



DIN EN ISO 9001:2000
certified



ADDI-DATA GmbH
Dieselstraße 3
D-77833 OTTERSWEIER
+49 (0)7223 / 9493 - 0

Technical description

APCI-1500, CPCI-1500

Digital I/O board, optically isolated

Edition: 12.08 - 01/2008

Product information

This manual contains the technical installation and important instructions for correct commissioning and usage, as well as production information according to the current status before printing. The content of this manual and the technical product data may be changed without prior notice. ADDI-DATA GmbH reserves the right to make changes to the technical data and the materials included herein.

Warranty and liability

The user is not permitted to make changes to the product beyond the intended use, or to interfere with the product in any other way.

ADDI-DATA shall not be liable for obvious printing and phrasing errors. In addition, ADDI DATA, if legally permissible, shall not be liable for personal injury or damage to materials caused by improper installation and/or commissioning of the board by the user or improper use, for example, if the board is operated despite faulty safety and protection devices, or if notes in the operating instructions regarding transport, storage, installation, commissioning, operation, thresholds, etc. are not taken into consideration. Liability is further excluded if the operator changes the board or the source code files without authorisation and/or if the operator is guilty of not monitoring the permanent operational capability of working parts and this has led to damage.

Copyright

This manual, which is intended for the operator and its staff only, is protected by copyright. Duplication of the information contained in the operating instructions and of any other product information, or disclosure of this information for use by third parties, is not permitted, unless this right has been granted by the product licence issued. Non-compliance with this could lead to civil and criminal proceedings.

ADDI-DATA software product licence

Please read this licence carefully before using the standard software. The customer is only granted the right to use this software if he/she agrees with the conditions of this licence.

The software must only be used to set up the ADDI-DATA boards.

Reproduction of the software is forbidden (except for back-up and for exchange of faulty data carriers). Disassembly, decompilation, decryption and reverse engineering of the software are forbidden. This licence and the software may be transferred to a third party if this party has acquired a board by purchase, has agreed to all the conditions in this licence contract and the original owner does not keep any copies of the software.

Trademarks

- ADDI-DATA is a registered trademark of ADDI-DATA GmbH.
- Turbo Pascal, Delphi, Borland C, Borland C++ are registered trademarks of Borland Insight Company.
- Microsoft C, Visual C++, Windows XP, 98, Windows 2000, Windows 95, Windows NT, EmbeddedNT and MS DOS are registered trademarks of Microsoft Corporation.
- LabVIEW, LabWindows/CVI, DasyLab, Diadem are registered trademarks of National Instruments Corp.
- CompactPCI is a registered trademark of PCI Industrial Computer Manufacturers Group.
- VxWorks is a registered trademark of Wind River Systems Inc.

WARNING

The following risks result from improper implementation and from use of the board contrary to the regulations:



- ◆ Personal injury
- ◆ Damage to the board, PC and peripherals
- ◆ Pollution of the environment

◆ **Protect yourself, the others and the environment!**

◆ **Read carefully the safety precautions (yellow leaflet).**

If this leaflet is not with the documentation, please contact us and ask for it.

◆ **Observe the instructions of the manual.**

Make sure that you do not forget or skip any step. We are not liable for damages resulting from a wrong use of the board.

◆ **Used symbols:**



IMPORTANT!

designates hints and other useful information.



WARNING!

It designates a possibly dangerous situation.

If the instructions are ignored the board, PC and/or peripheral may be destroyed.

1	DEFINITION OF APPLICATION	8
1.1	Intended use	8
1.2	Usage restrictions.....	8
1.3	General description of the board	8
2	USER	10
2.1	Qualification	10
2.2	Country-specific regulations.....	10
3	HANDLING OF THE BOARD	11
4	TECHNICAL DATA	12
4.1	Electromagnetic compatibility (EMC)	12
4.2	Physical set-up of the board.....	12
4.3	Options	13
4.4	Limit values.....	13
4.5	Component scheme.....	16
5	INSTALLATION OF THE BOARD	18
5.1	Installing the APCI-1500.....	18
5.1.1	Opening the PC	18
5.1.2	Selecting a free slot	18
5.1.3	Plugging the board into the slot.....	19
5.1.4	Closing the PC	19
5.2	Installing the CPCI-1500	20
6	SOFTWARE	22
6.1	Board registration with ADDIREG.....	23
6.1.1	Program description	23
6.1.2	Registering a new board.....	27
6.1.3	Changing the registration of a board	27
6.2	Questions and software downloads on the web.....	28
7	CONNECTING THE PERIPHERAL.....	29
7.1	Connector pin assignment.....	29
7.2	Connection principle	29
7.3	Connection examples.....	30
8	FUNCTIONS OF THE BOARD	32

8.1	Description of the board	32
8.1.1	Block diagram	32
8.2	Functions	33
8.2.1	Digital inputs	33
	Special input functions of the digital inputs	34
	1) Interrupt logic of the digital inputs 1-14	34
	2) Counter	36
8.2.2	Digital outputs	37
	Special functions	38
8.2.3	Interrupt	38
8.2.4	Counter/timer	39
	Input frequencies	40
	Data	40
	Option (only implemented for the APCI-1500)	40
9	STANDARD SOFTWARE	41
9.1	Introduction	41
9.2	Software functions	42
9.2.1	Base address and interrupt	42
	1) i_APCI1500_InitCompiler (..)	42
	2) i_APCI1500_CheckAndGetPCISlotNumber (...)	43
	3) i_APCI1500_SetBoardInformation (...)	44
	4) i_APCI1500_GetHardwareInformation (...)	45
	5) i_APCI1500_CloseBoardHandle (..)	46
9.2.2	Interrupt	47
	1) i_APCI1500_SetBoardIntRoutineDos (..)	47
	2) i_APCI1500_SetBoardIntRoutineVBDos (..)	50
	3) i_APCI1500_SetBoardIntRoutineWin16 (..)	52
	4) i_APCI1500_SetBoardIntRoutineWin32 (..)	55
	5) i_APCI1500_TestInterrupt (..)	61
	6) i_APCI1500_ResetBoardIntRoutine (..)	62
9.2.3	Kernel functions	63
	1) i_APCI1500_KRNL_Read16DigitalInput (...)	63
	2) v_APCI1500_KRNL_Set16DigitalOutputOn (...)	64
9.2.4	Digital input channel	65
	1) i_APCI1500_Read1DigitalInput (...)	65
	2) i_APCI1500_Read8DigitalInput (...)	65
	3) i_APCI1500_Read16DigitalInput (...)	66
9.2.5	Digital input channel - events	67
	1) i_APCI1500_SetInputEventMask (...)	67
	2) i_APCI1500_StartInputEvent (...)	69
	3) i_APCI1500_StopInputEvent (...)	69
9.2.6	Digital output channel	70
	1) i_APCI1500_SetOutputMemoryOn (...)	70
	2) i_APCI1500_SetOutputMemoryOff (...)	70
	3) i_APCI1500_Set1DigitalOutputOn (...)	71
	4) i_APCI1500_Set1DigitalOutputOff (...)	71

5)	i_APCI1500_Set8DigitalOutputOn (...)	72
6)	i_APCI1500_Set8DigitalOutputOff (...)	73
7)	v_APCI1500_Set16DigitalOutputOn (...)	74
8)	v_APCI1500_Set16DigitalOutputOff (...)	75
9.2.7	Timer/counter and watchdog	75
1)	i_APCI1500_InitTimerInputClock (...)	75
2)	i_APCI1500_InitTimerCounter1 (...)	76
3)	i_APCI1500_InitTimerCounter2 (...)	77
4)	i_APCI1500_InitWatchdogCounter3 (...)	79
5)	i_APCI1500_StartTimerCounter1(...)	80
6)	i_APCI1500_StartTimerCounter2 (...)	80
7)	i_APCI1500_StartCounter3 (...)	81
8)	i_APCI1500_StopTimerCounter1 (...)	81
9)	i_APCI1500_StopTimerCounter2 (...)	82
10)	i_APCI1500_StopCounter3 (...)	82
11)	i_APCI1500_TriggerTimerCounter1 (...)	82
12)	i_APCI1500_TriggerTimerCounter2 (...)	83
13)	i_APCI1500_TriggerCounter3 (...)	83
14)	i_APCI1500_Watchdog (...)	83
15)	i_APCI1500_ReadTimerCounter1 (...)	84
16)	i_APCI1500_ReadTimerCounter2 (...)	84
17)	i_APCI1500_ReadCounter3 (...)	85
10	GLOSSARY	86
11	INDEX	89

Figures

Fig. 3-1: Correct handling of the CPCI-1500	11
Fig. 3-2: Correct handling of the APCI-1500	11
Fig. 4-1: Component scheme of the APCI-1500	16
Fig. 4-2: Component scheme of the CPCI-1500	17
Fig. 5-1: PCI-5V slot (32-bit)	18
Fig. 5-2: Inserting the board	19
Fig. 5-3: Fastening the board at the back cover	19
Fig. 5-4: Types of slots for CompactPCI boards.....	20
Fig. 5-5: Pushing a CPCI board into a rack	20
Fig. 5-6: Connector keying	21
Fig. 6-1: ADDIREG registration program (example).....	23
Fig. 6-2: Configuring a new board	25
Fig. 6-3: PCI Boards	26
Fig. 7-1: 37-pin SUB-D male connector	29
Fig. 7-2: Connection principle of the input and output channels	29
Fig. 7-3: Connection examples for the input and output channels .	30
Fig. 7-4: Connection to the screw terminal panel.....	31
Fig. 8-1: Block diagram of the APCI-1500	32
Fig. 8-2: Block diagram of the CPCI-1500	32
Fig. 8-3: Interrupt logic – Digital inputs 1-8.....	35
Fig. 8-4: Protection circuitry for the inputs	36
Fig. 8-5: Protection circuitry for the outputs	38

Tables

Table 9-1: Type Declaration for Dos and Windows 3.1X	41
Table 9-2: Type Declaration for Windows 95/98/NT.....	41
Table 9-3: Interrupt mask.....	48

1 DEFINITION OF APPLICATION

1.1 Intended use

The **APCI-1500** board must be inserted in a PC with PCI 5V/32-bit slots which is used as electrical equipment for measurement, control and laboratory pursuant to the norm EN 61010-1 (IEC 61010-1). The used personal computer (PC) must fulfil the requirements of IEC 60950-1 or EN 60950-1 and 55022 or IEC/CISPR 22 and EN 55024 or IEC/CISPR 24.

The use of the board **APCI-1500** in combination with external screw terminal panels requires correct installation according to IEC 60439-1 or EN 60439-1 (switch cabinet / switch box).

The **CPCI-1500** board must be inserted in a CompactPCI/PXI computer with Compact PCI 5V/32-bit slots which is used as electrical equipment for measurement, control and laboratory pursuant to the norm EN 61010-1 (IEC 61010-1). The used personal computer (PC) must fulfil the requirements of IEC 60950-1 or EN 60950-1 and 55022 or IEC/CISPR 22 and EN 55024 or IEC/CISPR 24.

The use of the board **CPCI-1500** in combination with external screw terminal panels requires correct installation according to IEC 60439-1 or EN 60439-1 (switch cabinet / switch box).

1.2 Usage restrictions

The **APCI-/CPCI-1500** board must not be used as safety related part (SRP).

The board must not be used for safety related functions, for example for emergency stop functions.

The **APCI-/CPCI-1500** board must not be used in potentially explosive atmospheres.

The **APCI-/CPCI-1500** board must not be used as electrical equipment according to the Low Voltage Directive 2006/95/EC.

1.3 General description of the board

Data exchange between the **APCI-/CPCI-1500** board and the peripheral is to occur through a shielded cable. This cable must be connected to the 37-pin SUB-D male connector of the **APCI-/CPCI-1500** board

The board has 16 input channels and 16 output channels for processing digital 24 V signals. An external 24 V supply voltage is necessary to run the output channels. The screw terminal panel **PX901-D** and the relay board **PX8500** allow connecting the 24 V supply voltage through a shielded cable

The connection with our standard cable **ST010** complies with the following specifications:

- metallized plastic hoods
- shielded cable
- cable shield folded back and firmly screwed to the connector housing.

The use of the board according to its intended purpose includes observing all advises given in this manual and in the safety leaflet. Uses beyond these specifications are not allowed. The manufacturer is not liable for any damages which would result from the non-observance of this clause.

Make sure that the board remains in its protective blister pack **until it is used**.

Do not remove or alter the identification numbers of the board.
If you do, the guarantee expires.

2 USER

2.1 Qualification

Only persons trained in electronics are entitled to perform the following works:

- installation
- use,
- maintenance.

2.2 Country-specific regulations

Consider the country-specific regulations about:

- the prevention of accidents
- electrical and mechanical installations
- radio interference suppression.

3 HANDLING OF THE BOARD

Fig. 3-1: Correct handling of the CPCI-1500

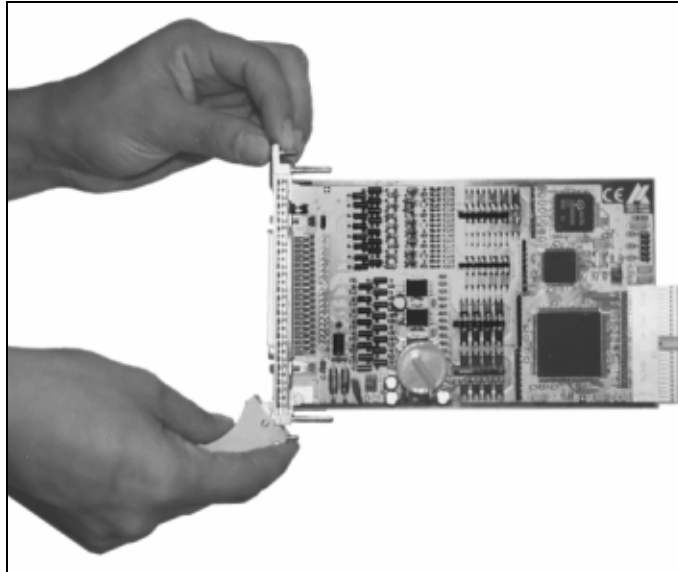
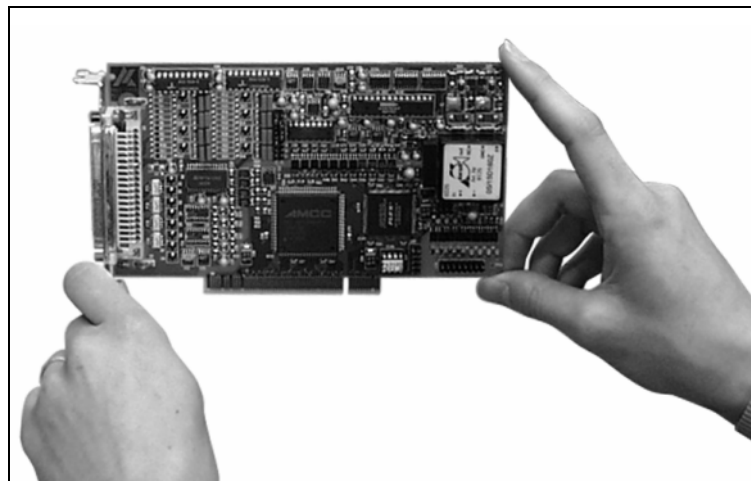


Fig. 3-2: Correct handling of the APCI-1500



4 TECHNICAL DATA

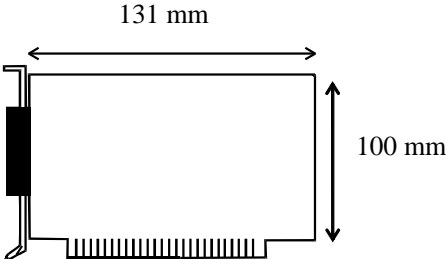
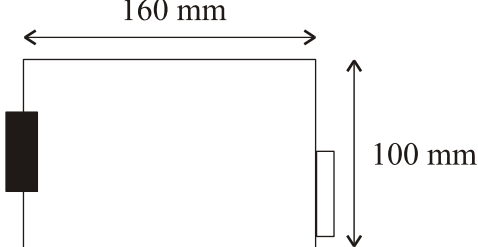
4.1 Electromagnetic compatibility (EMC)

The board **APCI-/CPCI-1500** complies with the European EMC directive. The tests were carried out by a certified EMC laboratory in accordance with the norm from the EN 61326 series (IEC 61326). The limit values as set out by the European EMC directive for an industrial environment are complied with.

The respective EMC test report is available on request.

4.2 Physical set-up of the board

The board is assembled on a 4-layer printed circuit card.

	APCI-1500	CPCI-1500
Dimensions		
Weight	160 g	200 g
Installation	PCI-5V (32-bit) slot or PCI-5V (64-bit) slot	CompactPCI-5V (32-bit) slot or CompactPCI-5V (64-bit) slot
Connection to the peripheral	37-pin SUB-D male connector	37-pin SUB-D male connector



WARNING!

The supply lines must be installed safely against mechanical loads.

4.3 Options

The board **CPCI-1500** is available in 2 versions:

- **CPCI-1500-3U:** 3U front plane
- **CPCI-1500-6U:** 6U front plane

4.4 Limit values

Max. altitude: 2000 m above NN

Operating temperature:..... 0 to 60°C

Storage temperature: -25 to + 70°C

Relative humidity at indoor installation

50% at +40 °C

80% at +31 °C

Minimum PC requirements (APCI-1500):

- PCI BIOS
- operating system: MS DOS 3.3 or >
Windows 3.1, NT, 95, 98
- bus speed: < 33 MHz

Minimum system requirements (CPCI-1500):

- 32-bit CompactPCI bus (5 V)
- PCI BIOS, PCI 2.1 specification and complying with **CompactPCI** 2.1
- operating system MS DOS 3.3 or >
Windows 3.1, NT, 95, 98
- bus speed..... ≤ 33 MHz

Energy requirements

- operating voltage of the PC: 5 V ± 5%
- current consumption in mA (without load): typ. See table ± 10%

	APCI-1500	CPCI-1500
+ 5 V of the PC	400 mA	220 mA
+ 24 V extern	-	10 mA

24 V digital input channels

Input type: common ground according to IEC1131-2

Number of input channels: 16

Nominal voltage: 24 VDC

Input current at nominal voltage: 6 mA

Logic input level: U_H ¹⁾ max.: 30 V, 9 mA.

U_H min.: 19 V, 3.3 mA typ.

¹ U_H : input voltage (= logic "1")

$U_L^{2)}$ max.: 14 V, 0.7 mA typ.

U_L min.: 0 V, 0 mA typ.

Signal delay: 70 μ s, (at nominal voltage)

Maximum input frequency: 5 kHz (at nominal voltage)

24 V digital output channels

Output type: high side (load at ground)

Number of output channels: 16

Nominal voltage: 24 VDC

Range of the supply voltage: 10 V to 36 VDC
(over 24 V ext. pins)

Max. output current for the

16 output channels: 3 A typ. (fused through
PTC resistors)

Max. output current / output channel: 500 mA

Short-circuit current / output channel at 24 V,

APCI-1500: $R_{load} < 0,1 R$: 1,5 A max. (switches off the
output channel)

CPCI-1500: $R_{load} < 0,01 R$: 2,5 A max. (switches off the
output channel)

ON-resistor of the output channel

(R_{DS} ON resistor): 0,4 R max.

Overtemperature: 170°C (switches off the component
i.e. the 4 output channels)

Temperature hysteresis: 20°C

Switch ON time at 24 V, R_{load} 500 mA: ... 100 μ s typ.

Switch OFF time at 24 V, R_{load} 500 mA: . 60 μ s typ.

Interruptible diagnostics, read back through status bit

\mathcal{G} -diagnostic: Pin 19 (24V/10 mA) is
switched on in case of overload
of the outputs or overtemperature

V_{cc} -diagnostic: is switched on in case of voltage
drop < 5 V

² U_L : input voltage (= logic "0")

Safety

Optical isolation

(DIN VDE 0411-100): 1000 V (from the PC to the external peripheral).

Logic: positive

Shut down logic: See *g*-diagnosticWatchdog: resets all the output channels, if no software trigger has happened. Times from 10 μ s to 37 s are available.

Counter input channels: max. 10 kHz, 24 V

APCI-1500

Option: fast counter input channels: max. 140 kHz, 24 V

CPCI-1500**Status LEDs on front connector:**

Green LED on: ext. voltage supply > 8 V

Red LED on: error signal from the outputs

4.5 Component scheme

Fig. 4-1: Component scheme of the APCI-1500

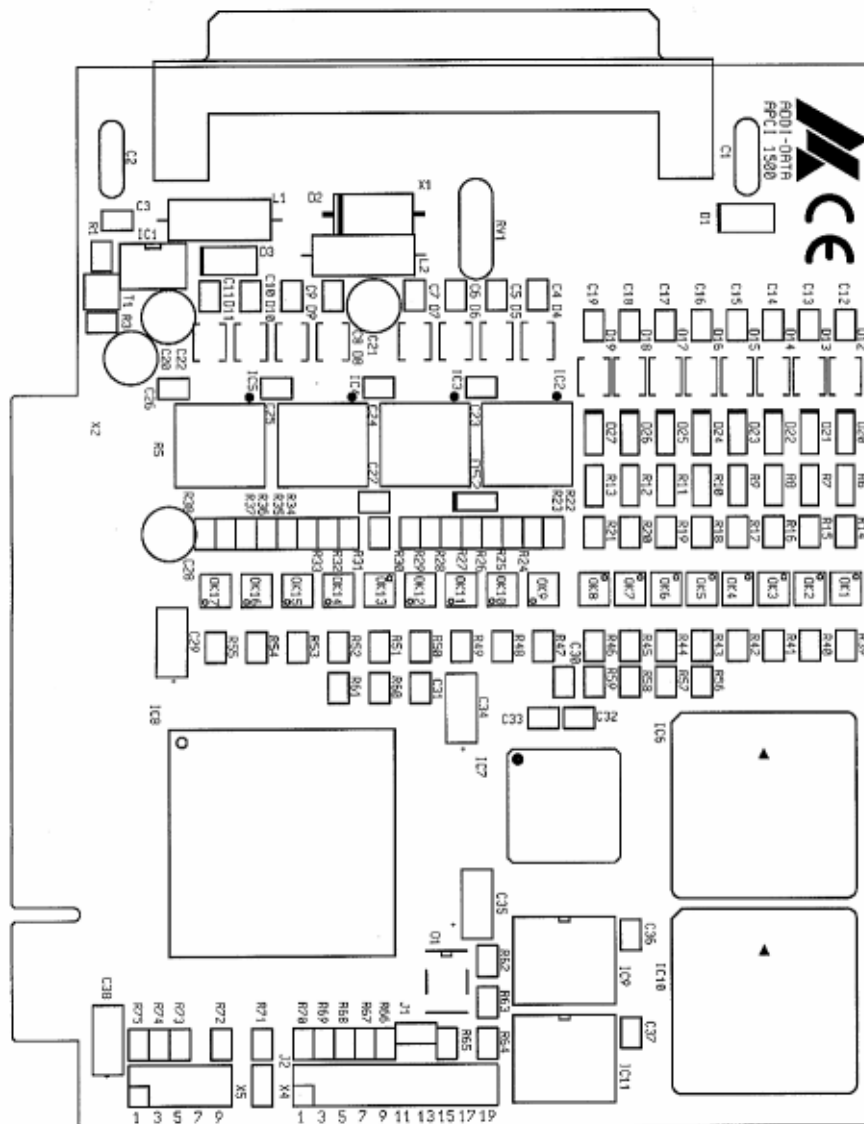
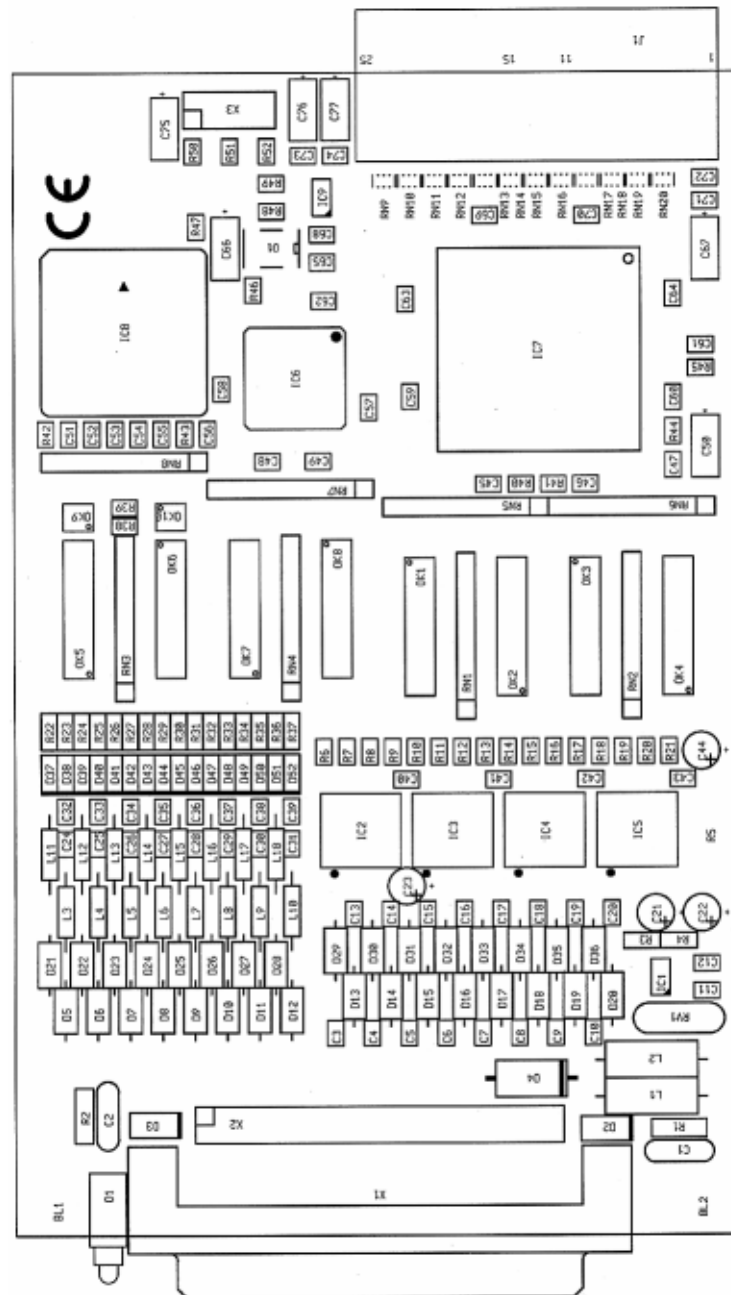


Fig. 4-2: Component scheme of the CPCI-1500



5 INSTALLATION OF THE BOARD



IMPORTANT!

Do observe the safety precautions (yellow leaflet)!

5.1 Installing the APCI-1500

5.1.1 Opening the PC

- ◆ Switch off your PC and all the units connected to the PC
- ◆ Pull the PC mains plug from the socket.
- ◆ Open your PC as described in the manual of the PC manufacturer.

5.1.2 Selecting a free slot

Insert the board in a free PCI-5V slot (32-bit).

Fig. 5-1: PCI-5V slot (32-bit)



32 bits

Remove the back cover of the selected slot according to the instructions of the PC manufacturer. Keep the back cover. You will need it if you remove the board

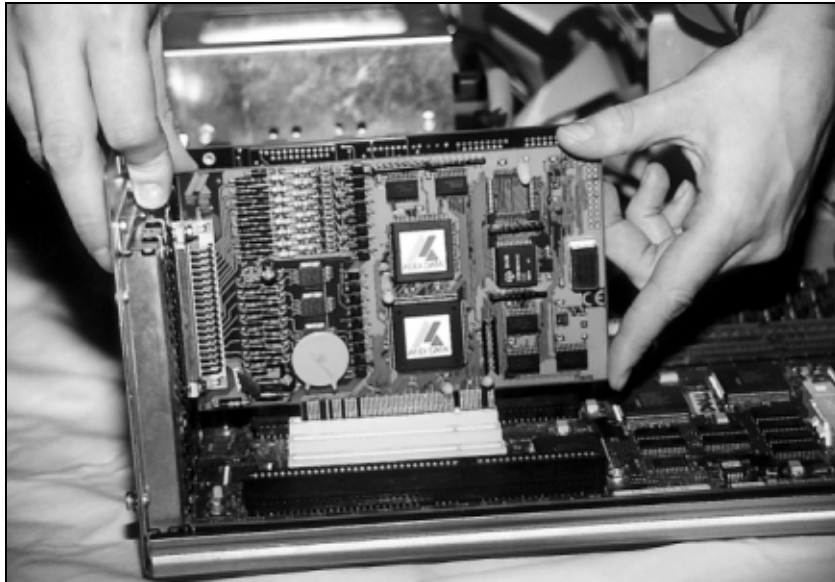
Discharge yourself from electrostatic charges.

Take the board out of its protective pack.

5.1.3 Plugging the board into the slot

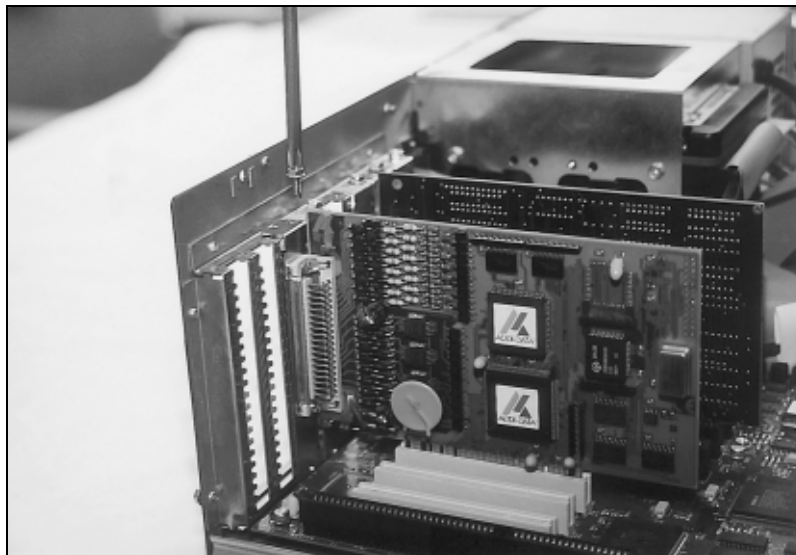
- ◆ Insert the board **vertically** into the chosen slot.

Fig. 5-2: Inserting the board



- ◆ Fasten the board to the rear of the PC housing with the screw which was fixed on the back cover.

Fig. 5-3: Fastening the board at the back cover



- ◆ Tighten all the loosen screws.

5.1.4 Closing the PC

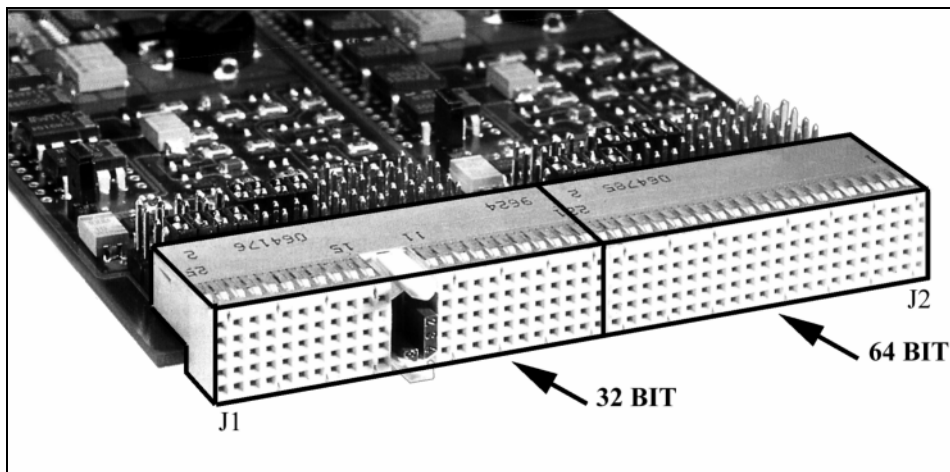
- ◆ Close your PC as described in the manual of the PC manufacturer.

5.2 Installing the CPCI-1500

The following **Compact PCI** slot types are available for 5V systems:
CPCI-5V (32-bit) and **CPCI-5V** (64-bit)

See in the computer manual which types of slots are free.

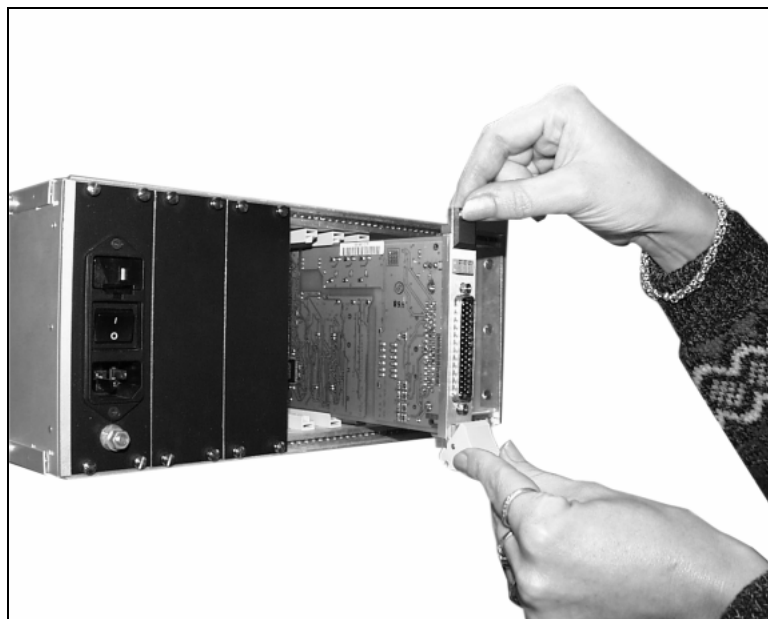
Fig. 5-4: Types of slots for CompactPCI boards



- ◆ Discharge yourself from electrostatic charges as described in the leaflet "Safety precautions".
- ◆ Hold the board at its grip (See board handling in chapter 3).
- ◆ Insert the board into the guiding rails and push it to the back cover of the rack.

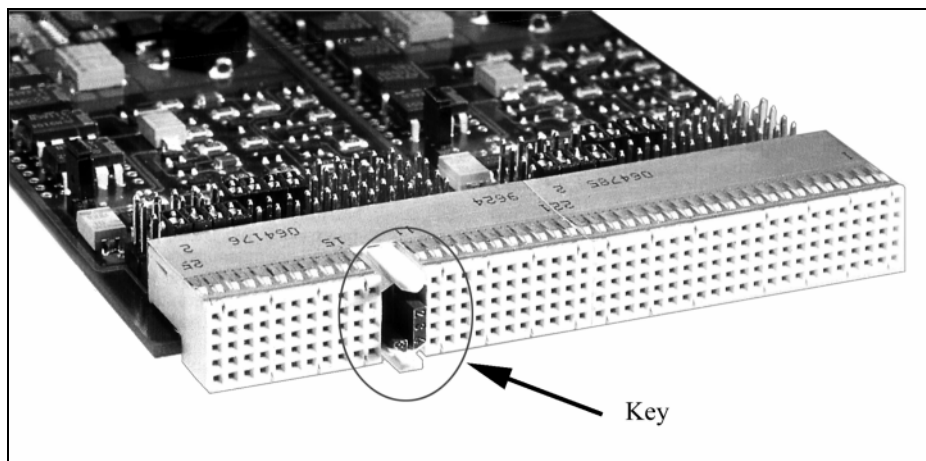
In order to fully insert the board, a small resistance has to be overcome.

Fig. 5-5: Pushing a CPCI board into a rack



- ◆ Make sure that the board is correctly connected by connecting the key of the board to the key of the backplane (blue key connectors if the board operates with 5 V)

Fig. 5-6: Connector keying



- ◆ If there is a screw at the upper part of the front plate, use this screw to fasten the board.

Note:

In order to pull the board out of the rack, pull it to the front at its grip. In some cases the grip has to be tilted upwards first.

6 SOFTWARE

In this chapter you will find a description of the delivered software and its possible applications.

i

IMPORTANT!

Further information for installing and uninstalling the different drivers is to be found in the delivered description "**Installation instructions for the PCI and ISA bus**".

A link to the corresponding PDF file is available in the navigation pane (Bookmarks) of Acrobat Reader.

The board is supplied with a CD-ROM (CD1) containing:

- the driver and software samples for Windows NT 4.0 and Windows XP/2000/98,
- the ADDIREG registration program for Windows NT 4.0 and Windows XP/2000/98.

6.1 Board registration with ADDIREG

The ADDIREG registration program is a 32-bit program for Windows NT 4.0 and Windows XP/2000/98.

The user can register all hardware information necessary to operate the ADDI-DATA PC boards.



IMPORTANT!

If you use one or several resources of the board, you cannot start the ADDIREG program.

6.1.1 Program description



IMPORTANT!

Insert the ADDI-DATA boards to be registered before starting the ADDIREG program.

If the board is not inserted, the user cannot test the registration.

Once the program is called up, the following dialog box appears.

Fig. 6-1: ADDIREG registration program (example)

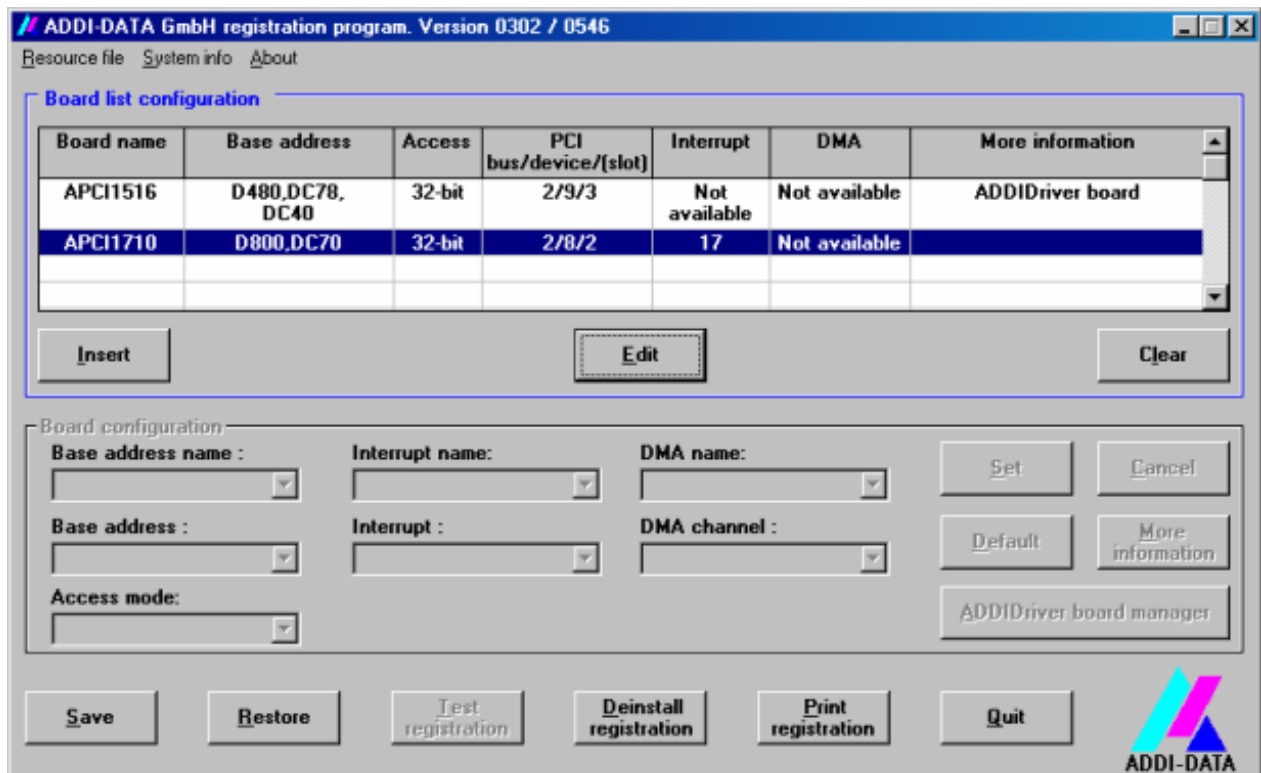


Table:**Board name:**

Names of the different registered boards (e.g.: APCI-3200).

Base address:

Selected base address of the board. For PCI boards the base address is allocated through BIOS.

Access:

Selection of the access mode for the ADDI-DATA digital boards.
Access in 8-bit or 16-bit or 32-bit mode.

PCI bus/device/(slot):

Number of the used PCI bus, slot, and device. If the board is no PCI board, the message "NO" is displayed.

Interrupt:

Used interrupt of the board. If the board supports no interrupt, the message "Not available" is displayed. **For PCI boards the interrupt is allocated through BIOS.**

DMA (ISA boards only):

Indicates the selected DMA channel or "Not available" if the board uses no DMA or if the board is no ISA board.

More information:

Additional information like the identifier string or the installed COM interfaces. It also displays whether the board is programmed with ADDIDRIVER or if a **PCI DMA** memory is allocated to the board.

Text boxes:**Base address name:**

Description of the used base addresses for the board. Select a name through the pull-down menu. The corresponding address range is displayed in the field below (Base address).

Interrupt name:

Description of the used IRQ lines for the board. Select a name through the pull-down menu. The corresponding interrupt line is displayed in the field below (Interrupt).

DMA name (for ISA boards only):

When the board supports 2 DMA channels, you can select which DMA channel is to be changed.

DMA channel (for ISA boards only):

Selection of the used DMA channel.

Buttons:

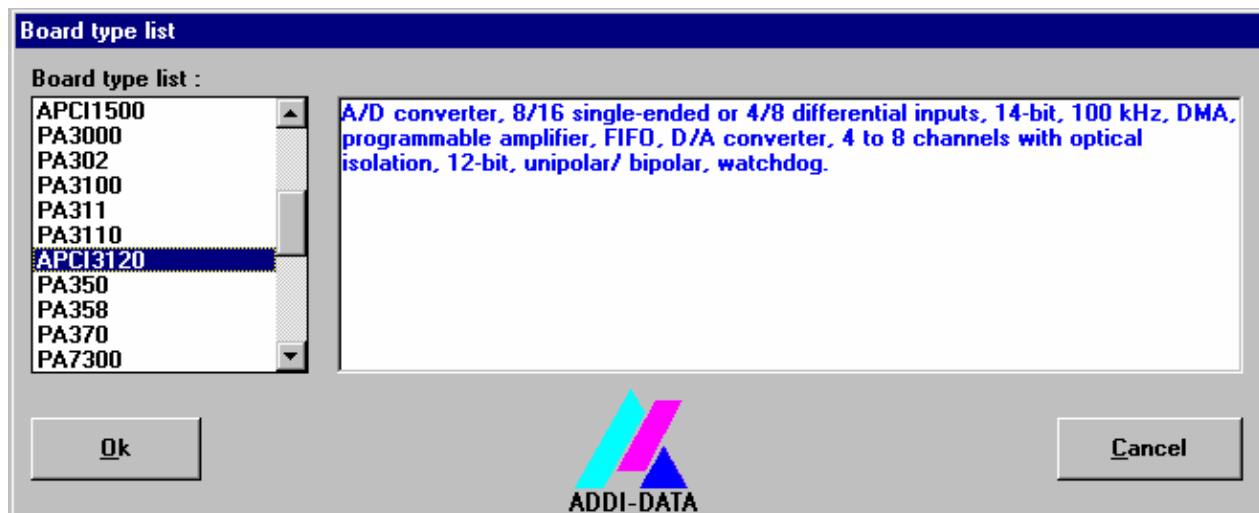
Edit:

Selection of the highlighted board with the different parameters set in the text boxes.

Insert:

When you want to insert a new board, click on "Insert". The following dialog window appears:

Fig. 6-2: Configuring a new board



All boards you can register are listed on the left. Select the wished board. (The corresponding line is highlighted).

On the right you can read technical information about the board(s).

Activate with "OK"; You come back to the former screen.

Clear:

You can delete the registration of a board. Select the board to be deleted and click on "Clear".

Set:

Sets the parametered board configuration. The configuration should be set before you save it.

Cancel:

Reactivates the former parameters of the saved configuration.

Default:

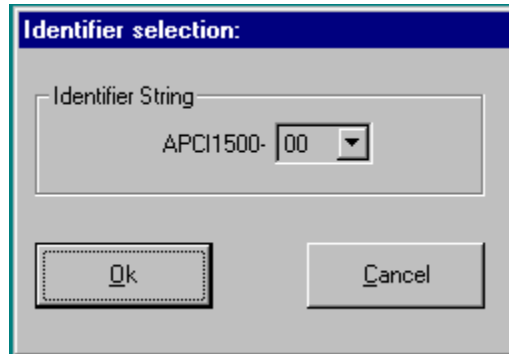
Sets the standard parameters of the board.

More information (not available for the boards with ADDIPACK)

You can change the board specific parameters like the identifier string, the COM number, the operating mode of a communication board, etc...

If your board does not support these information, you cannot activate this button.

Fig. 6-3: PCI Boards



With this option you can select the identifier string by entering the corresponding number and activating with "OK".

With one click on "cancel" you can select the former string.

Save:

Saves the parameters and registers the board.

Restore:

Reactivates the last saved parameters and registration.

ADDIDriver Board Manager (only for boards with ADDIPACK):

Under Edit/ADDIDriver Board Manager you can check or change the current settings of the board set through the ADDEVICE Manager.

ADDevice Manager starts and displays a list of all resources available for the virtual board.

Test registration:

Controls if there is a conflict between the board and other devices installed in the PC. A message indicates the parameter which has generated the conflict. If no conflict has occurred, "Test of device registration OK" is displayed.

Deinstall registration:

Deinstalls the registrations of all boards listed in the table and deletes the entries of the boards in the Windows Registry.

Print registration:

Prints the registration parameter on your standard printer.

Quit:

6.1.2 Registering a new board



IMPORTANT!

To register a new board, you must have administrator rights. Only an administrator is allowed to register a new board or change a registration.

- ◆ **Call up the ADDIREG program.**

The Fig. 6-1 is displayed on the screen.

- ◆ **Click on "Insert". Select the wished board.**
- ◆ **Click on "OK".**

The default address, interrupt, and the other parameters are automatically set in the lower fields. The parameters are listed in the lower fields.

If the parameters are not automatically set by the BIOS, you can change them.

- ◆ **Click on the wished scroll function(s) and choose a new value.**
- ◆ **Activate your selection with a click.**
- ◆ **Once the wished configuration is set, click on "Set".**
- ◆ **Save the configuration with "Save".**

You can test if the registration is "OK".

This test controls if the registration is right and if the board is present.

If the test has been successfully completed you can quit the ADDIREG program.

The board is initialised with the set parameters and can now be operated.

In case the registration data is to be modified, it is necessary to boot your PC again. A message asks you to do so. When it is not necessary you can quit the ADDIREG program and directly begin with your application.

6.1.3 Changing the registration of a board

- ◆ **Call up the ADDIREG program and select the board to be changed.**

The board parameters (Base address, DMA channel, ..) are listed in the lower fields.

- ◆ **Click on the parameter(s) you want to set and open the scroll function(s).**
- ◆ **Select a new value. Activate it with a click.**
- ◆ **Repeat the operation for each parameter to be modified.**
- ◆ **Once the wished configuration is set, click on "Set".**
- ◆ **Save the configuration with "Save".**

Under "Test registration" you can test if the registration is "OK". This test controls if the registration is right and if the board is present. If the test has been successfully completed you can quit the ADDIREG program. The board is initialised with the set parameters and can now be operated.

In case the registration data is to be modified, it is necessary to boot your PC again. A message asks you to do so. When it is not necessary you can quit the ADDIREG program and directly begin with your application.

6.2 Questions and software downloads on the web

Do not hesitate to e-mail us your questions.

per e-mail: info@addi-data.de or
 hotline@addi-data.de

Free downloads of standard software

You can download the latest version of the software for the board **APCI-/CPCI-1500**.

<http://www.addi-data.com>

i

IMPORTANT!

Before using the board or in case of malfunction during operation, check if there is an update of the product (technical description, driver). The current version can be found on the internet or contact us directly.

7 CONNECTING THE PERIPHERAL

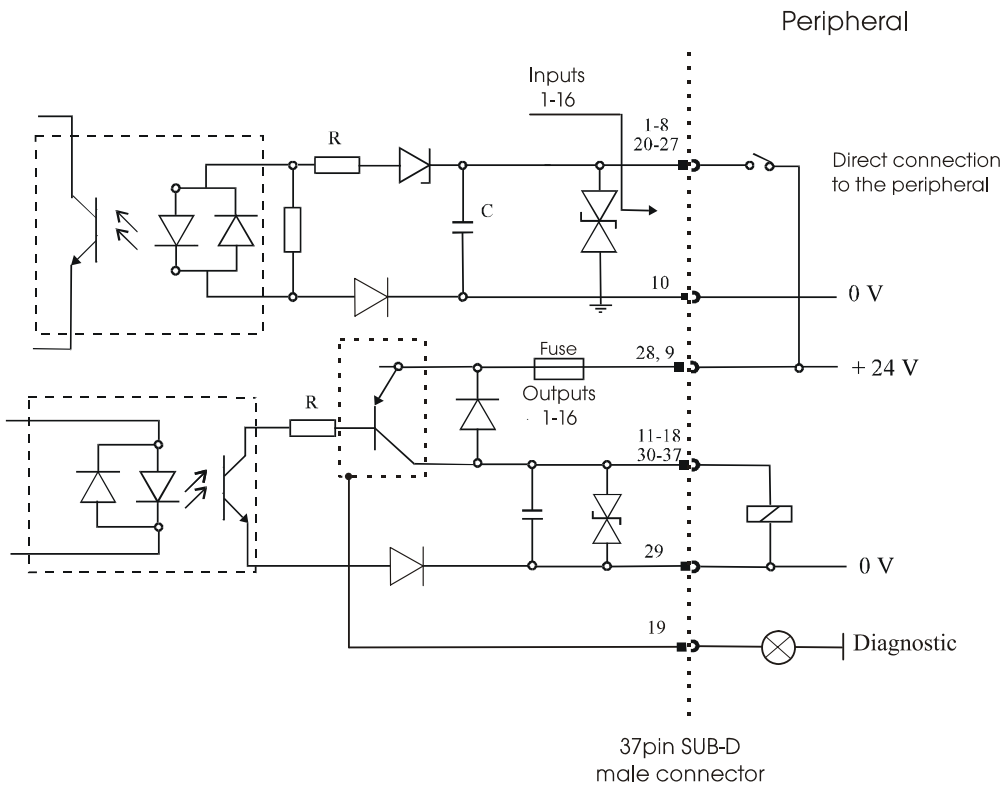
7.1 Connector pin assignment

Fig. 7-1: 37-pin SUB-D male connector

User designation					User designation
.....	Dig. input 2	20	1	Dig. input 1
.....	Dig. input 4	21	2	Dig. input 3
.....	Dig. input 6	22	3	Dig. input 5
.....	Dig. input 8	23	4	Dig. input 7
.....	Dig. input 10	24	5	Dig. input 9
.....	Dig. input 12	25	6	Dig. input 11
.....	Dig. input 14	26	7	Dig. input 13
.....	Dig. input 16	27	8	Dig. input 15
.....	24 V ext.	28	9	24 V ext.
.....	(Outputs) 0 V ext.	29	10	(Inputs) 0 V ext.
.....	Dig. output 2	30	11	Dig. output 1
.....	Dig. output 4	31	12	Dig. output 3
.....	Dig. output 6	32	13	Dig. output 5
.....	Dig. output 8	33	14	Dig. output 7
.....	Dig. output 10	34	15	Dig. output 9
.....	Dig. output 12	35	16	Dig. output 11
.....	Dig. output 14	36	17	Dig. output 13
.....	Dig. output 16	37	18	Dig. output 15
			19	9-Diagnostic

7.2 Connection principle

Fig. 7-2: Connection principle of the input and output channels



7.3 Connection examples

Fig. 7-3: Connection examples for the input and output channels

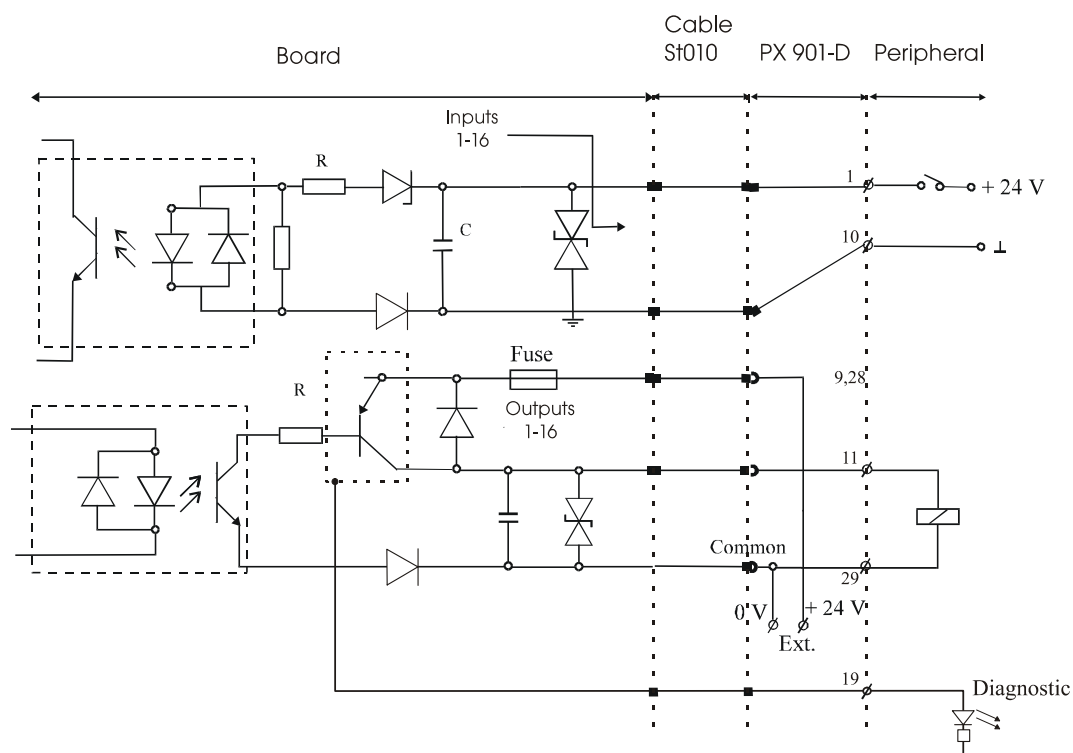
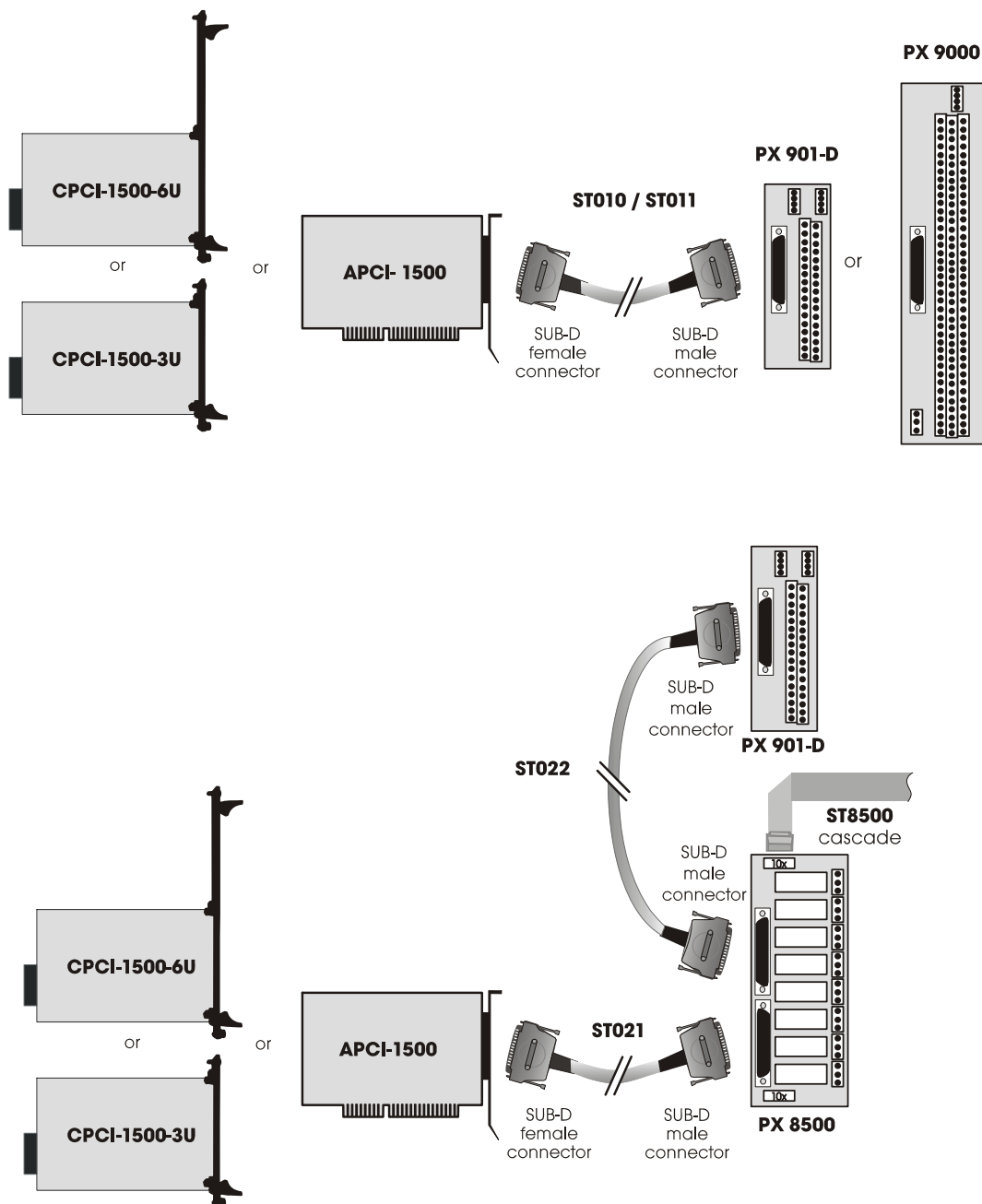


Fig. 7-4: Connection to the screw terminal panel



i

IMPORTANT!

Insert the FB on the connector with the red cable lead on the side of the pin 1.

8 FUNCTIONS OF THE BOARD

8.1 Description of the board

8.1.1 Block diagram

Fig. 8-1: Block diagram of the APCI-1500

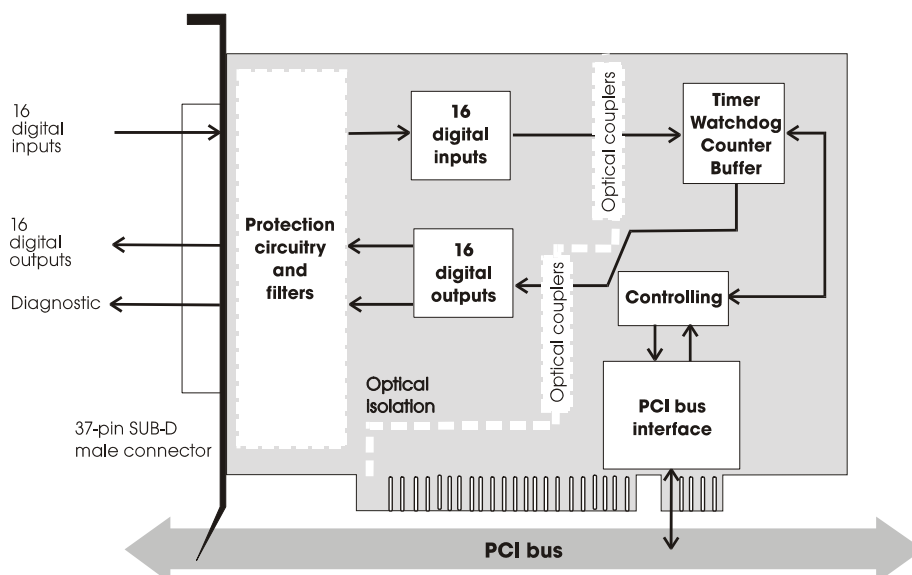
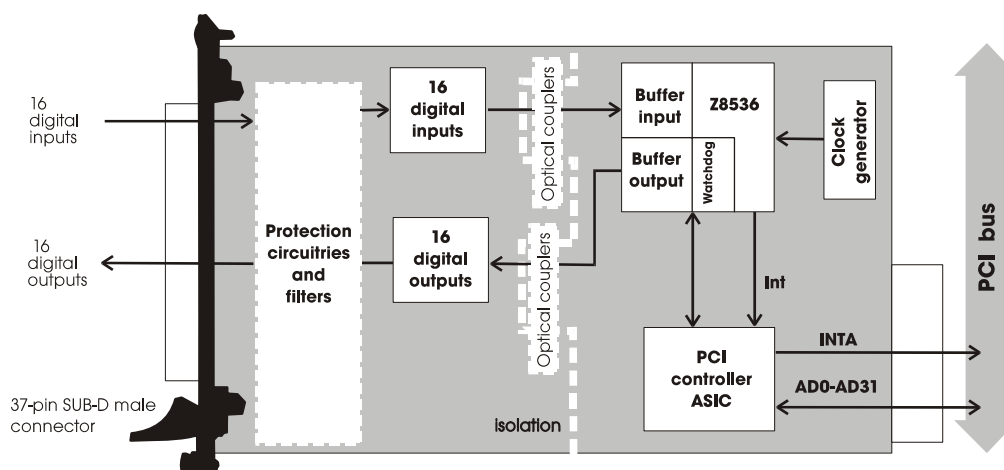


Fig. 8-2: Block diagram of the CPCI-1500



The board **APCI-/CPCI-1500** is intended for parallel input/output for digital signals in 24 V industrial environment. The peripheral and the system have a simultaneous optical isolation.

The board offers:

- **16 digital inputs:** 14 are interruptible.
- **3 counters** (or timers): programmable by software
- **1 timer:** can be used as a **watchdog** for the outputs.
- 16 digital outputs:
 - short-circuit current,
 - protection against overtemperature,
 - small ON-resistor,
 - wide supply voltage range,
 - the outputs are switched off if the voltage drops below the limit value (5 V).

The base address and the interrupt lines are automatically set through the BIOS.

EMC: design in accordance with CE regulations.

8.2 Functions

8.2.1 Digital inputs



WARNING!

Do not operate the board **simultaneously in several modes**. Otherwise you may damage the board, PC and/or the peripheral.

The board **APCI-/CPCI-1500** supplies 16 optically isolated inputs.

The inputs comply with the 24 V industry standard (IEC1131-2):

- logic "1" corresponds to an input voltage > 19 V
- logic "0" corresponds to an input voltage < 14 V.

All the inputs have a common current ground:

0V Ext. (inputs), pin 10 of the 37-pin SUB-D male connector.

The current input is at 6 mA with a nominal voltage of 24 V.

**WARNING!**

Do you operate all inputs with the same voltage supply? The voltage supply must deliver at least $16 \times 6 = 96 \text{ mA}$.

The maximum input voltage is 30 V.

Transil diodes, Z diodes, C filters and optical couplers protect the system bus from noise emitted by the peripheral. The effects of inductive and capacitive noise are thus reduced.

The board requires no initialisation to read the 24 V digital information. After successful power ON reset, data is immediately available on the board.

Special input functions of the digital inputs

1) Interrupt logic of the digital inputs 1-14

The inputs 1 to 14 can generate an interrupt. With a logic AND- or OR-functions of the inputs, the condition (=event) is defined in order to generate an interrupt.

At the AND-logic the conditions (=events) for generating an interrupt must be available at the same time.

The AND-function is limited to the inputs 1-8 (port 1).



Should you use events, do observe that the AND-logic can not be used on Port 2.

At the OR-logic one event is sufficient for generating an interrupt.

The OR-logic can be used on all inputs (port 1 and port 2). Port 2 has only 6 inputs.

The definition for the event, which can generate an interrupt, is defined through the event mask. See software function `[i_APCI1500_SetInputEventMask (...)]`. Digital inputs, which shall not influence the interrupt generation, will be masked out.

At the AND-logic a low-level (typ. 0 V), a high-level (typ. 24 V), a falling edge (signal switching from 24 V to 0 V) and/or a rising edge (signal switching from 0 V to 24 V) can be defined. Max. one edge switching can be defined.

- „X“: The input is not used for the event
- „0“: The input must be on „0“ (typ. 0 V)
- „1“: The input must be on „1“ (typ. 24 V)
- „2“: The input reacts to a falling edge 
(signal switching from 24 V to 0 V)
- „3“: The input reacts to a rising edge 
(signal switching from 0 V to 24 V)
- „4“: The input reacts to both edges

Example: Port 1 (digital inputs 1-8), AND-logic

Event for generating an interrupt:

Digital input	8 7 6 5 4 3 2 1
Event mask	3 X 1 0 0 1 0 1

The following events generate an interrupt:

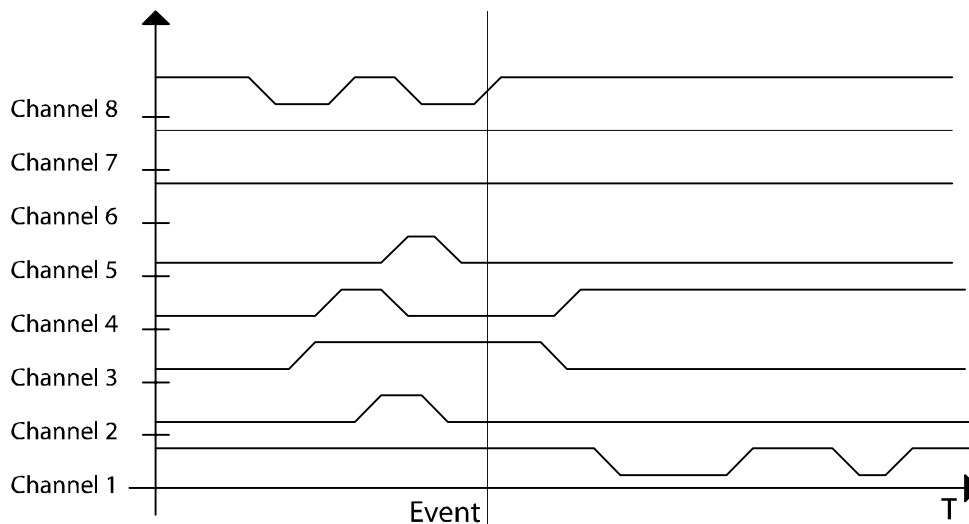
Inputs 2 and 4 and 5 = 0 V

Inputs 1 and 3 and 6 = 24 V

Input 8 = rising edge

Input 7: No meaning

Fig. 8-3: Interrupt logic – Digital inputs 1-8

**Example: Port 1 (digital inputs 1-8), OR-logic**

Event for generating an interrupt:

Digital input	8 7 6 5 4 3 2 1
Event mask	X X X X 0 0 1 1

The inputs 5-8 have no meaning.

An interrupt will be generated if a voltage of 24 V will be connected to input 1 or input 2.

Please observe: The OR-event is fulfilled as long as at least one of the two inputs is on 24 V.

The meaning in this case is:

- If input 1 switches to 24 V and input 2 stays on 0 V, an interrupt will be generated
- If input 2 switches from 0 V to 24 V, while input 1 stays on 24 V, no

interrupt will be generated because the OR-event is not fulfilled

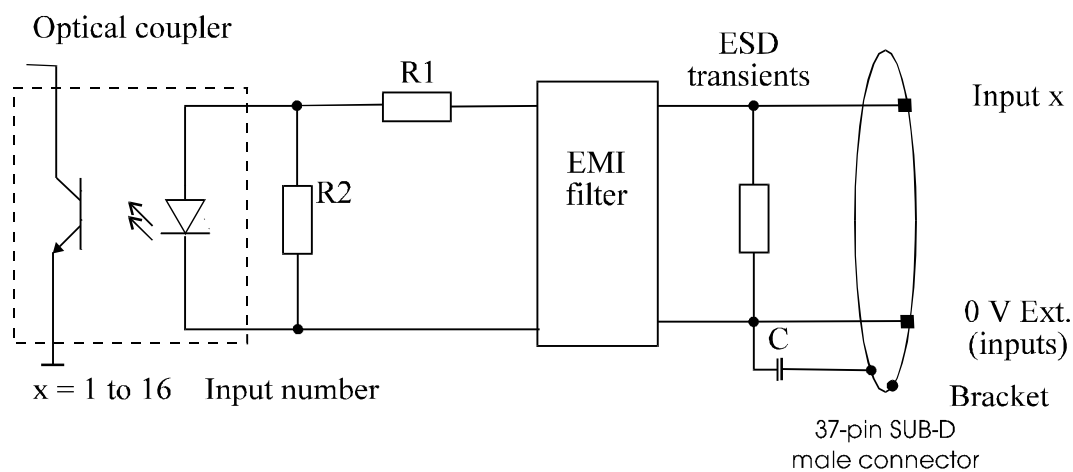
2) Counter

Counter 1: **Input 14** signal input

Counter 2: **Input 10** signal input
Input 11 can be used for a "trigger" function
Input 12 can be used for a "gate" function.

Counter 3 **Input 15** signal input
Input 16 can be used as a "gate" function

Fig. 8-4: Protection circuitry for the inputs



8.2.2 Digital outputs

The board supplies 16 optically isolated outputs.

The outputs comply with the 24 V industry standard (IEC1131-2):

The positive logic is used

- logic "1": sets the output by software (switch on ON),
- logic "0": resets the output (switch on OFF).

The outputs switch the **+24V ext.** outside to the load. One end of the load is connected with the ground of 0V EXT (outputs).

All outputs have a common ground: 0V ext. (outputs) pin 29 of the 37-pin SUB-D male connector.



WARNING!

If you use all outputs with the same voltage supply,
the voltage supply must deliver at least the power required for your application.

The maximum supply voltage is 36 V. Each output can switch 500 mA current. But the current is limited for all the outputs on approx. 3 A by a self-resetting fuse.

Features of the outputs:

- Short-circuit current for the 16 outputs
- Protection against overtemperature: shut down logic. Each group of 4 outputs is switched off: 1 to 4, 5 to 8, 9 to 12, 13 to 16.
- The outputs are switched off if the ext. supply voltage drops below 5 V.
- Diagnostic report through status register or interrupt to the PC in case of short-circuit, overtemperature, voltage drop or watchdog.

Transil diodes, C filters and optical couplers filter noise from the peripheral to the system bus. Thus the effects of inductive and capacitive noise are reduced. Possible noise emissions are also reduced by C filters.

The board requires no initialisation to output the 24V digital information. You can program the outputs immediately after successful power ON reset

State after power ON reset : all the outputs are reset (switch on OFF).

Special functions

2 diagnostic bits are available on the board.

The \mathcal{G} -diagnostic (Pin 19 on the front connector) is released:

- when short-circuit has occurred on an output or
- in case of overtemperature on an output component (8 channels).

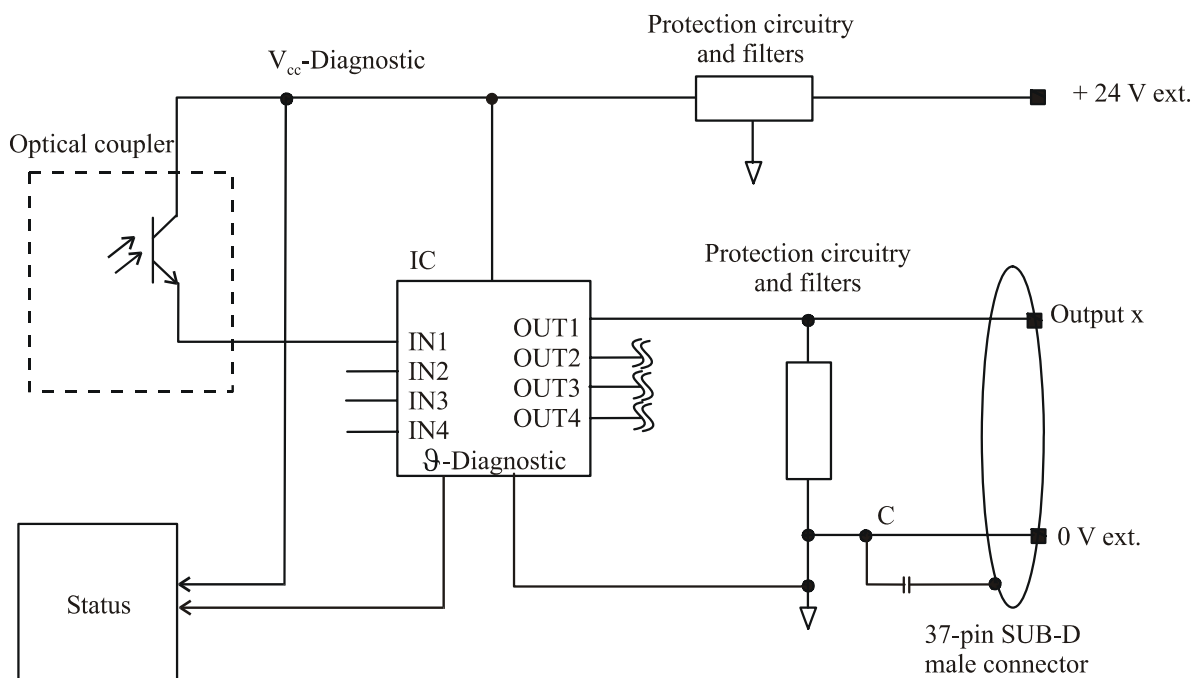
The V_{cc} -diagnostic informs that:

- the external voltage supply has dropped < 5 V.

These error data are available through an interrupt routine.

See API functions: `i_APCI1500_SetBoardIntRoutineXX`,
`i_APCI1500_ResetBoardIntRoutine`.

Fig. 8-5: Protection circuitry for the outputs



x = 1 to 16 Number of the output

8.2.3 Interrupt

The board has an interrupt line. An interrupt line of the PCI bus is allocated to the board through the BIOS.

Possible interrupt sources:

- Event 1 has occurred (input 1-8),
- Event 2 has occurred (input 9-14),
- Counter/Timer 1 has run down
- Counter/Timer 2 has run down
- Counter/Timer 3 has run down
- Watchdog has run down, the outputs are reset,
- Voltage error (the external voltage supply has dropped below 5 V),
- Short-circuit error, overtemperature error.

The interrupt source information are available on the user program through an interrupt routine

See API functions: `i_APCI1500_SetBoardIntRoutineXX`,
`i_APCI1500_ResetBoardIntRoutine`.

An **event** indicates a change of status (level):

- on an input (ex. "0" → "1")
- or on several inputs if a logic has been defined between the inputs.

Running-down: if the counter changes from 1 → 0.

8.2.4 Counter/timer

On the board three 16-bit counters/timers are available in the component Z8536 (downwards counting). Each counter/timer can be programmed by software.

If the component Z8536 operates as a counter, the corresponding inputs are used as follows:

Counter

- Counter 1: **Input 14** signal input.
- Counter 2: **Input 10** signal input
Input 11 can be used as a "trigger" function
Input 12 can be used as a "gate" function.
- Counter 3: **Input 15** signal input
Input 16 can be used as a "gate" function.

Timer

If the component Z8536 is used as a timer, the frequency is used as a reference. "Gate" and "trigger" are possible through the inputs.

Gate: can be driven by software or an input can be set. The polarity of the input can be programmed. This "gate" stops counting when it is set.

Trigger : can be driven by software or an input can be set. The polarity of the input can be programmed. This "trigger" re-loads the counter/timer with the initial counter value.

The following functionalities are available:

- Initialising the counters/timers,
- Starting the counters/timers,
- Stopping the counters/timers,
- Reading the counter value of the counters/timers

The counter/timer 3 has a special function: **Watchdog Timer**. The function **Watchdog Timer** allows to supervise the software or PC.

The principle is:

The counter/timer 3 is programmed as a non reloadable timer.

The timer is started. The outputs are reset when the timer has run down (switch OFF).

The user software must be built in such a way that the output channels are always set to "ON" again. You thus avoid that the watchdog runs down.

Input frequencies

The input frequency for the timer is selected through the software function `i_APCI1500_InitTimerInputClock (...)`

Available frequencies:

111.86 kHz \pm 100 ppm,

3.49 kHz \pm 100 ppm,

1.747 kHz \pm 100 ppm.



IMPORTANT!

The timer component internally operates with half of the input frequency.

Data

Approximate watchdog times:	Input frequency
17.9 μ s \rightarrow 1175 ms	111.86 kHz
574 μ s \rightarrow 37.65 s	3.49 kHz
1.14 ms \rightarrow 74.9 s	1.747 kHz

Option (only implemented for the APCI-1500)

The counter input channels 1, 2 and 3 can be equipped with fast optical couplers.

The maximum input frequency is then 140 kHz. (Standard frequency: 10 kHz)

9 STANDARD SOFTWARE

9.1 Introduction

Table 9-1: Type Declaration for Dos and Windows 3.1X

	Borland C	Microsoft C	Borland Pascal	Microsoft Visual Basic Dos	Microsoft Visual Basic Windows
VOID	void	void	pointer		any
BYTE	unsigned char	unsigned char	byte	integer	integer
INT	int	int	integer	integer	integer
UINT	unsigned int	unsigned int	word	long	long
LONG	long	long	longint	long	long
PBYTE	unsigned char *	unsigned char *	var byte	integer	integer
PINT	int *	int *	var integer	integer	integer
PUINT	unsigned int *	unsigned int *	var word	long	long
PCHAR	char *	char *	var string	string	string

Table 9-2: Type Declaration for Windows 95/98/NT

	Borland C	Microsoft C	Borland Pascal	Microsoft Visual Basic Dos	Microsoft Visual Basic Windows
VOID	void	void	pointer		any
BYTE	unsigned char	unsigned char	byte	integer	integer
INT	int	int	integer	integer	integer
UINT	unsigned int	unsigned int	long	long	long
LONG	long	long	longint	long	long
PBYTE	unsigned char *	unsigned char *	var byte	integer	integer
PINT	int *	int *	var integer	integer	integer
PUINT	unsigned int *	unsigned int *	var long	long	long
PCHAR	char *	char *	var string	string	string

9.2 Software functions

Remark: The **CPCI-1500** board is entirely compatible with the **APCI-1500** board regarding the software installation and the device driver. The program ADDIREG will thus make no difference between the systems (PCI board or *Compact PCI* board).

The API functions of the device driver are also the same.

9.2.1 Base address and interrupt

1) i_APCI1500_InitCompiler (..)

Syntax:

<Return value> = i_APCI1500_InitCompiler (BYTE b_CompilerDefine)

Parameters:

- Input:

BYTE b_CompilerDefine	The user has to choose the language under Windows in which he/she wants to program DLL_COMPILER_C: The user programs in C. DLL_COMPILER_VB: The user programs in Visual Basic for Windows. DLL_COMPILER_VB_5: The user programs in Visual Basic 5 for Windows NT or Windows 95/98. DLL_COMPILER_PASCAL: The user programs in Pascal or Delphi. DLL_LABVIEW: The user programs in Labview.
--------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

- Output:

No output signal has occurred.

Task:

If you want to use the DLL functions, parameter in the language which you want to program in. This function must be the first to be called up.



IMPORTANT!

This function is only available with a Windows environment.

Calling convention:

ANSI C :

```
int    i_ReturnValue;
      i_ReturnValue = i_APCI1500_InitCompiler (DLL_COMPILER_C);
```

Return value:

0: No error

-1: Compiler parameter is wrong

2) i_APCI1500_CheckAndGetPCISlotNumber (...)

Syntax:

<Return value> = i_APCI1500_CheckAndGetPCISlotNumber
(PBYTE pb_SlotNumberArray)

Parameters:

- Input:

No input signal has to occur.

- Output:

PBYTE pb_SlotNumberArray List of the slot numbers.

Task:

Checks all **xPCI-1500**¹ and returns the slot number of each **xPCI-1500** board. Each *pb_SlotNumberArray* parameter contains the slot number (1 to 10) of one **xPCI-1500** board.

Calling convention:

ANSI C:

```
int          i_ReturnValue;
unsigned char b_SlotNumberArray [10];
```

```
i_ReturnValue = i_APCI1500_CheckAndGetPCISlotNumber
                (b_SlotNumberArray);
```

Return value:

Returns the number of **xPCI-1500** boards which are inserted in your PC.
If the return value equals 0 then no **xPCI-1500** board was found in your PC.

¹ Common designation for the boards **APCI-1500** and **CPCI-1500**

3) i_APCI1500_SetBoardInformation (...)

Syntax:

<Return value> = i_APCI1500_SetBoardInformation
(BYTE b_SlotNumber,
PBYTE pb_BoardHandle)

Parameters:**- Input:**

BYTE b_SlotNumber: Slot number of the **xPCI-1500** board

- Output:

PBYTE pb_BoardHandle: Handle¹ of board xPCI-1500 for using the functions

Task:

Checks if the board **xPCI-1500** is present and stores the slot number.

A handle is returned to the user which allows to use the following functions.

Handles allow to operate several boards.

Return value:

0: No error

-1: Slot number is not available

-2: Board not present

-3: No handle is available for the board (up to 10 handles can be used)

-4: Error by opening the kernel driver under Windows NT/95/98

¹ Identification number of the board

4) i_APCI1500_GetHardwareInformation (...)

Syntax:

```
<Return value> = i_APCI1500_GetHardwareInformation
                    (BYTE  b_BoardHandle,
                     PUINT  pui_BaseAddress,
                     PBYTE  pb_InterruptNbr,
                     PBYTE  pb_SlotNumber)
```

Parameters:

- Input

BYTE b_BoardHandle Handle of board **xPCI-1500**

- Output:

PUINT pui_BaseAddress **xPCI-1500** base address
 PBYTE pb_InterruptNbr **xPCI-1500** interrupt channel.
 PBYTE pb_SlotNumber **xPCI-1500** slot number

Task:

Returns the base address, the interrupt and the slot number of the **xPCI-1500**.

Calling convention:

ANSI C:

```
int                    i_ReturnValue;
unsigned int          ui_BaseAddress;
unsigned char         b_InterruptNbr;
unsigned char         b_SlotNumber;
unsigned char         b_BoardHandle;
```

```
i_ReturnValue = i_APCI1500_GetHardwareInformation
                (b_BoardHandle,
                 &ui_BaseAddress,
                 &b_InterruptNbr,
                 &b_SlotNumber);
```

Return value:

0: No error

-1: The handle parameter of the board is wrong

5) i_APCI1500_CloseBoardHandle (..)



IMPORTANT!

Call up this function each time you want to leave the user program!

Syntax:

<Return value> = i_APCI1500_CloseBoardHandle (BYTE b_BoardHandle)

Parameters:

- Input:

BYTE b_BoardHandle Handle of board **xPCI-1500**

- Output:

No output signal has occurred.

Task:

Releases the board handle. Blocks the access to the board.

Calling convention:

ANSI C:

```
int                      i_ReturnValue;  
unsigned char            b_BoardHandle;
```

```
i_ReturnValue = i_APCI1500_CloseBoardHandle (b_BoardHandle);
```

Return value:

0: No error

-1: The handle parameter of the board is wrong

9.2.2 Interrupt



IMPORTANT!

This function is only available for C/C++ and Pascal for DOS.

1) i_APCI1500_SetBoardIntRoutineDos (..)

Syntax:

```
<Return value> = i_APCI1500_SetBoardIntRoutineDos
                                     (BYTE b_BoardHandle,
                                     VOID v_FunctionName
                                     (BYTEb_BoardHandle,
                                     BYTEb_InterruptMask
                                     BYTEb_InputChannelNbr))
```

Parameters:

- Input:

BYTE	b_BoardHandle	Handle of board xPCI-1500
VOID	v_FunctionName	Name of the user interrupt routine

- Output:

No output has occurred.

Task:

This function must be called up for each **xPCI-1500** on which an interrupt action is to be enabled. It installs an user interrupt function in all boards on which an interrupt is to be enabled.

First calling (first board):

- the user interrupt routine is installed
- interrupts are enabled.

If you operate several boards **xPCI-1500** which have to react to interrupts, call up the function as often as you operate boards **xPCI-1500**. The variable *v_FunctionName* is only relevant **for the first calling**.

From the second call of the function (next board):

- interrupts are enabled. The first board can receive IRQ.

Interrupt

The user interrupt routine is called up by the system when an interrupt is generated.

An interrupt is generated when:

- the counter/timer has run down
- an event is generated
- the watchdog has run down

The following errors are possible

- overtemperature
- short-circuit
- no voltage is available

If several boards are operated and if they have to react to interrupts, the variable *b_BoardHandle* returns the identification number (handle) of the board which has generated the interrupt.

The user interrupt routine must have the following Syntax:

```
VOID  v_FunctionName      (BYTE  b_BoardHandle,
                           BYTE  b_InterruptMask,
                           BYTE  b_InputChannelNbr)
```

v_FunctionName Name of the user interrupt routine

b_BoardHandle Handle of the **xPCI-1500** which has generated the interrupt

b_InterruptMask Mask of the events which have generated the interrupt.

b_InputChannelNbr If an interrupt is generated with a Mask 0000 0001 and if you use the OR-PRIORITY logic, this variable gives the input number, which have generated the interrupt.

Table 9-3: Interrupt mask

Mask	Meaning
0000 0001	Event 1 has occurred
0000 0010	Event 2 has occurred
0000 0100	Counter/timer 1 has run down
0000 1000	Counter/timer 2 has run down
0001 0000	Counter 3 has run down
0010 0000	Watchdog has run down
0100 0000	Voltage error
1000 0000	Short-circuit error

The user can give another name for *v_FunctionName*, *b_BoardHandle*, *b_InterruptMask*, *b_InputChannelNbr*.

Calling convention:ANSI C:

```
void    v_FunctionName    (unsigned char  b_BoardHandle,  
                           unsigned char  b_InterruptMask,  
                           unsigned int   b_InputChannelNumber)  
  
    {  
        .  
        .  
    }
```

```
int      i_ReturnValue;  
unsigned char b_BoardHandle;
```

```
i_ReturnValue = i_APCI1500_SetBoardIntRoutineDos  
                (b_BoardHandle,  
                 v_FunctionName );
```

Return value:

- 0: No error
- 1: The handle parameter of the board is wrong
- 2: Interrupt already installed

**IMPORTANT!**

This function is only available for Visual Basic DOS.

2) **i_APCI1500_SetBoardIntRoutineVBDos (..)**

Syntax:

```
<Return value> = i_APCI1500_SetBoardIntRoutineVBDos
                                     (BYTE    b_BoardHandle)
```

Parameters:**- Input:**

BYTE b_BoardHandle Handle of board **xPCI-1500**

- Output:

No output signal has occurred.

Task:

This function must be called up for each **xPCI-1500** on which an interrupt action is to be enabled. It installs an user interrupt function in all boards on which an interrupt is to be enabled.

First calling (first board):

- the user interrupt routine is installed
- interrupts are enabled.

If you operate several boards **xPCI-1500** which have to react to interrupts, call up the function as often as you operate boards **xPCI-1500**. The variable *v_FunctionName* is only relevant **for the first calling**.

Interrupt

The user interrupt routine is called up by the system when an interrupt is generated.

The following errors are possible

- overtemperature
- short-circuit
- no voltage is available

Controlling the interrupt management

Please use the following functions

"ON UEVENT GOSUB xxxxxxxxx" of Visual Basic for DOS

and

"i_APCI1500_TestInterrupt"

This function tests the interrupt of the **xPCI-1500**. It is used for obtaining the values of *b_BoardHandle* , *b_InterruptMask* and *b_InputChannelNbr*.

Calling convention:Visual Basic DOS:

```

Dim Shared i_ReturnValue      As Integer
Dim Shared i_BoardHandle      As Integer
Dim Shared i_InterruptMask    As Integer
Dim Shared l_InputChannelNbr  As Integer

```

IntLabel:

```

i_ReturnValue = i_APCI1500_TestInterrupt      (i_BoardHandle, _
                                                i_InterruptMask, _
                                                i_InputChannelNbr)

.
.
.

Return

ON UEVENT GOSUB IntLabel
UEVENT ON
i_ReturnValue = i_APCI1500_SetBoardIntRoutineVBDos(b_BoardHandle)

```

Return value:

0: No error
-1: The handle parameter of the board is wrong
-2: Interrupt already installed

**IMPORTANT!**

This function is only available for Windows 3.1 and Windows 3.11

3) i_APCI1500_SetBoardIntRoutineWin16 (..)**Syntax:**

```
<Return value> = i_APCI1500_SetBoardIntRoutineWin16
                (BYTE    b_BoardHandle,
                 VOID     v_FunctionName
                 (BYTE    b_BoardHandle,
                  BYTE    b_InterruptMask,
                  BYTE    InputChannelNbr))
```

Parameters:**- Input:**

BYTE	b_BoardHandle	Handle of board xPCI-1500
VOID	v_FunctionName	Name of the user interrupt routine

- Output:

No output signal has occurred.

Task:

This function must be called up for each **xPCI-1500** on which an interrupt action is to be enabled. It installs one user interrupt function in all boards on which an interrupt is to be enabled.

First calling (first board):

- the user interrupt routine is installed
- interrupts are enabled.

If you operate several boards **xPCI-1500** which have to react to interrupts, call up the function as often as you operate boards **xPCI-1500**. The variable *v_FunctionName* is only relevant **for the first calling**.

From the second call of the function (next board):

- interrupts are allowed.

Interrupt

The user interrupt routine is called up by the system when an interrupt is generated.

An interrupt is generated when:

- the counter/timer has run down
- an event is generated
- the watchdog has run down

The following errors are possible

- overtemperature
- short-circuit
- no voltage is available

If several boards are operated and if they have to react to interrupts, the variable *b_BoardHandle* returns the identification number (handle) of the board which has generated the interrupt.

The user interrupt routine must have the following Syntax:

```
VOID  v_FunctionName      (BYTE  b_BoardHandle,
                           BYTE  b_InterruptMask,
                           BYTE  b_InputChannelNbr)

v_FunctionName            Name of the user interrupt routine
b_BoardHandle              Handle of the xPCI-1500 which has generated
                           the interrupt
b_InterruptMask            Mask of the events which have generated the
                           interrupt.
b_InputChannelNbr          If an interrupt is generated with a Mask 0000
                           0001 and if you use the OR-PRIORITY logic,
                           this variable gives the input number, which
                           have generated the interrupt.
```

Mask	Meaning
0000 0001	Event 1 has occurred
0000 0010	Event 2 has occurred
0000 0100	Counter/timer 1 has run down
0000 1000	Counter/timer 2 has run down
0001 0000	Counter 3 has run down
0010 0000	Watchdog has run down
0100 0000	Voltage error
1000 0000	Short-circuit error

The user can give another name for *v_FunctionName*, *b_BoardHandle*, *b_InterruptMask*, and *b_InputChannelNbr*.

i**IMPORTANT!**

If you use Visual Basic for Windows the following parameters have no signification. You must use "i_APCI1500_TestInterrupt".

```
VOID  v_FunctionName  (BYTE      b_BoardHandle,
                       BYTE      b_InterruptMask,
                       BYTE      b_InputChannelNbr)
```

Calling convention:ANSI C:

```
void  v_FunctionName  (unsigned char  b_BoardHandle,
                       unsigned char  b_InterruptMask,
                       unsigned char  b_InputChannelNbr)
{
    .
    .
}
```

```
int      i_ReturnValue;
unsigned char b_BoardHandle;
```

```
i_ReturnValue = i_APCI1500_SetBoardIntRoutineWin16
                (b_BoardHandle,
                 v_FunctionName );
```

Return value:

- 0: No error
- 1: The handle parameter of the board is wrong
- 2: Interrupt already installed

4) i_APCI1500_SetBoardIntRoutineWin32 (..)

Syntax:

```
<Return value> = i_APCI1500_SetBoardIntRoutineWin32
                                (BYTE    b_BoardHandle,
                                BYTE    b_UserCallingMode,
                                ULONG    ul_UserSharedMemorySize,
                                VOID **  ppv_UserSharedMemory,
                                VOID     v_FunctionName
                                (BYTE    b_BoardHandle,
                                BYTE    b_InterruptMask,
                                BYTE    b_InputChannelNbr,
                                BYTE    b_UserCallingMode,
                                VOID *  pv_UserSharedMemory))
```

Parameters:

- Input:

BYTE	b_BoardHandle	Handle of board xPCI-1500
BYTE	b_UserCallingMode	APCI1500_SYNCHRONOUS_MODE: The user routine is directly called by the driver interrupt routine. APCI1500_ASYNCHRONOUS_MODE: The user routine is called by the driver interrupt thread.
VOID	v_FunctionName	Name of the user interrupt routine
ULONG	ul_UserSharedMemorySize	Determines the size in bytes of the user shared memory. Only used if you have selected APCI1500_SYNCHRONOUS_MODE

i

IMPORTANT!

The size of the User Shared Memory is limited to 63 MB. It could cause problems if more memory is required.

- Output:

VOID **	ppv_UserSharedMemory	User shared memory address. Only used if you have selected APCI1500_SYNCHRONOUS_MODE.
---------	----------------------	---------------------------------------------------------------------------------------

Task:**i****WINDOWS 32-BIT INFORMATION**

For Windows NT and Windows 2000/9x, 4 running rings (ring 0 to ring 3) are available

- The user application operates in ring 3. This ring gives no access to hardware.
- VXD and SYS driver operate in ring 0 and give access to hardware.
- Ring 0 has no direct access to global variable from ring 3. It has to use a shared memory.
- Ring 0 and ring 3 have a pointer that points on this shared memory. These 2 pointers are not configured under the same address.

This function must be called up for each **xPCI-1500** on which an interrupt is to be enabled. It installs one user interrupt function in all boards on which an interrupt is to be enabled.

First calling (first board):

- the user interrupt routine is installed
- interrupts are enabled
- user shared memory is allocated if APCI1500_SYNCHROUNOUS_MODE has been selected.

If you operate several boards **xPCI-1500** which have to react to interrupts, call up the function as often as you operate boards **xPCI-1500**.

The variable *v_FunctionName* is only relevant **for the first calling**.

From the second call of the function (next board):

- interrupts are allowed.

Interrupt

The user interrupt routine is called up by the system when an interrupt is generated.

An interrupt is generated when:

- the counter/timer has run down
- an event is generated
- the watchdog has run down

The following errors are possible

- overtemperature
- short-circuit
- no voltage is available

If several boards are operated and if they have to react to interrupts, the variable *b_BoardHandle* returns the identification number (handle) of the board which has generated the interrupt.

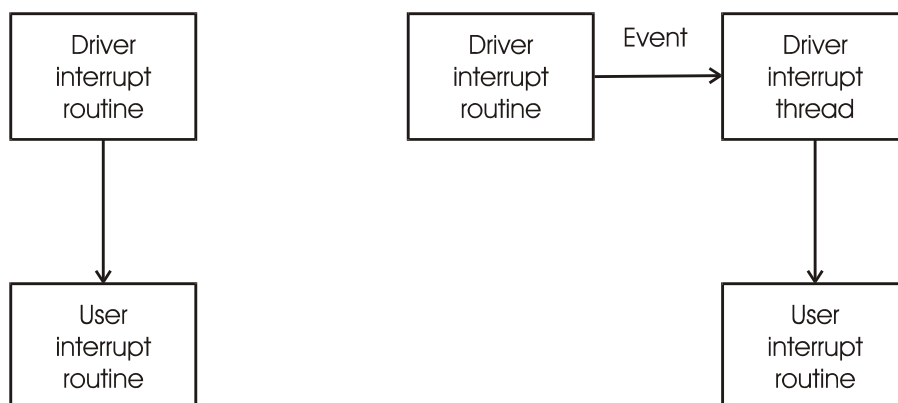
User interrupt routine can be called :

- directly by the driver interrupt routine (Synchronous mode). The code of the user interrupt routine directly operates in ring 0.
- by the driver interrupt thread (Asynchronous mode). An event is generated and the interrupt thread calls up the user interrupt routine. The code of the user interrupt routine operates in ring 3.

The driver interrupt thread have the highest priority (31) from system.

Synchronous mode

Asynchronous mode



SYNCHRONOUS-MODE	
ADVANTAGE	The code of the user interrupt routine is directly called by driver interrupt routine (ring 0). The time between interrupt and the user interrupt routine is reduced.
RESTRICTION	The user cannot debug the user interrupt routine.
	The user routine cannot call Windows API functions.
	The user routine cannot call functions which give access to global variables. The user can still use a shared memory.
	This mode is not available for Visual Basic

ASYNCHRONOUS-MODE	
ADVANTAGE	The user can debug the user interrupt routine provided he has not programmed in Visual Basic 5
	The user routine can call Windows API functions.
	The user routine can call functions which give access to global variables.
	The user routine can call all xPCI-1500 driver functions with the following extension: "i_APCI1500_XXXX"
RESTRICTION	The code of the user interrupt routine is called by driver interrupt thread routine (ring 3). The time between interrupt and the user interrupt routine is increased.

Shared memory

If you have selected the `APCI1500_SYNCHRONOUS_MODE` you cannot have access to global variables. But you have the possibility to create a shared memory (*ppv_UserSharedMemory*). The user shared memory can have all predefined compiler types or user define types.

The variable `ul_UserSharedMemorySize` indicates the size in bytes of the selected user type. A pointer of the variable *ppv_UserSharedMemory* is given to the user interrupt routine with the variable *pv_UserSharedMemory*. This is not possible for Visual Basic.

The user interrupt routine must have the following Syntax:

```
VOID v_FunctionName (BYTE b_BoardHandle,
                    BYTE b_InterruptMask,
                    BYTE b_InputChannelNbr,
                    BYTE b_UserCallingMode,
                    VOID * pv_UserSharedMemory)
```

`v_FunctionName` Name of the user interrupt routine

`b_BoardHandle` Handle of the **xPCI-1500** which has generated the interrupt

`b_InterruptMask` Mask of the events which have generated the interrupt.

`b_InputChannelNbr` Is not used. But stays for compatibility reasons

`b_UserCallingMode` `APCI1500_SYNCHRONOUS_MODE`:
The user routine is directly called by the driver interrupt routine.
`APCI1500_ASYNCHRONOUS_MODE`:
The user routine is called by driver the interrupt thread

`pv_UserSharedMemory` Pointer of the user shared memory.

Mask	Meaning
0000 0001	Event 1 has occurred
0000 0010	Event 2 has occurred
0000 0100	Counter/timer 1 has run down
0000 1000	Counter/timer 2 has run down
0001 0000	Counter 3 has run down
0010 0000	Watchdog has run down
0100 0000	Voltage error
1000 0000	Short-circuit error

The user can give another name for `v_FunctionName`, `b_BoardHandle`, `b_InterruptMask`, `b_InputChannelNbr`, `b_UserCallingMode`, `pv_UserSharedMemory`.

i**IMPORTANT!**

If you use Visual Basic 4 the following parameters have not signification.
You must used the "i_APCI1500_TestInterrupt" function

```

BYTE      b_UserCallingMode,
ULONG     ul_UserSharedMemorySize,
VOID **   ppv_UserSharedMemory,
VOID      v_FunctionName      (BYTE  b_BoardHandle,
                               BYTE  b_InterruptMask,
                               BYTE  b_InputChannelNbr,
                               BYTE  b_UserCallingMode,
                               VOID * pv_UserSharedMemory)

```

Calling convention:ANSI C:

```

typedef struct
{
    .
    .
    .
}str_UserStruct;
str_UserStruct * ps_UserSharedMemory;
void v_FunctionName      (unsigned char  b_BoardHandle,
                          unsigned char  b_InterruptMask,
                          unsigned char  b_InputChannelNbr,
                          unsigned char  b_UserCallingMode,
                          void *         pv_UserSharedMemory)
{
    str_UserStruct * ps_InterruptSharedMemory;
    ps_InterruptSharedMemory = (str_UserStruct *) pv_UserSharedMemory;
    .
    .
    .
}

int          i_ReturnValue;
unsigned char b_BoardHandle;
i_ReturnValue = i_APCI1500_SetBoardIntRoutineWin32
                (b_BoardHandle,
                 APCI1500_SYNCHRONOUS_MODE,
                 sizeof (str_UserStruct),
                 (void **) &ps_UserSharedMemory,
                 v_FunctionName);

```

Visual Basic 5:

```

Sub      v_FunctionName      (ByVal i_BoardHandle      As Integer,
                              ByVal i_InterruptMask    As Integer,
                              ByVal i_InputChannelNbr  As Integer,
                              ByVal b_UserCallingMode  As Integer,
                              ByVal l_UserSharedMemory As Long)

End Sub
Dim i_ReturnValue As Integer
Dim i_BoardHandle As Integer

```

```
i_ReturnValue = i_APCI1500_SetBoardIntRoutineWin32  
    (i_BoardHandle,  
     APCI1500_ASYNCHRONOUS_MODE,  
     0,  
     0,  
     AddressOf v_FunctionName)
```

Return value:

- 0: No error
- 1: The handle parameter of the board is wrong
- 2: Interrupt already installed
- 3: Calling mode selection of the user interrupt routine is wrong
- 4: No memory available for the user shared memory

5) i_APCI1500_TestInterrupt (..)

Syntax:

```
<Return value> = i_APCI1500_TestInterrupt
                    (PBYTE pb_BoardHandle,
                     PBYTE pb_InterruptMask,
                     PBYTE pb_ChannelNbr)
```

Parameters:

- Input:

No input signal is to occur.

- Output:

PBYTE pb_BoardHandle Handle of the board **PCI-1500** which has generated the interrupt,

PBYTE pb_InterruptMask Error mask of the event which has generated the interrupt. Several errors can occur simultaneously.

PBYTE pb_ChannelNbr Is not used. But stays for compatibility reasons.

Mask	Meaning
0000 0001	Event 1 has occurred
0000 0010	Event 2 has occurred
0000 0100	Counter/timer 1 has run down
0000 1000	Counter/timer 2 has run down
0001 0000	Counter 3 has run down
0010 0000	Watchdog has run down
0100 0000	Voltage error
1000 0000	Short-circuit error

Task:

Checks if a board **xPCI-1500** has generated an interrupt. If yes, the function returns the board handle and the interrupt source.



IMPORTANT!

This function is only in Visual Basic Dos and Windows available.

Calling convention:ANSI C :

```

unsigned char b_BoardHandle;
unsigned char b_InterruptMask;
unsigned char b_ChannelNbr;
int          i_Irq;

```

```

Irq = i_APCI1500_TestInterrupt (&b_BoardHandle,
                                &b_InterruptMask,
                                &b_ChannelNbr);

```

Return value:

-1: No interrupt
 > 0: IRQ number

6) i_APCI1500_ResetBoardIntRoutine (..)**Syntax:**

```

<Return value> = i_APCI1500_ResetBoardIntRoutine
                  (BYTE b_BoardHandle)

```

Parameters:**- Input:**

BYTE b_BoardHandle Handle of board **xPCI-1500**

- Output:

No output signal has occurred.

Task:

Stops the interrupt management of board **xPCI-1500**.

Deinstalls the interrupt routine if the management of interrupts of all **xPCI-1500** is stopped.

Calling convention:ANSI C :

```

unsigned char b_BoardHandle;
Irq = i_APCI1500_ResetBoardIntRoutine (b_BoardHandle);

```

Return value:

0: No error
 -1: Handle parameter of the board is wrong
 -2: Interrupt routine is not installed

9.2.3 Kernel functions

1) i_APCI1500_KRNL_Read16DigitalInput (...)

Syntax:

```
<Return value> = i_APCI1500_KRNL_Read16DigitalInput
                    (UINT  ui_BaseAddress,
                     PLONG pl_InputValue)
```

Parameters:

- Input:

UINT ui_BaseAddress **xPCI-1500** base address

- Output:

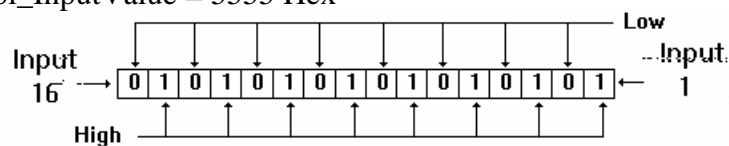
PLONG pl_InputValue State of the digital input channels of both ports
(0 to 65535)

Task:

Indicates the state of both ports. Reads the 16 input channels at once.

Example:

pl_InputValue = 5555 Hex



A voltage is present on the input channels 1, 3, 5, 7, 9, 11, 13, 15 .

A voltage is not present on the input channels 2, 4, 6, 8, 10, 12, 14, 16.

Return value:

0: No error

2) v_APCI1500_KRNL_Set16DigitalOutputOn (...)

Syntax:

```
<Return value> = v_APCI1500_KRNL_Set16DigitalOutputOn
                    (UINT  ui_BaseAddress,
                     LONG   l_Value)
```

Parameters:

- Input:

UINT ui_BaseAddress **xPCI-1500** base address
 LONG l_Value Output value (0 to 65535)

- Output:

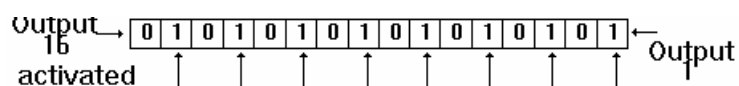
No output signal has occurred.

Task:

Sets one or several output channels of board **xPCI-1500**

Example:

l_Value = 5555 Hex



The output channels 1, 3, 5, 7, 9, 11, 13, 15 are set.

The output channels 2, 4, 6, 8, 10, 12, 14, 16 are reset.

Return value:

0: No error

9.2.4 Digital input channel

1) i_APCI1500_Read1DigitalInput (...)

Syntax :

```
<Return value> = i_APCI1500_Read1DigitalInput
                    (BYTE  b_BoardHandle,
                     BYTE  b_Channel,
                     PBYTE pb_ChannelValue)
```

Parameters:

BYTE	b_BoardHandle	Handle of board xPCI-1500
BYTE	b_Channel	The number of the input channel to be read (1 to 16)
PBYTE	pb_ChannelValue	State of the digital input channel: 0 -> low 1 -> high

Task:

Indicates the state of an input channel. The variable *b_Channel* passes the input channel to be read (1 to 16). A value is returned with the variable *pb_ChannelValue* : 0 (low) or 1 (high).

Return value:

0: No error
 -1: The handle parameter of the board is wrong
 -2: The input number is not between 1 and 16

2) i_APCI1500_Read8DigitalInput (...)

Syntax:

```
<Return value> = i_APCI1500_Read8DigitalInput
                    (BYTE  b_BoardHandle,
                     BYTE  b_Port,
                     PBYTE pb_PortValue)
```

Parameters:

BYTE	b_BoardHandle	Handle of the xPCI-1500
BYTE	b_Port	Number of the input port you want to read (1 or 2)
PBYTE	pb_PortValue	State of the digital input port (0 to 255)

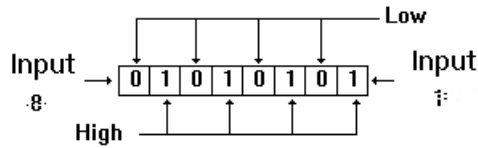
Task:

Indicates the state of a port. The variable *b_Port* passes the port to be read (1 or 2). A value is returned with the variable *pb_PortValue* .

Example:

b_Port = 1

pb_PortValue = 55 Hex



A voltage is present on the input channels 1, 3, 5, 7

A voltage is not present on the input channels 2, 4, 6, 8.

Return value:

0: No error

-1: Handle parameter of the board is wrong

-2: The parametered port number is wrong (parameter 1 or 2)

3) i_APCI1500_Read16DigitalInput (...)**Syntax:**

```
<Return value> = i_APCI1500_Read16DigitalInput
                    (BYTE  b_BoardHandle,
                     PLONG pl_InputValue)
```

Parameters:

BYTE b_BoardHandle

Handle of the **xPCI-1500**

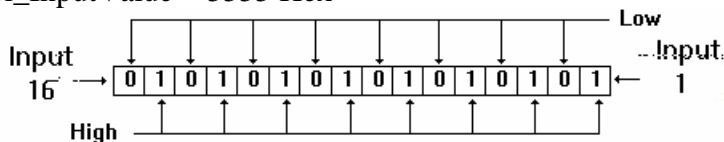
PLONG pl_InputValue

State of the digital input channels
of both ports (0 to 65535)**Task:**

Indicates the state of both ports. Reads the 16 input channels at once.

Example:

pl_InputValue = 5555 Hex



A voltage is present on the input channels 1, 3, 5, 7, 9, 11, 13, 15 .

A voltage is not present on the input channels 2, 4, 6, 8, 10, 12, 14, 16.

Return value:

0: No error

-1: The handle parameter of the board is wrong



9.2.5 Digital input channel - events

1) i_APCI1500_SetInputEventMask (...)

Syntax:

```
<Return value> = i_APCI1500_SetInputEventMask
                    (BYTE  b_BoardHandle,
                     BYTE  b_PortNbr,
                     BYTE  b_Logik,
                     PCHAR pc_EventMask)
```

Parameters:

BYTE	b_BoardHandle	Handle of board xPCI-1500
BYTE	b_Port	Number of the input port to be masked (1 or 2)
BYTE	b_Logik	Event logic Three possibilities for the first port: APCI1500_AND: This logic connects the input channels with an AND logic. APCI1500_OR: This logic connects the input channels with an OR logic. APCI1500_OR: This logic connects the input channels with a OR logic.
PCHAR	pc_EventMask	This 8-digit character string (port 1) and 6-digit character string (port 2) defines the mask of the event. Each digit indicates the state of the input channel. The state is identified by one of the following characters: "X": This input channel is not used for event "0": The input channel must be on "0" "1": The input channel must be on "1" "2": The input channel reacts to a falling edge  "3": The input channel reacts to a rising edge  "4": The input channel reacts to both edges <u>Port 1</u> : from the left to the right, the first digit of the character string is input 8 and the last digit is input 1. <u>Port 2</u> : from the left to the right, the first digit of the character string is input 14 and the last digit is input 9.



IMPORTANT!

If you use the APCI1500_AND logic, you can only use one edge event

Task:

An event can be generated for each port.

The first event is related to the first 8 input channels (port 1).

The second event is related to the next 6 input channels (port 2).

An interrupt is generated when one or both events have happened.

An event is a change of state (ex. Low→ high) on one or several input channels if an AND/OR/OR_PRIORITY logic has been defined.

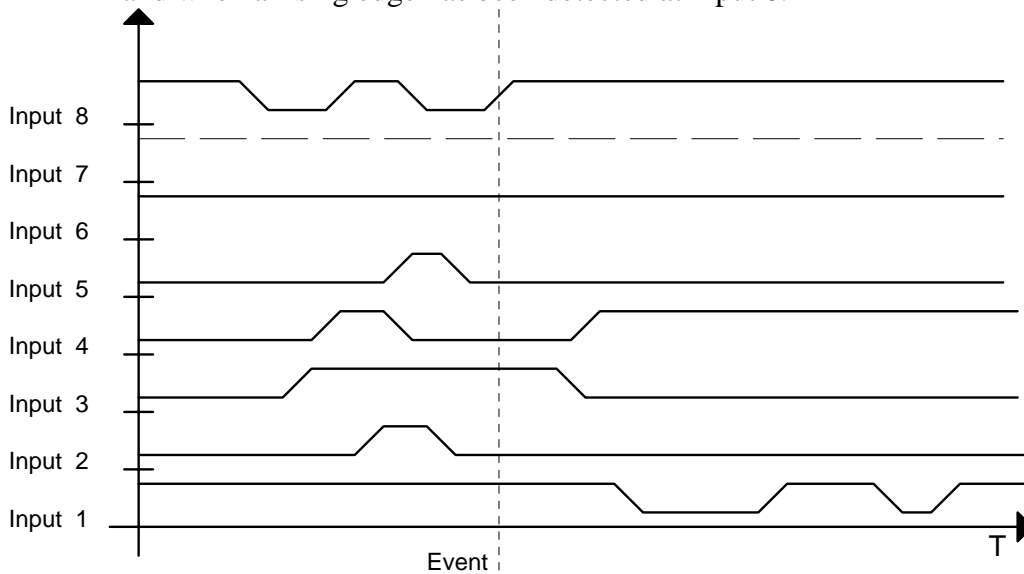
Examples:

Example 1:

b_PortNbr = 1
b_Logik = APCI1500_AND
pc_EventMask = "3X100101"

An event is generated:

- when the input channels 2, 4 and 5 are on "0".
- when the input channels 1, 3 and 6 are on "1"
- and when a rising edge has been detected at input 8.



Return value:

0: No error

- 1: The handle parameter of the board is wrong
- 2: The parameterized port number is wrong (parameter 1 or 2)
- 3: Error with the logic parameter . *b_Logik* has not the expected value
- 4: Error with the mask parameter . *pc_EventMask* has not the expected value
- 5: Interrupt routine not installed
- 6: More than 1 edge event has been declared for an AND logic

2) i_APCI1500_StartInputEvent (...)**Syntax :**

```
<Return value> = i_APCI1500_StartInputEvent
                    (BYTE  b_BoardHandle,
                     BYTE  b_PortNbr)
```

Parameters:

BYTE	b_BoardHandle	Handle of the xPCI-1500
BYTE	b_Port	Number of the input port (1 or 2)

Task:

As soon as the function is called, it is possible to process an event on one port.
First mask the input channels with the following function
i_APCI1500_SetInputEventMask .

Return value:

0: No error
-1: Handle parameter of the board is wrong
-2: The parameterized port number is wrong
-3: Event has not been initialised with the function "i_APCI1500_SetInputEvent".

3) i_APCI1500_StopInputEvent (...)**Syntax:**

```
<Return value> = i_APCI1500_StopInputEvent
                    (BYTE b_BoardHandle,
                     BYTE b_PortNbr)
```

Parameters:

BYTE	b_BoardHandle	Handle of the xPCI-1500
BYTE	b_Port	Number of the input port (1 or 2)

Task:

As soon as the function is called, it is not possible to process an event on one port.

Return value:

0: No error
-1: Handle parameter of the board is wrong
-2: The parametered port number is wrong
-3: Event has not been initialised with the function "i_APCI1500_SetInputEvent"

9.2.6 Digital output channel

1) i_APCI1500_SetOutputMemoryOn (...)

Syntax:

<Return value> = i_APCI1500_SetOutputMemoryOn
(BYTE b_BoardHandle)

Parameters:

BYTE b_BoardHandle Handle of the xPCI-1500

Task:

Activates the digital output memory.

After calling this function, the output channels you have previously activated with the functions "i_APCI1500_SetXDigitalOutputOn" are not reset.

You can reset them with the function "i_APCI1500_SetXDigitalOutputOff".

Return value:

0: No error

-1: Handle parameter of the board is wrong

2) i_APCI1500_SetOutputMemoryOff (...)

Syntax:

<Return value> = i_APCI1500_SetOutputMemoryOff (BYTE b_BoardHandle)

Parameters:

BYTE b_BoardHandle Handle of the **xPCI-1500**

Task:

Deactivates the digital output memory.

Return value:

0: No error

-1: Handle parameter of the board is wrong

3) i_APCI1500_Set1DigitalOutputOn (...)

Syntax:

<Return value> = i_APCI1500_Set1DigitalOutputOn
(BYTE b_BoardHandle,
BYTE b_Channel)

Parameters:

BYTE	b_BoardHandle	Handle of the xPCI-1500
BYTE	b_Channel	Number of the output channel you want to set (1 to 16)

Task:

Sets the output channel which has been passed with *b_Channel*.
Setting an output channel means setting an output channel on high.

Switching on the digital output memory (ON)

see function "i_APCI1500_SetOutputMemoryOn (...)
b_Channel= 1

The output channel 1 is set. The others output channels hold their state.

Switching off the digital output memory (OFF)

see function "i_APCI1500_SetOutputMemoryOff (...)
b_Channel= 1

The output channel 1 is set. The others output channels are reset.

If you have switched off the digital output memory (OFF), all the others input channels are set to "0".

Return value:

0: No error
-1: Handle parameter of the board is wrong
-2: Input number is not between 1 and 16

4) i_APCI1500_Set1DigitalOutputOff (...)

Syntax :

<Return value> = i_APCI1500_Set1DigitalOutputOff
(BYTE b_BoardHandle,
BYTE b_Channel)

Parameters:

BYTE	b_BoardHandle	Handle of the xPCI-1500
BYTE	b_Channel	Number of the output channel you want to reset (1 to 16)

Task:

Resets the output channel you have passed with *b_Channel*. Resetting an output channel means setting to low.



IMPORTANT!

You can use this function only if the digital output memory is ON. See function i_APCI1500_SetOutputMemoryOn (..).

Return value:

0: No error

-1: The handle parameter of the board is wrong

-2: The input number is not between 1 and 16

-3: Digital output memory OFF.

First use the function "i_APCI1500_SetOutputMemoryOn"

5) i_APCI1500_Set8DigitalOutputOn (...)**Syntax:**

```
<Return value> = i_APCI1500_Set8DigitalOutputOn
                    (BYTE b_BoardHandle,
                     BYTE b_Port,
                     BYTE b_Value)
```

Parameters:

BYTE	b_BoardHandle	Handle of the board xPCI-1500
BYTE	b_Port	Number of the output port (1 or 2)
BYTE	b_Value	Output value (0 to 255)

Task:

Sets one or several output channels of a port. Setting an output channel means setting to high.

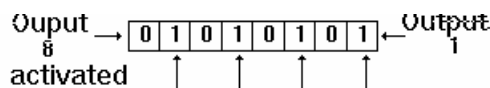
If you have switched off the digital output memory (OFF), the input channels are set to "0".

Example:***Switching on the digital output memory (ON)***

see function "i_APCI1500_SetOutputMemoryOn (...)"

b_Port = 1

b_Value = 55 Hex



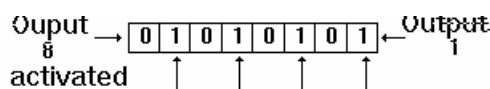
The output channels 1, 3, 5, 7 are set. The other output channels hold their state.

Switching off the digital output memory (OFF)

see function "i_APCI1500_SetOutputMemoryOff (...)"

b_Port = 1

b_Value = 55 Hex



The output channels 1, 3, 5, 7 are set. The other output channels are reset.

Return value:

0: No error

-1: The handle parameter of the board is wrong

-2: The port number is not 1 or 2

6) i_APCI1500_Set8DigitalOutputOff (...)

Syntax:

```
<Return value> = i_APCI1500_Set8DigitalOutputOff
                    (BYTE b_BoardHandle,
                     BYTE b_Port,
                     BYTE b_Value)
```

Parameters:

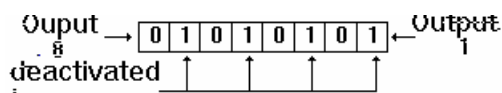
BYTE	b_BoardHandle	Handle of the xPCI-1500
BYTE	b_Port	Number of the output port (1 or 2)
BYTE	b_Value	Output value (0 to 255)

Task:

Resets one or several output channels of one port. Resetting means setting to high.

Example:

```
b_Port = 1
b_Value = 55 Hex
```



The output channels 1, 3, 5, 7 are reset.



IMPORTANT!

You can use this function only if the digital output memory is ON. See function i_APCI1500_SetOutputMemoryOn (..).

Return value:

- 0: No error
- 1: The handle parameter of the board is wrong
- 2: The port number is not 1 or 2
- 3: The digital output memory is OFF. Please first use the function i_APCI1500_SetOutputMemoryOn

7) v_APCI1500_Set16DigitalOutputOn (...)

Syntax:

<Return value> = v_APCI1500_Set16DigitalOutputOn
 (BYTE b_BoardHandle,
 LONG l_Value)

Parameters:

BYTE b_BoardHandle Handle of the **xPCI-1500**
 LONG l_Value Output value (0 to 65535)

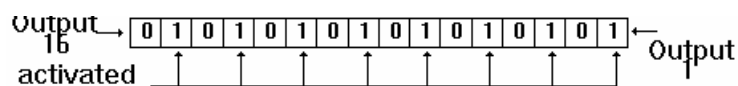
Task:

Sets one or several output channels of board **xPCI-1500**

Example:

Switching on the digital output memory (ON)

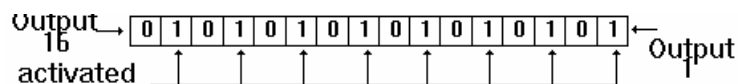
see function "i_APCI1500_SetOutputMemoryOn (...)
 l_Value = 5555 Hex



The output channels 1, 3, 5, 7, 9, 11, 13, 15 are set. The other output channels hold their state.

Switching off the digital output memory (OFF)

see function "i_APCI1500_SetOutputMemoryOff (...)
 l_Value = 5555 Hex



Outputs 1, 3, 5, 7, 9, 11, 13, 15 are set. Outputs 2, 4, 6, 8, 10, 12, 14, 16 are reset.

Return value:

0: No error
 -1: The handle parameter of the board is wrong

8) v_APCI1500_Set16DigitalOutputOff (...)

Syntax:

<Return value> = v_APCI1500_Set16DigitalOutputOff
(BYTE b_BoardHandle,
LONG l_Value)

Parameters:

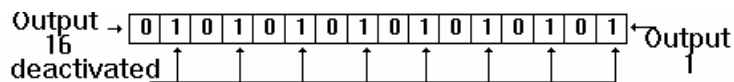
BYTE	b_BoardHandle	Handle of the xPCI-1500
LONG	l_Value	Output value (0 to 65535)

Task:

Resets one or several output channels of board **xPCI-1500**.

Example:

l_Value = 5555 Hex



The output channels 1, 3, 5, 7, 9, 11, 13, 15 are reset.



IMPORTANT!

You can use this function only if the digital output memory is ON. See function i_APCI1500_SetOutputMemoryOn (...).

Return value:

0: No error
-1: The handle parameter of the board is wrong
-2: The digital output memory is OFF. Please first use the function "i_APCI1500_SetOutputMemoryOn"

9.2.7 Timer/counter and watchdog

1) i_APCI1500_InitTimerInputClock (...)

Syntax:

<Return value> = i_APCI1500_InitTimerInputClock
(BYTE b_BoardHandle,
BYTE b_InputClockCase)

Parameters:

BYTE	b_BoardHandle	Handle of the xPCI-1500
BYTE	b_InputClockCase	Select the input clock for the 3 timers: - APCI1500_115_KHZ : 111.5 kHz - APCI1500_3_6_KHZ : 3.6 kHz - APCI1500_1_8_KHZ : 1.8 kHz

**IMPORTANT!**

The timer component internally operates with half of the input frequency.

Task:

Selects the input clock timing for all timers

Return value:

0: No error

-1: The handle-parameter of the board is wrong

-2: The parameter for selecting the input clock is wrong

2) i_APCI1500_InitTimerCounter1 (...)**Syntax:**

```
<Return value> = i_APCI1500_InitTimerCounter1
                    (BYTE b_BoardHandle,
                     BYTE b_CounterOrTimerSelect,
                     LONG l_ReloadValue,
                     BYTE b_ContinuousOrSingleCycleSelect,
                     BYTE b_InterruptHandling)
```

Parameters:

BYTE	b_BoardHandle:	Handle of the xPCI-1500
BYTE	b_CounterOrTimerSelect	Select the mode of the first counter/timer - APCI1500_TIMER: the first counter/timer is used as timer - APCI1500_COUNTER: The first counter/timer is used as counter
LONG	l_ReloadValue	This parameter has two meanings. If the counter/timer is used as a counter, it loads the start value of the counter. If the counter/timer is used as a timer, it loads the divider factor for the output.
BYTE	b_ContinuousOrSingleCycleSelect	- APCI1500_CONTINUOUS: Each time the counter value or timer value is set to "0", <i>l_ReloadValue</i> is loaded. - APCI1500_SINGLE: If the counter or timer value is set to "0", the counter or timer is stopped.
BYTE	b_InterruptHandling	Interrupts can be generated, when the counter has run down, or when the timer output is on high. With this parameter the user decides if interrupts are used or not. APCI1500_ENABLE: Interrupts are enabled. APCI1500_DISABLE: Interrupts are disabled.

Task:

Selects the operating mode of the first counter/timer. The user enters its start value.

You have to decide:

- if the counter/timer must execute once or several times the counting operation,
- if the counter/timer is used as a counter or a timer,
- and if an interrupt must be generated when the counter/timer has run down.

Return value:

0: No error

-1: The handle-parameter of the board is wrong

-2: The parameter for selecting the counter or the timer is wrong
(APCI1500_COUNTER or APCI1500_TIMER)

-3: Error with the interrupt selection
(APCI1500_ENABLE or APCI1500_DISABLE)

-4: The user interrupt routine is not installed

-5: Cycle parameter is wrong (APCI1500_CONTINUOUS or APCI1500_SINGLE)

3) i_APCI1500_InitTimerCounter2 (...)**Syntax:**

```
<Return value> = i_APCI1500_InitTimerCounter2
                                (BYTE  b_BoardHandle,
                                BYTE  b_CounterOrTimerSelect,
                                LONG  l_ReloadValue,
                                BYTE  b_ContinuousOrSingleCycleSelect,
                                BYTE  b_HardwareOrSoftwareTriggerSelect,
                                BYTE  b_HardwareOrSoftwareGateSelect,
                                BYTE  b_InterruptHandling)
```

Parameters:

BYTE	b_BoardHandle	Handle of the xPCI-1500
BYTE	b_CounterOrTimerSelect	<p>Selects the mode of the 2nd counter/timer</p> <ul style="list-style-type: none"> - APCI1500_TIMER: The 2nd counter/timer is used as a timer - APCI1500_COUNTER: The 2nd counter/timer is used as a counter
LONG	l_ReloadValue	<p>This parameter has two meanings.</p> <p>If the counter/timer is used as a counter, it loads the start value of the counter.</p> <p>If the counter/timer is used as a timer, it loads the divider factor for the output.</p>
BYTE	b_ContinuousOrSingleCycleSelect	<ul style="list-style-type: none"> - APCI1500_CONTINUOUS: Each time the counter value or timer value is set to "0", <i>l_ReloadValue</i> is loaded. - APCI1500_SINGLE: If the counter or timer value is set to "0", the counter or timer is stopped.

- BYTE b_HardwareOrSoftwareTriggerSelect:
- APCI1500_HARDWARE_TRIGGER: The input 12 is used for the trigger. If this input channel is on high, the start value is re-loaded.
 - APCI1500_SOFTWARE_TRIGGER: Input 12 has no influence on the trigger
- BYTE b_HardwareOrSoftwareGateSelect:
- APCI1500_HARDWARE_GATE: The input 13 is used for the gate. If this input channel is on high, the counter/timer is started. If this input channel is on low, the counter/timer is stopped.
 - APCI1500_SOFTWARE_GATE: Input 13 has no influence on the gate.
- BYTE b_InterruptHandling
- Interrupts can be generated, when the counter has run down, or when the timer output is on high. With this parameter the user decides if interrupts are used or not.
- APCI1500_ENABLE: Interrupts are enabled.
 - APCI1500_DISABLE: Interrupts are disabled.

Task:

Selects the operating mode of the second counter/timer. Enter its start value.

You have to decide:

- if the counter/timer must execute the counting operation once or several times,
- if the counter/timer is used as a counter or a timer,
- if an interrupt must be generated when the counter/timer has run down,
- if the external trigger is used and if the external gate is used.

Return value:

0: No error

- 1: The handle-parameter of the board is wrong
- 2: Wrong selection for counter/timer
(APCI1500_COUNTER or APCI1500_TIMER)
- 3: Error with the interrupt selection
(APCI1500_ENABLE or APCI1500_DISABLE)
- 4: User interrupt routine is not installed
- 5: Cycle parameter is wrong
(APCI1500_CONTINUOUS or APCI1500_SINGLE)
- 6: Wrong gate parameter
(APCI1500_SOFTWARE_GATE or APCI1500_HARDWARE_GATE)
- 7: Wrong trigger parameter
(APCI1500_SOFTWARE_TRIGGER or APCI1500_HARDWARE_TRIGGER)

4) i_APCI1500_InitWatchdogCounter3 (...)

Syntax:

```
<Return value> = i_APCI1500_InitWatchdogCounter3
                                (BYTE b_BoardHandle,
                                 BYTE b_WatchdogOrCounterSelect,
                                 LONG l_ReloadValue,
                                 BYTE b_ContinuousOrSingleCycleSelect,
                                 BYTE b_HardwareOrSoftwareGateSelect,
                                 BYTE b_InterruptHandling)
```

Parameters:

BYTE	b_BoardHandle	Handle of the xPCI-1500
BYTE	b_WatchdogOrCounterSelect	<p>Selects the mode of the third counter/watchdog</p> <ul style="list-style-type: none"> - APCI1500_WATCHDOG: The third counter/watchdog is used as a watchdog. - APCI1500_COUNTER: The third counter/watchdog is used as a counter.
LONG	l_ReloadValue	<p>This parameter has 2 meanings.</p> <p>If the counter/watchdog is used as a counter, it loads the limit value of the counter. If the counter/watchdog is used as a watchdog, it loads the watchdog time.</p>
BYTE	b_ContinuousOrSingleCycleSelect	<ul style="list-style-type: none"> - APCI1500_CONTINUOUS: Each time the counting or timer value is set to "0", <i>l_ReloadValue</i> is loaded. - APCI1500_SINGLE: if the counter or timer value is set to "0", the counter or timer is stopped.
BYTE	b_HardwareOrSoftwareGateSelect	<ul style="list-style-type: none"> - APCI1500_HARDWARE_GATE: Input 16 is used for the gate. - APCI1500_SOFTWARE_GATE: Input 16 has no influence on the gate.
BYTE	b_InterruptHandling	<p>Interrupts can be generated, when the counter or watchdog has run down. With this parameter the user decides to use interrupts or not.</p> <ul style="list-style-type: none"> - APCI1500_ENABLE: Interrupts are enabled. - APCI1500_DISABLE: Interrupts are disabled.

Task:

Selects the operating mode of the third counter/watchdog. Enter its limit.

You have to decide:

- if the counter must execute once or several times the counting operation,
- if the counter/watchdog is used as a counter or a watchdog,
- if an interrupt must be generated when the counter/watchdog has run down,
- and if the external gate is used (if it is used as a counter).

Return value:

No error

- 1: The handle parameter of the board is wrong
- 2: The parameter for selecting the counter or the timer is wrong
(APCI1500_COUNTER or APCI1500_WATCHDOG)
- 3: Interrupt selection error (APCI1500_ENABLE or APCI1500_DISABLE)
- 4: User interrupt routine is not installed
- 5: Cycle parameter is wrong (APCI1500_CONTINUOUS or APCI1500_SINGLE)
- 6: Gate parameter is wrong (APCI1500_SOFTWARE_GATE or
APCI1500_HARDWARE_GATE)

5) i_APCI1500_StartTimerCounter1(...)**Syntax:**

<Return value> = i_APCI1500_StartTimerCounter1
(BYTE b_BoardHandle)

Parameters:

BYTE b_BoardHandle Handle of the xPCI-1500

Task:

Starts the first counter/timer. Please initialise it previously with the function "i_APCI1500_InitTimerCounter1".

If the counter is used, it is now ready for counting.

If the timer is used, it is now running.

Return value:

0: No error

- 1: The handle parameter of the board is wrong
- 2: The counter or timer has not been initialised.
Please use function "i_APCI1500_InitTimerCounter1"

6) i_APCI1500_StartTimerCounter2 (...)**Syntax:**

<Return value> = i_APCI1500_StartTimerCounter2 (BYTE b_BoardHandle)

Parameters:

BYTE b_BoardHandle Handle of the xPCI-1500

Task:

Starts the second counter/timer, Please initialise it previously with the function "i_APCI1500_InitTimerCounter2".

If the counter is used, it is now ready for counting.

If the timer is used, it is now running.

Return value:

0: No error

- 1: The handle parameter of the board is wrong
- 2: The counter or timer has not been initialised.
Please use function "i_APCI1500_InitTimerCounter2"

7) i_APCI1500_StartCounter3 (...)**Syntax:**

<Return value> = i_APCI1500_StartCounter3 (BYTE b_BoardHandle)

Parameters:

BYTE b_BoardHandle Handle of the **xPCI-1500**

Task:

Starts the third counter. Please initialise it previously with the function "i_APCI1500_InitWatchdogCounter3".

Return value:

0: No error

-1: The handle parameter of the board is wrong

-2: Counter has not been initialised

-3: The counter/watchdog has been initialised as a watchdog.

"i_APCI1500_InitWatchdogCounter3"

8) i_APCI1500_StopTimerCounter1 (...)**Syntax:**

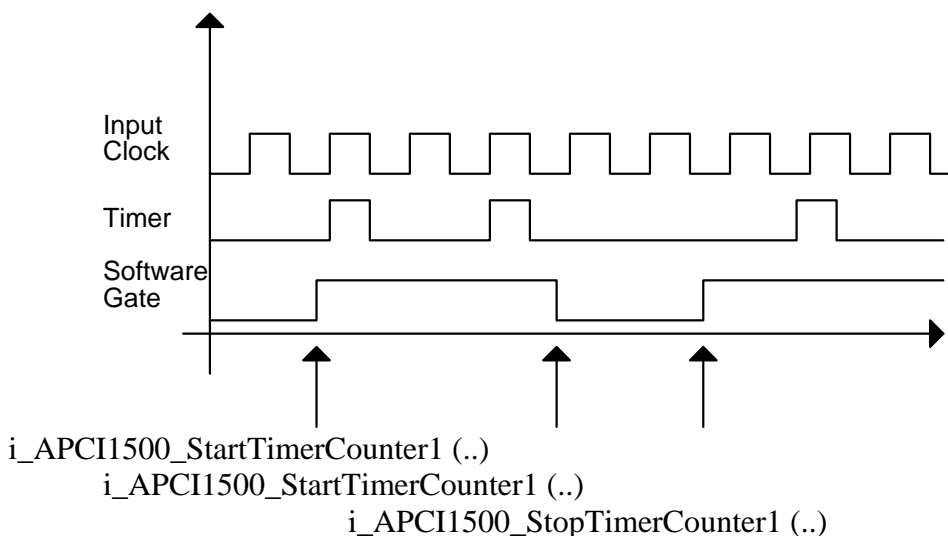
<Return value> = i_APCI1500_StopTimerCounter1 (BYTE b_BoardHandle)

Parameters:

BYTE b_BoardHandle Handle of the **xPCI-1500**

Task:

Stops the first counter/timer. The timer counter value is freezed. It has the same influence as a hardware gate.

**Return value:**

0: No error

-1: The handle parameter of the board is wrong

-2: The counter or timer has not been initialised.

Please use function "i_APCI1500_InitTimerCounter1"

9) i_APCI1500_StopTimerCounter2 (...)

Syntax:

<Return value> = i_APCI1500_StopTimerCounter2 (BYTE b_BoardHandle)

Parameters:

BYTE	b_BoardHandle	Handle of the xPCI-1500
------	---------------	--------------------------------

Task:

Stops the second counter/timer. The timer counter value is frozen.

Return value:

0: No error

-1: The handle parameter of the board is wrong

-2: The counter or timer has not been initialised.

Please use function "i_APCI1500_InitTimerCounter2"

10) i_APCI1500_StopCounter3 (...)

Syntax:

<Return value> = i_APCI1500_StopCounter3 (BYTE
b_BoardHandle)

Parameters:

BYTE	b_BoardHandle	Handle of the xPCI-1500
------	---------------	--------------------------------

Task:

Stops the third counter. The counter value is freezed.

Return value:

0: No error

-1: The handle parameter of the board is wrong

-2: The counter has not been initialised

-3: The counter/watchdog has been initialised as a watchdog.

Please use function "i_APCI1500_InitWatchdogCounter3"

11) i_APCI1500_TriggerTimerCounter1 (...)

Syntax:

<Return value> = i_APCI1500_TriggerTimerCounter1 (BYTE b_BoardHandle)

Parameters:

BYTE b BoardHandle Handle of the **xPCI-1500**

Task:

Triggers the first counter/timer. The start value is loaded in the counter/timer.

Return value:

0: No error

-1: The handle parameter of the board is wrong

-2: The counter or timer has not been initialised.

Please use function "i_APCI1500_InitTimerCounter1"

12) i_APCI1500_TriggerTimerCounter2 (...)**Syntax:**

<Return value> = i_APCI1500_TriggerTimerCounter2 (BYTE b_BoardHandle)

Parameters:

BYTE b_BoardHandle Handle of the **xPCI-1500**

Task:

Triggers the second counter/timer. The start value is loaded in the counter/timer.

Return value:

0: No error

-1: The handle parameter of the board is wrong

-2: The counter or timer has not been initialised.

Please use function "i_APCI1500_InitTimerCounter2"

13) i_APCI1500_TriggerCounter3 (...)**Syntax:**

<Return value> = i_APCI1500_TriggerCounter3 (BYTE b_BoardHandle)

Parameters:

BYTE b_BoardHandle Handle of the **xPCI-1500**

Task:

Triggers the third counter. The start value is loaded in the counter.

Return value:

0: No error

-1: The handle parameter of the board is wrong

-2: The counter has not been initialised

-3: The counter/watchdog has been initialised as a watchdog.

Please use function "i_APCI1500_InitWatchdogCounter3"

14) i_APCI1500_Watchdog (...)**Syntax:**

<Return value> = i_APCI1500_Watchdog (BYTE b_BoardHandle)

Parameters:

BYTE b_BoardHandle Handle of the **xPCI-1500**

Task:

Triggers the watchdog. The start value is loaded in the watchdog.

Return value:

0: No error

-1: The handle parameter of the board is wrong

-2: The watchdog has not been initialised

-3: The counter/watchdog has been initialised as a counter.

Please use function "i_APCI1500_InitWatchdogCounter3"

15) i_APCI1500_ReadTimerCounter1 (...)**Syntax:**

```
<Return value> = i_APCI1500_ReadTimerCounter1
                                (BYTE      b_BoardHandle,
                                PLONG      pl_ReadValue)
```

Parameters:

BYTE	b_BoardHandle	Handle of the xPCI-1500
PLONG	pl_ReadValue	This parameter has two meanings. When the counter/timer is used as a counter, it returns the current value of the counter. When the counter/timer is used as a timer, it returns the current value of the timer.

Task:

Reads the current value of the first counter/timer if used as a counter or reads the timer content if used as a timer.

Counter: the counter value is decremented each time the input channel changes from low to high. This counter value can be read with this function.

Timer: the timer value is decremented each time the input clock changes from low to high. This timer value can be read with this function.

Return value:

0: No error

-1: The handle parameter of the board is wrong

-2: The counter or timer has not been initialised.

Please use function "i_APCI1500_InitTimerCounter1"

16) i_APCI1500_ReadTimerCounter2 (...)**Syntax:**

```
<Return value> = i_APCI1500_ReadTimerCounter2
                                (BYTE      b_BoardHandle,
                                PLONG      pl_ReadValue)
```

Parameters:

BYTE	b_BoardHandle	Handle of the xPCI-1500
PLONG	pl_ReadValue	This parameter has two meanings. When the counter/timer is used as a counter it returns the current value of the counter. When the counter/timer is used as a timer, it returns the current value of the timer.

Task:

Reads the current value of the second counter/timer if used as a counter or reads the timer content if used as a timer.

Counter: the counter value is decremented each time the input channel changes from low to high. This counter value can be read with this function.

Timer: the timer value is decremented each time the input clock changes from low to high. This timer value can be read with this function.

Return value:

0: No error

-1: The handle parameter of the board is wrong

-2: The counter or timer has not been initialised.

Please use function "i_APCI1500_InitTimerCounter2"

17) i_APCI1500_ReadCounter3 (...)**Syntax:**

```
<Return value> = i_APCI1500_ReadCounter3
                                (BYTE  b_BoardHandle,
                                PLONG_ pl_ReadValue)
```

Parameters:

BYTE	b_BoardHandle	Handle of the xPCI-1500
PLONG	pl_ReadValue	When the counter/watchdog is used as a counter, it returns the current value of the counter.

Task:

Reads the current value of the third counter/watchdog if used as a counter.

Counter: the counter value is decremented each time the input changes from low to high. This counter value can be read with this function.

Return value:

0: No error

-1: The handle parameter of the board is wrong

-2: The counter has not been initialised.

-3: The counter/watchdog has been initialised as a watchdog.

Please use function "i_APCI1500_InitWatchdogCounter3"

10 GLOSSARY

Table 10-1: Glossary

Term	Description
A/D converter	= <i>ADC</i> An electronic device that produces a digital output directly proportional to an analog signal output.
Acquisition	The process by which data is gathered by the computer for analysis or storage.
Bus	The group of conductors that interconnect individual circuitry in a computer. Typically, a bus is the expansion vehicle to which I/O or other devices are connected. Examples of PC buses are PCI, PC Card (PCMCIA), ISA (AT), and EISA bus.
Clock	A circuit that generates time and clock pulses for the synchronisation of the conversion
Counter	A circuit that counts pulses or measures pulse duration
Creeping distance	In order to avoid the danger of the effects of electrical voltages and currents for electrical-mechanical components, it is required to keep minimum isolation distances. The creeping distance is the shortest distance alongside of an isolation surface between two reference points (contact elements).
D/A converter	= <i>DAC</i> A device that converts digital information into a corresponding analog voltage or current.
Data acquisition	Gathering information from sources such as sensors and transducers in an accurate, timely and organized manner. Modern systems convert this information to digital data which can be stored and processed by a computer.
Diagnostic program	A utility program used to isolate hardware malfunctions on-board, or software malfunctions in the program.
DC voltage	= <i>Direct current voltage</i> DC voltage means that the voltage is constant respecting the time. It will always fluctuate slightly. Especially at switching on and switching off the transition behaviour is of high significance.
Digital signal	A signal which has distinct states. Digital computers process data as binary information having either 1 or 0 states.
Disturb signal	Interferences that occur during the transfer caused by reduced bandwidth, attenuation, gain, noise, delay time etc.
Driver	A part of the software that is used to control a specific hardware device such as a data acquisition board or a printer.
FIFO	= <i>First In First Out</i> The first data into the buffer is the first data out of the buffer.
Gain	The factor by which an incoming signal is multiplied.
Ground	A common reference point for an electrical system.
Impedance	The reciprocal of admittance. Admittance is the complex ratio of the voltage across divided by the current flowing through a device, circuit element, or network.

Inductive loads	The voltage over the inductor is $U=L \cdot (dI/dt)$, whereas L is the inductivity and I is the current. If the current is switched on fast, the voltage over the load can become very highly for a short time.
Input impedance	The measured resistance and capacitance between the high and low inputs of a circuit.
Input level	The input level is the logarithmic relation of two electric units of the same type (voltage, current or power) at the signal input of any receive device. The receive device is often a logic level that refers to the input of the switch. The input voltage that corresponds with logic "0" is here between 0 and 15 V, and the one that corresponds with logic "1" is between 17 and 30 V.
Instrumentation amplifier	= <i>IA</i> Instrumentation amplifiers (IA) are precise measuring amplifiers with high input impedance, low output impedance, significantly high common-mode suppression and adjustable gain with high continuity respecting the time.
Interrupt	A signal to the CPU indicating that the board detected the occurrence of a specified condition or event.
Limit value	Exceeding the limit values, even for just a short time, can lead to the destruction or to a loss of functionality.
MUX	= <i>Multiplexer</i> An array of semiconductor or electromechanical switches with a common output used for selecting one of a number of input signals.
Noise immunity	Noise immunity is the ability of a device to work during an electromagnetic interference without reduced functions.
Noise suppression	The suppression of undesirable electrical interferences to a signal. Sources of noise include the ac power line, motors, generators, transformers, fluorescent lights, CRT displays, computers, electrical storms, welders, radio transmitters, and others.
Operating voltage	The operating voltage is the voltage that occurs during the continuous operation of the device. It may not exceed the continuous limit voltage. Furthermore, any negative operation situations, such as net overvoltages over one minute at switching on the device must be taken in consideration.
Optical isolation	The technique of using an optoelectric transmitter and receiver to transfer data without electrical continuity, to eliminate high-potential differences and transients.
Opto-coupler	With an opto-coupler direct current voltage can be transferred. The advantage is the small size.
Output current	The maximum amount of current the sensor can supply across the output signal, expressed as amps DC (A DC).
Output voltage	The nominal voltage output reading when shaft is rotated to full range, expressed in volts DC /Vo DC)
Parameter	The parameters of a control comprise all for the control process required numeric values, e.g. for limit values and technological number.

PCI bus	PCI bus is a fast local bus with a clock rate up to 33 MHz. This bus is used for processing a great number of data. The PCI bus is not limited like the ISA and EISA systems.
PLD	= <i>Programmable Logic Device</i> Prorammmable logic circuitry
Protective circuitry	A protective circuitry of the active part is done in order to protect the control electronic. The simplest protective circuitry is the parallel switching of a resistance.
Reference voltage	Reference voltages are stable voltages that are used as reference unit. From them voltages can be derived that are required for example in current supplies and in other electronic circuitries.
Resolution	The smallest significant number to which a measurement can be determined. For example a converter with 12-bit resolution can resolve 1 part in 4096.
Sensor	A device that responds to physical stimuli (heat, light, sound, pressure, motion, etc.) and produces a corresponding electrical output.
Settling time	The time required, after application of a step input signal, for the output voltage to settle and remain within a specified error band around the final value. The settling time of a system includes that of all of the components of the system.
Short circuit	A short circuit of two clamps of an electric switch is when the concerning clamp voltage is zero.
Short circuit current	Short circuit current is the current between tow short-circuited clamps.
Signal delay	The change of a signal affects the following circuitries with finite velocity; the signal will be delayed. Besides the signal delay times that are not wanted, the signal delay can be extended by time switches and delay lines.
Synchronous	In hardware, it is an event that occurs in a fixed time relationship to another event. In software, it refers to a function that begins an operation and returns to the calling program only when the operation is complete.
Throughput rate	The maximum repetitive rate at which data conversion system can operate with a specified accuracy. It is determined by summing the various times required for each part of the system and then by taking the inverse of this time.
Timer	The timer allows the adaptation of program processes between processor and peripheral devices. It usually contains from each other independent counters and can be programmed for several operation types over a control word register.

11 INDEX

A

ADDIREG registration program 23

B

Base address and interrupt
 Software functions 42
 Block diagram 32
 Board registration with ADDIREG 23

C

Changing the registration of a board 27
 Component scheme 16
 Configuring a new board 25
 Connection principle 29
 Connection to the screw terminal panel 31
 Counter
 Software functions 76
 Counter/timer
 Function description 39

D

Definition of application 8
 digital inputs 13
 Digital inputs
 Function description 33
 Software functions 66, 68
 digital outputs 13
 Digital outputs
 Function description 37
 Software functions 71
 Dimensions 12

E

EMC
 Electromagnetic compatibility 12
 Energy requirements 13

F

Functions of the board 32

G

Glossary 87

H

Handling of the board 11

I

Installation of the board 18

Intended use 8
 Internet 28
 Interrupt
 Function description 38
 Software functions 48

K

Kernel functions
 Software functions 64

L

Limit values 13

O

Optical isolation 15
 Options 13

P

Personal protection
 User 10
 Physical set-up of the board 12
 Pin assignment 29

Q

Qualification
 User 10

R

Registering a new board 27

S

safety 15
 Slots
 APCI-1500 18
 CPCI-1500 20
 Software 22
 Software download 28
 Standard software 41

T

Technical data 12
 Timer
 Software functions 76

U

Update 28
 Usage restrictions 8

W

Watchdog
 Software functions 76
Weight 12