

FST Drivers Reference

This manual describes the drivers and modules of the FST Runtime Library.

Contents

- [Input and Output Modules](#)
- [FEC and HC0x](#)
- [AS-Interface \(CP96\)](#)
- [Festo Fieldbus Master \(CP61\)](#)
- [Festo Fieldbus Slave \(CP61\)](#)
- [Profibus DP \(CP62\)](#)
- [Profibus FMS \(CP62\)](#)
- [FEC Remote I/O Expansion](#)
- [File Handling Modules](#)
- [Strings](#)
- [Screen and keyboard](#)
- [32-Bit Arithmetic](#)
- [Incremental Encoder \(IM2X\)](#)
- [Fast counters \(FEC, HC0X\)](#)
- [Positioning \(AMxx\)](#)
- [Servopneumatic Positioning \(AM30\)](#)
- [PID driver](#)
- [Serial Communication](#)
- [TCP/IP driver](#)

Input and output modules

This chapter describes the input and output modules supported by the FST IPC. This chapter is not intended to replace the manual supplied, however, but focuses on special points of FST IPC application. For pin assignment and technical data please refer to the manual supplied with the hardware.

1. PLC security

Most I/O modules are equipped with a PLC security function. This allows the runtime system (FST IPC Kernel) to monitor the presence of configured modules when a project is started and the function of the module at runtime. An FST error is triggered if a fault is detected. The error numbers used are

11	I/O stage defective
12	I/O module not found , or duplicated.

Error 12

If Error 12 occurs, the configured module could not be found or there were several modules with the same physical I/O address on the busboard.

Error 11

If Error 11 occurs, either the module is faulty, you have not connected the required external power supply, or there is a short-circuit. For information on this, see the status input word for the appropriate module. An additional FST input word has been assigned to each module that supports this feature. The assignment of the individual bits is module-specific as shown in the following table.

Please note that the status input word is updated during each cycle, to be sure that the status is available later you should write an error program.

Procedure for overload/short-circuit

In the event of a short-circuit (or an overload), please observe the following points.

It must be ensured that the output can not be set until the cause of the problem has been eliminated.

If you are working in machine mode and have not set an error program, all outputs are automatically switched off when an error occurs.

In system mode – or if an error program has been started – you yourself must ensure that the output involved is reset. You can evaluate the error information and the status input word to determine the output involved. Here you should bear in mind that the outputs are normally grouped in sets of 8 that must be reset together.

Working without PLC security

If you do not wish to use the PLC security function for a module, you can select the appropriate module driver in the FST IPC I/O configurator. The following table shows the I/O modules and the appropriate drivers without security.

Module	Driver without PLC security
OM11	OM10
OM40	OM10
OM21	OM20
IM11	IM10
IM51	IM10

2. Digital input and output modules

The following table shows an overview of the digital modules available. They are described in the following sections.

Module	Features	PLC security Ident/Check/Status
IM10	2x8 digital inputs	
IM11	2x8 digital inputs	*/*/1
IM12	4x8 digital inputs	*/*/2

2. IM11 PLC security

2x8 opto-isolated digital inputs

with PLC security and control LEDs for the inputs

Inputs:	16
Outputs:	
Input words:	1
Output words:	
Switch positions:	1 to 9
Sum of inputs:	144
Sum of outputs:	
Special features:	
Module identification:	Yes
Module check:	Yes
Status information (additional input words):	1

Assignment of input words:

IW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	E1								E0							
1	Status E1								Status E0							

Assignment of status registers:

Bit no.	Name	Function
0	-	Reserved
1	-	Reserved
2	-	Reserved
3	NO 24V	No external voltage
4	-	Reserved
5	-	Reserved
6	-	Reserved
7	-	Reserved

3. IM12 PLC security

4x8 opto-isolated digital inputs

with PLC security and

control LEDs for external 24V supply

Inputs:	32
Outputs:	
Input words:	2
Output words:	
Switch positions:	1 to F
Sum of inputs:	480

Sum of outputs:	
Special features:	
Module identification:	Yes
Module check:	Yes
Status information (additional input words):	2

Assignment of input words:

IW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	E1								E0							
1	E3								E2							
2	Status E1								Status E0							
3	Status E3								Status E2							

Assignment of status registers:

Bit no.	Name	Function
0	-	Reserved
1	-	Reserved
2	-	Reserved
3	NO 24V	No external voltage
4	-	Reserved
5	-	Reserved
6	-	Reserved
7	-	Reserved

4. IM12

4x8 opto-isolated digital inputs

with PLC security and

control LEDs for external 24V supply

Inputs:	32
Outputs:	
Input words:	2
Output words:	
Switch positions:	1 to F
Sum of inputs:	480
Sum of outputs:	
Special features:	
Module identification:	No
Module check:	No
Status information (additional input words):	-

Assignment of input words:

IW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	E1								E0							
1	E3								E2							

5. IM51 PLC security

2x8 opto-isolated digital inputs

with PLC security and control LEDs for the inputs

Inputs:	16
Outputs:	
Input words:	1
Output words:	
Switch positions:	1 to 9
Sum of inputs:	144
Sum of outputs:	
Special features:	npn (negative switching)
Module identification:	Yes
Module check:	Yes
Status information (additional input words):	1

Assignment of input words:

IW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	E1								E0							
1	Status E1								Status E0							

Assignment of status registers:

Bit no.	Name	Function
0	-	Reserved
1	-	Reserved
2	-	Reserved
3	NO 24V	No external voltage
4	-	Reserved
5	-	Reserved
6	-	Reserved
7	-	Reserved

6. OM10

2x8 opto-isolated digital outputs

Inputs:	
Outputs:	16
Input words:	
Output words:	1

Switch positions:	1 to 9
Sum of inputs:	
Sum of outputs:	144
Special features:	
Module identification:	
Module check:	
Status information (additional input words):	

Assignment of output words:

OW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	A1								A0							

7. OM11 PLC security

2x8 opto-isolated digital outputs

with PLC security and control LEDs for the outputs

Inputs:	
Outputs:	16
Input words:	
Output words:	1
Switch positions:	1 to 8
Sum of inputs:	
Sum of outputs:	128
Special features:	
Module identification:	Yes
Module check:	Yes
Status information (additional input words):	1

Assignment of input and output words:

OW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	A1								A0							

IW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Status A1								Status A0							

Assignment of status registers:

Bit no.	Name	Function
0	-	Reserved
1	OVRLD	Overload / short circuit
2	-	Reserved

3	NO 24V	No external voltage
4	-	Reserved
5	-	Reserved
6	-	Reserved
7	-	Reserved

8. OM12 PLC security

4x8 opto-isolated digital outputs

with PLC security and control LEDs for external 24V supply and overload

Inputs:	
Outputs:	32
Input words:	
Output words:	2
Switch positions:	1 to F
Sum of inputs:	
Sum of outputs:	480
Special features:	
Module identification:	Yes
Module check:	Yes
Status information (additional input words):	2

Assignment of input and output words:

OW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	A1								A0							
1	A3								A2							

IW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Status A1								Status A0							
1	Status A3								Status A2							

Assignment of status registers:

Bit no.	Name	Function
0	-	Reserved
1	OVRD	Overload / short circuit
2	-	Reserved
3	NO 24V	No external voltage
4	-	Reserved
5	-	Reserved
6	-	Reserved
7	-	Reserved

9. OM12

4x8 opto-isolated digital outputs

with control LEDs for external 24V supply and overload

Inputs:	
Outputs:	32
Input words:	
Output words:	2
Switch positions:	1 to F
Sum of inputs:	
Sum of outputs:	480
Special features:	
Module identification:	No
Module check:	No
Status information (additional input words):	-

Assignment of input and output words:

OW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	A1								A0							
1	A3								A2							

10. OM40 PLC security

2x8 opto-isolated digital outputs

with PLC security and control LEDs for the outputs

Inputs:	
Outputs:	16
Input words:	
Output words:	1
Switch positions:	1 to 8
Sum of inputs:	
Sum of outputs:	128
Special features:	2.5 A max. ; 1 A nom. per channel
Module identification:	Yyes
Module check:	Yes
Status information (additional input words):	1

Assignment of input and output words:

OW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	A1								A0							

IW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Status A1								Status A0							

Assignment of status registers:

Bit no.	Name	Function
0	-	Reserved
1	OVRLD	Overload / short circuit
2	-	Reserved
3	NO 24V	No external voltage
4	-	Reserved
5	-	Reserved
6	-	Reserved
7	-	Reserved

11. OM20

2x8 opto-isolated digital outputs,

2x8 opto-isolated digital inputs

Inputs:	8
Outputs:	8
Input words:	1
Output words:	1
Switch positions:	1 to 9
Sum of inputs	72
Sum of outputs	72
Special features	
module identification:	
Module check:	
Status information (additional input words):	

Assignment of input and output words:

IW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																E0

OW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																A0

12. OM21 PLC security

2x8 opto-isolated digital outputs,

2x8 opto-isolated digital inputs

with PLC security and
control LEDs for the inputs and outputs

Inputs:	8
Outputs:	8
Input words:	1

Output words:	1
Switch positions:	1 to 8
Sum of inputs	64
Sum of outputs	64
Special features	
Module identification:	Yes
Module check:	Yes
Status information (additional input words):	1

Assignment of input and output words:

IW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0									E0							
1	Status E0								Status A0							

OW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0									A0							

Assignment of status register A0:

Bit no.	Name	Function
0	-	Reserved
1	OVRD	Overload / short circuit
2	-	Reserved
3	NO 24V	No external voltage
4	-	Reserved
5	-	Reserved
6	-	Reserved
7	-	Reserved

Assignment of status register E0:

Bit no.	Name	Function
0	-	Reserved
1	-	Reserved
2	-	Reserved
3	NO 24V	No external voltage
4	-	Reserved
5	-	Reserved
6	-	Reserved
7	-	Reserved

13. OM22 PLC security

2x8 opto-isolated digital outputs,

2x8 opto-isolated digital inputs

with PLC security and
control LEDs for external 24V supply and overload

Inputs:	16
Outputs:	16
Input words:	1
Output words:	1
Switch positions:	1 to F
Sum of inputs	240
Sum of outputs	240
Special features	
Module identification:	Yes
Module check:	Yes
Status information (additional input words):	2

Assignment of input and output words:

IW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	E1								E0							
1	Status E0								Status A0							
2	Status E1								Status A1							

OW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	A1								A0							

Assignment of status register Ax:

Bit no.	Name	Function
0	-	Reserved
1	OVRLD	Overload / short circuit
2	-	Reserved
3	NO 24V	No external voltage
4	-	Reserved
5	-	Reserved
6	-	Reserved
7	-	Reserved

Assignment of status register Ex:

Bit no.	Name	Function
0	-	Reserved
1	-	Reserved
2	-	Reserved
3	NO 24V	No external voltage
4	-	Reserved
5	-	Reserved
6	-	Reserved
7	-	Reserved

2x8 opto-isolated digital outputs,

2x8 opto-isolated digital inputs

with control LEDs for external 24V supply and overload

Inputs:	16
Outputs:	16
Input words:	1
Output words:	1
Switch positions:	1 to F
Sum of inputs	240
Sum of outputs	240
Special features	
Module identification:	No
Module check:	No
Status information (additional input words):	-

Assignment of input and output words:

IW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	E1								E0							

OW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	A1								A0							

15. OM70 PLC security

6 relay outputs (changeover contacts)

with PLC security and control LEDs

Inputs:	
Outputs:	6
Input words:	
Output words:	1
Switch positions:	1 to 9
Sum of inputs:	
Sum of outputs:	56
Special features:	Relay, Changeover contacts
Module identification:	Yes
Module check:	Yes
Status information (additional input words):	

Allocation of the output words:

OW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0											K5	K4	K3	K2	K1	K0

16. **OM70**

6 relay outputs (changeover contacts)

with indicator LEDs

Inputs:	
Outputs:	6
Input words:	
Output words:	1
Switch positions:	1 to 9
Sum of inputs:	
Sum of outputs:	56
Special features:	Relay, Changeover contacts
Module identification:	
Module check:	
Status information (additional input words):	

Assignment of output words:

OW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0											K5	K4	K3	K2	K1	K0

17. OM74 PLC security

16 relay outputs (break contacts)

with PLC security and control-LEDs

Inputs:	
Outputs:	16
Input words:	
Output words:	1
Switch positions:	1 to F
Sum of inputs:	
Sum of outputs:	240
Special features:	Reed relay, break contacts
Module identification:	Yes
Module check:	Yes
Status information (additional input words):	

Assignment of output words:

OW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	K1								K0							

18. OM74

16 relay outputs (break contacts)

with indicator LEDs

Inputs:	
Outputs:	16
Input words:	
Output words:	1
Switch positions:	1 to F
Sum of inputs:	
Sum of outputs:	240
Special features:	Reed relay, break contacts
Module identification:	
Module check:	
Status information (additional input words):	

Assignment of output words:

OW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	K1								K0							

19. OM75 PLC security

16 relay outputs (break contacts)

with PLC security and control-LEDs

max. switching current per relay contact 2 A

Inputs:	
Outputs:	16
Input words:	
Output words:	1
Switch positions:	1 to F
Sum of inputs:	
Sum of outputs:	240
Special features:	Reed relay, break contacts
Module identification:	Yes
Module check:	Yes
Status information (additional input words):	

Assignment of output words:

OW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	K1								K0							

20. OM75

16 relay outputs (break contacts)

with indicator LEDs

max. switching current per relay contact 2 A

Inputs:	
Outputs:	16
Input words:	
Output words:	1
Switch positions:	1 to F
Sum of inputs:	
Sum of outputs:	240
Special features:	Reed relay, break contacts
Module identification:	
Module check:	
Status information (additional input words):	

Assignment of output words:

OW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	K1								K0							

21. TM10 PLC security

16 digital outputs (TTL),

16 digital inputs (TTL),

with PLC security

Inputs:	16
Outputs:	16
Input words:	1
Output words:	1
Switch positions:	1 to 9
Sum of inputs:	144
Sum of outputs:	144
Special features:	TTL (5V)
Module identification:	Yes
Module check:	Yes
Status information (additional input words):	

Assignment of input and output words:

IW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	E1								E0							

OW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	A1								A0							

22. TM10

16 digital outputs (TTL),

16 digital inputs (TTL),

Inputs:	16
Outputs:	16
Input words:	1
Output words:	1
Switch positions:	1 to 9
Sum of inputs:	144
Sum of outputs:	144
Special features:	TTL (5V)
Module identification:	
Module check:	
Status information (additional input words):	

Assignment of input and output words:

IW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	E1								E0							

OW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	A1								A0							

23. AS11

Universal display/switch-module
with 8 pusbuttons and 8 LEDs

Inputs:	8
Outputs:	8
Input words:	1
Output words:	1
Switch positions:	1 to 6
Sum of inputs:	48
Sum of outputs:	48
Special features:	Pusbuttons and LEDs only
Module identification:	

Module check:	
Status information (additional input words):	

Assignment of input and output words:

IW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		ESC		7		6		5		4		3		2		1

OW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		ESC		7		6		5		4		3		2		1

24. AS12

Universal display/switch-module

with 32 LEDs (16 red, 16 green)

Inputs:	
Outputs:	32
Input words:	
Output words:	2
switch positions:	1 to 5
Sum of inputs:	
Sum of outputs:	160
Special features:	LEDs only
Module identification:	
Module check:	
Status information (additional input words):	

Assignment of output words:

OW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	A (green)															
1	B (red)															

25. AS13

Universal display/switch-module

with 16 switches (E-M-E)

Inputs:	32
Outputs:	
Input words:	2
Output words:	
Switch positions:	1 to 5
Sum of inputs:	160

Sum of outputs:	
Special features:	16 switches (E-M-E)
Module identification:	
Module check:	
Status information (additional input words):	

Assignment of input words:

IW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	A															
1	B															

26. AS14

Universal display/switch-module

with 32 LEDs (green)

Inputs:	
Outputs:	32
Input words:	
Output words:	2
Switch positions:	1 to 5
Sum of inputs:	
Sum of outputs:	160
Special features:	LEDs only
Module identification:	
Module check:	
Status information (additional input words):	

Assignment of output words:

OW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	A															
1	B															

27. Address assignment matrix

The following tables are very helpful for planning addressing of the individual modules.

The columns show the I/O modules, the rows show the processor I/O addresses. The module address is set by means of the rotary switch on the underside of the module. Depending on the module and switch position, the range shown for the switch position is assigned. All addresses are shown in hexadecimal notation.

The same switch position may occur for different modules, but any single processor I/O address can only be assigned once. This means that a module assigns all table rows associated with a switch position. There is a separate address range for inputs and outputs.

The FST IPC I/O configurer supports you in assigning addresses / switch positions. You would always begin with the module that takes up the fewest I/O addresses, as it is then easier to find space for the remaining modules.

	AS11
120-123	1
220-223	2
320-323	3
1A0-1A3	4
2A0-2A3	5
3A0-3A3	6

Table 2: Address assignment digital output modules part I

	OM10	OM11 + OM40	OM12	OM74 + OM75	OM20 + OM70	OM21	OM22	TM10	AS12	AS13
110	1	1	1	1	1	1	1	1	1	1
111					2	2				
112	2	2		2	3	3	2	2		
113					4	4				
210	3	3	2	3	5	5	3	3	2	2
211					6	6				
212	4	4		4	7	7	4	4		
213					8	8				
310	5	5	3	5	9		5	5	3	3
311										
312	6	6		6			6	6		
313										
170	7	7	4	7			7	7	4	4
171										
172	8	8		8			8	8		
173										
270	9		5	9			9	9	5	5
271										
272				A			A			
273										
370			6	B			B			
371										
372				C			C			
373										
114			7	D			D			

115										
116				E			E			
117										
214			8+	F			F			
215										

Table 3: Address assignment digital output modules part II

	AS11
120-123	1
220-223	2
320-323	3
1A0-1A3	4
2A0-2A3	5
3A0-3A3	6

Table 4: Address assignment digital input modules part I

	IM10	IM11	IM12	OM20	OM21	OM22	TM10	AS13
110	1	1	1	1	1	1	1	1
111				2	2			
112	2	2		3	3	2	2	
113				4	4			
210	3	3	2	5	5	3	3	2
211				6	6			
212	4	4		7	7	4	4	
213				8	8			
310	5	5	3	9		5	5	3
311								
312	6	6				6	6	
313								
170	7	7	4			7	7	4
171								
172	8	8				8	8	
173								
270	9	9	5+			9	9	5
271								
272						A+		
273								

Table 5: Address assignment digital input modules part II

3. Analog input and output modules

The following table gives an overview of available analog modules. These are described in the following sections.

Module	Features	Range(s)	Resolution	PLC security
IO10	8 analog inputs	0 to 4.096V	12-bit	-
IO11	8 analog inputs	0 to 20mA	12-bit	-
IO12	8 analog inputs	0 to 10V	12-bit	-
IO40	4 analog inputs	-10V to +10V 0 to 10V -5V to +5V 0 to +5V	12-bit	*/-/-
IO41	4 analog inputs	-10V to +10V 0 to 10V -5V to +5V	16-bit	*/-/-
IO48	4 analog inputs	0 to 20mA 4 to 20mA	12-bit	*/-/-
IO60	4 analog outputs	-10V to +10V	12-bit	-
IO61	4 analog outputs	-10V to +10V	12-bit	-
IO70	4 analog outputs	-10V to +10V 0 to 10V	12-bit	-
IO71	4 analog outputs	-10V to +10V 0 to 10V	12-bit	-
IO73	4 analog outputs	0 to 20mA 4 to 20mA	12-bit	-

Table 6: Overview of analog input and output modules

1. IO10

8 analog inputs, 0 to 4.096V, 12-bit

Input words (channels):	8
Output words (channels):	
Switch positions:	1 to 4
Sum of inputs:	32
Sum of outputs:	
Range:	0 to 4,096V
Resolution:	12-bit
Special features:	
Module identification:	
Module check:	

Assignment of input words:

IW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Register value	Physical value
0	0 mA
205	1 mA
1024	5 mA
2048	10 mA
3072	15 mA
4095	20 mA

3. IO12

8 analog inputs, 0 to 10V, 12-bit

Input words (channels):	8
Output words (channels):	
Switch positions:	1 to 4
Sum of inputs:	32
Sum of outputs:	
Range:	0 to 10V
Resolution:	12-bit
Special features:	
Module identification:	
Module check:	

Assignment of input words:

IW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					Channel #1											
1					Channel #2											
2					Channel #3											
3					Channel #4											
4					Channel #5											
5					Channel #6											
6					Channel #7											
7					Channel #8											

Register value	Physical value
0	0V
410	1V
819	2V
2048	5V
4095	10V

4. IO40 -10V to +10V

4 analog inputs, -10V to +10V , 12-bit

Input words (channels):	4
Output words (channels):	
Switch positions:	1 to 6
Sum of inputs:	24
sum of outputs:	
Range:	-10V to +10V
Resolution:	12-bit
Special features:	
Module identification:	Yes
Module check:	

Assignment of input words:

IW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					Channel #1											
1					Channel #2											
2					Channel #3											
3					Channel #4											

Register value	Physical value
-2048	-10 V
-1024	-5 V
0	0 V
205	1 V
1023	5 V
2047	10 V

5. IO40 0V to 10V

4 analog inputs, 0V to 10V , 12-bit

Input words (channels):	4
Output words (channels):	
Switch positions:	1 to 6
Sum of inputs:	24
Sum of outputs:	
Range:	0V to 10V
Resolution:	12-bit
Special features:	
Module identification:	Yes
Module check:	

Assignment of input words:

IW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					Channel #1											
1					Channel #2											
2					Channel #3											
3					Channel #4											

Register value	Physical value
0	0 V
410	1 V
819	2 V
2048	5 V
4095	10 V

6. IO40 -5V to +5V

4 analog inputs, -5V to +5V , 12-bit

Input words (channels):	4
Output words (channels):	
Switch positions:	1 to 6
Sum of inputs:	24
Sum of outputs:	
Range:	-5V to +5V
Resolution:	12-bit
Special features:	
Module identification:	Yes
Module check:	

Assignment of input words:

IW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					Channel #1											
1					Channel #2											
2					Channel #3											
3					Channel #4											

Register value	Physical value
-2048	-5.0 V
-1024	-2.5 V
0	0.0 V
205	1.0 V
1023	2.5 V
2047	5.0 V

7. IO40 0V to 5V

4 analog inputs, 0V to 5V , 12-bit

Input words (channels):	4
Output words (channels):	
Switch positions:	1 to 6
Sum of inputs:	24
Sum of outputs:	
Range:	0V to +5V
Resolution:	12-bit
Special features:	
Module identification:	Yes
Module check:	

Assignment of input words:

IW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					Channel #1											
1					Channel #2											
2					Channel #3											
3					Channel #4											

Register value	Physical value
0	0.0 V
410	0.5 V
819	1.0 V
1638	2.0 V
2048	2.5 V
4095	5.0 V

8. IO41 -10V to +10V

4 analog inputs, -10V to +10V , 16-bit

Input words (channels):	4
Output words (channels):	
Switch positions:	1 to 6
Sum of inputs:	24
Sum of outputs:	
Range:	-10V to +10V
Resolution:	16-bit
Special features:	
Module identification:	Yes
Module check:	

Assignment of input words:

IW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Channel #1															
1	Channel #2															
2	Channel #3															
3	Channel #4															

Register value	Physical value
-32768	-10 V
-16384	-5 V
0	0 V
3279	1 V
16383	5 V
32787	10 V

9. IO41 0V to 10V

4 analog inputs, 0V to 10V , 16-bit

Input words (channels):	4
Output words (channels):	
Switch positions:	1 to 6
Sum of inputs:	24
Sum of outputs:	
Range:	0V to 10V
Resolution:	16-bit
Special features:	
Module identification:	Yes
Module check:	

Assignment of input words:

IW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Channel #1															
1	Channel #2															
2	Channel #3															
3	Channel #4															

Register value	Physical value
0	0 V
6554	1 V
13107	2 V
32768	5 V
65535	10 V

10. IO41 -5V to +5V

4 analog inputs, -5V to +5V , 16-bit

Input words (channels):	4
Output words (channels):	
Switch positions:	1 to 6
Sum of inputs:	24
Sum of outputs:	
Range:	-5V to +5V
Resolution:	16-bit
Special features:	
Module identification:	Yes
Module check:	

Assignment of input words:

IW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Channel #1															
1	Channel #2															
2	Channel #3															
3	Channel #4															

Register value	Physical value
-32768	-5.0 V
-16384	-2.5 V
0	0.0 V
6554	1.0 V
16383	2.5 V
32767	5.0 V

11. IO48 0 to 20mA

4 analog inputs, 0 to 20mA , 12-bit

Input words (channels):	4
Output words (channels):	
Switch positions:	1 to 6
Sum of inputs:	24
Sum of outputs:	
Range:	0 to 20mA
Resolution:	12-bit
Special features:	
Module identification:	Yes
Module check:	

Assignment of input words:

IW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

0					Channel #1
1					Channel #2
2					Channel #3
3					Channel #4

Register value	Physical value
0	0 mA
204	1 mA
819	4 mA
1024	5 mA
2048	10 mA
4095	20 mA

12. IO48 4mA to 20mA

4 analog inputs, 4mA to 20mA , 12-bit

Input words (channels):	4
Output words (channels):	
Switch positions:	1 to 6
Sum of inputs:	24
Sum of outputs:	
Range:	4mA to 20mA
Resolution:	12-bit
Special features:	
Module identification:	Yes
Module check:	

Assignment of input words:

IW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																Channel #1
1																Channel #2
2																Channel #3
3																Channel #4

Register value	Physical value
0	4 mA
1024	8 mA
2048	12 mA
3072	16 mA
4095	20 mA

13. IO60/IO61 -10V to +10V

4 analog outputs, 0V to 10V, 12-bit

Input words (channels):	
Output words (channels):	4
Switch positions:	0 to 4
Sum of inputs:	
Sum of outputs:	20
Range:	-10V to 10V
Resolution:	12-bit
Special features:	
Module identification:	
Module check:	

Assignment of output words:

OW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					Channel #1											
1					Channel #2											
2					Channel #3											
3					Channel #4											

Register value	Physical value
-2048	-10V
-1024	-5 V
0	0 V
1024	5 V
2047	10 V

14. IO70/IO71 0V to 10V

2 analog outputs, 0V to 10V , 12-bit

Input words (channels):	
Output words (channels):	2
Switch positions:	0 to 4
Sum of inputs:	
Sum of outputs:	10
Range:	0V to 10V
Resolution:	12-bit
Special features:	
Module identification:	
Module check:	

Assignment of output words:

OW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

0					Channel #1
1					Channel #2

Register value	Physical value
0	0 V
410	1 V
819	2 V
2048	5 V
4095	10 V

15. IO70/IO71 -10V to +10V

2 analog outputs, 0V to 10V , 12-bit

Input words (channels):	
Output words (channels):	2
Switch positions:	0 to 4
Sum of inputs:	
Sum of outputs:	10
Range:	-10V to 10V
Resolution:	12-bit
Special features:	
Module identification:	
Module check:	

Assignment of output words:

OW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																
1																

Register value	Physical value
-2048	-10V
-1024	-5 V
0	0 V
1024	5 V
2047	10 V

16. IO73 0 to 20mA

2 analog outputs, 0 to 20mA, 12-bit

Input words (channels):	
Output words (channels):	2
Switch positions:	0 to 4

Sum of inputs:	
Sum of outputs:	10
Range:	0 to 20mA
Resolution:	12-bit
Special features:	
Module identification:	
Module check:	

Assignment of output words:

OW	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					Channel #1											
1					Channel #2											

Register value	Physical value
0	0 mA
205	1 mA
819	4 mA
2048	10 mA
4095	20 mA

17. Address assignment matrix

The following tables are very helpful for planning addressing of the individual modules.

The columns show the I/O modules, the rows show the processor I/O addresses. The module address is set by means of the rotary switch on the underside of the module. Depending on the module and switch position, the range shown for the switch position is assigned. All addresses are shown in hexadecimal notation.

The same switch position may occur for different modules, but any single processor I/O address can only be assigned once. This means that a module assigns all table rows associated with a switch position. There is a separate address range for inputs and outputs.

The FST IPC I/O configurer supports you in assigning addresses / switch positions. You would always begin with the module that takes up the fewest I/O addresses, as it is then easier to find space for the remaining modules.

	IO10	IO11	IO12
140-15F	1	1	1
240-25F	2	2	2
1C0-1DF	3	3	3
2C0-2DF	4	4	4

Table 7: Address assignment analog outputs part I

	IO40	IO41	IO48
1A4	1	1	1
1A5			

2A4	2	2	2
2A5			
3A4	3	3	3
3A5			
1A6	4	4	4
1A7			
2A6	5	5	5
2A7			
3A6	6	6	6
3A7			

Table 8: Address assignment analog outputs part II

	IO60	IO61	IO70	IO71	IO73
300-303	0	0	0	0	0
304-307			1	1	1
130-133	1	1	2	2	2
134-137			3	3	3
230-233	2	2	4	4	4
234-237					
1B0-1B3	3	3			
1B4-1B7					
2B0-2B3	4	4			
2B4-2B7					

Table 9: Address assignment analog outputs part III

	IO10	IO11	IO12
140-15F	1	1	1
240-25F	2	2	2
1C0-1DF	3	3	3
2C0-2DF	4	4	4

Table 10: Address assignment analog inputs part I

	IO40	IO41	IO48
1A4	1	1	1
1A5			
2A4	2	2	2

2A5			
3A4	3	3	3
3A5			
1A6	4	4	4
1A7			
2A6	5	5	5
2A7			
3A6	6	6	6
3A7			

Table 11: Address assignment analog inputs part II

	IO60	IO61	IO70	IO71	IO73
300	0	0	0	0	0
304			1	1	1
130	1	1	2	2	2
134			3	3	3
230	2	2	4	4	4
1B0	3	3			
2B0	4	4			

Table 12: Address assignment analog inputs part III

FEC and HC0x

The special IO drivers for FEC and HC0x controllers are described in this topic.

Local IOs

FEC Standard

In the IO configuration, select the module that corresponds with your hardware, e.g. **FC440**. The inputs of the FEC Standard are grouped into FST input and output words. If you select the "word" option that is available for each FEC Standard type, 16 inputs or outputs are accessible through each FST input or output word. If you select the default IO script (without "word"), 8 inputs or outputs are accessible through the 8 least significant bits each FST input or output word.

For the analog IO separate modules must be selected: **Standard Analog Input 0-20mA** and **Standard Analog Output 0-20mA**.

PLC Safety is available through a separate module: **FEC Standard PLC Safety**. This module will generate FST error 12 if the hardware cannot be identified as a FEC Standard. Error 11 will be generated if no supply voltage is present on the outputs or an overload occurs on an output. The reason will be made available through the FST input word you specified when including the IO module.

```
BIT 0 = A0 overload
BIT 1 = A1 overload
BIT 2 = nc
BIT 3 = nc
BIT 4 = A0 no 24V
BIT 5 = A1 no 24V
```

Note! The input word reflects the actual status, to be sure that the reason is available later you should write an [error program](#).

FEC Compact

In the IO configuration, select the module **FEC**. You do not have to enter a switch position here. Enter the number of the FST input and output words that are to be used for the local IOs (e.g. both 0).

The 12 inputs of the FEC Compact are divided into two groups. 8 on the first, 4 on the second group. The 8 bits of the first group are represented in the 8 least significant bits of the input word given (e.g. I0.0 to I0.7). The 4 bits of the second group are represented in the 4 least significant bits of the following input word (e.g. I1.0 to I1.3).

The 8 outputs are represented in the 8 least significant bits of the output word given (e.g. O0.0 to O0.7).

HC0X

The 8 IOs of HC0X can be used as inputs or outputs.

Enter the module **HC0X** into the IO configuration. You do not have to enter a switch position here. Enter the number of the FST input and output words that are to be used for the local IOs (e.g. both 0). This will allocate one input and two output words.

The 8 IO bits are represented as the 8 least significant bits of input and output word given (e.g. I0.0 to I0.7, O0.0 to O0.7). The 8 least significant bits of the second output word (e.g. OW1) is used to define which pins are used as input or output: if a bit is set this means that the corresponding pin is an output, otherwise it is an input.

Debouncing of Inputs

The inputs have a delay time of 5 ms.

An additional delay can be achieved by using the logical IO module **DEBOUNCE**. Enter it into the IO configuration but behind the definition of the input word you want to debounce. The complete IW you specify will be debounced. The status of its bits will only change if the new value has been read for two consecutive IO scans. You can increase the number of consecutive IO scans by entering **DEBOUNCE** for several times for the same input word.

IO extension

HC0x and FEC Compact can be connected to an (additional) FEC via the EXT interface and a special (short!) cable.

Note! For the FEC Standard series the IO extension is not available.

If you want to have more than one extension module, use the IO extension driver which is described in a separate chapter.

To access the remote IO from your programs you have to enter the module **Remote FEC** (HC0X) or **FEC Slave** (FEC Compact) into the IO configuration. You have to give the number of the FST input and output words (e.g. 2 for both).

The 12 inputs of the remote FEC are divided into two groups: 8 on the first, 4 on the second group. The 8 bits of the first group are represented in the 8 least significant bits of the input word given (e.g. I2.0 to I2.7). The 4 bits of the second group are represented in the 4 least significant bits of the following input word (e.g. I3.0 to I3.3).

The 8 outputs are represented in the 8 least significant bits of the output word given (e.g. O2.0 to O2.7).

The remote FEC must not have a control program running. The FEC must be in the factory settings. Delete any files from drive B: using the File transfer utility of FST IPC.

The RUN/STOP switch must be in position STOP.

FST Error 99 will occur if a FEC has local IOs configured and is also accessed as remote FEC.

Analogue Potentiometer

FEC Compact and HC01 are equipped with an analogue potentiometer. It can be adjusted using a screwdriver in the range 1 to 63.

To read the values from your programs enter the module **Trimmer** into your IO configuration and specify the input word you want to use for the actual trimmer value.

Rotary Switch

FEC Standard controllers are equipped with a rotary switch with 16 positions (0-15) which also serves as run/stop switch, where 0 means stop and all other positions run.

To read the switch position (1-15) from your programs enter the module **Rotary Switch** into your IO configuration and specify the input word you want to use for the actual switch position.

For compatibility reasons for the FEC Standard the module **Trimmer** is still available. The switch positions will result in the following values:

0	1
1	1
2	5
3	8
4	12
5	16
6	21
7	26
8	32
9	36
10	41
11	46
12	50
13	54
14	59
15	63

Fast counters

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 HC0x (I0.0 and I0.1) can be used as 1 or 2 independent, high-speed counters. Still, the inputs can be read as standard inputs.

For the FEC Standard series the following inputs are used:

FC4xx, FC5xx - I1.6 and I1.7
FC6xx - I3.6 and I3.7

These counters are interrupt driven and once activated, operate independently of the user's control programs and therefore are not effected by factors such as control program scan time.

Include the module **Fast counter** to your IO configuration. Enter 0 as switch if you are using counter 0, 1 for counter 1. Specify the input word you want to hold the counter value.

The counter value is incremented on each rising edge of the corresponding input signal.

Please note that the counter values cannot be reset. If this is required the driver described in a separate chapter can be used instead.

Incremental Encoder

The first two inputs of the second group of the FEC Compact (I1.0 and I1.1) as well as the inputs I0.2 and I0.3 of HC0x can be used as incremental encoder up to 200Hz. Optionally a reset signal can be connected to I0.7. Still, the inputs can be read as standard inputs.

For the FEC Standard series the following inputs are used:

FC4xx, FC5xx - I1.4, I1.5 and I1.70
FC6xx - I3.4, I3.5 and I3.0.

This encoder is interrupt driven and once activated, operates independently of the user's control programs and therefore are not effected by factors such as control program scan time.

Include the module **Incremental Encoder** to your IO configuration and specify the input word for the counter value (e.g. IW5). The following input word will hold the direction (e.g. IW6).

ABRESET

To reset the counter value call the module ABRESET.

Input parameters	none
Output parameters	none

ABMODE

Pin I0.7/I1.0/I3.0 can be used to reset the counter. The counter will be reset if I0.7/I1.0/I3.0 is active and a positive edge is detected on one of the A/B inputs. You can choose between the following modes:

Input parameters	FU32	reference/reset mode:
		0 no reset
		1 direction 0, single
		3 direction 0, continuous
		5 direction 1, single
		7 direction 1, continuous
Output parameters	none	

Actuator Sensor Interface

The Actuator Sensor Interface (ASi) connects sensors and actuators quickly and cheaply to the IPC-PS1. Up to 4 CP96 AS-Interface modules can be used, each with up to 31 slaves.

The inputs and outputs of the slaves are copied to the user defined local function units of the FST IPC.

Selecting and parameterising the driver

If you want to use the AS-Interface in a FST IPC project, you first have to add and parameterise the ASi driver in the driver configuration.

Target drive:

Enter the drive that contains the AS-Interface driver ASIDRV.EXE or to which it is to be loaded.

The ASi configurer

Start the configurer in the FST software via the Extras > AS-Interface configuration from the menu.

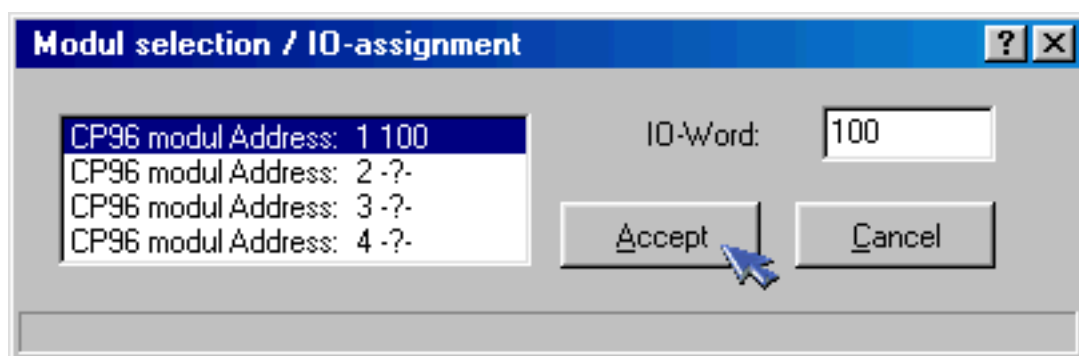
Note! In order to be able to use the online features of the configuration tool, the AS-Interface driver must be active on the IPC-PS1. It is sufficient to load (into the controller) the project for which you have configured the ASi driver, without programs. The IPC must then be rebooted. This will be done automatically.

Selecting the master

Selecting Edit > Modul Number/IO-assignment from the menu opens a selection window allowing you to choose one of 4 masters. The number of the master corresponds to the address set on the CP96 module.

This selection window also defines the base address for the I/O image to which the ASi slave/I/Os are copied. This address must be given for correct configuration of the master module. A CP96 module (ASi master) always occupies 8 I/O words in the local I/O area of the FST IPC.

Figure 1: Selecting the Master module and assigning an I/O address



Configuring individual stations

The main function of the configurer is the configuration of the individual ASi bus stations (slaves). Here, configuration means setting the ID and I/O codes of a slave. Configuration can be done offline, that is, without a link to the IPC. If there are several ASi masters in one IPC system, this procedure has to be repeated for each master. The selection of the desired master is done by Edit > Modul Number/IO-assignment from the menu.

At runtime – during the initialisation phase – the configuration data of each master is compared with the data in the reference list. A master is only reconfigured if the data does not match. This approach prevents a relatively long delay when a project is restarted (as would be caused by reprogramming of the master lists). This delay only occurs the first time a project is started, when the configuration is changed, and when a master is swapped.

Selecting Edit > Configuration from the menu shows the configured slaves from 1 to 31 are shown with I/O address, ID, I/O code and parameters. Any slave can be selected for further processing with the cursor keys or the mouse.

Edit

Double clicking on a slave or selecting Edit from the right mousebutton popup menu opens an input window allowing editing of the ID code, I/O code and parameters of the selected slave. ID and I/O code should be taken from the appropriate documentation for the slave. Entry of a parameter is optional. This parameter (0 to F) is transmitted to the slave when the master is restarted. When slaves are designed accordingly, such parameters can be used to set certain properties (such as the sensitivity of an ultrasound sensor).

Figure 2: Entering a slave into the configuration

Slave-Adr.	IO-Adr.	ID-Code	IO-Code	4	3	2	1	Para-meter
1	100.04	0	0	1	1	1	1	F

OK Cancel

Range 0..F

Delete

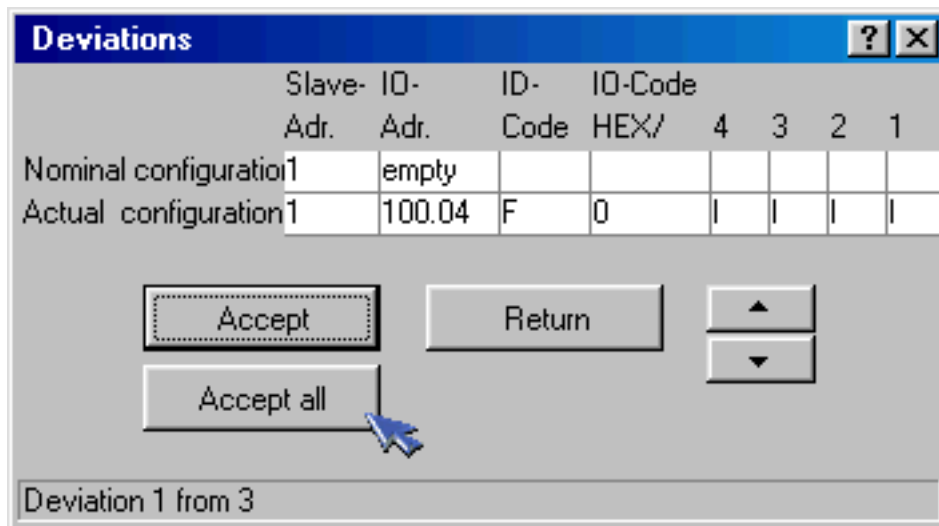
Pressing the Delete key allows you to delete the selected slave from the nominal list.

Nominal-actual comparison


When selecting Nominal/actual compare from the right mousebutton popup menu a comparison of the slaves in the nominal list with the slaves present on the ASi bus is started. All deviations are displayed in a dialog. If the nominal list is empty – for example in the case of a new project – all slaves on the bus are inserted into the nominal list. Naturally, the reference list can be edited in the normal way.

There must be a connection to the IPC. The ASi driver must be installed. The IPC main runtime program must be active.

Figure 3: Result of the nominal-actual comparison



Printing the configuration

Selecting File > Print from the menu or pressing  prints the configuration.

Address programming of individual slaves

Note! There must be a link to the IPC. The ASi driver must be installed. The IPC main runtime program must be active.

By default, all ASi slaves have the address 0. Each ASi slave can be assigned a new address by address programming. This can also be done for slaves that already have an address other than 0. Each ASi slave must have a unique address (per master). The addresses can be modified by first selecting Edit > Slave Addresses and then from the list use drag and drop to change the slave.

Online manipulation of individual slaves

Note! There must be a link to the IPC. The ASi driver must be installed. The IPC main runtime program must be active.

The slaves detected on the bus are displayed with their I/Os in the ASi online display. The display is dynamic. In other words, the display is automatically updated to reflect the status of the inputs.

Manipulating slave outputs

First position the selection highlight on the slave whose outputs are to be manipulated. Now it possible to manipulate the outputs by doubleclicking on the output, by pressing keys F5 to F8, or by using the right mousebutton popup menu.

CI commands

```
!7DLES          Display list of current slaves

!7DSxx          Display slave (IO, ID, +inputs)
                 (xx = 1 to 1F)

!7DOUF          Display control flags

!7DHARD         Display hard/soft reaction
```

	(0 = soft, 1 = hard)
!7MMA	Modify Master Address (= 1 to 4) (Valid for all following commands).
!7MSA	Modify Slave Address (= old 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 runtime error)

All numeric values given for the Modify command must be entered in hex. The command is not case sensitive. The commands are used in the first line of the ASi configurer. The commands are not intended for manual entry. They are only documented here for the sake of completeness.

Error numbers

No	Meaning of error Notes on elimination
700	No configuration data found for ASi driver
701	No master found
702	
703	
704	
711	Slave failure
712	
713	
714	

The cyclical part of the driver monitors the Config OK flags of all configured masters. If "hard" error reaction is set, the failure of a slave results in error number 710 + master address (1 – 4) being entered into the IPC error word. Program execution is stopped. If an error handling routine has been entered into the IPC configuration, this is now started. If this behavior is not desired, hard error reaction can be switched off with the AS-I_Mode module (see below).

Function modules

Overview

ASI_Mode	Sets the reaction of the ASi driver to configuration errors
ASI_Stat	Interrogates execution control level (OUF) flags

ASI_Para Transfer a parameter to an
 ASi slave at runtime

ASI_Res Restart cyclic update

ASI_Mode

Sets the reaction of the AS-I driver to configuration errors

Input parameters	FU32	= 0	Soft reaction
		> 0	Hard reaction
Output parameters	FU32	= -1	OK
		= 0	Driver not loaded

Note: The ASI_Mode module can only be used with the ASi driver.

ASI_Stat

Interrogates execution control level (OUF) flags

Input parameters	FU32	Master address (1 to 4)
Output parameters	FU32	= -1 OK
		= 1 Master address invalid
		= 10 Master not found
	FU33	Status flags

For the significance of the individual bits, please refer to the ASi master documentation. If hard error reaction has been switched off with the ASi module, the user can use ASI_Stat to interrogate the Config OK flag of the individual master and derive the desired reaction to the error from the result.

AS-I_Para

Transfer a parameter to an ASi slave at runtime

Input parameters	FU32	Master address (1 to 4)
	FU33	Slave address (1 to 31)
	FU34	Parameter value (0 to \$0F)
Output parameters	FU32	= -1 OK
		= 1 Master address invalid
		= 2 Slave address invalid
		= 3 Parameter value invalid
		= 10 Master not found
		= 11 Master timeout
	FU33	Error code

Overview of possible error codes:

HFUN_OK	0x00	Execution successful
HFUN_NOK	0x01	Execution not successful
HFUN_SNA	0x02	Slave not in LAS
HFUN_MOFF	0x0C	Master is offline
MAS-IBUSY	0xFD	Master not ready
HOSTTOUT	0xFE	Master timeout

Note: If the slave is not found, error code 0x02 is returned (Slave not in LAS).

ASI_Res

Restart cyclical update (including error monitoring) following a runtime error (slave failure)

Input parameters	None	
Output parameters	FU32	= -1 OK

= 0 Driver not loaded

Note: The ASI_Res module can only be used with the ASi driver.

Festo Fieldbus Master

Version 2.22

The fieldbus allows you to link up distant I/O modules (plant-floor units) with your IPC-PS1 to form a system network. Communication between the modules and the IPC is controlled by the PS1-CP61 fieldbus module. This module has a fieldbus interface to which you can connect up to 31 participants, when using repeater technology up to 99. Additional FST operands are available for cyclical inputs and outputs of the participants. Modules can be used to interrogate statuses and execute a-cyclic commands.

Set Festo fieldbus parameters

If you want to use the FESTO fieldbus in a project, you must enter and parameterise the FESTOBUS driver in the configurator.

Destination Drive:

Enter here the target drive for the Festo fieldbus driver IPCFB22.EXE.

CP61 switch setting:

Enter the switch position here which you have set on the rear of your fieldbus module (PS1-CP61).

Interrupt number:

Enter the number of the interrupt that the fieldbus module is allowed to use for communication with the CPU. Check which interrupts have already been assigned for other modules.

fieldbus baud rate:

Here you can specify the baud rate for data transmission on the fieldbus. The permissible values are 375, 187.5, 62.5 and 31.25 kilobaud.

Highest station address:

Enter here the highest participant number you are using. the permissible range is 1..99.

Soft error behaviour:

Enter here if you want to use the soft error behaviour. Valid entries are Yes and No.

Use nominal configuration:

Enter here if you want to use the nominal configuration. Valid entries are Yes and No.

Soft error behaviour

Programmers can choose between two options for the response to configuration problems during start-up:

The default is the "hard" error response. In the event of errors during start-up, neither Program 0 nor the error program is started.

Optionally, "soft" error response can be selected. In the event of errors during start-up, the response is as for other errors, that is, setting of the error word, and project stop or error program execution.

The desired error response can be set in the configuration dialog for the fieldbus (see above). Hard error response is the default setting.

Using the nominal configuration

Alternatively, the programmer can define the slave list to the FB master in the form of a nominal configuration. If a configured participant is not found during start-up, normal error handling for transmission errors result. The missing participant can then join the fieldbus at a later stage without reinitialisation of the bus.

Please note that only those participants can be entered that are also used in one of the loaded programs. It is immaterial whether the program is ever executed or not.

The individual participants must be used with their full scope of I/Os, as otherwise the participant is not addressed.

If the user reinitialises with CFM48,1 or CI, the Actual configuration is used as usual.

The Festo fieldbus configurer

The planning and control instrument for program generation is the fieldbus configurer. You use this to define the (nominal) configuration of the fieldbus.

You can then:

- Print out the configuration data that you have entered and connect the participants with the aid of the list
- Compare the actual and nominal configurations in order to correct connection errors.

You use the fieldbus configurer to create the nominal configuration of the fieldbus. You should do this before you enter the allocation list (and the programs) so that you can immediately assign symbolic identifiers to the operands that are created.

Note! As part of the syntax test carried out on the programs there is a check as to whether the fieldbus operands that have been entered have a corresponding entry in the fieldbus configuration.

In the following the fieldbus I/O modules are referred to as participants.

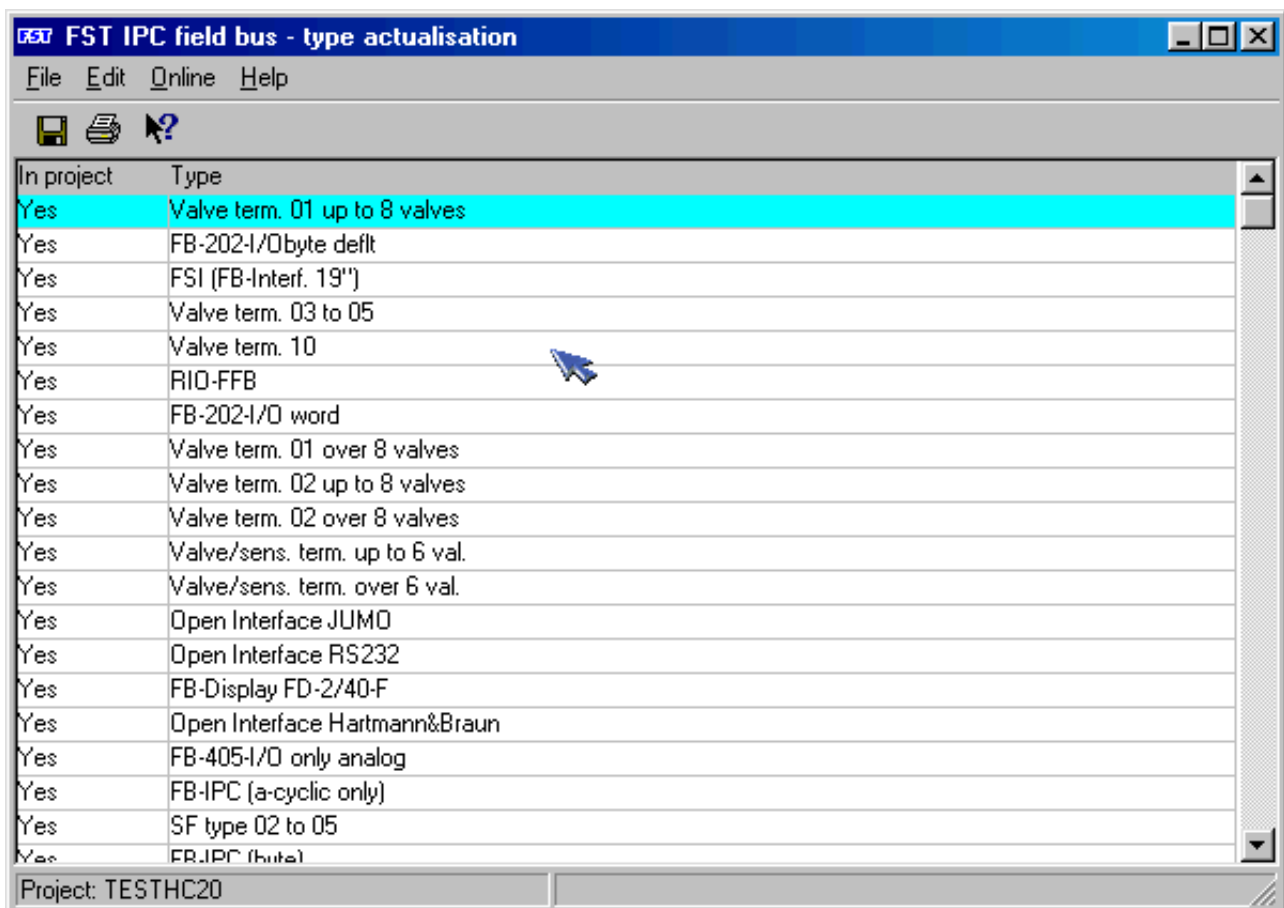
Calling the fieldbus configurer:

The FESTO Fieldbus configurer is configured as a [external tool](#), and can be activated in the Extras menu.

Selecting the type:

If you have not yet created a fieldbus configuration in the project, you must use Edit > Type actual. from the menu. A window appears showing you the types that are available from Festo. Here you can select the types that are to be used in your project for example by doubleclicking on them with the mouse.

Figure 1: Defining fieldbus participant types



Note! If you remove a type from the project-specific type file for which there is already an entry in the configuration, the designation Unknown type will subsequently appear in the configurer instead of the type. You should delete any such entries.

Edit configuration

Once you have completed type selection for your current project you can enter the participants by selecting Edit > Configuration from the menu.

After selecting Insert from the local menu a window appears where you configure the participant:

Figure 2: Configuring fieldbus participant

Pa.	Type	IW	OW	Comment
01				

OK Cancel

Range 1..99

Enter the participants number and type that you want to insert. The permissible entries are explained in the table below:

Abbr.	Meaning	Permissible entry
PA	Participant number	1 to 99
Type	Type of participant	The fieldbus participants stored in the type file are offered for selection in a window.
IW	Number of input units	It is only possible to specify the input units in the case of types where the number of inputs may vary (1 to 16).
OW	Number of output units	It is only possible to specify the output units in the case of types where the number of outputs may vary (1 to 16).

Conclude your entries by pressing the OK button.

The fieldbus participant is inserted in the configuration file in ascending order according to the participant number.

A message in the message shows the time in which the controller forwards I/O information to the fieldbus peripherals.

Modify:

Highlight the participant that you want to modify, and activate the Modify function by doubleclicking or selecting Modify from the local menu. As in the case of Insert, a window appears similar to that in Figure 2. The entries can then be overwritten.

Delete:

You can delete a fieldbus participant by selecting delete from the local menu.

Test functions

In order to ensure proper operation on the fieldbus, on the one hand various tests are carried out by the IPC run-time system while on the other hand the fieldbus configurer also offers a test option.

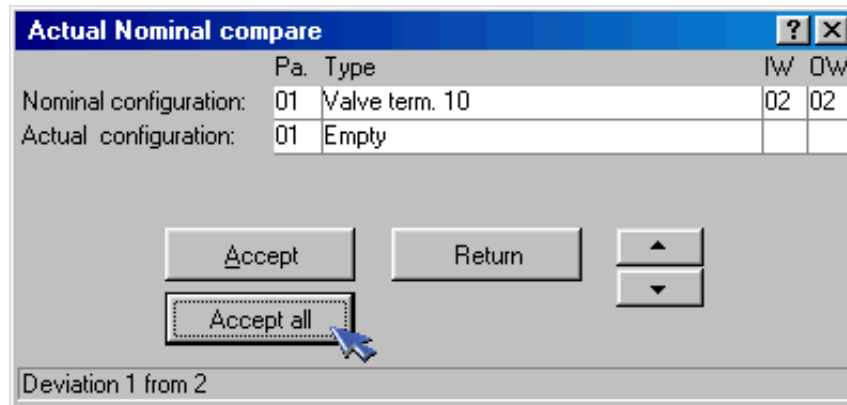
When a project is started, the run- time system detects the actual configuration and compares it with the nominal configuration. If the nominal configuration is not a subset of the actual configuration, an error is reported.

The Compare nominal/actual function supports the testing of the configuration data from the PC. To be able to execute this function, your PC must be connected to the IPC in the proper way.

Compare nominal/actual:

When you activate this function from the local menu the actual configuration is read from the IPC and is compared with the nominal configuration that was drawn up on the PC. If the configurations are identical, a message appears in the message line. If discrepancies occur during the comparison, the following window appears after you activate this function.

Figure 3: Nominal and actual comparison




You are shown the number of deviations between the nominal configuration and the actual configuration.

If there is no nominal configuration file in the project directory, the actual configuration that is read from the IPC is stored as the nominal configuration. The file contains no comments, however, so these details need to be added.

This function can be used for initial creation of the configuration file. The precondition for this, however, is that the fieldbus and all participants are available.

Print fieldbus configuration

Selecting File > Print from the menu or pressing  prints the configuration.

Programming fieldbus operands

fieldbus inputs and outputs can be addressed with the following syntax in Ladder diagram or Statement list programs:

```
Inputs (bits):      Ip.w.b
Outputs (bits):     Op.w.b

Input words:        IWp.w
Output words:       OWp.w

p   =   participant number
w   =   word number
b   =   bit number
```

The word number is assigned from 0 onwards for each participant, separately for inputs and outputs. For example, if you use the fieldbus module I/O extension 405 with 3 input cards and 2 output cards (1 word each) as participant 5, the following operands are created: I5.0.0 to I5.2.15 and IW5.0 to IW5.2, but also O5.0.0 to O5.1.15 and OW5.0 to OW5.1.

Most fieldbus modules occupy only one word; this is then word 0.

In the printout of the fieldbus configuration you will find an aid to addressing for each participant.

Function blocks

Overview

```
F40    Configuration of a fieldbus participant.
F41    Read parameter field.
F42    Write parameter field.
F43    Reset all cyclical outputs on the fieldbus.
F44    Status interrogation for a fieldbus participant.
F47    Set error handling.
F48    General configuration.
```

F40

Configuration of a fieldbus participant.

```
Input parameters
FU32    fieldbus participant number, 1 to 99
Output parameters
FU32    = -1    Nominal and actual data agree
         = 0    Nominal and actual data differ
FU33    Participant: actual type
```

FU34 Participant: number of inputs in bytes
 FU35 Participant: number of outputs in bytes

If you are using the nominal configuration (see Section 1.2) nominal and actual data will always agree by definition.

F41

Read parameter field of an “intelligent” participant.

Input parameters
 FU32 fieldbus participant number, 1 to 99
 FU33 Address of the participant word, 0 to 255
 Output parameters
 FU32 = -1 Command successfully processed
 = 0 Error, fieldbus participant not responding
 FU33 Participant status
 FU34 Participant word of address
 FU35 Participant word of address + 1
 FU36 Participant word of address + 2
 FU37 Participant word of address + 3

For a description of the participant status see F44.

F42

Write parameter field of an “intelligent” participant.

Input parameters
 FU32 fieldbus participant number, 1 to 99
 FU33 Number of word parameters, 1 to 4
 FU34 Address of participant word, 0 to 255
 FU35 Participant word for address
 FU36 Participant word for address + 1
 FU37 Participant word for address + 2
 FU38 Participant word for address + 3
 Output parameters
 FU32 = -1 Command successfully processed
 = 0 Error, fieldbus participant not responding
 FU33 Participant status

For a description of the participant status see F44.

F43

Reset all cyclical outputs.

Input parameters None
 Output parameters None

F44

Status interrogation for a participant.

Input parameters
 FU32 <> 0 Number of the participant
 = 0 Reset all error entries
 Output parameters
 FU32 Status of the participant
 FU33 number of short term errors
 FU34 number of long term errors
 FU35 total error count
 FU36 Participant address for last error

The status (FU32) contains the following information:

Bit 7 =1 ASCII data present
 Bit 6 error bits
 Bit 5 (see manual of field bus participant)
 Bit 4 .
 Bit 3 .
 Bit 2 .
 Bit 1 .
 Bit 0 =0 a-cyclic participant
 =1 cyclic participant

The number of short term errors is incremented in case of cyclic communication error (no response, checksum error etc.). The maximum value is 255. It will be reset to 0 after successful cyclic communication

Checking for broken communication can be done by using F44 and then testing for FU33 > some value The number of long term errors is incremented if communication had error (short term <> 0) and communication is restored

The total error count is incremented in case of cyclic communication error (no response, checksum error etc.)

F47

Status interrogation for a participant (FU32=1). Set exception handling programs (FU32=1).

```
Input parameters
FU32      = 1      Exception handling program assignment
FU33      Bit0 = 1 FU34 has the program number
           Bit1 = 1 FU35 has the program number
           Bit2 = 1 FU36 has the program number
FU34      Program number for transmission errors
FU35      Program number for errors in a field bus participant
FU36      Program number for ASCII data with field bus participant
Output parameters
None
```

All programs must always be given. If a non-existent or invalid program is given or if the bit is deleted in the mask, the program is deactivated for event handling.

Example:

```
THEN      CFM47
          WITH V1
          WITH V3
          WITH V3
          WITH V17
          WITH V0
```

In the event of a transmission error (such as cable break), Program 3 is set. If a participant error occurs (such as short-circuit at participant output), Program 17 is started. If a response to an a-cyclic command arrives (such as the participant is holding an ASCII file ready), no program is started.

The activated programs are executed in the normal sequence (that is, by increasing program number). Following completion of handling, they should deactivate themselves in order to allow themselves to be called again.

```
Input parameters
FU32      = 2      Status interrogation
Output parameters
FU32      Always 0
FU33      Status of the participant
FU34      Value of short-term error
FU35      Value of long-term error
FU36      Total number of errors
FU37      fieldbus address of the participant with last error
```

F48

Register actual configuration and status or use nominal configuration.

```
Input parameters
FU32      = 1      Configuration acquisition
           = 2      Status interrogation after
                     configuration acquisition
           = 3      use nominal configuration
Output parameters
if FU32 = 1      FU32      = -1      Configuration acquisition running

Output parameters
if FU32 = 2:      FU32      = 1      Configuration acquisition running
                   = 2      Actual configuration = nominal
                           configuration
                   = 3      Actual configuration <>
                           nominal configuration
                   = 4      No configuration available

Output parameters
if FU32 = 3      None
```

All fieldbus outputs are deleted prior to configuration acquisition. fieldbus inputs are not updated for a period of up to several seconds.

The use of the function with FU=2 does not make sense. When a participant list is given (see Section 1.2) actual and nominal data of this option match by definition.

Example program for F48:

```
STEP 1
THEN   CFM 48
       WITH K1      "Configuration acquisition

STEP 2
THEN   CFM 48
```

```

        WITH K2          "Status interrogation
IF      FU32
        <> K1
THEN    NOP

STEP 3
IF      FU32
        <> K2
THEN    ...

```

CI commands

The Festo fieldbus has the driver number 0. For historical reasons, "!0" can be entered instead of "\$".

```

!0FA      Assign nominal configuration

!0FC      Display participant number

!0FI      Reinitialisation
           (acquisition of actual configuration)
           Response "=1" for success, else "=0".

!0FN      Cyclic display of participant

```

Displays the definition of one participant per cycle. Output comprises participant number, type number, number of output bits and number of input bits.

```

!0FR      Start/continue cyclic transfer of data
!0FS      Stop cyclic transfer of data

```

Error numbers

No	Meaning of error Notes on elimination
14	Critical driver error Project cannot be started even with "soft" errors. This error occurs if the fieldbus driver was not found or problems occur starting the firmware of the CP61.
60	Actual configuration is not a superset of the nominal configuration. Correct fieldbus configuration or connect missing participant.

Festo Fieldbus Slave

This package allows you to operate the IPC as an intelligent slave in a Festo fieldbus.

Selecting the driver and assigning parameters

If you want to use the IPC to become a fieldbus slave, you must enter and parameterise the FBSLAVE driver in the configurator. The following entries are required:

Destination Drive:

Enter here the target drive for the Festo CP61 firmware FBSLAVE.BIN.

CP61 switch setting:

Enter the switch position here which you have set on the rear of your fieldbus module (CP61).

How to use the module FBSLAVE

Import the FBSLAVE file into your FST-IPC project. In the following examples I will assume that it was imported as a program module with number 0 (CMP 0).

Initialisation/configuration

First the firmware must know things like fieldbus address, baudrate etc.

Input parameters	FU32	1 = Initialisation/configuration
	FU33	Switch setting of CP61 (4..7)
	FU34	Baudrate (0..3) 0 = 31.25 Kbit/s 1 = 62.5 Kbit/s 2 = 187.5 Kbit/s 3 = 375 Kbit/s
	FU35	Participant number (slave address) (1..99)
	FU36	Output bytes (0..24) or words (0..12)
	FU37	Input bytes (0..24) or words (0..12)
	FU38	1 = Byte oriented 2 = Word oriented
	Output parameters	None

Repeated initialising has no effect, only after reloading the binary code (FBSLAVE.BIN) can the CP61 be reconfigured. You need to reboot the IPC to do so.

If inputs and outputs are mentioned I will use them as seen from the slave's perspective. So if in this document I use the word input, it means input as seen by the slave and output as seen by the master.

Example:

IF		NOP	
THEN	CMP 0		'FBSLAVE
	WITH	V1	" Initialise
	WITH	V4	" Switch setting CP61
	WITH	V3	" Baudrate (3=375 Kbit/s)
	WITH	V1	" Participant number (slave address)
	WITH	V24	" Inputs
	WITH	V24	" Outputs

WITH V1 " Byte oriented

From the master this slave will be seen as follows:

PA	Type	IW	OW
1	FB-IPC (byte)	24	24

Example:

```

IF
THEN  CMP 0
      WITH V1
      WITH V4
      WITH V3
      WITH V1
      WITH V8
      WITH V12
      WITH V2
      NOP
      'FBSLAVE
      " Initialise
      " Switch setting CP61
      " Baudrate (3=375 Kbit/s)
      " Participant number ( slave address )
      " Inputs
      " Outputs
      " Byte oriented

```

From the master this slave will be seen as follows:

PA	Type	IW	OW
1	FB-IPC (word)	8	12

Cyclic updates

The input and output data of the IPC fieldbus slave are kept in a flag word area defined by the user. The contents of these flags can be read to and from inputs and outputs or processed elsewhere with the help of an STL or LDR program.

To read/write the data which is cyclically updated to the master we use a range of flagwords which are read from/copied to the dual ported RAM of the CP61 by calling CFMxx as follows:

Input parameters	FU32	2 = Cyclic update
	FU33	1st flagword for outputs
	FU34	1st flagwords for inputs
Output parameters	None	

Example:

```

IF
THEN  CMP 0
      WITH V2
      WITH V10
      WITH V20
      NOP
      'FBSLAVE
      " Cyclic update
      " 1st flag word for outputs1
      " 1st flag word for inputs)

```

Get status

The communication status can be requested by calling CMPxx as follows:

Input parameters	FU32	0 = get status
Output parameters	FU32	status

The status is returned into FU32, and has the following meaning:

bit 0	reserved can be 0 or 1
bit 1	reserved can be 0 or 1
bit 2	1 if there is no cyclic communication with a fieldbus master
bit 3	1 if there is cyclic communication to a fieldbus master
bit 4	reserved can be 0 or 1
bit 5	reserved can be 0 or 1
bit 6	reserved can be 0 or 1
bit 7	reserved can be 0 or 1
bit 8 - 15	reserved always 0

A-cyclic updates

It is possible to access a datafield of 256 words. To write a value to a parameter you would use the following:

Input parameters	FU32	3 = write parameter
	FU33	parameter field number
	FU34	parameter field value
Output parameters	None	

To read the parameter:

Input parameters	FU32	4 = read parameter
	FU33	parameter field number
Output parameters	FU32	parameter field value

Sample program

```
" " Example program for Festo fieldbus Slave

STEP init
IF
THEN  CFM 62
      WITH V1
      WITH V$D400
      WITH V3
      WITH V1
      WITH V24
      WITH V24
      WITH V1
      " fieldbus slave module
      " Initialise
      " Base address CP61 module
      " Baudrate (3=375 Kb)
      " Slave address
      " Output bytes
      " Input bytes
      " Byte oriented

STEP loop
IF
THEN  LOAD IW1
      TO   FW9000
      " Some inputs
      " fieldbus output byte 0

IF
THEN  CFM 62
      WITH V2
      WITH V9000
      WITH V9100
      " fieldbus slave module
      " Cyclic update info
      " fieldbus outputs start at FW9000
      " fieldbus inputs start at FW9100

IF
THEN  LOAD FW9100
      TO   OW1
      " fieldbus input byte 0
      " LEDs on AS14

IF
THEN  JMP TO loop
```

Profibus DP (CP62)

With this driver and the CP62 module you can connect Profibus DP slaves to the IPC and to use them with the FST IPC.

Note! Before you can use the CP62 module it must be configured using the SyCon configuration software from the company Hilscher.

The inputs/outputs of the modules are copied cyclically to the local function units (inputs, outputs or flag words) configured by the user.

Selecting the driver and assigning parameters

If you want to use the Profibus DP driver in an FST IPC project you must enter the PDP driver in the driver configurator and assign the necessary parameters.

Note! The driver uses the same driver number as PROFI-DP for CP60. It can not be used together in the same project.

Destination drive:

Specify the IPC drive on which the Profibus DP driver PDP.EXE is located or onto which it is to be loaded.

CP62 switch position:

Enter the address switch position of the CP62 module here.

DR-RAM size:

This can be either 2 or 8.

Use FW as IW:

If you select "Y" then the process data will be copied into FWs instead of IWs.

Use FW as OW:

If you select "Y" then the process data will be copied into FWs instead of OWs.

Offset for Inputs:

Enter the index where the input area of the process data starts.

Offset for Outputs:

Enter the index where the output area of the process data starts.

Configuration

Before you can use the CP62 module it must be configured using the SyCon configuration software from the company Hilscher.

For DP-Master, following Parameters should be used:

Start up behaviour:

Automatic release of communication by the system after system initialised.

Watchdog time:

300ms

Addressing mode:

Word address (Important!)

Process Data exchange:

Buffered, program controlled data transfer

Error reaction:

The error reaction "Hard" or "Soft" (e.g.: participant missing or non-configured participant on the bus) depends on the Bus- Parameter setting "Auto Clear".

"Auto Clear On" := Hard reaction.

Error messages from the driver

The driver enters the following error numbers in the FPC error word:

No	Meaning of the error Pointer to error correction
1001	There is no communication between driver and Profibus card Probable cause: <ul style="list-style-type: none">- Card address is not the same with the driver configuration.- No valid configuration in the card system configuration (SyCon)
1004	Configured Slave not in the Bus (Only for Hard reaction)
1005	Input range exceeding
1006	Output range exceeding

Modules

Overview

DP_USIF	Get the current USIF status
DP_GETSL	Checks whether a diagnostic request is present
DP_GETDG	Obtains diagnostic information from a DP slave
DP_CONTR	Profibus function "Global Control Request"

DP_USIF

The User Interface status can not be edited; it will give back the actual status. The module operates in conjunction with the DP driver (PDP).

Input parameters	None	
Output parameters	FU32	= -1 OK
		= 0 driver not loaded
	FU33	Current status or
		0 = system not correctly initialised

DP_GETSL

Checks whether a diagnostic request is present.

Input parameters	None	
Output parameters	FU32	= -1 OK
		= 0 driver not loaded
	FU33	address of first participant with diagnostic requisition or
		\$FFFF = no request present
	FU34	address of first missing participant

DP_GETDG

Request diagnostic information from a DP slave. This should be used in conjunction with the DP_GETSL module.

Input parameters	FU32	Address of the slave (0 to 126)
	FU33	Offset in diagnostic information
	FU34	Group select
Output parameters	FU32	= -1 OK
		= 0 driver not loaded
		= 1 slave address invalid
	FU33	Profibus status of the operation
	FU34	Length of diagnostic information in bytes (not incl. length byte)
	FU35-38	The diagnostic information in consecutive order

Profibus status of the operation, typical values:

00	OK
C3	Partner not responding
04	No parallel master-slave function possible

Will give back max. 4 Words of diagnostic information for every call.

Note! The module call must be repeated as many times as necessary until the value 0 is returned in FE33. The other values are not valid until this is the case.

DP_CONTR

Profibus function "Global Control Request".

Input parameters	FU32	Slave address or
		127 = all slaves (broadcast)
	FU33	Control command
	FU34	Group select
Output parameters	FU32	= -1 OK
		= 0 driver not loaded
	FU33	Profibus status of the operation

All transfer parameters in BYTE format.

Profibus status of the operation, typical values:

00	OK
C3	Partner not responding
04	No parallel master-slave function possible

Note! The module call must be repeated as many times as necessary until the value 0 is returned in FE33. The other values are not valid until this is the case.

Profibus FMS (CP62)

With this driver you can operate the Festo IPC in a Profibus FMS network using the CP62 module of the IPC product range.

Note! Before you can use the CP62 module it must be configured using the SyCon configuration software from the company Hilscher.

There are no external configuration files. The bus parameters, Communication Reference List and Object Directory must be configured through SyCon.

Configuring the driver and assigning parameters

If you want to use the Profibus FMS driver in a FST IPC project, you must enter the PROFIFMS driver in the driver configurer and assign the necessary parameters.

Destination drive:

Specify the drive on which the Profibus FMS driver PROFIFMS.EXE is located or onto which it is to be loaded.

CP62 switch position:

Enter the switch position as set on the CP62 module. The default of CA means that the CP62 module uses a base memory segment of CA00h

Note! Make sure that the memory used by the CP62 is not in use by other modules.

Extended CI commands for PROFIFMS

This driver extends the IPC command interpreter with the following commands:

```
!39          Display driver identification
              and version number
```

Display driver info and version number. This information will also be displayed if an unknown command is entered (for example !39?).

Function modules

Overview

```
FMSREAD Read (polled response)
FMSWRIT Write (polled response)
```

FMSREAD

Read (polled response)

Input parameters	FU32	Communication reference (CR=1 to 32)
	FU33	Object index (20 to 65535)
	FU34	Object subindex (0 to 241)
	FU35	Number of the flag word for status variable
	FU36	Low byte, identification of the operand type for data
		0 = Input word
		1 = Output word
		2 = Flag word
		3 = Register
		High byte, identification of the 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-of-day
		13 = Time-difference
		14 = Bit-string
	FU37	Number of the (first) operand word for data
	FU38	Expected length in bytes
Output parameters	FU32	Error number
		= 0 Successfully processed
		> 0 Error (see table)
Status variable		= 0 Successfully completed
		= 1 Still being processed
		> 1 Terminated by error (see table)

FMSWRIT

Write (polled response)

Input parameters	FU32	
	...	
	FU38	See FMSREAD
Output parameters	FU32	See FMSREAD
Status variable		See FMSREAD

Object directory

If other Profibus FMS participants want to access FST operands, the operands must be configured in the Object Directory (in SyCon configurator). Entering operands is optional, if they are missing in the Object Directory the operands are simply not accessible.

Please note that not all operands are accessible. Only Inputwords, Outputwords, Registers and a limited range of Flagwords can be accessed.

When operands are entered they should be entered according to the following table:

Index	Designation as per IPC	Type	Access
100	EW0-EW63	Array(64)*U16	Read
101	EW64-EW127	Array(64)*U16	Read
102	EW128-EW191	Array(64)*U16	Read
103	EW192-EW255	Array(64)*U16	Read
110	AW0-AW63	Array(64)*U16	Read / Write
111	AW64-AW127	Array(64)*U16	Read / Write
112	AW128-AW191	Array(64)*U16	Read / Write
113	AW192-AW255	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	MW0-MW63	Array(64)*U16	Read / Write
201	MW64-MW127	Array(64)*U16	Read / Write
202	MW128-MW191	Array(64)*U16	Read / Write
203	MW192-MW255	Array(64)*U16	Read / Write
204	MW256-MW319	Array(64)*U16	Read / Write
205	MW320-MW383	Array(64)*U16	Read / Write
206	MW384-MW447	Array(64)*U16	Read / Write
207	MW448-MW511	Array(64)*U16	Read / Write
208	MW512-MW575	Array(64)*U16	Read / Write
209	MW576-MW639	Array(64)*U16	Read / Write
210	MW640-MW703	Array(64)*U16	Read / Write
211	MW704-MW767	Array(64)*U16	Read / Write
212	MW768-MW831	Array(64)*U16	Read / Write
213	MW832-MW895	Array(64)*U16	Read / Write
214	MW896-MW959	Array(64)*U16	Read / Write
215	MW960-MW1023	Array(64)*U16	Read / Write
216	MW1024-MW1087	Array(64)*U16	Read / Write
217	MW1088-MW1151	Array(64)*U16	Read / Write
218	MW1152-MW1215	Array(64)*U16	Read / Write
219	MW1216-MW1279	Array(64)*U16	Read / Write
220	MW1280-MW1343	Array(64)*U16	Read / Write
221	MW1344-MW1407	Array(64)*U16	Read / Write
222	MW1408-MW1471	Array(64)*U16	Read / Write
223	MW1472-MW1535	Array(64)*U16	Read / Write
224	MW1536-MW1599	Array(64)*U16	Read / Write
225	MW1600-MW1663	Array(64)*U16	Read / Write
226	MW1664-MW1727	Array(64)*U16	Read / Write
227	MW1728-MW1791	Array(64)*U16	Read / Write
228	MW1792-MW1855	Array(64)*U16	Read / Write
229	MW1856-MW1919	Array(64)*U16	Read / Write
230	MW1920-MW1983	Array(64)*U16	Read / Write
231	MW1984-MW2047	Array(64)*U16	Read / Write

Error return codes from function modules

If FU32 is <> 0 then the function module returned an error, following table lists the error codes:

198	Function not implemented (call Festo)
199	PROFIFMS driver not loaded
200	Invalid parameter
201	No message number available (255 outstanding requests)

Error codes in status variable

Following table lists some of the possible error values:

42h	66	Connection has been aborted
43h	67	Too many parallel services on one CR
80h	128	Connection could not be opened
81h	129	Error in application of remote partner

Heterogeneous networks

When integrating FPC-405 or PS1-CP60 into the Profibus network make sure to verify the bus parameters. In the SyCon configurator set the baud rate to 500 kBits/second and optimise to user. Then edit the bus parameters.

When using the FPC-405 with default bus parameters set as follows:

Edit Bus Parameter

Baud rate500kBits/s

Slot Time3500tBit

Min. Station Delay of Responders100tBit

Max. Station Delay of Responders1000tBit

Quiet Time0tBit

Setup Time80tBit

Target Rotation Time100000tBit

Target Rotation Time200.0000ms

GAP Actualization Factor100

Max Retry Limit1

Highest Station Address3

Tid1226tBit

Tid21000tBit

Poll Timeout10ms

Data Control Time1200ms

Min Slave Interval2.000ms

Watchdog control200ms

Auto Clear

☒Auto clear modus OFF

☐Auto clear modus ON

OK

Cancel

Also the Communication references differ slightly from the default settings, viewed from the CP62:

Communication Reference List (CRL)



General

Local SAP

33

Remote SAP

31

Remote address/device

3 / FPC405

Communication type

- ☒ Master-Master acyclic (MMAC)
☐ Master-Slave acyclic (MSAC)
☐ Master-Slave cyclic (MSCY)
☐ Multicast
☐ Broadcast

FMS/FDL Commun.

- ☒ FMS
☐ FDL defined
☐ FDL transparent

Dynaset navigation

New CRL

New CRL new station address

Delete

CR 2 / 2

|<

<

>

>|

OK

CRL Table...

<< Less Details

Confirmed Counter / Services

SCC

1

Client Services

- ☒ Write
☒ Read

☒ Get-OV Long

Server Services

- ☒ Write
☒ Read

☐ Get-OV Long

Unconfirmed Counters / Services

SAC

0

Client Services

- ☐ Inform. Report
☐ Event Notification

Server Services

- ☐ Inform. Report
☐ Event Notification

Timer / Definitions

LLI-Timer (Control Interval)

0

* 10 ms

ALI-Timer

0

* 10 ms

Max PDU Size

128

FEC Remote I/O Expansion

Introduction

This document describes the how to expand an FEC Compact controller with 1 to 4 additional modules to provide up to 100 I/O points.

Note! Ther I/O Expansion is not available for the FEC Standard series.

The architecture consists of several FEC Compacts in a Master-Slave arrangement. The Master FEC contains all of usual user-application programs as well as a special software driver to manage communication with the slave FECs. If only a single extension module is required you can also use the IO script **FEC Slave** without installtting drivers. See the chapter "FEC and HC0X" for details.

The Slave FECs do not contain any user-application programs but must be loaded with a special Slave software driver. The balance of this document provides detailed instructions.

System Overview

Systems should be created using FEC-FC30 only. FEC-FC20 modules can be used at the ends of the daisy chain. However, doing so has been experienced as not being stable in some situations and is therefore not recommended.



The modules must be connected together using Festo cable part number 183635. When planning a system, please note that I/O updates may require up to 25 milliseconds due to the nature of the serial data bus.

Configuring Slave FECs

Every FEC module which you will use as a slave device has to be configured in the following manner. It is suggested that you create a separate FST project called FECSLAVE which will contain data that must be loaded into every slave. No user written programs are required. It is only necessary to add the FCSLAVE driver to the FECSLAVE project and load it into the controller. The follow provides step-by-step instructions:

1. Create the FCSLAVE project
2. Select the FCSLAVE project as the current project
3. Open the Driver Configuration
4. Insert the FCSLAVE driver
5. You will be prompted for a Destination Drive. Accept the default B response.
6. A listing of ALL configured drivers will be shown. The FEC SLAVE driver "FCSLAVE" should be listed.

Loading the Slave Software into the FEC

1. Connect your PC to the COMM port of the FEC Slave using the appropriate cable
2. Set the Run/Stop switch of the slave FEC to STOP
3. Switch on the Power to the Slave FEC
4. Download the FECSLAVE project
5. You will receive a warning: "No IO scan table, no programs." This is correct, as only drivers will be loaded.
6. Set the Run/Stop switch of the slave FEC to the RUN position.

Note! Until all slaves and the master are connected and the power is cycled, it is likely that a Red or Orange LED will be shown. This is normal.

7. After the next power cycle, the FEC Slave should operate in the slave mode.
8. Repeat this Loading process for any other FEC Slaves.

Configuring the Master FEC

Adding the Driver

The master FEC is programmed and executes all of the user-application programs. As with standalone FEC controllers, the system I/O configuration has to be recorded.

Additionally, the special FCMaster software driver must to be included in the FST project.

The following provides step-by-step instructions:

1. Create (or select) your FST project.
2. Insert the FCMaster driver to the Driver Configuration
3. You will be prompted for a Destination Drive. Accept the default B response
4. A listing of ALL configured drivers will be shown. The FEC Master driver "FCMASTER" should be listed.

Selecting and adding I/O Entries

The user now must select one of two available I/O entries in the Master Project. The step-by-step process is present below. Please refer to the following section. However, the user must first select which I/O entry is best suited to the application.

The possible selections are:

FEC Slave

If this I/O entry is selected, then during system initialisation the master checks if all expected slave devices respond. If a configured slave does not respond then system error 11 is generated. During runtime, if any slave device fails to respond, system error 11 will be generated.

-or-

FEC Slaves without error 11

This I/O entry reacts in the same way during the initialisation phase. However, excessive runtime communication errors between the master and any slave device will not generate an error.

Each of the FEC Slave nodes must be entered into the FST Project I/O configuration of the FEC Master.

You will be prompted for:

Switch:

Enter 0 for the first slave, 1 for the second slave etc.

IW:

Enter the starting IW (Input Word) number for the Slave. Each FEC requires 2 Input words and each unit must be assigned unique, non-overlapping ranges.

OW:

Enter the starting OW (Output Word) number for the Slave. Each FEC requires 1 Output word and each unit must be assigned unique, non-overlapping ranges.

Note! Don't forget that the FEC master also must be entered into the I/O configuration!

Optional I/O Entry

If desired the user may also include the FEC Slaves Error Counter I/O entry in the master project. This will track the total global expansion bus errors that occur.

If this is desired then include the module "FEC Slaves Error Counter" into the IO configuration. For the IW enter the number of a non-conflicting "virtual" IW (Input Word) which will be used to store the global error count.

Runtime Functioning

The following describes the functionality of a configured and wired system.

Master at Runtime

Upon power up the master will initialise the remote (slave) FEC modules. If there is a discrepancy between the expected versus the actual configuration (the expected slaves do not respond); then an I/O stage defective error will be generated and system error 11 will be stored.

If during operation a communications problem develops (and the appropriate I/O entry is included) then excessive communications errors result in system error 11 being generated and the Master will stop attempting to update the slave FEC I/O points.

The system can be reinitialised by toggling the Run/Stop switch or cycling power.

Slave at Runtime

Upon power up the slave will check the Run/Stop switch. If the switch is in the Stop position, the slave software driver will not be processed and no access from the master can occur. If the switch is in the Run position, then the FEC slave software is processed.

Interpreting the RUN Led in Slave Devices

Red	Communications error
Flashing Orange	No communications with master
Flashing Green	Slave being initialised by master
Green	Communications normal

Diagnostic Program Module (REMDIAG)

The Remote FEC software package also includes software for monitoring the status of the FEC Remote I/O devices from within the user application program. If you want to include this capability in your application, you need to import the REMDIAG Remote FEC error counter into your FEC MASTER project.

Using the REMDIAG Module

Use the following guidelines to access to REMDIAG module from within your Statement List or Ladder Diagram program:

Get error counters

Input parameters	FU32 1	(get error counter)
Output parameters	FU32	Short term error count
		0 if communication OK
		<> 0 if error in last cycle(s)
	FU33	Total error count,
		wraps from 65535 back to 0

Reset total error counter

Input parameters	FU32 2	(reset total error counter)
------------------	--------	-----------------------------

Set retry counter

Input parameters	FU32 3	(set total error counter)
	FU33	Value for retry counter.
		0 = disable

File Handling Modules

Some standard CFM give access to the file systems on diskettes, hard disks, RAM-disks, etc. as supported by the MS-DOS operating system on HC1X and HC2X CPUs. It is possible to have up to 6 open files at a time. If more files than this are opened the CFM returns with error number 4 (too many open files).

The naming convention uses numbers for files. If the file number is in the range 0 to 32767, that is if it is a positive number, it will be translated into a file name directly. For example file number 17 results in a "17" file name in the current working directory of the current disk. For a negative file number the CFM search the string driver and use the string with the opposite number as file name. For example if the file number is "-3" and the string number 3 contains "C:\MYDATA.BIN" this will become the name of the associated file.

All file access CFM pass jobs to the MS-DOS task for further processing and return immediately. The real job will be done by the MS-DOS task somewhat later on a first comes first served base. So more than one operation can be issued at the same time.

All read or write data functions use flag words as storage for read or written data. The reason is, that only the flag words offer a meaningful amount of memory.

All CFM return a result and support an operation status variable. A CFM result of zero means the operation has been started, but need not be complete (and usually is not). A none zero CFM result shows an error condition. The operation is not started and the actual returned value indicates the error type, for example invalid parameters. A status variable value of zero means success. While the value is -1, the operation is still busy, not yet completed. If any error occurs, the value will become >0 and the actual value can be used to obtain further information of the kind of the error. For a list of error values see below.

Do not remove a diskette while there is an open file on it. Doing such in MS-DOS will produce lost clusters or even worse.

Remember, MS-DOS updates the file allocation table not steadily. The method used by MS-DOS would take too much time. MS-DOS updates it when a file is closed. To make sure that all written data is part of a file and all allocated memory (clusters) is noted in the file allocation table the file has to be closed. A simple but effective trick is to position to the desired end of the file, write zero bytes to that location and close the file. This will force MS-DOS to allocate enough memory from the disk and update the file allocation table. Doing this results in the best chance to save the integrity of the file system on a disk.

Another strategy uses a "check point a file" method. Request and save the current position of the file read/write pointer. Then close the file, open it again, and seek to the saved position. This also ensures all data will be written to the disk and the file allocation table will be updated. If the CHKDSK-command finds lost clusters on a disk, the file allocation table has not been updated at the right moment. The first method prevents this nearly always, the second is less sure but even saves partially written clusters to the disk.

The CI commands S and Y and the QUIT and EXIT commands will automatically close all files.

Overview

FCREATE	File create
FOPEN	File open
FCLOSE	File close
FCLOSALL	File close all
FDELETE	File delete
FSEEK	File seek
FSEEKX	File seek extended
FWRITE	File write
FREAD	File read
FWRITSTR	File write string
FREADSTR	File read string

FCREAT File Create

Create a new file or open an existing file for reading and writing. If the file exists it will be truncated to zero length.

Input parameters:

FU32 File number for the MS-DOS
file name. A positive
number is directly converted
into a file name.

The number 17 translates into a file name "17". A negative number is used as opposite index into the string table in the string driver. The associated string contains the file name.

FU33 Number of flag word FW used
for status report.

The status is

FW[FU33] = -1 while function is busy,
= 0 when completed successfully,
= >0 error number after failure.

FU34 Number of flag word FW for
the requested file handle.

Valid when the status flag word shows 0. This handle has to be used with all further operations with the opened file.

Output parameters:

FU32 = 0 OK, function has been
started,
= >0 error number after
failure. Function is not
started.

FOPEN File Open

Open an existing file in the given mode.

Input parameters:

FU32 File number for the MS-DOS
file name. A positive number
is directly converted into a
file name.

The number 17 translates into a file name "17". A negative number is used as opposite index into the string table in the string driver. The associated string contains the file name.

FU33 Number of flag word FW used
for status report.

The status is

FW[FU33] = -1 while function is busy,
= 0 when completed successfully,
= >0 error number after failure.

FU34 Number of flag word FW for
the requested file handle.

Valid when the status flag word shows 0. This handle has to be used with all further operations with the opened file.

FU35 Mode in which to open the file.
0 = read-only access,
1 = write-only access,
2 = read/write access.

Output parameters:

FU32 = 0 OK, function has been started,
= >0 error number after failure. Function is not started.

FCLOSE File Close

Close a previously opened file. After closing a file the file handle is no longer valid and cannot be used for any further operation.

Input parameters:

FU32 File handle for the MS-DOS file as obtained from file open or file create functions.
FU33 Number of flag word FW used for status report.

The status is

FW[FU33] = -1 while function is busy,
= 0 when completed successfully,
= >0 error number after failure.

Output parameters:

FU32 = 0 OK, function has been started,
= >0 error number after failure. Function is not started.

FCLOSALL File Close All

Close all currently open files. After closing all files the file handles are no longer valid and cannot be used for any further operation.

Input parameters:

FU32 Number of flag word FW used for status report.

The status is

FW[FU33] = -1 while function is busy,
= 0 when completed successfully,
= >0 error number after failure.

Output parameters:

```
FU32  =  0  OK, function has been
        started,
        = >0  error number after
        failure. Function is not
        started.
```

FDELETE File Delete

Delete a file. The file to be deleted may not be open.

Input parameters:

```
FU32  File number for the MS-DOS
        file name.
```

The number 17 translates into a file name "17".

```
FU33  Number of flag word FW used
        for status report.
```

The status is

```
FW[FU33] = -1 while function is busy,
           =  0 when completed successfully,
           = >0 error number after failure.
```

Output parameters:

```
FU32  =  0  OK, function has been
        started,
        = >0  error number after
        failure. Function is not
        started.
```

FSEEK File Seek

Move the file read/write pointer to a specified position. The position is given in bytes.

Input parameters:

```
FU32  File handle for the MS-DOS
        file as obtained from file
        open or file create
        functions.

FU33  Number of flag word FW used
        for status report. The
        status is
```

The status is

```
FW[FU33] = -1 while function is busy,
           =  0 when completed successfully,
           = >0 error number after failure.
```

```
FU34  Low word of 4 byte file
        pointer position.
```

```
FU35  High word of 4 byte file
        pointer position.
```

Output parameters:

```
FU32  = 0 OK, function has been
        started,
        = >0 error number after
        failure. Function is not
        started.
```

FSEEKX File Seek Extended

Move the file read/write pointer to a specified absolute or relative position. The position is given in bytes.

Input parameters:

```
FU32  File handle for the MS-DOS
        file as obtained from file
        open or file create
        functions.

FU33  Number of flag word FW used
        for status report.
```

The status is

```
FW[FU33] = -1 while function is busy,
           = 0 when completed successfully,
           = >0 error number after failure.

FU34  Low word of absolute or
        relative 4 byte file pointer
        position.

FU35  High word of absolute or
        relative 4 byte file pointer
        position.

FU36  0=absolute, start from
        beginning of file
        1=relative, start from
        current position
        2=relative, start from end
        of file.

FU37  Number of flag word FW used
        to return the new position.
```

The new position is given as follows

```
FW[FU37]          Low word of absolute 4
                   byte file pointer
                   position.
FW[FU37+1]        High word of absolute 4
                   byte file pointer
                   position.
```

Output parameters:

```
FU32  = 0 OK, function has been
        started,
        = >0 error number after
        failure. Function is not
        started.
```


FWRITE Write File

Write some portion of the flag words to an open file. The amount is given in bytes.

Input parameters:

FU32 File handle for the MS-DOS
file as obtained from file
open or file create
functions.

FU33 Number of flag word FW used
for status report.

The status is

FW[FU33] = -1 while function is busy,
= 0 when completed successfully,
= >0 error number after failure.

FU34 Number of the first flag
word FW that has to be
written.

FU35 Number of the bytes that
have to be written.

Writing 0 bytes will truncate the file at the current file pointer position. If the number of bytes is odd, only the low byte of the last flag word will be written.

Output parameters:

FU32 = 0 OK, function has been
started,
= >0 error number after
failure. Function is not
started.

FREAD Read File

Read some portion of the flag words from an open file. The amount is given in bytes.

Input parameters:

FU32 File handle for the MS-DOS
file as obtained from file
open or file create
functions.

FU33 Number of flag word FW used
for status report.

The status is

FW[FU33] = -1 while function is busy,
= 0 when completed successfully,
= >0 error number after failure.

FU34 Number of the first flag
word FW that has to be read.

FU35 Number of the bytes that

have to be read.

If the number of bytes is odd, only the low byte of the last flag word will be read.

FU36 Number of flag word FW for
the number of bytes that
actually have been read.

Output parameters:

FU32 = 0 OK, function has been
started,
= >0 error number after
failure. Function is not
started.

FREADSTR Read string from file

Read string from file until delimiter is found.

Input parameters:

FU32 File handle for the MS-DOS
file as obtained from file
open or file create
functions.

FU33 Number of string.

FU34 Number of the bytes that
will be checked (max).

FU35 Delimiter (0..255).
0: delimiter is CR LF.

Output parameters:

FU32 =0 DONE, function has been
executed successfully
200 String driver not found
201 Reached EOF before
delimiter was found. File
pointer remains
unchanged.
202 Checked given amount of
characters before
delimiter was found.
File pointer remains
unchanged.
>0 Error, see below.

FWRITSTR Write string to file.

Write string plus delimiter to file.

Note! This module needs to be called more than once in order to execute the function.

Input parameters:

FU32 File handle for the MS-DOS
file as obtained from file
open or file create
functions.

FU33 Number of string.

FU34 Number of the bytes that
will be written (max).

FU35 Delimiter (0..255).
0: delimiter is CR LF, -1:
no delimiter appended

Output parameters:

FU32 =0 DONE, function has been
executed successfully
200 String driver not found
>0 Error, see below.

Operation result codes

Most functions will set a status value in a flag word. The status value has following meanings.

-1 BUSY Function not yet
completed,
0 OK Function completed
successfully,
>0 ERROR Function failed to
complete
successfully, that
is an error, value
specifies error
type.

If the status value is >0 it is regarded an error.

2 File not found
3 Path not found
4 Too many open files
5 Access denied
6 Invalid handle number
12 Invalid access code
100 Invalid parameter
101 Disk full
102 Out of memory

Strings

Version 1.01

The FST string driver makes a new additional data type available; the STRING data type. The default setting provides support for 256 strings. Individual strings are addressed by way of the string number. Strings may contain any characters with the exception of the NUL character (hexadecimal \$00). Strings can be as long as necessary within the scope of the set memory requirement. The strings are not retentive.

Configuring the driver and assigning parameters

If you want to use strings in an FST IPC project, you must enter the STRINGS driver in the driver configurer and assign the necessary parameters.

IPC drive:

Specify the drive on which the string driver STRINGS.EXE is located or onto which it is to be loaded.

Reserved memory in bytes:

Enter the maximum memory capacity to be used for strings. The permissible range is from 5 to 65000 bytes. The default setting is 5 kilobytes. This setting can also be specified or modified using the STRINIT module.

Number of strings:

Enter the maximum number of strings. The permissible range is from 5 to 1024 strings. The default setting is 256 strings.

File with default assignments:

Enter the name of the file that contains the initialisation values for the strings. The format of this file is described in the following section.

Initialisation of strings

A simple text file is used for initialising strings. The file name must have the extension "TXT". Each line in the file is a string. The first line is string 0. Missing strings are initialised as blank strings. Any new-line characters (CR or LF) in the text file are removed. Special characters are represented by a combination of two characters, the '\' character and another character. The following special characters are possible:

\a	alert	Bell character (audible signal)
\b	backspace	Move position one character to the left
\f	formfeed	Form feed (FF)
\n	linefeed	Line feed (LF)
\r	return	New line (CR)
\t	tab	Tabulator character
\<no>	Hexadecimal definition of a character; <>no> must begin with a digit, for example "\0A8" (correct) rather than "\A8" (incorrect).	

The '\' character is represented by two backslashes: '\\'.

Extended CI commands for strings

When strings are used in a project, in other words when the string driver has been loaded, the command set of the CI is supplemented by the string driver. The additional commands for the string driver are shown below.

```
!3Dx      Display string x

!3Mx=text  Set string x with the sequence
           of characters 'text'

!3S        Status indication
```

Result "=count=<XX>,storage=<YY>,<ZZ>", where

```
XX  = Number of strings
YY  = Reserved memory capacity
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; in this case it is in driver 3 for strings.

Modules for handling strings

There are a number of ready- prepared modules which can be used for handling strings. These must be imported into a project in the usual way. The various strings concerned are specified with the string number. In order to delete string 5, that is to reduce it to length 0, the STRCLR module must be invoked. The STRCLR module may have been imported as CMP 73, for example.

In STL the program lines appear as follows:

```
THEN      . . .
          CMP 73  " delete with STRCLR
          WITH V5 " from string 5
```

In order to copy string 6 into string 12, the STRCPY module must be invoked. The STRCPY module may have been imported as CMP 74, for example. In STL the program lines appear as follows:

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

Most string modules return a result for error detection.

Overview of modules

STRADDR	Determine internal address of a string
STRAPPND	Append a character at the end of a string
STRATOH	Convert a hexadecimal string into a word
STRATOI	Convert a string into a signed word
STRATOIX	Convert a string into a signed word
STRATOU	Convert a string into an unsigned word
STRCAT	Combine two strings in a third string
STRCHECK	Memory check
STRCHGET	Extract a character from a string
STRCHSET	Replace a character in a string
STRCI	Execute a CI command
STRCLR	Delete a string
STRCMP	Compare two strings character by character, differentiating between upper and lower case
STRCPY	Copy string
STRDEL	Delete part of a string
STRDUMP	Display a number of strings
STRFILL	Create a string with a specified number of identical characters
STRFILLW	Fill a string with another string, right- or left-justified

STRFINDC	Find a character in a string
STRFINDS	Find a substring in a string
STRGROW	Enlarge the string memory for an individual string
STRHTOA	Convert a word into a hexadecimal string
STRICMP	Compare two strings character by character, not differentiating between upper and lower case
STRINIT	Initialisation or re-initialisation
STRINSRT	Insert a string into another string
STRITOA	Convert a signed word into a string
STRLEFT	Left substring
STRLEN	Length of a string
STRLOWER	Convert a string to lower case
STRMID	Middle substring
STRNCMP	Compare the first characters of two strings, differentiating between upper and lower case
STRNICMP	Compare the first characters of two strings, not differentiating between upper and lower case
STRRIGHT	Right substring
STRSTAT	Status of string driver
STRUPPER	Convert a string to upper case
STRUSAGE	Used and free memory
STRUTOA	Convert an unsigned word into a string

STRADDR

Determine the internal address of a string.

Input parameters	FU32	Number of the string
Return parameters	FU32	Offset of the address
		0 if string number is invalid
	FU33	Segment of the address
		0 if string number is invalid

As the lengths of the strings concerned are subject to dynamic change, the strings may need to be moved within the string memory. The direct consequence of this is that after the address of a string has been determined it may change immediately as a result of string operations with other strings. The addresses of strings are therefore to be used under special conditions.

STRAPPND

Append a character at the end of a string.

Input parameters	FU32	Number of the string
	FU34	character to append
Return parameters	FU32	0 if successful, otherwise error

STRATOH

Convert a hexadecimal string into a word.

Input parameters	FU32	Number of the string
Return parameters	FU32	0 if successful, otherwise error
	FU33	Resultant value

Space characters and tabulator characters at the beginning and end of the string are allowed. A '\$' character before the hexadecimal numeric characters is also allowed.

STRATOI

Convert a string into a signed word.

Input parameters	FU32	Number of the string
Return parameters	FU32	0 if successful, otherwise error
	FU33	Resultant value

Space characters and tabulator characters at the beginning and end of the string are allowed. A '+' or '-' character before the numeric characters is also allowed.

STRATOIX

Convert a string into a signed word.

Input parameters	FU32	Number of the string
	FU33	String position for the conversion
Return parameters	FU32	0 if successful, otherwise error
	FU33	Resultant value
	FU34	Number of characters used for the conversion

Space characters and tabulator characters at the beginning and end of the string are allowed. A '+' or '-' character before the numeric characters is also allowed.

STRATOU

Convert a string into an unsigned word.

Input parameters	FU32	Number of the string
Return parameters	FU32	0 if successful, otherwise error
	FU33	Resultant value

Space characters and tabulator characters at the beginning and end of the string are allowed. A '+' character before the numeric characters is also allowed.

STRCAT

Combine two strings in a third string.

Input parameters	FU32	Number of the first source string
	FU33	Number of the second source string
	FU34	Number of the destination string
Return parameters	FU32	0 if successful, otherwise error

STRCHECK

Memory check for the string memory.

Input parameters	None	
Return parameters	FU32	0 if successful, otherwise error

STRCHGET

Extract an indexed character from a string.

Input parameters	FU32	Number of the string
	FU33	Character index, 1 for first character of the string
Return parameters	FU32	0 if successful, otherwise error
	FU33	Extracted character

STRCHSET

Replace an indexed character in a string.

Input parameters	FU32	Number of the string
	FU33	Character index, 1 for first character
	FU34	Replacement character in lower-order byte
Return parameters	FU32	0 if successful, otherwise error

STRCI

Execute a CI command.

Input parameters	FU32	Number of the string with the CI command
	FU33	Number of the string for the result of the command
Return parameters	FU32	0 if successful, otherwise error

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

STRCLR

Delete a string.

Input parameters	FU32	Number of the string
Return parameters	FU32	0 if successful, otherwise error

This is the same as clearing a string or reducing its length to 0.

STRCMP

Compare two strings character by character.

Input parameters	FU32	Number of the first string
	FU33	Number of the second string
Return parameters	FU32	0 if successful, otherwise error
	FU33	1 if characters in first string > characters in second string
		0 if both strings are identical
		-1 if characters in first string < characters in second string

Upper-case and lower-case letters are differentiated. Upper-case letters have a lower value than lower-case letters.

STRCPY

Copy one string into another string.

Input parameters	FU32	Number of the source string
	FU33	Number of the destination string
Return parameters	FU32	0 if successful, otherwise error

STRDEL

Delete part of a string.

Input parameters	FU32	Number of the string
	FU33	Index for first character to be deleted, 1 for first character
	FU34	Number of characters
Return parameters	FU32	0 if successful, otherwise error

STRDUMP

Produce a debug output of a series of strings on the monitor of the IPC.

Input parameters	FU32	Number of the first string
	FU33	Number of the last string
Return parameters	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 a string with a specified number of identical characters.

Input parameters	FU32	Number of the string
	FU33	Number of characters

	FU34	Filler character
Return parameters	FU32	0 if successful, otherwise error

STRFILLW

Fill a string with another string, right- or left-justified.

Input parameters	FU32	Number of the string to be created
	FU33	Number of characters for this string >0 for right-justified <0 for left-justified
Return parameters	FU34	Number of the string to be transferred
	FU32	0 if successful, otherwise error

If the length specified for the string to be created is too short, the resulting string is truncated.

STRFINDC

Find a character in a string.

Input parameters	FU32	Number of the string
	FU33	Character to be found
Return parameters	FU32	0 if successful, otherwise error
	FU33	>0 position, 1 for first character 0 if character not found

STRFINDS

Find a substring in a string.

Input parameters	FU32	Number of the string in which to search
	FU33	Number of the substring to be found
Return parameters	FU32	0 if successful, otherwise error
	FU33	>0 position, 1 for first character 0 if substring not found

STRGROW

Enlarge the string memory for a single string from the available memory.

Input parameters	FU32	Number of the string
	FU33	New maximum size of the string in characters (without \0 at the end of the string)
Return parameters	FU32	0 if successful, otherwise error

The existing string is retained unchanged. After this a string with a length up to the maximum length set here can be written to the address determined with STRADDR. Each other invocation of a string module can change the set value again.

STRHTOA

Convert a word into a hexadecimal string.

Input parameters	FU32	Value to be converted
	FU33	Number of the string
Return parameters	FU32	0 if successful, otherwise error

A '\$' character is inserted before the four hexadecimal numeric characters.

STRICMP

Compare two strings character by character.

Input parameters	FU32	Number of the first string
	FU33	Number of the second string
Return parameters	FU32	0 if successful, otherwise error

FU33	1	if characters in first string > characters in second string
	0	if both strings are identical
	-1	if characters in first string < characters in second string

There is no differentiation between upper- and lower-case letters.

STRINSRT

Insert a string into another string from a specified position.

Input parameters	FU32	Number of the string into which the other string is to be inserted
	FU33	Position before which the string is to be inserted, 1 = before first character
	FU34	Number of the string that is to be inserted
Return parameters	FU32	0 if successful, otherwise error

STRINIT

Set the size of the memory for strings and the maximum number of strings, and reinstate the initialisation values of the strings.

Input parameters	FU32	Size of the string memory (minimum 1000)
	FU33	Maximum number of strings (minimum 10)
Return parameters	FU32	0 if successful, otherwise error

STRITOA

Convert a signed word into a string.

Input parameters	FU32	Value to be converted
	FU33	Number of the string
Return parameters	FU32	0 if successful, otherwise error

If necessary a '-' character is inserted before the numeric characters.

STRLEFT

Transfer the left substring of a specified length into a string.

Input parameters	FU32	Number of the source string
	FU33	Number of characters
	FU34	Number of the destination string
Return parameters	FU32	0 if successful, otherwise error

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

STRLEN

Length of a string.

Input parameters	FU32	Number of the string
Return parameters	FU32	0 if successful, otherwise error
	FU33	Length

STRLOWER

Convert a string to lower case.

Input parameters	FU32	Number of the string
Return parameters	FU32	0 if successful, otherwise error

Note! Country-specific characters like Ä, Ö, Ü are not converted.

STRMID

Convert a middle substring from a specified starting position and of a specified length into a string.

Input parameters	FU32	Number of the source string
	FU33	Start character in the source string, 1 for the first character of the source string
	FU34	Number of characters
Return parameters	FU35	Number of the destination string
	FU32	0 if successful, otherwise error

If the starting position is not found in the source string, a blank string is produced. If the source string is too short, the destination string is shortened accordingly.

STRNCMP

Compare the first characters of two strings.

Input parameters	FU32	Number of the first string
	FU33	Number of the second string
	FU34	Number of characters
Return parameters	FU32	0 if successful, otherwise error
	FU33	1 if characters in first string > characters in second string
		0 if both strings are identical
		-1 if characters in first string < characters in second string

Upper-case and lower-case letters are differentiated. Upper-case letters have a lower value than lower-case letters.

STRNICMP

Compare the first characters of two strings.

Input parameters	FU32	Number of the first string
	FU33	Number of the second string
	FU34	Number of characters
Return parameters	FU32	0 if successful, otherwise error
	FU33	1 if characters in first string > characters in second string
		0 if both strings are identical
		-1 if character in first string < character in second string

There is no differentiation between upper- and lower-case letters.

STRRIGHT

Transfer a right substring of a specified length to a string.

Input parameters	FU32	Number of the source string
	FU33	Number of characters
	FU34	Number of the destination string
Return parameters	FU32	0 if successful, otherwise error

If the source string is too short, the destination string is shortened accordingly.

STRSTAT

Query the status of the string driver.

Input parameters	None	
Return parameters	FU32	The set memory size for strings
	FU33	Maximum number of strings
	FU34	Memory used by strings
	FU35	Free memory remaining

STRUPPER

Convert a string to upper case.

Input parameters	FU32	Number of the string
Return parameters	FU32	0 if successful, otherwise error

Note! Country-specific characters like ä, ö, ü are not converted.

STRUSAGE

Determine the amount of used and free memory for strings.

Input parameters	None	
Return parameters	FU32	Memory used by strings
	FU33	Free memory remaining

STRUTOA

Convert an unsigned word into a string.

Input parameters	FU32	Value to be converted
	FU33	Number of the string
Return parameters	FU32	0 if successful, otherwise error

STR2FLAG

Copy a string into a flag word area.

Input parameters	FU32	Number of the string
	FU33	Maximum number of characters to be copied
	FU34	Number of the flag word
Return parameters	FU32	0 if successful, otherwise error
	FU33	Number of characters actually copied

FLAG2STR

Copy a flag word area into a string.

Input parameters	FU32	Number of the string
	FU33	Number of characters to be copied
	FU34	Number of the flag word
Return parameters	FU32	0 if successful, otherwise error

Each flag word contains two characters.

The STR2FLAG and FLAG2STR modules are particularly useful in conjunction with the file modules FREAD and FWRITE when it is necessary to write strings to a file. The specified flag word area is used for intermediate storage.

Screen and Keyboard

Version 1.02

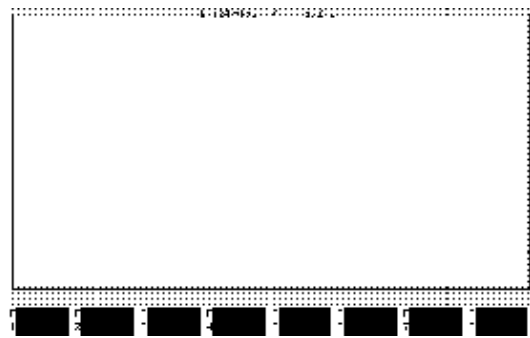
The purpose of this driver is to allow the design of a simple, text- based operating interface for projects using the FST IPC software, working directly on the IPC.

It is not intended to replace sophisticated user interfaces with this driver. In particular, there are no graphical representations involved or complex windowing, nor even list boxes or such features. The emphasis here is placed clearly on simplicity.

For many applications, however, a simple user interface of this kind is entirely adequate. Typical projects which may be able to use this user interface would have only a small number of displays for alphanumeric values and would allow only a few inputs or selections to be made.

Introduction

The user interface is based on the familiar FST software. It has very much the same look-and-feel as this does. It has a blue frame with a title for the work area, a message line and eight function keys. The method of operating this user interface is similar to that of operating the FST.



The user interface is divided into individual screen pages. The various screen pages are accessed via a number, beginning at 0. Each screen page contains a user- definable number of fixed frames, text blocks, display areas and input fields. Switching from one screen page to another is normally taken care of by the user program in the FST. There are appropriate program modules which deal with this.

All screen pages are independent of each other.

The amount of memory space provided allows for up to 50 screen pages, each with a maximum of 50 input fields.

All input fields use the string data type from the string driver. There are no dynamic displays. It is an easy matter to create such displays, however, by invoking the SCRWRITE module.

It is possible to switch between the FST user interface provided by this driver and the command interpreter of the main runtime program (FSTPCR22).

If the necessary extra effort is made, it is also possible to set up special-purpose screen representations (for example to suit particular customer requirements). There is a number of modules for this purpose which disregard the conditions that otherwise generally apply and which access the screen and keyboard directly.

Operation of the user interface

The method of operating the user interface is based on that for the FST software. The table below shows the functions of the keyboard but without the standard letters that are entered in the respective input fields.

Keyboard	Function
TAB	Move to the next input field
ENTER	
Down arrow	
Shift- TAB	Move to the previous input field
Up arrow	
HOME	Move to the first input field
END	Move to the last input field
BS	In overwrite mode: one character to the left; in insert mode: delete the character to the left of the cursor
Left arrow	One character to the left
Right arrow	One character to the right
INS	Toggle between overwrite mode (cursor is an underline character) and insert mode (cursor is a block)
DEL	Delete the character at the position of the cursor

Alt-F10 Switch to the screen of the main runtime program (can be deactivated)

Processing of the entry of ESC, F1 to F10, CTRL-UP ARROW, CTRL-DOWN ARROW, PAGE-UP, PAGE-DOWN, CTRL-HOME, CTRL-END, CTRL-PAGE-UP and CTRL-PAGE-DOWN must be dealt with by the user program.

All inputs of standard characters (letters and digits) are made directly into the various strings that are provided for each input field.

Functionality

If necessary, strings assigned for input fields are shortened to the maximum length of the defined input field. In the case of input fields for passwords, the length is reduced to zero before editing; i.e. the string is deleted.

It is possible to switch between the FST user interface and the command interpreter of the runtime main program. This can be done in two ways: either directly via the keyboard or by the user program in the FST via the SCRFLIP module:

In the CI:

- The F command switches to the current screen page.
- The D command switches to a specifiable page.

Switching by a program:

- The SCRFLIP module switches between the current page of the user interface and the runtime main program.
- The SCRXCHG module switches between the pages of the user interface.

To switch to the screen of the main runtime program from the screen page itself:

- Enter ALT-F10 on the keyboard (this function can be deactivated).

Note! The string driver is a prerequisite and is always used for handling strings.

Attributes

The FST software makes use of several standardised colour attributes for representing parts of the screen. These standard attributes are accessible via a numbering system.

Number	Attribute
0	Normal screen attribute
1	Highlighting screen attribute
2	Attribute of the frame
3	Attribute of edited fields
4	Attribute of function keys
5	Attribute of the message line
6	Attribute of the message line in the event of errors

In some cases the attribute can be freely chosen. It consists of one word (16 bits); its higher-order byte contains the attribute and its lower-order byte must be equal to zero.

The higher-order byte contains the following:

Bits	Use
0 to 7	Always 0
8 to 11	Foreground colour 0 = black, 1 = blue, 2 = green, 3 = cyan, 4 = red, 5 = magenta, 6 = brown, 7 = light grey, 8 = dark grey, 9 = light blue, 10 = light green, 11 = light cyan, 12 = light red, 13 = light magenta, 14 = yellow, 15 = white
12 to 14	Background colour 0 = black, 1 = blue, 2 = green, 3 = cyan, 4 = red, 5 = magenta, 6 = brown, 7 = light grey
15	= 0 for normal display = 1 for flashing display

This means, for example, that the attribute \$9E (= \$80 + \$10 + \$0E) is the attribute for flashing display of yellow characters on a blue background.

Feature flags

The following properties can be set with the use of feature flags.

B	V	Meaning
i	a	
t	l	
	u	
	e	
0	0	ALT-F10 switching enabled
	1	ALT-F10 switching disabled
1	0	Normal use of the keyboard
	1	Direct reading of the keyboard for special applications
2	0	Normal use of the keyboard
	1	Reading of the keyboard with handshake via SCRKBD
3	0	No automatic activation
	1	Automatic activation (== ACTIVATE)
15		Set by the driver in the event of syntax errors

If bit 2 is set (handshake with SCRKBD), it must be ensured that all inputs will be picked up by the user program because otherwise input will be blocked. Bit 15 is set by the driver during processing of the initialisation data.

Configuring the driver and assigning parameters

If you want to use the screen driver in an FST IPC project, you must enter the SCREEN driver in the driver configurer and assign the necessary parameters. The driver for the user interface makes internal use of the driver for strings. This driver must therefore also be configured.

IPC drive:

Specify the drive on which the screen driver SCREEN.EXE is located or onto which it is to be loaded.

Screen type:

Here you must specify the type of video card that is connected:

CGA	CGA card with 80x25 characters
MGA	MGA card with 80x25 characters
HGC	HGC card with 80x25 characters
EGA	EGA card with 80x25 characters
VGA	VGA card with 80x25 characters
BG20	BG20 on the IPC with 40x30 characters

The default setting is CGA.

File with screen layout [.TXT]: Here you must specify the name of the file that contains the data for initialising the user interface.

Initialising the user interface

In principle the user interface does not require any initialisation data because it can be configured and parameterised entirely via modules during execution of the user program. It is however considerably simpler and clearer, as well as being more economical with available space, to carry out initialisation with separate files intended solely for this purpose.

Initialising the user interface in text form

The text form supports the syntax described in the following. Each line may contain only one statement relating to the screen layout. Lines beginning with a semicolon ';' are comments. Blank lines may be inserted to improve readability. A statement consists of one keyword and a larger or smaller number of additional parameters, depending on the type of statement. Additional comments after a complete statement in the same line as the statement itself are allowed, but they must also begin with ';'.

There are two types of statement: initialisation statements and statements for the definition of screen pages. Initialisation statements must always be placed before the statements for screen pages. Numeric values must always be specified in decimal form. Strings are enclosed between single or double quotation marks, such as in 'Text' or "Text". Strings for the titles of frames must contain the usual FST characters, such as in "[Title]".

Initialisation statements

ACTIVATE

Display on the screen is activated immediately after the project is loaded.

FEATURE_FLAGS <value>

The feature flags are set to the value that is passed here. As the feature flags also contain the ACTIVATE flag, it is necessary either to leave out ACTIVATE or to place it after FEATURE_FLAGS. The meanings of the individual bits of the feature flags are explained in Section 3.2.

Statements for screen pages

```
SCREEN          <string>
SCREEN_INITIAL  <string>
```

Start of the definition of a screen page with its title. The SCREEN_INITIAL statement must not be used more than once, because only one page can be the first page.

```
TEXT           <X position> <Y position>
               <attribute> <string>
```

For display of non-varying text on the screen page. If the X position is negative, the text is displayed such that the string ends at the specified position (right-justified).

```
TEXT_STRING    <X position> <Y position>
               <attribute> <string number>
TEXT_SIGNED     <X position> <Y position>
               <attribute> <flag word number>
TEXT_UNSIGNED   <X position> <Y position>
               <attribute> <flag word number>
TEXT_HEXWORD    <X position> <Y position>
               <attribute> <flag word number>
```

For display of the contents of a string or of a flag word as a signed value, an unsigned value or a hexadecimal value. When the hexadecimal form is used, there are always 4 characters displayed. Unwanted characters may need to be overwritten. A typical situation is display of a short string at a position that previously contained a longer string. If the X position is negative, the text is displayed such that the string ends at the specified position (right-justified).

```
MESSAGE         <string>
MESSAGE_ERROR    <string>
```

Display for the message line; a maximum of 78 characters. In the case of MESSAGE_ERROR the display is made with the attribute for error messages.

```
FUNCTION_KEY    <key number> <upper string>
               <lower string>
```

Strings for the upper and lower halves of a label for a function key, each consisting of up to 8 characters.

```
INPUT_STRING    <X position> <Y position>
               <width> <string number>
INPUT_PASSWORD   <X position> <Y position>
               <width> <string number>
```

Definition of a positioned input field. All inputs are made in strings. These must then be evaluated by the user program by means of appropriate string functions. For details refer to the description of the string driver. The width specified here is the maximum length of the edited string, and the string number indicates the number of the string (in the string driver) that is used for the edited string. When INPUT_PASSWORD is used, the contents of the string are not displayed.

```
FRAME          <X position> <Y position>
               <width> <height>
               <string for title>
```

For drawing a frame, positioned with the top left-hand corner (x,y) and a frame title. Even if the title is blank it must be specified. The title must not be longer than the specified width less two.

Example

Below is a simple example consisting of three screen pages for the production of an extraordinarily important basic foodstuff for mankind.

```
; default settings
FEATURE_FLAGS 1
ACTIVATE

; screen 0, first screen
SCREEN_INITIAL "[ Chewing  Gum Production ]"
TEXT 30 10 0 "Chewing  Gum Production is Ready"
FUNCTION_KEY 1 "START" ""

; screen 1, interrogation of production quantity
SCREEN "[ Chewing Gum Production ]"
TEXT 20 10 0 "How many ?"
INPUT_STRING 45 10 5 1
FUNCTION_KEY 1 "START" "SYSTEM"
FUNCTION_KEY 2 "STOP" "SYSTEM"

; screen 2, error
SCREEN "[ Chewing Gum Production ]"
TEXT 20 10 1 "Production Interrupted"
MESSAGE_ERROR "Nothing Works Without Gum"
FUNCTION_KEY 1 "continue" ""
```

Switching between the individual screen pages and evaluation of the input of the production quantity takes place within the control program and is not shown here.

Extended CI commands for the user interface

This driver includes a command interpreter add-on to allow operation of the user interface via the command interpreter.

The additional commands are as follows:

```
!4F    Toggle between the user
```



```
interface and the screen of
the main runtime program
(flip display).
```

Within the user interface you can switch back to the screen of the main runtime program via the keyboard by pressing ALT-F10 (this function can be deactivated). The response returned is "=0" when the system switches to the screen of the runtime main program, and "=1" is returned when the system switches to the screen of the user interface.

```
!4Dx  Select screen page x.
```

The response returned is "=0" if it was not possible to select the page, whereas "=1" is returned if the system succeeds in switching to the specified screen of the user interface.

```
!4S   Status output.
```

If the screen of the main runtime program is currently displayed, the response that is returned is "=0,<feature flags>,<number of pages>". If a page of the user interface is displayed, the response is "=1,<feature flags>,<number of pages>,<page number>".

In these commands, "!4" is the prefix for a CI call in a driver; in this case it is in driver 4 for the user interface.

Modules for accessing the user interface

There are a number of ready- prepared modules that can be used for operating the user interface. These must be imported into a project in the usual way, where they can be used as CFMs or CMPs.

Overview of modules

SCRATTR	Calculate screen attribute
SCRCLEAR	Clear screen
SCRUGPO	Determine current position of cursor
SCRUSPO	Set current position of cursor
SCRUTYP	Set cursor type
SRDOKEY	Execution of a keyboard input
SCREDIT	Add an input field
SCREKBD	Clear keyboard buffer
SCRFILL	Fill an area with space characters
SCRFKEY	Specify the label of a function key
SCRFLAGS	Set feature flags
SCRFLIP	Toggle between screen page and FST CI
SCRFRAME	Draw a frame
SCRKBD	Interrogate the last input on the keyboard
SCRMSG	Display a string in the message line
SCRNEW	Draw a new, blank screen
SCRPUTS	Write a string at a set position on the screen
SCRUPDT	Refresh screen contents
SCRWRITE	Write a string at a set position on the screen
SCRW0808	8x8 block characters
SCRW0814	8x14 block characters
SCRXCHG	Switch between screen pages

The SCRCLEAR and SCRPUTS modules should not be used unless absolutely necessary because they circumvent the check for an activated screen page.

SCRATTR

Calculate screen attribute

Input parameters	FU32	Foreground colour
	FU33	Background colour
	FU34	Flashing option if not equal to 0
Output parameters	FU32	Attribute

SCRCLEAR

Clear screen.

Input parameters	None
Output parameters	None

This module should be used for special applications only and also works on the screen of the runtime main program. The module is not necessary for normal applications.

SCRUGPO

Determine current position of cursor.

Input parameters	None	
Output parameters	FU32	X starting position, column, 0 to 79, in the case of errors -1
	FU33	Y starting position, line, 0 to 23, in the case of errors -1

This module should be used for special applications only and also works on the screen of the runtime main program. The module is not necessary for normal applications.

SCRUSPO

Set current position of cursor.

Input parameters	FU32	X starting position, column, 0 to 79
	FU33	Y starting position, line, 0 to 23
Output parameters	None	

This module should be used for special applications only and also works on the screen of the runtime main program. The module is not necessary for normal applications.

SCR CUTYP

Set cursor type.

Input parameters	FU32	0 = No cursor 1 = Underline cursor 2 = Block cursor
Output parameters	None	

This module should be used for special applications only and also works on the screen of the runtime main program. The module is not necessary for normal applications.

SCRDOKEY

Execution of a keyboard input.

Input parameters	FU32	Keyboard input
Output parameters	None	

This module should be used for special applications only and also works on the screen of the runtime main program. The module is not necessary for normal applications.

SCREDIT

Add an input field.

Input parameters	FU32	X position
	FU33	Y position
	FU34	Width
	FU35	= 0, standard input field = 1, input field for password (string not visible)
Output parameters	FU36	String number
	FU32	= 0 if display appears = -1 if the position is not admissible, the string does not exist or the user interface is not activated

Before editing is carried out, the string that is passed is shortened to the maximum length as specified in width. In the case of non-visible strings for passwords, the string is deleted before editing (length 0). The SCREDIT module adds an input field to the current screen. This is not retained after switching to a different screen page, however.

SCREKBD

Clear keyboard buffer

Input parameters	None
Output parameters	None

SCR FILL

Fill an area with space characters.

Input parameters	FU32	X starting position, column, 0 to 79
	FU33	Y starting position, line, 0 to 24
	FU34	Width, >= 1
	FU35	Height, >= 1
	FU36	Attribute
Output parameters	FU32	= 0 if display appears = -1 if position is not admissible or the user interface is not activated

SCR FKEY

Specify the label of a function key.

Input parameters	FU32	Number of the function key, 1 to 8
	FU33	Number of the string for the upper half of the function key
	FU34	Number of the string for the lower half of the function key
Output parameters	FU32	= 0 if display appears = -1 if the string does not exist, the number of the function key is not admissible or the user interface is not activated

interface is not activated

Each function key has an upper and a lower string. Each string may contain up to 8 characters.

SCR FLAGS

Set properties of the user interface (feature flags).

Input parameters	FU32	0 = Interrogate current setting 1 = New setting
If FU32 = 1	FU33	Feature flags
Output parameters	FU32	Current setting of the feature flags

The basic setting is 0.

SCRFLIP

Toggle the screen display between the currently selected screen page in this driver and the screen of the main runtime program FSTPCR22.

Input parameters	FU32	0 FSTPCR22 screen 1 Currently selected screen page in this driver
Output parameters	None	

SCRFRAME

Draw a frame.

Input parameters	FU32	X starting position, column, 0 to 78
	FU33	Y starting position, line, 0 to 23
	FU34	Width, >= 2
	FU35	Height, >= 2
Output parameters	FU32	= 0 if display appears = -1 if position is not admissible or the user interface is not activated

SCRKBD

Interrogate the last input on the keyboard.

Input parameters	None	
Output parameters	FU32	= 0 if no input is available <>0 last input

Currently the following inputs are supported.

Key	Hexadecimal	Decimal Low-order byte
ESC	001B	27 Character codes
F1	013B	315 Scan codes
F2	013C	316
F3	013D	317
F4	013E	318
F5	013F	319
F6	0140	320
F7	0141	321
F8	0142	322
F9	0143	323
F10	0144	324
PAGE UP	0149	329
PAGE DOWN	0151	337
CTRL ARROW LEFT	0173	371
CTRL ARROW RIGHT	0174	372
CTRL END	0175	373
CTRL PAGE DOWN	0176	374
CTRL HOME	0177	375
CTRL PAGE UP	0184	388

If direct reading of the keyboard is activated (feature flags bit 1 = 1), all inputs are returned without processing. Normal processing can be carried out using the SCRDOKEY module.

SCRMSG

Display a string in the message line, optionally as an error message.

Input parameters	FU32	Number of the string to be displayed
	FU33	0 for normal display 1 for an error message
Output parameters	FU32	= 0 if display appears = -1 if the string does not exist or the user interface is not activated

To delete the message line, use a blank string.

SCRNEW

Draw a new, blank screen in the same way as FST.

Input parameters	None
Output parameters	None

This module should only be used if the option of initialisation is not used but instead the screen is built directly by programming in the user program.

SCRPUTS

Write a string at a set position on the screen.

Input parameters	FU32	Number of the string to be displayed
------------------	------	--------------------------------------

	FU33	X starting position, column, 0 to 79
	FU34	Y starting position, line, 0 to 24
	FU35	0 for normal display 1 for highlighted display
Output parameters	FU32	= 0 if display appears = -1 if the string does not exist, the position is invalid or the user interface is not activated

This module should be used for special applications only and also works on the screen of the runtime main program. The module is not necessary for normal applications.

SCRUPDT

Refresh screen contents.

Input parameters	None
Output parameters	None

The only items redisplayed are data from the operand memory and strings, but not edited strings.

SCRWRITE

Write a string at a set position on the screen

Input parameters	FU32	Number of the string to be displayed
	FU33	X starting position, column, 0 to 79 or negative
	FU34	Y starting position, line, 0 to 24
	FU35	0 for normal display 1 for highlighted display
Output parameters	FU32	= 0 if display appears = -1 if the string does not exist, the position is invalid or the user interface is not activated

If the X position is negative, the string is displayed in such a way that it ends at the specified position (right-justified).

SCRW0808

Write a string comprising large 8x8 block characters at a set position on the screen.

Input parameters	FU32	Number of the string to be displayed
	FU33	X starting position, column, 0 to 72 or negative
	FU34	Y starting position, line, 0 to 14
	FU35	0 for normal display 1 for highlighted display
Output parameters	FU32	= 0 if display appears = -1 if the string does not exist, the position is invalid or the user interface is not activated

If the X position is negative, the string is displayed in such a way that it ends at the specified position (right-justified). Internally the module uses the string with the number 0 as an interim storage location.

SCRW0814

Write a string comprising large 8x14 block characters at a set position on the screen.

Input parameters	FU32	Number of the string to be displayed
	FU33	X starting position, column, 0 to 72 or negative
	FU34	Y starting position, line, 0 to 8
	FU35	0 for normal display 1 for highlighted display
Output parameters	FU32	= 0 if display appears = -1 if the string does not exist, the position is invalid or the user interface is not activated

If the X position is negative, the string is displayed in such a way that it ends at the specified position (right-justified). Internally the module uses the string with the number 0 as an interim storage location.

SCRXCHG

Switch between the predefined screen pages in the driver.

Input parameters	FU32	Number of the screen page, indexing from 0
Output parameters	None	

The predefined screen pages are transferred from the project on startup of the runtime main program. This module does not switch (flip) between the screen page of the user interface and the screen of the runtime main program.

Complete example

A complete (though simple) example of the application of the user interface is shown below. It relates to a hypothetical production plant for chewing gum. The program uses four screen pages.

Scr	Purpose and program status
een	

```

0   Basic screen for STEP screen_0
1   Ask for production quantity
    in STEP screen_1
2   Status of production in STEP
    screen_2, production is
    simulated using timer T0
3   Screen for system fault in
    STEP screen_3

```

The script for the definition of the screen pages is given below.

```

; default settings
FEATURE_FLAGS 1
ACTIVATE

; screen 0
SCREEN_INITIAL "[ Chewing Gum Production ]"
TEXT 20 7 0 "The Most Important Food in the World"
TEXT 30 10 40704 " CHEWING GUM "
TEXT 22 15 0 "Chewing Gum Production is Ready."
FUNCTION_KEY 1 "START" ""

; screen 1
SCREEN "[ Chewing Gum Production ]"
TEXT 15 10 0 "How many ?"
INPUT_STRING 30 10 5 1
FUNCTION_KEY 1 "START" ""

; screen 2
SCREEN "[ Chewing Gum Production ]"
TEXT 25 9 0 "requested"
TEXT 25 11 0 "produced"
TEXT_UNSIGNED -42 9 0 1
FUNCTION_KEY 2 "STOP" ""

; screen 3
SCREEN "[ Chewing Gum Production ]"
TEXT 26 10 1 "Missing Liquid Rubber."
MESSAGE_ERROR " No Chewing without Rubber."
FUNCTION_KEY 1 "continue" ""

```

In the basic settings it is established that it is not to be permitted to switch from a screen page to the screen of the runtime main program. As far as the plant operating staff are concerned, therefore, the CI is not accessible. For the automation engineer, on the other hand, the CI can be reached at any time via a connected host. The program for the production plant is listed below. It was not considered necessary to show the allocation list in this context.

```

0001      "" Chewing Gum Production
0002
0003      "" a trivial example for string and screen driver
0004
0005      "" Dr. F. Haase, D 40597 Düsseldorf
0006
0007
0008      "" -----
0009
0010      STEP Init                      (1)      " some initial settings
0011
0012          THEN      LOAD      K315      " value for F1 key
0013                  TO        F1_KEY      'value for F1 key
0014
0015                  LOAD      K316      " value for F2 key
0016                  TO        F2_KEY      'value for F2 key
0017
0018                  LOAD      K0         " value of zero
0019                  TO        ZERO       'zero value
0020
0021
0022      "" -----
0023
0024      STEP Screen_0                  (2)      " initial screen
0025
0026          THEN      CMP 1              'SCRKBD
0027
0028                  LOAD      PARAM_1    'returned parameter 1
0029                  TO        KEYSTROKE    'the obvious
0030
0031          IF          KEYSTROKE          'the obvious
0032                  =          F1_KEY      'value for F1 key
0033
0034          THEN      LOAD      ZERO       'zero value
0035                  TO        AMOUNT      'amount to produce
0036                  TO        NPRODUCED    'produced so far
0037
0038                  CMP 2
0039                  WITH      K1          " flip to screen 1
0040
0041
0042      "" -----
0043
0044      STEP Screen_1                  (3)      " request amount
0045
0046          THEN      CMP 1              'SCRKBD
0047
0048                  LOAD      PARAM_1    'returned parameter 1
0049                  TO        KEYSTROKE    'the obvious
0050

```

```

0051         IF          =          KEYSTROKE      'the obvious
0052                                     F1_KEY 'value for F1 key
0053
0054         THEN      CMP 5          'STRATOU
0055         WITH      K1          "convert user input in string 1
0056
0057                 LOAD      PARAM_1 'returned parameter 1
0058         TO        NERROR      "0 if a valid unsigned number
0059
0060         LOAD      PARAM_2 'returned parameter 2
0061         TO        AMOUNT      "user input value, amount to produce
0062
0063         IF          " ? user input valid
0064         (          NERROR      'error flag word
0065         =          ZERO )      " has to be zero
0066         AND        (          AMOUN      'amount to produce
0067         >          ZERO )      " should be > 0
0068
0069         THEN      CMP 2          'SCRXCHG
0070         WITH      K2          " flip to screen 2
0071
0072                 " timer 0 for production simulation
0073         LOAD      K100         " 1 sec
0074         TO        TV0         " to produce a single chewing gum
0075
0076         SET       T0          " start timer (i.e. production)
0077
0078
0079 " -----
0080
0081     STEP Screen_2      (4)          " production display
0082
0083         THEN      CMP 1          'SCRKBD
0084
0085         LOAD      PARAM_1 'returned parameter 1
0086         TO        KEYSTROKE      'the obvious
0087
0088         IF          =          KEYSTROKE      'the obvious
0089                                     F2_KEY 'value for F2 key
0090
0091                 " stop production
0092         THEN      CMP 2          'SCRXCHG
0093         WITH      K0          " flip to screen 0
0094
0095         JMP TO    Screen_0 (2)
0096
0097         IF          N          T0          " 1 sec elapsed, simulate production
0098
0099         THEN      INC          NPRODUCED      'produced so far
0100
0101                 CMP 3          'STRUTOA
0102         WITH      NPRODUCED      'produced so far
0103         WITH      K2          " to string 2
0104
0105                 CMP 4          'SCRWRITE
0106         WITH      K2          " string 2
0107         WITH      K-42         " X
0108         WITH      K11         " Y
0109         WITH      K1          " highlighted attribute
0110
0111         SET       T0          " start again
0112
0113         IF          " ? done
0114                 NPRODUCED      'produced so far
0115         >          AMOUNT      'amount to produce
0116
0117         THEN      CMP 2          'SCRXCHG
0118         WITH      K0          " flip to screen 0
0119
0120         JMP TO    Screen_0 (2)      " done, back
0121
0122         IF          PLANTERR      'error in plant
0123
0124         THEN      CMP 2          'SCRXCHG
0125         WITH      K3          " flip to screen 3
0126
0127
0128 " -----
0129
0130     STEP Screen_3      (5)          " error display
0131
0132         THEN      CMP 1          'SCRKBD
0133
0134         LOAD      PARAM_1 'returned parameter 1
0135         TO        KEYSTROKE      'the obvious
0136
0137         IF          (          KEYSTROKE      'the obvious
0138         =          F1_KEY )      'value for F1 key
0139
0140         AND N      PLANTERR      'error in plant
0141
0142         THEN      CMP 2          'SCRXCHG
0143         WITH      K2          " flip to screen 2
0144
0145         SET       T0          " restart timer
0146
0147         JMP TO    Screen_2 (4)      " continue production

```

In the example, the steps and the screens are synchronised with each other. In real-world situations a somewhat more complex method of synchronising will be required. In such cases it is worth separating the actual automation tasks and representation on the screen into two different programs. In order to save on computing time for screen display, the program for this can be started as a cyclical program, for example every 500 msec (cf. function module F4).

Special applications

For the purposes of this driver for user interfaces, special applications are applications where representation and operation do not follow the rules of the FST software but instead are supposed to follow different rules, which are usually customer-specific. Bit 1 must be set in the FEATURE_FLAGS. This is done with the SCRFLAGS module. Representation on the screen is then created with two modules.

SCRCLEAR	Clear screen
SCRUGPO	Determine current position of cursor
SCRUSPO	Set current position of cursor
SCRUTYP	Set cursor type
SCRPUTS	Write a string at a set position on the screen

These modules should be used for special applications only and also work on the screen of the runtime main program. The modules are not necessary for normal applications. The module for keyboard interrogation returns all inputs from the keyboard.

SCRKBD	Interrogate the last input on the keyboard
--------	--

You can be returned to normal further processing via the SCRDOKEY module.

SCRDOKEY	Process input from the keyboard
----------	---------------------------------

32-Bit Arithmetic

These modules allow performing arithmetic operations on 32 bit values that are hold in two 16-bit FST operands.

Overview

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	Change of sign for a 32-bit value.
LSUB	Subtraction of 32-bit values.

LADD

Addition of 32-bit values.

Input parameters	FU32	Low word of the 1st operand
	FU33	High word of the 1st operand
	FU34	Low word of the 2nd operand
	FU35	High word of the 2nd operand
Output parameters	FU32	Low word of the result
	FU33	High word of the result

LCMP

Comparison of 32-bit values.

Input parameters	FU32	Low word of the 1st operand
	FU33	High word of the 1st operand
	FU34	Low word of the 2nd operand
	FU35	High word of the 2nd operand
Output parameters	FU32	Result in bit form
	\$xx01	1st operand < 2nd operand (signed)
	\$xx02	1st operand == 2nd operand (signed)
	\$xx04	1st operand > 2nd operand (signed)
	\$0lxx	1st operand < 2nd operand (unsigned)
	\$02xx	1st operand == 2nd operand (unsigned)
	\$04xx	1st operand > 2nd operand (unsigned)

LDIV

Division of 32-bit values (result = 1st operand / 2nd operand).

Input parameters	FU32	Low word of the 1st operand
	FU33	High word of the 1st operand
	FU34	Low word of the 2nd operand
	FU35	High word of the 2nd operand
Output parameters	FU32	Low word of the result
	FU33	High word of the result

LMUL

Multiplication of 32-bit values.

Input parameters	FU32	Low word of the 1st operand
	FU33	High word of the 1st operand
	FU34	Low word of the 2nd operand
	FU35	High word of the 2nd operand
Output parameters	FU32	Low word of the result

FU33	High word of the result
------	-------------------------

LNEG

Change of sign for a 32-bit value.

Input parameters	FU32	Low word of the operand
	FU33	High word of the operand
Output parameters	FU32	Low word of the result
	FU33	High word of the result

LSUB

Subtraction of 32-bit values (result = 1st operand - 2nd operand).

Input parameters	FU32	Low word of the 1st operand
	FU33	High word of the 1st operand
	FU34	Low word of the 2nd operand
	FU35	High word of the 2nd operand
Output parameters	FU32	Low word of the result
	FU33	High word of the result

Incremental Encoder

This program module has been designed to use the incremental encoder module IM2x for your FST IPC projects.

Import

The module IM2X needs to be imported to your project as a program module (e.g. CMP 0).

Functions of the IM2x module

Overview

Function (passed in FE33)

-1	Init/Reset counters
0	Read counter registers (values)
1	Load counter 1 with new value
2	Load counter 2 with new value
3	Load counter 3 with new value

Init/Reset

Input parameters	FU32	KSW setting (1 or 2)
	FU33	-1
	FU34	Operating mode (see table below)
Output parameters	FU32	Current status of counter 1
	FU33	Current status of counter 2
	FU34	Current status of counter 3

The numbers 1, 2 and 4 in the following table refer to 1-, 2- or 4-edge counting on the given channel. If only a single number is specified for two channels this means that they are cascaded.

Please consult the IM20/IM21 data sheets for more details and operating modes. It is, however, not possible to use the IRQ possibilities of the IM2x hardware with this software module.

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

Read counter registers (values)

Input parameters	FU32	KSW setting (1 or 2)
	FU33	0
Output parameters	FU32	Current status of counter 1
	FU33	Current status of counter 2
	FU34	Current status of counter 3

Load counter 1 with new value

Input parameters	FU32	KSW setting (1 or 2)
	FU33	1
	FU34	New given counter value for counter 1
Output parameters	FU32	Current status of counter 1
	FU33	Current status of counter 2
	FU34	Current status of counter 3

Load counter 2 with new value

Input parameters	FU32	KSW setting (1 or 2)
	FU33	2
	FU34	New given counter value for counter 1
Output parameters	FU32	Current status of counter 1
	FU33	Current status of counter 2
	FU34	Current status of counter 3

Load counter 3 with new value

Input parameters	FU32	KSW setting (1 or 2)
	FU33	3
	FU34	New given counter value for counter 3
Output parameters	FU32	Current status of counter 1
	FU33	Current status of counter 2
	FU34	Current status of counter 3

Fast counters

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 HC0x (I0.0 and I0.1) can be used as 1 or 2 independent, high-speed counters. Still the inputs can be read as standard inputs. For the FEC Standard series the following inputs are used:

FC4xx, FC5xx - I1.6 and I1.7
FC6xx - I3.6 and I3.7

These counters are interrupt driven and once activated, operate independently of the user's control programs and therefore are not effected by factors such as control program scan time.

Using the module FECCNTR

The program module provides functions for:

- Defining the operating parameters of each Counter
- Activating each Counter
- Resetting each Counter
- Checking the status and current value of each Counter

Upon achieving the preset value, the fast counter module can be configured to either:

- Modify an output word (OW)
- Start a defined program
- Modify an output word (OW) and Start a defined program.

Each fast counter can be independently defined for automatic or manual restarting.

Required drivers

You need to include the driver FECCNTR to you driver configuration of your project. And use the "Import Module" function to add the FECCNTR module to your control project.

Using the Fast Counter Module

The Fast Counter module is easily accessed from either Statement List or Ladder Diagram programs using conventional FST program module syntax. The following sections present the various functions provided by the Fast Counter module.

Reset of a Fast Counter

input parameters	FU32	0 = reset of counter defined in FU33
	FU33	0 = counter 0
		1 = counter 1
output parameters	none	

This function provides a means to deactivate a Fast Counter. No harm will occur if a non-active Fast Counter is Reset.

Example:

IF	...	" User defined conditions
THEN	SET	" Call Fast Counter module
	WITH	" Specify RESET function
	WITH	" Specify which Fast Counter:
		" y=0: first counter
		" y=1: second counter

Defining the operation of a Fast Counter

input parameters	FU32	1 = parameter settings of counter defined in FU33 (only possible if the counter is inactive)
	FU33	0=first counter 1=second counter

FU34	counter preselect value (16-bit decimal or HEX) (range 0 ...65535, \$0...\$FFFF)
FU35	define desired action when preselected count value is reached. 0 = no activities (default) 1 = modify an output word 2 = start a program 3 = start a program and modify an output word
FU36	high byte = program number low byte = output word number All values must be specified in HEX. Example: program no. 12, output word no. 0: \$0C00
FU37	high byte = output mask low byte = value (set or reset) All values must be specified in HEX. Example: set output x.7, outputs x.0...x.6 unchanged: \$8080 set output x.7, outputs x.0...x.6 are reset: \$FF80
FU38	0 = no automatic restart of a counter 1 = automatic restart of a counter
output parameters	none

This function provides the means to define how each of the Fast Counters will operate. This function should only be called before a Fast Counter is activated or use the Reset function prior to calling this function.

Each valid input signal pulse is summed and upon reaching this preselect value, the action defined in parameter 4 will be performed. Four choices are permitted:

1. No action is taken.
2. A specified digital output word (OW) is be modified
3. A specified Statement List or Ladder Diagram program is started
4. A specified digital output word is modified AND a specified program is started.

Parameter 5 serves to further define the action(s) specified in parameter 4. This parameter must be specified in hexadecimal format whereby the high byte indicates the program number (to start) and the low byte indicates the output word to be modified. If parameter 4 <> V3, then either or both unused bytes are simply ignored.

Examples:

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

Parameter 6 only has meaning if Parameter 5 was assigned a value of 1 or 3, indicating that a defined output word (OW) is to be modified when the defined Fast Counter reaches the preselect value.

This parameter must be specified in hexadecimal format whereby the high byte indicates the output "mask" to apply while the low byte which bits within the specified output word (OW) are to be Set or Reset. By separating these values, the user can separately specify which bits are enabled or disabled from modification.

Examples:

```
V$8080 Set output x.7, leave outputs x.0 - x.6 untouched
V$FF01 Set output x.0, reset outputs x.1 - x.7
```

Parameter 7 allows the programmer to decide whether or not the specified Fast Counter should be automatically restarted after having reached the preselected value.

Starting a Fast Counter

input parameters	FU32	2= start of counter defined in FU33
	FU33	0 = first counter 1 = second counter
output parameters	none	

This function provides a means to activate a Fast Counter. Before this function is called, the Fast Counter should first be defined. Note: If you have started a Fast Counter (without Auto-Restart enabled) and the counter value reaches the preselect value, you must Reset the counter before attempting to start the counter again.

Status of a Fast Counter

input parameters	FU32	3 = status of counter defined in FU33
	FU33	0 = first counter
		1 = second counter
return parameters	FU32	Status
		0 = counter active
		1 = counter inactive
	FU33	current fast counter value
	FU34	always 0

This function allows the application program to check the status of a Fast counter.

The status will be equal to V1 in only if ALL of the following conditions are true: The Fast Counter has been configured AND the preselected count was reached AND Auto-Restart is NOT enabled. In all other cases FU32 returns V0.

Example:

```
IF      ...      " User defined conditions
THEN   SET      CMPx " Call Fast Counter module
        WITH    V3   " Specify Status function
        WITH    Vy   " Specify which Fast Counter to check
        " Now check results...

IF      (FU32 = V0) " Then counter is still active
THEN   ...      " User defined actions
        LOAD    FU33 " Get current counter value and
        TO      R21  " Save it in Register 21
```

Positioning

This program module adds support for stepper and servo motors (AM1x and AM20) to your FST IPC projects.

Import

Import the module AMXX into your project using Project Management - > Import File etc. You now have available a function module which expects some input (up to 7 integers) and produces some output (up to 7 integers) via FU32 .. FU38.

The module does not really check whether the given parameters fall within the allowed value range. You will have to look into the current AM1x and AM20 documentation to get an idea about allowed ranges.

How to use the module

The first input parameter is always the command number (i.e. 0 for get status, 1 for move absolute, 2 for move relative etc.), the second parameter is always the motor number.

Supported are motor numbers from 1 up to 12, where motor 1 will always correspond to S0 on the module with switch selector 1, motor 2 will always correspond to S1 on the module with switch selector 1, motor 3 will always correspond to S0 on the module with switch selector 2 etc.

Motor	KSW	Label
1	1	S0
2	1	S1
3	2	S0
4	2	S1
...
11	6	S0
12	6	S1

The first return parameter (FU32) is always a status value.

Additional parameters are returned if the command was a request for information.

Overview

FU32	Function
0	Get status
1	Move absolute
2	Move relative
3	Reference move
4	Override speed
10	Configure polarity
11	Configure start speed and acceleration
12*	Set PID values
13	Set actual position
14*	Set positioning parameters
15*	Set DA output polarity (AM20 < S2.2)
16*	Set P division (AM20 > S2.2)
17*	Polarity for counting of increments (AM20 >= 2.2)
18*	Multiply acceleration by 16 (AM20 >= 2.3)
19*	Set Offset for D/A converter
20	Reset module
21	Stop motor
22*	Enable or disable motor
23*	Delete contouring error
30	Get revision
31	Get speeds / target position
32	Get polarity and PID parameters
33	Get status for AM20 user
40	Move with interpolation I
41	Move with interpolation II

Get status

Input parameters	FU32	0
	FU33	Motor number
Output parameters	FU32	Status (see Table 1)
	FU33	Switch status (see Table 2)
	FU34	Extended status, (see Table 3)
	FU35	Low word of current position
	FU36	High word of current position
AM20 only	FU37	In position status (see Table 4)
AM20 only	FU38	Current contouring error

Table 1: Status return values

Value	Description
-1	No module present for motor number
0	Motor is stopped
1	Motor is moving
2	Reference move is active

Table 2: Switch status

Bit	Description
0	0= Limit switch X (AM1x) or + (AM20) inactive 1= Limit switch X (AM1x) or + (AM20) activated
1	0= Limit switch Y (AM1x) or - (AM20) inactive 1= Limit switch Y (AM1x) or - (AM20) activated
2	0= Reference switch inactive 1= Reference switch activated

Table 3: Extended status

Bit	Description
0	0= move direction positive 1= move direction negative
1	0= Linear ramp 1= no ramp (move with starting frequency without ramp)
2	0= Reference move is inactive / finished 1= Reference move is active
3	0= Reset is finished 1= Reset is active
4	0= Brake ramp terminated 1= Brake ramp active 1= Parameter error (after axis has started moving)
5	0= No limit switch r error 1= Limit switch r error is active
6	0= Relative move to actual position 1= Absolute move
7	0= Axis is at a standstill 1= Axis is moving

Table 4: In position status / Contouring error / Enable (AM20 only)

Bit	Description
0	Bit is set if axis is within position window
1	Bit is set if axis has contouring error greater than specified maximum

Move absolute

Input parameters	FU32	1
	FU33	Motor number
	FU34	Low word of new position (steps / increments) Range: 0-65535
	FU35	High word of new position (steps / increments) Range: -128 - +127
	FU36	Moving speed Range / [unit] AM10: - 0-32000 / [steps / sec] Range / [unit] AM11: - 0-16000 / [steps / sec] Range / [unit] AM20: - 0-32767 ; [10* incr/sec] < S2.3 - 0-65535 ; [10* incr/sec] >= S2.4
Output parameters	FU32	Status (see Table 1)
	FU33	Extended status (see Table 3)

Move relative

Input parameters	FU32	2
	FU33	Motor number
	FU34	Low word of relative position (steps / increments) Range and unit see absolute move
	FU35	High word of relative position (steps / increments) Range and unit see absolute move
	FU36	Maximum speed Range and unit see absolute move
	FU37	Direction 1 = positive -1 = negative
Output parameters	FU32	Status (see Table 1)
	FU33	Extended status (see Table 3)

Reference move

Input parameters	FU32	3
	FU33	Motor number
	FU34	Speed for reference move Range and unit see absolute move
	FU35	Direction (see Table 5)
Output parameters	FU32	Status (see Table 1)
	FU33	Extended status (see Table 3)

Important: the speed given for the reference move will be used for the entire movement. This means you have to give a speed that is suitable as a start/stop frequency since there is no ramp applied to the reference move. Table 5: Reference move options

Value	Description
1	move in positive direction until reference switch reached
-1	move in negative direction until reference switch reached
2	move in positive direction until reference switch reached, then move in negative direction until reference switch freed
-2	move in negative direction until reference switch reached, then move in positive direction until reference switch freed
3	move in positive direction until reference switch reached, then move in positive direction until reference switch freed
-3	move in negative direction until reference switch reached, then move in negative direction until reference switch freed

Override Speed

Input parameters	FU32	4
	FU33	Motor number

Output parameters	FU34	Override (0% ... 100%)
	FU32	Status (see Table 1)

Configure polarity

Input parameters	FU32	10
	FU33	Motor number
	FU34	Polarity of digital outputs, only AM10 and AM11. Value for AM20 is always "0" 0 = active low 1 = active high
	FU35	Polarity of limit switches 0 = normally closed 1 = normally opened
	FU36	Polarity of reference switch 0 = normally closed 1 = normally opened
Output parameters	FU32	Status (see Table 1)

Configure start speed and acceleration

Input parameters	FU32	11
	FU33	Motor number
	FU34	Start speed Range / [unit] AM10: - 0-32000 / [steps / sec] Range / [unit] AM11: - 0-16000 / [steps / sec] Range / [unit] AM20: - 0-32767 ; [10* incr/sec]
	FU35	Acceleration / Deceleration Range / [unit] AM10, AM11: - 0-32767 / [10*steps / sec ²] Range / [unit] AM20: - 0-32767 ; [10* incr/sec ²] < S2.3 - 0-65535 ; [10* incr/sec ²] >= S2.4
	FU32	Status (see Table 1)
Output parameters	FU32	Status (see Table 1)

Configure PID values (AM20 only)

Input parameters	FU32	12
	FU33	Motor number
	FU34	P-Factor (range 0-255)
	FU35	I-Factor (range 0-255)
	FU36	D-Factor (range 0-255)
Output parameters	FU37	V-Factor (range 0-255)
	FU32	Status (see Table 1)

Set actual position

Input parameters	FU32	13
	FU33	Motor number
	FU34	Low word of new position
	FU35	High word of new position
Output parameters	FU32	Status (see Table 1)

Set positioning parameters (AM20 only)

Input parameters	FU32	14
	FU33	Motor number
	FU34	maximum allowed contouring error Range / [unit]: - 0-32767 / [increments]
	FU35	In position window Range / [unit]:

Output parameters	FU32	- 0-2557 / [increments] Status (see Table 1)
-------------------	------	---

Set D/A output polarity (AM20 > S 2.2 only)

Input parameters	FU32	15
	FU33	Motor number
	FU34	Polarity of D/A output 0 = normal 1 = inverted
Output parameters	FU32	Status (see Table 1)

Set P division (AM20 > S2.2 only)

Input parameters	FU32	16
	FU33	Motor number
	FU34	Division 0 = P factor will function normally 1 = P factor will be divided by 16
Output parameters	FU32	Status (see Table 1)

Set counter input polarity (AM20 >= S 2.3 only)

Input parameters	FU32	17
	FU33	Motor number
	FU34	Polarity of counter input 0 = normal 1 = inverted
Output parameters	FU32	Status (see Table 1)

Set acceleration multiplier (AM20 >= S 2.3 only)

Input parameters	FU32	18
	FU33	Motor number
	FU34	Multiplier 0 = acceleration normal 1 = acceleration will be multiplied by 16
Output parameters	FU32	Status (see Table 1)

Set Offset for D/A converter (AM20 >= S 2.3 only)

Input parameters	FU32	19
	FU33	Motor number
	FU34	Offset: 0 - 127: positive 128-255: negative
Output parameters	FU32	Status (see Table 1)

Reset module

Input parameters	FU32	20
	FU33	Motor number
Output parameters	FU32	Status (see Table 1)

Note that if you reset one motor of a two motor module the other one will be reset as well.

Stop motor

Input parameters	FU32	21
	FU33	Motor number
Output parameters	FU32	Status (see Table 1)

Stop movement using the defined acceleration value.

Set / Reset enable (AM20 only)

Input parameters	FU32	22
	FU33	Motor number
	FU34	0 = reset enable output 1 = set enable output
Output parameters	FU32	Status (see Table 1)

Delete contouring error (AM20 only)

Input parameters	FE32	23: up driver version V1.17
	FE33	Motor number
Output parameters	FE32	Status (see Table 1)

Get revision (AM20 only)

Input parameters	FU32	30
	FU33	Motor number
Output parameters	FU32	Status (see Table 1)
	FU33	Revision number

Get speeds / target position

Input parameters	FU32	31
	FU33	Motor number
Output parameters	FU32	Status (see Table 1)
	FU33	Start frequency
	FU34	Acceleration
	FU35	Maximum frequency
	FU36	Target position low word
	FU37	Target position high word

Get polarity and PID parameters

Input parameters	FU32	32
	FU33	Motor number
Output parameters	FU32	Status (see Table 1)
	FU33	Polarity register (see Table 6)
	FU34	P-Factor
	FU35	I-Factor
	FU36	D-Factor
	FU37	V-Factor

Table 6: Polarity

Bit	Description
0	0= Digital outputs low active 1= Digital outputs high active Only AM10 and AM11!
1	0= Divisor for P-Factor is off 1= Divisor for P-Factor is on
2	0= Limit switch low active 1= Limit switch high active
3	0= Polarity of the D/A converter is normal 1= Polarity of the D/A converter is inverted
4	0= Reference switch low active 1= Reference switch high active

- | | |
|---|---|
| 5 | 0= Interpolation inactive
1= Interpolation active |
| 6 | 0= Multiplier for acceleration is off
1= Multiplier for acceleration is on |
| 7 | 0= Count direction is normal
1= Count direction is inverted |

Get status for AM20 user

Input parameters	FU32	33
	FU33	Motor number
Output parameters	FU32	Status (see Table 1)
	FU33	Status flags (see Table 7)
	FU34	P-Factor
	FU35	I-Factor
	FU36	D-Factor
	FU37	V-Factor

Table 7: Status for AM20 user

Bit	Description
0	0= Enable is off 1= Enable is on
1	1= Motor is stopped 1= Motor is moving
2	0= Reference move is stopped / finished 1= Reference move is active
3	0= Axis is not in position window 1= Axis is within position window
4	0= contouring error is not active 1= Axis is stopped with contouring error
5	0= Positive limit switch inactive 1= Positive limit switch activated
6	0= Negative limit switch inactive 1= Negative limit switch activated
7	0= Reference switch inactive 1= Reference switch activated

Move with interpolation (AM20 >= 2.3 only)

Input parameters	FU32	40
	FU33	Motor (1,3,5,7,9,11)
	FU34	Low word of new position / distance for motor 1
	FU35	High word of new position / distance for motor 1
	FU36	Low word of new position / distance for motor 2
	FU37	High word of new position / distance for motor 2
Output parameters	FU32	Status (see Table 1)
	FU33	Error code: 0 = OK -1 = wrong motor number

Move with interpolation (AM20 >= 2.3 only)

Input parameters	FU32	41
	FU33	Motor (1,3,5,7,9,11)
	FU34	Move relative or absolute 0 = absolute 1 = relative
	FU35	Maximum speed

	FU36	Direction for motor 1 (if relative move) 1 = positive -1 = negative
	FU37	Direction for motor 2 (if relative move) 1 = positive -1 = negative
Output parameters	FU32	Status (see Table 1)
	FU33	Error code: 0 = OK -1 = wrong motor number -2 = Function 40 not yet called

Sample programs

Sample program for stepper motor (AM1x)

```

" " Sample program for stepper motor
" " Uses R0 for motor number

STEP init
" " Start with configuration
IF      NOP
THEN    LOAD      V1
        TO        R0          'Motor Number

IF      NOP
THEN    CFM 51
        WITH      V20        " AMxx module
        WITH      R0          " Reset
                                   'Motor Number

        CFM 51              " AMxx module
        WITH      V10        " Set polarity
        WITH      R0          'Motor Number
        WITH      V1          " Outputs ACTIVE HIGH
        WITH      V0          " Limit switches normally closed
        WITH      V1          " Reference switch normally open

        CFM 51              " AMxx module
        WITH      V11        " Set start speed and acceleration
        WITH      R0          'Motor Number
        WITH      V10        " 10 steps / sec
        WITH      V1000      " 10*1000 steps / sec²

STEP loop

IF      NOP
THEN    CFM 51              " AMxx module
        WITH      V0          " Get status
        WITH      R0          'Motor Number

        LOAD      FU34
        TO        FW34      'FU34=extended status

IF      F1.0              'Start Flag
        AND      N        F34.7      'Run Flag
THEN    CFM 51              " AMxx module
        WITH      V1          " Start absolute move
        WITH      R0          'Motor Number
        WITH      V2000      " Position low word
        WITH      V0          " Position high word
        WITH      V2500      " Moving speed 2500 Hz

STEP
IF      NOP
THEN    CFM 51              " AMxx module
        WITH      V0          " Get status
        WITH      R0          'Motor Number

        LOAD      FU34
        TO        FW34      'FU34=extended status

```

```

IF          F34.7      'Run Flag
THEN
NOP

STEP
IF
THEN  CFM 51          " AMxx module
      WITH            " Get status
      WITH            'Motor Number

      LOAD            FU34
      TO              FW34          'FU34=extended status

IF          N          F34.7      'Run Flag
THEN  CFM 51          " AMxx module
      WITH            " Start absolute move
      WITH            R0          'Motor Number
      WITH            V0          " Position low word
      WITH            V0          " Position high word
      WITH            V5000       " Moving speed 5000 Hz

STEP wait
IF
THEN  CFM 51          " AMxx module
      WITH            " Get status
      WITH            R0          'Motor Number

      LOAD            FU34
      TO              FW34          'FU34=extended status

IF          F34.7      'Run Flag
THEN  JMP TO loop
OTHRW JMP TO wait

```

Sample program for servo motor (AM20)

```

"" Sample program for servo motor
"" Uses R0 for motor number

STEP init
"" Start with configuration
IF          NOP
THEN  LOAD  V3
      TO    R0          'Motor Number

IF          NOP
THEN  CFM 51          " AMxx module
      WITH            " Reset
      WITH            R0          'Motor Number

      CFM 51          " AMxx module
      WITH            " Set polarity
      WITH            R0          'Motor Number
      WITH            V0          " Always 0 for AM20
      WITH            V0          " Limit switches normally closed
      WITH            V1          " Reference switch normally open

      CFM 51          " AMxx module
      WITH            V11         " Set start speed and acceleration
      WITH            R0          'Motor Number
      WITH            V5          " 10*5 steps / sec
      WITH            V1000       " 10*1000 steps / sec²

      CFM 51          " AMxx module
      WITH            V12         " Set PID parameters
      WITH            R0          'Motor Number
      WITH            V1          " P factor
      WITH            V2          " I factor
      WITH            V0          " D factor
      WITH            V3          " V factor

      CFM 51          " AMxx module

```

	WITH		V14	" Set positioning parameters
	WITH		R0	'Motor Number
	WITH		V10000	" Max contouring error [Incr]
	WITH		V20	" Inposition window [Incr]
CFM 51				" AMxx module
WITH			V22	" enable/disable
WITH			R0	'Motor Number
WITH			V1	" set enable
STEP loop				
IF			NOP	
THEN	CFM 51			" AMxx module
	WITH		V0	" Get status
	WITH		R0	'Motor Number
	LOAD		FU34	
	TO		FW34	'FU34=extended status
IF			F1.0	'Start Flag
	AND	N	F34.7	'Run Flag
THEN	CFM 51			" AMxx module
	WITH		V1	" Start absolute move
	WITH		R0	'Motor Number
	WITH		V2000	" Position low word
	WITH		V0	" Position high word
	WITH		V2500	" 10*2500 incr/sec
STEP				
IF			NOP	
THEN	CFM 51			" AMxx module
	WITH		V0	" Get status
	WITH		R0	'Motor Number
	LOAD		FU34	
	TO		FW34	'FU34=extended status
IF			F34.7	'Run Flag
THEN			NOP	
STEP				
IF			NOP	
THEN	CFM 51			" AMxx module
	WITH		V0	" Get status
	WITH		R0	'Motor Number
	LOAD		FU34	
	TO		FW34	'FU34=extended status
IF		N	F34.7	'Run Flag
THEN	CFM 51			" AMxx module
	WITH		V1	" Start absolute move
	WITH		R0	'Motor Number
	WITH		V0	" Position low word
	WITH		V0	" Position high word
	WITH		V5000	" 10*5000 incr/sec
STEP wait				
IF			NOP	
THEN	CFM 51			" AMxx module
	WITH		V0	" Get status
	WITH		R0	'Motor Number
	LOAD		FU34	
	TO		FW34	'FU34=extended status
IF			F34.7	'Run Flag
THEN	JMP TO loop		ELSE	
	JMP TO wait			

Servopneumatic Positioning

Driver configuration

A driver and a function module are required for controlling the PS1 AM30 module.

If you wish to use driver AM30DRV, this must be entered in the driver configurator and parametrised. When parametrising, you must specify the destination drive (default C).

The interface between the user program and the driver is implemented with function module AM30CFM. This function module must be imported in the project. With the aid of the module, the user program transfers to the driver the commands for controlling.

The software supplied also contains an FST demo program which serves for controlling module PS1 AM30. A brief description of how to use the program can be found at the end of this topic.

The functions of module AM30CFM

Note! Parameter values which lie outside the specified value range will be ignored. In this way, certain parameters can be switched out when a module is accessed.

Overview

0	Stop axis
1	Enable axis
2	Move absolutely at current speed and acceleration
3	Move relatively negative at current speed and acceleration
4	Move relatively positive at current speed and acceleration
5	Move absolutely at specified speed and acceleration
6	Move relatively negative at specified speed and acceleration
7	Move relatively positive at specified speed and acceleration
8	Set positioning parameters
9	Set ramp characteristics
10	Override speed
20	Set polarity
21	Transfer closed loop controller parameters
22	Transfer application parameters
23	Transfer fitting parameters
24	Transfer calibration factor
25	Transfer axis parameters 1/2
26	Transfer axis parameters 2/2
30	Read polarity settings
31	Read closed loop controller parameters
32	Read application parameters
33	Read fitting parameters
34	Read calibration factor
35	Read axis parameters 1/2
36	Read axis parameters 2/2
37	Read out positioning parameters
38	Read out ramp characteristics
80	Delete hardware fault
81	Delete software fault
82	Delete timeout fault
83	Delete EMERGENCY STOP
84	Reset/Init measuring system bus
85	Reset axis
86	Reset module
87	Static identification of axis
88	Dynamic identification of axis with minimum load
89	Dynamic identification of axis with maximum load
90	Do not save axis parameters in Flash
91	Delete axis parameters in Flash
92	Reset identification and adaptaion parameters
101	Status interrogation 1/3
102	Status interrogation 2/3
103	Status interrogation 3/3

Instructions

1. For all functions, the function code will be passed in parameter FU32; the number of the axis will be passed in parameter FU33.

```
FU32    = function code
FU33    = axis number (1..12)
```

2. All functions return two parameters:

```
FU32    = status / axis exists
FU33    = error code
```

3. With the write functions, an additional internal check is made of the transferred parameters. The result of the check is entered in FU38. FU38 has the following meaning:

```
FU38    = 0: All transferred parameters are within the permitted limits.
          <>0: Non-permitted values have been transferred.
              The module transfers only those values which
              lie within the permitted value range.
```

In the event of an error, FU38 has the following coding:

```
Bit 0      error in FU32
Bit 1      error in FU33
Bit 2      error in FU34
Bit 3      error in FU35
Bit 4      error in FU36
Bit 5      error in FU37
Bit 6      error in FU38
Bit 7      not assigned

Bits 8-15   the command number with which incorrect
              parameters were transferred
```

If a non-existent axis or incorrect command is accessed, no further parameters will be checked and the appropriate error code will be returned.

Stop axis

Input parameters	FU32	0
	FU33	Axis (1..12)
Output parameters	FU32	= status
	FU33	= error code

Enable axis

Input parameters	FU32	1
	FU33	Axis (1..12)
	FU34	enable on/off
		0 = off
		1 = on
Output parameters	FU32	= status
	FU33	= error code

Move absolutely at current speed and acceleration

Input parameters	FU32	2
	FU33	Axis (1..12)
	FU34	= set path {-32000..32000} ; unit [0.1 mm]
Output parameters	FU32	= status
	FU33	= error code

Move relatively negative at current speed and acceleration

Input parameters	FU32	3
------------------	------	---

	FU33	Axis (1..12)
	FU34	= set path {-32000..32000} ; unit [0.1 mm]
Output parameters	FU32	= status
	FU33	= error code

Move relatively positive at current speed and acceleration

Input parameters	FU32	4
	FU33	Axis (1..12)
	FU34	= set path {-32000..32000} ; unit [0.1 mm]
Output parameters	FU32	= status
	FU33	= error code

Move absolutely at specified speed and acceleration

Input parameters	FU32	5
	FU33	Axis (1..12)
	FU34	= set path {-32000..32000} ; unit [0.1 mm]
	FU35	= speed {1..100} ; unit [0.1 m/s]
	FU36	= acceleration {1..1000} ; unit [0.1 m/s ²]
	FU37	= position window {1..50} / unit [0.1 mm]
Output parameters	FU32	= status
	FU33	= error code

Move relatively negative at specified speed and acceleration

Input parameters	FU32	6
	FU33	Axis (1..12)
	FU34	= set path {-32000..32000} ; unit [0.1 mm]
	FU35	= speed {1..100} ; unit [0.1 m/s]
	FU36	= acceleration {1..1000} ; unit [0.1 m/s ²]
	FU37	= position window {1..50} / unit [0.1 mm]
Output parameters	FU32	= status
	FU33	= error code

Move relatively positive at specified speed and acceleration

Input parameters	FU32	7
	FU33	Axis (1..12)
	FU34	= set path {-32000..32000} ; unit [0.1 mm]
	FU35	= speed {1..100} ; unit [0.1 m/s]
	FU36	= acceleration {1..1000} ; unit [0.1 m/s ²]
	FU37	= position window {1..50} / unit [0.1 mm]
Output parameters	FU32	= status
	FU33	= error code

Set positioning parameters

Input parameters	FU32	8
	FU33	Axis (1..12)
	FU34	= speed {1..100} ; unit [0.1 m/s]
	FU35	= acceleration {1..1000} ; unit [0.1 m/s ²]
	FU36	= position window {1..50} / unit [0.1 mm]
	FU37	= positioning quality (class)
		1 = fast stop
		2 = fast stop with damping time
		3 = precision stop
		4 = precision stop with damping time
		5 = precision stop with speed control
		6 = precision stop with speed control and damping time
Output parameters	FU32	= status

FU33 = error code

Set ramp characteristics

Input parameters	FU32	9
	FU33	Axis (1..12)
	FU34	= with or without ramp 0 = move with ramp 1 = move without ramp (jump)
	FU35	= ramp type {0,1} 0 = linear ramp 1 = sin ² ramp
Output parameters	FU32	= status
	FU33	= error code

Override speed

Input parameters	FU32	10
	FU33	Axis (1..12)
	FU34	= Override speed 0-100%
Output parameters	FU32	= status
	FU33	= error code

Transfer polarity settings

Input parameters	FU32	20
	FU33	Axis (1..12)
	FU34	= sensor polarity (only potentiometer) -1 = negative 1 = positive
	FU35	= valve polarity -1 = negative 1 = positive
Output parameters	FU32	= status
	FU33	= error code

Transfer closed loop controller parameters

Input parameters	FU32	21
	FU33	Axis (1..12)
	FU34	= gain factor of controller {1..100} corresponds internally to the range {0.1 .. 10.0}
	FU35	= damping factor of controller {1..100} corresponds internally to the range {0.1 .. 10.0}
	FU36	= filter factor of controller {1..100} corresponds internally to the range {0.1 .. 10.0}
	FU37	= positioning timeout {0..10000}; unit [0.1 sec]
Output parameters	FU32	= status
	FU33	= error code

Transfer application parameters

Input parameters	FU32	22
	FU33	Axis (1..12)
	FU34	= tool load {1..20000}; unit [0.1 kg]
	FU35	= work load {0.0..20000}; unit [0.1 kg]
	FU36	= supply pressure {30 ..100}; unit [0.1 bar]
Output parameters	FU32	= status
	FU33	= error code

Transfer fitting parameters

Input parameters	FU32	23
	FU33	Axis (1..12)
	FU34	= lower software end position: {0..32000}; unit [0.1 mm]
	FU35	= upper software end position: {0..32000}; unit [0.1 mm]
	FU36	= fitting position {-900..900}; unit [0.1°]
	FU37	= fitting offset {-16000..16000}; unit [0.1 mm]
	FU38	= project zero point {0..32000}; unit [0.1 mm]
Output parameters	FU32	= status
	FU33	= error code

Transfer calibration factor

Input parameters	FU32	24
	FU33	Axis (1..12)
	FU34	= calibration factor {90 .. 110}; unit [0.01]
Output parameters	FU32	= status
	FU33	= error code

Transfer axis parameters 1/2

Input parameters	FU32	25
	FU33	Axis (1..12)
	FU34	= cylinder design: 0 = linear with piston rod 1 = linear without piston rod 2 = rotary
	FU35	= cylinder length {500..32000}; unit [0.1 mm]
	FU36	= cylinder diameter {120..3200}; unit [0.1 mm]
	FU37	= measuring system design: 1 = potentiometer 2 = Temposonic
	FU38	= measuring system length {500..32000}; unit [0.1 mm]
Output parameters	FU32	= status
	FU33	= error code

Transfer axis parameters 2/2

Input parameters	FU32	26
	FU33	Axis (1..12)
	FU34	= valve type: 0 = VT-MPYE-5-M5 1 = VT-MPYE-5-1/8LF 2 = VT-MPYE-5-1/8HF 3 = VT-MPYE-5-1/4 4 = VT-MPYE-5-3/8HF
	FU35	= flow chain factor {1..15} corresponds internally to the range {0.1 .. 1.5}
Output parameters	FU32	= status
	FU33	= error code

Read polarity settings

Input parameters	FU32	30
	FU33	Axis (1..12)
Output parameters	FU32	= status
	FU33	= error code
	FU34	= sensor polarity (only potentiometer) -1 = negative; 1 = positive
	FU35	= valve polarity -1 = negative; 1 = positive

Read closed loop controller parameters

Input parameters	FU32	31
	FU33	Axis (1..12)
Output parameters	FU32	= status
	FU33	= error code
	FU34	= amplification factor of controller {1..100} corresponds internally to the range {0.1 .. 10.0}
	FU35	= cushioning factor of controller {1..100} corresponds internally to the range {0.1 .. 10.0}
	FU36	= filter factor of controller {1..100} corresponds internally to the range {0.1 .. 10.0}
	FU37	= positioning timeout {0..10000}; unit [0.1 sec].

Read application parameters

Input parameters	FU32	32
	FU33	Axis (1..12)
Output parameters	FU32	= status
	FU33	= error code
	FU34	= tool load {1..20000}; unit [0.1 kg]
	FU35	= work load {0.0..20000}; unit [0.1 kg]
	FU36	= supply pressure {30 ..100}; unit [0.1 bar]

Read fitting parameters

Input parameters	FU32	33
	FU33	Axis (1..12)
Output parameters	FU32	= status
	FU33	= error code
	FU34	= lower software end position {0..32000}; unit [0.1 mm]
	FU35	= upper software end position {0..32000}; unit [0.1 mm]
	FU36	= fitting position {-900..900}; unit [0.1°]
	FU37	= fitting offset {-16000..16000}; unit [0.1 mm]
	FU38	= project zero point {0..32000}; unit [0.1 mm]

Read calibration factor

Input parameters	FU32	34
	FU33	Axis (1..12)
Output parameters	FU32	= status
	FU33	= error code
	FU34	= Calibration factor {90 .. 110}; unit [0.01]

Read axis parameters 1/2

Input parameters	FU32	35
	FU33	Axis (1..12)
Output parameters	FU32	= status
	FU33	= error code
	FU34	= cylinder design: 0 = linear with piston rod 1 = linear without piston rod 2 = rotary
	FU35	= cylinder length {500..32000}; unit [0.1 mm]
	FU36	= cylinder diameter {120..3200}; unit [0.1 mm]
	FU37	= measuring system design: 1 = potentiometer 2 = Temposonic
	FU38	= measuring system length {500..32000}; unit [0.1 mm]

Read axis parameters 2/2

Input parameters	FU32	36
	FU33	Axis (1..12)
Output parameters	FU32	= status
	FU33	= error code
	FU34	= valve design:
		0 = VT-MPYE-5-M5
		1 = VT-MPYE-5-1/8LF
		2 = VT-MPYE-5-1/8HF
		3 = VT-MPYE-5-1/4
		4 = VT-MPYE-5-3/8HF
	FU35	= flow chain factor {1..15} corresponds internally to the range {0.1 .. 1.5}

Read out positioning parameters

Input parameters	FU32	37
	FU33	Axis (1..12)
Output parameters	FU32	= status
	FU33	= error code
	FU34	= speed {1..100}; unit [0.1 m/s]
	FU35	= acceleration {1..1000}; unit [0.1 m/s ²]
	FU36	= position window {1..50} / unit [0.1 mm]
	FU37	= override speed 0-100 %
	FU38	= position quality (class)
		1 = fast stop
		2 = fast stop with damping time
		3 = precision stop
		4 = precision stop with damping time
		5 = precision stop with speed control
		6 = precision stop with speed control and damping time

Read out ramp characteristics

Input parameters	FU32	38
	FU33	Axis (1..12)
Output parameters	FU32	= status
	FU33	= error code
	FU34	= with or without ramp
		0 = move with ramp
		1 = move without ramp (jump)
	FU35	= ramp type {0,1}
		0 = linear ramp
		1 = sin ² ramp

Delete hardware fault

Input parameters	FU32	80
	FU33	Axis (1..12)
Output parameters	FU32	= status
	FU33	= error code

Deletes the bit HWERR if possible (see axis flags)

Delete software fault

Input parameters	FU32	81
	FU33	Axis (1..12)
Output parameters	FU32	= status
	FU33	= error code

Deletes the bit SWERR if possible (see axis flags)

Delete timeout fault

Input parameters	FU32	82
	FU33	Axis (1..12)
Output parameters	FU32	= status
	FU33	= error code

Deletes the bit TOUT (see axis flags)

Delete EMERGENCY STOP

Input parameters	FU32	83
	FU33	Axis (1..12)
Output parameters	FU32	= status
	FU33	= error code

Deletes the bit EMERGENCY STOP (see axis flags) after "Reset module" if programmable reaction is set.

Reset/Init measuring system bus

Input parameters	FU32	84
	FU33	Axis (1..12) (one of the two axes!)
Output parameters	FU32	= status
	FU33	= error code

The Reinitialisation function of the measuring system bus triggers a process during which the AM30 module re-initialises all the slaves on the measuring system bus. When the command is executed, the axis parameters are retrieved from the Flash memory.

Reset axis

Input parameters	FU32	85
	FU33	Axis (1..12)
Output parameters	FU32	= status
	FU33	= error code

Axis is started again.

If the axis is moving, it will be stopped, the controller will be blocked and re-initialised. The axis loads the default values.

- The function must not be activated when the controller is enabled.
- The function must not be activated when a Flash function is active.
- The function must not be activated when the system is being re-initialised

Reset module

Input parameters	FU32	86
	FU33	Axis (1..12) (one of the two axes!)
Output parameters	FU32	= status
	FU33	= error code

Hard reaction (see jumper setting):

The DSP will be stopped and communication with the measuring system slaves will be interrupted. The result of this reaction is that the axes will be stopped immediately and with a jump.

Note! Uncontrolled axis movements may occur!!!

Programmable reaction:

This triggers an immediate interruption in the module. The module then stops the axis immediately and blocks all controller functions.

The axis can only be enabled again after command 83.

Static identification of axis

Input parameters	FU32	87
	FU33	Axis (1..12)
	FU34	1 = start 0 = stop
Output parameters	FU32	= status
	FU33	= error code

Dynamic identification of axis with minimum load

Input parameters	FU32	88
	FU33	Axis (1..12)
	FU34	1 = start 0 = stop
Output parameters	FU32	= status
	FU33	= error code

Dynamic identification of axis with maximum load

Input parameters	FU32	89
	FU33	Axis (1..12)
	FU34	1 = start 0 = stop
Output parameters	FU32	= status
	FU33	= error code

Do not save axis parameters in Flash

Input parameters	FU32	90
	FU33	Axis (1..12)
Output parameters	FU32	= status
	FU33	= error code

Delete axis parameters in Flash

Input parameters	FU32	91
	FU33	Axis (1..12)
Output parameters	FU32	= status
	FU33	= error code

Reset identification and adaptation parameters

Input parameters	FU32	92
	FU33	Axis (1..12)
Output parameters	FU32	= status
	FU33	= error code

Status interrogation 1/3

Input parameters	FU32	101
	FU33	Axis (1..12)
Output parameters	FU32	= status
	FU33	= error code
	FU34	= axis flags
	FU35	= command flags
	FU36	= software limit switch flags

FU37 = polarity flags
 FU34 = in position flags

Status interrogation 2/3

Input parameters	FU32	102
	FU33	Axis (1..12)
Output parameters	FU32	= status
	FU33	= error code
	FU34	= actual position
	FU35	= set position
	FU36	= not assigned
	FU37	= not assigned

Status interrogation 3/3

Input parameters	FU32	103
	FU33	Axis (1..12)
Output parameters	FU32	= status
	FU33	= error code
	FU34	= firmware version number
	FU35	= firmware date (day)
	FU36	= firmware date (month)
	FU37	= firmware date (year)

Meaning of the command flags

The individual bits have the following meaning in this parameter:

(7)	(6)	(5)	(4)	(3)	(2)	(1)	(0)
RUN	ABS	END	0	-	0	RDIS	DIR
RUN	:	0: axis stopped 1: axis moves					
ABS	:	(Absolute) 0: move relatively 1: move absolutely This determines whether the specified destination position is to be understood as absolute or relative					
END	:	(End position) 0: end position not reached 1: end position reached This status bit specifies whether one of the two software end positions has been reached or exceeded.					
RDIS	:	(Ramp disable) 0: ramp enables 1: ramp disabled This determines whether the start and stop procedures are to be carried out via the predefined ramp or "in the quickest way".					
DIR	:	(Direction) 0: positive 1: negative This determines the direction of travel with relative movements.					

Please note:

- The bit RUN is relevant for the FST programmer.

Meaning of the axis flags

This status register signals the status of the axis. A read access to this register informs the programmer whether an error has

occurred which can block the movement of the axis. Some of the information, which appears in this register, also appears in other (different) status registers for reasons of compatibility. In order to minimise the necessary bus accesses, this information is grouped together in the status register.

(7)	(6)	(5)	(4)	(3)	(2)	(1)	(0)
RUN	CAL	FLASH	END	STOP	TOUT	SWERR	HWERR
RUN	:	(Run) 0: axis stopped 1: axis moves This is the same status bit as in the axis control register and status register, but this bit cannot be modified by the status register.					
CAL	:	(Calibrated) 0: axis not calibrated 1 : axis calibrated After each "Power on" or "Module reset" the operating system ascertains the configuration of the measuring system bus. If it finds (per axis) the same slaves (AIF type / measuring system type), the parameters of which were saved in Flash, it will unload these parameters from Flash and set the CAL status bit (read only).					
FLASH	:	(Flash) 0: Flash function not active 1: Flash function active This status bit indicates whether a Flash function (programming or deleting) has been triggered. The bit can only be deleted if it is set at same time as the TOUT bit (Flash programmer/delete error).					
END	:	(End position) 0: End position not reached 1: End position reached This status bit (read only) indicates whether one of the two end positions has been reached or exceeded. It is the same status bit, as in the axis control register and status register.					
STOP	:	(Emergency Stop) 0: Emergency stop not active 1: Emergency stop active This status bit (readable and only deletable) indicates whether an Emergency stop has been triggered via the ISA bus (hardware reset) or via the PLC safety functions (software reset). (Of course only when no physical reset procedure is triggered when the configuration jumpers are set).					
TOUT	:	(Timeout) 0: No timeout 1: Timeout This status bit (readable and only deletable) is set when the controller cannot reach the last specified destination position within the calculated movement time + an additional timeout time or the Flash programming/delete procedure cannot be completed within the fixed set time.					
SWERR	:	(Software error) 0: No error 1: Controller error has occurred This status bit is set when the controller has ascertained a critical error in itself. This reaction of the controller has not yet been sufficiently documented in detail. Theoretically this error message should never occur.					
HWERR	:	(Hardware error) 0: No error 1: Error has occurred. This status bit is set under the following circumstances: - when this axis (1 or 2) has not been installed					

- on the measuring system bus;
- when a critical error has occurred on the measuring system bus.

The status bit is set for both axes.

- when a slave (AIF or CP) registers an error on the measuring system bus;
- when the system cannot program the Flash module correctly or delete it.

The status bit is set for both axes.

If one of the five lower-value status bits (END / STOP / TOUT / SWERR / HWERR) is set, no positioning task (for the relevant axis) will be accepted by the operating system. These status bits must first be deleted (acknowledged) by the user, before the axis can move again.

Positioning quality class

This parameter specifies how the controller is to react in the vicinity of the destination position. Possible specifications are as follows:

Positioning quality	Meaning
1	Fast stop without damping time. The axis registers the movement is finished as soon as it enters the tolerance area. Result: fast but not accurate positioning.
2	Fast stop with damping time. The axis registers the movement is finished as soon as it enters the tolerance area and the recovery time has expired.
3	Precision stop without damping time. The controller registers the movement is finished when the axis is in the tolerance window for the duration of the monitoring time.
4	Precision stop with damping time. The controller registers the movement is finished when the axis is in the tolerance window for the duration of the monitoring time and the recovery time.
5	Precision stop with speed control. The controller registers the movement as finished when the axis is in the tolerance window for the duration of the monitoring time and remains almost still.
6	Precision stop with speed control and damping time. The controller registers the movement as finished when the axis is in the tolerance window for the duration of the monitoring time and recovery time and remains almost still.

The length of the damping time and monitoring time is specified automatically by the controller.

Value range: 1 .. 6

Error messages of the module

Error code	Meaning
0	No error
1	1st parameter in parameter block outside the permitted range
2	2nd parameter in parameter block outside the permitted range
3	3rd parameter in parameter block outside the permitted range
4	4th parameter in parameter block outside the permitted range
5	5th parameter in parameter block outside the permitted range
6	6th parameter in parameter block outside the permitted range
7	7th parameter in parameter block outside the permitted range
8	8th parameter in parameter block outside the permitted range
9	9th parameter in parameter block outside the permitted range
10	10th parameter in parameter block outside the permitted range

11	11th parameter in parameter block outside the permitted range
12	12th parameter in parameter block outside the permitted range
13	Invalid parameter block number
14	Parameter block can only be read
15	Error in configuring the controller
20	Invalid register number
21	Parameter can only be read
22	Parameter outside permitted range
23	Controller not enabled
24	Controller is enabled
25	Controller enable active
26	No slave on this axis
27	Controller cannot be enabled
28	Jump without dynamic identification not possible
29	Identification active
30	Positioning task active
31	Positioning task not permitted (status error bit active)
32	End position reached
40	Flash function active
41	Reinitialisation of axis active
42	Reinitialisation of measuring system bus active
43	Invalid special function code
44	Indirect register can only be read
50	Hardware error cannot be deleted
51	CP error
60	Trace function error

The table contains all the error messages which are generated by the AM30.

As only a few of the functions are implemented in the FST module, not all the error messages will occur. The first time access is made to axis 2 (after Power on), error 21 may occur (functioning is however not impeded). Just ignore this error. At the next access, the error will be deleted automatically.

Commissioning a servopneumatic axis with the FST program AM30_DRV

Necessary hardware:

- CPU HC16F
- AM30

Proceed as follows:

1. Load the project.
2. Boot the controller.
3. Access program 62 in the STL status display mode.
4. In step 3 of the program, enter all the parameter values (of the application) in the registers.
5. Start program 62 (miniterminal / online).
6. If everything is OK, P62 will run through and delete itself. If everything is not OK, an error will be generated (diagnosis in step 2 of P62).
7. Start identification program 61 (miniterminal / online). The program will carry out all the identifications and saves the data in Flash. If the axis moves uncontrolled into an end position when the program is started:
 - the controller must be switched off and then on again;
 - you must modify the valve polarity in P62 after booting;
 - you must continue with point 5.
8. If P61 is completed successfully, the data are saved in Flash and after each "Power on" the axis will run with these parameters.
9. Now access program 1 in STL online display mode.
10. Enter the set positions (R2 and R102).
11. Program 1 can now be started in the miniterminal or online.

Note! Program 0 will be started automatically and only carries out the status interrogation of the system. All other programs must be started/stopped in online mode.

PID driver

This driver adds 16 PID control loops to your FST-IPC projects. In this document some terms related to control technology are used, for explanation of these no additional information is available at the moment.

Driver overview

The driver uses a table with the PID parameters, controller input and controller output. This table is located in FlagWords. The location where the table starts is set through a function module.

Configuring the driver and assigning parameters

If you want to use PID in an FST- IPC project, you must enter the PID driver in the driver configurer and assign the necessary parameters.

Destination drive:

Specify the drive on which the PID driver PIDDRV.EXE is located or onto which it is to be loaded.

Extended CI commands for PID

This driver extends the IPC command interpreter with the following commands:

!29 Display version number

Display driver info and version number. This information will also be displayed if an unknown command is entered (for example !29abcdef).

Module for PID

Overview of modules

PIDCFM Setup and start the driver

PIDCFM

Setup and start the driver.

Input parameters	FU32	Number of the 1st flagword used for the parameter table
Output parameters	FU32	0 if successful 256 if driver not installed

Note! Calling the PIDCFM function module will reset all flagwords in the complete parameter table to 0.

Parameter table

The following parameter table is the result if the PIDCFM function module is called with parameter 500:

	PID loop 0	PID loop 1	PID loop 15
Man/Auto	FW500	FW516	FW740
Setpoint	FW501	FW517	FW741
Process Value	FW502	FW518	FW742
Output Value	FW503	FW519	FW743
Manual Value	FW504	FW520	FW744
Kp	FW505	FW521	FW745
Ki	FW506	FW522	FW746
Kd	FW507	FW523	FW747
Time constant	FW508	FW524	FW748

Explanation of parameters

Man/Auto:

If this parameter is 0, the control loop is inactive and the manual value is copied to the output value.
If this parameter is 1, the control loop is active and the output value is calculated according to setpoint, process value and control parameters.

Setpoint:

The target for the controller in auto mode.

Process value:

The input value for the control loop.

Output value:

The output value from the control loop.

Manual value:

An operator override in case of manual mode.

Kp:

The proportional parameter for the control loop.

Ki:

The integral parameter for the control loop.

Kd:

The differential parameter for the control loop.

Time constant:

Indicator for the interval time with which the control loop calculates its output value.

If the time constant is 0 the control loop calculates approximately every 14 ms, if the time constant is 1 the control loop calculates approximately every 28 ms, etc.

The Setpoint, Process value, Output value and Manual value are all in the range from 0 to 4095.

The Kp, Ki and Kd values are divided by 100 inside the control loop. So if you set a Kp of 125 in the table, the control loop calculates with 1.25.

Serial Communication

Version 1.33

This driver allows characters to be sent and received via standard COM ports.

HC1X and HC2X

Up to 4 ports are supported. These can be either **COM1..COM4** or **COM2..COM5**.

Note! The serial port on the HC2X CPUs labelled "COM" is the COM1 port!

HC0X

- The interfaces **COM** and **EXT** can be used for serial communication.
- Additional COM ports on CP3x modules can be used:
 - Due to mutual exclusions of external interrupt sources (IRQs) you can either use **COM2** or **COM4**. If you do so, for HC01 the trimmer no longer can be used. For HC0X this also requires to change jumper J1 (see the hardware manual for details). On the HC02 there is no trimmer anyway, and no jumpers have to be changed.
 - Due to mutual exclusions of external interrupt sources (IRQs) you can either use **COM3** or **COM5**.
 - If you want to use **COM1** on HC02 you can do so, but this does not allow to use the ethernet port any longer and requires to change another jumper (see the hardware manual for details). On the HC01 you can use **COM1** without restrictions.

FECs

The interfaces **COM** and **EXT** can be used for serial communication.

Due to very limited memory in the FC20, only FEC Compacts of the FC3x series and FEC Standards should be used for serial communication.

Beginning with version 1.31 the driver supports FIFOs. Check the version number with the CI-Command "!8" which is available since version 1.31.

Selecting and parameterising the driver

If you wish to use the serial driver in an FST project, you must enter the correct driver(s) according to the table below in the driver configurator and parameterise it.

HC1X and HC2X

Please use the SERIALDR driver.

HC0X

If you are using only COM and/or EXT please use the driver COMEXT.

If you are using CP3x please use HC0XCOM instead (this driver includes the routines for COM and EXT as well).

For HC0X < S2.00 you also need to include the driver FOSEXT if you want to use the EXT port (HC0X < S1.11 requires the use of FOSSIL2 instead of FOSEXT).

FEC Compact

Please use the driver COMEXT.

For the EXT port you also need to include the driver FOSEXT (FC20 < V1.21 and FC30 < S1.10 require the use of FOSSIL2 instead of FOSEXT).

FEC Standard

Please use the driver COMEXT.

Note! Some of the modules provided also require the STRINGS driver.

Destination drive:

Enter the drive containing the serial driver or to which it is to be loaded.

Modules

The modules for serial communication require the serial interface that is used as parameter (FU32). Please note the following assignment of port numbers:

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

Note! You must not use the serial port that is used for the standard CI with these modules, except when it is referred to as port 255 (COM).

The standard COM port for the CI is set in the Controller Settings of the FST project.

The default setting for the standard COM port is

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

Note! It is not possible to send or receive #0 or Ctrl-T characters over the COM port (255).

Overview

OPENCOM	Open serial interface
CLOSECOM	Close serial interface
GETCOM	Send character
PUTCOM	Receive character
PRINTCOM	Send string
READCOM	Receive string (incl. delimiter)
READLCOM	Receive string (excl. delimiter)
F30	Set interface parameters
F31	Activate CI
F32	Clear buffers
F34	Interrogate status
F35	Change delimiter
BREAKCOM	Set/Reset hardware break (SERIALDR only)
SETRTS	Active handshake for RS485 (SERIALDR only)
IS485	Check if handshake is active (SERIALDR only)

OPENCOM

Open (initialise) a serial interface with 9600 baud, 8 data bits, no parity.

Input parameters	FU32	Serial interface
Output parameters	FU32	0 = DONE
		1 = ERROR

Use F30 instead of OPENCOM for other com port settings.

CLOSECOM

Close a previously opened serial interface.

Input parameters	FU32	Serial interface
Output parameters	FU32	0 = DONE
		1 = ERROR

GETCOM

Read character from a serial interface.

Input parameters	FU32	Serial interface
Output parameters	FU32	0 = DONE 1 = ERROR -1 = NOTHING RECEIVED
For FU32 = DONE	FU33	Received char. (0 to 255)

PUTCOM

Write character to a serial interface.

Input parameters	FU32	Serial interface
	FU33	Char. to be sent (0 to 255)
Output parameters	FU32	0 = DONE 1 = ERROR

PRINTCOM

Write an FST string to a serial interface.

Input parameters	FU32	Serial interface
	FU33	Number of FST string to be sent
Output parameters	FU32	0 = DONE 1 = ERROR

READCOM

Read characters from a serial string to a delimiter and save to an FST string.

Input parameters	FU32	Serial interface
	FU33	Number of FST string for the character string.
	FU34	Maximum length
	FU35	Delimiter (0 to 255)
Output parameters	FU32	0 = DONE 1 = ERROR -1 = NOTHING RECEIVED

The delimiter is not written to the string.

READLCOM

Read characters from serial interface to selected standard end-of-record character (default is CR) and save to FST string.

Input parameters	FU32	Serial interface
	FU33	Number of the FST string for the character string.
	FU34	Max. length including end-of-record character.
Output parameters	FU32	0 = DONE 1 = ERROR -1 = NOTHING RECEIVED

The delimiter is written with the string.

F30

Open port and set interface parameter.

Input parameters	FU32	Serial interface
	FU33	Interface parameter
Output parameters	None	
7 6 5 4 3 2 1 0		
Baud rate		Char. length Parity

Bit	7	6	5	4	Baud
	1	0	0	0	19200
	1	0	0	1	9600 (default)
	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

Bit	3	2	Bits/characters
	0	0	5
	0	1	6
	1	0	7
	1	1	8 (default)

Bit	1	0	Parity
	0	0	NONE (default)
	0	1	ODD
	1	0	NONE
	1	1	EVEN

Note! For the COM port (255) only the settings N,8,1 are accepted.

Note! For the COM port (255) of FC20 only the baudrates 9600 or less are useful due to the optocouplers.

F31

Activate CI.

Input parameters	FU32	Serial interface
	FU33	Driver mode:
		0=Disable CI
		1=Enable CI,
		disabled on receiving the CI command "X"
		2=Enable CI,
		ignore the CI command "X"
Output parameters	None	

Note! The standard COM port for the CI cannot be disabled.

The standard COM port for the CI is set in the Controller Settings of the FST project.

The default setting for the standard COM port is

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

F32

Erase interface buffer.

Input parameters	FU32	Serial interface
Output parameters	FU32	0 = DONE
		1 = ERROR

F34

Interrogate settings and status of a serial interface.

Input parameters	FU32	Serial interface
Output parameters	FU32	Number of records in receive buffer
		(separated by standard end-of-record characters).

F35

Change standard end-of-record character. Default is CR (13).

Input parameters	FU32	Serial interface
	FU33	End-of-record characters (0 to 255)
Output parameters	None	

BREAKCOM

Send hardware BREAK.

Note! This function is only implemented for the SERIALDR (HC1X and HC2x).

Input parameters	FU32	Serial interface
	FU33	1 = BREAK on 0 = BREAK off
Output parameters	None	

IS495

Returns current port setting regarding RS485 support.

Input parameters	FU32	Serial interface
Output parameters	FU32	0 = DONE 1 = ERROR
	FU33	1=RTS control for RS485 active 0=normal RS232 operation

Note! FST driver SERIALDR Version 1.10 or greater is required.

Note! This function is only implemented for the SERIALDR (HC1X and HC2x).

SETRTS

Enables/disables RTS control for RS485 serial communication via SM30 IPC module.

Input parameters	FU32	Serial interface
	FU33	1 for RS485 operation, 0 for normal RS232 operation
Output parameters	FU32	0 = DONE 1 = ERROR

Note! Has to be called before opening the port.

Note! FST driver SERIALDR Version 1.10 or greater is required.

Note! This function is only implemented for the SERIALDR (HC1X and HC2x).

TCP/IP driver

Version 1.12

This driver allows the IPC to communicate with other PCs and Controllers using the UDP or TCP protocol from the TCP/IP protocol suite.

Introduction

In this document terms related to TCP/IP are used, for explanation please refer to one of the many books and internet sites. A good starting point is <http://www.whatis.com> where all kinds of technical terms are explained.

Driver overview

To function properly every TCP/IP participant (host) needs an IP address and netmask. The FST TCP/IP driver allows 3 ways for configuring these: configuration in the Driver configuration screen, using function blocks or dynamically using the BOOTP or DHCP protocol.

Note! If the IP address is 0 (0.0.0.0) either through configuration or change through a function block the BOOTP/DHCP protocol will be automatically activated.

If the IP address is set the BOOTP/DHCP protocol will be deactivated.

The FST TCP/IP driver implements the following protocols / services for communication and testing:

Protocol	Port	Service	Description
ICMP	n.a.	n.a.	Used for PING
UDP	7	echo	Returns all data received
UDP	9	discard	Discards all data received
UDP	13	daytime	Returns string with date and time
UDP	19	chargen	Returns string with characters
UDP	37	time	Returns time as 32bit number
UDP	991	CI	Command interpreter
UDP	992	CI_EXT	Extended command interpreter
UDP	993	CI_BIN	Binary data exchange
UDP	995	EasyIP	Data exchange protocol
TCP	7	echo	Returns all data received
TCP	9	discard	Discards all data received
TCP	13	daytime	Returns string with date and time
TCP	19	chargen	Sends continual stream of characters
TCP	991	CI	Command interpreter (via telnet)

The TCP command interpreter at port 991 can be accessed with a program like TELNET. Start TELNET with parameters IP number and 991 (i.e. TELNET 10.10.10.1 991), now it is possible to send and receive just as via the normal RS-232 command interpreter. Please note that only 1 connection is allowed at the same time and the connection will automatically be closed after 60 seconds of no interaction.

The command interpreter can also be accessed as follows: send an UDP datagram with the CI command (without CR / LF) to the FST controller at port number 991 and the result will be returned. For example DR0 will result in something like =1099.

Using the extended command interpreter on port 992 is similar with a small extension. The response will be the original question and the CI reply separated by a NULL character. So when sending DR0 the result will be DR0 NULL character =1099.

Using the TCP (telnet) way is more convenient for users, using UDP is better suited for programs that need to access FST operands.

Note! Do not use the Y! command. This command will stop all programs and drivers, since TCP/IP is implemented as a driver it will be stopped as well !

Note! The LE commands are not available via the telnet interface.

Configuring the driver and assigning parameters

If you want to use TCP/IP in a FST IPC project, you must enter the TCP/IP driver in the driver configuration and assign the necessary parameters.

Different driver names exist for different target systems, the following table lists the supported target systems and their driver:

TCPIPDRV	PS1-HC1x with PS1-CP10/CP11/CP14 PS1-HC1x with PS1-CP12 (8 bit modus) PS1-HC2x with PS1-CP10/CP11/CP14 PS1-HC2x with PS1-CP12 (8 bit modus)
TCPIPFEC	FEC Compact (FC34,FC44)
TCPIPFEC2	FEC Standard (FC440,FC560,FC640,FC660)
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

Configuration of TCP/IP driver for PS1-CP10, CP11, CP12 and CP14:

Destination drive:

Specify the drive on which the TCP/IP driver TCPIPDRV.EXE is located or onto which it is to be loaded.

Interrupt of CP10/CP11/CP12/CP14 module:

The interrupt as configured on the module. Allowed values are 2, 3, 5, 6 and 7. Make sure that the jumper on the network module is set accordingly and that there are no conflicts with other IPC modules.

IO port configuration:

The IO port (in hexadecimal !) as configured on the module. Allowed values are 300, 320, 340 and 360. Make sure that the switches on the network module are set accordingly and that there are no conflicts with other IPC modules.

IP address:

The IP address. Leave the address to 0.0.0.0 if the address is configured through the IP_IP function block or dynamically through the BOOTP protocol.

IP netmask:

The IP netmask. Leave the mask to 255.255.255.0 if the netmask is configured through the IP_MASK function block or dynamically through the BOOTP protocol.

IP address gateway:

The IP address for the standard gateway. Leave the address to 0.0.0.0 if there is no gateway or the gateway address is configured through the IP_GATE function block or dynamically through the BOOTP protocol.

Configuration of TCP/IP driver for PS1-CP15:

Only destination drive and IP configuration are needed. Additionally there is a configuration file PKT.INI (in the FST runtime directory) which must be edited to reflect the settings to be used. Most important parameters are BitRate, Channel, RadioType and SystemId.

The PKT.INI will be downloaded into the IPC when downloading the project. Please make sure that after modification the

file is downloaded since the project download will only freshen files with different sizes.

Configuration of TCP/IP driver for FEC Compact, FEC Standard, PS1-HC02:

Only destination drive and IP configuration are needed.

Configuration of TCP/IP driver for 2 network cards:

Lots of parameters are needed. Destination drive, port and interrupt number for both cards, IP configuration for both cards.

Please be careful when using the TCPIPXXX driver:

- On the second card BOOTP/DHCP is not possible. The second card only allows a local network (no gateway option for routing).
- Functions like EASY_S, EASY_R, UDP_SEND etc. will automatically select the first or second network card based upon IP configuration and destination IP.
- For performance reasons it is advised to use HC20 or better CPUs.

Extended CI commands for TCP/IP

This driver extends the IPC command interpreter with the following commands:

```
!26      Display version number
```

Display driver info and version number. This information will also be displayed if an unknown command is entered (for example !26?).

```
!26Axx   Display ARP resolution  
         table information
```

Displays information from the Address Resolution Protocol (ARP). The TCP/IP driver holds a table of 32 IP addresses and Ethernet addresses. If the position (xx) is omitted the displayed information will automatically cycle through the used positions.

```
!26B      Display BOOTP/DHCP status
```

Displays status for the BOOTP / DHCP process and the time before the DHCP lease expires if the IP address was 'leased' from a DHCP server.

```
!26C      Display date / time
```

Displays the actual date, time and timezone offset in the format: YYYY.MM.DD HH:MM:SS.tt followed by the UTC offset in hours.

```
!26Da.b.c Resolve hostname to IP address
```

Resolves the hostname a.b.c to an IP address. For example !26Dwww.festo.com would search the IP address for www.festo.com

```
!26D      Display name resolver status
```

Displays the status of the last hostname search. Possible return texts: resolving, resolved followed by IP address, resolver timed out and resolver error followed by an error code.

When searching for a hostname first enter !26D followed by the hostname, then enter !26D until the resolver is no longer 'resolving'.

```
!26Hxx   Display handler table information
```

Displays information about currently defined TCP and UDP handlers and their state. If the position (xx) is omitted the displayed information will automatically cycle through the defined handlers.

```
!26I      Display IP configuration
```

Displays IP address, IP netmask and IP gateway.

```
!26M      Display Ethernet MAC  
         address
```

Displays the MAC address of the network card.

```
!26N      Display hostname and domainname
```

Displays the hostname and domainname.

```
!26Pa.b.c.d Ping IP address
```

Sends a ping to the IP address a.b.c.d For example !26P10.10.10.10 would ping 10.10.10.10

```
!26P      Display ping status
```

Displays the status of the last ping. Possible return texts: not pinging, pinging, host is alive and timeout.

When pinging a host first enter !26P followed by the IP address, then enter !26P until the return message is either host is alive or timeout.

```
!26R      Display runtime
```

Displays the time since the controller restart. Display format: days:hours:minutes:seconds.

```
!26SH     Display interrupt count
```

Displays the number of interrupts generated by the network module. This information is only available on HC1X, HC2X controllers with CP10, CP11, CP12 or CP14 network module.

```
!26SI     Display IP statistics
```

Displays IP packet statistics: total send, total received, checksum errors, wrong destination address.

```
!26SP     Display Ethernet statistics
```

Displays Ethernet packet statistics: total send, total received, overflow, packet too large, cannot send.

```
!26ST     Display TCP statistics
```

Displays IP packet statistics: total send, total received, checksum errors, wrong destination address, no handler for TCP port.

```
!26SU     Display UDP statistics
```

Displays IP packet statistics: total send, total received, checksum errors, wrong destination address, no handler for UDP port, unhandled broadcast.

```
!26Txx    Display IP table contents
```

Displays the IP addresses currently defined in the IP table. The TCP/IP driver can hold a total of 32 IP addresses for easy reference.

Note! The CI commands are mainly intended for diagnostic purposes. In case of problems start with !26 first. If the IPC returns an ACCESS ERROR this means that the TCP/IP driver is not included in the project or the hardware configuration is wrong. Then check for packets send and received with !26SP. If no packets are received this could indicate that the CP10/11/12 interrupt is wrongly configured.

Modules for TCP/IP

Overview of generally used modules

EASY_R	Request a block of operands from another IPC
EASY_S	Send a block of operands to another IPC
IP_ALIVE	Check whether IP address is known
IP_IP	Get/set our IP address
IP_MASK	Get/set our IP netmask
IP_TABLE	Get/set IP address from/to table

All modules return an error code in FU32, see the table with all errorcodes for an explanation of the individual codes.

EASY_R

Request a block of operands from another controller.

Input parameters

FU32 Index number in IP table
FU33 Operand type,
1=flags,
2=inputs,
3=outputs,
4=registers,
11=strings
FU34 Number of operands wanted (maximum 256)
FU35 Number of first local operand to store response
FU36 Number of first operand in remote controller
FU37 Number of flagword for status

Output parameters

FU32 0 if request send, otherwise error

This function module requests a block of operands.

Note! The operand TYPE specified in FU33 (source of remote data requested) also defines the Local Operand TYPE where the operand will be stored in the local (requesting) controller. For example:

```
IF...  
THEN  CMP 12      'EASY_R  
      WITH  V3     " Table Index 3  
      WITH  V4     " get Registers  
      WITH  V10    " get 10 Registers  
      WITH  V150   " 1st Remote Reg -> Local R150  
      WITH  V34    " Remote Register data block  
      " is R34 to R43  
      WITH  V99    " Status in FlagWord 99
```

If no response is received after approximately 50 milliseconds the status flagword will indicate a timeout. Only 1 request can be active for each index number.

Status values are:

-1 Packet send, no response or timeout yet
0 OK response received, data accepted by partner or data received from partner
1 Operand type error, partner indicates that the specified type is not supported
2 Offset error, partner indicates that the specified offset is not allowed. For example requesting FW20000
4 Size error, partner indicates that the number of operands send or requested is too large
128 Timeout, no response received from partner

EASY_S

Send a block of operands to another controller.

Input parameters

FU32 Index number in IP table
FU33 Operand type,
1=flags,
2=inputs,
3=outputs,
4=registers,
11=strings
FU34 Number of operands to send (maximum 256)
FU35 Number of first operand to send
FU36 Number of first operand in remote controller
FU37 Number of flagword for status (-1 if no acknowledge wanted)

Output parameters

FU32 0 if data send, otherwise error

This function module sends a block of operands.

Note! The operand TYPE specified in FU33 (source of data to be sent) also defines the Remote Operand Type where the data will be stored in the REMOTE (destination) controller. For example:

```
IF...
THEN  CMP 22      'EASY_S
      WITH  V3     " Table Index 3
      WITH  V4     " send Registers
      WITH  V10    " send 10 Registers
      WITH  V23    " send local Registers
      WITH  V234   " R23-R32 to remote controller
      WITH  V234   " Remote controller will store
      WITH  V98    " Data in R234-R43.
      WITH  V98    " Use FW98 for status
```

If no acknowledgment is received after approximately 50 milliseconds the status flagword will indicate a timeout. Only 1 unacknowledged send can be active for each index number.

Status values are:

-1	Packet send, no response or timeout yet
0	OK response received, data accepted by partner or data received from partner
1	Operand type error, partner indicates that the specified type is not supported
2	Offset error, partner indicates that the specified offset is not allowed. For example requesting FW20000
4	Size error, partner indicates that the number of operands send or requested is too large
128	Timeout, no response received from partner

IP_ALIVE

Checks the ARP table, to see if an IP address is known.

```
Input parameters
FU32  Index number in IP table
Output parameters
FU32  0 if successful, otherwise error
FU33  0 if IP address unknown
      1 if IP address known
      2 if IP address must be reached through a gateway
```

Be aware that if a controller is stopped, disconnected or otherwise no longer available it can take up to 10 minutes before IP_ALIVE indicates that the IP address is unknown.

Other alternatives are available, for example testing the status value from EasyIP packets or using the function module PING if no EasyIP communication is used.

IP_IP

Set or get the IP address.

```
Input parameters
FU32  1 to set the IP address
      2 to get the IP address
FU33  IP address
FU34  IP address
FU35  IP address
FU36  IP address
Output parameters
FU32  0 if successful, otherwise error
FU33  IP address
FU34  IP address
FU35  IP address
FU36  IP address
```

This module is normally only used in projects with identical controllers (with the same programs). By using this function

module and retentative variables it is then possible to use identical FST projects on multiple controllers. Other use is when using dynamic addresses so the controller knows when the IP address is received.

IP_MASK

Set or get the IP netmask.

```
Input parameters
FU32      1 to set the IP mask
           2 to get the IP mask
FU33      IP mask
FU34      IP mask
FU35      IP mask
FU36      IP mask
Output parameters
FU32      0 if successful, otherwise error
FU33      IP mask
FU34      IP mask
FU35      IP mask
FU36      IP mask
```

IP_TABLE

Set or get an IP address into or from table.

```
Input parameters
FU32      1 to set the IP address
           2 to get the IP address
FU33      index number in IP table
FU34      IP address
FU35      IP address
FU36      IP address
FU37      IP address
Output parameters
FU32      0 if successful, otherwise error
FU33      index number in IP table
FU34      IP address
FU35      IP address
FU36      IP address
FU37      IP address
```

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

More modules for TCP/IP

Overview of special modules

These modules are listed separately because their use requires special expertise, they are rarely used etc.

DNS_NAME	Get/set hostname and domainname
DNSRESOL	Resolve hostname to IP address
IP_DNS	Get/set IP address for DNS server
IP_GATE	Get/set IP address for gateway
IP_MAC	Get Ethernet MAC address from network module
PING	Ping
SNTPTIME	Start time synchronisation
TCP_CLOS	Close TCP connection
TCP_HAND	Activate TCP handler
TCP_OPEN	Open TCP connection
TCP_RES	Reset TCP handler
TCP_SEND	Send a TCP datapacket
TCP_STAT	Status of TCP connection
TCP_STR	Send a string through TCP
TFTPFILE	Send / request a file
UDP_FW	Send a FW block to another IPC
UDP_HAND	Activate UDP handler
UDP_SEND	Send an UDP datapacket
UDP_STR	Send a string through UDP

All modules return an error code in FU32, see the table with all errorcodes for an explanation of the individual codes.

DNS_NAME

Get or set the hostname and domainname.

```
Input parameters
FU32      1 to set hostname
          2 to get hostname
          3 to set domainname
          4 to get domainname
FU33      Number of the source string (set),
          or destination string (get).
Output parameters
FU32      0 if successful, otherwise error
```

TCP/IP hosts normally have names in the format name.some.domain.com. In this case the hostname is 'name' and the domainname is 'some.domain.com'. These names can be set through BOOTP/DHCP or with this module. This function requires that the FST string driver is installed in the project.

DNSRESOL

Resolve a hostname to IP address

```
Input parameters
FU32      1 to start hostname resolution
          2 to get status
FU33      Number of the string with host-name to resolve
FU34      Index number in IP table
Output parameters
FU32      0 if successful, otherwise error
FU33      Resolver status
          -3 if resolver not yet started
          -2 if resolver already busy, wait and try again
          -1 if resolver busy
          0 if resolver finished
          1 if resolver timed out
```

Tries to find the IP address for a hostname. The IP address for DNS server(s) must be known. This function requires that the FST string driver is installed in the project.

IP_DNS

Set or get an IP address for a DNS server

```
Input parameters
FU32      1 to set the IP address
          2 to get the IP address
FU33      DNS server number (0, 1 or 2)
FU34      IP address
FU35      IP address
FU36      IP address
FU37      IP address
Output parameters
FU32      0 if successful, otherwise error
FU33      DNS server number
FU34      IP address
FU35      IP address
FU36      IP address
FU37      IP address
```

Up to 3 DNS servers are used when resolving a hostname into an IP address. The servers are used consecutive (0, 1 then 2).

IP_GATE

Set or get the IP address for gateway.

```
Input parameters
FU32      1 to set the IP address
          2 to get the IP address
FU33      IP address
FU34      IP address
```

```

FU35    IP address
FU36    IP address
Output parameters
FU32    0 if successful, otherwise error
FU33    IP address
FU34    IP address
FU35    IP address
FU36    IP address

```

IP_MAC

Get the Ethernet MAC address.

```

Input parameters
FU32    Number of string to store the address
Output parameters
FU32    0 if successful, otherwise error

```

The FST STRING driver must be installed in the project.

PING

Send ping or get ping status

```

Input parameters
FU32    1 to send ping
        2 to get status
FU33    IP address
FU34    IP address
FU35    IP address
FU36    IP address
Output parameters
FU32    0 if successful, otherwise error
FU33    Ping status
        -1 = pinging
        0 = host is alive
        1 = timed out (no response after 5 seconds)
        2 = not yet pinging

```

Sends a ping and wait for response. Use this to actively test if a host is alive.

SNTPTIME

Starts time synchronisation and get status

```

Input parameters
FU32    0 to get status
        1 to start synchronisation
        2 to start listen for broadcasts
FU33    Time offset to Greenwich Mean Time in hours
FU34    IP address
FU35    IP address
FU36    IP address
FU37    IP address
Output parameters
FU32    0 if successful, otherwise error
FU33    status
FU34    number of seconds since last synchronisation
        -1 if not synchronised

```

The time synchronisation uses standard time codes (to allow worldwide synchronisation) therefore the time difference with Greenwich Mean Time must be known. For some sample values see the table at the end. The status codes are: -1 busy, 0 time synchronised, 1 timedout (no response from NTP server), 2 if listening active.

Timeserver programs are available from freeware to commercial packages and can be run on standard windows PCs.

TCP_CLOS

Closes a TCP connection

```

Input parameters

```

```
FU32    Index number for handler
Output parameters
FU32    0 if successful, otherwise error
```

TCP_HAND

Installs a handler to receive TCP datapackets

```
Input parameters
FU32    local port number to use
FU33    Number of first flagword to receive data
Output parameters
FU32    0 if successful, otherwise error
FU33    Index number for handler
```

Some additional data is written if a datapacket is received. See table below for layout. The handler number returned in FU33 must be stored and used to send TCP datapackets.

TCP_OPEN

Actively opens a TCP connection

```
Input parameters
FU32    Index number for handler
FU33    Index number in IP table
FU34    Destination port number
Output parameters
FU32    0 if successful, otherwise error
```

TCP_RES

Resets closed TCP handler to listen state

```
Input parameters
FU32    Index number for handler
Output parameters
FU32    0 if successful, otherwise error
```

TCP_SEND

Send a TCP datapacket

```
Input parameters
FU32    Index number for handler
FU33    number of bytes to send
FU34    number of first flagword to send
Output parameters
FU32    0 if successful, otherwise error
```

TCP_STAT

Returns status of TCP connection

```
Input parameters
FU32    Index number for handler
Output parameters
FU32    0 if successful, otherwise error
FU33    1 if connected (send possible)
FU34    Extended state
FU35    Number of unacknowledged bytes in send buffer
```

Extended state values are:

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 through TCP

```

Input parameters
FU32    Index number for handler
FU33    Number of string to send
Output parameters
FU32    0 if successful, otherwise error

```

The FST STRING driver must be installed in the project.

TFTPFILE

Send or requests a file.

```

Input parameters
FU32    1 to send a file
        2 to request a file
FU33    index number in IP table
FU34    number of the string with our file-name
FU35    number of the string with the remote filename
FU36    number of flagword for status
Output parameters
FU32    0 if successful, otherwise error

```

Uses the string driver for filenames. A waiting time of around 1 second must be inserted between 2 transmissions with TFTPFILE.

Status values:

-1	Busy with file transfer.
0	File transfer successfully finished
1	Timeout error
2	Local file not found
3	Error reading from local file
4	Local file already exists (overwrite is not allowed, use the FDELETE function module first)
5	Error writing to local file
127	Unexpected message received during filetransfer
128	Received unknown error message
129	Received NOFILE error message
130	Received access error message
131	Received disk full error message
132	Received illegal operation error message
133	Received TID error message
134	Received file exists error message
135	Received NOUSER error message
136	Received option error message

UDP_FW

Send a block of flagwords to another IPC.

```

Input parameters
FU32    index number in IP table
FU33    number of flagwords to send (maximum 256)
FU34    number of first flagword to send
FU35    number of first flagword in target IPC
Output parameters
FU32    0 if successful, otherwise error

```

This function module just sends a block of flagwords, no retry is done if the target is unavailable and no indication is given if

the target did not receive the flagwords. This module has lost its use, use the module EASY_S instead.

UDP_HAND

Installs a handler to receive UDP datapackets

```
Input parameters
FU32    local port number to use
FU33    Number of first flagword to receive data
Output parameters
FU32    0 if successful, otherwise error
```

Some additional data is written if a datapacket is received. See table below for layout.

UDP_SEND

Send an UDP datapacket

```
Input parameters
FU32    local port number to use
FU33    index number in IP table
FU34    Destination port number
FU35    number of bytes to send
FU36    number of first flagword to send
Output parameters
FU32    0 if successful, otherwise error
```

UDP_STR

Send a string through UDP

```
Input parameters
FU32    Local port number to use
FU33    Index number in IP table
FU34    Destination port number
FU35    Number of string to send
Output parameters
FU32    0 if successful, otherwise error
```

The FST STRING driver must be installed in the project.

Modules for second network card

```
IP_IP2          Get/set IP address for 2 nd network card
IP_MASK2        Get/set IP netmask for 2 nd network card
```

IP_IP2

Set or get the IP address.

```
Input parameters
FU32    1 to set the IP address
         2 to get the IP address
FU33    IP address
FU34    IP address
FU35    IP address
FU36    IP address

Output parameters
FU32    0 if successful, otherwise error
FU33    IP address
FU34    IP address
FU35    IP address
FU36    IP address
```

IP_MASK2

Set or get the IP netmask.

Input parameters

FU32	1 to set the IP mask
	2 to get the IP mask
FU33	IP mask
FU34	IP mask
FU35	IP mask
FU36	IP mask

Output parameters

FU32	0 if successful, otherwise error
FU33	IP mask
FU34	IP mask
FU35	IP mask
FU36	IP mask

EasyIP status values

Status values are:

-1	Packet send, no response or timeout yet
0	OK response received, data accepted by partner or data received from partner
1	Operand type error, partner indicates that the specified type is not supported
2	Offset error, partner indicates that the specified offset is not allowed. For example requesting FW20000
4	Size error, partner indicates that the number of operands send or requested is too large
128	Timeout, no response received from partner

Data received by handlers

Handlers write data to flagwords if a packet is received. Also written is the IP address, port number of the datapackets sender. The following table lists all data stored (FWx is the installed flagword):

FWx	Number of packets received by handler
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 databytes
FWx + 10	Start of actual datapacket

Note that the received size is the number of bytes, not the number of words.

Time offsets

Some values for the time offset:

Anchorage	-9	
Buenos Aires		-3
London	0	
Wetzlar	+1	
Cairo	+2	
Moscow	+3	
Jakarta	+6	

Error codes

If FU32 is ≤ 0 then the function module returned an error, following table lists the error codes:

99	Invalid parameter
100	TCP/IP driver not loaded
101	Illegal IP address
102	Illegal table index (> 15)
103	Table position empty
104	Invalid port number
105	Invalid handler index (>15)
106	TCP send not possible, not connected
107	String driver not loaded
108	Illegal string number (too high)
109	Error in host- or domainname
110	Error modifying string
111	Invalid TCP handler
112	Unknown operand type
113	Datablock too large
114	Invalid value for status flagword
115	Table position already waiting for response
116	Invalid operand number to store response
117	Cannot send, cable disconnected, collision or other error
118	TFTP already sending or receiving