2 FB 73 - RTU_MB_SLAVE - Modbus RTU slave

Description

This function block allows the operation of the internal serial RS485 interface of a CPU from VIPA or a System SLIO CP 040 as Modbus RTU slave.

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

Call parameter

Name	Declaration	Type	Description
ENABLE	IN	BOOL	Activation/Deactivation Modbus server.
HARDWARE	IN	BYTE	1 = System SLIO CP 040 / 2 = VIPA SPEED7 CPU
LADDR	IN	INT	Logical address of the System SLIO CP 040 (parameter is ignored with the VIPA SPEED7 CPU).
MB_UNIT_ID	IN	BYTE	Modbus: Device identification = own address (1 ... 247).
MB_DATA_PTR	IN	ANY	Modbus: Data buffer (only flag area or data block of type Byte allowed).
NDR *	OUT	BOOL	New data were written by the Modbus client.
DR *	OUT	BOOL	Data were read by the Modbus client.
ERROR *	OUT	BOOL	Job is ready with error - parameter *STATUS* has error information.
STATUS *	OUT	WORD	Extended status and error information.
*) Parameter is available until the next call of the FB			

Parameter in instance DB

Name	Declaration	Type	Description
INIT	STAT	BOOL	With an edge 0-1 an synchronous reset is established at the System SLIO CP 040.
REQUEST_COUNT	STAT	WORD	Counter for each received frame.
MESSAGE_COUNT	STAT	WORD	Counter for each valid Modbus request.
BROADCAST_COUNT	STAT	WORD	Counter for each valid Modbus broadcast request.
EXCEPTION_COUNT	STAT	WORD	Counter for each negatively acknowledged Modbus request.
SUCCESS_COUNT	STAT	WORD	Counter for each positively acknowledged Modbus request.
BAD_CRC_COUNT	STAT	WORD	Counter for each valid Modbus request with CRC error.

Name	Decla-ration	Type	Description
FC1_ADDR_OUTPUT_START	STAT	WORD	Modbus *Function code 01h* start register for Q0.0 Default: 0
FC1_ADDR_OUTPUT_END	STAT	WORD	Modbus *Function code 01h* end register for Qx.y Default: 19999
FC1_ADDR_MEMORY_START	STAT	WORD	Modbus *Function code 01h* start register for M0.0 Default: 20000
FC1_ADDR_MEMORY_END	STAT	WORD	Modbus *Function code 01h* end register for Mx.y Default: 39999
FC2_ADDR_INPUT_START	STAT	WORD	Modbus *Function code 02h* start register for I0.0 Default: 0
FC2_ADDR_INPUT_END	STAT	WORD	Modbus *Function code 02h* end register for Ix.y Default: 19999
FC2_ADDR_MEMORY_START	STAT	WORD	Modbus *Function code 02h* start register for M0.0 Default: 20000
FC2_ADDR_MEMORY_END	STAT	WORD	Modbus *Function code 02h* end register for Mx.y Default: 39999
FC4_ADDR_INPUT_START	STAT	WORD	Modbus *Function code 04h* start register for IW0 Default: 0
FC4_ADDR_INPUT_END	STAT	WORD	Modbus *Function code 04h* end register for IWx Default: 19999
FC4_ADDR_MEMORY_START	STAT	WORD	Modbus *Function code 04h* start register for MW0 Default: 20000
FC4_ADDR_MEMORY_END	STAT	WORD	Modbus *function-Code 04 h* end register for MW0 Default: 39999
FC5_ADDR_OUTPUT_START	STAT	WORD	Modbus *Function code 05h* start register for Q0.0 Default: 0

Name	Declaration	Type	Description
FC5_ADDR_OUTPUT_END	STAT	WORD	Modbus *Function code 05h* end register for Qx.y Default: 19999
FC5_ADDR_MEMORY_START	STAT	WORD	Modbus *Function code 05h* start register for M0.0 Default: 20000
FC5_ADDR_MEMORY_END	STAT	WORD	Modbus *Function code 05h* end register for Mx.y Default: 39999
FC15_ADDR_OUTPUT_START	STAT	WORD	Modbus *Function code 0Fh* start register for Q0.0 Default: 0
FC15_ADDR_OUTPUT_END	STAT	WORD	Modbus *Function code 0Fh* end register for Qx.y Default: 19999
FC15_ADDR_MEMORY_START	STAT	WORD	Modbus *Function code 0Fh* start register for M0.0 Default: 20000
FC15_ADDR_MEMORY_END	STAT	WORD	Modbus *Function code 0Fh* end register for Mx.y Default: 39999

The following must be observed:

- The *call parameters* must be specified with the block call. Besides the *call parameters* all parameters are located in the instance DB.
- The interface to be used must be configured before:
  – VIPA System SLIO CP 040: Configuration as ASCII module with 60 byte IO-Size in the hardware configuration.
  – Internal serial RS485 interface of a VIPA CPU:
    Configuration via SFC 216 (SER_CFG) with protocol "ASCII".
- FB 60 SEND and FB 61 RECEIVE (or FB 65 SEND_RECV) are required for the use of the block, even if the internal serial RS485 interface of a CPU from VIPA is used.
- Broadcast request via MB_UNIT_ID = 0 are only accepted for writing functions.
- The INPUT/OUTPUT Modbus addresses of a *Function code* must be located in front of the MEMORY Modbus address and thus always be lower.
- Within a *Function code* no Modbus address may be defined multiple times - also not 0!
- The slave can only process one job simultaneously. New Modbus requests during job processing are ignored and not answered.

Status and error indication

The function block reports via STATUS the following status and error information.

STATUS	NDR	DR	ERROR	Description
0000h	0 or 1 *		0	Operation executed without error.
7000h	0	0	0	No connection established or communication error.
7005h	0	0	0	Data are sent.
7006h	0	0	0	Data are received.
8210h	0	0	1	The hardware is incompatible with the block library Modbus RTU/TCP.
8380h	0	0	1	CRC error
8381h	0	0	1	*Exception code 01h*, *Function code* is not supported.
8382h	0	0	1	*Exception code 03h*, data length or data value are not valid.
8383h	0	0	1	*Exception code 02h*, invalid start address or address range.
8384h	0	0	1	*Exception code 04h*, area length error when accessing inputs, outputs or bit memories.
8187h	0	0	1	MB_DATA_PTR not valid.
8201h	0	0	1	HARDWARE not valid.
8202h	0	0	1	MB_UNIT_ID not valid.
8203 h	0	0	1	

*) Error free Modbus job with *Function code 05h, 06h, 0Fh* or *10h* returns NDR=1 and DR=0.

Error free Modbus job with *Function code 01h, 02h, 03h, 04h* return DR=1 and NDR=0.

7.2.2.1 Example

Task

The CPU provides 100 byte data in the flag area starting from MB200 for a Modbus master via the Modbus register 0 ... 49. Data can be read by the Modbus master via *Function code 03h* and written with *Function code 06h, 10h*. The CPU output Q1.0 is to be controlled by a Modbus master via *Function code 05h* and the start address 5008. Errors are to be stored. The Modbus RTU slave with the address 99 is realized via the internal serial RS485 interface of a VIPA CPU.

OB100

```
 CALL  SFC  216
       Protocol :=B#16#1
       Parameter :=DB10
       Baudrate:=B#16#9
       CharLen:=B#16#3
       Parity:=B#16#2
       StopBits:=B#16#1
       FlowControl:=B#16#1
       RetVal:=MW100
```

OB100 - Description

1. Calling of the SFC 216 (SER_CFG) to configure the internal serial interface of the CPU from VIPA.
2. Protocol: "ASCII", 9600 baud, 8 data bit, 1 stop bit, even parity, no flow control.
3. DB10 has a variable of type WORD and must be passed as "Dummy".

OB1

```
      L     5000
      T     DB73.DBW   58

      CALL  FB     73 , DB73
       ENABLE      :=M101.0
       HARDWARE    :=B#16#2
       LADDR       :=
       MB_UNIT_ID  :=B#16#63
       MB_DATA_PTR:=P#M 200.0 BYTE 100
       NDR         :=M101.1
       DR          :=M101.2
       ERROR       :=M101.3
       STATUS      :=MW102

      UN    M    101.3
      SPB   ERR1
      L     MW   102
      T     MW   104
ERR1: NOP   0
```

OB1 - Description

1. Calling the handling block of the Modbus/TCP server with the correct parameters.
2. If the interface was correctly initialized in the OB100, the slave can be used and MW102 returns 7006h.
3. The Modbus start register in the process image, which can be reached by *Function code 05h*, may be changed in the example by the parameter FC5_ADDR_OUTPUT_START (word 58 in the instance data block).
4. Set M101.0 in the CPU to TRUE.

   ⇨ The Modbus slave now works.

5. The master sends a Modbus request with *Function code 03h* start address 10 and quantity 30.

   ⇨ The slave responds with 60byte starting from MB200. DR is set for one CPU cycle and thus M101.2 is set to "1".

6. The master sends a Modbus request with *Function code 05h* start address 5008 and the value FF00h.

   ⇨ The salve acknowledges the request and writes "1" to the output Q1.0. NDR is set for one CPU cycle and thus M101.1 is set to "1".

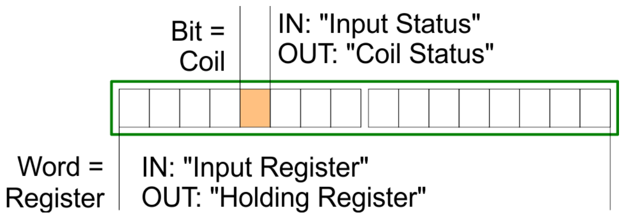
7. The master sends a Modbus request with *Function code 03h* start address 50 (does not exist!) and quantity 1.

   ⇨ The server responds with *Exception code 02h* and sets ERROR/STATUS for one CPU cycle. MW104 returns 8383h.

7.3 FKT Codes

Naming convention

Modbus has some naming conventions:



- Modbus differentiates between bit and word access; Bits = "Coils" and Words = "Register".
- Bit inputs are referred to as "Input-Status" and bit outputs as "Coil-Status".
- Word inputs are referred to as "Input-Register" and word outputs as "Holding-Register".

Range definitions

Normally the access with Modbus happens by means of the ranges 0x, 1x, 3x and 4x.

0x and 1x gives you access to *digital* bit areas and 3x and 4x to *analog* word areas.

For the Ethernet coupler from VIPA is not differentiating digital and analog data, the following assignment is valid:

0x - Bit area for master output

Access via function code 01h, 05h, 0Fh

1x - Bit area for master input

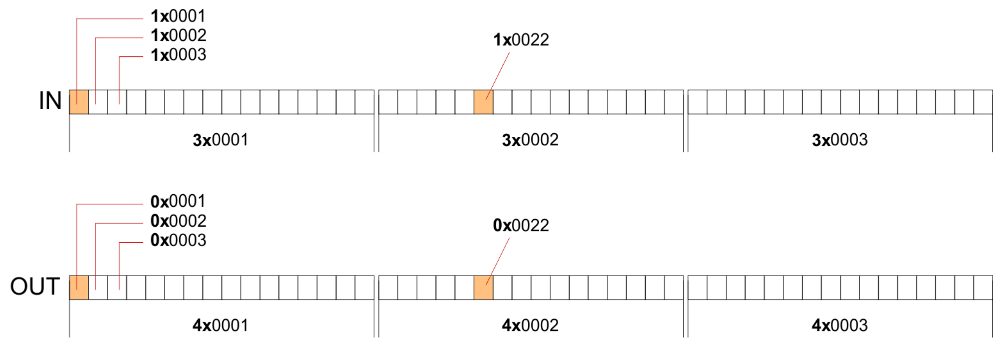
Access via function code 02h

3x - Word area for master input

Access via function code 04h

4x - Word area for master output

Access via function code 03h, 06h, 10h, 16h



Overview

With the following Modbus function codes a Modbus master can access a Modbus slave. The description always takes place from the point of view of the master:

Code	Command	Description
01h	Read n Bits	Read n bits of master output area 0x
02h	Read n Bits	Read n bits of master input area 1x
03h	Read n Words	Read n words of master output area 4x
04h	Read n Words	Read n words master input area 3x
05h	Write 1 Bit	Write 1 bit to master output area 0x
06h	Write 1 Word	Write 1 word to master output area 4x
0Fh	Write n Bits	Write n bits to master area 0x
10h	Write n Words	Write n words to master area 4x
16h	Mask 1 Word	Mask 1 word in master output area 4x
17h	Write n Words and Read m Words	Write n words into master output area 4x and the respond contains m read words of the master input area 3x

Byte sequence in a word

1 word	
High byte	Low byte

Respond of the coupler

If the slave announces an error, the function code is sent back with a "OR" and 80h. Without an error, the function code is sent back.

Coupler answer:	Function code OR 80h	→ Error & error number
	Function code	→ OK

If the slave announces an error, the function code is sent back with a "OR" and 80h. Without an error, the function code is sent back.

01h: Function number is not supported

02h: Addressing errors

03h: Data errors

04h: System SLIO bus is not initialized

07h: General error

Read n Bits 01h, 02h

Code 01h: Read n bits of master output area 0x.

Code 02h: Read n bits of master input area 1x.

Command telegram

Modbus/TCP-Header						Slave address	Function code	Address1. bit	Number of bits
x	x	0	0	0	6				
6byte						1byte	1byte	1word	1word

Respond telegram

Modbus/TCP-Header						Slave address	Function code	Number of read bytes	Data 1. byte	Data 2. byte	...
x	x	0	0	0							
6byte						1byte	1byte	1byte	1byte	1byte	
									max. 252byte		

Read n words 03h, 04h

03h: Read n words of master output area 4x.

04h: Read n words master input area 3x.

Command telegram

Modbus/TCP-Header						Slave address	Function code	Address word	Number of words
x	x	0	0	0	6				
6byte						1byte	1byte	1word	1word

Respond telegram

Modbus/TCP-Header						Slave address	Function code	Number of read bytes	Data 1. word	Data 2. word	...
x	x	0	0	0							
6byte						1byte	1byte	1byte	1word	1word	
									max. 126words		

Write 1 bit 05h

Code 05h: Write 1 bit to master output area 0x.

A status change is via "Status bit" with following values:

"Status bit" = 0000h → Bit = 0

"Status bit" = FF00h → Bit = 1

Command telegram

Modbus/TCP-Header						Slave address	Function code	Address bit	Status bit
x	x	0	0	0	6				
6byte						1byte	1byte	1word	1word

Respond telegram

Modbus/TCP-Header						Slave address	Function code	Address bit	Status bit
x	x	0	0	0	6				
6byte						1byte	1byte	1word	1word

Write 1 word 06h

Code 06h: Write 1 word to master output area 4x.

Command telegram

Modbus/TCP-Header						Slave address	Function code	Address word	Value word
x	x	0	0	0	6				
6byte						1byte	1byte	1word	1word

Respond telegram

Modbus/TCP-Header						Slave address	Function code	Address word	Value word
x	x	0	0	0	6				
6byte						1byte	1byte	1word	1word

Write n bits 0Fh

Code 0Fh: Write n bits to master output area 0x.

Please regard that the number of bits are additionally to be set in byte.

Command telegram

Modbus/TCP-Header						Slave address	Function code	Address 1. bit	Number of bits	Number of bytes	Data 1. byte	Data 2. byte	...
x	x	0	0	0									
6byte						1byte	1byte	1word	1word	1byte	1byte	1byte	1byte
											max. 248byte		

Respond telegram

Modbus/TCP-Header						Slave address	Function code	Address 1. bit	Number of bits
x	x	0	0	0	6				
6byte						1byte	1byte	1word	1word

Write n words 10h

Code 10h: Write n words to master output area 4x.

Command telegram

Modbus/TCP-Header						Slave address	Function code	Address 1. word	Number of words	Number of bytes	Data 1. word	Data 2. word	...
x	x	0	0	0									
6byte						1byte	1byte	1word	1word	1word	1word	1word	1word
											max. 124byte		

Respond telegram

Modbus/TCP-Header						Slave address	Function code	Address 1. word	Number of words
x	x	0	0	0	6				
6byte						1byte	1byte	1word	1word

Mask a word 16h

Code 16h: This function allows to mask a word in the master output area 4x.

Command telegram

Modbus/TCP-Header						Slave address	Function code	Address word	AND Mask	OR Mask
x	x	0	0	0	8					
6byte						1byte	1byte	1word	1word	1word

Respond telegram

Modbus/TCP-Header						Slave address	Function code	Address word	AND Mask	OR Mask
x	x	0	0	0	8					
6byte						1byte	1byte	1word	1word	1word

8 Serial Communication

8.1 Serial communication

8.1.1 SFC 207 - SER_CTRL - Modem functionality PtP

Description

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

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

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

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)

8.1.2 Overview

You may de-activate the DP master integrated in the SPEED7-CPU via a hardware configuration using Object properties and the parameter "Function RS485". Thus release the RS485 interface for PtP (point-to-point) communication. The RS485 interface supports in PtP operation the serial process connection to different source res. destination systems.

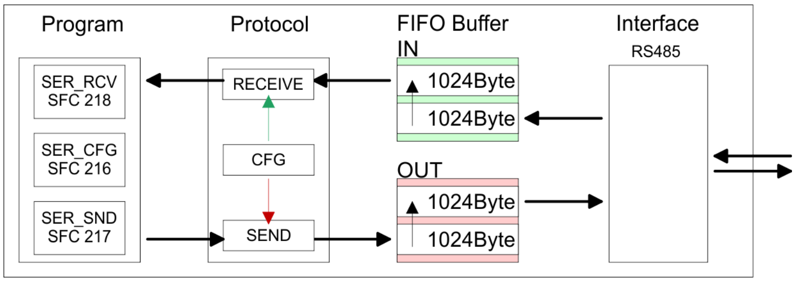
Parametrization

The parametrization happens during runtime using the FC/SFC 216 (SER_CFG). For this you have to store the parameters in a DB for all protocols except ASCII.

Communication

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

RS485 PtP communication



Overview FC/SFCs for serial communication

The following FC/SFCs are used for the serial communication:

FC/SFC		Description
FC/SFC 216	SER_CFG	RS485 parametrize
FC/SFC 217	SER_SND	RS485 send
FC/SFC 218	SER_RCV	RS485 receive

8.1.3 FC/SFC 216 - SER_CFG - Parametrization PtP

Description

The parametrization happens during runtime deploying the FC/SFC 216 (SER_CFG). You have to store the parameters for STX/ETX, 3964R, USS and Modbus in a DB.

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

Parameters

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

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

Example:

- Wanted time 8ms at a baudrate of 19200baud
- Calculation: 19200bit/s x 0.008s ≈ 154bit → (9Ah)
- Hex value is 9Ah.

PROTOCOL

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

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

PARAMETER (as DB)

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

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

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

With not used IDs please always enter FFh!

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

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

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

BAUDRATE

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

CHARLEN

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

PARITY

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

0: NONE	1: ODD	2: EVEN

STOPBITS

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

1: 1bit	2: 1.5bit*	3: 2bit
*) Only permitted when *CHARLEN* = 0 (5bit)		

FLOWCONTROL

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

RETVAL FC/SFC 216 (Return values)

Return values send by the block:

Error code	Description
0000h	no error
809Ah	Interface not found e. g. interface is used by PROFIBUS In the VIPA SLIO CPU with FeatureSet PTP_NO only the ASCII protocol is configurable. If another protocol is selected the FC/SFC216 also left with this error code.
8x24h	Error at FC/SFC-Parameter x, with x: 1: Error at *PROTOCOL* 2: Error at *PARAMETER* 3: Error at *BAUDRATE* 4: Error at *CHARLENGTH* 5: Error at *PARITY* 6: Error at *STOPBITS* 7: Error at *FLOWCONTROL*
809xh	Error in FC/SFC parameter value x, where x: 1: Error at *PROTOCOL* 3: Error at *BAUDRATE* 4: Error at *CHARLENGTH* 5: Error at *PARITY* 6: Error at *STOPBITS* 7: Error at *FLOWCONTROL* (parameter is missing)
8092h	Access error in parameter DB (DB too short)
828xh	Error in parameter x of DB parameter, where x: 1: Error 1. parameter 2: Error 2. parameter ...

8.1.4 FC/SFC 217 - SER_SND - Send to PtP

Description

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

> ***VIPA specific block***
> *The VIPA specific blocks can be found in the VIPA library.* *Chapter 4 'Include VIPA library' on page 103*

Parameters

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

DATAPTR

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

Example:

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

DATALEN

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

RETVAL FC/SFC 217 (Return values)

Return values of the block:

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

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

Protocol specific RETVAL values

ASCII

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

STX/ETX

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

3964R

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

USS

Error code	Description
2000h	Send ready without error
8080h	Receive buffer overflow (no space for receipt)
8090h	Acknowledgement delay time exceeded
80F0h	Wrong checksum in respond
80FEh	Wrong start sign in respond
80FFh	Wrong slave address in respond

Error code	Description
9000h	Buffer overflow (no data send)
9001h	Data too long (>1024byte)
9002h	Data too short (<2byte)

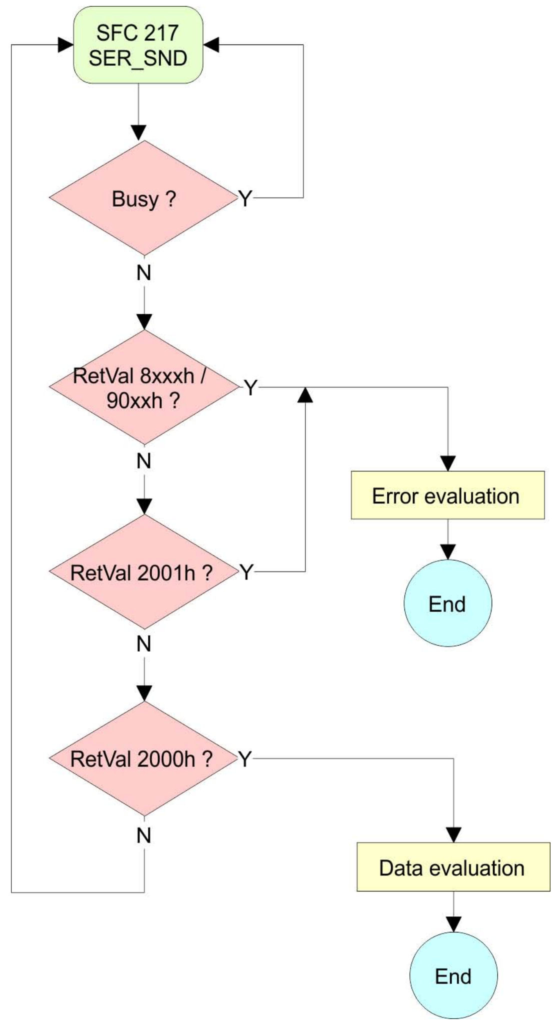
Modbus RTU/ASCII Master

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

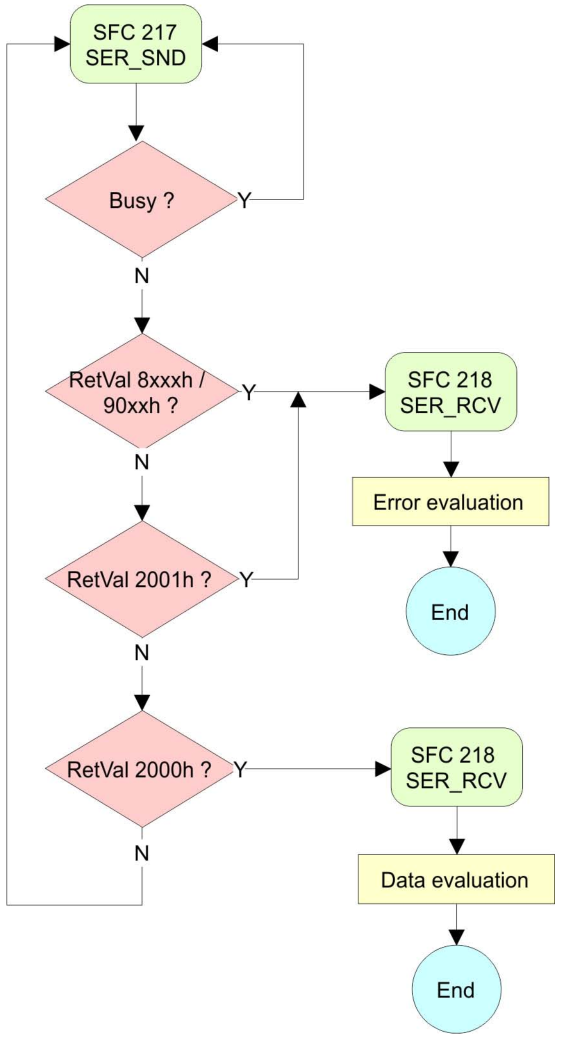
Principles of programming

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

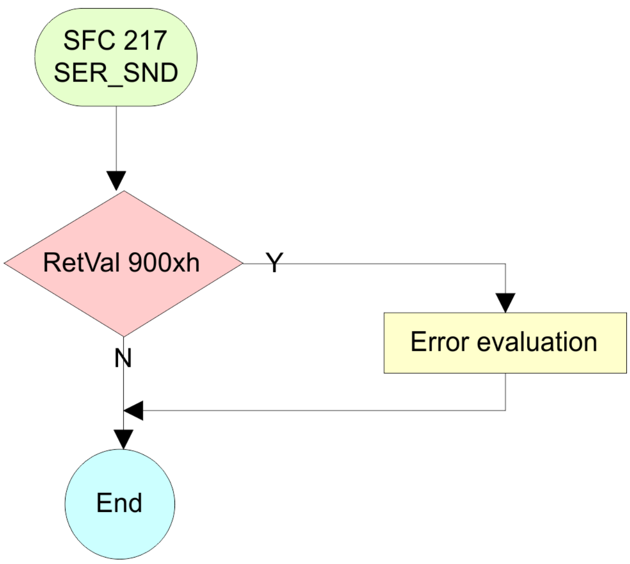
3964R



USS / Modbus



ASCII / STX/ETX



8.1.5 FC/SFC 218 - SER_RCV - Receive from PtP

Description

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

> ***VIPA specific block***
> *The VIPA specific blocks can be found in the VIPA library.* *Chapter 4 'Include VIPA library' on page 103*

Parameters

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

DATAPTR

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

Example:

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

DATALEN

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

ERROR

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

ASCII

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

STX/ETX

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

3964R / Modbus RTU/ASCII Master

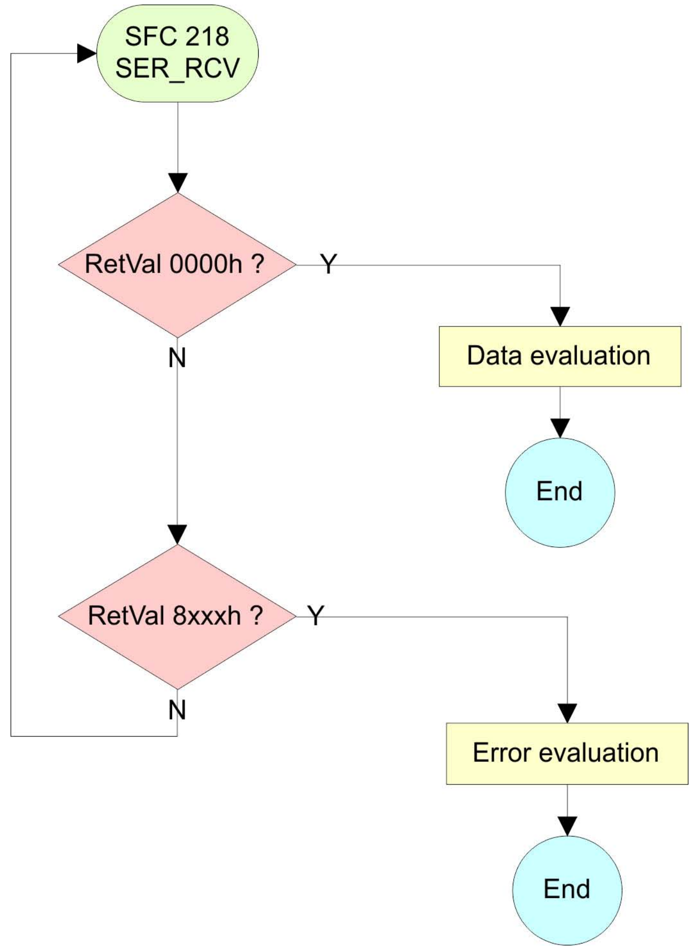
Bit	Error	Description
0	overflow	The received telegram exceeds the size of the receive buffer.

RETVAL FC/SFC 218 (Return value)

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

Principles of programming

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



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

Description

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

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *↳ Chapter 4 'Include VIPA library' on page 103*

Parameter

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

EMPF_PUFFER

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

Example:

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

ER_BYTE

This word gets an entry in case of error.

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

8.1.7 FB 7 - P_RCV_RK - Receive from CP 341

Description

The FB 7 P_RCV_RK transfers data from the CP to a data area of the CPU specified by the parameter *DB_NO, DBB_NO* and *LEN*. For data transfer the FB is to be called either cyclically or statically by a timer-driven program.

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

Parameter

Parameter	Declaration	Data type	Description
EN_R	IN	BOOL	Enables data read
R	IN	BOOL	Aborts request - current request is aborted and receiving is blocked.
LADDR	IN	INT	Logical basic address of the CP - corresponds to the address of the hardware configuration of the CP.
DB_NO	IN	INT	Data block number - number of the receive DB, zero is not allowed.
DBB_NO	IN	INT	Data byte number - received data as of data byte 0 ≤ *DBB_NO* ≤ 8190
L_...	OUT	-	These parameters are not relevant for ASCII and 3964(R). But they may be used by loadable protocols.
NDR*	OUT	BOOL	Request complete without errors, data received Parameter *STATUS* = 00h
ERROR*	OUT	BOOL	Request complete with error Parameter *STATUS* contains error details
LEN*	OUT	BOOL	Length of the received telegram in byte 1 ≤ *LEN* ≤ 1024
STATUS*	OUT	WORD	Specification of the error on *ERROR* = 1
*) Parameter is available until the next call of the FB.			

Release and cancel a request

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

Mechanism for startup synchronization

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

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

Error indication

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

Addressing

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

Data area

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

8.1.8 FB 8 - P_SND_RK - Send to CP 341

Description

The FB 8 - P_SND_RK transfers a data block of a DB to the CP, specified by the parameters *DB_NO*, *DBB_NO* and *LEN*. For data transfer the FB is to be called either cyclically or statically by a timer-driven program.

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *↳ Chapter 4 'Include VIPA library' on page 103*

Parameter

Parameter	Declaration	Data type	Description
SF	IN	CHAR	S = Send, F = Fetch. At ASCII and 3964R the default value "S" for Send may be used
REQ	IN	BOOL	Initiates request with positive edge
R	IN	BOOL	Aborts request - current request is aborted and sending is blocked.
LADDR	IN	INT	Logical basic address of the CP - corresponds to the address of the hardware configuration of the CP.
DB_NO	IN	INT	Data block number - number of the send DB, zero is not allowed.
DBB_NO	IN	INT	Data byte number - transmitted data as of data byte 0 ≤ *DBB_NO* ≤ 8190

Parameter	Declaration	Data type	Description
LEN	IN	INT	Length of message frame to be sent in byte 1 ≤ *LEN* ≤ 1024
R_...	IN	-	These parameters are not relevant for ASCII and 3964(R). But they may be used by loadable protocols. With Modbus enter here "X".
DONE*	OUT	BOOL	Request complete without errors, data sent Parameter *STATUS* = 00h
ERROR*	OUT	BOOL	Request complete with error Parameter *STATUS* contains error details
STATUS*	OUT	WORD	Specification of the error on *ERROR* = 1
*) Parameter is available until the next call of the FB.			

Release and cancel a request

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

Mechanism for startup synchronization

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

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

Error indication

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

Addressing

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

Data area

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

8.2 CP040

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

Description

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

> ***VIPA specific block***
>
> *The VIPA specific blocks can be found in the VIPA library.* *Chapter 4 'Include VIPA library' on page 103*

Parameters

Name	Declaration	Type	Description
REQ	IN	BOOL	Release SEND with positive edge.
R	IN	BOOL	Release synchronous reset.
LADDR	IN	INT	Logical base address of the CP.
DB_NO	IN	INT	Number of DB containing data to send.
DBB_NO	IN	INT	Data byte number - send data starting from data byte.
LEN	IN	INT	Length of telegram in byte, to be sent.
IO_SIZE	IN	WORD	Configured IO size of the module.
DONE *	OUT	BOOL	Send order finished without errors.
ERROR *	OUT	BOOL	Send order finished with errors. Parameter *STATUS* contains the error information.
STATUS *	OUT	WORD	Specification of the error with *ERROR* = 1.
CONTROL	IN_OUT	BYTE	Divided byte with RECEIVE handling block: SEND (bit 0 … 3), RECEIVE (bit 4 … 7).

*) Parameter is available until the FB is called.

REQ

Request - Send release:

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

R

Synchronous reset:

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

LADDR

Peripheral address:

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

DB_NO

Data block number:

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

DBB_NO

Data byte number:

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

LEN

Length:

- Length of the user data to be sent.
- It is: 1 ≤ *LEN* ≤ 1024.

IO_SIZE

Size I/O area:

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

DONE

DONE:

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

ERROR

ERROR:

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

STATUS

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

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

CONTROL

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

Error indication

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

8.2.2 FB 61 - RECEIVE - Receive from System SLIO CP 040

Description

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

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

Parameters

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

EN_R

Enable Receive - Release to read:

- With signal status "1" at *EN_R* the examination, whether data from the CP are read, is released. Depending upon the number of data, a data transfer can run over several program cycles.
- At any time a current order may be cancelled with signal state "0" of *EN_R*. Here the cancelled receipt order is finished with an error message (*STATUS*).
- The Receive function is deactivated as long as *EN_R* is statically set to "0".

R

Synchronous reset:

- For the initialization RECEIVE is once to be called in the start-up OB with every parameter and set *R*.
- At any time a current order may be cancelled and the FB may be set to initial state with signal state "1" of *R*.
- The Receive function is deactivated as long as *R* is statically set to "1".

LADDR

Peripheral address:

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

DB_NO

Data block number:

- Number of the data block, which contains the data are read.
- Zero is not permitted.

DBB_NO

Data byte number:

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

IO_SIZE

Size I/O area:

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

LEN

Length:

- Length of the user data to be sent.
- It is: 1 ≤ *LEN* ≤ 1024.

NDR

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

ERROR

ERROR:

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

STATUS

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

STATUS	Description
0000h	No error found
0202h	Handling block and CP are not synchronous (Remedy: Start synchronous reset)

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

CONTROL

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

Error indication

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

8.2.3 FB 65 - CP040_COM - Communication SLIO CP 040

Description

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

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *➪ Chapter 4 ‘Include VIPA library’ on page 103*

Parameters

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

REQ_SEND

Request - Send release:

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

EN_RECV

Enable receive data.

RESET

Synchron Reset:

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

ADDR_IN

Peripheral input address:

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

ADDR_OUT

Peripheral output address:

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

DB_NO_SEND

Number of the DB SEND:

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

DBB_NO_SEND

Data byte number SEND:

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

LEN_SEND

Length SEND:

- Length of the user data to be sent.
- It is: 1 ≤ *LEN_SEND* ≤ 1024.

DB_NO_RECV

Number of the DB RECV:

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

DBB_NO_RECV

Data byte number RECV:

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

IO_SIZE

Size I/O area:

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

DONE_SEND

DONE_SEND is set at order ready without errors and *STATUS_SEND* = 0000h.

ERROR_SEND

ERROR_SEND is set at order ready with error. Here *STATUS_SEND* contains the corresponding error message.

STATUS_SEND

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

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

LEN_RCV

Length Receive:

- Length of the received telegram in byte.
- 1 ≤ *LEN_RCV* ≤ 1024

NDR_RCV

New data ready:

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

ERROR_RCV

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

STATUS_RCV

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

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

Error indication

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

8.3 CP240

8.3.1 FC 0 - SEND - Send to CP 240

Description

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

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

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.

8.3.2 FC 1 - RECEIVE - Receive from CP 240

Description

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

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *↳ Chapter 4 'Include VIPA library' on page 103*

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

Name	Declara-tion	Type	Comment
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.

8.3.3 FC 8 - STEUERBIT - Modem functionality CP 240

Description

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

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

VIPA specific block
The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

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)

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.

8.3.4 FC 9 - SYNCHRON_RESET - Synchronization CPU and CP 240

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

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

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

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.

8.3.5 FC 11 - ASCII_FRAGMENT - Receive fragmented from CP 240

Description

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

> ***VIPA specific block***
> *The VIPA specific blocks can be found in the VIPA library.* *Chapter 4 'Include VIPA library' on page 103*

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

Name	Declaration	Type	Comment
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.

9 EtherCAT Communication

9.1 SDO Communication

9.1.1 FB 52 - SDO_READ - Read access to Object Dictionary Area

Description

With this block, you will have read access to the object directory of the EtherCAT slave stations and EtherCAT master. The block operates asynchronously, that is, processing covers multiple FB calls. Start the job by calling FB 52 with REQ = 1. The job status is displayed via the output parameters BUSY and RETVAL. The record set transmission is completed when the output parameter BUSY = FALSE. The error handling happens with the parameters ERROR, ERROR_ID and RETVAL.

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

Parameters

Parameter	Declaration	Data type	Description
REQ	IN	BOOL	REQ = 1: activates the SDO access at rising edge.
ID	IN	WORD	Logical base address of the EtherCAT slave station respectively master in the hardware configuration. With an output module bit 15 must be set (example for address 5: ID:=DW#16#8005). With a combination module you have to set the lower one of the two addresses.
INDEX	IN	WORD	Index of the object for the SDO access.
SUBINDEX	IN	BYTE	Sub index of the object for the SDO access.
COMPL_ACCESS	IN	BOOL	This parameter defines whether only a single sub-index, or the entire object is to be read.
MLEN	IN	INT	Maximum length of the data to be read.
VALID	OUT	BOOL	indicates that a new record set was received and is valid.
BUSY	OUT	BOOL	This parameter indicates the status of the SDO access. *BUSY* = 1: SDO access is not yet terminated.
ERROR	OUT	BOOL	*ERROR* = 1: A read error has occurred.
RETVAL	OUT	INT	Return value (0 = OK)
ERROR_ID	OUT	DWORD	Bus specific error code. If there was an error during the SDO access, the SDO abort error code (EtherCAT error code) can be found here.

Parameter	Declaration	Data type	Description
LEN	OUT	INT	Length of the read data.
RECORD	IN_OUT	ANY	Area of the read data.

Special features at *COMPL_ACCESS* (CompleteAccess)

With the activation of the parameter *COMPL_ACCESS* the following is to be considered:

- With *COMPL_ACCESS* = true only *SUBINDEX* 0 or 1 is allowed! Otherwise you will get an error message.
- With *COMPL_ACCESS* = true for *SUBINDEX* 0 2 bytes are read, because *SUBINDEX* 1 has an offset of 2 byte.

RETVAL (return value)

In addition to the module specific error codes, which are listed here, also the general error codes for FC/SFC as return value are possible. *Chapter 2.1 'General and Specific Error Information RET_VAL' on page 67*

RETVAL	Description	Error code in *ERROR_ID*
0x80A0	Negative acknowledgement while reading the module.	yes
0x80A1	Negative acknowledgement while writing the module.	yes
0x80A3	General protocol error.	yes
0x80A5	Internal error.	Value = 0: no
		Value ≠ 0: yes
0x80A7	Module is occupied (Timeout).	yes
0x80A9	Feature not supported by the module.	yes
0x80AA	Module reports a manufacturer-specific error in its application.	yes
0x80B0	Data record not known in module / Illegal data record number.	yes
0x80B4	Module reports access to an invalid area.	yes
0x80B5	Module not ready.	yes
0x80B6	Module denies access.	yes
0x80B7	Module reports an invalid range for a parameter or value.	yes
0x80B8	Module reports an invalid parameter.	yes
0x80B9	Module reports an invalid type: Buffer too small (reading subsets is not possible).	yes
0x80C2	The module currently processes the maximum possible jobs for a CPU.	yes

RETVAL	Description	Error code in *ERROR_ID*
0x80C3	The required operating resources are currently occupied.	no
0x80C4	Internal temporary error: Job could not be carried out.	yes
0x80C5	Module not available.	yes
0x80D2	Error on reading an SDO due to wrong call parameters.	yes

ERROR_ID

On a *RETVAL* more information can be found in the *ERROR_ID* if available. Otherwise *ERROR_ID* is 0.

Internal error	Description
0x00000000	No error
0x98110001	Feature not supported
0x98110002	Invalid Index
0x98110003	Invalid Offset
0x98110005	Invalid Size
0x98110006	Invalid Data
0x98110007	Not ready
0x98110008	Busy
0x9811000A	No Memory left
0x9811000B	Invalid Parameter
0x9811000C	Not Found
0x9811000E	Invalid state
0x98110010	Timeout
0x98110011	Open Failed
0x98110012	Send Failed
0x98110014	Invalid Command
0x98110015	Unknown Mailbox Protocol Command
0x98110016	Access Denied
0x98110024	Slave error
0x9811002D	Ethernet link cable disconnected
0x98110031	No mailbox support

CoE Error codes	Description	CoE slave abort code
0x98110040	SDO: Toggle bit not alternated	0x05030000
0x98110041	SDO protocol timed out	0x05040000
0x98110042	SDO: Client/server command specifier not valid or unknown	0x05040001

CoE Error codes	Description	CoE slave abort code
0x98110043	SDO: Invalid block size (block mode only)	0x05040002
0x98110044	SDO: Invalid sequence number (block mode only)	0x05040003
0x98110045	SDO: CRC error (block mode only)	0x05040004
0x98110046	SDO: Out of memory	0x05040005
0x98110047	SDO: Unsupported access to an object	0x06010000
0x98110048	SDO: Attempt to read a write only object	0x06010001
0x98110049	SDO: Attempt to write a read only object	0x06010002
0x9811004A	SDO: Object does not exist in the object dictionary	0x06020000
0x9811004B	SDO: Object cannot be mapped to the PDO	0x06040041
0x9811004C	SDO: The number and length of the objects to be mapped would exceed PDO length	0x06040042
0x9811004D	SDO: General parameter incompatibility reason	0x06040043
0x9811004E	SDO: General internal incompatibility in the device	0x06040047
0x9811004F	SDO: Access failed due to an hardware error	0x06060000
0x98110050	SDO: Data type does not match, length of service parameter does not match	0x06070010
0x98110051	SDO: Data type does not match, length of service parameter too high	0x06070012
0x98110052	SDO: Data type does not match, length of service parameter too low	0x06070013
0x98110053	SDO: Sub-index does not exist	0x06090011
0x98110054	SDO: Value range of parameter exceeded (only for write access)	0x06090030
0x98110055	SDO: Value of parameter written too high	0x06090031
0x98110056	SDO: Value of parameter written too low	0x06090032
0x98110057	SDO: Maximum value is less than minimum value	0x06090036
0x98110058	SDO: General error	0x08000000
0x98110059	SDO: Data cannot be transferred or stored to the application	0x08000020
0x9811005A	SDO: Data cannot be transferred or stored to the application because of local control	0x08000021
0x9811005B	SDO: Data cannot be transferred or stored to the application because of the present device state	0x08000022
0x9811005C	SDO: 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)	0x08000023
0x9811005D	SDO: Unknown code	unknown
0x9811010E	Command not executed	Slave is not present at the bus

9.1.2 FB 53 - SDO_WRITE - Write access to Object Dictionary Area

Description

With this block, you will have write access to the object directory of the EtherCAT slave stations and EtherCAT master. The block operates asynchronously, that is, processing covers multiple FB calls. Start the job by calling FB 53 with REQ = 1. The job status is displayed via the output parameters BUSY and RETVAL. The record set transmission is completed when the output parameter BUSY = FALSE.

The error handling happens with the parameters ERROR, ERROR_ID and RETVAL.

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. Chapter 4 'Include VIPA library' on page 103

Parameters

Parameter	Declaration	Data type	Description
REQ	IN	BOOL	REQ = 1: activates the SDO access at rising edge.
ID	IN	WORD	Logical base address of the EtherCAT slave station respectively master in the hardware configuration. With an output module bit 15 must be set (example for address 5: ID:=DW#16#8005). With a combination module you have to set the lower one of the two addresses.
INDEX	IN	WORD	Index of the object for the SDO access.
SUBINDEX	IN	BYTE	Sub index of the object for the SDO access.
COMPL_ACCESS	IN	BOOL	This parameter defines whether only a single sub-index, or the entire object is to be written.
LEN	IN	INT	Maximum length of the data to be written.
DONE	OUT	BOOL	indicates that a new record set was written.
BUSY	OUT	BOOL	This parameter indicates the status of the SDO access. *BUSY* = 1: SDO access is not yet terminated.
ERROR	OUT	BOOL	*ERROR* = 1: A write error has occurred.
RETVAL	OUT	INT	Return value (0 = OK)
ERROR_ID	OUT	DWORD	Bus specific error code. If there was an error during the SDO access, the SDO abort error code (EtherCAT error code) can be found here.
LEN	OUT	INT	Length of the data to be written.
RECORD	IN_OUT	ANY	Area of the data to be written.

Special features at *COMPL_ACCESS* (CompleteAccess)

With the activation of the parameter *COMPL_ACCESS* the following is to be considered:

- With *COMPL_ACCESS* = true only *SUBINDEX* 0 or 1 is allowed! Otherwise you will get an error message.
- With *COMPL_ACCESS* = true for *SUBINDEX* 0 2 bytes are written, because *SUBINDEX* 1 has an offset of 2 bytes.

RETVAL (return value)

In addition to the module specific error codes, which are listed here, also the general error codes for FC/SFC as return value are possible. *Chapter 2.1 'General and Specific Error Information RET_VAL' on page 67*

RETVAL	Description	Error code in *ERROR_ID*
0x80A0	Negative acknowledgement while reading the module.	yes
0x80A1	Negative acknowledgement while writing the module.	yes
0x80A3	General protocol error.	yes
0x80A5	Internal error.	Value = 0: no
		Value ≠ 0: yes
0x80A7	Module is occupied (Timeout).	yes
0x80A9	Feature not supported by the module.	yes
0x80AA	Module reports a manufacturer-specific error in its application.	yes
0x80B0	Data record not known in module / Illegal data record number.	yes
0x80B4	Module reports access to an invalid area.	yes
0x80B5	Module not ready.	yes
0x80B6	Module denies access.	yes
0x80B7	Module reports an invalid range for a parameter or value.	yes
0x80B8	Module reports an invalid parameter.	yes
0x80B9	Module reports an invalid type: Buffer too small (writing subsets is not possible).	yes
0x80C2	The module currently processes the maximum possible jobs for a CPU.	yes
0x80C3	The required operating resources are currently occupied.	no
0x80C4	Internal temporary error: Job could not be carried out.	yes
0x80C5	Module not available.	yes
0x80D2	Error on reading an SDO due to wrong call parameters.	yes

ERROR_ID

On a *RETVAL* more information can be found in the *ERROR_ID* if available. Otherwise *ERROR_ID* is 0.

Internal error	Description
0x00000000	No error
0x98110001	Feature not supported
0x98110002	Invalid Index
0x98110003	Invalid Offset
0x98110005	Invalid Size
0x98110006	Invalid Data
0x98110007	Not ready
0x98110008	Busy
0x9811000A	No Memory left
0x9811000B	Invalid Parameter
0x9811000C	Not Found
0x9811000E	Invalid state
0x98110010	Timeout
0x98110011	Open Failed
0x98110012	Send Failed
0x98110014	Invalid Command
0x98110015	Unknown Mailbox Protocol Command
0x98110016	Access Denied
0x98110024	Slave error
0x9811002D	Ethernet link cable disconnected
0x98110031	No mailbox support

CoE Error codes	Description	CoE slave abort code
0x98110040	SDO: Toggle bit not alternated	0x05030000
0x98110041	SDO protocol timed out	0x05040000
0x98110042	SDO: Client/server command specifier not valid or unknown	0x05040001
0x98110043	SDO: Invalid block size (block mode only)	0x05040002
0x98110044	SDO: Invalid sequence number (block mode only)	0x05040003
0x98110045	SDO: CRC error (block mode only)	0x05040004
0x98110046	SDO: Out of memory	0x05040005
0x98110047	SDO: Unsupported access to an object	0x06010000
0x98110048	SDO: Attempt to read a write only object	0x06010001
0x98110049	SDO: Attempt to write a read only object	0x06010002

CoE Error codes	Description	CoE slave abort code
0x9811004A	SDO: Object does not exist in the object dictionary	0x06020000
0x9811004B	SDO: Object cannot be mapped to the PDO	0x06040041
0x9811004C	SDO: The number and length of the objects to be mapped would exceed PDO length	0x06040042
0x9811004D	SDO: General parameter incompatibility reason	0x06040043
0x9811004E	SDO: General internal incompatibility in the device	0x06040047
0x9811004F	SDO: Access failed due to an hardware error	0x06060000
0x98110050	SDO: Data type does not match, length of service parameter does not match	0x06070010
0x98110051	SDO: Data type does not match, length of service parameter too high	0x06070012
0x98110052	SDO: Data type does not match, length of service parameter too low	0x06070013
0x98110053	SDO: Sub-index does not exist	0x06090011
0x98110054	SDO: Value range of parameter exceeded (only for write access)	0x06090030
0x98110055	SDO: Value of parameter written too high	0x06090031
0x98110056	SDO: Value of parameter written too low	0x06090032
0x98110057	SDO: Maximum value is less than minimum value	0x06090036
0x98110058	SDO: General error	0x08000000
0x98110059	SDO: Data cannot be transferred or stored to the application	0x08000020
0x9811005A	SDO: Data cannot be transferred or stored to the application because of local control	0x08000021
0x9811005B	SDO: Data cannot be transferred or stored to the application because of the present device state	0x08000022
0x9811005C	SDO: 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)	0x08000023
0x9811005D	SDO: Unknown code	unknown
0x9811010E	Command not executed	Slave is not present at the bus

10 Device Specific

10.1 Frequency Measurement

10.1.1 FC 300 ... 303 - Frequency measurement SLIO consistent

Overview

The following VIPA specific functions are used to control the System SLIO frequency measurement modules, which are connected via PROFIBUS, PROFINET or EtherCAT. The usage with EtherCAT is only possible at an EtherCAT CPU from VIPA. By this functions SFC 14 - DPRD_DAT respectively SFC 15 - DPWR_DAT for consistent read respectively write access to the data are internally called. Error messages of these blocks are reported by the parameter *ERROR*.

Function	Symbol	Comment
FC 300	FM_SET_CONTROL	Function to control the frequency measurement with integrated consistent access.
FC 301	FM_GET_PERIOD	Function to calculate the period duration with integrated consistent access.
FC 302	FM_GET_FREQUENCY	Function to calculate the frequency with integrated consistent access.
FC 303	FM_GET_SPEED	Function to calculate the rotational speed with integrated consistent access.

10.1.2 FC 300 - FM_SET_CONTROL - Control frequency measurement consistent

Description

The System SLIO Frequency measurement module is controlled by the FC 300 FM_SET_CONTROL. By this function the SFC 15 - DPWR_DAT for consistent write access of data is called. Here error messages of the block are reported by *ERROR*.

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

10.1.2.1 Parameters

Parameter	Declaration	Data type	Memory block	Description
ENABLE_FM	INPUT	BOOL	I, Q, M, D, L	Enable frequency measurement
LADDR_OUT	INPUT	WORD	I, Q, M, D, L	Logical base address
PRESET_CH0	INPUT	DINT	I, Q, M, D, L	Channel 0: Measurement period
PRESET_CH1	INPUT	DINT	I, Q, M, D, L	Channel 1: Measurement period
DONE	OUTPUT	BOOL	I, Q, M, D, L	Ready signal (TRUE = OK)
ERROR	OUTPUT	WORD	I, Q, M, D, L	Return value (0 = OK)

ENABLE_FM

With setting *ENABLE_FM* the *measuring periods*, which were preset by PRESET_CH0/1, are transferred to the channels and the measurement of both channels are started. Both frequency meters are stopped by resetting *ENABLE_FM*.

Only while ENABLE_FM is set, evaluated values can be retrieved from the module. Otherwise you get the error message that the channels are disabled.

LADDR_OUT

Configured base address of the output area of the System SLIO frequency measurement module, which is to be written to. The address must be in hexadecimal notation.

(Example: Address 100: *LADDR_OUT* : = W#16#64).

PRESET_CHx

Enter here the measurement period in µs for the corresponding channel.

Range of values: 1µs ... 8 388 607µs

DONE

Ready signal of the function

- TRUE: Function was finished without error.
- FALSE: Function is not active respectively there is an error.

ERROR (Return value)

The following code can be reported:

Code	Description
0x0000	No error
0x80D2	Channel 0: Input value measurement period ≤ 0
0x80D3	Channel 1: Input value measurement period ≤ 0
0x80D4	Channel 0: Input value measurement period > 8 388 607µs
0x80D5	Channel 1: Input value measurement period > 8 388 607µs

10.1.2.2 Errors of the internally called SFC 15

Code	Description
0x808x	System error on the bus coupler
0x8090	*LADDR_OUT* is wrong, possible reasons: ■ there is no module configured on this address ■ limitation of the length of consistent data was not considered ■ Basic address in parameter *LADDR_OUT* was not entered in hexadecimal type
0x8093	There is no bus coupler existing for *LADDR_OUT*, from which consistent data can be read.
0x80A0	An access error was detected during peripheral access.
0x80B0	System error on the bus coupler
0x80B1	Specified length of the source area does not correspond to the configured user data length.
0x80B2	System error on the bus coupler
0x80B3	System error on the bus coupler
0x80C1	The data from the previous read request on the module are not processed by the module, yet.
0x80C2	System error on the bus coupler
0x80Fx	System error on the bus coupler
0x85xy	System error on the bus coupler
0x8xyy	General error information For details, please refer to OPL_SP7 "Integrated Standard SFCs" at "General and Specific Error Information RET_VAL".

10.1.3 FC 301 - FM_GET_PERIOD - Calculate period duration consistent

Description

With the FC 301 FM_GET_PERIOD, you can calculate the period duration of the input signals of both channels. By this function internally SFC 14 - DPRD_DAT for consistent reading of user data is called. Here, the error messages of the function block are returned by *ERROR*.

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

10.1.3.1 Parameters

Parameter	Declaration	Data type	Memory block	Description
LADDR_IN	INPUT	WORD	I, Q, M, D, L	Logical base input address
DONE	OUTPUT	BOOL	I, Q, M, D, L	Ready signal (TRUE = OK)
ERROR	OUTPUT	WORD	I, Q, M, D, L	Return value (0 = OK)
PERIOD_CH0	OUTPUT	DINT	I, Q, M, D, L	Channel 0: Period duration
PERIOD_CH1	OUTPUT	DINT	I, Q, M, D, L	Channel 1: Period duration

LADDR_IN

Configured base address of the input area of the System SLIO frequency measurement module, which is to be read from. The address must be in hexadecimal notation.

(Example: Address 100: *LADDR_IN*: = W#16#64).

DONE

Ready signal of the function

- TRUE: Function was finished without error.
- FALSE: Function is not active respectively there is an error.

PERIOD_CHx

Currently determined period duration of the corresponding channel.

ERROR (Return value)

The following codes can be returned:

Code	Description
0x0000	No error
0x80D0	Channel 0 not in status active
0x80D1	Channel 1 not in status active
0x80DC	Channel 0: Measured time value < 0
0x80DD	Channel 1: Measured time value < 0
0x80DE	Channel 0: Measured time value > 0x7FFFFFF
0x80DF	Channel 1: Measured time value > 0x7FFFFFF
0x80E0	Channel 0: Determined number of edges = 0
0x80E1	Channel 1: Determined number of edges = 0
0x80E2	Channel 0: Determined number of edges < 0
0x80E3	Channel 1: Determined number of edges < 0
0x80E4	Channel 0: Determined number of edges > 0xFFFFFF
0x80E5	Channel 1: Determined number of edges > 0xFFFFFF

Code	Description
0x80E8	Channel 0: No valid measurement within the entered measurement period.
0x80E9	Channel 1: No valid measurement within the entered measurement period.

10.1.3.2 Error of the internal called SFC 14

Code	Description
0x808x	System error on the bus coupler
0x8090	*LADDR_IN* is not correct, possible reasons: ■ there is no module configured on this address ■ limitation of the length of consistent data was not considered ■ Basic address in parameter *LADDR_IN* was not entered in hexadecimal type
0x8093	There is no bus coupler existing for *LADDR_IN*, to which consistent data can be written.
0x80A0	An access error was detected during peripheral access.
0x80B0	System error on the bus coupler
0x80B1	Specified length of the source area does not correspond to the configured user data length.
0x80B2	System error on the bus coupler
0x80B3	System error on the bus coupler
0x80C1	The data from the previous write request on the module are not processed by the module, yet.
0x80C2	System error on the bus coupler
0x80Fx	System error on the bus coupler
0x85xy	System error on the bus coupler
0x8xyy	General error information For details, please refer to OPL_SP7 "Integrated Standard SFCs" at "General and Specific Error Information RET_VAL".

10.1.4 FC 302 - FM_GET_FREQUENCY - Calculate frequency consistent

Description

With the FC 302 FM_GET_FREQUENCY, you can calculate the frequency of the input signals of both channels. By this function internally SFC 14 - DPRD_DAT for consistent reading of user data is called. Here, the error messages of the function block are returned by *ERROR*.

VIPA specific block
The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

10.1.4.1 Parameters

Parameter	Declaration	Data type	Memory block	Description
LADDR_IN	INPUT	WORD	I, Q, M, D, L	Logical base input address
DONE	OUTPUT	BOOL	I, Q, M, D, L	Ready signal (TRUE = OK)
ERROR	OUTPUT	WORD	I, Q, M, D, L	Return value (0 = OK)
FREQUENCY_CH0	OUTPUT	DINT	I, Q, M, D, L	Channel 0: Frequency
FREQUENCY_CH1	OUTPUT	DINT	I, Q, M, D, L	Channel 1: Frequency

LADDR_IN

Configured base address of the input area of the System SLIO frequency measurement module, which is to be read from. The address must be in hexadecimal notation.

(Example: Address 100: *LADDR_IN*: = W#16#64).

DONE

Ready signal of the function

- TRUE: Function was finished without error.
- FALSE: Function is not active respectively there is an error.

FREQUENCY_CHx

Currently determined frequency of the corresponding channel in mHz.

ERROR (Return value)

The following codes can be returned:

Code	Description
0x0000	No error
0x80D0	Channel 0 not in status active
0x80D1	Channel 1 not in status active
0x80DA	Channel 0: Measured time value = 0
0x80DB	Channel 1: Measured time value = 0
0x80DC	Channel 0: Measured time value < 0
0x80DD	Channel 1: Measured time value < 0
0x80DE	Channel 0: Measured time value > 0x7FFFFFF

Code	Description
0x80DF	Channel 1: Measured time value > 0x7FFFFFF
0x80E2	Channel 0: Determined number of edges < 0
0x80E3	Channel 1: Determined number of edges < 0
0x80E4	Channel 0: Determined number of edges > 0xFFFFFF
0x80E5	Channel 1: Determined number of edges > 0xFFFFFF
0x80E6	Channel 0: Frequency > 600kHz
0x80E7	Channel 1: Frequency > 600kHz
0x80E8	Channel 0: No valid measurement within the entered measurement period.
0x80E9	Channel 1: No valid measurement within the entered measurement period.

10.1.4.2 Error of the internal called SFC 14

Code	Description
0x808x	System error on the bus coupler
0x8090	*LADDR_IN* is not correct, possible reasons: ■ there is no module configured on this address ■ limitation of the length of consistent data was not considered ■ Basic address in parameter *LADDR_IN* was not entered in hexadecimal type
0x8093	There is no bus coupler existing for *LADDR_IN*, to which consistent data can be written.
0x80A0	An access error was detected during peripheral access.
0x80B0	System error on the bus coupler
0x80B1	Specified length of the source area does not correspond to the configured user data length.
0x80B2	System error on the bus coupler
0x80B3	System error on the bus coupler
0x80C1	The data from the previous write request on the module are not processed by the module, yet.
0x80C2	System error on the bus coupler
0x80Fx	System error on the bus coupler

Code	Description
0x85xy	System error on the bus coupler
0x8xyy	General error information For details, please refer to OPL_SP7 "Integrated Standard SFCs" at "General and Specific Error Information RET_VAL".

10.1.5 FC 303 - FM_GET_SPEED - Calculate rotational speed consistent

Description

With the FC 303 FM_GET_SPEED, you can calculate the rotational speed of the input signals of both channels. By this function internally SFC 14 - DPRD_DAT for consistent reading of user data is called. Here, the error messages of the function block are returned by *ERROR*.

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *➭ Chapter 4 'Include VIPA library' on page 103*

10.1.5.1 Parameters

Parameter	Declaration	Data type	Memory block	Description
LADDR_IN	INPUT	WORD	I, Q, M, D, L	Logical base input address
RESOLUTION_CH0	INPUT	DINT	I, Q, M, D, L	Channel 0: Resolution of the sensor
RESOLUTION_CH1	INPUT	DINT	I, Q, M, D, L	Channel 1: Resolution of the sensor
DONE	OUTPUT	BOOL	I, Q, M, D, L	Ready signal (TRUE = OK)
ERROR	OUTPUT	WORD	I, Q, M, D, L	return value (0 = OK)
SPEED_CH0	OUTPUT	DINT	I, Q, M, D, L	Channel 0: Rotational speed
SPEED_CH1	OUTPUT	DINT	I, Q, M, D, L	Channel 1: Rotational speed

LADDR_IN

Configured base address of the input area of the System SLIO frequency measurement module, which is to be read from. The address must be in hexadecimal notation.

(Example: Address 100: *LADDR_IN*: = W#16#64).

RESOLUTION_CHx

Enter here the resolution in increments per revolution for the corresponding channel .

DONE

Ready signal of the function

- TRUE: Function was finished without error.
- FALSE: Function is not active respectively there is an error.

SPEED_CHx

Currently determined rotational speed of the corresponding channel in revolutions per minute (rpm).

ERROR (Return value)

The following codes can be returned:

Code	Description
0x0000	No error
0x80D0	Channel 0 not in status active
0x80D1	Channel 1 not in status active
0x80D6	Channel 0: Input value RESOLUTION_CH0 = 0
0x80D7	Channel 1: Input value RESOLUTION_CH1 = 0
0x80D8	Channel 0: Input value RESOLUTION_CH0 < 0
0x80D9	Channel 1: Input value RESOLUTION_CH1 < 0
0x80DA	Channel 0: Measured time value = 0
0x80DB	Channel 1: Measured time value = 0
0x80DC	Channel 0: Measured time value < 0
0x80DD	Channel 1: Measured time value < 0
0x80DE	Channel 0: Measured time value > 0x7FFFFFF
0x80DF	Channel 1: Measured time value > 0x7FFFFFF
0x80E2	Channel 0: Determined number of edges < 0
0x80E3	Channel 1: Determined number of edges < 0
0x80E4	Channel 0: Determined number of edges > 0xFFFFFF
0x80E5	Channel 1: Determined number of edges > 0xFFFFFF
0x80E6	Channel 0: Determined rotational speed > max (DINT)
0x80E7	Channel 1: Determined rotational speed > max (DINT)

Code	Description
0x80E8	Channel 0: No valid measurement within the entered measurement period.
0x80E9	Channel 1: No valid measurement within the entered measurement period.

10.1.5.2 Error of the internal called SFC 14

Code	Description
0x808x	System error on the bus coupler
0x8090	*LADDR_IN* is not correct, possible reasons: ■ there is no module configured on this address ■ limitation of the length of consistent data was not considered ■ Basic address in parameter *LADDR_IN* was not entered in hexadecimal type
0x8093	There is no bus coupler existing for *LADDR_IN*, to which consistent data can be written.
0x80A0	An access error was detected during peripheral access.
0x80B0	System error on the bus coupler
0x80B1	Specified length of the source area does not correspond to the configured user data length.
0x80B2	System error on the bus coupler
0x80B3	System error on the bus coupler
0x80C1	The data from the previous write request on the module are not processed by the module, yet.
0x80C2	System error on the bus coupler
0x80Fx	System error on the bus coupler
0x85xy	System error on the bus coupler
0x8xyy	General error information For details, please refer to OPL_SP7 "Integrated Standard SFCs" at "General and Specific Error Information RET_VAL".

10.1.6 FC 310 ... 313 - Frequency measurement SLIO

Overview

The following VIPA specific functions are used to control the System SLIO frequency measurement modules, if the consistency of the data are ensured by the bus protocol and consistent reading respectively writing with SFC 14 respectively SFC 15 is not possible. Within the functions there are "FM_..." parameters, whose content is to be consistently connected to the corresponding input or output area of the frequency measurement module by means of the bus system. By calling the appropriate function the corresponding "FM_..." parameters are automatically filled by the function.

Function	Symbol	Comment
FC 310	FM_CONTROL	Function to control the frequency measurement
FC 311	FM_CALC_PERIOD	Function to calculate the period duration
FC 312	FM_CALC_FREQUENCY	Function to calculate the frequency
FC 313	FM_CALC_SPEED	Function to calculate the rotational speed

10.1.7 FC 310 - FM_CONTROL - Control frequency measurement

Description

The System SLIO Frequency measurement module is controlled by the FC 310 FM_CONTROL. Since this FC does not internally call a block for consistent write access of data, you have to ensure consistent data transfer in your system.

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *↳ Chapter 4 'Include VIPA library' on page 103*

10.1.7.1 Parameters

Parameter	Declaration	Data type	Memory block	Description
ENABLE_FM	INPUT	BOOL	I, Q, M, D, L	Enable frequency measurement
PRESET_CH0	INPUT	DINT	I, Q, M, D, L	Channel 0: Measurement period
PRESET_CH1	INPUT	DINT	I, Q, M, D, L	Channel 1: Measurement period

Parameter	Declaration	Data type	Memory block	Description
DONE	OUTPUT	BOOL	I, Q, M, D, L	Ready signal (TRUE = OK)
ERROR	OUTPUT	WORD	I, Q, M, D, L	return value (0 = OK)
FM_PRESET_PERIOD_CH0	OUTPUT	DWORD	I, Q, M, D, L	Setpoint value for frequency measurement module output address: +0
FM_PRESET_PERIOD_CH1	OUTPUT	DWORD	I, Q, M, D, L	Setpoint value for frequency measurement module output address: +4
FM_CONTROL_CH0	OUTPUT	WORD	I, Q, M, D, L	Setpoint value for frequency measurement module output address: +8
FM_CONTROL_CH1	OUTPUT	WORD	I, Q, M, D, L	Setpoint value for frequency measurement module output address: +10

ENABLE_FM

With setting *ENABLE_FM* the corresponding CONTROL is generated and issued via *FM_CONTROL_CHx*. The measurement of both channels is started as soon as the content of *FM_CONTROL_CHx* was consistent transferred by the bus system to the frequency measurement module. The measurement of both channels is stopped by resetting *ENABLE_FM*, after *FM_CONTROL_CHx* was consistent transferred to the frequency measurement module.

Only as long as the frequency meters are started, evaluated values can be retrieved from the module. Otherwise you get the error message that the channels are disabled.

PRESET_CHx

Enter here the measurement period in µs for the corresponding channel.

Range of values: 1µs ... 8 388 607µs

DONE

Ready signal of the function

- TRUE: Function was finished without error.
- FALSE: Function is not active respectively there is an error.

FM_PRESET_ PERIOD_CHx

This parameter contains the measuring period for channel 0 respectively channel 1. The content is to be consistent connected with address +0 respectively +4 of the output area of the frequency measurement module, via the according bus system.

FM_CONTROL_CHx

This parameter contains CONTROL, which is generated by *ENABLE_FM*. The content for channel 0 respectively channel 1 is to be consistent connected with address +8 respectively +10 of the output area of the frequency measurement module, via the according bus system.

ERROR (Return value)

The following code can be reported:

Code	Description
0x0000	No error
0x80D2	Channel 0: Input value measurement period ≤ 0
0x80D3	Channel 1: Input value measurement period ≤ 0
0x80D4	Channel 0: Input value measurement period > 8 388 607µs
0x80D5	Channel 1: Input value measurement period > 8 388 607µs

10.1.8 FC 311 - FM_CALC_PERIOD - Calculate period duration

Description

With the FC 311 FM_CALC_PERIOD, you can calculate the period duration of the input signals of both channels. Since this FC does not internally call a block for consistent read access of data, you have to ensure consistent data transfer in your system.

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *⇨ Chapter 4 'Include VIPA library' on page 103*

10.1.8.1 Parameters

Parameter	Declaration	Data type	Memory block	Description
FM_PERIOD_CH0	INPUT	DWORD	I, Q, M, D, L	Actual value of frequency measurement module input address: +0
FM_PERIOD_CH1	INPUT	DWORD	I, Q, M, D, L	Actual value of frequency measurement module input address: +4
FM_RISING_EDGES_CH0	INPUT	DWORD	I, Q, M, D, L	Actual value of frequency measurement module input address: +8
FM_RISING_EDGES_CH1	INPUT	DWORD	I, Q, M, D, L	Actual value of frequency measurement module input address: +12
FM_STATUS_CH0	INPUT	WORD	I, Q, M, D, L	Actual value of frequency measurement module input address: +16
FM_STATUS_CH1	INPUT	WORD	I, Q, M, D, L	Actual value of frequency measurement module input address: +18
DONE	OUTPUT	BOOL	I, Q, M, D, L	Ready signal (TRUE = OK)
ERROR	OUTPUT	WORD	I, Q, M, D, L	Return value (0 = OK)
PERIOD_CH0	OUTPUT	DINT	I, Q, M, D, L	Channel 0: Period duration
PERIOD_CH1	OUTPUT	DINT	I, Q, M, D, L	Channel 1: Period duration

FM_PERIOD_CHx

This parameter contains the measured time value of channel 0 respectively channel 1. The content is to be consistent connected with address +0 respectively +4 of the input area of the frequency measurement module, via the according bus system.

FM_RISING_ EDGES_CHx

This parameter contains the determined number of rising edges for channel 0 respectively channel 1. The content is to be consistent connected with address +8 respectively +12 of the input area of the frequency measurement module, via the according bus system.

FM_STATUS_CHx

This parameter contains the status of channel 0 respectively channel 1. The content is to be consistent connected with address +16 respectively +18 of the input area of the frequency measurement module, via the according bus system.

DONE

Ready signal of the function

- TRUE: Function was finished without error.
- FALSE: Function is not active respectively there is an error.

PERIOD_CHx

Currently determined period duration of the corresponding channel in 100ns.

ERROR (Return value)

The following codes can be returned:

Code	Description
0x0000	No error
0x80D0	Channel 0 not in status active
0x80D1	Channel 1 not in status active
0x80DC	Channel 0: Measured time value < 0
0x80DD	Channel 1: Measured time value < 0
0x80DE	Channel 0: Measured time value > 0x7FFFFFF
0x80DF	Channel 1: Measured time value > 0x7FFFFFF
0x80E0	Channel 0: Determined number of edges = 0
0x80E1	Channel 1: Determined number of edges = 0
0x80E2	Channel 0: Determined number of edges < 0
0x80E3	Channel 1: Determined number of edges < 0
0x80E4	Channel 0: Determined number of edges > 0xFFFFFF
0x80E5	Channel 1: Determined number of edges > 0xFFFFFF
0x80E8	Channel 0: No valid measurement within the entered measurement period.
0x80E9	Channel 1: No valid measurement within the entered measurement period.

10.1.9 FC 312 - FM_CALC_FREQUENCY - Calculate frequency

Description

With the FC 312 FM_CALC_FREQUENCY, you can calculate the period duration of the input signals of both channels. Since this FC does not internally call a block for consistent read access of data, you have to ensure consistent data transfer in your system.

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

10.1.9.1 Parameters

Parameter	Declara-tion	Data type	Memory block	Description
FM_PERIOD_CH0	INPUT	DWORD	I, Q, M, D, L	Actual value of frequency measurement module input address: +0
FM_PERIOD_CH1	INPUT	DWORD	I, Q, M, D, L	Actual value of frequency measurement module input address: +4
FM_RISING_EDGES_CH0	INPUT	DWORD	I, Q, M, D, L	Actual value of frequency measurement module input address: +8
FM_RISING_EDGES_CH1	INPUT	DWORD	I, Q, M, D, L	Actual value of frequency measurement module input address: +12
FM_STATUS_CH0	INPUT	WORD	I, Q, M, D, L	Actual value of frequency measurement module input address: +16
FM_STATUS_CH1	INPUT	WORD	I, Q, M, D, L	Actual value of frequency measurement module input address: +18
DONE	OUTPUT	BOOL	I, Q, M, D, L	Ready signal (TRUE = OK)
ERROR	OUTPUT	WORD	I, Q, M, D, L	Return value (0 = OK)

Parameter	Declara-tion	Data type	Memory block	Description
FREQUENCY_CH0	OUTPUT	DINT	I, Q, M, D, L	Channel 0: Calculated frequency
FREQUENCY_CH1	OUTPUT	DINT	I, Q, M, D, L	Channel 1: Calculated frequency

FM_PERIOD_CHx

This parameter contains the measured time value of channel 0 respectively channel 1. The content is to be consistent connected with address +0 respectively +4 of the input area of the frequency measurement module, via the according bus system.

FM_RISING_ EDGES_CHx

This parameter contains the determined number of rising edges for channel 0 respectively channel 1. The content is to be consistent connected with address +8 respectively +12 of the input area of the frequency measurement module, via the according bus system.

FM_STATUS_CHx

This parameter contains the status of channel 0 respectively channel 1. The content is to be consistent connected with address +16 respectively +18 of the input area of the frequency measurement module, via the according bus system.

DONE

Ready signal of the function

- TRUE: Function was finished without error.
- FALSE: Function is not active respectively there is an error.

FREQUENCY_CHx

Currently determined frequency of the corresponding channel in mHz.

ERROR (Return value)

The following codes can be returned:

Code	Description
0x0000	No error
0x80D0	Channel 0 not in status active
0x80D1	Channel 1 not in status active
0x80DA	Channel 0: Measured time value = 0
0x80DB	Channel 1: Measured time value = 0
0x80DC	Channel 0: Measured time value < 0
0x80DD	Channel 1: Measured time value < 0
0x80DE	Channel 0: Measured time value > 0x7FFFFFF
0x80DF	Channel 1: Measured time value > 0x7FFFFFF

Code	Description
0x80E2	Channel 0: Determined number of edges < 0
0x80E3	Channel 1: Determined number of edges < 0
0x80E4	Channel 0: Determined number of edges > 0xFFFFFF
0x80E5	Channel 1: Determined number of edges > 0xFFFFFF
0x80E6	Channel 0: Frequency > 600kHz
0x80E7	Channel 1: Frequency > 600kHz
0x80E8	Channel 0: No valid measurement within the entered measurement period.
0x80E9	Channel 1: No valid measurement within the entered measurement period.

10.1.10 FC 313 - FM_CALC_SPEED - Calculate rotational speed

Description

With the FC 313 FM_CALC_SPEED, you can calculate the velocity of the input signals of both channels. Since this FC does not internally call a block for consistent read access of data, you have to ensure consistent data transfer in your system.

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

10.1.10.1 Parameters

Parameter	Declaration	Data type	Memory block	Description
FM_PERIOD_CH0	INPUT	DWORD	I, Q, M, D, L	Actual value of frequency measurement module input address: +0
FM_PERIOD_CH1	INPUT	DWORD	I, Q, M, D, L	Actual value of frequency measurement module input address: +4
FM_RISING_EDGES_CH0	INPUT	DWORD	I, Q, M, D, L	Actual value of frequency measurement module input address: +8

Parameter	Declaration	Data type	Memory block	Description
FM_RISING_EDGES_CH1	INPUT	DWORD	I, Q, M, D, L	Actual value of frequency measurement module input address: +12
FM_STATUS_CH0	INPUT	WORD	I, Q, M, D, L	Actual value of frequency measurement module input address: +16
FM_STATUS_CH1	INPUT	WORD	I, Q, M, D, L	Actual value of frequency measurement module input address: +18
RESOLUTION_CH0	INPUT	DINT	I, Q, M, D, L	Channel 0: Resolution of the sensor
RESOLUTION_CH1	INPUT	DINT	I, Q, M, D, L	Channel 1: Resolution of the sensor
DONE	OUTPUT	BOOL	I, Q, M, D, L	Ready signal (TRUE = OK)
ERROR	OUTPUT	WORD	I, Q, M, D, L	Return value (0 = OK)
SPEED_CH0	OUTPUT	DINT	I, Q, M, D, L	Channel 0: Calculated rotational speed
SPEED_CH1	OUTPUT	DINT	I, Q, M, D, L	Channel 1: Calculated rotational speed

FM_PERIOD_CHx

This parameter contains the measured time value for channel 0 respectively channel 1. The content is to be consistent connected with address +0 respectively +4 of the input area of the frequency measurement module, via the according bus system.

FM_RISING_EDGES_CHx

This parameter contains the determined number of rising edges for channel 0 respectively channel 1. The content is to be consistent connected with address +8 respectively +12 of the input area of the frequency measurement module, via the according bus system.

FM_STATUS_CHx

This parameter contains the status of channel 0 respectively channel 1. The content is to be consistent connected with address +16 respectively +18 of the input area of the frequency measurement module, via the according bus system.

RESOLUTION_CHx

Enter here the resolution in increments per revolution for the corresponding channel.

DONE

Ready signal of the function

- TRUE: Function was finished without error.
- FALSE: Function is not active respectively there is an error.

SPEED_CHx

Currently determined rotational speed of the corresponding channel in revolutions per minute (rpm).

ERROR (Return value)

The following codes can be returned:

Code	Description
0x0000	No error
0x80D0	Channel 0 not in status active
0x80D1	Channel 1 not in status active
0x80D6	Channel 0: Input value RESOLUTION_CH0 = 0
0x80D7	Channel 1: Input value RESOLUTION_CH1 = 0
0x80D8	Channel 0: Input value RESOLUTION_CH0 < 0
0x80D9	Channel 1: Input value RESOLUTION_CH1 < 0
0x80DA	Channel 0: Measured time value = 0
0x80DB	Channel 1: Measured time value = 0
0x80DC	Channel 0: Measured time value < 0
0x80DD	Channel 1: Measured time value < 0
0x80DE	Channel 0: Measured time value > 0x7FFFFFF
0x80DF	Channel 1: Measured time value > 0x7FFFFFF
0x80E2	Channel 0: Determined number of edges < 0
0x80E3	Channel 1: Determined number of edges < 0
0x80E4	Channel 0: Determined number of edges > 0xFFFFFF
0x80E5	Channel 1: Determined number of edges > 0xFFFFFF
0x80E6	Channel 0: Determined rotational speed > max (DINT)
0x80E7	Channel 1: Determined rotational speed > max (DINT)
0x80E8	Channel 0: No valid measurement within the entered measurement period.
0x80E9	Channel 1: No valid measurement within the entered measurement period.

10.2 Energy Measurement

10.2.1 FB 325 - EM_COM_1 - Communication with 031-1PA00

Overview

This module enables the communication with the module 031-1PA00 for energy metering and power measurement. For the communication a data block is necessary. Here the DB gets its structure from the UDT 325 EM_COM_1. The block has the following functionalities:

- Load default parameters after start-up
- Storage of parameters, limit values, measured values and messages
- Transfer of consistent measured values
- Definition of the measured values by means of an UDT structure
- Communication by means of telegram type and ID
- Functional diagnostics, connection monitoring and error message evaluation

> ***VIPA specific block***
>
> *The VIPA specific blocks can be found in the VIPA library.* *Chapter 4 'Include VIPA library' on page 103*

Parameter

Parameter	Declaration	Data type	Description
MODE	INPUT	BYTE	■ 0x01 = Data exchange via process data Currently only the MODE = 1 is supported
MEAS_DATA	IN_OUT	UDT	■ UDT for the measured values *Chapter 10.2.2 '4.1 UDT 325 - EM_DATA_R1 - Data structure for FB 325' on page 275*
CHANNEL_ IN	INPUT	ANY	Pointer to the input data ■ With MODE = 0x01 exclusively data type BYTE and length 16 are permitted. Example: P#E100.0 BYTE 16 or P#DB10.DBX0.0 BYTE 16
CHANNEL_OUT	INPUT	ANY	Pointer to the output data ■ With MODE = 0x01 exclusively data type BYTE and length 16 are permitted. Example: P#A100.0 BYTE 16 or P#DB10.DBX16.0 BYTE 16

10.2.2 4.1 UDT 325 - EM_DATA_R1 - Data structure for FB 325

UDT - Header

Name	Declaration	Data type	Description
Timeout	INPUT	TIME	■ Timeout for reading measured values
Polltime	INPUT	TIME	■ Interval for the periodic reading

Name	Declaration	Data type	Description
Control_Global	INPUT	BYTE	0: de-activated, 1: activated ■ Bit 0: Periodic execution according to the *Poll-time* (default) ■ Bit 1: Immediate execution - bit is to be reset after the execution. ■ Bit 6 ... 2: reserved ■ Bit 7: Re-initialization of the block by the configuration is sent again
Status_Global	OUTPUT	BYTE	Block status ■ 0x00: Not processed ■ 0x01: In process (BUSY) ■ 0x02: Ready without error (DONE) ■ 0x80: Error on processing (ERROR)
Status Alarm_Global	OUTPUT	BYTE	Corresponds to B3: Header byte 3 - *Common status* ■ Bit 0: Frequency *F_MAX* exceeded ■ Bit 1: Frequency *F_MIN* undershot ■ Bit 2: Temperature *T_MAX* exceeded ■ Bit 3: Voltage *VRMS_MAX* exceeded ■ Bit 4: Voltage *VRMS_MIN* undershot ■ Bit 5: Efficiency *PF_MIN* undershot ■ Bit 6: Current *IRMS_MAX* exceeded ■ Bit 7: reserved
Cmd	INPUT	BYTE	0: de-activated, 1: activated ■ Bit 0: Reset the energy counters ■ Bit 1: Trigger Reset at current transformer ■ Bit 2: Reset *status measurement* – If several bits are set, they are sequentially processed.
Status_Cmd	OUTPUT	BYTE	Status command ■ 0x00: Not processed ■ 0x01: In process (BUSY) ■ 0x02: Ready without error (DONE) ■ 0x80: Error on processing (ERROR)
Jobtime	OUTPUT	TIME	■ Duration to read the measured values respectively to run a command
DsID	OUTPUT	BYTE	Number of the current DS-ID
Frame_ID	OUTPUT	BYTE	Number of the current FR-ID
Error_ID	OUTPUT	WORD	Detailed error information
Reserved		ARRAY of BYTE (1...28)	reserved

UDT - data

After the header data, in the UDT there are the measurands sequentially listed with the following structure:

Name	Declaration	Data type	Description
Name	IN_OUT	STRUCT	■ Name of the measurand
Read_Mode	INPUT	BYTE	■ Bit 0: Accessing the measured value – 0: Measured value is not read – 1: Measured value is read
Value	OUTPUT	DWORD	■ Current measured value

ERROR IDs

ERROR ID	Description
0x0000	no error
0x8070	Error: Parameter MODE
0x8073	Error: Parameter CHANNEL_IN does not match MODE
0x8074	Error: Parameter CHANNEL_OUT does not match MODE
0x8080	Error: Write parameter: Data length is beyond 1 or 2 byte
0x8081	Error: Write parameter: Timeout detected when writing
0x8091	Error: Read measured value: Timeout detected when reading
0x80A1	Error: Telegram type not available - invalid request
0x80A2	Error: Frame not defined
0x80A3	Error: Measurand not available
0x80A4	Error: Telegram length
0x80A5	Error: Frame too big
0x80A6	Error: No new measured values available
0x80A7	Error: DS-ID
0x80A8	Error: "CMD Frame" - Command could not be executed
0x80AF	Internal error - Please contact the hotline! On an internal error (0x0F) all the measurements are stopped and a reset to the default parameters of the module is triggered! Here all counter values and Frame definitions are deleted!

10.3 Motion Modules

10.3.1 FB 320 - ACYC_RW - Acyclic access to the System SLIO motion module

Description

With this block you can access the object dictionary of the System SLIO motion modules by means of your user program. Here the block uses an acyclic communication channel based on a request/response sequence. This is part of the input/output area of motion module.

Due to the blocks FB 320 and FB 321 access the same data base, for each channel (if multichannel) you can use only one of these blocks in your user program! Also this block must be called per cycle only once!

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

Parameters

Parameter	Declaration	Data type	Description
REQUEST	IN	BOOL	The job is started with edge 0-1.
MODE	IN	BYTE	Enter 0x01 for the acyclic protocol
COMMAND	IN	BYTE	0x11 = Reading a data object (max. 4byte) 0x21 = Writing a data object (max. 4byte)
INDEX	IN	WORD	Index of the object
SUBINDEX	IN	BYTE	Subindex of the object
WRITE_LENGTH	IN	DINT	Length of the data to be written in byte (max. 4byte)
WRITE_DATA	IN	ANY	Pointer to the data to be written.
READ_DATA	IN	ANY	Pointer to the received data.
CHANNEL_IN	IN	ANY	Pointer to the beginning of the acyclic channel in the input area of the motion module. Enter as length 10bytes. Examples P#E100.0 BYTE 10 or P#DB10.DBX0.0 BYTE 10
CHANNEL_OUT	IN	ANY	Pointer to the beginning of the acyclic channel in the output area of the motion module. Enter as length 8bytes. Examples P#A100.0 BYTE 8 or P#DB10.DBX10.0 BYTE 8
READ_LENGTH	OUT	DInt	Length of the received data in byte. This value is to be rounded up to a multiple of 4, because the length specification is not transmitted.
DONE	OUT	BOOL	1: Job has been executed without error
BUSY	OUT	BOOL	0: There is no job being executed 1: Job is currently being executed
ERROR	OUT	BOOL	0: No Error 1: There is an error. The cause of the error is shown on the *ERROR_ID* parameter
ERROR_ID	OUT	WORD	Detailed error information

Please note that the parameters WRITE_DATA and READ_DATA are not checked for data type and length!

Behavior of the block parameters

- Exclusiveness of the outputs
  – The outputs *BUSY, DONE* and *ERROR* are mutually exclusive. There can only one of these outputs be TRUE at the same time.
  – As soon as the input *REQUEST* is TRUE, one of the outputs must be TRUE.
- Output status
  – The outputs *DONE, ERROR, ERROR_ID* and *READ_LENGTH* are reset by an edge 1-0 at the input *REQUEST*, when the function block is not active (*BUSY* = FALSE).
  – An edge 1-0 at *REQUEST* does not affect the job processing.
  – If *REQUEST* is already reset during job processing, so it is guaranteed that one of the outputs is set at the end of the command for a PLC cycle. Only then the outputs are reset.
- Input parameter
  – The input parameters are taken with edge 0-1 at *REQUEST*. To change parameters, you have to trigger the job again.
  – If there is again an edge 0-1 at *REQUEST* during the job processing, an error is reported, no new command is activated and the answer rejected by the current command!
- Error handling
  – The block has 2 error outputs for displaying errors during order processing. ERROR indicates the error and ERROR_ID shows an additional error number.
  – The outputs *DONE* and *READ_LENGTH* designates a successful command execution and are not set when *ERROR* becomes TRUE.
- Behavior of the *DONE* output
  – The *DONE* output is set, when a command was successfully executed.
- Behavior of the *BUSY* output
  – The *BUSY* output indicates that the function block is active.
  – Busy is immediately set with edge 0-1 of *REQUEST* and will not be reset until the job was completed successfully or failed.
  – As long as *BUSY* is TRUE, the function block must be called cyclically to execute the command.

If there is again an edge 0-1 at REQUEST during the job processing, an error is reported, no new command is activated and the answer rejected by the current command!

ERROR_ID

ERROR_ID	Description
0x0000	There is no Error
0x8070	Faulty parameter *MODE*
0x8071	Faulty parameter *COMMAND*
0x8072	Parameter *WRITE_LENGTH* exceeds the maximum size
0x8073	Parameter *CHANNEL_IN* does not fit the parameter *MODE*
0x8074	Parameter *CHANNEL_OUT* does not fit the parameter *MODE*
0x8075	Impermissible command (edge 0-1 at *REQUEST* during job is executed)
0x8081	Error - read access - data do not exist Command rejected!
0x8091	Error - write access - data do not exist Command rejected!
0x8092	Error - write access - data out of range Command rejected!
0x8093	Error - write access - data can only be read Command rejected!
0x8094	Error - write access - data are write protected Command rejected!
0x8099	Error during acyclic communication Command rejected!

Program code

If no job is active, all output parameters must be set to 0 (Command = IDLE). With an edge 0-1 at *REQUEST*, with the following approach a job is activated:

1. Check if a job is already active, if necessary terminate job and output error.
   ⇨ Wait until Status = IDLE
2. Check input parameters:
   - MODE
   - COMMAND
   - WRITE_LENGTH
   - CHANNEL_IN
   - CHANNEL_OUT

   ⇨ Terminate job on error, otherwise continue with step 3.
3. Save input parameters internally.

4. Execute the desired command and wait until this has been carried out.
5. Save and output the result of the command execution internally.
6. Set the command to IDLE again.

10.3.2 FB 321 - ACYC_DS - Acyclic parametrization System SLIO motion module

Description

With this block you can parametrize you motion module motion module by means of your user program. Here you can store your parameters as *Object list* in a data block an transfer them via the acyclic communication channel in your motion module

Due to the blocks FB 320 and FB 321 access the same data base, for each channel (if multichannel) you can use only one of these blocks in your user program! Also this block must be called per cycle only once!

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

Parameter

Parameter	Declaration	Data type	Description
REQUEST	IN	BOOL	The job is started with edge 0-1.
MODE	IN	BYTE	Enter 0x01 for the acyclic protocol.
READ_BACK	IN	BOOL	0: Written objects are not read back. 1: Written objects are read back immediately after the write operation and compared.
GROUP	IN	WORD	0x01...0x7F: Selection of a group in the object list. 0xFF: Section of all the objects in the object list.
OBJECT_DATA	IN	ANY	Pointer to the UDT. *Chapter 10.3.3 'UDT 321 - ACYC_OBJECT-DATA - Data structure for FB 321' on page 284*
CHANNEL_IN	IN	ANY	Pointer to the beginning of the input data of the *Acyclic channel* of the motion module.
CHANNEL_OUT	IN	ANY	Pointer to the beginning of the output data of the *Acyclic channel* of the motion module.
DONE	OUT	BOOL	1: Job has been executed without error.
BUSY	OUT	BOOL	0: There is no job being executed. 1: Job is currently being executed.
DATASET_INDEX	OUT	INT	Object that is currently being processed.

Parameter	Declaration	Data type	Description
ERROR	OUT	BOOL	0: No Error 1: There is an error. The cause of the error is shown on the *ERROR_ID* parameter.
ERROR_ID	OUT	WORD	Detailed error information

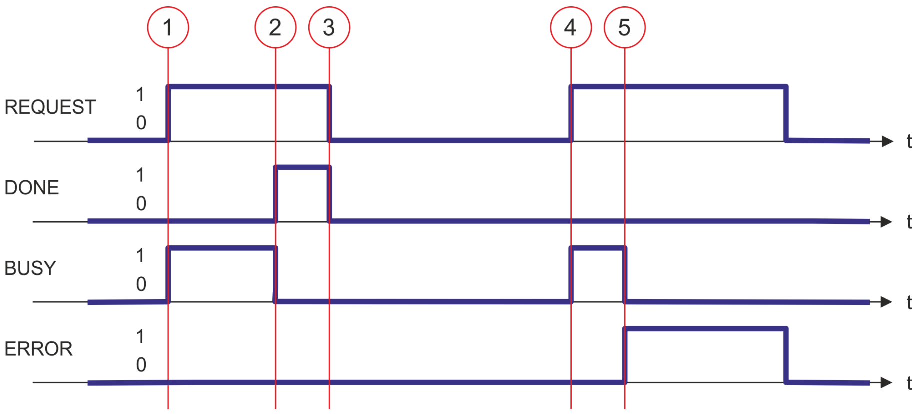
Behavior of the block parameters

- Exclusiveness of the outputs:
  – The outputs *BUSY*, *DONE* and *ERROR* are mutually exclusive. There can only one of these outputs be TRUE at the same time.
  – As soon as the input *REQUEST* is TRUE, one of the outputs must be TRUE.
- Output status
  – The outputs *DONE, ERROR, ERROR_ID* and *DATASET_INDEX* are reset by an edge 1-0 at the input *REQUEST*, when the job is finished.
  – If *REQUEST* is already reset during job processing, so it is guaranteed that the whole object list is processed.
  – At the end of the job with no error, *DONE* is set for one PLC cycle. Only then the outputs are reset.
- Input parameter
  – The input parameters are taken with edge 0-1 at *REQUEST*. To change parameters, you have to trigger the job again.
  – If there is again an edge 0-1 at *REQUEST* during the job, an error is reported (invalid command sequence) and the processing of the object list is finished.
- Input parameter *READ_BACK*
  – With activated parameter *READ_BACK* written objects are read back immediately after the write operation by a read job.
  – The written an read values are compared.
    If they are identical, the next object is handled
    If they are not identical, an error message (*ERROR ID* = 0x8079) is returned and the development of the object list is finished.
- Input parameter *GROUP*
  – For a better structure you can assign a group to each object.
  – Via *GROUP* you define the group whose parameters are to be transferred.
    0x01...0x7F: Transfer the objects of the selected group.
    0xFF: Transfer the objects of all the groups.
- Error handling
  – The block has error outputs to show errors during job processing. *ERROR* indicates the error, *ERROR_ID* shows an additional error number and *DATASET_INDEX* informs at which object the error occurred.
  – The output *DONE* designates a successful job execution and is not set when ERROR becomes TRUE.
- Behavior of the *DONE* output
  – The *DONE* output is set, when a command was successfully executed.

- Behavior of the *BUSY* output
  - The *BUSY* output indicates that the function block is active.
  - *BUSY* is immediately set with edge 0-1 of *REQUEST* and will not be reset until the job was completed successfully or failed.
  - As long as *BUSY* is TRUE, the function block must be called cyclically to execute the command.
- Behavior of the *DATASET_INDEX* output
  - The *DATASET_INDEX* output indicates, which object of the object list is currently being processed.
  - If there is no job active, *DATASET_INDEX* = 0 is returned.
  - If there is an error during the object processing, *DATASET_INDEX* shows the faulting object.

> *If there is again an edge 0-1 at REQUEST during the job processing, an error is reported (ERROR_ID = 0x8075), no new command is activated and the answer rejected by the current command!*

Status diagram



(1) The job is started with edge 0-1 at *REQUEST* and *BUSY* becomes TRUE.
(2) At the time (2) the job is completed. *BUSY* has the value FALSE and *DONE* den value TRUE.
(3) At the time (3) the job is completed and *REQUEST* becomes FALSE and thus each output parameter FALSE respectively 0.
(4) At the time (4) with an edge 0-1 at *REQUEST* the job is started again and *BUSY* becomes TRUE.
(5) At the time (5) an error occurs during the job. *BUSY* has the value FALSE and *ERROR* den value TRUE.

ERROR_ID

ERROR_ID	Description
0x0000	There is no Error
0x8070	Faulty parameter *MODE*
0x8071	Faulty parameter *OBJECT_DATA*
0x8075	Invalid command (edge 0-1 at *REQUEST* during job is executed)
0x8078	Faulty parameter *GROUP*

ERROR_ID	Description
0x8079	*READ_BACK* detects an error (written and read value unequal)
0x807A	Pointer at *OBJECT_DATA* not valid

Within the function block the FB 320 is called. Here, any error of the FB 320 is passed to the FB 321.
‘ERROR_ID’ on page 280

10.3.3 UDT 321 - ACYC_OBJECT-DATA - Data structure for FB 321

Data structure for the object list

The parameters are to be stored in a data block as *object list*, which consists of individual *objects*. The structure of an *objects* is defined via an UDT.

Structure of an object

Variable	Declaration	Data type	Description
Group	IN	WORD	0 < *Group* < 0x80 permitted
COMMAND	IN	BYTE	0x11 = Read from the object list 0x21 = Write to the object list
Index	IN	WORD	Index of the object
Subindex	IN	BYTE	Subindex of the object
Write_Length	IN	BYTE	Length of the data to be written in byte
Data_Write	IN	DWORD	Data to be written.
Data_Read	OUT	DWORD	Read data
State	OUT	BYTE	0x00 = never processed 0x01 = *BUSY* - in progress 0x02 = *DONE* - successfully processed 0x80 = *ERROR* - an error has occurred during the processing

Please note that you always specify the appropriate length for the object during a write job!

Example DB

Addr.	Name	Type	Start value	Current value	Comment
0.0	Object(1).Group	WORD			1. Object
2.0	Object(1).Command	BYTE			

Addr.	Name	Type	Start value	Current value	Comment
4.0	Object(1).Index	WORD			
6.0	Object(1).Subindex	BYTE			
7.0	Object(1).Write_Length	BYTE			
8.0	Object(1).Data_Write	DWORD			
12.0	Object(1).Data_Read	DWORD			
16.0	Object(1).State	BYTE			
18.0	Object(2).Group	WORD			2. Object
...	...	...			
34.0	Object(2).State	BYTE			
36.0	Object(3).Group	WORD			3. Object
...	...	...			
52.0	Object(3).State	BYTE			
...	...	...			...

10.4 WLD

10.4.1 FB 240 - RAM_to_s7prog.wld - RAM to s7prog.wld

Description

With *REQ* = TRUE this block copies the currently loaded project of a CPU on an inserted memory card as s7prog.wld. With a SPEED7 CPU from VIPA the s7prog.wld is automatically read from an inserted memory card always after an overall reset. The FB 240 internally calls the block SFB 239 with the corresponding parameters. Here the values of *BUSY* and *RET_VAL* are returned from the SFB 239 to the FB 240.

> ***VIPA specific block***
> *The VIPA specific blocks can be found in the VIPA library.* *Chapter 4 'Include VIPA library' on page 103*

Parameter

Name	Declara-tion	Data type	Memory area	Description
REQ	IN	BOOL	I, Q, M, D, L	Function request with *REQ* = 1
BUSY	OUT	BOOL	I, Q, M, D, L	Return value of the SFB 239
RET_VAL	OUT	WORD	I, Q, M, D, L	Return value of the SFB 239

10.4.2 FB 241 - RAM_to_autoload.wld - RAM to autoload.wld

Description

With *REQ* = TRUE this block copies the currently loaded project of a CPU on an inserted memory card as autoload.wld. With a SPEED7 CPU from VIPA the s7prog.wld is automatically read from an inserted memory card always after PowerON. The FB 241 internally calls the block SFB 239 with the corresponding parameters. Here the values of *BUSY* and *RET_VAL* are returned from the SFB 239 to the FB 241.

VIPA specific block
The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

Parameter

Name	Declara-tion	Data type	Memory area	Description
REQ	IN	BOOL	I, Q, M, D, L	Function request with *REQ* = 1
BUSY	OUT	BOOL	I, Q, M, D, L	Return value of the SFB 239
RET_VAL	OUT	WORD	I, Q, M, D, L	Return value of the SFB 239

10.5 Onboard I/O System 100V

10.5.1 SFC 223 - PWM - Pulse duration modulation

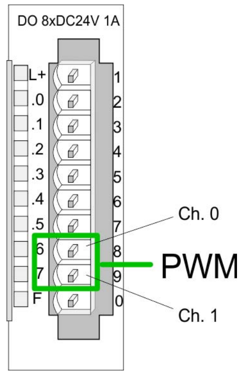
Description

This block serves the parameterization of the pulse duration modulation for the last two output channels of X5.

VIPA specific block
The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

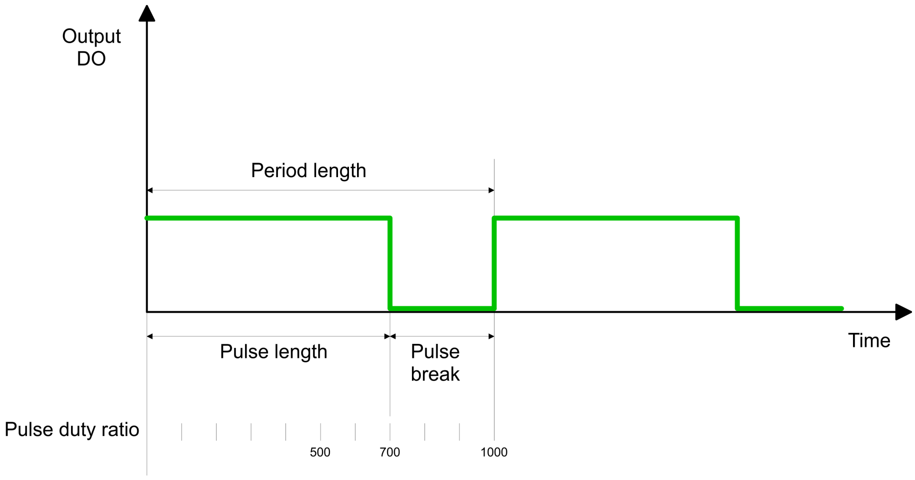
Parameters

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*
- IIf the calculated pulse duration is no multiplication of the *TIME-BASE*, 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 param-eter.
9002h	Internal hardware error - Please contact the hotline.
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!).

10.5.2 SFC 224 - HSC - High-speed-Counter

Description

This SFC serves for parameterization of the counter functions (high speed counter) for the first 4 inputs.

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 ‘Include VIPA library’ on page 103*

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

Value	Description
8009h	Parameter *CHANNEL* outside the permissible range
9001h	Internal error - There was no valid address for a parameter.
9002h	Internal hardware error - Please contact the hotline.

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

10.5.3 SFC 225 - HF_PWM - HF pulse duration modulation

Description

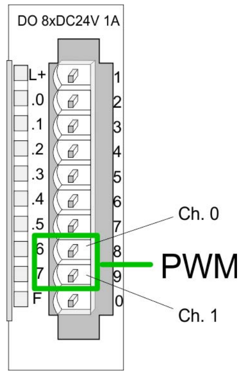
This block serves the parameterization of the pulse duration modulation for the last two output channels. This block is function identical to SFC 223. Instead of *TIMEBASE* and *PERIOD*, the SFC 225 works with a predefined frequency (up to 50kHz).

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

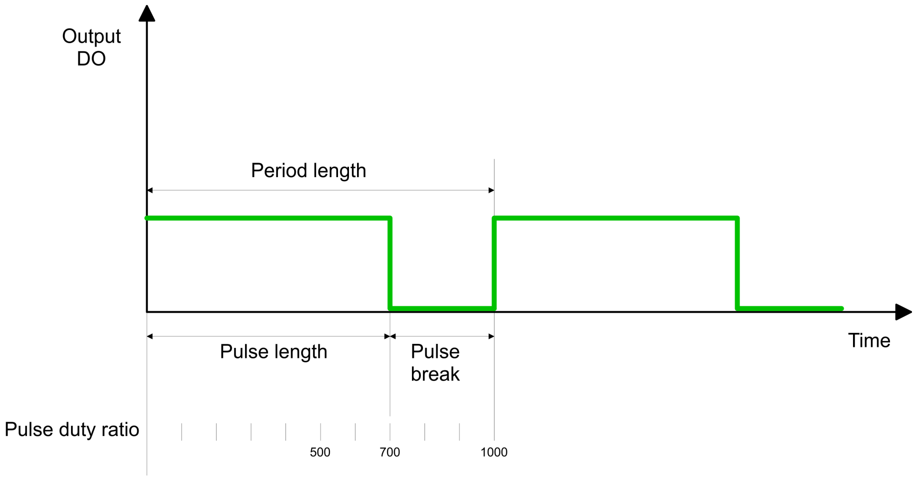
Parameters

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 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 = 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 *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 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 contact the hotline.
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!).

11 Integrated Standard

11.1 Standard Functions

11.1.1 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

11.1.2 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*.

11.1.3 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.

11.1.4 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.

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*.

11.1.5 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

11.1.6 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

11.1.7 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.

Parameter

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	Submodule 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

Submodule slot

(when submodules 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

11.1.8 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	Structure element PRORITY shows the priority class of the current OB.
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

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

TOP_SI Structure element	Data type	Logical Variable	Data type
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

11.1.9 SFC 7 - DP_PRAL - Triggering a hardware interrupt on the DP master

Description

With SFC 7 DP_PRAL you trigger a hardware interrupt on the DP master from the user program of an intelligent slave. This interrupt starts OB 40 on the DP master. Using the input parameter AL_INFO, you can identify the cause of the hardware interrupt. This interrupt identifier is transferred to the DP master and you can evaluate the identifier in OB 40 (variable OB40_POINT_ADDR). The requested hardware interrupt is uniquely specified by the input parameters *IOID* and *LADDR*. For each configured address area in the transfer memory, you can trigger exactly one hardware interrupt at any time.

How the SFC operates

SFC 7 DP_PRAL operates asynchronously, in other words, it is executed over several SFC calls. You start the hardware interrupt request by calling SFC 7 with *REQ* = 1. The status of the job is indicated by the output parameters *RET_VAL* and *BUSY*, see Meaning of the Parameters *REQ*, *RET_VAL* and *BUSY* with Asynchronous SFCs. The job is completed when execution of OB 40 is completed on the DP master.

If you operate the DP slave as a standard slave, the job is completed as soon as the diagnostic frame is obtained by the DP master.

Identifying a job

The input parameters *IOID* and *LADDR* uniquely specify the job. If you have called SFC 7 DP_PRAL on a DP slave and you call this SFC again before the master has acknowledged the requested hardware interrupt, the way in which the SFC reacts depends largely on whether the new call involves the same job: if the parameters *IOID* and *LADDR* match a job that is not yet completed, the SFC call is interpreted as a follow-on call regardless of the value of the parameter *AL_INFO*, and the value W#16#7002 is entered in *RET_VAL*.

Parameters

Parameter	Declaration	Data Type	Memory Area	Description
REQ	INPUT	BOOL	E, A, M, D, L, Constant	*REQ* = 1: Hardware interrupt on the DP master belonging to the slave
IOID	INPUT	BYTE	E, A, M, D, L, Constant	Identifier of the address area in the transfer memory (for the perspective of the DP slave): ■ B#16#00:Bit15 of *LADDR* specifies whether a an input (Bit15=0) or output address (Bit15=1) is involved. ■ B#16#54: Peripheral input (PI) ■ B#16#55: Peripheral output (PQ) If a mixed module is involved, the area identifier of the lower address must be specified. If the addresses are the same, B#16#54 must be specified.
LAADR	INPUT	WORD	E, A, M, D, L, Constant	Start address of the address range in the transfer memory (from the point of view of the DP slave). If this is a range belonging to a mixed module, specify the lower of the two addresses.

Parameter	Declaration	Data Type	Memory Area	Description
AL_INFO	INPUT	DWORD	E, A, M, D, L, Constant	Interrupt ID This is transferred to the OB40 that will be started on the DP master (variable OB40_POINT_ADDR). If you operate the intelligent slave with a remote master, you must evaluate the diagnostic frame on the master.
RET_VAL	OUTPUT	INT	E, A, M, D, L	If an error occurs while the function is being executed, the return value contains an error code.
BUSY	OUTPUT	BOOL	E, A, M, D, L	*BUSY* = 1: The triggered hardware interrupt has not yet been acknowledged by the DP master.

RET_VAL (Return value)

Value	Description
0000h	The job was executed without errors.
7000h	First call with *REQ* = 0. No hardware interrupt request is active; *BUSY* has the value 0.
7001h	First call with *REQ* = 1. A hardware interrupt request has already been sent to the DP master; *BUSY* has the value 1.
7002h	Interim call (*REQ* irrelevant): the triggered hardware interrupt has not yet been acknowledged by the DP master; *BUSY* has the value 1.
8090h	Start address of the address range in the transfer memory is incorrect.
8091h	Interrupt is blocked (block configured by user)
8093h	The parameters *IOID* and *LADDR* address a module that is not capable of a hardware interrupt request.
80B5h	Call in the DP master not permitted.
80C3h	The required resources (memory, etc.) are occupied at this time.
80C5h	Distributed I/O device is not available at this time (i.e. station failure).
80C8h	The function is not permitted in the current DP master operating mode.
8xyyh	General error information *Chapter 2.1 'General and Specific Error Information RET_VAL' on page 67*

11.1.10 SFC 12 - D_ACT_DP - 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.

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.

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.

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.

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.)

Value	Description
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.
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.

11.1.11 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.

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	return value
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 over-flow-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.

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.

> *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.

11.1.12 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 128Byte. 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

11.1.13 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 128Byte. 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.

Value	Description
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

11.1.14 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.

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

When generating messages with the SFC 17 and SFC 18, the operating system uses one system resource for the duration of the signal cycle.

For SFC 18 , the signal cycle lasts from the SFC call *SIG* = "1" until another call with SIG = "0". For SFC 17, this time period also includes the time until the incoming signal is acknowledged by one of the reported display devices, if necessary.

If, during the signal cycle, the message-generating block is overloaded or deleted, the associated system resource remains occupied until the next restart.

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 indicated 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 acknowledgment 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.

11.1.15 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.

11.1.16 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).

Interruptibility

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.

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)

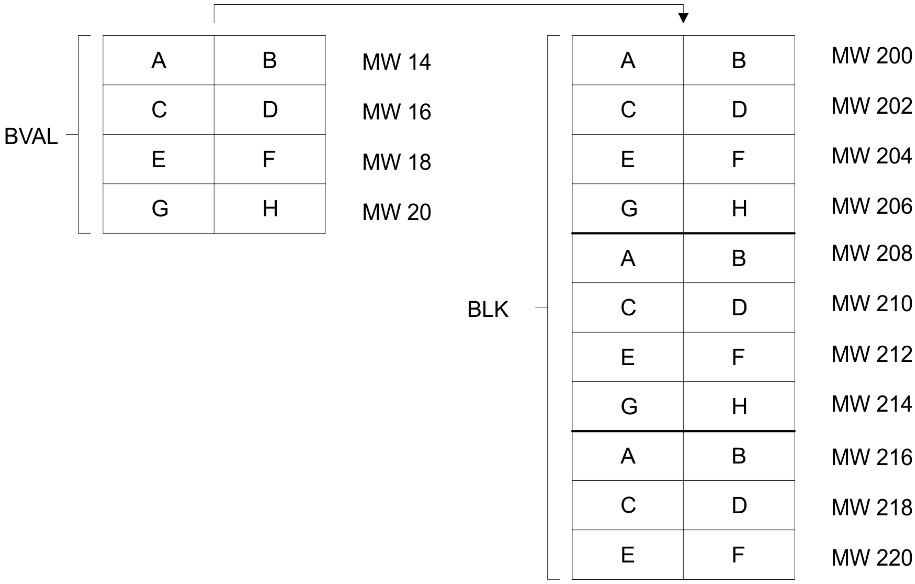
The return value contains an error code if an error is detected when the function is being processed.

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

11.1.17 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.



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 returns no specific error information.

11.1.18 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.

Interruptibility

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_NUMBER*

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

Value	Description
80B2h	Insufficient memory available
80B3h	Insufficient contiguous memory available (compress the memory!)

11.1.19 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.

Interruptibility

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.

Value	Description
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 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

11.1.20 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.

11.1.21 SFC 25 - COMPRESS - Compressing the User Memory

Gaps in Memory

Gaps can occur in the load memory and in the work memory if data blocks are deleted and reloaded several times. These gaps reduce the effective memory area.

Description

With SFC 25 COMPRESS, you start compression of the RAM section of both the load memory and the work memory. The compression function is the same as when started externally in the RUN mode (mode selector setting).

If compression was started externally and is still active (via Module Status Information), the SFC 25 call will result in an error message.

Parameters

Parameter	Declaration	Data type	Memory block	Description
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Error information
BUSY	OUTPUT	BOOL	I, Q, M, D, L	Indicates whether the compression function started by an SFC 25 call is still active. (1 means active)
DONE	OUTPUT	BOOL	I, Q, M, D, L	Indicates whether the compression function started by SFC 25 was completed successfully. (1 means completed successfully)

Checking the Compression Function

If SFC 25 COMPRESS is called once, the compression function is started.

Call SFC 25 cyclically. First evaluate the parameter *RET_VAL* after every call. Provided that its value is 0, the parameters *BUSY* and *DONE* can be evaluated. If *BUSY* = 1 and *DONE* = 0, this indicates that the compression function is still active. When *BUSY* changes to value 0 and *DONE* to the value 1, this indicates that the compression function was completed successfully.

If SFC 25 is called again afterwards, the compression function is started again.

11.1.22 SFC 28 ... SFC 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

11.1.22.1 SFC 28 - SET_TINT - Set time-of-day interrupt

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

Parameter	Declaration	Data type	Memory block	Description
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.

11.1.22.2 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.

11.1.22.3 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 (Rückgabewert)

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.

11.1.22.4 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

Param-eter	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.

11.1.23 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

11.1.24 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 can-celled (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.

11.1.25 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 can-celled (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.

11.1.26 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.

11.1.27 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.

11.1.28 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.

11.1.29 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*.

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 *Chapter 2.1 'General and Specific Error Information RET_VAL' on page 67*

11.1.30 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 *Chapter 2.1 'General and Specific Error Information RET_VAL' on page 67*

11.1.31 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).

11.1.32 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.

11.1.33 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.

11.1.34 SFC 44 - REPL_VAL - Replace value to ACCU1

Description

The SFC 44 REPL_VAL (replace value) transfers a value into ACCU1 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 ACCU1 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).

11.1.35 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.

11.1.36 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.

Interruptibility

The SFC 47 may be interrupted by high priority OBs.

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.

11.1.37 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 *AREA* below
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 *SUB-ADDR* 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	Decentralized 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*.

11.1.38 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*.
80A0h	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.

11.1.39 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*.

IInformation 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 setN_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).

11.1.40 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 *SSL_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

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 (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)

11.1.41 FC/SFC 53 - uS_Tick - Time measurement

This block allows you to read the µs ticker integrated in the SPEED7-CPU. The µs ticker is a 32bit µs time counter that starts at every reboot with 0 and counts to 2^{32-1}µs. At overflow the counter starts again with 0. With the help of the difference creation of the *RETVAL* results of 2 FC/SFC 53 calls before and after an application you may thus evaluate the runtime of the application in µs.

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

Runtime in dependence of the operating mode

Status	µs system time
Start-up	Starts with 0 and is permanently updated
RUN	is permanently updated
STOP	is stopped (time cannot be read)
Reboot	Starts again with 0

Parameters

Name	Declaration	Type	Comment
RETVAL	OUT	DINT	System time in µs

RETVAL

The parameter *RETVAL* contains the read system time in the range of 0 ... 2^{32}-1µs.

> *Please note for further calculations that the system time is returned in a signed data type.*

11.1.42 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)

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. 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).
  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.

Value	Description
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.

11.1.43 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.

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)
RECORD	INPUT	ANY	I, Q, M, D, L	Record

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.
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.

Value	Description
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.
80D5h	The record is static.

11.1.44 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.

Value	Description
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.

11.1.45 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.
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)
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.

Value	Description
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.

11.1.46 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!

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 range: 2 ... 240)
RECORD	INPUT	ANY	I, Q, M, D, L	Record Only data type BYTE is valid

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.
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

Value	Description
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. 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.

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.

11.1.47 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!

Parameters

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.
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 per-mitted.

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* indi-cates 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 acknowledgment 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.

Value	Description
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.

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.

11.1.48 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.

11.1.49 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.

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.
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.

Value	Description
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.

11.1.50 SFC 66 - X_RCV - Receive data

Description

The SFC 66 X_RCV 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.

Parameter	Declaration	Data type	Memory block	Description
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)
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 acknowledgment.
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 acknowledgment contains an unknown error code.
80BAh	The answer of the communication partner does not fit into the communication telegram.
80C0h	The answer of the communication partner does not fit into the communication telegram.

Value	Description
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.

11.1.51 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.

Parameter	Declaration	Data type	Memory block	Description
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" 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 started; *BUSY* has the value 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.

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 acknowledgment 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.

11.1.52 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.

Value	Description
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.

11.1.53 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.

11.1.54 SFC 70 - GEO_LOG - Determining the Start Address of a Module

Description

Assumption: the associated module slot of the module is known from the channel of a signal module. With SFC 70 GEO_LOG (convert geographical address to logical address) you can determine the associated start address of the module, that is, the smallest I address or Q address. If you use SFC 70 on power modules or modules with packed addresses, the diagnostic address is returned.

Parameters

Parameter	Declaration	Data Type	Memory Area	Description
MASTER	INPUT	INT	E, A, M, D, L, constant	Area ID: ■ 0, if the slot is located in one of the racks 0-3 (S7-300) or 0 bis 21 (S7-400) ■ 1 to 32: DP master system ID of the associated field device if the slot is located in a field device on PROFIBUS ■ 100 to 115: PROFINET IO system ID of the associated field device if the slot is located in a field device on PROFINET
STATION	INPUT	INT	E, A, M, D, L, constant	■ No. of rack, if area ID= 0 ■ Station number of field device if area ID > 0
SLOT	INPUT	INT	E, A, M, D, L, constant	Slot no.
SUBSLOT	INPUT	INT	E, A, M, D, L constant	Interface module slot (if no interface module can be inserted, enter 0 here)
RET_VAL	OUTPUT	INT	E, A, M, D, L	Error information
LADDR	OUTPUT	WORD	E, A, M, D, L	Start address of the module Bit 15 of *LADDR* indicates whether an input address (bit 15 = 0) or an output address (bit 15 = 1) is present

RET_VAL (Return value)

Value	Description
0000h	The job was executed without errors.
8094h	No subnet was configured with the specified *SUBNETID*.
8095h	Invalid value for *STATION* parameter

Value	Description
8096h	Invalid value for *SLOT* parameter
8097h	Invalid value for *SUBSLOT* parameter
8099h	The slot is not configured.
809Ah	The interface module address is not configured for the selected slot.
8xyyh	General error information *Chapter 2.1 'General and Specific Error Information RET_VAL' on page 67*

11.1.55 SFC 71 - LOG_GEO - Determining the slot belonging to a logical address

Description

SFC 71 LOG_GEO (convert logical address to geographical address) lets you determine the module slot belonging to a logical address as well as the offset in the user data area of the module.

Parameters

Parameter	Declaration	Data Type	Memory Area	Description
LADDR	INPUT	WORD	E, A, M, D, L, constant	Any logical address of the module In bit 15 you indicate whether an input address (bit 15 = 0) or an output address (bit 15 = 1) is present.
RET_VAL	OUTPUT	INT	E, A, M, D, L,	Error information
AREA	OUTPUT	INT	E, A, M, D, L,	Area ID: indicates how the remaining parameters are to be interpreted.
MASTER	OUTPUT	INT	E, A, M, D, L constant	Area ID: ■ 0, if the slot is located in one of the racks 0 - 3 (S7-300) or 0 - 21 (S7-400) ■ 1 to 32: DP master system ID of the associated field device if the slot is located in a field device on PROFIBUS ■ 100 to 115: PROFINET IO system ID of the associated field device if the slot is located in a field device on PROFINET
STATION	OUTPUT	INT	E, A, M, D, L	■ No. of rack, if area ID= 0 ■ Station number of field device if area ID > 0
SLOT	OUTPUT	INT	E, A, M, D, L	Slot no.
SUBSLOT	OUTPUT	INT	E, A, M, D, L	Interface module number
OFFSET	OUTPUT	INT	E, A, M, D, L	Offset in user data area of the associated module

AREA Output Parameter

Value of AREA	System	Meaning of RACK, SLOT and SUB-ADDR
0	S7-400	■ *MASTER*: 0 ■ *STATION*: Rack no. ■ *SLOT*: Slot no. ■ *SUBSLOT*: 0 ■ *OFFSET*: Difference between the logical address and the logical base address.
1	S7-300	■ *MASTER*: 0 ■ *STATION*: Rack no. ■ *SLOT*: Slot no. ■ *SUBSLOT*: 0 ■ *OFFSET*: Difference between the logical address and the logical base address.
2	PROFIBUS DP	■ *MASTER*: DP master system ID ■ *STATION*: Station number ■ *SLOT*: Slot no. in the station ■ *SUBSLOT*: 0 ■ *OFFSET*: Offset in user data address area of the associated module
	PROFINET IO	■ *MASTER*: PROFINET IO-System-ID ■ *STATION*: Station number ■ *SLOT*: Slot no. in the station ■ *SUBSLOT*: Submodulnummer ■ *OFFSET*: Offset in user data address area of the associated module
3	S5-P area	■ *MASTER*: 0 ■ *STATION*: Rack no. ■ *SLOT*: Slot no. of the adapter module ■ *SUBSLOT*: 0 ■ *OFFSET*: Address in the S5 x area
4	S5-Q area	■ *MASTER*: 0 ■ *STATION*: Rack no. ■ *SLOT*: Slot no. of the adapter module ■ *SUBSLOT*: 0 ■ *OFFSET*: Address in the S5 x area

Value of AREA	System	Meaning of RACK, SLOT and SUB-ADDR
5	S5-IM3 area	■ *MASTER*: 0 ■ *STATION*: Rack no. ■ *SLOT*: Slot no. of the adapter module ■ *OFFSET*: Address in the S5 x area
6	S5-IM4 area	■ *MASTER*: 0 ■ *STATION*: Rack no. ■ *SLOT*: Slot no. of the adapter module ■ *SUBSLOT*: 0 ■ *OFFSET*: Address in the S5 x area

RET_VAL (Return value)

Value	Description
0000h	The job was executed without errors.
8090h	Specified logical address invalid
8xyyh	General error information *Chapter 2.1 'General and Specific Error Information RET_VAL' on page 67*

11.1.56 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: FB, SFB, FC, SFC, 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.

Interruptibility

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.

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.

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.

11.1.57 SFC 101 - RTM - Handling Runtime meters

Description

Call SFC 101 RTM (runtime meter) to set, start, stop and read a 32-bit runtime meter of your CPU. To fetch the values of all 32-bit runtime meters of your CPU, call SFC 51 RDSYSST with SZL_ID=W#16#0132 and INDEX=W#16#000B (for runtime meters 0 … 7) or INDEX=W#16#000C (for runtime meters 8 … 15).

Parameters

Parameter	Deklaration	Datentyp	Speicher-bereich	Beschreibung
NR	INPUT	BYTE	E, A, M, D, L, Konstante	Number of the runtime meter Numbering starts at 0. You will find the number of runtime meters of your CPU in the technical specifications.
MODE	INPUT	BYTE	E, A, M, D, L, Konstante	Job ID: ■ 0: fetch (the status is then written to CQ and the current value to CV). After the runtime meter has reached (2E31) -1 hours, it stops at the highest value that can be displayed and outputs an "Overflow" error message. ■ 1: start (at the last counter value) ■ 2: stop ■ 4: set (to the value specified in *PV*) ■ 5: set (to the value specified in *PV*) and then start ■ 6: set (to the value specified in *PV*) and then stop
PV	INPUT	DINT	E, A, M, D, L, Konstante	New value for the runtime meter
RET_VAL	OUTPUT	INT	E, A, M, D, L	The return value will contain an error code if an error occurs while the function is being processed.
CQ	OUTPUT	BOOL	E, A, M, D, L	Status of the runtime meter (1: running)
CV	OUTPUT	DINT	E, A, M, D, L	Current value of the runtime meter

Compatibility to programs for a CPU with 16-bit runtime meters

You can also operate your 32-bit runtime meters with the SFCs 2 SET_RTM, SFC 3 CTRL_RTM and SFC 4 READ_RTM. In this case however, the 32-bit runtime meters operate in the same way as 16-bit meters (Range of values: 0 to 32767 hours). The partial list extract with SSL ID W#16#0132 and index W#16#0008 displays the 32-bit runtime meters 0 to 7 in 16-bit mode. This means that you can continue to use programs developed for a CPU with 16-bit runtime meters that use partial list extract with SSL ID W#16#0132 and index W#16#0008.

RET_VAL (Return value)

Error code	Description
0000h	The job was executed without errors.
8080h	Wrong runtime meter number
8081h	A negative value was passed to parameter *PV*.
8082h	Overflow of the runtime meter.

Error code	Description
8091h	Illegal value in input parameter *MODE*.
8xyyh	General error information *Chapter 2.1 'General and Specific Error Information RET_VAL' on page 67*

11.1.58 SFC 102 - RD_DPARA - Reading Predefined Parameters

Description

With SFC 102 RD_DPARA you can read the record set with the number *RECNUM* of a selected module from system data configured with STEP7. The read record set is entered into the target area opened with the parameter *RECORD*.

Operating principle

The SFC 102 RD_DPARA operates asynchronously, that is, processing covers multiple SFC calls.

Start the job by calling SFC 102 with REQ = 1. The job status is displayed via the output parameters *RET_VAL* and *BUSY*. Refer also to Meaning of *REQ*, *RET_VAL* and *BUSY* with Asynchronously Operating SFCs.

Parameters

Parameter	Declaration	Data type	Memory block	Description
REQ	INPUT	BOOL	I, Q, M, D, L	*REQ* = 1: Read request
LADDR	INPUT	WORD	I, Q, M, D, L, constant	Address of the module. For an output address, the highest value bit must be set.
RECNUM	INPUT	BYTE	I, Q, M, D, L, constant	Record set number (permitted values: 0 ... 240).
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. If no error occurred during the transmission, the following two cases are distinguished: ■ *RET_VAL* contains the length of the actually read record set in bytes if the destination area is larger than the read record set. ■ *RET_VAL* contains 0 if the length of the read record set is equal to the length of the destination area.

Parameter	Declaration	Data type	Memory block	Description
BUSY	OUTPUT	BOOL	I, Q, M, D, L	*BUSY* = 1: The job is not yet closed.
RECORD	OUTPUT	ANY	I, Q, M, D, L	Destination area for the read record set. Only data type BYTE is permitted. Note: Note that the *RECORD*parameter of CPUs always required the full specification of the DB parameters. (for example: P#DB13.DBX0.0 byte 100). Omitting an explicit DB number is not permitted for CPUs and causes an error message in the user program.

Error Information

Chapter 11.1.45 'SFC 57 - PARM_MOD - Parameterize module' on page 351

11.1.59 SFC 105 - READ_SI - Reading Dynamic System Resources

Overview

When messages are generated with SFCs 107 ALARM_DQ and 108 ALARM_D, the operating system occupies temporarily system memory space. For example, if you do not delete a FB that exists in the CPU with SFC 107 or SFC 108 calls it may happen that corresponding system resources stay permanently occupied.

If you reload the FB with SFC 108 or SFC 108 calls, it may happen that the SFCs 107 and 108 are not processed properly anymore.

Description

With SFC 105 READ_SI you can read currently used system resources occupied with the SFCs 107 and 108 when messages were generated.

This is done via the values of *EV_ID* and *CMP_ID* used in this place. The values are passed on to SFC 105 READ_SI in parameter *SI_ID*.

SFC 105 READ_SI has four possible operating modes that we explain in the table below. Set the desired operating mode via the*MODE* parameter.

MODE	Which of the system resources occupied by SFC 107/SFC 108 are read?
1	All (call of SFC 105 with *SI_ID*: =0)
2	The system resource occupied by the call of SFC 107-/SFC 108 with *EV_ID*:= ev_id (call of the SFC 105 with *SI_ID*: = ev_id)

MODE	Which of the system resources occupied by SFC 107/SFC 108 are read?
3	The system resource occupied by the call of SFC 107-/SFC 108 with *CMP_ID*: = cmp_id (call of the SFC 105 with *SI_ID*: = ev_id)
0	Additional system resources that could not be read with the previous call in *MODE* =1 or *MODE* =3 because you have specified a target field *SYS_INST* that is too small.

Operating principle

If you have not selected a sufficiently large *SYS_INST* target area when you called the SFC 105 in *MODE* =1 or *MODE* =3, it contains the content of all currently occupied system resources selected via *MODE* parameter.

High system load on resources will cause a correspondingly high SFC runtime. That is, a high load on CPU performance may result in overshoot of the maximum configurable cycle monitoring time. You can work around this runtime problem as follows: Select a relatively small *SYS_INST* target area.

RET_VAL = 0001h informs you if the SFC cannot enter all system resources to be read in *SYS_INST*. In this case, call SFC 105 with *MODE* =0 and the same *SI_ID* as for the previous call until the value of *RET_VAL* is 0000h.

Since the operating system does not coordinate the SFC 105 calls that belong to the read job, you should execute all SFC 105 calls with the same priority class.

Target Area SYS_INST

The target area for the fetched occupied system resource must lie within a DB. You should appropriately define the target area as a field of structures, whereby a structure is constructed as follows:

Structure element	Data type	Description
SFC_NO	WORD	No. of the SFC that occupies the system resource
LEN	BYTE	Length of the structures in bytes, incl. *SFC_NO* and *LEN*: 0Ch
SIG_STAT	BOOL	Signal state
ACK_STAT	BOOL	Acknowledgement status of the incoming event (positive edge)
EV_ID	DWORD	Message number
CMP_ID	DWORD	Partial system ID

Parameters

Parameter	Declaration	Data type	Memory Area	Description
MODE	INPUT	INT	I, Q, M, D, L, constant	Job identifier Permissible values: ■ 1: Read all system resources ■ 2: Read the system resource that was occupied with *EV_ID* = ev_id when SFC 107-/SFC 108 was called ■ 3: Read the system resources that were occupied with *CMP_ID* = cmp_id when SFC 107-/SFC 108 was called ■ 0: subsequent call
SI_ID	INPUT	DWORD	I, Q, M, D, L, constant	ID for the system resource(s) to be read Permissible values ■ 0, if *MODE* = 1 ■ Message number ev_id, if *MODE* = 2 ■ ID cmp_id for identification of the system section, if *MODE* = 3
RET_VAL	OUTPUT	INT	I, Q, M, D, L,	Return value (error information or job status)
N_SI	OUTPUT	INT	I, Q, M, D, L,	Number of output system resources with *SYS_INT*
SYS_INST	OUTPUT	ANY	D	Target area for the fetched system resources.

RET_VAL (Return value)

Error code	Description
0000h	No error occurred.
0001h	Not all system resources could be read because the *SYS_INT* target range you have selected is too short.
8081h	(only with *MODE* =2 or 3) You have assigned the value 0 to *SI_ID*.
8082h	only with *MODE* =1) You have assigned one of 0 different values to *SI_ID*.
8083h	(only with *MODE* =0) You have assigned *SI_ID* a value other than at the preceding call of the SFC with *MODE* =1 or 3.
8084h	You have assigned an illegal value to *MODE*.
8085h	SFC 105 is already being processed in another OB.

Error code	Description
8086h	Target area *SYS_INST* too small for a system resource.
8087h or 8092h	Target area *SYS_INST* does not exist in a DB or error in the ANY pointer.
8xyyh	General error information *Chapter 2.1 'General and Specific Error Information RET_VAL' on page 67*

11.1.60 SFC 106 - DEL_SI - Reading Dynamic System Resources

Overview

When messages are generated with SFCs 107 ALARM_DQ and 108 ALARM_D, the operating system occupies temporarily system memory space.

For example, if you do not delete a FB that exists in the CPU with SFC 107 or SFC 108 calls it may happen that corresponding system resources stay permanently occupied. If you reload the FB with SFC 108 or SFC 108 calls, it may happen that the SFCs 107 and 108 are not processed properly anymore.

Description

With SFC 106 DEL_SI you can delete currently used system resources.

SFC 106 DEL_SI has three possible operating modes explained in the table below. Set the desired operating mode via the *MODE* parameter.

MODE	Which of the system resources occupied by SFC 107/SFC 108 are deleted?
1	All (call of SFC 106 with *SI_ID*: = 0)
2	The system resource occupied by the call of SFC 107-/SFC 108 with *EV_ID*: = ev_id (call of the SFC 106 with *SI_ID*: = ev_id)
3	The system resource occupied by the call of SFC 107-/SFC 108 with *CMP_ID*:= cmp_id (call of the SFC 106 with *SI_ID*: =e v_id)

Parameters

Parameter	Declara-tion	Data type	Memory Area	Description
MODE	INPUT	INT	I, Q, M, D, L, constant	Job identifier Permissible values ■ 1: delete all system resources ■ 2: delete the system resource that was occupied with *EV_ID* = ev_id when SFC 107-/SFC 108 was called ■ 3: delete the system resources that were occupied with *CMP_ID* = cmp_id when SFC 107-/SFC 108 was called
SI_ID	INPUT	DWORD	I, Q, M, D, L, constant	ID of the system resource(s) to be deleted Permissible values ■ 0, if *MODE* = 1 ■ Message number ev_id, if *MODE* = 2 ■ ID cmp_id for identification of the system section, if *MODE* = 3
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Error Information

RET_VAL (Return value)

Error code	Description
0000h	No error occurred.
8081h	(only with *MODE* = 2 or 3) You have assigned the value 0 to *SI_ID*.
8082h	(only with *MODE* = 1) You have assigned one of 0 different values to *SI_ID*.
8084h	You have assigned an illegal value to MODE.
8085h	SFC 106 is currently being processed.
8086h	Not all selected system resources could be deleted because at least one of them was being processed when SFC 106 was called.
8xyyh	General error information *Chapter 2.1 'General and Specific Error Information RET_VAL' on page 67*

11.1.61 SFC 107 - ALARM_DQ and SFC 108 - ALARM_D

Description

With every call the SFCs 107 ALARM_DQ (Generating Acknowledgeable Block Related Messages) and 108 ALARM_D (Permanently Acknowledged Block Related Messages) generate a message to which you can append an associated value. Thus, you correspond with SFCs 17 ALARM_SQ and 18 ALARM_S.

When generating messages with SFCs 107 ALARM_DQ and 108 ALARM_D, the operating system temporarily occupies a system resource for the duration of the signal cycle.

The signal cycle time for SFC 108 ALARM_D starts at the SFC call with *SIG* = 1 and ends at a new call with *SIG* = 0. This interval for SFC 107 ALARM_DQ may be extended by the time expiring until the incoming signal is acknowledged at a logged in displaying device.

For SFC 108 ALARM_D, the signal cycle lasts from the SFC call *SIG* = 1 until another call with *SIG* = 0. For SFC 107 ALARM_DQ, this time period also includes the time until the incoming signal is acknowledged by one of the reported display devices, if necessary.

If, during the signal cycle, the message-generating block is over-loaded or deleted, the associated system resource remains occupied until the next restart.

The additional functionality of SFCs 107 ALARM_DQ and 108 ALARM_D compared to SFCs 17 and 18 is now that you can manage these occupied system resources:

- With the help of SFC 105 READ_SI you can fetch information related to occupied system resources.
- With SFC 106 DEL_SI you can release occupied system resources again. This is of special significance for permanently occupied system resources. A currently occupied system resource, for example, stays occupied until the next restart if you, in the course of a program change, delete an FB call that contains SFC 107 or SFC 108 calls. When you change the program, and reload an FB with SFC 107 or SFC 108 calls, it may happen that the SFCs 107 and 108 do not generate anymore messages.

Description Parameter

The SFCs 107 and 108 contain one parameter more than the SFCs 17 and 18, namely the input *CMP_ID*. Use this input to assign the messages generated with SFCs 107 and 108 to logical areas, for example to parts of the system. If you call SFC 107/SFC 108 in an FB the obvious thing to do is to assign the number of the corresponding instance DB to *CMP_ID*.

Parameters

Parameter	Declaration	Data type	Memory block	Description
SIG	INPUT	BOOL	I, Q, M, D, L	The message triggering signal
ID	INPUT	WORD	I, Q, M, D, L, constant	Data channel for messages: EEEEh
EV_ID	INPUT	DWORD	I, Q, M, D, L, constant	Message number (not allowed: 0)

Parameter	Declaration	Data type	Memory block	Description
CMP_ID	INPUT	DWORD	I, Q, M, D, L, constant	Component identifier (not allowed: 0) ID for the partial system to which the-corresponding message is assigned Recommended values: ■ Low-Word: 1 ... 65535 ■ High-Word: 0 You will not be confronted with any con-flicts if you are compliant with these rec-ommendations.
SD	INPUT	ANY	I, Q, M, D, T, C	Associated value Maximum length: 12 bytes Permitted are only data of the type BOOL (not allowed: Bit field),BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Error Information

RET_VAL (Return value)

Error code	Description
0000h	No error occurred.
0001h	■ The length of the associated value exceeds the maximum permissible length, or ■ Access to user memory not possible (for example, access to deleted DB).The activated message is sent. ■ The associated value points to a value in the local data area. The message is sent. (S7-400 only)
0002h	Warning: The last free message acknowledge memory was occupied. (S7-400 only)
8081h	The specified *EV_ID* lies outside the valid range.
8082h	Message loss because your CPU has no more resource for generating block related messages with SFCs.
8083h	Message loss, the same signal transition is already present but could not be sent yet (signal overflow).
8084h	With the current and the previous SFC 107-/SFC-108 call the message triggering signal SIG has the same value.
8085h	There is no logon for the specified *EV_ID*.
8086h	An SFC call for the specified *EV_ID* is already being processed in a lower priority class.
8087h	At the initial call of SFC 107/SFC 108 the message triggering signal had the value 0.
8088h	The specified *EV_ID* is already in use by another system resource (to SFC 17, 18, 107, 108).

Error code	Description
8089h	You have assigned the value 0 to *CMP_ID*.
808Ah	*CMP_ID* not fit to *EV_ID*
8xyyh	General error information *Chapter 2.1 ‘General and Specific Error Information RET_VAL’ on page 67*

11.2 Standard Function Blocks

11.2.1 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
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

11.2.2 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:

- "1", if $CV \leq 0$
- else *Q* = "0"

CV

Current count:

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

11.2.3 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 $CV \geq PV$.

- 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 up counter:

- *QU* = "1" if *CV* ≥ *PV* (Current count ≥ Preset value)
- else *QU* = "0"

QD

Status of the down counter:

- *QD* is set to "1", if 0 ≥ CV (Current count smaller/= 0)
- else *QU* = "0"

CV

Current count

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

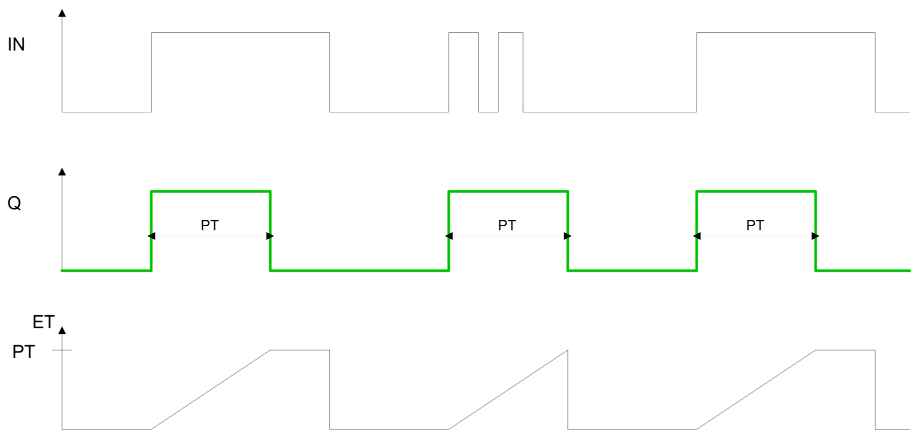
11.2.4 SFB 3 - TP - Create pulse

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.

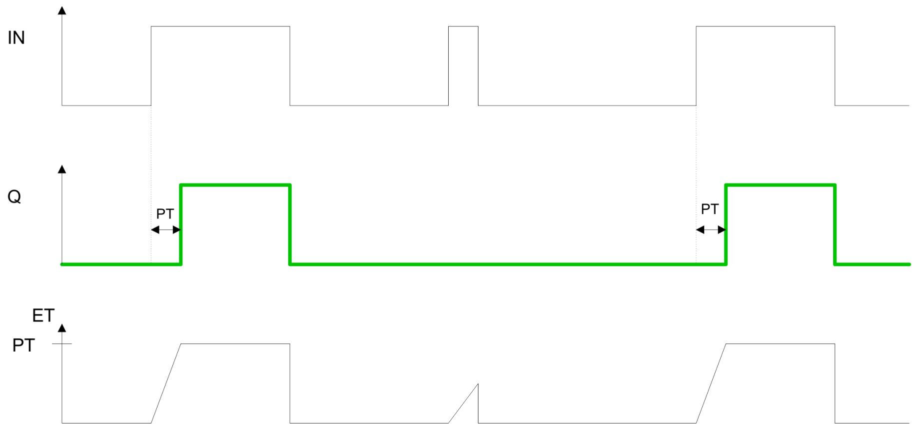
11.2.5 SFB 4 - TON - Create turn-on delay

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* to *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".

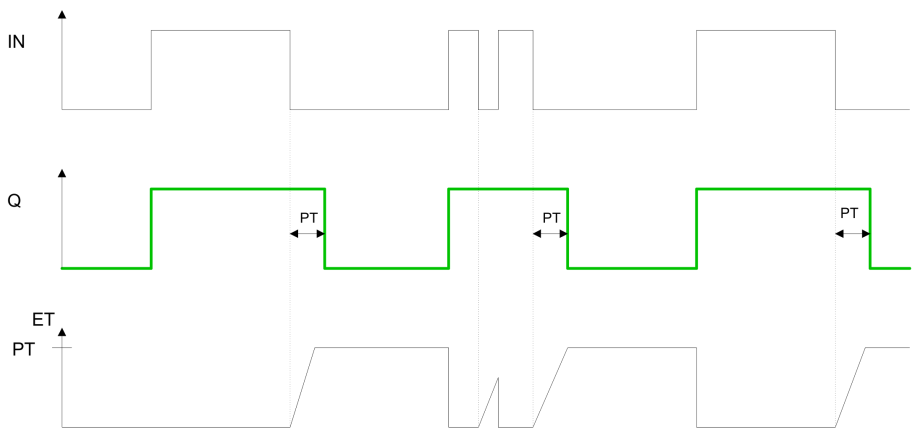
11.2.6 SFB 5 - TOF - Create turn-off delay

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* to *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* 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".

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

Description

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

Depending upon communication function the following behavior is present:

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

Function

- If there is a rising edge at control input *R*, the current data transfer is cancelled.
- Successful completion of the transfer is indicated by the status parameter *DONE* having the value 1.

- A new send job cannot be processed until the previous send process has been completed if the status parameter *DONE* or *ERROR* have the value 1.
- Due to the asynchronous data transmission, a new transmission can only be initiated if the previous data have been retrieved by the call of the partner FB/SFB. Until the data are retrieved, the status value 7 will be given when the FB/SFB BSEND is called.

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

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

Parameters

Parameter	Declaration	Data type	Memory block	Description
REQ	INPUT	BOOL	I, Q, M, D, L	Control parameter request, a rising edge activates the data exchange (with respect to the most recent FB/SFB call)
R	INPUT	BOOL	I, Q, M, D, L, constant	control parameter reset: terminates the active task
ID	INPUT	WORD	I, Q, M, D, constant	A reference for the connection. Format W#16#xxxx
R_ID	INPUT	DWORD	I, Q, M, D, L, constant	Address parameter *R_ID*. Format DW#16#wxyzWXYZ.
DONE	OUTPUT	BOOL	I, Q, M, D, L	Status parameter *DONE*: 0: task has not been started or is still being executed. 1: task was executed without error.
ERROR	OUTPUT	BOOL	I, Q, M, D, L	Status parameter *ERROR*: ■ *ERROR* = 0 + *STATUS* = 0000h – No warnings or errors. ■ *ERROR* = 0 + *STATUS* unequal to 0000h – A Warning has occurred. *STATUS* contains detailed information. ■ *ERROR* = 1 – An error has occurred.

Parameter	Declaration	Data type	Memory block	Description
STATUS	OUTPUT	WORD	I, Q, M, D, L	Status parameter *STATUS*, returns detailed information about the type of error.
SD_1	IN_OUT	ANY	I, Q, M, D, T, C	Pointer to the send data buffer. The length parameter is only utilized when the block is called for the first time after a start. It specifies the maximum length of the send buffer. Only data type BOOL is valid (Bit field not permitted), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER.
LEN	IN_OUT	WORD	I, Q, M, D, L	The length of the send data block in bytes.

Error information

ERROR	STATUS (decimal)	Description
0	11	Warning: the new task is not active since the previous task has not completed.
0	25	The communication process was initiated. The task is being processed.
1	1	Communication failures, e.g.: ■ Connection parameters not loaded (local or remote) ■ Connection interrupted (e.g. cable, CPU turned off, CP in STOP)
1	2	Negative acknowledgment received from the partner FB/SFB. The function cannot be executed.
1	3	*R_ID* is not available to the communication link specified by ID or the receive block has never been called.
1	4	Error in send buffer pointer *SD_1* with respect to the length or the data type, or parameter *LEN* was set to 0 or an error has occurred in the receive data buffer pointer *RD_1* of the respective FB/SFB 13 BRCV
1	5	Reset request was executed.
1	6	The status of the partner FB/SFB is DISABLED (*EN_R* has a value of 0)
1	7	The status of the partner FB/SFB is not correct (the receive block has not been called after the most recent data transfer).
1	8	Access to the remote object in application memory was rejected.
1	10	Access to local application memory not possible (e.g. access to deleted DB).

ERROR	STATUS (decimal)	Description
1	12	The call to the FB/SFB ■ contains an instance DB that does not belong to the FB/SFB 12 ■ contains a global DB instead of an instance DB ■ could not locate an instance DB (load a new instance DB from the PG)
1	18	*R_ID* already exists in the connection ID.
1	20	Not enough memory.

Data consistency

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

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

Description

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

Depending upon communication function the following behavior is present:

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

> ***VIPA specific block***
> *The VIPA specific blocks can be found in the VIPA library.* *Chapter 4 'Include VIPA library' on page 103*

Parameters

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

Function

- The FB/SFB 13 is ready for reception when control input *EN_R* is set to 1. Parameter *RD_1* specifies the start address of the receive data buffer. An acknowledgment is returned to the partner FB/SFB after reception of each data segment and parameter *LEN* of the FB/SFB 13 is updated accordingly. If the block is called during the asynchronous reception process a warning is issued via the status parameter *STATUS*.
- Should this call be received with control input *EN_R* set to 0 then the receive process is terminated and the FB/SFB is reset to its initial state. When all data segments have been received without error parameter *NDR* is set to 1. The received data remains unaltered until FB/SFB 13 is called again with parameter *EN_R* = 1.

Error information

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

ERROR	STATUS (decimal)	Description
1	18	*R_ID* already exists in the connection *ID*.
1	20	Not enough memory.

Data consistency

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

- When copying has been completed (parameter *NDR* is set to 1) FB/SFB 13 must again be called with parameter *EN_R* set to 0 in order to ensure that the receive data block is not overwritten before it has bee evaluated.
- The most recently used receive data block *RD_1* must have been evaluated completely before the block is denoted as being ready to receive (calls with parameter *EN_R* set to 1).

Receiving Data S7-400

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

11.2.9 FB/SFB 14 - GET - Remote CPU read

Description

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

Depending upon communication function the following behavior is present:

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

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

Parameters

Parameter	Declaration	Data type	Memory block	Description
REQ	INPUT	BOOL	I, Q, M, D, L	control parameter request, a rising edge activates the data exchange (with respect to the most recent FB/SFB-call)
ID	INPUT	WORD	I, Q, M, D, constant	A reference for the connection. Format: W#16#xxxx
NDR	OUTPUT	BOOL	I, Q, M, D, L	Status parameter *NDR*: data from partner CPU has been accepted.
ERROR	OUTPUT	BOOL	I, Q, M, D, L	Status parameter *ERROR*: ■ *ERROR* = 0 + *STATUS* = 0000h – No warnings or errors. ■ *ERROR* = 0 + *STATUS* unequal to 0000h – A Warning has occurred. *STATUS* contains detailed information. ■ *ERROR* = 1 – An error has occurred.
STATUS	OUTPUT	WORD	I, Q, M, D, L	Status parameter *STATUS*, returns detailed information about the type of error.
ADDR_1	IN_OUT	ANY	e.g. I, Q, M, D	Pointer indicating the buffers in the partner CPU that must be read
ADDR_2	IN_OUT	ANY	e.g. I, Q, M, D	Pointer indicating the buffers in the partner CPU that must be read
ADDR_3	IN_OUT	ANY	e.g. I, Q, M, D	Pointer indicating the buffers in the partner CPU that must be read
ADDR_4	IN_OUT	ANY	e.g. I, Q, M, D	Pointer indicating the buffers in the partner CPU that must be read
RD_i,1≤ I ≤4	IN_OUT	ANY	I, Q, M, D, T, C	Pointers to the area of the local CPU in which the read data are entered. Only data type BOOL is valid (bit field not permitted), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER.

Function

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

Error information

ERROR	STATUS (decimal)	Description
0	11	Warning: the new task is not active since the previous task has not completed.
0	25	The communication process was initiated. The task is being processed.
1	1	Communication failures, e.g. ■ Connection parameters not loaded (local or remote) ■ Connection interrupted (e.g.: cable, CPU turned off, CP in STOP)
1	2	Negative acknowledgment from partner device. The function cannot be executed.
1	4	Error in receive data buffer pointer *RD_i* with respect to the length or the data type.
1	8	Partner CPU access error
1	10	Access to local application memory not possible (e.g. access to deleted DB).
1	12	The call to the FB/SFB ■ contains an instance DB that does not belong to the FB/SFB 14 ■ contains a global DB instead of an instance DB ■ could not locate an instance DB (load a new instance DB from the PG)
1	20	Not enough memory.

Data consistency

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

11.2.10 FB/SFB 15 - PUT - Remote CPU write

Description

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

Depending upon communication function the following behavior is present:

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

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

Parameters

Parameter	Declaration	Data type	Memory block	Description
REQ	INPUT	BOOL	I, Q, M, D, L	control parameter request, a rising edge activates the data exchange (with respect to the most recent FB/SFB-call)
ID	INPUT	WORD	I, Q, M, D, constant	A reference for the connection. Format W#16#xxxx
DONE	OUTPUT	BOOL	I, Q, M, D, L	Status parameter *DONE*: function completed.
ERROR	OUTPUT	BOOL	I, Q, M, D, L	Status parameter *ERROR*: ■ *ERROR* = 0 + *STATUS* = 0000h – No warnings or errors. ■ *ERROR* = 0 + *STATUS* unequal to 0000h – A Warning has occurred. *STATUS* contains detailed information. ■ *ERROR* = 1 – An error has occurred.
STATUS	OUTPUT	WORD	I, Q, M, D, L	Status parameter *STATUS*, returns detailed information about the type of error.
ADDR_1	IN_OUT	ANY	e.g. I, Q, M, D	Pointer indicating the buffers in the partner CPU into which data is written

Parameter	Declaration	Data type	Memory block	Description
ADDR_2	IN_OUT	ANY	e.g. I, Q, M, D	Pointer indicating the buffers in the partner CPU into which data is written
ADDR_3	IN_OUT	ANY	e.g. I, Q, M, D	Pointer indicating the buffers in the partner CPU into which data is written
ADDR_4	IN_OUT	ANY	e.g. I, Q, M, D	Pointer indicating the buffers in the partner CPU into which data is written
SD_i,1≤I ≤4	IN_OUT	ANY	I, Q, M, D, T, C	Pointer to the data buffers in the local CPU that contains the data that must be sent. Only data type BOOL is valid (Bit field not permitted), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER.

Function

- The partner CPU stores the data at the respective address and returns an acknowledgment.
- This acknowledgment is tested and when an error is detected in the data transfer parameter *DONE* is set to 1 with the next call of FB/SFB 15.
- The write process can only be activated again when the most recent write process has been completed. The amount, length and data type of the buffer areas that were defined by means of parameters *ADDR_i* and *SD_i*, 1 ≤ I ≤ 4 must be identical.

Error information

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

ERROR	STATUS (decimal)	Description
1	12	The call to the FB/SFB contains an instance DB that does not belong to the FB/SFB 15. contains a global DB instead of an instance DB. could not locate an instance DB (load a new instance DB from the PG).
1	20	Not enough memory.

Data consistency

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

11.2.11 SFB 31 - NOTIFY_8P - Messages without acknowledge display (8x)

Description

Generating block related messages without acknowledgement display for 8 signals.

- SFB 31 NOTIFY_8P represents an extension of SFB 36 "NOTIFY" to 8 signals.
- A message is generated if at least one signal transition has been detected. A message is always generated at the initial call of SFB 31. All 8 signal are allocated a common message number that is split into 8 sub-messages on the displaying device.
- One memory with 2 memory blocks is available for each instance of SFB 31 NOTIFY_8P.
- The displaying device shows the last two signal transitions, irrespective of message loss.

Before you call SFB 31 NOTIFY_8P in a automation system, you must insure that all connected displaying devices know this block. More information about this may be found in the manuals of the components used.

Parameter

Parameter	Declaration	Data type	Memory block	Description
SIG_i,	INPUT	BOOL	I, Q, M, D, L	i-th signal to be monitored
ID	INPUT	WORD	Constant (I, Q, M, D, L)	Data channel for messages: EEEEh. ID is only evaluated at the first call.
EV_ID	INPUT	DWORD	Constant (I, Q, M, D, L)	Message number (not permitted: 0)
SEVERITY	INPUT	WORD	Constant (I, Q, M, D, L)	Weighting of the event
DONE	OUTPUT	BOOL	I, Q, M, D, L	Status parameter DONE: Message generation completed.
ERROR	OUTPUT	BOOL	I, Q, M, D, L	Status parameter ERROR
STATUS	OUTPUT	WORD	I, Q, M, D, L	Status parameter STATUS: Display of an error information
SD_i	IN_OUT	ANY	I, Q, M, D, T, C	i-th associated value

SIG_i,

i-th signal to be monitored It is valid 1 ≤ I ≤ 8.

ID

Data channel for messages: EEEEh. *ID* is only evaluated at the first call.

EV_ID

EV_ID is only evaluated at the first call. Subsequently, the message number used for the first call applies to every call of SFB with the corresponding instance DB. The message number is automatically assigned by the Siemens STEP®7 programming tool. So the consistency of the message numbers is guaranteed. The message numbers within a user program must be unique.

SEVERITY

Weighting of the event Here the value 0 is the highest weighting. This parameter is irrelevant for processing the message. Possible values: 0 ... 127 (default value: 64)

DONE

Status parameter *DONE*, Message generation completed.

SD_i

i-th associated value It is valid 1 ≤ i ≤ maxNumber. The max. number of associated values may be found in the technical data of your CPU. Permitted are only data of the type BOOL, (not permitted: bit field), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND _TIME.

When the ANY pointer accesses a DB, the DB always must be specified (e.g.: P# DB10.DBX5.0 Byte 10).

Error information ERROR / STATUS

ERROR = TRUE indicates that an error has occurred during processing. For details refer to parameter *STATUS*. The following table contains all the error information specific to SFB 31 that can be output with the *ERROR* and *STATUS* parameters.

ERROR	STATUS (decimal)	Description
0	11	Message lost: The previous signal change or the previous message could not be sent and will be replaced by the current message.
0	22	■ Error in the pointer to the associated values *SD_i*: – relating to the data length or the data type – Associated values in the user memory not accessible, for example, due to deleted DB or area length error. The activated message is sent without associated values or if necessary with even possible number of associated values. ■ The actual parameter you have selected for *SEVERITY* is higher than the permitted range. The activated message is sent with *SEVERITY* = 127.
0	25	Communication was initiated. The message is being processed.
1	1	Communication problems: Disconnection or no logon
1	4	At the first call the specified *EV_ID* is outside the permitted range or the ANY pointer *SD_i* has a formal error or the maximum memory area that can be sent for the CPU per SFB 31 was exceeded.
1	10	Access to local user memory not possible (for example, access to a deleted DB)
1	12	When the SFB was called: an instance DB that does not belong to SFB 31 was specified or a shared DB instead of an instance DB was specified.
1	18	*EV_ID* was already being used by one of the SFBs 31 or 33 ... 36.
1	20	Not enough working memory.
1	21	The message with the specified *EV_ID* is disabled.

11.2.12 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).
- 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*.
- 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".

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

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.

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 ≤ I ≤ 16	INPUT	BOOL	I, Q, M, D, L, constant	Event bit No. I (belongs to step I)
OUTj,0 ≤ j ≤ 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 pro-cessed
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

Parameter	Declaration	Data type	Memory block	Description
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. *'Error information' on page 417*

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.

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)

11.2.13 SFB 33 - ALARM - Messages with acknowledgement display

Description

Generating block-related messages with acknowledgement display:

- SFB 33 ALARM monitors a signal:
  - Acknowledgement triggered reporting is disabled (default): The block generates a message both on a rising edge (event entering state) and on a falling edge (event leaving state) to which associated values can be added.
  - Acknowledgement triggered reporting is enabled: After an incoming message is generated for the signal, the block will no longer generate messages until you have acknowledged this incoming message on a displaying device.
- When the SFB is first called, a message with the current signal state is sent. The message is sent to all stations logged on for this purpose.
- Once your acknowledgement has been received from a logged on display device, the acknowledgement information is passed on to all other stations logged on for this purpose.
- One message memory with 2 memory blocks is available for each instance of SFB 33 ALARM.
- SFB 33 ALARM complies with the IEC 1131-5 standard.

Parameter

Parameter	Declaration	Data type	Memory block	Description
EN_R	INPUT	BOOL	I, Q, M, D, L	Control parameter
SIG	INPUT	BOOL	I, Q, M, D, L	The signal to be monitored.
ID	INPUT	WORD	Constant (I, Q, M, D, L)	Data channel for messages: EEEEh. *ID* is only evaluated at the first call.

Parameter	Declaration	Data type	Memory block	Description
EV_ID	INPUT	DWORD	Constant (I, Q, M, D, L)	Message number (not allowed: 0)
SEVERITY	INPUT	WORD	Constant (I, Q, M, D, L)	Weighting of the event
DONE	OUTPUT	BOOL	I, Q, M, D, L	Status parameter *DONE*: Message generation completed.
ERROR	OUTPUT	BOOL	I, Q, M, D, L	*ERROR* status parameter
STATUS	OUTPUT	WORD	I, Q, M, D, L	*STATUS* parameter: Display of an error information
ACK_DN	OUTPUT	BOOL	I, Q, M, D, L	Outgoing event was acknowledged
ACK_UP	OUTPUT	BOOL	I, Q, M, D, L	Incoming event was acknowledged.
SD_i	IN_OUT	ANY	I, Q, M, D, T, C	i-th associated value

EN_R

Control parameter (enabled to receive) that decides whether the outputs *ACK_UP* and *ACK_DN* are updated at the first block call (*EN_R* = 1) or not (*EN_R* = 0). If *EN_R* = 0 the output parameters *ACK_UP* and *ACK_DN* remain unchanged.

SIG

The signal to be monitored.

ID

Data channel for messages: EEEEh. *ID* is only evaluated at the first call.

EV_ID

EV_ID is only evaluated at the first call. Subsequently, the message number used for the first call applies to every call of SFB with the corresponding instance DB. The message number is automatically assigned by the Siemens STEP®7 programming tool. So the consistency of the message numbers is guaranteed. The message numbers within a user program must be unique.

SEVERITY

Weighting of the event Here the value 0 is the highest weighting. This parameter is irrelevant for processing the message. Possible values: 0 ... 127 (default value: 64)

DONE

Status parameter *DONE*, Message generation completed.

ACK_DN

Outgoing event has been acknowledged on a display device. Initialization status: 1. The *ACK_DN* output is reset at the negative edge. It is set when your acknowledgement of the event leaving the state is received from a logged on display device.

ACK_UP

Incoming event has been acknowledged on a display device. Initialization status: 1 The *ACK_UP* output is reset at the rising edge. It is set when your acknowledgement of the event entering the state has arrived from a logged on display device.

SD_i

i-th associated value It is valid 1 ≤ i ≤ maxNumber. The max. number of associated values may be found in the technical data of your CPU. Permitted are only data of the type BOOL, (not permitted: bit field), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND _TIME.

When the ANY pointer accesses a DB, the DB always must be specified. (e.g.: P# DB10.DBX5.0 Byte 10).

Error information *ERROR / STATUS*

ERROR = TRUE indicates that an error has occurred during processing. For details refer to parameter *STATUS*. The following table contains all the error information specific to SFB 33 that can be output with the *ERROR* and *STATUS* parameters.

ERROR	*STATUS* (decimal)	Description
0	11	Message lost: The previous signal change or the previous message could not be sent and will be replaced by the current message.
0	22	■ Error in the pointer to the associated values *SD_i*: – relating to the data length or the data type – Associated values in the user memory not accessible, for example, due to deleted DB or area length error. The activated message is sent without associated values or if necessary with even possible number of associated values. ■ The actual parameter you have selected for *SEVERITY* is higher than the permitted range. The activated message is sent with *SEVERITY* = 127.
0	25	Communication was initiated. The message is being processed.
1	1	Communication problems: Disconnection or no logon With acknowledgement-triggered reporting active: temporary display, if no display devices support acknowledgement-triggered reporting.
1	4	At the first call the specified *EV_ID* is outside the permitted range or the ANY pointer *SD_i* has a formal error or the maximum memory area that can be sent for the CPU per SFB 31 was exceeded.
1	10	Access to local user memory not possible (for example, access to a deleted DB)
1	12	When the SFB was called: an instance DB that does not belong to SFB 31 was specified or a shared DB instead of an instance DB was specified.
1	18	*EV_ID* was already being used by one of the SFBs 31 or 33 ... 36.

ERROR	*STATUS* (decimal)	Description
1	20	Not enough working memory.
1	21	The message with the specified *EV_ID* is disabled.

After the first block call, the ACK_UP and ACK_DN outputs have the value 1 and it is assumed that the previous value of the SIG input was 0.

11.2.14 SFB 34 - ALARM_8 - Messages without associated values (8x)

Description

Generating block-related messages without associated values for 8 signals.

- SFB 34 ALARM_8 is identical to SFB 35 ALARM_8P.
- Except the associated values are not transferred.

Parameter

Parameter	Declaration	Data type	Memory block	Description
EN_R	INPUT	BOOL	I, Q, M, D, L Constant	Control parameter
SIG_i	INPUT	BOOL	I, Q, M, D, L	i-th signal to be monitored
ID	INPUT	WORD	Constant (I, Q, M, D, L)	Data channel for messages: EEEEh. *ID* is only evaluated at the first call.
EV_ID	INPUT	DWORD	Constant (I, Q, M, D, L)	Message number (not allowed: 0)
SEVERITY	INPUT	WORD	Constant (I, Q, M, D, L)	Weighting of the event
DONE	OUTPUT	BOOL	I, Q, M, D, L	Status parameter *DONE*: Message generation completed.
ERROR	OUTPUT	BOOL	I, Q, M, D, L	Status parameter *ERROR*
STATUS	OUTPUT	WORD	I, Q, M, D, L	Status parameter *STATUS*: Display of an error information
ACK_STATE	OUTPUT	WORD	I, Q, M, D, L	Bit field acknowledgement status of all 8 messages

EN_R

Control parameter (enabled to receive) that decides whether the output *ACK_STATE* is updated (*EN_R* = 1) when the block is called or not (*EN_R* = 0).

SIG_i

i-th signal to be monitored It is valid 1 ≤ i ≤ 8.

ID

Data channel for messages: EEEEh. *ID* is only evaluated at the first call.

EV_ID

EV_ID is only evaluated at the first call. Subsequently, the message number used for the first call applies to every call of SFB with the corresponding instance DB. The message number is automatically assigned by the Siemens STEP®7 programming tool. So the consistency of the message numbers is guaranteed. The message numbers within a user program must be unique.

SEVERITY

Weighting of the event Here the value 0 is the highest weighting. This parameter is irrelevant for processing the message. Possible values: 0 ... 127 (default value: 64)

DONE

Status parameter *DONE*: Message generation completed.

ACK_STATE

Bit field with the current acknowledgement status of all 8 messages.

- Bit 7 ... 0: incoming event of SIG_1 ... SIG_8
- Bit 15 ... 8: outgoing event of SIG_1 ... SIG_8

(1: Event acknowledged, 0: Event not acknowledged):

Initialization status: FFFFh, this means, all incoming and outgoing events have been acknowledged.

Error information *ERROR / STATUS*

ERROR = TRUE indicates that an error has occurred during processing. For details refer to parameter *STATUS*. The following table contains all the error information specific to SFB 34 that can be output with the *ERROR* and *STATUS* parameters.

ERROR	*STATUS* (decimal)	Description
0	11	Message lost: The previous signal change or the previous message could not be sent and will be replaced by the current message.
0	22	The actual parameter you have selected for *SEVERITY* is higher than the permitted range. The activated message is sent with *SEVERITY* = 127.
0	25	Communication was initiated. The message is being processed.
1	1	Communication problems: Communications problems: connection abort or no logon With acknowledgement-triggered reporting active: temporary display, if no display devices support acknowledgement-triggered reporting.
1	4	At the first call, the specified *EV_ID* is outside the permitted range.

ERROR	*STATUS* (decimal)	Description
1	10	Access to local user memory not possible (for example, access to a deleted DB)
1	12	When the SFB was called: an instance DB that does not belong to SFB 34 was specified or a shared DB instead of an instance DB was specified.
1	18	EV_ID was already being used by one of the SFBs 31 or 33 ... 36.
1	20	Not enough working memory.
1	21	The message with the specified *EV_ID* is disabled.

After the first block call. all the bits of the ACK_STATE output are set and it is assumed that the previous values of inputs SIG_i, 1≤ i ≤ 8 were 0.

11.2.15 SFB 35 - ALARM_8P - Messages with associated values (8x)

Description

Generating block-related messages with associated values for 8 signals.

- SFB 35 ALARM_8P represents a linear extension of SFB 33 ALARM to 8 signals.
- As long as you have not enabled acknowledgement triggered reporting, a message will always be generated when a signal transition is detected at one or more signals (exception: a message is always sent at the first block call). All 8 signal are allocated a common message number that is split into 8 sub-messages on the displaying device. You can acknowledge each individual message separately or a group of messages.
- You can use the *ACK_STATE* output parameter to process the acknowledgement state of the individual messages in your program. If you disable or enable a message of an ALARM_8P block, this always affects the entire ALARM_8P block. Disabling and enabling of individual signals is not possible.
- One message memory with 2 memory blocks is available for each instance of SFB35 ALARM_8P.

Parameter

Parameter	Declaration	Data type	Memory block	Description
EN_R	INPUT	BOOL	I, Q, M, D, L	Control parameter
SIG_i,	INPUT	BOOL	I, Q, M, D, L	i-th signal to be monitored
ID	INPUT	WORD	Constant (I, Q, M, D, L)	Data channel for messages: EEEEh. ID is only evaluated at the first call.

Parameter	Declaration	Data type	Memory block	Description
EV_ID	INPUT	DWORD	Constant (I, Q, M, D, L)	Message number (not allowed: 0)
SEVERITY	INPUT	WORD	Constant (I, Q, M, D, L)	Weighting of the event
DONE	OUTPUT	BOOL	I, Q, M, D, L	Status parameter *DONE*: Message generation completed.
ERROR	OUTPUT	BOOL	I, Q, M, D, L	*ERROR* status parameter
STATUS	OUTPUT	WORD	I, Q, M, D, L	Status parameter *STATUS*: Display of an error information
ACK_STATE	OUTPUT	WORD	I, Q, M, D, L	Bit field acknowledgement status of all 8 messages
SD_j	IN_OUT	ANY	I, Q, M, D, T, C	j-th associated value

EN_R

Control parameter (enabled to receive) that decides whether the output *ACK_STATE* is updated (*EN_R* = 1) when the block is called or not (*EN_R* = 0).

SIG_i

i-th signal to be monitored It is valid 1 ≤ i ≤ 8.

ID

Data channel for messages: EEEEh. ID is only evaluated at the first call.

EV_ID

EV_ID is only evaluated at the first call. Subsequently, the message number used for the first call applies to every call of SFB with the corresponding instance DB. The message number is automatically assigned by the Siemens STEP®7 programming tool. So the consistency of the message numbers is guaranteed. The message numbers within a user program must be unique.

SEVERITY

Weighting of the event Here the value 0 is the highest weighting. This parameter is irrelevant for processing the message. Possible values: 0 ... 127 (default value: 64)

DONE

Status parameter *DONE*, Message generation completed.

ACK_STATE

Bit field with the current acknowledgement status of all 8 messages.

- Bit 7 ... 0: incoming event of SIG_1 ... SIG_8
- Bit 15 ... 8: outgoing event of SIG_1 ... SIG_8

1 Event acknowledged, 0: Event not acknowledged):

Initialization status: FFFFh, this means, all incoming and outgoing events have been acknowledged.

SD_i

i-th associated value It is valid 1 ≤ i ≤ maxNumber. The max. number of associated values may be found in the technical data of your CPU. Permitted are only data of the type BOOL, (not permitted: bit field), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND _TIME.

When the ANY pointer accesses a DB, the DB always must be specified. (e.g.: P# DB10.DBX5.0 Byte 10).

Error information
ERROR / STATUS

ERROR = TRUE indicates that an error has occurred during processing. For details refer to parameter *STATUS*. The following table contains all the error information specific to SFB 35 that can be output with the *ERROR* and *STATUS* parameters.

ERROR	*STATUS* (decimal)	Description
0	11	Message lost: The previous signal change or the previous message could not be sent and will be replaced by the current message.
0	22	■ Error in the pointer to the associated values SD_i: – relating to the data length or the data type – No access to associated values in user memory, for example, due to deleted DB or area length error. The activated message is sent without associated values. ■ The actual parameter you have selected for *SEVERITY* is higher than the permitted range. The activated message is sent with *SEVERITY* = 127.
0	25	Communication was initiated. The message is being processed.
1	1	Communication problems: Disconnection or no logon With acknowledgement-triggered reporting active: temporary display, if no display devices support acknowledgement-triggered reporting.
1	4	At the first call the specified *EV_ID* is outside the permitted range or the ANY pointer SD_i has a formal error or the maximum memory area that can be sent for the CPU per SFB 35 was exceeded.
1	10	Access to local user memory not possible (for example, access to a deleted DB)
1	12	When the SFB was called: an instance DB that does not belong to SFB 34 was specified or a shared DB instead of an instance DB was specified.
1	18	EV_ID was already being used by one of the SFBs 31 or 33 ... 36.
1	20	Not enough working memory.
1	21	The message with the specified *EV_ID* is disabled.

After the first block call. all the bits of the ACK_STATE output are set and it is assumed that the previous values of inputs SIG_i, 1≤ i ≤ 8 were 0.

11.2.16 SFB 36 - NOTIFY - Messages without acknowledgement display

Description

Generating block-related messages without acknowledgement display.

- SFB 36 NOTIFY monitors a signal. It generates a message both on a rising edge (event entering state) and on a falling edge (event leaving state) with associated values.
- When the SFB is first called, a message with the current signal state is sent. The message is sent to all stations logged on for this purpose.
- The associated values are queried when the edge is detected and assigned to the message.

Parameter

Parameter	Declaration	Data type	Memory block	Description
SIG	INPUT	BOOL	I, Q, M, D, L	The signal to be monitored.
ID	INPUT	WORD	Constant (I, Q, M, D, L)	Data channel for messages: EEEEh. *ID* is only evaluated at the first call.
EV_ID	INPUT	DWORD	Constant (I, Q, M, D, L)	Message number (not allowed: 0)
SEVERITY	INPUT	WORD	Constant (I, Q, M, D, L)	Weighting of the event
DONE	OUTPUT	BOOL	I, Q, M, D, L	Status parameter *DONE*: Message generation completed.
ERROR	OUTPUT	BOOL	I, Q, M, D, L	*ERROR* status parameter
STATUS	OUTPUT	WORD	I, Q, M, D, L	*STATUS* parameter: Display of an error information
SD_i	IN_OUT	ANY	I, Q, M, D, T, C	i-th associated value

SIG

The signal to be monitored.

ID

Data channel for messages: EEEEh. *ID* is only evaluated at the first call.

EV_ID

EV_ID is only evaluated at the first call. Subsequently, the message number used for the first call applies to every call of SFB with the corresponding instance DB. The message number is automatically assigned by the Siemens STEP®7 programming tool. So the consistency of the message numbers is guaranteed. The message numbers within a user program must be unique.

SEVERITY

Weighting of the event Here the value 0 is the highest weighting. This parameter is irrelevant for processing the message. Possible values: 0 ... 127 (default value: 64)

DONE

Status parameter *DONE*: Message generation completed.

SD_i

i-th associated value It is valid 1 ≤ I ≤ maxNumber. The max. number of associated values may be found in the technical data of your CPU. Permitted are only data of the type BOOL, (not permitted: bit field), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND _TIME.

When the ANY pointer accesses a DB, the DB always must be specified. (e.g.: P# DB10.DBX5.0 Byte 10).

Error information *ERROR / STATUS*

The following table contains all the error information specific to SFB 36 that can be output with the *ERROR* and *STATUS* parameters.

ERROR	*STATUS* (decimal)	Description
0	11	Message lost: The previous signal change or the previous message could not be sent and will be replaced by the current message.
0	22	■ Error in the pointer to the associated values SD_i: – relating to the data length or the data type – Associated values in the user memory not accessible, for example, due to deleted DB or area length error. The activated message is sent without associated values or if necessary with even possible number of associated values. ■ The actual parameter you have selected for *SEVERITY* is higher than the permitted range. The activated message is sent with *SEVERITY* = 127.
0	25	Communication was initiated. The message is being processed.
1	1	Communication problems: Disconnection or no logon
1	4	At the first call the specified *EV_ID* is outside the permitted range or the ANY pointer *SD_i* has a formal error or the maximum memory area that can be sent for the CPU per SFB 36 was exceeded.

ERROR	*STATUS* (decimal)	Description
1	10	Access to local user memory not possible (for example, access to a deleted DB)
1	12	When the SFB was called: an instance DB that does not belong to SFB 36 was specified or a shared DB instead of an instance DB was specified.
1	18	*EV_ID* was already being used by one of the SFBs 31 or 33 ... 36.
1	20	Not enough working memory.
1	21	The message with the specified *EV_ID* is disabled.

11.2.17 SFB 47 - COUNT - Counter controlling

Description

The SFB 47 is a specially developed block for compact CPUs for controlling of the counters. The SFB is to be called with the corresponding instance DB. Here the parameters of the SFB are stored. With the SFB COUNT (SFB 47) you have following functional options:

- Start/Stop the counter via software gate *SW_GATE*
- Enable/control digital output DO
- Read the status bit
- Read the actual count and latch value
- Request to read/write internal counter registers

Parameters

Name	Data type	Address (Instance DB)	Default value	Comment
LADDR	WORD	0.0	300h	This parameter is not evaluated. Always the internal I/O periphery is addressed.
CHANNEL	INT	2.0	0	Channel number
SW_GATE	BOOL	4.0	FALSE	Enables the Software gate
CTRL_DO	BOOL	4.1	FALSE	Enables the output False: Standard Digital Output
SET_DO	BOOL	4.2	FALSE	Parameter is not evaluated
JOB_REQ	BOOL	4.3	FALSE	Initiates the job (edge 0-1)
JOB_ID	WORD	6.0	0	Job ID
JOB_VAL	DINT	8.0	0	Value for write jobs
STS_GATE	BOOL	12.0	FALSE	Status of the internal gate
STS_STRT	BOOL	12.1	FALSE	Status of the hardware gate
STS_LTCH	BOOL	12.2	FALSE	Status of the latch input
STS_DO	BOOL	12.3	FALSE	Status of the output
STS_C_DN	BOOL	12.4	FALSE	Status of the down-count Always indicates the last direction of count. After the first SFB call *STS_C_DN* is set FALSE.
STS_C_UP	BOOL	12.5	FALSE	Status of the up-count Always indicates the last direction of count. After the first SFB call *STS_C_UP* is set TRUE.
COUNTVAL	DINT	14.0	0	Actual count value
LATCHVAL	DINT	18.0	0	Actual latch value
JOB_DONE	BOOL	22.0	TRUE	New job can be started
JOB_ERR	BOOL	22.1	FALSE	Job error
JOB_STAT	WORD	24.0	0	Job error ID

Local data only in instance DB

Name	Data type	Address (Instance DB)	Default value	Comment
RES00	BOOL	26.0	FALSE	reserved
RES01	BOOL	26.1	FALSE	reserved
RES02	BOOL	26.2	FALSE	reserved
STS_CMP	BOOL	26.3	FALSE	Comparator Status * Status bit *STS_CMP* indicates that the comparison condition of the comparator is or was reached. *STS_CMP* also indicates that the output was set. (*STS_DO* = TRUE).
RES04	BOOL	26.4	FALSE	reserved
STS_OFLW	BOOL	26.5	FALSE	Overflow status *
STS_UFLW	BOOL	26.6	FALSE	Underflow status *
STS_ZP	BOOL	26.7	FALSE	Status of the zero mark * The bit is only set when counting without main direction. Indicates the zero mark. This is also set when the counter is set to 0 or if is start counting.
JOB_OVAL	DINT	28.0		Output value for read request.
RES10	BOOL	32.0	FALSE	reserved
RES11	BOOL	32.1	FALSE	reserved
RES_STS	BOOL	32.2	FALSE	Reset status bits: Resets the status bits: STS_CMP, STS_OFLW, STS_ZP. The SFB must be twice called to reset the status bit.
*) Reset with RES_STS				

Per channel you may call the SFB in each case with the same instance DB, since the data necessary for the internal operational are stored here. Writing accesses to outputs of the instance DB is not permissible.

Counter request interface

To read/write counter registers the request interface of the SFB 47 may be used. So that a new job may be executed, the previous job must have be finished with *JOB_DONE* = TRUE.

Proceeding

The deployment of the request interface takes place at the following sequence:

1. Edit the following input parameters:

Name	Data type	Address (DB)	Default	Comment
JOB_REQ	BOOL	4.3	FALSE	Initiates the job (edges 0-1) *
JOB_ID	WORD	6.0	0	Job ID: 00h Job without function 01h Writes the *count value* 02h Writes the *load value* 04h Writes the *comparison value* 08h Writes the *hysteresis* 10h Writes the *pulse duration* 20h Writes the *end value* 82h Reads the *load value* 84h Reads the *comparison value* 88h Reads the *hysteresis* 90h Reads the *pulse duration* A0h Reads the *end value*
JOB_VAL	DINT	8.0	0	Value for write jobs
*) State remains set also after a CPU STOP-RUN transition.				

2. Call the SFB. The job is processed immediately. *JOB_DONE* only applies to SFB run with the result FALSE. *JOB_ERR* = TRUE if an error occurred. Details on the error cause are indicated at *JOB_STAT*.

Name	Data type	Address (DB)	Default	Comment
JOB_DONE	BOOL	22.0	TRUE	New job can be started
JOB_ERR	BOOL	22.1	FALSE	Job error
JOB_STAT	WORD	24.0	0000h	Job error ID 0000h No error 0121h *Comparison value* too low 0122h *Comparison value* too high 0131h *Hysteresis* too low 0132h *Hysteresis* too high 0141h *Pulse duration* too low 0142h *Pulse duration* too high 0151h *Load value* too low 0152h *Load value* too high 0161h *Count value* too low 0162h *Count value* too high 01FFh Invalid *job ID*

3. A new job may be started with *JOB_DONE* = TRUE.
4. A value to be read of a read job may be found in *JOB_OVAL* in the instance DB at address 28.

Permitted value range for JOB_VAL

Continuous count:

Job	Valid range
Writing *counter* directly	-2147483647 ($-2^{31}+1$) ... +2147483646 ($2^{31}-2$)
Writing the *load value*	-2147483647 ($-2^{31}+1$) ... +2147483646 ($2^{31}-2$)
Writing *comparison value*	-2147483648 (-2^{31}) ... +2147483647 ($2^{31}-1$)
Writing *hysteresis*	0 ... 255
Writing *pulse duration**	0 ... 510ms

Single/periodic count, no main count direction:

Job	Valid range
Writing *counter* directly	-2147483647 ($-2^{31}+1$) ... +2147483646 ($2^{31}-2$)
Writing the *load value*	-2147483647 ($-2^{31}+1$) ... +2147483646 ($2^{31}-2$)
Writing *comparison value*	-2147483648 (-2^{31}) ... +2147483647 ($2^{31}-1$)
Writing *hysteresis*	0 ... 255
Writing *pulse duration**	0 ... 510ms

Single/periodic count, main count direction up:

Job	Valid range
End value	2 ... +2147483646 ($2^{31}-1$)
Writing *counter* directly	-2147483648 (-2^{31}) ... *end value* -2
Writing the *load value*	-2147483648 (-2^{31}) ... *end value* -2
Writing *comparison value*	-2147483648 (-2^{31}) ... *end value* -1
Writing *hysteresis*	0 ... 255
Writing *pulse duration**	0 ... 510ms

Single/periodic count, main count direction down:

Job	Valid range
Writing *counter directly*	2 ... +2147483647 ($2^{31}-1$)
Writing the *load value*	2 ... +2147483647 ($2^{31}-1$)
Writing *comparison value*	1 ... +2147483647 ($2^{31}-1$)
Writing *hysteresis*	0 ... 255
Writing *pulse duration**	0 ... 510ms
*) Only even values allowed. Odd values are automatically rounded.	

Latch function

As soon as during a count process an edge 0-1 is recognized at the "Latch" input of a counter, the recent counter value is stored in the according latch register.

You may access the latch register via *LATCHVAL* of the SFB 47.

A just in *LATCHVAL* loaded value remains after a STOP-RUN transition.

11.2.18 SFB 48 - FREQUENC - Frequency measurement

Description

The SFB 48 is a specially developed block for compact CPUs for frequence measurement.

- The SFB FREQUENC should cyclically be called (e.g. OB 1) for controlling the frequency measurement.
- The SFB is to be called with the corresponding instance DB. Here the parameters of the SFB are stored.
- Among others the SFB 48 contains a request interface. Hereby you get read and write access to the registers of the frequency meter.
- So that a new job may be executed, the previous job must have be finished with *JOB_DONE* = TRUE.
- Per channel you may call the SFB in each case with the same instance DB, since the data necessary for the internal operational are stored here. Writing accesses to outputs of the instance DB is not permissible.
- With the SFB FREQUENC (SFB 48) you have following functional options:
  - Start/Stop the frequency meter via software gate SW_GATE
  - Read the status bit
  - Read the evaluated frequency
  - Request to read/write internal registers of the frequency meter.

Parameters

Name	Declaration	Data type	Address (Inst.-DB)	Default value	Comment
LADDR	INPUT	WORD	0.0	300h	This parameter is not evaluated. Always the internal I/O periphery is addressed.
CHANNEL	INPUT	INT	2.0	0	Channel number
SW_GATE	INPUT	BOOL	4.0	FALSE	Enables the Software gate
JOB_REQ	INPUT	BOOL	4.3	FALSE	Initiates the job (edge 0-1)
JOB_ID	INPUT	WORD	6.0	0	Job ID
JOB_VAL	INPUT	DINT	8.0	0	Value for write jobs
STS_GATE	OUTPUT	BOOL	12.0	FALSE	Status of the internal gate
MEAS_VAL	OUTPUT	DINT	14.0	0	Evaluated frequency
JOB_DONE	OUTPUT	BOOL	22.0	TRUE	New job can be started.
JOB_ERR	OUTPUT	BOOL	22.1	FALSE	Job error
JOB_STAT	OUTPUT	WORD	24.0	0	Job error ID

Local data only in instance DB

Name	Data type	Address (Instance DB)	Default	Comment
JOB_OVAL	DINT	28.0	-	Output value for read request.

Per channel you may call the SFB in each case with the same instance DB, since the data necessary for the internal operational are stored here. Writing accesses to outputs of the instance DB is not permissible.

Frequency meter request interface

To read/write the registers of the frequency meter the request interface of the SFB 48 may be used.

So that a new job may be executed, the previous job must have be finished with *JOB_DONE* = TRUE.

Proceeding

The deployment of the request interface takes place at the following sequence:

→ Edit the following input parameters:

Name	Data type	Address (DB)	Default	Comment
JOB_REQ	BOOL	4.3	FALSE	Initiates the job (edges 0-1)
JOB_ID	WORD	6.0	0	Job ID: 00h Job without function 04h Writes the integration time 84h Read the integration time
JOB_VAL	DINT	8.0	0	Value for write jobs. Permitted value for integration time: 10 ... 10000ms

→ Call the SFB. The job is processed immediately. *JOB_DONE* only applies to SFB run with the result FALSE. *JOB_ERR* = TRUE if an error occurred. Details on the error cause are indicated at *JOB_STAT*.

Name	Data type	Address (DB)	Default	Comment
JOB_DONE	BOOL	22.0	TRUE	New job can be started
JOB_ERR	BOOL	22.1	FALSE	Job error
JOB_STAT	WORD	24.0	0000h	Job error ID 0000h No error 0221h Integration time too low 0222h Integration time too high 02FFh Invalid job ID 8001h Parameter error 8009h Channel no. not valid

1. A new job may be started with *JOB_DONE* = TRUE.
2. A value to be read of a read job may be found in *JOB_OVAL* in the instance DB at address 28.

Channel no. not valid

(8009h and Parameter error 8001h)

If you have preset a CHANNEL number greater than 3, the error "Channel no. not valid " (8009h) is reported. if you have preset a CHANNEL number greater than the maximum channel number of the CPU, "Parameter error" (8001h) is reported.

Controlling frequency meter

The frequency meter is controlled by the internal gate (I gate). The I gate is identical to the software gate (SW gate).

SW gate:

open (activate): In the user program by setting *SW_GATE* of SFB 48

close (deactivate): In the user program by resetting *SW_GATE* of SFB 48

11.2.19 SFB 49 - PULSE - Pulse width modulation

Description

The SFB 49 is a specially developed block for compact CPUs for pulse width modulation.

- The SFB PULSE should cyclically be called (e.g. OB 1) for controlling the frequency measurement.
- The SFB is to be called with the corresponding instance DB. Here the parameters of the SFB are stored.
- Among others the SFB 49 contains a request interface. Hereby you get read and write access to the registers of the pulse width modulation.
- So that a new job may be executed, the previous job must have be finished with *JOB_DONE* = TRUE.

- Per channel you may call the SFB in each case with the same instance DB, since the data necessary for the internal operational are stored here. Writing accesses to outputs of the instance DB is not permissible.
- With the SFB PULSE (SFB 49) you have following functional options:
  – Start/Stop the pulse width modulation via software gate SW_GATE
  – Enabling/controlling of the PWM output
  – Read status bits
  – Request to read/write internal registers of the pulse width modulation

Parameters

Name	Declaration	Data type	Address (Inst.-DB)	Default value	Comment
LADDR	INPUT	WORD	0.0	300h	This parameter is not evaluated. Always the internal I/O periphery is addressed.
CHANNEL	INPUT	INT	2.0	0	Channel number
SW_EN	INPUT	BOOL	4.0	FALSE	Enables the Software gate
OUTP_VAL	INPUT	INT	6.0	0	Output value
JOB_REQ	INPUT	BOOL	8.0	FALSE	Initiates the job (edge 0-1)
JOB_ID	INPUT	WORD	10.0	0	Job ID
JOB_VAL	INPUT	DINT	12.0	0	Value for write jobs
STS_EN	OUTPUT	BOOL	16.0	FALSE	Status of the internal gate
JOB_DONE	OUTPUT	BOOL	16.3	TRUE	New job can be started.
JOB_ERR	OUTPUT	BOOL	16.4	FALSE	Job error
JOB_STAT	OUTPUT	WORD	18.0	0	Job error ID

Local data only in Instance DB

Name	Data type	Address (Instance DB)	Default	Comment
JOB_OVAL	DINT	20.0	-	Output value for read request.

Per channel you may call the SFB in each case with the same instance DB, since the data necessary for the internal operational are stored here. Writing accesses to outputs of the instance DB is not permissible.

PWM Request interface

To read/write the registers of the pulse width modulation the request interface of the SFB 49 may be used.

So that a new job may be executed, the previous job must have be finished with *JOB_DONE* = TRUE.

Proceeding

The deployment of the request interface takes place at the following sequence:

→ Edit the following input parameters:

Name	Data type	Address (DB)	Default	Comment
JOB_REQ	BOOL	8.0	FALSE	Initiates the job (edges 0-1)
JOB_ID	WORD	10.0	0	Job ID: 00h Job without function 01h write period duration 02h write on-delay 04h write minimum pulse duration 81h read period duration 82h read on-delay 84h read minimum pulse duration
JOB_VAL	DINT	8.0	0	Value for write jobs. -2147483648 (-2^{31}) to +2147483647 (2^{31}-1)

→ Call the SFB. The job is processed immediately. *JOB_DONE* only applies to SFB run with the result FALSE. *JOB_ERR* = TRUE if an error occurred. Details on the error cause are indicated at *JOB_STAT*.

Name	Data type	Address (DB)	Default	Comment
JOB_DONE	BOOL	22.0	TRUE	New job can be started
JOB_ERR	BOOL	22.1	FALSE	Job error
JOB_STAT	WORD	24.0	0000h	Job error ID 0000h No error 0411h Period duration time too low 0412h Period duration time too high 0421h On-delay too low 0422h On-delay too high 0431h Minimum pulse duration too low 0432h Minimum pulse duration too high 04FFh Invalid job ID 8001h Parameter error 8009h Channel no. not valid

1. A new job may be started with *JOB_DONE* = TRUE.
2. A value to be read of a read job may be found in *JOB_OVAL* in the instance DB at address 28.

Channel no. not valid (8009h) and Parameter error (8001h)

If you have preset a CHANNEL number greater than 3, the error "Channel no. not valid" (8009h) is reported. if you have preset a CHANNEL number greater than the maximum channel number of the CPU, "Parameter error" (8001h) is reported.

Controlling PWM

The pulse width modulation is controlled by the internal gate (I gate). The I gate is identical to the software gate (SW gate).

SW gate:

open (activate): In the user program by setting *SW_EN* of SFB 49

close (deactivate): In the user program by resetting *SW_EN* of SFB 49

If values during the PWM output are changed, the new values will be issued until the beginning of a new period. A just started period runs always to the end!

11.2.20 SFB 52 - RDREC - Reading record set

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 can read a record set with the number INDEX from a module 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!

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 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

Parameter	Declaration	Data type	Memory block	Description
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

Chapter 11.2.22 'SFB 54 - RALRM - Receiving an interrupt from a periphery module' on page 441

11.2.21 SFB 53 - WRREC - Writing record set

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 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!

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 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.
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.
RECORD	IN_OUT	ANY	I, Q, M, D, L	Record set

Error information

Chapter 11.2.22 'SFB 54 - RALRM - Receiving an interrupt from a periphery module' on page 441

11.2.22 SFB 54 - RALRM - Receiving an interrupt from a periphery module

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 or a component of the corresponding bus slave and provides this information to its output parameters. The information contained in the input parameters contains the start information of the called OB as well as information from 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.

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	TRUE: A new interrupt was received. FALSE: No new interrupt was received.
STATUS	OUTPUT	DWORD	I, Q, M, D, L	C0000000h: no error C080C300h: Resources are presently occupied C0809000h: Invalid logical start address **Only PROFINET IO:** C080A000h: Read error C080B700h: Invalid area
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.

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

TINFO

<table>
<tr><th colspan="7">TINFO PROFIBUS: Data structure of the target area (task information)</th></tr>
<tr><th>Byte</th><th>Data type</th><th colspan="5">Description</th></tr>
<tr><td>0 ... 19</td><td></td><td colspan="5">Start information of the OB in which the 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</td></tr>
<tr><td>20 ... 27</td><td></td><td colspan="5">Management information:</td></tr>
<tr><td>20</td><td>Byte</td><td colspan="5">centralized: 0
decentralized: DP master system ID (possible values: 1 ... 255)</td></tr>
<tr><td>21</td><td>Byte</td><td colspan="5">central: Module rack number (possible values: 0 ... 31)
distributed: Number of DP station (possible values: 0 ... 127)</td></tr>
<tr><td rowspan="8">22</td><td rowspan="8">Byte</td><td colspan="5">centralized: 0</td></tr>
<tr><td rowspan="7">decentral-ized:</td><td rowspan="5">Bit 3 ... 0</td><td rowspan="5">Slave type</td><td>0000:</td><td>DP</td></tr>
<tr><td>0001:</td><td>DPS7</td></tr>
<tr><td>0010:</td><td>DPS7 V1</td></tr>
<tr><td>0011:</td><td>DP-V1</td></tr>
<tr><td>ab 0100:</td><td>reserved</td></tr>
<tr><td rowspan="2">Bit 7 ... 4</td><td rowspan="2">Profile type</td><td>0000:</td><td>DP</td></tr>
<tr><td>ab 0001:</td><td>reserved</td></tr>
<tr><td rowspan="7">23</td><td rowspan="7">Byte</td><td colspan="5">centralized: 0</td></tr>
<tr><td rowspan="6">decentral-ized:</td><td rowspan="4">Bit 3 ... 0</td><td rowspan="4">Interrupt info type</td><td>0000:</td><td>Transparent
(Interrupt originates from a configured decentralized module)</td></tr>
<tr><td>0001:</td><td>Representative
(Interrupt originating from a non-DP-V1 slave or a slot that is not configured)</td></tr>
<tr><td>0010:</td><td>Generated interrupt (generated in the CPU)</td></tr>
<tr><td>as of 0011:</td><td>reserved</td></tr>
<tr><td rowspan="2">Bit 7 ... 4</td><td rowspan="2">Structure ver-sion</td><td>0000:</td><td>Initial</td></tr>
<tr><td>as of 0001:</td><td>reserved</td></tr>
<tr><td rowspan="2">24</td><td rowspan="2">Byte</td><td colspan="5">centralized: 0</td></tr>
<tr><td colspan="5">decentralized: Flags of the DP master interface</td></tr>
</table>

TINFO PROFIBUS: Data structure of the target area (task information)			
Byte	**Data type**	**Description**	
		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	

TINFO PROFINET IO: Data structure of the target area (task information)			
Byte	**Declaration**	**Data type**	**Description**
0 ... 19	OB Startinfo	BYTE	Start information of the OB in which the SFC 54 was currently called:
20 ... 21	Addressinfo	WORD	Bit 0 ... 10: Number of the DP station (0-2047) Bit 11 ... 14: the last two digits of the PROFINET IO system ID (0-15), to get the whole PROFINET IO system ID you have to add 100 (decimal). Bit 15: 1
22	Slavetype	BYTE	Bit 0 ... 3: 1000: Fixed value for PROFINET IO Bit 4 ... 7: reserved
23	Alarminfo	BYTE	Bit 0 ... 3: 0000: Transparent, which is always the case for PROFINET IO (interrupt originates from a configured distributed module) Bit 4 ... 7: reserved
24	PROFINET IO controller interface	BYTE	Flags of the PROFINET IO controller interface module Bit 0: 0: Interrupt originating from an integrated interface Bit 0: 1: Interrupt originating from an external interface Bit 1 ... 7: reserved
25	Flags of the PROFINET IO controller interface	BYTE	Bit 0: AR data status failure bit of the interrupt message frame or "0" if there is no information in the interrupt Bit 0: 1: IO device is faulty Bit 1... 7: reserved

TINFO PROFINET IO: Data structure of the target area (task information)			
Byte	**Declaration**	**Data type**	**Description**
26 ... 27	PROFINET IO device ID number	WORD	PROFINET IO device ID number as unique identifier of the PROFINET IO device
28 ... 29		WORD	Manufacturer ID
30 ... 31	ID	WORD	ID number of the instance

TINFO EtherCAT: Data structure of the target area (task information)			
Byte	**Declaration**	**Data type**	**Description**
0 ... 19	OB Startinfo	BYTE	Start information of the OB in which the SFC 54 was currently called.
20 ... 21	Addressinfo	WORD	Bit 0 ... 10: Master/Slave Bit 11 ... 14: System ID EtherCAT network - 100 Bit 15: 1: Bit for EtherCAT (PROFINET "look and feel")
22	Slavetype	BYTE	Bit 0 ... 3: 1000: 0b1111 EtherCAT[1] Bit 4 ... 7: reserved
23	Alarminfo	BYTE	Bit 0 ... 3: 0000: Transparent, interrupt originates from a configured distributed module Bit 4 ... 7: reserved
24	EC Flags I	BYTE	Flags of the EtherCAT IO controller interface Bit 0: 0: Interrupt originating from an integrated interface 1: Interrupt originating from an external interface Bit 1 ... 7: reserved
25 ... 31			reserved
1) At 0b1001 PROFINET IO			

AINFO

AINFO PROFIBUS: 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 any-more
			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]	

AINFO PROFINET IO: Data structure of the target area (interrupt information)			
Byte	**Declaration**	**Data type**	**Description**
0 ... 1	Block type	WORD	Bit 0 ... 7: Block type Bit 8 ... 15: reserved
2 ... 3	Block length	WORD	Length of the received interrupt information in byte MIN: 0 MAX: 1536 (1.5kbyte)
4 ... 5	Version	WORD	Bits 0 ... 7: low byte Bits 8 ... 15: high byte

AINFO PROFINET IO: Data structure of the target area (interrupt information)			
Byte	**Declaration**	**Data type**	**Description**
6 ... 7	Interrupt type	WORD	Identifier for the interrupt type: 1: Diagnostic interrupt (incoming) 2: Hardware interrupt 3: Removal interrupt 4: Insertion interrupt 5: Status interrupt 6: Update interrupt 7: Redundancy interrupt 8: Controlled by supervisor 9: Released by supervisor 10: Configured module not inserted 11: Return of submodule 12: Diagnostic interrupt (exiting state) 13: Direct data exchange connection message 14: Neighbourhood change message 15: Clock synchronization message (from bus) 16: Clock synchronization message (from device) 17: Network component message 18: Time synchronization message (from bus) 19 to 31: Reserved 32 to 127: Vendor-specific interrupt 128 ... 65535: reserved, without the following VIPA specific interrupt types: 38CAh: Recovery of the controller 48CAh: Configuration of the controller accepted 39CAh: Controller failure 49CAh: Failure of the controller due to the watchdog 38CBh: Recovery of the device 38CCh: Failure of the recovery of the device 38CDh: Another device is detected during the recovery of the device. 38CEh: Parameter error during the recovery of the device 39CBh: Device failure
8 ... 11	API	DWORD	API (Application Process Identifier)
12 ... 13	Slot number	WORD	Slot number of the component triggering the interrupt (range of values 0 to 65535)

AINFO PROFINET IO: Data structure of the target area (interrupt information)			
Byte	**Declaration**	**Data type**	**Description**
14 ... 15	Interface module slot number	WORD	Interface module slot number of the component triggering the interrupt (range of values 0 to 65535)
16 ... 19	Module ID	DWORD	Specific information on the source of the interrupt: Module ID
20 ... 23	Submodule ID	DWORD	Specific information on the source of the interrupt: Submodule ID
24 ... 25	Interrupt specifier	WORD	Bits 0 to 10: Sequence number (range of values: 0 to 2047) Bit 11: Channel diagnostics: 0: No channel diagnostics available 1: Channel diagnostics available Bit 12: Status of manufacturer-specific diagnostics: 0: No manufacturer-specific status information available 1: Manufacturer-specific status information available Bit 13: Status of diagnostics for interface module: 0: No status information available; all errors corrected 1: Diagnostics for at least one channel and/or status information available Bit 14: reserved Bit 15: Application Relationship Diagnosis State 0: None of the configured modules within this AR is reporting a diagnosis 1: At least one of the configured modules within this AR is reporting a diagnosis
26 ... 1535	Interrupt specifier	WORD	**Note:** The additional interrupt specifier can also be omitted.

AINFO EtherCAT: Data structure of the target area (interrupt information)			
Byte	**Declaration**	**Data type**	**Description**
0, 1	Length	WORD	Length of the received interrupt information in byte: MIN: 0 MAX: 1535 (1.5kbyte)

2, 3	InterruptType	WORD	ID of the interrupt type:
			0001h: DIAGNOSTICS_INTERRUPT_COMMING
			0002h: HARDWARE_INTERRUPT
			000Ch: DIAGNOSTICS_INTERRUPT_GOING
			0020h: MANUFACTOR_SPECIFIC_ALARM_MIN
			// VIPA specific:
			39CAh: CONTROLLER_FAILURE
			49CAh: CONTROLLER_FAILURE_WATCHDOG
			// EtherCAT specific:
			8001h: BUS_STATE_CHANGED
			8002h: SLAVE_STATE_CHANGED
			8003h: TOPOLOGY_OK
			8004h: TOPOLOGY_MISMATCH

4, 5	RackSlot	WORD	Slot number of the EtherCAT master
6, 7	Master/Slave ID	WORD	EtherCAT master/slave address
8, 9	InterruptSpecifier	WORD	Value depends on the interrupt type: InterruptType: Value BUS_STATE_CHANGED: new bus status [1] DIAGNOSTICS_INTERRUPT_GOING: reserved DIAGNOSTICS_INTERRUPT_COMMING: reserved HARDWARE_INTERRUPT: reserved MANUFACTOR_SPECIFIC_ALARM_MIN: reserved SLAVE_STATE_CHANGED: new bus status CONTROLLER_FAILURE: reserved CONTROLLER_FAILURE_WATCHDOG: reserved TOPOLOGY_OK: reserved TOPOLOGY_MISMATCH: reserved

AINFO EtherCAT: Data structure of the target area (interrupt information)			
Byte	**Declaration**	**Data type**	**Description**
10 ... n	Data	BYTE	Content depends on the InterruptType: AlarmType: Content BUS_STATE_CHANGED: Data structure [2] DIAGNOSTICS_INTERRUPT_GOING: CoE-Emergency [3] DIAGNOSTICS_INTERRUPT_COMMING: CoE-Emergency PROCESS_INTERRUPT: CoE-Emergency MANUFACTOR_SPECIFIC_INTERRUPT_MIN: CoE-Emergency SLAVE_STATE_CHANGED: AL Status Code [4] CONTROLLER_FAILURE: Failure code [5] CONTROLLER_FAILURE_WATCHDOG: reserved TOPOLOGY_OK: reserved TOPOLOGY_MISMATCH: reserved

1) EtherCAT-States ➡ *453*

2) Data structure BUS_STATE_CHANGED ➡ *455*

3) CoE emergency ➡ *454*

4) AL Status Code ➡ *454*

5) Failure code ➡ *454*

11.2.22.1 EtherCAT-States

The bus states are coded as follows

Name	Code	Description
Undefined/Unknown	0x00	This status has a slave before he could carry out its initialization routines. For the VIPA EtherCAT master a slave has the undefined state, if there is a slave failure (disconnect).
Init	0x01	There is no direct communication between master and slaves. In this state the master initializes the configuration register of the ESC. There is no process data or mailbox communication.
PreOp	0x02	In this state mailbox communication is possible, but there is no process data communication.
BootStrap	0x03	Special state of the EtherCAT slave, there only mailbox communication takes place. For a firmware update of the salve, the slave must be switched in this state.

Name	Code	Description
SafeOp	0x04	In the state SafeOp mailbox communication is possible an process input data can be exchanged. However, there will be no exchange of process output data.
Op	0x08	In this state mailbox and process data can be exchanged.

11.2.22.2 Cause of controller failure

On a controller failure the alarm specifier provides information about the cause of the failure

Name	Code	Description
REASON_UNKNOWN	0	The reason is unknown
ALARM_OVERFLOW	1	Overflow of interrupts
MESSAGE_QUEUE_OVERFLOW	2	Overflow of EtherCAT events
CYCLIC_FRAMES_NOT_IN_BUSCYCLE	3	EtherCAT receive telegram was not received within the bus cycle time
APPL_BUSCYCLE_ERROR	4	Bus cycle time could not be fetched e.g. due to a high system load

11.2.22.3 CoE emergency

A CoE emergency is a special type of mailbox communication in the EtherCAT slave. Here the EtherCAT slave can signalise the EtherCAT master that an error has occurred. It has the following structure:

Name	Data type	Description
Error Code	WORD	Error Code
Error Register	BYTE	EtherCAT state on the error of the salve
Data	BYTE[5]	Manufacturer Specific Error Field (MEF), contains additional diagnostics data

11.2.22.4 AL Status Code

AL is the abbreviation for Application Layer. The AL status code is an error code of the slave application.

11.2.22.5 Data structure BUS_STATE_CHANGED

Header

NrOfSlavesTotal	-	Number of slaves, which are not in master state
NrOfSlavesUndefined	-	Number of slaves in state *undefined*
NrOfSlavesInit	-	Number of slaves in state *Init*
NrOfSlavesPreop	-	Number of slaves in state *PreOp*
NrOfSlavesBoostrap	-	Number of slaves in state *Bootstrap*
NrOfSlavesSafeop	-	Number of slaves in state *SafeOp*
NrOfSlavesOp	-	Number of slaves in state *Op*

DeviceId

DeviceId[0] ...	-	EtherCAT address of the slave as defined in the configuration
DeviceId[NrOfSlavesTotal-1]	-	EtherCAT address of the slave as defined in the configuration

TINFO and AINFO

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.

Target Area					
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-CPU: 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

***STATUS[3]* (Error ID) can have the following values:**			
Error_Decode	**Error_Code_1**	**Explanation according to DP-V1**	**Description**
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	DP protocol error at layer 2
	A3h	reserved, pass	DP protocol error with Direct-Data-Link-Mapper or User-Inter-face/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 manufac-turer 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 param-eter *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

STATUS[3] **(Error ID) can have the following values:**			
Error_Decode	**Error_Code_1**	**Explanation according to DP-V1**	**Description**
	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,
81h	00h ... FFh		Error in the initial call parameter (with SFB 54: MODE)

***STATUS[3]* (Error ID) can have the following values:**			
Error_Decode	**Error_Code_1**	**Explanation according to DP-V1**	**Description**
	00h		Illegal operating mode
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 zero 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 zero 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

12 Standard

12.1 Converting

12.1.1 FB 80 - LEAD_LAG - Lead/Lag Algorithm

Description

The Lead/Lag Algorithm LEAD_LAG function block allows signal processing to be done on an analog variable. An output *OUT* is calculated based on an input *IN* and the specified gain *GAIN*, lead *LD_TIME*, and lag *LG_TIME* values. The gain value must be greater than zero. The LEAD_LAG algorithm uses the following equation:

$$und\ OUT = \left[\frac{LG\_TIME}{LG\_TIME + SAMPLE\_T}\right] PREV\_OUT + GAIN \left[\frac{LD\_TIME + SAMPLE\_T}{LG\_TIME + SAMPLE\_T}\right] IN - GAIN \left[\frac{LD\_TIME}{LG\_TIME + SAMPLE\_T}\right] PREV\_IN$$

Typically, LEAD_LAG is used in conjunction with loops as a compensator in dynamic feed-forward control. LEAD_LAG consists of two parts. Phase lead shifts the phase of the function block's output so that it leads the input whereas phase lag shifts the output so that it lags the input. Because the lag operation is equivalent to an integration, it can be used as a noise suppressor or a low-pass filter. A lead operation is equivalent to a differentiation and is thus a high-pass filter. LEAD_LAG combined can cause the output phase to lag input at low frequency, and to lead input at high frequency, and can thus be used as a band-pass filter.

Parameters

Parameter	Declaration	Data Type	Memory Area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input with signal state of 1 activates the box
ENO	Output	BOOL	I, Q, M, D, L	Enable output has a signal state of 1 if the function block is executed without error
IN	Input	REAL	I, Q, M, D, L, P, constant	The input value of the current sample period to be processed
SAMPLE_T	Output	INT	I, Q, M, D, L, P, constant	Sample time
OUT	Output	REAL	I, Q, M, D, L, P, constant	The result of the LEAD_LAG operation
ERR_CODE	Output	WORD	E, A, M, D, L, P	Returns a value of W#16#0000 if the instruction executes without error; see Error Information for values other than W#16#0000
LD_TIME	Static	REAL	I, Q, M, D, L, P, constant	Lead time in minutes
LG_TIME	Static	REAL	I, Q, M, D, L, P, constant	Lag time in minutes
GAIN	Static	REAL	I, Q, M, D, L, P, constant	Gain as % / % (the ratio of the change in output to the change in input as a steady state).

Parameter	Declaration	Data Type	Memory Area	Description
PREV_IN	Static	REAL	I, Q, M, D, L, P, constant	Previous input
PREV_OUT	Static	REAL	I, Q, M, D, L, P, constant	Previous output

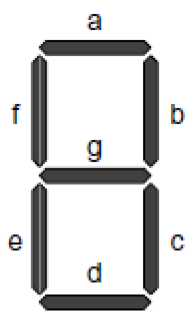
Error Information

If *GAIN* is less than or equal to 0, the function block is not executed. The signal state of *ENO* is set to 0 and *ERR_CODE* is set equal to W#16#0009.

12.1.2 FC 93 - SEG - Seven Segment Decoder

Description

The Seven Segment Decoder SEG function converts each of the four hexadecimal digits in the designated source data word *IN* into four equivalent 7-segment display codes and writes it to the output destination double word *OUT*. The Figure below shows the relationship between the input hex digits and the output bit patterns.



Parameters

Digit	– g f e d c b a	Display
0 0 0 0	0 0 1 1 1 1 1 1	0
0 0 0 1	0 0 0 0 0 1 1 0	1
0 0 1 0	0 1 0 1 1 0 1 1	2
0 0 1 1	0 1 0 0 1 1 1 1	3
0 1 0 0	0 1 1 0 0 1 1 0	4
0 1 0 1	0 1 1 0 1 1 0 1	5
0 1 1 0	0 1 1 1 1 1 0 1	6
0 1 1 1	0 0 0 0 0 1 1 1	7
1 0 0 0	0 1 1 1 1 1 1 1	8
1 0 0 1	0 1 1 0 0 1 1 1	9
1 0 1 0	0 1 1 1 0 1 1 1	A
1 0 1 1	0 1 1 1 1 1 0 0	b
1 1 0 0	0 0 1 1 1 0 0 1	C
1 1 0 1	0 1 0 1 1 1 1 0	d
1 1 1 0	0 1 1 1 1 0 0 1	E
1 1 1 1	0 1 1 1 0 0 0 1	F

Parameters

Parameter	Declaration	Data Type	Memory Area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input with signal state of 1 activates the box
ENO	Output	BOOL	I, Q, M, D, L	Enable output has a signal state of 1 if the function is executed without error
IN	Input	WORD	I, M, D, P, or constant	Source data word in four hexadecimal digits
OUT	Output	DWORD	Q, M, D, L, P	Destination bit pattern in four bytes

Error Information

This function does not detect any error conditions.

12.1.3 FC 94 - ATH - ASCII to Hex

Description

The ASCII to Hex (ATH) function converts the ASCII character string pointed to by *IN* into packed hexadecimal digits and stores these in the destination table pointed to by *OUT*. Since 8 bits are required for the ASCII character and only 4 bits for the hexadecimal digit, the output word length is only half of the input word length. The ASCII characters are converted and placed into the hexadecimal output in the same order as they are read in. If there is an odd number of ASCII characters, the hexadecimal digit is padded with zeros in the right-most nibble of the last converted hexadecimal digit.

Parameters

Parameter	Declaration	Data Type	Memory Area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input with signal state of 1 activates the box
ENO	Output	BOOL	I, Q, M, D, L	Enable output has a signal state of 1 if the function is executed without error
IN	Input	Pointer *)	I, Q, M, D, L	Points to the starting location of an ASCII string
N	Input	INT	I, Q, M, L, P	Number of ASCII input characters to be converted
RET_VAL	Output	WORD	I, Q, M, D, L, P	Returns a value of W#16#0000 if the instruction executes without error; see Error Information for values other than W#16#0000
OUT	Output	Pointer *)	Q, M, D, L	Points to the starting location of the table

*) Double word pointer format for area-crossing register indirect addressing

Error Information

If any ASCII character is found to be invalid, it is converted as 0. The signal state of *ENO* is set to 0 and *RET_VAL* is set equal to W#16#0007.

12.1.4 FC 95 - HTA - Hex to ASCII

Description

The Hex to ASCII (HTA) function converts packed hexadecimal digits, pointed to by *IN*, and stores them in the destination string pointed to by *OUT*. Since 8 bits are required for the character and only 4 bits for the hex digit, the output word length is two times that of the input word length. Each nibble of the hexadecimal digit is converted into a character in the same order as they are read in (left-most nibble of a hexadecimal digit is converted first, followed by the right-most nibble of that same digit).

Parameters

Parameter	Declaration	Data Type	Memory Area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input with signal state of 1 activates the box
ENO	Output	BOOL	I, Q, M, D, L	Enable output has a signal state of 1 if the function is executed without error
IN	Input	Pointer *)	I, Q, M, D	Points to the starting location of the hexadecimal digit string
N	Input	WORD	I, Q, M, L, P	Number of hex input bytes to be converted
OUT	Output	Pointer *)	Q, M, D, L	Points to the starting location of the destination table
*) Double word pointer format for area-crossing register indirect addressing				

Error Information

This function does not detect any error conditions.

12.1.5 FC 96 - ENCO - Encode Binary Position

Description

The Encode Binary Position ENCO function converts the contents of *IN* to the 5-bit binary number corresponding to the bit position of the right-most set bit in *IN* and returns the result as the function's value. If *IN* is either 0000 0001 or 0000 0000, a value of 0 is returned.

Parameters

Parameter	Declaration	Data Type	Memory Area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input with signal state of 1 activates the box
ENO	Output	BOOL	I, Q, M, D, L	Enable output has a signal state of 1 if the function is executed without error
IN	Input	DWORD	I, M, D, L, P, or constant	Value to be encoded
RET_VAL	Input	INT	Q, M, D, L, P	Value returned (contains 5-bit binary number)

Error Information This function does not detect any error conditions.

12.1.6 FC 97 - DECO - Decode Binary Position

Description The Decode Binary Position DECO function converts a 5-bit binary number (0 – 31) from input *IN* to a value by setting the corresponding bit position in the function's return value. If *IN* is greater than 31, a modulo 32 operation is performed to get a 5-bit binary number.

Parameters

Parameter	Declaration	Data Type	Memory Area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input with signal state of 1 activates the box
ENO	Output	BOOL	I, Q, M, D, L	Enable output has a signal state of 1 if the function is executed without error
IN	Input	WORD	I, M, D, L, P, constant	Variable to decode
RET_VAL	Output	DWORD	Q, M, D, L, P	Value returned

Error Information This function does not detect any error conditions.

12.1.7 FC 98 - BCDCPL - Tens Complement

Description The Tens Complement BCDCPL function returns the Tens complement of a 7-digit BCD number *IN*. The mathematical formula for this operation is the following:

$$\begin{array}{r} 10000000 \ (in \ BCD) \\ - \ 7\text{-}digit \ BCD \ value \\ \hline = \ Tens \ complement \ value \ (in \ BCD) \end{array}$$

Parameters

Parameter	Declaration	Data Type	Memory Area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input with signal state of 1 activates the box
ENO	Output	BOOL	I, Q, M, D, L	Enable output has a signal state of 1 if the function is executed without error
IN	Input	DWORD	I, M, D, L, P, constant	7-digit BCD number
RET_VAL	Output	DWORD	Q, M, D, L, P	Value returned

Error Information This function does not detect any error conditions.

12.1.8 FC 99 - BITSUM - Sum Number of Bits

Description

The Sum Number of Bits BITSUM function counts the number of bits that are set to a value of 1 in the input *IN* and returns this as the function's value.

Parameter

Parameter	Deklaration	Datentyp	Speicherbereich	Beschreibung
EN	Input	BOOL	I, Q, M, D, L	Enable input with signal state of 1 activates the box
ENO	Output	BOOL	I, Q, M, D, L	Enable output has a signal state of 1 if the function is executed without error
IN	Input	DWORD	I, M, D, L, P, constant	Variable to count bits in
RET_VAL	Output	INT	Q, M, D, L, P	Value returned

Error Information

This function does not detect any error conditions.

12.1.9 FC 105 - SCALE - Scaling Values

Description

The Scaling Values SCALE function takes an integer value *IN* and converts it to a real value in engineering units scaled between a low and a high limit *LO_LIM* and *HI_LIM*. The result is written to *OUT*. The SCALE function uses the equation:

$$OUT = [((FLOAT\ (IN) - K1) / (K2 - K1)) \cdot (HI\_LIM - LO\_LIM)] + LO\_LIM$$

The constants K1 and K2 are set based upon whether the input value is *BIPOLAR* or *UNIPOLAR*.

- *BIPOLAR*:
  - The input integer value is assumed to be between -27648 and 27648, therefore,
    K1 = -27648,0 and K2 = +27648,0.
- *UNIPOLAR*:
  - The input integer value is assumed to be between 0 and 27648, therefore,
    K1 = 0,0 and K2 = +27648,0.

If the input integer value is greater than K2, the output *OUT* is clamped to *HI_LIM*, and an error is returned. If the input integer value is less than K1, the output *OUT* is clamped to *LO_LIM*, and an error is returned. Reverse scaling can be obtained by programming *LO_LIM* > *HI_LIM*. With reverse scaling, the value of the output decreases as the value of the input increases.

Parameters

Parameter	Declaration	Data Type	Memory Area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input with signal state of 1 activates the box
ENO	Output	BOOL	I, Q, M, D, L	Enable output has a signal state of 1 if the function executed without error
IN	Input	INT	I, Q, M, D, L, constant	The input value to be scaled to a REAL value in engineering units
HI_LIM	Input	REAL	I, Q, M, D, L, P, constant	Upper limit in engineering units
LO_LIM	Input	REAL	I, Q, M, D, L, P, constant	Lower limit in engineering units
BIPOLAR	Input	BOOL	I, Q, M, D, L	A signal state of 1 indicates the input value is bipolar, a signal state of "0" indicates unipolar
OUT	Output	REAL	I, Q, M, D, L, P	The result of the scale conversion
RET_VAL	Input	WORD	I, Q, M, D, L, P	Returns a value of W#16#0000 if the instruction executes without error; see Error Information for values other than W#16#0000

Error Information

If the input integer value is greater than K2, the output *OUT* is clamped to *HI_LIM*, and an error is returned. If the input integer value is less than K1, the output *OUT* is clamped to *LO_LIM*, and an error is returned. The signal state of *ENO* is set to 0 and *RET_VAL* is set equal to W#16#0008.

12.1.10 FC 106 - UNSCALE - Unscaling Values

Description

The Unscaling Values UNSCALE function takes a real input value *IN* in engineering units scaled between a low and a high limit *LO_LIM* and *HI_LIM* and converts it to an integer value. The result is written to *OUT*. The UNSCALE function uses the equation:

$$OUT = [((IN - LO\_LIM) / (HI\_LIM - LO\_LIM)) \cdot (K2 - K1)] + K1$$

and sets the constants K1 and K2 based upon whether the input value is *BIPOLAR* or *UNIPOLAR*.

- *BIPOLAR*:
  – The input integer value is assumed to be between -27648 and 27648, therefore,
    K1 = -27648,0 and K2 = +27648,0.
- *UNIPOLAR*:
  – The input integer value is assumed to be between 0 and 27648, therefore,
    K1 = 0,0 and K2 = +27648,0.

If the input value is outside the *LO_LIM* and *HI_LIM* range, the output *OUT* is clamped to the nearer of either the low limit or the high limit of the specified range for its type (*BIPOLAR* or *UNIPOLAR*), and an error is returned.

Parameters

Parameter	Declaration	Data Type	Memory Area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input with signal state of 1 activates the box
ENO	Output	BOOL	I, Q, M, D, L	Enable output has a signal state of 1 if the function executed without error
IN	Input	REAL	I, Q, M, D, L, P, constant	The input value to be unscaled to an integer value
HI_LIM	Input	REAL	I, Q, M, D, L, P, constant	Upper limit in engineering units
LO_LIM	Input	REAL	I, Q, M, D, L, P, constant	Lower limit in engineering units
BIPOLAR	Input	BOOL	I, Q, M, D, L	A signal state of 1 indicates the input value is bipolar and a signal state of "0" indicates unipolar
OUT	Output	INT	I, Q, M, D, L, P	The result of the scale conversion
RET_VAL	Output	WORD	I, Q, M, D, L, P	Returns a value of W#16#0000 if the instruction executes without error; see Error Information for values other than W#16#0000

Error Information

If the input real value is outside the *LO_LIM* and *HI_LIM* range the output *OUT* is clamped to the nearer of either the low limit or the high limit of the specified range for its type (*BIPOLAR* or *UNIPOLAR*), and an error is returned. The signal state of *ENO* is set to 0 and *RET_VAL* is set equal to W#16#0008.

12.1.11 FC 108 - RLG_AA1 - Issue an Analog Value

Description

The function RLG_AA1 (Issue an Analog Value) transforms an Input Value *XE* (Fixed Point Number) into an output value for an analog output module in accordance with the nominal range between *OGR* and *UGR*. If the nominal range is exceeded, an error message is displayed.

Parameter	Datentyp	Speicherbereich	Beschreibung
XE	INT	I, Q, M, L, D, constant	Input value *XE* as a fixed point number
BG	INT	I, Q, M, L, D, constant	Specify the module address

Parameter	Datentyp	Speicherbereich	Beschreibung
KNKT	WORD	I, Q, M, L, D, con-stant	Channel number KN Channel type KT
OGR	INT	I, Q, M, L, D, con-stant	Upper limit of the input value *XE*
UGR	INT	I, Q, M, L, D, con-stant	Lower limit of the input value *XE*
FEH	BOOL	I, Q, M, L, D	Error bit
BU	BOOL	I, Q, M, L, D	Range excess

Differences between S5 and S7

- The BG parameter
  – There is no address check. The range is the whole P area.

This function is only used to convert the FB251 of an existing S5 program of an S5 CPU 941 to 944 to a function of an S7 program for the S7-400 programmable controller.

12.1.12 FC 109 - RLG_AA2 - Write Analog Value 2

Description

The function RLG_AA2 (Issue an Analog Value) transforms an Input Value *XE* (Floating Point Number) into an output value for an analog output module in accordance with the nominal range between *OGR* and *UGR*. If the nominal range is exceeded, an error message is displayed.

Parameter	Data Type	Memory Area	Description
XE	REAL	I, Q, M, L, D, con-stant	Input value *XE* as a floating point number
BG	INT	I, Q, M, L, D, con-stant	Specify the module address
P_Q	WORD	I, Q, M, L, D, con-stant	Peripheriebereich normal/erweitert
KNKT	WORD	I, Q, M, L, D, con-stant	Channel number KN Channel type KT
OGR	REAL	I, Q, M, L, D, con-stant	Upper limit of the input value *XE*
UGR	REAL	I, Q, M, L, D, con-stant const.	Lower limit of the input value *XE*

Parameter	Data Type	Memory Area	Description
FEH	BOOL	I, Q, M, L, D	Error bit
BU	BOOL	I, Q, M, L, D	Range excess

Differences between S5 and S7

- The BG parameter
  – There is no address check. The range is the whole P area.
- In S7, no value is assigned to the parameter *P_Q*.
- A process image of the S5 I/O areas P/Q/IM3/IM4 is made in the S7 I/O area. You must assign the I/O area in the configuration table.

This function is only used to convert the FB41 of an existing S5 program of an S5 CPU 928B, 945 or 948 to a function of an S7 program for the S7-400 programmable controller.

12.1.13 FC 110 - PER_ET1 - Read/Write Ext. Per. 1

Description

The function PER_ET1 (Reading and Writing for Expanded Peripheries) transfers either a peripheral area into a CPU-internal area or vice-versa (depending on the parameter assignment). In this way, input bytes can be read from, and output bytes written to, the expanded I/O. If a data block is selected as an internal area, the block must have been set up by the user with the necessary length prior to calling up the function.

Parameter	Data Type	Memory Area	Description
PBIB	WORD	I, Q, M, L, D, constant	Specify the areas to be processed
ANF	INT	I, Q, M, L, D, constant	Beginning of the internal area
ANEN	WORD	I, Q, M, L, D, constant	Beginning and end of the block on the interface module
E_A	BOOL	I, Q, M, L, D, constant	Transfer direction
PAFE	BOOL	E, A, M, L, D	Parameter assignment error

Differences between S5 and S7

- The *PBIB* parameter
  – In S7, the I/O area is assigned values as follows:

	S5		S7
P area	0 to 255	P area	0 to 255

Q area	0 to 255	P area	256 to 511
IM3 area	0 to 255	P area	512 to 767
IM4 area	0 to 255	P area	768 to 1023
DB	0 to 255	DB	0 to 255
DX	0 to 255	DB	256 to 511
M	0 to 199	M	0 to 199
S		Error message: "Invalid range"	

- A process image of the S5 I/O areas P/Q/IM3/IM4 is made in the S7 I/O area. You must assign the I/O area in the configuration table.

> *This function is only used to convert the FB196 of an existing S5 program of an S5 CPU 95U, 103, 941 to 944, 945, 928B, 948 to a function of an S7 program for the S7-300/400 programmable controller.*

12.1.14 FC 111 - PER_ET2 - Read/Write Ext. Per. 2

Description

The function PER_ET2 (Reading and Writing for Expanded Peripheries) transfers either a peripheral area into a CPU-internal area or vice-versa (depending on the parameter assignment). In this way, input bytes can be read from, and output bytes written to, the expanded I/O. If a data block is selected as an internal area, the block must have been set up by the user with the necessary length prior to calling up the function.

Differences between S5 and S7:

- The *PBIB* parameter (defined in DB)
  – In S7, the I/O area is assigned values as follows:

	S5		S7
P area	0 to 255	P area	0 to 255
Q area	0 to 255	P area	256 to 511
IM3 area	0 to 255	P area	512 to 767
IM4 area	0 to 255	P area	768 to 1023
DB	0 to 255	DB	0 to 255
DX	0 to 255	DB	256 to 511
M	0 to 199	M	0 to 199
S		Error message: "Invalid range"	

- A process image of the S5 I/O areas P/Q/IM3/IM4 is made in the S7 I/O area. You must assign the I/O area in the configuration table.

This function is only used to convert the FB197 of an existing S5 program of an S5 CPU 95U, 103, 941 to 944, 945, 928B, 948 to a function of an S7 program for the S7-300/400 programmable controller.

12.2 IEC

12.2.1 Date and time as complex data types

Actual parameters for DATE_AND_TIME

The DATE_AND_TIME data type is a complex data type like ARRAY, STRING, and STRUCT. The permissible memory areas for complex data types are the data block (DB) and local data (L stack) areas. If you use the data type DATE_AND_TIME as formal parameter in an instruction, due to the complex data type you can specify only one of the following formats:

- A block-specific symbol from the variable declaration table for a specific block
- A symbolic name for a data block, such as e.g. "DB_sys_info.System_Time", made up of the following parts:
  - A name defined in the symbol table for the number of the data block (e.g. "DB_sys_info" for DB 5)
  - A name defined within the data block for the DATE_AND_TIME element (e.g. "Time" for a variable of data type DATE_AND_TIME contained in DB 5)

You cannot pass constants as actual parameters to formal parameters of the complex data types, including DATE_AND_TIME. Also, you cannot pass absolute addresses as actual parameters to DATE_AND_TIME.

12.2.2 FC 1 - AD_DT_TM - Add duration to instant of time

Description

The function FC 1 adds a duration *D* (time) to an instant of time *T* (date and time) and provides a new instant of time (date and time) as the result. The instant of time *T* must be in the range DT#1990-01-01-00:00:00.000 ... DT#2089-12-31-23:59:59.999. The function does not check the input parameters. If the result of the addition is not within the valid range, the result is limited to the corresponding value and the binary result (BR) bit of the status word is set to "0".

Parameter

Parameter	Declaration	Data type	Memory area	Description
T*	INPUT	DATE_AND_TIME	D, L	Instant of time in format DT
D	INPUT	TIME	I, Q, M, D, L Constant	Duration in Format TIME

Parameter	Declaration	Data type	Memory area	Description
RET_VAL*	OUTPUT	DATE_AND_TIME	D, L	Sum in format DT
*) You can assign only a symbolically defined variable for the parameter.				

12.2.3 FC 2 - CONCAT - Concatenate two STRING variables

Description

The function FC 2 concatenates two STRING variables together to form one string. If the resulting string is longer than the variable given at the output parameter, the result string is limited to the maximum set length and the BR bit is set to "0".

Parameter

Parameter	Declaration	Data type	Memory area	Description
IN1*	INPUT	STRING	D, L	Input variable in format STRING
IN2*	INPUT	STRING	D, L	Input variable in format STRING
RET_VAL*	OUTPUT	STRING	D, L	Concatenated string
*) You can assign only a symbolically defined variable for the parameter.				

12.2.4 FC 3 - D_TOD_DT - Combine DATE and TIME_OF_DAY

Description

The function FC 3 combines the data formats DATE and TIME_OF_DAY (TOD) and converts these formats to the data format DATE_AND_TIME (DT). The input value *IN1* must be in the range DATE#1990-01-01 ... DATE#2089-12-31. The function does not check the input parameters and does not report any errors.

Parameter

Parameter	Declaration	Data type	Memory area	Description
IN1	INPUT	DATE	I, Q, M, D, L Constant	Input variable in format DATE
IN2	INPUT	TIME_OF_DAY	I, Q, M, D, L Constant	Input variable in format TOD
RET_VAL*	OUTPUT	DATE_AND_TIME	D, L	Return value in format DT
*) You can assign only a symbolically defined variable for the parameter.				

12.2.5 FC 4 - DELETE - Delete in a STRING variable

Description

The function FC 4 deletes a number of characters *L* from the character at position *P* (inclusive) in a string. The function does not report any errors.

- If *L* and/or *P* are equal to zero or if *P* is greater than the current length of the input string, the input string is returned.
- If the sum of *L* and *P* is greater than the input string, the string is deleted up to the end.
- If *L* and/or *P* is negative, a blank string is returned and the BR bit is set to "0".

Parameter

Parameter	Declaration	Data type	Memory area	Description
IN*	INPUT	STRING	D, L	STRING variable to be deleted in
L	INPUT	INT	I, Q, M, D, L Constant	Number of characters to be deleted
P	INPUT	INT	I, Q, M, D, L Constant	Position of 1. character to be deleted
RET_VAL*	OUTPUT	STRING	D, L	Result string
*) You can assign only a symbolically defined variable for the parameter.				

12.2.6 FC 5 - DI_STRNG - Convert DINT to STRING

Description

The function FC 5 converts a variable in DINT data format to a string. The string is shown preceded by a sign. If the variable given at the return parameter is too short, no conversion takes place and the BR bit is set to "0".

Parameter

Parameter	Declaration	Data type	Memory area	Description
I	INPUT	DINT	I, Q, M, D, L Constant	Input value
RET_VAL*	OUTPUT	STRING	D, L	Result string
*) You can assign only a symbolically defined variable for the parameter.				

12.2.7 FC 6 - DT_DATE - Extract DATE from DT

Description

The function FC 6 extracts the data format DATE from the format DATE_AND_TIME. DATE value is between the limits DATE#1990-1-1 and DATE#2089-12-31. The function does not report any errors.

Parameter

Parameter	Declaration	Data type	Memory area	Description
IN*	INPUT	DATE_AND_TIME	D, L	Input variable in format DT
RET_VAL	OUTPUT	DATE	I, Q, M, D, L	Return value in format DATE
*) You can assign only a symbolically defined variable for the parameter.				

12.2.8 FC 7 - DT_DAY - Extract day of the week from DT

Description

The function FC 7 extracts the day of the week from the format DATE_AND_TIME. The function does not report any errors. The day of the week is returned as INTEGER value.

- 1: Sunday
- 2: Monday
- 3: Tuesday
- 4: Wednesday
- 5: Thursday
- 6: Friday
- 7: Saturday

Parameter

Parameter	Declaration	Data type	Memory area	Description
IN*	INPUT	DATE_AND_TIME	D, L	Input variable in format DT
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Return value in format INT
*) You can assign only a symbolically defined variable for the parameter.				

12.2.9 FC 8 - DT_TOD - Extract TIME_OF_DAY from DT

Description

The function FC 8 extracts the data format TIME_OF_DAY from the format DATE_AND_TIME. The function does not report any errors.

Parameter

Parameter	Declaration	Data type	Memory area	Description
IN*	INPUT	DATE_AND_TIME	D, L	Input variable in format DT
RET_VAL	OUTPUT	TIME_OF_DAY	I, Q, M, D, L	Return value in format TOD
*) You can assign only a symbolically defined variable for the parameter.				

12.2.10 FC 9 - EQ_DT - Compare DT for equality

Description

The function FC 9 compares the contents of two variables in the data type format DATE_AND_TIME to determine if they are equal and outputs the result of the comparison as a return value. The return value has the signal state "1" if the time at parameter *DT1* is the same as the time at parameter *DT2*. The function does not report any errors.

Parameter

Parameter	Declaration	Data type	Memory area	Description
DT1*	INPUT	DATE_AND_TIME	D, L	Input variable in format TD
DT2*	INPUT	DATE_AND_TIME	D, L	Input variable in format TD
RET_VAL	OUTPUT	BOOL	I, Q, M, D, L	Comparison result
*) You can assign only a symbolically defined variable for the parameter.				

12.2.11 FC 10 - EQ_STRNG - Compare STRING for equal

Description

The function FC 10 compares the contents of two variables in the format STRING to determine if they are equal and outputs the result of the comparison as a return value. The return value has the signal state "1" if the string at parameter *S1* is the same as the string at parameter *S2*. The function does not report any errors.

Parameter

Parameter	Declaration	Data type	Memory area	Description
S1*	INPUT	STRING	D, L	Input variable in format STRING
S2*	INPUT	STRING	D, L	Input variable in format STRING
RET_VAL	OUTPUT	BOOL	I, Q, M, D, L	Comparison result
*) You can assign only a symbolically defined variable for the parameter.				

12.2.12 FC 11 - FIND - Find in a STRING variable

Description

The function FC 11 provides the position of the second string *IN2* within the first string *IN1*. The search starts on the left; the first occurrence of the string is reported. If the second string is not found in the first, zero is returned. The function does not report any errors.

Parameter

Parameter	Declaration	Data type	Memory area	Description
IN1*	INPUT	STRING	D, L	STRING variable to be searched in
IN2*	INPUT	STRING	D, L	STRING variable to be found
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Position of the string found
*) You can assign only a symbolically defined variable for the parameter.				

12.2.13 FC 12 - GE_DT - Compare DT for greater than or equal

Description

The function FC 12 compares the contents of two variables in the data format DATE_AND_TIME to determine if one is greater or equal to the other and outputs the result of the comparison as a return value. The return value has the signal state "1" if the time at parameter *DT1* is greater (younger) than the time at parameter *DT2* or if both instants of time are the same. The function does not report any errors.

Parameter

Parameter	Declaration	Data type	Memory area	Description
DT1*	INPUT	DATE_AND_TIME	D, L	Input variable in format TD
DT2*	INPUT	DATE_AND_TIME	D, L	Input variable in format TD
RET_VAL	OUTPUT	BOOL	I, Q, M, D, L	Comparison result
*) You can assign only a symbolically defined variable for the parameters.				

12.2.14 FC 13 - GE_STRNG - Compare STRING for greater than or equal

Description

The function FC 13 compares the contents of two variables in the data format STRING to determine if one is greater or equal to the other and outputs the result of the comparison as a return value. The return value has the signal state "1" if the string at parameter *S1* is greater than or equal to the string at parameter *S2*. The characters are compared by their ASCII code (e.g. 'a' is greater than 'A'), starting from the left. The first character to be different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the longer string is considered as greater. The function does not report any errors.

Parameter

Parameter	Declaration	Data type	Memory area	Description
S1*	INPUT	STRING	D, L	Input variable in format STRING
S2*	INPUT	STRING	D, L	Input variable in format STRING
RET_VAL	OUTPUT	BOOL	I, Q, M, D, L	Comparison result
*) You can assign only a symbolically defined variable for the parameter.				

12.2.15 FC 14 - GT_DT - Compare DT for greater than

Description

The function FC 14 compares the contents of two variables in the data format DATE_AND_TIME to determine if one is greater to the other and outputs the result of the comparison as a return value. The return value has the signal state "1" if the time at parameter *DT1* is greater (younger) than the time at parameter *DT2*. The function does not report any errors.

Parameter

Parameter	Declaration	Data type	Memory area	Description
DT1*	INPUT	DATE_AND_TIME	D, L	Input variable in format TD
DT2*	INPUT	DATE_AND_TIME	D, L	Input variable in format TD
RET_VAL	OUTPUT	BOOL	I, Q, M, D, L	Comparison result
*) You can assign only a symbolically defined variable for the parameter.				

12.2.16 FC 15 - GT_STRNG - Compare STRING for greater than

Description

The function FC 15 compares the contents of two variables in the data format STRING to find out if the first is greater than the other and outputs the result of the comparison as a return value. The return value has the signal state "1" if the string at parameter *S1* is greater than the string at parameter *S2*. The characters are compared by their ASCII code (e.g. 'a' is greater than 'A'), starting from the left. The first character to be different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the longer string is considered as greater. The function does not report any errors.

Parameter

Parameter	Declaration	Data type	Memory area	Description
S1*	INPUT	STRING	D, L	Input variable in format STRING
S2*	INPUT	STRING	D, L	Input variable in format STRING

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	OUTPUT	BOOL	I, Q, M, D, L	Comparison result
*) You can assign only a symbolically defined variable for the parameter.				

12.2.17 FC 16 - I_STRNG - Convert INT to STRING

Description

The function FC 16 converts a variable in DINT data format to a string. The string is shown preceded by a sign. If the variable given at the return parameter is too short, no conversion takes place and the BR bit is set to "0".

Parameter

Parameter	Declaration	Data type	Memory area	Description
I	INPUT	INT	I, Q, M, D, L Constant	Input value
RET_VAL*	OUTPUT	STRING	D, L	Result string
*) You can assign only a symbolically defined variable for the parameter.				

12.2.18 FC 17 - INSERT - Insert in a STRING variable

Description

The function FC 17 inserts a string at parameter *IN2* into the string at parameter *IN1* after the character at position *P*.

- If *P* equals zero, the second string is inserted before the first string.
- If *P* is greater than the current length of the first string, the second string is appended to the first.
- If *P* is negative, a blank string is output and the BR bit is set to "0". The binary result bit is also set to "0" if the resulting string is longer than the variable given at the output parameter; in this case the result string is limited to the maximum set length.

Parameter

Parameter	Declaration	Data type	Memory area	Description
IN1*	INPUT	STRING	D, L	STRING variable to be inserted into
IN2*	INPUT	STRING	D, L	STRING variable to be inserted
P	INPUT	INT	I, Q, M, D, L Constant	Insert position
RET_VAL*	OUTPUT	STRING	D, L	Result string
*) You can assign only a symbolically defined variable for the parameter.				

12.2.19 FC 18 - LE_DT - Compare DT for smaller than or equal

Description

The function FC 18 compares the contents of two variables in the format DATE_AND_TIME to determine if one is smaller or equal to the other and outputs the result of the comparison as a return value. The return value has the signal state "1" if the time at parameter *DT1* is smaller (older) than the time at parameter *DT2* or if both instants of time are the same. The function does not report any errors.

Parameter

Parameter	Declaration	Data type	Memory area	Description
DT1*	INPUT	DATE_AND_TIME	D, L	Input variable in format TD
DT2*	INPUT	DATE_AND_TIME	D, L	Input variable in format TD
RET_VAL*	OUTPUT	BOOL	I, Q, M, D, L	Comparison result
*) You can assign only a symbolically defined variable for the parameter.				

12.2.20 FC 19 - LE_STRNG - Compare STRING for smaller then or equal

Description

The function FC 19 compares the contents of two variables in the format STRING to determine if one is smaller or equal to the other and outputs the result of the comparison as a return value. The return value has the signal state "1" if the string at parameter *S1* is smaller than or equal to the string at parameter *S2*. The characters are compared by their ASCII code (e.g. 'A' smaller than 'a'), starting from the left. The first character to be different decides the result of the comparison. If the left part of the longer character string and the shorter character string are the same, the shorter string is smaller. The function does not report any errors.

Parameter

Parameter	Declaration	Data type	Memory area	Description
S1*	INPUT	STRING	D, L	Input variable in format STRING
S2*	INPUT	STRING	D, L	Input variable in format STRING
RET_VAL	OUTPUT	BOOL	I, Q, M, D, L	Comparison result
*) You can assign only a symbolically defined variable for the parameter.				

12.2.21 FC 20 - LEFT - Left part of a STRING variable

Description

The function FC 20 provides the first *L* characters of a string.

- If *L* is greater than the current length of the STRING variable, the input value is returned.
- With *L* = 0 and with a blank string as the input value, a blank string is returned.
- If *L* is negative, a blank string is returned and the BR bit of the status word is set to "0".

Parameter

Parameter	Declaration	Data type	Memory area	Description
IN*	INPUT	STRING	D, L	Input variable in format STRING
L	INPUT	INT	I, Q, M, D, L Constant	Length of the left character string
RET_VAL*	OUTPUT	STRING	D, L	Output variable in format STRING
*) You can assign only a symbolically defined variable for the parameter.				

12.2.22 FC 21 - LEN - Length of a STRING variable

Description

A STRING variable contains two lengths:

- Maximum length
  – It is given in square brackets when the variables are being defined.
- Current length
  – This is the number of currently valid characters.

The current length is smaller or equal to the maximum length. The number of bytes occupied by a string is 2 greater than the maximum length. The function FC 21 outputs the current length of a string (number of valid characters) as a return value. A blank string (' ') has the length zero. The maximum length is 254. The function does not report any errors.

Parameter

Parameter	Declaration	Data type	Memory area	Description
S*	INPUT	STRING	D, L	Input variable in format STRING
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Number of current charac-ters
*) You can assign only a symbolically defined variable for the parameter.				

12.2.23 FC 22 - LIMIT

Description

The function FC 22 limits the number value of a variable to limit values which can have parameters assigned.

- Variables of the data types INT, DINT, and REAL are permitted as input values.
- All variables with parameters assigned must be of the same data type.
- The variable type is recognized by the ANY pointer.
- *MN* may not be greater as *MX*.
- The output value remains unchanged and the BR bit is set to "0" if:
  - a variable with parameters assigned has an invalid data type.
  - all variables with parameters assigned do not have the same data type.
  - the lower limit value is greater than the upper limit value.
  - a REAL variable does not represent a valid floating-point number.

Parameter

Parameter	Declaration	Data type	Memory area	Description
MN	INPUT	ANY	I, Q, M, D, L	Lower limit
IN	INPUT	ANY	I, Q, M, D, L	Input variable
MX	INPUT	ANY	I, Q, M, D, L	Upper limit
RET_VAL	OUTPUT	ANY	I, Q, M, D, L	Limited output variable

12.2.24 FC 23 - LT_DT - Compare DT for smaller than

Description

The function FC 23 compares the contents of two variables in the format DATE_AND_TIME to determine if one is smaller to the other and outputs the result of the comparison as a return value. The return value has the signal state "1" if the time at parameter *DT1* is smaller (older) than the time at parameter *DT2*. The function does not report any errors.

Parameter

Parameter	Declaration	Data type	Memory area	Description
DT1*	INPUT	DATE_AND_TIME	D, L	Input variable in format TD
DT2*	INPUT	DATE_AND_TIME	D, L	Input variable in format TD
RET_VAL	OUTPUT	BOOL	I, Q, M, D, L	Comparison result
*) You can assign only a symbolically defined variable for the parameter.				

12.2.25 FC 24 - LT_STRNG - Compare STRING for smaller

Description

The function FC 24 compares the contents of two variables in the format STRING to determine if one is smaller to the other and outputs the result of the comparison as a return value. The return value has the signal state "1" if the string at parameter *S1* is smaller than the string at parameter *S2*. The characters are compared by their ASCII code (e.g. 'A' smaller than 'a'), starting from the left. The first character to be different decides the result of the comparison. If the left part of the longer character string and the shorter character string are the same, the shorter string is smaller. The function does not report any errors.

Parameter

Parameter	Declaration	Data type	Memory area	Description
S1*	INPUT	STRING	D, L	Input variable in format STRING
S2*	INPUT	STRING	D, L	Input variable in format STRING
RET_VAL	OUTPUT	BOOL	I, Q, M, D, L	Comparison result
*) You can assign only a symbolically defined variable for the parameter.				

12.2.26 FC 25 - MAX - Select maximum

Description

The function FC 25 selects the largest of three numerical variable values.

- Variables of the data types INT, DINT, and REAL are permitted as input values.
- All variables with parameters assigned must be of the same data type.
- The variable type is recognized by the ANY pointer.
- The output value remains unchanged and the BR bit is set to "0" if:
  - a variable with parameters assigned has an invalid data type.
  - all variables with parameters assigned do not have the same data type.
  - a REAL variable does not represent a valid floating-point number.

Parameter

Parameter	Declaration	Data type	Memory area	Description
IN1	INPUT	ANY	I, Q, M, D, L	1. Input value
IN2	INPUT	ANY	I, Q, M, D, L	2. Input value
IN3	INPUT	ANY	I, Q, M, D, L	3. Input value
RET_VAL	OUTPUT	ANY	I, Q, M, D, L	Largest of the input values

The admitted data types INT, DINT and REAL must be entered in the ANY pointer. Such parameters as "MD20" are also admitted, but you must define the corresponding data type of "MD20" in "Symbol".

Example in STL:

```
CALL FC 25
IN1 := P#M 10.0 DINT 1
IN2 := MD20
IN3 := P#DB1.DBX 0.0 DINT 1
RET_VAL := P#M 40.0 DINT 1
= M 0.0
```

12.2.27 FC 26 - MID - Middle part of a STRING variable

Description

The function FC 26 provides the middle part of a string (*L* characters from the character *P* inclusive).

- If the sum of *L* and (*P*-1) exceeds the current length of the STRING variables, a string is returned from the character *P* to the end of the input value.
- In all other cases (*P* is outside the current length, *P* and/or *L* are equal to zero or negative), a blank string is returned and the BR bit is set to "0".

Parameter

Parameter	Declaration	Data type	Memory area	Description
IN*	INPUT	STRING	D, L	Input variable in format STRING
L	INPUT	INT	I, Q, M, D, L Constant	Length of the middle character string
P	INPUT	INT	I, Q, M, D, L Constant	Position of first character
RET_VAL*	OUTPUT	STRING	D, L	Output variable in format STRING

*) You can assign only a symbolically defined variable for the parameter.

12.2.28 FC 27 - MIN - Select minimum

Description

The function FC 27 selects the smallest of three numerical variable values.

- Variables of the data types INT, DINT, and REAL are permitted as input values.
- All variables with parameters assigned must be of the same data type.
- The variable type is recognized by the ANY pointer.
- The output value remains unchanged and the BR bit is set to "0" if:
  – a variable with parameters assigned has an invalid data type.
  – all variables with parameters assigned do not have the same data type.
  – a REAL variable does not represent a valid floating-point number.

Parameter

Parameter	Declaration	Data type	Memory area	Description
IN1	INPUT	ANY	I, Q, M, D, L	1. Input value
IN2	INPUT	ANY	I, Q, M, D, L	2. Input value
IN3	INPUT	ANY	I, Q, M, D, L	3. Input value
RET_VAL	OUTPUT	ANY	I, Q, M, D, L	Smallest of the input values

The admitted data types INT, DINT and REAL must be entered in the ANY pointer. Such parameters as "MD20" are also admitted, but you must define the corresponding data type of "MD20" in "Symbol".

Example in STL:

```
CALL FC 27
IN1 := P#M 10.0 DINT 1
IN2 := MD20
IN3 := P#DB1.DBX 0.0 DINT 1
RET_VAL := P#M 40.0 DINT 1
= M 0.0
```

12.2.29 FC 28 - NE_DT - Compare DT for unequal

Description

The function FC 28 compares the contents of two variables in the format DATE_AND_TIME to determine if they are unequal and outputs the result of the comparison as a return value. The return value has the signal state "1" if the time at parameter *DT1* is unequal the time at parameter *DT2*. The function does not report any errors.

Parameter

Parameter	Declaration	Data type	Memory area	Description
DT1*	INPUT	DATE_AND_TIME	D, L	Input variable in format TD
DT2*	INPUT	DATE_AND_TIME	D, L	Input variable in format TD
RET_VAL	OUTPUT	BOOL	I, Q, M, D, L	Comparison result
*) You can assign only a symbolically defined variable for the parameter.				

12.2.30 FC 29 - NE_STRNG - Compare STRING for unequal

Description

The function FC 29 compares the contents of two variables in the format STRING to determine if they are unequal and outputs the result of the comparison as a return value. The return value has the signal state "1" if the string at parameter *S1* is unequal to the string at parameter *S2*. The function does not report any errors.

Parameter

Parameter	Declaration	Data type	Memory area	Description
S1*	INPUT	STRING	D, L	Input variable in format STRING
S2*	INPUT	STRING	D, L	Input variable in format STRING
RET_VAL	OUTPUT	BOOL	I, Q, M, D, L	Comparison result
*) You can assign only a symbolically defined variable for the parameter.				

12.2.31 FC 30 - R_STRNG - Convert REAL to STRING

Description

The function FC 30 converts a variable in REAL data format to a string.

- The string is shown with 14 digits:
  ±v.nnnnnnnE±xx
  – ±: Sign
  – v: 1 digit before the decimal point
  – n: 7 digits after the decimal point
  – x: 2 exponential digits
- If the variable given at the return parameter is too short or if no valid floating-point number is given at parameter IN, no conversion takes place and the BR bit is set to "0".

Parameter

Parameter	Declaration	Data type	Memory area	Description
IN	INPUT	REAL	I, Q, M, D, L Constant	Input value
RET_VAL*	OUTPUT	STRING	D, L	Result string
*) You can assign only a symbolically defined variable for the parameter.				

12.2.32 FC 31 - REPLACE - Replace in a STRING variable

Description

The function FC 31 replaces a number of characters *L* of the first string *IN1* starting at the character at position *P* (inclusive) with the entire second string *IN2*.

- If *L* is equal to zero and *P* is not equal to zero, the first string is returned.
- If *L* is equal to zero and *P* is equal to zero, the second string is precent to the first string.
- If *L* is not equal to zero and *P* is equal to zero or one, the string is replaced from the 1. character (inclusive).
- If *P* is outside the first string, the second string is appended to the first string.
- If *L* and/or *P* is negative, a blank string is returned and the BR bit is set to "0". The BR bit is also set to "0" if the resulting string is longer than the variable given at the output parameter; in this case the result string is limited to the maximum set length.

Parameter

Parameter	Declaration	Data type	Memory area	Description
IN1*	INPUT	STRING	D, L	STRING variable to be inserted into
IN2*	INPUT	STRING	D, L	STRING variable to be inserted
L	INPUT	INT	I, Q, M, D, L Constant	Number of characters to be replaced
P	INPUT	INT	I, Q, M, D, L Constant	Position of 1. character to be replaced
RET_VAL*	OUTPUT	STRING	D, L	Result string
*) You can assign only a symbolically defined variable for the parameter.				

12.2.33 FC 32 - RIGHT - Right part of a STRING variable

Description

The function FC 32 provides the last *L* characters of a string.

- If *L* is greater than the current length of the STRING variable, the input value is returned.
- With *L* = 0 and with a blank string as the input value, a blank string is returned.
- If *L* is negative, a blank string is returned and the BR bit is set to "0".

Parameter

Parameter	Declaration	Data type	Memory area	Description
IN*	INPUT	STRING	D, L	Input variable in format STRING
L	INPUT	INT	I, Q, M, D, L Constant	Length of the right character string
RET_VAL*	OUTPUT	STRING	D, L	Output variable in format STRING
*) You can assign only a symbolically defined variable for the parameter.				

12.2.34 FC 33 - S5TI_TIM - Convert S5TIME to TIME

Description

The function FC 33 converts the data format S5TIME to the data format TIME. If the result of the conversion is outside the TIME range, the result is limited to the corresponding value and the binary result (BR) bit is set to "0".

Parameter

Parameter	Declaration	Data type	Memory area	Description
IN	INPUT	S5TIME	I, Q, M, D, L Constant	Input variable in format S5TIME
RET_VAL	OUTPUT	TIME	I, Q, M, D, L	Return value in format TIME

12.2.35 FC 34 - SB_DT_DT - Subtract two instants of time

Description

The function FC 34 subtracts two instants of time *DTx* (date and time) and provides a duration (time) as the result. The instants of time *DTx* must be in the range DT#1990-01-01-00:00:00.000 ... DT#2089-12-31-23:59:59.999. The function does not check the input parameters. It is valid:

- With *DT1* > *DT2* the result is positive.
- With *DT1* < *DT2* the result is negative.
- If the result of the subtraction is outside the TIME range, the result is limited to the corresponding value and the binary result (BR) bit is set to "0".

Parameter

Parameter	Declaration	Data type	Memory area	Description
DT1*	INPUT	DATE_AND_TIME	D, L	1. instant of time in format DT
DT2*	INPUT	DATE_AND_TIME	D, L	2. Instant of time in format DT
RET_VAL	OUTPUT	TIME	I, Q, M, D, L	Difference in format TIME
*) You can assign only a symbolically defined variable for the parameter.				

12.2.36 FC 35 - SB_DT_TM - Subtract a duration from a time

Description

The function FC 35 subtracts a duration *D* (TIME) from a time *T* (DT) and provides a new time (DT) as the result. The time *T* must be between DT#1990-01-01-00:00:00.000 and DT#2089-12-31-23:59:59.999. The function does not run an input check. If the result of the subtraction is not within the valid range, the result is limited to the corresponding value and the binary result (BR) bit of the status word is set to "0".

Parameter

Parameter	Declaration	Data type	Memory area	Description
T*	INPUT	DATE_AND_TIME	D, L	Time in format DT
D	INPUT	TIME	E, A, M, D, L, Constant	Duration in format TIME
RET_VAL *	OUTPUT	DATE_AND_TIME	D, L	Difference in format DT
*) You can assign only a symbolically defined variable for the parameter.				

12.2.37 FC 36 - SEL - Binary selection

Description

The function FC 36 selects one of two variable values depending on a switch *G*.

- Variables with all data types which correspond to the data width bit, byte, word, and double word (not data types DT and STRING) are permitted as input values at the parameters *IN0* and *IN1*.
- *IN0*, *IN1* and *RET_VAL* must be of the same data type.
- The output value remains unchanged and the BR bit is set to "0" if:
  - a variable with parameters assigned has an invalid data type.
  - all variables with parameters assigned do not have the same data type.
  - a REAL variable does not represent a valid floating-point number.

Parameter

Parameter	Declaration	Data type	Memory area	Description
G	INPUT	BOOL	I, Q, M, D, L Constant	Selection switch
IN0	INPUT	ANY	I, Q, M, D, L	1. Input value
IN1	INPUT	ANY	I, Q, M, D, L	2. Input value
RET_VAL	OUTPUT	ANY	I, Q, M, D, L	Selected input value

12.2.38 FC 37 - STRNG_DI - Convert STRING to DINT

Description

The function FC 37 converts a string to a variable in DINT data format.

- The first character in the string may be a sign or a number, the characters which then follow must be numbers.
- If the length of the string is equal to zero or greater than 11, or if invalid characters are found in the string, no conversion takes place and the BR bit is set to "0".
- If the result of the conversion is outside the DINT range, the result is limited to the corresponding value and the BR bit is set to "0".

Parameter

Parameter	Declaration	Data type	Memory area	Description
S*	INPUT	STRING	D, L	Input string
RET_VAL	OUTPUT	DINT	I, Q, M, D, L	Result
*) You can assign only a symbolically defined variable for the parameter.				

12.2.39 FC 38 - STRNG_I - Convert STRING to INT

Description

The function FC 38 converts a string to a variable in INT data format.

- The first character in the string may be a sign or a number, the characters which then follow must be numbers.
- If the length of the string is equal to zero or greater than 6, or if invalid characters are found in the string, no conversion takes place and the BR bit is set to "0".
- If the result of the conversion is outside the INT range, the result is limited to the corresponding value and the BR bit is set to "0".

Parameter

Parameter	Declaration	Data type	Memory area	Description
S*	INPUT	STRING	D, L	Input string
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Result
*) You can assign only a symbolically defined variable for the parameter.				

12.2.40 FC 39 - STRNG_R - Convert STRING to REAL

Description

The function FC 39 converts a string to a variable in REAL data format.

- The string must have the following format:
  ±v.nnnnnnnE±xx
  - ±: Sign
  - v: 1 digit before the decimal point
  - n: 7 digits after the decimal point
  - x: 2 exponential digits
- If the length of the string is smaller than 14, or if it is not structured as shown above, no conversion takes place and the BR bit is set to "0".
- If the result of the conversion is outside the REAL range, the result is limited to the corresponding value and the BR bit is set to "0".

Parameter

Parameter	Declaration	Data type	Memory area	Description
S*	INPUT	STRING	D, L	Input string
RET_VAL	OUTPUT	REAL	I, Q, M, D, L	Result
*) You can assign only a symbolically defined variable for the parameter.				

12.2.41 FC 40 - TIM_S5TI - Convert TIME to S5TIME

Description

The function FC 40 converts the data format TIME to the format S5TIME. Here is always rounded down. If the input parameter is greater than the displayable S5TIME format (TIME#02:46:30.000), S5TIME#999.3 is output as result and the binary result (BR) bit is set to "0".

Parameter

Parameter	Declaration	Data type	Memory area	Description
IN	INPUT	TIME	I, Q, M, D, L Constant	Input variable in format TIME
RET_VAL	OUTPUT	S5TIME	I, Q, M, D, L	Return value in format S5TIME

12.3 IO

12.3.1 FB 20 - GETIO - PROFIBUS/PROFINET read all Inputs

Description

With the FB 20 GETIO you consistently read out all inputs of a PROFIBUS DP slave/PROFINET IO device. In doing so, FB 20 calls the SFC 14 DPRD_DAT. If there was no error during the data transmission, the data that have been read are entered in the target area indicated by *INPUTS*. The target area must have the same length that you configured for the selected component. In the case of a PROFIBUS DP slave with a modular structure or with several DP IDs, you can only access the data for one component/DP ID with an FB 20 call each time at the configured start address.

Parameter	Declaration	Data Type	Memory Area	Description
ID	INPUT	DWORD	I, Q, M, D, L constant	■ Low word: logical address of the DP slave/PROFINET IO component (module or submodule) ■ High word: irrelevant
STATUS	OUTPUT	DWORD	I, Q, M, D, L	Contains error information for SFC 14 DPRD_DAT in the form DW#16#40xxxx00
LEN	OUTPUT	INT	I, Q, M, D, L	Amount of data read in bytes
INPUTS	IN_OUT	ANY	I, Q, M, D	Target area for the read data. It must have the same length as the area that you configured for the selected DP slave/PROFINET IO component. Only the data type BYTE is permitted.

Error Information

Chapter 11.1.12 'SFC 14 - DPRD_DAT - Read consistent data' on page 309

12.3.2 FB 21 - SETIO - PROFIBUS/PROFINET write all Outputs

Description

With the FB 21 SETIO you consistently transfer the data from the source area indicated by *OUTPUTS* to the addressed PROFIBUS DP slave/PROFINET IO device, and, if necessary, to the process image (in the case where you have configured the affected address area for the DP standard slave as a consistency area in a process image). In doing so, FB 21 calls the SFC 15 DPWR_DAT. The source area must have the same length that you configured with for the selected component. In the case of a DP standard slave with a modular structure or with several DP IDs, you can only access the data for one component/DP ID with an FB 20 call each time at the configured start address.

Parameter	Declaration	Data Type	Memory Area	Description
ID	INPUT	DWORD	I, Q, M, D, L, constant	■ Low word: logical address of the DP slave/PROFINET IO component ■ (module or submodule) ■ High word: irrelevant
LEN	INPUT	INT	E, A, M, D, L	Irrelevant
STATUS	OUTPUT	DWORD	E, A, M, D, L	Contains error information for SFC 15 DPRD_DAT in the form DW#16#40xxxx00
OUTPUTS	IN_OUT	ANY	E, A, M, D	Source area for the read data to be read. It must have the same length as the area that you configured for the selected DP slave/PROFINET IO component. Only the data type BYTE is permitted.

Error Information

Chapter 11.1.13 'SFC 15 - DPWR_DAT - Write consistent data' on page 311

12.3.3 FB 22 - GETIO_PART - PROFIBUS/PROFINET read a part of the Inputs

Description

With the FB 22 GETIO_PART you consistently read a part of the process image area belonging to a PROFIBUS DP slave/PROFINET IO device. In doing so, FB 22 calls the SFC 81 UBLKMOV.

You must assign a process image partition for inputs to the OB in which FB 22 GETIO_PART is called. Furthermore, before calling FB 22 you must add the associated PROFIBUS DP slave or the associated PROFINET IO device to this process image partition for inputs. If your CPU does not recognize any process image partitions or you want to call FB 22 in OB 1, you must add the associated PROFIBUS DP slave or the associated PROFINET IO device to this process image partition for inputs before calling FB 22. You use the OFFSET and LEN parameters to specify the portion of the process image area to be read for the components addressed by means of their ID. If there was no error during the data transmission, ERROR receives the value FALSE, and the data that have been read are entered in the target area indicated by INPUTS. If there was an error during the data transmission, ERROR receives the value TRUE, and STATUS receives the SFC 81 error information UBLKMOV. If the target area (INPUTS parameter) is smaller than LEN, then as many bytes as INPUTS can accept are transferred. ERROR receives the value FALSE. If the target area is greater than LEN, then the first LEN bytes in the target area are written. ERROR receives the value FALSE.

The FB 22 GETIO_PART does not check the process image for inputs for delimiters between data belonging to different PROFIBUS DP or PROFINET IO components. Because of this, you yourself must make sure that the process image area specified by means of OFFSET and LEN belongs to one component. Reading of data for more than one component cannot be guaranteed for future systems and compromises the transferability to systems from other manufacturers.

Parameter	Declaration	Data Type	Memory Area	Description
ID	INPUT	DWORD	I, Q, M, D, L constant	■ Low word: logical address of the DP slave/ PROFINET IO component (module or submodule) ■ High word: irrelevant
OFFSET	INPUT	INT	I, Q, M, D, L constant	Number of the first byte to be read in the process image for the component (smallest possible value: 0)
LEN	INPUT	INT	I, Q, M, D, L constant	Amount of bytes to be read
STATUS	OUTPUT	DWORD	I, Q, M, D, L	Contains error information for SFC 81 UBLKMOV in the form DW#16#40xxxx00 if *ERROR* = TRUE

Parameter	Declaration	Data Type	Memory Area	Description
ERROR	OUTPUT	BOOL	I, Q, M, D, L	Error display: *ERROR* = TRUE if an error occurs when calling SFC 81 UBLKMOV.
INPUTS	IN_OUT	ANY	I, Q, M, D	Target area for read data: ■ If the target area is smaller than *LEN*, then as many bytes as *INPUTS* can accept are transferred. ERROR receives the value FALSE. ■ If the target area is greater than *LEN*, then the first *LEN* bytes of the target area are written. *ERROR* receives the value FALSE.

Error Information

Chapter 11.1.56 'SFC 81 - UBLKMOV - Copy data area without gaps' on page 380

12.3.4 FB 23 - SETIO_PART - PROFIBUS/PROFINET write a part of the Outputs

Description

With the FB 23 SETIO_PART you transfer data from the source area indicated by *OUTPUTS* into a part of the process image area belonging to a PROFIBUS DP slave/PROFINET IO device. In doing so, FB 23 calls the SFC 81 UBLKMOV.

You must assign a process image partition for outputs to the OB in which FB 23 SETIO_PART is called. Furthermore, before calling FB 23 you must add the associated PROFIBUS DP slave or the associated PROFINET IO device to this process image partition for outputs. If your CPU does not recognize any process image partitions or you want to call FB 23 in OB 1, you must add the associated PROFIBUS DP slave or the associated PROFINET IO device to this process image partition for outputs before calling FB 23. You use the OFFSET and LEN parameters to specify the portion of the process image area to be written for the components addressed by means of their ID. If there was no error during the data transmission, ERROR receives the value FALSE. If there was an error during the data transmission, ERROR receives the value TRUE, and STATUS receives the SFC 81 error information UBLKMOV. If the source area (OUTPUTS parameter) is smaller than LEN, then as many bytes as OUTPUTS contains are transferred. ERROR receives the value FALSE. If the source area is greater than LEN, then the first LEN bytes are transferred from OUTPUTS. ERROR receives the value FALSE.

The FB 23 SETIO_PART does not check the process image for inputs for delimiters between data that belong to different PROFIBUS DP or PROFINET IO components. Because of this, you yourself must make sure that the process image area specified by means of OFFSET and LEN belongs to one component. Writing of data for more than one component cannot be guaranteed for future systems and compromises the transferability to systems from other manufacturers.

Parameter	Declaration	Data Type	Memory Area	Description
ID	INPUT	DWORD	I, Q, M, D, L, constant	■ Low word: logical address of the DP slave/PROFINET IO component (module or submodule) ■ High word: irrelevant
OFFSET	INPUT	INT	I, Q, M, D, L, constant	Number of the first byte to be written in the process image for the component (smallest possible value: 0)
LEN	INPUT	INT	I, Q, M, D, L, constant	Amount of bytes to be written
STATUS	OUTPUT	DWORD	I, Q, M, D	Contains error information for SFC 81 UBLKMOV in the form DW#16#40xxxx00 if *ERROR* = TRUE
ERROR	OUTPUT	BOOL	I, Q, M, D	Error display: *ERROR* = TRUE if an error occurs when calling SFC 81 UBLKMOV.
OUTPUTS	IN_OUT	ANY	I, Q, M, D	Source area for the data to be written: ■ If the source area is smaller than *LEN*, then as many bytes as OUTPUTS contains are transferred. *ERROR* receives the value FALSE. ■ If the source area is greater than *LEN*, then the first *LEN* bytes are transferred from *OUTPUTS*. *ERROR* receives the value FALSE.

Error Information

Chapter 11.1.56 'SFC 81 - UBLKMOV - Copy data area without gaps' on page 380

13 System Blocks

13.1 Fetch/Write Communication

13.1.1 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.

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!

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

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

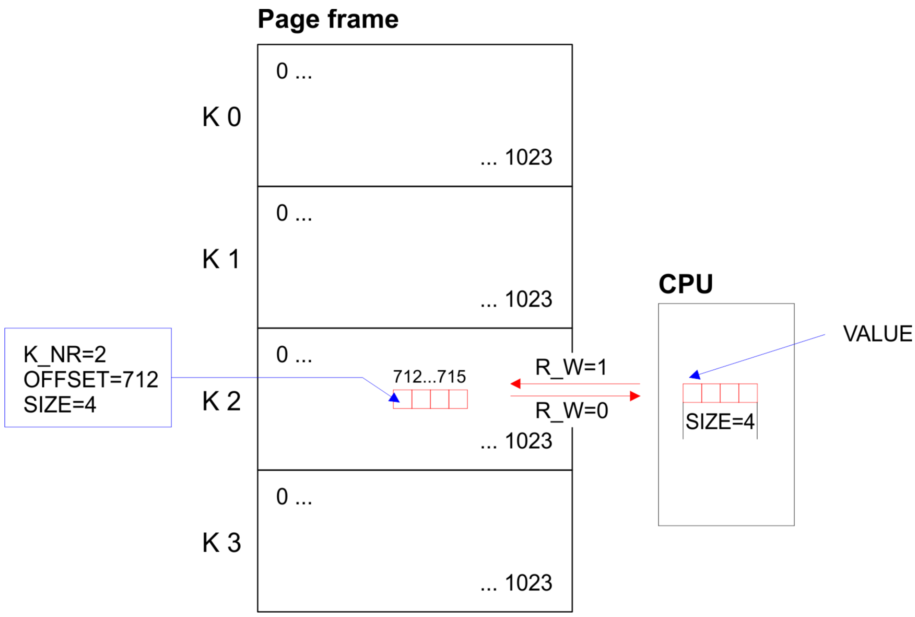
In-/output area

- 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

Value	Description
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 ist ≠ 0 or 1

13.1.2 SFC 230 ... 238 - Page frame communication

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

13.1.2.1 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

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

> *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.
  – Parameterart: Zeiger
  – Sinnvoller Bereich: DB, M, A, E

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: AB 0 ... AB127, MB 0...MB 255

13.1.2.2 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

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

```
CALL  SFC  230
       SSNR:=MW10
       ANR :=MW12
       IND :=MW14
       QANF:=P#DB10.DBX0.0 BYTE 16
       PAFE:=MB80
       ANZW:=MD48
```

Please note that you have to load the bit memory words with the corresponding values before.

13.1.2.3 Source res. destination definition

Overview

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	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#DBa.DBX b.0 BYTE` `CP#DB10.DBX 0.0 BYTE 8`	`P#M b.0 BYTE` `cP#M 5.0 BYTE 10`	`P#A b.0 BYTE` `cP#A 0.0 BYTE 2`	`P#E b.0 BYTE` `cP#E 20.0 BYTE 1`
DB, MB, AB, EB 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#A` The data is stored in the output byte.	`P#E` 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` Valid range for "c"	Length of the Source/ Destination data blocks in Words. 1 ... 2048	Length of the Source/ Destination data blocks in bytes. 1 ... 255	Length of the Source/ Destination data blocks in bytes. 1 ... 128	Length of the Source/ Destination data blocks in bytes. 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	IND = 1		IND = 2		
Definition	Indirect addressing for source **or** destination parameters. The source or destination parameters are stored in a DB. *QANF/ZANF:*		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	DW +0	Data type source	Description data source
	+2	DB-Nr. at type "DB", otherwise irrelevant	+2	DB-Nr. at type "DB", otherwise irrelevant	
	+4	Start address	+4	Start address	
	+6	Length in Byte	+6	Length in Byte	
			+8	Data type destin.	Description data destination
			+10	DB-Nr. at type "DB", otherwise irrelevant	
			+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	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*			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	Description data source/ destination	DW +0	Data type source	Description data source

QTYP/ZTYP	IND = 5			IND = 6		
	+2	DB-Nr. at type "DB", otherwise irrelevant		+2	DB-Nr. at type "DB", otherwise irrelevant	
	+4	Start address		+4	Start address	
	+6	Length in Byte		+6	Length in Byte	
	+8	Data type destin.	Description indicator word	+8	Data type destin.	Description data destination
	+10	DB-Nr. at type "DB", otherwise irrelevant		+10	DB-Nr. at type "DB", otherwise irrelevant	
	+12	Start address		+12	Start address	
				+14	Length in Byte	
				+16	Data type source	Description indicator word
				+18	DB-Nr. at type "DB", otherwise irrelevant	
				+20	Start address	
valid DB-No.	0 ... 32767			0 ... 32767		
Data word Definition	DW-Nr., where the stored data starts			DW-Nr., 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		

13.1.2.4 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 – 1: released – 0: 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 highbyte 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

The following section lists important status and error messages of the CPU 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.

- 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).
- 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.
- X 0 X 8 - The connection has been established again (e.g. after a CP reboot), the SEND order is released (SEND-communication order).
- X 0 X 9 - The connection has been established again, the RECEIVE order is released (RECEIVE-communication order).
- X 0 2 4 - SEND has been worked off without errors, the data was transferred.
- X 0 4 5 - RECEIVE was successful, the data arrived at the AG.
- 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

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
Telegramm da	X 0 X 1	.....	.....

State at H1	Prio 0/1	Prio 2	Prio 3/4
State at TCP/IP	**Prio 1**	**Prio 2**	**Prio 3**
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	

13.1.2.5 Parameterization error *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 ... 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

13.1.3 SFC 230 - SEND - Send to page frame

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.

> ***VIPA specific block***
> *The VIPA specific blocks can be found in the VIPA library.* *Chapter 4 'Include VIPA library' on page 103*

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.

13.1.4 SFC 231 - RECEIVE - Receive from page frame

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).

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

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

Name	Declaration	Type	Description
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.

13.1.5 SFC 232 - FETCH - Fetch from page frame

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).

> ***VIPA specific block***
> *The VIPA specific blocks can be found in the VIPA library.* *Chapter 4 'Include VIPA library' on page 103*

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

Name	Declaration	Type	Description
PAFE	OUT	BYTE	Parameterization error
ANZW	IN_OUT	DWORD	Indicator word

Information for indirect parameterization ↳ Chapter 13.1.2.3 'Source res. destination definition' on page 502

13.1.6 SFC 233 - CONTROL - Control page frame

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.

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. ↳ Chapter 4 'Include VIPA library' on page 103

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).

13.1.7 SFC 234 - RESET - Reset page frame

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.

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

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

13.1.8 SFC 235 - SYNCHRON - Synchronization page frame

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.

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

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

13.1.9 SFC 236 - SEND_ALL - Send all to page frame

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.

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

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, that is parameterized in the SEND_ALL block, the current order number is stored (0 means standby). The amount of the transmitted data for one order is shown in the data word of SEND_ALL which follows the indicator word.

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.*

13.1.10 SFC 237 - RECEIVE_ALL - Receive all from page frame

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.

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

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, 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.

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.*

13.1.11 SFC 238 - CTRL1 - Control1 page frame

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.

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *↳ Chapter 4 'Include VIPA library' on page 103*

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.

13.2 MMC Functions standard CPUs

13.2.1 SFC 220 ... 222 - MMC Access

Overview

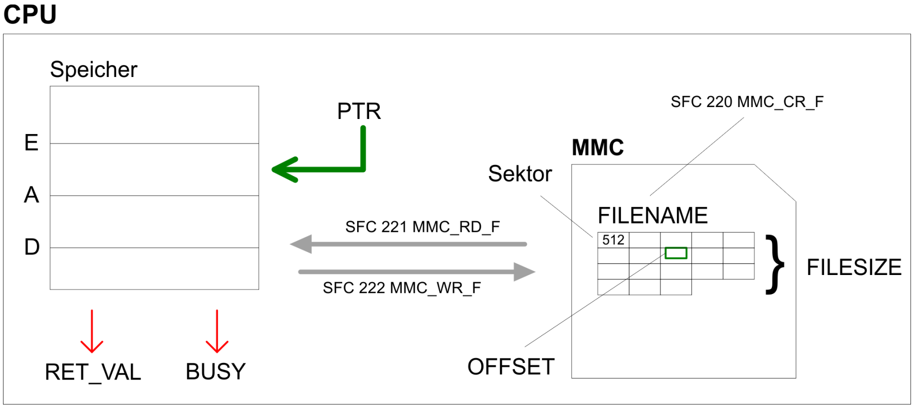
By means of these blocks there is the possibility to integrate MMC access to your application program. Here a new file may be created respectively an existing file may be opened for accessed when a MMC is plugged-in. As long as you do not open another file, you may access this file via read/write commands.

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 fragmented, for only complete data blocks may be read res. written.
- When transferring data to the MMC from an external reading device, they may be fragmented, 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 fragmented.
- 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:



For read and write accesses to the MMC, you firstly have to open the file with SFC 220!

13.2.2 SFC 220 - MMC_CR_F - create or open MMC file

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. For more detailed information to this and to the restrictions *Chapter 13.2.1 ‘SFC 220 ... 222 - MMC Access’ on page 521*.

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.

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 ‘Include VIPA library’ on page 103*

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

> *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.

Value	Description
Diagnostic messages	
0000h	No errors (appears if new file is generated).
0001h	File already exists, is not fragmented 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 fragmented 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.

13.2.3 SFC 221 - MMC_RD_F - read from MMC file

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 *Chapter 13.2.1 'SFC 220 ... 222 - MMC Access' on page 521*.

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

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.

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.

13.2.4 SFC 222 - MMC_WR_F - write to MMC file

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 ➪ *Chapter 13.2.1 'SFC 220 ... 222 - MMC Access' on page 521*.

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. ➪ Chapter 4 'Include VIPA library' on page 103

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.

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.

13.3 File Functions SPEED7 CPUs

13.3.1 FC/SFC 195 and FC/SFC 208...215 - Memory card access

Overview

The FC/SFC 195 and FC/SFC 208 ... FC/SFC 215 allow you to include the memory card access into your user application. The following parameters are necessary for the usage of the FC/SFCs:

HANDLE, FILENAME

The access takes place via a *HANDLE* number. That is assigned to a *FILENAME* via a call of the FC/SFC 208 FILE_OPN res. FC/SFC 209 FILE_CRE. At the same time a max. of 4 *HANDLE* may be opened (0 ... 3). To close an opened file call the FC/SFC 210 FILE_CLO and thus release the *HANDLE* again.

MEDIA

As media format set 0 for the MMC. Other formats are not supported at this time.

ORIGIN, OFFSET

Read and write start with the position of a write/read flag. After opening res. creation of a file, the write/read flag is at position 0. With FC/SFC 213 FILE_SEK you may shift the write/read flag from an *ORIGIN* position for an *OFFSET* (number Bytes).

REQ, BUSY

- With *REQ* = 1 you activate the according function.
- *REQ* = 0 returns the current state of a function via *RETVAL*.
- *BUSY* = 1 monitors that the according function is in process.

RETVAL

After the execution of a function *RETVAL* returns a number code:

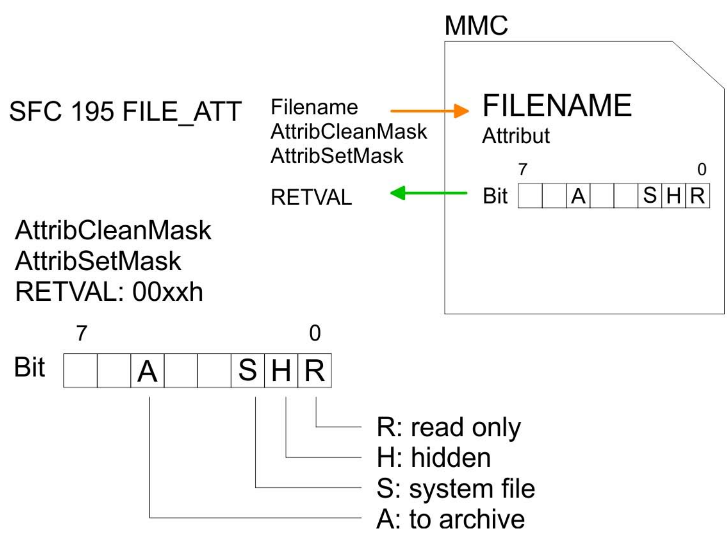
RETVAL = 0:	Function has been executed without errors.
0 < RETVAL < 7000h:	*RETVAL* = Length of the transferred data (only FC/SFC 211 and FC/SFC 212).
7000h ≤ RETVAL < 8000h:	Monitors the execution state of the function.
RETVAL ≥ 8000h:	Indicates an error that is described more detailed in the according FC/SFC.

> **CAUTION!**
> For the access of the memory card you must regard the following hints. Nonobservance may cause data loss at the memory card:
> – A max. of 4 Handle (0 ... 3) may be used at the same time!
> – File names must follow the 8.3 format or special character!
> – These FC/SFCs only gives you access to the top directory level (Root directory) of the memory card!
> – You may only rename or delete files that you've closed before with FC/SFCs 210 FILE_CLO!

13.3.2 FC/SFC 195 - FILE_ATT - Change file attributes

Description

In the root directory of the memory card the file attributes may be changed by FILE_ATT. Here enter a file name. The corresponding attributes may be reset with *ATTRIBCLEANMASK* respectively set with *ATTRIBSETMASK* by given bit pattern. Setting takes priority over resetting. After job execution the current state of the attributes is returned with *RETVAL* 00xxh. For determination of the current file attributes by *RETVAL*, the parameters *ATTRIBCLEANMASK* and *ATTRIBSETMASK* may be set to value 00h.



> ***VIPA specific block***
> *The VIPA specific blocks can be found in the VIPA library.* *↪ Chapter 4 'Include VIPA library' on page 103*

Parameters

Parameter	Declaration	Data type	Description
REQ	IN	BOOL	Activate function
MEDIA	IN	INT	0 = MMC
FILENAME	IN	STRING[254]	Name of file (must be in 8.3 format)
ATTRIBCLEANMASK	IN	BYTE	Bit pattern of attributes to clean
ATTRIBSETMASK	IN	BYTE	Bit pattern of attributes to set
RETVAL	OUT	WORD	Return value (00xxh=OK with xx: attributes)
BUSY	OUT	BOOL	Function is busy

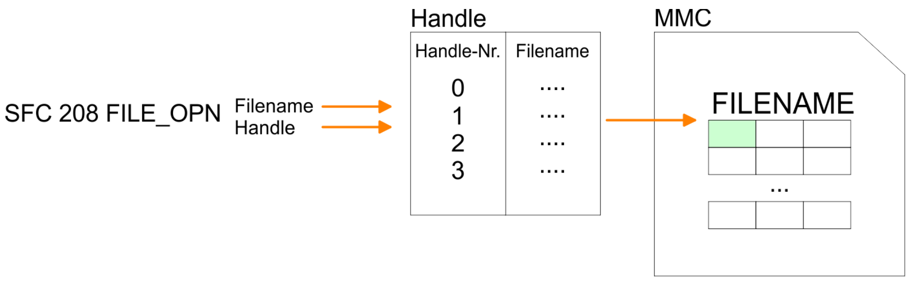
RETVAL (Return value) Return codes of *RETVAL*:

Code	Description
00xxh	OK, attributes have been changed with xx: attributes
7000h	*REQ* = 0, *BUSY* = 0 (nothing present)
7001h	*REQ* = 1, 1. call
7002h	Block is executed
A001h	The defined *MEDIA* type is not valid
A002h	Error in parameter *ATTRIBSETMASK*
A004h	File *FILENAME* is not found
A005h	*FILENAME* is a directory
A006h	File is just open
A007h	Memory card is write protected
A010h	File error FILENAME
A100h	General file system error (e.g. no memory card plugged)

13.3.3 FC/SFC 208 - FILE_OPN - Open file

Description

You may open a file on the memory card with FC/SFC 208. Here a *HANDLE* is connected to a *FILENAME*. By using the *HANDLE* you now have read and write access to the file until you close the file again with the FC/SFC 210 FILE_CLO. *REQ* = 1 initializes the function. After the opening the read/write flag is at 0.



VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

Parameters

Parameter	Declaration	Data type	Description
REQ	IN	BOOL	Activate function
MEDIA	IN	INT	0 = MMC
FILENAME	IN	STRING[254]	Name of file (must be in 8.3 format)
HANDLE	IN	INT	Index of file 0 ... 3
RETVAL	OUT	WORD	Return value (0 = OK)
BUSY	OUT	BOOL	Function is busy

RETVAL (Return value)

Codes that are returned by *RETVAL*:

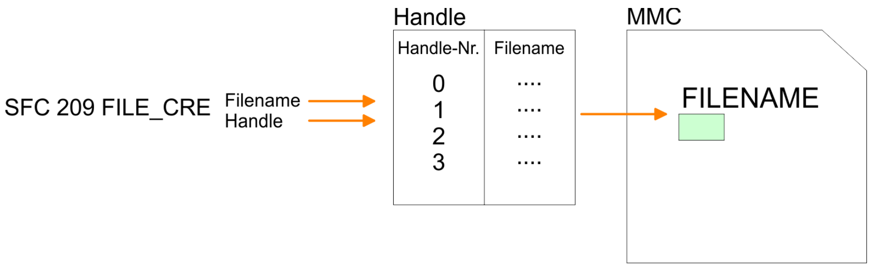
Code	Description
0000h	OK
7000h	*REQ* = 0, *BUSY* = 0 (nothing present)
7001h	*REQ* = 1, 1. call
7002h	Block is executed
8010h	Parameter *FILENAME* is not present (e.g. DB not loaded).
8011h	Error *FILENAME* (not conform with 8.3 or special character)
8100h	The defined *HANDLE* is not valid
9001h	*HANDLE* is assigned to another file
9002h	Another function has been called via this *HANDLE* and is ready
9003h	Another function has been called via this *HANDLE* and is ready
A000h	System internal error occurred
A001h	The defined *MEDIA* type is not valid

Code	Description
A003h	A general error in the file system occurred
A004h	The in *FILENAME* defined file doesn't exist or is a directory
A100h	General file system error (e.g. no memory card plugged)

13.3.4 FC/SFC 209 - FILE_CRE - Create file

Description

By using this block you may create a new file with the entered file name on the memory card (if plugged) and open it for read/write access. Please regard that you may only create files at the top directory level. *REQ* = 1 initializes the function. After opening, the write / read flag is at 0.



VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *➪ Chapter 4 'Include VIPA library' on page 103*

Parameters

Parameter	Declaration	Data type	Description
REQ	IN	BOOL	Activate function
MEDIA	IN	INT	0 = MMC
FILENAME	IN	STRING[254]	Name of file (must be in 8.3 format)
HANDLE	IN	INT	Index of file 0 ... 3
RETVAL	OUT	WORD	Return value (0 = OK)
BUSY	OUT	BOOL	Function is busy

RETVAL (Return value)

Codes that are returned by RETVAL:

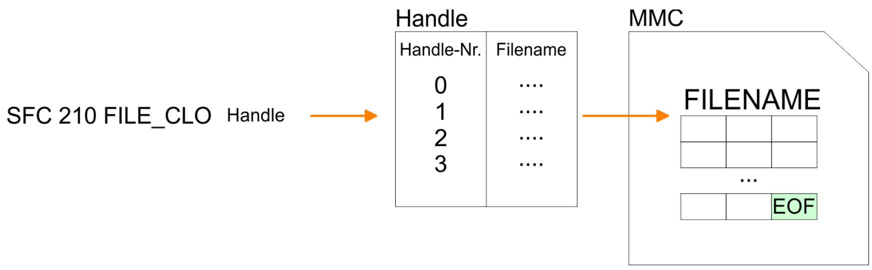
Code	Description
0000h	OK
7000h	*REQ* = 0, *BUSY* = 0 (nothing present)
7001h	*REQ* = 1, 1. call

Code	Description
7002h	Block is executed
8010h	Parameter *FILENAME* is not present (e.g. DB not loaded)
8011h	Error *FILENAME* (not conform with 8.3 or special character)
8100h	The defined HANDLE is not valid
9001h	HANDLE is assigned to another file
9002h	Another function has been called via this *HANDLE* and is ready
9003h	Another function has been called via this *HANDLE* and is not ready
A000h	System internal error occurred
A001h	The defined *MEDIA* type is not valid
A003h	A general error in the file system occurred
A004h	No root-entry is available in the directory
A005h	Memory card is write-protected
A100h	General file system error (e.g. no memory card plugged)

13.3.5 FC/SFC 210 - FILE_CLO - Close file

Description

This block allows you to close an opened file. Here an EOF (**E**nd **o**f **F**ile) is added, the file is closed and the *HANDLE* released. *REQ* = 1 initializes the function.



> ***VIPA specific block***
> *The VIPA specific blocks can be found in the VIPA library.* *Chapter 4 'Include VIPA library' on page 103*

Parameters

Parameter	Declaration	Data type	Description
REQ	IN	BOOL	Activate function
HANDLE	IN	INT	Index of file 0 ... 3
RETVAL	OUT	WORD	Return value (0 = OK)
BUSY	OUT	BOOL	Function is busy

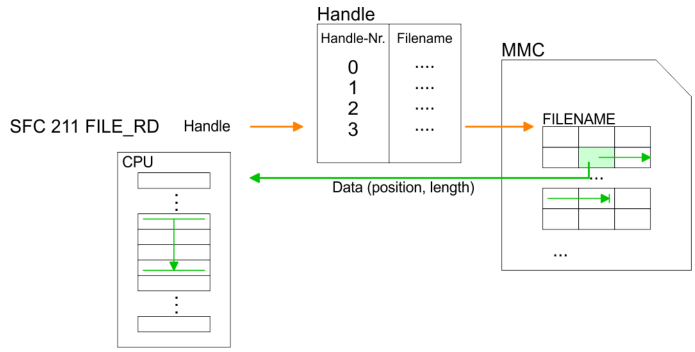
RETVAL (Return value) Codes that are returned by *RETVAL*:

Code	Description
0000h	OK
7000h	*REQ* = 0, *BUSY* = 0 (nothing present)
7001h	*REQ* = 1, 1. call
7002h	Block is executed
8100h	The defined *HANDLE* is invalid
9001h	The *HANDLE* is not assigned to a file name
9002h	Another function has been called via this *HANDLE* and is ready
9003h	Another function has been called via this *HANDLE* and is not ready
A000h	System internal error occurred
A100h	General file system error (e.g. no memory card plugged)

13.3.6 FC/SFC 211 - FILE_RD - Read file

Description

This allows you to transfer data from the memory card to the CPU via the opened *HANDLE* starting from an ORIGIN position (position of the read-/write flag). During every call you may transfer a max. of 512byte. By setting of *DATA* you define storage place and length of the write area in the CPU. *REQ* = 1 initializes the function.



VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

Parameters

Parameter	Declaration	Data type	Description
REQ	IN	BOOL	Activate function
HANDLE	IN	INT	Index of file 0 ... 3
DATA	IN	ANY	Pointer to PLC memory and data length

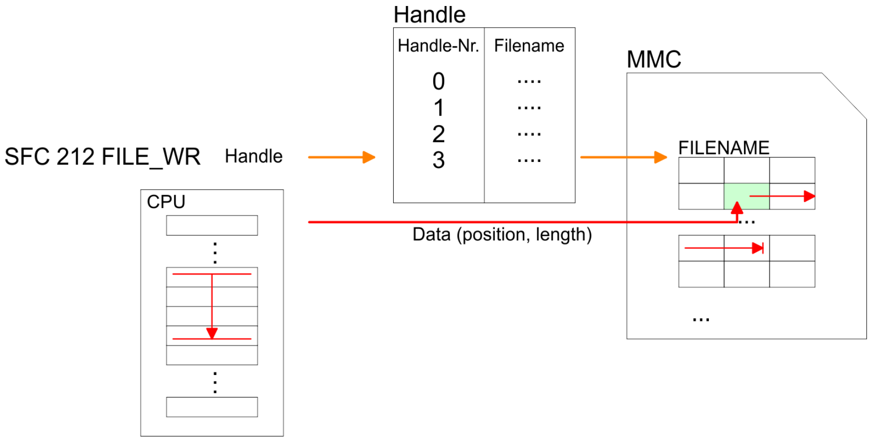
Parameter	Declaration	Data type	Description
RETVAL	OUT	WORD	Return value (0 = OK)
BUSY	OUT	BOOL	Function is busy

RETVAL (Return value) Codes that are returned by *RETVAL*:

Code	Description
0xxxh	0 = OK, 0xxx = Length of read data
7000h	*REQ* = 0, *BUSY* = 0 (nothing present)
7001h	*REQ* = 1, 1. call
7002h	Block is executed
8010h	Pointer in *DATA* has type *BOOL*
8011h	Pointer in *DATA* cannot be decoded (e.g. DB not loaded)
8012h	Data length exceeds 512byte
8013h	A write access to a write-protected DB happened
8100h	The defined *HANDLE* is not valid
9001h	For this *HANDLE* no file is opened.
9002h	Another function has been called via this *HANDLE* and is ready
9003h	Another function has been called via this *HANDLE* and is not ready
A000h	System internal error occurred
A003h	Internal error
A100h	General file system error (e.g. no memory card plugged)

13.3.7 FC/SFC 212 - FILE_WR - Write file

Description Use this block for write access to the memory card. This writes data from the position and length of the CPU defined under *DATA* to the memory card via the according *HANDLE* starting at the write/read position. During every call you may transfer a max. of 512byte. *REQ* = 1 initializes the function.



VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

Parameters

Parameter	Declaration	Data type	Description
REQ	IN	BOOL	Activate function
HANDLE	IN	INT	Index of file 0 ... 3
DATA	IN	ANY	Pointer to PLC memory and data length
RETVAL	OUT	WORD	Return value
BUSY	OUT	BOOL	Function is busy

The parameter *RETVAL* returns the length of the written data. The block doesn't announce an error message that the MMC is full. The user has to check himself if the number of the bytes to write corresponds to the number of written bytes returned by *RETVAL*.

RETVAL (Return value)

Codes that are returned by *RETVAL*:

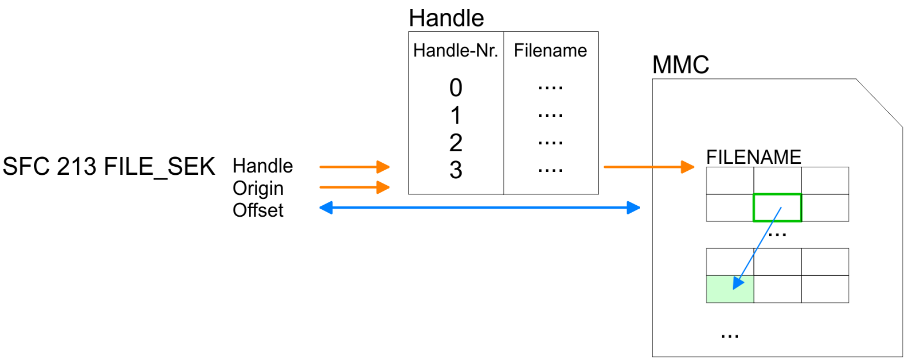
Code	Description
0xxxh	0 = OK, 0xxx = Length of written data
7000h	*REQ* = 0, *BUSY* = 0 (nothing present)
7001h	*REQ* = 1, 1. call
7002h	Block is executed
8010h	Pointer in *DATA* has type BOOL
8011h	Pointer in *DATA* cannot be decoded (e.g. DB not loaded)
8012h	Data length exceeds 512byte
8100h	The defined *HANDLE* is not valid

Code	Description
9001h	For this Handle no file is opened
9002h	Another function has been called via this *HANDLE* and is ready
9003h	Another function has been called via this *HANDLE* and is not ready
A000h	System internal error occurred
A002h	File is write-protected
A003h	Internal error
A004h	Memory card is write-protected
A100h	General file system error (e.g. no memory card plugged)

13.3.8 FC/SFC 213 - FILE_SEK - Position pointer

Description

FILE_SEK allows you to detect res. alter the position of the write-/read flag of the according *HANDLE*. By setting *ORIGIN* as start position and an *OFFSET* you may define the write-/read flag for the according *HANDLE*. *REQ* = 1 starts the function.



VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

Parameters

Parameter	Declaration	Data type	Description
REQ	IN	BOOL	Activate function
HANDLE	IN	INT	Index of file 0 ... 3
ORIGIN	IN	INT	0 = file start, 1 = current position, 2 = file end
RETVAL	OUT	WORD	Return value (0 = OK)
BUSY	OUT	BOOL	Function is busy
OFFSET	INOUT	DINT	Offset write-/read flag

RETVAL (Return value)

Codes that are returned by *RETVAL*:

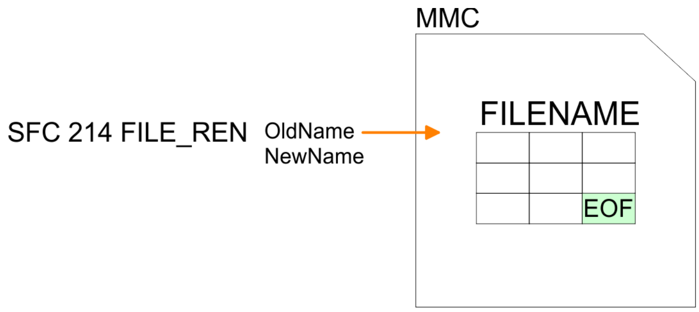
Code	Description
0000h	OK, *OFFSET* contains the current write-/read position
7000h	*REQ* = 0, *BUSY* = 0 (nothing present)
7001h	*REQ* = 1, 1. call
7002h	Block is executed
8100h	The defined *HANDLE* is not valid
9001h	For this *HANDLE* no file is opened
9002h	Another function has been called via this *HANDLE* and is ready
9003h	Another function has been called via this *HANDLE* and is not ready
A000h	System internal error occurred
A004h	*ORIGIN* parameter is defective
A100h	General file system error (e.g. no memory card plugged)

13.3.9 FC/SFC 214 - FILE_REN - Rename file

Description

Using FILE_REN you may alter the file name defined in *OLDNAME* to the file name that you type in *NEWNAME*.

CAUTION!
Please regard that you may only rename files that you've closed before with FILE_CLO. Nonobservance may cause data loss at the memory card!



VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

Parameters

Parameter	Declaration	Data type	Description
REQ	IN	BOOL	Activate function
MEDIA	IN	INT	0 = MMC
OLDNAME	IN	STRING[254]	Old name of file (must be in 8.3 format)
NEWNAME	IN	STRING[254]	New name of file (must be in 8.3 format)
RETVAL	OUT	WORD	Return value (0 = OK)
BUSY	OUT	BOOL	Function is busy.

RETVAL (Return value) Codes that are returned by *RETVAL*:

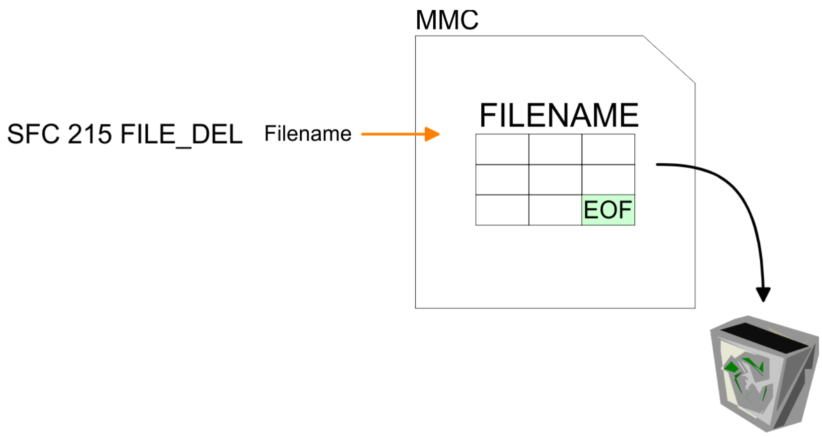
Code	Description
0000h	OK, file has been renamed
7000h	*REQ* = 0, *BUSY* = 0 (nothing present)
7001h	*REQ* = 1, 1. call
7002h	Block is executed
8010h	Parameter *OLDNAME* is not present (e.g. DB not loaded)
8011h	Error *OLDNAME* (not conform with 8.3 format or special character)
8020h	Parameter *NEWNAME* is not present (e.g. DB not loaded)
8021h	Error *NEWNAME* (not conform with 8.3 format or special character)
A000h	System internal error occurred
A001h	The defined MEDIA type is not valid
A003h	The new filename NEWNAME already exists
A004h	File *OLDNAME* is not found
A006h	File *OLDNAME* is just open
A007h	Memory card write-protected
A100h	Error occurs when file creation (e.g. no memory card plugged)

13.3.10 FC/SFC 215 - FILE_DEL - Delete file

Description

This block allows you to delete a file at the memory card. For this, type the file name of the file to delete under *FILENAME*.

CAUTION!
Please regard that you may only delete files that you've closed before with FILE_CLO. Nonobservance may cause data loss at the memory card!



VIPA specific block
The VIPA specific blocks can be found in the VIPA library. *🡆 Chapter 4 'Include VIPA library' on page 103*

Parameters

Parameter	Declaration	Data type	Description
REQ	IN	BOOL	Activate function
MEDIA	IN	INT	0 = MMC
FILENAME	IN	STRING[254]	Name of file (must be in 8.3 format)
RETVAL	OUT	WORD	Return value (0 = OK)
BUSY	OUT	BOOL	Function is busy.

RETVAL (Return value) Codes that are returned by *RETVAL*:

Code	Description
0000h	OK, file has been deleted
7000h	*REQ* = 0, *BUSY* = 0 (nothing present)
7001h	*REQ* = 1, 1. call
7002h	Block is executed
8010h	Parameter *FILENAME* is not available (e.g. DB not loaded)

Code	Description
8011h	*FILENAME* is defective (e.g. is not conform with 8.3 format or special character)
A000h	System internal error occurred
A001h	The defined *MEDIA* type is not valid
A002h	The file is write-protected
A004h	File *FILENAME* is not found
A005h	*FILENAME* is a directory - you cannot delete
A006h	File is just open
A007h	Memory card is write-protected
A100h	General file system error (e.g. no memory card plugged)

13.4 System Functions

13.4.1 SFC 75 - SET_ADDR - Set PROFIBUS MAC address

Description

With this SFC you can change the MAC address of the integrated PROFIBUS interface of a CPU. The function is only possible in the passive DP slave mode. To identify the diagnostic address is used. The SFC is asynchronous and can be applied only to one interface. At STOP and subsequent warm start the set network address is retained. With PowerOFF-PowerON or on overall reset the interface gets the configured node number The DP slave consistently assumes the identity of the DP slave with the new address. For the DP master the DP slave with the old address fails and a DP slave with the new address returns. If an address is selected, which is already used by another node on the DP line, then both slaves fail in accordance to the DP communication.

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

Parameters

Parameter	Declaration	Data type	Memory area	Description
REQ	INPUT	BOOL	I, Q, M, D, L	Function request with *REQ* = 1
LADDR	INPUT	WORD	I, Q, M, D, L	Identification of the interface
ADDR	INPUT	BYTE	I, Q, M, D, L	New node address
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Error code
BUSY	OUTPUT	BOOL	I, Q, M, D, L	*BUSY* = 1: In progress

RET_VAL (return value)

Value	Description
0000h	Job has been executed without error
7000h	Function request with *REQ* = 0 (call without processing) *BUSY* is set to 0, no data transfer is active
7001h	First call with *REQ* = 1: Data transfer started *BUSY* is set to 1
7002h	Intermediate call (*REQ* irrelevant): Data transfer started *BUSY* is set to 1
8xyyh	General error information *Chapter 2.1 'General and Specific Error Information RET_VAL' on page 67*
8090h	Identification of the interfaces: Logical address is not valid
8091h	New node address is not valid
8093h	Identification of the interfaces: Logical address is no interface
809Bh	Function not executable (e.g.. interface is no DP slave or active)
80C3h	There are no resources (e.g. multiple call of the SFC)

13.4.2 FC/SFC 193 - AI_OSZI - Oscilloscope-/FIFO function

Description

- The FC/SFC 193 serves for controlling the oscilloscope-/FIFO function of analog input channels with this functionality.
- It allows to start the recording and to read the buffered data.
- Depending upon the parameterization there are the following possibilities:

Oscilloscope operation

- Depending on the trigger condition at edge evaluation the monitoring of the configured channel may be started respectively at manual operation the recording may be started.
- The recorded measuring values may be accessed by the FC/SFC 193 as soon as the buffer is full.

FIFO operation

- Start the recording.
- Read the puffer at any time.

The FC/SFC may only be called from on level of priority e.g. only from OB 1 or OB 35.

The module is to be parameterized before.

For starting and reading in each case the FC/SCF 193 is to be called. The differentiation of both variants takes place in the parameter MODE.

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *↳ Chapter 4 'Include VIPA library' on page 103*

Parameters

Parameter	Declara-tion	Data type	Function depending on MODE
REQ	IN	BOOL	Execute function (start/read)
LADR	IN	WORD	Base address of the module
MODE	IN	WORD	Mode (start/read)
CHANNEL	IN	BYTE	Channel to be read
OFFSET	IN	DWORD	Address offset for reading (not FIFO operation)
RECORD	IN	ANY	Memory for the read data
RETVAL	OUT	WORD	Return value (0 = OK)
BUSY	OUT	BOOL	Function is busy
TIMESTAMP	OUT	DWORD	Time stamp (only at edge evaluation)
LEN	INOUT	DWORD	Number of values to be handled per channel

REQ

- Depending on the set *MODE* when the bit is set the recording respectively the reading may be started.
- Depending on the trigger condition at edge evaluation the monitoring of the configured channel may be started respectively at manual operation the recording may be started.
- The data are read from the module, if "read" is set at *MODE*.

LADR

Logical basic address of the module.

MODE

The FC/SFC 193 may be called with 3 different modes. The corresponding mode may be set by the parameter *MODE*. The configured mode is executed by setting *REQ*. The following values are supported:

- 01h: Starts recording respectively edge monitoring depending upon the parameterization.
- 00h: Read data within several cycles until *BUSY* = 0.
- 80h: Read data with one access.

CHANNEL

Here the channel is specified to be read. With each call one channel may be read. This parameter is irrelevant at start calls with *MODE* = 01h.

OFFSET

- Offset specifies an address offset for the reading process. By this you get access to sub-ranges of the recorded data.
- The value for the maximum offset depends on the number of values, which were recorded per channel.
- *OFFSET* is not supported in FIFO operation. It will be ignored.

RECORD

- Here an area for the read values to be stored at may be defined.
- In FIFO operation every value of the selected channel may be read, which were stored up to the time of start reading.
- Please regard that the buffer has a sufficient size for the data to be buffered, otherwise an error is reported.

BUSY

- *BUSY* = 1 indicates that the function just processed.
- *BUSY* = 0 indicates that the function is finished.

TIMESTAMP

- There is an internal clock with a resolution of 1μs running in every SPEED-Bus module.
- The returned value corresponds to the time at the SPEED-Bus module, on which the trigger event occurred.
- *TIMESTAMP* is only available at the edge triggered oscilloscope operation.
- It is valid as long as the job is running (*RETVAL* = 7xxxh) and bit 4 of byte 0 is set respectively the job has been finished without an error (*RETVAL* = 0000h).

LEN

The length parameter realized as IN/OUT is variably interpreted depending on the selected mode at the function call.

Mode: start (MODE: = 01h)

At *MODE* = 01h this parameter may only be used at the manual oscilloscope start. Here the requested number of values per channel to be buffered may be assigned. In this mode there is no value reported by *LEN*.

Mode: read (MODE: = 00h or 80h)

At *MODE* = 00h respectively 80h the number of values to be read may be set. This parameter is ignored in FIFO operation. The number of the read values is returned by *LEN*.

RETVAL (Return value)

In addition to the module specific error codes listed here, there general FC/SFC error information may be returned as well.

RETVAL	Description depending on the BUSY-Bit	BUSY
Byte		
0	Bit 1, 0:	
	00: Call with REQ: = 0 (idle, waiting for *REQ* = 1)	0
	01: First call with *REQ*: = 1	1

RETVAL	Description depending on the BUSY-Bit	BUSY
	10: Subsequent call with *REQ*: = 1	1
	11: Oscilloscope is just recording	1
	Bit 2: REQ: = 1, but recording was not yet started. (*MODE*: = 00h or *MODE*: = 80h)	0
	Bit 3: reserved	-
	Bit 4: Trigger event occurred and recording is just running.	1
	Bit 5: Waiting for trigger event	1
	Bit 7…6: reserved	-
1	Bit 0: reserved	-
	Bit 1: The number of recorded values exceeds the target area defined by *RECORD* (in words).	0
	Bit 2: The number of the recorded values exceeds the area defined by *LEN* and *OFFSET*.	0
	Bit 3: Buffer overflow in FIFO operation.	0
	Bit 7...4:	
	0000: Job finished without an error	0
	0111: Job still running	1
	1000: Job finished with error	0

Job finished without an error

RETVAL	Description depending on the BUSY-Bit	BUSY
0000h	Job was finished without an error.	0

Job finished with error

RETVAL	Description depending on the BUSY-Bit	BUSY
8002h:	Oscilloscope-/FIFO function is not configured.	0
8003h:	An internal error occurred - please contact VIPA.	0
8005h:	The selected channel may not be read - wrong channel number.	0
8007h:	The value at *OFFSET* exceeds the number of recorded values.	0
8090h:	There is no SPEED-Bus module with this address available.	0
80D2h:	*LADR* exceeds the peripheral address area.	0

13.4.3 FC/SFC 194 - DP_EXCH - Data exchange with CP342S

Description

With the FC/SFC 194 you can exchange data between your CPU and a PROFIBUS DP master, which is connected via SPEED-Bus. Normally each PROFIBUS DP master embeds its I/O area into the peripheral area of the CPU. Here you can address a periphery range of 0 ... 2047 via the hardware configuration. Since this limits the maximum number of PROFIBUS DP master modules at the SPEED-Bus, there is the possibility to deactivate the mapping at the appropriate DP master and to activate instead the access via handling blocks. Here you can write data from the CPU in a defined area of the DP master and read data from a defined area of the DP master.

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

Parameters

Parameter	Declaration	Data type	Functionality depending on MODE
LADR	IN	WORD	Base address of the DP master module on the SPEED-Bus
MODE	IN	WORD	Modus (0 = read / 1 = write)
LEN	IN	WORD	Length of the data area in the DP master
OFFSET	IN	DWORD	Begin of the data area in the DP master
RETVAL	OUT	WORD	Return value (0 = OK)
DATA	IN OUT	ANY	Pointer to the data area of the CPU

LADR

Logical base address of the module.

MODE

Den FC/SFC 194 may be called with the following modes:

- 0000 = Transfer data from the DP master to the CPU.
- 0001 = Transfer data from the CPU to the DP master.

LEN

Here the length of the data area in the DP master is defined.

OFFSET

Here the beginning of the data area in the DP master is defined. Please consider that the area defined via *OFFSET* and *LEN* does not exceed the area defined of the DP master by the hardware configuration.

RETVAL (Return value)

In addition to the module-specific error codes listed here, as return value there are also general error codes possible for FC/SFCs . *Chapter 2.1 'General and Specific Error Information RET_VAL' on page 67*

RETVAL	Description
0000h	No error
8001h	*LADR* could not be assigned to a DP master at the SPEED-Bus.
8002h	The value of the parameter *MODE* is out of range.
8003h	The value of the parameter *LEN* is 0.
8004h	The value of the parameter *LEN* is greater than the data area defined at *DATA*.
8005h	The area defined by *OFFSET* and *LEN* is out of the range 0 …2047.
8006h	The DP master specified by *LADR* is not configured for access via handling block. Activate in the properties of the DP master "IO-Mode HTB".
8008h	There are gap(s) in the input area.
8009h	There are gap(s) in the output area.
8010h	Error while accessing the input area (e.g. DP master is not reachable)
8011h	Error while accessing the output area (e.g. DP master is not reachable)
8Fxxh	Error at DATA (xx) *Chapter 2.1 'General and Specific Error Information RET_VAL' on page 67*

13.4.4 FC/SFC 219 - CAN_TLGR - CANopen communication

FC/SFC 219 CAN_TLGR SDO request to CAN master

Every SPEED7-CPU provides the integrated FC/SFC 219. This allows you to initialize a SDO read or write access from the PLC program to the CAN master. For this you address the master via the slot number and the destination slave via its CAN address. The process data is defined by the setting of *INDEX* and *SUBINDEX*. Via SDO per each access a max. of one data word process data can be transferred.

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *Chapter 4 'Include VIPA library' on page 103*

Parameters

Parameter	Declaration	Data type	Description
REQUEST	IN	BOOL	Activate function
SLOT_MASTER	IN	BYTE	SPEED-Bus slot (101 ... 116)
NODEID	IN	BYTE	CAN address (1 ... 127)
TRANSFERTYP	IN	BYTE	Type of transfer
INDEX	IN	DWORD	CANopen Index
SUBINDEX	IN	DWORD	CANopen sub index
CANOPENERROR	OUT	DWORD	CANopen error

Parameter	Declaration	Data type	Description
RETVAL	OUT	WORD	Return value (0 = OK)
BUSY	OUT	BOOL	Function is busy
DATABUFFER	INOUT	ANY	Data Buffer for FC/SFC communication

REQUEST

Control parameter: 1: Initialization of the order

SLOT_MASTER

101...116: slot 1 ... 16 from master at SPEED-Bus

NODELD

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 sub index

SLOT_MASTER

0:	System 200 CPU 21xCAN
1...32:	System 200 IM 208CAN
101...115:	System 300S 342-1CA70

CANOPENERROR

When no error occurs, *CANOPENERROR* returns 0. In case of an error *CANOPENERROR* contains one of the following error messages that are created by the CAN master:

Code	Description
0503 0000h	Toggle Bit not alternated
0504 0000h	SDO Time out value reached
0504 0001h	Client/server command specify not valid, unknown
0504 0002h	Invalid block size (only block mode)
0504 0003h	Invalid sequence number (only block mode)
0504 0004h	CRC error (only block mode)
0504 0005h	Insufficient memory
0601 0000h	Attempt to read a write only object
0601 0001h	Attempt to write a read only object

Code	Description
0602 0000h	Object does not exist in the object dictionary
0604 0041h	Object cannot be mapped to the PDO
0604 0042h	The number and length of the objects to be mapped would exceed PDO length.
0604 0043h	General parameter incompatibility reason
0604 0047h	General internal incompatibility reason in the device
0606 0000h	Access failed because of an hardware error
0607 0010h	Data type does not match, length of service parameter does not match.
0607 0012h	Data type does not match, length of service parameter exceeded.
0607 0013h	Data type does not match, length of service parameter shortfall.
0609 0011h	Sub index does not exist
0609 0030h	Value range of parameter exceeded (only for write access)
0609 0031h	Value of parameter written too high
0609 0032h	Value of parameter written too low
0609 0036h	Maximum value is less than minimum value
0800 0000h	General error
0800 0020h	Data cannot be transferred or stored to the application.
0800 0021h	Data cannot be transferred or stored to the application because of local control.
0800 0022h	Data cannot be transferred or stored to the application because of the present device state.
0800 0023h	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

When the function has been executed without error, the return value contains the valid length of the response data: 1: BYTE, 2: WORD, 4: DWORD. If an error occurs during execution, the return value contains one of the following error codes.

Code	Description
F021h	Invalid slave address (call parameter equal 0 or higher 127)
F022h	Invalid transfer type (value not equal to 40h, 23h, 2Bh, 2Fh)
F023h	Invalid data length (data buffer too small, at SDO read access this should be at least 4byte, at SDO write access at least 1byte, 2byte or 4byte).
F024h	FC/SFC is not supported.
F025h	Write buffer in CANopen master overflow, service cannot be processed at this time.

Code	Description
F026h	Read buffer in CANopen master overflow, service cannot be processed at this time.
F027h	SDO read or write access with defective response *‘CANOPENERROR’ on page 546*.
F028h	SDO timeout (no CANopen station with this node-ID found).

BUSY

As long as *BUSY* = 1, the current order is not finished.

DATABUFFER

- Data area via that the FC/SFC communicates. Set here an ANY pointer of the type Byte.
- SDO read access: Destination area for the read user data.
- SDO write access: Source area for the user data to write.

When the SDO request has been executed without errors, RETVAL contains the length of the valid response data (1, 2 or 4byte) and CANOPENERROR the value 0.

13.4.5 FC/SFC 254 - RW_SBUS - IBS communication

Description

This block serves the INTERBUS-FCs 20x as communication block between INTERBUS master and CPU.

For the usage of the INTERBUS-FCs 20x the FC/SFC 254 must be included in your project as block.

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. Chapter 4 ‘Include VIPA library’ on page 103

Parameters

Parameter	Declaration	Type	Description
READ/WRITE	IN	Byte	0 = Read, 1 = Write
LADDR	IN	WORD	Logical Address INTERBUS master
IBS_ADDR	IN	WORD	Address INTERBUS Master
DATAPOINTER	IN	ANY	Pointer to PLC data
RETVAL	OUT	WORD	Return value (0 = OK)

READ/WRITE

This defines the transfer direction seen from the CPU. *READ* reads the data from the Dual port memory of the INTERBUS master.

LADDR

Enter the address (**L**ogical **Addr**ess) from where on the register of the master is mapped in the CPU. At the start-up of the CPU, the INTERBUS master are stored in the I/O address range of the CPU following the shown formula if no hardware configuration is present:

Start address = 256× (slot-101)+2048

The slot numbers at the SPEED-Bus start with 101 at the left side of the CPU and raises from the right to the left. For example the 1. slot has the address 2048, the 2. the address 2304 etc.

IBS_ADDR

Address in the address range of the INTERBUS master.

DATAPOINTER

Pointer to the data area of the CPU.

RETVAL

Value that the function returns. 0 means OK.

13.5 System Function Blocks

13.5.1 SFB 7 - TIMEMESS - Time measurement

In opposite to the SFC 53, the SFB 7 returns the difference between two calls in µs. With *RESET* = 1 the current timer value is transferred to InstDB. Another call with *RESET* = 0 displays the difference in µs via *VALUE*.

VIPA specific block

The VIPA specific blocks can be found in the VIPA library. *↳ Chapter 4 'Include VIPA library' on page 103*

Parameters

Name	Declaration	Type	Comment
RESET	IN	BOOL	*RESET* = 1 start timer
VALUE	OUT	DWORD	Difference in µs

RESET

RESET = 1 transfers the current timer value to InstDB. Here *VALUE* is not influenced.

VALUE

After a call with *RESET* = 0, *VALUE* returns the time difference between the two SFB 7 calls.

14 SSL System status list

14.1 Overview SSL

SSL

This chapter describes all the partial lists of the system status list, readable via SFC 51 RDSYSST or via Hardware configurator. SSL partial lists, which are only for internal usage, are not described here.The SSL (**s**ystem **s**tatus **l**ist) describes the current status of a automation system. It contains the following information:

- System data
  – These are fixed or assigned characteristics data of a CPU as configuration of the CPU, status of the priority classes and communication.
- Module status data in the CPU
  – This describes the current status of the components monitored by system diagnostic functions.
- Diagnostics data
  – The diagnostics data of modules with diagnostic capabilities assigned to the CPU.
- Diagnostics buffer
  – Diagnostic entries of the diagnostic buffer in the order in which they occur.

SSL partial list

- Only partial lists of the SSL may be accessed. The partial lists are virtual list, this means, they are only created by the operating system of the CPUs when specifically requested and my only be read.
- A partial list or a list extract may be read e.g. by means of the SFC 51 RDSYSST.
- Here with the parameters SSL_ID and INDEX you define the kind of information to read.

A partial list always has the following structure:

- Header
  – SSL-ID
  – Index
  – Length of the record set in byte
  – Number of record sets of the partial list
- Record sets
  – A record set of a partial list has a certain length, depending on the information of the partial list. It depends on the partial list as the data words are used in a record set.

SSL-ID

Structure

	SSL-ID															
	High byte								Low byte							
Bit number	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Module class: CPU: 0000 IM: 0100 FM: 1000 CP: 1100				Number of the partial list extract: Definition of the subset of the partial list				Number of the partial list: Definition of the partial list of the SSL							

14.2 Overview - SSL partial lists

SSL partial lists

In the following all the possible SSL partial lists with additional SSL-ID are listed, which are supported by the SPEED7 system.

SSL partial lists, which are only for internal usage, are no more described.

SSL partial list	SSL-ID
Module identification	xy11h
CPU characteristics	xy12h
User memory areas	xy13h
System areas	xy14h
Block Types	xy15h
Status of all LEDs	xy19h
Identification of the component	xy1Ch
Interrupt status	xy22h
Communication status data	xy32h
Ethernet details of the module	xy37h
Status of the TCON Connections	xy3Ah
Status of the LEDs	xy74h
Status information CPU	xy91h
Stations status information (DPM)	xy92h
Stations status information (DPM, PROFINET-IO and EtherCAT)	xy94h
Module status information (PROFIBUS DP, PROFINET-IO, EtherCAT)	xy96h
Diagnostic buffer of the CPU	xyA0h
Module diagnostic information (record set 0)	xyB1h

SSL partial list	SSL-ID
Module diagnostic information (record set 1) via physical address	xyB2h
Module diagnostic information (record set 1) via logical address	xyB3h
Diagnostic data of a DP slave	xyB4h
Information EtherCAT master/slave	xyE0h
EtherCAT bus system	xyE1h
Statistics information to OBs	xyFAh
Status of the VSC features from the System SLIO CPU	xyFCh

14.3 Module Identification - SSL-ID: xy11h

Description

With the *SSL-ID* xy11h you obtain the module identification data of your module.

Parameters

SSL_ID	INDEX	Description
0011h	-	All identification data
0111h		Selection of the identification data:
	0001h	Identification data of the module
	0006h	Identification data of the basic hardware
	0007h	Identification data of the basic firmware
	0081h	Identification data of the VIPA firmware
	0082h	Identifikation der SVN-Version
0F11h	-	only SSL partial list header information

LENTHDR	One record set is 14words long (28bytes).
N_DR	Number of record sets

Record set

SSL_ID: xy11h

Do not configure CPU as Siemens 318-2AJ00

Name	Length	Description
index	1word	Number of a identification record set
mlfb	20byte	■ 0001h and 0006h: – Order number (MlfB) of the module; string of 19 characters and one blank (20h) e.g. 6ES7 315-2EH14 ■ 0007h: – Space (20h) ■ 0081h: – VIPA product name and hardware release: e.g. VIPA 015-CEFPR00-0100 ■ 0082h: – Text: "SVN revision"
bgtyp	1word	reserved
ausbg1	1word	■ 0001h and 0006h: – Hardware release of the module ■ 0007h: – "V" and first digit of the version ID ■ 0081h: – VIPA version ID: First digit in ASCII, second digit in hex ■ 0082h: – High word of "SVN revision" in hex
ausbg2	1word	■ 0001h and 0006h: – reserved ■ 0007h: – remaining digits of the version ID ■ 0081h: – VIPA version ID: third and fourth digit in hex ■ 0082h: – Low word of "SVN revision" in hex

Configure CPU as Siemens 318-2AJ00

Name	Length	Description
index	1word	Number of an identification record set
mlfb	20byte	■ 0001h and 0006h: – Order number (MlfB) of the module; string of 19 characters and one blank (20h) e.g. 6ES7 318-2AJ00-0AB0 ■ 0007h: – VIPA product name and hardware release: e.g. VIPA 317-4NE12-0119
bgtyp	1word	reserved

Name	Length	Description
ausbg1	1word	■ 0001h and 0006h: – Hardware release of the module ■ 0007h: – "V" and first digit of the version ID
ausbg2	1word	■ 0001h and 0006h: – reserved ■ 0007h: – remaining digits of the version ID

14.4 CPU characteristics - SSL-ID: xy12h

Description

Here you can determine the hardware-specific characteristics of your CPU by specifying the appropriate feature code.

Parameters

SSL_ID	INDEX	Description
0012h	-	All CPU characteristics
0112h		CPU characteristics of one group:
	0000h	MC7 processing unit
	0100h	Time system
	0200h	System response
	0300h	MC7 language description of the CPU
0E11h	0F12h	SSL partial list header information

LENTHDR	One record set is 1word long (2bytes).
N_DR	Number of record sets

Record set

SSL_ID: 0012h

All record sets of the CPU characteristics relevant for your CPU are listed. They follow completely one behind the other. One record set is 1word long. For each feature there is an ID. This ID is 1word long. You will find the list of the characteristics IDs on the following page.

SSL_ID: 0112h

All data records relevant for the group are listed. They follow completely one behind the other.

Characteristics identifier

Identifier	Description
0000h - 00FFh	**MC7 processing unit**
0001h	MC7 processing generating code

Identifier	Description
0002h	MC7 interpreter
0100h - 01FFh	**Time system**
0101h	1ms resolution
0102h	10ms resolution
0103h	no real time clock
0104h	BCD time-of-day format
0105h	all time-of-day functions (set time-of-day, set and read time-of-day, time-of-day synchronization: time-of-day slave and time-of-day master)
0300h - 03FFh	**MC7 language description of the CPU**
0301h	reserved
0302h	all 32 bit fixed-point instructions
0303h	all floating-point instructions
0304h	sin, asin, cos, acos, tan, atan, sqr, sqrt, in, exp
0305h	ACCU3/ACCU4 with corresponding instructions (ENT, PUSH, POP, LEAVE)
0306h	Master Control Relay instructions
0307h	Address register 1 exists with corresponding instructions
0308h	Address register 2 exists with corresponding instructions
0309h	Operations for area-crossing addressing
030Ah	Operations for area-internal addressing
030Bh	all memory-indirect addressing instructions via M
030Ch	all memory-indirect addressing instructions via DB
030Dh	all memory-indirect addressing instructions via DI
030Eh	all memory-indirect addressing instructions for L
030Fh	all instructions for parameter transfer in FCs
0310h	Memory bit edge instructions via I
0311h	Memory bit edge instructions via Q
0312h	Memory bit edge instructions via M
0313h	Memory bit edge instructions via DB
0314h	Memory bit edge instructions via DI
0315h	Memory bit edge instructions via L

Identifier	Description
0316h	Dynamic evaluation of the FC bits
0317h	Dynamic local data area with the corresponding instructions

14.5 User memory areas - SSL-ID: xy13h

Description

With the partial list with the SSL-ID xy13h you obtain information about the memory areas of the CPU.

Parameters

SSL_ID	INDEX	Description
0013h	xxxx	Record sets for any memory areas

SSL_ID	INDEX	Description
0013h	xxxx	Record sets for any memory areas
	0001h	Work memory
	0002h	Load memory integrated
	0003h	Load memory plugged
	0004h	Max. plug-in load memory
	0005h	Size of backup memory
0F13h	xxxx	SSL partial list header information

LENTHDR	One record set is 18words long (36byte).
N_DR	Number of record sets

Record set

***SSL_ID:* xy13h**

Name	Length	Description
index	1word	Not relevant
code	1word	Type of memory: ■ 0001h: volatile memory (RAM) ■ 0002h: non volatile memory (RAM) ■ 0003h: mixed memory (RAM and EPROM)
größe	2words	Total size of the selected memory (total of area Ber1 and Ber2)

Name	Length	Description
modus	1word	Logical mode of the memory: ■ Bit 0: RAM ■ Bit 1: EPROM ■ Bit 2: RAM and EPROM For work memory: ■ Bit 3: Code and data separated ■ Bit 4: Code and data together
granu	1word	0 (fix)
ber1	2words	Size of the RAM in byte
belegt1	2words	Size of the RAM being used
block1	2words	Largest free block in the RAM ■ "0": no information available or cannot be determined.
ber2	2words	Size of the EPROM in byte
belegt2	2words	Size of the EPROM being used
block2	2words	Largest free block in the EPROM ■ "0": no information available or cannot be determined.

14.6 System areas - SSL-ID: xy14h

Description

If you read the partial list with SSL-ID xy14h, you obtain information about the system areas of the CPU.

Parameters

SSL_ID	INDEX	Description
0014h	-	All system areas of a CPU
0F14h	-	SSL partial list header information

LENTHDR	One record set is 4words long (8byte)
N_DR	Number of record sets ■ You must at least assign a number of 9 record sets. ■ If you select a target area, which is too small, the SFC 51 RDSYSST does not provide a record set.

Record set

SSL_ID: xy14h

Name	Length	Description
index	1word	Index of the system area ■ 0001h: PII (quantity in byte) ■ 0002h: PIQ (quantity in byte) ■ 0003h: Memory (number in bits) – This index is only provided by the CPU, where the number of flags can be shown in one word. If your CPU does not provide this value, you must evaluate index 0008h ■ 0004h: Timers (quantity) ■ 0005h: Counters (quantity) ■ 0006h: Quantity of bytes in the logical address area. ■ 0007h: Local data (entire local data area of the CPU in byte) – This index is only provided by the CPU, where the number of local data area can be shown in one word. If your CPU does not provide this value, you must evil index 0009h ■ 0008h: Memory (number in bytes) ■ 0009h: Local data (entire local data area of the CPU in kbytes)
code	1word	Memory type: ■ 0001h: RAM ■ 0002h: EPROM
quantity	1word	Number of elements of the system area defined by *INDEX*.
remain	1word	Number of retentive elements defined by *INDEX*.

14.7 Block Types - SSL-ID: xy15h

Description

You obtain the block types (OBs, DBs, SDBs, FCs and FBs) that exists on the CPU.

Parameters

SSL_ID	INDEX	Description
0015h	-	Record sets of all block types of a CPU (Standard blocks)
0115h	xxxh	Record set of a block type of a CPU
0815h	xxxh	Record set of a block type of a CPU (VIPA specific blocks)
0F15h	-	Returns the number of records and the size of the data sets for standard blocks
8F15h	-	Returns the number of records and the size of the data sets for VIPA blocks

LENTHDR	one record set is 5words long (10byte)
N_DR	Number of record sets

Record set

***SSL-ID*: 0115h**

Name	Length	Description
INDEX	1word	Block type number: ■ 0800h: OB ■ 0A00h: DB ■ 0B00h: SDB ■ 0C00h: FC ■ 0E00h: FB ■ 8800h: VOB ■ 8A00h: VDB ■ 8B00h: VSDB ■ 8C00h: VFC ■ 8E00h: VFB
MaxAnz	1word	Maximum number of blocks of the type: ■ at OBs: – max. possible number of OBs for a CPU ■ at DBs: – max. possible number of DBs including DB0 ■ at SDBs: – max. possible number of SDBs including SDB2 ■ at FCs and FBs: – max. possible number of loadable blocks
MaxLng	1word	Maximum total size of the object to be loaded in kbytes
Maxabl	2words	Maximum length of the work memory part of a block in bytes

Record set

***SSL-ID*: 0815h**

Name	Length	Description
INDEX	1word	Block type number (VIPA specific) ■ 8800h: VOB ■ 8A00h: VDB ■ 8B00h: VSDB ■ 8C00h: VFC ■ 8E00h: VFB
MaxAnz	1word	Maximum number of blocks of the type: ■ at OBs: – max. possible number of OBs for a CPU ■ at DBs: – max. possible number of DBs including DB0 ■ at SDBs: – max. possible number of SDBs including SDB2 ■ at FCs and FBs: – max. possible number of loadable blocks
MaxLng	1word	Maximum total size of the object to be loaded in kbytes
Maxabl	2words	Maximum length of the work memory part of a block in bytes

14.8 Status of all LEDs - SSL-ID: xy19h

You obtain information about the status of all LEDs from your CPU.

Parameters

***SSL-ID*: xy19h**

SSL_ID	INDEX	Description
		Status of the LEDs
0019h	-	Status of all LEDs (without VIPA specific)
0119h	xxxxh	Status of one LED, to specify via INDEX
0E19h	0000h	Status of all VIPA specific LEDs
0F19h	-	SSL partial list header information

LENTHDR	one record set is 2words long (4byte)
N_DR	Number of record sets

Record set

***SSL-ID*: xy19h**

Name	Length	Description
INDEX	1word	LED-Kennung ■ 0001h: SF (Group error) ■ 0004h: RUN ■ 0005h: STOP ■ 0006h: FRCE (Force) ■ 0008h: BATF (always "0") ■ 000Bh: DP master BUSF1 (Bus error interface 1) ■ 000Ch: BF LED only at EtherCAT and PROFINET ■ 0015h: MT LED ■ 0100h: CP LED (VIPA specific) ■ 1000h: Access to memory card LED (VIPA specific) ■ 1001h: PROFIBUS Data Exchange slave LED (VIPA specific) ■ 2000h: PROFIBUS master RUN LED (VIPA specific, SLIO CPU = 0) ■ 2001h: PROFIBUS master ERR LED (VIPA specific) ■ 2002h: PROFIBUS master Data Exchange LED (VIPA specific) ■ 2003h: PROFIBUS master IF LED (VIPA specific, SLIO CPU = 0)
Led_on	1byte	Status of one LED: ■ 0: off ■ 1: on
Blink Code	1byte	Flashing status of the LEDs: (decimal) ■ 0: off ■ 1: flashing normally (2Hz) ■ 2: flashing slowly (0.5Hz) – **Note:** EtherCat systemic flashing frequency 1Hz ■ 3: flashing with 1Hz (VIPA specific LED) ■ 4: flashing with 4Hz (VIPA specific LED) ■ 5: flashing with 2.5Hz (VIPA specific LED) ■ 6: flashing with 10Hz (VIPA specific LED) ■ 7: cyclically: short (200 ms) flashes once then off for 1000ms. (VIPA specific LED) ■ 8: cyclical: flashes twice briefly (200ms) then off for 1000ms. (VIPA specific LED) ■ 9: cyclically: three short flashes (200ms) then off for 1000ms. (VIPA specific LED) ■ 10: cyclical: remains 4 seconds, then 2 seconds off. (VIPA specific LED)

14.9 Identification of the component - SSL-ID: xy1Ch

Description

If you read the partial list you can identify the CPU or the automation system.

Parameters

SSL_ID	INDEX	Description
001Ch	-	Identification of all components
011Ch		Identification of one component:
	0001h	Name of the automation system
	0002h	Name of the module
	0003h	Plant identification of the module
	0005h	Serial number of the module
	0006h	Reserved for the operating system
	0007h	Module type name
	0008h	Serial number of the memory card - CID without CardType
	000Ah	OEM identification of the module
	000Bh	Location identifier of the module
	00E0h	Serial number at the key file in the activated memory card (only at *SSL_ID 011Ch*)
	00E1h	Serial number at the key file in the plugged memory card (only at *SSL_ID 011Ch*)
	00FFh	Serial number of the memory card - CID with CardType (only at *SSL_ID 011Ch*)
0F1Ch	-	SSL partial list header information

LENTHDR	■ A record set is 17words long (34byte): – at INDEX < 00E0h ■ A record set is 5words long (10byte): – at INDEX = 00E0h, 00E1h ■ A record set is 19words long (38byte): – at INDEX = 00FFh
N_DR	Number of record sets ■ 0009h: at SSL: 001Ch ■ 0001h: at SSL: 011Ch

Record set

A record set of the partial list with *SSL_ID: 011Ch* has the following structure:

INDEX	Name	Length	Description
0001h	name	12words	Name of the automation system (max. 24 characters) *
	res	4words	reserved

INDEX	Name	Length	Description
0002h	name	12words	Name of the module (max. 24 characters) *
	res	4words	reserved
0003h	tag	16words	Plant identification of the module (max. 32 characters) *
	res	4words	reserved
0005h	serialn	12words	Serial number of the module (max. 24 characters) *
	res	3words	reserved
0007h	cputypname	16words	Module type name as character string (max. 32 characters) *
0008h	sn_cid	16words	Serial number of the memory card CID without Card-Type: (max. 32 characters) * CID without CardType:
			at MMC card: "MMC " + serial number
			at SD card: "SD " + serial number (Product serial number from CID)
			if no card is plugged: 0
000Ah	oem	1word	OEM identification of the module
000Bh	ok	1word	Location identifier of the module
00E0h	sn_act	1word	Serial number at the key file in the activated memory card (only at *SSL_ID* x11Ch)
00E1h	sn_plug	1word	Serial number at the key file in the plugged memory card (only at *SSL_ID* x11Ch)
00FFh	cid		Serial number of the memory card (only at *SSL_ID* x11Ch) CID with CardType:
		2words	Manufacturer ID
		2words	Application ID
		4words	Product Name
		2words	Product Revision
		2words	Product Serial Number
		2words	Manufacturer Month
		2words	Manufacturer Year
		2words	Card Type: ■ 0 = MMC ■ 1 = SD ■ 2 = SDHC
*) If names and designations are shorter than the corresponding max. characters, the gaps are filled with 00h.			

14.10 Interrupt status - SSL-ID: xy22h

Description

This partial list contains information about the current status of interrupt processing and interrupt generation in the module.

Parameters

SSL_ID	INDEX	Description
0222h		Record set on the specified interrupt. The interrupt class is to be specified via INDEX:
	0001h	OB 1 (Free cycle)
	000Ah	OB 10 (Time-of-day interrupt)
	000Bh	OB 11 (Time-of-day interrupt)
	0014h	OB 20 (Time-delay interrupt)
	0015h	OB 21 (Time-delay interrupt)
	001Ch	OB 28 (VIPA Watchdog Interrupt)
	001Dh	OB 29 (VIPA Watchdog Interrupt)
	0020h	OB 32 (Watchdog Interrupt)
	0021h	OB 33 (Watchdog Interrupt)
	0022h	OB 34 (Watchdog Interrupt)
	0023h	OB 35 (Watchdog Interrupt)
	0028h	OB 40 (Hardware Interrupt)
	0029h	OB 41 (Hardware Interrupt)
	0037h	OB 55 (Status Interrupt)
	0038h	OB 56 (Update Interrupt)
	0039h	OB 57 (Manufacturer Specific Interrupt)
	003Dh	OB 61 (Clock synchronous error)
	0050h	OB 80 (Asynchronous error)
	0051h	OB 81 (Asynchronous error)
	0052h	OB 82 (Asynchronous error)
	0053h	OB 83 (Asynchronous error)
	0055h	OB 85 (Asynchronous error)
	0056h	OB 86 (Asynchronous error)
	0057h	OB 87 (Asynchronous error)
	0064h	OB 100 (Reboot)
	0066h	OB 102 (Reboot)
	0079h	OB 121 (Synchronous error)
	007Ah	OB 122 (Synchronous error)

LENTHDR	A record set is 14words long (28bytes)
N_DR	Number of record sets (here always 1)

Record set

***SSL_ID*: xy22h**

Name	Length	Description
info	10words	Start info for the given OB, with following exceptions: ■ OB 1 provides the current minimum (in bytes 8 and 9) and maximum cycle time (in bytes 10 and 11) (time base: ms, byte count begins at 0). ■ When a job is active for a time-delay interrupt, bytes 8 ... 11 (byte count begins at) get the remaining time in ms left of the delay time set as a parameter. ■ OB 80 contains the configured minimum (in bytes 8 and 9) and maximum cycle time (in bytes 10 and 11) (time base: ms, byte count begins at 0). ■ Error interrupts without the current information. ■ Interrupts contain the status info from the current parameter settings of the interrupt source. ■ In the case of synchronous errors, the priority class entered is 7Fh if the OBs were not yet processed; otherwise, the priority class of the last call. ■ If an OB has several start events and these have not yet occurred at the information time, then event no. xyzzh is returned with: x: event class y: undefined zz: smallest defined number in the group Otherwise, the number of the last start event that occurred is used.
al 1	1word	Processing identifiers: ■ Bit 0: Interrupt event is caused by parameters: – 0: enabled – 1: disabled ■ Bit 1: Interrupt event as per SFC 39 DIS_IRT – 0: not locked – 1: locked ■ Bit 2: 1: Interrupt source is active (generation job ready for time interrupts, time-of-day/time-delay interrupt OB started, cyclic interrupt OB was configured) ■ Bit 4: Interrupt OB – 0: is not loaded – 1: is loaded ■ Bit 5: Interrupt OB is by TIS: – 0: enabled – 1: disabled ■ Bit 6: Entry in diagnostic buffer – 0: enabled – 1: disabled ■ Bit 15 ... 7: reserved

Name	Length	Description
al 2	1word	Reaction with not loaded/locked OB ■ Bit 0: 1: Lock interrupt source ■ Bit 1: 1: Generate interrupt event error ■ Bit 2: 1: CPU goes into STOP mode ■ Bit 3: 1: Interrupt only discarded ■ Bit 15 ... 4: reserved
al 3	2words	Discard by TIS functions

Record set

SSL_ID: 0222h **INDEX: 003Dh**

The data set contains the local data of OB 61 and further information on the status to the OB 61.

Name	Length	Description
OB61_EV_CLASS	1byte	Event class and identifiers: 11h: Alarm is active
OB61_STRT_INF	1byte	64h: Start request for OB 61
OB61_PRIORITY	1byte	Assigned priority class Default value: 25
OB61_OB_NUMBR	1byte	OB number: 61 ... 64
OB61_RESERVED_1	1byte	reserved
OB61_RESERVED_2	1byte	reserved
OB61_GC_VIOL	1bit	GC violation at PROFIBUS DP
OB61_FIRST	1bit	First run after start up or stop state
OB61_MISSED_EXEC	1byte	Number of failed OB 61 starts since the last OB 61 execution
OB61_DP_ID	1byte	PROFINET-IO system ID of the clock synchronous PN IO system (100 ... 115)
OB61_RESERVED_3	1byte	reserved
OB61_RESERVED_4	2bytes	reserved
OB61_DATE_TIME	8bytes	Date and time of day when the OB was called
al 1	2bytes	Processing identifiers (see below)
al 2	2bytes	Reaction with not loaded/locked OB (see below)
al 3	4bytes	Discard by TIS functions (see below)

Additional status information OB 61:

Name	Length	Description
al 1	2 Bytes	Processing identifiers: ■ Bit 0: Interrupt event is caused by parameters: – 0: enabled – 1: disabled ■ Bit 1: Interrupt event as per SFC 39 DIS_IRT: – 0: not locked – 1: locked ■ Bit 2: (Generation job ready for time interrupts, time-of-day/ time-delay interrupt OB started, cyclic interrupt OB was configured) – 0 = not active – 1 = Interrupt source is active ■ Bit 4: Interrupt OB: – 0: is not loaded – 1: is loaded ■ Bit 5: Interrupt OB is by TIS: – 0: enabled – 1: disabled ■ Bit 6: Entry in diagnostic buffer: – 0: enabled – 1: disabled ■ Bit 15 ... 7: reserved
al 2	2 Bytes	Reaction with not loaded / locked OB ■ Bit 0: 1: Lock interrupt source ■ Bit 1: 1: Generate interrupt event error ■ Bit 2: 1: CPU goes into STOP mode ■ Bit 3: 1: Interrupt only discarded ■ Bit 15 ... 4: reserved
al 3	4 Bytes	Discard by TIS functions Bit number x set means: The event number that is larger than the smallest event to x the according event number is discard by TIS function.

14.11 Communication status data - SSL-ID: xy32h

Description

If you read this partial list you obtain the status data of the CPU communication section.

Parameters

SSL_ID	INDEX	Description
0132h		Status data of the CPU communication section
	0001h	General of communication status data.

SSL_ID	INDEX	Description
	0002h	TIS status data
	0004h	Protection status data
	0006h	Data exchange via communication SFB
	0008h	Time system (16bit Run-time meter 0 ... 7)
	0009h	MPI status data
	000Ah	K-Bus status data
	000Bh	Time system (32bit Run-time meter 0 ... 7)

LENTHDR	A record set is 20words long (40bytes) The assignment depends on the INDEX parameter
N_DR	Number of record sets

Record set

SSL_ID: 0132h *INDEX: 0001h*

The partial list extract contains information about general of communication status data.

Name	Length	Description
INDEX	1word	General condition data for communication
	1word	Reserved number of PG connections (Default = 1)
	1word	Reserved Number of OP connections (Default = 1)
	1word	Number of occupied PG connections
	1word	Number of occupied OP connections
	1word	Number of configured S7 connections (Default = 0)
	1word	Number of occupied S7 connections
	1word	Number of unused connection resources
	1word	reserved
	1word	Max. preset communication load of the CPU in % (Default = 20%)
	6words	reserved (0000h)
	1byte	reserved (00h)
	1byte	Reserved number S7 basic communication connections (Default = 0)
	1byte	Number of occupied S7 basic communication connections (XPut/XGet/MPI)
	1byte	reserved (00h)

Name	Length	Description
	1word	Number of occupied other connections
	1word	Dialog mode switching (communication dialog) via Siemens SIMATIC Manager: ■ 0000h: communication dialog – Siemens CPU 318 – VIPA CPU 317-4NE12 ■ 0001h: communication dialog – VIPA CPU 315-2AG10 – VIPA CPU 317-2AJ10 ■ 0002h: reserved ■ 0003h: communication dialog – Siemens CPU 315-2EH13 FW: V2.6 – Siemens CPU 317-4EK14 FW: V3.x

Record set

SSL_ID: 0132h *INDEX: 0002h*

The partial list extract contains information about the TIS status data.

Name	Length	Description
INDEX	1word	0002h: TIS status
	1word	Number of furnished TIS orders
	18words	reserved

Record set

SSL_ID: 0132h *INDEX: 0004h*

The partial list extract contains information about protection status data.

Name	Length	Description
INDEX	1word	0004h: Protection status
	1word	Protection at the key switch (possible value : 1, 2 or 3)
	1word	Configured protection level (possible values: 0, 1, 2 or 3 0: no password, parameterized protection level is invalid)
	1word	Valid protection level of the CPU (possible values: 1, 2 or 3)

Name	Length	Description
	1word	Position of the mode switch: ■ 0: undefined or can not be determined ■ 1: RUN ■ 2: RUN_P ■ 3: STOP ■ 4: MRES
	1word	Position of the mode CRST/WRST: ■ 0: undefined or can not be determined ■ 1: CRST (Cold Restart) ■ 2: WRST (Warm Restart)
	14words	reserved

Record set

SSL_ID: 0132h *INDEX: 0006h*

The partial list extract contains information about data exchange via communication SFB of configured connections.

Name	Length	Description
INDEX	1word	0006h: Data exchange via communication SFB of configured connections
	1words	Used SFB blocks
	1byte	reserved
	1word	Number of loaded SFB instances
	1word	Number of multicast used blocks
	25byte	reserved

Record set

SSL_ID: 0132h *INDEX* 0008h

The partial list extract contains information about the status of the 16bit run-time meter 0 ... 7.

Name	Length	Description
index	1word	0008h: Time system status
zykl	1word	Cycle time of the synchronization telegram
korr	1word	Correction factor for the time
clock 0	1word	Run-time meter 0: time in hours
clock 1	1word	Run-time meter 1: time in hours
clock 2	1word	Run-time meter 2: time in hours
clock 3	1word	Run-time meter 3: time in hours
clock 4	1word	Run-time meter 4: time in hours

Name	Length	Description
clock 5	1word	Run-time meter 5: time in hours
clock 6	1word	Run-time meter 6: time in hours
clock 7	1word	Run-time meter 7: time in hours
time	4words	Current date and time (format: date_and_time)
bszl_0	1byte	■ Bit x: Run-time meter x with 0 ≤ x ≤ 7 – 1: Run-time meter active
bszl_1	1byte	reserved
bszü_0	1byte	■ Bit x: Run-time meter overflow x with 0 ≤ x ≤ 7 – 1: overflow
res	1byte	reserved
res	3words	reserved

status	Time status															
	High byte								Low byte							
Bit number	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SG	correction value					-	-	hr	su/ wi	-	res		-	-	sync

Bit	Description	Default value
0	Synchronization failure This parameter indicates whether the time transmitted in the frame from an external time master is synchronized. ■ 0: synchronization failed ■ 1: synchronization occurred **Note:** Evaluation of this bit in a CPU is only useful if there is continuous external time synchronization.	0
1	Parameter is not used.	0
2	Parameter is not used.	0
4, 3	Time resolution ■ 00: 0.001s ■ 01: 0.01s ■ 10: 0.1s ■ 11: 1s	00
5	Parameter is not used.	0

Bit	Description	Default value
6	Summer/winter time indicator The parameter indicates whether the local time calculated using the correction value is summer or winter time. ■ 0: winter time ■ 1: summer time	0
7	Notification hour This parameter indicates whether the next time adjustment also includes a switchover from summer to winter time or vice versa. ■ 0: no adjustment made ■ 1: adjustment made	0
8	reserved	0
9	reserved	0
14 ... 10	Correction value (Local time = basic time ± correction value * 0.5h) This correction takes into account the time zone and the time difference.	00000
15	Sign for the correction value ■ 0: positive ■ 1: negative	0

Record set

SSL_ID: 0132h *INDEX: 0009h*

The partial list extract contains information about the status data of the MPI.

Name	Length	Description
Index	1word	0009h: MPI status data
	1words	Used Baud rate (hexadecimal code)
	17words	reserved

Record set

SSL_ID: 0132h *INDEX: 000Ah*

The partial list extract contains information about the status data of the K-Bus.

Name	Length	Description
Index	1word	000Ah: K-Bus status data
	2words	Used Baud rate (hexadecimal code)
	17words	reserved

Record set

SSL_ID: 0132h *INDEX: 000Bh*

The partial list extract contains information about the status of the 32bit run-time meter 0 ... 7.

Name	Length	Description
index	1word	000Bh: Time system status
bszl_0	1byte	■ Bit x: Run-time meter x with 0 ≤ x ≤ 7 – 1: Run-time meter active
res	1byte	reserved
bszü_0	1byte	■ Bit x: Run-time meter overflow x with 0 ≤ x ≤ 7 – 1: overflow
res	1byte	reserved
clock 0	1Dword	Run-time meter 0: time in hours
clock 1	1Dword	Run-time meter 1: time in hours
clock 2	1Dword	Run-time meter 2: time in hours
clock 3	1Dword	Run-time meter 3: time in hours
clock 4	1Dword	Run-time meter 4: time in hours
clock 5	1Dword	Run-time meter 5: time in hours
clock 6	1Dword	Run-time meter 6: time in hours
clock 7	1Dword	Run-time meter 7: time in hours
res	1word	reserved

14.12 Ethernet details of the module - SSL xy37h

Description

With this partial list you get information about the configuration of the TCP/IP stack, the vendor specified MAC address and the connection properties on layer 2 - security layer (data link layer) of the CP interface of the PROFINET CPU.

Parameters

SSL_ID	INDEX	Description
0037h		Details of all Ethernet interfaces
	0000h	if the details of all Ethernet interfaces are requested
0137h		Details of an Ethernet interface
	xxxxh	Logical base address of the Ethernet interface, which details are requested
0F37h	xxxxh	SSL partial list header information

LENTHDR	One record set is 24words long (48bytes)
N_DR	Number of record sets

Record set *SSL_ID*: xy37h

Name	Length	Description
logaddr	2byte	Logical base address of the interface
ip_addr	4byte	IP address The IP address is stored in the following format (at the example a.b.c.d): Offset x: a, Offset x+1: b, Offset x+2: c, Offset x+3: d
subnetmask	4byte	subnet mask The subnet mask is stored in the following format (at the example a.b.c.d): Offset x: a, Offset x+1: b, Offset x+2: c, Offset x+3: d
defaultrouter	4byte	IP address of the default router If you have not configured a default router, here the IP address of the interface is entered.
mac_addr	6byte	MAC address
source	1byte	Origin of the IP address: ■ 00h: IP address is not initialized ■ 01h: IP address was configured ■ 02h: IP address was set by DCP ■ 03h: IP address comes from a DHCP server ■ 04h ... FFh: reserved
reserved	1byte	reserved
dcp_mod_ timestamp	8byte	Time stamp of the last change of the IP address via DCP **Note:** The content of this field may be evaluated only when bit 1 is set in source.

Name	Length	Description
phys_mode1	1byte	State of port 1: ■ Bit 0: Duplex mode (only relevant if AUI-Mode = 0): – 1: phys. Layer works full duplex – 0: phys. Layer works half duplex ■ Bit 1: Baud rate ID (only relevant if AUI-Mode = 0): – 1: phys. Layer works with 100MBaud – 0: phys. Layer works with 10MBaud ■ Bit 2: Link status: – 1: phys. Layer has link pulses – 0: phys. Layer has no link pulses ■ Bit 3: Auto mode: – 1: phys. Layer has to automatically adjust to the LAN medium – 0: phys. Layer will not automatically adjust to the LAN medium ■ Bit 6 ... 4: 0 ■ Bit 7: Validity: – 0: phys_mode1 has no valid data – 1: phys_mode1 has valid data The numbering of the ports is identical to the numbering in the configuration. If the interface has only one port , whose physical properties are entered at port 1.
phys_mode2	1byte	State of port 2 (structure like phys_mode1)
...	...	...
phys_mode16	1byte	State of port 16 (structure like phys_mode1)
reserved	2byte	reserved

If you have not carried out any IP configuration, the variables ip_addr , subnetmask and defaultrouter each have the value zero.

14.13 TCON Connection - SSL-ID: xy3Ah

Description

If you read this partial list, you obtain information of the TCON connection from qualified CPUs.

The "Open Communication via Industrial Ethernet" in the Siemens SIMATIC Manager dialog is visible only when the SSL 003Ah and 0F3Ah exist and are available. For this, you must be entered in the table of contents (SSL 0000h).

The diagnostic data that can be read by the SSL, will be updated by the system with a period of one second.

Parameters

SSL_ID	INDEX	Description
xy3Ah		Status TCON connection
003Ah	xxxxh	Read diagnostic information
0F3Ah	xxxxh	Only header

LENTHDR	Length of following record set is 74words (148byte)
N_DR	■ 0: TCON Online Diagnostics is not possible ("Diagnostics" button in the Siemens SIMATIC Manager = "gray"). It is delivered only the header and no further user data. ■ >0: TCON Online Diagnosis enabled

Record set

SZL_ID: xy3Ah *INDEX: 003Ah*

If you read this partial list, you obtain information of the TCON connection from qualified CPUs.

Name	Length	Description
003Ah	1word	0100h: unknown
	1word	"current connection number": not connection ID
	1word	Block_length[3] 40h: Up to Offset 4 … 67 = 64 byte
	1word	ID[3]: connection ID
	1byte	■ connection_type[3]: – 11h = TCP/IP – 12h = ISO on TCP – 13h = UDP – 01h = TCP (compatibility mode)
	1byte	active_est[3]
	1byte	local_device_id[3] 02h: CPU Type
	1byte	local_tsap_id_len[3]
	1byte	rem_subnet_id_len[3]
	1byte	rem_staddr_len[3] 04h: für IP-Adresse
	1byte	rem_tsap_id_len[3]
	1byte	next_staddr_len[3]
	16byte	local_tsap_id (include TSAP or Port number)[3]

Name	Length	Description
	6byte	rem_subnet_id[3] for routing
	6byte	rem_staddr (remote IP address)[3]
	16byte	rem_tsap_id (include TSAP or Port number)[3]
	6byte	next_staddr (next IP address)[3] for routing
	1word	spare[3]
	4byte	local_staddr (local IP address)[3]
	8byte	1. timestamp[1] timestamp for 1. call attempt
	8byte	2. timestamp[1] Storage for timestamp 4 at disconnection
	8byte	3. timestamp[1] Timestamp, the error message of the last disconnection In this purpose there is an error number (Offset: 132)
	8byte	4. timestamp[1] Timestamp for successful connection Is copied in disconnection by timestamp 2 and deleted (reset all to 0)
	8byte	5. timestamp[1] Timestamp of the last failed connection attempt In this purpose there is an error number (Offset: 130)
	4byte	rem_ip_addr (remote IP address)[4]
	2byte	rem_port_nr (remote Port number)[4]
	2byte	spare[4]
	4byte	rem_ip_addr (remote IP address)[5]
	2byte	rem_port_nr (remote Port number)[5]
	2byte	spare[5]
	1word	■ State of connection: – 0000h: no display – 0001h: Connection is established – 0002h: no display – 0003h: Connection is established passive – 0004h: Connection is established active – 0005h: Connection is established passive – > 0005h: no display

Name	Length	Description
	1word	■ Error message of the last connection attempt: – 0000h: no error – 0001h: local network error – 0002h: participant not available – 0003h: local abort – 0004h: abort by partner – 0005h: abort due to timeout – 0006h: abort by protocol error – 0007h: system internal error (7) – 0008h: system internal error (8) – 0009h: system internal error (9) – 000Ah: system internal error (10) – 000Bh: call attempt to own station address – 000Ch: double addressing – ≥ 000Dh: unknown error
	1word	■ – Error message from the last disconnection: – 0000h: no error – 0001h: local network error – 0002h: participant not available – 0003h: local abort – 0004h: abort by partner – 0005h: abort due to timeout – 0006h: abort by protocol error – 0007h: system internal error (7) – 0008h: system internal error (8) – 0009h: system internal error (9) – 000Ah: system internal error (10) – 000Bh: Call attempt to own station address – 000Ch: double addressing – ≥ 000Dh: unknown error
	1word	Current connection attempts; is reset when connected
	1Dword	Number of bytes sent
	1Dword	Number of bytes received
	1word	Number of successful connection attempts
	1word	0000h: unknown

1) Time stamp (data type: S7 Date and Time), resolution in seconds, milliseconds are zeroed

3) Fields corresponding TCON Config DB (UDT65). Fields rem_staddr_len, rem_tsap_id_len, rem_staddr and rem_tsap_id updated when connected with address data of the communication partner

4) Fields according to the address of DB TUSEND (UDT66)

5) Fields according to the address of DB TURCV by calling (UDT66)

14.14 Status of the LEDs - SSL-ID: xy74h

Description

This partial list contains information about the LEDs of the CPU.

Parameters

SSL_ID	INDEX	Description
0074h	-	Record sets of all CPU LEDs
0174h		Record set of one CPU LED:
	0001h	SF (group error)
	0004h	RUN
	0005h	STOP
	0006h	FRCE (Force)
	0008h	BATF: 0 (fix) INDEX exists only at CPUs that are configured as CPU 318.
	000Bh	BF (Bus error interface 1)
	000Ch	BF LED only at EtherCAT, PROFINET and PROFIBUS (ERROR)
	0015h	MT LED (VIPA specific)
	0100h	CP LED (VIPA specific)
	1000h	Memory card (VIPA specific)
	1001h	PROFIBUS slave DE (VIPA specific, not for SLIO CPU)
	2000h	PROFIBUS master RUN (VIPA specific)
	2001h	PROFIBUS master ERR (VIPA specific)
	2002h	PROFIBUS master DE (VIPA specific)
	2003h	PROFIBUS master IF (VIPA specific)
0E74h		Records sets of all CPU status LEDs also PROFIBUS DP master/slave if available.
	0000h	INDEX = 0000h (mandatory)

LENTHDR	A record set is 2words long (4bytes).
N_DR	Number of record sets

Record set

***SSL_ID*: xy74h**

Name	Length	Description
index	1word	■ INDEX of the LED (if exist) – 0001h: SF (group error) – 0004h: RUN – 0005h: STOP – 0006h: FRCE (Force) – 0008h: BATF: 0 (fix) – 000Bh: DP master BUSF1: 0 (fix) – 000Ch: DP master group error (ERROR) – 0015h: Virtual PN SF LED (VIPA specific) – 1000h: Memory card (VIPA specific) – 1001h: PROFIBUS slave DE (VIPA specific) – 2000h: PROFIBUS master RUN (VIPA specific) – 2001h: PROFIBUS master ERR (VIPA specific) – 2002h: PROFIBUS master DE (VIPA specific) – 2003h: PROFIBUS master IF (VIPA specific)
led_on	1byte	■ Status of the LED: – 0: off – 1: on
blink_code	1byte	■ Flashing status of the LED (decimal): – 0: not flashing – 1: flashing normally (2Hz) – 2: flashing slowly (0.5Hz) – 3: flashing with 1Hz (VIPA specific LED) – 4: flashing with 4Hz (VIPA specific LED) – 5: flashing with 2.5Hz (VIPA specific LED) – 6: flashing with 10Hz (VIPA specific LED) – 7: cyclically: short (200 ms) flashes once then off for 1000ms. (VIPA specific LED) – 8: cyclical: flashes twice briefly (200ms) then off for 1000ms. (VIPA specific LED) – 9: cyclically: three short flashes (200ms) then off for 1000ms. (VIPA specific LED) – 10: cyclical: remains 4 seconds, then 2 seconds off. (VIPA specific LED)

14.15 Status information CPU - SSL-ID: xy91h

Description

If you read the partial list, you obtain the status information of modules assigned to the CPU. In this manual are only the available partial lists for the EtherCAT-CPUs described. Not described are SSL partial list: W#16#0191, W#16#0291, W#16#0391, W#16#0591, W#16#0991.

Parameters

SSL_ID	INDEX	Description
0091h	-	Module status information of all plugged and projected modules/submodules from the CPU
0A91h	-	Module status information of a module in the central structure or at an integrated bus system (PROFIBUS, PROFINET or EtherCAT) via the logical base address.
0C91h	adr	Module status information of a module an external bus interface (PROFIBUS, PROFINET or EtherCAT) via the logical base address. xxxx: Bits 0 ... 14: any logical address of the module ■ Bit 15: – 0 = Input – 1 = Output
4C91h	xxxxh	Module status information of all modules of the rack or the station (central, decentral PROFIBUS DP, PROFINET-IO or EtherCAT). xxxx: Bits 0 ... 14: any logical address of the module ■ Bit 15: – 0 = Input – 1 = Output
0D91h		Module status information of all configured modules (central, decentral PROFIBUS DP, PROFINET-IO or EtherCAT)
	xx00h	Modules or submodules from the rack or station number. With xx you have to specify the number of the rack.

SSL_ID	INDEX	Description
	xxyyh	xxyy: all modules of a DP station or a PROFINET-IO station or an EtherCAT station PROFIBUS DP: xx include master system ID, yy station number; ■ PROFINET-IO: – Bit 0 ... 10: Device number – Bit 11 ... 14: the last two digits of the PN IO Subsystem ID – Bit 15: 1 ■ EtherCAT: – Bit 0 ... 10: Slave number – Bit 11 ... 14: the last two digits of the EtherCAT Subsystem ID – Bit 15: 1
0E91h	-	Module status information of all assigned modules.

LENTHDR	A record set is 8words long (16bytes).
N_DR	Number of record sets; product-specific, the number of transmitted record set can be less

Additional Record sets

In the case of *SSL_ID* 0091h, 0191h and 0F91h two additional record sets are supplied per rack:

- A record for the power supply if it exists
- A record set for the rack

The sequence of the records in case of a centralized structure is:

Power supply, slots 1 ... n, rack

Record set

***SSL_ID*: xy91h:**

Name	Length	Description
adr1	1word	*'adr1' on page 584*
adr2	1word	*'adr2' on page 585*
logadr	1word	First assigned logical I/O address (base address).
solltyp	1word	Target type: only at PROFINET or EtherCAT (otherwise reserved)
isttyp	1word	Actual type: only at PROFINET or EtherCAT (otherwise reserved)

Name	Length	Description
reserved	1word	■ At PROFINET-IO or EtherCAT (otherwise reserved): – SSL-ID=0C91h: Number of really existing submodules (without submodule 0) – SSL-ID=0D91h: Number of submodules (without submodule 0) – SSL-ID=4C91h: Number of really existing submodules (without submodule 0) – SSL-ID=4D91h: Number of really existing submodules (without submodule 0)
eastat	1word	■ I/O status: – Bit 0: 1: Module error (detected by diagnostic interrupt) – Bit 1: 1: Module exists – Bit 2: 1: Module does not exist – Bit 3: 1: Module disabled – Bit 4: 1: Station error – Bit 5: 1: A CiR event at this module /station is busy or not yet completed. – Bit 6: 1: reserved – Bit 7: 1: Module in local bus segment – Bit 8 ... 15: Data ID for logical address (Input: B4h, Output: B5h, DP interface: FFh)
ber_bgbr	1word	■ Area ID/module width – Bit 0 ... 2: Module width – Bit 3: reserved – Bit 4 ... 6: Area ID
		0: Siemens S7-400
		1: Siemens S7-300
		2: ET area (PROFIBUS/PROFINET/EtherCAT-decentralized)
		3: P area
		4: Q area
		5: IM3 area
		6: IM4 area
		7: Consistent area (PROFIBUS slave)
		Bit 7 ... 15: reserved

adr1 *At a centralized configuration*

	adr1															
	High byte								Low byte							
Bit number	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0								Rack number (0 ... 31)							

At a decentralized configuration with PROFIBUS DP

Bit 15: 0 is the ID for PROFIBUS

	adr1															
	High byte								Low byte							
Bit number	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	DP master system ID (1 ... 32)							Station number (0 ... 127)							

At a decentralized configuration with PROFINET IO or EtherCAT

To obtain the full PROFINET IO system ID, you have to add 100 (decimal) to bit 12 ... 14.

Bit 15: 1 is the ID for PROFINET or EtherCAT

	adr1															
	High byte								Low byte							
Bit number	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	PROFINET IO system ID (0 ... 15)			Station number (0 ... 2047)											

adr2

At a centralized respectively decentralized structure with PROFIBUS DP

	adr2															
	High byte								Low byte							
Bit number	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Slot number								Submodule slot number							

adr2

Slot number: for a decentralized configuration with PROFINET-IO or EtherCAT.

14.16 Stations status information (DPM) - SSL-ID: xy92h

Description

If you read this partial list, you obtain information about the expected and the current hardware configuration of centrally installed stations of a DP master system, connected via a DP interface.

Parameters

SSL_ID	INDEX	Description
0092h	DPM-ID	Expected status of the central stations of a DP master system.
0292h	DPM-ID	Actual status of the stations of a DP master system.
0692h	DPM-ID	Diagnostic status of the expansion racks in the central configuration of the stations of a DP master system.
4092h	DPM-ID	Expected status of a DP master system, which is connected via an **external** DP switch.
4192h	DPM-ID	Activation status of a DP master system, which is connected via an **external** DP switch.
4292h	DPM-ID	Actual status of a DP master system, which is connected via an **external** DP switch.
4692h	DPM-ID	Diagnostic status of the expansion racks of a DP master system, which is connected via an **external** DP switch.

LENTHDR	One record set is 8words long (16bytes).
N_DR	Number of record sets

Record set

SSL_ID: xy92h:

Name	Length	Description		
status_0 ... status_15	16byte	Rack status / station status or backup status (the backup status is only relevant for DP modules).		
		0092h:	0:	Rack/station not configured
			1:	Rack/station configured
		4092h:	0:	Station not configured
			1:	Station configured
		0292h:	0:	Rack/station failure, deactivated or not configured
			1:	Rack/station exists, is activated and has not failed
		0692h:	0:	All modules of the expansion rack / of a station exist, are available with no problems and activated.

Name	Length	Description		
			1:	At least 1 module of the expansion rack / of a station is not OK or the station is deactivated.
		4692h:	0:	All modules of a station exist, are available with no problems, and activated.
			1:	At least 1 module of a station is not OK or the station is deactivated.
status_0	1byte	Bit 0:		Central rack (INDEX = 0) or station 1 (INDEX > 0)
		Bit 1:		1. Expansion rack or station 2
		...		...
		Bit 7:		7. Expansion rack or station 8
status_1	1byte	Bit 0:		8. Expansion rack or station 9
		...		...
		Bit 7:		15. Expansion rack or station 16
status_2	1byte	Bit 0:		16. Expansion rack or station 17
		...		...
		Bit 5:		21. Expansion rack or station 22
		Bit 6:		0: or station 23
		Bit 7:		0: or station 24
status_3	1byte	Bit 0:		0: or station 25
		...		...
		Bit 5:		0: or station 30
		Bit 6:		Expansion rack in Siemens S5 area or station 31
		Bit 7:		0: or station 32
status_4	1byte	Bit 0:		0: or station 33
		...		...
		Bit 7:		0: or station 40
...	...	...		...
status_15	1byte	Bit 0:		0: or station 121
		...		...
		Bit 7:		0: or station 128

14.17 Stations status information (DPM, PROFINET-IO and EtherCAT) - SSL-ID: xy94h

Description

If you read this partial list, you obtain information about the expected and the current hardware configuration of centrally installed stations of a DP master system / PROFINET IO controller system or EtherCAT master system.

Parameters

SSL_ID	INDEX	Description
0094h	PN-ID	Expected status of the central stations of a PROFINET-IO control system / PN IO sub-system-ID With EtherCAT only the stations configured as *mandatory* are registered. ■ Status bit = 1: – Rack/station configure
0194h	PN-ID	Activation status of a station of an IO controller system, which is configured and disabled. ■ Status bit = 1
0294h	PN-ID	Actual state of the rack in the central configuration of the stations of an IO controller system ■ Status bit = 1: – Rack/station exists, activated and not failed
0694h	PN-ID	Diagnostic status of the expansion racks in the central configuration of the stations of a PROFINET-IO control system / PN IO subsystem-ID ■ Status bit = 1: – at least one module of rack/station has malfunction or is de-activated: coming diagnostics interrupt, neighbourhood interrupt, remove/fit interrupt, failure mandatory station
0794h	PN-ID	Diagnostic / Maintenance condition of the central stations of a PROFINET-IO control system / PN IO subsystem-ID ■ Status bit = 0: – no problem and no maintenance necessary ■ status bit = 1: – rack/station has a problem or maintenance requirement or maintenance request

SSL_ID	INDEX	Description
0994h	PN-ID	Set point - actual value difference ■ Status bit = 1: – Set point - actual value difference in station exists ModDiffBlock, EC state unequal master state
0A94h	PN-ID	Set point state of the stations of an EtherCAT IO controller system. In this partial list besides the *mandatory* stations additionally the *optional* configured stations are registered. ■ Status bit = 1: – Rack/station configured
0F94h		only header information

LENTHDR	One record set is 129words long (258bytes).
N_DR	Number of record sets

Record set

SSL_ID: xy94h

Name	Length	Description
INDEX	1word	■ 0: Central module ■ 1 ... 31: Distributed module at PROFIBUS DP ■ 100 ... 115: Distributed module at PROFINET-IO / EtherCAT-IO
status_0	BOOL	■ Group information: – 1: one of the following status bits has the value 1 – 0: all subsequent status bits have the value 0
status_1	BOOL	Status station 1
status_2	BOOL	Status station 2
...		...
status_2047	BOOL	Status station 2047

A status bit of non-configured racks/stations/devices has the value 0.

Note!

Important difference to the previous SSL ID xy92h:

Compared to the previous SSL ID xy92h, the data have been shifted by one bit since bit status_0 is used for group information.

Local SLIO-Bus

- *A virtual PN device on the PROFINET network is configured for the SLIO CPU of a local SLIO bus. The corresponding SSLs xy94h is filled with this configured station number.*
- *If no virtual PN Device for the SLIO bus is configured, then natively for the station number 2047 is used.*

EtherCAT-Bus

- *A virtual PN device is configured on the PROFINET network for the EtherCAT network. The corresponding SSLs xy94h is filled with this configured station number.*
- *The EtherCAT master (controller) has normally the station number 0. This can not be located in the SSL xy94h because the bit 0 is used as a group bit. Thus is set in Topology Mismatch in the SSL xy94h the bit for the station 512 (maximum station number for EtherCAT).*

14.18 Status information PROFINET/EtherCAT/PROFIBUS DP - SSL-ID: xy96h

Description

This partial list contains status information on all the modules assigned to the CPU. It provides information specific to PROFINET IO as well as information on PROFIBUS DP modules, EtherCAT modules and central modules. Complementing *SSL_ID* xy91 you get on the partial list with the *SSL_ID* xy96 additional state information of modules and submodules.

Parameters

SSL_ID	INDEX	Description
0696h	logadr	Module status information of all submodules of a specified module (only with integrated interface with PROFINET IO) address with IO identifier.
0C96h	logadr	Module status information of a module/ submodule located centrally or on a PROFIBUS DP/PROFINET/EtherCAT interface module over the start address.

LENTHDR	Length of the data record is 24words (48byte).
N_DR	Number of record sets

Record set

***SSL_ID:* xy96h:**

<table>
<tr><th>Name</th><th>Length</th><th colspan="2">Description</th></tr>
<tr><td>logadr</td><td>1word</td><td colspan="2">■ Bits 0 ... 14: Address of the module
■ Bit 15: 0 = Input, 1 = Output</td></tr>
<tr><td>System</td><td>1word</td><td colspan="2">ID for centralized module / DP master system ID / PROFINET IO system ID / EtherCAT system:
■ 0: Central Module
■ 1 ... 31: Decentralized module at PROFIBUS DP
■ 100 ... 115: Decentralized module at PROFINET-IO / EtherCAT-IO</td></tr>
<tr><td>res</td><td>2words</td><td colspan="2">Not relevant</td></tr>
<tr><td>Station</td><td>1word</td><td colspan="2">Rack no./station number/device number</td></tr>
<tr><td>Slot</td><td>1word</td><td colspan="2">Slot number</td></tr>
<tr><td>Subslot</td><td>1word</td><td colspan="2">Submodule slot number
(Enter 0 if no submodule can be installed)</td></tr>
<tr><td>res</td><td>1word</td><td colspan="2">Not relevant</td></tr>
<tr><td rowspan="9">Set point type</td><td rowspan="2">7words</td><td colspan="2">Set point type:
With PROFINET-IO the structure of the set point type is hierarchical</td></tr>
<tr><td>PROFINET-IO / EtherCAT-IO</td><td>PROFIBUS DP</td></tr>
<tr><td>1. word:</td><td>Vendor number or profile ID</td><td>0000</td></tr>
<tr><td>2. word:</td><td>Product code (High Word)</td><td>0000</td></tr>
<tr><td>3. word:</td><td>Product code (Low Word)</td><td>0000</td></tr>
<tr><td>4. word:</td><td>1. Word of the double word Module identification</td><td>Type ID</td></tr>
<tr><td>5. word:</td><td>2. Word of the double word Module identification</td><td>0000</td></tr>
<tr><td>6. word:</td><td>1. Word of the double word submodule identification
with EtherCAT-IO: reserved</td><td>0000</td></tr>
<tr><td>7. word:</td><td>2. Word of the double word submodule identification
with EtherCAT-IO: reserved</td><td>0000</td></tr>
<tr><td>Soll_ungleic _Ist_typ</td><td>1word</td><td colspan="2">ID set point/actual
■ Bit 0 = 0: Set point equal actual
■ Bit 0 = 1: Set point unequal actual
■ Bit 1 ... 15: reserved</td></tr>
<tr><td>reserved</td><td>1word</td><td colspan="2">reserved</td></tr>
</table>

Name	Length	Description
eastat	1word	I/O status: ■ Bit 0: 1: Module has malfunction (detected by diagnostics) ■ Bit 1: 1: Module exists ■ Bit 2: 1: Module not available ■ Bit 3: 1: Module de-activated ■ Bit 4: 1: Station has malfunction ■ Bit 5, 6: reserved ■ Bit 7: 1: Module in local bus segment ■ Bit 8: 1: Module maintenance required ■ Bit 9: 1: Module maintenance request ■ Bit 10 ... 15: reserved
ber_bgbr	1word	Area ID/module width ■ Bit 0 ... 2: Module width ■ Bit 3: reserved ■ Bit 4 ... 6: Area ID – 0: Siemens S7-400 – 1: Siemens S7-300 – 2: PROFINET IO (decentralized) – 3: P area – 4: Q area – 5: IM3 area – 6: IM4 area – 7: EtherCAT (decentralized) ■ Bit 7 ... 15: reserved
reserve	5words	reserved

Note!

Partial List with SSL-ID 0696h for modules on PROFIBUS DP: This results in the error message "sub-module level not present".

14.19 Diagnostic buffer of the CPU/CP - SSL-ID: xyA0h

Description

If you read the partial list, you obtain the entries of the diagnostic buffer of your CPU or your CP.

Parameters

SSL_ID	INDEX	Description
00A0h	-	Shows all entries of the diagnostics buffer, which are possible in the current mode.
01A0h	xxxxh	Shows the most recent entries of the diagnostics buffer. Here you specify the number of INDEX.
0FA0h	-	SSL partial list header information

LENTHDR	A record set is 10words long (20bytes).
N_DR	Number of record sets

Record set

SSL_ID: 00A0h and 01A0h

Name	Length	Description
ID	1word	Event ID
Pk	1byte	Depending on the diagnostic buffer entry
ObNr	1byte	Depending on the diagnostic buffer entry
DatId	1word	Depending on the diagnostic buffer entry
ZInfo1	1word	Information about the event
ZInfo2	1word	Information about the event
ZInfo3	1word	Information about the event
time	4words	Time stamp of the event (DATE_AND_TIME)

DATE_AND_TIME

DATE_AND_TIME: BCD format

Bytes	Description	Area
0	year	1990 ... 2089
1	month	01 ... 12
2	day	1 ... 31
3	hour	0 ... 23
4	minute	0 ... 59
5	second	0 ... 59
6	■ 2 MSD from ms – MSD: Most Significant Decade	00 ... 99

Bytes	Description	Area
7 (4 MSB)	■ LSD from ms – LSD: Least Significant Decade	0 ... 9
7 (4 LSB)	weekday	1 ... 7 (1 = Sunday)

Diagnostic buffer

More information about the events in the diagnostics buffer of your CPU may be found in the manual of your CPU or in the manual of you programming software.

14.20 Module diagnostic information - SSL-ID: 00B1h

Description

If you read this partial list, you obtain the first 4 diagnostic bytes of a module with diagnostic capability.

Parameters

SSL_ID	INDEX	Description
00B1h	adr	Shows the first 4 diagnostic bytes of a module. Here the following is to be specified via INDEX: ■ Bit 0 ... 14: Logical base address of the module ■ Bit 15: – 0: Input – 1: Output

LENTHDR	A record set is 2words long (4bytes).
N_DR	Number of record sets

Record set

SSL_ID: 00B1h

Name	Length	Description
byte0	1byte	■ Bit 0: Module fault (group fault ID) ■ Bit 1: Internal fault ■ Bit 2: External fault ■ Bit 3: Channel error exists ■ Bit 4: No external auxiliary voltage ■ Bit 5: No front connector ■ Bit 6: Module not assigned parameters ■ Bit 7: Wrong parameters on module
byte1	1byte	■ Bit 0 ... 3: Module class – 0000: CPU – 0101: Analog modules – 1000: FM – 1100: CP – 1111: Digital modules – 0011: DP Norm slave – 0100: IM ■ Bit 4: Channel information exists ■ Bit 5: User information exists ■ Bit 6: Diagnostic interrupt from substitute ■ Bit 7: Maintenance requirement (PROFINET IO only)
byte2	1byte	■ Bit 0: User module incorrect / does not exist ■ Bit 1: Communication fault ■ Bit 2: Mode 0: RUN, 1: STOP ■ Bit 3: Watchdog responded ■ Bit 4: Internal module power supply failed ■ Bit 5: Battery exhausted ■ Bit 6: Entire buffer failed ■ Bit 7: Maintenance requirement (PROFINET IO only)
byte3	1byte	■ Bit 0: Expansion rack failure (detected by IM) ■ Bit 1: Processor failure ■ Bit 2: EPROM error ■ Bit 3: RAM error ■ Bit 4: ADC/DAC error ■ Bit 5: Fuse blown ■ Bit 6: Hardware error lost ■ Bit 7: reserved (fix 0)

14.21 Module diagnostic information via physical address - SSL-ID: 00B2h

Description

If you read this partial list, you obtain the diagnostic record set 1 of a module in a central rack (not for PROFIBUS DP or submodules). The diagnostic record 1 contains the 4 bytes of diagnostic data that are also in data record 0, plus module-specific diagnostics data that describe the state of a channel or a channel group. The module is to be specified via rack and slot number.

Parameter

SSL_ID	INDEX	Description
00B2h	xxyyh	Shows diagnostic record set 1 of a module. Here the following is to be specified via INDEX: ■ xx: Number of the rack ■ yy: Slot number of the module

LENTHDR	The length of the record set depends on the module.
N_DR	1 (Number of record set)

Record set

Information to length and structure of the diagnostic record set may be found in the corresponding manual of your diagnosable module.

14.22 Module diagnostic information via logical address - SSL-ID: 00B3h

Description

If you read this partial list, you obtain all the diagnostic data of a module. You can also obtain this information for PROFIBUS DP and submodules. The diagnostic record 1 contains the 4 bytes of diagnostic data that are also in data record 0, plus module-specific diagnostics data that describe the state of a channel or a channel group. The module is to be specified via the logical base address.

Parameters

SSL_ID	INDEX	Description
00B3h	adr	Shows all the diagnostic data of a module. Here the following is to be specified via INDEX: ■ Bit 0 ... 14: Logical base address of the module ■ Bit 15: 0: Input, 1: Output

LENTHDR	The length of the record set depends on the module.
N_DR	1 (Number of record set)

Record set

Information to length and structure of the diagnostic data may be found in the corresponding manual of your diagnosable module.

14.23 Diagnostic data of a DP slave - SSL-ID: 00B4h

Description

If you read this partial list, you obtain the diagnostic data of a PROFIBUS DP slave. This diagnostic data is structured in compliance with EN 50 170 Volume 2, PROFIBUS. The module is to be specified via the configured diagnostic address.

Parameters

SSL_ID	INDEX	Description
00B4h	diagadr	Shows all the diagnostic data of a PROFIBUS DP slave. Here the configured diagnostic address of the DP slave is to be specified with INDEX.

LENTHDR	Length of a record set The maximum length is 240bytes. For standard slaves, which have a diagnostic data length of more than 240bytes up to a maximum of 244bytes, the first 240bytes are read and the overflow bit is set in the data.
N_DR	1 (Number of record set)

Record set

***SSL_ID*: 00B4h**

Name	Length	Description
status1	1byte	Station status 1
status2	1byte	Station status 2
status3	1byte	Station status 3
stat_nr	1byte	Number of master station
ken_hi	1byte	Vendor ID (high byte)
ken_lo	1byte	Vendor ID (low byte)
...	...	Further diagnostic data specific to the particular slave

14.24 Information EtherCAT master/slave - SSL-ID: xyE0h

Description

This SSL partial list is a VIPA specific SSL to request EtherCAT states of master/slave via logical and geographical addresses.

Parameters

SSL-ID: xyE0h

SSL_ID	INDEX	Description
x0E0h		State info of a master + all the configured slaves via the ID of the EtherCAT network
	xxxxh	■ Bit 0 ... 10: – not relevant (all devices, max. 512+1)
		■ Bit 11 ... 14: – System ID [1] of the EtherCAT network - 100

SSL_ID	INDEX	Description
		■ Bit 15: – 1: ID bit for EtherCAT (PROFINET "look and feel")
xCE0h		State info of the EtherCAT master/slave via logical base address
	xxxxh	■ Bits 0 ... 14: – logical base address of the EtherCAT device
		■ Bit 15: – 0 = Input – 1 = Output
xDE0h		State info of a EtherCAT master/slave via the geographical address
	xxxxh	■ Bit 0 ... 10: – Master/slave ID
		■ Bit 11 ... 14: – System ID [1] of the EtherCAT network-100
		■ Bit 15: – 1: ID bit for EtherCAT (PROFINET "look and feel")
xFE0h		Only header
	xxxxh	not relevant

1) Refer PROFINET IO system ID, because EtherCAT is configured as PROFINET in the Siemens SIMATIC Manager.

LENTHDR	A record set is 1byte long
N_DR	■ 00E0h: Number of record sets – 512 slaves + 1 master ■ 0CE0h, 0DE0h: Number of record sets

Record set

***SSL_ID*: xyE0h**

Name	Length	Value	Description
ecstate	1 byte	B#16#00	Undefined/Unknown
		B#16#01	Init
		B#16#02	PreOp
		B#16#03	BootStrap
		B#16#04	SafeOp

Name	Length	Value	Description
		B#16#08	Op
		B#16#FF	NotProjected (for not projected EtherCAT periphery)

14.25 EtherCAT bus system - SSL-ID: xyE1h

Description

This SSL partial list is a VIPA specific SSL to request information from the EtherCAT bus system.

Parameters

***SSL-ID*: xyE1h**

SSL_ID	INDEX	Description
0CE1h		State info of the EtherCAT master via logical base address
	xxxxh	■ Bits 0 ... 14: – logical base address of the EtherCAT master (diagnostics address of the interface)
		■ Bit 15: – 0 = Input – 1 = Output
0DE1h		State info of a EtherCAT master via the geographical address
	xxxxh	■ Bits 0 ... 10: – not relevant
		■ Bits 0 ... 14: – System ID [1] of the EtherCAT network - 100
		■ Bit 15: – 1: ID bit for EtherCAT (PROFINET "look and feel")
0FE1h		Only header
	xxxxh	not relevant
1) Refer PROFINET IO system ID, because EtherCAT is configured as PROFINET in the Siemens SIMATIC Manager.		

LENTHDR	A record set is 2words long (4bytes).
N_DR	Number of record sets (1)

Record set

SSL_ID: xyE1h

Name	Length	Value	Description
Bus system	2words	W#32xxxxxxxx	Information via the EtherCAT bus system
			■ Bit 0: – 0: Topology OK – 1: Topology Mismatch
			■ Bit 1: – 0: DC master out of "sync" – 1: DC master in "sync"
			■ Bit 2 ... 31: reserved

14.26 Statistics information to OBs - SSL-ID: xyFAh

This partial list contains statistical information about the OBs (additionally OB 60 and OB 61).

Parameters

SSL-ID: xyFAh

SSL_ID	INDEX	Description
00FAh		All statistical information for OB xx (5 record sets with 24bytes)
01FAh		Response time: time between the request and the start of execution
02FAh		Process image of the inputs (only relevant for OBs are assigned a process image)
03FAh		OB execution time: included alarm interrupts
04FAh		Process image of the outputs (only relevant for OBs are assigned a process image)
05FAh		Processing time: Time for an execution cycle of request until the completion of processing follow up
0FFAh	-	SSL partial list header information
	xx00h	Statistical information for all used OBs (additionally OB 60 and OB 61)
	xx3Ch	Statistical information for OB 60
	xx3Dh	Statistical information for OB 61

LENTHDR	one record set is 12words long (24byte)
N_DR	Number of record sets

- *The times must be specified in µs*
- *During startup, the times are reset to zero - without minimum times.*
- *The minimum times are assigned with the value FFFFh.*

Record set

SSL-ID: 01FAh

The data set includes the response time. This is the time between the request and the start of execution. This time also includes a process input image.

Length	Value	Description
1byte	01h	Number of partial list: *SSL Sub ID*
1byte	xxh	OB Number: statistical information for OB xx (INDEX see above)
1word	xxxxh	reserved
2words	xxxxxxxxh	Minimum execution time: The smallest measured time
2words	xxxxxxxxh	Maximum execution time: Maximum measured time
2words	xxxxxxxxh	Last run time: Last measured time
2words	xxxxxxxxh	Average execution time: The time is determined by the last 1000 recorded times.
2words	xxxxxxxxh	reserved

- *The times must be specified in µs*
- *The measurement of time starts with the first transition from Startup to RUN.*

Record set

SSL-ID: 02FAh

The data set includes the time taken to create the process image of inputs. Only relevant for OBs which a process image is assigned.

Length	Value	Description
1byte	02h	Number of partial list: *SSL Sub ID*
1byte	xxh	OB Number: statistical information for OB xx (INDEX see above)
1word	xxxxh	reserved
2words	xxxxxxxxh	Minimum execution time: The smallest measured time
2words	xxxxxxxxh	Maximum execution time: Maximum measured time

Length	Value	Description
2words	xxxxxxxxh	Last run time: Last measured time
2words	xxxxxxxxh	Average execution time: The time is determined by the last 1000 recorded times.
2words	xxxxxxxxh	reserved

- *The times must be specified in µs*
- *The measurement of time starts with the first transition from Startup to RUN.*

Record set

SSL-ID: 03FAh

The data set contains the execution time of the OBs. This is the time between the start of the OBs until leaving the OB including all alarm interrupts and SFC operations. The time from a higher priority OB is executed by a synchronous or asynchronous error is counted with.

Length	Value	Description
1byte	03h	Number of partial list: *SSL Sub ID*
1byte	xxh	OB Number: statistical information for OB xx (INDEX see above)
1word	xxxxh	reserved
2words	xxxxxxxxh	Minimum execution time: The smallest measured time
2words	xxxxxxxxh	Maximum execution time: Maximum measured time
2words	xxxxxxxxh	Last run time: Last measured time
2words	xxxxxxxxh	Average execution time: The time is determined by the last 1000 recorded times.
2words	xxxxxxxxh	reserved

- *The times must be specified in µs*
- *The measurement of time starts with the first transition from Startup to RUN.*

Record set

SSL-ID: 04FAh

The data set includes the time for creating the process image of outputs. Only relevant for OBs which a process image is assigned.

Length	Value	Description
1byte	04h	Number of partial list: *SSL Sub ID*
1byte	xxh	OB Number: statistical information for OB xx (INDEX see above)
1word	xxxxh	reserved
2words	xxxxxxxxh	Minimum execution time: The smallest measured time
2words	xxxxxxxxh	Maximum execution time: Maximum measured time
2words	xxxxxxxxh	Last run time: Last measured time
2words	xxxxxxxxh	Average execution time: The time is determined by the last 1000 recorded times.
2words	xxxxxxxxh	reserved

- *The times must be specified in µs*
- *The measurement of time starts with the first transition from Startup to RUN.*

Record set

SSL-ID: 05FAh

The data set contains the determined times for one execution cycle. This is the time between the request and the full completion of the processing.

Length	Value	Description
1byte	05h	Number of partial list: *SSL Sub ID*
1byte	xxh	OB Number: statistical information for OB xx (INDEX see above)
1word	xxxxh	reserved
2words	xxxxxxxxh	Minimum execution time: The smallest measured time
2words	xxxxxxxxh	Maximum execution time: Maximum measured time
2words	xxxxxxxxh	Last run time: Last measured time
2words	xxxxxxxxh	Average execution time: The time is determined by the last 1000 recorded times.
2words	xxxxxxxxh	Error counter: This counter is increased at the time, when the execution cycle is longer than 60% of the projected Sync clock.

- *The times must be specified in µs*
- *The measurement of time starts with the first transition from Startup to RUN.*
- *The cycle time of the Sync signal is set (HW configuration) via the CPU properties.*

14.27 VSC features - SSL-ID: xyFCh

General

Via this partial list you get the current status of the VSC features of the System SLIO CPU. There are features at the VIPA memory card to unlock e.g. additional memory or PROFIBUS functionality.

Parameters

SSL-ID: xyFCh

SSL_ID	INDEX	Description
00FCh	-	Status of all the VSC features
01FCh		Specifies the VSC feature, whose state is requested
	0001h	VSC feature PROFIBUS
	0002h	VSC feature memory extension
	0003h	VSC feature Timeout
	0004h	VSC feature CP fieldbus
	0005h	VSC feature motion

LENTHDR	Length of the following record set in byte
N_DR	Number of record sets

Record set

***SSL_ID*: 0xFCh**

Name	Length	Value	Description
VSC_Feature PROFIBUS-DP	2words	000xh	■ 0 = PROFIBUS_NO ■ 1 = PROFIBUS_MASTER ■ 2 = PROFIBUS_SLAVE
VSC_Feature MemKeySize	2words	xxxxh	Size of the memory extension via VSC card in byte
VSC TimeOut	2words	xxxxh	Remaining time of the CPU with removed VSC card in ms (for S7 data type Time)
VSC_Feature CpFieldbus	2words	xxxxh	■ 0 = FEATURE_SET_CP_FIELDBUS_NO ■ 1 = FEATURE_SET_CP_FIELDBUS_ETHERCAT
VSC_Feature Motion	2words	xxxxh	■ 0 = FEATURE_SET_MOTION_NO ■ 1 = FEATURE_SET_MOTION_8AXIS ■ 2 = FEATURE_SET_MOTION_20AXIS

15 Index

H

I