



DIN EN ISO 9001:2000  
certified



ADDI-DATA GmbH  
Dieselstraße 3  
D-77833 OTTERSWEIER



Technical support:  
+49 (0)7223 / 9493 - 0

**Technical description**

**CPCI-1710, CPCI-1711**

**Multifunction counter board,  
optically isolated**

Edition: 06.07 - 11/2006

## Copyright

All rights reserved. This manual is intended for the manager and its personnel.  
No part of this publication may be reproduced or transmitted by any means.  
Offences can have penal consequences.

## Guarantee and responsibility

Basically are effective our "general terms of delivery and payment". The manager receives them at the latest with the invoice. Claims for guarantee and responsibility in case of injuries and material damages are excluded, if they are due to one or some of the following causes:

- if the board has not been used for the intended purpose
- improper installation, operation and maintenance of the board
- if the board has been operated with defective safety devices or with not appropriate or non-functioning safety equipment
- non-observance of the instructions concerning: transport, storage, inserting the board, use, limit values, maintenance, device drivers
- altering the board at the user's own initiative
- altering the source files at the user's own initiative
- not checking properly the parts which are subject to wear
- disasters caused by the intrusion of foreign bodies and by influence beyond the user's control.

## Licence for ADDI-DATA software products

Read carefully this licence before using the standard software. The right for using this software is given to the customer, if he/she agrees to the conditions of this licence.

- this software can only be used for configuring ADDI-DATA boards.
- copying the software is forbidden (except for archiving/ saving data and for replacing defective data carriers).
- deassembling, decompiling, decoding and reverse engineering of the software are forbidden.
- this licence and the software can be transferred to a third party, so far as this party has purchased a board, declares to agree to all the clauses of this licence contract and the preceding owner has not kept copies of the software.

## Trademarks

Borland C and Turbo Pascal are registered trademarks of Borland International, INC.

Intel is a registered trademark of Intel Corporation.

AT, IBM, ISA and XT are registered trademarks of International Business Machines Corporation.

Microsoft, MS-DOS, Visual Basic and Windows are registered trademarks of Microsoft Corporation.

***The original version of this manual is in German. You can obtain it on request.***

# WARNING

In case of wrong uses and if the board is not used for the purpose it is intended:



◆ people may be injured,



◆ the board, PC and peripheral may be destroyed,



◆ the environment may be polluted

◆ **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.

<b>1</b>	<b>DEFINITION OF APPLICATION .....</b>	<b>8</b>
1.1	Intended use .....	8
1.2	Usage restrictions.....	8
1.3	General description of the board .....	8
<b>2</b>	<b>USER .....</b>	<b>10</b>
2.1	Qualification .....	10
2.2	Personal protection.....	10
<b>3</b>	<b>HANDLING OF THE BOARD .....</b>	<b>11</b>
<b>4</b>	<b>TECHNICAL DATA.....</b>	<b>12</b>
4.1	Electromagnetic compatibility (EMC) .....	12
4.2	Physical set-up of the board .....	12
4.3	Versions .....	13
4.4	Limit values.....	13
4.4.1	Inputs.....	14
	24 V option .....	15
4.4.2	Outputs.....	15
4.4.3	Safety .....	17
4.5	Component scheme.....	18
<b>5</b>	<b>AVAILABLE FUNCTIONS OF THE CPCI-1710 .....</b>	<b>19</b>
5.1	Available signals.....	19
5.1.1	Connectable signal lines.....	19
5.1.2	Maximal signal wirings of the CPCI-1710 .....	19
5.2	Available functions.....	20
5.2.1	Programmable functions of the counter board .....	20
5.2.2	Connection facilities depending on the chosen function .....	20
5.2.3	Delivered manuals .....	21
<b>6</b>	<b>INSTALLATION OF THE BOARD .....</b>	<b>22</b>
6.1	Opening the CompactPCI system .....	22
6.2	Installation .....	22
<b>7</b>	<b>SOFTWARE .....</b>	<b>24</b>
7.1	Delivered software .....	24
7.2	Configuration of the CPCI-1710 with ADDIREG .....	24
7.2.1	Registration of a new board .....	28

7.2.2	Changing the registration of an existing board .....	29
<b>7.3</b>	<b>Loading a function into a function module.....</b>	<b>30</b>
7.3.1	Module configuration with SET1710 .....	30
7.3.2	Setting a module configuration.....	33
	Module configuration through mouse click.....	33
	Module configuration through keyboard.....	33
<b>7.4</b>	<b>ADDI-DATA on the Internet .....</b>	<b>34</b>
<b>8</b>	<b>CONNECTING THE PERIPHERAL.....</b>	<b>35</b>
<b>8.1</b>	<b>Connector pin assignment.....</b>	<b>35</b>
8.1.1	50-pin SUB-D front connector ST1 .....	35
8.1.2	Terminal KL1.....	38
<b>8.2</b>	<b>Connection of mass-related inputs and outputs .....</b>	<b>39</b>
<b>8.3</b>	<b>Connection of the differential I/O .....</b>	<b>40</b>
<b>9</b>	<b>FUNCTIONS OF THE BOARD .....</b>	<b>42</b>
<b>9.1</b>	<b>Description .....</b>	<b>42</b>
<b>9.2</b>	<b>Digital inputs and outputs.....</b>	<b>44</b>
9.2.1	Description .....	44
9.2.2	Inputs.....	46
9.2.3	Outputs.....	48
<b>9.3</b>	<b>PCI bus interface.....</b>	<b>49</b>
<b>10</b>	<b>STANDARD SOFTWARE .....</b>	<b>52</b>
<b>10.1</b>	<b>Introduction.....</b>	<b>52</b>
<b>10.2</b>	<b>Software functions.....</b>	<b>54</b>
10.2.1	Initialization .....	54
	1) I_APCI1710_InitCompiler (...).....	54
	2) I_APCI1710_CheckAndGetPCISlotNumber (...).....	56
	3) I_APCI1710_SetBoardInformation (...).....	57
	4) I_APCI1710_ConfigureAllModule .....	58
	5) I_APCI1710_GetHardwareInformation.....	59
	6) I_APCI1710_CloseBoardHandle (...) .....	60
10.2.2	Interrupt .....	61
	7) I_APCI1710_SetBoardRoutineDos (...).....	61
	8) i_APCI1710_SetBoardRoutingVBDOs (..) .....	63
	9) i_APCI1710_SetBoardIntRoutineWin16 (..).....	65
	10) i_APCI1710_SetBoardInRoutineWin32 (..) .....	67
	11) i_APCI1710_TestInterrupt.....	73
	12) i_APCI1710_ResetBoardIntRoutine (..) .....	74
10.2.3	Initialisation input filter .....	75
	13) i_APCI1710_InitInputFilter (..) .....	75
	14) i_APCI1710_CheckInputFilter40MHzStatus (..) .....	77

## Figures

Fig. 3-1: Correct handling .....	11
Fig. 4-1: Output current versus differential output voltage.....	15
Fig. 4-2: Component scheme .....	18
Fig. 6-1: Slot types for CompactPCI boards .....	22
Fig. 6-2: Inserting a board .....	23
Fig. 6-3: Fastening the board at the back cover .....	23
Fig. 7-1: Registration program ADDIREG.....	25
Fig. 7-2: Configuration of a new board.....	27
Fig. 7-3: PCI boards.....	28
Fig. 7-4: SET1710: Configuration program for modules.....	30
Fig. 7-5: Selection of a APCI-/CPCI-170.....	31
Fig. 7-6: General information .....	31
Fig. 7-7: Function list and module configuration.....	32
Fig. 7-8: Setting a module configuration by mouse click.....	33
Fig. 7-9: Setting a module configuration with the keyboard .....	34
Fig. 8-1: 50-pin SUB-D male connector (ST1) .....	35
Fig. 8-2: Terminal KL1 .....	38
Fig. 8-3: Connection of a mass-related input .....	39
Fig. 8-4: Connection of a mass-related output.....	39
Fig. 8-5: Connection of a differential input .....	40
Fig. 8-6: Example: Connection of 2 incremental encoders to FM1 .....	40
Fig. 8-7: Connection of a differential output.....	41
Fig. 9-1: Block diagram of the CPCI-1710.....	42
Fig. 9-2: Block diagram of the CPCI-1711 .....	43
Fig. 9-3: Block diagram of the digital inputs and outputs (1 function module) .....	45
Fig. 9-4: Basic circuit of the differential inputs (5 V).....	46
Fig. 9-5: Basic circuit of the differential inputs 5 V; used as TTL inputs.....	46
Fig. 9-6: Basic circuit of the 24 V differential inputs (OPTION).....	47
Fig. 9-7: Basic circuit of the digital inputs 24 V .....	47
Fig. 9-8: Basic circuit of the digital inputs 5 V (OPTION).....	47
Fig. 9-9: Basic circuit of the digital outputs 5V – differential.....	48
Fig. 9-10: Basic circuit of the digital output H – 24 V .....	48
Fig. 9-11: Basic circuit of a digital output H – 5 V (OPTION).....	49
Fig. 10-1: Synchronous and asynchronous mode.....	69
Fig. 10-2: Synchronous and asynchronous mode.....	69

## Tables

Table 5-1: Maximal input lines on the board .....	19
Table 5-2: Maximal input lines on a module .....	19
Table 5-3: Maximal input lines on the board .....	19
Table 5-4: Maximal input lines on a module .....	20
Table 5-5: Possible applications of the counter board .....	20
Table 5-6: Maximal application functions on the board .....	20
Table 5-7: Supplied function manuals .....	21
Table 8-1: Pin assignment for function module No. 1 (FM1) .....	36
Table 8-2: Pin assignment for function module No. 2 (FM2) .....	36
Table 8-3: Pin assignment for function module No. 3 (FM3) .....	37
Table 8-4: Pin assignment for function module No. 4 (FM4) .....	37
Table 8-5: Special pin assignment .....	37
Table 9-1: Assignment of the function modules .....	50
Table 9-2: I/O range .....	51
Table 10-1: Type declaration for DOS and Windows 3.1x .....	52
Table 10-2: Type declaration for Windows 95/NT .....	53
Table 10-3: Define value .....	53
Table 10-4: Filter time .....	76

# 1 DEFINITION OF APPLICATION

## 1.1 Intended use

The board **CPCI-1710** must be inserted in a CompactPCI system with PCI 5V/32-bit slots, which is used as electrical equipment for measurement, control and laboratory use as defined in the norm IEC 61010-1.

## 1.2 Usage restrictions

The board **CPCI-1710** must not to be used as safety related part for securing emergency stop functions.

The board **CPCI-1710** must not be used in potentially explosive atmospheres.

## 1.3 General description of the board

The board **CPCI-1710** is the interface between industrial process technology and a CompactPCI system.

Data exchange between the **CPCI-1710** board and the peripheral is to occur through a shielded cable. The cable ST370-16 must be connected to the 50-pin SUB-D male connector of the **CPCI-1710** board

The board has:

- 1. Up to 16 differential inputs** for 5 V signal acquisition.
  - The inputs (RS422) are intended to be connected to incremental encoders. They can also be used as TTL inputs.
  - **Option:** These inputs can also be adapted for processing 24 V signals.
- 2. Up to 12 mass-related inputs** which can be operated for any function provided they are used within the defined limit values.
  - **Option:** these inputs can also be adapted for processing 5 V signals.
- 3. Up to 12 outputs** for the emission of 5 V or 24 V signals. (In this case 20 inputs are available on the board.)
  - Up to 8 **differential outputs** (RS422) can be connected to SSI (Synchronous Serial Interface) encoders.
  - The other 4 **mass-related outputs** can be processed by any function provided they are used within the defined limit values.
  - **Option:** The 4 mass-related 24 V outputs can also be operated for the processing of 5 V TTL signals.

The **CPCI-1710** board consists of 4 “function modules”. Digital inputs and outputs are allocated to each function module. For each module 8 channels are available: (see block diagram of the inputs and outputs).

The channels are apportioned according to the board as follows:

**Input channels:**

- 2 x TTL, RS422 (signals C,D)



- 3 x 24 V, 5 V optional (signals E, F, G)  
**Output channels**
- 1 x 24 V, TTL optional (signal H)  
**Free definable channels (inputs and outputs)**
- 2 x TTL, RS422, (signals A, B)

The use of the board **CPCI-1710** in combination with external screw terminal panels or relay boards is to occur in a closed switch cabinet. **Check the shielding capacity** of the PC housing and of the cable prior to putting the device into operation.

The connection with our standard cable **ST370-16** complies with the following specifications:

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

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.

The use of the board according to its intended purpose includes observing all advises given in this manual and in the safety leaflet. Therefore, please

The use of the board in a CompactPCI system could change the PC features regarding noise emission and immunity. Increased noise emission or decreased noise immunity could result in the system is not being conform anymore.

The board must not be used in potentially explosive atmospheres. The board must not be exposed to vibrations without any additional keying.

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.

Observe also the regulations of the respective associations.

## **2 USER**

### **2.1 Qualification**

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

- Installation
- operation
- use
- maintenance

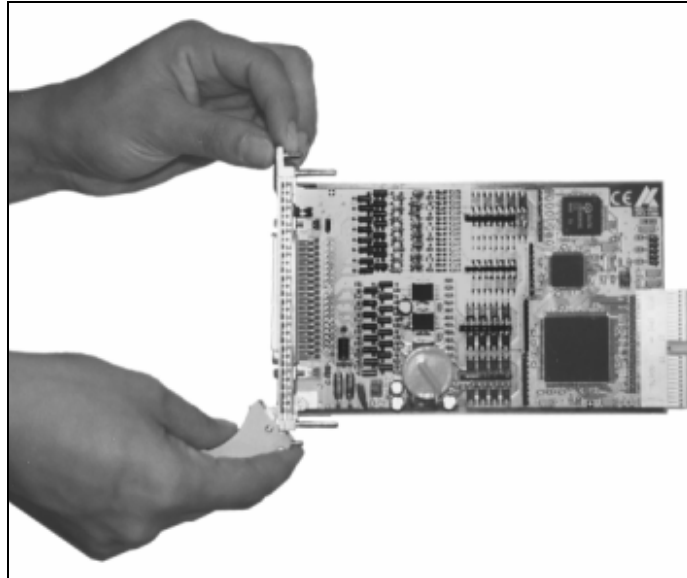
### **2.2 Personal protection**

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



## 4 TECHNICAL DATA

### 4.1 Electromagnetic compatibility (EMC)

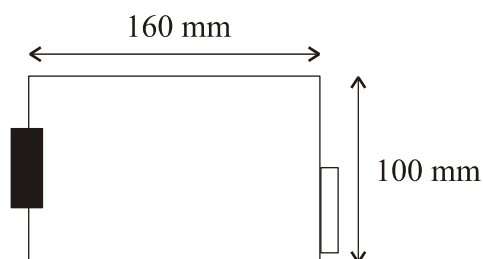
The CompactPCI system has been subjected to EMC tests in an accredited laboratory. The board complies with the limit values set by the norms IEC61326 as follows:

	True value	Set value
ESD (Discharge by contact/air) .....	4/8 kV	4/8 kV
Fields .....	10 V/m	10 V/m
Burst .....	4 kV	2 kV
Conducted radio interferences .....	10 V	10 V

### 4.2 Physical set-up of the board

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

#### Dimensions:



Weight: .....	approx. 200 g
Installation in: .....	32/64-bit CompactPCI slot, 5 V
Connection to the peripheral: .....	50-pin SUB-D male connector
Accessories <sup>1</sup> : .....	Standard cable <b>ST370-16</b> Screw terminal panel <b>PX8000</b>

<sup>1</sup> Not included in the standard delivery.

## 4.3 Versions

The board **CPCI-1710** is available in the following versions:

Version	Onboard Function modules	Option
<b>CPCI-1710</b>	4	5 V mass-related inputs 5 V digital inputs 24 V - 6U front bracket
<b>CPCI-1711</b>	2	5 V mass-related inputs 5 V digital inputs 24 V - 6U front bracket

## 4.4 Limit values

Max. altitude: ..... 2000 m  
 Operating temperature: ..... 0 to 60°C  
 Relative humidity: ..... 30% to 99% without condensing  
 Storage temperature: ..... -25 to 70°C

### Minimum PC requirements:

- 32-bit CompactPCI bus (5 V)
- PCI BIOS, PCI 2.1, specifications and CompactPCI 2.1 “compliant”
- 3 U format factor according to IEEE-1101

Bus speed: ..... < 33 MHz  
 Data bus access: ..... 32-bit  
 Decoding: ..... in the 64 K I/O area of the PC  
 “Target Only” operation  
 Operating system: ..... Windows NT 4.0, 9x, 2000, XP,  
 Linux

### Resources

I/O area  
 3 areas: ..... 64 bytes,  
 ..... 8 bytes,  
 ..... 256 bytes  
 IRQs: ..... INTA of the Compact PCI bus

### Energy requirements:

Operating voltage of the PC: ..... 5 V  $\pm$  5%  
 Current consumption (without load): ..... if + 5 V from PC: 877 mA ( $\pm$ 10 %)  
 ..... if + 24 V ext.: 10 mA ( $\pm$ 10 %)

### 4.4.1 Inputs

#### CPCI-1710

Number of inputs: .....	28
Input type:	
Differential inputs or rather TTL .....	16
24 V mass-related inputs .....	12

#### CPCI-1711

Number of inputs: .....	14
Input type:	
Differential inputs or rather TTL .....	8
24 V mass-related inputs .....	6

#### Differential inputs, 5 V

Fulfills the EIA standard RS485

Nominal voltage: .....	5 VDC
Common mode range: .....	+12/-7 V
Max. differential voltage: .....	$\pm 12$ V
Input sensitivity: .....	200 mV
Input hysteresis: .....	50 mV
Input impedance: .....	12 k $\Omega$
Load resistance (type): .....	150 $\Omega$ in series with 10 nF
Max input frequency: .....	1 MHz (at nominal voltage)
“Open Circuit Fail Safe Receiver Design” “1” = inputs open	
ESD protection .....	: up to $\pm 15$ kV

#### Mass-related inputs, 24 V (channels E, F, G)

Nominal voltage: .....	24 VDC
Input current (when nominal voltage): .....	11 mA
Logic input level: .....	U <sub>H</sub> max.: 30 V
	U <sub>H</sub> min.: 17 V
	U <sub>L</sub> max.: 15 V
	U <sub>L</sub> min.: 0 V
Signal delay: .....	120 ns (at nominal voltage)
Max. input frequency: .....	2.5 MHz (at nominal voltage)

#### Mass-related inputs, 5 V (channels E, F, G)

Nominal voltage: .....	5 VDC
Input current (when nominal voltage): .....	10 mA
Logic input level: .....	U <sub>H</sub> max.: 7 V
	U <sub>H</sub> min.: 2 V
	U <sub>L</sub> max.: 0.8 V
	U <sub>L</sub> min.: 0 V
Signal delay: .....	120 ns (at nominal voltage)
Max. input frequency: .....	5 MHz (at nominal voltage)

24 V option

**OPTION: Inputs, 24 V (channels A±, B±, C±, D±)**

The 24 V signals get a 5 V voltage through an adapter switch. According to the encoder the input switch is different.

The 24 V option offers a lot of application possibilities.

Our technical support will be pleased to answer your questions respecting these possibilities.

The differential inputs can be used either in the TTL mode or as 24 V mass-related inputs (OPTION). For this one input of the respective differential pair shall be laid onto the +UREF Pin 34 of the SUB-D connector.

- In the TTL compatible mode +UREF = +1.4 as switch wave.

- for the 24 V inputs is +UREF = +3.0 V as switch wave (OPTION).

Nominal voltage: ..... 24 VDC/11 mA

Logic input level: .....  $U_H$  max.<sup>1</sup>: 25 V

$U_H$  min.: 20 V

$U_L$  max.<sup>2</sup>: 17 V

$U_L$  min.: 0 V

Max. input frequency: ..... 1 MHz (at nominal voltage)

4.4.2 Outputs

**Differential outputs, 5 V**

Fulfil the EIA standard RS485

Nominal voltage: ..... 5 VDC

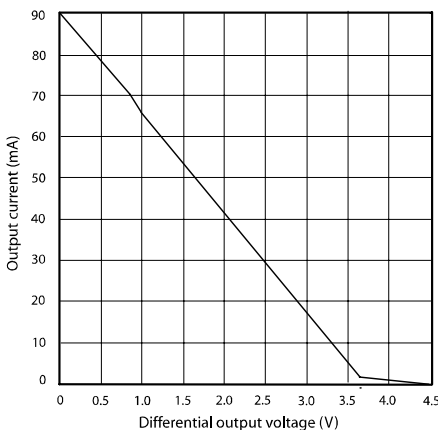
Max. output frequency: ..... 5 MHz

Max. number of outputs:

**CPCI-1710:** ..... 8 (if they are not reserved as differential inputs)

**CPCI-1711:** ..... 4 (if they are not reserved as differential inputs)

**Fig. 4-1**Output current versus differential output voltage



<sup>1</sup>  $U_H$ : Input voltage that corresponds with logic “1”

<sup>2</sup>  $U_L$ : Input voltage that corresponds with logic “0”

### Digital outputs, 24 V

Output type:.....High-Side (load on ground)

Number of outputs: ..... 4 for the CPCI-1710  
  2 for the CPCI-1711

Nominal voltage:.....24 VDC

Range of distribution voltage: 10 V to 36 VDC (over 24 V ext. pin)

Max. output current for the 4 outputs: 2 A type. (to be limited at the distribution voltage)

Max. output current:..... 500 mA

Short circuit current / output at 24 V,

R<sub>last</sub> < 0.1 R: ..... 1.5 A max. (output switches off)

ON resistance of the output

(R<sub>DS</sub> ON resistance): ..... 0.4 R max.

Excess temperature: ..... 170 °C (all outputs switch off)

**CPCI-1710, 24 V**

The inputs/outputs A to B can be used only as inputs, i.e. that the function PWM and digital I/O (not all as inputs) cannot be used with this board.

### Protection against overtemperature (24 V outputs)

Activation from: ..... approx. 150-170 °C (chip temperature)

Deactivation (automatically) from: .....approx. 125-140 °C (chip temperature)

Outputs (at overtemperature): .....outputs switch off

**Undervoltage** (active when  $V_{\text{ext}} < 5 \text{ V}$ )

Outputs (when undervoltage):.....all outputs switch off.

### Switch characteristics of the outputs

( $V_{\text{ext}} = 24 \text{ V}$ ,  $T = 25 \text{ }^{\circ}\text{C}$ , ohmic resistance: 500 m $\Omega$ )

Switch ON time:.....200  $\mu$ s

Switch OFF time: ..... 15  $\mu$ s

### Digital outputs, 5 V (OPTION)

Output type:.....TTL

[illegible]

Nominal voltage:.....5 VDC

### Switch characteristics of the outputs (T=25 °C, TTL load)

Switch ON time:.....0.06  $\mu$ s

Switch OFF time: ..... 0.02  $\mu$ s



#### 4.4.3 Safety

Optical isolation

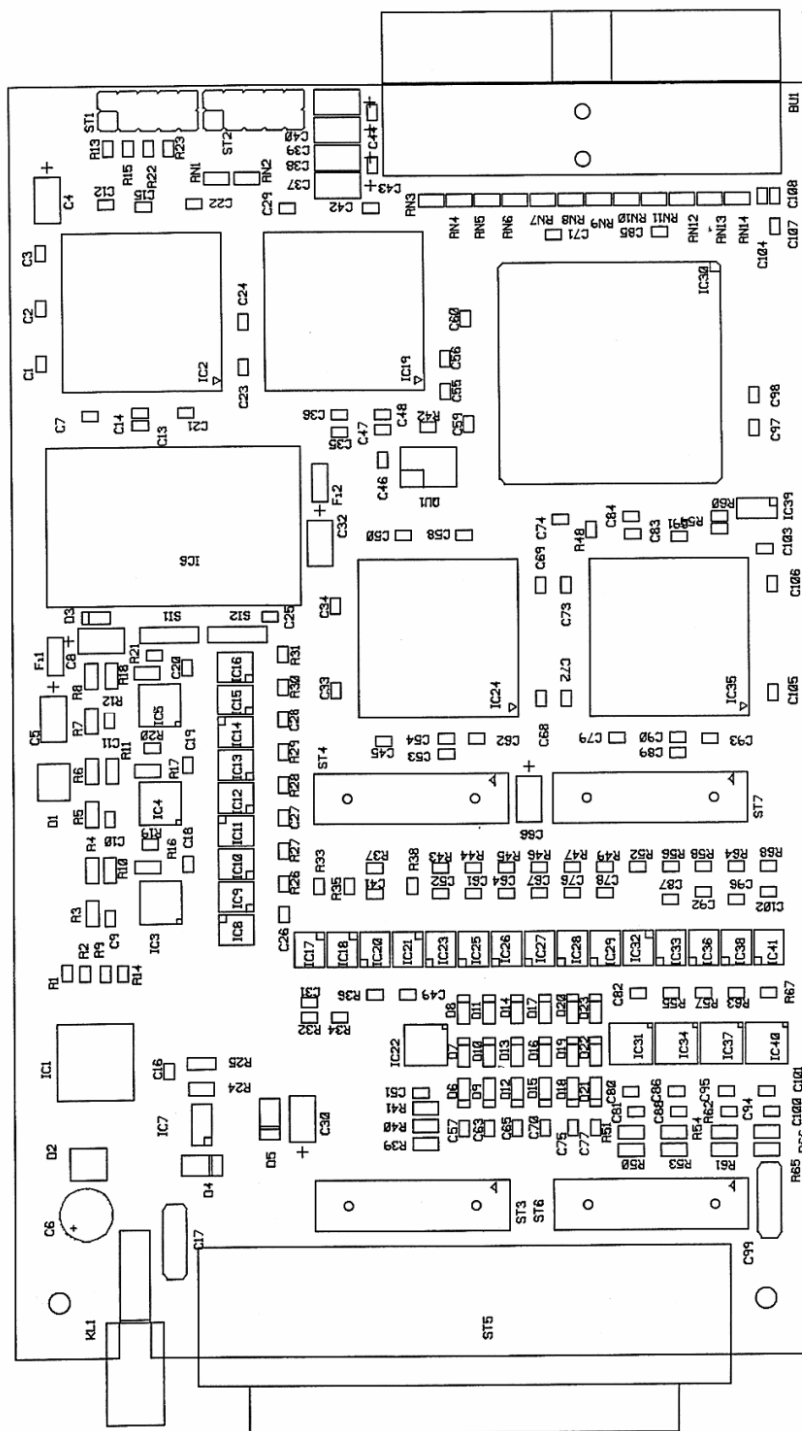
(DIN VDE 0411-100): ..... 1000 V (from the PC to  
external periphery)

Logic: ..... positive

Creeping distance: ..... 3.2 mm on the printed board

## 4.5 Component scheme

Fig. 4-2: Component scheme



## 5 AVAILABLE FUNCTIONS OF THE CPCI-1710

### 5.1 Available signals

#### 5.1.1 Connectable signal lines

The lines are divided according to the board as follows:

**Input lines:**

- 2 x TTL, RS422 (signals C, D)
- 3 x 24 V, 5 V optional (signals E, F, G)

**Output lines:**

- 1 x 24 V, TTL optional (signal H)

**Free definable lines (input or output):**

- 2 x TTL, RS422 (signals A, B)

#### 5.1.2 Maximal signal wirings of the CPCI-1710



**IMPORTANT!**

The CPCI-1711 has always only half the number of signals.

**Table 5-1: Maximal input lines on the board**

Inputs	Available outputs
28	4 (H)
of them are - 16 differential inputs and - 12 x 24 V inputs	

**Table 5-2: Maximal input lines on a module**

Inputs	Available outputs
7	1
of them are - 4 differential - 3 x 24 V inputs	

**Table 5-3: Maximal input lines on the board**

Outputs	Available inputs
12	20
<b>Of them are</b> - 8 differential outputs and - 4 x 24 V outputs	<b>Of them are</b> 8 differential 12 x 24 V inputs (E,F,G)

Table 5-4: Maximal input lines on a module

Outputs	Available inputs
3 output lines	1
Of them are - 2 differential and - 1 x 24 V output	

## 5.2 Available functions

### 5.2.1 Programmable functions of the counter board

Table 5-5: Possible applications of the counter board

Function	Reserved input channels	Reserved output channels
Incremental counter	7 inputs (A to G)	1 output (H)
SSI	6 inputs (B to G)	2 outputs (A, H)
Chronos	5 inputs (C to G)	3 outputs (A, B, H)
Counter/Timer	5 inputs (C to G)	3 outputs (A, B, H)
Digital I/O	7 inputs (A to G)	1 output (H)
or	5 inputs (C to G)	3 outputs (A, B, H)
PWM	5 inputs (C to G)	3 outputs (A, B, H)
TOR	2 inputs (C, D)	2 outputs (A, B)
Pulse counter	4 inputs	1 output (H)
ETM	4 inputs (A, B, C, D)	

### 5.2.2 Connection facilities depending on the chosen function

Table 5-6: Maximal application functions on the board

Application	On the board	Per function module	Programmed function
Incremental encoder	4 (32-bit counting depth) 8 (16-bit counting depth)	1 (32-bit) 2 (16-bit)	Incremental counter
Absolute encoder	12	3	SSI
Pulse counter	16	4	Pulse counter
Frequency, duty cycle determination	4	1	Chronos
Frequency output trigger control	12	3	Counter / Timer
PWM	8	2	PWM
Gate measurement	8	2	TOR
Edge Time Measurement	8	2	ETM

### 5.2.3 Delivered manuals

According to the function used you can find the required reserve and programming information in the respecting manuals.

**Table 5-7: Supplied function manuals**

Function	PDF file (CD2 technical manuals)		Function description in SET1710	CFG file
	German	English		
Incremental counter	Inkr_zähler_d.pdf	Incr_counter_e.pdf	Incremental counter	inc_cpt.cfg
SSI	SSI_d.pdf	ssi_e.pdf	SSI	ssi.cfg
SSI monitor	SSI-Monitor_d	SSIMonitor_e.pdf	SSI Monitor	ssi_mon.cfg
Chronos	chronos_d.pdf	chronos_e.pdf	Chronos	chronos.cfg
Counter/timer	Zähler_timer_d.pdf	Counter_timer_e.pdf	counter/timer	82x54.cfg
TOR	TOR_d.pdf	TOR_e.pdf	TOR	tor.cfg
PWM	PWM_d.pdf	PWM_e.pdf	Pulse width modulation	PWM.cfg
TTL	TTL_IO_d.pdf	TTL_IO_e.pdf	TTL I/O	ttl_io.cfg
Digital I/O	dig_EA_d.pdf	dig_IO_e.pdf	Digital I/O	dig_IO.cfg
Pulse counter	Impulszähler_d.pdf	pulseCounter_e.pdf	Pulse counter	imp_cpt.cfg
ETM (Edge time measurement)	ETM_d.pdf	ETM_e.pf	Edge time measurement	etm.cfg

## 6 INSTALLATION OF THE BOARD



### IMPORTANT!

Do observe the safety precautions!

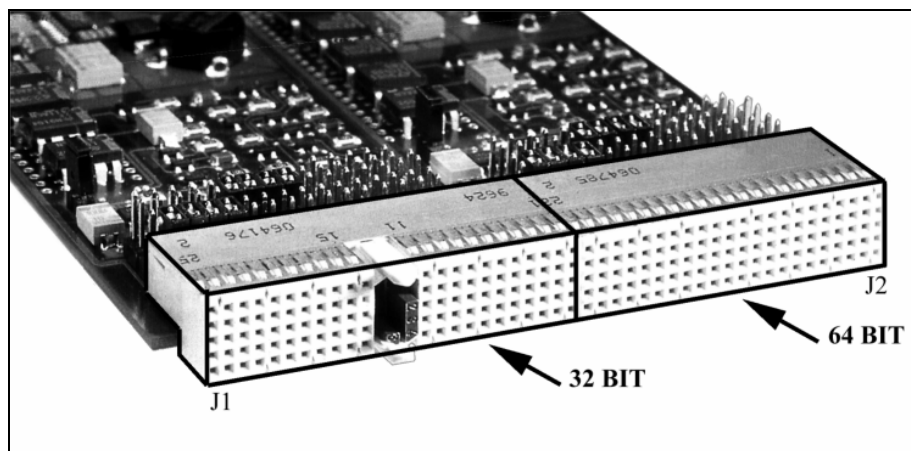
### 6.1 Opening the CompactPCI system

- ◆ Switch off your system and all the units connected to the system
- ◆ Pull the CompactPCI system's mains plug from the socket.
- ◆ Remove the front bracket of a free CompactPCI slot
- ◆ Take the board out of its protective pack

### 6.2 Installation

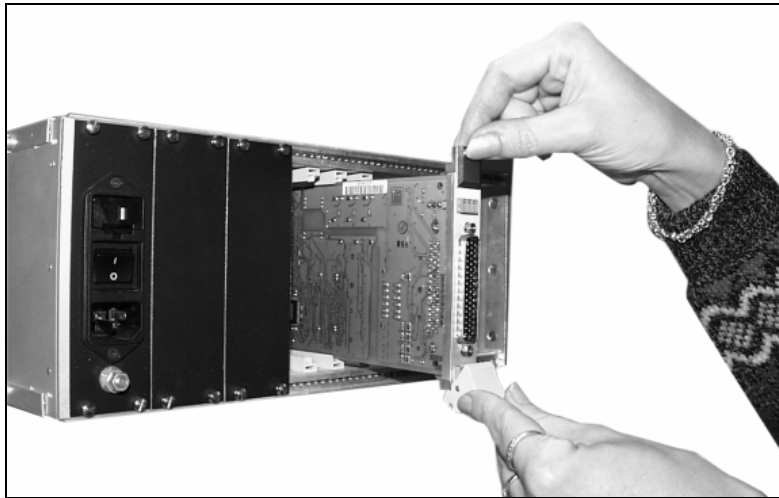
There are various slot types of CompactPCIs for 5 V systems available: CompactPCI-5V (32-bit) and CompactPCI-5V (64-bit).

Fig. 6-1: Slot types for CompactPCI boards



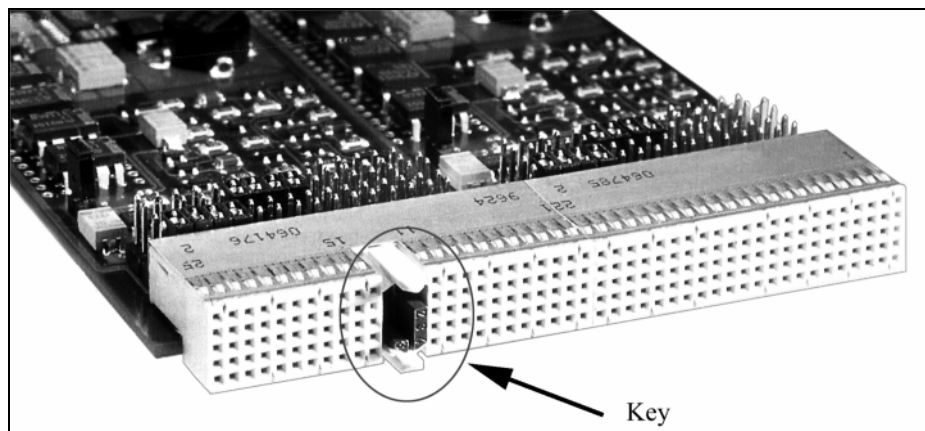
- ◆ Discharge yourself.
- ◆ Keep the board at the grip (see chapter 3)
- ◆ Insert the board **into the guide bar of the base and shift it up to the back wall of the housing.** In order to insert it completely, a slight resistance must be noticed.

Fig. 6-2: Inserting a board



- ◆ Please check that the board is inserted correctly: The coding key of the Compact PCI slot on the backplane shall fit with the coding key of the board (blue coding key if the board is operated with 5 V).

Fig. 6-3: Fastening the board at the back cover



- ◆ Fix the board at the upper end of the front plane with screw, if available.

**Note:**

To withdraw the board, pull it to the front with the grip. If it is a foldable grip, you firstly shall push it slightly upwards.

## 7 SOFTWARE

### 7.1 Delivered software

The board is delivered with a driver CD (CD 1).

The CD contains:

- ADDIREG for Windows NT 4.0 and Windows 2000/95/98
- Standard software for the ADDI-DATA boards:  
SET1710 program for the configuration of the function modules
- All functions that are implemented for the **CPCI-1710**.

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



#### **IMPORTANT!**

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

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

### 7.2 Configuration of the CPCI-1710 with ADDIREG

The ADDIREG registration program is a 32-bit program. With this program the user can register all hardware information that is required for the operation of the ADDI-DATA PC boards.



#### **IMPORTANT!**

First, install the ADDI-DATA board that you want to register before you call up the ADDIREG program.

If the board is not installed in the PC, the registration can not be controlled. When calling up the program, the screen displays the following window.



Fig. 7-1: Registration program ADDIREG

ADDI-DATA GmbH registration program. Version 0302 / 0546

Resource file System info About

**Board list configuration**

Board name	Base address	Access	PCI bus/device/(slot)	Interrupt	DMA	More information
APCI1516	D480.DC78, DC40	32-bit	2/9/3	Not available	Not available	ADDIDriver board
APCI1710	D800.DC70	32-bit	2/8/2	17	Not available	

Insert Edit Clear

**Board configuration**

Base address name : Interrupt name: DMA name: Set Cancel

Base address : Interrupt : DMA channel : Default More information

Access mode: ADDIDriver board manager

Save Restore Test registration Deinstall registration Print registration Quit

ADDI-DATA

**Table:**

The mid table shows the registered boards and parameters.

**Board name:**

The names of the different registered boards are shown (for example APCI-7800). When you use the program for the first time, no board appears under this entry.

**Base address:**

Chosen base address of the board.

**Access:**

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

**PCI slot:**

Used PCI slot. If the board is no APCI board appears "NO".

**Interrupt:**

Used interrupt of the board. If the board do not have an interrupt, appears "Not available".

**DMA:**

Shows the selected DMA channel or “Not available” if the board do not use one.

**More information:**

Further information is shown by the dialog box, for example the sign chain for the identifier (e.g. PCI1500-50) or the installed COM interfaces. If the board is programmed with ADDIDRIVER this will be shown.

**Text boxes:**

In this table you can find 6 boxes for text input with which you can modify the board's parameter.

**Base address name:**

If the board is operated by various base addresses (one for the first interface, one for the second interface etc.) you can decide which base address is to be changed.

**Base address:**

In this box you can select your PC-board's base address. All vacant base addresses are listed. A base address that is already used is not displayed in this entry.

**Interrupt name:**

Selection of the interrupt number that shall be used by the board.

**DMA name**

When the board supports 2 DMA channels, you can select which DMA channel you want to change.

**DMA channel:**

Selection of the used DMA channel.

**Buttons:****Edit:**

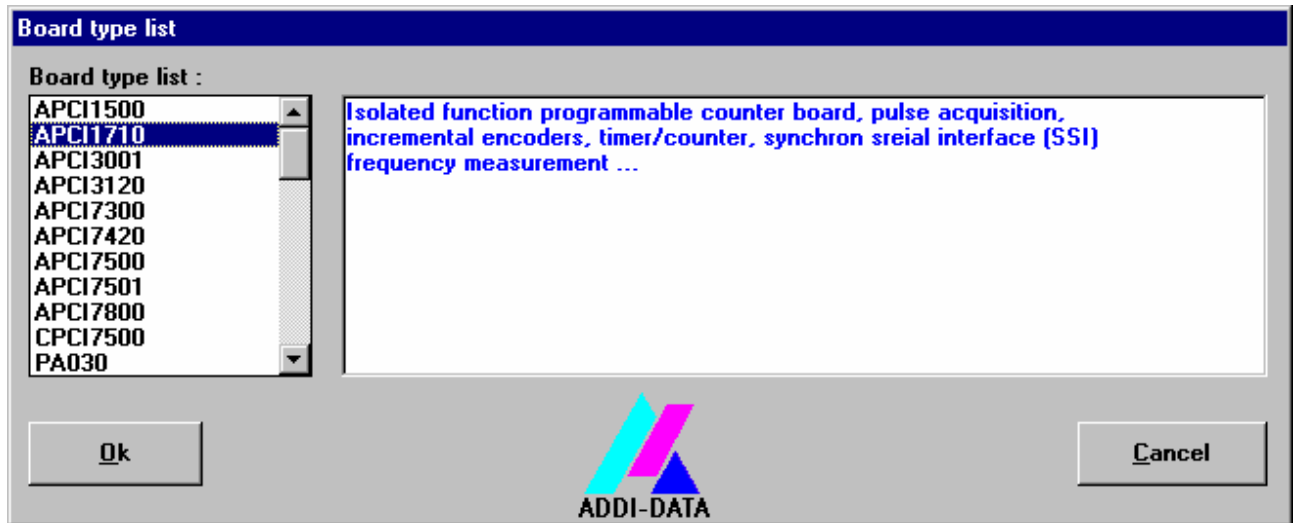
Selection of the highlighted board with the different parameters set in the text boxes.

Click on Edit to confirm the entries or click twice on the selected board. Sets the parameterized board configuration. The configuration should be set before you save it.

**Insert:**

If you want to insert a new board click on “Insert. Then the following box is displayed:

Fig. 7-2: Configuration of a new board



On the left side you can see all the boards that you can register. Please click on the selected board (the respecting line will be highlighted).

On the right side of the box you can see some technical specifications about the board.

Confirm with “OK”. Then you will return to the first screen.

**Clear:**

You can delete the registration of your board. Highlight the board to be deleted and click on “Clear”.

**Set:**

Sets the parameterized board configuration. The configuration shall be set before you save it.

**Cancel:**

Resets the modified parameter to the currently saved configuration. parameterized board configuration.

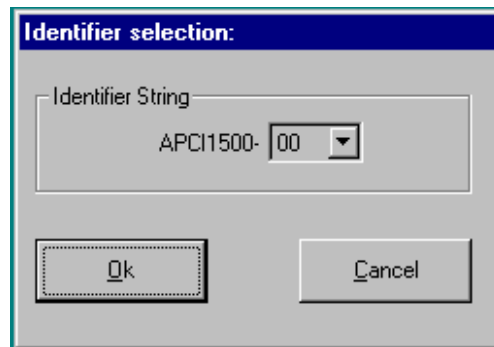
**Default:**

Sets the board’s standard parameter.

**More information**

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

Fig. 7-3: PCI boards



With this option you can select the identifier for the string by specifying the number and confirming with “OK”.

With clicking on “Cancel” you choose the old string.

#### **ADDIDriver Board Manager:**

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

ADDevice Manager starts and displays a list of all resources available of the virtual board. As the **CPCI-1710** is not programmed with ADDIPACK, this button is deactivated.

#### **Save:**

Saves the parameter and registers the board.

#### **Default:**

Reactivation of the parameter and registration that were saved at last.

#### **Test registration:**

Checks if there is a conflict between the board and other devices. A message indicates the parameter that has generated the conflict. If there is no conflict, "OK" is displayed.

#### **Deinstall registration:**

Deinstalls all registrations of all boards listed in the table.

#### **Print registration:**

Prints the registration parameter on your standard printer.

#### **Quit:**

Quits the ADDIREG program

### 7.2.1 Registration of a new board



#### **IMPORTANT!**

For the registration of a new board you need administrator right. Only an administrator is allowed to record a new board or

modify an already existing registration.

◆ **Call up the ADDIREG program.**

Figure 7-1 will be displayed. Click on “Insert” and then select the required board.

◆ **Click on “OK”.**

The Default address, Interrupt and the other parameters will be set automatically. The parameters will be listed in the lower areas. If the parameters are not set automatically by BIOS you can modify them. Click on the required scroll function/functions and select a new value. Confirm it with a click.

◆ **When the required is set, click on “Set”**

◆ **Save the configuration with “Save”.**

You can control if the registration is right by conducting a test. If the test result is positive you can leave the ADDIREG program. Then the board will be initialized with the set parameters and can be operated. You will be requested to restart your PC. Otherwise, the parameters will be saved in files so that there is no need to restart the PC.

## 7.2.2 Changing the registration of an existing board



### **IMPORTANT!**

In order to register a new board, administrator rights are required. Only an administrator is allowed to register a new board or to change an already existing registration.

◆ **Call the ADDIREG program. Mark the board whose parameters you want to change.**

The board's parameters (base address, DMA, channel etc) are listed in the lower areas.

◆ **Click on the parameters that you want to set newly and open the scrolling function.**

◆ **Select a new value. Confirm it with a click. Please repeat this for each parameter to be changed.**

◆ **When the required configuration is set, click on “Set”.**

◆ **Save the configuration with “Save”.**

## 7.3 Loading a function into a function module

The SET1710 configuration program is a 16-bit or 32-bit program for Windows 3.11 and Windows XP/NT/2000/98. It will be installed automatically when installing the driver in 32-bit (Windows XP/NT/2000/98).

At delivery time all function modules are preset with the function “**incremental counter**”.

The functions are programmed **once** and **separate** for each function module through “**SET1710.exe**”, which is delivered with the board. After the PC’s new start the board **CPCI-1710** is operable.

The user also can replace the configuration of the modules through the software (see software function *i\_APCI1710\_ConfigureAllModule* in chapter 9).

### 7.3.1 Module configuration with SET1710

**i**

#### IMPORTANT!

Firstly, set the board with the registration program ADDIREG under Windows XP/NT/2000/98 before configuring the function modules with the program SET1710.

Fig.7-4: SET1710: Configuration program for modules

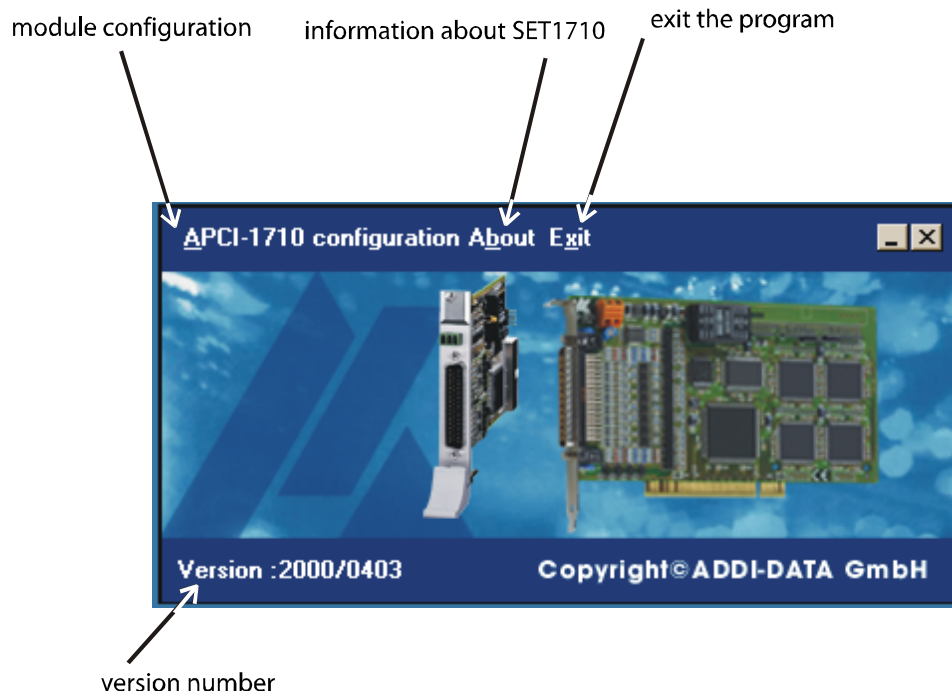


Fig. 7-5: Selection of a APCI-/CPCI-170



Under “About” you find general information about the board and our customer service.

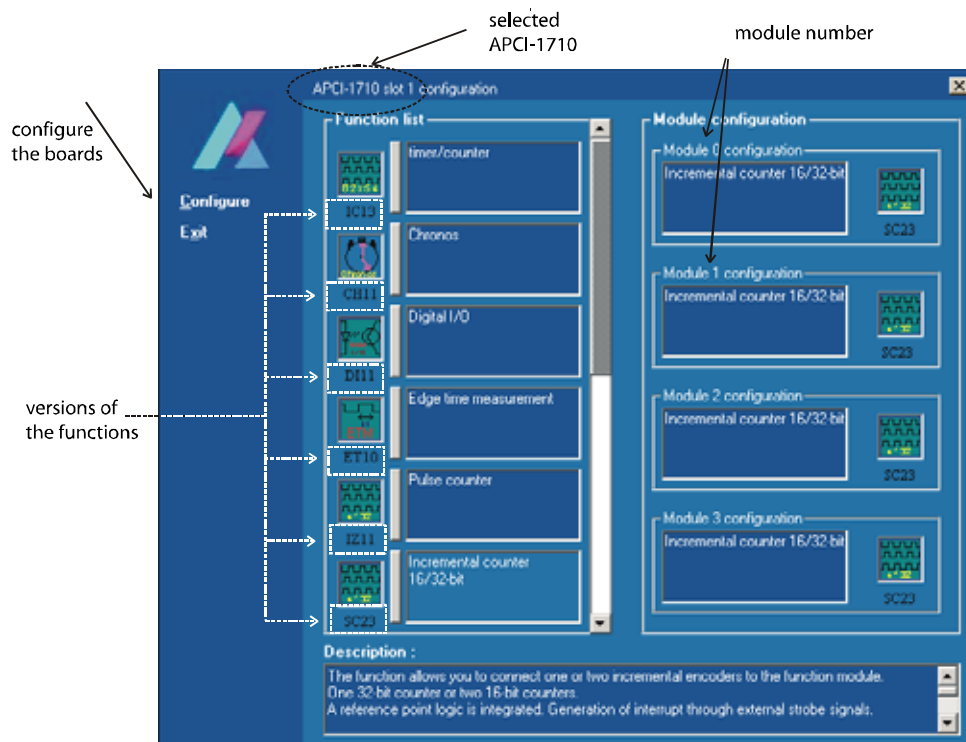
Fig. 7-6: General information



With “Exit” you leave the SET1710 program.

After having selected the **CPCI-1710** the following box is displayed:

Fig. 7-7: Function list and module configuration

**Function list:**

In this box all available module functions are listed. Each function will be set in a configuration file into the directory CFG.

**Module configuration:**

Current module configuration of the selected board

**Description:**

Description of the functions that are selected in the list. The version of the function, the update description and any other information about the module function is listed in this field.

**Configure:**

Programs the APCI-/CPCI-1710 with the current module configuration.

**Exit:**

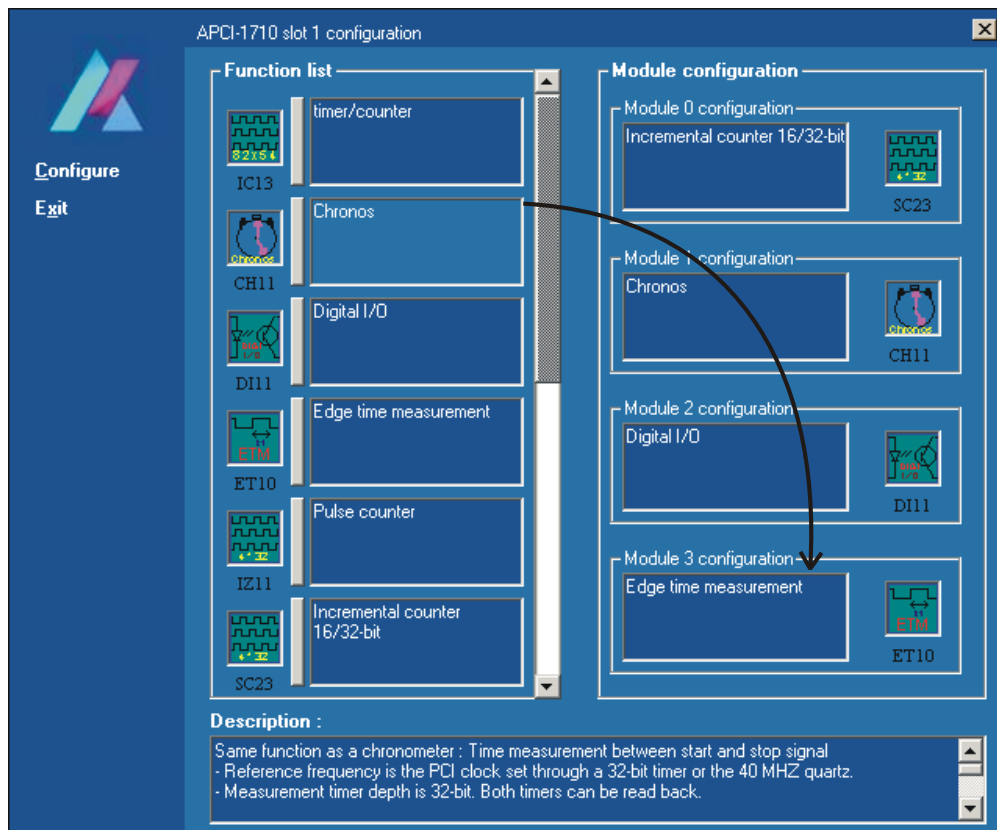
Closes the box.



### 7.3.2 Setting a module configuration

#### Module configuration through mouse click

Fig.7-8: Setting a module configuration by mouse click

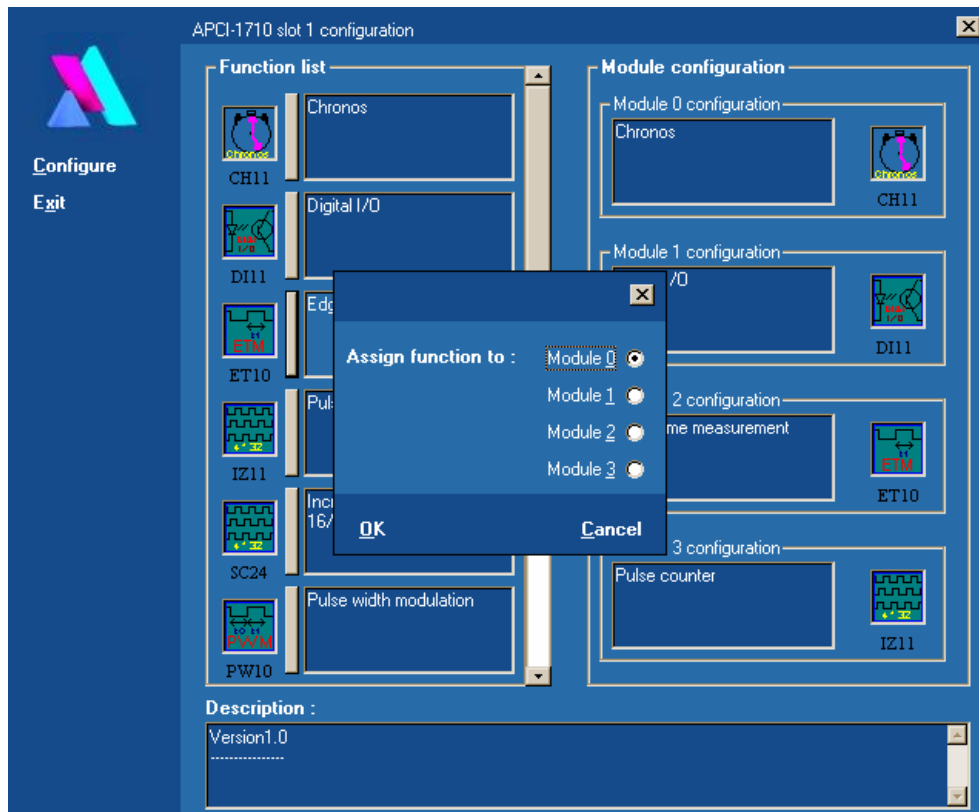


Click with the right mouse button on the function in the list and move it into the required module.

#### Module configuration through keyboard

Select the function in the list (tabulator). Then press the “Return” tab. The following dialogue box will be displayed:

Fig. 7-9: Setting a module configuration with the keyboard



Confirm the required module with “OK”.

## 7.4 ADDI-DATA on the Internet

Do not hesitate to e-mail us your questions.

E-mail: **info@addi-data.de** or **hotline@addi-data.de**

### Free downloads of standard software

You can download the driver for your board from the internet:

[www.addi-data.com](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.

## 8 CONNECTING THE PERIPHERAL



### IMPORTANT!

Interferences are emitted and inserted through the connection cable. Therefore, a wrong cable may endanger the operation and function safety of your system.

**Use our standard connection cable.**

**During placing the connection cable observe the following:**

- There shall be sufficient distance to sensitive analog signals
- The distance to potential sources of interference like frequency converter, mains supply circuit, shall be as long as possible.

If you operate the outputs with maximum load, you shall place the connection cable freely and well ventilated.

### 8.1 Connector pin assignment



### IMPORTANT!

The function modules are described with different definitions in the hardware and software descriptions.

For the connector pin assignment (*hardware*) the modules 1 to 4 are numbered. For the SET1710 program or the software function (*Software*) the module numbering **BEGINS** with 0.

#### 8.1.1 50-pin SUB-D front connector ST1

Fig. 8-1: 50-pin SUB-D male connector (ST1)

Pin		Pin				Pin	
34	Output voltage/ 24 V voltage supply			34	18	1	ext. GND for all inputs and outputs
35	FM 1 output	18	input or output	35		2	input or output
36	FM 2 output	19	input or output	36		3	input or output
37	FM 3 output	20	input or output	37		4	input or output
38	FM 4 output	21	input or output	38		5	input or output
39	FM 1 input	22	input	39		6	input
40	FM 2 input	23	input	40		7	input
41	FM 3 input	24	input	41		8	input
42	FM 4 input	25	input	42		9	input
43	FM 1 input	26	input or output	43		10	input or output
44	FM 2 input	27	input or output	44		11	input or output
45	FM 3 input	28	input or output	45		12	input or output
46	FM 4 input	29	input or output	46		13	input or output
47	FM 1 input	30	input	47		14	input
48	FM 2 input	31	input	48		15	input
49	FM 3 input	32	input	49		16	input
50	FM 4 input	33	input	50	33	17	input



### IMPORTANT!

The CPCI-1711 assigns the signals of the modules FM1 and FM2.

The following tables show the pin assignment of the digital inputs and outputs and the respecting signals and functions. The inputs and outputs shall be switched on the module after the programming of the function (direction change-over depends on the selected function).

**Table 8-1: Pin assignment for function module No. 1 (FM1)**

Pin	Description	Input/Output	Signal type	Option
2	A1+	Input/Output	Diff. / TTL	24 V input
3	A1-	Input/Output	Diff. / TTL	- <sup>1</sup>
4	B1+	Input/Output	Diff. / TTL	24 V input
5	B1-	Input/Output	Diff. / TTL	-
6	C1+	Input	Diff. / TTL	24 V
7	C1-	Input	Diff. / TTL	-
8	D1+	Input	Diff. / TTL	24 V
9	D1-	Input	Diff. / TTL	-
35	H1	Digital Output	24 V	24 V
39	E1	Digital Input	24 V	24 V
43	F1	Digital Input	24 V	24 V
47	G1	Digital Input	24 V	24 V

**Table 8-2: Pin assignment for function module No. 2 (FM2)**

Pin	Description	Input/Output	Signal type	Option
10	A2+	Input/Output	Diff. / TTL	24 V input
11	A2-	Input/Output	Diff. / TTL	-
12	B2+	Input/Output	Diff. / TTL	24 V input
13	B2-	Input/Output	Diff. / TTL	-
14	C2+	Input	Diff. / TTL	24 V
15	C2-	Input	Diff. / TTL	-
16	D2+	Input	Diff. / TTL	24 V
17	D2-	Input	Diff. / TTL	-
36	H2	Digital Output	24 V	24 V
40	E2	Digital Input	24 V	24 V
44	F2	Digital Input	24 V	24 V
48	G2	Digital Input	24 V	24 V

<sup>1</sup> The Ax-, Bx-, Cx-, Dx- pins have no function at the 24 V board, see Fig. 9-6: Basic circuit of the 24 V differential inputs (OPTION)

**Table 8-3: Pin assignment for function module No. 3 (FM3)**

Pin	Description	Input/Output	Signal type	Option
18	A3+	Input/Output	Diff. / TTL	24 V input
19	A3-	Input/Output	Diff. / TTL	-
20	B3+	Input/Output	Diff. / TTL	24 V input
21	B3-	Input/Output	Diff. / TTL	-
22	C3+	Input	Diff. / TTL	24 V
23	C3-	Input	Diff. / TTL	-
24	D3+	Input	Diff. / TTL	24 V
25	D3-	Input	Diff. / TTL	-
37	H3	Digital Output	24 V	24 V
41	E3	Digital Input	24 V	24 V
45	F3	Digital Input	24 V	24 V
49	G3	Digital Input	24 V	24 V

**Table 8-4: Pin assignment for function module No. 4 (FM4)**

Pin	Description	Input/Output	Signal type	Option
26	A4+	Input/Output	Diff. / TTL	24 V input
27	A4-	Input/Output	Diff. / TTL	-
28	B4+	Input/Output	Diff. / TTL	24 V input
29	B4-	Input/Output	Diff. / TTL	-
30	C4+	Input	Diff. / TTL	24 V
31	C4-	Input	Diff. / TTL	-
32	D4+	Input	Diff. / TTL	24 V
33	D4-	Input	Diff. / TTL	-
38	H4	Digital Output	24 V	24 V
42	E4	Digital Input	24 V	24 V
46	F4	Digital Input	24 V	24 V
50	G4	Digital Input	24 V	24 V

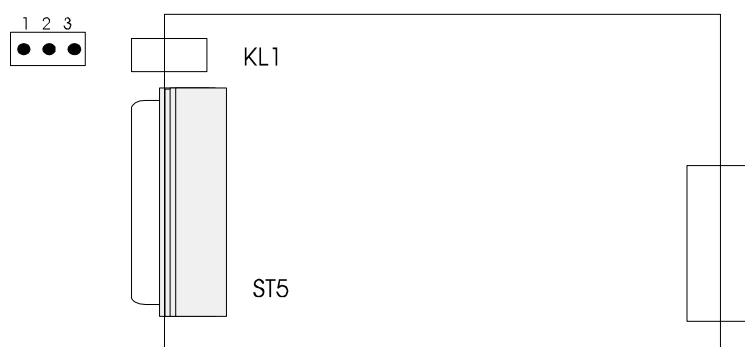
**Table 8-5: Special pin assignment**

Pins with fix assignment

Pin	Description	Function	Note
1	EXTGND	Reference potential	For all inputs and outputs
34	+ UREF	Voltage output for TTL signals	Reference voltage for the connection of TTL signals

### 8.1.2 Terminal KL1

Fig. 8-2: Terminal KL1



Pin	Description	Function	Note
1	24 V supply	Voltage supply of the 24 V outputs (channels Hx)	
2	EXTGND	Reference potential	Connected with Pin 1 of the SUB-D connector
3	5 V ext.	Supply of external sensors, max. 200 mA	

## 8.2 Connection of mass-related inputs and outputs

Fig. 8-3: Connection of a mass-related input

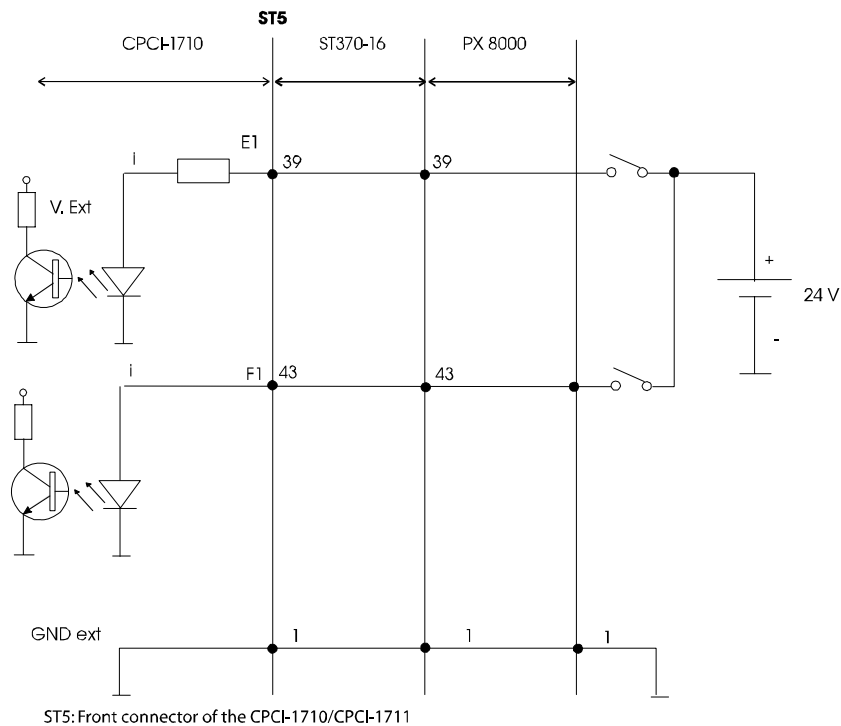
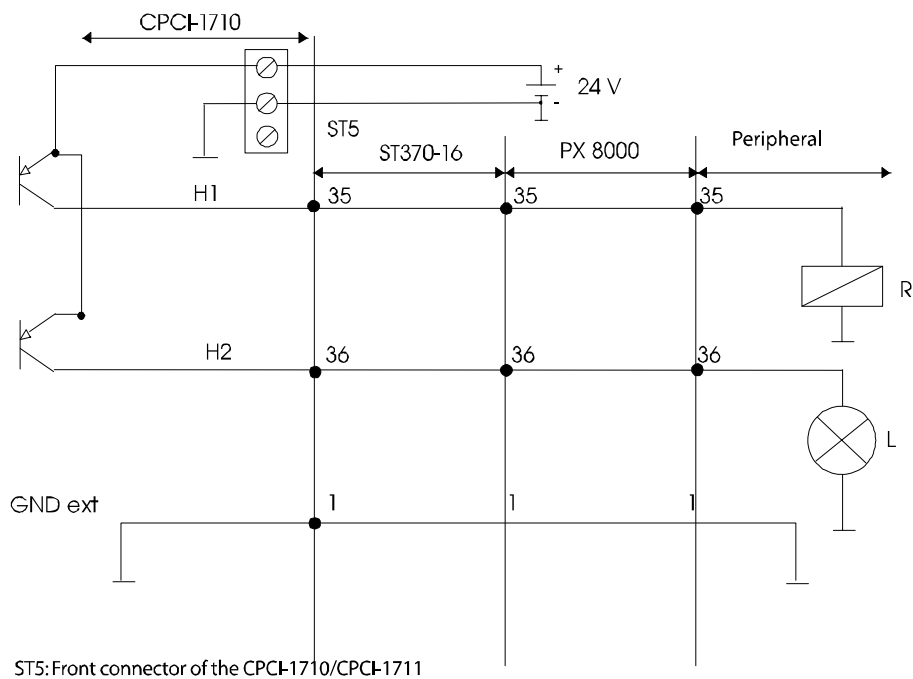


Fig. 8-4: Connection of a mass-related output



### 8.3 Connection of the differential I/O

Fig. 8-5: Connection of a differential input

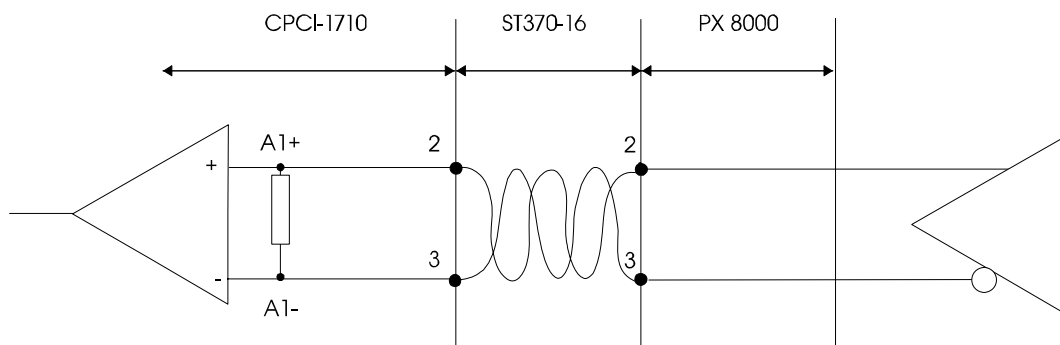


Fig. 8-6: Example: Connection of 2 incremental encoders to FM1

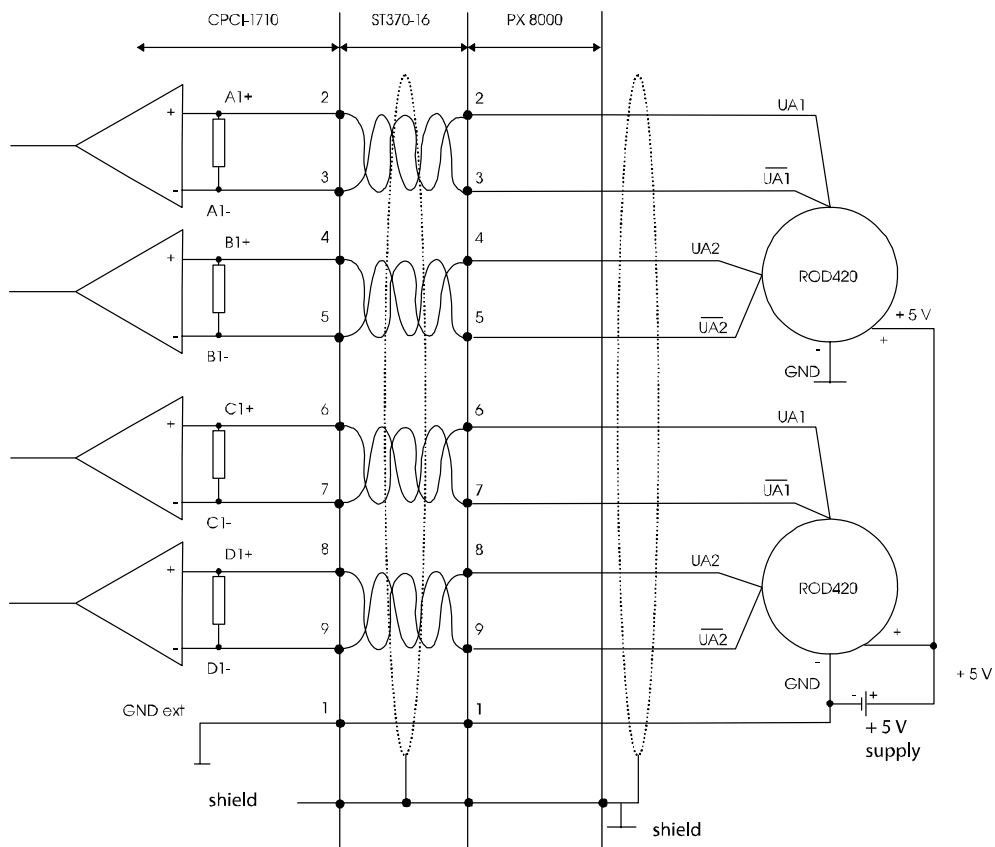
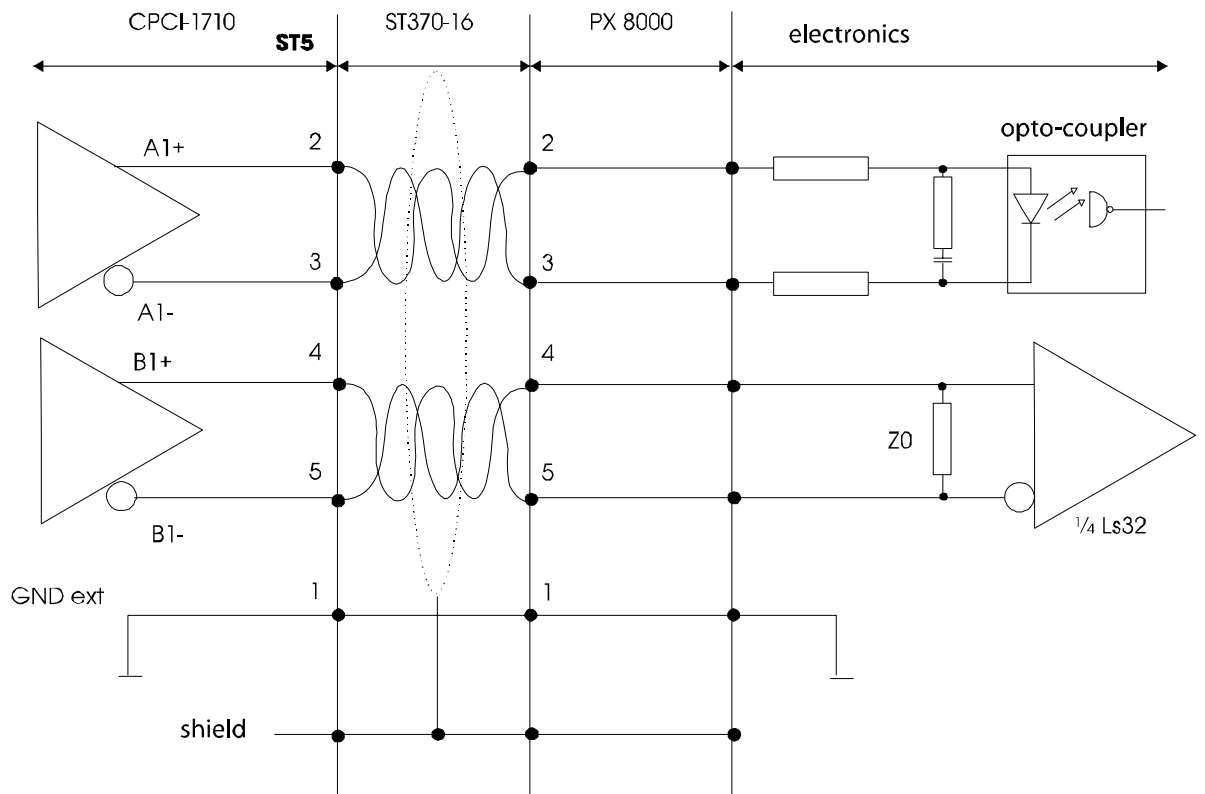




Fig. 8-7: Connection of a differential output



## 9 FUNCTIONS OF THE BOARD

### 9.1 Description

The **CPCI-1710** is an extension board for the CompactPCI bus and it is compatible with the PCI specification 2.1. This type of board is mainly considered for the processing of digital signals with focus on “counting and time measurement”.

The digital signals are connected through a 50 pin SUB-D front connector ST5 to the “function modules” of the **CPCI-1710** board. They are separated optically through the opto-coupler.

The **CPCI-1710** board consists of 4 “function modules” that are again assembled with 4 programmable CPLDs (Complex Programmable Logic Devices). Digital inputs and outputs are allocated firmly to each function module.

**Fig. 9-1: Block diagram of the CPCI-1710**

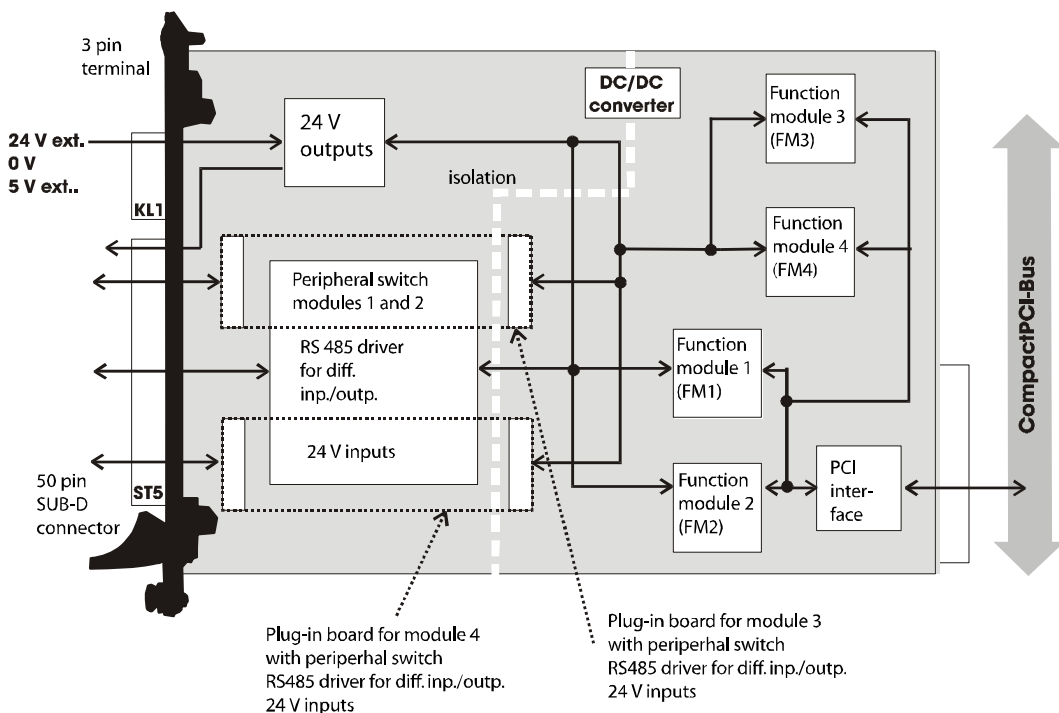
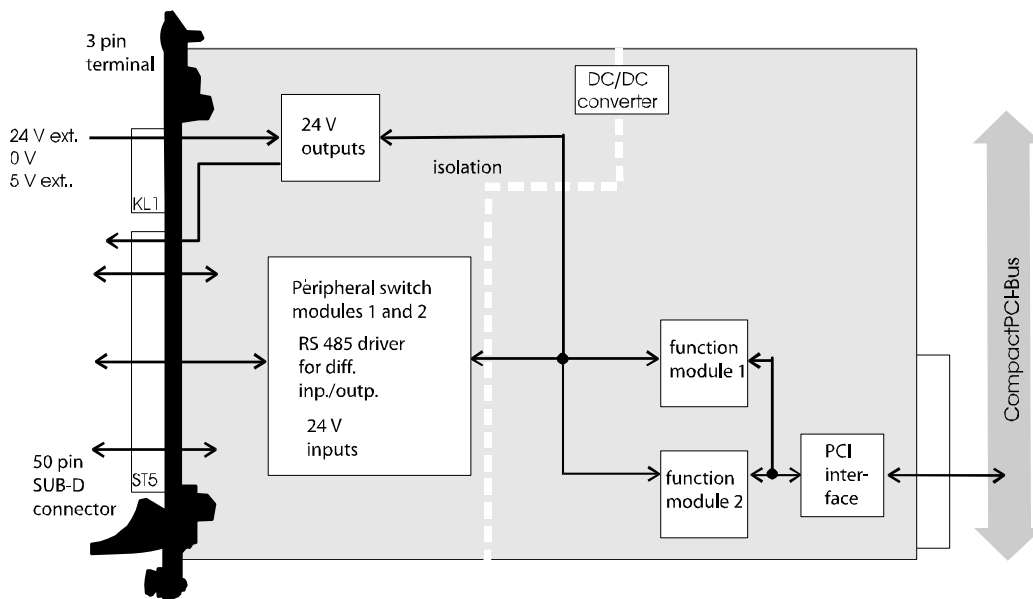


Fig. 9-2: Block diagram of the CPCI-1711



The user can reprogram the board without uninstalling it. Hereof the following advantages result:

- easy extension of the board's functionality
- implementation of customer requests
- easy download of newly developed functions
- reduced storage place
- etc.

The function modules allow to connect digital input and output signals and to process them on hardware base (in real time) before they are passed to the PC.

One function module is a physical unit that consists of:

- digital inputs
- digital outputs
- one function programmable component (hardware)

The function modules are connected also between each other through an internal bus.

## 9.2 Digital inputs and outputs

### 9.2.1 Description

To each function module digital inputs and outputs are allocated firmly. However, a few inputs also can be set as outputs for certain functions, which are implemented on the function module.

8 lines are available for each module: (see Fig. 9-3: block diagram of inputs and outputs) (1 function module)

The lines are divided according to the board as follows:

**Input lines**

- 2 x TTL, RS422 (signals C, D)
- 3 x 24 V, 5 V optional (signals E, F, G)

**Output lines**

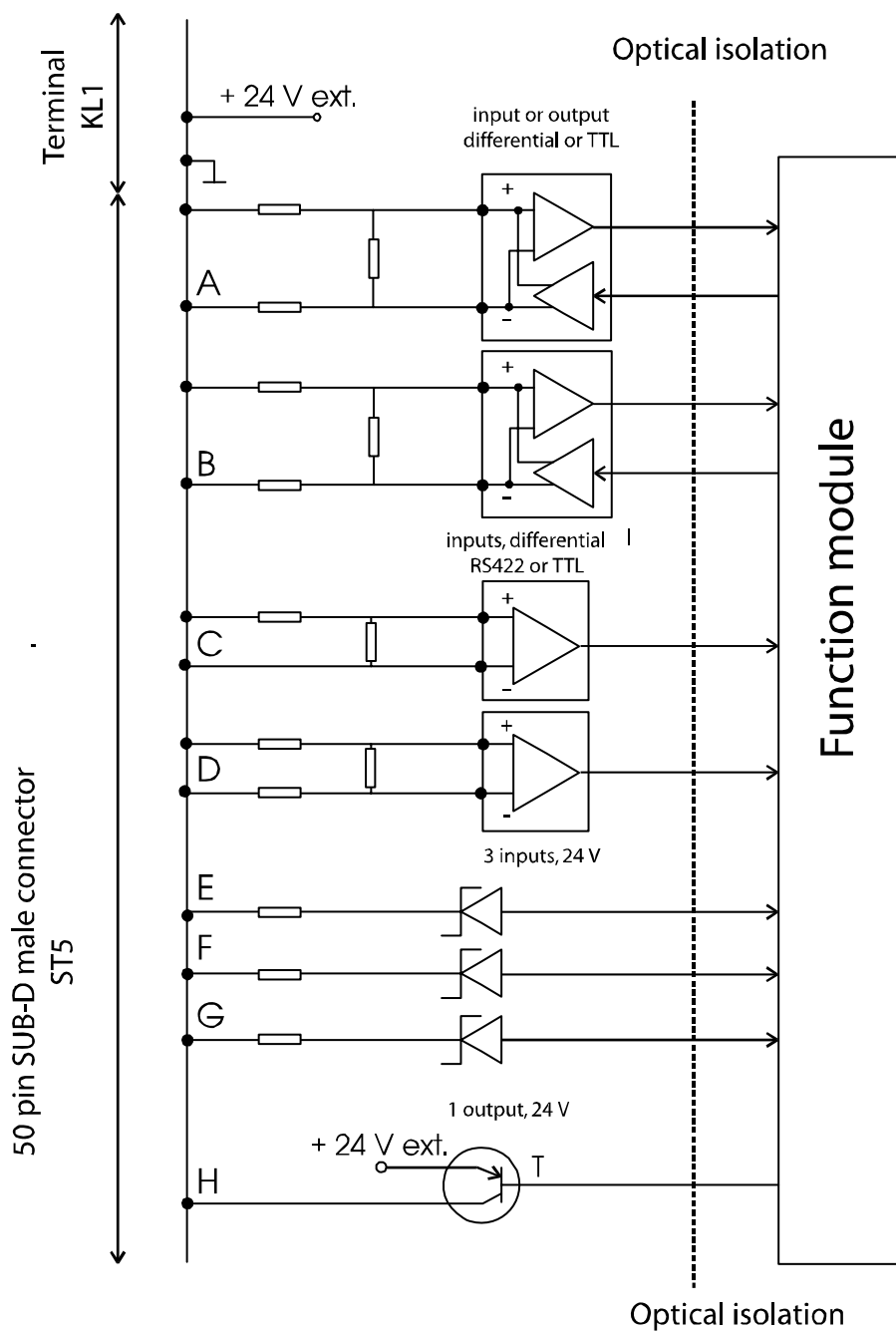
- 1 x 24 V, TTL optional (signal H)

**Free definable lines (input or output)**

- 2 x TTL, RS422 (signals A, B)

The function of the digital inputs and outputs depends on the function that is programmed on the function module and is described respectively in the corresponding documentations. The following sections describe only the general characteristics of the inputs and outputs.

**Fig. 9-3: Block diagram of the digital inputs and outputs (1 function module)**



## 9.2.2 Inputs

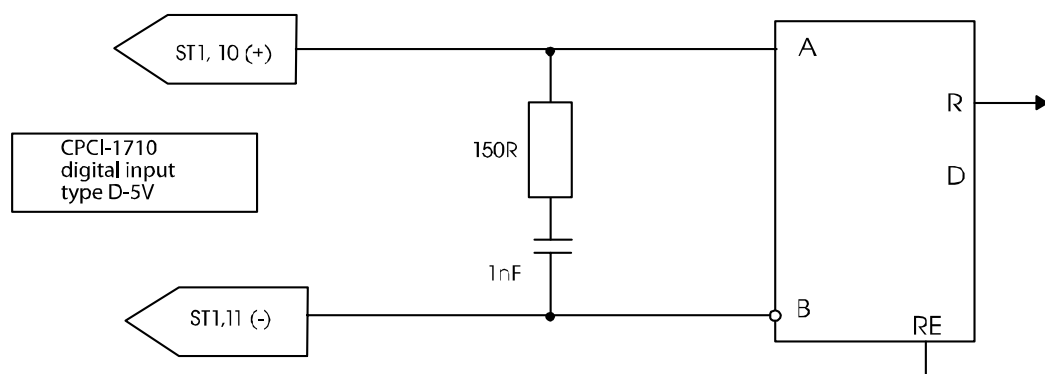
The inputs are distinguished as follows:

- differential inputs for very fast signals
- mass-related inputs

### Differential inputs

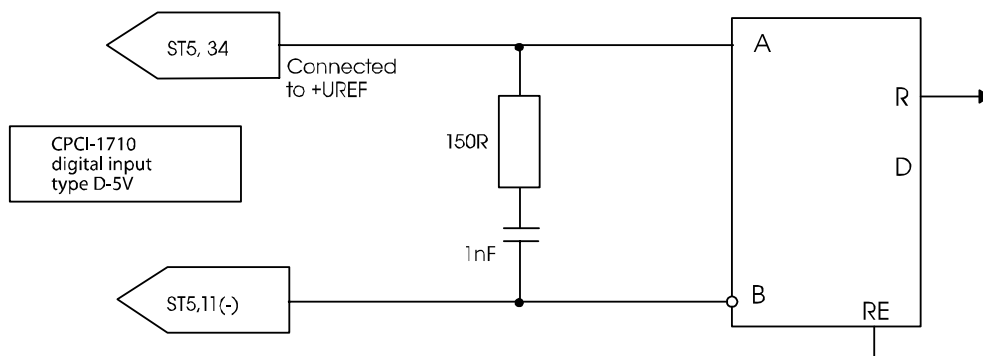
Max. 4 differential inputs (A, B, C and D) are available for each “function module”. The levels in the standard delivery correspond with the RS485 (5 V) standard.

**Fig. 9-4: Basic circuit of the differential inputs (5 V)**

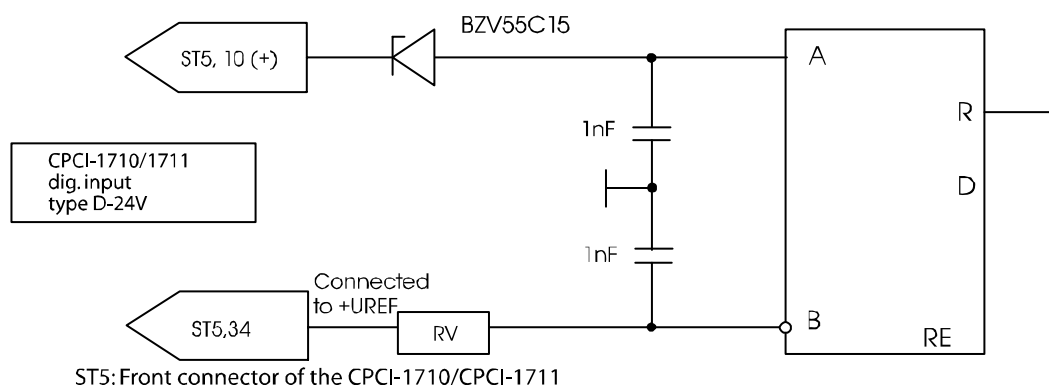


Alternatively, also TTL signals can be connected with these inputs. Please note that the other input of the differential receiver is wired on “+UREF”, so that it also can be a difference. So the TTL signal is, according to the wiring, inverted or not inverted to the function module.

**Fig. 9-5: Basic circuit of the differential inputs 5 V; used as TTL inputs**

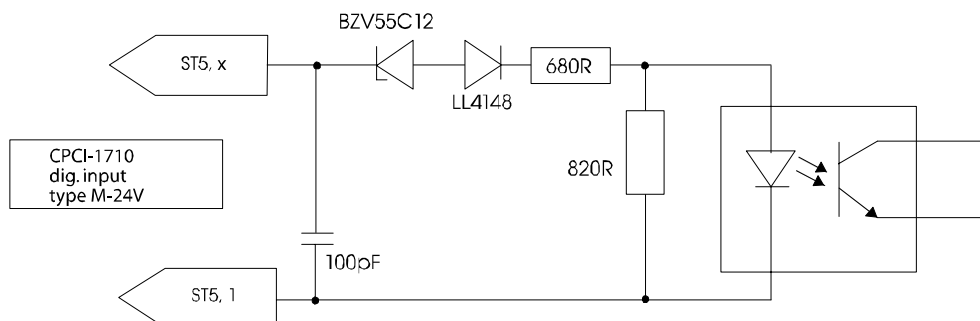


As an option, these differential inputs (A to D) can also be designed for the connection to a 24 V impulse transmitter / signal transmitter.

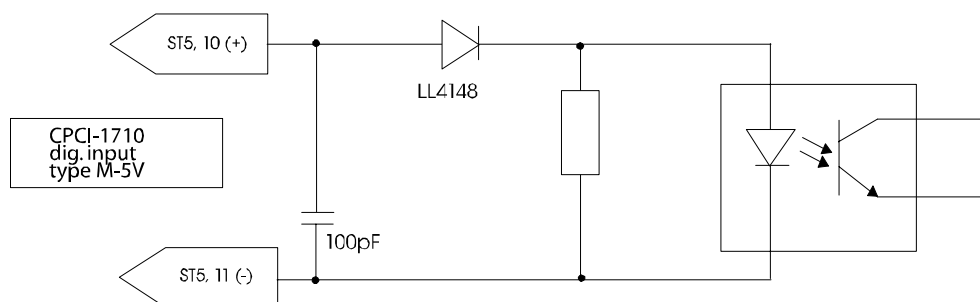
**Fig. 9-6: Basic circuit of the 24 V differential inputs (OPTION)****Mass-related inputs**

Max. 3 mass-related inputs (E, F and G) are available for one function module. The levels in the standard delivery correspond with the 24 V standard (IEC1131-2 / type 1).

These inputs have a common ground line.

**Fig. 9-7: Basic circuit of the digital inputs 24 V**

These inputs can be delivered on request for another signal level (option).

**Fig. 9-8: Basic circuit of the digital inputs 5 V (OPTION)**

### 9.2.3 Outputs

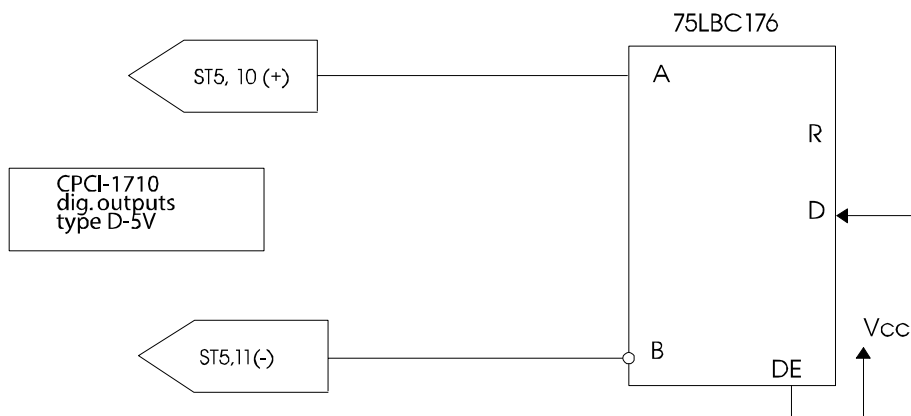
The outputs are distinguished as follows:

- differential outputs for very fast signals
- mass-related outputs

#### Differential outputs

There are max. 2 differential outputs (A and B) available for one “function module”. The levels in the standard delivery correspond with the RS485 (5 V) standard. In this case A and B can not be used as common inputs.

**Fig. 9-9: Basic circuit of the digital outputs 5V – differential**

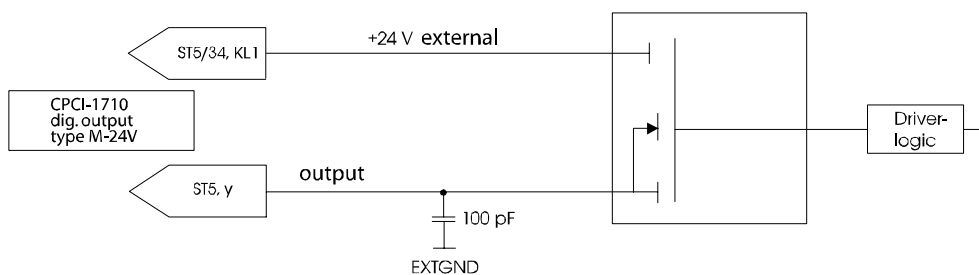


#### Mass-related outputs

Max. one mass-related output (H) is available for one function module. The levels in the standard delivery correspond with the 24 V standard. (IEC1131-2 / High-Side driver).

**Fig. 9-10: Basic circuit of the digital output H – 24 V**

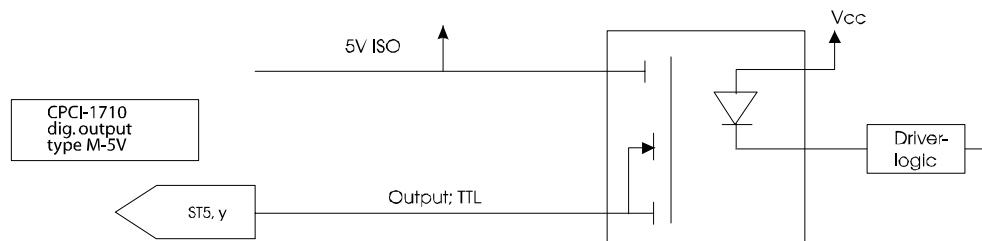
The 24 V can be fed either through the connector ST1 (jumper in position A) or separately through the terminal ST2.



The output can be delivered on request also as opto-coupler output (TTL compatible).



Fig. 9-11: Basic circuit of a digital output H – 5 V (OPTION)



### 9.3 PCI bus interface

The **CPCI-1710** is an extension board for the PCI/CompactPCI bus.

Hereof result the following advantages:

- the PCI/CompactPCI bus is able for “Plug & Play”, i.e. the software sets the addresses, interrupts automatically.
- the board is identified automatically through the software program
- the **CPCI-1710** allows a 32-bit access on the peripheral for faster data transfers.
- the board is to be operated as “Target only” on the PCI bus. It has a common interrupt that is wired to the INTA pin of the PCI connector.

After the PC has booted and the application software has got the respecting base addresses for the **CPCI-1710** through the BIOS function, the board can be accessed via the usual I/O write/read commands.

The four function modules need 256 bytes of the 64 kilobytes I/O range of the PC, always 64 bytes for each function module.



#### **WARNING!**

The addresses allocated by the BIOS should not be reprogrammed.

The following table shows the I/O assignment:

**Table 9-1: Assignment of the function modules**

	IORD / IOWR			
	D31...D24	D23...D16	D15.....D8	D7.....D0
BYTES	HIGHBYTE	MIDHIGHBYTE	MIDLOWBYTE	LOWBYTE
BASE+0	FUNCTION MODULE 1			
BASE+63	FUNCTION MODULE 1			
BASE+64	FUNCTION MODULE 2			
BASE+127	FUNCTION MODULE 2			
BASE+128	FUNCTION MODULE 3			
BASE+191	FUNCTION MODULE 3			
BASE+192	FUNCTION MODULE 4			
BASE+255	FUNCTION MODULE 4			

The function modules (FM) can seize the following addresses:

- FM 1 can occupy the addresses: BASE +0 to BASE +63.
- FM 2 can occupy the addresses: BASE +64 to BASE +127
- FM 3 can occupy the addresses: BASE +128 to BASE +191
- FM 4 can occupy the addresses: BASE +192 to BASE +255

**Example:**

The incremental transmitters are programmed with reference point logic in the function modules 1 to 4. The corresponding I/O-function range for the “incremental transmitter” will be copied into the therefore allocated places of the function modules.

The following I/O range is built:

Table 9-2: I/O range

	IORD / IOWR			
	D31...D24	D23...D16	D15.....D8	D7.....D0
BYTES	HIGHBYTE	MIDHIGHBYTE	MIDLOWBYTE	LOWBYTE
BASE+0	Incremental counter 1			
BASE +59	Incremental counter 1			
BASE +60 bis +63	Function module 1 - Info <b>SCxx</b>			
BASE+64	Incremental counter 2			
BASE+ 123	Incremental counter 2			
BASE + 124 bis + 127	Function module 2 - Info SCxx			
BASE+128	Incremental counter 3			
BASE+187	Incremental counter 3			
BASE +188 bis +191	Function module 3 - Info SCxx			
BASE+192	Incremental counter 4			
BASE+251	Incremental counter 4			
BASE +252 bis +255	Function module 4 - Info SCxx			

SC: Incremental counter      xx: Number of version, e.g. SC23 Version 2.3

All settings on the board are realized through the software (API), through the connector assignment or through a reprogramming of the function module (without a jumper on the board).

## 10 STANDARDSOFTWARE



### IMPORTANT!

The **CPCI-1710/CPCI-1711** is compatible with the **APCI-1710** in respect of the software installation. The ADDIREG program draws no distinction between the both system (PCI board or CompactPCI board). The API functions of the standard software are also the same.

### 10.1 Introduction



### IMPORTANT!

Remember the following style conventions in the text.

Function:     *“i\_APCI1710\_SetBoardInformation”*

Variable:     *ui\_Address*

**Table 10-1: Type declaration for DOS and Windows 3.1x**

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

Table 10-2: Type declaration for Windows 95/NT

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

Table 10-3: Define value

Define name	Decimal value	Hexadecimal value
<b>DLL_COMPILER_C</b>	1	1
<b>DLL_COMPILER_VB</b>	2	2
<b>DLL_COMPILER_PASCAL</b>	3	3
<b>DLL_LABVIEW</b>	4	4
<b>DLL_COMPILER_VB_5</b>	5	5
<b>APCI1710_DISABLE</b>	0	0
<b>APCI1710_ENABLE</b>	1	1

## 10.2 Software functions

### 10.2.1 Initialization

In this chapter you can find the common functions for each function module. The software function depending on the function of the board **CPCI-1710** is in the respecting manuals.

#### 1) I\_APCI1710\_InitCompiler (...)

**Syntax:**

<Return-Wert>= i\_APCI1710\_InitCompiler  
(BYTE b\_CompilerDefine)

**Parameter:**

**- Input:**

BYTE	b_CompilerDefine	The user shall select under Windows the language, which he wants to use for programming.
		- DLL_COMPILER_C: The user programs in C.
		- DLL_COMPILER_VB: Programming in Visual Basic for Windows.
		- DLL_COMPILER_VB_5: Programming in Visual Basic 5 for Windows NT or Windows 95/98.
		- DLL_COMPILER_PASCAL: Programming in Pascal or Delphi.
		- DLL_LABVIEW: Programming in Labview.

**- Output:**

No output signal has occurred.

**Task:**

If you want to use the DLL functions, please specify the language, which you want to use for programming. Call this function firstly.



**IMPORTANT!**

**This function is only available under Windows.**

**Calling convention:**ANSI C :`int i_ReturnValue;``i_ReturnValue = i_APCI1710_InitCompiler (DLL_COMPILER_C);`**Return value:**`0: No error``-1: Compiler Parameter is wrong`

## 2) I\_APCI1710\_CheckAndGetPCISlotNumber (...)

### Syntax:

<Return-Wert> = i\_APCI1710\_CheckAndGetPCISlotNumber  
(PBYTE pb\_SlotNumberArray)

### Parameter:

#### - Input:

There is no input.

#### - Output

PBYTE pb\_SlotNumberArray List of slot numbers

### Task:

Controls all **CPCI-1710** and shows the slot number of each board.  
Each parameter *pb\_SlotNumberArray* contains the slot number  
(1 to 8) of a **CPCI-1710** board.

### Calling convention:

#### ANSI C:

```
int i_ReturnValue;
```

```
unsigned char b_SlotNumberArray [8];
```

```
i_ReturnValue = i_APCI1710_CheckAndGetPCISlotNumber  
                (b_SlotNumberArray);
```

### Return value:

Returns the number of **CPCI-1710** boards that are installed in the PC. If the  
Return Value shows "0" then no **CPCI-1710** board was found on your PC.



### 3) I\_APCI1710\_SetBoardInformation (...)

#### Syntax:

```
<Return-Wert> = i_APCI1710_SetBoardInformation
                    (BYTE    b_SlotNumber,
                     PBYTE   pb_BoardHandle)
```

#### Parameter:

##### - Input:

BYTE b\_SlotNumber slot number of the board

##### - Ausgabe:

PBYTE pb\_BoardHandle Handle of the board to use the functions.

#### Task:

Controls if there is a **xPCI-1710** board and save the slot number. The user gets back a handle so that the next functions can be used. Handles allow the administration of various boards.

#### Calling convention:

##### ANSI C:

```
int i_ReturnValue;
unsigned char b_BoardHandle;
i_ReturnValue = i_APCI1710_SetBoardInformation (1, &b_BoardHandle);
```

#### Return value:

- 0: No error
- 1: Slot number not available
- 2: There is no board
- 3: No handle for the board available (limited to 10 handles)
- 4: Error during opening of the driver in Windows NT/ Windows 95

#### 4) I\_APCI1710\_ConfigureAllModule

##### Syntax:

```
<Return-Wert> = i_APCI1710_ConfigureAllModule
                                (BYTE      b_BoardHandle,
                                PCHAR      pc_FileName1,
                                PCHAR      pc_FileName2,
                                PCHAR      pc_FileName3,
                                PCHAR      pc_FileName4,
                                PULONG     pul_WriteAddressError);
```

##### Parameter:

###### - Input:

BYTE	b_BoardHandle	Handle of the board
PCHAR	pc_FileName1	Name of the configuration file for FM <sup>1</sup> 0
PCHAR	pc_FileName2	Name of the configuration file for FM 1
PCHAR	pc_FileName3	Name of the configuration file for FM 2
PCHAR	pc_FileName4	Name of the configuration file for FM 3

###### - Output:

PULONG	pul_WriteAddressError	In case of an error, address of the read/write register
--------	-----------------------	---

##### Task:

Initializes the function modules through software.

##### Calling convention:

###### ANSI C:

```
int          i_ReturnValue;
unsigned char b_BoardHandle;
unsigned long ul_WriteAddressError;

i_ReturnValue = i_APCI1710_ConfigureAllModule
                                (b_BoardHandle,
                                "CFG\INC_CPT.CFG",
                                "CFG\INC_CPT.CFG",
                                "CFG\INC_CPT.CFG",
                                "CFG\INC_CPT.CFG",
                                &ul_WriteAddressError);
```

##### Return value:

- 0: No error
- 1: The handle parameter of the board is wrong
- 2: Error during loading of the file
- 3: The decoding of the Altera file is wrong
- 4: Error during deleting the EEPROM component (Flash)
- 5: The programming is wrong
- 6: Error during read-out preparation of the flash
- 7: The read-out of the flash is wrong
- 8: Comparison between writing and reading of the flash is not correct
- 9: Error during loading of the Flex component
- 10: The loading test of the Flex component is wrong.

---

<sup>1</sup> FM: Function module

## 5) I\_APCI1710\_GetHardwareInformation

### Syntax:

```
<Return-Wert> = i_APCI1710_GetHardwareInformation
                    (BYTE    b_BoardHandle,
                     PUINT    pui_BaseAddress,
                     PBYTE    pb_InterruptNbr,
                     PBYTE    pb_SlotNumber)
```

### Parameter:

#### - Input:

BYTE	b_BoardHandle	Handle of the board
------	---------------	---------------------

#### - Output:

PUINT	pui_BaseAddress	Base address of the board
PBYTE	pb_InterruptNbr	Interrupt channel of the board
PBYTE	pb_SlotNumber	Slot number of the board

### Task:

Returns the slot's base address, the interrupt and the slot number.

### 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_APCI1710_GetHardwareInformation
                    (b_BoardHandle,
                     &ui_BaseAddress,
                     &b_InterruptNbr,
                     &b_SlotNumber);
```

### Return-Wert:

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

## 6) I\_APCI1710\_CloseBoardHandle (...)



### IMPORTANT!

Call this function every time you want to quit the user program!

#### Syntax:

```
<Return-Wert> = i_APCI1710_CloseBoardHandle  
                (BYTE      b_BoardHandle)
```

#### Parameter:

##### - Input:

BYTE b\_BoardHandle Handle der **xPCI-1710**

##### - Output:

No output signal has occurred

#### Task:

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

#### Calling convention:

ANSI C :

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

```
i_ReturnValue = i_APCI1710_CloseBoardHandle (b_BoardHandle);
```

#### Return value:

0: No error

-1: Handle parameter of the board is wrong

## 10.2.2 Interrupt

### 7) I\_APCI1710\_SetBoardRoutineDos (...)



#### IMPORTANT!

This function can be used only for C/C++ and Pascal for DOS

#### Syntax:

```
<Return-Wert> = i_APCI1710_SetBoardIntRoutineDos
                (BYTE  b_BoardHandle,
                 VOID   v_FunctionName
                 (BYTE  b_BoardHandle,
                  BYTE  b_ModuleMask,
                  ULONG ul_InterruptMask,
                  ULONG ul_CounterLatchValue))
```

#### Parameter:

##### - Input:

BYTE	b_BoardHandle	Handle der <b>xPCI-1710</b> Karte
VOID	v_FunctionName	Name der Benutzer-Interruptroutine

##### - Output:

No output signal has occurred.

#### Task:

This function is to be called for all **xPCI-1710** board on which you want to activate an interrupt.

At the first calling of the function (first board):

- the user interrupt routine is installed
- interrupts are made possible

If you operate various **xPCI-1710** boards that shall react to interrupts, you shall call the function as many times as how many **xPCI-1710** boards you are operating.

The variable *v\_FunctionName* is only at the first calling of importance. From the second calling of the function on (next boards), interrupts are enabled.

#### Interrupt

If an interrupt is generated, the user interrupt routine is invoked from the system. If various boards are operated and shall react to interrupts, the variable *b\_BoardHandle* shows the identification number (handle) of the board, which has generated the interrupt.

The user interrupt routine shall have the following syntax:

```
VOID v_FunctionName (BYTE  b_BoardHandle,
                    BYTE  b_ModuleMask
                    ULONG  ul_InterruptMask,
                    ULONG  ul_CounterLatchValue)
```

<i>v_FunctionName</i>	Name of the user interrupt routine
<i>b_BoardHandle</i>	Number of the <b>xPCI-1710</b> -handle, which has generated the interrupt.
<i>b_ModulMask</i>	Mask of the module, which has generated the interrupt. (See table of the interrupt mask in the respecting manual)
<i>ul_InterruptMask</i>	Mask of the event, which has generated the interrupt. (See table of the interrupt mask in the respecting manual)
<i>ul_CounterLatchValue</i>	The latched values of the time are returned (See table of the interrupt mask in the respecting manual)

The user can specify another name for *v\_FunctionName*, *b\_BoardHandle*, *ul\_InterruptMask*, *ul\_CounterLatchValue*

### Calling convention:

ANSI C:

```

void    v_FunctionName    (unsigned char b_BoardHandle,
                           unsigned char b_ModuleMask,
                           unsigned long ul_InterruptMask
                           unsigned long ul_CounterLatchValue)
    {
        .
        .
    }
int      i_ReturnValue;
unsigned char b_BoardHandle;

i_ReturnValue = i_APCI1710_SetBoardIntRoutineDos
                (b_BoardHandle,
                 v_FunctionName );

```

### Return value:

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

8) **i\_APCI1710\_SetBoardRoutingVBDos (..)****IMPORTANT!**

This function can be used only for Visual Basic DOS.

**Syntax:**

```
<Return-Wert> = i_APCI1710_SetBoardIntRoutineVBDos  
                (BYTE    b_BoardHandle)
```

**Parameter:****- Input:**

BYTE b\_BoardHandle Handle der Karte

**- Output:**

No output signal has occurred.

**Task:**

This function is to be called for all **xPCI-1710** boards on which you want to activate an interrupt. When an interrupt is activated, a Visual Basic Event will be generated.

At the first calling of a function (first board):

- the interrupts for the selected board are made possible.

If you operate various **xPCI-1710** boards that shall react to interrupts, you shall call the function as many times as how many **xPCI-1710** boards you are operating.

***Interrupt***

If an interrupt is generated, the user interrupt routine of the system is called.

***Controlling the interrupt administration:***

Use the functions "ON UEVENT GOSUB xxxxxxxxx" of Visual Basic DOS and "i\_APCI1710\_TestInterrupt"

This function tests the interrupt of the xPCI-1710. It is used to get the values of *b\_BoardHandle*, *b\_ModuleMask*, *ul\_InterruptMask* and *ul\_CounterLatchValue*

**Calling convention:**

Visual Basic DOS:

```
Dim Shared i_ReturnValue    As Integer  
Dim Shared i_BoardHandle    As Integer  
Dim Shared i_ModuleMask     As Integer  
Dim Shared l_InterruptMask  As Long  
Dim Shared l_CounterLatchValue As Long  
IntLabel:
```

```
i_ReturnValue = i_APCI1710_TestInterrupt (i_BoardHandle, _  
                                           i_ModuleMask, _  
                                           l_InterruptMask, _  
                                           l_CounterLatchValue)
```

Return

```
ON UEVENT GOSUB IntLabel  
UEVENT ON
```

```
i_ReturnValue = i_APCI1710_SetBoardIntRoutineVBDos(b_BoardHandle)
```

**Return value:**

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



9) **i\_APCI1710\_SetBoardIntRoutineWin16** (..)**i****IMPORTANT!**

This function can be used only for Windows 3.1 and Windows 3.11.

**Syntax:**

```
<Return-Wert> = i_APCI1710_SetBoardIntRoutineWin16
                (BYTE  b_BoardHandle,
                 VOID   v_FunctionName
                 (BYTE  b_BoardHandle,
                  BYTE  b_ModuleMask,
                  ULONG l_InterruptMask,
                  ULONG ul_CounterLatchValue))
```

**Parameter:****- Input:**

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

**- Output:**

No output signal has occurred

**Task:**

This function is to be called for all **xPCI-1710** boards, on which you want to activate an interrupt.

At the first calling of a function (first board):

- the user interrupt routine is installed
- the interrupts are made possible.

If you operate various **xPCI-1710** boards that shall react to interrupts, you shall call the function as many times as how many xPCI-1710 boards you are operating.

The variable *v\_FunctionName* is only at **the first calling** of importance. From the second calling of a function on (next boards) interrupts are enabled.

**Interrupt**

If an interrupt is being generated, the user interrupt routine will be called of the system.

If various boards are operated and shall react to interrupts, the variable *b\_BoardHandle* shows the identification number (handle) of the board, which has generated the interrupt.

The user interrupt routine shall have the following syntax:

```
VOID v_FunctionName (BYTE  b_BoardHandle,
                    BYTE  b_ModuleMask,
                    ULONG  ul_InterruptMask,
                    ULONG  ul_CounterLatchValue)
```

*v\_FunctionName*                      Name of the user interrupt routine

*b\_BoardHandle*                      Handle of the **xPCI-1710**, which has generated the interrupt.

<i>b_ModuleMask</i>	Mask of the module, which has generated the interrupt (see table of the interrupt mask in the respecting manual)
<i>ul_InterruptMask</i>	Mask of the events, which have generated the interrupt (see table of the interrupt mask in the respecting manual)
<i>ul_CounterLatchValue</i>	The latched values of the time rare returned (see table of the interrupt mask in the respecting manual)

The user can specify another name for *v\_FunctionName*, *b\_BoardHandle*, *b\_ModuleMask*, *ul\_InterruptMask* und *ul\_CounterLatchValue*.

# i

## IMPORTANT!

If you use Visual Basic for Windows the following parameter does not exist!

Use the function: "i\_APCI1710\_TestInterrupt"!

```
VOID v_FunctionName (BYTE b_BoardHandle,
                     BYTE b_ModuleMask,
                     ULONG ul_InterruptMask,
                     ULONG ul_CounterLatchValue)
```

## Calling convention:

ANSI C :

```
void v_FunctionName (unsigned char b_BoardHandle,
                    unsigned char b_ModuleMask,
                    unsigned long ul_InterruptMask,
                    unsigned long ul_CounterLatchValue)
{
    .
    .
}
```

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

```
i_ReturnValue = i_APCI1710_SetBoardIntRoutineWin16
                (b_BoardHandle,
                 v_FunctionName );
```

## Return-Wert:

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

## 10) i\_APCI1710\_SetBoardInRoutineWin32 (..)

**i****IMPORTANT!**

This function is only available for Windows NT and Windows 9x.

**Syntax:**

```
<Return-Wert> = i_APCI1710_SetBoardInRoutineWin32
                (BYTE      b_BoardHandle,
                 BYTE      b_UserCallingMode,
                 ULONG     ul_UserSharedMemorySize,
                 VOID **   ppv_UserSharedMemory,
                 VOID      v_FunctionName
                 (BYTE     b_BoardHandle,
                  BYTE     b_ModuleMask,
                  ULONG    ul_InterruptMask,
                  ULONG    ul_CounterLatchValue,
                  BYTE     b_UserCallingMode,
                  VOID *   pv_UserSharedMemory))
```

**Parameter:****- Input:**

BYTE	b_BoardHandle	Handle of the board <b>xPCI-1710</b>
BYTE	b_UserCallingMode	APCI1710_SYNCHRONOUS_MODE: The user routine is called directly through the interrupt routine of the driver. APCI1710_ASYNCHRONOUS_MODE: The user routine is called directly through the interrupt thread of the driver
VOID	v_FunctionName	Name of the interrupt routine
ULONG	ul_UserSharedMemorySize	Defines the size in bytes of the user shared memory. You can use the parameter only if you have selected the following mode: APCI1710_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	Address of the user shared memory You can use the parameter only if you have selected the following mode: APCI1710_SYNCHRONOUS_MODE
---------	----------------------	---

**i****WICHTIG!**

For Windows NT and Windows 95 are 4 rings available (ring 0 to ring 3).

- The user application program runs under ring 3. In this ring no access to hardware is available.
- VXD and SYS drivers run under ring 0 and have hardware access.
- Ring 0 can not access to the variable of ring 3 and therefore shall use the Shared Memory.
- Ring 0 and ring 3 have an indicator, which identifies this Shared Memory. Both rings have a different address.

This function is to be called for all **xPCI-1710** boards, on which you want to activate an interrupt.

At the first calling of a function (first board):

- the user interrupt routine is installed
- the interrupts are made possible.
- the user shared memory will be allocated if the following mode is activated:

`APCI1710_SYNCHRONOUS_MODE`.

If you operate various xPCI-1710 boards that shall react to interrupts, you shall call the function as many times as how many **xPCI-1710** boards you are operating.

The variable `v_FunctionName` is only at **the first calling** of importance. From the second calling of a function on (next boards) interrupts are enabled.

***Interrupt***

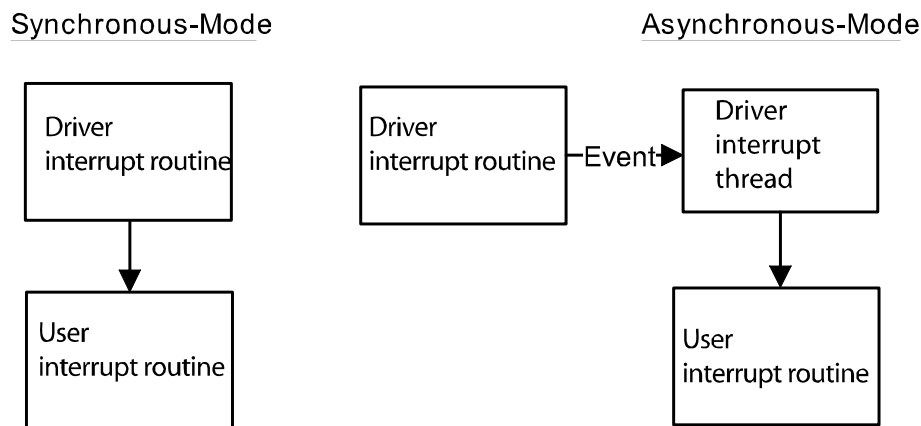
If an interrupt is being generated, the user interrupt routine of the system will be called.

If various boards are operated and shall react to interrupts, the variable `b_BoardHandle` shows the identification number (handle) of the board, which has generated the interrupt.

The user interrupt can be called as follows:

- directly from the interrupt routine of the driver (synchronous mode). The code of the user interrupt routine runs directly under ring 0.
- or of the Interrupt Thread of the driver (asynchronous mode). An event will be generated and the Interrupt Thread calls the user interrupt routine. The code of the user interrupt routine runs under ring 3.

The interrupt thread of the driver has the first priority (31) in the system.

**Fig. 10-1: Synchronous and asynchronous mode****Fig. 10-2: Synchronous and asynchronous mode**

<b>SYNCHRONOUS MODE</b>	
<b>ADVANTAGES</b>	The user interrupt routine is called directly from the interrupt routine of the driver (ring 0). The time between interrupt and user interrupt routine is reduced.
<b>LIMITS</b>	A debug of the user interrupt routine is not possible.
	The user interrupt routine can not call the Windows API functions.
	The user interrupt routine can not call the functions, which have access to the shared variable. However, the user can use a shared memory.
	The user interrupt routine can call only the functions of the xPCI-1710 device driver, which have the following extension: "i_APCI1710_KRNL_XXX"
	This mode can not be used under Visual Basic.

<b>ASYNCHRONOUS MODE</b>	
<b>ADVANTAGES</b>	A debug of the user interrupt routine is possible.
	The user interrupt routine can call the API functions.
	The user interrupt routine can call the functions, which have access to the shared variable.
	The user interrupt routine can call all functions of the xPCI-1710 device driver. Syntax: "i_APCI1710_XXX"
<b>LIMITS</b>	The code of the user interrupt routine is called from the interrupt thread of the driver (ring 3). The time between interrupt and user interrupt routine is longer.

**Shared memory:**

If you have chosen the APCI1710\_Synchronous\_Mode, you have no access to the common functions. However, you have the possibility to create a shared memory (ppv\_UserSharedMemory), in which all given compilers or user defines are saved.

The variable *ul\_UserSharedMemorySize* identifies the size in byte of the selected user type.

An indicator of the variable *pv\_UserSharedMemory* will be returned to the interrupt routine with the variable *pv\_UserSharedMemory*. This function is not possible in Visual Basic.

The user interrupt routine shall have the following syntax:

```
VOID  v_FunctionName      (BYTE    b_BoardHandle,
                           BYTE    b_ModuleMask,
                           ULONG   ul_InterruptMask,
                           ULONG   ul_CounterLatchValue,
                           BYTE    b_UserCallingMode,
                           VOID *  pv_UserSharedMemory)
```

*v\_FunctionName*      Name of the user interrupt routine

*b\_BoardHandle*      Handle of the **xPCI-1710**, which has generated the interrupt.

*b\_ModuleMask*      Mask of the module, which has generated the interrupt (see table in the interrupt of the respecting manual)

*ul\_InterruptMask*    Mask of the event, which has generated the interrupt (see table of the interrupt mask in the respecting manual)

*ul\_CounterLatchValue*    The latched values of the counter are returned. (See table of the interrupt mask in the respecting manual)

*b\_UserCallingMode*    APCI1710\_SYNCHRONOUS\_MODE:  
The user routine is called directly from the driver interrupt routine  
APCI1710\_ASYNCHRONOUS\_MODE:  
The user interrupt routine is called directly from the driver interrupt thread.

*pv\_UserSharedMemory*    Indicator of the user shared memory.  
The user can use other names for *v\_FunctionName*, *b\_BoardHandle*, *b\_ModuleMask*, *ul\_InterruptMask*, *ul\_CounterLatchValue*, *b\_UserCallingMode* und *pv\_UserSharedMemory*.

**i****IMPORTANT!**

If you use Visual Basic 4 the following parameters are of no importance. Use the function: "i\_APCI1710\_TestInterrupt".

```
BYTE    b_UserCallingMode,
ULONG   ul_UserSharedMemorySize,
VOID ** ppv_UserSharedMemory,
VOID    v_FunctionName      (BYTE    b_BoardHandle,
```

```

BYTE    b_ModuleMask,
ULONG   ul_InterruptMask,
ULONG   ul_CounterLatchValue,
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_ModuleMask,
                    unsigned long    ul_InterruptMask,
                    unsigned long    ul_CounterLatchValue,
                    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_APCI1710_SetBoardIntRoutineWin32
                (b_BoardHandle,
                 APCI1710_SYNCHRONOUS_MODE,
                 sizeof(str_UserStruct),
                 (void **) &ps_UserSharedMemory,
                 v_FunctionName);

```

**Visual Basic 5:**

```

Sub v_FunctionName (ByVal i_BoardHandle As Integer,
                   ByVal i_ModuleMask As Integer,
                   ByVal l_InterruptMask As Long,
                   ByVal l_CounterLatchValue As Long,
                   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_APCI1710_SetBoardIntRoutineWin32

```

```
(i_BoardHandle,  
APCI1710_ASYNCHRONOUS_MODE,  
0,  
0,  
AddressOf v_FunctionName)
```

**Return value:**

0: No error

-1: Handle parameter of the board is wrong

-2: Interrupt is already installed

-3: The selected calling mode of the user interrupt routine is wrong

-4: No memory space for the user shared memory available.



**11) i\_APCI1710\_TestInterrupt****Syntax:**

```
<Return-Wert> = i_APCI1710_TestInterrupt
                                (BYTE      b_BoardHandle,
                                BYTE      b_ModuleMask
                                PULONG    pul_InterruptMask,
                                PULONG    pul_CounterLatchValue)
```

**Parameter:****- Input:**

There is no input

**- Output:**

PBYTE	pb_BoardHandle	Handle of the <b>xPCI-1710</b> , which has generated the interrupt. (see table of the interrupt mask in the respecting manual)
PBYTE	pb_ModuleMask	Mask of the module, which has generated the interrupt. (see table of the interrupt mask in the respecting manual)
PULONG	pul_InterruptMask	Mask of the events, which have generated the interrupt. (see table of the interrupt mask in the respecting manual)
PULONG	pul_CounterLatchValue	Returns the latched values of the counter.

**Task:**

Controls if a xPCI-1710 has generated an interrupt. If yes, it returns the handle of the board and the source of the interrupt.

**i****IMPORTANT!**

This function can be used only in Visual Basic for DOS and Windows.

**Calling convention:**

ANSI C :

```
int          i_ReturnValue;
unsigned char b_BoardHandle;
unsigned char b_ModuleMask;
unsigned char ul_InterruptMask;
unsigned int  ul_CounterLatchValue;
```

```
i_ReturnValue = i_APCI1710_TestInterrupt (&b_BoardHandle,
                                           &b_ModuleMask
                                           &ul_InterruptMask,
                                           &ul_CounterLatchValue;
```

**Return value:**

-1: No interrupt

> 0: IRQ number

**12) i\_APCI1710\_ResetBoardIntRoutine (..)****Syntax:**

<Return-Wert> = i\_APCI1710\_ResetBoardIntRoutine  
(BYTE b\_BoardHandle)

**Parameter:**

- Input:  
    BYTE     b\_BoardHandle           Handle of the board
- Output:  
    No output signal has occurred

**Task:**

Stops the interrupt administration of the xPCI-1710.  
Deinstalls the interrupt routine if the interrupt administration of all xPCI-1710 is stopped.

**Calling convention:**

ANSI C :

```
int               i_ReturnValue;  
unsigned char     b_BoardHandle;  
i_ReturnValue = i_APCI1710_ResetBoardIntRoutine  
                                                 (b_BoardHandle);
```

**Return value:**

- 0: No error
- 1: Handle parameter of the board is wrong
- 2: No interrupt installed with the function  
    *"i\_APCI1710\_SetBoardIntRoutineXXX"*

## 10.2.3 Initialisation input filter

### 13) i\_APCI1710\_InitInputFilter (..)

**Syntax:**

```
<Return Wert> = i_APCI1710_InitInputFilter
                    (BYTE   b_BoardHandle,
                     BYTE   b_Modul,
                     BYTE   b_TimeBase,
                     BYTE   b_Filter)
```

**Parameter:**

**-Input:**

BYTE	b_BoardHandle	Handle of the board x <b>PCI-1710</b>
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_TimeBase	Selection of the filter clock - APCI1710_30MHZ: The PC has a PCI bus clock of 30 MHz - APCI1710_33MHZ: The PC has a PCI bus clock of 33 MHz - APCI1710_40MHZ: The board has a 40 MHz quartz
BYTE	b_Filter	Selection of the filter. See Table 10-4

**-Output:**

There is no output.

**Task:**

Disables or enables the filter of the selected module (b\_Modul).  
b\_Filter indicates the filter time.

Table 10-4: Filter time

<i>b_TimeBase</i>	APCI1710_30MHZ	APCI1710_33MHZ	APCI1710_40MHZ
<i>b_Filter</i>			
0	Filter not used	Filter not used	Filter not used
1	133 ns	121 ns	100 ns
2	200 ns	182 ns	150 ns
3	267 ns	242 ns	200 ns
4	333 ns	303 ns	250 ns
5	400 ns	364 ns	300 ns
6	467 ns	424 ns	350 ns
7	533 ns	485 ns	400 ns
8	600 ns	545 ns	450 ns
9	667 ns	606 ns	500 ns
10	733 ns	667 ns	550 ns
11	800 ns	727 ns	600 ns
12	867 ns	788 ns	650 ns
13	933 ns	848 ns	700 ns
14	1000 ns	909 ns	750 ns
15	1067 ns	970 ns	800 ns

# i

## IMPORTANT!

This function is only available for the following modules:  
Chronos, pulse counter, incremental counter

### Calling convention:

ANSI C:

```
int i_ReturnValue;
unsigned char b_BoardHandle;
i_ReturnValue = i_APCI1710_InitInputFilter
(b_BoardHandle
0,
APCI1710_40MHz,
9);
```

### Return value:

- 0: No error
- 1: Handle parameter of the board is wrong.
- 2: The selected module number is wrong.
- 3: The selected module has no input filter.
- 4: The selected filter clock is wrong.
- 5: The selected filter time is wrong.
- 6: No 40 MHz quartz available on the board.

## 14) i\_APCI1710\_CheckInputFilter40MHzStatus (..)

### Syntax:

```
<Return Wert> = i_APCI1710_CheckInputFilter40MHzStatus  
                (BYTE      b_BoardHandle,  
                 PBYTE     pb_40MHz_Status)
```

**Parameter:**

**-Input:**

BYTE	b_BoardHandle	Handle of the board
------	---------------	---------------------

**-Output:**

PBYTE	pb_40MHz_Status	Availability 40 MHz on the board
		0: 40 MHz not available
		1: 40 MHz available

### Task:

Checks the availability of the 40 MHz clock on the board

**i**

## IMPORTANT!

This function is only available for the following modules:  
Chronos, pulse counter, incremental counter

### Calling convention:

ANSI C :

```
int i_ReturnValue;
unsigned char b_BoardHandle;
unsigned char b_40MHz_Status;
```

```
i_ReturnValue = i_APCI1710_CheckInputFilter40MHzStatus
                (b_BoardHandle
                 &b 40MHz Status);
```

### Return value:

0: No error  
-1: Handle parameter of the board is wrong.  
-2: The function is in none of the four function modules available.

# 11 INDEX

## C

Changing the registration of an existing board 29  
 Component scheme 18  
 Configuration of a new board 27  
 Configuration with ADDIREG 24  
 Connecting the peripheral 36  
 Connection of mass-related inputs and outputs 40  
 Connection of the differential I/O 41

## D

Definition of application 8  
 Digital inputs and outputs  
   Function description 45

## E

EMC  
   Electromagnetic compatibility 12  
 Energy requirements 13

## F

Function manuals 21  
 Functions 20

## H

Handling of the board 11

## I

Initialisation input filter  
   Software functions 76  
 Initialization  
   Software functions 55  
 Inputs  
   Limit values 14  
 Installation of the board 22  
 Intended use 8  
 Internet 35  
 Interrupt  
   Software functions 62

## L

Limit values 13  
 Loading a function into a function module  
   SET1710 30

## M

Module configuration with SET1710 30

## O

Optical isolation 17  
 Outputs  
   Limit values 15

## P

PCI bus interface  
   Function description 50  
 Personal protection  
   User 10  
 Physical set-up of the board 12  
 Pin assignment 36

## Q

Qualification  
   User 10

## R

Registration of a new board 28

## S

SET1710 30  
 Setting a module configuration  
   SET1710 34  
 Signals 19  
 Slots 22  
 Software 24  
 Standardsoftware 53

## T

Technical data 12  
 Terminal KL1 39

## U

Update 35  
 Usage restrictions 8

## V

Versions 13