

Information about hazardous substances

Please find a list of hazardous substances respecting the multifunction counter board **APCI-1710** in the following table:

Coarse classification	Chemical substance symbol					
	Pb	Hg	Cd	Cr (VI)	PBB	PBDE
Printed circuit board	0	0	0	0	0	0
Connectors	0	0	0	0	0	0
Logic ICs	0	0	0	0	0	0
Power IC (VN340)	X	0	0	0	0	0
Capacitors	0	0	0	0	0	0
Resistors	X	0	0	0	0	0
Diodes (with glass housings)	X	0	0	0	0	0

X = Limit value exceeded

0 = Limit value kept

Should you have any questions that you do not find in the manual or on our website (<http://www.addi-data.com>) please contact us by phone or e-mail.



DIN EN ISO 9001:2000
certified



ADDI-DATA GmbH
Dieselstraße 3
D-77833 OTTERSWEIER



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

Technical description

APCI-1710

**Multifunction counter board,
optically isolated**

Edition: 07.09 - 04/2007

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.

1	DEFINITION OF APPLICATION	9
1.1	Intended use	9
1.2	Usage restrictions.....	9
1.3	General description of the board	9
2	USER	11
2.1	Qualification	11
2.2	Personal protection.....	11
3	HANDLING OF THE BOARD	12
4	TECHNICAL DATA.....	13
4.1	Electromagnetic compatibility (EMC)	13
4.2	Physical set-up of the board	13
4.3	Versions	14
4.4	Limit values.....	14
4.4.1	Inputs.....	15
4.4.2	Outputs.....	16
4.4.3	Safety	18
4.5	Component scheme.....	19
5	AVAILABLE FUNCTIONS OF THE APCI-1710.....	20
5.1	Available signals.....	20
5.1.1	Connectable signal lines.....	20
5.1.2	Maximal signal wirings of the APCI-1710	20
5.2	Available functions.....	21
5.2.1	Programmable functions of the counter board	21
5.2.2	Connection facilities depending on the chosen function...	22
5.2.3	Delivered manuals	22
6	INSTALLATION OF THE BOARD	23
6.1	Opening the PC.....	23
6.2	Selecting a free slot	23
6.3	Plugging the board into the slot	24
6.4	Closing the PC	24
7	SOFTWARE	25
7.1	Delivered software	25
7.2	Configuration of the APCI-1710 with ADDIREG.....	25

7.2.1	Registration of a new board	30
7.2.2	Changing the registration of an existing board	30
7.3	Loading a function into a function module.....	31
7.3.1	Module configuration with SET1710	32
7.3.2	Setting a module configuration.....	34
7.4	ADDI-DATA on the internet	35
8	CONNECTING THE PERIPHERAL.....	36
8.1	Connector pin assignment.....	36
8.1.1	50-pin SUB-D front connector ST1	36
8.1.2	24 V supply for 24 V digital outputs (channel H).....	39
8.1.3	50-pin ribbon cable connector ST5	40
8.2	Connection of mass-related inputs and outputs	42
8.3	Connection of the differential digital I/O	43
8.4	Connection at the TTL inputs and outputs.....	45
9	FUNCTIONS OF THE BOARD	46
9.1	Description	46
9.2	Digital inputs and outputs.....	47
9.2.1	Description	47
9.2.2	Inputs.....	48
9.2.3	Outputs.....	50
9.3	TTL inputs and outputs	52
9.3.1	Common signals for all function modules.....	52
9.3.2	Single signals	52
9.4	PCI bus interface.....	53
10	STANDARD SOFTWARE	55
10.1	Introduction.....	55
10.2	Software functions.....	57
10.2.1	Initialisation	57
1)	I_APCI1710_InitCompiler (...)	57
2)	I_ACPI1710_CheckAndGetPCISlotNumber (...)	59
3)	I_APCI1710_SetBoardInformation (...)	60
4)	I_APCI1710_ConfigureAllModule	61
5)	I_ACPI1710_GetHardwareInformation.....	62
6)	I_APCI1710_CloseBoardHandle (...)	63
10.2.2	Interrupt	64
7)	I_APCI1710_SetBoardRoutineDos (...)	64
8)	i_APCI1710_SetBoardRoutingVBDos (..)	66
9)	i_APCI1710_SetBoardIntRoutineWin16 (..)	68
10)	i_APCI1710_SetBoardInRoutineWin32 (..)	70

11)	i_APCI1710_TestInterrupt.....	76
12)	i_APCI1710_ResetBoardIntRoutine (..).....	78
10.2.3	Initialisation input filter.....	79
13)	i_APCI1710_InitInputFilter (..).....	79
14)	i_APCI1710_CheckInputFilter40MHzStatus (..)	81
11	INDEX	82

Figures

Fig. 3-1: Correct handling	12
Fig. 4-1: Output current versus differential output voltage.....	16
Fig. 6-1: PCI-5V slot (32-bit).....	23
Fig. 6-2: Inserting the board	24
Fig. 6-3: Fastening the board at the back cover	24
Fig. 7-1: Registration program ADDIREG.....	26
Fig. 7-2: Configuration of a new board.....	28
Fig. 7-3: PCI boards.....	29
Fig. 7-4: SET1710: Configuration program for modules.....	32
Fig. 7-5: Selection of a APCI-/CPCI-170.....	32
Fig. 7-6: General information	33
Fig. 7-7: Function list and module configuration.....	33
Fig. 7-8: Setting a module configuration by mouse click.....	34
Fig. 7-9: Setting a module configuration by keyboard.....	35
Fig. 8-1: 50-pin SUB-D male connector (ST1)	36
Fig. 8-2: Terminal ST2	39
Fig. 8-3: Position of the connector ST5 on the board.....	40
Fig. 8-4: Pin assignment of the ribbon cable connector.....	40
Fig. 8-5: Connection of a mass-related input	42
Fig. 8-6: Connection of a mass-related output.....	43
Fig. 8-7: Connection of a differential input	43
Fig. 8-8: Example: Connection of 2 incremental encoders at FM1 ..	44
Fig. 8-9: Connection of a differential output.....	44
Fig. 8-10: Connection at TTL inputs and outputs	45
Fig. 9-1: Block diagram of the APCI-1710	46
Fig. 9-2: Block diagram of the digital inputs and outputs (1 function module)	48
Fig. 9-3: Basic circuit of the differential inputs (5 V).....	49
Fig. 9-4: Basic circuit of the differential inputs 5 V; used as TTL inputs	49
Fig. 9-5: Basic circuit of the 24 V inputs (APCI-1710-24V)	49
Fig. 9-6: Basic circuit of the digital inputs 24 V	50
Fig. 9-7: Basic circuit of the digital inputs 5 V (OPTION).....	50
Fig. 9-8: Basic circuit of the digital outputs 5 V – differential.....	51
Fig. 9-9: Basic circuit of the digital output H – 24 V	51
Fig. 9-10: Basic circuit of a digital output H – 5 V (OPTION).....	51
Fig. 10-1: Synchronous and asynchronous mode.....	72
Fig. 10-2: Synchronous and asynchronous mode.....	72

Tables

Table 5-1: Maximal input lines on the board	20
Table 5-2: Maximal input lines on a module	20
Table 5-3: Maximal output lines on the board	20
Table 5-4: Maximal output lines on a module.....	21
Table 5-5: Possible applications of the counter board.....	21
Table 5-6: Maximal application functions on the board	22

Table 5-7: Delivered manuals	22
Table 8-1: Pin assignment for function module No. 1 (FM1)	37
Table 8-2: Pin assignment for function module No. 2 (FM2)	37
Table 8-3: Pin assignment for function module No. 3 (FM3)	38
Table 8-4: Pin assignment for function module No. 4 (FM4)	38
Table 8-5: Special pin assignment	38
Table 8-6: External supply through terminal ST2	39
Table 8-7: Description of the pin assignment	41
Table 9-1: Assignment of the function modules	53
Table 9-2: I/O range	54
Table 10-1: Type declaration for DOS and Windows 3.1x	55
Table 10-2: Type declaration for Windows 95/NT	56
Table 10-3: Define value	56
Table 10-4: Filter time	80

1 DEFINITION OF APPLICATION

1.1 Intended use

The board **APCI-1710** must be inserted in a PC 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 **APCI-1710** must not be used as safety related part for securing emergency stop functions.

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

1.3 General description of the board

Data exchange between the **APCI-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 **APCI-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.
 - **24 V version (APCI-1710-24):** these inputs can also be used 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 board **APCI-1710** 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)

24 V version (ACPI-1710-24 V)

7 x 24 V inputs (signals A to G)

1 x 24 V output (signal H)

The use of the board **APCI-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 PC 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 change 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

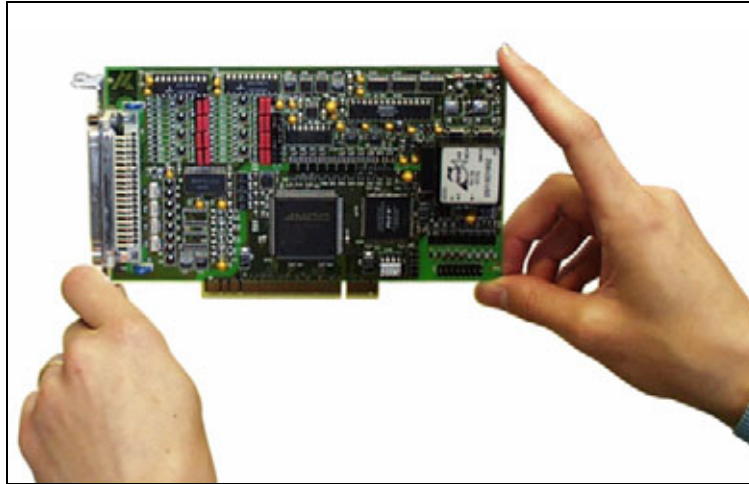
2.2 Personal protection

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)

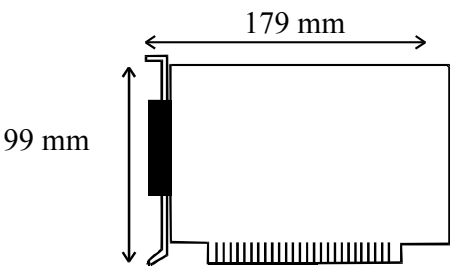
The board has been subjected to EMC tests in an accredited laboratory. The board complies with the limit values set by the norms IEC61326 as follows:

	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. 150 g
Installation in:	32/64-bit PCI slot, 5 V
Connection to the peripheral:	50-pin SUB-D male connector

Accessories¹:.....Standard cable **ST370-16**
Screw terminal panel **PX8000**



WARNING!
The supply lines must be installed safely against mechanical loads.

¹ Not included in the standard delivery.

4.3 Versions

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

Version	Format	Onboard function modules	Options
APCI-1710	PCI	4	5 V inputs (mass-related) Digital 5 V outputs
APCI-1710-24V	PCI	4	-

4.4 Limit values

Max. altitude: 2000 m
 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:

PCI BIOS from Version 1.0

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,
 Linux

Resources

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

Energy requirements:

Operating voltage of the PC: 5 V ± 5%
 Current consumption (without load): if + 5 V from PC: 1.15 A (±10 %)
 if + 24 V ext: 10 mA (±10 %)
 (for standard version APCI-1710)

4.4.1 Inputs

Number of inputs:	28
Input type:	
Differential inputs or rather TTL	16
Mass-related inputs	12

APCI-1710

Differential inputs, 5 V

Fulfills the EIA standard RS485

Nominal voltage:	5 VDC
Common-mode range:	+12/-7 V
Max. differential voltage:	± 12 V
Input sensitivity:	200 mV
Input hysteresis:	50 mV
Input impedance:	12 kΩ
Load resistance (type):	150 Ω in series with 10 nF
Signal delay:	120 nS (at nominal voltage)
Max. input frequency:	5 MHz (at nominal voltage)
“Open Circuit Fail Safe Receiver Design” “1” = inputs open	
ESD protection:	up to ±15 kV

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

Nominal voltage:	24 VDC
Input current (when nominal voltage):	11 mA (typical)
Logic input level:	U _H max.: 30 V
	U _H min.: 19 V
	U _L max.: 15 V
	U _L min.: 0 V
Signal delay:	120 ns (at nominal voltage)
Max. input frequency:	2.5 MHz (at nominal voltage)

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

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

APCI-170-24 V

24 V inputs (channels A to G). This board version is especially designed for 24 V – encoders designed. On the inputs only 24 V signals can be connected.

Nominal voltage: 24 VDC/11 mA

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

Logic input level: U_H max.: 30 V
 U_H min.: 19 V
 U_L max.: 15 V
 U_L min.: 0 V

4.4.2 Outputs**Differential outputs, 5 V**

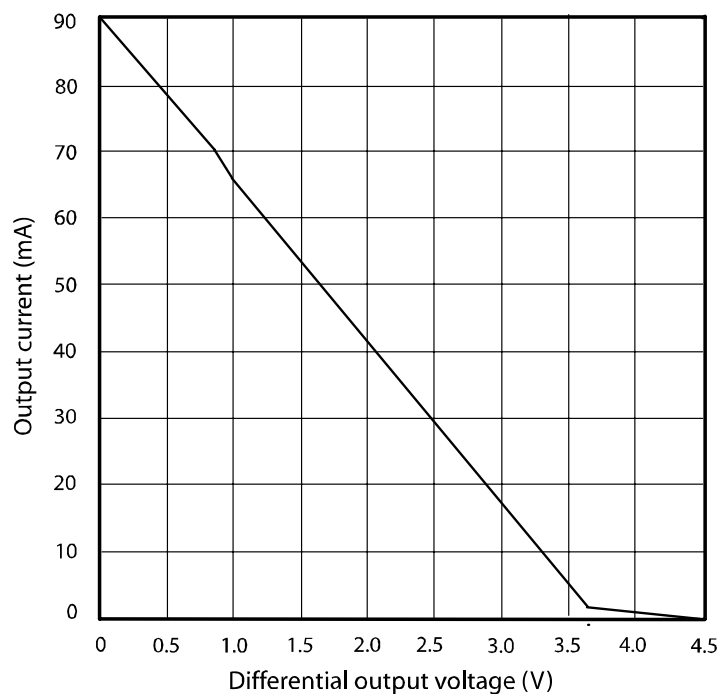
Fulfil the EIA standard RS485

Nominal voltage: 5 VDC

Max. output frequency: 5 MHz

Max. number of outputs: 8 (if not used as
differential inputs)

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



Digital outputs, 24 V

Output type:.....	High-Side (load on mass)
Number of outputs:	4
Nominal voltage:.....	24 VDC
Area 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 supply voltage)
Max. output current:.....	500 mA
Short circuit current / output at 24 V, $R_{last} < 0.1 R$:	1.5 A max. (output switches off)
ON resistance of the output (R_{DS} ON resistance):	0.4 R max.
Overtemperature:.....	170 °C (all outputs switch off)

APCI-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 (when excess temperature):	outputs switch off

Undervoltage protection (active when $V_{ext} < 5$ V)

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

Switch characteristics of the outputs

(V _{ext} = 24 V, T=25 °C, ohmic resistance: 500 mA)	
Switch ON time:.....	200 µs
Switch OFF time:	15 µs

Digital outputs, 5 V (OPTION)

Output type:.....	TTL
Number of outputs:	4
Nominal voltage:.....	5 VDC
Output current:	10 mA

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

Switch ON time:.....	0.06 µs
Switch OFF time:	0.02 µs

4.4.3 Safety

Optical isolation

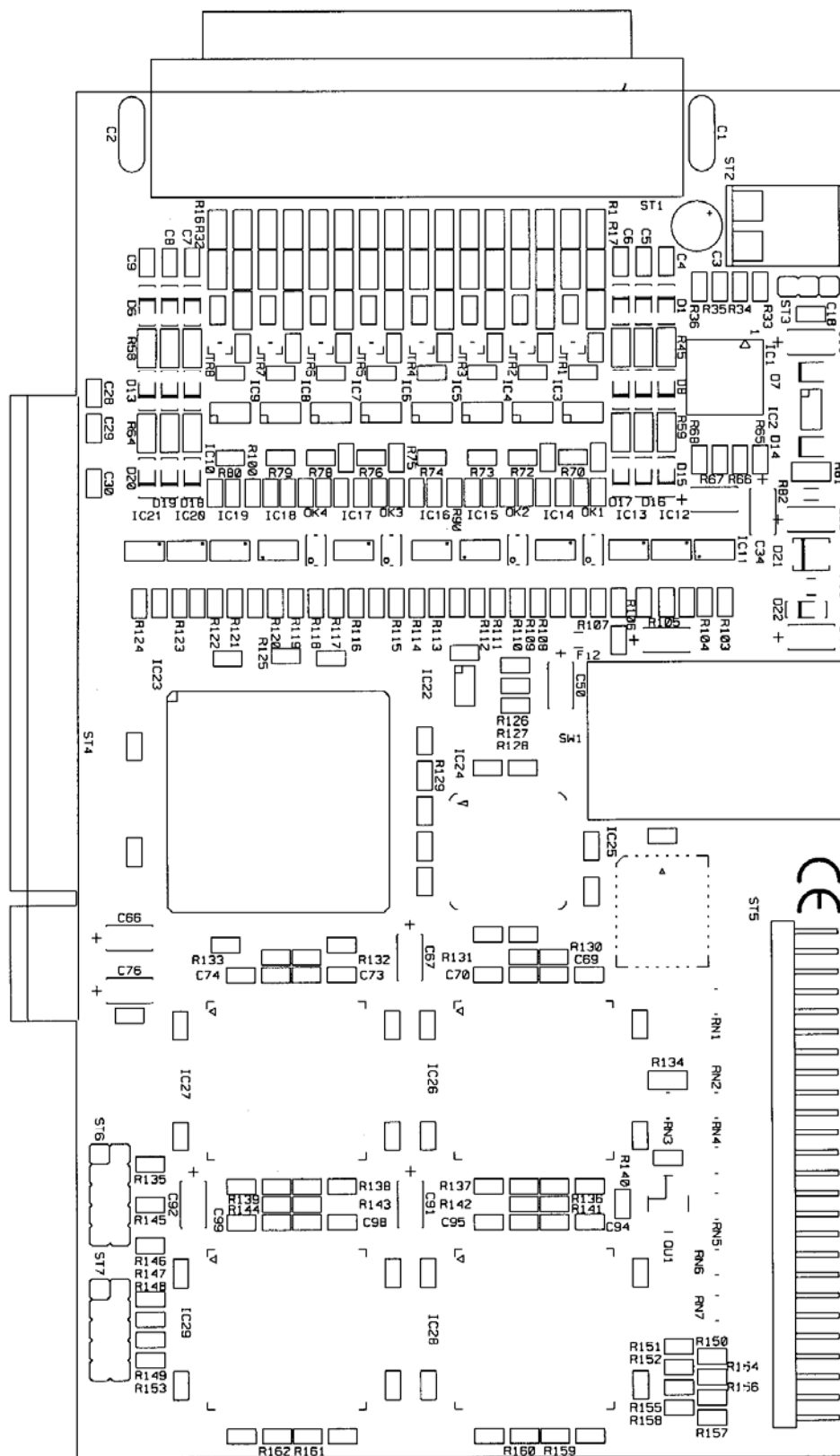
(DIN VDE 0411-100): 1000 V (from the PC to the external periphery)

Logic: positive

Creeping distance: 3.2 mm on the printed board

4.5 Component scheme

Figure 4-1: Component scheme



5 AVAILABLE FUNCTIONS OF THE APCI-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)

24 V version (APCI-1710-24V):

- 7 x 24 V inputs (signals A to G)
- 1 x 14 V output, TTL optional (signal H)

5.1.2 Maximal signal wirings of the APCI-1710

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 output lines on the board

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

Table 5-4: Maximal output 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)
TTL	24 digital channels configurable as inputs or outputs (PA0..7, PB0..7, PC0..7) 2 inputs and outputs (I,J) Only possible on 1 function module	
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 counter 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
TTL inputs and outputs	1		TTL
Gate measurement	8	2	TOR
Edge Time Measurement	8	2	ETM

5.2.3 Delivered manuals

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

Table 5-7: Delivered 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 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.

6.2 Selecting a free slot

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

Fig. 6-1: 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

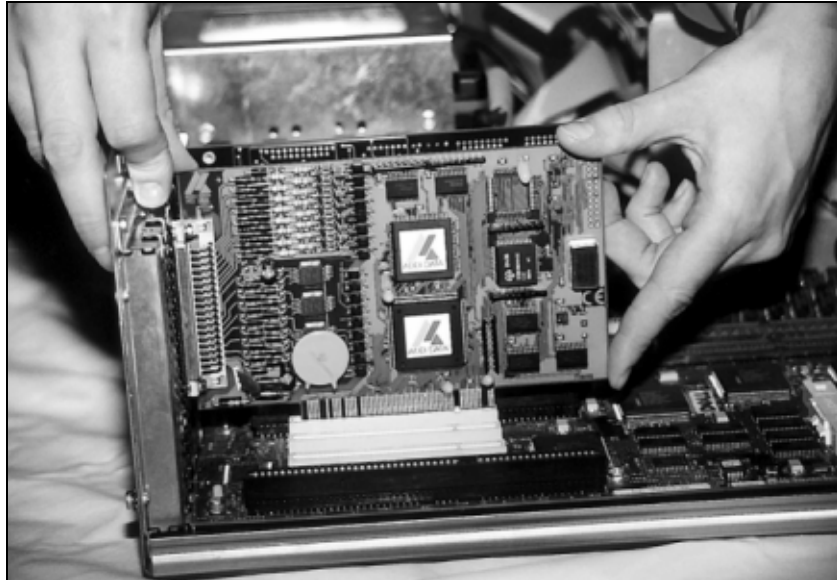
Discharge yourself from electrostatic charges.

Take the board out of its protective blister pack.

6.3 Plugging the board into the slot

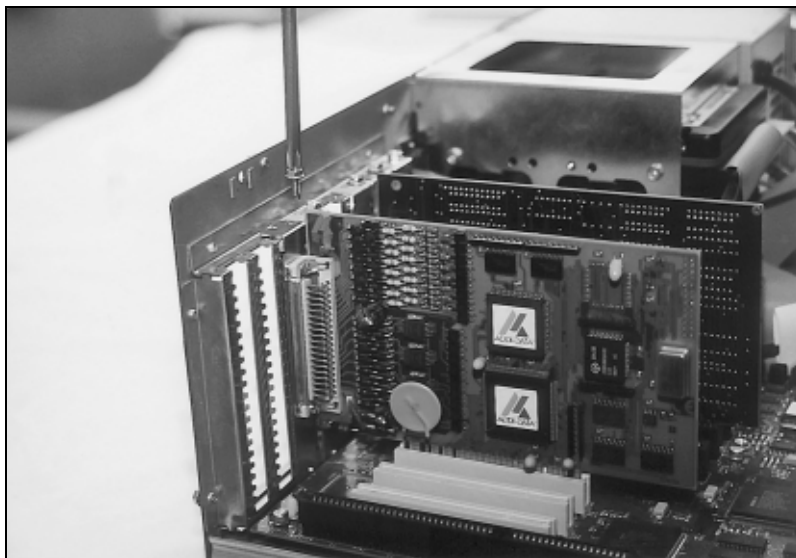
- ◆ Insert the board vertically into the chosen slot.

Fig. 6-2: Inserting the board



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

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



- ◆ Tighten all the loosen screws.

6.4 Closing the PC

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

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 board:
SET1719 program for the configuration of the function modules
- All functions that are implemented for the APCI-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

"Installations instructions for the PCI and ISA bus".

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

7.2 Configuration of the APCI-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.

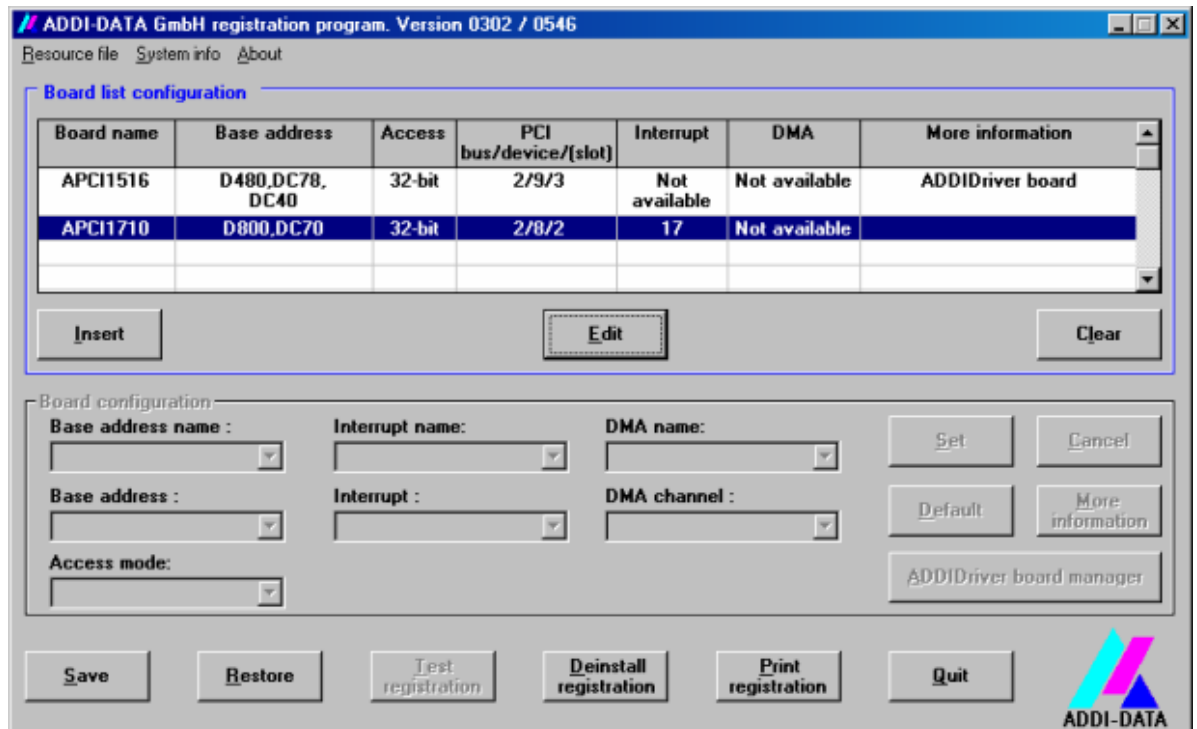


IMPORTANT!

Firstly, 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

**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 (for ISA boards only):

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.

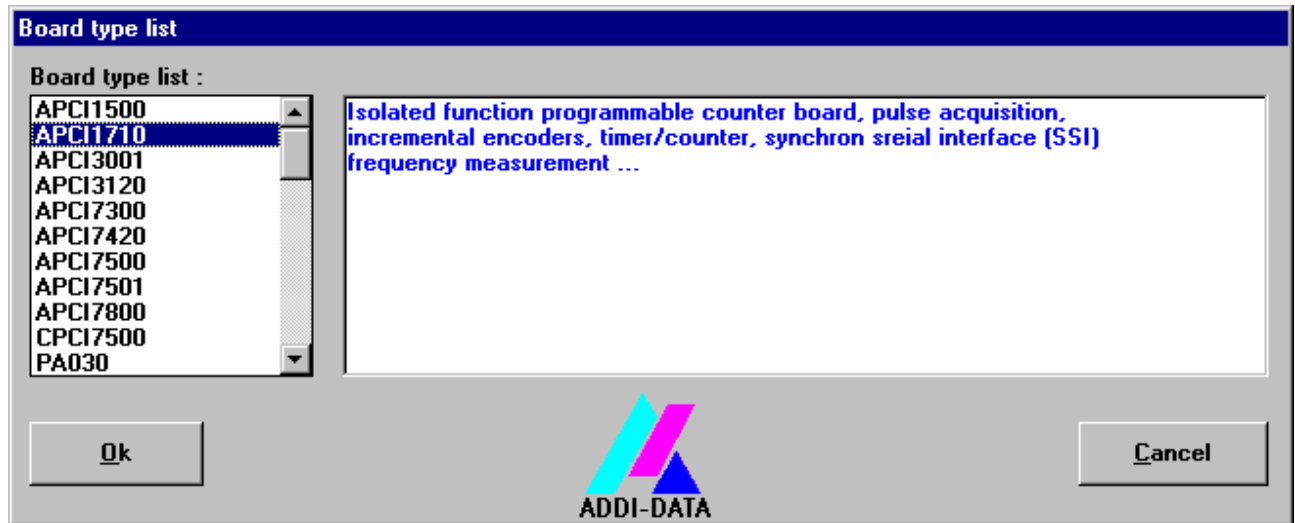
Click on Edit to confirm the entries or click twice on the selected board.

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 **APCI-1710** is not programmed with ADDIPACK, this button is deactivated.

Save:

Saves the parameter and registrates 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”.

You can check if the register is correct by a test. If the test is positive you can exit the ADDIREG program.

The board will be initialised with the set parameters and now can be distributed.

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 **APCI-1710** is operable.

The user also can set 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

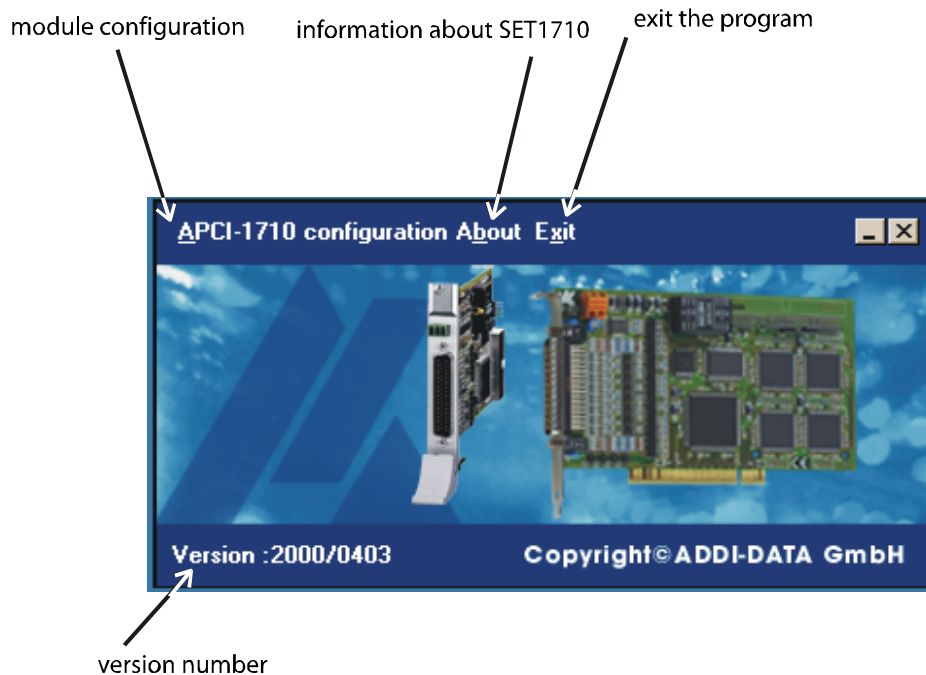


Fig. 7-5: Selection of a APCI-/CPCI-170



Under "About" you can find general information about the board and our customer service.

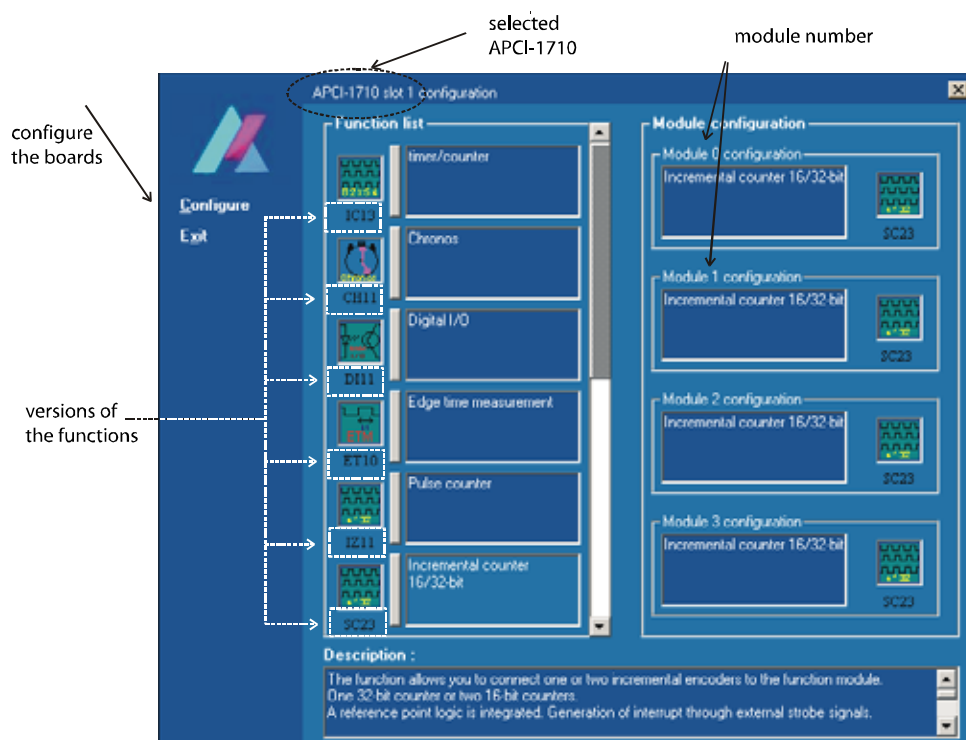
Fig. 7-6: General information



Quit with “Exit” the SET1710 program.

After having selected the **APCI-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.

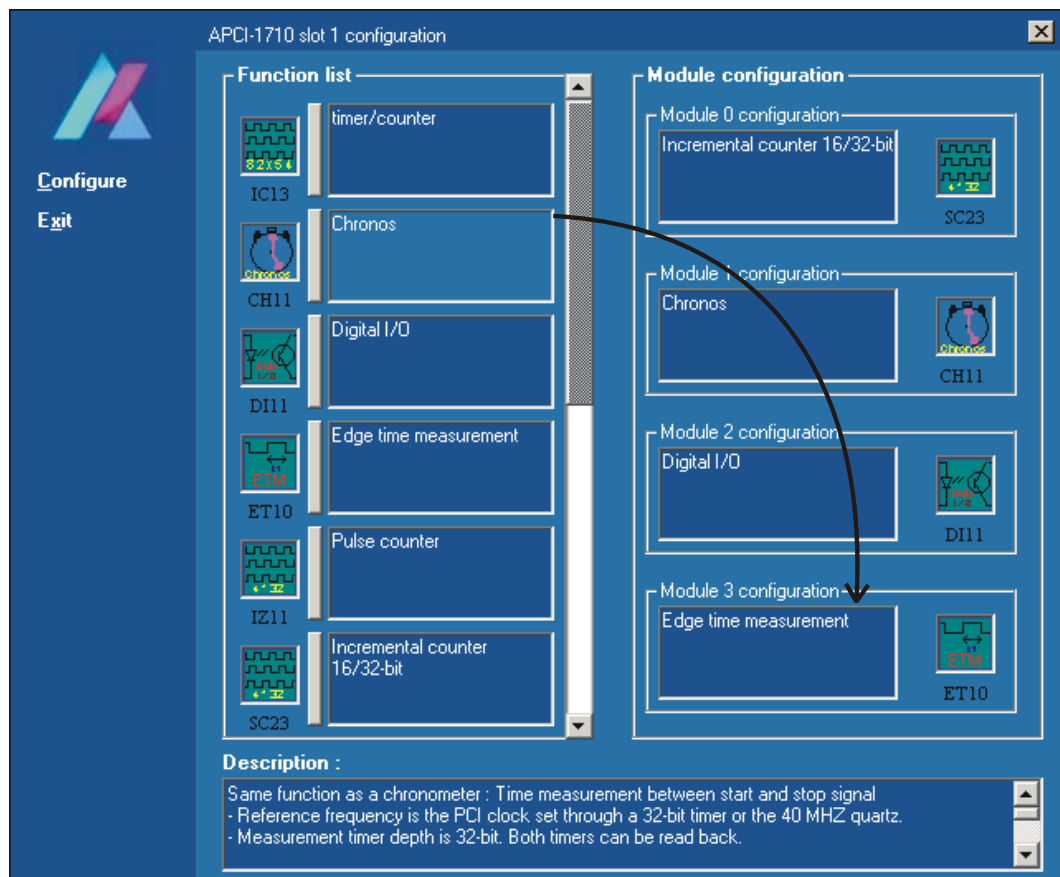
Exit:

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

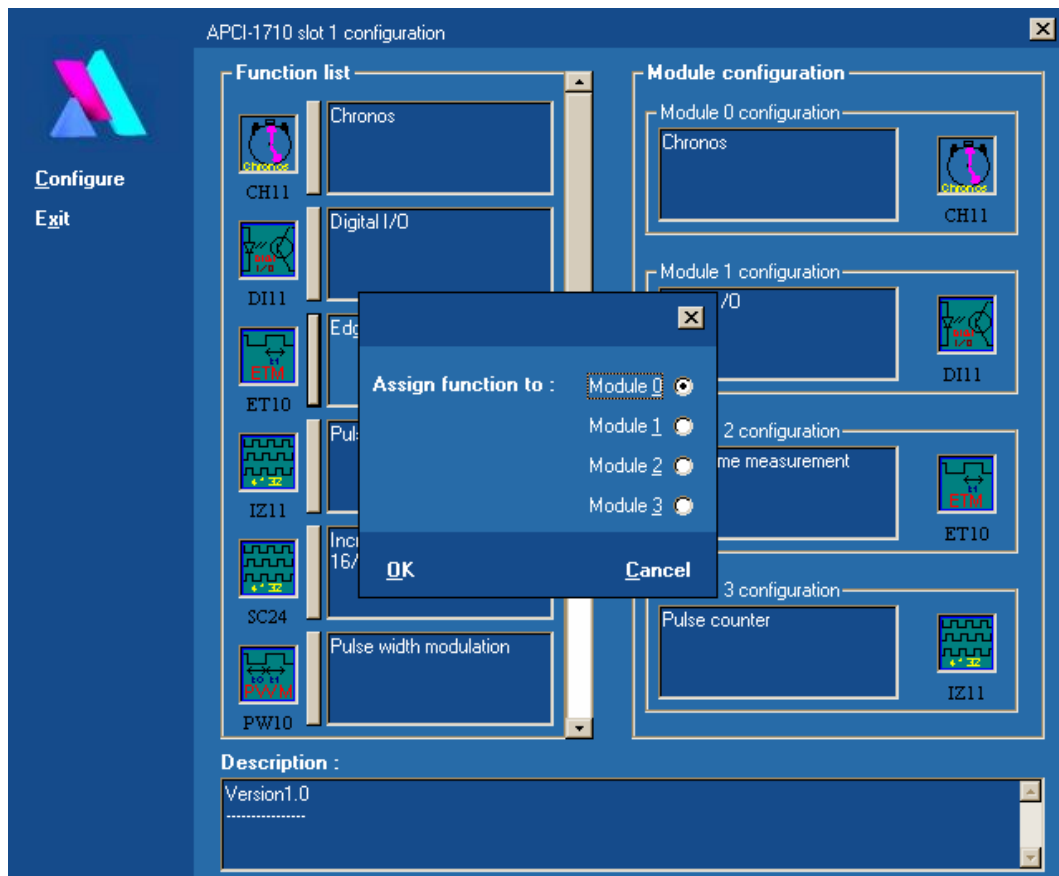


Click with the right mouse button on the function in the list and move it into the required module.

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 by 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

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	18	input or output	34	18	1	ext. GND for all inputs and outputs
35	FM 1 output	19	input or output	35		2	input or output
36	FM 2 output	20	input or output	36		3	input or output
37	FM 3 output	21	input or output	37		4	input or output
38	FM 4 output	22	input	38		5	input or output
39	FM 1 input	23	input	39		6	input
40	FM 2 input	24	input	40		7	input
41	FM 3 input	25	input	41		8	input
42	FM 4 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

**WARNING!**

Pin 34 of the front connector has a double assignment function. If you do not connect Pin 34 correctly, **your board can be destroyed.**

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	APCI-1710	APCI-1710-24 V
2	A1+	Input/Output ¹	Diff. / TTL	24 V input
3	A1-	Input/Output ¹	Diff. / TTL	- ²
4	B1+	Input/Output ¹	Diff. / TTL	24 V input
5	B1-	Input/Output ¹	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	APCI-1710	APCI-1710-24 V
10	A2+	Input/Output ¹	Diff. / TTL	24 V input
11	A2-	Input/Output ¹	Diff. / TTL	-
12	B2+	Input/Output ¹	Diff. / TTL	24 V input
13	B2-	Input/Output ¹	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

¹ On the 24 V signals (APCI-1710-24 V) only inputs can be connected

² Ax-, Bx-, Cx-, Dx- pins do not have a function on the 24 V board (see Fig. 9-5)

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

Pin	Description	Input/Output	APCI-1710	APCI-1710-24 V
18	A3+	Input/Output ¹	Diff. / TTL	24 V input
19	A3-	Input/Output ¹	Diff. / TTL	-
20	B3+	Input/Output ¹	Diff. / TTL	24 V input
21	B3-	Input/Output ¹	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	ACPI-1710	APCI-1710-24 V
26	A4+	Input/Output ¹	Diff. / TTL	24 V input
27	A4-	Input/Output ¹	Diff. / TTL	-
28	B4+	Input/Output ¹	Diff. / TTL	24 V input
29	B4-	Input/Output ¹	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	Input/Output	Function	Note
1	EXTGND	-	Reference potential	For all inputs and outputs
34	+ UREF Or 24 V ext.	Output/Input	- voltage output for TTL signals or - 24 V supply for 24 V outputs	Configurable through ST3

8.1.2 24 V supply for 24 V digital outputs (channel H)



WARNING!

Pin 34 of the front connector has a double assignment function. If you do not connect Pin 34 correctly, **your board can be destroyed.**

According to which input type is connected, the user shall observe the connection of the 24 V output channel H.

◆ Use of 5 V differential inputs

Channels A_{\pm} , B_{\pm} , C_{\pm} , D_{\pm} delivery status

- The 24 supply of the outputs H1 to H4 is conducted through Pin 34 of the front connector ST1. Jumper ST3 is on position A.

◆ Use of TTL inputs

The 24 V supply of the outputs H1 to H4 is conducted **through terminal ST2**. Jumper ST3 must be set on position B. Pin 34 is considered for the auxiliary voltage +UREF for the adaptation of the 5 V differential inputs on TTL signals.

Fig. 8-2: Terminal ST2

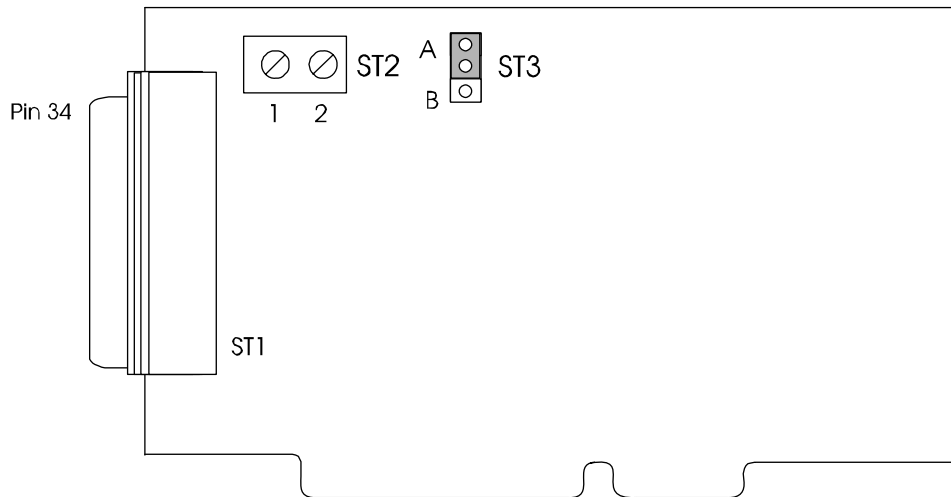


Table 8-6: External supply through terminal ST2

Pin	Description	Input/Output	Function	Note
ST2-1	24 V supply	24 V supply		
ST2-2	EXTGND	EXTGND	Reference potential	Connected with Pin 1 of the SUB-D connector

8.1.3 50-pin ribbon cable connector ST5

Fig. 8-3: Position of the connector ST5 on the board

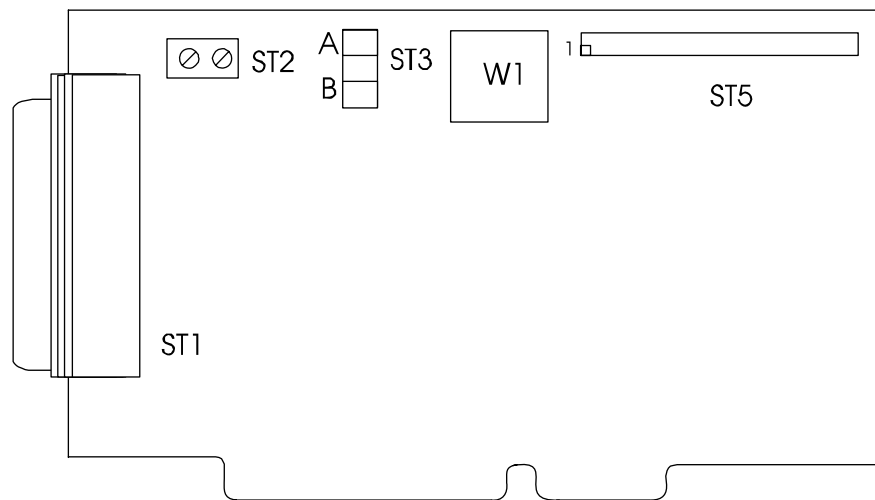


Fig. 8-4: Pin assignment of the ribbon cable connector

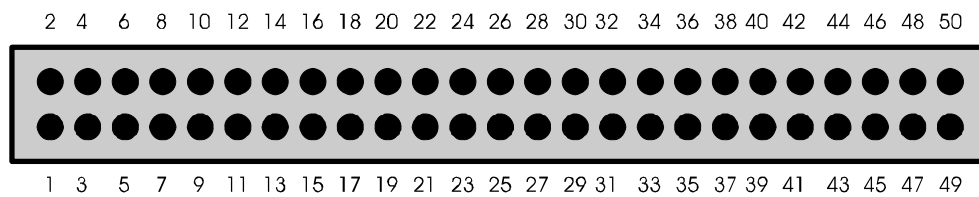


Table 8-7: Description of the pin assignment

Pin no. at the connector	Name	Description	Pin no. at FB8001
1	PC7 ¹	TTL, input or output; after reset: input	1
2	GND	PC GND, without isolation	34
3	PC6	TTL, input or output; after reset: input	18
4	GND	PC GND, without isolation	2
5	PC5	TTL, input or output; after reset: input	35
6	GND	PC GND, without isolation	19
7	PC4	TTL, input or output; after reset: input	3
8	GND	PC GND; without isolation	36
9	K1	TTL, outp.; same signal as H1 at front connector, FM1	20
10	GND	PC GND, without isolation	4
11	K2	TTL outp., same signal as H2 at front connector, FM2	37
12	GND	PC GND; without isolation	21
13	K3	TTL outp.; same signal as H3 at front connector, FM3	5
14	GND	PC GND; without isolation	38
15	K4	TTL outp.; same signal as H4 at front connector, FM4	22
16	GND	PC GND; without isolation	6
17	PA0	TTL, input or output; after reset: input	39
18	PA1	TTL, input or output; after reset: input	23
19	PA2	TTL, input or output; after reset: input	7
20	PA3	TTL, input or output; after reset: input	40
21	GND	PC GND, without isolation	24
22	PA4	TTL, input or output; after reset: input	8
23	PA5	TTL, input or output; after reset: input	41
24	PA6	TTL, input or output; after reset: input	25
25	PA7	TTL, input or output; after reset: input	9
26	V. ext		42
27	PB0	TTL, input or output; after reset: input	26
28	PB1	TTL, input or output; after reset: input	10
29	PB2	TTL, input or output; after reset: input	43
30	PB3	TTL, input or output; after reset: input	27
31	GND		11
32	PB4	TTL, input or output; after reset: input	44
33	PB5	TTL, input or output; after reset: input	28
34	PB6	TTL, input or output; after reset: input	12
35	PB7	TTL, input or output; after reset: input	45
36	V ext		29
37	PC0	TTL, input or output; after reset: input	13
38	PC1	TTL, input or output; after reset: input	46
39	PC2	TTL, input or output; after reset: input	30
40	PC3	TTL, input or output; after reset: input	14
41	GND		47
42	I4 ²	TTL, input or output; after reset: output, FM4	31
43	J4 ²	TTL, input or output; after reset: output, FM4	15
44	I3	TTL, input or output; after reset: output, FM3	48
45	J3 ²	TTL, input or output; after reset: output, FM3	32
46	V ext	PC + 5 V voltage supply	16

47	I2 ²	TTL, input or output; after reset: output, FM2	49
48	J2 ²	TTL, input or output; after reset: output, FM2	33
49	I1 ²	TTL, input or output; after reset: output, FM1	17
50	J1 ²	TTL, input or output; after reset: output, FM1	50

¹ PA, PB and PC : Pullup on 5 V

² Serial resistance 100 Ω , PD

PA, PB, PC and PD can be used through the function module “TTL I/O”.

8.2 Connection of mass-related inputs and outputs

Fig. 8-5: Connection of a mass-related input

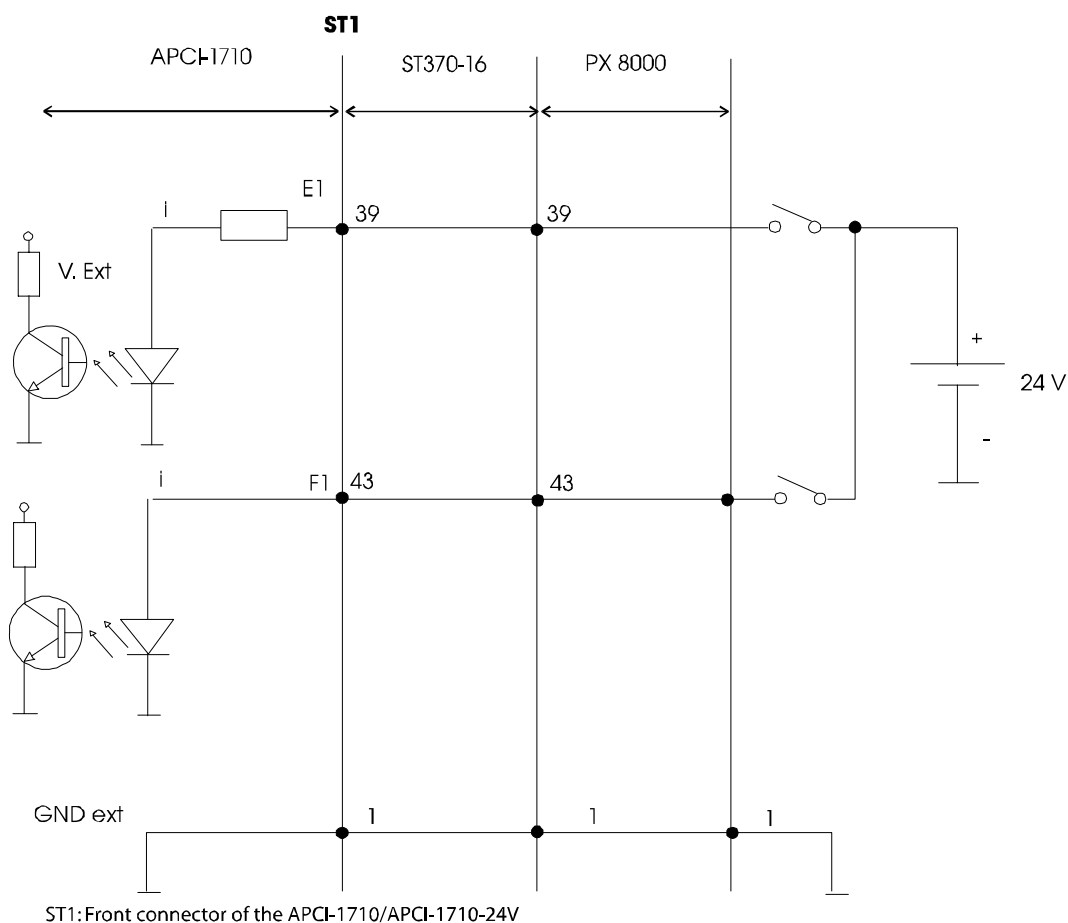
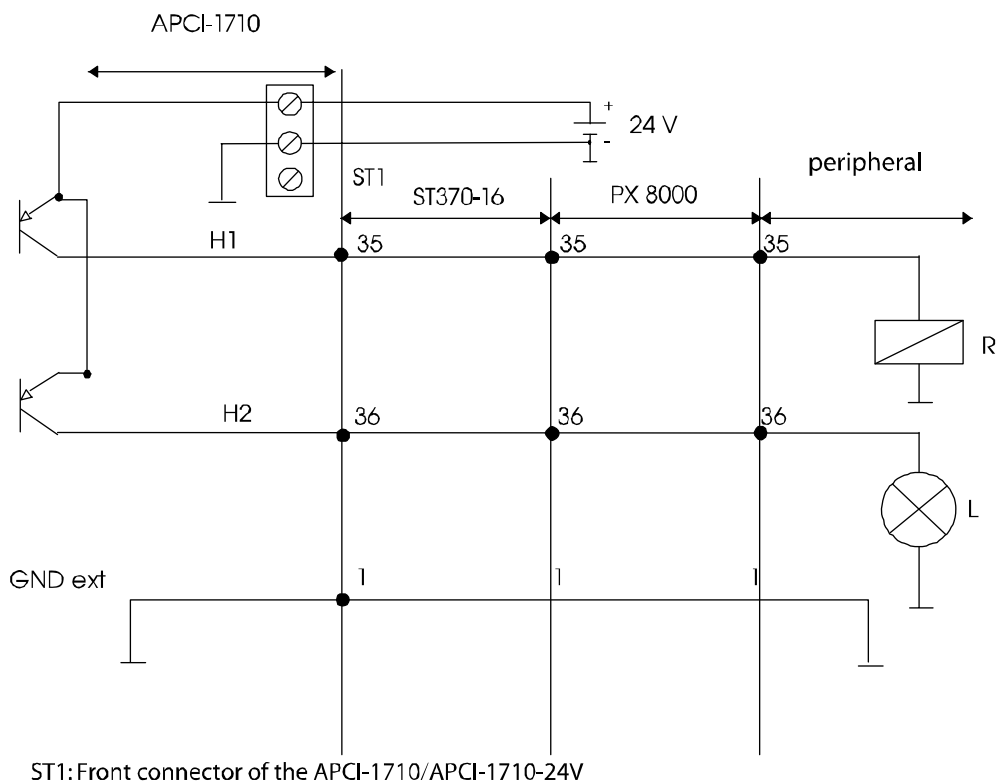


Fig. 8-6: Connection of a mass-related output

8.3 Connection of the differential digital I/O

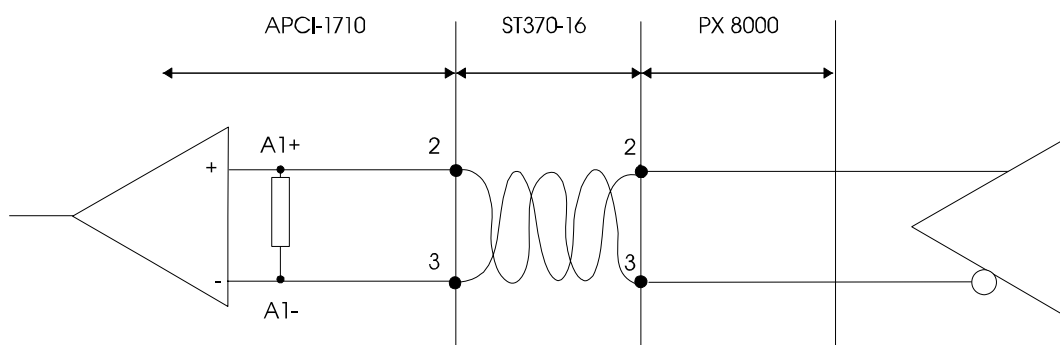
Fig. 8-7: Connection of a differential input

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

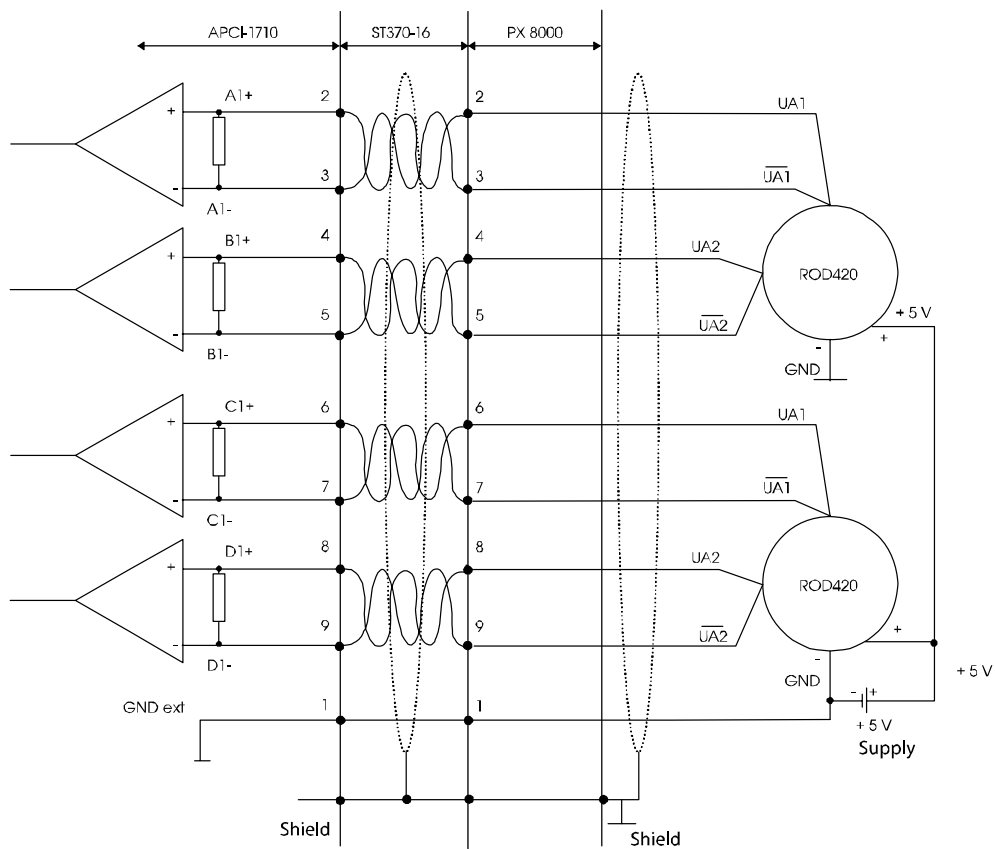
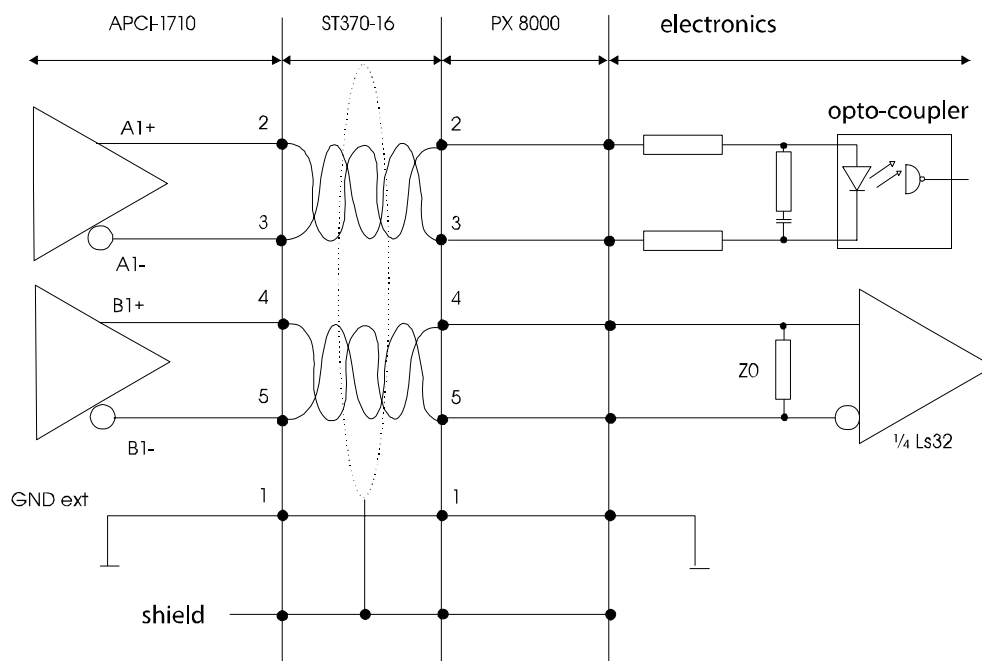
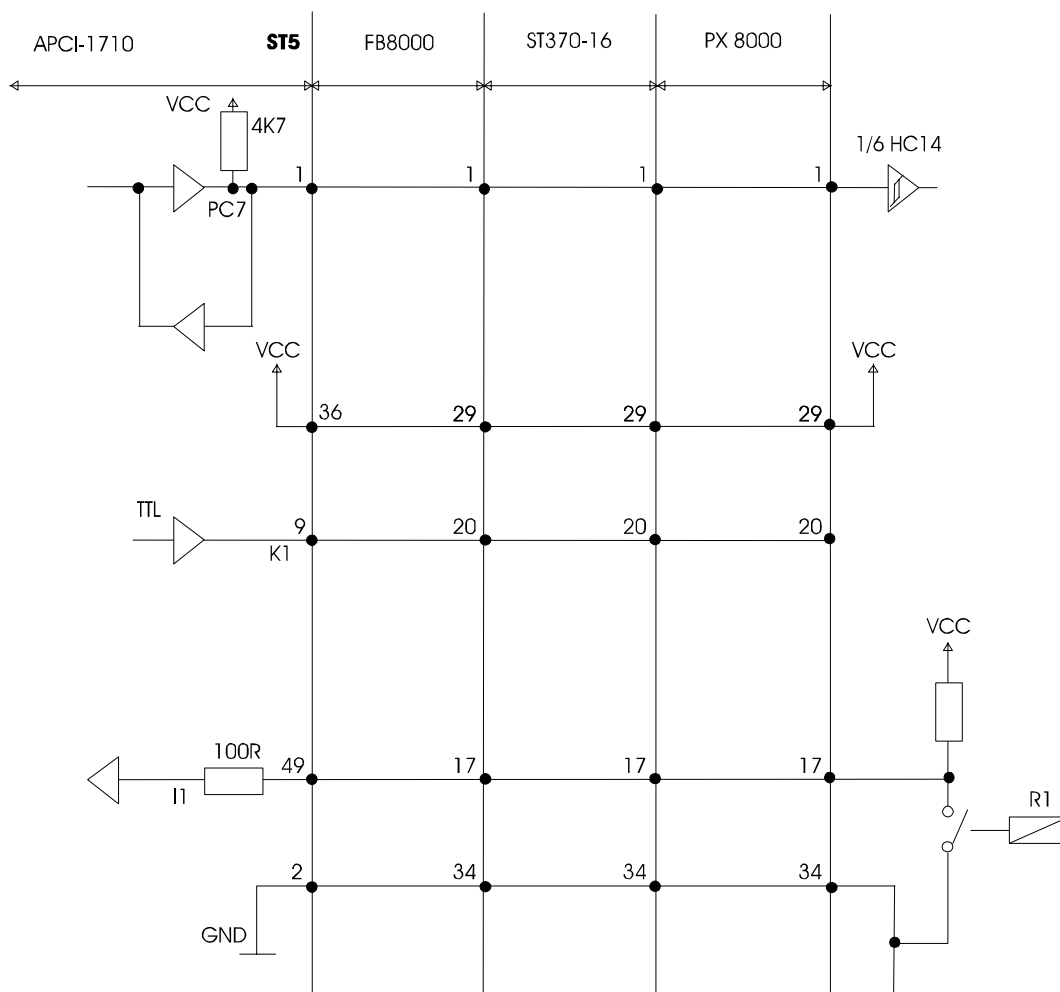


Fig. 8-9: Connection of a differential output



8.4 Connection at the TTL inputs and outputs

Fig.8-10: Connection at TTL inputs and outputs



ST5: Connector of the APCI-1710/APCI-1710-24V for connecting ribbon cable FB8000

9 FUNCTIONS OF THE BOARD

9.1 Description

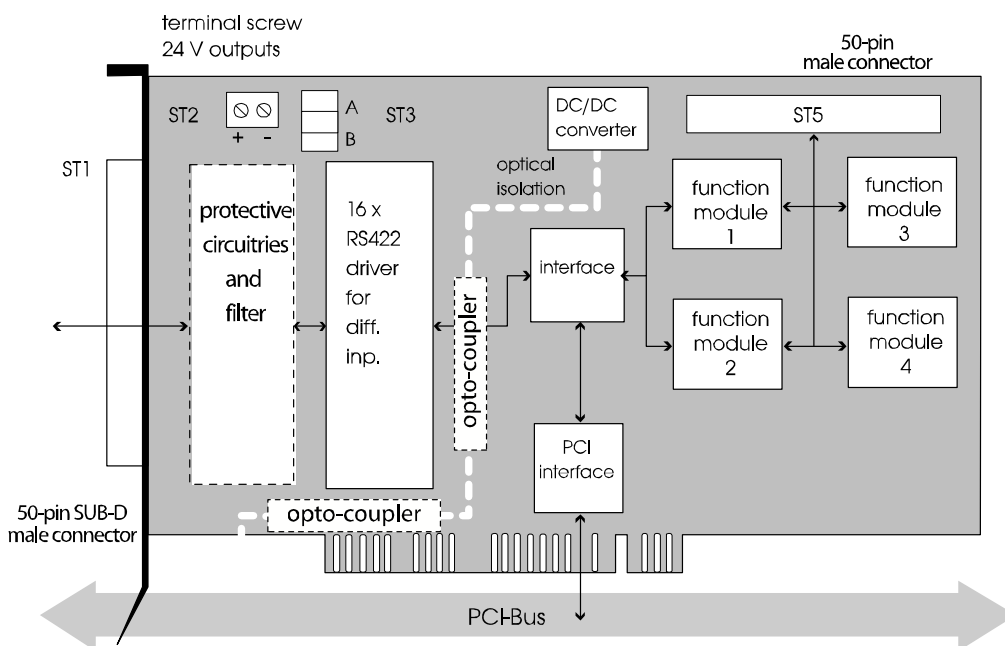
The **APCI-1710** is an extension board for the PCI bus and it is compatible with the PCI specification 2.1.. The board is used for the processing of digital signals with focus on “counting and time measurement”.

The digital signals are connected through a 50-pin SUB-D front connector ST1 to the “function modules” of the **APCI-1710** board. They are separated optically through opto-couplers.

A second 50-pin connector ST5 is assembled only for the connection of TTL signals to the board. These signals are not isolated optically from the PC.

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

Fig. 9-1: Block diagram of the APCI-1710



The user can reprogram the board without uninstalling it. This has the following advantages:

- 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 also connected to 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. 9-2: block diagram of digital inputs and outputs) (1 function module)

The lines are divided according to the board as follows:

APCI-1710

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)

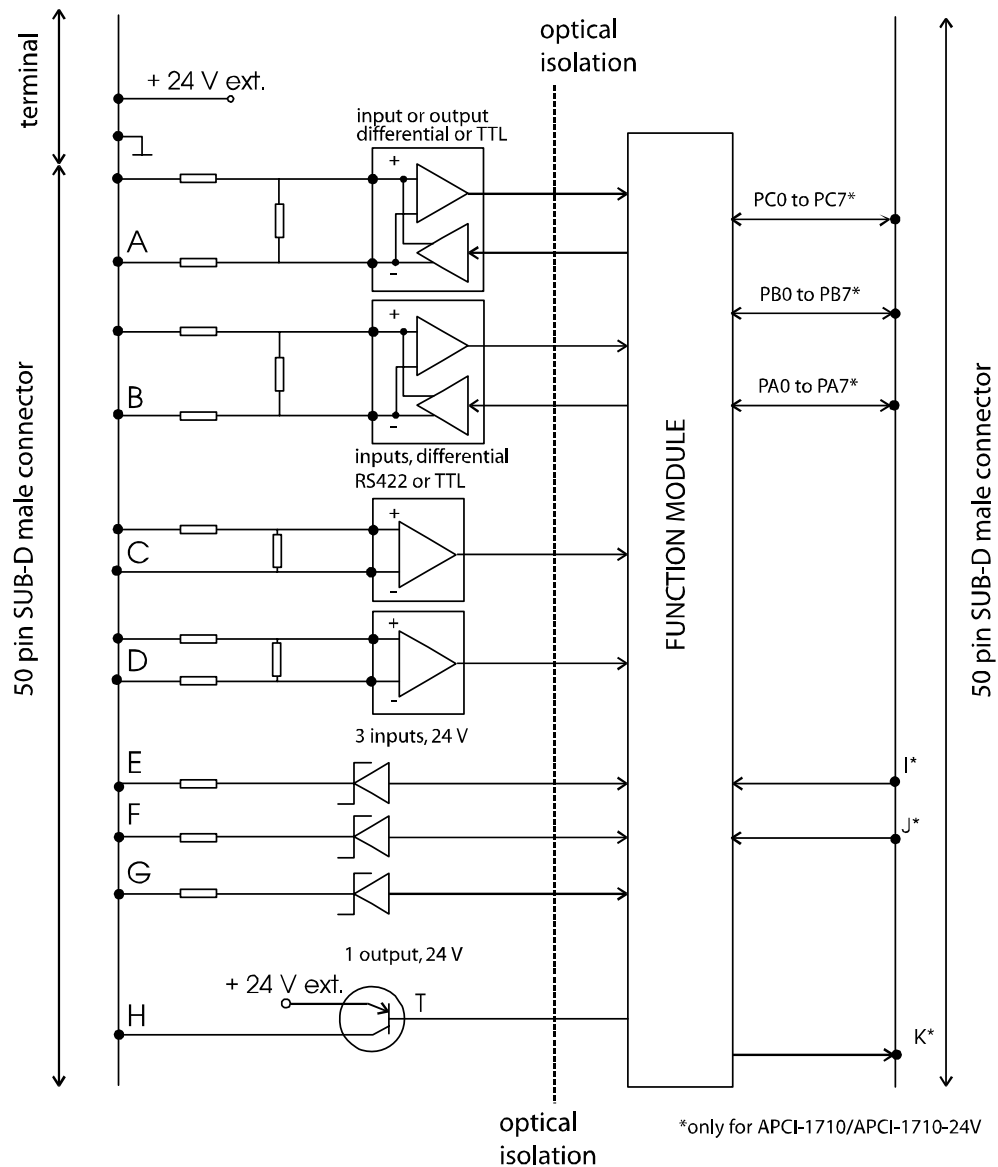
- 2 x TTL, RS422 (signals A, B)

APCI-1710-24 V

- 7 x 24 V inputs (signals A to G)
- 1 x 24 V output, TTL optional (signal H)

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. 9-2: Block diagram of the digital inputs and outputs (1 function module)



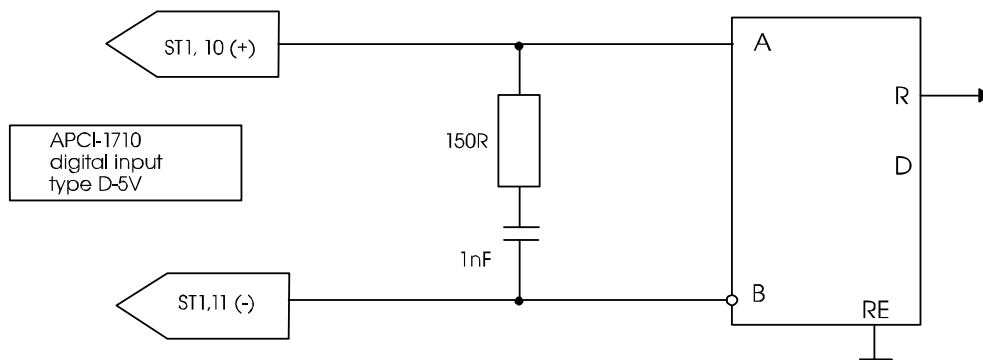
9.2.2 Inputs

The inputs are distinguished as follows:

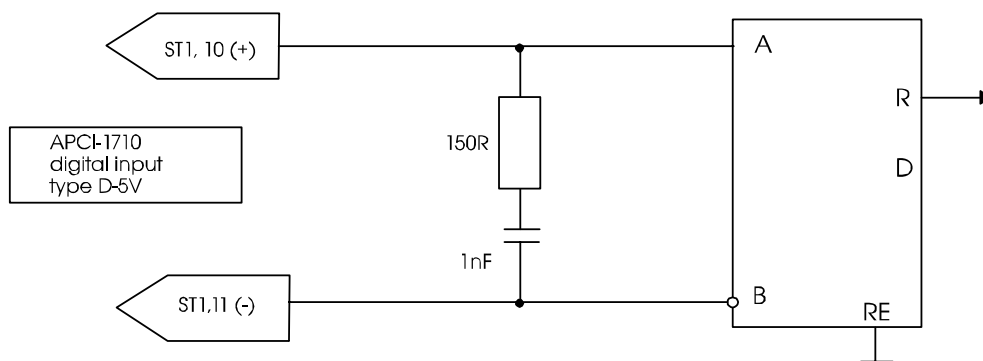
- differential inputs for very fast signals
- mass-related inputs

Differential inputs

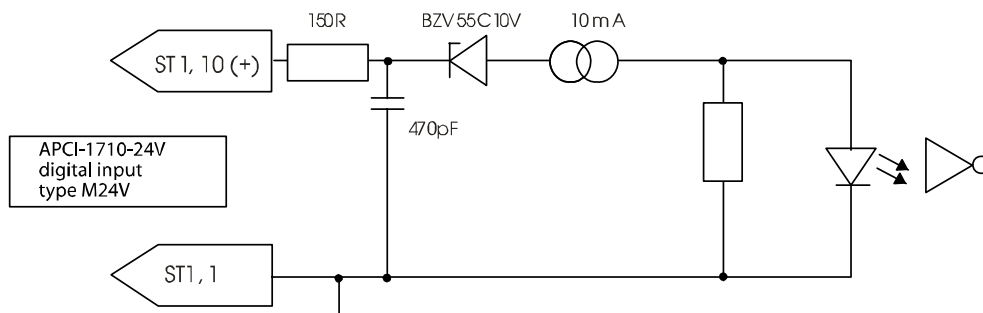
Max. 4 differential inputs (A, B, C and D) are available for each “function module”. The levels in the standard delivery correspond with the RS485 (5 V) standard.

Fig. 9-3: Basic circuit of the differential inputs (5 V)

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

Fig. 9-4: Basic circuit of the differential inputs 5 V; used as TTL inputs

In version **APCI-1710-24** the differential inputs (A to D) are individually designed for the connection to a 24 V impulse transmitter / signal transmitter.

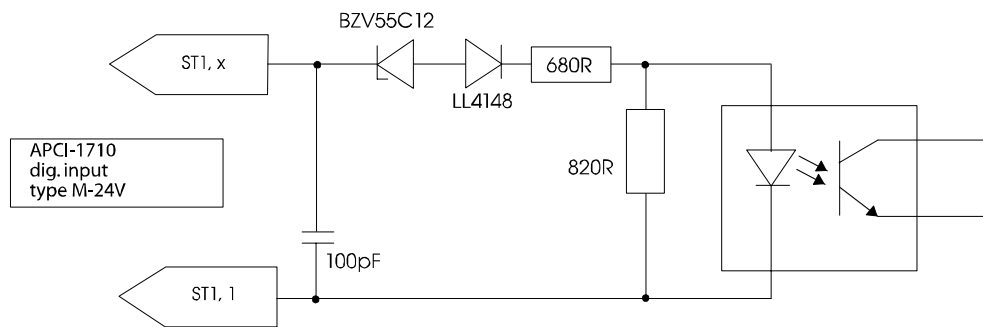
Fig. 9-5: Basic circuit of the 24 V inputs (APCI-1710-24V)

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).

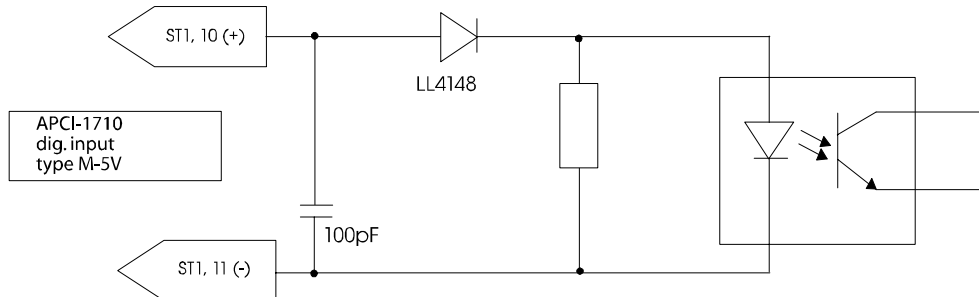
These inputs have a common ground line.

Fig. 9-6: Basic circuit of the digital inputs 24 V



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

Fig. 9-7: Basic circuit of the digital inputs 5 V (OPTION)



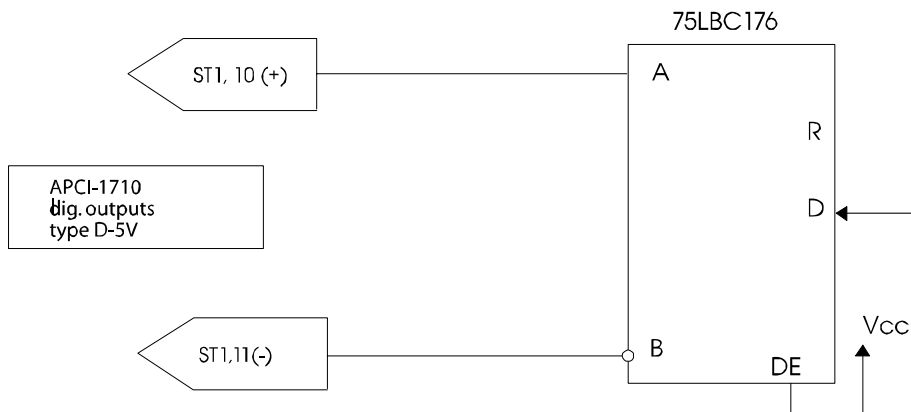
9.2.3 Outputs

The outputs are distinguished as follows:

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

Differential outputs

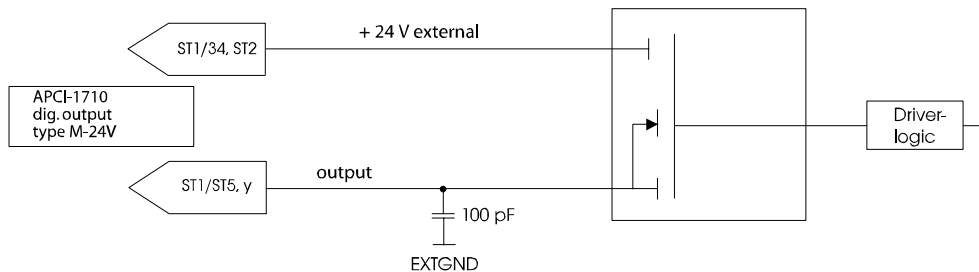
Max. 2 differential outputs (A and B) are available for one “function module”. The levels in the standard delivery correspond with the RS485 (5 V) standard. In this case A and B can not be used as common inputs.

Fig. 9-8: Basic circuit of the digital outputs 5 V – differential**Mass-related outputs**

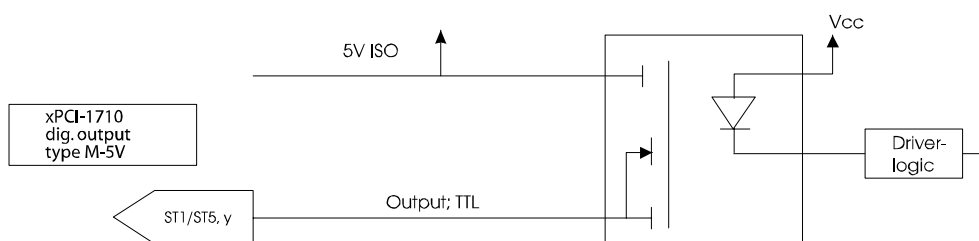
Max. one mass-related output (H) is available for one function module. The levels in the standard delivery correspond with the 24 V standard. (IEC1131-2 / High-Side driver).

Fig. 9-9: 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 also be delivered on request as opto-coupler output (TTL compatible).

Fig. 9-10: Basic circuit of a digital output H – 5 V (OPTION)

9.3 TTL inputs and outputs

At the connector ST5 of the APCI-1710 are further I/O, GND and Vcc of the PC available. These signals must correspond with the TTL level and be treated carefully in order not to damage the board if other signals shall be connected.

With the cable FB8001 the board can be connected through the connector ST5 to the peripheral.

9.3.1 Common signals for all function modules

The signals PA0 to PA7, PB0 to PB7 and PC0 to PC7 are connected with all “function modules”. They are available for max. one “function module” (e.g. “TTL I/O”) available.

Example: Only the function module No. 4 (FM4) is programmed with the function TTL I/O. The signals can be set as inputs or outputs.

9.3.2 Single signals

2 TTL inputs and outputs (I and J) are available for each “function module”. **These can be used only together with function module TTL I/O.**

9.4 PCI bus interface

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

Hereof result the following advantages:

- the PCI/CompactPCI bus can “Plug & Play”, this means the software sets the addresses, interrupts automatically.
- the board is identified automatically through the software program
- the **APCI-1710** allows a 32-bit access on the peripheral for faster data transfer.
- 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 APCI-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 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 SCxx			
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 (AP), through the connector assignment or through a reprogramming of the function module (no jumper on the board).

10 STANDARD SOFTWARE

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

	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 10-2: Type declaration for Windows 95/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

Table 10-3: Define value

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

10.2 Software functions

10.2.1 Initialisation



IMPORTANT!

The following chapter lists the common functions for each function module.

You can find the single software functions according to the function of the board **APCI-1710** 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 für Windows NT or Windows 95/98. - DLL_COMPILER_PASCAL: Programming in Pascal or Delphi. - DLL_LABVIEW: Programming in Labview.
--------------------------	---

- Output:

No output signal.

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 **APCI-1710** and shows the slot number of each board.

Each parameter *pb_SlotNumberArray* contains the slot number (1 to 8) of a **APCI-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 **APCI-1710** boards that are installed in the PC. If the Return Value shows "0" then no **APCI-1710** 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 saves the slot number. The user gets back a handle so that the next functions can be used. Handles enable the administration of several 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 ¹ 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.

¹ 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 base address, the interrupt and the slot number of the board.

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 anytime you want to exit 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.

Aufgabe:

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 only be used for C/C++ and Pascal for DOS b

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 xPCI-1710 Karte
VOID	v_FunctionName	Name der Benutzer-Interruptroutine

- Output:

No output signal.

Task:

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

At the first calling 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 important at the first calling. From the second calling of the function on (next boards), interrupts are enabled.

Interrupt

If an interrupt is generated, the user interrupt routine is called up by 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 must 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 xPCI-1710-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.

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 being 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* und *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.

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 important at **the first calling**. From the second calling of a function on (next boards) interrupts are enabled.

Interrupt

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

b_ModuleMask Mask of the module, which has generated the interrupt

(see table of the interrupt mask in the respecting manual)

ul_InterruptMask Mask of the events, which have generated the interrupt
(see table of the interrupt mask in the respecting manual)

ul_CounterLatchValue The latched values of the timer are 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 xPCI-1710
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 on 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
---------	----------------------	---



WICHTIG!

For Windows NT and Windows 95, 4 rings are 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 cannot access to the variable of ring 3 and therefore shall use the Shared Memory.
- Ring 0 und ring 3 have an indicator, which identifies this Shared Memory. Both rings have different addresses.

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

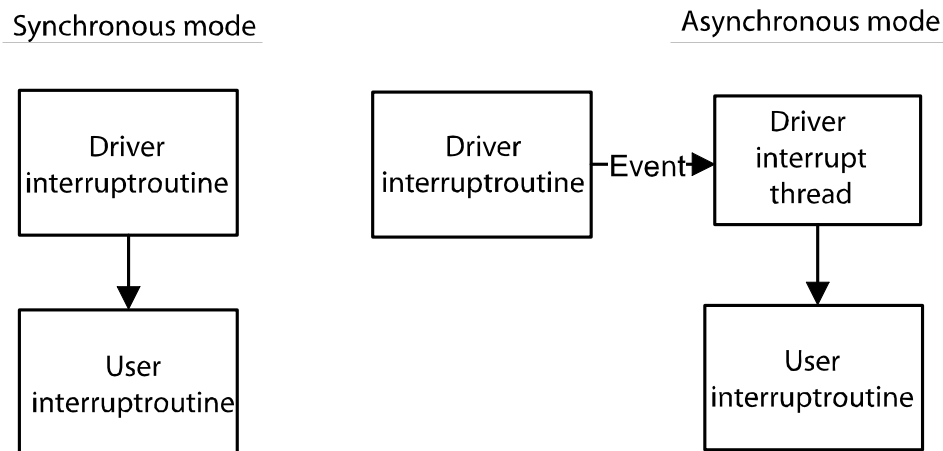
If an interrupt is generated, the user interrupt routine of the system is called.

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 (synchronous 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 highest priority (31) in the system.

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

	SYNCHRONOUS MODE
ADVANTAGES	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.
LIMITS	A debug of the user interrupt routine is not possible.
	The user interrupt routine cannot call the Windows API functions.
	The user interrupt routine cannot call the functions, which have access to the shared variable. However, the user can use a shared memory.
	The user interrupt routine can only call 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.

	ASYNCHRONOUS MODE
ADVANTAGES	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"
LIMITS	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 function. 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:

<code>VOID v_FunctionName</code>	(<code>BYTE b_BoardHandle,</code> <code>BYTE b_ModuleMask,</code> <code>ULONG ul_InterruptMask,</code> <code>ULONG ul_CounterLatchValue,</code> <code>BYTE b_UserCallingMode,</code> <code>VOID * pv_UserSharedMemory)</code>
<code>v_FunctionName</code>	Name of the user interrupt routine
<code>b_BoardHandle</code>	Handle of the APCI-1710 , which has generated the interrupt.
<code>b_ModuleMask</code>	Mask of the module, which has generated the interrupt (see table in the interrupt of the respecting manual)
<code>ul_InterruptMask</code>	Mask of the event, which has generated the interrupt (see table of the interrupt mask in the respecting manual)
<code>ul_CounterLatchValue</code>	The latched values of the counter are returned. (See table of the interrupt mask in the respecting manual)
<code>b_UserCallingMode</code>	<code>APCI1710_SYNCHRONOUS_MODE</code> : The user routine is called directly from the driver interrupt routine <code>APCI1710_ASYNCCHRONOUS_MODE</code> : The user interrupt routine is called directly from the driver interrupt thread.
<code>pv_UserSharedMemory</code>	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 have no meaning. Use the function: `"i_APCI1710_TestInterrupt"`.

<code>BYTE</code>	<code>b_UserCallingMode,</code>
<code>ULONG</code>	<code>ul_UserSharedMemorySize,</code>
<code>VOID **</code>	<code>ppv_UserSharedMemory,</code>
<code>VOID</code>	<code>v_FunctionName (BYTE b_BoardHandle,</code>

```

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 xPCI-1710, 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, 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 PCI-1710
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_TimeBase	Selection of the filter clock - APCI1710_30 MHZ: The PC has a PCI bus clock of 30 MHz - APCI1710_33MHZ: The PC has a PCI bus clock of 33 MHz - APCI1710_40 MHz: The board has a 40 MHz quartz
BYTE	b_Filter	Selection of the filter. (See Table 10-4: Filter time)

-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: 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 is 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: This function is in not available in any of the function modules.

11 INDEX

A

Accessories 13

B

Block diagram 48

C

Changing the registration of an existing board 31
 Component scheme 19
 Configuration of a new board 29
 Configuration with ADDIREG 26
 Connecting the peripheral 38
 Connection of a mass-related input 44
 Connection of a mass-related output 45
 Connection of the differential digital I/O 45

D

Definition of application 9
 Digital inputs and outputs
 Description 49
 Dimensions 13

E

EMC
 Electromagnetic compatibility 13
 Energy requirements 14

F

Functions 22
 Functions of the board 48

H

Handling of the board 12

I

Initialisation
 Software functions 59
 Initialisation input filter
 Software functions 81
 Inputs
 Limit values 15
 Installation of the board 24
 Intended use 9
 Internet 36
 Interrupt
 Software functions 66

L

Limit values 14
 Loading a function into a function module
 SET1710 32

M

Manuals 23
 Module configuration with SET1710 33

O

Optical isolation 18
 Outputs
 Limit values 16

P

PCI bus interface
 Description 55
 Personal protection
 User 11
 Physical set-up of the board 13
 Pin assignment 38

Q

Qualification
 User 11

R

Registration of a new board 31
 Registration program ADDIREG 27

S

SET1710 32
 Setting a module configuration 35
 Signals 21
 Slots 24
 Software 26
 Standard software 57

T

Technical data 13
 TTL inputs and outputs
 Description 54

U

Update 36
 Usage restrictions 9

V

Versions 14

W

Weight 13