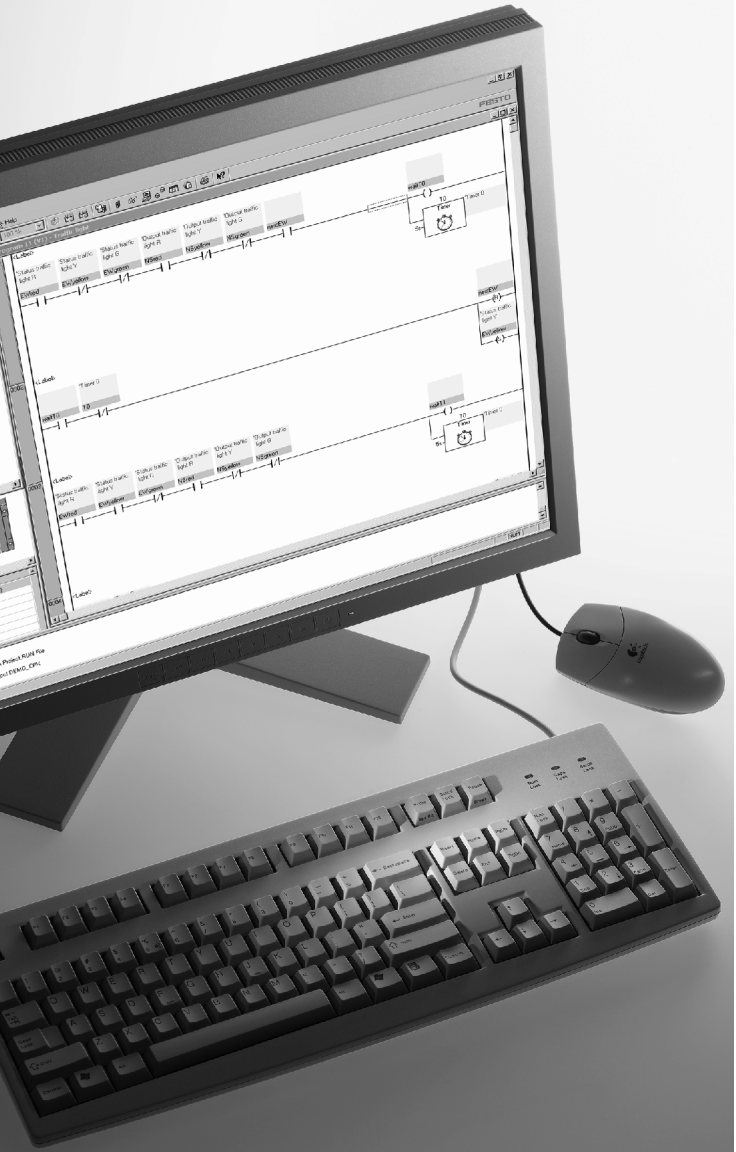
Festo Software Tools



FESTO

Software package FST

Version 4

Volume 2

Drivers and modules

Manual
en 0403NH
[682 298]

Authors ..... S. Breuer, Dr. F. Haase, Z. Kirch, J. Römer I. Walter, O. Westrik

Editor ..... M. Holder

Original ..... de

Edition ..... en 0403NH

Designation P.BE-FST4-B2-EN

Order no. of the software package 682 298

Internet: http://www.festo.com
E-Mail: service_international@festo.com

Microsoft® Windows®	is a registered trade mark of the Microsoft Corporation
Microsoft Internet Explorer®	is a registered trade mark of the Microsoft Corporation
AS-Interface®	is a registered trade mark of the AS-Interface Association
MODBUS®	is a registered trademark of the Schneider Automation

Contents

Designated use

With this software package, the user who is familiar with the relevant PLC/IPC, can undertake the configuration, programming and commissioning of the PLC/IPC supported by the software package.

Observe also the standards specified in the relevant chapters, as well as national and local laws and technical regulations.

Target group

This manual is intended exclusively for technicians trained in control and automation technology who have experience in installing, commissioning, programming and diagnosing PLC/IPCs.

Service

Please consult your local Festo repair service if you have any technical problems.

Notes on the use of this manual

This manual contains information on the drivers and modules available in the FST, as well on the hardware configuration and addressing with FEC Standard, FEC Compact and PS1 (see chapter 6).

Special information on the supported PLC/IPCs can be found in the hardware documentation for the relevant product.

Drivers and modules for the FST PLC operating system

The functionality of the FST PLC operating system can be extended by means of pre-produced drivers and modules. Modules can be imported into the project and then loaded into the controller.

Please note
This manual refers to the driver versions listed in Appendix A.2.

Modules can be created in all available programming languages (STL, LDR, C). Information on creating modules in C can be found on the data storage medium.
Detailed information on this can be found in the file README.TXT in the installation directory of the FST software.

Numerous C modules are available for special tasks. These can be accessed either as function modules or as program modules.

Most drivers support the use of special modules. Before these modules can be used, the relevant driver must be entered and parametrized in the driver configurator. Details on this can be found in volume 1.

Standard drivers and standard modules

Chapter 1

Contents

Contents of this chapter

This chapter provides an overview of the standard drivers and standard modules for the FST PLC operating system. The drivers and modules described in this chapter can be used with the following PLC/IPCs from Festo:

- CPX terminal with CPX-FEC
- FEC Compact
- FEC Standard
- PS1

Further information

Information on further drivers and modules, which can be used only with some of the PLC/IPCs named, can be found in the following chapters:

- Drivers and modules for CPX-FEC; chapter 2
- Drivers and modules for FEC Compact, FEC Standard and PS1; chapter 3
- Further drivers and modules for FEC Compact; chapter 4
- Further drivers and modules for PS1; chapter 5

1.1 General modules

Overview of modules

Modules	Description
BLINK	General flashing bits
FIFO	“First-in-first-out” memory
INRANGE	Checks whether the value lies within a certain range
MINMAX	Ring buffer with minimum, maximum and medium value
SCALE	Scales a 16-bit value

BLINK General flashing bits

Input parameter
None

Return parameter			
FU32	4 flashing bits		
	Bit	Signal duration [s]	Frequency [Hz]
	0	0.25	2
	1	0.5	1
	2	1	0.5
	3	2	0.25

This module does not use a timer. The module must be accessed cyclically for flashing bits to be set.

FIFO

"First-in-first-out" memory (FIFO memory)

Input parameter	
FU32	Function 0 = Reset FIFO 1 = Transfer value from FU33 to the FIFO 2 = Read next value from the FIFO 3 = Register the number of values saved in the FIFO 4 = Read any element in the FIFO memory 5 = Write any element in the FIFO memory
FU33	IF function (input parameter FU32) = 1: New value to be transferred to the FIFO memory = 4: Position in the FIFO memory = 5: Position in the FIFO memory
FU34	IF function (input parameter FU32) = 5: Value to be transferred to the FIFO memory

Return parameter	
FU32	IF function (input parameter FU32) = 1: 0: Operation carried out successfully 1: No memory space for extending the FIFO memory 2: FIFO memory is full = 2: Next value from the FIFO or 0 if FIFO is empty = 3: Number of values saved in the FIFO = 4: Value from the FIFO or 0 if FIFO is empty = 5: Supplies 0 if successful

Maximum 10000 values can be saved in the FIFO memory.

INRANGE

Checks whether the value lies within a certain range.

Input parameter	
FU32	Value
FU33	Rated value
FU34	+/- deviation from rated value

Return parameter	
FU32	-1: Value is smaller than rated value - deviation 0: Value is greater than rated value - deviation and smaller than rated value + deviation 1: Value is greater than rated value + deviation
FU33	Difference between actual value and rated value (absolute)
FU34	Like FU32, but with values with a sign
FU35	Like FU33, but with values with a sign

With a comparison of values without sign (0 ... 65535), FU32 and FU33 must be evaluated. With a comparison of numbers with a sign (-32768 to 32767), FU34 and FU35 must be evaluated.

Please note
The calculated difference is always positive.

MINMAX

Ring buffer with the possibility of interrogating minimum, maximum and medium values

Input parameter	
FU32	Mode 0 = Initialize or reset buffer 1 = Add value 2 = Calculate minimum, maximum and medium values 3 = Like 2, but with numbers with a sign 4 = Copy buffer into flag word range
FU33	Depending on mode: Mode 0: Buffer size (1 ... 1024) Mode 1: New value Mode 4: Start flag word

Return parameter	
FU32	Fault code (0 = no fault)
FU33	Number of values in the buffer
FU34	Minimum value (modes 2 and 3)
FU35	Maximum value (modes 2 and 3)
FU36	Medium value (modes 2 and 3)

The range for numbers with signs is from -32768 to 32767, otherwise from 0 to 65535.

Fault codes	Description
-1	Not initialized
1	Mode not supported (invalid value in FU32)
2	Buffer size too large (>1024)

SCALE

Scale 16-bit values

Scales a 16-bit value in accordance with this formula:

$$y = x \cdot \left(\frac{a}{b}\right) + c$$

Input parameter	
FU32	x input value
FU33	a range output value
FU34	b range input value
FU35	c offset

Return parameter	
FU32	y output value

This function supports you in converting values into another range. For example the return value of an analogue card (usually 0 ... 4095) in millivolts.

You convert 0 ... 4095 in 0 ... 10000 mV:

```
CFM0                  " Scale: X * (A/B) + C
 WITH   IW10          " X
 WITH   V10000        " A
 WITH   V4095         " B
 WITH   V0            " C
```

You convert -2048 ... 2047 in -10000 ... +10000 mV:

```
CFM 0                 " Scale: X * (A/B) + C
 WITH   IW10          " X
 WITH   V20000        " A
 WITH   V4095         " B
 WITH   V0            " C
```

You convert 0 ... 4095 in -10000 ... +10000 mV:

```
CFM 0                    " Scale: X * (A/B) + C
 WITH   IW10             " X
 WITH   V20000           " A
 WITH   V4095            " B
 WITH   V-10000          " C
```

You convert 0 ... 4095 in 0 ... 20 mA:

```
CFM 0                    " Scale: X * (A/B) + C
 WITH   IW10             " X
 WITH   V20              " A
 WITH   V4095            " B
 WITH   V0               " C
```

You convert 4 ... 4095 in 4 ... 20 mA:

```
CFM 0                    " Scale: X * (A/B) + C
 WITH   IW10             " X
 WITH   V16              " A
 WITH   V4095            " B
 WITH   V4               " C
```

1.2 Modules for controlling program processing

Overview of modules

Modules	Description
F4	Cyclic starting of a program
F8	Stopping all cyclic programs
F23	Interrogates whether a program is ready for processing
F26	Controls the program, the number of which is saved in a variable

F4 Cyclic starting of a program

Input parameter	
FU32	Program number, 0 to 63
FU33	Time in milliseconds, 14 to 65535 or 0 in order to deactivate

Return parameter
None

Resolution is:

– in the case of FEC Compact and HC0X with operating system version ‹ S2.00: 13.74 ms

– in the case of FEC Standard, CPX-FEC, FEC Compact and HC0X with operating system version ≥ S2.00: 5 ms.

All time specifications in FU33 are rounded to a multiple of the resolution named.

Function module F4 can also be used for stopping cyclically started programs. Call the module with the value 0 in FU33 (see also module F8).

A cyclic program must reset itself with RESET Pn (stop), in order that it can be set again when the set time has expired.

F8 Stopping all cyclic programs

This function module concludes cyclic processing of the programs. All programs started cyclically with F4 will be stopped. In order to remove an individual program from cyclic processing, you must use F4.

Input parameter
None

Return parameter
None

F23 Interrogates whether a program is ready for processing

Input parameter	
FU32	Program number

Return parameter	
FU32	0 = Program does not exist -1 = Program can be started

F26

Controls the program the number of which is saved in a variable

Input parameter	
FU32	Program number
FU33	0 = Start program 1 = Stop program 2 = Continue interrupted program 3 = Interrupt program

Return parameter
None

1.3 Modules for error treatment

Overview of modules

Modules	Description
F21	Interrogate or set the number of the error program
F22	Set error treatment
F25	Set error treatment for I/O errors

F21 Interrogate or set the number of the error program

Input parameter		
FU32	0: 1...63: ≥ 64:	Reset function (no error program) Number of the error program Interrogate current status

Return parameter	
FU32	Number of the current error program

The error program can also be defined in the PLC settings for the project.

F22 Set error treatment

Input parameter	
FU32	0 = Interrogate error data 1 = Interrogate error data and delete error 2 = Process programs further, interrogate error data and delete error 3 = Error cannot be eliminated, stop programs

Return parameter	
FU32	Error number
FU33	Program number
FU34	Step number
FU35	Always 0 (error address)

F25 Set error treatment for I/O errors

This module has only one meaning for the I/O error recognition in respect of errors 11 and 12 (only relevant with FEC Compact, FEC Standard and PS1).

With CPX-FEC the I/O error recognition (monitoring) can be parametrized (see CPX system manual).

Input parameter	
FU32	0 = Switch off error recognition 1 = Switch on error recognition, presetting

Return parameter
None

1.4 Modules for modifying operands

Overview of modules

Modules	Description
CHECKSUM	Checksum of part of the range of flag words
COPY	Copy part of the range of flag words
DINDEXMW	Flag word indexed decrementing
F9	Clear operands
FLAGBIT	Flag bit indexed setting or resetting
IINDEXMW	Flag word indexed incrementing
NINDEXMW	Delete flag word range
RINDEXMW	Flag word indexed reading
WINDEXMW	Flag word indexed writing

CHECKSUM

Checksum of part of the range of flag words

Input parameter	
FU32	Number of the first flag word
FU33	Number of the last flag word

Return parameter	
FU32	Checksum

You can form the checksum by adding together the flag words.

COPY

Copy part of the range of flag words

Input parameter	
FU32	Number of the first source flag word
FU33	Number of the first target flag word
FU34	Number of flag words

Return parameter
None

The flag word ranges may overlap.

DINDEXMW

Flag word indexed decrementing

Input parameter	
FU32	Number of the flag word

Return parameter	
FU32	New value

The specified flag word is decremented by 1, if it is not yet zero.

F9 Clear operands

Input parameter	
FU32	0 = Clear all registers, counters, timers, flags 1 = Clear all registers 2 = Clear all flags 3 = Clear all timers 4 = Clear all counters

Return parameter
None

FLAGBIT Flag bit indexed setting or resetting

Input parameter	
FU32	Flag word number
FU33	Bit number
FU34	0 = Reset bit 1 = Set bit 2 = Return bit value in FU33 3 = Switch bit

Return parameter	
FU32	0 = Module processed successfully 1 = Non-permitted flag word number 2 = Non-permitted bit number 3 = Invalid function
FU33	Bit value 0 = Bit not set 1 = Bit set

IINDEXMW

Flag word indexed incrementing

Input parameter	
FU32	Number of the flag word

Return parameter	
FU32	New value

The specified flag word is incremented by 1, if the maximum value 65535 ($FFFF) is not yet reached.

NINDEXMW

Delete flag word range

Input parameter	
FU32	Number of the first flag word
FU33	Number of flag words to be deleted

Return parameter
None

RINDEXMW

Flag word indexed reading

Input parameter	
FU32	Number of the flag word

Return parameter	
FU32	Read value

WINDEXMW Flag word indexed writing

Input parameter	
FU32	Number of the flag word
FU33	New value

Return parameter
None

1.5 Setting the real time clock or the system clock

Real time clock

Real time clocks are clock modules which still function when the controller is switched off. They are supplied with power separately, e. g. by means of a battery. They can therefore always provide the current time and the current date. Only the CPUs HC0X, HC1X and HC2X provide real time clocks.

System clock

The system clock is part of the operating system of the controller and is therefore always available. It works only when the controller is switched on. When the system clock is switched on, it always starts with the same time. (usually 1980-01-01 00:00:00). If a real time clock is also available, the system clock will be synchronized with the real time clock when the system is started (after booting or switching on). During running time, the system clock will be updated by the system timer.

Setting the clocks with the FST

The FST offers modules F10 to F13 for reading and setting the system clock or the real time clock (see also section 1.5.1). If a real time clock is not available, the system clock will be influenced directly. If a real time clock is available, it will be influenced directly and the system clock will be synchronized with the real time clock each time the above-named modules are accessed.

In order to set the clock with the FST, you must load a project into the controller with the modules F10 to F13. The modules can be processed with the CI commands “RF” or “RB.”

Example: If module F10 is imported e.g. as CFM 10, the following CI command will set the clock to 08:30:

```
›RF10,8,30,0,0
›
```

Real time clocks of CPUs HC0X, HC1X and HC2X

Target system	Description
HC0X	A real time clock is available in the CPU. It is supplied with power by a battery for approx. 10 years.
HC1X	The system can contain 2 different real time clocks: 1. the I2C-bus real time clock 2. the SRAM real time clock The I2C-bus real time clock is not located in the CPU, but either on the bus board or in the DC/DC voltage converter module, depending on the bus board. Systems on a three-bus board do not have an I2C-bus real time clock. A Goldcap capacitor supplies the real time clock with power for 3 to 5 days when the voltage has been switched off. The SRAM real time clock is located in the CPU module and is battery-backed; the battery lasts for approx. 10 years. The CPU module possesses a SRAM real time clock if it is equipped with a ZL16 or ZL17 SRAM. If a SRAM real time clock is found in the system, this will be preferred rather than the I2C-bus real time clock.
HC2X	This CPU is equipped with a SRAM real time clock. The SRAM real time clock is located in the CPU module and is battery-backed; the battery lasts for approx. 10 years.

Setting the clock manually in the HC1X

The I2C-bus real time clock in the HC1X can be set in the BIOS setup. How to carry out the setting is described in the hardware documentation. A screen and a keyboard must always be connected for this purpose.

1.5.1 Modules for setting the real time clock or the system clock

Overview of modules

Modules	Description
F10	Set the time
F11	Set the date
F12	Interrogate the time
F13	Interrogate the date

F10 Setting the time

Input parameter	
FU32	Hour (0 to 23)
FU33	Minute (0 to 59)
FU34	Second (0 to 59)
FU35	Hundredth of a second (0 to 99)

Return parameter
None

F11 Setting the date

Input parameter	
FU32	Year (1980 to 2099)
FU33	Month (1 to 12)
FU34	Day (1 to 31)

Return parameter
None

F12 Interrogate time

Input parameter
None

Return parameter	
FU32	Hour (0 to 23)
FU33	Minute (0 to 59)
FU34	Second (0 to 59)
FU35	Hundredth of a second (0 to 99)

F13 Interrogate date

Input parameter
None

Return parameter	
FU32	Year (1980 to 2099)
FU33	Month (1 to 12)
FU34	Day (1 to 31)
FU35	Day of the week (0 = Sunday, 6 = Saturday)

1.6 Modules for 32-bit arithmetic

With the following modules it is possible to carry out calculation and comparison operations on 32-bit values. For this purpose the numbers are distributed on two consecutive 16-bit operands.

Modules	Description
LADD	Addition of 32-bit values
LCMP	Comparison of 32-bit values
LDIV	Division of 32-bit values
LMUL	Multiplication of 32-bit values
LNEG	Sign change with a 32-bit value
LSUB	Subtraction of 32-bit values

LADD Addition of 32-bit values

Input parameter	
FU32	Lower-value word of the 1st. operand
FU33	Higher-value word of the 1st. operand
FU34	Lower-value word of the 2nd. operand
FU35	Higher-value word of the 2nd. operand

Return parameter	
FU32	Lower-value word of the result
FU33	Higher-value word of the result

LCMP Comparison of 32-bit values

Input parameter	
FU32	Lower-value word of the 1st. operand
FU33	Higher-value word of the 1st. operand
FU34	Lower-value word of the 2nd. operand
FU35	Higher-value word of the 2nd. operand

Return parameter	
FU32	Result in bit form $xx01: 1st. operand ‹ 2nd. operand, with sign (signed) $xx02: 1st. operand == 2nd. operand, with sign (signed) $xx04: 1st. operand › 2nd. operand, with sign (signed) $01xx: 1st. operand ‹ 2nd. operand, without sign (unsigned) $02xx: 1st. operand == 2nd. operand, without sign (unsigned) $04xx: 1st. operand › 2nd. operand, without sign (unsigned)

LDIV Division of 32-bit values

Input parameter	
FU32	Lower-value word of the 1st. operand
FU33	Higher-value word of the 1st. operand
FU34	Lower-value word of the 2nd. operand
FU35	Higher-value word of the 2nd. operand

Return parameter	
FU32	Lower-value word of the result
FU33	Higher-value word of the result

LMUL Multiplication of 32-bit values

Input parameter	
FU32	Lower-value word of the 1st. operand
FU33	Higher-value word of the 1st. operand
FU34	Lower-value word of the 2nd. operand
FU35	Higher-value word of the 2nd. operand

Return parameter	
FU32	Lower-value word of the result
FU33	Higher-value word of the result

LNEG

Sign change with a 32-bit value

Input parameter	
FU32	Lower-value word of the operand
FU33	Higher-value word of the operand

Return parameter	
FU32	Lower-value word of the result
FU33	Higher-value word of the result

LSUB

Subtraction of 32-bit values

(Result = 1st. operand – 2nd. operand)

Input parameter	
FU32	Lower-value word of the 1st. operand
FU33	Higher-value word of the 1st. operand
FU34	Lower-value word of the 2nd. operand
FU35	Higher-value word of the 2nd. operand

Return parameter	
FU32	Lower-value word of the result
FU33	Higher-value word of the result

1.7 Floating point operations (driver FPMATHDR)

The driver FPMATHDR supports the following functions:

- Basic floating point operations (multiplication, division, addition, subtraction)
- Conversion between:
  - floating point numbers and 16-bit integers
  - floating point numbers and 32-bit integers.

In order to carry out floating point operations in your FST project, add this driver (FPMATHDR) to the driver list of the project. The driver does not require any parameters.

Mathematical operations/conversions of the FST program are carried out with the aid of modules (see section 1.7.3). Using modules without the driver will cause fault 1, which will be returned by the module.

The driver supports operations in the format single float (also supported as type float of standard C compilers, Microsoft C/C++, Borland C/C++ etc.).

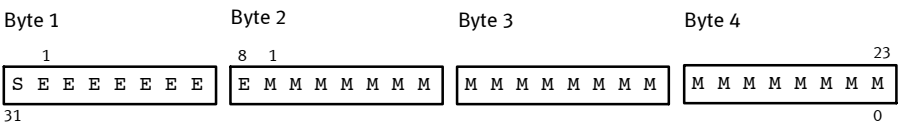


Fig. 1/1: Floating point format (single float)

E = exponent; M = mantissa; S = sign bit: 0 = +, 1 = –

The mantissa has a length of 24 positions – F1 to F23 plus a hidden position (F0). The hidden position (hidden bit) is ascertained internally as follows:

– F0 = **0**, with exponent -127

– F0 = **1**, with exponent -126 ... +127

– F0 is not relevant with exponent +128

Bit sample in the exponent		**Exponent**	**Hidden bit (F0)**	**Represented floating point number**
00000000	0	-127	0	± (**0.** F1 F2 ... F23) * 2^{-127}
00000001	1	-126	1	± (**1.** F1 F2 ... F23) * 2^{-126}
	...			
01111111	127	0	1	± (**1.** F1 F2 ... F23) * 2^{0}
10000001	128	1	1	± (**1.** F1 F2 ... F23) * 2^{1}
	...			
11111110	254	127	1	± (**1.** F1 F2 ... F23) * 2^{127}
11111111	255	128	–	∞ if F1=...=F23=0, otherwise NaN (**n**ot **a n**umber)

Calculating the value

$$\text{Value} = 0. * 2^{x} + M1 * 2^{x-1} + M2 * 2^{x-2} + M3 * 2^{x-3} + \ldots + M23 * 2^{x-23}$$

$$\text{Value} = 1. * 2^{x} + M1 * 2^{x-1} + M2 * 2^{x-2} + M3 * 2^{x-3} + \ldots + M23 * 2^{x-23}$$

x = exponent, M = mantissa position

Example

Floating point number 550.0 is saved as follows:

0x44098000
(0x44 is byte 1; 0x09 is byte 2; 0x80 is byte 3; ...).

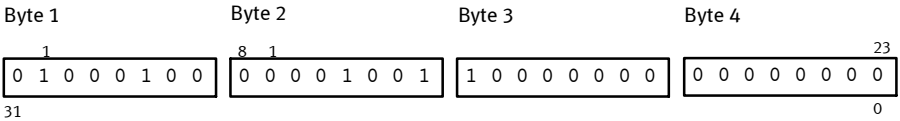


Fig. 1/2: Floating point format (single float)

Sign bit: 0 (+)

Exponent: 136 - 127 = 9

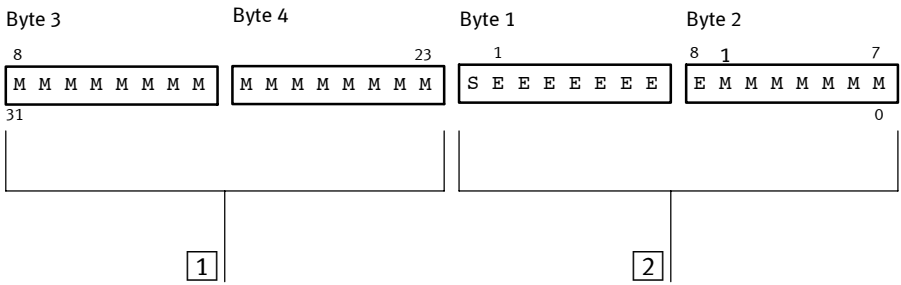
F0 = 1

Result: $1 * 2^9 + 1 * 2^5 + 1 * 2^2 + 1 * 2^1 = 550$

A note on the float format of the MODBUS

Please note
Note that the MODBUS protocol uses the same format for representing floating point numbers, but with a different byte sequence.

Float format of the MODBUS



1 MODBUS address X

2 MODBUS address X + 1

Fig. 1/3: Float format of the MODBUS

S = sign bit, E = exponent; M = mantissa

Example

With MODBUS floating point number 550.0 is saved as follows:

0x80004409 (byte 3 is saved first, ...).

1.7.1 Configuration of the driver and assignment of parameters

If you wish to use the FPMATH modules in an FST IPC project, you must enter the FPMATH driver in the driver configuration and assign the necessary parameters.

Target disc drive

Specify the drive on which the FPMATH driver FPMATHDR.EXE can be found or onto which it must be loaded.

1.7.2 Additional CI commands

This driver supplements the CI with the following commands:

CI command	Description
!46	Shows driver information and version number. This information is also shown if an unknown command is entered (e.g. !46abcdef).

1.7.3 Modules

Overview of modules

Module	Description
FPA2F	Conversion of string to float
FPF2A	Conversion of float to string
FPF2I	Conversion of float to 16-bit integer
FPF2L	Conversion of float to 32-bit integer
FPI2F	Conversion of 16-bit integer to float
FPL2F	Conversion of 32-bit integer to float
FPBINOP	Basic floating point binary operations
FPABCD	Carries out (A*B)/(C*D)
FPM1	Carries out ((A-B)*C – (D-E)*(F-G))/(100-H)
FPROM1	Evaluating customer-specific expressions
FPSQRT	Calculating square roots
FPGONIO	Trigonometrical functions

FPA2F Conversion of string to float

Input parameter	
FU32	String number

Return parameter	
FU32	Status 0 = OK 1 = fault in accessing FpMathDr-driver 2 = number outside range 3 = mathematical fault during conversion
FU33	Converted number (16-bit integer with sign)
FU34	Value range of converted number

FPF2A Conversion of float to string

Input parameter	
FU32	Byte_3, Byte_4 of the float number
FU33	Byte_1, Byte_2 of the float number
FU34	Number of post-decimal positions
FU35	String number

Return parameter	
FU32	Status 0 = OK 1 = fault in accessing FpMathDr-driver 2 = number outside range 3 = mathematical fault during conversion

FPF2I — Conversion of float to 16-bit integer

Use in FST Insert function module FPF2I in the project, declare as type F (function module) and assign a module number of your choice. This function module converts the standard floating point number to the corresponding 16-bit integer with sign.

Input parameter	
FU32	Byte_3, Byte_4 of the float number (contents of MODBUS address X of the float number saving)
FU33	Byte_1, Byte_2 of the float number (contents of MODBUS address X+1 of the float number saving)
FU34	Range specification: 0 = no range 1 = automatic range with max. accuracy 2 = automatic range with scientific format 3 = use certain value range (FU35)
FU35	Certain value range Valid only for FU34 = 3

Return parameter	
FU32	Status 0 = OK 1 = fault in accessing FpMathDr-driver 2 = number outside range 3 = mathematical fault during conversion
FU33	Converted number (16-bit integer with sign)
FU34	Value range of converted number

Description of the value range
(input parameter FU35, output parameter FU34)

The value range is the exponent of the number on the basis of 10. The number NUM specified and value range RANGE_VALUE, give the following result:

Result = NUM * 10 ^ RANGE_VALUE

The value range is an integer with sign.

Examples

– IntValue = 4567, RangeValue = 3:
  Result = $4567 * 10^3 = 4567000$

– IntValue = 4567, RangeValue = 1:
  Result = $4567 * 10^1 = 45670$

– IntValue = 4567, RangeValue = -5:
  Result = $4567 * 10^{-5} = 0.04567$

Description of the range specification
(input parameter FU34)

FU34	Description
0	There is no range allocation for the conversion, e.g. value range = 0 at output
1	Automatic range with max. accuracy The number is converted so that it fits preferably in the range (-32000, +32000), with the maximum number of figures. The value range will be calculated and returned at output in FU34.
2	Automatic range with scientific format Similar as above, but the number is converted so that the value range corresponds to a multiple of 3 (value range in [...,-6,-3,0,+3,+6,...]).
3	Value range in FU35 at input correct Output value range in FU34 will be the same. The number will be converted in accordance with the specified value range.

FPF2L

Conversion of float to 32-bit integer of type long

Use in FST Insert function module FPF2L in the project, declare as type F (function module) and assign with a module number of your choice. This function module converts the floating point number to the corresponding 32-bit integer with sign.

Input parameter	
FU32	Byte_3, Byte_4 of the float number (contents of MODBUS address X of the float number saving)
FU33	Byte_1, Byte_2 of the float number (contents of MODBUS address X+1 of the float number saving)
FU34	Range specification: 0 = no range 1 = automatic range with max. accuracy 2 = automatic range with scientific format 3 = use certain value range (FU35)
FU35	Certain value range Valid only for FU34 = 3

Return parameter	
FU32	Status 0 = OK 1 = fault in accessing FpMathDr-driver 2 = number outside range 3 = mathematical fault during conversion
FU33	Low word (= lower-value word) of the converted 32-bit integer of type long
FU34	High word (= higher-value word) of the converted 32-bit integer of type long
FU35	Value range of converted number

Description of the value range
(input parameter FU35, output parameter FU35)

Please note the description of function module FPF2I.

Description of the range specification
(input parameter FU34)

Please note the description of function module FPF2I.

FPI2F Conversion of 16-bit integer to float

Use in FST Insert function module FP12F in the project, declare as type F (function module) and assign with a module number of your choice. This function module converts 16-bit integers with sign into the corresponding floating point numbers.

Conversion carried out:
Result (FU33, 34) = float (FU32 * 10 ^ FU33)

Input parameter	
FU32	16-bit integer with sign
FU33	Certain value range

Return parameter	
FU32	Status 0 = OK 1 = fault in accessing FpMathDr-driver 2 = number outside range 3 = mathematical fault during conversion
FU33	Byte_3, Byte_4 of the float number (contents of MODBUS address X of float number saving)
FU34	Byte_1, Byte_2 of the float number (contents of MODBUS address X+1 of float number saving)

FPL2F Conversion of 32-bit integer of type long to float

Use in FST Insert function module FPL2F in the project, declare as type F (function module) and assign with a module number of your choice. This function module converts 32-bit integers of type long with sign into the corresponding floating point numbers.

Conversion carried out:
Result (FU33, 34) =
float((long)(FU32 + 65536 * FU33) * 10 ^ FU34)

Input parameter	
FU32	Low word (= lower-value word) of the converted 32-bit integer of type long with sign
FU33	High word (= higher-value word) of the 32-bit integer of type long with sign
FU34	Certain value range

Return parameter	
FU32	Status 0 = OK 1 = fault in accessing FpMathDr-driver 2 = number outside range 3 = mathematical fault during conversion
FU33	Byte_3, Byte_4 of the float number (contents of MODBUS address X of float number saving)
FU34	Byte_1, Byte_2 of the float number (contents of MODBUS address X+1 of float number saving)

FPBINOP — Basic floating point binary operations

Use in FST Insert function module FPBINOP in the project, declare as type F (function module) and assign with a module number of your choice. This function carries out basic binary operations with floating point numbers. The result is also output as a floating point number.

Input parameter	
FU32	Operation code (BIN_OP): 0 = multiplication 1 = division 2 = addition 3 = subtraction 4 = power 5 = natural logarithm (requires only FU33 and FU34)
FU33	Byte_3, Byte_4 of float number 1
FU34	Byte_1, Byte_2 of float number 1
FU35	Byte_3, Byte_4 of float number 2
FU36	Byte_1, Byte_2 of float number 2

Return parameter	
FU32	Status 0 = OK 1 = fault in accessing FpMathDr-driver 2 = number outside range 3 = mathematical fault during conversion 4 = incorrect operation code
FU33	Operation code (the same as for FU32 when entering)
FU34	Byte_3, Byte_4 of the float result
FU35	Byte_1, Byte_2 of the float result

FPABCD

Carries out (A * B) / (C * D)

Use in FST Insert function module FPABCD in the project, declare as type F (function module) and assign with a module number of your choice. This function module carries out the following operation:

Result = (A * B) / (C * D)

Input numbers (A, B, C and D) are 16-bit integers with signs. The result is output as a floating point number.

Input parameter	
FU32	A (16-bit integer with sign)
FU33	B (16-bit integer with sign)
FU34	C (16-bit integer with sign)
FU35	D (16-bit integer with sign)

Return parameter	
FU32	Status 0 = OK 1 = fault in accessing FpMathDr-driver 2 = number outside range 3 = mathematical fault during conversion
FU33	Byte_3, Byte_4 of the float result
FU34	Byte_1, Byte_2 of the float result

FPM1

Carries out ((A-B)*C - (D-E)*(F-G)) / (100 - H)

Use in FST Insert function module FPM1 in the project, declare as type F (function module) and assign with a module number of your choice. This function module carries out the following operation:

Result = (A-B)*C – (D-E)*(F-G))/(100-H)

Input numbers (A, B, C, D, E, F, G, H) are all 16-bit integers with signs. The result is also output as an integer with sign.

Input parameter	
FU32	Starting flag word from the range of 8 consecutive flag words which contain the input parameters A to H.
FU33	Range specification: 0 = no range 1 = automatic range with max. accuracy 2 = automatic range with scientific format 3 = use certain value range (FU34)
FU34	Certain value range Valid only for FU33 = 3

Return parameter	
FU32	Status 0 = OK 1 = fault in accessing FpMathDr-driver 2 = number outside range 3 = mathematical fault during conversion
FU33	The result (16-bit integer with sign)
FU34	Value range of result

Description of the value range
(input parameter FU34, output parameter FU34)

The value range is the number exponent on the basis of 10. The end result after this function module has been carried out should be interpreted as follows:

Result = FU33 * 10 ^ FU34.

Both the result (FU33) and the value range are integers with signs.

Examples

– FU33 = 4567, FU34 = 3:
  Result = $4567 * 10^3 = 4567000$

– FU33 = 4567, FU34 = 1:
  Result = $4567 * 10^1 = 45670$

– FU33 = 4567, FU34 = -5:
  Result = $4567 * 10^{-5} = 0.04567$

Description of the range specification
(input parameter FU34)

FU34	Description
0	There is no range allocation for the conversion, e.g. value range = 0 at output
1	Automatic range with max. accuracy The number is converted so that it fits preferably in the range (-32000, +32000), with the maximum number of figures. The value range will be calculated and returned at output in FU34.
2	Automatic range with scientific format similar as above, but the result (FU33 with return) is calculated so that the value range (FU34 with return) corresponds to a multiple of 3 (value range in [...,-6,-3,0,+3,+6,...]).
3	Value range is defined in FU34 at input. Output value range in FU34 will be the same. Result (FU33 at return) is calculated according to the defined value range.

FPROM1

Carries out Param1 * 0.001 * 0.5719 * (Param2 + 1) ^ 0.9545

Use in FST Insert function module FPROM1 in the project, declare as type F (function module) and assign with a module number of your choice. The function module requires FpMathDr.exe version 1.40 or later.

This function module carries out the following operation:

Result = Param1 * 0.001 * 0.5719 * (Param2 + 1) ^ 0.9545

– The input parameters (Param1, Param2) are both floating point numbers. The result is also output as a floating point number.

Input parameter	
FU32	Byte_3, Byte_4 of float parameter 1
FU33	Byte_1, Byte_2 of float parameter 1
FU34	Byte_3, Byte_4 of float parameter 2
FU35	Byte_1, Byte_2 of float parameter 2

Return parameter	
FU32	Status 0 = OK 1 = fault in accessing FpMathDr-driver 2 = number outside range 3 = mathematical fault during conversion
FU33	Byte_3, Byte_4 of the float result
FU34	Byte_1, Byte_2 of the float result

FPSQRT

Carries out Sqrt (FloatNumber)

Use in FST Insert function module FPSQRT in the project, declare as type F (function module) and assign with a module number of your choice. The function module requires FPMATHDR version 1.50 or later.

This function module carries out the following operation:

– The input parameter FloatNumber is a floating point number. The result is output as a floating point number.

Input parameter	
FU32	Byte_3, Byte_4 of FloatNumber
FU33	Byte_1, Byte_2 of FloatNumber

Return parameter	
FU32	Status 0 = OK 1 = fault in accessing FpMathDr-driver 2 = number outside range 3 = mathematical fault during conversion
FU33	Byte_3, Byte_4 of the float result
FU34	Byte_1, Byte_2 of the float result

FPGONIO Trigonometrical functions

The function module requires FPMATHDR version 1.60 or later. The angles to be calculated must be specified in circular measure (radian).

Input parameter	
FU32	Function: 1 = sin, 2 = cos, 3 = tan, 4 = asin, 5 = acos, 6 = atan
FU33	Byte_3, Byte_4 of FloatNumber
FU34	Byte_1, Byte_2 of FloatNumber

Return parameter	
FU32	Status 0 = OK 1 = fault in accessing FpMathDr-driver 2 = number outside range 3 = mathematical fault during conversion 4 = invalid function pre-select value
FU33	Byte_3, Byte_4 of the float result
FU34	Byte_1, Byte_2 of the float result

Notes on conversion of float – longint (= integers of type long)

Note 1 – accuracy

According to the documentation concerning IEEE-754 – floating point numbers (represented using 4 bytes), its accuracy is approx. 7 figures. This leads to a loss of accuracy with conversions of integers of type long into float, if long integers are used which consist of more than 7 figures.

Example 1

LongInt number = 1966080500 (753001F4h)

After conversion into float using FPL2F.COM function module the following applies:

Float number (bytes 1, 2, 3, 4) = 4E EA 60 04

After conversion back into long integer using FPL2F.COM, the following applies:

LongInt number = 1966080512 (75300200h)

Example 2

LongInt number = 196608500 (0BB801F4h)

After conversion into float using FPL2F.COM function module, the following applies:

Float number (bytes 1, 2, 3, 4) = 4D 3B 80 1F

1) After conversion back into long integer using FPF2L.COM and range specification = 0, the following applies gilt:

LongInt number = 196608496 (0BB801F0h)

2) After conversion back into long integer using FPF2L.COM and range specification = 1, the following applies:

LongInt number = 1966084992 (75301380h) and value range = -1 (FFFFh)

Note that the conversion with automatic range (2.) produces a more exact value.

Note 2 – with or without sign?

Note that the conversion function modules assume that all long-integer values have a sign.
Example:

LongInt number = -65535 (FFFF0001h)

After conversion into float using FPL2F.COM function module, the following applies:

Float number (bytes 1,2,3,4) = C7 7F FF 00

1) After conversion back into long integer using FPF2L.COM and range specification = 0, the following applies:

LongInt number = -65535 (FFFF0001h)

2) After conversion back into long integer using FPF2L.COM and range specification = 1, the following applies:

LongInt number = -655350016 (D8F02700h) and value range = -4 (FFFCh)

1.7.4 Example: Use of the modules

The following example shows the use of modules FPI2F, FPBINOP and FPF2A.

Example

```
STEP
"" Convert FW0 to float – result is FW0 * (10 ^ FW1)
 IF                NOP
 THEN  CFM 0                     " Convert 16-bit integer to float
         WITH      FW0           " 16-bit integer with sign
         WITH      FW1           " Value range
"" If conversion is successful, then load result to FW20,FW21
 IF                FU32          'Parameter 1
         =         V0
 THEN  LOAD        FU33          'Parameter 2
         TO        FW20
       LOAD        FU34          'Parameter 3
         TO        FW21

STEP
"" Convert FW2 to float – result is FW2 * (10 ^ FW3)
 IF                NOP
 THEN  CFM 0                     " Convert 16-bit whole number to float
         WITH      FW2           " 16-bit integer with sign
         WITH      FW3           " Value range

"" If conversion is successful, then load result to FW22,FW23
 IF                FU32          'Parameter 1
         =         V0
 THEN  LOAD        FU33          'Parameter 2
         TO        FW22
       LOAD        FU34          'Parameter 3
         TO        FW23

STEP
"" Division of intermediate results 1 and 2
 IF                NOP
 THEN  CFM 1                     " Floating point binary operations
         WITH      V1            " Operation code
                                 " (0=*,1=/,2=+,3=-,4=^,5=ln)
         WITH      FW20          " Bytes 3,4 of parameter 1 (F1)
         WITH      FW21          " Bytes 1.2 of parameter 1 (F1)
         WITH      FW22          " Bytes 3,4 of parameter 2 (F2)
         WITH      FW23          " Bytes 1.2 of parameter 2 (F2)
```

```
"" If division is successful, then load result to FW30, FW31
 IF               FU32           'Parameter 1
        =         V0
 THEN  LOAD       FU34           'Parameter 3
         TO       FW30
       LOAD       FU35           'Parameter 4
         TO       FW31

STEP
"" Convert result to string with 5 post-decimal positions
 IF               NOP
 THEN  CFM 2                     " Convert float to string
         WITH     FW30           " Bytes 3,4 of float number
         WITH     FW31           " Bytes 1.2 of float number
         WITH     V5             " Number of post-decimal positions
         WITH     V1             " Number of the string for the
                                 " Result
 IF               FU32           'Parameter 1
        =         V0
 THEN             NOP
```

1.8 String driver (driver STRINGS)

The String Driver of the FST provides a new additional data type STRING. 256 strings are supported as a pre-setting. Individual strings are addressed with a number which represents the string number. Strings can contain any characters except the NUL character (hexadecimal $00). Strings can be of any length within the framework of the set memory requirement. The strings are not remanent.

1.8.1 Configuring and parametrizing the driver

If you wish to use strings in an FST IPC project, you must enter and parametrize the driver STRINGS in the driver configurator.

IPC drive

Specify the drive on which the string driver STRINGS.EXE can be found or onto which it must be loaded.

Reserved memory in bytes

Specify the maximum memory space for strings. Permitted range from 5 to 65000 bytes. The pre-setting is 2000 bytes. This setting can also be carried out or be modified by the module STRINIT.

Number of strings

Specify the maximum number of strings. Permitted range from 5 to 1024 strings. The pre-setting is 256 strings.

File with pre-assignments

Specify the name of the file which contains the initialization values for the strings. The format of this file is described in the following section.

1.8.2 Initializing strings

A simple text file is used for initializing the strings. It must have the extension “.TXT”. Each line of this file is a string. The first line becomes string 0. Missing strings will be initialized as empty. Line feed characters (CR and LF) will be removed from the text file.

Special characters

Special characters are represented by a combination of two characters, the character ’\’ and a further character. The following special characters are possible:

Special characters	**Meaning**	**Description**
\a	alert	Bell tone (signal tone)
\b	backspace	Positioning one character backwards
\f	formfeed	Page feed (FF)
\n	linefeed	Linefeed (LF)
\r	return	New line (CR)
\t	tab	Tabulator character
\‹Nr›		Hexadecimal definition of a character; ‹Nr› must begin with a figure, e.g. “\0A8” (correct) instead of “\A8” (incorrect).
\\	\	The character ’\’ is represented by two ’\\’.

1.8.3 Additional CI commands

The command scope of the CI is increased by the string driver. The additional commands for the string driver are:

CI command	Description
!3	Shows driver information and version number. This information is also shown if an unknown command is entered (e.g. !3abcdef).
!3Dx	Display of string x
!3Mx=text	Setting string x with the character sequence text
!3S	Status display, Result "= count=‹XX›, storage=‹YY›, ‹ZZ›" with: – XX = number of strings – YY = reserved memory location – ZZ = OK, if string memory is OK, or = BAD, if string memory is defective

In these commands, "!3" is the prefix for a CI call in a driver, here in driver 3 for strings.

1.8.4 Modules for dealing with strings

There is a series of modules for dealing with strings. These must be imported in a project as usual. The strings concerned will each be specified with the string number.

Example

In order to clear string 5, i.e. to bring it to length 0, the module STRCLR must be accessed. Module STRCLR should be imported as CFM 73. The following should be programmed in STL:

```
THEN     . . .
 CFM 73        " Access STRCLR for deleting
  WITH V5      " of string 5
```

In order to copy string 6 into string 12, you must access module STRCPY. Module STRCPY should be imported as CFM 74. The following should be programmed in STL:

```
THEN     . . .
 CFM 74        " Access STRCPY
  WITH R0      " copy string with number
               " in R0
  WITH V12     " to string 12
```

Most string modules return a result for fault recognition.

Overview of modules

Module	Brief description
STRADDR	Ascertain internal address of a string
STRAPPND	Append character to a string
STRATOH	Convert hexadecimal string into word
STRATOI	Convert string into word with sign (signed)
STRATOIX	Convert string into word with sign (signed)
STRATOU	Convert string into word without sign (unsigned)
STRCAT	Combine two strings into a third
STRCHECK	Memory check
STRCHGET	Extract character from a string
STRCHSET	Replace character in a string
STRCI	Carry out a CI command
STRCLR	Clear string
STRCMP	Character-by-character comparison of two strings, a distinction is made between lower and upper case letters
STRCPY	Copy string
STRDEL	Remove part of a string
STRDUMP	Output some strings
STRFILL	Create string with specified number of equal characters
STRFILLW	Fill string with another string right or left-justified
STRFINDC	Search for a character in a string
STRFINDS	Search for part of a string in a string
STRGROW	Increase string memory for an individual string
STRHTOA	Convert word into hexadecimal string

Module	Brief description
STRICMP	Character-by-character comparison of two strings, no distinction between lower and upper case letters
STRINIT	Initialization or re-initialization
STRINSRT	Inserting a string into another
STRITOA	Convert word with sign into string
STRLEFT	Transfer left string part
STRLEN	Length of a string
STRLOWER	Convert string into lower case letters
STRMID	Transfer centre string part
STRNCMP	Comparison of the first characters of two strings, a distinction is made between lower and upper case letters
STRNICMP	Comparison of the first characters of two strings, no distinction between lower and upper case letters
STRRIGHT	Transfer right string part
STRSTAT	Status of string driver
STRUPPER	Convert string into upper case letters
STRUSAGE	User and free memory
STRUTOA	Convert word without sign into string
STR2FLAG	Copy a string into a flag word range
FLAG2STR	Copy a flag word range into a string

STRADDR — Ascertain internal address of a string

Input parameter	
FU32	Number of the string

Return parameter	
FU32	Offset of address, 0 with non-permitted string number
FU33	Offset of address, 0 with non-permitted string number

As the lengths of the strings concerned can change dynamically, the strings are shifted within the string memory, if this is necessary. It then follows that a string address once ascertained will be modified as a result of string operations with other strings. The addresses of strings are therefore usable only under special conditions.

STRAPPND — Hang a character onto a string

Input parameter	
FU32	Number of the string
FU33	Character

Return parameter	
FU32	0 if successful, otherwise fault

STRATOH

Convert hexadecimal string into word

Input parameter	
FU32	Number of the string

Return parameter	
FU32	0 if successful, otherwise fault
FU33	Ascertained value

Empty spaces and tabulator characters are permitted at the beginning and end of the string. A '$' character in front of the hexadecimal figure characters is also permitted. Only positive values can be converted. The hexadecimal figures must be contained in the string as upper case letters. Four figures can be converted (e.g. ABC2).

STRATOI

Convert string into word with sign (signed)

Input parameter	
FU32	Number of the string

Return parameter	
FU32	0 if successful, otherwise fault
FU33	Ascertained value

Empty spaces and tabulator characters are permitted at the beginning and end of the string. A '+' or '-' character in front of the figure characters is also permitted.

STRATOIX

Convert string into word with sign (signed)

Input parameter	
FU32	Number of the string
FU33	Position as from which conversion is made

Return parameter	
FU32	0 if successful, otherwise fault
FU33	Ascertained value
FU34	Number of characters read

Empty spaces and tabulator characters are permitted at the beginning and end of the string. A ‘+’ or ‘-’ character in front of the figure characters is also permitted.

STRATOU

Convert string into word without sign (unsigned)

Input parameter	
FU32	Number of the string

Return parameter	
FU32	0 if successful, otherwise fault
FU33	Ascertained value

Empty spaces and tabulator characters are permitted at the beginning and end of the string. A ’+’ character in front of the figure characters is also permitted.

STRCAT

Grouping two strings into a third

Input parameter	
FU32	Number of the first source string
FU33	Number of the second source string
FU34	Number of the target string

Return parameter	
FU32	0 if successful, otherwise fault

STRCHECK

Checking the string memory

Input parameter
None

Return parameter	
FU32	0 if successful, otherwise fault

STRCHGET Extract indexed character from a string

Input parameter	
FU32	Number of the string
FU33	Character index, 1 for first character of the string

Return parameter	
FU32	0 if successful, otherwise fault
FU33	Extracted character

STRCHSET Replace indexed character in a string

Input parameter	
FU32	Number of the string
FU33	Character index, 1 for first character
FU34	Replacement character in lower-value byte

Return parameter	
FU32	0 if successful, otherwise fault

STRCI

Carry out a CI command

Input parameter	
FU32	Number of string with the CI command
FU33	Number of string for the result of the command

Return parameter	
FU32	0 if successful, otherwise fault

The resulting string of the command must not be longer than 80 characters. The result of the CI command is not interpreted.

STRCLR

Clear string

i.e. empty the string, bring its length to 0.

Input parameter	
FU32	Number of the string

Return parameter	
FU32	0 if successful, otherwise fault

STRCMP

Character-by-character comparison of two strings

Input parameter	
FU32	Number of the first string
FU33	Number of the second string

Return parameter	
FU32	0 if successful, otherwise fault
FU33	1 = Character in the first string › as character in the second string 0 = Both strings are identical -1 = Character in the first string ‹ as character in the second string

A distinction is made between upper and lower case letters. Upper case letters have less value than lower case letters.

STRCPY

Copy string into another

Input parameter	
FU32	Number of the source string
FU33	Number of the target string

Return parameter	
FU32	0 if successful, otherwise fault

STRDEL

Remove part of a string

Input parameter	
FU32	Number of the string
FU33	Index for first character to be removed, 1 for first character
FU34	Number of characters

Return parameter	
FU32	0 if successful, otherwise fault

STRDUMP

Debug output of some strings on the monitor of the IPC

Input parameter	
FU32	Number of the first string
FU33	Number of the last string

Return parameter
None

This module should not be used in normal operation. If necessary, the extended CI commands of the string driver can be used.

STRFILL

Create string with specified number of equal characters

Input parameter	
FU32	Number of the string
FU33	Number of characters
FU34	Fill character

Return parameter	
FU32	0 if successful, otherwise fault

STRFILLW

Fill string with another string right or left-justified

Input parameter	
FU32	Number of the string to be created
FU33	Number of characters for this string › 0 for right-justified ‹ 0 for left-justified
FU34	Number of the string to be transferred

Return parameter	
FU32	0 if successful, otherwise fault

If the string to be created has been specified too small, the resulting string will be cut short.

STRFINDC

Search for a character in a string

Input parameter	
FU32	Number of the string
FU33	Search character

Return parameter	
FU32	0 if successful, otherwise fault
FU33	› 0 Position 1 for first character 0 for character not contained

STRFINDS

Search for part of a string in a string

Input parameter	
FU32	Number of string in which the search is made
FU33	Number of the string part to be searched

Return parameter	
FU32	0 if successful, otherwise fault
FU33	› 0 Position 1 for first character 0 if part string is not contained or is empty

STRGROW

Increase string memory for an individual string from the existing memory

Input parameter	
FU32	Number of the string
FU33	New maximum size of string in characters (without \0 at end of string)

Return parameter	
FU32	0 if successful, otherwise fault

The existing string is retained without modification. A string with maximum the set length can then be written on the address ascertained with STRADDR. Every other call of a string module can modify the set value again.

STRHTOA

Convert word into hexadecimal string

Input parameter	
FU32	Value to be converted
FU33	Number of the string

Return parameter	
FU32	0 if successful, otherwise fault

A '$' character is inserted in front of the four hexadecimal figure characters.

STRICMP

Character-by-character comparison of two strings

Input parameter	
FU32	Number of the first string
FU33	Number of the second string

Return parameter	
FU32	0 if successful, otherwise fault
FU33	1 = Character in the first string › as character in the second string 0 = Both strings are identical -1 = Character in the first string ‹ as character in the second string

No distinction between upper and lower case letters.

STRINIT

Setting the memory size for strings and the maximum number of strings and restoring the initialization values of the strings

Input parameter	
FU32	Size of the string memory (minimum 1000)
FU33	Maximum number of strings (minimum 10)

Return parameter	
FU32	0 if successful, otherwise fault

STRINSRT

Inserting a string into another string as from a specified position

Input parameter	
FU32	Number of the string into which the other is to be inserted
FU33	Position in front of which insertion is to be made, 1 = in front of first character
FU34	Number of the string which is to be inserted

Return parameter	
FU32	0 if successful, otherwise fault

STRITOA

Convert word with sign into string

Input parameter	
FU32	Value to be converted
FU33	Number of the string

Return parameter	
FU32	0 if successful, otherwise fault

A '-' character is inserted in front of the figure characters, if necessary.

STRLEFT

Transfer left string part with specified length into a string

Input parameter	
FU32	Number of the source string
FU33	Number of characters
FU34	Number of the target string

Return parameter	
FU32	0 if successful, otherwise fault

If the source string is shorter than the specified length, the source string will be copied.

STRLEN

Length of a string

Input parameter	
FU32	Number of the string

Return parameter	
FU32	0 if successful, otherwise fault
FU33	Length

STRLOWER

Convert string into lower case letters

Input parameter	
FU32	Number of the string

Return parameter	
FU32	0 if successful, otherwise fault

Country-specific characters such as the umlauts Ä, Ö, Ü will not be modified.

STRMID

Transfer centre string part

Transfer left string part as from a specified starting position and with specified length into a string

Input parameter	
FU32	Number of the source string
FU33	Starting character in source string, 1 for first character of the source string
FU34	Number of characters
FU35	Number of the target string

Return parameter	
FU32	0 if successful, otherwise fault

If the starting position is not contained in the source string, an empty string will be created. If the source string is too short, the target string will be shortened accordingly.

STRNCMP

Comparison of the first characters of two strings

Input parameter	
FU32	Number of the first string
FU33	Number of the second string
FU34	Number of characters

Return parameter	
FU32	0 if successful, otherwise fault
FU33	1 = Character in the first string › as character in the second string 0 = Both strings are identical -1 = Character in the first string ‹ as character in the second string

A distinction is made between upper and lower case letters. Upper case letters have less value than lower case letters.

STRNICMP

Comparison of the first characters of two strings

Input parameter	
FU32	Number of the first string
FU33	Number of the second string
FU34	Number of characters

Return parameter	
FU32	0 if successful, otherwise fault
FU33	1 = Character in the first string › as character in the second string 0 = Both strings are identical -1 = Character in the first string ‹ as character in the second string

No distinction between upper and lower case letters.

STRRIGHT

Transfer right string part with specified length into a string

Input parameter	
FU32	Number of the source string
FU33	Number of characters
FU34	Number of the target string

Return parameter	
FU32	0 if successful, otherwise fault

If the source string is too short, the target string will be shortened accordingly.

STRSTAT

Interrogate status of string driver

Input parameter
None

Return parameter	
FU32	Set memory size for strings
FU33	Maximum number of strings
FU34	Memory used by strings
FU35	Remaining free memory space

STRUPPER

Convert string into upper case letters

Input parameter	
FU32	Number of the string

Return parameter	
FU32	0 if successful, otherwise fault

Country-specific characters such as the umlauts ä, ö, ü will not be modified.

STRUSAGE

Ascertain and use free memory for strings

Input parameter
None

Return parameter	
FU32	Memory used by strings
FU33	Remaining free memory space

STRUTOA

Convert word without sign into string

Input parameter	
FU32	Value to be converted
FU33	Number of the string

Return parameter	
FU32	0 if successful, otherwise fault

STR2FLAG

Copy a string into a flag word range

Input parameter	
FU32	Number of the string
FU33	Maximum number of characters to be copied
FU34	Number of the starting flag word

Return parameter	
FU32	0 if successful, otherwise fault
FU33	Number of characters actually copied [1]

[1] If the maximum number is greater than the actual string length, a concluding zero will be copied and counted in the case of an odd number of characters.

FLAG2STR

Copy a flag word range into a string

Input parameter	
FU32	Number of the string
FU33	Number of characters to be copied
FU34	Number of the flag word

Return parameter	
FU32	0 if successful, otherwise fault

Each flag word contains two characters.

Modules STR2FLAG and FLAG2STR can be used together with the file modules FREAD and FWRITE for writing strings into a file. The specified flag word range will thereby be used as intermediate memory.

1.9 PID controller (PID driver)

The PID driver provides 16 PID controllers for your FST IPC projects.

1.9.1 Introduction

The control parameters as well as the controller input and output values are provided via flag words. 16 consecutive flag words are assigned for each controller. A coherent range of 256 flag words is then assigned consecutively for all 16 controllers.

The assigned flag word range is called a parameter table. You can set the first flag word of this range with a module.

1.9.2 Configuring and parametrizing the driver

In order to use the PID controller in an FST project, the PID driver must be entered in the driver configuration and the necessary parameters must be specified.

Target disc drive

Specify the drive on which the PID driver can be found or onto which it must be loaded.

1.9.3 Additional CI commands

This driver supplements the CI with the following commands:

CI command	Description
!29	Shows driver information and version number. This information is also shown if an unknown command is entered (e.g. !29abcdef).

1.9.4 Module for the PID driver

PIDCFM Set parameter table and start driver

Determines the flag word range for the parameter table and starts the PID driver.

Please note
If module PIDCFM is accessed, all flag words in the parameter table will be reset to 0. All 16 controllers are then deactivated and can be started again with the parameter Man/Auto.

Input parameter	
FU32	Number of the first flag word for the parameter table

Return parameter	
FU32	0 if successful 256 if driver is not installed

Parameter table

The following parameter table is the result of accessing PIDCFM with 500.

Parameters	PID controller 0	PID controller 1	PID controller ...	PID controller 15
Man/Auto	FW500	FW516	...	FW740
Nominal value	FW501	FW517	...	FW741
Actual value	FW502	FW518	...	FW742
Variable	FW503	FW519	...	FW743
Specified value	FW504	FW520	...	FW744
Kp	FW505	FW521	...	FW745
Ki	FW506	FW522	...	FW746
Kd	FW507	FW523	...	FW747
Time constant	FW508	FW524	...	FW748
Reserved	FW509 ... FW515	FW525 ... FW531	...	FW749 ... FW755

Explanation of the parameters

Parameters	Description
Man/Auto	If 0 is entered here, the controller is switched off. The specified value is output as a positioning variable. If 1 is entered here, the controller is switched on. The positioning variable is then calculated as a factor of the positioning value, the processing variables and the PID parameters.
Nominal value	Target value for the controller
Actual value	The input value for the controller
Positioning variable	The output word of the controller
Specified value	Is output in manual operation as the positioning variable.
Kp	Proportional value of the controller
Ki	Integral value of the controller
Kd	Differential value of the controller
Time constant	Determines the time interval during which the controller calculates a new value.

If the time constant is 0, calculation will take place approx. every 14 ms; if it is 1, calculation will take place every 28 ms etc.

Nominal value, actual value, positioning variable and specified value lie in the range 0 to 4095.

Kp, Ki and Kd are divided internally by 100. If you wish to use e.g. 1.25, enter 125 in the table.

1.10 Serial communication

This driver enables characters to be sent to and received from serial interfaces as per RS232.

PLC/IPC	Description
CPX-FEC	The programming interface (RS232) can be used for serial communication.
FEC Compact and FEC Standard	The interfaces COM and EXT can be used for serial communication. Due to the very limited memory capacity of the FC20, only FEC Compact controllers of the FC3x series and FEC Standard controllers should be used for serial communication.
HC0X	The interfaces COM and EXT can be used for serial communication. Additional COM interfaces on CP3x modules can be used as follows: – In order to prevent a collision between the external interrupt sources (IRQs), use either COM2 or COM4. In this case the trimmer for the HC01 cannot be used. The jumper J1 in the HC0X must be connected differently (see Hardware Manual). There is no trimmer on the HC02 and jumpers do not need to be connected differently. – In order to prevent a collision between the external interrupt sources (IRQs), you can use either COM3 or COM5. – With the HC02 it is possible to use COM1, but in this case the Ethernet interface cannot be used and a jumper must be connected differently (see Hardware Manual for details). COM1 can be used without restrictions on the HC01.
HC1X and HC2X	Up to 4 interfaces are supported. These are either COM1..COM4 or COM2..COM5. Please note. The serial interface marked “COM” on the HC2X CPU is COM1. The serial interface marked “EXT” on the HC20 CPU is COM2.

As from version 1.31 the driver supports the FIFOs if they are available. The version number of the loaded driver can be interrogated as from version 1.31 with the CI command “!8”.

1.10.1 Selecting and parametrizing the driver

If you wish to use the serial diver in an FST IPC project, you must enter and parametrize the relevant driver in the driver configurator.

Target system	Driver	Description
CPX-FEC	COMEXT	Use the COMEXT driver
FEC Compact	COMEXT	Use the COMEXT driver. The driver FOSEXT must also be entered for the EXT interface. FC20 ‹ V1.21 and FC30 ‹ S1.10 require FOSSIL2 instead of FOSEXT.
	FOSEXT	
	FOSSIL2	
FEC Standard	COMEXT	Use the COMEXT driver
HC0X	COMEXT	– If you use only COM and/or EXT, please use the driver COMEXT. – If you use a CP3x module, use HC0XCOM (this driver also contains the modules for COM and EXT). – With HC0X ‹ S2.00 the driver FOSEXT must also be entered for the EXT interface (HC0x ‹ S1.11 requires FOSSIL2 instead of FOSEXT).
	HC0XCOM	
	FOSEXT	
	FOSSIL2	
HC1X and HC2X	SERIALDR	Use the SERIALDR driver.

Configuration with the SERIALDR driver

Target disc drive

Specify the drive on which the serial driver SERIALDR.EXE can be found or onto which it must be loaded.

1.10.2 Modules for serial communication

The modules for serial communication acquire the used interface as parameter (FU32). Please note the following interface number assignment:

Interface	Parameter
COM	255
EXT	0
COM1	1
COM2	2
COM3	3
COM4	4
COM5	5

Please note
The modules must not use the same interface as the command interpreter, unless it has the number 255 (COM).

The standard interface for the command interpreter is determined in the PLC settings of the project. The presetting for the standard interface is:

Target system	Port	Number
CPX-FEC	Programming interface	255
FEC Compact and FEC Standard	COM	255
HC0X	COM	255
HC1X	COM1	1
HC2X	COM	1

Note: The characters #0 or Ctrl-T cannot be sent or received via the interface with number 255.

Overview of modules

Module	Brief description
OPENCOM	Open serial interface
OPENCOMX	Open serial interface and set parameters
CLOSECOM	Close serial interface
GETCOM	Read a character from a serial interface
PUTCOM	Send a character to a serial interface
PRINTCOM	Write an FST string to a serial interface
READCOM	Read character chain and save in FST string (without data delimiter)
READLCOM	Read character chain and save in FST string (with data delimiter)
F30	Set interface parameter
F31	Activate CI
F32	Empty interface buffer
F34	Interrogate number of records in receive buffer
F35	Modify standard data delimiter
BREAKCOM	Create hardware break (only SERIALDR)
SETRTS	Switch on/off RTS for RS485 (only SERIALDR)
IS485	Interrogate whether interface is in RS485 mode (only SERIALDR)

OPENCOM

Open serial interface (with 9600 Baud, 8 data bits, no parity)

Input parameter	
FU32	Serial interface

Return parameter	
FU32	0 = ready 1 = fault

Use F30 or OPENCOMX for other settings.

OPENCOMX

Open serial interface and set parameters

Input parameter	
FU32	Serial interface
FU33	Baud rate
FU34	Data bits (5...8)
FU35	Stop bits (1, 2)
FU36	Parity 0 = no parity 1 = odd parity 2 = even parity
FU37	CI mode 0 = no CI 1 = activate CI [1)] 2 = activate CI (ignore command “X”) [1)]

[1)] If the CI is activated with 1, it can be activated again after receiving the “X” command only with module OPENCOMX or F31. If the CI is activated with 2, the “X” command will be ignored.

Return parameter	
FU32	0 = ready 1 = fault

When the interface has been opened successfully, the input and output buffers are emptied automatically.

Please note
Not all baud rates are supported by each platform.

The following table gives an overview of the baud rates supported by the software. It is possible that the hardware does not support all the settings.

PLC/IPC	300	1200	2400	9600	19200	38400	57600
CPX-FEC	X	X	X	X	X	X	X
FEC Compact (Firmware ‹ S2.00)	–	X	X	X	X	X	–
FEC Compact	–	X	X	X	X	X	–
FEC Compact (Firmware ›= S2.04)	X	X	X	X	X	X	X
FEC Standard	–	X	X	X	X	X	–
FEC Standard (Firmware ›= S1.10)	X	X	X	X	X	X	X
HC0x EXT port (Firmware ‹ S2.00)	–	X	X	X	X	X	X
HC0x EXT port	–	X	X	X	X	X	–
HC0x EXT port (Firmware ›= S2.03)	X	X	X	X	X	X	X
HC0x CP31	X	X	X	X	X	X	X
HC20-40	X	X	X	X	X	X	X

Please note
FEC Compact, FEC Standard and HC0x (EXT port) require drivers version 1.40 or later for 57600 baud.

CLOSECOM

Close opened serial interface

Input parameter	
FU32	Serial interface

Return parameter	
FU32	0 = ready 1 = fault

GETCOM

Read a character from a serial interface

Input parameter	
FU32	Serial interface

Return parameter	
FU32	0 = ready 1 = fault -1 = nothing received
FU33	If FU32 = 0 (ready), FU33 will contain the character received (0 to 255).

PUTCOM

Send a character to a serial interface

Input parameter	
FU32	Serial interface
FU33	Character to be sent (0 to 255)

Return parameter	
FU32	0 = ready 1 = fault

PRINTCOM

Write an FST string to a serial interface

Input parameter	
FU32	Serial interface
FU33	Number of the FST string

Return parameter	
FU32	0 = ready 1 = fault

READCOM

Read character chain from the interface and save in FST string

Input parameter	
FU32	Serial interface
FU33	Number of the FST string for the characters
FU34	Maximum length
FU35	Data delimiters (0...255)

Return parameter	
FU32	0 = ready 1 = fault -1 = nothing received

The data delimiter is not written in the string.

READLCOM

Read character chain from the serial interface and save in FST string (with data delimiter)

Input parameter	
FU32	Serial interface
FU33	Number of the FST string for the characters
FU34	Maximum length with data delimiters

Return parameter	
FU32	0 = ready 1 = fault -1 = nothing received

The data delimiter is transferred to the string. The presetting for the data delimiter is CR.

F30 Set interface parameter

Input parameter	
FU32	Serial interface
FU33	Interface parameter

Return parameter
None

Interface parameter

	Baud rate				Character length		Parity	
Bit	7	6	5	4	3	2	1	0

Baud rate

Bit	7	6	5	4	Baud
	1	0	0	0	19200
	1	0	0	1	9600 (presetting)
	1	0	1	0	4800
	1	0	1	1	2400
	1	1	0	0	1200
	1	1	0	1	600
	1	1	1	0	300
	1	1	1	1	110

Character length			
Bit	**3**	**2**	**Bits/characters**
	0	0	5
	0	1	6
	1	0	7
	1	1	8 (presetting)

Parity			
Bit	**1**	**0**	**Parity**
	0	0	NONE (presetting)
	0	1	ODD
	1	0	NONE
	1	1	EVEN

Please note

– Only the setting N,8,1 is permitted for the COM interface (255).

– Due to the optocouplers, only 9600 Baud or a lower baud rate is permitted for the COM interface (255) of the FC20.

F31 Activate CI

Input parameter	
FU32	Serial interface
FU33	Driver mode: 0 = Disable CI (presetting) 1 = Enable CI, disable, if the CI command “X” is received 2 = Enable CI, ignore CI command “X”

Return parameter
None

The standard CI interface cannot be switched off.

The standard interface for the command interpreter is determined in the PLC settings of the FST project. The presetting for the standard interface is:

Controller	Port	Number
FEC	COM	255
HC0X	COM	255
HC1X	COM1	1
HC2X	COM	1

F32 Empty interface buffer

Input parameter	
FU32	Serial interface

Return parameter	
FU32	0 = ready 1 = fault

F34 Interrogate number of records in receive buffer

Input parameter	
FU32	Serial interface

Return parameter	
FU32	Interrogate number of records in receive buffer (separated by standard block end characters)

F35 Modify standard data delimiter

Input parameter	
FU32	Serial interface
FU33	Data delimiter (0 to 255); presetting is CR (13)

Return parameter
None

BREAKCOM Create hardware break

Input parameter	
FU32	Serial interface
FU33	1 = Output BREAK 0 = Withdraw BREAK

Return parameter
None

Information on the effects of a “break” can be found in the documentation for the device connected.

IS485

Interrogate whether interface is in RS485 mode

Input parameter	
FU32	Serial interface

Return parameter	
FU32	0 = ready 1 = fault
FU33	1 = RTS switched on for RS485 0 = normal RS232 operation

FST driver version 1.40 or later is required for FEC and HC0X.
FST driver SERIALDR version 1.10 or later is required for HC1X and HC2X.

SETRTS

Switch on/off RTS for RS485 serial communication via IPC module SM30 or SM35

Input parameter	
FU32	Serial interface
FU33	1 = switch on RTS for RS485 0 = normal RS232 operation

Return parameter	
FU32	0 = ready 1 = fault

Note

– FEC Compact supports RS485 as from firmware S2.04. Driver version 1.40 or later is required. SETRTS must be accessed when the interface has been opened.

- FEC Standard supports RS485 as from firmware S1.10. Driver version 1.40 or later is required. SETRTS must be accessed when the interface has been opened.
- HC0x supports RS485 as from firmware S2.03. Driver version 1.40 or later is required. SETRTS must be accessed when the interface has been opened.
- FST IPC driver SERIALDR version 1.10 or later is required for HC1X and HC2X. SETRTS must be accessed before the interface is opened.

1.11 TCP/IP (driver TCPIP...)

With the TCP/IP driver your PLC/IPC is able to communicate via the Ethernet interface with other PCs or PLC/IPCs by means of the UDP or TCP protocol of the TCP/IP protocol series.

Each TCP/IP participant (host) requires an IP address and IP net mask. The FST TCP/IP driver offers 3 possibilities for carrying out these settings:

– in the driver configuration

– by means of module access

– dynamically via the BOOTP/DHCP protocol.

Many local networks use DHCP servers which automatically assign an address to new devices. The FST TCP/IP driver supports such DHCP servers.

Please note
If the IP address is set to 0.0.0.0 (by driver configuration or module access), the BOOTP protocol will automatically be activated. When an IP address is set, the BOOTP protocol is switched off.

If the DHCP server is used, programming for communication between the controllers will take somewhat longer. A specific target IP address is required each time data is sent/received. The controllers must first exchange these IP addresses with each other.

If the IP address of the controller is assigned by the DHCP server, the address must be read out in the program and then written by broadcast to all connected controllers in a previously agreed flag word. For the communication the target address can then be read from this flag word.

The FST TCP/IP driver supports the following protocol/services for communication and for tests:

Protocol	Port	Name	Description
ICMP	–	–	Used for PING
UDP	7	echo	Returns each character received
UDP	9	discard	Discard all data received
UDP	13	daytime	Returns date and time as character chain
UDP	19	charge	Returns character chain
UDP	37	time	Returns time as 32-bit number
UDP	991	CI	Command interpreter
UDP	992	CI_EXT	Extended command interpreter
UDP	993	CI_BIN	Binary exchange of data
UDP	995	EasyIP	Data exchange protocol
TCP	7	echo	Returns each character received
TCP	9	discard	Discard all data received
TCP	13	daytime	Returns date and time as character chain
TCP	19	charge	Sends characters at regular short intervals
TCP	991	CI	Command interpreter (via telnet)

The TCP command interpreter at port 991 can be accessed by a program such as TELNET.

Start TELNET with the parameters, IP number and 991 (i.e. TELNET 10.10.10.1 991). Now it is possible to send commands and replies as with the normal RS232 command interpreter.

Please note
Please note that only one connection at a time is possible. The connection will be interrupted automatically if no message is sent within 60 seconds.

The command interpreter can also be accessed as follows: Send a UPD datagram with the CI command (without CR/LF) to the FST controller, port 991. The result will be returned. The result of DR0 is e.g. =1234. Operation of the extended CI on port 992 is similar, with a small extension. The original command will be returned followed by a zero character and the result, e.g.:

DR0‹Zero character›=1234.

The use of the TCP (Telnet) is more suitable for users, while the use of the UDP is intended more for programs for accessing FST operands.

Caution
The command Y! must not be entered. This command will stop all programs and drivers. As the TCP/IP is implemented as a driver, this will also be stopped.

Please note
The LE commands cannot be accessed via Telnet.

Selecting and parametrizing drivers

If you wish to use TCP/IP in an FST IPC project, you must enter and parametrize the relevant TCP/IP driver in the driver configuration. The drivers of the individual target systems have different names, see the following table:

Driver name	Target system
TCPIPCPX	– CPX terminal with CPX-FEC
TCPIPFEC	– FEC Compact (FC34, FC44)
TCPIPFC2	– FEC Standard (FC440, FC560, FC640, FC660)
TCPIPDRV	– PS1-HC1x with PS1-CP10/CP11/CP14 – PS1-HC1x with PS1-CP12 (8-bit mode) – PS1-HC2x with PS1-CP10/CP11/CP14 – PS1-HC2x with PS1-CP12 (8-bit mode)
TCPIPHC0	– PS1-HC02
TCPIP_15	– PS1-HC1x with PS1-CP15 – PS1-HC2x with PS1-CP15
TCPIPXXX	– PS1-HC1x with 2 PS1-CP10/CP11/CP14 – PS1-HC2x with 2 PS1-CP10/CP11/CP14

1.11.1 Configuration of the TCP/IP driver

The following settings are required for the configuration of a TCP/IP driver, depending on the handware used:

Driver settings	Description
Target disc drive	IPC drive on which the TCP/IP driver can be found or onto which it must be loaded.
IP address	The IP address consists of 4 bytes, i.e. 4 numbers between 0 and 255. The addresses 0.0.0.0 and 255.255.255.255 are not permitted. In a closed-loop local network, consisting only of your controllers and programming PC, you can select the address freely. If your network is connected to your company network, you must usually ask your data processing department for the permitted IP addresses. Leave the address 0.0.0.0, if it has been assigned with module IP_IP, or dynamically by the BOOTP/DHCP protocol.
IP network mask	The mask limits the use of the addresses. A bit of the network mask corresponds to each individual bit of an IP address. If the bit of the network mask is 1, then the bit of the IP addresses between two communicating partners must be identical. Leave the address 255.255.255.0 if it has been assigned with module IP_MASK, or dynamically by the BOOTP/DHCP protocol.
IP address of the gateway	If your controller network is to be connected to another network with the aid of a gateway, the address of this gateway must be entered here. Leave the address 0.0.0.0, if you are not using a gateway.

Configuration of the TCP/IP driver for CPX-FEC, FEC Compact and FEC Standard

Only the above-named IP addresses and the IP network mask must be specified.

Configuration of the TCP/IP driver for PS1-CP15

Only the above-named drive and the IP addresses/mask must be specified. Also, file PKT.INI (in the relevant RUNTIME.xxx directory) must be edited. The most important parameters are BitRate, Channel, RadioType and SystemId.

File PKT.INI is loaded into the IPC when the project is loaded.

Please note
Please make sure that file PKT.INI is actually loaded after a modification, because the loading procedure updates only files the length of which differes from that saved in the controller.

Configuration of the TCP/IP driver for PS1-HC02

Only the drive and the IP addresses/mask must be specified.

Please note
The Ethernet interface occupies the address range 300h ... 31Fh! No further PS1 modules may lie in this address range.

The following table provides an overview of the possible overlappings. If you wish to use a module which is not listed in the table, check the assigned address range with the aid of the hardware manual.

PS1 modules for HC0X which can be operated in the address range 300h ... 31Fh:

PS1 module	KSW	Address	KSW	Address
AS12	3	310-313	–	–
AS13	3	310-313	–	–
AS14	3	310-313	–	–
IM10/11	5	310/311	6	312/313
IM12	3	310-313	9	314-317
IM20/21	0	300-307	3	318-31F
IM51	5	310/311	6	312/313
OM10/11	5	310/311	6	312/313
OM12	3	310/313	9	314/317
OM20	9	310	–	–
OM22	5	310/311	6	312/313
OM40	5	310/311	6	312/313
OM70	9	310	–	–
OM74	5	310/311	6	312/313
OM75	5	310/311	6	312/313
TM10	5	310/311	6	312/313
IO60/61/64	0	300-303		
IO70/71/73	0	300/301		

Configuration of the TCP/IP driver for 2 network cards

The following parameters are required:

– Target disc drive

– Port and interrupt numbers for these two cards

– IP configuration for both cards

Please be careful when using driver TCPIPXXX.

– BOOTP/DHCP is not possible on the second card. The second card enables only a local network (no gateway option for routing).

– Modules such as EASY_S, EASY_R, UDP_SEND etc. automatically select the correct network card due to the IP configuration and target IP.

– For reasons of speed, we recommend that an HC20 or better CPU be used.

1.11.2 Additional CI commands

The TCP/IP driver extends the scope of the command interpreter with the following CI commands:

CI command	Brief description	Description
!26	Display version number	Display version number and driver information. This display is also shown if an unknown command is entered (e.g. !26?).
!26Axx	Display the ARP table	Display information on the Address Resolution Protocol (ARP). The TCP/IP driver contains a table with 32 IP and Ethernet addresses. If no position number (xx) is specified, a line of the table will be displayed for each access.
!26B	Displays the BOOTP/DHCP status	Display of the status for the BOOTP / DHCP procedure and the time duration before the DHCP address expires, when the IP address has been “borrowed” by a DHCP server.
!26C	Displaying the date and time	Displays the date, the time and the time zones offset in the following format: – YYYY.MM.DD HH:MM:SS.tt, followed by the UTC offset in hours.
!26Da.b.c	Supplies the IP address for the host name	Supplies the IP address for the host name for example, !26Dwww.festo.com supplies the IP address for www.festo.com.
!26D	Displays the status of the address resolution (resolver)	Possible return values: – “resolving” (busy) – “resolved” followed by the IP address found – “resolver timed out” (time exceeded) – “resolver error” followed by a fault code When searching for a host name, enter first !26D and then the host name. Then continue to enter !26D until the address resolution (resolver) is ready.
!26Hxx	Display of the handler table	Display of information on the currently defined TCP and UDP handler and its status. If no position number (xx) is specified, a line of the table will be displayed for each access.

CI command	Brief description	Description
!26I	Display of the IP configuration	Display of IP address, IP network mask and IP address of the gateway.
!26M	Display of the Ethernet MAC address	Displays the MAC address of the network card.
!26N	Display host name and domain name	Supplies the host name and the domain name.
!26Pa.b.c.d	Send ping to IP address	Sends a PING to the IP address a.b.c.d.
!26P	Display ping status	Displays the status of the last PING. Possible replies: – “Not pinging” (no Ping active) – “Pinging” (busy) – “Host is alive” (Host can be accessed) – “Timeout” (time exceeded) When pinging a host, enter first !26P and then the IP address. Then continue to enter !26P until the reply no longer “pings”.
!26R	Display of the run time duration	Displays the time duration since the controller was started. Format: – days, hours, minutes, seconds
!26SH	Displays the number of interrupts	Displays the number of interrupts generated by the network card. This information is only accessible on controllers types HC1X, HC2X with CP10, CP11, CP12 or CP14 network modules.
!26SI	Display of the IP statistics	Display of the IP statistics: – sent altogether, – received altogether, – checksum fault, – lincorrect target address.
!26SP	Display of the Ethernet statistics	Display of the Ethernet statistics: – sent altogether, – received altogether, – overrun, – package too big, cannot send.

CI command	Brief description	Description
!26ST	Display of the TCP statistics	Display of the TCP statistics: – sent altogether, – received altogether, – checksum fault, – incorrect target address, – no handler for TCP port.
!26SU	Display of the UDP statistics	Display of the UDP statistics: – sent altogether, – received altogether, – checksum fault, – incorrect target address, – no handler for UDP port, unprocessed broadcast.
!26Txx	Display the IP table	Display of the currently defined IP addresses in the table. The TCP/IP driver can manage altogether 32 IP addresses for simple access.

Please note
The CI commands are intended principally for diagnostic purposes. If there is a fault, begin with !26. If the IPC registers an ACCESS ERROR, this means that the TCP/IP driver is not contained in the project, or that the hardware configuration is not correct. Then check the sent and received packages with !26SP. If a package has not been received, this can mean that the CP10/11/12/14 interrupt is not configured correctly.

1.11.3 Modules for TCP/IP

Overview of modules

Module	Description
EASY_R	Request operand range from another IPC.
EASY_S	Send operand range to another IPC.
EASY_IO	Exchange inputs and outputs with another IPC.
IP_ALIVE	Check whether IP address is known.
IP_IP	Set/interrogate own IP address.
IP_MASK	Set/interrogate own IP network mask.
IP_TABLE	Set/interrogate IP address in table.

All modules return a fault code in FU32.
The meaning of the individual fault codes can be found in section 1.11.9.

EASY_R

Request operand range from another IPC

Input parameter	
FU32	Index for IP table
FU33	Operand type: 1 = flag, 2 = inputs, 3 = outputs, 4 = register, 5 = timer preselect, 11 = strings
FU34	Number of requested operands (max. 256)
FU35	Number of the first local operand for the requested operands
FU36	Number of the first operand in the remote controller
FU37	Flag word number for status

Return parameter	
FU32	0 if request is sent, otherwise fault

Please note
For the operand type strings (11) only one string can be transferred per access call. The operand type transferred in FU33 applies to both controllers (local and remote).

Example

```
IF  ...
THEN  CMP 12          ' Module EASY_R
       WITH   V3      " Table index 3
       WITH   V4      " Request register
       WITH   V10     " 10 Request register
       WITH   V150    " Save local as from R150
       WITH   V34     " Fetch from remote as from
                      " Register 34
       WITH   V99     " Status in flag word 99
```

If no reply arrives within approx. 50 milliseconds, a timeout fault will be entered in the status flag word. Only one request can be active for each index.

Status values	
-1	Request sent, no reply yet, no timeout
0	OK, partner has accepted the data or partner has received data
1	Fault in operand type. Partner informs that specified operand type is not supported.
2	Offset fault. Partner informs that specified operand number is not permitted (e.g. if FW20000 is requested).
4	Fault in operand number. Partner informs that the number of requested/sent operands is too large.
128	Timeout, no reply received from partner.

EASY_S

Send operand range to another IPC

Input parameter	
FU32	Index for IP table
FU33	Operand type: 1 = flag, 2 = inputs, 3 = outputs, 4 = register, 5 = timer preselect, 11 = strings
FU34	Number of operands to be sent (max. 256)
FU35	Number of the first local operand to be sent
FU36	Number of the first operand in the remote controller
FU37	Flag word number for status (-1 if confirmation is not desired)

Return parameter	
FU32	If data is sent, otherwise fault

Please note
For the operand type strings (11) only one string can be transferred per access call. The operand type transferred in FU33 applies to both controllers (local and remote).

Example

```
IF  ...
THEN  CMP 22          ' Module EASY_S
       WITH   V3      " Table index 3
       WITH   V4      " Send register
       WITH   V10     " 10 send register
       WITH   V23     " As from R23 local (up to R32
                      " local)
       WITH   V234    " Save in remote as from R234
       WITH   V98     " FW98 for status
```

If no confirmation arrives within approx. 50 milliseconds, a timeout fault will be entered in the status flag word. Only one non-confirmed request to send can be active for each index.

Status values	
-1	Package sent, no reply yet, no timeout
0	OK, partner has accepted the data or partner has received data
1	Fault in operand type. Partner informs that specified operand type is not supported.
2	Offset fault Partner informs that specified operand number is not permitted (e.g. if data are to be saved as from FW20000).
4	Fault in operand number. Partner informs that the number of requested/sent operands is too large.
128	Timeout, no reply received from partner.

EASY_IO

Exchange inputs and outputs with another IPC

Input parameter	
FU32	Index for IP table
FU33	Number of input words to be transferred
FU34	Number of the first local input word
FU35	Number of output words to be transferred
FU36	Number of the first local output word
FU37	Flag word number for status

Return parameter	
FU32	0, if data are sent, otherwise fault

This module transfers a block of output words and requests a block of input words. With this functon module you can use a different controller as remote I/O.

Please note
On the other contrroller the inputs and outputs as from number 0 are always used.

Status values	
-1	Package sent, no reply yet, no timeout
0	OK, partner has accepted the data or partner has received data
1	Fault in operand type. Partner informs that specified operand type is not supported.
2	Offset fault. Partner informs that specified operand number is not permitted (e.g. if data are to be saved as from FW20000).
4	Fault in operand number. Partner informs that the number of requested/sent operands is too large.
128	Timeout, no confirmation received from partner.

The following example copies the local outputs 10 and 11 to output words 0 and 1 on the other controller, and reads the input words 0, 1, 2 and 3 from the other controller and saves them in the local input words 20, 21, 22 and 23.

Example

```
IF  ...
THEN CFM          'EASY_IO
      WITH  V3    " Table index 3
      WITH  V4    " Request 4 input words
      WITH  V20   " Save input words as from IW20
      WITH  V2    " Transfer 2 output words
      WITH  V10   " Send output words as from OW10
      WITH  V98   " FW98 for status
```

IP_ALIVE

Check whether a specified IP address is known

Input parameter	
FU32	Index for IP table

Return parameter	
FU32	0 if successful, otherwise fault
FU33	0 = IP address unknown 1 = IP address known 2 = IP address can be accessed via gateway

Please note
If a controller is stopped, disconnected from the network, or is not accessible for some other reason, it may take up to 10 minutes until IP_ALIVE registers the IP address as unknown.

There are other possibilities for checking an IP address, e.g. testing the status value of an EasyIP package or by using the module PING.

IP_IP

Set/interrogate own IP address

Input parameter	
FU32	1 for setting the IP address 2 for interrogating the IP address
FU33	IP address
FU34	IP address
FU35	IP address
FU36	IP address

Return parameter	
FU32	0 if successful, otherwise fault code
FU33	IP address
FU34	IP address
FU35	IP address
FU36	IP address

This module is usually used only in projects with identical controllers (with the same programs). If this module and remanent operands are used, identical FST projects can be used in several controllers.

Another application is the dynamic assignment of addresses in order to allocate the IP address to a certain time point.

IP_MASK

Set/interrogate own IP network mask

Input parameter	
FU32	1 for setting the IP network mask 2 for interrogating the IP network mask
FU33	IP network mask
FU34	IP network mask
FU35	IP network mask
FU36	IP network mask

Return parameter	
FU32	0 if successful, otherwise fault code
FU33	IP network mask
FU34	IP network mask
FU35	IP network mask
FU36	IP network mask

IP_TABLE Set/interrogate IP address in table

Input parameter	
FU32	1 for setting the IP address 2 for interrogating the IP address
FU33	Index for table
FU34	IP address
FU35	IP address
FU36	IP address
FU37	IP address

Return parameter	
FU32	0 if successful, otherwise fault code
FU33	Index for table
FU34	IP address
FU35	IP address
FU36	IP address
FU37	IP address

The IP table is a list of brief addresses which are used by other modules.

1.11.4 Further modules for TCP/IP

Overview of the special modules

These modules are listed separately because their use requires special knowledge and because they are seldom used.

Module	Description
DNS_NAME	Set/interrogate host name and domain name
DNSRESOL	Supply IP address for host name
IP_DNS	Set/interrogate IP address for DNS server
IP_GATE	Set/interrogate IP address for gateway
IP_MAC	Interrogate Ethernet MAC address via network module
PING	Ping
SNTPTIME	Start time synchronization
TCP_CLOS	Close TCP connection
TCP_HAND	Activate TCP handler
TCP_OPEN	Open TCP connection
TCP_RES	Reset TCP handler
TCP_SEND	Send TCP data package
TCP_STAT	Status of TCP connection
TCP_STR	Send a string via TCP
TFTPFILE	Send/request file
UDP_FW	Send flag word range to another IPC
UDP_HAND	Activate UDP handler
UDP_SEND	Send UDP data package
UDP_STR	Send a string via UDP
IP_IP2	Set/interrogate IP address for second network card
IP_MASK2	Set/interrogate IP network mask for second network card

All modules return a fault code in FU32. The meaning of the individual fault codes can be found in section 1.11.9.

The principal procedure for sending data is as follows:

1 Module accesses in the active controller

2 Module accesses in the passive controller

3 Send data

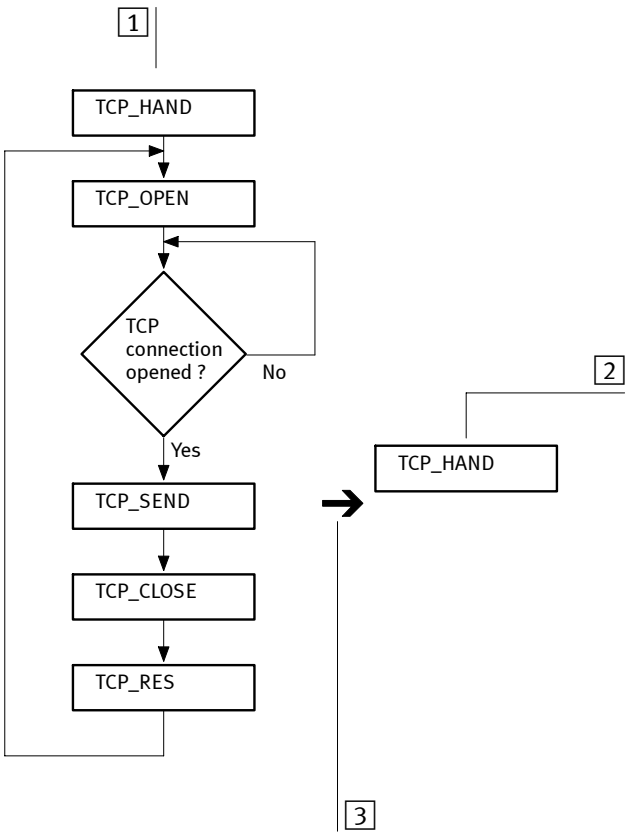


Fig. 1/4: Send data via a TCP connection

If the TCP connection is closed via TCP_CLOSE, e.g. because the connection to the passive controller has been lost, the TCP handler must be reset with TCP_RES and the TCP connection opened again with TCP_OPEN.

DNS_NAME Set/interrogate host name and domain name

Input parameter	
FU32	1 = Set host name 2 = Interrogate host name 3 = Set domain name 4 = Interrogate domain name
FU33	Set the number of the source string or interrogate the number of the target string

Return parameter	
FU32	0 if successful, otherwise fault code

TCP/IP hosts usually have names in the format:

– name.some.domain.com

In this case the host name is 'name'; the domain name is 'some.domain.com'. These names can be set with BOOTP/ DHCP or with this module. The module assumes that the FST string driver is included in the project.

DNSRESOL

Supply IP address for host name

Input parameter	
FU32	1 = Start procedure (resolver) 2 = Status interrogation
FU33	Number of string with the CI host name
FU34	Index for IP table

Return parameter	
FU32	0 if successful, otherwise fault code
FU33	Resolver status -3 = Resolver not yet started -2 = Resolver already busy, wait and try again -1 = Resolver busy 0 = Resolver ready 1 = Resolver timeout

This module tries to find the IP address for a host name. The IP address od the DNS server must be known.
The module assumes that the FST string driver is included in the project.

IP_DNS

Set/interrogate IP address for DNS server

Input parameter	
FU32	1 = Set IP address 2 = Interrogate IP address
FU33	Number of the DNS server (0, 1 or 2)
FU34	IP address
FU35	IP address
FU36	IP address
FU37	IP address

Return parameter	
FU32	0 if successful, otherwise fault code
FU33	Number of the DNS server
FU34	IP address
FU35	IP address
FU36	IP address
FU37	IP address

Up to 3 DNS servers are used for searching for the IP address of a host name. The servers are used in sequence (0, 1 then 2).

IP_GATE

Set/interrogate IP address for gateway

Input parameter	
FU32	1 = Set IP address 2 = Interrogate IP address
FU33	IP address
FU34	IP address
FU35	IP address
FU36	IP address

Return parameter	
FU32	0 if successful, otherwise fault code
FU33	IP address
FU34	IP address
FU35	IP address
FU36	IP address

IP_MAC

Interrogate Ethernet MAC address via network module

Input parameter	
FU32	String number for the address

Return parameter	
FU32	0 if successful, otherwise fault code

Please note
The FST string driver must be included in the project.

PING

Send ping or register ping status

Input parameter	
FU32	1 = Send PING 2 = Status interrogation
FU33	IP address
FU34	IP address
FU35	IP address
FU36	IP address

Return parameter	
FU32	0 if successful, otherwise fault code
FU33	Ping status -1 = Ping runs 0 = Host exists 1 = Timeout (no reply after 5 seconds) 2 = Ping not started

Send a PING and wait for reply. Can be used to check whether a host is accessible.

SNTPTIME

Start time synchronization and interrogate status

Input parameter	
FU32	0 Status interrogation 1 Start one-time synchronization 2 Wait for regular message from server
FU33	Time difference from Greenwich Mean Time in hours
FU34	IP address
FU35	IP address
FU36	IP address
FU37	IP address

Return parameter	
FU32	0 if successful, otherwise fault code
FU33	Status -1 = Busy (one-time synchronization) 0 = Time synchronized (one-time synchronization) 1 = Timeout (no reply from server) (one-time synchronization) 2 = Wait for message from server
FU34	Number of seconds since last synchronization -1 if not synchronized

The time synchronization uses standard time codes (in order to enable worldwide synchronisation). The difference from Greenwich Mean Time must therefore be known (some examples of time differences can be found in section 1.11.8).

Time server programs for Windows PCs are available as freeware or as commercially-available packages.

TCP_CLOS

Close TCP connection

Input parameter	
FU32	Index for handlers

Return parameter	
FU32	0 if successful, otherwise fault code

TCP_HAND

Installs a handler for receiving TCP data packages

Input parameter	
FU32	Local port number (permitted range: 1024 ... 65535)
FU33	Number of the first flag word for the received data

Return parameter	
FU32	0 if successful, otherwise fault code
FU33	Index for handlers

When a data package is received, additional data are written at the beginning of the flag word range (see section 1.11.7). The handler index, which is provided in FU33, must be saved and used for sending TCP data packages.

TCP_OPEN

Open TCP connection active

Input parameter	
FU32	Index for handlers
FU33	Index for IP table
FU34	Target port number

Return parameter	
FU32	0 if successful, otherwise fault code

TCP_RES

Resets closed TCP handler

Input parameter	
FU32	Index for handlers

Return parameter	
FU32	0 if successful, otherwise fault code

TCP_SEND

Send TCP data package

Input parameter	
FU32	Index for handlers
FU33	Number of bytes to be sent
FU34	Number of the first flag word with the data to be sent

Return parameter	
FU32	0 if successful, otherwise fault code

TCP_STAT

Supplies the status of the TCP connection

Input parameter	
FU32	Index for handlers

Return parameter	
FU32	0 if successful, otherwise fault code
FU33	1 if connection is set up
FU34	Detailed status
FU35	Non-confirmed bytes in the send buffer

Possible detailed status values are:

Value	Description	
0	LISTEN	Waiting for tcp_open request from remote
1	SYNSENT	tcp_open send, waiting for remote
2	SYNRCVD	tcp_open received, acknowledge send, waiting for remote
3	ESTABLISHED	Connection open, data can be transferred
4		
5	FINWAIT1	tcp_close send, waiting for remote
6	FINWAIT2	Close acknowledged
7	CLOSEWAIT	Not used
8	CLOSING	Our close acknowledged and remote close received
9	LASTACK	Close received, close send, waiting for acknowledge
10	TIMEWAIT	After closing, timer is started after that -› CLOSED
11	CLOSED	Connection closed waiting for TCP_RES

TCP_STR

Send a string via TCP

Input parameter	
FU32	Index for handlers
FU33	Number of the string

Return parameter	
FU32	0 if successful, otherwise fault code

The driver FST STRING must be installed in the project.

TFTPFILE

Send/request file

Input parameter	
FU32	1 = Send data 2 = Request data
FU33	Index for IP table
FU34	Number of the string with the name of the local file [1]
FU35	Number of the string with the name of the remote file [1]
FU36	Number of the flag word for status
[1] Drive must also be specified (e.g. B:\FileN.ame)	

Return parameter	
FU32	0 if successful, otherwise fault code

Please note
TFTPFILE requires the string driver for the file names.
There must be a waiting time of approx. 1 second between two transmissions with TFTPFILE.

Status values	
-1	Busy with file transfer
0	File transfer completed successfully
1	Timeout fault
2	Local file not found
3	Fault in reading local file

Status values	
4	Local file exists already (overwriting is not permitted, first delete file with module FDELETE)
5	Fault in writing local file
127	Unexpected message received during file transfer
128	Unknown fault message received
129	NOFILE fault message received
130	Access fault message received
131	Fault message “Disc full” received
132	Fault message “non-permitted operation” received
133	TID fault message received
134	Fault message “File exists” received
135	NOUSER fault message received
136	Fault message “option not supported” received

UDP_FW

Send flag word range to another IPC

Input parameter	
FU32	Index for IP table
FU33	Number of flag words to be sent (max. 256)
FU34	Number of the first flag word with the data to be sent
FU35	Number of the first flag word in target IPC

Return parameter	
FU32	0 if successful, otherwise fault code

This module simply sends a flag word range; no new attempt is made if the target cannot be addressed. No message will be sent if the target does not receive the data. Therefore the module EASY_S should be used instead.

UDP_HAND

Installs a handler for receiving UDP data packages

Input parameter	
FU32	Local port number (permitted range: 1024 ... 65535)
FU33	Number of the first flag word for the received data

Return parameter	
FU32	0 if successful, otherwise fault code

When a data package is received, additional data are written at the beginning of the flag word range (see section 1.11.7).

UDP_SEND

Send UDP data package

Input parameter	
FU32	Local port number
FU33	Index for IP table
FU34	Target port number
FU35	Number of bytes to be sent
FU36	Number of the first flag word with the data to be sent

Return parameter	
FU32	0 if successful, otherwise fault code

UDP_STR Send a string via UDP

Input parameter	
FU32	local port number
FU33	Index for IP table
FU34	Target port number
FU35	Number of the string

Return parameter	
FU32	0 if successful, otherwise fault code

1.11.5 Modules for handling a second network card

Modules	Description
IP_IP2	Set/interrogate IP address for second network card
IP_MASK2	Set/interrogate IP network mask for second network card

IP_IP2

Set/interrogate IP address for second network card

Input parameter	
FU32	1 = Set IP address 2 = Interrogate IP address
FU33	IP address
FU34	IP address
FU35	IP address
FU36	IP address

Return parameter	
FU32	0 if successful, otherwise fault code
FU33	IP address
FU34	IP address
FU35	IP address
FU36	IP address

IP_MASK2

Set/interrogate IP network mask for second network card

Input parameter	
FU32	1 = Set the IP network mask 2 = Interrogate the IP network mask
FU33	IP network mask
FU34	IP network mask
FU35	IP network mask
FU36	IP network mask

Return parameter	
FU32	0 if successful, otherwise fault code
FU33	IP network mask
FU34	IP network mask
FU35	IP network mask
FU36	IP network mask

1.11.6 EasyIP status values

Status values	
-1	Package sent, no reply yet, no timeout
0	OK, confirmation received, partner has accepted the data or partner has received data.
1	Fault in operand type. Partner informs that specified operand type is not supported.
2	Offset fault. Partner informs that specified operand number is not permitted (e.g. if FW20000 is requested).
4	Fault in operand number. Partner informs that the number of requested/sent operands is too large.
128	Timeout, no reply received from partner.

1.11.7 Receiving data from handler

The handlers write in the specified flag words if data are received. Information on the IP address and port number of the sender, etc. are stored at the beginning of the range (FWx is the first FW of the range):

Flag word	Contents
FWx	Number of packages received
FWx + 1	Sender IP address
FWx + 2	Sender IP address
FWx + 3	Sender IP address
FWx + 4	Sender IP address
FWx + 5	Sender port number
FWx + 6	Number of data bytes
FWx + 10	Beginning of actual data

Please note
Please note that the amount of data recived is specified in bytes, not words.

1.11.8 Time difference

Some values for the time difference	
Anchorage	-9
Buenos Aires	-3
London	0
Esslingen	+1
Cairo	+2
Moscow	+3
Jakarta	+6

1.11.9 Fault codes

Supplies return parameter FU32 ‹› 0, a fault has occurred:

Fault codes	Description
99	Non-permitted parameter
100	TCP/IP driver not loaded
101	Non-permitted IP address
102	Non-permitted table index (› 15)
103	Table position empty
104	Non-permitted port number
105	Non-permitted handler index (› 15)
106	TCP cannot be sent, no connection
107	String driver not loaded
108	Non-permitted string number (too large)
109	Fault in host or domain names
110	Fault in string modification
111	Non-permitted TCP handler
112	Unknown operand type
113	Data block too large
114	Non-permitted value for status flag word
115	Table position waits for reply
116	Save non-permitted operand number for reply
117	Sending not possible, station disconnected, collision or other fault
118	TFTP already sending or receiving

1.12 Web-Server (driver WEB_SRVR)

A Web-Server is a computer which provides data in a network. This data can be accessed with the aid of an Internet Browser. The computer, which accesses the avilable data with the aid of an Internet Browser, is called a Client. The data are usually provided in HTML format. The Browser loads the data provided, e.g. the HTML pages of the Web-Server, and displays them.

Further basic information on the theme Web-Server can be found in section 1.12.5. A brief introduction on creating HTML pages can be found in section 1.12.6.

1.12.1 Installing the driver

For use as Web-Server you must also install driver WEB_SRVR in addition to the relevant TCP/IP driver (see section 1.11).

Web-Server root directory

When installing the Web-Server driver, simply select the directory in which the data or HTML pages for the Web-Server are saved.

As soon as a project has been loaded with the TCP/IP driver and the Web-Server driver, the controller can communicate via the Ethernet interface (TCP/IP driver) and be addressed with a Browser (WEB-Server).

1.12.2 Possibilities and limits of the Web-Server

Possibilities

- HTML pages, media files and Java-Applets can be loaded into the controller. All media formats and all representation elements of the HTML format are permitted. Access can be made to these data via the Ethernet interface with the aid of any Browser.
- CI commands can be incorporated in the HTML pages in HTML code. In this way the operands of the controller can be observed or modified with the aid of a Browser.
- With the aid of JavaScripts and Java-Applets, Web pages can be made dynamic (e.g. for representing processes).
- When an HTML page is accessed, CI commands can be added to the page names as an HTTP query.
- The driver WEB_SRVR already contains standard HTML pages. The standard Homepage is called Index.htm (see also Fig. 1/5).

Limits

- Web pages **cannot** be dynamically generated with CGI or PHP programming or similar.
- The possibilities are limited by the available memory space on the controller used.
- File names must conform with the name conventions of MS-DOS (8+3 characters).

Please note
Please note that a control task in the controller has priority over communication with devices which request data from Web-Server drivers.

The standard HTML pages

The Web-Server driver already contains standard HTML pages. The standard Homepage is called Index.htm. The standard HTML pages offer read access to the operands of the controller. If the IP address of the controller is specified in the address bar of the Browser without the HTML page,

– page Main.htm will be displayed, if it exists

– the standard Homepage Index.htm will be displayed, if no HTML page with the name Main.htm exists.

Fig. 1/5: Standard Homepage Index.htm

From the standard Homepage you can open the HTML pages for the relevant operands. 16 operands are displayed there. By clicking “PageUp” and “PageDown” you can scroll through the operands.

The standard HTML pages will be automatically updated cyclically approx. every 5 seconds.

CI command as HTTP query

HTTP query

CI commands can be added to the HTML page names as an HTTP query. The HTML page named will then be accessed and at the same time the CI command will be issued.

Example

```
http://10.8.65.119/main.htm?ci:maw0=128
```

Access the page main.htm and at the same time send the CI command maw0=128 to the command interpreter (maw0 stands for **m**odify **o** utput:**w**ord **0**).

http_in_ci

An internal page, which shows only the result of a CI command, can also be accessed with an HTTP query.

Example

```
http://10.8.65.119/http_in_ci?ci:daw0
```

An HTML page is shown which shows the contents of output word 0 (daw0 stands for **d**isplay **o**utput**w**ord **0**).

Incorporating CI commands in HTML

FSTCI-Tag

The Web-Server driver supports a special HTML-Tag. This consists of the abbreviation fstci and the desired CI command.

Example

```
Display IW0: <fstci dew0>
```

When the HTML page is accessed, the text “Display IW0:” and the contents of input word 0 will be shown (dew0 stands for **d**isplay **i**input**w**ord **0**).

Link-Tag

By means of a Link-Tag, you can send CI commands by clicking a link on the command interpreter.

Example

```
<A href="main.htm?ci:maw0=255"> output word 0 =
255</A>
```

By clicking the text “Output word 0 = 255” the CI command “maw0=255” will be sent (maw0 stands for **m**odify **o**utput**w**ord **0**).

Form-Tag

With the Form-Tag you can group several CI commands in a form. You can transfer the CI commands by clicking the Send button.

Example

```
<form method="POST" action="">
Load to flag word 1:
      <input type="text" name="MFW1" value="<FSTCI dmw1>"size="6"
      maxlength="6"><P></P>
Load to flag word 2: <select name="mmw3">
      <option value="1">1</option>
      <option value="2">2</option>
      <option value="3">3</option></select><P></P>
Load 1 to flag word 3
      <INPUT type="radio" name="MMW2" value="1" checked><P></P>
Load 2 to flag word 3
      <INPUT type="radio" name="MMW2" value="2" ><P></P>
Load 3 to flag word 3
      <INPUT type="radio" name="MMW2" value="3" ><P></P>
<input type="submit" name="send" value="Send">
<input type="reset" value="reset" name="reset"> <P></P></form>
```

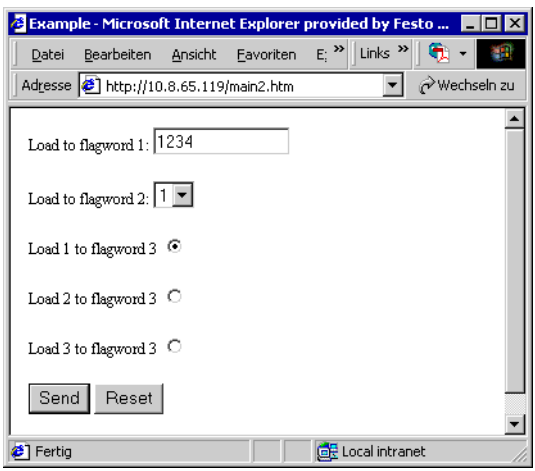


Fig. 1/6: Example of Form-Tag

1.12.3 Transfer files for the Web-Server to the controller

In order to transfer files (e.g. HTML pages) for the Web-Server into the controller with FST, proceed as follows:

1. Open the window File transfer with [Online] [File transfer].
2. Select drive “B:/” in the upper part of the list in the window “File transfer”.
3. If the Web directory exists on your controller, open the Web directory in drive B.
4. Now click on the blue arrow which points downwards, in order to load a file into the controller.
5. Select the desired file in the subsequent dialogue and confirm your selection with “Open”. The file will then be transferred to the controller.

If you have already transferred the TCP/IP driver and the Web-Server driver by loading a project, you can access the HTML pages with an Internet Browser.

1.12.4 Accessing HTML pages with an Internet-Browser

In order to be able to access HTML pages with a PC via an Internet-Browser, you must provide an Ethernet connection between the PC and the controller.

Direct connection

You will require a crossover cable for a direct connection.

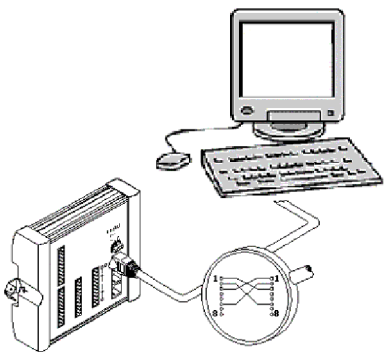


Fig. 1/7: Direct connection with crossover cable

Indirect connection

An indirect connection via a HUB can be made with a patch cable.

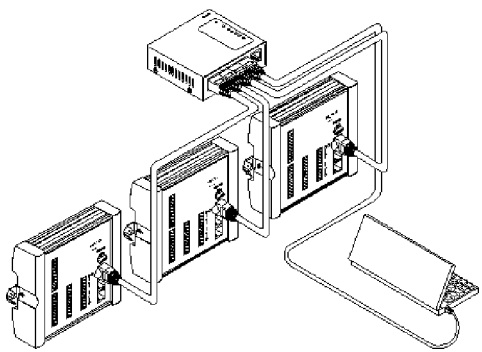


Fig. 1/8: Indirect connection with patch cable

Setting up your PC

In order to access the controller, you must set up your PC accordingly.

If you are using a direct connection without a house network:

– configure your network connection under Windows with the system controller. Match the IP address of your PC with the IP address and the IP network mask of your controller.

  Example:

IP address of the PC	192.168.0.1
IP address of the controller	192.168.0.2
Net mask address	255.255.255.0.

Please note
If you are using your controller in your house network, please consult your system administrator. He will be able to give you further assistance.

Accessing the HTML pages

You can access the HTML pages in the controller as follows:

1. Open your Internet-Browser.
2. If you are using a direct connection without the house network, make sure that use of the Proxy Server is switched off in the Browser settings.
3. Enter the IP address of the controller in the box “Address” and confirm the entry with Enter.

If you have already loaded a self-created HTML page with the name Main.htm into the controller, this will be displayed. If no Main.htm exists, the standard Homepage of the controller will be displayed (see Fig. 1/5).

1.12.5 Basic principles of the theme Web-Server

HTML	HTML stands for Hypertext Markup Language. This is the display language in which the pages of the World Wide Web (www) are produced. The basic principle of a display language is the marking (displaying) of certain elements of a document. The marking commands are known as HTML-Tag. HTML documents can be created with a simple text editor like text documents. The W3C coordinates the standard HTML (http://www.w3c.org).
URL	URL stands for Uniform Resource Locator. With this term the address, under which you can access different pages, will be named, e.g. www.festo.com.
Internet-Browser	The Internet-Browser is a program with which you can display HTML pages. MS Internet Explorer, Netscape or Opera are the most important Internet-Browsers.
HTTP	HTTP stands for Hypertext Transfer Protocol. As the name suggests, the HTML pages are transmitted with this protocol. Before each URL you should normally always write an “http://”. The present-day Internet-Browsers place this automatically before each URL.
Website	Website is the name given to a complete HTML page construct. The Website usually consists of several HTML pages. These are connected to each other by means of so-called “links” (references). The starting page is called the Homepage. After each link there is a further URL. These links can refer to one’s own addresses or to external URLs.

TCP/IP

TCP/IP is a further important protocol in “WWW”. All data is sent via the Internet with this protocol. The HTTP protocol uses the TCP/IP protocol for transmitting the HTML pages.

Further information

You can obtain further information and programs under the following addresses:

– www.w3c.org
  W3C is the World Wide Web Consortium.

– selfhtml.teamone.de
  Internet page with a lot of information on HTML.

– www.evrsoft.com
  HTML Editor (Freeware).

– www.macromedia.com/software/dreamweaver/
  Professional graphic HTML Editor.

1.12.6 Brief introduction on creating HTML pages

An HTML file can be created with a simple text editor. You can use the standard Editor under Windows (see under [Start] [Programs] [Accessories] [Editor]).

File names of the HTML pages must conform with the name conventions of MS-DOS (8+3 characters). We recommend as Homepage the name “main.htm”. The standard Homepage (see Fig. 1/5) contains a link (user homepage) to the page main.htm.

Basic information on the syntax of the HTML:

– Commands in HTML are called Tags or Elements.

– Each Tag is placed in angled brackets: ‹Command›.

– A lot of Tags start with ‹command› and end with ‹/command›. The actual information, e.g. a text, is placed between these Tags.

– Line breaks created in the Editor will not be considered.

The HTML basic structure

The following example shows the “basic structure” of an HTML page:

Example

```
<!DOCTYPE HTML PUBLIC ”-//W3C//DTD HTML 4.0 Transitional//EN”>
<HTML>
   <HEAD>
      <TITLE>Title of page</TITLE>
   </HEAD>

   <BODY>
      <!-- Contents of the page -->
   </BODY>
</HTML>
```

HTML Tags used for the HTML basic grid

HTML Tag	Description
‹!-- .. --›	Comment With "‹!DOCTYPE HTML PUBLIC"-//W3C//DTD HTML 4.0 Transitional//EN"›" you can determine the specification with which the HTML document has been created.
‹HTML›	Start of the HTML document Informs the Browser that an HTML-coded program is loaded. The opening and closing ‹html› Tags represent the beginning and the end of the HTML document.
‹HEAD›	The head is opened with this Tag. Here you will find information on the document which does not directly influence the appearance of the document.
‹TITLE›	The title of the page is specified within the head. This title is then shown in the title bar of the Browser, e.g. under www.festo.com.
‹/TITLE›	This concludes the title.
‹/HEAD›	Concludes the page head.
‹BODY›	The body of the page is opened. The actual contents of the page can be found here.
‹/BODY›	The body with the contents is now closed again.
‹/HTML›	The end of the page is now determined.

Further important HTML-Tags are shown in the following table.

Further important HTML Tags

HTML Tags	Brief description
‹BR›	Creates a line break. The flow of text and pictures on the line is concluded and started again on the next line at the left-hand edge. There is no closing Tag.
‹P›	Marks the beginning of a paragraph.
‹/P›	Marks the end of a paragraph.
‹HR›	Horizontal line
‹IMG›	Inserts a graph or a photograph into the text flow or between other pictures. Example: ‹IMG SRC=”FILENAME”› FILENAME must be replaced by an appropriate path (with or without sub-directory, relative to the directory in which the HTML file is located) and the file name of the picture.
‹A HREF=”...”› NAME‹/A›	Link (reference) to another internal or external file Examples Internal link: ‹A HREF=”start.htm”LINKNAME ‹/A› External link: ‹A HREF=”http://www.festo.com”› LINKNAME ‹/A› LINKNAME must be replaced by a text which appears in the HTML document. You can open this link by clicking on the name.

Example

Insert these lines in your Editor and save the file under “main.htm”.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
   <HEAD>
      <TITLE> Test </TITLE>
   </HEAD>
      <BODY>My first HTML page
         <HR>
         It is not difficult to create, <BR>
         HTML documents
      </BODY>
</HTML>
```

Open this “main.htm” with your Internet-Browser or with a double click on the file in the File Manager. You should see the following page displayed.

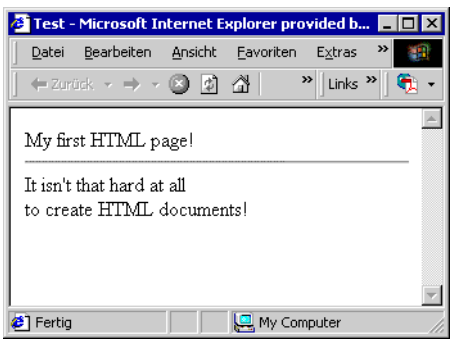


Fig. 1/9: Example of HTML page

1.13 E-mail driver (SMTP driver)

With the SMTP driver your PLC/IPC is able to send E-mails via SMTP protocol. The controller cannot, however, receive any E-mails.

1.13.1 Overview

In order to use the SMTP driver, you must install the string driver STRINGS (siehe section 1.8) in addition to the relevant TCP/IP driver (see section 1.11).

In order that E-mails can be sent, you will require a mail host which sends the E-mail to the desired receiver. As the SMTP driver does not have automatic repeating and offers only limited diagnostic possibilities, this mail host should function reliably and preferably be located in the local network. The following diagram shows the possible path of an E-mail:

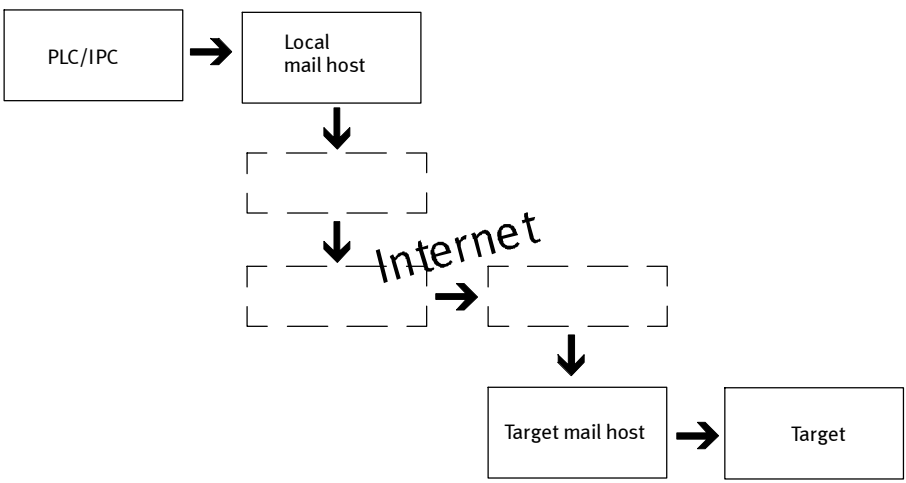


Fig. 1/10: Possible path of an E-mail

The mail host must accept E-mail messages from the controller. This usually means that the controller must be known as the “user”. For this purpose, the driver offers the possibility of setting a user name. If the local mail host is defined with a name, the controller must have a configured DNS Server.

1.13.2 Configuring and parametrizing the driver

If you wish to send E-mails in an FST IPC project, you must enter and parametrize the SMTP driver in the driver configuration.

Target drive (only HC1x and HC2x)

Specify the drive onto which the SMTP driver is to be loaded.

1.13.3 Additional CI commands

The SMTP driver extends the scope of the command interpreter with the following CI commands:

CI command	Brief description	Description
!34	Display version number	Display version number and driver information. This display is also shown if an unknown command is entered (e.g. !34?).
!34C	Display sender address and mail host	Shows the currently configured sender address and the mail host
!34D	Display host name and domain name	Displays the currently configured (in the TCP/IP driver) host name and domain name.
!34T	Send E-mail	Send an empty test mail. Example: !34Tsomeone@somewhere.com sends an E-mail to someone@somewhere.com. 1)
1) Before an E-mail can be sent, the mail host and the user name must be entered correctly.		

1.13.4 Module for the SMTP driver

Overview of modules

Module	Description
SMTPCFM	– Status interrogation (FU32=0) – Determine sender address and host name (FU32=1) – Send E-mail (FU32=2)

Module SMTPCFM returns a fault code in return parameter FU32 (see section 1.13.5).

SMTPCFM Status interrogation

Input parameter	
FU32	0 = function: Status interrogation
FU33	Number of the string for text message

Return parameter	
FU32	0 if successful, otherwise fault code
FU33	Status code, 0 if sending is completed
FU34	Fault code
FU35	Extended fault code

With the status interrogation you can ascertain whether sending is completed (see following table).

Status	Return values
Sending runs	FU32 = 0 and FU33 ‹› 0
Sending completed successfully	FU32 = 0 and FU33 = 0
Sending not completed successfully (fault)	FU32 ‹› 0

SMTPCFM

Determine sender address and mail host

Input parameter	
FU32	1 = function: Determine sender address and mail host
FU33	Number of the string with E-mail address of the sender
FU34	Number of the string with name or IP address of the mail host

Return parameter	
FU32	0 if successful, otherwise fault code
FU33	Status code, 0 if sending is completed
FU34	Fault code
FU35	Extended fault code

SMTPCFM Send E-mail

Input parameter	
FU32	2 = function: Send E-mail
FU33	Number of the string with E-mail address of the receiver
FU34	Number of the string with E-mail subject
FU35	Number of the string with which the message began
FU36	Number of the string with the message contents

Return parameter	
FU32	0 if successful, otherwise fault code
FU33	Status code, 0 if sending is completed
FU34	Fault code
FU35	Extended fault code

1.13.5 Fault codes

Supplies return parameter FU32 ‹› 0, a fault has occurred:

Fault code in FU32	Description
1	SMTP driver not in idle mode
2	Invalid string number for the sender address
3	Invalid string length for the sender address
4	Invalid string number for mail host
5	Invalid string length for mail host
6	Invalid string number for the receiver address
7	Invalid string length for the receiver address
8	Invalid string number for subject
9	Invalid string length for subject
10	Invalid string number(s) for message
99	Invalid parameters
100	SMTP driver not loaded
101	TCP/IP driver not loaded
102	STRING driver not loaded

The following fault codes can appear in FU33:

Fault code in FU33	Description
99	Invalid parameters
100	SMTP driver not loaded
101	TCPIP driver not loaded
102	STRING driver not loaded
103	Fault in deleting the mail host (DNS)
104	Timeout in deleting the mail host (DNS)
105	Timeout in connection to the mail host
106	Timeout, no (more) replies received from mail host
107	TCP connection to mail host lost
255	Mail host has registered a fault. Check the fault code in FU35.

1.13.6 Example program

This example program expects the following strings to have a fixed assignment:

String no.	Contents	Description
10	Ipc@somedomain.com	E-mail address (sender)
11	Mail.somedomain.com	Mail host (name or IP address)
12	Destination@someotherdomain.com	E-mail address (receiver)
13	Message from the PLC/IPC	Mail subject
14	Hallo,	E-mail text (string 1)
15	Here a current E-mail from the PLC/IPC	E-mail text (string 2)

In the following program extract the user name and the mail host are first determined and then an E-mail is sent.

Example

```
STEP
"" Set user name and mail host
IF    NOP
THEN  CMP 30      'E-MAIL module
       WITH V1    "0:status interrog.,1:set name and mail host,2:send mail
       WITH V10   "string#,0:message,1:user name,2:target address
       WITH V11   "string#,1:mail host,2:mail subject

IF    FU32        'Parameter 1
   =  V0
THEN  NOP

STEP
"" Send e-mail
IF    NOP
THEN  CMP 30      'E-MAIL module
       WITH V2    "0:status interrog.,1:set name and mail host,2:send mail
       WITH V12   "string#,0:message,1:user name,2:target address
       WITH V13   "string#,1:mail host,2:mail subject
       WITH V14   "string#,2:mail text
       WITH V2    "Number of strings

IF    FU32        'Parameter 1
   =  V0
THEN  NOP

STEP
"" Wait until E-mail is sent
IF    NOP
THEN  CMP 30      'E-MAIL module
       WITH V0    "0:status interrog.,1:set name and mail host,2:send mail
       WITH V20   "string#,0:message,1:user name,2:target address

IF    FU33        'Parameter 2
   =  V0
THEN  LOAD   FU34 'Parameter 3
        TO   FW34 'SMTP fault code
      LOAD   FU35 'Parameter 4
        TO   FW35 'SMTP additional fault code
```

Drivers and modules for CPX-FEC

Chapter 2

Contents

Contents of this chapter

This chapter provides an overview of the special modules for the CPX terminal. The necessary FECCPX driver will be transferred automatically when a project is loaded.

Further information

The CPX terminal supports all standard drivers and modules. Information on this can be found in chapter 1.

General information on commissioning CPX terminals as well as a detailed description of the individual parameters and data of CPX terminals can be found in the CPX system manual (P.BE-CPX-SYS-..) as well as in the manual for the relevant module (e.g. P.BE-CPX-EA-...).

Information on commissioning CPX terminals with the Front End Controller type CPX-FEC can be found in the manual for the CPX Front End Controller (P.BE-CPX-FEC-...).

2.1 Access to internal parameters and data (FECCPX)

The CPX terminal provides several internal parameters and data. The FECCPX driver enables access to all parameters and data. Parameter modifications are however not saved remanently. When the device is switched on again, the start parametrizing, which was set with the Handware Configurator, is once again valid.

2.1.1 Additional CI commands

The FECCPX driver extends the scope of the command interpreter with the following CI commands:

CI command [1]	Description
!49	Display version number and driver information. This display is also shown if an unknown command is entered (e.g. !49?).
!49DM	Display number of registered CPX modules
!49DMm[–]	Display module code and module type
!49DMEm[–] !49DMAm[–]	Display the number, size and data type of the appropriate channels
!49DMEm.n !49DMAm.n	Display value of channel named
!49DS	Display system diagnosis
!49DSm	Display diagnostic data of module m
!49DSEm.n[–] !49DSAm.n[–]	Display diagnostic data of the input channel (I) or output channel (O)

[1] m = module number, n = channel number, O = output channel, I = input channel,
– = mass display, [ ] = character in brackets can also be specified

2.1.2 Modules

Overview of modules

Modules	Description
C_ST_rd	Read CPX internal parameters and data
C_ST_wr	Write CPX internal parameters
C_STATUS	Interrogate diagnostic status
C_MD_rd	Read module diagnostic data
C_TR_rd	Read entries in diagnostic memory
C_MP_rd	Read general module parameters
C_MP_wr	Write general module parameters
C_AP_rd	Read special analogue module parameters
C_AP_wr	Write special analogue module parameters

Modules C_ST_rd and C_ST_wr offer access to all parameters and data of the CPX terminal via the so-called function numbers. The other modules offer access to certain parameters without the need for the function number to be made known.

The function numbers named in this section will assist you in finding information in other manuals. A detailed description of the individual parameters and data as well as the relevant assigned function numbers can be found in the CPX system manual (P.BE-CPX-SYS-..) as well as in the manual for the relevant module (e.g. P.BE-CPX-EA-...).

Module status

All modules return the so-called module status. The meaning of the module status is explained in the following table.

Return value of module status	
0	Module processed successfully
-1	C-bus driver missing
1	Function number outside the permitted range (› 8192)
2	Parameter value outside the permitted range
3	Function number is write-protected
4	Function number not assigned (reserved)
5	Internal fault when writing a parameter (e.g. due to overrun of an internal buffer)
6 ... 10	Reserved
11	Number of the first flag word not valid (≥ 10000)
12	Number for the first entry in the diagnostic memory not valid (≥ 40)
13	Module number not valid (≥ 48 or does not exist)
14	Channel number not valid

C_ST_rd

Read CPX internal parameters and data

Enables all parameters and data of the CPX terminal to be read after specification of the relevant function number.

Input parameter	
FU32	Function number

Return parameter	
FU32	Module status
FU33	Value

C_ST_wr

Write CPX internal parameters

Enables all parameters of the CPX terminal to be written after specification of the relevant function number.

Input parameter	
FU32	Function number
FU33	Value

Return parameter	
FU32	Module status

C_STATUS Interrogate diagnostic status

The system diagnostic data of the CPX terminal can be read with this module (FU33 ... FU35). You can then ascertain the module number for which there is diagnostic information (FU36 ... FU38). Each bit stands for the relevant module number (0 ... 47). If the bit supplies a 1-signal, there is diagnostic information for this module.

Input parameter	
FU32	–

Return parameter		***)**
FU32	Module status	–
FU33	CPX status bits	1936
FU34	First module with fault	1937
FU35	Fault message of the first module with a fault	1938
FU36	Diagnostic information exists module 0 ... 15 **)	–
FU37	Diagnostic information exists module 16 ... 31 **)	
FU38	Diagnostic information exists module 32 ... 47 **)	

*) Parameter corresponds to the function number named
**) 1 = there is diagnostic information; 0 = no diagnostic information

C_MD_rd — Read module diagnostic data (FU33 = 0)

Returns all diagnostic data of the module named.

Input parameter	
FU32	Module number (0 ... 47)
FU33	Function 0 = Read module diagnostic data

Return parameter		***)**
FU32	Module status	–
FU33	Channel number of the first faulty channel	2008 + m * 4 + 0
FU34	Module fault number	2008 + m * 4 + 1
FU35	Information 2 (reserved)	2008 + m * 4 + 2
FU36	Information 3 (reserved)	2008 + m * 4 + 3
*) Parameter corresponds to the function number named m = module number (0 ... 47)		

C_MD_rd — Read fault numbers of the channels (FU33 = 1)

Supplies the fault numbers of maximum 6 channels. The starting number of the first channel, as from which the fault numbers are to be read, will be displayed in FU34.

Detailed information on the possible fault numbers can be found in the CPX system manual as well as in the manual for the relevant module.

Input parameter	
FU32	Module number
FU33	Function 1: Read fault numbers of the channels
FU34	Number of the first channel x

Return parameter	
FU32	Module status
FU33	Fault number of channel x
FU34	Fault number of channel x + 1
FU35	Fault number of channel x + 2
FU36	Fault number of channel x + 3
FU37	Fault number of channel x + 4
FU38	Fault number of channel x + 5

C_TR_rd — Read entries in diagnostic memory

Enables the diagnostic memory to be read out. The diagnostic memory contains up to 40 diagnostic entries. A diagnostic entry consists of 10 bytes. The first five bytes contain information on the time of the fault. The last five bytes contain information on the fault.

More about the composition of the diagnostic entries can be found in the CPX system manual.

Input parameter	
FU32	Number of the first flag word in which the data are to be saved (0 ... 9999)
FU33	Number of the first entry in the diagnostic memory as from which reading is to start (0 ... 39)
FU34	Number of entries (0 ... 40) *)
*) With 0, no diagnostic entries are read, but only the information in the return parameters FU33 and FU34 is supplied.	

Return parameter		***)**
FU32	Module status	
FU33	Number of available entries	3482
FU34	Overrun and status – Bit 0: Overrun (more than 40 entries) – Bit 1: Registering inactive	3483
*) Parameter corresponds to the function number named		

C_MP_rd

Read general module parameters

Returns the general module parameters of the module named.

Input parameter	
FU32	Module number (0 ... 47)

Return parameter		***)**
FU32	Module status	–
FU33	Parameter byte 0	4828 + m * 64 + 0
FU34	Parameter byte 1	4828 + m * 64 + 1
FU35	Parameter byte 2	4828 + m * 64 + 2
FU36	Parameter byte 3	4828 + m * 64 + 3
FU37	Parameter byte 4	4828 + m * 64 + 4
FU38	Parameter byte 5	4828 + m * 64 + 5
*) Parameter corresponds to the function number named m = module number (0 ... 47)		

Special parameters of analogue modules can be read with module C_AP_rd.

C_MP_wr — Write general module parameters

Enables the general module parameters of the module named to be written.

Detailed information on the module parameters of the module you are using can be found in the manual for the relevant module. There you will also find information on possible parameter values and their presettings.

Input parameter		***)**
FU32	Module number (0 ... 47)	–
FU33	Parameter byte 0	4828 + m * 64 + 0
FU34	Parameter byte 1	4828 + m * 64 + 1
FU35	Parameter byte 2	4828 + m * 64 + 2
FU36	Parameter byte 3	4828 + m * 64 + 3
FU37	Parameter byte 4	4828 + m * 64 + 4
FU38	Parameter byte 5	4828 + m * 64 + 5
*) Parameter corresponds to the function number named m = module number (0 ... 47)		

Return parameter	
FU32	Module status

Special parameters of analogue modules can be written with module C_AP_wr.

C_AP_rd Read analogue module parameters

Input parameter	
FU32	Module number (0 ... 47)
FU33	Channel number

Return parameter		*)
FU32	Module status	–
FU33	Reserved	–
FU34	Monitoring channel 0, 1	4828 + m * 64 + 6 ... 7
FU35	Lower limit value	Depending on type **)
FU36	Upper limit value	Depending on type **)
FU37	Measured value smoothing (with input modules)	4828 + m * 64 + 9

*) Parameter corresponds to the function number named
m = module number (0 ... 47)
**) Function number depends on module type (see manual for module)

C_AP_wr Write analogue module parameters

Input parameter		*)
FU32	Module number (0 ... 47)	
FU33	Channel number	
FU34	Monitoring channel 0, 1	4828 + m * 64 + 6 ... 7
FU35	Lower limit value	Depending on type **)
FU36	Upper limit value	Depending on type **)
FU37	Measured value smoothing (with input modules)	4828 + m * 64 + 9

*) Parameter corresponds to the function number named
m = module number (0 ... 47)
**) Function number depends on module type (see manual for module)

Return parameter	
FU32	Module status

2.1.3 Fault message

If a fault occurs, the driver will enter the following fault message in the fault word of the CPX-FEC:

Fault message	Description
42,‹CPX fault no.›,‹module no.›	CPX fault number 1) and module number of the CPX module on which the fault occurred.

1) See CPX system manual P.BE-CPX-SYS-...

2.2 Communication via MODBUS/TCP

MODBUS/TCP is a transmission protocol for controlling and monitoring automation devices which are connected to a network via the Ethernet interface. It supports the master/slave communication.

With the MODBUSTCP driver, the CPX-FEC can be coupled as an Ethernet field bus slave to a MODBUS/TCP network. Up to 2 masters can build up a connection to the CPX-FEC at the same time.

2.2.1 Configuration of the MODBUSTCP driver

When a new project is created, the MODBUSTCP driver will be included automatically in the driver configurator.

When the project is loaded, it will be saved on drive A. If the driver is not required, it can be removed from the driver configurator in order to save storage space.

Flag word

The input and output data are exchanged between the CPX-FEC and the MODBUS/TCP master via a data field of up to 256 flag words. When configuring the driver, enter the number of the starting flag word.

2.2.2 Additional CI commands

The MODBUSTCP driver extends the scope of the command interpreter with the following CI commands:

CI command	Description
!35	Display version number and driver information. This display is also shown if an unknown command is entered (e.g. !35?).
!35TS	Status display of the connections (see following table)

Possible status values are:

Value	Description	
0	LISTEN	Waiting for tcp_open request from remote
1	SYNSENT	tcp_open send, waiting for remote
2	SYNRCVD	tcp_open received, acknowledge send, waiting for remote
3	ESTABLISHED	Connection open, data can be transferred
4		
5	FINWAIT1	tcp_close send, waiting for remote
6	FINWAIT2	Close acknowledged
7	CLOSEWAIT	Not used
8	CLOSING	Our close acknowledged and remote close received
9	LASTACK	Close received, close send, waiting for acknowledge
10	TIMEWAIT	After closing, timer is started after that -› CLOSED
11	CLOSED	Connection closed waiting for TCP_RES

2.2.3 Communication via MODBUS/TCP

Detailed information on using the MODBUSTCP driver can be found in the manual for the CPX-FEC (P.BE-CPX-FEC-...).

Drivers and modules for FEC Compact, FEC Standard and PS1

Chapter 3

Contents

Contents of this chapter

This chapter provides an overview of the drivers and modules for the following PLC/IPCs from Festo:

- FEC Compact
- FEC Standard
- PS1

Further information

Standard drivers and modules can be found in chapter 1. Information on further drivers and modules, which can be used only with some of the PLC/IPCs named, can be found in the following chapters:

- Further drivers and modules for FEC Compact; chapter 4
- Further drivers and modules for PS1; chapter 5

3.1 Fast counter (FECCNTR)

FEC Compact and HC0X

The last two inputs of the second group of the FEC Compact (I1.2 and I1.3) as well as the first two inputs of the HC0x (I0.0 and 0.1) can also be used as 1 or 2 independent fast counters.

Inputs, which are configured as counters, can be read at the same time as normal digital inputs.

FEC Standard

The following inputs are used with the FEC Standard:

- FC4xx - I1.6 and I1.7
- FC6xx - I3.6 and I3.7

These counters are interrupt-controlled. Once activated, they run independently of the user programs. They are not therefore influenced by the cycle time of the user programs.

3.1.1 Drivers and modules required

The FECCNTR driver must be entered in the driver configuration. The module with the same name “FECCNTR” is used for handling the fast counters. This must be imported into the project as usual, and used there either as CFM or CMP.

3.1.2 Using the module

The module can be accessed by the control programs (STL or LDR) as a function or program module. The desired function is transferred in the first input parameter (FU32).

Overview of modules

Module	Module functions	FU32
FECCNTR	Resetting the fast counter	0
	Parametrizing the fast counter	1
	Activating the fast counter	2
	Interrogating the status and current counter value	3

The fast counters can be parametrized so that when a preselect value is reached:

– an output word (OW) can be modified

– a specified program can be started

– an output word (OW) can be modified and a program started.

Each counter can be set for automatic or manual restart irrespective of the others.

FECCNTR

Resetting the fast counter

The fast counter is deactivated.

Input parameter	
FU32	0 = function: Reset
FU33	0 = counter 0 1 = counter 1

Return parameter	
FU32	None

Nothing will happen if an inactive counter is reset.

Example

```
IF      ...             " Condition
THEN    SET CMPx        " Access the fast
                        " counter module
        WITH V0         " Function: Reset
        WITH Vy         " Definition of fast counter:
                        " y=0: first counter
                        " y=1: second counter
```

FECCNTR — Parametrizing the fast counter

Definition of the counter preselect and the activities when the counter preselect value is reached. The function must be accessed in the inactive status of the fast counter (before activation or after the reset function).

Input parameter	
FU32	1 = function: parameter setting
FU33	0 = first counter; 1 = second counter
FU34	Value for counter preselect, (presetting = 0) (16-bit decimal or HEX, range 0...65535, $0...$FFFF)
FU35	Specify activities when counter preselect value is reached. 0 = No activities (presetting) 1 = Influence output word 2 = Start program 3 = Start program and influence output word
FU36	Higher-value byte: Program number Lower-value byte: Output word number All values must be specified in HEX format. Example: Program 12, output word 0: $0C00
FU37	Higher-value byte: Output mask In order to modify a bit in the output word, the same bit must be set in the mask. Lower-value byte: Value for the outputs. Set (1) or Reset (0) the output if the bit is set in the output mask. All values must be specified in HEX format. Example: Set output x.7, outputs x.0...x.6 unmodified: $8080 Set output x.7, reset outputs x.0...x.6: $FF80
FU38	0 = no automatic restart of counter 1 = automatic restart of counter

Return parameter	
FU32	None

Each pulse at the input increments the counter status and, if the preselect value is reached, the activity defined in parameter 4 will be carried out:

1. No activity
2. An output word is modified.
3. An STL or LDR program is started.
4. An output word is modified AND a program is started.

Parameter 5 defines in the higher-value byte the number of the program to be started, in the lower-value byte the number of the output word to be modified. The value must be specified in hexadecimal format. If parameter 4 ‹ › is 3, the non-used byte will simply be ignored.

Examples:

V$0600 defines output word (OW) 0 and program 6.
V$0901 defines output word (OW) 1 and program 9.

Parameter 6 is only of importance if parameter 5 has the value 1 or 3, i.e. when the preselect value is reached, an output word is modified.

The higher-value byte contains the output mask, the set bits of which specify the bits to be modified in the output word. If the same bit is also set in the lower-value byte, the corresponding output bit will be set, otherwise it will be reset. Due to this separation of the mask and the value, certain bits can be protected from modification. The value must be specified in hexadecimal format.

Examples:

V$8080 Set output x.7, outputs x.0...x.6 remain unmodified
V$FF01 Set output x.0, reset outputs x.1...x.7

In parameter 7 the user can determine whether or not there is to be an automatic restart of the counter when the preselelct value is reached.

FECCNTR

Activating the fast counter

When the module is thus accessed, the set parameters will be transferred and the fast counter will be activated.

Input parameter	
FU32	2 = function: activate counter
FU33	0 = first counter 1 = second counter

Return parameter	
FU32	None

The counter must be parametrized before this function is triggered.

If a counter is started without automatic restart and reaches the preselect value, it must be reset before it can be started again.

FECCNTR

Interrogating the status and current counter value

When the module is thus accessed, the status and the current counter value of a fast counter will be interrogated.

Input parameter	
FU32	3 = function: status interrogation
FU33	0 = counter 0 1 = counter 1

Return parameter	
FU32	Status 0 = counter active 1 = counter inactive
FU33	Current counter value
FU34	With FEC always 0 (in order to be compatible with 32-bit counters)

Please note
The status in FU32 supplies a sensible value only if the automatic restart is **not** permitted.

Example

```
IF       ...      " Condition
THEN  CMPx        " Access the fast
                  " counter module
      WITH V3     " Function: Status interrogation
      WITH Vy     " Definition of fast counter

"Check result  ...
IF    (FU32 =V0)  " Counter is still active
THEN   ...        " Other activities
      LOAD  FU33  " Save current counter value
      TO    R21   " in register 21
```

3.2 Fast outputs (FASTOUT)

The FASTOUT driver enables up to 8 outputs to be controlled at relatively high speed, irrespective of the FST project.

Overview of drivers

The FASTOUT driver controls up to 8 outputs. A “switch-on time” and a “switch-off time” can be set for each output. The driver functions only with:

– FEC Compact (20, 30, 34)

– FEC Standard (400, 440, 600, 620, 640, 660)

– HC01/HC02 controllers.

Caution
Relay outputs are not suitable as fast outputs, as they permit only a limited number of switching cycles. Use only transistor outputs for this purpose.

A complete group of 8 outputs is used. If more than one output group is available (FC600, FC620, FC640 and FC660), the last group will be used.

The bi-directional I/Os on the module will be used for HC01/HC02. These can no longer be used as inputs; and outputs 6 and 7 are not available.

The outputs which are not used are available as normal FST output words. These outputs function after the driver has been initialized with the FASTOUT module.

The driver uses the same internal resources as fast counter 1, this counter cannot therefore be used. However, fast counter 0 can be used.

3.2.1 Configuring and parametrizing the driver

If you wish to use fast outputs in an FST IPC project, you must enter and parametrize the FASTOUT driver in the driver configurator.

FST output word

Specify the FST output word which is to be used for the fast outputs.

I/O configuration

If you are using the FASTOUT driver:

– With FEC Compact and FEC Standard, use the special I/O scripts with the addition “fast outputs“ in your project instead of the normal I/O scripts.

– With HC01/HC02, the I/O script for the local I/Os must not be used or must be removed from the project.

Please remember to switch back to the normal I/O scripts if you delete the FASTOUT driver from the project.

3.2.2 FASTOUT module

FASTOUT Initializing, starting and stopping outputs

Input parameter	
FU32	0 = Initializing the driver 1 = Stop output 2 = Start output 3 = Start output, limit pulse 4 = Start output with 1 5 = Start output with 1, limit pulse 10 = Interrogate output status
FU33	Output number (0..7)
FU34	Switch-on time (x 0.5 msec)
FU35	Switch-off time (x 0.5 msec)
FU36	Number of pulses (with limited number of pulses)

Return parameter	
FU32	0 if successful 100 if the driver is not found
FU33	1 if the output is started, but not yet finished (with limited number of pulses)
FU34	Number of remaining pulses (with limited number of pulses)

If you start the output with function 2 or 3, the output will at first be switched off and then switched on again when the switch-off time has expired. If you start the output with function 4 or 5, the output will at first be switched on and then switched off again when the switch-on time has expired.

3.2.3 Additional CI commands

This driver extends the scope of the command interpreter with the following CI commands:

CI command	Brief description
!48	Display version number

3.2.4 Examples for FASTOUT

Example 1

In order to initialize the driver and to switch output 2 on for 10 ms and off for 25 ms again and again, use the following code:

```
       CFM 0                    " Fast output
         WITH     V0            " 0:Init; 1:Stop; 2:Start

...

       CFM 0                    " Fast output
         WITH     V2            " 0:Init; 1:Stop; 2:Start
         WITH     V2            " Output number (0..7)
         WITH     V20           " Switch-on time (* 0.5 ms)
         WITH     V50           " Switch-off time (* 0.5 ms)
```

Example 2

In order to initialize the driver and to switch output 3 on for 12 ms and off for 40 ms 12 times, use the following code:

```
Start STEP
 IF                NOP
 THEN  CFM 0                      " Fast output
         WITH      V0             " 0:Init; 1:Stop; 2:Start
       CFM 0                      " Fast output
         WITH      V3             " 0:Init; 1:Stop; 2:Start
         WITH      V3             " Output number (0..7)
         WITH      V40            " Switch-on time (* 0.5 ms)
         WITH      V80            " Switch-off time (* 0.5 ms)
         WITH      V12            " Pulse number

STEP wait
 IF                NOP
 THEN  CFM 0                      " Fast output
         WITH      V10            " 0:Init; 1:Stop; 2:Start
         WITH      V3             " Output number (0..7)
 IF                FU33
         =         V0
 THEN              NOP
```

3.2.5 Notes and limitations

If outputs are started simultaneously (in the same program step), the outputs do not start synchronously. The distance between the pulses is 0.5 ms. It is not possible to synchronize outputs.
The FASTOUT driver influences the accuracy of the FST timer. The deviation is up to 8 ms.

The switching frequency is limited by the minimum length of the switch-on and switch-off times (each times 0.5 ms). The maximum frequency which can be reached is 1000 Hz with an accuracy of approx. ± 2. However, at this frequency there is no longer a pulse ratio of 1:1, it corresponds more to 2:1. A well-balanced pulse ratio (52:48) will only be achieved with frequencies below 250 Hz.

Please note
Note that the pulse ratio to be aimed at depends on the type of controller used.

The table gives a few measured values:

Frequency [Hz] (nominal 1:1)	FEC FC440 Ratio	FEC FC660 Ratio
1000	60:40	100:0 (no pulse)
500	–	83:17
250	52:48	66:33
100	51:49	56:44

If the frequency is too high, the FC660 will lose pulses and the output will no longer switch. No frequencies higher than 100 Hz should be used.

Active communication between a PC (e.g. FST online display) or FED and the controller influences the reaction time of the input to the starting signal for an output.

3.3 Stepping motor driver (STEPLITE)

Please note
Can only be used with FEC Standard.

STEPLITE is a simple stepping motor driver which uses the normal inputs and outputs of the FEC Standard controller in order to control a stepping motor.

The software consists of the following parts:

– driver and installation file (STEPLITE)

– module (StepLT)

– I/O script (HCOxStep).

I/O assignment

I/Os	FEC Standard
Ix.0	Not connected
Ix.1	Not connected
Ix.2	Not connected
Ix.3	Not connected
Ix.4	Not connected
Ix.5	Negative limit switch
Ix.6	Positive limit switch
Ix.7	Reference switch
Ox.0	Not connected
Ox.1	Not connected
Ox.2	Not connected

I/Os	FEC Standard
Ox.3	Sequence
Ox.4	Direction
Ox.5	Not connected
Ox.6	Not connected
Ox.7	Not connected

The 'x' used above (as in Ix.0) stands for the word number as selected in the I/O configuration.

The first input and output bytes are always used, irrespective of whether you have selected word or byte format.

Limitations

Please note
STEPLITE works interrupt-controlled and uses a built-in hardware timer. This means that the CPU can be heavily loaded during a positioning procedure. In order to prevent the system from being "overloaded," similar drivers or I/O scripts like, e.g. the fast counter should not be used at the same time.

Please note
STEPLITE uses "normal" I/Os with "normal" optocouplers. Therefore no higher frequency than approx. 1 to 1.5 kHz can be used. Otherwise the rise time and fall time of the outputs will generate a status which will be interpreted as "always on" or "always off" depending on the switching point of the stepping motor controller. If the frequency is too high, steps will inevitably be lost.

3.3.1 Using STEPLITE in a project

STEPLITE can only be used with FEC Standard.

I/O configuration

You must first determine the I/O assignment in a project. This is carried out in the normal manner – there is nothing special to note.

Adding the driver

Select the STEPLITE driver in the list of installed drivers in the driver configuration.

Insert the StepLT module

The PLC programs exchange information with the driver via the StepLT module. Import the StepLT either as CMP or CFM.

3.3.2 StepLT module functions

The StepLT module uses up to 7 input and output parameters in order to address the driver. The desired function is transferred with FU32. The following functions are available:

Module functions	**FU32**
Interrogate revision	0
Absolute positioning	1
Relative positioning in a positive direction	2
Relative positioning in a negative direction	-2
Reference travel	3
Move in positive direction	4
Move in negative direction	-4
Set polarity	10
Absolute positioning with speed profile	11
Relative positioning in a positive direction with speed profile	12
Relative positioning in a negative direction with speed profile	-12
Setting the actual position	13
Interrogate actual position and status	14
Fast stop without ramp	20
Stopping the motor with specified delay	21

STEPLT Interrogate revision status

Input parameter	
FU32	0 = function: Interrogate revision status

Return parameter	
FU32	Main revision number of the module
FU33	Secondary revision number of the module
FU34	Year
FU35	Month
FU36	Day
FU37	Main revision number of the driver
FU38	Secondary revision number of the driver

STEPLT Set polarity

Input parameter	
FU32	10 = function: Set polarity
FU33	Limit switch (0 = normally-closed, 1 = normally open)
FU34	Reference switch (0 = normally-closed, 1 = normally open)

Return parameter	
FU32	Fault number

STEPLT Reference travel

Input parameter	
FU32	3 = function: Reference travel
FU33	Reference speed, 38...1500 Hz
FU34	Mode (1, 2, 3 or -1, -2, -3) 1 = Move in positive direction until the reference switch is reached 2 = Move in positive direction until the reference switch is reached, then move in negative direction until the reference switch is left. 3 = Move in positive direction until the reference switch is reached, then move in positive direction until the reference switch is left. -1 = Move in negative direction until the reference switch is reached. -2 = Move in negative direction until the reference switch is reached, then move in positive direction until the reference switch is left. -3 = Move in negative direction until the reference switch is reached, then move in negative direction until the reference switch is left.

Return parameter	
FU32	Fault number

STEPLT Setting the actual position

Input parameter	
FU32	13 = function: Setting the actual position
FU33	New position, lower-value word (steps)
FU34	New position, higher-value word (steps), bit 15 determines the sign (+/-)

Return parameter	
FU32	Fault number

STEPLT Interrogate actual position and status

Input parameter	
FU32	14 = function: Interrogate actual position and status

Return parameter	
FU32	Fault number
FU33	Actual position, lower-value word (steps)
FU34	Actual position, higher-value word (steps), bit 15 determines the sign (+/-)
FU35	0 = ready, inactive >0 = active (number of function just carried out)
FU36	Negative limit switch (0 = not active; 1 = active)
FU37	Positive limit switch (0 = not active; 1 = active)
FU38	Reference switch (0 = not active; 1 = active)

Positioning

Please note
Do not exceed the following maximum frequencies:
- FC400/FC440 1500 Hz
- FC600/640 1000 Hz
- FC620/660 800 Hz

In order to carry out a movement without a ramp, simply enter the values for the maximum frequency. You can reach the maximum ramp length with the following settings:

- Maximum frequency (depending on type 800 ... 1500 Hz)
- Starting frequency 38 Hz
- Acceleration ramp 1 step/s ^{2}
- Delay ramp 1 step/s ^{2}

STEPLT Absolute positioning with speed profile

Input parameter	
FU32	1 = function: Absolute positioning with speed profile
FU33	New position, lower-value word (steps)
FU34	New position, higher-value word (steps), bit 15 determines the sign (+/-)
FU35	Maximum frequency (steps / s), 38...1500 Hz [1)]
FU36	Starting frequency (steps / s), 38...1500 Hz
FU37	Acceleration ramp (steps / s^2), 1...65535
FU38	Deceleration ramp (steps / s^2), 1...65535

[1)] Observe maximum permitted frequency (FC400/FC440: 1500 Hz; FC600/640: 1000 Hz; FC620/660: 800 Hz).

Return parameter	
FU32	Fault number

STEPLT

Absolute positioning with saved speed profile

Input parameter	
FU32	11 = function: Absolute positioning with saved speed profile
FU33	New position, lower-value word (steps)
FU34	New position, higher-value word (steps), bit 15 determines the sign (+/-)

Return parameter	
FU32	Fault number

STEPLT

Relative positioning in a positive direction with speed profile

Input parameter	
FU32	2 = function: Relative positioning in a positive direction with speed profile
FU33	Distance, lower-value word (steps)
FU34	Distance, higher-value word (steps)
FU35	Maximum frequency (steps / s), 38...1500 Hz [1]
FU36	Starting frequency (steps / s), 38...1500 Hz
FU37	Acceleration ramp (steps / s^2), 1...65535
FU38	Deceleration ramp (steps / s^2), 1...65535
[1] Observe maximum permitted frequency (FC400/FC440: 1500 Hz; FC600/640: 1000 Hz; FC620/660: 800 Hz).	

Return parameter	
FU32	Fault number

STEPLT

Relative positioning in a negative direction with speed profile

Input parameter	
FU32	-2 = function: Relative positioning in a negative direction with speed profile
FU33	Distance, lower-value word (steps)
FU34	Distance, higher-value word (steps)
FU35	Maximum frequency (steps / s), 38...1500 Hz [1)]
FU36	Starting frequency (steps / s), 38...1500 Hz
FU37	Acceleration ramp (steps / s^2), 1...65535
FU38	Deceleration ramp (steps / s^2), 1...65535
[1)] Observe maximum permitted frequency (FC400/FC440: 1500 Hz; FC600/640: 1000 Hz; FC620/660: 800 Hz).	

Return parameter	
FU32	Fault number

STEPLT

Relative positioning in a positive direction with saved speed profile

Input parameter	
FU32	12 = function: Relative positioning in a positive direction with saved speed profile
FU33	Distance, lower-value word (steps)
FU34	Distance, higher-value word (steps)

Return parameter	
FU32	Fault number

STEPLT

Relative positioning in a negative direction with saved speed profile

Input parameter	
FU32	-12 = function: Relative positioning in a negative direction with saved speed profile
FU33	Distance, lower-value word (steps)
FU34	Distance, higher-value word (steps)

Return parameter	
FU32	Fault number

Manual operation

The movement is limited to 4 294 967 295 steps. This corresponds to a positioning time of:

– 33.1 days at 1500 Hz

– 3.58 years at 38 Hz

STEPLT Move in positive direction at constant speed

Input parameter	
FU32	4 = function: Move in positive direction at constant speed
FU33	Speed (steps / s)

Return parameter	
FU32	Fault number

STEPLT Move in negative direction at constant speed

Input parameter	
FU32	-4 = function: Move in negative direction at constant speed
FU33	Speed (steps / s)

Return parameter	
FU32	Fault number

STEPLT

Fast stop without deceleration ramp

Input parameter	
FU32	20 = function: Fast stop without deceleration ramp

Return parameter	
FU32	Fault number

STEPLT

Stop motor with specified deceleration ramp

Input parameter	
FU32	21 = function: Stop motor with specified deceleration ramp

Return parameter	
FU32	Fault number

3.3.3 List of fault numbers

Fault number	Description
0	No fault
1	No valid function number, incorrect entry for FU32
2	Incorrect entry for FU33
3	Incorrect entry for FU34
4	Incorrect entry for FU35
5	Incorrect entry for FU36
6	Incorrect entry for FU37
7	Incorrect entry for FU38
8	Driver not found
9	Invalid driver
10	Driver not found
11	Position outside the valid range
12	Starting speed too high
13	Arithmetic fault, distance too large
14	Limit switch active
15	No speed profile saved (functions 11, 12 or -12)

3.4 Watchdog driver (WATCHDRV)

The WATCHDRV driver serves the CPU Watchdog. The Watchdog is normally served by the BIOS. This setting is not alway sufficient if there are faulty modules or drivers in the project. We recommend that the Watchdog driver be used in the test phase of a software system of C modules in order to avoid deadlocks. In normal operation the Watchdog should not be triggered if the hardware functions correctly.

The Watchdog driver modifies the setting so that the CPU is restarted (reboot), if the PLC cycle is stopped. The driver sets the Watchdog timer to approx. 1 second. The Watchdog timer is wound up again by the driver approximately every 500 ms.

Please note that the Watchdog is not activated if “normal” FST faults are generated. For example, the Watchdog will not be triggered when fault 11 (I/O stage defective) occurs.

3.4.1 Configuring and parametrizing the driver

If you wish to use the Watchdog in an FST IPC project, you must enter and parametrize the WATCHDOG driver in the driver configuration.

Target disc drive

Specify the drive on which the WATCHDOG driver can be found or onto which it must be loaded.

With HC1X, HC2X: Flag word for Watchdog status

Specify the number of a flag word which the driver can use in order to note that the Watchdog has been triggered. If this flag word contains a value not equal to 0, FST fault 13 will be triggered.

Please note
Please note that this function is only possible on HC1X CPUs and that the DRAD driver must be configured.

3.4.2 Additional CI commands

The WATCHDOG driver extends the scope of the command interpreter with the following command:

CI command	Brief description	Description
!37	Display version number	Display version number and driver information. This display is also shown if an unknown command is entered (e.g. !37?).

3.4.3 Modules for WATCHDOG

Overview of modules

Modules	Description
PAUSE	Stops the CPU
PAUSECLI	Stops the CPU and switches the Interrupts off

PAUSE

Stops the CPU

Input parameter	
FU32	Waiting time in 15 msec ticks Value 0 means wait permanently.

Return parameter	
FU32	None

This module performs an endless loop and thereby blocks the FST system. Without a Watchdog driver this would result in a “hung up” CPU. With this module the functioning of the WATCHDOG driver can be checked.

PAUSECLI

Stops the CPU and switches the Interrupts off

Input parameter	
FU32	None

Return parameter	
FU32	None

This module performs an endless loop and thereby blocks the FST system. The Interrupts will also be switched off. Without a Watchdog driver this would result in a restart of the system. With this module the functioning of the WATCHDOG driver can be checked.

Further drivers and modules for FEC Compact

Chapter 4

Contents

Contents of this chapter

This chapter describes how an FEC Compact with 1 to 4 additional modules can be extended in order to have up to 100 I/Os.

Further information

Standard drivers and modules can be found in chapter 1. Information on further drivers and modules, which can be used only with some of the PLC/IPCs named, can be found in the following chapters:

- Drivers and modules for FEC Compact, FEC Standard and PS1; chapter 3
- Further drivers and modules for PS1; chapter 5

4.1 FEC remote I/O extension (FCMASTER/FCSLAVE)

An FEC Compact can be extended with 1 to 4 additional modules. An FEC Compact can therefore control up to 100 I/Os.

I/O extensions cannot be connected to FEC Standard controllers.

Please note
Use only controllers of type FEC-FC30 for building up extended systems. Controllers of type FEC-FC20 are not permitted in this respect.

A system can consist of several FEC Compact controllers in a master-slave arrangement. The master FEC saves the user programs as well as the driver for monitoring communication with the slave FECs. If only one extension unit is required, you can use the I/O module FEC slave without a driver.

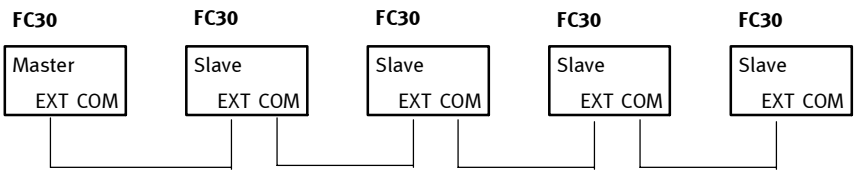


Fig. 4/1: System overview – FEC remote I/O extension

The modules must be connected together with the Festo cable type FEC-KSD4 (order number 183635). The slave FECs must not contain user programs. Only the appropriate driver for the slave function needs to be loaded.

Please note
When plannng the system, take into account the fact that I/O modifications can take up to 25 milliseconds of time, due to the nature of the serial data bus.

4.1.1 Configuration of the slave FECs

Each FEC module, which is to be used as a slave, must be configured accordingly. In the following it is assumed that you have set up a separate project FECSLAVE, which contains the data which must be loaded into the slave FECs. The project must not contain user programs. You simply have to enter the FCSLAVE driver in the driver configuration of the project and load the project into each slave FEC.

Proceed as follows:

1. Set up the project FECSLAVE or open the project FECSLAVE.
2. Open the driver configuration.
3. Enter the FCSLAVE driver.
4. Use the appropriate cable to connect your PC to the COM interface of the FEC.
5. Set the Run/Stop switch of the slave FEC to STOP.
6. Switch on the power supply for the slave FEC.
7. Load the FECSLAVE project.
   You have received a warning: “No I/O scan table, no program marked”. This is correct as only drivers will be loaded.
8. Set the Run/Stop switch of the FEC to RUN.
9. If the operating voltage is switched off then on again, the FEC will be in the slave status.
10. Repeat this procedure for each FEC slave.

4.1.2 Configuration of the master FEC

The master FEC processes the user programs. As with single FECs, you must create the I/O configuration. You must also enter the FCMASTER driver in the driver configuration of the project.

Supplementing the I/O configuration

The FEC slaves must be entered in the I/O configuration of the master project. Two I/O scripts are always available for operating the slave FECs. Select the I/O script best suited to your application.

I/O script	Description
External FEC / FEC slave	If you select this script, the master will check during the system initialization to see if all expected (configured) slaves reply. If one of the configured slaves does not reply, system error 11 will be generated. If one of the configured slaves does not reply during running time, system error 11 will also be generated.
FEC slave without error 11	This script reacts during system initialization exactly like the first one. At running time, however, error 11 is not triggered as a result of communication faults between the master and the slaves.

Each FEC slave must be entered in the I/O configuration of the master project. You must enter the following data:

– Switch position: Enter 0 for the first slave, 1 for the second slave, etc.

– IW: Enter here the first input word for the slave. Each FEC occupies 2 input words; the ranges of the individual slaves must not overlap.

– OW: Enter here the first output word for the slave. Each FEC occupies 2 output words; the ranges of the individual slaves must not overlap.

Please note
Do not forget that the FEC master must also be entered in the I/O configuration.

Optional I/O entry

If necessary, the faults can be counted on the extended bus. In order to do this enter the script “FEC slaves error counter” in the I/O configuration. Enter the number of an unused input word under “IW”. The number of bus faults will be entered here.

4.1.3 Run time behaviour

Run time behaviour of the master

When the system is switched on, the master initializes the slave FECs. If a deviation of the actual existing slaves from the configured slaves is thereby discovered (e.g. configured slaves do not reply), an “I/O module defective” error will be generated and system error 11 will be saved.

If the I/O script “External FEC / FEC slave” is contained, frequent faults in communication with a slave during running time will lead to system error 11, and the master will stop updating the slave.

The system can be reinitialized if the setting of the Run/Stop switch is changed or if the operating voltage is switched off then on again.

Run time behaviour of the slaves

When the slave is switched on the Run/Stop switch is checked. If this switch is in the Stop position, the slave driver will not be started and the master will not have access to the slave. If the switch is in the Run position, the slave driver will be started.

Explanation of the RUN LED of the slaves

RUN LED	**Description**
red	Communication fault
orange, flashing	No communication with the master
green, flashing	Slave is initialized by the master
green	Communication normal

4.1.4 Diagnostic module (REMDIAG)

The module library of the FST provides a module with which the status of the slave FECs can be interrogated from within the control programs. If you wish to use this function, you must import the module REMDIAG (slave FEC error counter) into the project of the FEC master.

Using the REMDIAG module

The REMDIAG module can be accessed in the STL and LDR programs with the aid of the following parameters:

REMDIAG Interrogate error counter

Input parameter	
FU32	1 = function: Interrogate error counter

Return parameter	
FU32	Number of short-time errors 0 if communication OK ‹› 0 if error is in the last cycles
FU33	Total number of errors, changes from 65535 to 0.

REMDIAG — Reset total error counter

Input parameter	
FU32	2 = function: Reset total error counter

Return parameter
None

REMDIAG — Set total error counter

Input parameter	
FU32	3 = function: Set total error counter
FU33	Value for total error counter 0 = Delete function

Return parameter
None

Further drivers and modules for PS1

Chapter 5

Contents

Contents of this chapter	This chapter describes special drivers and modules for PS1.
Further information	Standard drivers and standard modules are described in chapter 1. Information on further drivers und modules for PS1 can be found in chapter 3 “Drivers and modules for FEC Compact, FEC Standard and PS1”.

5.1 Module for Encoder module IM2... (module IM2X)

With module IM2X it is possible to use Encoder module IM2X for FST IPC projects. Module IM2X must be imported into every project, where it is to be used, as a function module (e.g. as B00). This is accomplished with the aiid of the Import function, e.g. via the menu command [Program] [Import].

The desired function is transferred with FU33. The following functions are available:

Module functions	FU33
Initialize and reset counter	-1
Read counter states	0
Load counter 1 with value	1
Load counter 2 with value	2
Load counter 3 with value	3

IM2X Initialize and reset counter (FU33 = -1)

Input parameter	
FU32	KSW setting (1 or 2)
FU33	-1 = function: Initialize and reset counter
FU34	Mode (see following table)

Return parameter	
FU32	Current status of counter 1
FU33	Current status of counter 2
FU34	Current status of counter 3

Modes of function module IM2X

The numbers 1, 2 and 4 in the following table stand for 1, 2 or 4-edge evaluation of the signals for the channel specified. If only one number is specified for two channels, this means that these are cascaded.

Mode (hex)	Channel 1	Channel 2	Channel 3
$1124	1	1	1
$1125	2	1	1
$112D	2	2	1
$116D	2	2	2
$1126	4	1	1
$112E	4	2	1
$116E	4	2	2
$1136	4	4	1
$1176	4	4	2
$11B6	4	4	4
$1320	1 *)		1
$1360	1 *)		2
$13A0	1 *)		4
$1321	2 *)		1
$1361	2 *)		2
$13A1	2 *)		4
$1322	4 *)		1
$1362	4 *)		2
$13A2	4 *)		4
*) Channels 1 + 2 cascaded for 32-bit resolution			

Further setting possibilities can be found in the IM2x hardware manual. However, it is not possible to use Interrupts with this function module.

IM2X — Read counter states (FU33 = 0)

Input parameter	
FU32	KSW setting (1 or 2)
FU33	0 = function: Read counter states

Return parameter	
FU32	Current status of counter 1
FU33	Current status of counter 2
FU34	Current status of counter 3

IM2X — Load counter 1 with value (FU33 = 1)

Input parameter	
FU32	KSW setting (1 or 2)
FU33	1 = function: Load counter 1 with value
FU34	New specified counter value

Return parameter	
FU32	Current status of counter 1
FU33	Current status of counter 2
FU34	Current status of counter 3

IM2X — Load counter 2 with value (FU33 = 2)

Input parameter	
FU32	KSW setting (1 or 2)
FU33	2 = function: Load counter 2 with value
FU34	New specified counter value

Return parameter	
FU32	Current status of counter 1
FU33	Current status of counter 2
FU34	Current status of counter 3

IM2X — Load counter 3 with value (FU33 = 3)

Input parameter	
FU32	KSW setting (1 or 2)
FU33	3 = function: Load counter 3 with value
FU34	New specified counter value

Return parameter	
FU32	Current status of counter 1
FU33	Current status of counter 2
FU34	Current status of counter 3

5.2 AS-Interface (ASI driver)

The AS-Interface connects sensors and actuators speedily and at low cost to the IPC. Up to 4 CP96 AS-Interface modules each with up to 31 slaves can then be used.

The inputs and outputs of the slaves are mapped onto the local function units of the FST IPC which have been configured by the user.

5.2.1 Selecting and parametrizing the driver

If you wish to use AS-Interface in an FST IPC project, you must enter and parametrize the ASI driver in the driver configurator.

Target disc drive

Specify the drive on which the AS-Interface driver ASIDRV.EXE can be found or onto which it must be loaded.

5.2.2 The AS-Interface configurator

Start the configurator in the FST IPC via the menu item [Extras] [AS-Interface configuration].

Please note
In order that you can use the online features of the configuration tool, the ASI driver must be active on the IPC. It will suffice if you load the project, for which you have configured the ASI driver, into the controller without programs.

5.2.3 Selecting the desired master

- Activate the menu command [Edit] [Modul selection / IO-assignment]. An entry window will open in which one of 4 masters can be selected.

The number of the master corresponds to the address set on the CP96 module. The basis address for the I/O image, onto which the AS-Interface slave I/Os are mapped, is also defined in this selection window. It is absolutely necessary that this addresss be specified in order that the master modules used can be correctly configured. A CP96 module (AS-Interface master) always occupies 8 I/O words in the local I/O range of the FST IPC.

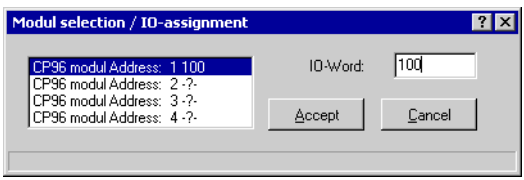


Fig. 5/1: Selecting the master module and assigning the I/O address

5.2.4 Configuration of the individual slaves

The central function of the configurator tool is the planning of the individual AS-Interface bus slaves. Planning is to be understood here as defining the ID and IO codes of a slave. Planning can be carried out here offline, i.e. without connection to an IPC. If there are several AS-Interface masters in an IPC system, this procedure must be carried out individually for each master. Selection of the desired masters is carried out via the menu command [Edit] › Module selection / IO-assignment.

In the initialization phase during operation, the configuration data of each master is compared with the data available to the driver as the nominal list. If the data is not the same, the driver will carry out a new configuration of the relevant master.

Due to this procedure, a relatively long delay in restarting a project (due to new programming of the master lists) can be avoided. This delay occurs only when a project is started the first time, when the configuration is modified and when a master is replaced.

When the menu command Edit › Configuration is selected, the planned slaves from 1..31 with I/O address, ID, IO code and parameter will be displayed. Any slave can be selected with the arrow keys or with the mouse for further processing.

Editing

After a double click on a slave or after selection of Edit in the local menu, an entry mask will be opened, in which the ID and IO codes as well as parameters of the selected slave can be edited. ID and IO codes can be found in the documentation for the relevant slaves. The specification of a parameter is optional. This parameter (0..F) is transmitted to the slave when the master is restarted. With the aid of such parameters, certain features can be influenced with appropriately designed slaves (e.g. sensitiveness of an ultrasonic sensor).

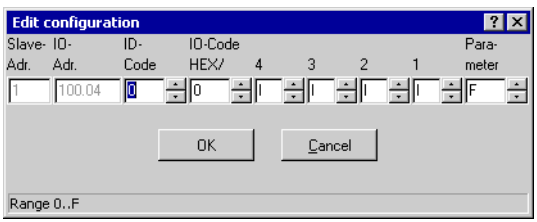


Fig. 5/2: Entering the slave in the configuration

Delete

The selected slave can be deleted from the nominal list with the DEL key.

Nominal-actual comparison

When the nominal-actual comparison is selected in the local menu, the slaves registered in the nominal list will be compared with the slaves on the AS-Interface bus. Any deviations will be displayed in a box which can be scrolled. If the nominal list is empty (e.g. at the start of a new project), the slaves available on the bus will be taken over as the nominal list. The list thus taken over can of course be processed further.

Prerequisites for the nominal-actual comparison There must be a connection to the IPC. The ASI driver must be installed. The IPC run time main program must be active.

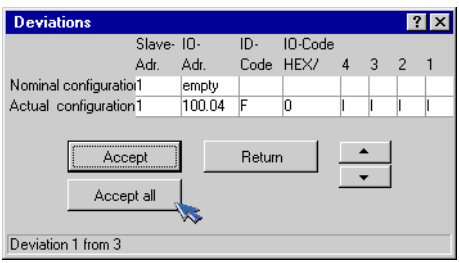


Fig. 5/3: Result of the nominal-actual comparison

Printing the configuration

In order to print the current configuration, select the menu command [File] [Print] or click on the icon “Print”.

Programming the address of individual slaves

Prerequisites: There must be a connection to the IPC. The ASI driver must be installed. The IPC run time main program must be active.

In the “original status” all AS-Interface slaves have address 0. Due to the address programming an AS-Interface slave can be assigned with a new address. This can also apply to slaves which already have an address other than 0. Each AS-Interface slave must have a clear address (per master). In order to modify an address, activate the menu command [Edit] [Slave addresses]. In this list you can now assign a new address to a slave with “Drag and drop”.

Online manipulation of individual slaves

Prerequisites: There must be a connection to the IPC. The ASI driver must be installed. The IPC run time main program must be active.

In the AS-Interface online display (Menu item [Online] [Display]) the slaves registered on the bus will be displayed with their I/Os. The display is always dynamic, i.e. the inputs will be updated automatically.

Manipulation of the slave outputs

Position the selection marking on the slave, the inputs of which are to be manipulated. The outputs can be set and reset with a double click, via the local menus (right-hand mouse key) or via the function keys F5...F8.

5.2.5 Additional CI commands

The ASI driver extends the scope of the command interpreter with the following CI commands:

CI command	Description
!7DLES	Display list of the current slaves
!7DSxx	Display slave (IO, ID + inputs) xx = 1..1F
!7DOUF	Display control flags
!7DHARD	Display hard/soft reaction (0 = soft, 1 = hard)
!7MMA	Modify master address (= 1..4) (valid for all further commands).
!7MSA	Modify slave address (= old slave address, new slave address)
!7MPM	Modify project mode (= 0 or = 1)
!7MA	Modify output (= slave address, output word)
!7MSP	Modify slave parameter (= slave address, parameter)
!7MHARD	Modify hard/soft reaction (= 0 or =1)
!7MERR	Modify error status (= 0) (restart after run time error)

All number values specified with the Modify commands must be entered as a HEX value. Upper/lower case letters is optional. The commands will be used mainly by the AS-Interface configurator. Manual use has no sense. They are only documented here for completeness.

5.2.6 Error numbers

Error number	Meaning of error
700	No configuration data available for the ASI driver.
701	Master 1 not available
702	Master 2 not available
703	Master 3 not available
704	Master 4 not available
711	Failure of a slave with master 1
712	Failure of a slave with master 2
713	Failure of a slave with master 3
714	Failure of a slave with master 4

In its cyclic part, the driver monitors the Config. OK flags of all configured masters. If “hard” is set as a error reaction (presetting), error number 710 + master address (1..4) will be entered in the IPC error word in the event of a slave failure. As a result, processing of the program will be stopped or a error handling program entered in the IPC configuration will be started. If this procedure is not desired, the hard error reaction can be switched off with the aid of module ASI_MODE.

5.2.7 Modules for AS-interface

Overview of modules

Module	Description
ASI_Mode	Set the reaction of the ASI driver to configuration faults
ASI_Stat	Interrogate the flags of the sequence control level (OUF)
ASI_Para	Transmit a parameter to an AS-Interface slave during run time
ASI_Res	Restart the cyclic update

ASI_MODE

Set the reaction of the ASI driver to configuration faults

Input parameter	
FU32	0 = soft reaction › 0 = hard reaction

Return parameter	
FU32	-1 = OK 0 = Driver not loaded

ASI_STAT

Interrogate the flags of the sequence control level (OUF)

Input parameter	
FU32	Master address (1..4)

Return parameter	
FU32	-1 = OK 1 = Master address not valid 10 = Master not available
FU33	Status flags

The meaning of the individual bits can be found in the documentation for the AS-Interface master. If the hard reaction is switched off with module ASI_Mode, the Config. OK flag of the relevant master can be interrogated with module ASI_Stat and a error reaction can be triggered.

ASI_PARA

Transmit a parameter to an AS-Interface slave during run time

Input parameter	
FU32	Master address (1..4)
FU33	Slave address (1..31)
FU34	Parameter value (0..$0F)

Return parameter	
FU32	-1 = OK 1 = Master address not valid 2 = Slave address not valid 3 = Parameter value not valid 10 = Master not available 11 = Master timeout
FU33	Error code Overview of possible error codes: HFUN_OK $00 Processing successful HFUN_NOK $01 Processing not successful HFUN_SNA $02 Slave is not in LAS HFUN_MOFF $0C Master is offline MAS-IBUSY $FD Master not ready HOSTTOUT $FE Master timeout

If the slave is not available, error $02 will be returned (Slave not in LAS).

ASI_RES

Restart the cyclic update

Restart the cyclic update (including error monitoring) after the occurrence of a run time error (slave failed).

Input parameter
None

Return parameter	
FU32	-1 = OK 0 = Driver not loaded

5.3 Festo field bus master (FESTOBUS)

The field bus enables you to couple further remote I/O modules (process-close units) to your IPC-PS1 to form a system network.

Communication between the modules and the IPC is controlled by field bus module PS1-CP61. This possesses a field bus interface to which you can connect up to 31 slaves, if using repeater technology up to 99 slaves.

Special additional FST operands are available for the cyclic inputs and outputs of the slaves. States can be interrogated and acyclic commands can be carried out by means of modules.

5.3.1 Setting the Festo field bus parameters

If you wish to use the Festo field bus in a project, you must enter and parametrize the FESTOBUS driver in the driver configurator.

Target disc drive

Specify the drive on which the Festo field bus driver IPCFB22.EXE can be found or onto which it must be loaded.

CP61 switch position

You must enter here the switch position which you have set on the rear of your field bus module (CP61).

Interrupt number

Enter the number of the Interrupt which the field bus module may use for communication with the CPU. Check which Interrupts have already been assigned to other modules.

Field bus baud rate

Here you can specify the baud rate of the data transmission on the field bus. Permitted values are 375, 187.5, 62.5 and 31.25 kBaud.

Highest station address

Enter here the highest station address (slave number) that you are using on the Festo field bus. Permitted are values in the range 1...99.

Soft error behavior

Enter here whether you wish to use a “soft” error behavior. Permitted are Yes or No.

For the reaction to configuration discrepancies during start-up, the programmer can select one of the following alternatives:

– Presetting is the so-called hard error behavior. If there is a fault during start-up, neither program 0 nor the error program will be started.

– Alternatively, a soft error behavior can be defined. If there are configuration discrepancies the same procedure should be adopted as with other faults, i.e. the error word should be set and Project Stop or the error program processed.

The desired error behavior can be specified in the configuration dialogue for the field bus (see above).

Use Nominal Configuration

Enter here whether you wish to use the nominal configuration. Permitted are Yes or No.

Specifying the slave list

Alternatively the slave list can be specified to the FB master in the form of the nominal configuration created by the programmer. If there is no configured slave during start-up, the usual handling procedure for transmission faults will be adopted. The slave which was missing at the beginning can operate on the field bus later without reinitialization.

Please note that only those slaves are entered which are also used in one of the loaded programs. It is of no importance here whether or not the program is ever processed.

The individual slaves must also be used with the complete scope of their I/Os, otherwise the slave will not be addressed.

If the user carries out reinitialization via CFM48,1 or CI, the actual configuration will, as usual, be used.

5.3.2 The Festo field bus configurator

The field bus configurator is the planning and control instrument for creating programs. With this you can determine the (NOMINAL) configuration of the field bus.

You can then:

– print out the configuration data entered and connect the slaves according to the list

– carry out a comparison between NOMINAL and ACTUAL configurations, in order to eliminate connection faults.

You should create the nominal configuration of the field bus before entering the assignment list (and the programs), in order to assign symbolic designators to the operands which thereby result.

Please note: During the syntax test of the programs, a check is made to see if the corresponding entry has been made in the field bus configuration for the field bus operands entered in the programs.

The field bus I/O modules are referred to in the following as slaves.

Accessing the field bus configurator

The Festo field bus configurator is configured as an external tool in the FST 4 and can be activated in the menu “Extras”.

Updating the type

If you have not yet created a field bus configuration in the project, you must first select [Edit] [Update type] from the menu. In a window you will see the types available from Festo. Here you can select the types used in your project, e.g. by double-clicking with the mouse.

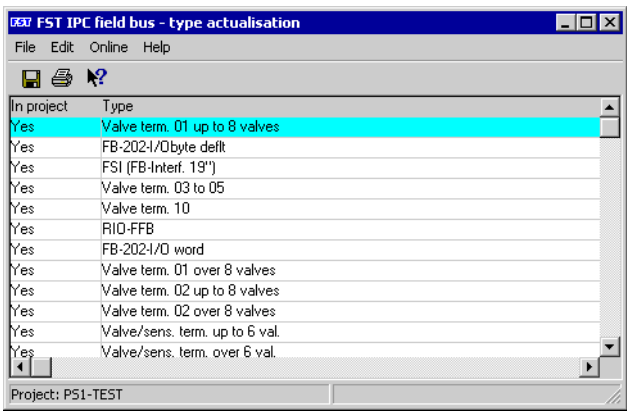


Fig. 5/4: Determining the field bus slave types

Please note
If you remove a type from the project-specific type file, for which there is already an entry in the configuration, the designator “Unknown type” will appear in the configurator instead of the type removed. Please delete such entries.

Editing the configuration

When you have selected the type for your current project, you can enter the slaves. In order to do this select the menu command [Edit] [Configuration].

If you select [Insert] from the local menu, the following window will appear in which you can configure the slaves:

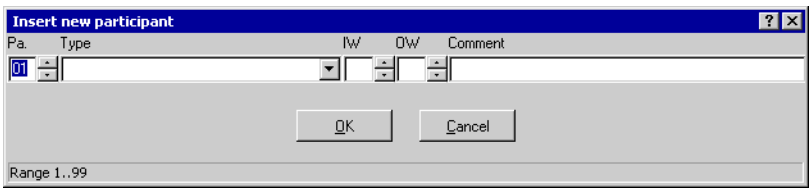


Fig. 5/5: Configuring the field bus slaves

- Enter the slave number and the slave which you wish to connect. Please refer to the following table for the permitted entries:

Abbreviation	Meaning	Permitted entry
Pa.	Slave number	- 1 to 99
Type	Type of slave	The field bus slaves saved in the project type file will be offered for selection in a window.
IW	Number of input units	Specification of the input units is only possible with those types on which the number of inputs can vary (1 to 16).
OW	Number of output units	Specification of the output units is only possible with those types on which the number of outputs can vary (1 to 16).

- Conclude entries by pressing the OK key.

The field bus slave is inserted in the configuration file in ascending sequence corresponding to the slave numbers. A display in the message bar shows the time in which the controller sends I/O information to the field bus peripherals.

Modify

Mark the field bus slave which you wish to modify and activate the function “Modify” by double-clicking or with the edit entry in the local menu. As with insert, a window appears as shown in Fig. 2. The entries can now be overwritten.

Delete

You can delete the field bus slave by selecting “Delete” in the local menu.

Checking functions

In order to guarantee faultless operation on the field bus, various checks are carried out on the one hand by the run time system of the IPC; on the other hand the field bus configurator also offers a checking possibility.

When a project is started, the actual configuration is registered by the run time system and compared with the nominal configuration. If the nominal configuration is not part of the actual configuration, a fault will be registered.

The nominal-actual comparison function supports the check of the configuration data from the PC. In order that the function can be carried out, your PC must be correctly connected to the controller.

Nominal-actual comparison

If you select the command [Nominal-actual comparison] in the context menu, the actual configuration will be downloaded from the IPC and compared with the nominal configuration created on the PC. If the configurations are identical, this will be indicated in the message bar. If there are discrepancies in the comparison, the following window will appear when this function is activated.

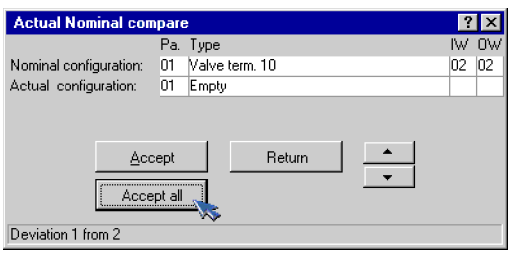


Fig. 5/6: Nominal-actual comparison

You will see the number of deviations between the nominal and the actual configurations. If there is no NOMINAL configuration file in the project directory, the downloaded ACTUAL configuration will be saved as the NOMINAL configuration. However, the file contains no comments. These specifications must therefore be added.

This function can be used when the configuration file is first created. The prerequisite is, however, that the field bus is available with all slaves.

Printing the field bus configuration

In order to print the current configuration, select the menu command [File] [Print] or click on the icon “Print”.

5.3.3 Programming field bus operands

Field bus inputs and outputs can be addressed in LDR or STL programs with the following syntax, whereby: t = slave number, w = word number, b = bit number:

– Inputs (as bit): It.w.b

– Outputs (as bit): Ot.w.b

– Input words: IWt.w

– Output words: OWt.w

The word number is assigned as from 0 for each slave separately for inputs and outputs. If, for example, you use the field bus module I/O extension 405 with 3 input cards and 2 output cards (1 word each) as slave 5, the following operands will arise:

– I5.0.0 to I5.2.15 and IW5.0 to IW5.2, or O5.0.0 to O5.1.15 and OW5.0 to OW5.1

Most field bus modules occupy only one word. That is then word 0. In the printout of the field bus configuration you will find an addressing aid for every slave.

5.3.4 Modules

Overview of modules

Modules	Description
F40	Check the configuration of a field bus slave
F41	Read the parameter field of an “intelligent” field bus slave
F42	Write the parameter field of an “intelligent” field bus slave
F43	Delete the cyclic outputs of all field bus slaves
F44	Status interrogation of a slave
F47	– FU32=1: Set the fault treatment and reaction to an acyclic command – FU32=2: Interrogate the slave status
F48	Handling the total configuration

F40 Check the configuration of a field bus slave

Input parameter	
FU32	Field bus slave number, 1...99

Return parameter	
FU32	-1 = nominal and actual data are identical 0 = nominal and actual data are different
FU33	Slave actual type
FU34	Slave: number of inputs in bytes
FU35	Slave: number of outputs in bytes

When the slave list is specified, nominal and actual data are always identical by definition of this option.

F41 Read the parameter field of an “intelligent” field bus slave

Input parameter	
FU32	Field bus slave number, 1...99
FU33	Address of the slave word, 0...255

Return parameter	
FU32	-1 = command processed successfully 0 = fault, field bus slave does not reply
FU33	Slave status; the meaning of the slave status is described in F44.
FU34	Slave word of the address
FU35	Slave word of the address + 1
FU36	Slave word of the address + 2
FU37	Slave word of the address + 3

F42 Write the parameter field of an “intelligent” field bus slave

Input parameter	
FU32	Field bus slave number, 1...99
FU33	Number of word parameters, 1...4
FU34	Address of the slave word, 0...255
FU35	Slave word for the address
FU36	Slave word for the address + 1
FU37	Slave word for the address + 2
FU38	Slave word for the address + 3

Return parameter	
FU32	-1 = command processed successfully 0 = fault, field bus slave does not reply
FU33	Slave status; the meaning of the slave status is described in F44.

F43 Delete the cyclic outputs of all field bus slaves

Input parameter
None

Return parameter
None

F44 Status interrogation of a slave

Input parameter	
FU32	‹›0 = number of the slave 0 = delete all fault entries

Return parameter	
FU32	Status of the slave Bit 7 = 1: ASCII data are available Bit 6...1: fault bits for further information on slave faults (see manual for the field bus slave) Bit 0 = 0: acyclic slave Bit 0 = 1: cyclic slave
FU33	Number of short time faults
FU34	Number of long time faults
FU35	Faults altogether
FU36	Slave address for last fault

The counter for short time faults will be incremented if there is a fault in cyclic communication (no reply, checksum fault, etc.). The maximum value is 255. The counter is reset as soon as cyclic communication is successful again.

You can check communication by accessing F44 and by interrogating the number of short time faults (FU33 › value).

The counter for long time faults is incremented if a communication fault occurs (number of short time faults ‹ › 0) and communication is restored again.

The total fault counter is incremented if cyclic communication faults occur. (no reply, checksum fault, etc.)

F47 Set programs for fault treatment (FU32 = 1)

Input parameter	
FU32	1 = function: Assign error program
FU33	Bit 0 = 1, then FU34 has the program number Bit 1 = 1, then FU35 has the program number Bit 2 = 1, then FU36 has the program number
FU34	Program number with transmission faults
FU35	Program number with slave faults
FU36	Program number for ASCII data with field bus slave

Return parameter
None

All programs must always be specified. If a non-existing or invalid program is specified, or if the bit in the mask is deleted, the program for the event treatment will be deactivated.

Example

```
THEN  CFM47
        WITH V1
        WITH V3
        WITH V3
        WITH V17
        WITH V0
```

If a transmision fault occurs (e.g. cable break), program 3 will be set. If a slave fault occurs (e.g. short circuit at a slave output), program 17 will be activated. If there is a reply to an acyclic command (e.g. the slave holds ASCII data ready), no program will be started.

The activated programs come for processing in the usual sequence (i.e. according to ascending program number). When treatment is finished, the programs should deactivate themselves so that they can be accessed again.

F47 Interrogate slave status (FU32 = 2)

Input parameter	
FU32	2 = function: Status interrogation of the slave with the last fault

Return parameter	
FU32	Always 0
FU33	Status of the slave
FU34	Short time fault value
FU35	Long time fault word
FU36	Total fault number
FU37	Field bus address of the slave with the last fault

F48 Treatment of the total configuration

Input parameter	
FU32	1 = registering the configuration 2 = status interrogation after registering the configuration 3 = specify the slave list

Return parameter	
FU32	If FU32 = 1: -1 = configuration registering runs If FU32 = 2: 1 = configuration registering runs 2 = actual configuration = nominal configuration 3 = actual configuration ‹› nominal configuration 4 = no configuration available If FU32 = 3: none

All field bus outputs will be deleted before the configuration is registered. None of the field bus inputs will be updated for several seconds.

The use of the function with FU = 2 is not sensible according to the specifications in the slave list. When the slave list is specified, nominal and actual data are always identical by definition of this option.

Example

```
STEP 1
THEN   CFM 48
         WITH  V1     "Configuration registering

STEP 2
THEN   CFM 48
         WITH  V2      "Status interrogation
IF     FU32
       <>V1
THEN   NOP

STEP 3
IF     FU32
       <>V2
THEN   ...
```

5.3.5 Additional CI commands

The Festo field bus has driver number 0. For historical reasons "$" can also be entered instead of "!0".

CI command	Description
!0FA	Assign nominal configuration
!0FC	Display of the number of slaves
!0FI	Reinitializing (registering the actual configuration) reply "=1" if successful, otherwise "=0"
!0FN	Cyclic display of the slaves The definition of each slave is displayed one at a time cyclically. Slave numbers, type numbers, the number of input bytes and the number of output bytes are output.

CI command	Description
!0FR	Start/continue the transmission of cyclic data
!0FS	Stop the transmission of cyclic data

5.3.6 Error numbers

Error number	Description
14	Critical driver faults Project cannot be started even with “soft” error behavior. This fault occurs if the field bus driver cannot be found or if faults occur in starting the firmware of the CP61.
60	Actual configuration is not a major part of the nominal configuration. Correct the field bus configuration or connect missing slaves.

5.3.7 CP61 LED fault codes

LED	Description
LED on	No firmware loaded into the CP61
LED out	Cyclic update runs (if necessary with empty configuration)
LED flashes slowly	Cyclic update stopped
LED flashes fast	Cyclic update interrupted as CPU does not reply

5.4 Festo field bus slave (FBSLAVE)

This driver enables the IPC to be used as an “intelligent” slave on a Festo field bus.

5.4.1 Selecting and parametrizing the driver

If the IPC is to become a Festo field bus slave, you must enter and parametrize the FBSLAVE driver in the driver configurator. The following entries are required:

Target disc drive

Specify the drive on which the necessary files can be found or onto which they must be loaded.

CP61 switch position

You must enter here the switch position which you have set on the rear of your field bus module (CP61).

5.4.2 Using the FBSLAVE module

The FBSLAVE module must first be imported like all modules, before it can be used. In the following examples it is assumed that the FBSLAVE module has been imported as CFM 0.

FBSLAVE Initialization/configuration

Before the IPC can work as the Festo field bus slave, the following data must be transferred to the firmware by means of a one-time module access.

Input parameter	
FU32	1 = function: Initialization/configuration
FU33	Switch position CP61 (4...7)
FU34	Baud rate (0...3): 0 = 31.25 Kbit/s 1 = 62.5 Kbit/s 2 = 187.5 Kbit/s 3 = 375 Kbit/s
FU35	Slave number (1...99)
FU36	Number of output bytes (0...24) or words (0...12)
FU37	Number of input bytes (0...24) or words (0...12)
FU38	1 = byte-orientated 2 = word-orientated

Return parameter	
FU32	None

Repeated accessing of this function has no effect. In order to reset the parameters, you must boot the IPC.

Inputs and outputs are specified as the slave sees them. If, therefore, the word “input” is used in this documentation, this indicates an input from the point of view of the slave. However, from the point of view of the master this is an output.

Example

```
IF           NOP
THEN   CMP 0           'FBSLAVE
         WITH V1       " Initialization
         WITH V4       " Switch position CP61
         WITH V3       " Baud rate (3=375 Kbit/s)
         WITH V24      " Number of inputs
         WITH V24      " Number of outputs
         WITH V1       " byte-orientated
```

This slave appears as follows in the slave list of the field bus master:

TN	Type	IW	OW
1	FB IPC (byte)	24	24

Example

```
IF     NOP
THEN   CMP 0           'FBSLAVE
         WITH V1       " Initialization
         WITH V4       " Switch position CP61
         WITH V3       " Baud rate (3=375 Kbit/s)
         WITH V1       " Slave number
         WITH V8       " Number of inputs
         WITH V12      " Number of outputs
         WITH V2       " word-orientated
```

This slave appears as follows in the slave list of the field bus master:

TN	Type	IW	OW
1	FB IPC (word)	8	12

Cyclic update

The input and output data of the IPC field bus slave are kept in a flag word range to be defined by the user. The contents of these flags can be copied from/to the inputs and outputs with the aid of an STL or LDR program or be processed further.

In order to update the values in the flags you must access the FBSLAVE module with FU32 = 2.

FBSLAVE

Updating the cyclic data

Input parameter	
FU32	2 = function: Updating the cyclic data
FU33	First flag word for outputs
FU34	First flag word for inputs

Return parameter	
FU32	None

Example

```
IF     NOP
THEN   CMP 0          'FBSLAVE
         WITH V2      " Cyclic update
         WITH V10     " First flag word for outputs
         WITH V20     " First flag word for inputs
```

Status interrogation

The status of the connection to the master can be interrogated with the module as follows:

FBSLAVE Status interrogation

Input parameter	
FU32	0 = function: Status interrogation

Return parameter	
FU32	Status The individual bits have the following meaning: Bit 0: Reserved, 0 or 1 Bit 1: Reserved, 0 or 1 Bit 2: 1, if there is no cyclic communication with a field bus master Bit 3: 1, if there is cyclic communication with a field bus master Bit 4: Reserved, 0 or 1 Bit 5: Reserved, 0 or 1 Bit 6: Reserved, 0 or 1 Bit 7: Reserved, 0 or 1 Bit 8...15: Reserved, always 0

Acyclic update

Access can be made to 256 parameter fields.

In order to enter a value in the parameter table, you should access the module as follows:

FBSLAVE Write parameters

Input parameter	
FU32	3 = function: Write parameters
FU33	Number of the parameter field
FU34	Value to be entered

Return parameter
None

In order to read a parameter value, as follows:

FBSLAVE Read parameters

Input parameter	
FU32	4 = read parameters
FU33	Number of the parameter field

Return parameter	
FU32	Value of the parameter field

Example

```
""Example program for the Festo field bus slave

STEP init
IF     NOP
THEN   CMP 0            " Field bus slave module
         WITH V1        " Initialization
         WITH V4        " Switch position CP61
         WITH V3        " Baud rate (3=375 Kbit/s)
         WITH V1        " Slave number
         WITH V24       " Number of outputs
         WITH V24       " Number of inputs
         WITH V1        " byte-orientated

STEP loop
IF     NOP
THEN   LOAD   IW1       " Physical input
       TO     FW9000    " Field bus output byte 0

IF     NOP
THEN   CMP 0            " Field bus slave module
         WITH V2        " Cyclic update
         WITH V9000     " Field bus outputs
                        " begin with FW9000
         WITH V9100     " Field bus inputs
                        " begin with FW9100

IF     NOP
THEN   LOAD   FW9100    " Field bus input byte 0
       TO     OW1       " Physical output

IF     NOP
THEN   JMP TO loop
```

5.5 PROFIBUS-DP with module CP62 (PDP driver)

With the PDP driver and module CP62 it is possible to connect PROFIBUS-DP slaves to the IPC and to use them with FST IPC.

Please note
The driver assumes that configuration has taken place correctly with the field bus project planning tool SyCon from Hilscher.

The inputs and outputs of the modules are copied cyclically onto the local function units (inputs and outputs or flags) which have been configured by the user.

5.5.1 Selecting and parametrizing the driver

If you wish to use the PROFIBUS-DP driver in an FST IPC project, you must enter and parametrize the PDP driver in the driver configuration.

Please note
The driver uses the same driver number as the PROFI-DP for CP60. It cannot therefore be used together in the same project.

Target disc drive

Specify the IPC drive on which the PROFIBUS-DP driver PDP.EXE can be found or onto which it must be loaded.

CP62 switch position (identifier 0...6)

Specify the identifier for the memory address range of the PROFIBUS module CP62 (see following table).

CP62 switch position (identifier)	Memory address range
0	CA000 ... CA7FF
1	CA800 ... CAFFF
2	CB000 ... CB7FF
3	CB000 ... CBFFF
4	CC000 ... CC7FF
5	CC800 ... CCFFF
6	D0000 ... D07FF

The memory address range of module CP62 can be set with the address selector switches KSW1 and KSW2 (see documentation for module CP62). The PDP driver does not use an Interrupt. Leave, therefore, the plug bridge J1 in the position as at delivery.

DR-RAM size

Enter here the appropriate DP-RAM size (2 or 8).

Use FW instead of IW

If the question is answered with “Yes”, the processing data will be copied into the corresponding flag word range (instead of into IWs).

Use FW instead of OW

If the question is answered with “Yes” the processing data will be copied into the corresponding flag word range (instead of into OWs).

Please note
Make sure that the I and O ranges in the flag words do not overlap.

Offset for inputs

Enter here the starting address for the input range of the processing data.

Offset for outputs

Enter here the starting address for the output range of the processing data.

5.5.2 Configuration

The driver assumes that configuration has taken place correctly with the field bus project planning tool SyCon from Hilscher. The following values should be selected with “Settings/DP master settings”:

Settings / DP master settings	**Values**
Startup behaviour after system initialization	Automatic release of the communication by the device
Monitoring time	300 ms
Addressing mode	Word addresses (important!)

Settings / DP master settings	Values
Handshake for the process data	Buffered, host-controlled
Fault reaction	Whether the system reacts “hard” or “soft” to missing slaves, depends on “Bus parameter / Edit bus parameter” Auto Clear. “Auto Clear on” = hard reaction

5.5.3 Fault messages

The driver enters the following error numbers in the error word of the IPC:

Error number	Description
1001	The driver cannot make connection with the card. Possible causes: – The card address is not the same as the value in the driver configuration. – No valid configuration data in the card (SyCon).
1004	Configured slave not on the bus (only with “hard reaction”)
1005	Input address range exceeded (offset)
1006	Output address range exceeded (offset)

5.5.4 Modules

Overview of modules

Modules	Description
DP_USIF	Interrogation of the USIF status
DP_GETSL	Asks whether there is a request for diagnosis
DP_GETDG	Request diagnostic information of a DP slave
DP_CONTR	PROFIBUS function “Global Control Request”

DP_USIF — Interrogation of the USIF status

Information on the current status. The user interface status cannot be modified.

Input parameter
None

Return parameter	
FU32	-1 = OK 0 = driver not loaded
FU33	New or current status or 0 = system not correctly initialized

DP_GETSL

Asks whether there is a request for diagnosis

Input parameter	
FU32	None

Return parameter	
FU32	-1 = OK 0 = driver not loaded
FU33	Address of the 1st. slave with diagnosis request or $FFFF = no request
FU34	Address of the 1st. missing slave

DP_GETDG

Request diagnostic information of a DP slave

This module should be used in conjunction with module DP_GETSL.

Input parameter	
FU32	Address of the slave (0...126)
FU33	Offset in diagnostic information
FU34	Group Select

Return parameter	
FU32	-1 = OK 0 = driver not loaded 1 = slave address not valid
FU33	PROFIBUS status of the operation; typical values: 00: OK C3: partner does not reply 04: no parallel master slave function possible.
FU34	Length of diagnostic information in bytes (without length byte)
FU35 ... FU38	The diagnostic information in continuous sequence

Maximum 4 words of diagnostic information are supplied per access.

Please note
Module access must be repeated until value 0 is returned in FU33. Only then are the other values binding.

DP_CONTR

Profibus function “Global Control Request”

All input parameters in BYTE format.

Input parameter	
FU32	Slave address or 127 = all slaves (broadcast)
FU33	Control command
FU34	Group select

Return parameter	
FU32	-1 = OK 0 = driver not loaded
FU33	PROFIBUS status of the operation; typical values: 00: OK C3: partner does not reply 04: no parallel master-slave function possible.

➔

Please note
Module access must be repeated until value 0 is returned in FU33. Only then are the other values binding.

5.6 PROFIBUS FMS (PROFIFMS driver)

With the PROFIFMS driver and module CP62 it is possible to operate the Festo IPC in a PROFIBUS FMS network.

Please note
The driver assumes that configuration has taken place correctly with the field bus project planning tool SyCon from Hilscher.

There are no external configuration files. The bus parameter, communication reference list and the object directory must be configured with SyCon.

5.6.1 Selecting and parametrizing the driver

If you wish to use the PROFIBUS-FMS driver in an FST IPC project, you must enter and parametrize the PROFIFMS driver in the driver configuration.

Target disc drive

Specify the IPC drive on which the PROFIBUS FMS driver PROFIFMS.EXE can be found or onto which it must be loaded.

CP62 switch position

Specify here the address switch setting of the CP62 module. The presetting CA indicates that the module uses a memory segment as from $CA00.

Please note
Make sure that the memory range used by the CP62 is not used by any other module.

5.6.2 Additional CI commands for PROFIFMS

This driver extends the scope of the command interpreter with the following CI commands:

CI command	Description
!39	Display driver information and version number Displays information on the driver and the version number. This display is also shown if an unknown command is entered (e.g. !39?).

5.6.3 Modules

Overview of modules

Modules	Description
FMSREAD	Read (polled response)
FMSWRIT	Write (polled response)

FMSREAD Read (polled response)

Input parameter	
FU32	Communication reference (CR=1...32)
FU33	Object index (20...65535)
FU34	Object subindex (0...241)
FU35	Flag word number for status variable
FU36	Lower-value byte, operand type for data 0 = Input word 1 = Output word 2 = Flag word 3 = Register Higher-value byte, PROFIBUS data type 1 = Boolean 2 = Integer8 3 = Integer16 4 = Integer32 5 = Unsigned8 6 = Unsigned16 7 = Unsigned32 8 = Floating point 9 = Visible string 10 = Octet string 11 = Date 12 = Time 13 = Time difference 14 = Bit string
FU37	Number of the first operand word for the data
FU38	Expected (maximum) length in bytes

Return parameter	
FU32	Fault number 0 = completed successfully, no fault › 0 = fault (see section 5.6.5)

Status variable

– 0 = completed successfully

– 1 = still being processed

– › 1 = concluded by fault (see table)

FMSWRIT Write (polled response)

Input parameter	
FU32 ... FU38	see FMSREAD

Return parameter	
FU32	see FMSREAD

Status variable see FMSREAD

5.6.4 Object directory

If other PROFIBUS FMS slaves wish to access FST operands, the operands must be configured in the object directory (in the SyCon configurator). Operands can be entered optionally. If they are missing from the object directory, they simply cannot be accessed.

Please note that not all operands can be accessed. Only input words, output words, registers and a range of the flag words can be accessed.

If operands are entered, this should take place in accordance with the following table:

Index	Designation in the IPC	Type	Access
100	IW0-IW63	Array(64)*U16	read
101	IW64-IW127	Array(64)*U16	read
102	IW128-IW191	Array(64)*U16	read
103	IW192-IW255	Array(64)*U16	read
110	OW0-OW63	Array(64)*U16	read/write
111	OW64-OW127	Array(64)*U16	read/write
112	OW128-OW191	Array(64)*U16	read/write
113	OW192-OW255	Array(64)*U16	read/write
120	R0/R63	Array(64)*U16	read/write
121	R64/R127	Array(64)*U16	read/write
122	R128/R191	Array(64)*U16	read/write
123	R192/R255	Array(64)*U16	read/write
200	FW0-FW63	Array(64)*U16	read/write
201	FW64-FW127	Array(64)*U16	read/write
202	FW128-FW191	Array(64)*U16	read/write
203	FW192-FW255	Array(64)*U16	read/write
204	FW256-FW319	Array(64)*U16	read/write
205	FW320-FW383	Array(64)*U16	read/write
206	FW384-FW447	Array(64)*U16	read/write
207	FW448-FW511	Array(64)*U16	read/write
208	FW512-FW575	Array(64)*U16	read/write
209	FW576-FW639	Array(64)*U16	read/write
210	FW640-FW703	Array(64)*U16	read/write

Index	Designation in the IPC	Type	Access
211	FW704-FW767	Array(64)*U16	read/write
212	FW768-FW831	Array(64)*U16	read/write
213	FW832-FW895	Array(64)*U16	read/write
214	FW896-FW959	Array(64)*U16	read/write
215	FW960-FW1023	Array(64)*U16	read/write
216	FW1024-FW1087	Array(64)*U16	read/write
217	FW1088-FW1151	Array(64)*U16	read/write
218	FW1152-FW1215	Array(64)*U16	read/write
219	FW1216-FW1279	Array(64)*U16	read/write
220	FW1280-FW1343	Array(64)*U16	read/write
221	FW1344-FW1407	Array(64)*U16	read/write
222	FW1408-FW1471	Array(64)*U16	read/write
223	FW1472-FW1535	Array(64)*U16	read/write
224	FW1536-FW1599	Array(64)*U16	read/write
225	FW1600-FW1663	Array(64)*U16	read/write
226	FW1664-FW1727	Array(64)*U16	read/write
227	FW1728-FW1791	Array(64)*U16	read/write
228	FW1792-FW1855	Array(64)*U16	read/write
229	FW1856-FW1919	Array(64)*U16	read/write
230	FW1920-FW1983	Array(64)*U16	read/write
231	FW1984-FW2047	Array(64)*U16	read/write

5.6.5 Fault code of the modules

If the return parameter FU32 ‹ › is 0, this is a fault code. The following fault codes are possible:

Fault code	Description
198	Function not implemented
199	PROFIFMS driver not loaded
200	Non-permitted parameter
201	No message number available (255 unfinished tasks)

5.6.6 Fault values in the status variable

The following table lists some of the possible fault values:

Fault values		Description
[hex]	[dec]	
$42	66	Connection interrupted
$43	67	Too many parallel services on one KR
$80	128	Connection cannot be opened
$81	129	Fault in the application of the remote partner

5.6.7 Heterogeneous networks

Check the bus parameters if an FPC-405 or PS1-CP60 is included in the PROFIBUS network. Set the baud rate to 500 kBits/s in the SyCon configurator and select for optimization: user-defined.

If you are using the FPC-405 with preset bus parameters, make the following settings:

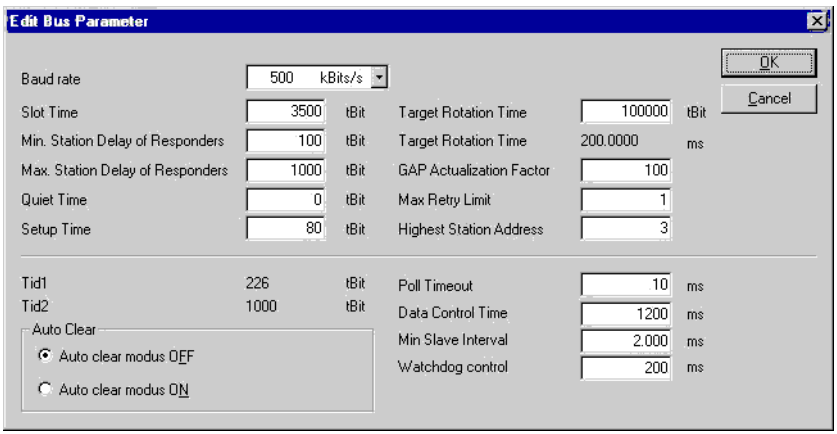


Fig. 5/7: Bus parameter

The communication references also deviate slightly from the presettings, seen from the point of view of the CP62:

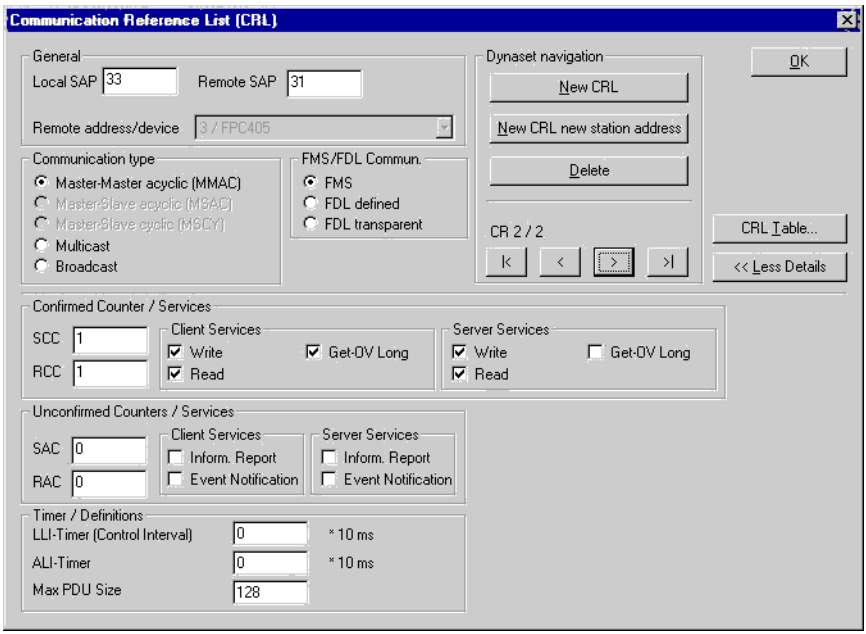


Fig. 5/8: Communication references

5.7 Modules for handling files

There are some standard modules for accessing file systems such as diskettes, hard discs, RAM discs, etc. They support drives inserted in the MS-DOS operating system.

With these modules it is possible to hold up to 6 files open simultaneously. If an attempt is made to open further files, the modules will return fault number 4 (“Too many open files”).

Please note
The modules can only be used on controllers of types HC1X and HC2X.

The files are identified by means of numbers. If the file number lies in the range 0 to 32767, and is therefore positive, it will be used immediately as the file name. For example, file number 17 will be converted to file name “17” in the current directory of the current drive. With negative file numbers, the modules search for the string driver and use the string with the opposite number as the file name. If the file number is -3 and the string with number 3 contains the value “C:\MYDATA.BIN”, this will be used as the name for the relevant file.

All modules for file access transfer their task to an MS-DOS task within the FST run time nucleus and return immediately. The basic task is then performed at a later stage by the MS-DOS task. The tasks received first will be carried out first. It is therefore possible that more than just one operation may be running simultaneously.

All modules for reading or writing use flag words as memory space for the data. The reason for this is that only the flag words offer sufficient space even for more complex tasks.

All modules return a result and use a status variable. A return value of zero means that the task has been started. It need not, however, be finished and this is usually the case. A return value which is not zero indicates a fault situation. The operation has then not been started and the value of the return indicates the type of fault, for example non-permitted parameters. A status variable with the value zero indicates the successful end of an operation. If the operation is not yet finished, the status variable will have the value -1. If a fault occurs, the value will become greater than zero. The relevant value also indicates the type of fault here. A list of faults is given below.

Please note
If you wish to carry out file operations although there is no diskette in the drive, you will not receive a fault message.

If a file is still open on a diskette, the diskette must not be removed from the drive. Under MS-DOS this will lead to ranges being lost and to even worse matters.

You must remember that MS-DOS does not continually supplement the table for the range division (FAT). This is not carried out until the file is closed. In order to ensure that all written data are contained in a file and that all ranges used have been noted, the file must be closed. A simple method is to go to the end of the file, write zero bytes in the file at this point and then close the file. In this way MS-DOS is compelled to reserve sufficient memory space on the storage medium and to supplement the table for the range division. You thus have the best chance of ensuring the integrity of a file system.

Another method of ensuring that the file is saved is to carry out certain checks. For this purpose the current position of the write/read pointer must be ascertained and saved. The file is then closed, opened again and positioned back at the position saved. This will also ensure that all the data is written into a file and the range table supplemented. If the CHKDSK program finds lost ranges on a diskette, the range table has not been supplemented correctly. The first method works nearly always; the second is less certain, but also writes partly written ranges onto the diskette.

The CI commands S, Y and the QUIT and EXIT commands automatically close all files.

5.7.1 Modules

Overview of modules

Module	Description
FCREATE	Create file
FOPEN	Open file
FCLOSE	Close file
FCLOSALL	Close all files
FDELETE	Delete a file
FSEEK	Position file pointer
FSEEKX	Position file pointer, return new position
FWRITE	Write in a file
FREAD	Read from a file
FWRITSTR	Write a string in a file
FREADSTR	Read a string from a file

FCREATE Create file

Creates a new file or opens an existing file for reading and writing. If the file already exists, it will be shortened to length 0.

Input parameter	
FU32	File number for the MS-DOS file system. A positive number is converted immediately to a file name. The number 17 produces the file name “17”. A negative number is used with reversed sign as an index in the string table of the string driver. The relevant string contains the file name.
FU33	Number of a flag word FW for returning a status. The status is: FW[FU33]= -1, if the function is still being processed 0, if the function has been completed successfully › 0, a fault number after a failure.
FU34	Number of a flag word FW for the file handle. This value is only valid if the flag word contains value 0 with the status. The file handle must be specified with all further operations with the opened file.

Return parameter	
FU32	0 = OK, the function has been started › 0 = fault number after a failure. The function has not been started.

FOPEN Open file

Opens an existing file in the mode specified.

Input parameter	
FU32	File number for the MS-DOS file system. A positive number is converted immediately to a file name. The number 17 produces the file name “17”. A negative number is used with reversed sign as an index in the string table of the string driver. The relevant string contains the file name.
FU33	Number of a flag word FW for returning a status. The status is: FW[FU33]= -1, if the function is still being processed 0, if the function has been completed successfully › 0, a fault number after a failure.
FU34	Number of a flag word FW for the file handle. This value is only valid if the flag word contains value 0 with the status. The file handle must be specified with all further operations with the opened file.
FU35	Mode in which the file must be opened. 0 = read only 1 = write only 2 = read and write

Return parameter	
FU32	0 = OK, the function has been started. › 0 = fault number after a failure. The function has not been started.

FCLOSE Close file

Close a previously opened file. When the file has been closed, the file handle is no longer valid and can no longer be used for further operations.

Input parameter	
FU32	File handle for the MS-DOS file, as supplied by the functions for opening or creating.
FU33	Number of a flag word FW for returning a status. The status is: FW[FU33]= -1, if the function is still being processed 0, if the function has been completed successfully › 0, a fault number after a failure.
FU34	

Return parameter	
FU32	0 = OK, the function has been started. › 0 = fault number after a failure. The function has not been started.

FCLOSEALL

Close all files

With this command all files still open will be closed immediately. All file handles become invalid after this operation and can no longer be used for further operations.

Input parameter	
FU32	Number of a flag word FW for returning a status. The status is: FW[FU32]= -1, if the function is still being processed 0, if the function has been completed successfully › 0, a fault number after a failure.

Return parameter	
FU32	0 = OK, the function has been started. › 0 = fault number after a failure. The function has not been started.

FDELETE — Delete a file

The file specified is deleted. A file which is to be deleted must not be open.

Input parameter	
FU32	File number for the MS-DOS file system. A positive number is converted immediately to a file name. The number 17 produces the file name “17”. A negative number is used with reversed sign as an index in the string table of the string driver. The relevant string contains the file name.
FU33	Number of a flag word FW for returning a status. The status is: FW[FU33]= -1, if the function is still being processed 0, if the function has been completed successfully › 0, a fault number after a failure.
FU34	

Return parameter	
FU32	0 = OK, the function has been started. › 0 = fault number after a failure. The function has not been started.

FSEEK Positioning in a file

Moves the write/read pointer of the file to a specified position. The position is specified in bytes.

Input parameter	
FU32	File handle for the MS-DOS file, as supplied by the functions for opening or creating.
FU33	Number of a flag word FW for returning a status. The status is: FW[FU33]= -1, if the function is still being processed 0, if the function has been completed successfully › 0 a fault number after a failure.
FU34	Lower-value word of a 4-byte file position
FU35	Higher-value word of a 4-byte file position

Return parameter	
FU32	0 = OK, the function has been started. › 0 = fault number after a failure. The function has not been started.

FSEEKX — Position file pointer, return new position

Position data pointer absolutely or relatively, with return of new file position.

Moves the write/read pointer of the file to or around a specified position. The position is specified in bytes.

Input parameter	
FU32	File handle for the MS-DOS file, as supplied by the functions for opening or creating.
FU33	Number of a flag word FW for returning a status. The status is: FW[FU33]= -1, if the function is still being processed 0, if the function has been completed successfully › 0 a fault number after a failure.
FU34	Lower-value word of an absolute or relative 4-byte file position
FU35	Higher-value word of an absolute or relative 4-byte file position
FU36	Operation: 0 = absolute from the beginning of the file 1 = relative to the current position 2 = relative to the end of the file
FU37	Number of the flag word for returning the new position; the new position is supplied as follows: FW[FU37] Lower-value word of an absolute 4-byte file position FW[FU37+1] Higher-value word of an absolute 4-byte file position

Return parameter	
FU32	0 = OK, the function has been started. › 0 = fault number after a failure. The function has not been started.

FWRITE — Write in a file

A range of the flag words is entered in the opened file. The specified amount must be calculated in bytes.

Input parameter	
FU32	File handle for the MS-DOS file, as supplied by the functions for opening or creating.
FU33	Number of a flag word FW for returning a status. The status is: FW[FU33]= -1, if the function is still being processed 0, if the function has been completed successfully › 0 a fault number after a failure.
FU34	Number of the first flag word which is to be written as data
FU35	Number of bytes which are to be written. Writing 0 bytes will shorten the file at the current position. If the number is odd, only the lower-value byte of the last flag word will be written.

Return parameter	
FU32	0 = OK, the function has been started. › 0 = fault number after a failure. The function has not been started.

FREAD

Read from a file

Data from the opened file is entered in a range of the flag words. The specified amount must be calculated in bytes.

Input parameter	
FU32	File handle for the MS-DOS file, as supplied by the functions for opening or creating.
FU33	Number of a flag word FW for returning a status. The status is: FW[FU33]= -1, if the function is still being processed 0, if the function has been completed successfully › 0 a fault number after a failure.
FU34	Number of the first flag word in which the data are to be read.
FU35	Number of bytes which are to be read. If the number is odd, only the lower-value byte of the last flag word will be read.
FU36	Number of a flag word FW in which the actual number of bytes read is entered.

Return parameter	
FU32	0 = OK, the function has been started. › 0 = fault number after a failure. The function has not been started.

FREADSTR — Read a string from a file

Characters are read out of the file into a string up to a specified data delimiter.

Input parameter	
FU32	File handle for the MS-DOS file, as supplied by the functions for opening or creating.
FU33	String number
FU34	Maximum number of bytes which are to be read.
FU35	Data delimiters (0...255). If 0 is specified, reading will be carried out to the next CR LF.

Return parameter		
FU32	0:	DONE, the function has been completed successfully.
	200:	String driver not found
	201:	The end of the file is reached before the data delimiter is found. No characters are copied into the string and the data pointer remains in the old position.
	202:	The specified maximum number of characters was read before the data delimiter was found. No characters are copied into the string and the data pointer remains in the old position.
	› 0:	A fault has occurred (see below).

FWRITSTR — Write a string in a file

The string is written in the file and a punctuation mark is added.

Input parameter	
FU32	File handle for the MS-DOS file, as supplied by the functions for opening or creating.
FU33	String number
FU34	Maximum number of bytes which are to be written.
FU35	Data delimiter If 0 is specified, CR LF will be added. If -1 is specified, no data delimiter will be added.

Return parameter	
FU32	0: DONE, the function has been completed successfully. 200: String driver not found › 0: A fault has occurred (see section 5.7.2).

5.7.2 Fault numbers and status values

Most file operations use a status word for returning information. This status variable has the following meanings.

Status variable	Description	
-1	BUSY	Function is not yet completed
0	OK	Function has been completed successfully.
› 0	ERROR	Fault. Function has not been completed successfully. One of the following fault numbers is returned (see following table).

Fault number	Description
2	File not found
3	Path not found
4	Too many open files
5	Access refused
6	Non-permitted file handle
12	Non-permitted access
100	Non-permitted parameter
101	Diskette, hard disc full
102	No memory space

5.8 Setting the log-in method

The log-in method can be set with the COM1METH module. Four methods are supported. All methods perrmit after log-in any desired setting of the baud rate (see CI command MV). After booting method 1 is set.

Method	Description
1	After BREAK, cyclic switching is made twice to 9600 Baud and once to 2400 Baud. This is the basic method (default).
2	9600 Baud is always set (old method).
3	2400 Baud is always set (sensible with slow modem connections and when the Field PC Net alias MpRAM is used).
4	After BREAK, cyclic switching is made three times to 2400 Baud and twice to 9600 Baud.

COM1METH Setting the log-in method

Input parameter	
FU32	0: Method 1 1: Method 2 2: Method 3 3: Method 4

Return parameter	
FU32	Current/new log-in method

Instructions on addressing

Chapter 6

Contents

Contents of this chapter	This chapter provides instructions on configuring and addressing the FEC Standard, FEC Compact, HC0X as well as the I/O modules of the PS1.
Further information	Further information on address assignment can be found in the documentation for the relevant PLC/IPC. Information on configuring CPX terminals with the CPX Front End Controller type CPX-FEC can be found in the manual for the CPX Front End Controller (P.BE-CPX-FEC-...).

6.1 Local inputs/outputs of FEC Standard, FEC Compact and HC0x

This section contains information on the local inputs and outputs as well as the I/O configuration of the FEC Standard, FEC Compact und HC0X controllers.

Please note
The inputs have a delay time of 5 ms.

6.1.1 FEC Standard

In order to program the inputs and outputs of the FEC Standard, you must select the I/O module which corresponds to the controller in the I/O configuration, e.g. FC440. It is not necessary to specify a switch position for the digital inputs/outputs. Specify the input words and output words which you wish to use (e.g. both 0). If you have selected an I/O module with the addition “Word”, you can access the inputs and outputs via all 16 bits of the FST input and output words. In the case of the modules without addition, only the 8 lower-value bits per FST word are used.

PLC safety is implemented via the separate module FEC Standard PLC safety. If the hardware cannot be identified as FEC Standard, FST error 12 will be generated. Error 11 will be generated if the power supply is not connected to the output modules, or if the outputs are overloaded. The cause will be displayed in the FST input word which you have entered for the I/O module:

Bit number	Description
BIT 0	O0 overload
BIT 1	O1 overload
BIT 2, 3	Reserved

Bit number	Description
BIT 4	O0 no 24 V
BIT 5	O1 no 24 V

Caution
The input word displays the current status. In order to make sure that the cause is available later, you must write a error program.

For the analogue inputs and outputs, please select the I/O modules Standard Analogue Input 0 to 20 mA and Standard Analogue Output 0 to 20 mA. Specify the input words and output words which you wish to use. You can not use here the same input/output words as for the digital inputs/outputs. It is not necessary to specify a switch position for the analogue output channel (default = switch position 0). In the case of the analogue inputs, please note the assignment of the inputs to the switch position as shown in the following table.

Analogue channel no.	Switch position
0	0
1	1
2	2

PLC analogue safety is implemented via the separate module FEC Standard PLC safety Analog. If the hardware cannot be identified as FEC Standard with analogue inputs or outputs, FST error 12 will be generated. Error 11 will be generated if the power supply is not connected to the output modules, or if the outputs are overloaded.

The cause will be displayed in the FST input word which you have entered for the I/O module:

Bit number	Description
Bit 0 ... Bit 7	Reserved
Bit 8	Oin 0 overload
Bit 9	Oin 1 overload
Bit 10	Oin 2 overload
Bit 11	Reserved
Bit 12	No 24 V (analogue output)
Bit 13 ... Bit 15	Reserved

6.1.2 FEC Compact

In order to program the inputs and outputs of the FEC Compact, you must select the I/O module FEC in the I/O configuration. It is not necessary to specify a switch position. Specify the input words and output words which you wish to use (e.g. both 0).

The 12 inputs of the FEC are divided into two groups: first group 8 inputs, second group 4 inputs. The first group is faded into the lower 8 bits of the first configured input word (e.g. I0.0 to I0.7); the second group is faded into the lower 4 bits of the following input word (e.g. I1.0 to I1.3).

The 8 outputs of the FEC are programmed via the lower 8 bits of the first configured output word (e.g. O0.0 to O0.7).

6.1.3 HC0X

The HC0X has 8 local I/O points. These can be used either as inputs or outputs.

Enter module HC0X in the I/O configuration. It is not necessary to specify a switch position. Determine the input and output words for programming the local I/Os (e.g. both 0). One input word and two output words (e.g. OW0 and OW1) will then be assigned.

The 8 I/O points represent the lower 8 bits of the input word or of the first output word (e.g. I0.0 to I0.7 or O0.0 to O0.7). By means of the lower 8 bits of the second output word (e.g. OW1) you can determine whether an I/O point is to be used as an input or an output. If a bit is set, the corresponding I/O point will be an output, otherwise an input.

6.1.4 Analogue potentiometer

FEC Compact and HC01 contain an analogue potentiometer, which can easily be set within a decimal value range from 1 to 63 with the aid of a screwdriver.

Enter the Trimmer module in the I/O configuration and determine the input word in which the current potentiometer setting is saved.

6.1.5 Rotary switch

FEC Standard controllers have a rotary switch with 16 positions (0...F). This switch functions at the same time as a start/stop switch; 0 indicates stop, all other positions start.

In order to interrogate the switch positions in control programs, enter the I/O module Rotary switch in the I/O configuration. Specify an FST input word in which the current switch position is to be made available.

For compatibility reasons you can also use the I/O Trimmer module with the FEC Standard. The switch positions are mapped as follows on values from 1...63:

Switch position	Value
0 1 2 3 4 5	1 1 5 8 12 16
6 7 8 9 A	21 26 32 36 41
B C D E F	46 50 54 59 63

Please note
In order to guarantee that switching can be made directly from program 1 to program 5 with the rotary switch (intended as a program preselect) without addressing programs 2, 3 or 4, a delay of approx. 2 seconds has been included in order to wait until the correct value is set. (With the Trimmer of the FEC Compact, the values will always be made available immediately without delay of the PLC function.)

6.1.6 Fast counter

The last two inputs of the second group of the FEC Compact (I1.2 and I1.3) as well as the first two inputs of the HC0X (I0.0 and 0.1) can also be used as 1 or 2 independent fast counters. Inputs, which are configured as counters, can be read at the same time as normal digital inputs.

The following inputs are used with the FEC Standard:

Type	Inputs
FC4xx	I1.6 and I1.7
FC6xx	I3.6 and I3.7

These counters are interrupt-controlled. Once activated, they run independently of the user programs. They are not therefore influenced by the cycle time of the user programs.

Enter the module Fast Counter in the I/O configuration. Switch position 0 corresponds to the counter at input I1.2 (FEC Compact) or I0.0 (HC0x) and switch position 1 to a counter at I1.3 (FEC Compact) respectively I0.1 (HC0x). Enter the input word which is to contain the counter value.

For each negative edge at the counter input, the counter status of the specified FST input word will be incremented.

It is not possible to reset the counter status. If it is necessary to reset the counter status, please use the driver for fast counters which is described in a separate chapter.

6.1.7 Software incremental encoder

HC0x and FEC have a 16 bit deep software incremental encoder. Connection is made via the two interrupt inputs I1.0 and I1.1 (FEC Compact) or I0.2 and I0.3 (HC0x).

The encoder runs up to 200 Hz. Resetting can be made via the input I0.7 (FEC Compact and HC0x). The inputs can be used simultaneously as normal digital inputs.

The following inputs are used with the FEC Standard:

Type	**Inputs**
FC4xx	I1.4, I1.5 and I1.0
FC6xx	I3.4, I3.5 and I3.0

The encoder is interrupt-controlled. Once activated, it runs independently of the user programs. It is not therefore influenced by the cycle time of the user programs.

Enter the Incremental Encoder module in the I/O configuration and specify the input word for the counter status (e.g. IW5). The direction of rotation is then contained in the following input word (e.g. IW6).

In order to reset the counter status, call the ABRESET module.

ABRESET Resetting the counter status

Input parameter
None

Return parameter
None

ABMODE Mode for positioning movement/reset

The counter can be reset by means of input I0.7/I1.0/I3.0. The counter is reset when input I0.7/I1.0/I3.0 is set and when there is a positive edge at one of the A/B inputs. You can select one of four different modes.

Input parameter	
FU32	Mode for positioning movement/reset 0 = no reset 1 = direction 0, once 3 = direction 0, continuously 5 = direction 1, once 7 = direction 1, continuously

Return parameter
None

6.2 Input and output modules (PS1)

This section describes the input and output modules which are supported by the FST IPC PS1. However, it does not replace the manual accompanying the hardware, but describes the special points concerning the use with FST. The pin assignment and technical data can be found in the manual provided with the hardware.

PLC safety

Most I/O modules are equipped with a so-called PLC safety function. This enables the run time system (FST IPC nucleus) to monitor the existence of planned modules at the start of the project and to monitor their function during running time. If an incorrect function is detected, one of the following FST errors will be triggered.

Error	Brief description	Description
12	I/O module not found or double	If error 12 occurs, a planned module cannot be found or there are several modules at the same physical I/O address on the bus board.
11	I/O module defective	If error 11 occurs, either the module is defective, you have not connected the necessary external power supply or there is a short circuit. You can read information on this out of the status input word of the relevant module. For this purpose a further FST input word is assigned to each module which supports this feature. The meaning of the individual bits is module-specific and is shown in the following tables.

Please note
The input word displays the current status. In order to make sure that the cause is available later, you must write a error program.

Procedure for overload/short circuit

If there is a short circuit (or an overload), you should observe the following points:

– You must make sure that the output is not set again until the cause of the problem has been eliminated.

– If you are working in machine mode and have not set a error program, all outputs will be switched off automatically when the fault occurs. However in system mode, or when a error program is started, you must make sure yourself that the relevant output is reset.

– In order to ascertain the relevant output, you can evaluate the fault information and the status input word. You should note here that the outputs are usually in groups of 8 which you must reset completely.

Assignment of the status register

Bit no.	Name	Function
0	–	Reserved
1	VRLD	Overload/short circuit
2	–	Reserved
3	No 24 V	No external power supply
4	–	Reserved
5	–	Reserved
6	–	Reserved
7	–	Reserved

Working without PLC safety

If you do not wish to use the PLC safety function for a module, you can select the relevant module driver in the I/O configurator of the FST IPC. In the following table I/O modules are shown opposite the relevant module driver without PLC safety which should be used.

Module	Driver without PLC safety
OM11	OM10
OM40	OM10
OM21	OM20
IM11	IM10
IM51	IM10

6.3 Digital input and output modules for PS1

The following table provides an overview of the digital modules available. They are described in the following sections.

Module	Function	PLC safety
IM10	2x8 digital inputs	–
IM11	2x8 digital inputs	Yes
IM12	4x8 digital inputs	Yes
IM51	2x8 digital inputs (NPN)	Yes
OM10	2x8 digital outputs	–
OM11	2x8 digital outputs	Yes
OM12	4x8 digital outputs	Yes
OM40	2x8 digital outputs (1 A)	Yes
OM20	1x8 digital inputs 1x8 digital outputs	–
OM21	1x8 digital inputs 1x8 digital outputs	Yes
OM22	4x8 digital inputs 4x8 digital outputs	Yes
OM70	6 relays, switch-over contacts	Yes
OM74	16 relays, switch-on contacts	Yes
TM10	16 digital inputs (TTL) 16 digital outputs (TTL)	Yes
AS11	8 buttons, 8 LEDs	–
AS12	32 LEDs (red/green)	Yes
AS13	32 switches (E-M-E)	Yes
AS14	32 LEDs (green)	Yes

Tab. 6/1: Overview of digital input and output modules

6.3.1 Digital input modules

Input module IM10

2x8 optocoupler-separated digital inputs

IM10	
Inputs	16
Input words	1
Switch positions	1...9
Sum of I	144

Assignment of the input words	
IW(n)	I0 (0...7)
	I1 (8...15)

Please note
Switch positions 7 and 8 are not permitted with HC20 controllers.

Input module IM11 (PLC safety)

2x8 optocoupler-separated digital inputs with PLC safety and LEDs for the inputs

IM11 PLC safety	
Inputs	16
Input words	1
Switch positions	1...9
Sum of I	144
Module identification	Yes
Module check	Yes
Status information (additional input words)	1

Assignment of the input words	
IW(n)	I0 (0...7)
	I1 (8...15)
IW(n+1)	Status I0 (0...7)
	Status I1 (8...15)

Please note
Switch positions 7 and 8 are not permitted with HC20 controllers.

Input module IM12 (PLC safety)

2x8 optocoupler-separated digital inputs with PLC safety and LEDs for the external 24 V supply

IM12 PLC safety	
Inputs	32
Input words	2
Switch positions	1...F
Sum of I	480
Module identification	Yes
Module check	Yes
Status information (additional input words)	2

Assignment of the input words	
IW(n)	I0 (0...7)
	I1 (8...15)
IW(n+1)	I2 (0...7)
	I3 (8...15)
IW(n+2)	Status I0/I1 (0...7)
IW(n+3)	Status I2/I3 (8...15)

Please note
Switch position 4 is not permitted with HC20 controllers.

Input module IM12

4x8 optocoupler-separated digital inputs with PLC safety and LEDs for the external 24 V supply

IM12	
Inputs	32
Input words	2
Switch positions	1...F
Sum of I	480

Assignment of the input words	
IW(n)	I0 (0...7)
	I1 (8...15)
IW(n+1)	I2 (0...7)
	I3 (8...15)

Please note
Switch position 4 is not permitted with HC20 controllers.

Input module IM51 (PLC safety)

2x8 optocoupler-separated digital inputs with PLC safety and LEDs for the inputs

IM51 PLC safety	
Inputs	32
Input words	2
Switch positions	1...F
Sum of I	480
Special features	NPN (switching to mass)
Module identification	Yes
Module check	Yes
Status information (additional input words)	1

Assignment of the input words	
IW(n)	I0 (0...7)
	I1 (8...15)
IW(n+1)	Status I0 (0...7)
	Status I1 (8...15)

Please note
Switch positions 7 and 8 are not permitted with HC20 controllers.

6.3.2 Digital output modules

Output module OM10

2x8 optocoupler-separated digital outputs

OM10	
Outputs	16
Output words	1
Switch positions	1...9
Sum of O	144

Assignment of the output words	
OW(n)	O0 (0...7)
	O1 (8...15)

Please note
Switch positions 7 and 8 are not permitted with HC20 controllers.

Output module OM11 (PLC safety)

2x8 optocoupler-separated digital inputs with PLC safety and LEDs for the outputs

OM11 PLC safety	
Outputs	16
Output words	1
Switch positions	1...8
Sum of O	128
Module identification	Yes
Module check	Yes
Status information (additional input words)	1

Assignment of the input and output words	
OW(n)	O0 (0...7)
	O1 (8...15)
IW(n)	Status O0 (0...7)
	Status O1 (8...15)

Please note
Switch positions 7 and 8 are not permitted with HC20 controllers.

Output module OM12 (PLC safety)

4x8 optocoupler-separated digital outputs with PLC safety and LEDs for the external 24 V supply and overload

OM12 PLC safety	
Outputs	32
Output words	2
Switch positions	1...F
Sum of O	480
Module identification	Yes
Module check	Yes
Status information (additional input words)	2

Assignment of the input and output words	
OW(n)	O0 (0...7)
	O1 (8...15)
OW(n+1)	O2 (0...7)
	O3 (8...15)
IW(m)	Status O0/O1 (0...7)
IW(m+1)	Status O2/O3 (0...7)

Please note
Switch position 4 is not permitted with HC20 controllers.

Output module OM12

4x8 optocoupler-separated digital outputs with LEDs for the external 24 V supply and overload

OM12	
Outputs	32
Output words	2
Switch positions	1...F
Sum of O	480

Assignment of the output words	
OW(n)	O0 (0...7)
	O1 (8...15)
OW(n+1)	O2 (0...7)
	O3 (8...15)

Please note
Switch position 4 is not permitted with HC20 controllers.

Output module OM20

1x8 optocoupler-separated digital outputs, 1x8 optocoupler separated digital inputs

OM20	
Inputs	8
Outputs	8
Input words	1
Output words	1
Switch positions	1...9
Sum of I	72
Sum of O	72

Assignment of the input and output words	
OW(n)	O0 (0...7)
IW(m)	I0 (0...7)

Output module OM21 (PLC safety)

1x8 optocoupler-separated digital outputs, 1x8 optocoupler-separated digital inputs with PLC safety and LEDs for the external 24 V supply and overload

OM21 PLC safety	
Inputs	8
Outputs	8
Input words	1
Output words	1
Switch positions	1...8
Sum of I	64
Sum of O	64
Module identification	Yes
Module check	Yes
Status information (additional input words)	1

Assignment of the input and output words	
OW(n)	O0 (0...7)
IW(m)	I0 (0...7)
IW(m+1)	Status I0 (0...7)
	Status O0 (8...15)

Output module OM22 (PLC safety)

2x8 optocoupler-separated digital outputs, 2x8 optocoupler-separated digital inputs with PLC safety and LEDs for the external 24 V supply and overload

OM22 PLC safety	
Inputs	16
Outputs	16
Input words	1
Output words	1
Switch positions	1...F
Sum of I	240
Sum of O	240
Module identification	Yes
Module check	Yes
Status information (additional input words)	2

Assignment of the input and output words	
OW(n)	O0 (0...7)
	O1 (8...15)
IW(m)	I0 (0...7)
	I1 (8...15)
IW(m+1)	Status O0 (0...7)
	Status I0 (8...15)
IW(m+2)	Status O1 (0...7)
	Status I1 (8...15)

Please note
Switch positions 7 and 8 are not permitted with HC20 controllers.

Output module OM22

2x8 optocoupler-separated digital outputs, 2x8 optocoupler-separated digital inputs with LEDs for the external 24 V supply and overload

OM22	
Inputs	16
Outputs	16
Input words	1
Output words	1
Switch positions	1...F
Sum of I	240
Sum of O	240

Assignment of the input and output words	
OW(n)	O0 (0...7)
	O1 (8...15)
IW(m)	I0 (0...7)
	I1 (8...15)

Please note
Switch positions 7 and 8 are not permitted with HC20 controllers.

Output module OM40 (PLC safety)

2x8 optocoupler-separated digital outputs with PLC safety and LEDs for the external 24 V supply and overload

OM40 PLC safety	
Outputs	16
Output words	1
Switch positions	1...8
Sum of O	128
Special features	2.5 A max.; 1 A nom. per channel
Module identification	Yes
Module check	Yes
Status information (additional input words)	1

Assignment of the input and output words	
OW(n)	O0 (0...7)
	O1 (8...15)
IW(m)	Status O0 (0...7)
	Status O1 (8...15)

Please note
Switch positions 7 and 8 are not permitted with HC20 controllers.

Output module OM70 (PLC safety)

6 relay outputs (switch-over contacts) with PLC safety and LEDs

OM70 PLC safety	
Outputs	6
Output words	1
Switch positions	1...9
Sum of O	56
Special features	Relays, switch-over contacts
Module identification	Yes
Module check	Yes

Assignment of the output words	
OW(n)	K5 (5)
	K4 (4)
	K3 (3)
	K2 (2)
	K1 (1)
	K0 (0)

Output module OM70

6 relay outputs (switch-over contacts) with PLC safety and LEDs

OM70	
Outputs	6
Output words	1
Switch positions	1...9
Sum of O	56
Special features	Relays, switch-over contacts

Assignment of the output words	
OW(n)	K5 (5)
	K4 (4)
	K3 (3)
	K2 (2)
	K1 (1)
	K0 (0)

Output module OM74 (PLC safety)

16 relay outputs (switch-on contacts) with PLC safety and LEDs

OM74 PLC safety	
Outputs	16
Output words	1
Switch positions	1...F
Sum of O	240
Special features	Reed relays, switch-on contacts
Module identification	Yes
Module check	Yes

Assignment of the output words	
OW(n)	K1 (0...7)
	K4 (8...15)

Please note
Switch positions 7 and 8 are not permitted with HC20 controllers.

Output module OM74

16-relay outputs (switch-on contacts) with LEDs

OM74	
Outputs	16
Output words	1
Switch positions	1...F
Sum of O	240
Special features	Reed relays, switch-on contacts

Assignment of the output words	
OW(n)	K1 (0...7)
	K4 (8...15)

Please note
Switch positions 7 and 8 are not permitted with HC20 controllers.

Output module OM75 (PLC safety)

16 relay outputs (switch-on contacts) with PLC safety and LEDs, max. switching current per relay 2 A

OM75 PLC safety	
Outputs	16
Output words	1
Switch positions	1...F
Sum of O	240
Special features	Reed relays, switch-on contacts
Module identification	Yes
Module check	Yes

Assignment of the output words	
OW(n)	K1 (0...7)
	K4 (8...15)

Please note
Switch positions 7 and 8 are not permitted with HC20 controllers.

Output module OM75

16 relay outputs (switch-on contacts) with LEDs, max. switching current per relay 2 A

OM75	
Outputs	16
Output words	1
Switch positions	1...F
Sum of O	140
Special features	Reed relays, switch-on contacts

Assignment of the output words	
OW(n)	K1 (0...7)
	K4 (8...15)

Please note
Switch positions 7 and 8 are not permitted with HC20 controllers.

6.3.3 Multi I/O module

Multi I/O module TM10 (PLC safety)

16 digital outputs (TTL), 16 digital inputs (TTL), with PLC safety

TM10 PLC safety	
Inputs	16
Outputs	16
Input words	1
Output words	1
Switch positions	1...9
Sum of I	144
Sum of O	144
Special features	TTL (5V)
Module identification	Yes
Module check	Yes

Assignment of the input and output words	
OW(n)	O0 (0...7)
	O1 (8...15)
IW(m)	I0 (0...7)
	I1 (8...15)

Please note
Switch positions 7 and 8 are not permitted with HC20 controllers.

Multi I/O module TM10

16 digital outputs (TTL), 16 digital inputs (TTL)

TM10	
Inputs	16
Outputs	16
Input words	1
Output words	1
Switch positions	1...9
Sum of I	144
Sum of O	144
Special features	TTL (5 V)

Assignment of the input and output words	
OW(n)	O0 (0...7)
	O1 (8...15)
IW(m)	I0 (0...7)
	I1 (8...15)

Please note
Switch positions 7 and 8 are not permitted with HC20 controllers.

6.3.4 Special modules

LED/button module AS11

Universal display and switch module with 8 buttons and 8 LEDs

AS11	
Inputs	8
Outputs	8
Input words	1
Output words	1
Switch positions	1...6
Sum of I	48
Sum of O	48
Special features	Only buttons and LEDs

Assignment of the input and output words			
OW(n)	1 (0)	IW(n)	1 (0)
	2 (2)		2 (2)
	3 (4)		3 (4)
	4 (6)		4 (6)
	5 (8)		5 (8)
	6 (10)		6 (10)
	7 (12)		7 (12)
	ESC (14)		ESC (14)

LED module AS12

Universal display and switch module with 32 LEDs
(16 red, 16 green)

AS12	
Outputs	32
Output words	2
Switch positions	1...5
Sum of O	160
Special features	Only LEDs

Assignment of the output words	
OW(n)	A (green)
OW(n+1)	B (red)

Switch module AS13

Universal switch module with 16 switches (E-M-E)

AS13	
Inputs	32
Input words	2
Switch positions	1...5
Sum of I	160
Special features	16 switches (E-M-E)

Assignment of the input words	
IW(n)	A
IW(n+1)	B

LED module AS14

Universal display module with 32 LEDs (green)

AS14	
Outputs	32
Output words	2
Switch positions	1...5
Sum of O	160
Special features	Only LEDs

Assignment of the output words	
OW(n)	A
OW(n+1)	B

6.4 Analogue I/O modules for PS1

The following table provides an overview of the analogue modules available. They are described in the following sections.

Module	Function	Range	PLC safety
IO10	8 analogue inputs (12 bits)	0...4096 mV	–
IO11	8 analogue inputs (12 bits)	0...20 mA	–
IO12	8 analogue inputs (12 bits)	0...10 V	–
IO40	4 analogue inputs (12 bits)	-10...+10 V -5...+5 V 0...10 V 0...5 V	Yes
IO41	4 analogue inputs (16 bits)	-10...+10 V -5...+5 V 0...10 V	Yes
IO48	4 analogue inputs (12 bits)	0...20 mA 4...20 mA	Yes
IO60	4 analogue outputs (12 bits)	-10...+10 V	–
IO61	4 analogue outputs (12 bits)	-10...+10 V	–
IO64	4 analogue inputs (12 bits)	-10...+10 V 0...10 V 0...20 mA	–
IO70	2 analogue outputs (12 bits)	-10...+10 V 0...10 V	–
IO71	2 analogue outputs (12 bits)	-10...+10 V 0...10 V	–
IO73	2 analogue outputs (12 bits)	0...20 mA 4...20 mA	–

Tab. 6/2: Overview of analogue input and output modules

6.4.1 Analogue input modules

Input module IO10 (0 to 4095 mV)

8 analogue inputs, 0 to 4095 mV, 12 bits

IO10 0 to 4095 mV	
Input words (channels)	8
Switch positions	1...4
Sum of I	32
Range	0...4095 mV
Resolution	12 bits

Assignment of the input words	
IW(n)	Channel #1
IW(n+1)	Channel #2
IW(n+2)	Channel #3
IW(n+3)	Channel #4
IW(n+4)	Channel #5
IW(n+5)	Channel #6
IW(n+6)	Channel #7
IW(n+7)	Channel #8

Register value	**Analogue value**
0	0 mV
100	100 mV
200	200 mV
500	500 mV
1000	1000 mV
2000	2000 mV
4095	4095 mV

Input module IO11 (0 to 20 mA)

8 analogue inputs, 0 to 20 mA, 12 bits

IO11 0 to 20 mA	
Input words (channels)	8
Switch positions	1...4
Sum of I	32
Range	0...20 mA
Resolution	12 bits

Assignment of the input words	
IW(n)	Channel #1
IW(n+1)	Channel #2
IW(n+2)	Channel #3
IW(n+3)	Channel #4
IW(n+4)	Channel #5
IW(n+5)	Channel #6
IW(n+6)	Channel #7
IW(n+7)	Channel #8

Register value	**Analogue value**
0	0 mA
205	1 mA
409	2 mA
1024	5 mA
2048	10 mA
4095	20 mA – 1LSB

Input module IO12 (0 to 10 V)

8 analogue inputs, 0 to 10 V, 12 bits

IO12 0 to 10 V	
Input words (channels)	8
Switch positions	1...4
Sum of I	32
Range	0...10 V
Resolution	12 bits

Assignment of the input words	
IW(n)	Channel #1
IW(n+1)	Channel #2
IW(n+2)	Channel #3
IW(n+3)	Channel #4
IW(n+4)	Channel #5
IW(n+5)	Channel #6
IW(n+6)	Channel #7
IW(n+7)	Channel #8

Register value	**Analogue value**
0	0 V
409	1 V
819	2 V
2048	5 V
4095	10 V – 1LSB

Input module IO40 (-10 to +10 V)

4 analogue inputs, -10 to +10 V, 12 bits

IO40 -10 to +10 V	
Input words (channels)	4
Switch positions	1...6
Sum of I	24
Range	-10...+10 V
Resolution	12 bits
Module identification	Yes

Assignment of the input words	
IW(n) IW(n+1) IW(n+2) IW(n+3)	Channel #1 Channel #2 Channel #3 Channel #4

Register value	**Analogue value**
-2048	-10 V
-1024	-5 V
0	0 V
1024	5 V
2047	10 V – 1LSB

Input module IO40 (0 to +10 V)

4 analogue inputs, 0 to 10 V, 12 bits

IO40 0 to 10 V	
Input words (channels)	4
Switch positions	1...6
Sum of I	24
Range	0...10 V
Resolution	12 bits
Module identification	Yes

Assignment of the input words	
IW(n) IW(n+1) IW(n+2) IW(n+3)	Channel #1 Channel #2 Channel #3 Channel #4

Register value	**Analogue value**
0	0 V
409	1 V
819	2 V
2048	5 V
4095	10 V – 1LSB

Input module IO40 (-5 to +5 V)

4 analogue inputs, -5 to +5 V, 12 bits

IO40 -5 to +5 V	
Input words (channels)	4
Switch positions	1...6
Sum of I	24
Range	-5...+5 V
Resolution	12 bits
Module identification	Yes

Assignment of the input words	
IW(n) IW(n+1) IW(n+2) IW(n+3)	Channel #1 Channel #2 Channel #3 Channel #4

Register value	**Analogue value**
-2048	-5 V
-819	-2 V
0	0 V
819	2 V
2047	5 V – 1LSB

Input module IO40 (0 to 5 V)

4 analogue inputs, 0 to 5 V, 12 bits

IO40 0 to 5 V	
Input words (channels)	4
Switch positions	1...6
Sum of I	24
Range	0...5 V
Resolution	12 bits
Module identification	Yes

Assignment of the input words	
IW(n) IW(n+1) IW(n+2) IW(n+3)	Channel #1 Channel #2 Channel #3 Channel #4

Register value	**Analogue value**
0	0 V
819	1 V
1638	2 V
4095	5 V – 1LSB

Input module IO41 (-10 to +10 V)

4 analogue inputs, -10 to +10 V, 16 bits

IO41 -10 to +10 V	
Input words (channels)	4
Switch positions	1...6
Sum of I	24
Range	-10...+10 V
Resolution	16 bits
Module identification	Yes

Assignment of the input words	
IW(n) IW(n+1) IW(n+2) IW(n+3)	Channel #1 Channel #2 Channel #3 Channel #4

Register value	**Analogue value**
-32768	-10 V
-16384	-5 V
0	0 V
16383	5 V
32767	10 V – 1LSB

Input module IO41 (0 to 10 V)

4 analogue inputs, 0 to 10 V, 16 bits

IO41 0 to 10 V	
Input words (channels)	4
Switch positions	1...6
Sum of I	24
Range	0...10 V
Resolution	16 bits
Module identification	Yes

Assignment of the input words	
IW(n) IW(n+1) IW(n+2) IW(n+3)	Channel #1 Channel #2 Channel #3 Channel #4

Register value	**Analogue value**
0	0 V
6554	1 V
13107	2 V
32768	5 V
65535	10 V – 1LSB

Input module IO41 (-5 to +5 V)

4 analogue inputs, -5 to +5 V, 16 bits

IO41 -5 to +5 V	
Input words (channels)	4
Switch positions	1...6
Sum of I	24
Range	-5...+5 V
Resolution	16 bits
Module identification	Yes

Assignment of the input words	
IW(n) IW(n+1) IW(n+2) IW(n+3)	Channel #1 Channel #2 Channel #3 Channel #4

Register value	**Analogue value**
-32768	-5 V
-16384	-2 V
0	0 V
16383	2 V
32767	5 V – 1LSB

Input module IO48 (0 to 20 mA)

4 analogue inputs, 0 to 20 mA, 12 bits

IO48 0 to 20 mA	
Input words (channels)	4
Switch positions	1...6
Sum of I	24
Range	0...20 mA
Resolution	12 bits
Module identification	Yes

Assignment of the input words	
IW(n) IW(n+1) IW(n+2) IW(n+3)	Channel #1 Channel #2 Channel #3 Channel #4

Register value	**Analogue value**
0	0 mA
205	1 mA
409	2 mA
1024	5 mA
2048	10 mA
4095	20 mA – 1LSB

Input module IO48 (4 to 20 mA)

4 analogue inputs, 4 to 20 mA, 12 bits

IO48 4 to 20 mA	
Input words (channels)	4
Switch positions	1...6
Sum of I	24
Range	4...20 mA
Resolution	12 bits
Module identification	Yes

Assignment of the input words	
IW(n) IW(n+1) IW(n+2) IW(n+3)	Channel #1 Channel #2 Channel #3 Channel #4

Register value	**Analogue value**
0	0 mA
205	1 mA
409	2 mA
1024	5 mA
2048	10 mA
4095	20 mA – 1LSB

6.4.2 Analogue output modules

Output module IO60/IO61/IO64 (-10 to +10 V)

4 analogue outputs, -10 to +10 V, 12 bits

IO60/IO61/IO64 -10 to +10 V	
Output words (channels)	4
Switch positions	0...4
Sum of O	20
Range	-10...+10 V
Resolution	12 bits

Assignment of the output words	
OW(n) OW(n+1) OW(n+2) OW(n+3)	Channel #1 Channel #2 Channel #3 Channel #4

Register value	**Analogue value**
-2048	-10 V
-1024	-5 V
0	0 V
1024	5 V
2047	10 V - 1LSB

Output module IO64 (0 to 10 V)

4 analogue outputs, 0 to 10 V, 12 bits

IO64 0 to 10 V	
Output words (channels)	4
Switch positions	0...4
Sum of O	20
Range	0...10 V
Resolution	12 bits

Assignment of the output words	
OW(n) OW(n+1) OW(n+2) OW(n+3)	Channel #1 Channel #2 Channel #3 Channel #4

Register value	**Analogue value**
0	0 V
409	1 V
819	2 V
2048	5 V
4095	10 V – 1LSB

Output module IO64 (0 to 20 mA)

4 analogue outputs, 0 to 20 mA, 12 bits

IO64 0 to 20 mA	
Output words (channels)	4
Switch positions	0...4
Sum of O	20
Range	0...20 mA
Resolution	12 bits

Assignment of the output words	
OW(n) OW(n+1) OW(n+2) OW(n+3)	Channel #1 Channel #2 Channel #3 Channel #4

Register value	**Analogue value**
0	0 mA
205	1 mA
409	2 mA
1024	5 mA
2048	10 mA
4095	20 mA – 1LSB

Output module IO70/IO71 (0 to 10 V)

2 analogue outputs, 0 to 10 V, 12 bits

IO70/IO71 0 to 10 V	
Output words (channels)	2
Switch positions	0...4
Sum of O	10
Range	0...10 V
Resolution	12 bits

Assignment of the output words	
OW(n) OW(n+1)	Channel #1 Channel #2

Register value	**Analogue value**
0	0 V
409	1 V
819	2 V
2048	5 V
4095	10 V – 1LSB

Output module IO70/IO71 (-10 V to +10 V)

2 analogue outputs, -10 V to +10 V, 12 bits

IO70/IO71 -10 V to +10 V	
Output words (channels)	2
Switch positions	0...4
Sum of O	10
Range	-10...+10 V
Resolution	12 bits

Assignment of the output words	
OW(n) OW(n+1)	Channel #1 Channel #2

Register value	**Analogue value**
-2048	-10 V
-1024	-5 V
0	0 V
1024	5 V
2047	10 V – 1LSB

Output module IO73 (0 to 20 mA)

2 analogue outputs, 0 to 20 mA, 12 bits

IO73 0 to 20 mA	
Output words (channels)	2
Switch positions	0...4
Sum of O	10
Range	0...10 mA
Resolution	12 bits

Assignment of the output words	
OW(n) OW(n+1)	Channel #1 Channel #2

Register value	**Analogue value**
0	0 mA
205	1 mA
409	2 mA
1024	5 mA
2048	10 mA
4095	20 mA – 1LSB

Overview of the drivers and modules

Appendix A

Contents

A.1 Modules

A.1.1 Standard modules

General modules	
Modules	**Description**
BLINK	General flashing bits
FIFO	“First-in-first-out” memory
INRANGE	Checks whether the value lies within a certain range
LOADSYNC	Load synchronization of project (new)
MINMAX	Ring buffer with minimum, maximum and medium value
SCALE	Scales a 16-bit value

Modules for controlling program processing	
Modules	**Description**
F4	Cyclic starting of a program
F8	Stop all cyclic programs
F23	Interrogates whether a program is ready for processing
F26	Controls the program the number of which is saved in a variable

Modules for error treatment	
Modules	**Description**
F21	Interrogate or set the number of the error program
F22	Set error treatment
F25	Set error treatment for I/O errors

Modules for modifying operands

Modules	Description
CHECKSUM	Checksum of part of the range of flag words
COPY	Copy part of the range of flag words
DINDEXMW	Flag word indexed decrementing
F9	Clear operands
FLAGBIT	Flag bit indexed setting or resetting
IINDEXMW	Flag word indexed incrementing
NINDEXMW	Delete flag word range
RINDEXMW	Flag word indexed reading
WINDEXMW	Flag word indexed writing

Modules for setting the real time clock

Modules	Description
F10	Set the time
F11	Set the date
F12	Interrogate time
F13	Interrogate date

Modules for 32-bit arithmetic

Modules	Description
LADD	Addition of 32-bit values
LCMP	Comparison of 32-bit values
LDIV	Division of 32-bit values
LMUL	Multiplication of 32-bit values
LNEG	Sign change with a 32-bit value
LSUB	Subtraction of 32-bit values

Modules for floating point operations

Modules	Description
FPA2F	Conversion of string to float
FPABCD	Carries out (A*B)/(C*D)
FPBINOP	Basic floating point binary operations
FPF2A	Conversion of float to string
FPF2I	Conversion of float to 16-bit integer
FPF2L	Conversion of float to 32-bit integer
FPGONIO	Trigonometrical functions
FpI2F	Conversion of 16-bit integer to float
FpL2F	Conversion of 32-bit integer to float
FPM1	Carries out ((A-B)*C – (D-E)*(F-G))/(100-H)
FPROM1	Evaluating customer-specific expressions
FPSQRT	Calculating square roots

Modules for string treatment

Modules	Brief description
FLAG2STR	Copy a flag word range into a string
STR2FLAG	Copy a string into a flag word range
STRADDR	Ascertain internal address of a string
STRAPPND	Append a character to a string
STRATOH	Convert hexadecimal string into word
STRATOI	Convert string into word with sign (signed)
STRATOIX	Convert string into word with sign (signed)
STRATOU	Convert string into word without sign (unsigned)
STRCAT	Combine two strings into a third

Modules	Brief description
STRCHECK	Memory check
STRCHGET	Extract character from a string
STRCHSET	Replace character in a string
STRCI	Carry out a CI command
STRCLR	Delete string
STRCMP	Character-by-character comparison of two strings, a distinction is made between lower and upper case letters
STRCPY	Copy string
STRDEL	Remove part of a string
STRDUMP	Output some strings
STRFILL	Create string with specified number of equal characters
STRFILLW	Fill string with another string right or left-justified
STRFINDC	Search for a character in a string
STRFINDS	Search for part of a string in a string
STRGROW	Increase string memory for an individual string
STRHTOA	Convert word into hexadecimal string
STRICMP	Character-by-character comparison of two strings, no distinction between lower and upper case letters
STRINIT	Initialization or re-initialization
STRINSRT	Inserting a string into another
STRITOA	Convert word with sign into string
STRLEFT	Transfer left string part
STRLEN	Length of a string
STRLOWER	Convert string into lower case letters
STRMID	Transfer centre string part

Modules	Brief description
STRNCMP	Comparison of the first characters of two strings, a distinction is made between lower and upper case letters
STRNICMP	Comparison of the first characters of two strings, no distinction between lower and upper case letters
STRRIGHT	Transfer right string part
STRSTAT	Status of string driver
STRUPPER	Convert string into upper case letters
STRUSAGE	User and free memory
STRUTOA	Convert word without sign into string

Module for the PID controller

Module	Description
PIDCFM	Set and start the driver

Modules for serial communication

Modules	Brief description
BREAKCOM	Create hardware break (only SERIALDR)
CLOSECOM	Close serial interface
F30	Set interface parameter
F31	Activate CI
F32	Empty interface buffer
F34	Interrogate number of records in receive buffer
F35	Modify standard data delimiter
GETCOM	Read a character from a serial interface
IS485	Interrogate whether interface is in RS485 mode (only SERIALDR)
OPENCOM	Open serial interface

Modules	Brief description
OPENCOMX	Open serial interface and set parameters
PRINTCOM	Write an FST string to a serial interface
PUTCOM	Send a character to a serial interface
READCOM	Read character chain and save in FST string (without data delimiter)
READLCOM	Read character chain and save in FST string (with data delimiter)
SETRTS	Switch on/off RTS for RS485 (only SERIALDR)

Modules for TCP/IP

Modules	Description
DNS_NAME	Set/interrogate host name and domain name
DNSRESOL	Supply IP address for host name
EASY_IO	Exchange inputs and outputs with another IPC
EASY_R	Request operand range from another IPC
EASY_S	Send operand range to another IPC
IP_ALIVE	Check whether IP address is known
IP_DNS	Set/interrogate IP address for DNS server
IP_GATE	Set/interrogate IP address for gateway
IP_IP	Set/interrogate own IP address
IP_IP2	Set/interrogate IP address for second network card
IP_MAC	Interrogate Ethernet MAC address via network module
IP_MASK	Set/interrogate own IP network mask
IP_MASK2	Set/interrogate IP network mask for second network card
IP_TABLE	Set/interrogate IP address in table
PING	Ping
SNTPTIME	Start time synchronization

Modules	Description
TCP_CLOS	Close TCP connection
TCP_HAND	Activate TCP handler
TCP_OPEN	Open TCP connection
TCP_RES	Reset TCP handler
TCP_SEND	Send TCP data package
TCP_STAT	Status of TCP connection
TCP_STR	Send a string via TCP
TFTPFILE	Send/request file
UDP_FW	Send flag word range to another IPC
UDP_HAND	Activate UDP handler
UDP_SEND	Send UDP data package
UDP_STR	Send a string via UDP

Module for SMTP (E-mail driver)

Module	Description
SMTPCFM	Interrogate status (FU32=0) Determine sender address and host name (FU32=1) Send E-mail (FU32=2)

A.1.2 Modules for CPX Front End Controller

Modules for parameters and data of the CPX terminal	
Modules	**Description**
C_AP_rd	Write analogue module parameters
C_AP_wr	Read analogue module parameters
C_MD_rd	Read module diagnostic data
C_MP_rd	Read general module parameters
C_MP_wr	Write general module parameters
C_ST_rd	Read CPX internal parameters and data
C_ST_wr	Write CPX internal parameters
C_STATUS	Interrogate diagnostic status
C_TR_rd	Read entries in diagnostic memory

A.1.3 Modules for FEC Compact, FEC Standard and PS1

Modules for the Software Incremental Encoder	
Modules	**Description**
ABMODE	Mode for positioning movement/reset
ABRESET	Reset the counter status

Module for fast counters	
Module	**Description**
FECCNTR	Reset the fast counter (FU32=0) Parametrize the fast counter (FU32=1) Activate the fast counter (FU32=2) Interrogate the status and current counter value (FU32=3)

Module for fast outputs

Module	Description
FASTOUT	Initialize driver (FU32=0) Stop output (FU32=1) Start output (FU32=2) Start output, limit pulse (FU32=3) Start output with 1 (FU32=4) Start output with 1, limit pulse (FU32=5) Interrogate output status (FU32=10)

Module for stepping motor driver

Module	Description
STEPLT	Interrogate revision (FU32=0) Position absolutely (FU32=1) Position relatively in a positive direction (FU32=2) Position relatively in a negative direction (FU32=-2) Reference travel (FU32=3) Move in positive direction (FU32=4) Move in a negative direction (FU32=-4) Set polarity (FU32=10) Position absolutely with speed profile (FU32=11) Position relatively in a positive direction with speed profile (FU32=12) Position relatively in a negative direction with speed profile (FU32=-12) Set the actual position (FU32=13) Interrogate actual position and status (FU32=14) Fast stop without ramp (FU32=20) Stop motor with specified delay (FU32=21)

Modules for WATCHDOG

Modules	Description
PAUSE	Stops the CPU
PAUSECLI	Stops the CPU and switches the Interrupts off

A.1.4 Further drivers and modules for FEC Compact

Module for FEC remote I/O extension	
Module	**Description**
REMDIAG	Function: Interrogate fault counter (FU32=1) Function: Reset total fault counter (FU32=2) Function: Set total fault counter (FU32=3)

A.1.5 Further drivers and modules for PS1

Module for Encoder module IM2...	
Module	**Description**
IM2X	Initialize and reset counter (FU33=-1) Read counter states (FU33=0) Load counter 1 with value (FU33=1) Load counter 2 with value (FU33=2) Load counter 3 with value (FU33=3)

Modules for AS-Interface	
Modules	**Description**
ASI_Mode	Set the reaction of the AS-Interface driver to configuration faults
ASI_Para	Transmit a parameter to an AS-Interface slave during run time
ASI_Res	Restart the cyclic update
ASI_Stat	Interrogate the flags of the sequence control level (OUF)

Modules for the Festo field bus master

Modules	Description
F40	Check the configuration of a field bus slave
F41	Read the parameter field of an “intelligent” field bus slave
F42	Write the parameter field of an “intelligent” field bus slave
F43	Delete the cyclic outputs of all field bus slaves
F44	Status interrogation of a slave
F47	– FU32=1: Set the fault treatment and reaction to an acyclic command – FU32=2: Interrogate the slave status
F48	Treatment of the total configuration

Module for the Festo field bus slave

Module	Description
FBSLAVE	Initialization/configuration (FU32=1) Updating the cyclic data (FU32=2) Status interrogation (FU32=0) Write parameters (FU32=3) Read parameters (FU32=4)

Modules for PROFIBUS-DP

Modules	Description
DP_CONTR	Profibus function “Global Control Request”
DP_GETDG	Request diagnostic information of a DP slave
DP_GETSL	Asks whether there is a request for diagnosis
DP_USIF	Interrogation of the USIF status

Modules for PROFIBUS FMS

Modules	Description
FMSREAD	Read (polled response)
FMSWRIT	Write (polled response)

Modules for handling files

Modules	Description
FCLOSALL	Close all files
FCLOSE	Close file
FCREATE	Create file
FDELETE	Delete a file
FOPEN	Open file
FREAD	Read from a file
FREADSTR	Read a string from a file
FSEEK	Position file pointer
FSEEKX	Position file pointer, return new position
FWRITE	Write in a file
FWRITSTR	Write a string in a file

A.2 Drivers

Driver name	Description	Version
ASI	AS-Interface	1.00
COMEXT	Serial communication	1.40
FECCPX	Drivers for CPX-FEC	0.44
FASTOUT	Fast outputs	0.20
FBSLAVE [1)]	Festo field bus slave	Aug. 1997
FCMASTER	FEC remote I/O extension	July 2001
FCSLAVE	FEC remote I/O extension	Sep. 2002
FECCNTR	Fast counter	1.00
FESTOBUS	Festo field bus master	2.22
FOSEXT [2)]	Serial communication	July 1999
FOSSIL2 [2)]	Serial communication	July 1999
FPMATHDR	Floating point operations	1.65
HC0XCOM	Serial communication	1.40
MODBUSTCP	MODBUS/TCP driver	0.37
PDP	PROFIBUS-DP	1.11
PIDDRV	PID driver	0.20
PROFIFMS	PROFIBUS FMS	0.50
SERIALDR	Serial communication	1.33
SMTP	E-mail driver	0.30
STEPLITE	Stepping motor driver	1.40
STRINGS	String driver	1.11
TCPIPCPX	TCP/IP driver	1.27
TCPIP_15	TCP/IP driver	1.27

Driver name	Description	Version
TCPIPDRV	TCP/IP driver	1.27
TCPIPFC2	TCP/IP driver	1.27
TCPIPFEC	TCP/IP driver	1.27
TCPIPHC0	TCP/IP driver	1.27
TCPIPXXX	TCP/IP driver	1.27
WATCHDRV	Watchdog driver	0.33
WEB_SRVR	Web server	1.03

[1] Help files for bus module
[2] Aids for data transmission onto older target systems

Index

Appendix B

Contents

B.1 Index

D

G

M

N

S

T

Conditions of use for ”Electronic documentation”

I. Protection rights and scope of use
The file of your choice is subject to safeguarding provisions. Festo or third parties have protection rights for this electronic documentation which Festo provides on portable data storage devices (diskettes, CD ROM, cartridge discs), as well as in Internet and/or Intranet, always referred to in the following as “electronic documentation”. In so far as third parties have whole or partial right of access to this electronic documentation, Festo has the appropriate rights of use. Festo permits the user the use under the following conditions:

1. Scope of use
a) The user of the electronic documentation is allowed to use this documentation for his own, exclusively company-internal purposes on any number of machines within his business premises (location). This right of use includes exclusively the right to save the electronic documentation on the central processors (machines) used at the location.
b) The electronic documentation may be printed out on a printer at the location of the user as often as desired, providing this printout is printed with or kept in a safe place together with these conditions of use and other user instructions.
c) With the exception of the Festo Logo, the user has the right to use pictures and texts from the electronic documentation for creating his own machine and system documentation. The use of the Festo logo requires written consent from Festo. The user himself is responsible for ensuring that the pictures and texts used match the machine/system or the relevant product.
d) Further uses are permitted within the following framework:
Copying exclusively for use within the framework of machine and system documentation from electronic documents of all documented supplier components.
Demonstrating to third parties exclusively under guarantee that no data material is stored wholly or partly in other networks or other data storage devices or can be reproduced there.
Passing on printouts to third parties not covered by the regulation in item 3, as well as any processing or other use, is not permitted.

2. Copyright note
Every ”Electronic document” receives a copyright note. This note must be included in every copy and in every printout.
Example: © 2003, Festo AG & Co. KG,
D-73726 Esslingen, Germany

3. Transferring the authorization of use
The user can transfer his authorization of use in the scope of and with the limitations of the conditions in accordance with items 1 and 2 completely to a third party. The third party must be made explicitly aware of these conditions of use.

II. Exporting the electronic documentation
When exporting the electronic documentation, the licence holder must observe the export regulations of the exporting country and those of the purchasing country.

III. Guarantee
1. Festo products are being further developed with regard to hardware and software. The hardware status and, where applicable, the software status of the product can be found on the type plate of the product. If the electronic documentation, in whatever form, is not supplied with the product, i.e. is not supplied on a data storage device (diskette, CD ROM, cartridge disc) as a delivery unit with the relevant product, Festo does not guarantee that the electronic documentation corresponds to every hardware and software status of the product. In this case, the printed documentation from Festo accompanying the product is alone decisive for ensuring that the hardware and software status of the product matches that of the electronic documentation.
2. The information contained in this electronic documentation can be amended by Festo without prior notice and does not commit Festo in any way.

IV. Liability/limitations of liability
1. Festo provides this electronic documentation in order to assist the user in creating his machine and system documentation. In the case of electronic documentation which in the form of portable data storage devices (diskettes, CD ROM, cartridge discs) does not accompany a product, i.e. which are not supplied together with that product, Festo does not guarantee that the electronic documentation separately available / supplied matches the product actually used by the user. The latter applies particularly to extracts of the documents for the user’s own documentation.
The guarantee and liability for separately available / supplied portable data storage devices, i.e. with the exception of the electronic documentation provided in Internet/Intranet, is limited exclusively to proper duplication of the software, whereby Festo guarantees that in each case the relevant portable data storage device or software contains the latest status of the documentation. In respect of the electronic documentation in Internet/Intranet it is not guaranteed that this has the same version status as the last printed edition.
2. Furthermore, Festo cannot be held liable for the lack of economic success or for damage or claims by third parties resulting from the use of the documentation by the user, with the exception of claims arising from infringement of the protection rights of third parties concerning the use of the electronic documentation.
3. The limitations of liability as per paragraphs 1 and 2 do not apply if, in cases of intent or wanton negligence or the lack of warranted quality, liability is absolutely necessary. In such a case, the liability of Festo is limited to the damage recognizable by Festo when the concrete circumstances are made known.

VI. Safety guidelines/documentation
Guarantee and liability claims in conformity with the regulations mentioned above (items III. and IV) can only be made if the user has observed the safety guidelines of the documentation in conjunction with the use of the machine and its safety guidelines. The user himself is responsible for ensuring that the electronic documentation, which is not supplied with the product, matches the product actually used by the user.