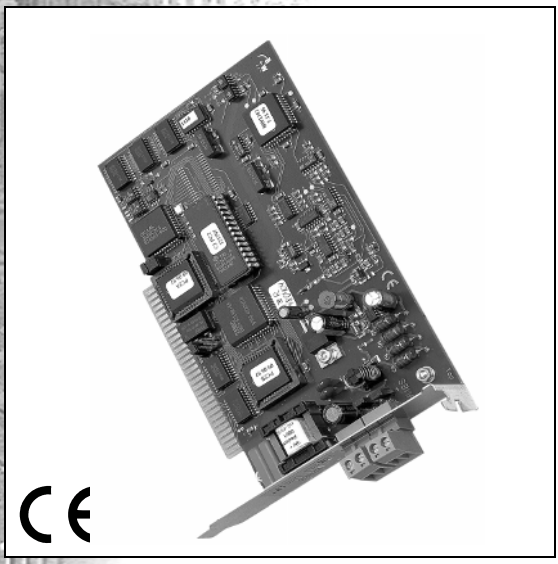
FACTORY AUTOMATION

MANUAL

VAM-CTR-PC2

AS-INTERFACE PC2-BOARD



PEPPERL+FUCHS

Table of Contents

Issue date 27.07.98

Issue date 27.07.98

1 Declaration of Conformity

The AS-i PC2 Board have been developed and produced in accordance with the applicable European standards and directives.

Note

Notice

The corresponding of conformity can be requested from the manufacturer.

The manufacturer of the product, Pepperl & Fuchs Group in D- 68307 Mannheim, possesses a certified quality assurance system in accordance with ISO 9001.

CE

ISO**9001**

Issue date 27.07.98

2 The Used Symbols

Warning

This symbol warns the user of possible danger. Failure to heed this warning can lead to personal injury or death and/or damage to equipment.

Attention

This symbol warns the user of a possible failure. Failure to heed this warning can lead to total failure of the equipment or any other connected equipment.

Note

This symbol gives the user important hints.

Issue date 27.07.98

3 Safety

3.1 Intended Use

Warning

The protection of operating personnel and the system against possible danger is not guaranteed if the control interface unit is not operated in accordance with its intended use.
The device may only be operated by appropriately qualified personnel in accordance with this operating manual.

3.2 General Safety Information

Warning

Safety and correct functioning of the device cannot be guaranteed if any operation other than that described in this operation manual is performed.
The connecting of the equipment and any maintenance work to be carried out with voltage applied to the equipment must only be performed by appropriately qualified electrotechnical personnel.
In the case that a failure cannot be repaired, the device must be taken out of operation and kept from inadvertently put back into operation.
Repair work is to be carried out by the manufacturer only. Additions or modifications to the equipment are not allowed and void the warranty.

Note

The operator is responsible for the observance of local safety standards.

Issue date 27.07.98

4 Features of the AS-i PC2 Board

The Pepperl+Fuchs AS-i PC2 board ...

- is a single AS-i Master with AS-i Control option (PLC), build as a board for the PC ISA bus.
- is able to execute a control program. It operates without taking an affect to the performance of your PC. The other way round the control program runs with a constant cycle time independent from the PC capacity required by other applications.
- exchanges data with the PC via Dual Port RAM (DPRAM).
- uses only 3 I/O addresses of the PC ISA bus for data exchange.
- ensures an easy way to program drivers in any programming language and for any operating system. Adapting existing PC software is also possible.
- does not need interrupts for standard operation.
- allows simultaneous operation of up to 8 AS-i PC2 boards in one PC system.
- is capable to generate an interrupt on the ISA bus, for example if a change in the AS-i data occurs.
- can share an interrupt with other AS-i PC2 boards
- is able to detect a breakdown of the PC if the build-in watchdog is activated. (The AS-i master will change to the Off-line phase if the watchdog is not triggered by a PC program.)
- includes the ‘Advanced Diagnostics’

Issue date 27.07.98

5 Installation of the AS-i PC2 Board

At the backplane of the board there are 4 terminals for the AS-i power supply. You can select with jumpers JP4 and JP5 whether the AS-i should be powered by its own power supply or by an external AS-i power supply.

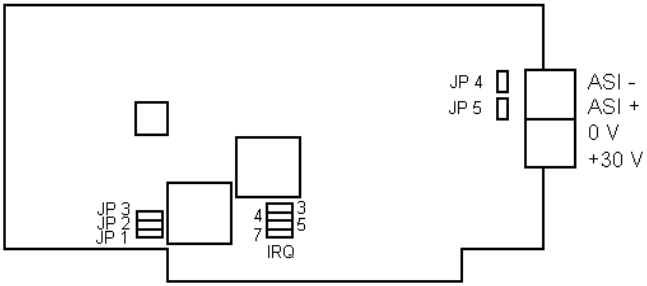


Figure 5.1: Backplane of the board

The individual terminals and jumpers have the following functions:

AS-i+	"AS-i +", Actuator Sensor Interface, positive terminal
AS-i–	"AS-i –", Actuator Sensor Interface, negative terminal
30V	standard power supply, positive terminal (24V-32V)
0V	standard power supply, negative terminal
J1 - J3	Jumpers for selecting the I/O address:

I/O-Basisadresse	J1	J2	J3
300_h	closed	closed	closed
304_h	open	closed	closed
308_h	closed	open	closed
$30C_h$	open	open	closed
320_h	closed	closed	open
324_h	open	closed	open
328_h	closed	open	open
$32C_h$	open	open	open

Issue date 27.07.98

J4, J5 Jumpers for selecting the power supply of AS-i:

both closed: The AS-i is powered by a standard power supply. This should be connected to the terminals '30V' and '0V'. The external power supply and the AS-i circuit are then decoupled with coils. This possibility is not AS-i conform and should be used for tests only.

both open: With the jumpers open the AS-i must be powered by a AS-i power supply. The AS-i power supply should be connected to the AS-i cable in the field.

IRQ With this jumper the ISA bus interrupt number (3, 4, 5 or 7) can be selected. It is not necessary for the correct operation of the AS-i PC2 to have an interrupt selected.

In opposition to standard ISA bus cards, the AS-i PC2 board is able to share its interrupt with other AS-i PC2 boards. It is not possible for a AS-i PC2 to share its interrupt with standard ISA bus cards.

In small applications, power of the AS-i Network can be supplied through a simple 30 V DC power supply. For use in a fully developed System, the AS-i network must be connected to an AS-i power supply. The following figures illustrate the possibilities for the power supply of the AS-i circuit.

Variant 1:
AS-i with standard power supply (J4 and J5 closed):

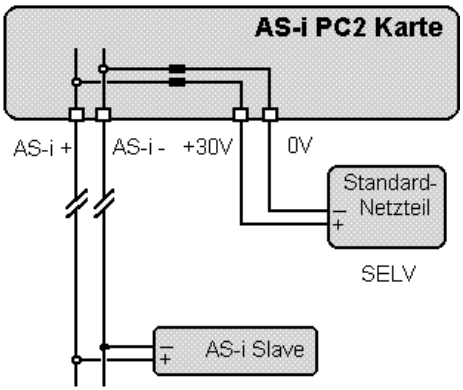


Figure 5.2: AS-i with standard power supply

The maximum current through the AS-i master is 2A.

Issue date 27.07.98

Variant 2:
AS-i with AS-i power supply directly connected to the network (J4 and J5 open):

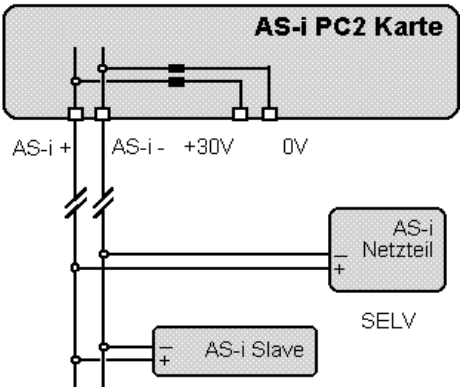


Figure 5.3: AS-i with AS-i power supply

The maximum current through the AS-i master is 2A.

Issue date 27.07.98

6 Advanced Diagnostics for AS-i Masters

To give the user the possibility to solve the problem to locate occasional occuring errors without additional diagnostics tools, Pepperl+Fuchs has implemented advanced diagnostics functionality inside the AS-i masters.

The "AS-i Control Tools" Pepperl+Fuchs's software for the comfortable commissioning of the AS-Interface and the programming of AS-i Control will include the "Advanced Diagnostics" with the *LCS* from version 3.0 on.

6.1 List of corrupted AS-i Slaves *(LCS)*

To locate occasionally occuring short-time configuration errors the AS-i masters with advanced diagnostics manage beside the list of projected slaves *(LPS)*, the list of detected slaves (LDS) and the list of activated slaves (LAS) a forth list, the List of Corrupted Slaves *(LCS)*. This *LCS* contains entries of all AS-i slaves which were responsible for at least one configuration error since powering up the AS-i master or reading the list. AS-i power fail is displayed in the *LCS* at the position of AS-i slave with address 0.

With every read access the *LCS* will be deleted.

6.2 Error Counter: Counter of corrupted data telegrams

The AS-i master with advanced diagnostics has an Error Counter for each AS-i slave, which is increased everytime there is a corrupted data telegram. This makes possible to judge the quality of the AS-i network, even if only a few corrupted telegrams occured and the AS-i slave did not cause any configuration errors.

The counter values can be read via the host interface and will be deleted with every read access. The counter value is limited to 254. 255 means counter overflow.

The "Error Counter" is included in the command "Advanced Diagnostics" of "AS-i Control Tools" version 3.0.

6.3 Offline Phase when Configuration Error

A new feature of AS-i masters with advanced diagnostics is the posibility to put themselves into the Offline Phase, when a configuration error on the AS-Interface occurs. In this way the security of the application can be ensured. The reaction to a configuration error is very fast and the host can be relieved. If there are any problems on the AS-i networks, the AS-i masters can ensure the security of the application.

Issue date 27.07.98

There are two different ways to use this feature:

- Every configuration error during normal operation in protected mode releases the Offline Phase.
- For each slave address can be chosen, if a configuration error on this address will release the Offline Phase. In this way the List of Offline Slave addresses *(LOS)* is set.
  The user himself can decide how the system reacts to a configuration error on the AS-Interface. The master can release the Offline-Phase in critical situations, i.e. only with certain slave addresses, while in less critical situations only an error message is sent to the host, but AS-i is still running.

The functionality "Offline Phase when Configuration Error" will be as well supported by the "AS-i Control Tools" version 3.0.

Issue date 27.07.98

7 Data Exchange via the Dual Port RAM (DPRAM)

7.1 Principle

The AS-i (Control) data is stored in a DPRAM which can be accessed by b*urst-memory-accesses*. That means that the device driver is very simple and can easily be written in any language and for any operating system.

7.2 DPRAM Access

The board uses 3 addresses for data exchange, the I/O base address and the two following bytes with offset 1 and 2:

offset	reading	writing
0	data register	data register
1	- - - -	INDEX-register
2	- - - -	PAGE-register (Bits 0,1) reset (Bit 4) (min. 200ms)

Reading and writing at address with offset 0 accesses the memory location which is determined by the PAGE and INDEX register (0x100*PAGE + INDEX). Every access at offset 0 increments the INDEX register so 256 consecutive bytes can be written or read with one writing of the PAGE and INDEX Register.

Reading beyond the border of the page is not possible. On overflow of the index register (after reading the 256th byte of the page) both the index register and the page register are cleared, i.e. the access continues on page 0 (byte 0).

After a reset of the board, the DPRAM is completely new written.

Example (assume the I/O base address is 300_h):

If you want to read the AS-i information which is represented in 17 consecutive bytes (DPRAM-addresses 102_h - 111_h), you first have to initialize the INDEX and the PAGE-Register. Since the information starts at DPRAM-address 101_h the INDEX-Register (I/O port 301_h) must be set to 2 and the PAGE-Register (I/O port 302_h) to 1. After that the data can be read at I/O port 300_h consecutively without further initialization. The write access works the same way.

Issue date 27.07.98

7.3 Detailed DPRAM Address Map

page	adress	size	data	access
page 0: version data	000_h–$0AF_h$	—	data	—
	$0B0_h$–$0DF_h$	48	—	r/-
	$0E0_h$–$0EF_h$	16	*master name and version*	r/-
	$0F0_h$–$0F7_h$	8	*installed software*	r/-
	0F8h–$0FF_h$	—	*version ID string*	—
page 1: AS-i data	100_h	1	—	r/w
	101_h	1	*watchdog enable*	r/w
	102_h-111_h	16	*watchdog counter*	r/-
	112_h	1	input data image IDI	r/-
	113_h-122_h	16	execution control flags ec-flags	r/w
	123_h	1	output data image *ODI* (inverted!)	r/w
	124_h-127_h	4	host interface flags hi-flags	r/-
	128_h-$12B_h$	4	list of active slaves LAS	r/-
	$12C_h$-$12F_h$	4	list of detected slaves LDS	r/w
	130_h-$14F_h$	32	list of projected slaves LPS	r/-
	150_h-$16F_h$	32	configuration data image CDI	r/w
	170_h-$17F_h$	16	permanent configuration data PCD	r/w
	180_h-$18F_h$	16	parameter image PI	r/w
	190_h-$1CF_h$	—	permanent parameter PP	—
	$1D0_h$-$1EF_h$	4	—	r/w
	$1D4_h$-$1EF_h$	—	list of 'offline slaves' *LOS*	—
	$1F0_h$	1	—	r/w
	$1F1_h$	1	'*command*'	r/w
	$1F2_h$	1	parameter #1: *data*	r/w
	$1F3_h$	1	parameter #2: *address*	r/w
	$1F4_h$-$1FC_h$	—	parameter #3: *address extension* (high byte)	—
	$1FD_h$	1	—	r/w
	$1FE_h$	1	interrupt enable	r/w
	$1FF_h$	1	interrupt event	r/w
For AS-i Control:				
page 2: AS-i Control memory	200_h-$2FF_h$	256	communication (user) memory	r/w

Issue date 27.07.98

page	adress	size	data	access
page 3: AS-i Control buffer	300_h-301_h	2	previous controller program cycle time	r/-
	302_h-303_h	2	maximum controller program cycle time	r/-
	304_h	1	AS-i Control flags	r/w
	305_h-$30F_h$	—	—	—
	310_h-$3FF_h$	240	program buffer	r/w

AS-i Master Execution Control Lists

All data of the AS-i master may be read at any time out of the DPRAM (addresses 102_h to $18F_h$).

The only lists, that are read cyclically by the AS-i master are the output data image *ODI*, the host interface flags (*hi-flags*) and the parameter image PI. Requests for changing all other *execution control lists* may be refused by the AS-i master or have to be copied to a non-volatile memory, so changes to the DPRAM have to be indicated by writing an appropriate value to the byte *command*.

The AS-i master writes cyclically the input data image *IDI*, the execution control flags (*ec-flags*), the list of active slaves *LAS*, the list of detected slaves *LDS* and the configuration data image *CDI*.

Additionally, the *installed software* string is updated.

Due to the internal processing of the output data image *ODI*, the user has to store it *inverted* to the DPRAM.

While a control program is running, it generates the new *ODI*, so the AS-i master writes it to the DPRAM image. If an *ODI* was written by the PC, it will be overwritten then.

Commands

The byte *command* is to issue orders to the host interface of the PC card.

These orders are acknowledged by clearing all bits of the byte *command* (00_h: positive acknowledgement; successful execution) or by setting all bits (FF_h: negative acknowledgement; failure).

If for carrying out the command the AS-i master executes a function which requires parameters that cannot be read out of the images of the AS-i master execution control lists in the DPRAM, the parameters has to be stored to the bytes *data* and *address*. If this function returns a value, the master will store this return value to the byte *data*.

Possible codes for the byte *command* are:

Issue date 27.07.98

3: *Set Permanent Parameter*

The image of the execution control list *PP* in the DPRAM is compared with the master-internal execution control list. For each difference the execution control function `Set_Permanent_Parameter()` is executed.

The AS-i master updates the image of *PP* in the DPRAM if it is not able or not allowed to carry out the order completely.

5: *Write Parameter*

The execution control function `Write_Parameter()` is executed. The slave address has to be stored in *address* and the parameters to send in *data*.

The slaves response to the parameter request is stored to the byte *data*. In Addition, the AS-i master updates the image of *PI* in the DPRAM.

⇒ If no return value is required, it is recommended to write directly to the image of the *PI* in the DPRAM. The AS-i master cyclically compares this to its internal copy of the *PI* and executes `Write_Parameter()` for each difference.

7: S*tore Actual Parameter*

The execution control function `Store_Actual_Parameter()` is executed and the images of the execution control lists *PP* and *PI* are updated in the DPRAM.

8: *Set Permanent Configuration*

The image of the execution control list *PCD* in the DPRAM is compared with the master-internal execution control list. For each difference the execution control function `Set_Permanent_ Configuration()` is executed.

The AS-i master updates the image of *PCD* in the DPRAM if it is not able or not allowed to carry out the order completely.

10: *Store Actual Configuration*

The execution control function `Store_Actual_Configuration()` is executed and the images of the execution control list *PCD* and the list of projected slaves *LPS* are updated in the DPRAM.

12: *Set LPS*

The execution control function `Set_LPS()` is executed using the image of the execution control list and the image of the execution control list *LPS* is updated in the DPRAM.

17: *Set Operation Mode*

If the value of *data* is not equal to zero, the AS-i master changes to the *configuration mode*. Else, the master tries to change to *protected mode*.

20: *Change Slave Address*

The execution control function `Change_Slave_Address` is executed. The new slave address has to be stored in *data* and the old slave address in *address*.

The AS-i master stores the extended status in the byte *data* (see chap. 10.2.7).

Issue date 27.07.98

22: *Execute Command*

The execution control function `Execute_Command()` is executed. The slave address has to be stored in *address* and the information part of the master request in *data*.

The AS-i master stores the slave response in the byte *data*.

50: *Write AS-i Control flags (AS-i Control)*

The AS-i master reads the image of the AS-i Control flags out of the DPRAM and copies it to the real flags; If the flag `reset` is set, the control program is copied from the EEPROM to the RAM (code space).

Finally, the AS-i master updates the image of the AS-i Control flags in DPRAM.

51: *Get Status* (AS-i Control)

The AS-i master updates the image of the AS-i control flags and the cycle time.

52: *Download (AS-i Control)*

The AS-i master reads up to 240 bytes control program from the program buffer in DPRAM and stores it to the EEPROM.

Size and start address of the control program block is read from the DPRAM bytes *data* (size), *address* and *address extension* (start address in EEPROM).

53*: Upload* (AS-i Control)

The AS-i master copies up to 240 bytes control program from the EEPROM to the program buffer in DPRAM.

Size and start address of the control program block is read from the DPRAM bytes *data* (size), *address* and *address extension* (start address in EEPROM).

54: *Write Communication Memory with Consistency* (AS-i Control)

The AS-i master reads a block of up to 240 bytes from the program buffer in DPRAM and copies it (with block-wide consistency) to the communication memory (also DPRAM).

Size and start address of the block to copy is read from the DPRAM bytes *data* (size) and *address* (start address).

⇒ If no consistency is needed, direct write accesses to the communication memory are possible at any time.

56: *Read Communication Memory with Consistency* (AS-i Control)

The AS-i master reads a block of up to 240 bytes from the communication memory in DPRAM and copies it (with block-wide consistency) to the program buffer (in the DPRAM, too).

Size and start address of the block to copy is read from the DPRAM bytes *data* (size) and *address* (start address).

⇒ If no consistency is needed, direct read accesses to the communication memory are possible at any time.

Issue date 27.07.98

70: Read and clear LCS (advanced diagnostics)

The *LCS* is copied to the bytes 0 to 3 of the program buffer. After that, the *LCS* is cleared.

This function is available only with the 'advanced diagnostics' extension.

⇒ The *LCS* is cleared each time this command is carried out.

71: Read and clear Transmission Error Counters (advanced diagnostics)

The Transmission *Error Counters* are copied to the bytes 0 to 32 of the program buffer (one byte for each AS-i slave address) and the *Error Counters* are cleared.

This function is available only with the 'advanced diagnostics' extension.

⇒ The transmission error counters are cleared each time this command is carried out.

72: Set LOS (advanced diagnostics)

The list of 'off-line slaves' *LOS* is updated using the image of the *LOS* and the image is updated in the DPRAM.

If a bit in the list of 'off-line slaves' is set, each configuration error at the corresponding AS-i slave address forces the execution control to enter the *Off-line phase*. This is effective only while the execution control is in normal operation mode within p*rotected mode*.

This function is available only with the 'advanced diagnostics' extension.

⇒ To leave the Off-line phase, the *LOS* has to be rewritten or cleared.

128: *Update DPRAM*

The whole DPRAM (except watchdog and output data image) is rewritten by the AS-i master in order to eliminate inconsistencies.

129: *Read AS-i Telegram and Error Counters*

The counters for the AS-i master requests and the erroneous or missing AS-i slave responses are copied to the bytes 0 to 3 of the program buffer. After that, the counters are cleared.

Note

This function is available only with the EMC test extension.

Issue date 27.07.98

130: *EMC Test mode (optional)*

The AS-i master changes to EMC test mode.

The EMC test mode parameters are read out of the program buffer:

Byte 0, Bit 0 = 1: repetitions are allowed.

Byte 0, Bit 1 = 1: the slave responses are checked for correct data.

Byte 0: Bits 3 to 7: the expected slave response

Byte 1: the request to send

Bytes 2 to 5: list of slaves to send requests to

The counters for the AS-i master requests and the erroneous or missing AS-i slave responses are copied to the bytes 8 to 11 of the program buffer.

After that, the counters are cleared.

Note

This function is available only with the EMC test extension.

Watchdog

If the contents of the byte *watchdog enable* is not equal to zero, the watchdog is enabled and the user has to write cyclically a value different to zero to the byte *watchdog counter*. The AS-i master decrements the watchdog counter every 10 ms. If Zero is reached, it changes to the Off-line Phase.

This way, the maximum watchdog time is written in units of 10 ms. That allows supervising times from 10 ms up to 2.55 seconds.

To disable the watchdog, the user has to write 00h to the byte *watchdog enable*.

Timing out may be recognized by *watchdog enable* = 0 and *watchdog counter* = 0.

Config_ok Delay

It is recommended to read the execution control flags every time the input data image *IDI* is read. Only if the *Config_ok* flag is set, the user can be sure that all input data is valid.

The PC needs a certain time to read the 17 bytes out of the DPRAM (in addition, the PC may be interrupted by other tasks while reading), and the AS-i master is able to alter the DPRAM at any time.

To guarantee the input data read by the PC is valid when it reads ‘configuration O.K.’, the 0-to-1 transition of the *Config_ok* flag is delayed. The user has to read both, input data and execution control flags before this delay runs out.

The DPRAM byte *Config_ok delay* holds the maximum time for this delay in units of 10ms. If the user is sure reading of *IDI* and flags does not take more than 10ms, he does not need to change the default value of 2.

Issue date 27.07.98

Interrupts

The PC board is able to release interrupts on the ISA bus of the Personal Computer. Each interrupt source has to be enabled by setting the corresponding bit in the byte *interrupt enable*. It is possible to activate several interrupt sources at one time.

The register *interrupt event* holds the source(s) of the pending interrupt and should be cleared by the users interrupt routine.

The interrupt sources are as follows:

Bit 0: Changes on *Config_ok*
Both, 0-to-1 and 1-to-0 transition of the execution control flag *Config_ok* release an interrupt.

Bit 1: Changes on the Input Data Image *IDI*
If this bit is set, the input data image is tested cyclically for changes by the PC card. As soon as a change is detected, an interrupt is released.

Bit 2: AS-i Cycle
During the *inclusion phase* of the execution control (i.e. each AS-i cycle in normal operation mode), an interrupt will be released.

The AS-i Cycle time for the Pepperl+Fuchs AS-i masters is between 300µs (one AS-i slave only) and about 5ms (31 AS-i slaves).

The interrupt number is selected by jumpers.

Issue date 27.07.98

8 PC Software

8.1 ASISHELL"

ASISHELL is an simple menu to start the other executables without knowledge about their command line parameters.

With the upper part of the menu items like the I/O address of the PC card or the name of the source code file for AS-i Control can be specified.

The second set of menu contains items to select the compiler language and to process the controlling program by calling an text editor or the Tokenizer/Loader ACF (see chap. 8.2.6).

ASI_VIS, ASI_MON or ASI_ID can be executed by selecting one of the three menu items below.

The fourth block contains items to control the execution of ASISHELL: leaving ASISHELL, calling the DOS command line interpreter and showing the output of previous called programs.

The last menu item is to specify the command line to call the text editor. This makes it possible for every user to use the editor he is the most familiar with. Selecting the menu item “edit” makes ASISHELL to pass this line to the operating system, after it replaced ”$f” by the name of the source file and ”$l” by the line number where ACF found an error.

8.2 Monitor Program "ASI_MON"

The AS-i Monitor is a simple, screen-oriented MS-DOS program for monitoring, displaying (monitor mode) and editing (editor mode) of all process data and conditions of the AS-i PC2-Masters.

8.2.1 Starting the Program

In its default setting, the AS-i Monitor searches for AS-i PC2 boards and uses the first one found. If you want to specify a certain board, add the I/O address to the command line. For example, the command

```
ASI_MON 320
```

runs the program using the AS-i PC2 board at I/O address 320h. Possible values are 300_{h}, 304_{h}, 308_{h}, $30C_{h}$, 320_{h}, 324_{h}, 328_{h} and $32C_{h}$ for the I/O address.

8.2.2 Screen Design

In order to display the large amount of data orderly, two screen pages are used. On both pages, the current AS-i flag conditions are displayed at the top. The lower halves of both screens contain the slave-related AS-i data. Two footer lines display the function key assignments and status information. An additional screen page is available for logging any errors that might occur.

Issue date 27.07.98

8.2.3 Status Messages

If necessary, status messages with the following meanings are displayed in the bottom left corner of the screen:

*	Communication is active.
r	The Master re-reads all AS-i data. Normally, only the data that can be modified by changes in the AS-i circuit are updated. The master-internal data such as the configuration are read automatically only at the start and in longer intervals.
p	One of the two AS-i functions `Store_Actual_Parameters` or `Store_Actual_Configuration` is currently active. This display was added since these functions can be carried out also in the monitor mode with the function keys F5 and F6.
err	An error occurred during the last communication. You can display a detailed error description on the error screen.

8.2.4 Function Keys

The AS-i monitor is controlled via function keys in both the monitor and the editor modes.

F1	Help
F3	Toggles between monitor and editor modes
F5	Executes AS-i function `Store_Actual_Parameters`
F6	Executes AS-i function `Store_Actual_Configuration`
F9	Updates all AS-i data including the master-internal data from the Master. This is necessary when you change the Master, for example.
F10	Exit AS-i Monitor
Home, End, PgUp, PgDn	Change the screen page; the error display screen is the last page.

8.2.5 Editor Mode

In the editor mode, you cannot only display all process data and conditions, but also change them. Just move the cursor to the information you want to change and overwrite it.

You can move the cursor only to the fields that can actually be changed. Any valid entries are executed immediately.

Except for the two functions `Store_Actual_Parameters` and `Store_Actual_Configuration` that are executed with function keys, all control level functions of the current AS-i Master specification can be modified.

Changing Hexadecimal Information

Since they have a value range between zero and 15, input and output data, slave parameters and configuration data are displayed in hexadecimal notation.

Issue date 27.07.98

In order to change these values, move the cursor to the respective position and enter the new value as a single-digit hexadecimal value. Any entries that cannot be interpreted as hexadecimal values are ignored.

Changing Binary Information

The three slave lists (*LDS*, *LAS*, and *LPS*) and the execution control flags (in the AS-i Master) are displayed in binary notation. A set bit is represented by an ”X”, and the erased bits are represented as a period.

After you move the cursor to the proper position, press the spacebar to toggle the values.

Changing the Slave Addresses

A slave address can be overwritten by entering a two-digit decimal number. Invalid keystrokes are ignored, but both digits must be entered one after the other.

8.2.6 Tokenizer/Loader "ACF"

The ACF program serves to read the control program from a source text file and convert it to a code that can be read by AS-i Control. This process is called ”tokenizing”.

The generated code is not written to a new file, but can—if desired—be stored directly in the AS-i Control’s EEPROM. Simultaneously, the AS-i configuration contained in the Setup section of the source text file is entered into the AS-i Control configuration data (”downloading”).

The loaded control program can optionally be activated in the AS-i Control.

During the downloading, the following AS-i host functions are executed:

- Slaves listed in the Setup sections are entered into the list of projected slaves (*LPS*).
- The parameters in the Setup sections (default Fhex) are sent to all slaves in the *LPS*.
- These parameters are entered into the AS-i Control permanent parameters list *PP*.
- I/O codes and configuration IDs for these slaves are entered into the AS-i Control permanent configuration data list *PCD*.

The control program is stored in the AS-i Control’s EEPROM. Before such a program can be started, it must be reset, i.e. it is read from the EEPROM.

In addition to the source text file name, the following parameters can be added when calling the ACF program:

`/i`*<settings>* ”interface settings”

The parameter <*settings*> is a string with format like ”IO<*port*>”. Use this command line parameter if you want to specify a certain board. Possible values for *<port>* are 300h, 304h, 308h, 30Ch, 320h, 324h, 328h and 32Ch.

`/l`*<path>* ”library path”

The parameter <*path*> names the path, where ACF looks for function modules and save all temporary files.

Issue date 27.07.98

`/c<code>` "compiler language"

Possible values of <code> are 001 for english and 049 for german.

`/u+` or `/u–` "upload"

This switch determines whether ACF should read the control program in AS-i Control and write it to the source text file (`/u+`).

`/o+` or `/o–` "overwrite source file"

(makes sense only in combination with `/u+`).
When the name of an existing source text file is indicated during the upload, it is normally overwritten after an additional confirmation by the user.
This process can be suppressed with `/o+`.

`/d+` or `/d–` "download"

This switch determines whether ACF should only check the source text file for correct syntax (`/d-`) or also store it in the AS-i Control's EEPROM (`/d+`).

`/s+`, `/s–` or `/s0` "start control program"

Determines whether the control program loaded with /d+ should be activated (`/s+`), stopped (`/s–`), or whether the start flag condition should remain unchanged (`/s0`).
When a control program is deactivated with `/s–`, AS-i Control acts like a normal serial AS-i Master even during normal operation in the protected operating mode.

`/v+` or `/v–` "produce verbose output"

`/v+` causes a very detailed screen output that displays the individual steps during the token generation and the download to AS-i Control.
Independent of `/v+` and `/v–`, the text file CTRL2.{I} is generated each time ACF is called. This file contains these detailed messages. The file CTRL.{a} contains the assembly listing of the control program.

Issue date 27.07.98

8.3 Display Program "ASI_VIS"

The ASI_VIS program displays the AS-i circuit's input and output data and the AS-i Control's user memory while a control program is running.

The command parameters have the following functions:

`/p`*<port>* "select I/O port "

In its default setting, ASI_VIS searches for AS-i PC2 boards and uses the first one found. Use this command line parameter if you want to specify a certain board.
Possible values for *<port>* are 300h, 304h, 308h, 30Ch, 320h, 324h, 328h and 32Ch.

`/h` or `/?` "display help"

Displays help text with the various settings and the default settings stored with `/w`.

Issue date 27.07.98

9 PC Driver

The driver allows compatibility to other Pepperl+Fuchs masters. The following sample program demonstrates how the driver is integrated into user programs.

If this compatibility is not necessary, it is better to access the board directly. Chapter 10.3.1 contains an ANSI-C sample program (ansidemo.c), that is also included on the enclosed disc.

```
#include <stdio.h>
#include <conio.h>
#include "intface.h"

main (int argc, char *argv[])

{
     ASI_DATA message, answer;
     int back, i;

     /* initialize driver */
     back = AsiPc2Init (0);
     if (back != 0) {
          printf ("Error: <%04x>\n", back);
          return 1;
     }

     do {
          /* read input data */
          back = ASI_com (EIN_DATEN_LESEN_SER, message, answer);

          /* error message */
          if (back != 0) {
               printf ("Error: <%04x>\n", back);
               return 1;
     }

     /* display input telegram */
     else {
          printf("data: ");
          for (i = 0; i < (int) ASI_answer_len; i++)
               printf (" %02X", answer [i]);
          putchar ('\r');
     }

/* repeat until key pressed */
} while (kbhit () == 0);

/* End */
return (0);
}
```

Issue date 27.07.98

9.1 New Data Types

The only newly introduced data type is ASI_DATA. ASI_DATA is a field with elements of the unsigned char type that is used to accept the user data. You can access the individual elements of this data field with the customary C subscripts. ”data[3]”, for example, is the third element in the ASI_DATA data field.

You can also pass your own data structure to the communication routines. Using ASI_DATA, however, ensures a sufficient data field size.

9.2 Global Static Variables

Some information that is normally not needed by the main program is stored in static variables. In most cases, these are status information for the most recent transmissions.

All of these variables except ASI_retry are set by the driver and should only be read by the main program.

ASI_message	type (AS-i function) of the PC’s most recent send message
ASI_message_len	its length
ASI_answer	type of the master’s most recent response message
ASI_answer_len	its length

9.3 Functions

```
int AsiPc2Init (unsigned int io_addr);
```

This function must be called prior to the first communication since it initializes the driver. The valid I/O addresses are 300_h, 304_h, 308_h, $30C_h$, 320_h, 324_h, 328_h and $32C_h$. If zero is passed as I/O address, the driver searches for AS-i PC2 boards and uses the first one found.

Return value:

00_h	No error
01_h	The value of the io_addr parameter is out of range.
02_h	No AS-i PC2 board found or wrong firmware version detected.
03_h	No AS-i PC2 board found.

```
int ASI_com (char command, ASI_DATA message, ASI_DATA answer);
```

This is the general function for the communication with the Master. A message of the command type is sent to the Master. The user data are transferred in message, and the user data of the response are stored in answer.

Issue date 27.07.98

The return value's low bytes (hexadecimal) have the following meanings:

00_h	error-free communication
01_h	`ASI_init()` had not yet been called
02_h	invalid AS-i command in `command`
03_h	invalid handshake response from the AS-i PC card
1106_h	handshake timeout

```
int ASI_is_command (char command);
```

This function is used to check for a valid message type.

Return value:

1	`command` is a valid message type
0	`command` is an invalid message type

```
int ASI_get_message_len (char command, ASI_DATA message);
int ASI_get_answer_len (char command, ASI_DATA answer);
```

With these functions, you can determine the number of user data bytes in a send or response message. For message types that permit a variable message length, the actual length is determined by the data actually transmitted in `message` or `answer`.

If `message` or `answer` are unknown, NULL can be transmitted. Both functions then return the maximum number of data bytes.

```
int AsiPc2Reset (void);
```

This function performs a power-up reset for the AS-i PC2 board.

Return value:

00h	No error
01h	`ASI_init()` had not yet been called

Issue date 27.07.98

10 Appendix

10.1 Driver Commands in compatibility with other masters of Pepperl+Fuchs

In the following table for each driver command there are both the value of the command byte *k* and the value of data byte b_i of host message and master message.

Note

The command byte k is only related to the driver commands, so don't confound it with the byte 'command' of the DPRAM table.
This commands shouldn't be used, if the compatiliby to other masters of Pepperl+Fuchs isn't necessary. In this case it is more usefully to communicate with the PC card directly through the DPRAM.

Commands According to the AS-i Master Specification			
message	*k*	b_i *(host message data)*	b_i *(master message data)*
read input data	*71hex*	-	b_1...b_{16}*:input data*
write output data	*70hex*	b_1...b_{16}*: output data*	b_1*: status*
write configured parameters	*61hex*	b_1*: slave address* b_2*: parameters*	b_1*: status*
read configured parameters	*62hex*	b_1*: slave address*	b_1*: parameters*
write actual parameters	*63hex*	b_1*: slave address* b_2*: parameters*	b_1*: counter read parameters (inverted in case of error)*
read actual parameters	*64hex*	b_1*: slave address*	b_1*: parameters*
configure actual parameters	*65hex*	-	b_1*: status*
write configuration data	*66hex*	b_1*: slave address* b_2*: configuration data*	b_1*: status*
read configuration data	*67hex*	b_1*: slave address*	b_1*: configuration data*
actual configuration	*68hex*	-	*b1: status*
read actual configuration	*69hex*	b_1*: slave address*	b_1*: configuration data*
write list of configured slaves	*6Ahex*	b_1 ... b_4*: LPS*	b_1*: status*
read list of configured slaves	*6Bhex*	-	b_1 ... b_4*: LPS*
read list of active slaves	*6Chex*	-	b_1 ... b_4*: LAS*

Issue date 27.07.98

read list of recognized slaves	*6Dhex*	*-*	*b_1 ... b_4: LDS*
read execution control flags	*72hex*	*-*	*b_1: execution control flags*
set operating mode	*73hex*	*b_1 = 0: protected operating mode* *b_1 = 1: configurating mode*	*b_1: status*
write host interface flags	*74hex*	*b_1: host interface flags*	*b_1: status*
change slave's operating address	*6Ehex*	*b_1: old slave address* *b_2: new slave address*	*b_1: status*
AS-i command call	*6Fhex*	*b_1: slave address* *b_2: AS-i information component*	*b_1: response from slave* *b_2: status*

Additional Driver Commands (not in AS-i Master Specification)			
message	*k*	*b_i (host message data)*	*b_i (master message data)*
replace all input and output data	*76hex*	*b_1...b_{16}: output data*	*b_1: execution control flags* *b_2...b_{17}: input data*
write selected output data	*77hex*	*b_1: first slave address* *b_2: number of slaves* *b_3...b_{18}: output data*	*b_1: status*
read selected input data	*78hex*	*b_1: first slave address* *b_2: number of slaves*	*b_1: execution control flags* *b_2...b_{17}: input data*

Issue date 27.07.98

Additional Driver Commands (not in AS-i Master Specification)			
read output data	*81hex*	*-*	*b_1...b_{16}: output data*
read master version	*7Dhex*	*b_1: ½ 0: version number (8 bytes)* *b_1: ½ 1: master name part1 (17 bytes)* *b_1: ½ 2: master name part2 (17 bytes)* *b_1: ½ 3: master version (17 bytes)* *b_1: ½ 4: installed software and host interface flags (17 bytes)*	*b_1: version information (8 oder 17 bytes)*

Note

The command "replace all input and output data" is the prefered command due to less overhead: the AS-i master has to wait only one time for the answer of the slave.

The functions 'write selected output data' and 'read selected input data' are only executed if the AS-i Master is in the normal mode.

10.2 Representation of Information in the User Data Bytes

10.2.1 Input and Output Data

For each slave, a four-digit binary number can be entered as input and output data. Input and output data can therefore range between 0 and 15.

For serial transmission, the data for two slaves are combined in a single byte. With message "q" (read input data), the Master therefore sends 32/2 = 16 bytes of user data.

The entries for low slave addresses are transmitted first. Byte 0, bits 0 through 3 (lower nibble) thus contains the input data of the slave with operating address zero; the upper nibble of the user data byte 15 contains the data of slave 31.

10.2.2 Slave Lists

In the slave lists (*LRS*, *LAS*, *LCS*), one bit is responsible for each slave address. Since 32 slaves can be connected to the AS-i, four bytes are required for each slave list.

Here, too, the data for slaves with the lowest addresses are transmitted first. Slave zero is entered in byte 0, bit 0 (20, LSB), slave 32 in byte 3, but 7 (27, MSB).

Issue date 27.07.98

10.2.3 Execution Control Flags (ec-flags)

Bit 0:	*config_OK*	no configuration error
Bit 1:	*LDS.0*	slave with address 0 present
Bit 2:	*Auto_Address_Assign*	automatic programming permitted
Bit 3:	*Auto_Address_Available*	automatic programming available
Bit 4:	*Configuration_Active*	configuration mode active
Bit 5:	*Normal_Operation_Active*	normal operation active
Bit 6:	*APF*	AS-i power failure
Bit 7:	*Offline_Ready*	offline mode active

10.2.4 Host Interface Flags (hi-flags)

Bit 0:	*data_exchange_active*	when bit 0 is set, the data communication between Master and slaves is enabled (the flag is transmitted inverted)
Bit 1:	*Offline*	When bit 1 is set, the Master is set to off-line mode
Bit 2:	*Auto_Address_Enable*	When bit 2 is set, automatic programming is disabled (the flag is transmitted inverted)

10.2.5 Installed Software/Flags of the Host Interface

At address 0E0h at the DPRAM a 16 bytes long character string is stored. This string contains the host interface states and the AS-i Master's capabilities as upper- and lower-case letters.

The letters have the following explanations:

Byte 0 `(C/c,D/d)`

The responding AS-i Master is an AS-i Control. The capital 'C' means that a control program is currently being executed. A lower-case 'c' means that either the start flag has not been set or that the AS-i Master's status does not permit the execution.
Is D/d displayed instead of C/c, the new software of AS-i Control is installed.

Issue date 27.07.98

Byte 1	(B/b) Bus-capable AS-i Master. The responding Master has a bus-capability (true for all AS-i PC2).
Byte 2	(F/f) The responding AS-i Master is featured with the optional AS-i error counter.
Byte 3	(E/e) The responding AS-i Master is featured with the optional EMC test mode.
Byte 4	(D/d) The responding AS-i Master is featured with the advanced diagnostics functionality
Byte 5	(C/c) The responding AS-i Master is featured with the function ‘Offline Phase when Configuration Error’
Byte 6-7	not used
Byte 8	(D/d) The *data_exchange_active* host interface flag is set/erased.
Byte 9	(O/o) The *offline* host interface flag is set/erased.
Byte 10	(A/a) The *auto_address_enable* host interface flag is set/erased.
Byte 11-13	not used
Byte 14	(W/w) The watchdog was activated/deactivated.
Byte 15	not used

Issue date 27.07.98

10.2.6 AS-i Control Flags, Start/Stop Code

Bit 0:	*start_flag*	When bit 0 is set, the control program is executed as soon as the AS-i Master's status permits the execution. (This flag is stored non-volatile.)
Bit 1:	*reset_bit*	The control program is read from the EEPROM prior to the start. In addition, the user memory (flag bytes) is erased. (Necessary after each download, not returned as AS-i Control flag.)
Bit 2:	*ignore_config_errors*;	When bit 2 is erased, the control program is stopped as soon as an AS-i configuration error occurs. (This flag is stored non-volatile.)
Bit 3:	*auto_start*	When bit 3 is set, AS-i Control waits for a push on the "set" button before it restarts the control programm. (This flag is stored non-volatile.)
Bit 4:	*map_counters*;	When bit 4 is set, the counter registers of the 15 counters can be accessed by `M 96.0` to `M 125.7.` (This flag is stored non-volatile.)

10.2.7 Status Byte

The Master returns the status byte if otherwise there would be no user data. Normally, it takes on only one of the two following values:

00	error while executing a host request
01	no error while executing a host request

If an error occurs by changing a slave address, detailed error messages are returned:

02	slave whose address should be changed does not exist
03	slave with operating address 0 present
04	address for which the slave should be programmed is already occupied
05	slave could not be programmed to address 0
06	slave could not be set for new operating address
07	new operating address could not be stored in slave's EEPROM

Issue date 27.07.98

10.3 Program Listings

10.3.1 Example for direct programming of the AS-i PC board in ANSI C

```
/* file: ansidemo.c */
/* ============================================================== */
/*                                                                */
/*      PEPPERL+FUCHS GMBH   MANNHEIM                           */
/*                                                                */
/* ============================================================== */
/* program:     ANSIDEMO                                          */
/*                                                                */
/* description: demo program for the ASi PC2 board                */
/*              (written in ANSI-C)                               */
/*              using one standard sensor (IO=0x01, ID=0x01)      */
/*              and one 4xout module (IO=0x08, ID=0x00)           */
/* ============================================================== */
/* author:      A.Quick                                           */
/* date:        04.09.96                                          */
/* ============================================================== */

/* ============================================================== */
/* I N C L U D E S                                                */

#include <stdio.h>
/* maybe these header names are different for your operation system */
#include <conio.h>
#include <time.h>

/* ============================================================== */
/* A S - i   C O N F I G U R A T I O N                            */

/* list of the projected slaves (LPS) */
static unsigned char lps[4] = {0x06,0x00,0x00,0x00};

/* list of the IO and ID data */
/* = permanet configuration data (PCD) */
static unsigned char pcd[32] = {0xFF,0x11,0x08,0xFF,
                                0xFF,0xFF,0xFF,0xFF,
                                0xFF,0xFF,0xFF,0xFF,
                                0xFF,0xFF,0xFF,0xFF,
                                0xFF,0xFF,0xFF,0xFF,
                                0xFF,0xFF,0xFF,0xFF,
                                0xFF,0xFF,0xFF,0xFF,
                                0xFF,0xFF,0xFF,0xFF};

/* list of the projected parameters (PP) */
static unsigned char pp[16] = {0xFF,0xFF,0xFF,0xFF,
                               0xFF,0xFF,0xFF,0xFF,
                               0xFF,0xFF,0xFF,0xFF,
                               0xFF,0xFF,0xFF,0xFF};

/* ============================================================== */
/* D E F I N E S                                                  */

/* base address of the AS-i PC2 board (has to be adapted to your */
/* own application, default address is 0x300) */
#define BASE_ADDRESS 0x300

/* offsets of the 3 registers of the AS-i PC2 board */
#define DATA 0
```

Issue date 27.07.98

```
#define INDEX 1
#define PAGE 2

/* addresses in the DPRAM */
#define D_COMMAND 0x1F0
#define D_DATA 0x1F1
#define D_LPS 0x12C
#define D_PCD 0x150
#define D_PP 0x180
#define D_PI 0x170
#define D_EC_FLAGS 0x112
#define D_WDOG_ENA 0x100
#define D_WDOG_CNT 0x101
#define D_IDI 0x102
#define D_ODI 0x113

/* command codes (used for the DPRAM's command byte) */
#define C_SET_MODE 17
#define C_SET_PP 3
#define C_SET_PCD 8
#define C_SET_LPS 12

/* macros for addressing, reading and writing of the DPRAM */
/* the functions "inp" and "outp" have to be adapted to your */
/* operating system */
#define Page(Loc)   outp(BASE_ADDRESS+PAGE, (Loc) >> 8)
#define Index(Loc)  outp(BASE_ADDRESS+INDEX, (Loc) & 0xFF)
#define Read(Var)   Var = (unsigned char)inp(BASE_ADDRESS+DATA)
#define Write(Var)  outp(BASE_ADDRESS+DATA, (Var))


/* ================================================================ */
/* F U N C T I O N   P R O T O T Y P E S                            */

static int send_cmd(unsigned char command);


/* ================================================================ */
/* FUNCTION:    main                                                */
/* ================================================================ */

void main (void)
{
    int error, i;
    int which_bar_to_print = 1;
    time_t timer;
    /* AS-i execution control flags */
    unsigned char flags;
    /* AS-i input data */
    unsigned char idata[16];
    /* AS-i output data */
    unsigned char odata[16] = {0xFF,0xFF,0xFF,0xFF,
                               0xFF,0xFF,0xFF,0xFF,
                               0xFF,0xFF,0xFF,0xFF,
                               0xFF,0xFF,0xFF,0xFF};

    /* disable watchdog */
    Page(D_WDOG_ENA);
    Index(D_WDOG_ENA);
    Write(0x00);
```

Issue date 27.07.98

```
/* switch to configuration mode */
Page(D_DATA);   /* write something bigger than 1 */
Index(D_DATA);  /* to the parameter "data" */
Write(0x01);
error = send_cmd(C_SET_MODE); /* command to set mode */
if(error)
{
    printf("can't switch to configuration mode.");
    return;
}

/* write new list of projected slaves (LPS) */
Page(D_LPS);
Index(D_LPS);
for(i=0 ; i<sizeof(lps) ; i++)  /* write LPS to DPRAM */
        Write( lps[i] );
error = send_cmd(C_SET_LPS);    /* command to update LPS */
if(error)
{
    printf("can't update LPS.");
    return;
}

/* write new permanent configuration data (PCD) */
Page(D_PCD);
Index(D_PCD);
for(i=0 ; i<sizeof(pcd) ; i++)  /* write PCD to DPRAM */
    Write( pcd[i] );
error = send_cmd(C_SET_PCD);    /* command to update PCD */
if(error)
{
    printf("can't update PCD.");
    return;
}

/* write new PP */
Page(D_PP);
Index(D_PP);
for(i=0 ; i<sizeof(pp) ; i++)   /* write PP to DPRAM */
    Write( pp[i] );
error = send_cmd(C_SET_PP);     /* command to update PP */
if(error)
{
    printf("can't update PP.");
    return;
}

/* return to protected operation mode */
Page(D_DATA);   /* write 0 to the parameter "data" */
Index(D_DATA);
Write(0x00);
error = send_cmd(C_SET_MODE); /* command to set mode */
if(error)
{
    printf("can't return to protected operation mode.");
    return;
}

/* wait for normal operation */
Page(D_EC_FLAGS);
time (&timer);  /* store actual time */
```

Issue date 27.07.98

```
do  /* loop until normal operation mode flag gets high */
{
    Index(D_EC_FLAGS);
    Read(flags);
    /* end program after 2 seconds timeout*/
    if(2 < difftime(time(NULL),timer))
    {
        printf ("can't start normal operation.");
        return;
    }

}while (0x00 == (flags & 0x20));

/* write projected parameter (PP) to parameter image (PI) */
/* AS-i PC2 board writes PI list only after power up! */
Page(D_PI);
Index(D_PI);
for(i=0 ; i<sizeof(pp) ; i++)
    Write( pp[i] );

/* initialize watchdog */
Page(D_WDOG_CNT);
Index(D_WDOG_CNT);
Write(0x0a);        /* load watchdog counter with 100 ms */
Page(D_WDOG_ENA);
Index(D_WDOG_ENA);
Write(0x01);        /* enable watchdog */

/* communication for ever */
do
{
    /* reload watchdog counter with 100 ms */
    Page(D_WDOG_CNT);
    Index(D_WDOG_CNT); /* index register will be incremented */
    Write(0x0a);       /* after every read or write access! */

    /* read input data from DPRAM */
    for(i=0;i<sizeof(idata);i++)
        Read( idata[i] );

    /* read execution control flags from DPRAM */
    Read(flags);

    /* copy sensor bit of slave 1 to bit 0 of slave 2 */
    odata[1] = idata[0] & 0x10; /* mask sensor bit of sl.1 */
    odata[1] = odata[1] >> 4;   /* shift from sl.3 to sl.2 */
    odata[1] = ~odata[1];       /* bitwise negation */

    /* write output data to DPRAM */
    for(i=0;i<sizeof(odata);i++)
        Write( odata[i] );

    /* print data to screen ("shift-" and "AND-operation" */
    /* necessary for right placing on screen!) */
    printf("input data (slave 1) = %1x  ", idata[0]>>4);
    printf("output data (slave 2)= %1x  ", odata[1]&0x0f);
    /* print "config OK" or "config err" to screen */
    if( 0x00 == (flags & 0x01) )    /* mask config flag */
        printf("config err...\r");
    else
    {
```

Issue date 27.07.98

```
            printf("config OK... ");
            /* print "active wheel" to screen */
            switch(which_bar_to_print)
            {
                case 1: printf("-\r");
                        which_bar_to_print = 2;
                        break;
                case 2: printf("/\r");
                        which_bar_to_print = 3;
                        break;
                case 3: printf("\\\r");
                        which_bar_to_print = 1;
                        break;
            }
        }

    } while(1);

} /* main */

/* ================================================================ */
/* FUNCTION:    send_cmd                                            */
/* DESCRIPTION: send a command to AS-i PC board                     */
/*                                                                  */
/* USAGE:       error = send_cmd(command);                          */
/*                                                                  */
/* PARAMETERS:                                                      */
/*      command: value for 'command'                                */
/*                                                                  */
/* RETURNS:                                                         */
/*      0x0000: o.k. (ACK)                                          */
/*      0x0001: o.k. (NAK)                                          */
/*      0x0003: invalid response                                    */
/*      0x0004: timeout                                             */
/* ================================================================ */

static int send_cmd(unsigned char command)
{
    time_t timer;
    register unsigned char c;

    Page(D_COMMAND);    /* write the command code */
    Index(D_COMMAND);   /* to DPRAM */
    Write(command);

    c = command;

    time(&timer);   /* store actual time */
    do
    {
        Index(D_COMMAND);   /* read command byte */
        Read(c);            /* from DPRAM! */
        if (c != command)   /* break loop, if command byte */
            break;          /* is changing... */

    }while(difftime(time(NULL),timer) < 0.2);
    /* ...or end loop after 200 ms (timeout)! */

    /* return codes depending on the command byte */
    if      (c == command)  return (0x0004);    /*  timeout */
    else if (c == 0x00)     return (0x0000);    /*  ACK     */
```

Issue date 27.07.98

```
    else if (c == 0xFF)     return (0x0001);    /*  NAK     */
    else                    return (0x0003);    /*  invalid */

}   /* send_cmd */

/* ================================================================ */
/* E N D   O F   M O D U L E                                        */
/* ================================================================ */
```

10.3.2 Example for direct programming of the AS-i PC in Pascal

```
{ file: pc2demo.pas }
{ ========================================================== }
{                                                            }
{               PEPPERL+FUCHS GMBH   MANNHEIM               }
{                                                            }
{ ========================================================== }
{ program:      pc2demo                                      }
{                                                            }
{ description:  demo program for the ASi PC2 board           }
{               using one standard sensor (IO=0x01, ID=0x01) }
{               and one 4xout module (IO=0x08, ID=0x00)      }
{ ========================================================== }
{ author:       A.Quick                                      }
{ date:         06.09.96                                     }
{ ========================================================== }

program ansidemo;

uses dos;

{ ========================================================== }
{ CONSTANTS

{ list of the projected slaves (LPS) }
const   lps : array [0..3] of byte = ($06,$00,$00,$00);

{ list of the IO and ID data }
{ = permanet configuration data (PCD) }
        pcd : array [0..31] of byte =  ($FF,$11,$08,$FF,
                                        $FF,$FF,$FF,$FF,
                                        $FF,$FF,$FF,$FF,
                                        $FF,$FF,$FF,$FF,
                                        $FF,$FF,$FF,$FF,
                                        $FF,$FF,$FF,$FF,
                                        $FF,$FF,$FF,$FF,
                                        $FF,$FF,$FF,$FF);

{ list of the projected parameters (PP) }
        pp : array [0..15] of byte =   ($FF,$FF,$FF,$FF,
                                        $FF,$FF,$FF,$FF,
                                        $FF,$FF,$FF,$FF,
                                        $FF,$FF,$FF,$FF);

{ base address of the AS-i PC2 board (has to be adapted to your }
{ own application, default address is 0x300) }
        BASE_ADDRESS    : word =$300;

{ offsets of the 3 registers of the AS-i PC2 board }
        DATA            : word = $0;
        INDEX           : word = $1;
```

Issue date 27.07.98

```
        PAGE            : word = $2;

{ addresses in the DPRAM }
        D_COMMAND       : word = $1F0;
        D_DATA          : word = $1F1;
        D_LPS           : word = $12C;
        D_PCD           : word = $150;
        D_PP            : word = $180;
        D_PI            : word = $170;
        D_EC_FLAGS      : word = $112;
        D_WDOG_ENA      : word = $100;
        D_WDOG_CNT      : word = $101;
        D_IDI           : word = $102;
        D_ODI           : word = $113;

{ command codes (used for the DPRAM's command byte) }
        C_SET_MODE      : byte = 17;
        C_SET_PP        : byte = 3;
        C_SET_PCD       : byte = 8;
        C_SET_LPS       : byte = 12;

{ ============================================================ }
{ GLOBAL VARIABLES                                             }

var     error, i                        : integer;
        flags                           : byte;
        idata                           : array[0..15] of byte;
        odata                           : array[0..15] of byte;
        hour, minute, second, sec100    : word;
        the_second                      : word;

{ ============================================================ }
{ PROCEDURE:    DPpage                                         }
{ DESCRIPTION:  write the page register of the DPRAM           }
{                                                              }
{ PARAMETERS:                                                  }
{               loc: Complete address to read or write from    }
{ ============================================================ }

procedure DPpage(loc : word);
begin
        port[BASE_ADDRESS + PAGE] := (loc shr 8); {shift right, 8 bit}
end;    { DPpage }


{ ============================================================ }
{ PROCEDURE:    DPindex                                        }
{ DESCRIPTION:  write the the index register of the DPRAM      }
{                                                              }
{ PARAMETERS:                                                  }
{               loc: Complete address to read or write from    }
{ ============================================================ }

procedure DPindex(loc : word);
begin
        port[BASE_ADDRESS + INDEX] := (loc AND $FF); {mask lower 8 bit}
end;    { DPindex }
{ ============================================================ }
{ PROCEDURE:    DPread                                         }
{ DESCRIPTION:  read from the address specified by page and    }
{               index register of the DPRAM                    }
```

Issue date 27.07.98

```
{                                                                }
{ RETURNS:      dat: DATA register of the DPRAM                  }
{ ============================================================== }

procedure DPread(var dat: byte);
begin
        dat := port[BASE_ADDRESS + DATA];
end;    { DPread }

{ ============================================================== }
{ PROCEDURE:    DPwrite                                          }
{ DESCRIPTION:  write to the address specified by page and       }
{               index register of the DPRAM                      }
{                                                                }
{ PARAMETERS:   variable: dat                                    }
{ ============================================================== }

procedure DPwrite(dat : byte);
begin
        port[BASE_ADDRESS + DATA] := dat;
end;    { DPwrite }

{ ============================================================== }
{ FUNCTION:     send_cmd                                         }
{ DESCRIPTION:  send a command to AS-i PC board                  }
{                                                                }
{ USAGE:        error := send_cmd(command);                      }
{                                                                }
{ PARAMETERS:                                                    }
{               command: value for 'command'                     }
{                                                                }
{ RETURNS:                                                       }
{               0:      o.k. (ACK)                               }
{               1:      o.k. (NAK)                               }
{               3:      invalid response                         }
{               4:      timeout                                  }
{ ============================================================== }

function send_cmd(command : byte): integer;
var     hour, minute, second, sec100    : word;
        the_sec100                      : word;
        c                               : byte;
begin
        c := command;

        DPpage(D_COMMAND);      { write command... }
        DPindex(D_COMMAND);     { ...to DPRAM }
        DPwrite(command);

        GetTime(hour, minute, second, sec100);
        the_sec100 := sec100;
        repeat
                DPindex(D_COMMAND);     { read command byte }
                DPread(c);              { from DPRAM }
                GetTime(hour, minute, second, sec100);

        until ( (sec100 - the_sec100 > 5) OR (c <> command) );
        { end loop, if command byte is changing... }
        { ... or after 500 ms (timeout) }

        { return codes depending on the command byte }
```

Issue date 27.07.98

```
        if      (c = command)   then send_cmd := 4
        else if (c = $00)       then send_cmd := 0
        else if (c = $FF)       then send_cmd := 1
        else                    send_cmd := 3;

end;    { send_cmd }

{ ============================================================ }
{ MAINPROGRAM                                                  }
{ ============================================================ }
begin

        { init }
        for i:=0 to 15 do
            odata[i] := $FF;

        { disable watchdog }
        DPpage(D_WDOG_ENA);
        DPindex(D_WDOG_ENA);
        DPwrite($00);

        { switch to configuration mode }
        DPpage(D_DATA);         { write something bigger than 0... }
        DPindex(D_DATA);        { ...to the parameter "data" }
        DPwrite($01);
        error := send_cmd(C_SET_MODE); { command to set mode }
        if error <> 0 then
        begin
                write('can`t switch to configuration mode.',error);
                exit;
        end;

        { write new list of projected slaves (LPS) }
        DPpage(D_LPS);
        DPindex(D_LPS);
        for i:=0 to 3 do        { write LPS to DPRAM }
                DPwrite( lps[i] );
        error := send_cmd(C_SET_LPS);   { command to update LPS }
        if error <> 0 then
        begin
                write('can`t update LPS.',error);
                exit;
        end;

        { write new permanent configuration data (PCD) }
        DPpage(D_PCD);
        DPindex(D_PCD);
        for i:=0 to 31 do       { write PCD to DPRAM }
                DPwrite( pcd[i] );
        error := send_cmd(C_SET_PCD);   { command to update PCD }
        if error <> 0 then
        begin
                write('can`t update PCD.',error);
                exit;
        end;

        { write new PP }
        DPpage(D_PP);
        DPindex(D_PP);
        for i:=0 to 15 do       { write PP to DPRAM }
                DPwrite( pp[i] );
```

Issue date 27.07.98

```
error := send_cmd(C_SET_PP);            { command to update PP }
if error <> 0 then
begin
        write('can`t update PP.',error);
        exit;
end;

{ return to protected operation mode }
DPpage(D_DATA); { write 0 to the parameter 'data' }
DPindex(D_DATA);
DPwrite($00);
error := send_cmd(C_SET_MODE); { command to set mode }
if error <> 0 then
begin
        write('can`t return to protected operation mode.',error);
        exit;
end;

{ wait for normal operation }
DPpage(D_EC_FLAGS);
GetTime(hour, minute, second, sec100);
the_second := second;   { store actual time }
repeat  { loop until normal operation mode flag gets high }
        DPindex(D_EC_FLAGS);
        DPread(flags);
        GetTime(hour, minute, second, sec100);
        { end program after 2 seconds timeout}
        if (second - the_second > 2) then
        begin
                write ('can`t start normal operation.');
                exit;
        end;

until ($00 <> (flags AND $20));

{ write projected parameter (PP) to parameter image (PI) }
{ AS-i PC2 board writes PI list only after power up! }
DPpage(D_PI);
DPindex(D_PI);
for i:=0 to 15 do
        DPwrite( pp[i] );

{ initialize watchdog }
DPpage(D_WDOG_CNT);
DPindex(D_WDOG_CNT);
DPwrite($0A);           { load watchdog counter with 100 ms }
DPpage(D_WDOG_ENA);
DPindex(D_WDOG_ENA);
DPwrite($01);           { enable watchdog }

{ communication for ever }
repeat
        { reload watchdog counter with 100 ms }
        DPpage(D_WDOG_CNT);
        DPindex(D_WDOG_CNT); { index register will be incremented }
        DPwrite($0A);       { after every read or write access! }

        { read input data from DPRAM }
        for i:=0 to 15 do
                DPread( idata[i] );
```

Issue date 27.07.98

```
                { read execution control flags from DPRAM }
                DPread(flags);

                { copy sensor bit of slave 1 to bit 0 of slave 2 }
                odata[1] := idata[0] AND $10;   { mask sensor bit of sl.1 }
                odata[1] := odata[1] shr 4;     { shift from sl.3 to sl.2 }
                odata[1] := NOT odata[1];       { bitwise negation }

                { write output data to DPRAM }
                for i:=0 to 15 do
                        DPwrite( odata[i] );

                { print data to screen ('shift-' and 'AND-operation' }
                { necessary for right placing on screen!) }
                write(' input data (slave 1) = ', idata[0] shr 4);
                write(' output data (slave 2)= ', odata[1] AND $0F);
                { print 'config OK' or 'config err' to screen }
                if (flags AND $01 = $00) then { mask config flag }
                        writeln(' config err...')
                else
                        writeln(' config OK... ');

        until (1<>1);   { communication for ever }

end.

{ END OF PROGRAM                                                }
{ ============================================================== }
```

10.3.3 Control Program for the I/O Addresses

```
/*****************************************************************
 *
 *  error = CheckAsiPc2 (IOport);
 *
 *  description:
 *      checks, if a AS-i PC2 board is present at the given I/O address
 *
 *  parameters:
 *      IOport: I/O address of AS-i PC2 to check
 *
 *  returns:
 *      0:  AS-i PC2 found at IOport
 *      1:  IOport out of range
 *      2:  no PC2 found at IOport
 *      3:  no PC2 found at IOport */

static int CheckAsiPc2 (int IOport)
{
    register int i;
    static unsigned char TestPad [512];
    static char AsiPc2Name = "AS-i PC2";

/* IOport out of range */
    if ((0x100 > IOport) || (0x3FF < IOport) || (0 != (0x0003 & IOport)))
        return (1);

/* read page 0 */
    for (i = 0; sizeof (TestPad) > i; i++)
        TestPad [i] = (unsigned char) inp (IOport +O_DATA);
```

Issue date 27.07.98

```
/* search for name string */
    for (i = 0; sizeof (TestPad) > i; i++)
        if (0 == strncmp (AsiPc2Name, (char *) (TestPad +i), 8))
            break;

/* no name string found */
    if (sizeof (TestPad) == i)
        return (3);

/* check firmware version identifier */
    if ((!isdigit (TestPad [i +64]))
     || (!isdigit (TestPad [i +65]))
     || (!isdigit (TestPad [i +66]))
     || (!isdigit (TestPad [i +67]))
     || (!isdigit (TestPad [i +68]))
     || (!isdigit (TestPad [i +69]))
     || (!isdigit (TestPad [i +70]))
     || (!isdigit (TestPad [i +71])))
        return (2);

/* AS-i PC2 found */
    return (0);
}
```

Issue date 27.07.98

Ausgabedatum 27.07.98

Pepperl+Fuchs GmbH · 68301 Mannheim · Telefon (06 21) 7 76-11 11 · Telefax (06 21) 7 76-10 00 · Internet http://www.pepperl-fuchs.com