



DIN EN ISO9001 certified



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



**Technical description**

**ADDIALOG APCI-/CPCI-3001**

**Analog input channels  
for the PCI / CompactPCI bus**

3<sup>rd</sup> edition 07/2001

## 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
- nonobservance 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 media).
- 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.

Burr-Brown is a registered trademark of Burr-Brown Corporation

Intel is a registered trademark of Intel Corporation

CompactPCI is a registered trademark of the PCI Industrial Computer Manufacturer Group (PICMG)

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

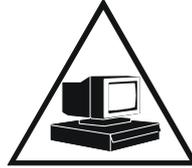
***The original version of this manual is in German. You can obtain it on request.***

# WARNING

**In case of improper handling and if the board is not used for the purpose it is intended for:**



**people may be injured**



**the board, PC and peripheral may be damaged**



**the environment may be polluted**

**★★★ Protect yourself, other people and the environment ★★★**

- **Do read the safety leaflet!**

If this leaflet is not with the manual, please contact us.

- **Observe the instructions in the manual!**

Make sure that you have not forgotten any step. We are not liable for damage resulting from a wrong use of the board.

- **Symbols used**



**WARNING!**

It designates a possibly dangerous situation.

If the instructions are ignored **the board, PC and/or peripheral devices may be damaged.**



**IMPORTANT!**

designates hints and other useful information.

- **Do you have any question?**

Our technical support is always glad to help you.



# Declaration of Conformity

This declaration is valid for the following product:

**ADDIALOG APCI-3001, CPCI-3001**  
**Analog inputs**

It is made by

ADDI-DATA GmbH  
Meß- und Steuerungstechnik  
Dieselstraße 3  
D-77833 Ottersweier

in sole responsibility and is valid on the understanding that the product is competently installed, used and maintained, according to the respective security regulations as well as to the manufacturer's instructions regarding its intended use.

This declaration states that the product complies with following EC Directives:

- **EWGRL 336/89 of 3.05.1989**
- **EWGRL 31/92 of 28.04.1992**
- **EWGRL 68/93 of 22.07.1993**

This declaration is valid for all units manufactured according to the regulations and procedures of the quality management system (DIN EN ISO 9001 certification).

Following norms have been applied to test the product regarding electromagnetic compatibility:

- **EN55011/03.91**
- **EN55022/08.94**
- **EN50082-2/03.95**

We point out that

- the conformity and herewith the permission of use expire if the user alters the product without consulting with the manufacturer.
- non-skilled users are to have the operational area of the product and the requirements resulting from it checked prior to putting into operation.
- by using this product in appliances coming under the EC EMC Directive, the user is to make sure they are conform to its regulations prior to putting into operation.
- by using this product in machines / installations coming under the EU Machine Directive, the user is to make sure they are conform to its regulations prior to putting into operation.

A copy of the EMC tests is at your disposal on request.

APCI-3001: 17<sup>th</sup> February 1999  
CPCI-3001: 7<sup>th</sup> September 2000

---

Antonio Agnetti  
Legally valid signature of the manufacturer

<b>1</b>	<b>INTENDED PURPOSE OF THE BOARD</b> .....	<b>1</b>
1.1	Limits of use .....	1
<b>2</b>	<b>USER</b> .....	<b>2</b>
2.1	Qualification .....	2
2.2	Personal protection .....	2
<b>3</b>	<b>HANDLING THE BOARD</b> .....	<b>3</b>
<b>4</b>	<b>TECHNICAL DATA</b> .....	<b>4</b>
4.1	Electromagnetic compatibility (EMC) .....	4
4.2	Physical set-up of the board .....	4
4.3	Options .....	5
4.4	Versions .....	5
4.5	Limit values .....	5
4.6	Component scheme .....	8
<b>5</b>	<b>INSTALLATION</b> .....	<b>10</b>
<b>5.1</b>	<b>Jumper settings</b> .....	<b>11</b>
5.1.1	Jumper location and settings at delivery .....	11
5.1.2	Jumper settings according to the function used .....	12
<b>5.2</b>	<b>Inserting the board</b> .....	<b>13</b>
5.2.1	Installing an APCI-3001 board .....	13
	Selecting a free slot .....	13
	Plugging the board .....	13
	Closing the PC .....	14
5.2.2	Installing a CPCI-3001 board .....	14
<b>6</b>	<b>BOARD CONFIGURATION WITH ADDIREG</b> .....	<b>16</b>
<b>6.1</b>	<b>ADDIREG installation</b> .....	<b>16</b>
<b>6.2</b>	<b>Program description</b> .....	<b>17</b>
6.2.1	"Board list configuration" .....	17
6.2.2	"Board configuration" .....	18
6.2.3	Buttons .....	19
<b>6.3</b>	<b>MORE information</b> .....	<b>20</b>
6.3.1	PCI analog input boards with DMA .....	20
	System information .....	21
	PCI DMA board list .....	21
	Buttons .....	22
	Single PCI DMA board configuration: .....	22
<b>6.4</b>	<b>Registering a new board</b> .....	<b>23</b>
<b>6.5</b>	<b>Changing the registration of an existing board</b> .....	<b>24</b>

<b>7</b>	<b>SOFTWARE .....</b>	<b>25</b>
<b>7.1</b>	<b>Driver installation .....</b>	<b>25</b>
7.1.1	Installation under DOS and Windows 3.11 .....	25
7.1.2	Installation under Windows NT .....	25
7.1.3	Installation under Windows 2000/9x .....	26
<b>7.2</b>	<b>Installation of the software samples .....</b>	<b>27</b>
7.2.1	Installation under DOS .....	27
7.2.2	Installation under Windows NT/9x/2000 .....	27
<b>7.3</b>	<b>The ADDI-UNINSTALL program .....</b>	<b>28</b>
7.3.1	Installation of ADDI-UNINSTALL .....	28
7.3.2	Software uninstalling with ADDI-UNINSTALL .....	28
	Uninstall ADDIREG .....	29
<b>7.4</b>	<b>Software downloads from the Internet .....</b>	<b>29</b>
<b>8</b>	<b>CONNECTION TO THE PERIPHERAL .....</b>	<b>30</b>
<b>8.1</b>	<b>Connection principle .....</b>	<b>30</b>
<b>8.2</b>	<b>Connector pin assignment .....</b>	<b>31</b>
<b>8.3</b>	<b>Connection examples .....</b>	<b>32</b>
8.3.1	Analog input channels .....	32
8.3.2	Digital input and output channels .....	33
8.3.3	Connection to screw terminal boards .....	33
<b>9</b>	<b>FUNCTIONS OF THE BOARD .....</b>	<b>34</b>
<b>9.1</b>	<b>Analog input channels .....</b>	<b>34</b>
<b>9.2</b>	<b>Time-multiplex system .....</b>	<b>35</b>
<b>10</b>	<b>SOFTWARE EXAMPLES .....</b>	<b>37</b>
<b>10.1</b>	<b>Initialisation .....</b>	<b>37</b>
10.1.1	Initialisation of an APCI-/CPCI-3001 board .....	37
	a) Flow chart .....	37
	b) Example in C .....	38
10.1.2	Initialisation of several APCI-/CPCI-3001 boards .....	39
	a) Flow chart .....	39
	b) Example in C .....	40
<b>10.2</b>	<b>Interrupt .....</b>	<b>41</b>
	a) Flow chart .....	41
	b) Example in C for DOS und Windows 3.1x .....	42
	c) Example in C for Windows NT/95/98 (asynchronous mode) .....	43
	d) Flow chart for Windows NT/95/98 (synchronous mode) .....	44
	e) Example in C for Windows NT/95/98 (synchronous mode) .....	45

<b>10.3 Direct conversion of analog input channels .....</b>	<b>46</b>
10.3.1 Test of 1 analog input channel .....	46
a) Flow chart .....	46
b) Example in C .....	47
10.3.2 Test of several analog input channels .....	48
a) Flow chart .....	48
b) Example in C .....	49
<b>10.4 Cyclic conversion of analog input channels.....</b>	<b>50</b>
10.4.1 Cyclic conversion without DMA, external trigger and delay.....	50
a) Flow chart.....	50
b) Pin assignment .....	51
c) Example in C for DOS .....	51
d) Example in C for Windows 3.1x.....	52
e) Example in C for Windows NT/95/98 (asynchronous mode).....	53
f) Example in C for Windows NT/95/98 (synchronous mode) .....	54
10.4.2 Cyclic conversion with DMA without external trigger and delay.....	55
a) Flow chart.....	55
b) Pin assignment .....	56
c) Example in C for DOS .....	56
d) Example in C for Windows 3.1x.....	57
e) Example in C for Windows NT/95/98 (asynchronous mode).....	58
f) Example in C for Windows NT/95/98 (synchronous mode) .....	59
<b>10.5 Timer.....</b>	<b>60</b>
10.5.1 Test of the timer interrupt.....	60
a) Flow chart .....	60
b) Example in C for DOS .....	61
c) Example in c for Windows 3.1x .....	62
d) Example in C for Windows NT/95/98 (asynchronous mode).....	63
e) Example in C for Windows NT/95/98 (synchronous mode) .....	64
<b>10.6 Digital Input channels .....</b>	<b>65</b>
10.6.1 Reading 1 digital input channel .....	65
a) Flow chart.....	65
b) Pin assignment .....	66
c) Example in C .....	66
10.6.2 Reading 4 digital input channels.....	67
a) Flow chart .....	67
b) Pin assignment .....	68
c) Example in C .....	68
<b>10.7 Digital output channels.....</b>	<b>69</b>
10.7.1 Test of the digital output memory .....	69
a) Flow chart .....	69
b) Example in C .....	70
<b>INDEX.....</b>	<b>A</b>

**Figures**

Fig. 3-1: Correct handling for the CPCI-3001 ..... 3

Fig. 3-2: Correct handling for the APCI-3001 ..... 3

Fig. 4-1: Component scheme of the APCI-3001 board ..... 8

Fig. 4-2: Component scheme of the CPCI-3001 board..... 9

Fig. 5-1: Opening the protective blister pack. .... 10

Fig. 5-2: Jumper location on the APCI-3001 ..... 11

Fig. 5-3: Jumper location on the CPCI-3001 ..... 11

Fig. 5-4: Settings at delivery ..... 12

Fig. 5-5: Types of slots ..... 13

Fig. 5-6: Inserting the board..... 13

Fig. 5-7: Fastening the board at the back cover ..... 14

Fig. 5-8: Types of slots for CompactPCI boards ..... 14

Fig. 5-9: Pushing a CPCI board into a rack ..... 15

Fig. 5-10: Connector keying ..... 15

Fig. 6-1: Installation of the ADDIREG program..... 16

Fig. 6-2: ADDIREG registration program ..... 17

Fig. 6-3: Board list under ADDIREG ..... 19

Fig. 6-4: PCI DMA management ..... 21

Fig. 7-1: Installation of the driver ..... 25

Fig. 7-2: Inquiry of the .inf files (German version)..... 26

Fig. 7-3: Installation of the samples ..... 27

Fig. 7-4: Installation of the ADDI-UNINSTALL program..... 28

Fig. 7-5: The ADDI\_UNINSTALL program ..... 28

Fig. 8-1: Current loop circuitry for the option DC..... 30

Fig. 8-2: 37-pin SU-D male connector ..... 31

Fig. 8-3: 16-pin male connector connected to a 37-pin SUB-D male connector ..... 31

Fig. 8-4: Analog input channels (SE)..... 32

Fig. 8-5: Analog input channels (diff.) ..... 32

Fig. 8-6: Digital output channels ..... 33

Fig. 8-7: Digital input channels..... 33

Fig. 8-8: Connection to the PX 901 screw terminal board ..... 33

**Tables**

Table 5-1: Setting the jumpers according to the function used..... 12

# 1 INTENDED PURPOSE OF THE BOARD

The **APCI-/CPCI-3001** board is the interface between an industrial process and a personal computer (PC). It is to be used in a PC equipped with a free PCI 5 V/32-bit PCI slot. The PC is to comply with the EU directive 89/336/EEC and the specifications for EMC protection.

Products complying with these specifications bear the normed  mark.

Data exchange between the **APCI-/CPCI-3001** board and the peripheral is to occur through a shielded cable. This cable must be connected to the 37-pin SUB-D male connector of the **APCI-/CPCI-3001** board.

The board has up to 16 input channels for processing analog signals and 4 input channels and 4 output channels for processing digital 24 V signals.

The use of the **APCI-/CPCI-3001** board in combination with external screw terminal boards is to occur in a closed switch cabinet; the installation is to be effected competently.

The connection with our standard cable ST010 complies with the minimum specifications as follows:

- 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 the *Technical description* and in the *Safety* leaflet.

## 1.1 Limits of use

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 not being conform anymore.

**Check the shielding capacity** of the PC housing and of the cable prior to putting the device into operation.

Make sure that the board remains in its protective blister pack **until it is used**. Do not remove or alter the identification numbers of the board. If you do, the guarantee expires.

## **2 USER**

### **2.1 Qualification**

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

- installation,
- use,
- maintenance.

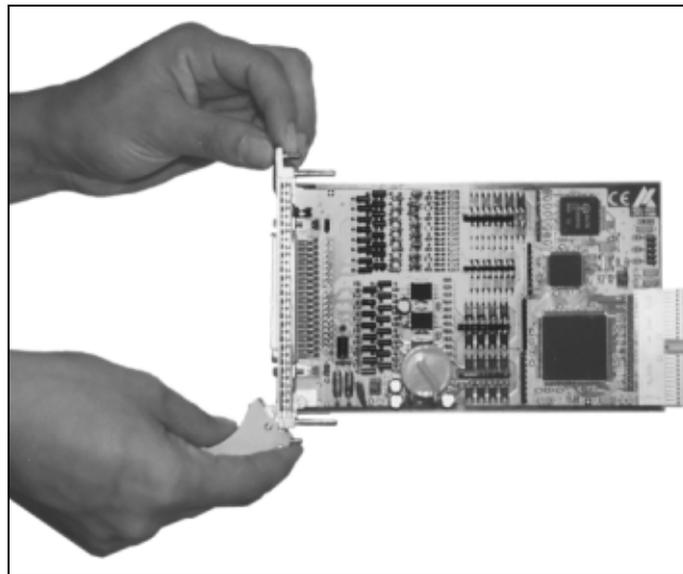
### **2.2 Personal protection**

Consider the country-specific regulations about

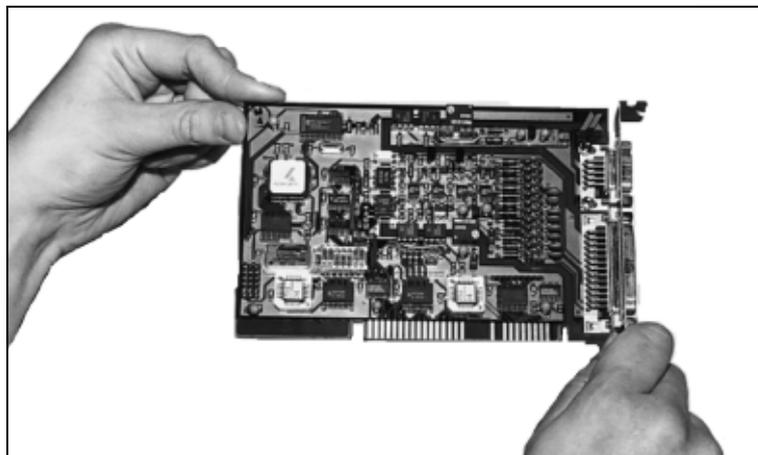
- the prevention of accidents
- electrical and mechanical installations
- radio interference suppression.

### 3 HANDLING THE BOARD

**Fig. 3-1: Correct handling for the CPCI-3001**



**Fig. 3-2: Correct handling for the APCI-3001**



## 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 EN50082-2, EN55011, EN55022 as follows:

	<u>True value</u>	<u>Set value</u>
ESD .....	4 kV	4 kV
Fields .....	10 V/m	10 V/m
Burst .....	2 kV	2 kV
Conducted radio interferences .....	10 V	10 V
Noise emissions .....		B-class



**WARNING!**

The EMC tests have been carried out in a specific appliance configuration.

We guarantee these limit values **only** in this configuration<sup>1</sup>.

Consider the following aspects:

- your test program must be able to detect operation errors.
- your system must be set up so that you can find out what caused errors.

### 4.2 Physical set-up of the board

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

	<b>APCI-3001</b>	<b>CPCI-3001</b>
Dimensions	<p>175 mm 99 mm</p>	<p>160 mm 100 mm</p>
Weight	156 g	200 g
Insertion in	PCI-5V (32-bit) or PCI-5V (64-bit) slot	CompactPCI-5V(32-bit) or CompactPCI-5V (64-bit) slot
Connection to the peripheral	37-pin SUB-D male connector	37-pin SUB-D male connector

<sup>1</sup> We transmit our appliance configuration on request

### 4.3 Options

- SF, DF: Precision filter for the analog input channels
- SC Precision current inputs 0-20 mA or 4-20 mA for the SE analog input channels
- DC Precision current inputs 0-20 mA or 4-20 mA for the diff. analog input channels (Expect for the board XPCI-3001-4)

Input range	At a resolution of 12-bit
0 mA	is 0 digit
4 mA	is 819 digits
20 mA	is 4095 digits

**Remark:** At 4-20 mA the accuracy is altered.

### 4.4 Versions

The board APCI-/CPCI-3001 is available in the following versions.

Version	Analog input channels
XPCI-3001-4 <sup>1</sup>	4 SE <sup>2</sup>
XPCI-3001-8	8 SE / 4 Diff. <sup>3</sup>
XPCI-3001-16	16 SE / 8 Diff.

### 4.5 Limit values

- Operating temperature: ..... 0 to 60°C
- Storage temperature: ..... -25 to 70°C
- Relative humidity: ..... 30% to 99% non condensing

**Minimum PC requirements (APCI-3001):**

- PCI BIOS, bus speed ≤ 33 MHz
- Operating system: ..... MS DOS 6.22 or higher  
Windows 3.1, NT 4.0, 95, 98

**Minimum system requirements (CPCI-3001):**

- 32-Bit CompactPCI bus (5 Volt), bus speed ≤ 33 MHz
- PCI BIOS, PCI 2.1 specification and CompactPCI 2.1 "compliant"
- 3 U format according to IEEE-1101
- Operating system: ..... MS DOS 6.22 or higher  
Windows 3.1, NT 4.0, 95, 98

---

<sup>1</sup> Common mane for the APCI-3001 and the CPCI-3001

<sup>2</sup> SE for Single-Ended

<sup>3</sup> diff. for differential

**Energy requirements:**

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

	<b>XPCI3001-x</b>
+ 5 V from the PC	886 mA

**Analog input channels:**

- Number of analog input channels: ..... 16 SE/ 8 diff. for **XPCI-3001-16**  
8 SE / 4 diff. for **XPCI3001-8**
- Analog resolution: ..... 12-bit, 1 among 4095
- Max. sampling rate (1 input channel): ..... 100 kHz
- Data transfer : ..... Data to the PC (16-bit only)  
via FIFO memory  
1) through I/O commands  
2) Interrupt at EOC <sup>1</sup> & EOS <sup>2</sup>  
3) DMA transfer at EOC
- Start of conversion: ..... 1) per software trigger  
2) TIMER 0  
3) TIMER 0 & 1  
4) external trigger
- Monotony: ..... 12-bit
- Offset error: ..... after calibration:  
- Bipolar: ± ½ LSB  
Unipolar: ± ½ LSB  
Drift (0°C to 60°C):  
- Bipolar: ± 2 ppm / °C  
Unipolar: ± 2 ppm / °C
- Gain error: ..... after calibration:  
- Bipolar: ± ½ LSB  
Unipolar: ± ½ LSB  
Drift (0°C to 60°C):  
- Bipolar: ±7 ppm / °C  
Unipolar: ±7 ppm / °C
- Analog input ranges: ..... Voltage  
Unipolar: 0-10 V  
Bipolar: ±10 V  
Selectable by software
- Analog input ranges: ..... Current  
Unipolar: 0-20 mA  
Selection of the range 0-10 V  
and of gain x2 is necessary
- Overvoltage protection: ..... 20 V when POWER ON

1 EOC: End of Conversion

2 EOS:(= End of Scan ): signals that the acquisition of a group of channels has been completed

**Analog input channels (continued):**

Common mode rejection: ..... DC up to 10 Hz, 90 dB mini.  
(Gain = 1)

Band width (-3dB): ..... Limited to 159 kHz (-3dB)  
with low-pass filter 1st order;  
yet the minimum SINAD is still  
83 dB at 49 kHz (fin)

Bias currents for each input channel  
(multiplexer) ..... ± 2 nA max.

Input impedance (PGA): ..... 10<sup>12</sup> Ω // 20 nF to GND

Integral non-linearity (INL): ..... ± 0.9 LSB

Differential non-linearity (DNL): ..... ± 0.9 LSB

Accuracy: ..... ± ½ LSB

Selectable gain: ..... via PGA gain 1, 2, 5, 10  
(selectable by software)

System noise: ..... Bipolar: Gain x1: ± ½ LSB  
Gain x2: ± ½ LSB  
Gain x10: ± 1 LSB  
Unipolar: Gain x1: ± ½ LSB  
Gain x2: ± ½ LSB  
Gain x10: ± 1 LSB

Digital coding: ..... linear

Analog Input		Binary Code	HEX Code
Bipolar	Unipolar		
- 10V	0V	0000000000000000	0000
0V	5V	1000000000000000	8000
+10V	10V	1111111111111111	FFFF

Optical isolation to the PC: ..... 500 VDC min.

Overvoltage protection: ..... ± 12 V

Data transfer: ..... The board is located in the I/O  
address space of the PC.  
The values are written on the  
board through 16-bit accesses  
and automatically updated.

**Digital input channels:**

Number: ..... 4

Input current at 24 V: ..... 3 mA typ.

Input voltage range: ..... 0-30 V

Optical isolation: ..... 1000 VAC

Logic „0“ level: ..... 0-5 V

Logic „1“ level ..... 10-30 V

**Digital output channels:**

Number: ..... 4

Max. switch current: ..... 5 mA typ.

Voltage range: ..... 5-30 V

Optical isolation: ..... 1000 VAC

Type: ..... Open Collector

## 4.6 Component scheme

Fig. 4-1: Component scheme of the APCI-3001 board

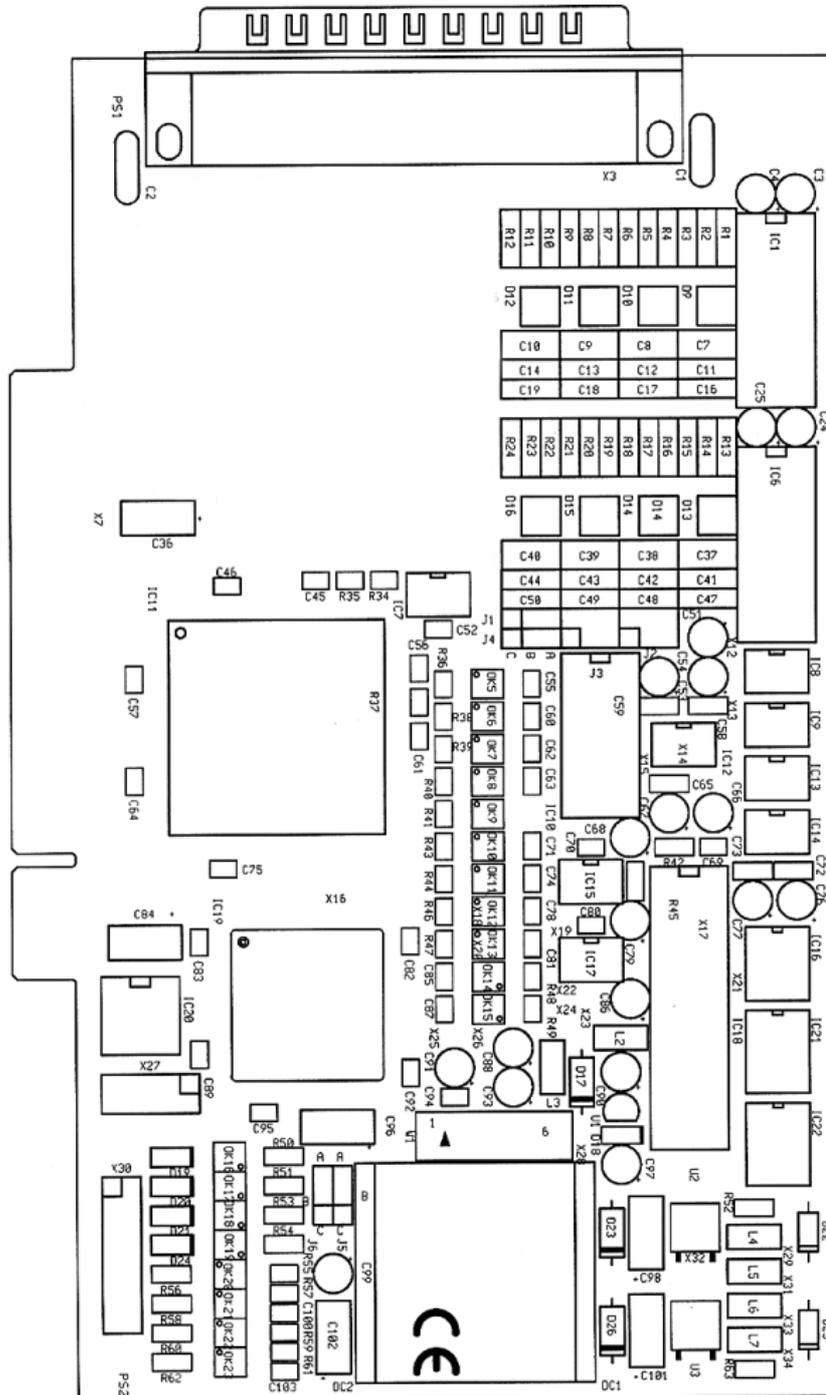
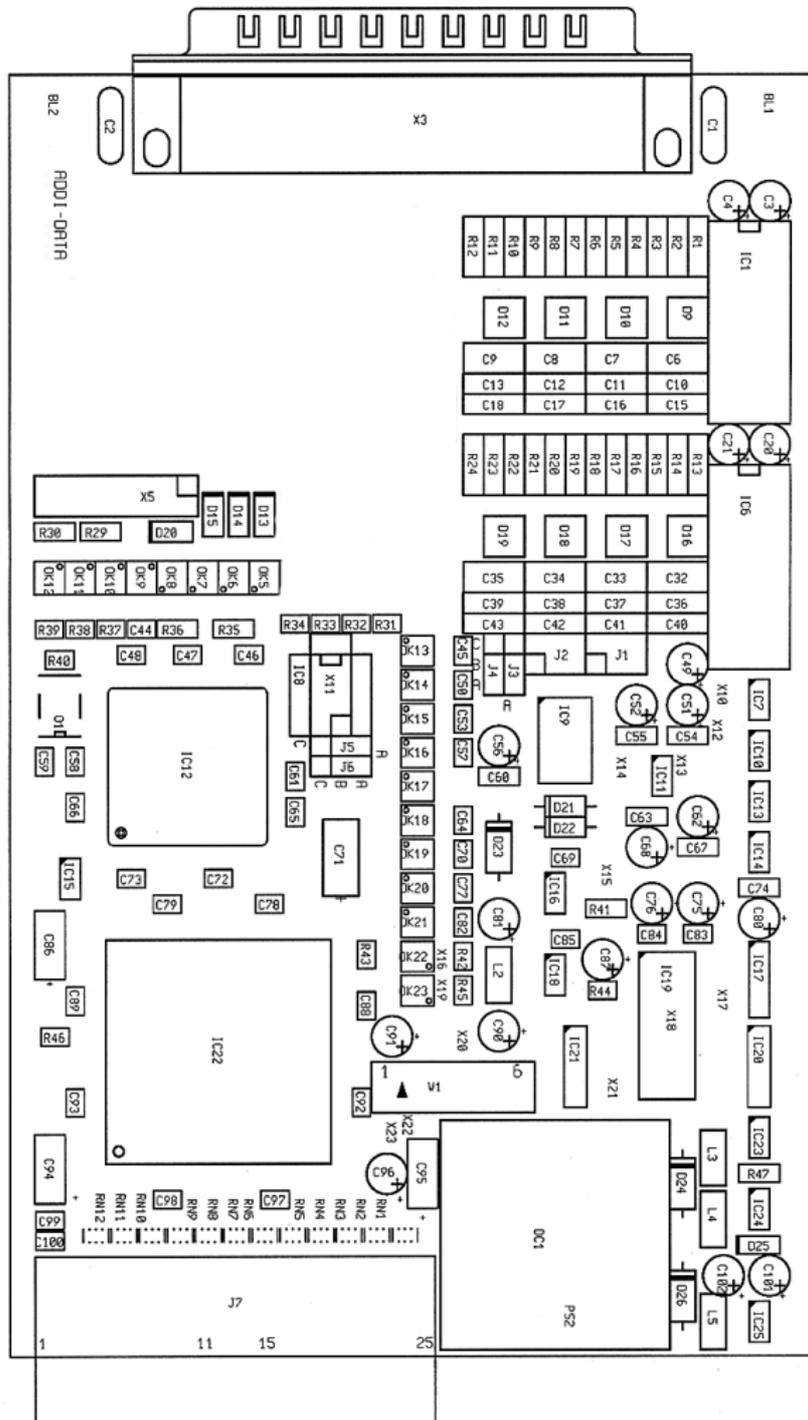


Fig. 4-2: Component scheme of the CPCI-3001 board



## 5 INSTALLATION

The board is supplied with a CD-ROM containing:

- ADDIREG for Windows NT 4.0 and Windows 95/98/2000.
- standard software for the ADDI-DATA boards:
  - 16-bit for MS-DOS and Windows 3.11
  - 32-bit for Windows NT/95/98/2000.

You can download the latest version of the ADDIREG program from the Internet.

### i

#### IMPORTANT!

If you want to install **several** ADDI-DATA boards **simultaneously**, consider the following procedure.

- **Install and configure** the boards one after the other.  
You will thus avoid configuration errors.
1. Switch off the PC
  2. Install the **first** board
  3. Start the PC
  4. Install ADDIREG (once is enough)
  5. Configure the board
  6. Install the driver and the samples if necessary
  7. Switch off the PC
  8. Install the **second** board
  9. Start the PC
  10. Configure the board
  11. Install the driver and the samples if necessary.  
etc.

### i

#### IMPORTANT!

**To install the new version of ADDIREG**, please uninstall first the current version from your PC with the **ADDI-UNINSTALL** program.

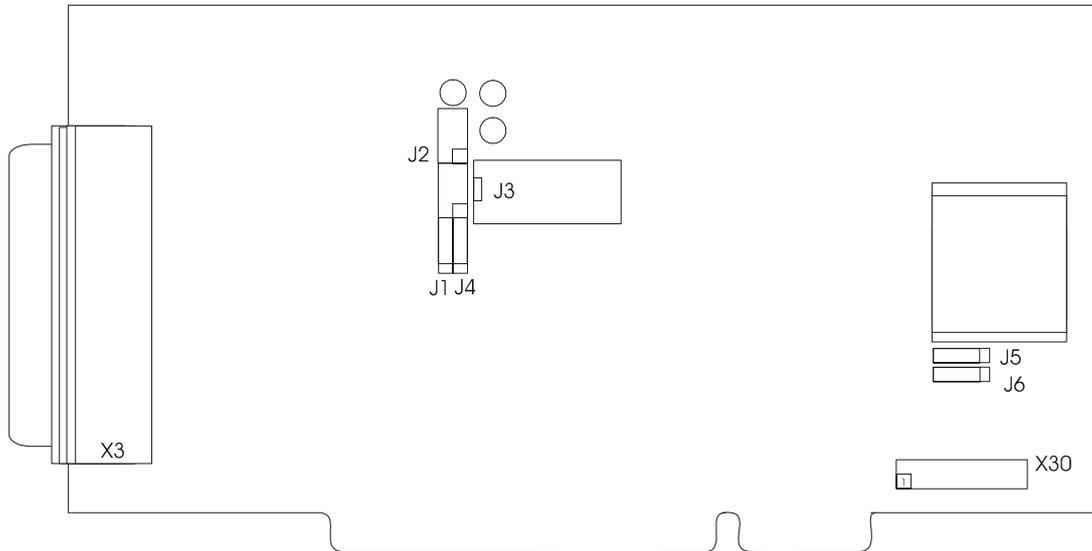
**Fig. 5-1: Opening the protective blister pack.**



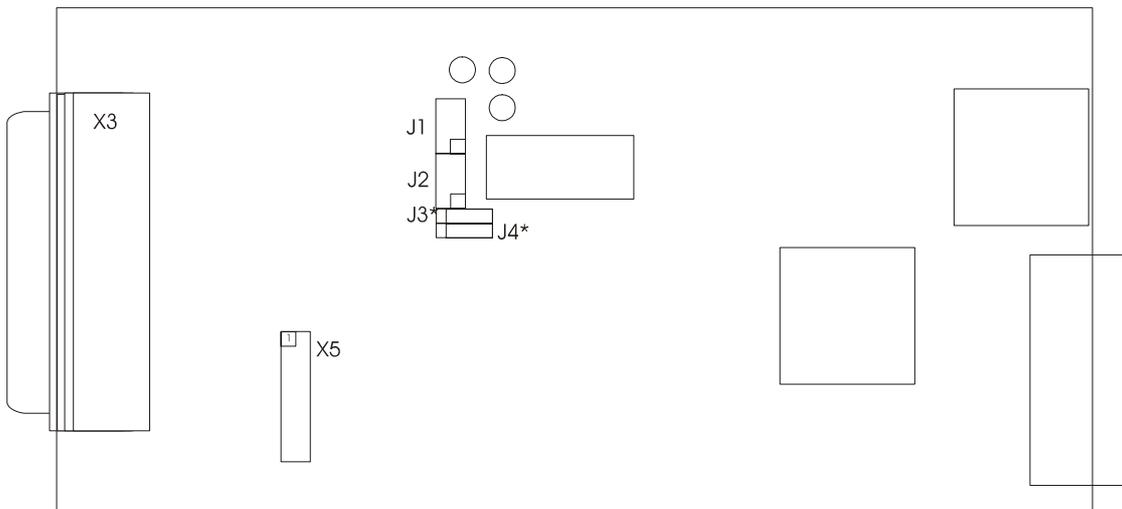
## 5.1 Jumper settings

### 5.1.1 Jumper location and settings at delivery

**Fig. 5-2: Jumper location on the APCI-3001**

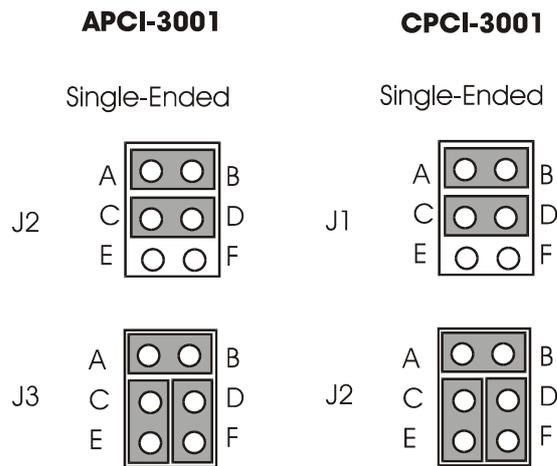


**Fig. 5-3: Jumper location on the CPCI-3001**



\* J3 and J4: do not consider

**Fig. 5-4: Settings at delivery**



**5.1.2 Jumper settings according to the function used**

**Table 5-1: Setting the jumpers according to the function used**

APCI-3001	CPCI-3001	Position	Function	Delivery settings
J2	J1	A-B, C-D	Single Ended measurement	✓
J2	J1	A-C, D-F	Differential measurement	
J3	J2	A-B, C-E, D-F	Single Ended measurement	✓
J3	J2	B-D, C-E	Differential measurement	

## 5.2 Inserting the board

**i**

**IMPORTANT!**

Do observe the *safety instructions*.

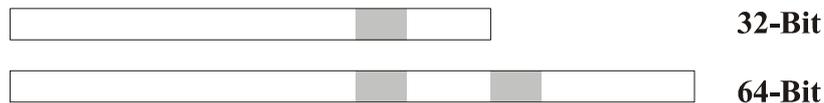
- Switch off your PC and all the units connected to the PC.
- Pull the PC mains plug from the socket.
- Open your PC as described in the manual of the PC manufacturer.

### 5.2.1 Installing an APCI-3001 board

#### Selecting a free slot

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

**Fig. 5-5: Types of slots**



Information on the slot types can be found in the PC manual.

**Decide in which type of slot to insert the board.**

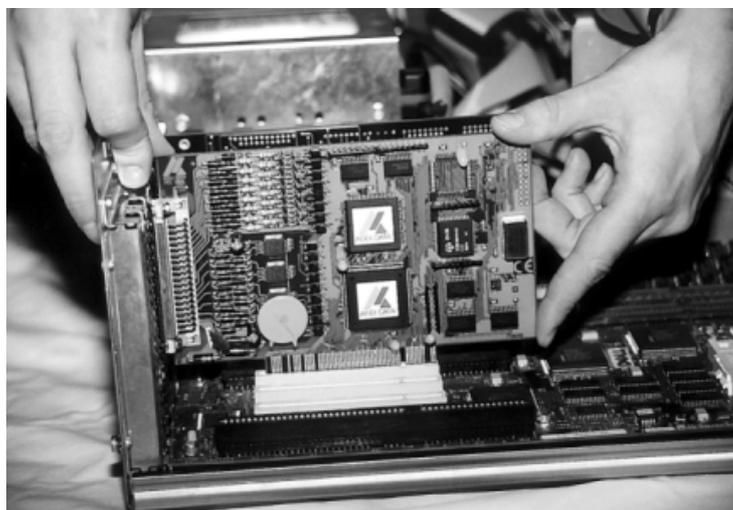
**Remove the back cover of the selected slot according to the instructions of the PC manufacturer.** Keep the back cover. You will need it if you remove the board.

Discharge yourself from electrostatic charges.

#### Plugging the board

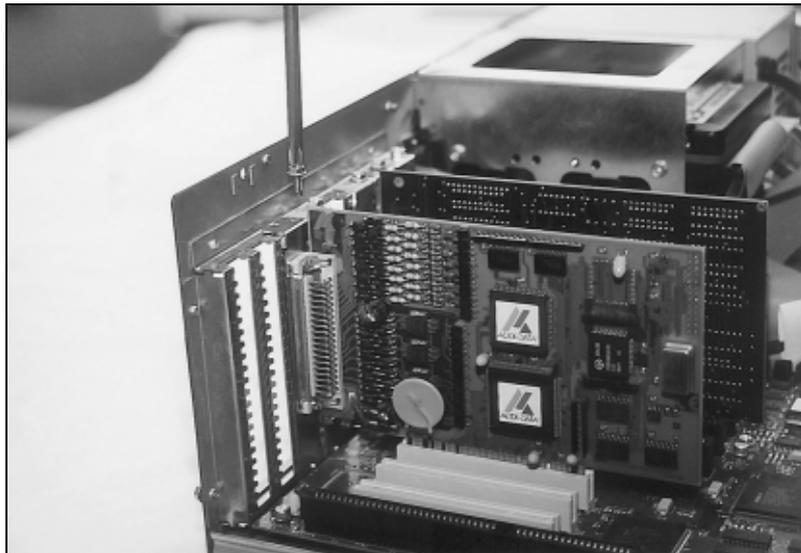
Insert the board vertically into the chosen slot.

**Fig. 5-6: Inserting the board**



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

**Fig. 5-7: Fastening the board at the back cover**



Tighten all the loosen screws.

**Closing the PC**

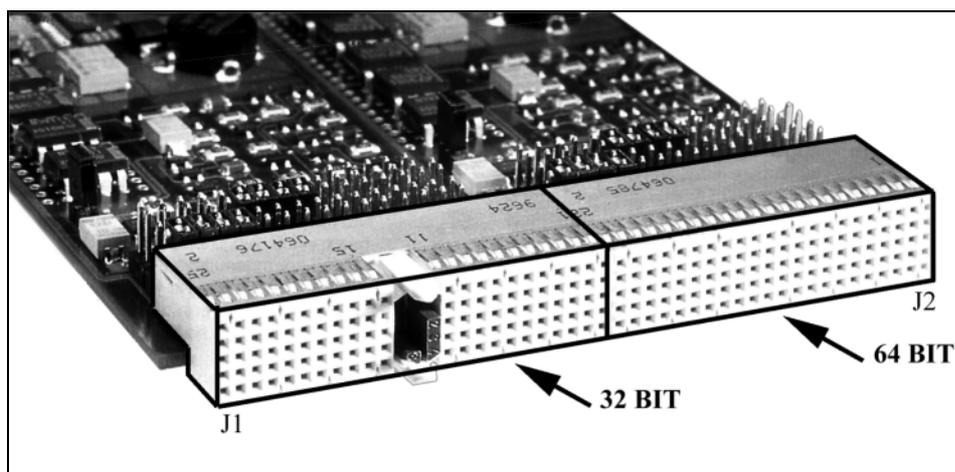
Close your PC as described in the manual of the PC manufacturer.

**5.2.2 Installing a CPCI-3001 board**

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

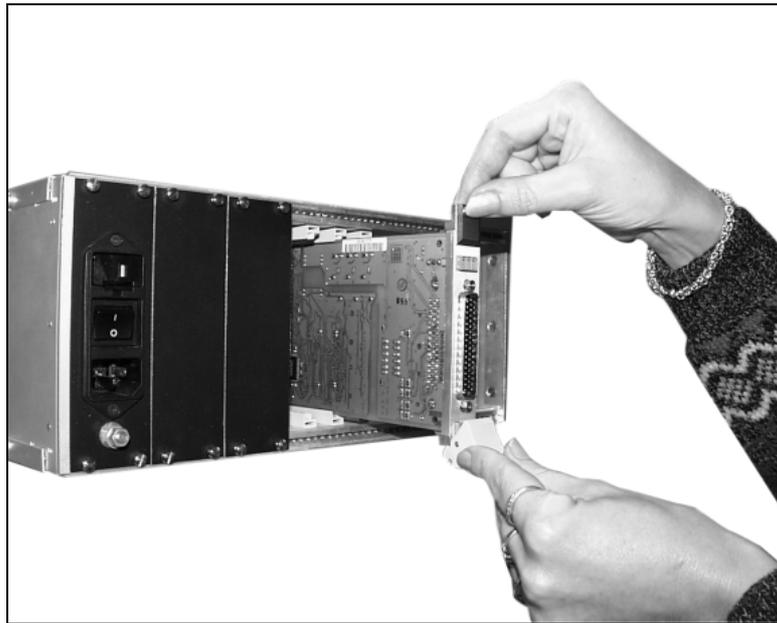
See in the computer manual which types of slots are free.

**Fig. 5-8: Types of slots for CompactPCI boards**



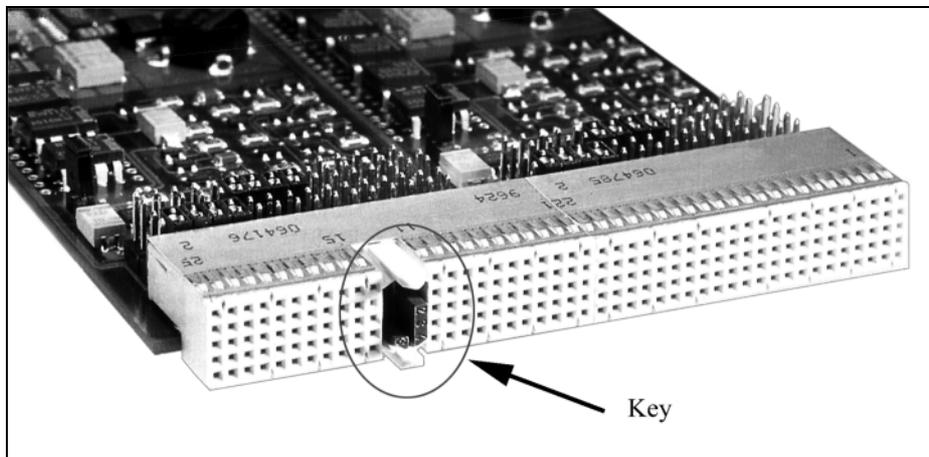
- Discharge yourself from electrostatic charges
- Hold the board at its grip (See handling of the board in chapter 3).
- Insert the board into the guiding rails and push it to the back cover of the rack.  
**In order to fully insert the board, a small resistance has to be overcome.**

**Fig. 5-9: Pushing a CPCI board into a rack**



- Make sure that the board is correctly connected by connecting the key of the board to the key of the backplane. (blue connector key if the board operates in 5 V).

**Fig. 5-10: Connector keying**



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

**Note:**

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

## 6 BOARD CONFIGURATION WITH ADDIREG

The ADDIREG registration program is a 32-bit program for Windows NT 4.0/200/9x.

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

### 6.1 ADDIREG installation

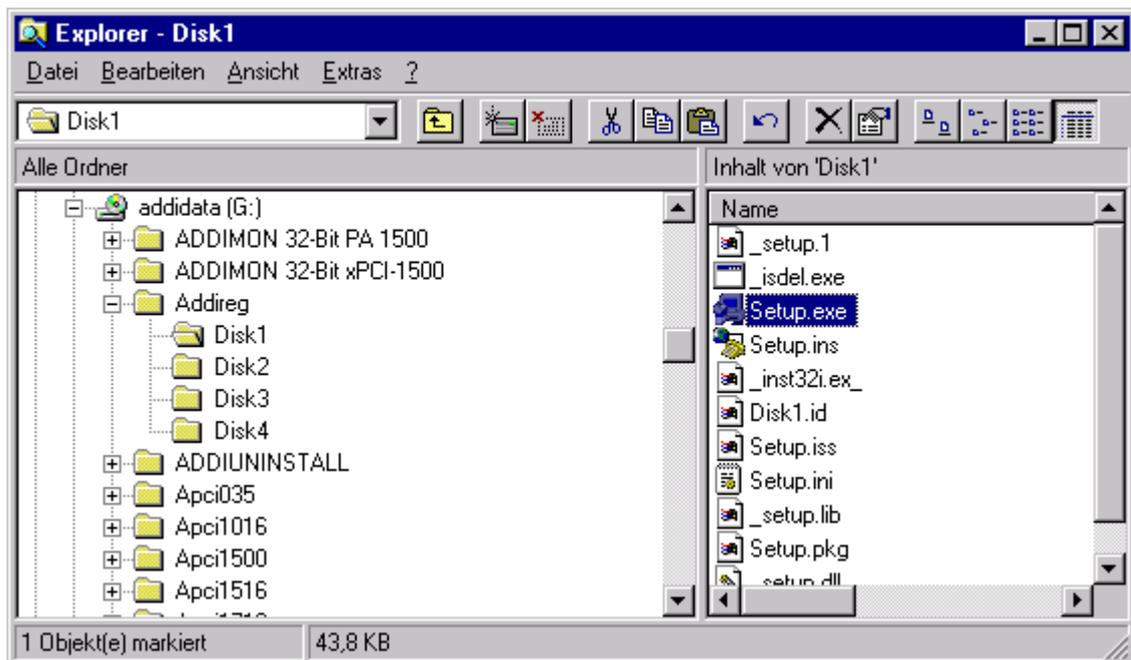
**i**

#### IMPORTANT!

Do install the ADDIREG Programm first before installing or starting any other application for the board.

- Change to the CD drive.

**Fig. 6-1: Installation of the ADDIREG program**



- Start the set-up program "setup.exe" (double click)
- Select one of the 3 parameters
  - 1- typical
  - 2- compact
  - 3- custom

Proceed as indicated on the screen and read attentively the "Software License" and "Readme". In "custom", you can select your operating system.

The installation program gives you further instructions.

If the message "Der Keyboard Kernel wurde noch nicht gestartet, ... soll der Kernel jetzt gestartet werden?" (Problem when installing the system) is displayed by starting the program, deinstall the ADDIREG program and install it anew.

## 6.2 Program description

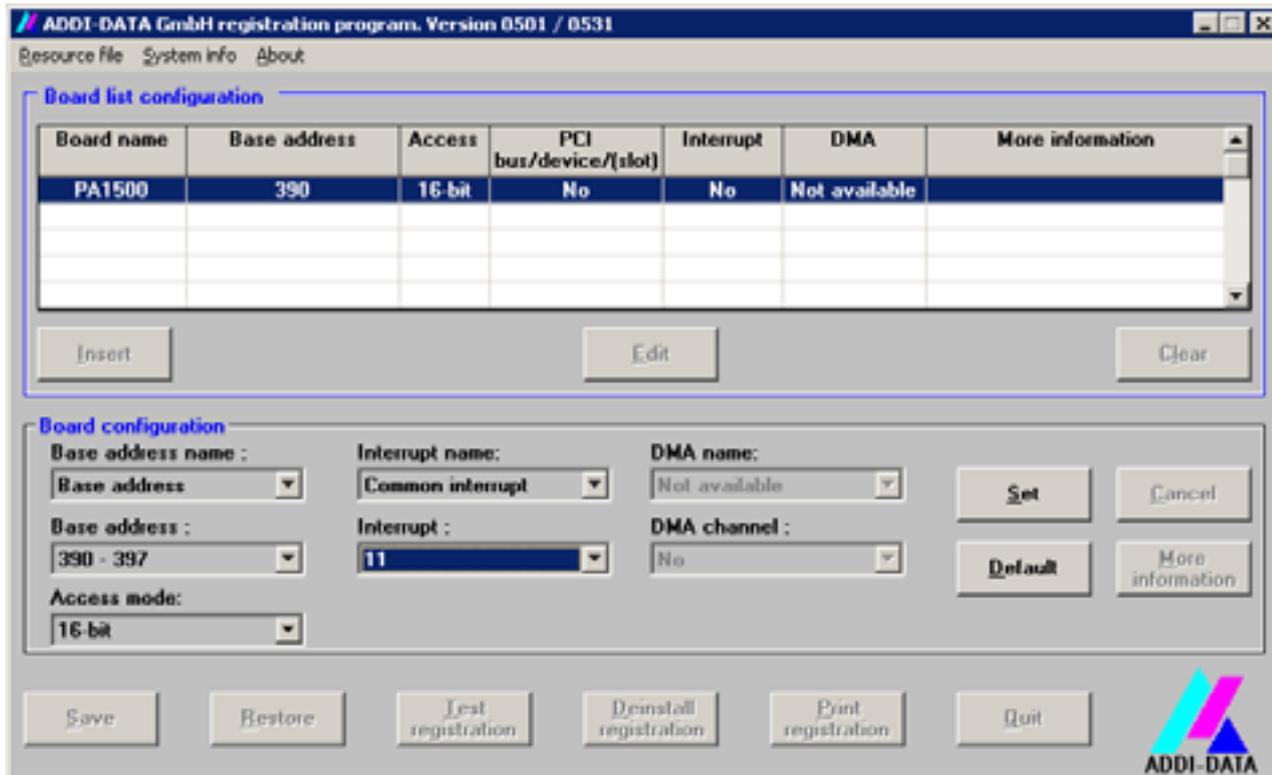
**i**

### IMPORTANT!

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

If the board is not inserted, the user cannot test the registration. Once the program is called up, the following dialog box appears.

**Fig. 6-2: ADDIREG registration program**



### 6.2.1 "Board list configuration"

The table in the middle contains the registered boards and their parameters.

**Board name:**

In this field the name of the registered boards will be displayed (e.g. APCI-1500). If you use the program for the first time, no board is displayed on the screen.

**Base address:**

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

**Access:**

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

**PCI bus/device/(slot):**

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

**Interrupt:**

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

**DMA (ISA boards only):**

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

**More information:**

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

## 6.2.2 "Board configuration"

Under the table are six text boxes. With this text boxes you can change the parameters of the boards.

**Base address name:**

When the board operates with several base addresses (One for port 1, one for port 2, etc.) you can select which base address is to be changed.

**Base address:**

In this box you can select the base addresses of your PC board. The free base addresses are listed. The used base addresses do not appear in this box. (boards without ADDIDRIVER)

**Interrupt name:**

When the board must support different interrupt lines (common or single interrupts), you can select them in this box.

**Interrupt (ISA boards only):**

Selection of the interrupt number which the board uses.

**DMA name (ISA boards only):**

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

**DMA channel (ISA boards only):**

Selection of the used the DMA channel.

### 6.2.3 Buttons

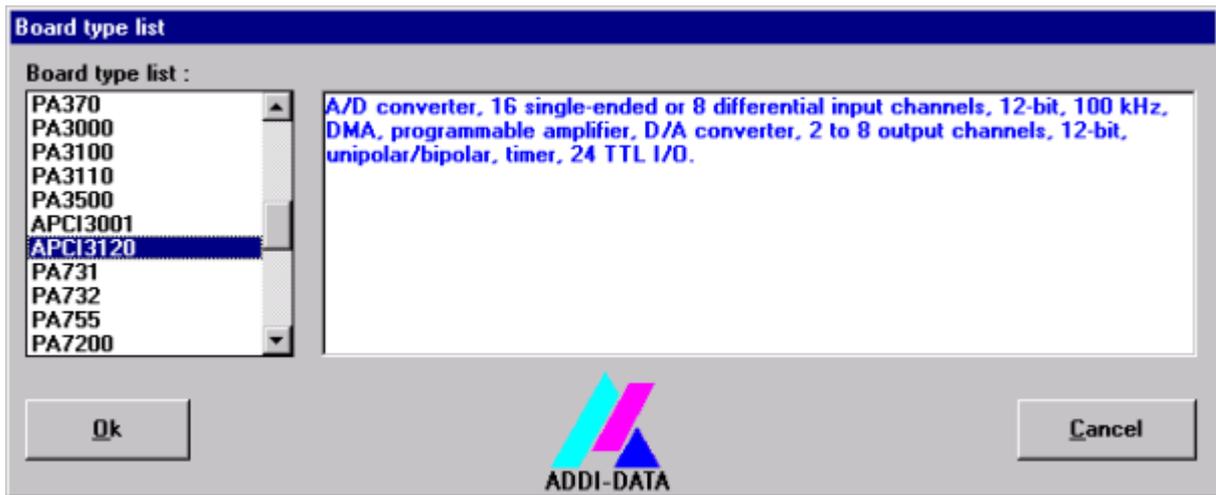
**Edit:**

Selection of the highlighted board with the different parameters set in the text boxes. A double click on the wished board has the same function.

**Insert:**

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

**Fig. 6-3: Board list under ADDIREG**



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

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

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

**Clear:**

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

**Set:**

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

**Cancel:**

Reactivates the parameters of the former configuration.

**Default:**

Sets the standard parameters of the board.

**Save:**

Saves the parameters and registers the board.

**Restore:**

Reactivates the previous parameters and registration.

**Test registration:**

Checks if there is a conflict between the board and another device.

A message prints either "OK" or the parameter which have generated the conflict.

**Deinstall registration:**

Deinstalls all registrations of all boards in the table.

**Print registration:**

Prints the registration parameters on your standard printer.

**Quit:**

Ends the ADDIREG program.

**More information**

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

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

**i**

**IMPORTANT!**

**According to the board type** the user has different possibilities (see next paragraph).

## 6.3 MORE information

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

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

### 6.3.1 PCI analog input boards with DMA

If you have inserted an APCI-3001 or CPCI-3001 the following dialog box is displayed when clicking on "More information".

Below is the example of 1,000,000 PCI DMA acquisitions (in continuous mode).

For the PCI DMA analog input acquisition, a linear memory buffer of the PC is used. The buffer size depends on the number of acquisitions. For 1 acquisition 2 bytes are needed.

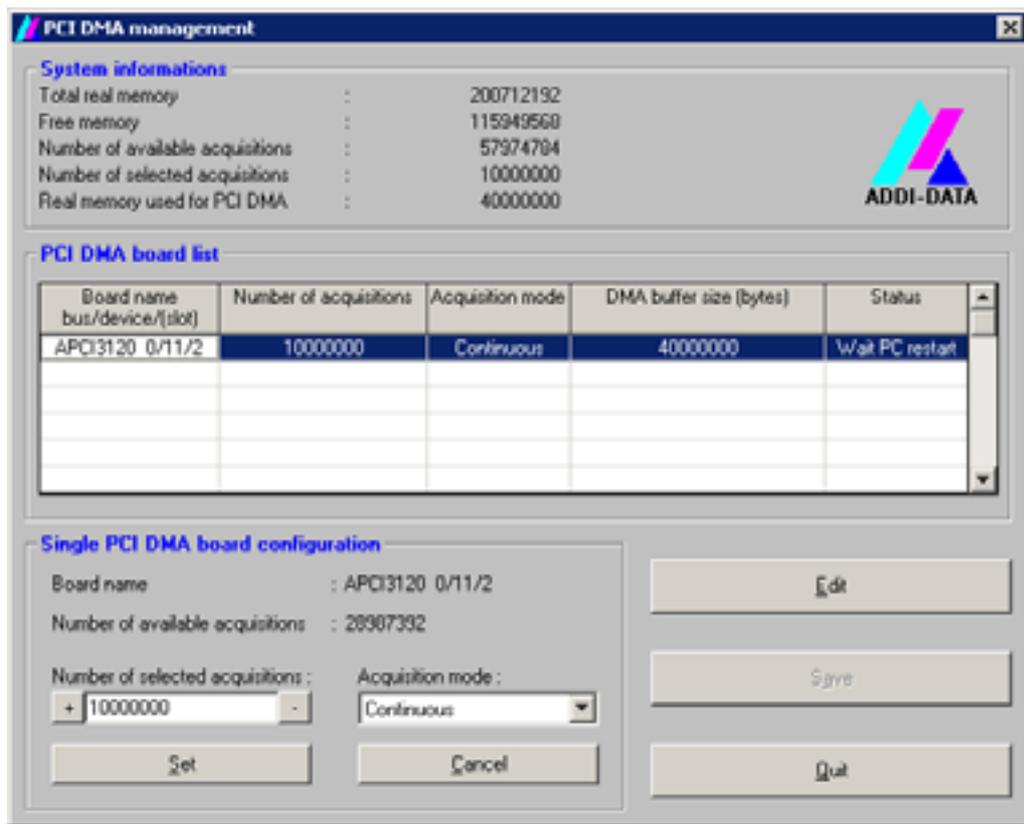
You can define the maximum number of acquisitions used for your application and allocate a large buffer after the PC has started.

If you have selected DMA\_USED in the function `i_PCI3001_InitAnalogInputAcquisition` the buffer(s) are used. (See technical description "Standard software")

For a single acquisition, only one buffer is allocated.

For a continuous acquisition, two buffers are allocated.

Fig. 6-4: PCI DMA management



**System information**

**Total real memory:**

Total real memory of the PC (in bytes).

**Free memory:**

Returns the PC memory (in bytes) available for PCI DMA acquisition.

**Number of available acquisitions:**

Returns the number of acquisitions which can be carried out in the single mode.

**Number of selected acquisitions:**

Returns the number of acquisitions selected by the user.

**Real memory used for PCI DMA:**

Returns the memory size (in bytes) used for the PCI DMA acquisition.

**PCI DMA board list**

List of all PCI boards which can use the PCI DMA analog input acquisition. For each board the user can select the number of acquisitions and the acquisition mode (single/continuous).

**Board name bus/device/(slot):**

Indicates the board name, the bus number, the device number and the slot number.

**Number of acquisitions:**

Number of acquisitions selected by the user.

**Acquisition mode:**

Acquisition mode selected by the user (single or continuous).

**DMA buffer size (in bytes):**

Size of the buffer used for this configuration.

**Status:**

Not used: The number of acquisitions selected by the user is equal to 0

Wait PC restart: Wait until the PC restarts to allocate the memory

Allocation OK: Buffer allocation OK

Allocation error: Buffer allocation error. The driver could not allocate a linear memory buffer for this acquisition.

**Buttons****Edit:**

Selection of the highlighted board with the different parameters set in the boxes of "Single PCI DMA board configuration". (See below)

**Save:**

Saves the configuration of all boards.

**Quit:**

Closes this window.

**Single PCI DMA board configuration:**

After selecting a board, click on Edit: the selected configuration of the board with PCI DMA is displayed in the "Single PCI DMA board configuration" box.

**Board name:**

Indicates the board name, the bus number, the device number and the slot number

**Number of available acquisitions:**

Indicates the number of acquisitions available **for the selected mode** (acquisition mode) and **for the next board** to be configured.

**Number of selected acquisitions:**

Number of acquisitions selected by the user ("Not used" means that no buffer is allocated for PCI DMA acquisition).

**i****IMPORTANT!**

You have to enter an **even number**.

An odd number of acquisitions will not be accepted and automatically replaced by the approaching even number.

**Acquisition mode:**

Acquisition mode selected the user:

Single: Only one acquisition cycle is used. After this cycle the acquisition is immediately stopped.

Continuous: The acquisition runs until the function `i_PCI3001_StopAnalogInputAcquisition` is called up.

**Set:**

Sets the user configuration.

**Cancel:**

Restores the former configuration

## 6.4 Registering a new board

### **i**

**IMPORTANT!**

To register a new board you need **Administrator rights**. If you have user rights you cannot register a new board or change an existing registration!!

- Call the ADDIREG program. The screen of figure 6-2 will be displayed. Click on "Insert". Select the board you need.
- Select the board to be registered and click on "Ok". The default address, interrupt and the other parameters are automatically selected. The parameters are set in the fields under the table. If the parameters are not automatically set by the BIOS (by the PCI boards for example), you can change the parameters. For this please use the scroll functions of the fields. Click on the scroll-function and select the new value. Validate it with a click. Do the same for all values you wish to change.
- When the configuration have been completed, click on "Set".
- Save the configuration with the button "Save".
- You can test if the registration is OK. For this, click on the "Test registration" button. This function tests the registered parameters and if the board is present on the selected base address. After this you can quit the ADDIREG program. Your board will be initialized with the configured parameters and is ready to be used. The PC should normally reboot. But if there is no message asking you to boot it, the parameters are written in files which do not command rebooting the PC to be saved.

## 6.5 Changing the registration of an existing board

**i****IMPORTANT!**

To register a new board you need **Administrator rights**. If you have user rights you cannot register a new board or change an existing registration!!

- Call the ADDIREG program. Please highlight the board you wish to change. The parameters of the board (like base address, DMA channel and so on) are displayed in the fields under the table.
- Click on the parameters to be changed and open the scroll function.
- Select the new value and validate it with a click. Do the same for each parameter you wish to change.
- Click on the button "SET" to validate your configuration.
- Save your registration with the "SAVE" button.
- Now you can test the registration with "Test registration". If everything is OK you can quit the ADDIREG program. When necessary, a message ask you to reboot your PC.

# 7 SOFTWARE

## 7.1 Driver installation



### IMPORTANT!

Install the **ADDIREG** program first before installing and starting any other application for the board!

### 7.1.1 Installation under DOS and Windows 3.11

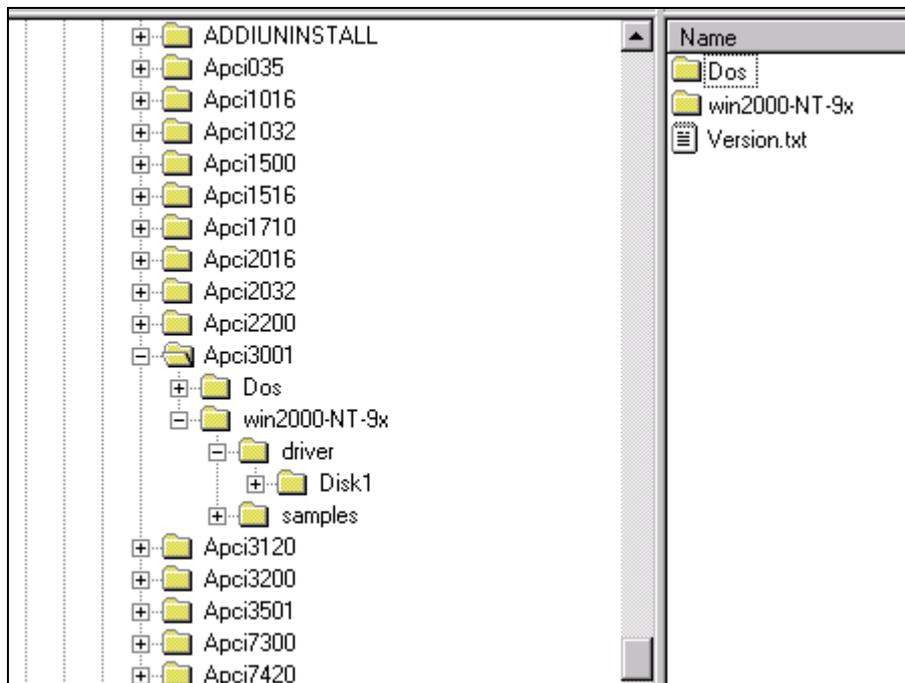
- Copy the contents of APCI3001/Dos/Disk1 on a diskette.
- If several diskettes are to be used, the directory content is stored in several sub-directories (Disk1, Disk2, Disk3...).
- Insert the (first) diskette into a driver and change to this drive.
- Enter <INSTALL>.

The installation program gives you further instructions.

### 7.1.2 Installation under Windows NT

- Change to the CD drive; The required driver and the corresponding samples are stored as follows:

**Fig. 7-1: Installation of the driver**



- Under APCI3001/Win2000-winNT-9x/Driver/Disk1 start the set-up program "setup.exe" (double click).

Proceed as indicated on the screen and read attentively the "Software License" and "Readme".

### 7.1.3 Installation under Windows 2000/9x

After installing your ADDI-DATA board **you have to reboot your PC**.  
Once rebooted, a **message** appears on the screen.

The PC requires information included in the **.inf files** and asks for the path where they are saved. The .inf files contains:

- the name of the board
- the name of the manufacturer
- the hardware type of the board (all ADDI-DATA boards are considered as multifunction boards)

The inf files are stored on the CD-ROM in the directory CD/INF

**i**

#### **IMPORTANT!**

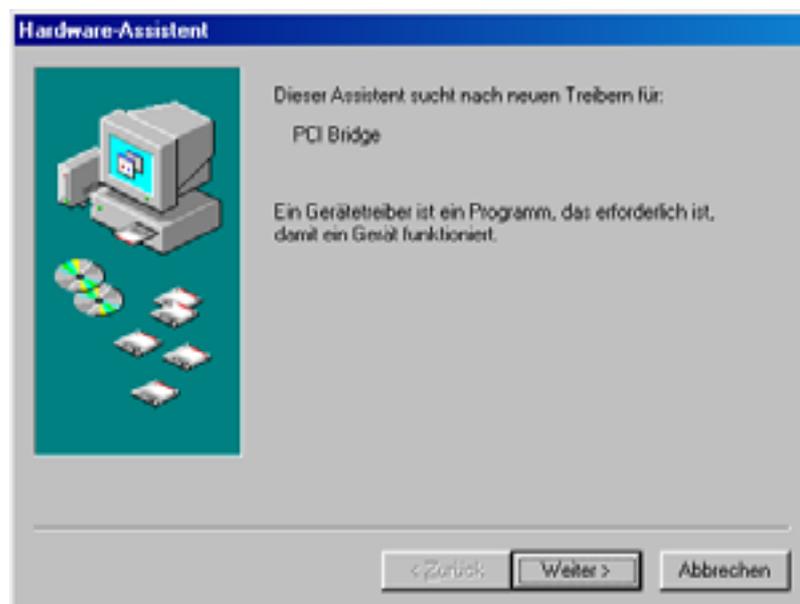
Make sure you also indicated the directory for the **used operating system**.

In case you only enter the path to the INF directory, the operating system may randomly choose one directory and thus install the wrong INF files.

Enter the directory of the used operating system:

- Win2000 for Windows 2000
- Win98 for Windows 9x

**Fig. 7-2: Inquiry of the .inf files (German version)**



Please enter the path of the needed files.

To install the standard software, proceed as described in the former paragraph.  
(Installation under Windows NT)

## 7.2 Installation of the software samples

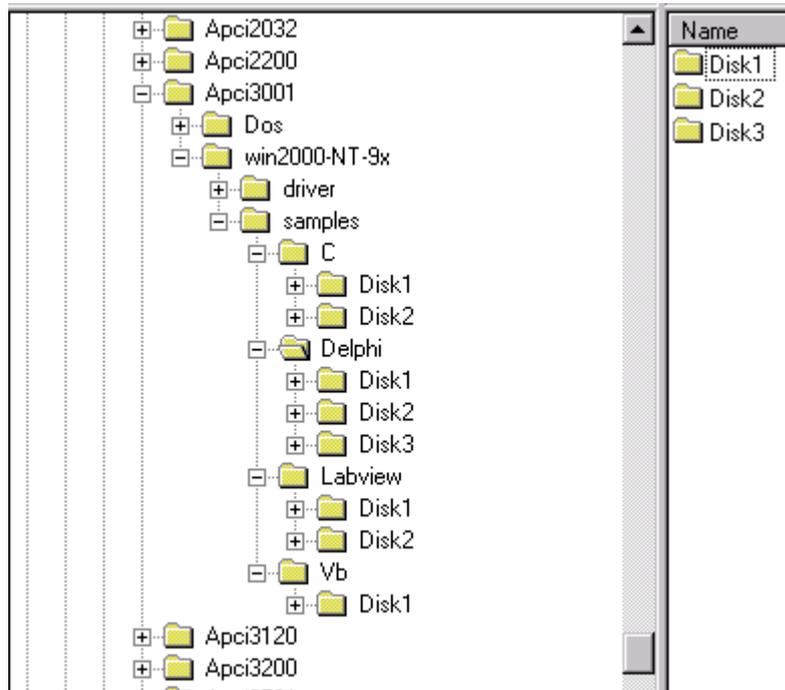
### 7.2.1 Installation under DOS

The software samples for DOS are automatically installed with the driver.

### 7.2.2 Installation under Windows NT/9x/2000

- Change to the CD drive. The files for the required samples are to be found as follows:

**Fig. 7-3: Installation of the samples**



- Select the required programming language.
- Start the set-up program "setup.exe" (double click)
- Select one of the 3 parameters
  - 1- typical: all files are installed (Samples, source code and exe files).
  - 2- compact: only the source code is installed.
  - 3- custom: you can select which files are to be installed.

Proceed as indicated on the screen and read attentively the "Software License" and "Readme". In "custom", you can select your operating system.

The installation program gives you further instructions.

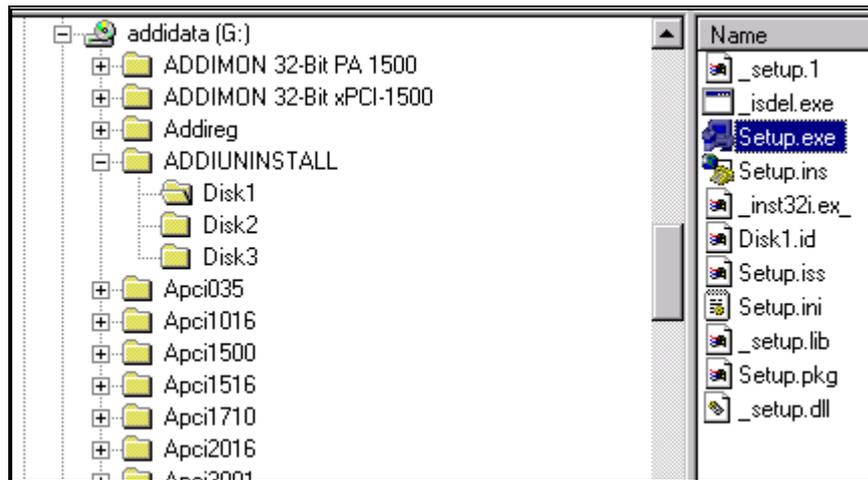
## 7.3 The ADDI-UNINSTALL program

### 7.3.1 Installation of ADDI-UNINSTALL

The ADDI-UNINSTALL program is delivered on the CD-ROM.

- Change to the CD drive and start the set-up file (double click).

**Fig. 7-4: Installation of the ADDI-UNINSTALL program**

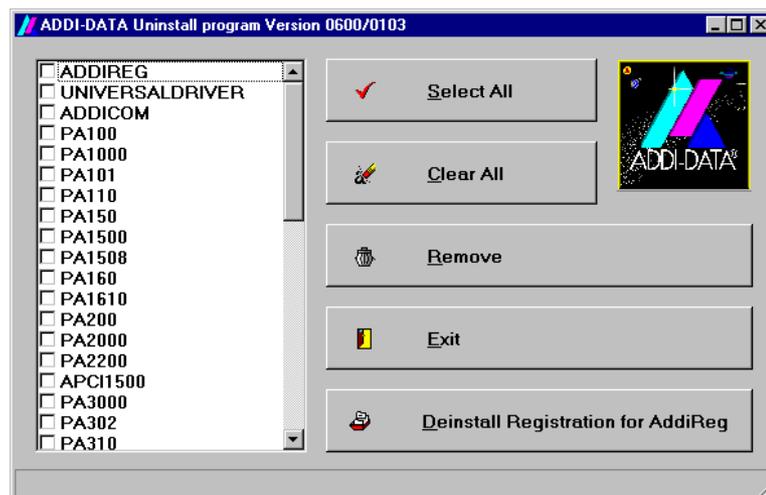


- Proceed as indicated on the screen.

### 7.3.2 Software uninstalling with ADDI-UNINSTALL

- Start the ADDI\_UNINSTALL program.

**Fig. 7-5: The ADDI\_UNINSTALL program**



- Select the software or the driver to be deinstalled. Enter it in the corresponding check box.
- Click on "Remove". Proceed as indicated until the complete removal of the program.

## Uninstall ADDIREG

- Click on "Deinstall registration for AddiReg".
- Proceed as indicated until the complete removal of ADDIREG.

You can also download the ADDI-UNINSTALL program from the Internet.

## 7.4 Software downloads from the Internet

Do not hesitate to visit us or e-mail your questions.

Our Internet page is accessed:

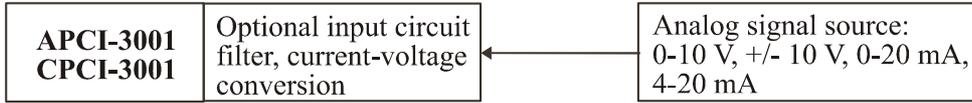
- per e-mail: [info@addi-data.de](mailto:info@addi-data.de)
- per Internet : <http://www.addi-data.de>. or  
<http://www.addi-data.com>

### Free downloads of the standard software

You can download the latest version of the software for the board **XPCI-3001**.

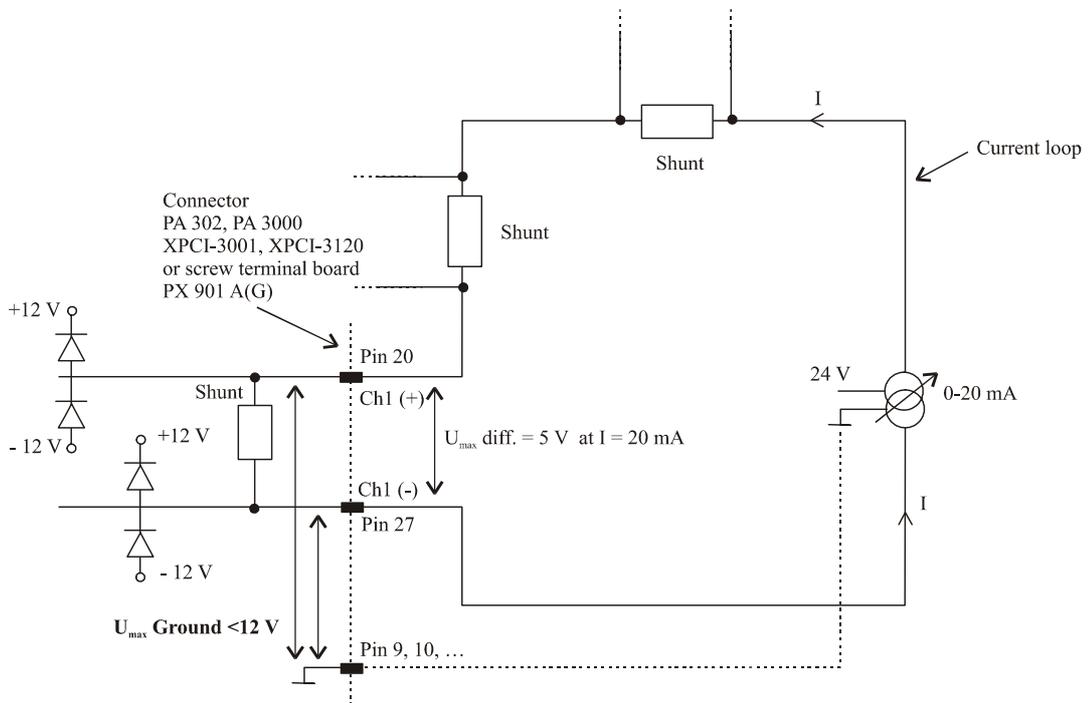
# 8 CONNECTION TO THE PERIPHERAL

## 8.1 Connection principle



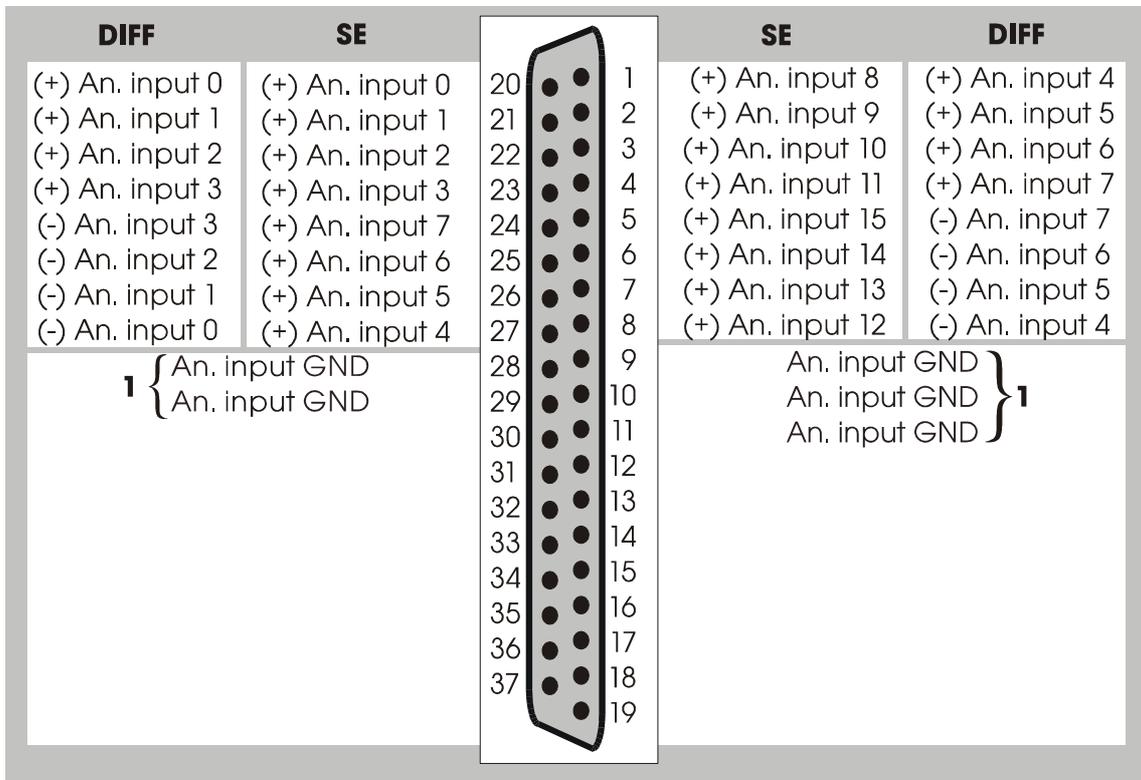
At option DC the board is to be inserted at the end of the current loop so that the voltage ( $U_{\max}$  Ground) amounts maximum 12 V between the differential input pin and the ground.

**Fig. 8-1: Current loop circuitry for the option DC**



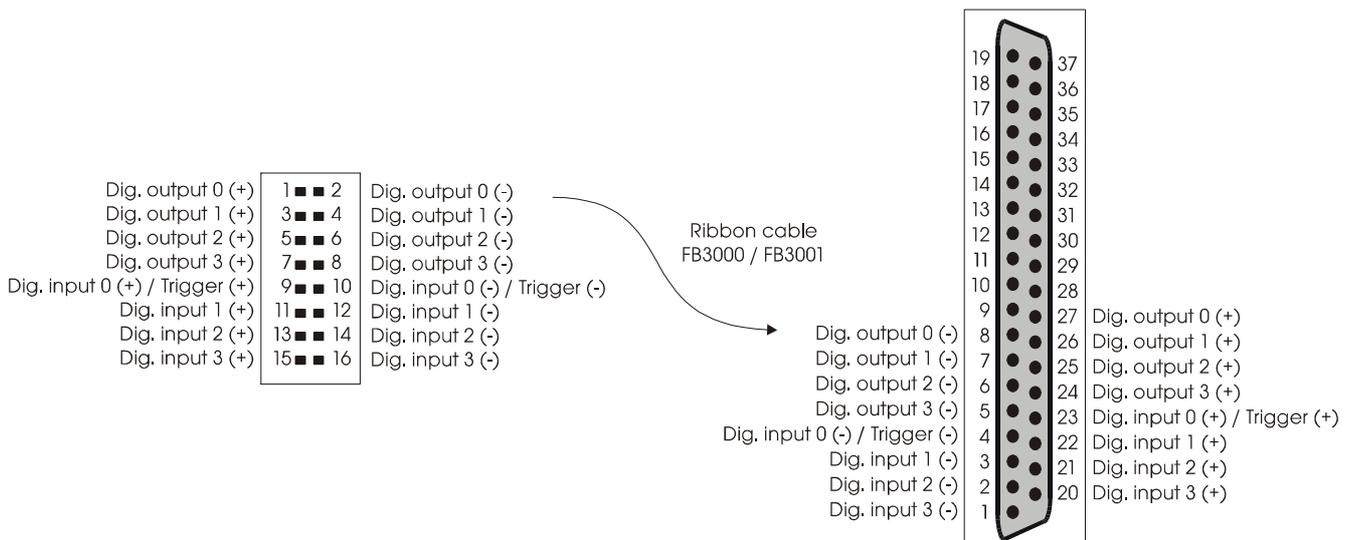
## 8.2 Connector pin assignment

**Fig. 8-2: 37-pin SU-D male connector**



1: The analog input channels have a common ground line

**Fig. 8-3: 16-pin male connector connected to a 37-pin SUB-D male connector**



**i**

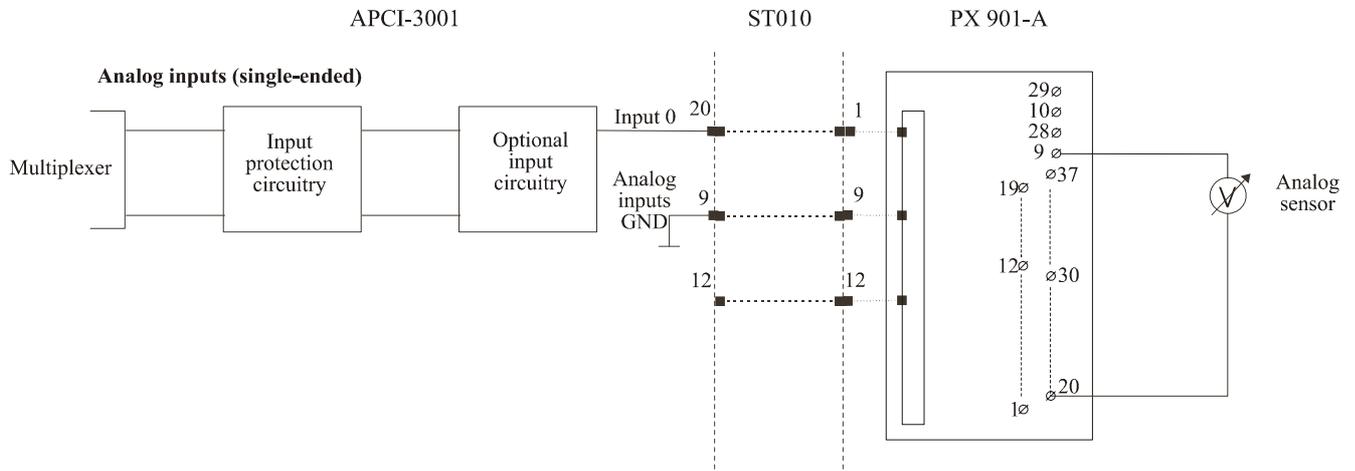
**IMPORTANT!**

Insert the FB3000/FB3001 on the connector with the red cable lead on the side of the pin 1. See 11 "Jumper location on the board".

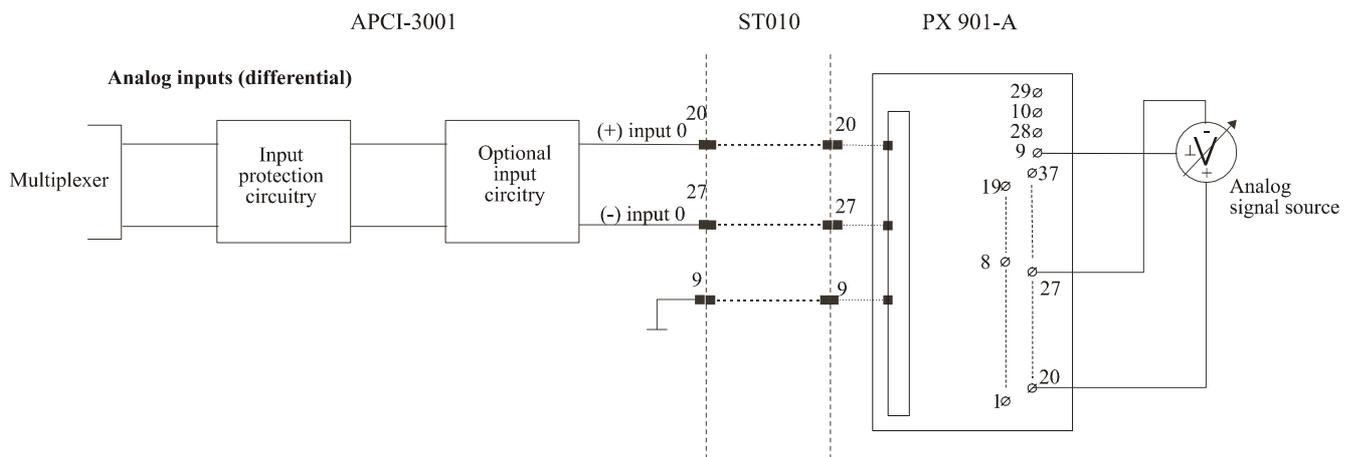
### 8.3 Connection examples

#### 8.3.1 Analog input channels

**Fig. 8-4: Analog input channels (SE)**



**Fig. 8-5: Analog input channels (diff.)**



### 8.3.2 Digital input and output channels

Fig. 8-6: Digital output channels

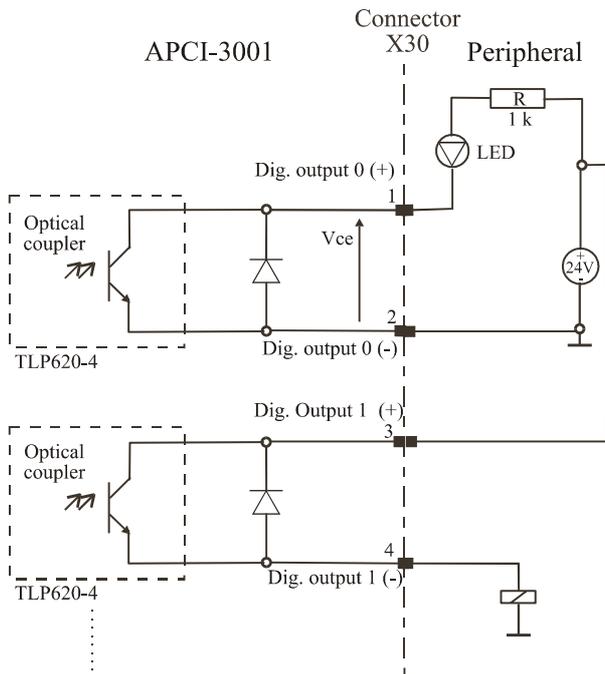
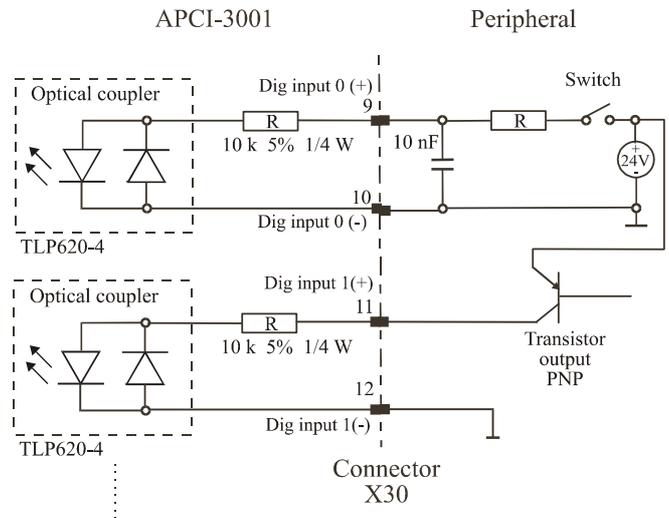
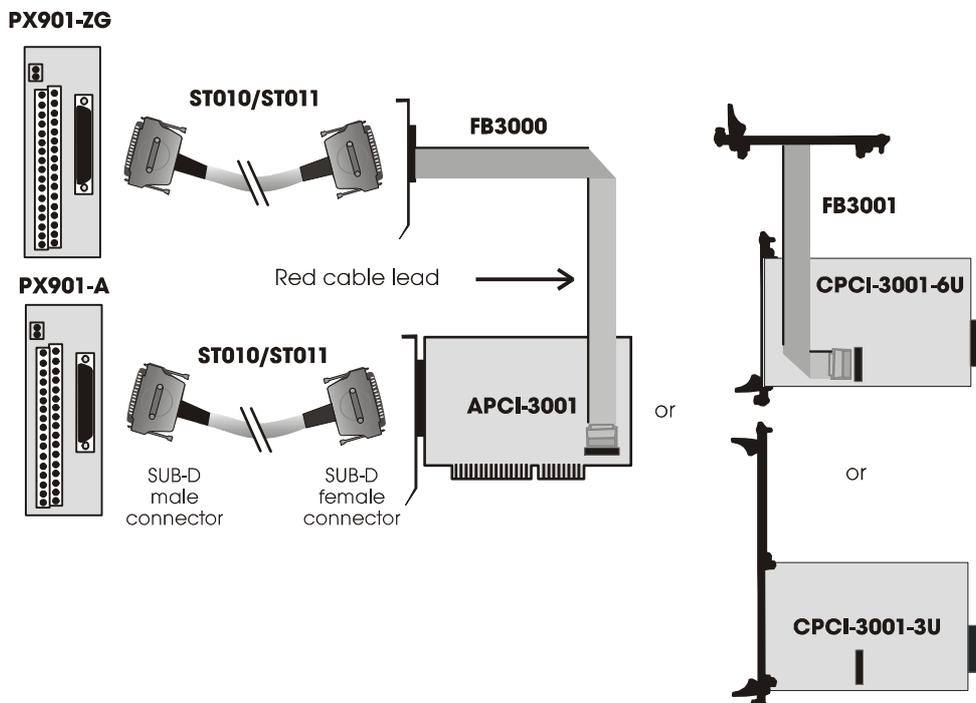


Fig. 8-7: Digital input channels



### 8.3.3 Connection to screw terminal boards

Fig. 8-8: Connection to the PX 901 screw terminal board



**i**

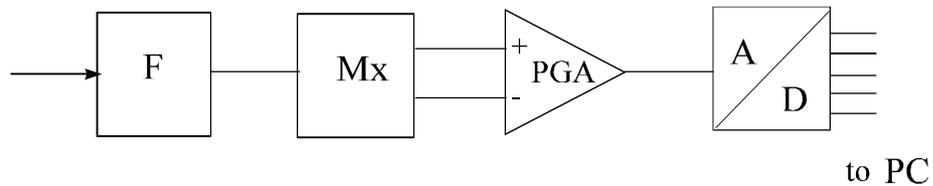
**IMPORTANT!**

Insert the FB3000/FB3001 on the connector with the red cable lead on the side of the pin 1. See 11 "Jumper location on the board".

## 9 FUNCTIONS OF THE BOARD

### 9.1 Analog input channels

Up to 16 analog signals can be connected to the board. It is possible to configure either ground-related or differential measuring (jumper-selectable).



After reaching the multiplexer via a filter (RC module), the signals are led through a programmable gain amplifier to the 16-bit A/D converter.

*The analog input voltage range (0-10 V; ±10 V) and the gain can be configured through software.*

**It is thus possible**

- to have for each channel a different voltage (or current)
- and to use the best resolution of the A/D converter.

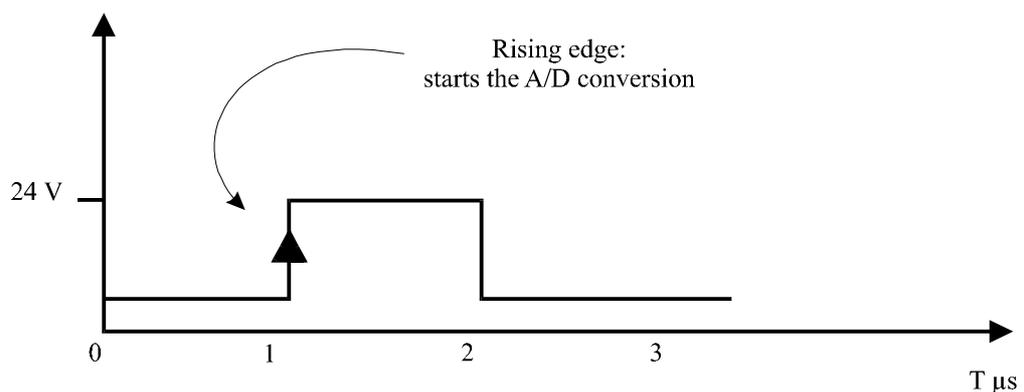
**Scan lists (list of the channel features)**

These lists can be located in a 16-byte deep RAM.

They allow more flexible data acquisitions. You determine their depth by software.

Possible acquisition of the **scan lists**:

- one channel after the other by software trigger (each channel must be triggered),
- the list of the channels is processed once (scans) by software trigger,
- through Timer0: the list can be cyclically handled,
- through Timer0 and 1: the list can be handled during a defined time interval.
- through Timer0, 1 and 2: a defined number of conversions or scans is processed.
- the programmed AD conversion is started by a 24 V signal via the input 0 of the 4 digital input channels.



Data exchanges to the PC occur through a 256 word-deep **FIFO**.

- Polling is possible, analysis of the EOC and EOS bit
- Interrupt at EOC, EOS, END OF DMA,
- DMA mode at EOC
- Data is partly loaded (according to the type of conversion) in the FIFO.

## 9.2 Time-multiplex system

Data acquisition with the **APCI-/CPCI-3001** is based on a time-multiplex system. The board is equipped with a single A/D converter to which the channels are led through an analog multiplexer (software and hardware controlled).

The programmable gain amplifier is highly resistive. It is equipped with a capacitive line from the output channel of the multiplexer to the input channel of the INA, so that each channel is protected by a low-pass filter (RC module).

By switching from one channel to another, the output capacity of the multiplexer is to be converted with the new value.

There is a certain delay between the channel conversion and the start of the A/D converter.

This time delay corresponds to the settling time of an end value. This value depends on the resolution of the acquisition. (e.g.: 0,01 % at 12-bit).

The time delay depends on the following factors:

- the switching time of the amplifier, about 3.5  $\mu$ s.
- the maximum voltage bounce from a channel to another.
- the source impedance of the sensory mechanism.
- Option SF = 10 K // 470 nF Fc/-3 dB ca. 30 Hz
- Option DF = 20 K // 470 nF Fc/-3 dB ca. 30 Hz

The delay is supported on the **APCI-/CPCI-3001** board by the 16-bit Timer 0.

You can set this time in steps of 1  $\mu$ s from 10  $\mu$ s to 32767  $\mu$ s. In the delivered API functions the delay time is the parameter *ui\_ConvertTiming* (or *ui\_AcquisitionTiming*; see documentation: Standard software).

With the scan list the next channel can switch on during the current conversion (Delay: 10  $\mu$ s). This enables to reach the maximum conversion rate of 100 kHz on several channels with low-impedant sensors.

The following table (without guarantee) gives indications for setting the variable *ui\_ConvertTiming* (or *ui\_AcquisitionTiming*). The optimum time depends on your system and can only be established through experiments.

When the data acquisition is run by software control (direct conversion) the following table applies:

<b>Rsource</b>	<b>ui_ConvertTiming ui_AcquisitionTiming</b>
<100R	10..20
< 500R	20..60
< 1K	500..700
< 10K	1000..2000
< 50K	10000...32767

When the data acquisition is run by hardware control (cyclic conversion) the user has to consider the A/D conversion time.

<b>Rsource</b>	<b>ui_ConvertTiming ui_AcquisitionTiming</b>
<100R	10 ..20
< 500R	30..80
< 1K	500.. 700
< 10K	1000..2000
< 50K	10000..32767

The time  $l\_DelayTiming$  must be higher than the number of channels to be converted x  $ui\_AcquisitionTiming$ .

## 10 SOFTWARE EXAMPLES

**i**

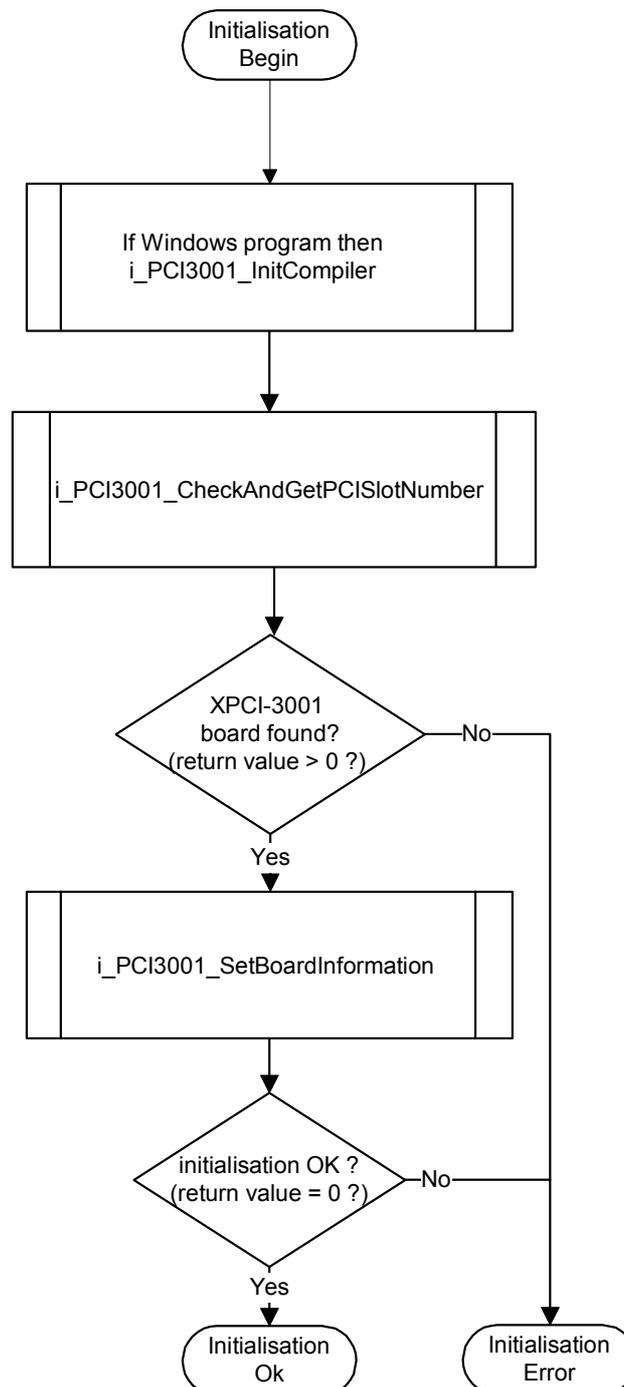
### IMPORTANT!

These **examples** are not the functions of a real-time application. They only represent the **possible functions** which can be processed with the board XPCI-3001.

### 10.1 Initialisation

#### 10.1.1 Initialisation of an APCI-/CPCI-3001 board

a) Flow chart



**b) Example in C**

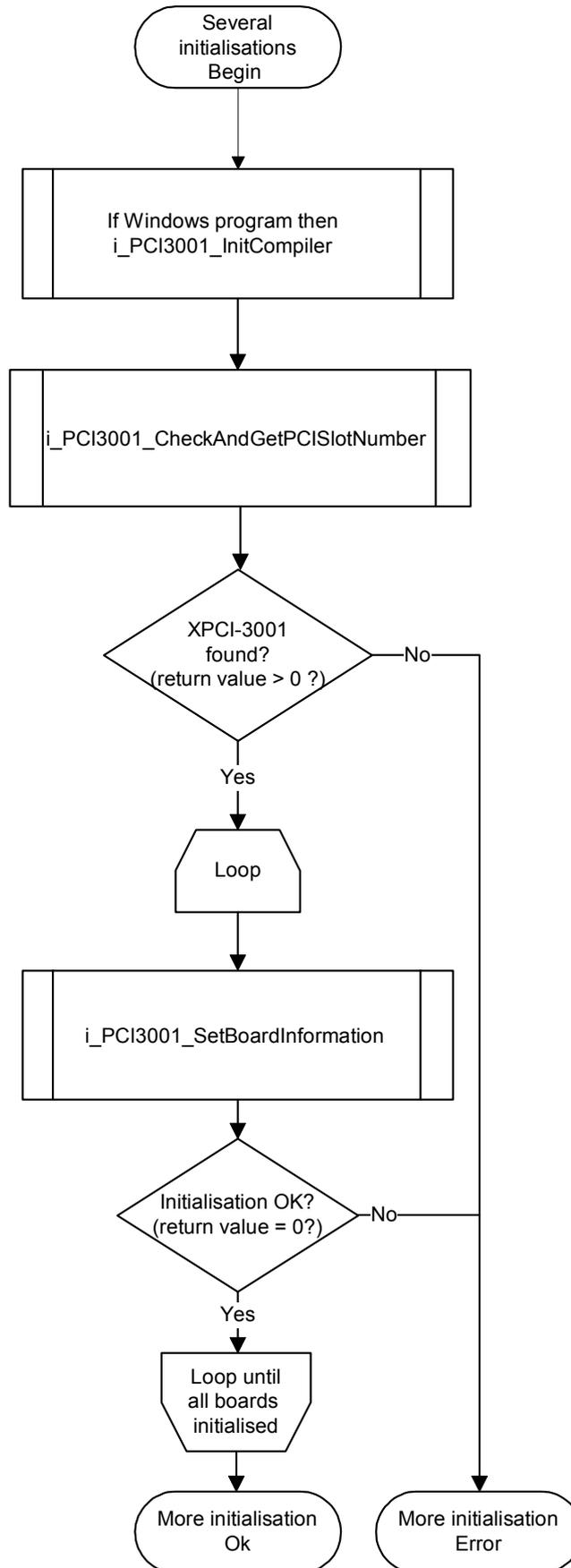
```
int Initialisation(unsigned char *pb_BoardHandle)
{
    unsigned char b_SlotNumberArray [8];

    #ifdef _Windows
        i_PCI3001_InitCompiler (DLL_COMPILER_C);
    #endif

    if(i_PCI3001_CheckAndGetPCISlotNumber (b_SlotNumberArray))
    {
        if(i_PCI3001_SetBoardInformation (b_SlotNumberArray[0],
                                         16,
                                         pb_BoardHandle) == 0)
        {
            return (0);      /* OK */
        }
        else
        {
            return (-1);     /* ERROR */
        }
    }
    else
    {
        return (-1);        /* ERROR */
    }
}
```

### 10.1.2 Initialisation of several APCI-/CPCI-3001 boards

a) Flow chart



**b) Example in C**

```
int MoreInitialisation(unsigned char *pb_BoardHandleArray)
{
    int          i_NbrOfBoard;
    int          i_Cpt;
    unsigned char b_SlotNumberArray [8];

#ifdef _Windows
    i_PCI3001_InitCompiler (DLL_COMPILER_C);
#endif

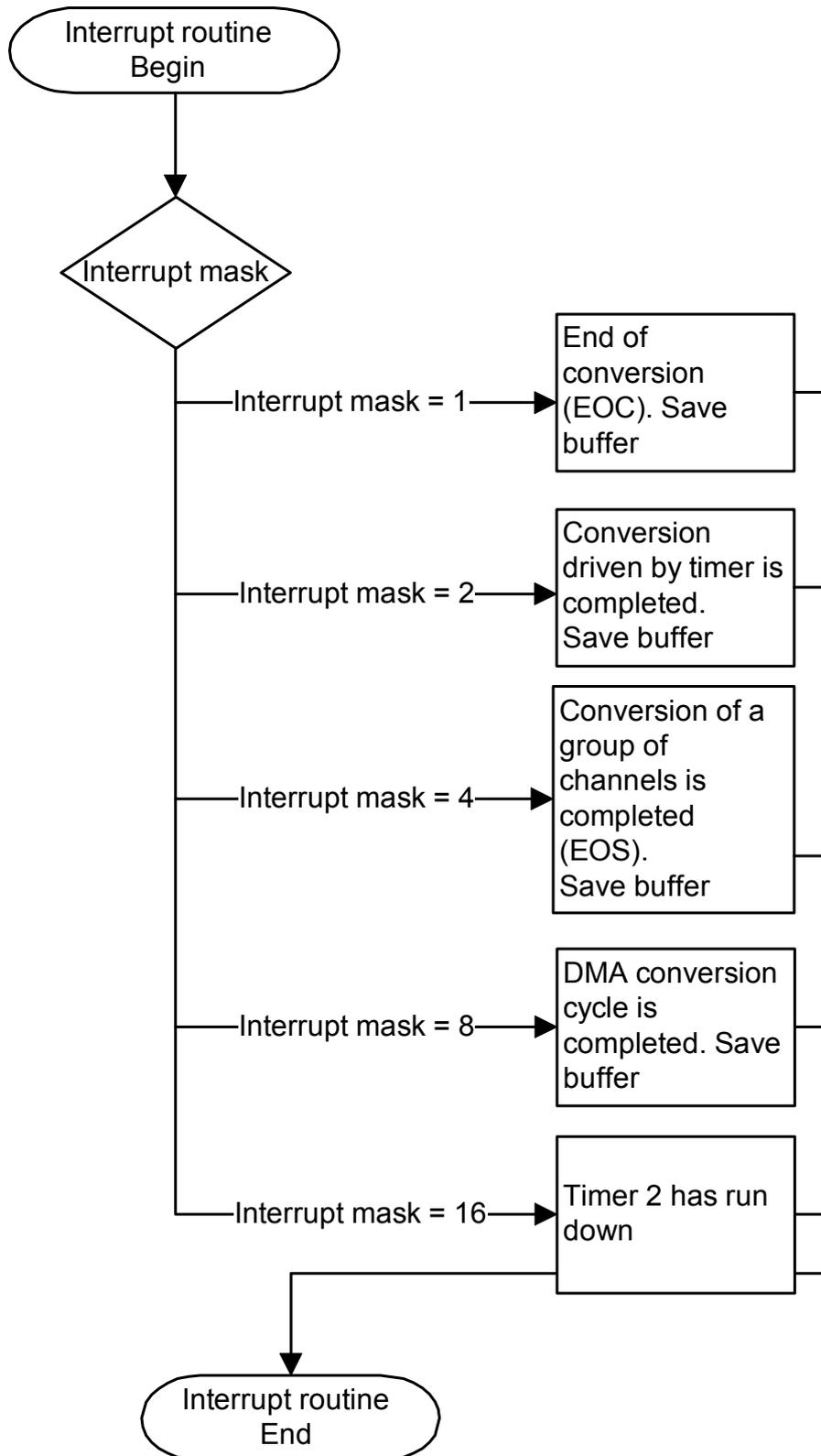
    i_NbrOfBoard = i_PCI3001_CheckAndGetPCISlotNumber (b_SlotNumberArray)

    if(i_NbrOfBoard > 0)
    {
        for (i_Cpt = 0; i_Cpt < i_NbrOfBoard; i_Cpt ++)
        {
            if (i_PCI3001_SetBoardInformation (b_SlotNumberArray[i_Cpt],
                                                16,
                                                &pb_BoardHandleArray [i_Cpt]) != 0)
            {
                break;
            }
        }

        if (i_Cpt == i_NbrOfBoard)
        {
            return (i_Cpt); /* Return number of board found */
        }
        else
        {
            return (-1);    /* ERROR */
        }
    }
    else
    {
        return (-1);      /* ERROR */
    }
}
```

## 10.2 Interrupt

### a) Flow chart



**b) Example in C for DOS und Windows 3.1x**

```
unsigned int  ui_SaveArray [16];          /* Global buffer          */
unsigned int  ui_TimerIntCpt  = 0; /* Timer interrupt counter*/
unsigned char b_ReceiveInterrupt = 0; /* Interrupt flag          */

_VOID_ v_InterruptRoutine (BYTE_ b_BoardHandle, BYTE_ b_InterruptMask, PUINT_
pui_ValueArray)
{
    unsigned int ui_Cpt;

    switch(b_InterruptMask)
    {
        case 1:
            /* EOC interrupt */
            ui_SaveArray[0] = pui_ValueArray[1];
            break;
        case 2:
            /* EOS interrupt Acquisition */
            for (ui_Cpt=0;ui_Cpt<pui_ValueArray [0];ui_Cpt++)
                ui_SaveArray [ui_Cpt] = pui_ValueArray [1+ui_Cpt];
            break;
        case 4:
            /* EOS interrupt Read More*/
            for (ui_Cpt=0;ui_Cpt<pui_ValueArray [0];ui_Cpt++)
                ui_SaveArray [ui_Cpt] = pui_ValueArray [1+ui_Cpt];
            break;
        case 8:
            /* DMA completed */
            for (ui_Cpt=0;ui_Cpt<16;ui_Cpt++)
                ui_SaveArray [ui_Cpt] = pui_ValueArray [ui_Cpt];
            break;
        case 16:
            /* Timer 2 has run down */
            ui_TimerIntCpt = ui_TimerIntCpt + 1;
            break;
        default :
            break;
    }
    b_ReceiveInterrupt = 1;
}
```

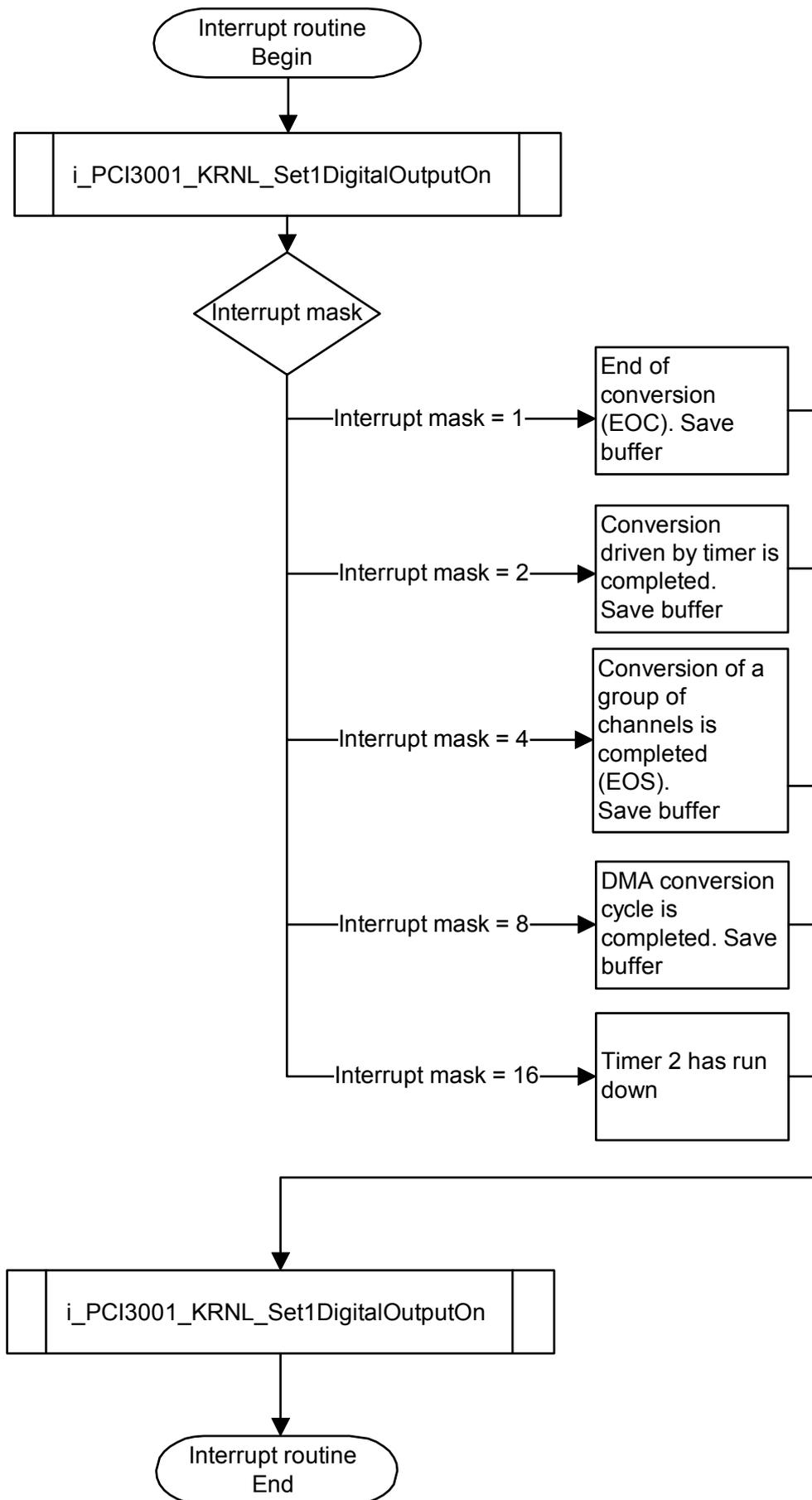
**c) Example in C for Windows NT/95/98 (asynchronous mode)**

```
unsigned int ui_SaveArray [16];          /* Global Buffer */
unsigned int ui_TimerIntCpt = 0;         /* Timer interrupt counter */
unsigned char b_ReceiveInterrupt = 0;   /* Interrupt flag */

_VOID_ v_InterruptRoutine ( BYTE_ b_BoardHandle, BYTE_ b_InterruptMask,
                           PUINT_ pui_ValueArray,
                           BYTE_ b_UserCallingMode, VOID *pv_UserSharedMemory)
{
    unsigned long ul_Cpt;
    unsigned short int *pusi_Index;

    pusi_Index = pui_ValueArray;
    switch(b_InterruptMask)
    {
        case 1:
            /* EOC interrupt */
            ui_SaveArray[0] = pui_ValueArray[1];
            break;
        case 2:
            /* EOS interrupt Acquisition */
            for (ul_Cpt=0;ul_Cpt<pui_ValueArray [0];ul_Cpt++)
                ui_SaveArray [ul_Cpt] = pui_ValueArray [1+ul_Cpt];
            break;
        case 4:
            /* EOS interrupt Read More*/
            for (ul_Cpt=0;ul_Cpt<pui_ValueArray [0];ul_Cpt++)
                ui_SaveArray [ul_Cpt] = pui_ValueArray [1+ul_Cpt];
            break;
        case 8:
            /* DMA completed */
            for (ul_Cpt=0;ul_Cpt<ul_NbrAcquisitionDMA;ul_Cpt++)
                ui_SaveArray [ul_Cpt] = pusi_Index[ul_Cpt];
            break;
        case 16:
            /* Timer 2 has run down */
            ui_TimerIntCpt = ui_TimerIntCpt + 1;
            break;
        default :
            break;
    }
    b_ReceiveInterrupt = 1;
}
```

d) Flow chart for Windows NT/95/98 (synchronous mode)



## e) Example in C for Windows NT/95/98 (synchronous mode)

```

typedef struct
{
    unsigned int ui_SaveArray [16];          /* Global Buffer */
    unsigned int ui_TimerIntCpt ;           /* Timer interrupt counter */
    unsigned char b_ReceiveInterrupt ;      /* Interrupt flag */
}str_UserStruct;
str_UserStruct *ps_GlobalUserStruct;

_VOID_ v_InterruptRoutine (BYTE_ b_BoardHandle,BYTE_ b_InterruptMask,
                           PUINT_ pui_ValueArray,
                           BYTE_ b_UserCallingMode,VOID *pv_UserSharedMemory)
{
    unsigned int ui_Cpt;
    unsigned short int *pusi_Index;

    str_UserStruct *ps_UserStruct = (str_UserStruct *) pv_UserSharedMemory;
    pusi_Index = pui_ValueArray;

    i_PCI3001_KRNL_Set1DigitalOutputOn(0x390,1);
    if ((b_InterruptMask&1) == 1) /* EOC interrupt */
    {
        ps_UserStruct->ui_SaveArray[0] = pui_ValueArray[1];
    }
    if ((b_InterruptMask&2) == 2) /* EOS interrupt Acquisition */
    {
        for (ui_Cpt= 1;ui_Cpt<= pui_ValueArray [0];ui_Cpt++)
            ps_UserStruct->ui_SaveArray [ui_Cpt] = pui_ValueArray [ui_Cpt];
    }
    if ((b_InterruptMask&4) == 4) /* EOS interrupt Read More*/
    {
        for (ui_Cpt=0;ui_Cpt<pui_ValueArray [0];ui_Cpt++)
            ps_UserStruct->ui_SaveArray [ui_Cpt] = pui_ValueArray [1+ui_Cpt];
    }

    if ((b_InterruptMask&8) == 8) /* DMA completed */

    {

        for (ui_Cpt=0;ui_Cpt<16;ui_Cpt++)

            ps_UserStruct->ui_SaveArray [ui_Cpt] = pusi_Index[ui_Cpt];

    }

    if ((b_InterruptMask&16) == 16)

    {

        /* Timer 2 has run down */

        ps_UserStruct->ui_TimerIntCpt = ps_UserStruct->ui_TimerIntCpt + 1;

    }

    i_PCI3001_KRNL_Set1DigitalOutputOn(0x390,2);

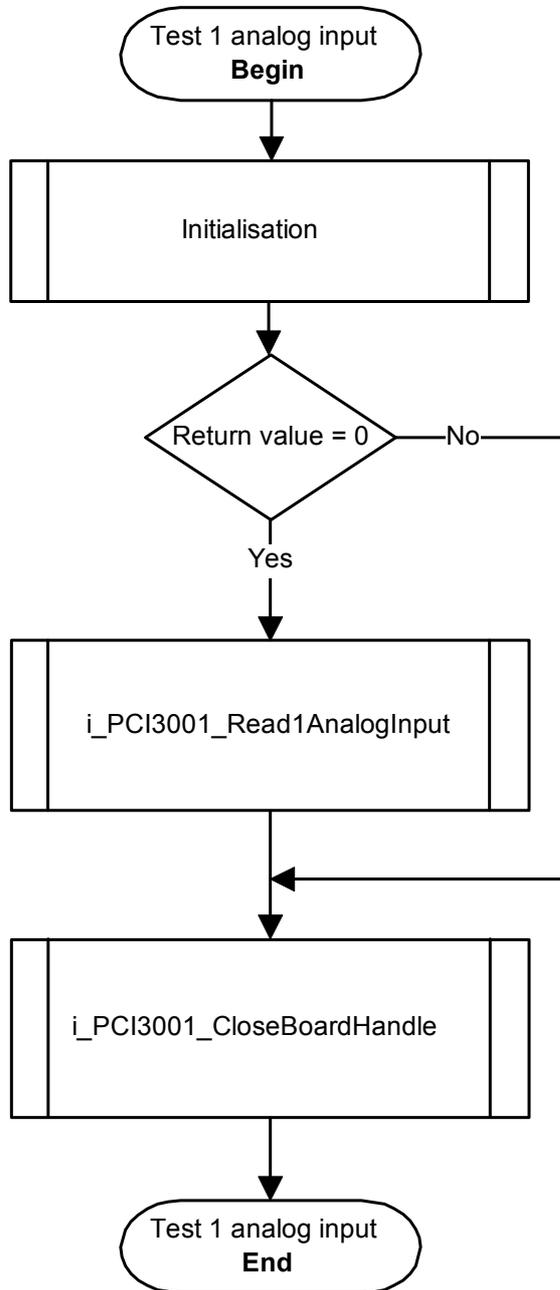
    ps_UserStruct->b_ReceiveInterrupt =ps_UserStruct->b_ReceiveInterrupt + 1;
}

```

### 10.3 Direct conversion of analog input channels

#### 10.3.1 Test of 1 analog input channel

a) Flow chart



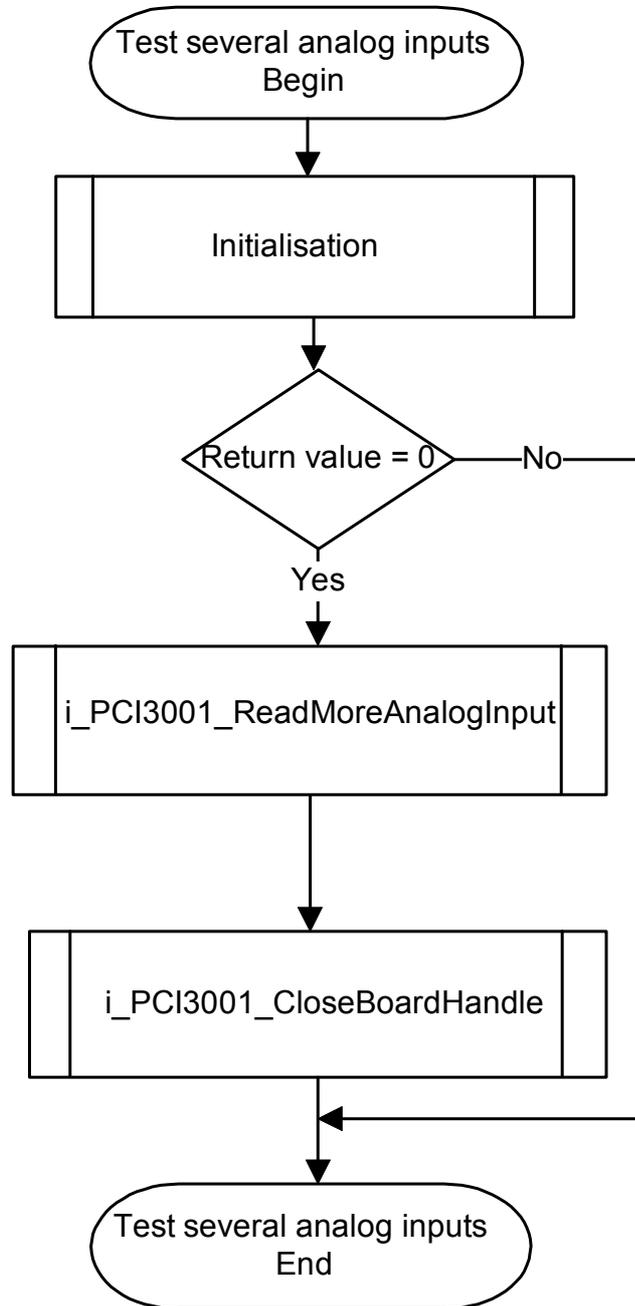
**b) Example in C**

```
void main (void)
{
    unsigned char b_BoardHandle;
    unsigned int  ui_ReadValue;

    if (Initialisation (&b_BoardHandle) == 0)
    {
        if (i_PCI3001_ReadAnalogInput (b_BoardHandle,
                                       PCI3001_CHANNEL_1,
                                       PCI3001_1_GAIN,
                                       PCI3001_UNIPOLAR,
                                       10,
                                       PCI3001_DISABLE,
                                       &ui_ReadValue) == 0)
        {
            printf ("ui_ReadValue = %u", ui_ReadValue);
        }
        else
        {
            printf ("Read value error");
        }
        i_PCI3001_CloseBoardHandle (b_BoardHandle);
    }
    else
    {
        printf ("Initialisation error");
    }
}
```

### 10.3.2 Test of several analog input channels

a) Flow chart



**b) Example in C**

```
void main (void)
{
    unsigned char b_BoardHandle;
    unsigned char b_Gain          [8];
    unsigned char b_Polar        [8];
    unsigned char b_Channel      [8];
    unsigned int  ui_ReadValueArray [8];

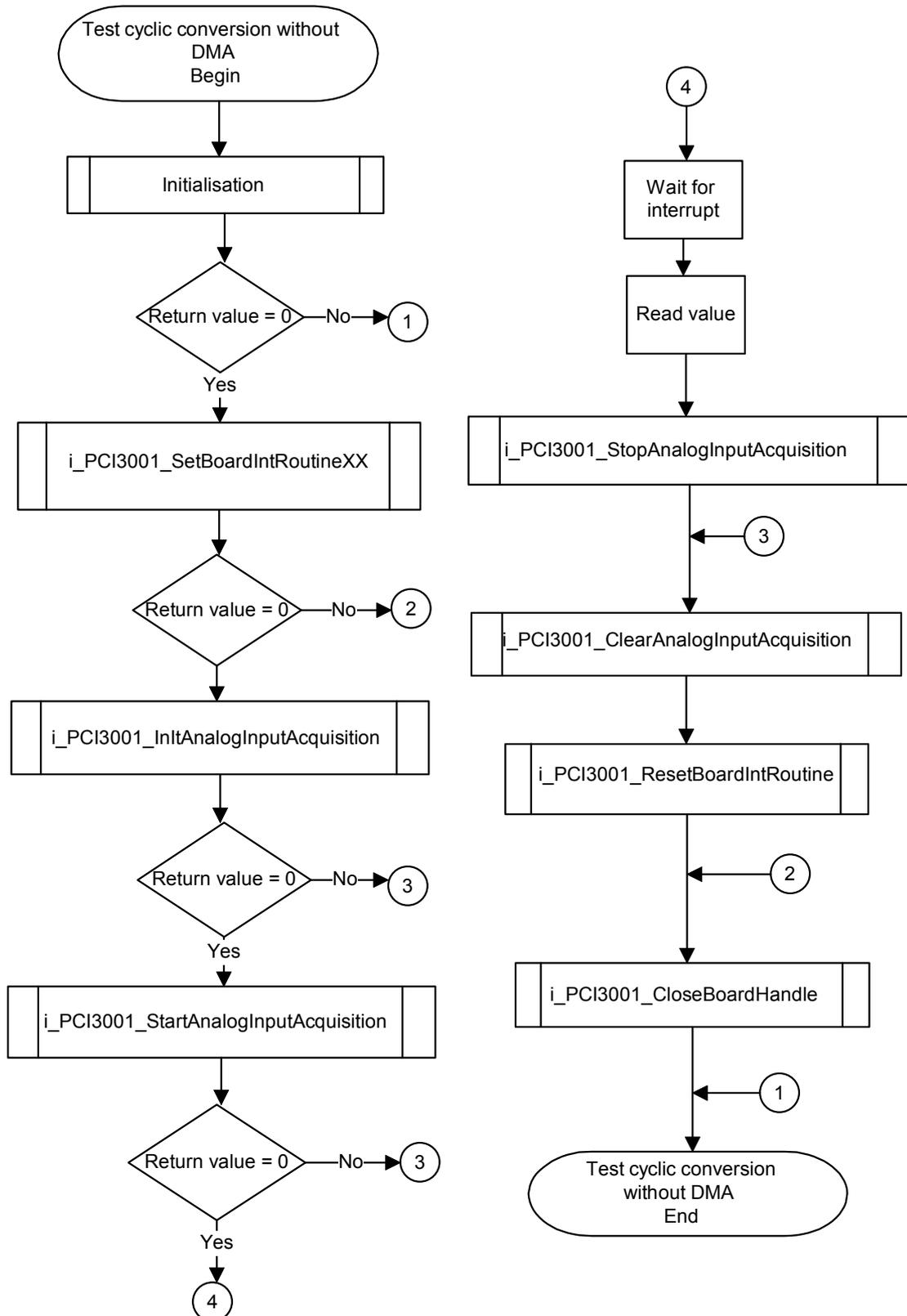
    b_Channel[0] = PCI3001_CHANNEL_0; b_Gain[0] = PCI3001_1_GAIN; b_Polar[0] = PCI3001_UNIPOLAR;
    b_Channel[1] = PCI3001_CHANNEL_1; b_Gain[1] = PCI3001_1_GAIN; b_Polar[1] = PCI3001_UNIPOLAR;
    b_Channel[2] = PCI3001_CHANNEL_2; b_Gain[2] = PCI3001_1_GAIN; b_Polar[2] = PCI3001_UNIPOLAR;
    b_Channel[3] = PCI3001_CHANNEL_3; b_Gain[3] = PCI3001_1_GAIN; b_Polar[3] = PCI3001_UNIPOLAR;
    b_Channel[4] = PCI3001_CHANNEL_4; b_Gain[4] = PCI3001_1_GAIN; b_Polar[4] = PCI3001_UNIPOLAR;
    b_Channel[5] = PCI3001_CHANNEL_5; b_Gain[5] = PCI3001_1_GAIN; b_Polar[5] = PCI3001_UNIPOLAR;
    b_Channel[6] = PCI3001_CHANNEL_6; b_Gain[6] = PCI3001_1_GAIN; b_Polar[6] = PCI3001_UNIPOLAR;
    b_Channel[7] = PCI3001_CHANNEL_7; b_Gain[7] = PCI3001_1_GAIN; b_Polar[7] = PCI3001_UNIPOLAR;

    if (Initialisation (&b_BoardHandle) == 0)
    {
        if (i_PCI3001_ReadMoreAnalogInput (b_BoardHandle, 8, b_Channel, b_Gain,
                                           b_Polar, 10, PCI3001_DISABLE,
                                           ui_ReadValueArray) == 0)
        {
            printf ("ui_ReadValue = %u %u %u %u %u %u %u %u",
                    ui_ReadValue [0], ui_ReadValue [1], ui_ReadValue [2], ui_ReadValue [3],
                    ui_ReadValue [4], ui_ReadValue [5], ui_ReadValue [6], ui_ReadValue [7]);
        }
        else
        {
            printf ("Read value error");
        }
        i_PCI3001_CloseBoardHandle (b_BoardHandle);
    }
    else
    {
        printf ("Initialisation error");
    }
}
```

## 10.4 Cyclic conversion of analog input channels

### 10.4.1 Cyclic conversion without DMA, external trigger and delay

#### a) Flow chart



**b) Pin assignment**

The analog input channels are Single Mode .(See Jumper settings).  
Acquisition of the analog input from 0 to 3 .

Set Pin 28 to - (0 V).

Set Pin 20 input 0

Pin 21 input 1

Pin 22 input 2

Pin 23 input 3 to 10 V.

The value to be read for each input is 4095

**c) Example in C for DOS**

```
void main(void)
{
    int i_Cpt;
    unsigned char b_Gain [4];
    unsigned char b_Polar [4];
    unsigned char b_Channel [4];
    unsigned char b_BoardHandle;
    if (Initialisation(&b_BoardHandle) == 0)
    {
        if (i_PCI3001_SetBoardIntRoutineDos(b_BoardHandle, v_InterruptRoutine) == 0)
        {
            b_Channel[0]=PCI3001_CHANNEL_0;b_Gain[0]=PCI3001_1_GAIN;b_Polar[0]=PCI3001_UNIPOLAR;
            b_Channel[1]=PCI3001_CHANNEL_1;b_Gain[1]=PCI3001_1_GAIN;b_Polar[1]=PCI3001_UNIPOLAR;
            b_Channel[2]=PCI3001_CHANNEL_2;b_Gain[2]=PCI3001_1_GAIN;b_Polar[2]=PCI3001_UNIPOLAR;
            b_Channel[3]=PCI3001_CHANNEL_3;b_Gain[3]=PCI3001_1_GAIN;b_Polar[3]=PCI3001_UNIPOLAR;
            if (i_PCI3001_InitAnalogInputAcquisition (b_BoardHandle,4,
                b_Channel,b_Gain,b_Polar,
                PCI3001_SIMPLE_MODUS,
                PCI3001_DISABLE,1500,
                0,4,PCI3001_DMA_NOT_USED,
                PCI3001_SINGLE)==0)
            {
                b_ReceiveInterrupt = 0;
                if (i_PCI3001_StartAnalogInputAcquisition (b_BoardHandle) == 0)
                {
                    for (i_Cpt=0;i_Cpt<4;i_Cpt++)
                    {
                        while (b_ReceiveInterrupt == 0);
                        b_ReceiveInterrupt = 0;
                        printf("\n Acquisition 1 %u %u %u %u",ui_SaveArray[0], ui_SaveArray[1],
                            ui_SaveArray[2], ui_SaveArray[3]);
                    }
                    i_PCI3001_StopAnalogInputAcquisition(b_BoardHandle);
                }
                else
                printf("\n Start acquisition error");
                i_PCI3001_ClearAnalogInputAcquisition(b_BoardHandle);
            }
            else
                printf("\n Acquisition initialisation error");
            i_PCI3001_ResetBoardIntRoutine(b_BoardHandle);
        }
        else
            printf("\n Interrupt routine initialisation error");
        i_PCI3001_CloseBoardHandle(b_BoardHandle);
    }
    else
        printf("\n Initialisation error");
}
```

## d) Example in C for Windows 3.1x

```

void main(void)
{
    int i_Cpt;
    unsigned char b_Gain [4];
    unsigned char b_Polar [4];
    unsigned char b_Channel [4];
    unsigned char b_BoardHandle;
    if (Initialisation(&b_BoardHandle) == 0)
    {
        if (i_PCI3001_SetBoardIntRoutineWin16(b_BoardHandle, v_InterruptRoutine) == 0)
        {
            b_Channel[0]=PCI3001_CHANNEL_0;b_Gain[0]=PCI3001_1_GAIN;b_Polar[0]=PCI3001_UNIPOLAR;
b_Channel[1]=PCI3001_CHANNEL_1;b_Gain[1]=PCI3001_1_GAIN;b_Polar[1]=PCI3001_UNIPOLAR;
b_Channel[2]=PCI3001_CHANNEL_2;b_Gain[2]=PCI3001_1_GAIN;b_Polar[2]=PCI3001_UNIPOLAR;
b_Channel[3]=PCI3001_CHANNEL_3;b_Gain[3]=PCI3001_1_GAIN;b_Polar[3]=PCI3001_UNIPOLAR;
            if (i_PCI3001_InitAnalogInputAcquisition (b_BoardHandle,4,b_Channel,
                b_Gain,b_Polar,
                PCI3001_SIMPLE_MODUS,
                PCI3001_DISABLE,1500,
                0,4,PCI3001_DMA_NOT_USED,
                PCI3001_SINGLE)==0)
            {
                b_ReceiveInterrupt = 0;
                if (i_PCI3001_StartAnalogInputAcquisition (b_BoardHandle) == 0)
                {
                    for (i_Cpt=0;i_Cpt<4;i_Cpt++)
                    {
                        while (b_ReceiveInterrupt == 0);
                        b_ReceiveInterrupt = 0;
                        printf("\n Acquisition 1 %u %u %u %u",ui_SaveArray[0], ui_SaveArray[1],
                            ui_SaveArray[2], ui_SaveArray[3]);
                    }
                    i_PCI3001_StopAnalogInputAcquisition(b_BoardHandle);
                }
                else
                    printf("\n Start acquisition error");
                i_PCI3001_ClearAnalogInputAcquisition(b_BoardHandle);
            }
            else
                printf("\n Acquisition initialisation error");
            i_PCI3001_ResetBoardIntRoutine(b_BoardHandle);
        }
        else
            printf("\n Interrupt routine initialisation error");
        i_PCI3001_CloseBoardHandle(b_BoardHandle);
    }
    else
        printf("\n Initialisation error");
}

```

e) Example in C for Windows NT/95/98 (asynchronous mode)

```

void main(void)
{
    int i_Cpt;
    unsigned char b_Gain [4];
    unsigned char b_Polar [4];
    unsigned char b_Channel [4];
    unsigned char b_BoardHandle;
    if (Initialisation(&b_BoardHandle) == 0)
    {
        if (i_PCI3001_SetBoardIntRoutineWin32(b_BoardHandle,0,NULL,v_InterruptRoutine) ==
0)
        {
            b_Channel[0]=PCI3001_CHANNEL_0;b_Gain[0]=PCI3001_1_GAIN;b_Polar[0]=PCI3001_UNIPOLAR;
            b_Channel[1]=PCI3001_CHANNEL_1;b_Gain[1]=PCI3001_1_GAIN;b_Polar[1]=PCI3001_UNIPOLAR;
            b_Channel[2]=PCI3001_CHANNEL_2;b_Gain[2]=PCI3001_1_GAIN;b_Polar[2]=PCI3001_UNIPOLAR;
            b_Channel[3]=PCI3001_CHANNEL_3;b_Gain[3]=PCI3001_1_GAIN;b_Polar[3]=PCI3001_UNIPOLAR;
            if (i_PCI3001_InitAnalogInputAcquisition (b_BoardHandle,4,b_Channel,
                b_Gain,b_Polar,
                PCI3001_SIMPLE_MODUS,
                PCI3001_DISABLE,1500,
                0,4,PCI3001_DMA_NOT_USED,
                PCI3001_SINGLE)==0)
            {
                b_ReceiveInterrupt = 0;
                if (i_PCI3001_StartAnalogInputAcquisition (b_BoardHandle) == 0)
                {
                    for (i_Cpt=0;i_Cpt<4;i_Cpt++)
                    {
                        while (b_ReceiveInterrupt == 0);
                        b_ReceiveInterrupt = 0;
                        printf("\n Acquisition 1 %u %u %u %u",ui_SaveArray[0], ui_SaveArray[1],
                            ui_SaveArray[2], ui_SaveArray[3]);
                    }
                    i_PCI3001_StopAnalogInputAcquisition(b_BoardHandle);
                }
                else
                    printf("\n Start acquisition error");
                i_PCI3001_ClearAnalogInputAcquisition(b_BoardHandle);
            }
            else
                printf("\n Acquisition initialisation error");
            i_PCI3001_ResetBoardIntRoutine(b_BoardHandle);
        }
        else
            printf("\n Interrupt routine initialisation error");
        i_PCI3001_CloseBoardHandle(b_BoardHandle);
    }
    else
        printf("\n Initialisation error");
}

```

## f) Example in C for Windows NT/95/98 (synchronous mode)

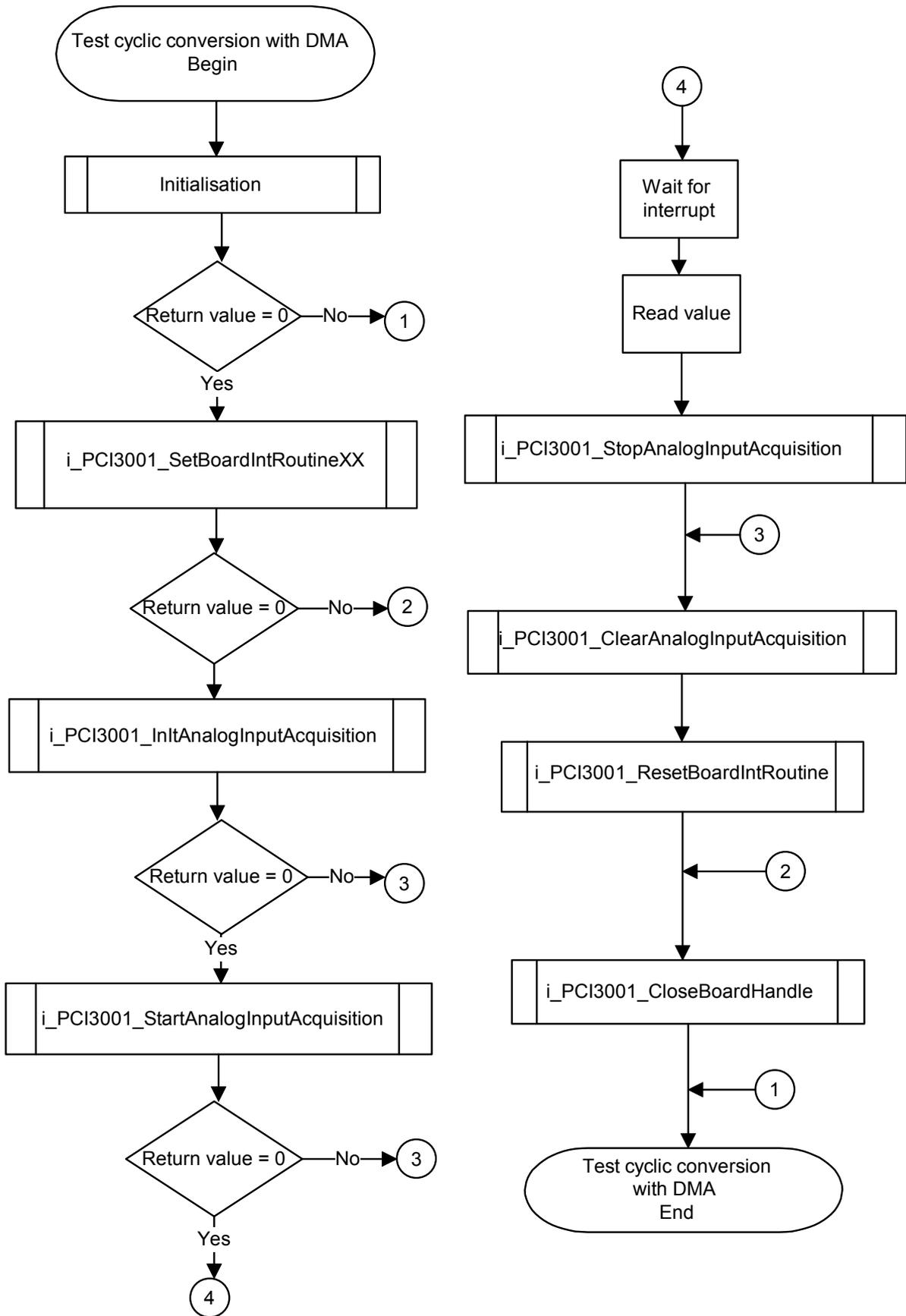
```

void main(void)
{
    int i_Cpt;
    unsigned char b_Gain [4]; unsigned char b_Polar [4]; unsigned char b_Channel [4];
    unsigned char b_BoardHandle;
    if (Initialisation(&b_BoardHandle) == 0)
    {
        if (i_PCI3001_SetBoardIntRoutineWin32(b_BoardHandle, sizeof(str_UserStruct),
            (void **) &ps_GlobalUserStruct,
            v_InterruptRoutine) == 0)
        {
            b_Channel[0]=PCI3001_CHANNEL_0;b_Gain[0]=PCI3001_1_GAIN;b_Polar[0]=PCI3001_UNIPOLAR;
            b_Channel[1]=PCI3001_CHANNEL_1;b_Gain[1]=PCI3001_1_GAIN;b_Polar[1]=PCI3001_UNIPOLAR;
            b_Channel[2]=PCI3001_CHANNEL_2;b_Gain[2]=PCI3001_1_GAIN;b_Polar[2]=PCI3001_UNIPOLAR;
            b_Channel[3]=PCI3001_CHANNEL_3;b_Gain[3]=PCI3001_1_GAIN;b_Polar[3]=PCI3001_UNIPOLAR;
            if (i_PCI3001_InitAnalogInputAcquisition (b_BoardHandle,4,b_Channel,b_Gain,
                b_Polar,PCI3001_SIMPLE_MODUS,
                PCI3001_DISABLE,1500,0,4,
                PCI3001_DMA_NOT_USED,PCI3001_SINGLE)==0)
            {
                ps_GlobalUserStruct -> b_ReceiveInterrupt = 0;
                if (i_PCI3001_StartAnalogInputAcquisition (b_BoardHandle) == 0)
                {
                    for (i_Cpt=0;i_Cpt<4;i_Cpt++)
                    {
                        while (ps_GlobalUserStruct -> b_ReceiveInterrupt == 0);
                        ps_GlobalUserStruct -> b_ReceiveInterrupt = 0;
                        printf("\n Acquisition 1 %u %u %u %u", ps_GlobalUserStruct ->
                            ui_SaveArray[0],
                            ps_GlobalUserStruct -> ui_SaveArray[1],
                            ps_GlobalUserStruct -> ui_SaveArray[2],
                            ps_GlobalUserStruct -> ui_SaveArray[3]);
                    }
                    i_PCI3001_StopAnalogInputAcquisition(b_BoardHandle);
                }
                else
                    printf("\n Start acquisition error");
                i_PCI3001_ClearAnalogInputAcquisition(b_BoardHandle);
            }
            else
                printf("\n Acquisition initialisation error");
            i_PCI3001_ResetBoardIntRoutine(b_BoardHandle);
        }
        else
            printf("\n Interrupt routine initialisation error");
        i_PCI3001_CloseBoardHandle(b_BoardHandle);
    }
    else
        printf("\n Initialisation error");
}

```

10.4.2 Cyclic conversion with DMA without external trigger and delay

a) Flow chart



**b) Pin assignment**

The analog input channels are in Single Mode. (See Jumper settings).

Acquisition of the analog input from 0 to 3 .

Set Pin 28 to - (0 V).

Set Pin 20 input 0

Pin 21 input 1

Pin 22 input 2

Pin 23 input 3 to 10 V.

The value to be read for each input is 4095.

**c) Example in C for DOS**

```

void main(void)
{
    int i_Cpt;
    unsigned char b_Gain [4];
    unsigned char b_Polar [4];
    unsigned char b_Channel [4];
    unsigned char b_BoardHandle;
    if (Initialisation(&b_BoardHandle) == 0)
    {
        if (i_PCI3001_SetBoardIntRoutineDos(b_BoardHandle, v_InterruptRoutine) == 0)
        {
            b_Channel[0]=PCI3001_CHANNEL_0;b_Gain[0]=PCI3001_1_GAIN;b_Polar[0]=PCI3001_UNIPOLAR;
            b_Channel[1]=PCI3001_CHANNEL_1;b_Gain[1]=PCI3001_1_GAIN;b_Polar[1]=PCI3001_UNIPOLAR;
            b_Channel[2]=PCI3001_CHANNEL_2;b_Gain[2]=PCI3001_1_GAIN;b_Polar[2]=PCI3001_UNIPOLAR;
            b_Channel[3]=PCI3001_CHANNEL_3;b_Gain[3]=PCI3001_1_GAIN;b_Polar[3]=PCI3001_UNIPOLAR;
            if (i_PCI3001_InitAnalogInputAcquisition
                (b_BoardHandle,4,b_Channel,b_Gain,b_Polar,
                 PCI3001_SIMPLE_MODUS,PCI3001_DISABLE,1500,0,16,PCI3001_DMA_USED,
                 PCI3001_SINGLE) == 0)
            {
                b_ReceiveInterrupt = 0;
                if (i_PCI3001_StartAnalogInputAcquisition (b_BoardHandle) == 0)
                {
                    while (b_ReceiveInterrupt == 0);
                    b_ReceiveInterrupt = 0;
                    printf("\n Acquisition 1 %u %u %u %u %u %u %u %u \n %u %u %u %u %u %u %u
%u",
                        ui_SaveArray[0],ui_SaveArray[1],ui_SaveArray[2],ui_SaveArray[3],
                        ui_SaveArray[4],ui_SaveArray[5],ui_SaveArray[6],ui_SaveArray[7],
                        ui_SaveArray[8],ui_SaveArray[9],ui_SaveArray[10],ui_SaveArray[11],
                        ui_SaveArray[12],ui_SaveArray[13],ui_SaveArray[14],ui_SaveArray[15]);
                    i_PCI3001_StopAnalogInputAcquisition(b_BoardHandle);
                    i_PCI3001_ClearAnalogInputAcquisition(b_BoardHandle);
                }
                else
                    printf("\n Start acquisition error");
            }
            else
                printf("\n Acquisition initialisation error");
            i_PCI3001_ResetBoardIntRoutine(b_BoardHandle);
        }
        else
            printf("\n Interrupt routine initialisation error");
        i_PCI3001_CloseBoardHandle(b_BoardHandle);
    }
    else
        printf("\n Initialisation error");
}

```

d) Example in C for Windows 3.1x

```

void main(void)
{
    int i_Cpt;
    unsigned char b_Gain [4];
    unsigned char b_Polar [4];
    unsigned char b_Channel [4];
    unsigned char b_BoardHandle;
    if (Initialisation(&b_BoardHandle) == 0)
    {
        if (i_PCI3001_SetBoardIntRoutineWin16(b_BoardHandle, v_InterruptRoutine) == 0)
        {
            b_Channel[0]=PCI3001_CHANNEL_0;b_Gain[0]=PCI3001_1_GAIN;b_Polar[0]=PCI3001_UNIPOLAR;
            b_Channel[1]=PCI3001_CHANNEL_1;b_Gain[1]=PCI3001_1_GAIN;b_Polar[1]=PCI3001_UNIPOLAR;
            b_Channel[2]=PCI3001_CHANNEL_2;b_Gain[2]=PCI3001_1_GAIN;b_Polar[2]=PCI3001_UNIPOLAR;
            b_Channel[3]=PCI3001_CHANNEL_3;b_Gain[3]=PCI3001_1_GAIN;b_Polar[3]=PCI3001_UNIPOLAR;
            if (i_PCI3001_InitAnalogInputAcquisition (b_BoardHandle,4,b_Channel,b_Gain,b_Polar,
                PCI3001_SIMPLE_MODUS,PCI3001_DISABLE,
                1500,0,16,
                PCI3001_DMA_USED,PCI3001_SINGLE) == 0)
            {
                b_ReceiveInterrupt = 0;
                if (i_PCI3001_StartAnalogInputAcquisition (b_BoardHandle) == 0)
                {
                    while (b_ReceiveInterrupt == 0);
                    b_ReceiveInterrupt = 0;
                    printf("\n Acquisition 1 %u %u %u %u %u %u %u %u \n %u %u %u %u %u %u %u",
                        ui_SaveArray[0],ui_SaveArray[1],ui_SaveArray[2],ui_SaveArray[3],
                        ui_SaveArray[4],ui_SaveArray[5],ui_SaveArray[6],ui_SaveArray[7],
                        ui_SaveArray[8],ui_SaveArray[9],ui_SaveArray[10],ui_SaveArray[11],
                        ui_SaveArray[12],ui_SaveArray[13],ui_SaveArray[14],ui_SaveArray[15]);
                    i_PCI3001_StopAnalogInputAcquisition(b_BoardHandle);
                    i_PCI3001_ClearAnalogInputAcquisition(b_BoardHandle);
                }
                else
                    printf("\n Start acquisition error");
            }
            else
                printf("\n Acquisition initialisation error");
            i_PCI3001_ResetBoardIntRoutine(b_BoardHandle);
        }
        else
            printf("\n Interrupt routine initialisation error");
        i_PCI3001_CloseBoardHandle(b_BoardHandle);
    }
    else
        printf("\n Initialisation error");
}

```

## e) Example in C for Windows NT/95/98 (asynchronous mode)

```

void main(void)
{
    int i_Cpt;
    unsigned char b_Gain [4];
    unsigned char b_Polar [4];
    unsigned char b_Channel [4];
    unsigned char b_BoardHandle;
    if (Initialisation(&b_BoardHandle) == 0)
    {
        if (i_PCI3001_SetBoardIntRoutineWin32(b_BoardHandle,0,NULL,v_InterruptRoutine) ==
0)
        {
            b_Channel[0]=PCI3001_CHANNEL_0;b_Gain[0]=PCI3001_1_GAIN;b_Polar[0]=PCI3001_UNIPOLAR;
            b_Channel[1]=PCI3001_CHANNEL_1;b_Gain[1]=PCI3001_1_GAIN;b_Polar[1]=PCI3001_UNIPOLAR;
            b_Channel[2]=PCI3001_CHANNEL_2;b_Gain[2]=PCI3001_1_GAIN;b_Polar[2]=PCI3001_UNIPOLAR;
            b_Channel[3]=PCI3001_CHANNEL_3;b_Gain[3]=PCI3001_1_GAIN;b_Polar[3]=PCI3001_UNIPOLAR;
            if (i_PCI3001_InitAnalogInputAcquisition (b_BoardHandle,4,b_Channel,b_Gain,b_Polar,
                PCI3001_SIMPLE_MODUS,PCI3001_DISABLE,
                1500,0,16,PCI3001_DMA_USED,
                PCI3001_SINGLE) == 0)
            {
                b_ReceiveInterrupt = 0;
                if (i_PCI3001_StartAnalogInputAcquisition (b_BoardHandle) == 0)
                {
                    while (b_ReceiveInterrupt == 0);
                    b_ReceiveInterrupt = 0;
                    printf("\n Acquisition 1 %u %u %u %u %u %u %u %u \n %u %u %u %u %u %u %u
%u",
                        ui_SaveArray[0],ui_SaveArray[1],ui_SaveArray[2],ui_SaveArray[3],
                        ui_SaveArray[4],ui_SaveArray[5],ui_SaveArray[6],ui_SaveArray[7],
                        ui_SaveArray[8],ui_SaveArray[9],ui_SaveArray[10],ui_SaveArray[11],
                        ui_SaveArray[12],ui_SaveArray[13],ui_SaveArray[14],ui_SaveArray[15]);
                    i_PCI3001_StopAnalogInputAcquisition(b_BoardHandle);
                    i_PCI3001_ClearAnalogInputAcquisition(b_BoardHandle);
                }
                else
                    printf("\n Start acquisition error");
            }
            else
                printf("\n Acquisition initialisation error");
            i_PCI3001_ResetBoardIntRoutine(b_BoardHandle);
        }
        else
            printf("\n Interrupt routine initialisation error");
        i_PCI3001_CloseBoardHandle(b_BoardHandle);
    }
    else
        printf("\n Initialisation error");
}

```

f) Example in C for Windows NT/95/98 (synchronous mode)

```

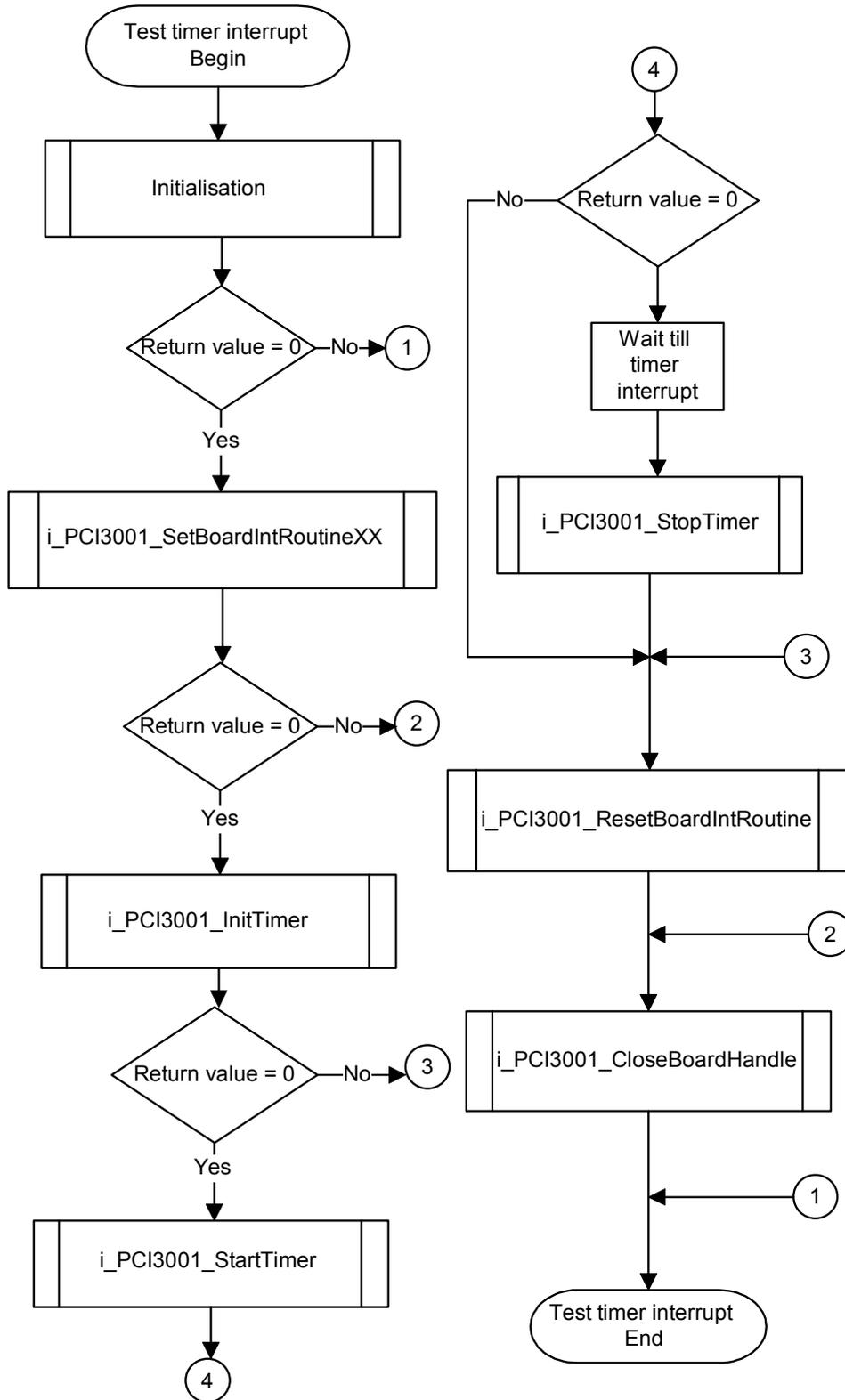
void main(void)
{
int i_Cpt; unsigned char b_Gain [4]; unsigned char b_Polar [4]; unsigned char b_Channel
[4];
unsigned char b_BoardHandle;
if (Initialisation(&b_BoardHandle) == 0)
{
if (i_PCI3001_SetBoardIntRoutineWin32(b_BoardHandle, sizeof(str_UserStruct),
(void **) &GlobalUserStruct, v_InterruptRoutine) == 0)
{
b_Channel[0]=PCI3001_CHANNEL_0;b_Gain[0]=PCI3001_1_GAIN;b_Polar[0]=PCI3001_UNIPOLAR;
b_Channel[1]=PCI3001_CHANNEL_1;b_Gain[1]=PCI3001_1_GAIN;b_Polar[1]=PCI3001_UNIPOLAR;
b_Channel[2]=PCI3001_CHANNEL_2;b_Gain[2]=PCI3001_1_GAIN;b_Polar[2]=PCI3001_UNIPOLAR;
b_Channel[3]=PCI3001_CHANNEL_3;b_Gain[3]=PCI3001_1_GAIN;b_Polar[3]=PCI3001_UNIPOLAR;
if (i_PCI3001_InitAnalogInputAcquisition (b_BoardHandle,4,b_Channel,b_Gain,b_Polar,
PCI3001_SIMPLE_MODUS,PCI3001_DISABLE,
1500,0,16,PCI3001_DMA_USED,PCI3001_SINGLE) ==
0)
{
ps_GlobalUserStruct -> b_ReceiveInterrupt = 0;
if (i_PCI3001_StartAnalogInputAcquisition (b_BoardHandle) == 0)
{
while (ps_GlobalUserStruct -> b_ReceiveInterrupt == 0);
ps_GlobalUserStruct -> b_ReceiveInterrupt = 0;
printf("\n Acquisition 1 %u %u %u %u %u %u %u %u \n %u %u %u %u %u %u",
ps_GlobalUserStruct -> ui_SaveArray[0], ps_GlobalUserStruct -> ui_SaveArray[1],
ps_GlobalUserStruct -> ui_SaveArray[2], ps_GlobalUserStruct -> ui_SaveArray[3],
ps_GlobalUserStruct -> ui_SaveArray[4], ps_GlobalUserStruct -> ui_SaveArray[5],
ps_GlobalUserStruct -> ui_SaveArray[6], ps_GlobalUserStruct -> ui_SaveArray[7],
ps_GlobalUserStruct -> ui_SaveArray[8], ps_GlobalUserStruct -> ui_SaveArray[9],
ps_GlobalUserStruct -> ui_SaveArray[10], ps_GlobalUserStruct -> ui_SaveArray[11],
ps_GlobalUserStruct -> ui_SaveArray[12], ps_GlobalUserStruct -> ui_SaveArray[13],
ps_GlobalUserStruct -> ui_SaveArray[14], ps_GlobalUserStruct -> ui_SaveArray[15]);
i_PCI3001_StopAnalogInputAcquisition(b_BoardHandle);
i_PCI3001_ClearAnalogInputAcquisition(b_BoardHandle);
}
else
printf("\n Start acquisition error");
}
else
printf("\n Acquisition initialisation error");
i_PCI3001_ResetBoardIntRoutine(b_BoardHandle);
}
else
printf("\n Interrupt routine initialisation error");
i_PCI3001_CloseBoardHandle(b_BoardHandle);
}
else
printf("\n Initialisation error");
}
}

```

## 10.5 Timer

### 10.5.1 Test of the timer interrupt

#### a) Flow chart



**b) Example in C for DOS**

```
void main (void)
{
    unsigned char b_BoardHandle;

    if (Initialisation (&b_BoardHandle) == 0)
    {
        if (i_PCI3001_SetBoardIntRoutineDOS (b_BoardHandle, v_InterruptRoutine) == 0)
        {
            ui_TimerIntCpt = 0;
            if (i_PCI3001_InitTimer (b_BoardHandle,1000, PCI3001_ENABLE) == 0)
            {
                if (i_PCI3001_StartTimer (b_BoardHandle) == 0)
                {
                    while (ui_TimerIntCpt == 0);
                    printf ("Receive timer interrupt");
                    i_PCI3001_StopTimer (b_BoardHandle);
                }
                else
                    printf ("Start timer error");
            }
            else
                printf ("Init timer error");
            i_PCI3001_ResetBoardIntRoutine (b_BoardHandle);
        }
        else
            printf ("Interrupt routine initialisation error");
        i_PCI3001_CloseBoardHandle (b_BoardHandle);
    }
    else
        printf ("Initialisation error");
}
```

## c) Example in c for Windows 3.1x

```
void main (void)
{
    unsigned char b_BoardHandle;

    if (Initialisation (&b_BoardHandle) == 0)
    {
        if (i_PCI3001_SetBoardIntRoutineWin16 (b_BoardHandle, v_InterruptRoutine) == 0)
        {
            ui_TimerIntCpt = 0;
            if (i_PCI3001_InitTimer (b_BoardHandle,1000, PCI3001_ENABLE) == 0)
            {
                if (i_PCI3001_StartTimer (b_BoardHandle) == 0)
                {
                    while (ui_TimerIntCpt == 0);
                    printf ("Receive timer interrupt");
                    i_PCI3001_StopTimer (b_BoardHandle);
                }
                else
                {
                    printf ("Start timer error");
                }
            }
            else
            {
                printf ("Init timer error");
            }
            i_PCI3001_ResetBoardIntRoutine (b_BoardHandle);
        }
        else
        {
            printf ("Interrupt routine initialisation error");
        }
        i_PCI3001_CloseBoardHandle (b_BoardHandle);
    }
    else
    {
        printf ("Initialisation error");
    }
}
```

**d) Example in C for Windows NT/95/98 (asynchronous mode)**

```
void main (void)
{
    unsigned char b_BoardHandle;

    if (Initialisation (&b_BoardHandle) == 0)
    {
        if (i_PCI3001_SetBoardIntRoutineWin32 (b_BoardHandle,PCI3001_ASYNCHRONOUS_MODE,
                                                0,NULL,v_InterruptRoutine) == 0)
        {
            ui_TimerIntCpt = 0;
            if (i_PCI3001_InitTimer (b_BoardHandle, 1000, PCI3001_ENABLE) == 0)
            {
                if (i_PCI3001_StartTimer (b_BoardHandle) == 0)
                {
                    while (ui_TimerIntCpt == 0);
                    printf ("Receive timer interrupt");
                    i_PCI3001_StopTimer (b_BoardHandle);
                }
                else
                    printf ("Start timer error");
            }
            else
                printf ("Init timer error");
            i_PCI3001_ResetBoardIntRoutine (b_BoardHandle);
        }
        else
            printf ("Interrupt routine initialisation error");
        i_PCI3001_CloseBoardHandle (b_BoardHandle);
    }
    else
        printf ("Initialisation error");
}
```

**e) Example in C for Windows NT/95/98 (synchronous mode)**

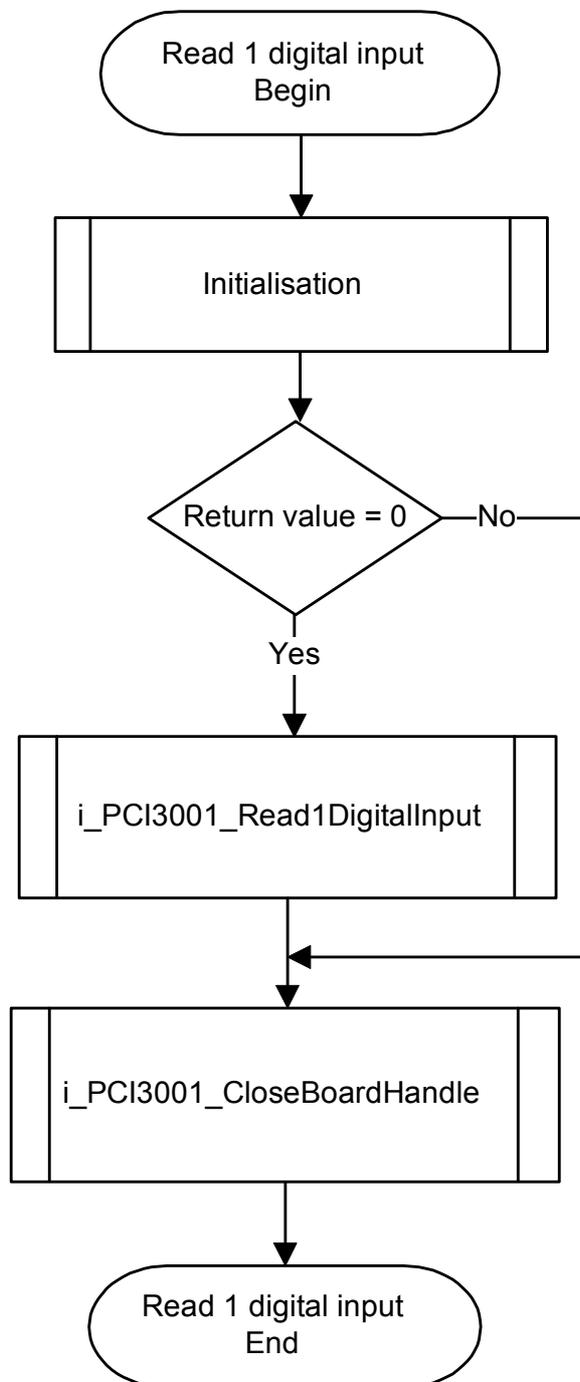
```
void main (void)
{
    unsigned char b_BoardHandle;

    if (Initialisation (&b_BoardHandle) == 0)
    {
        if (i_PCI3001_SetBoardIntRoutineWin32 (b_BoardHandle,PCI3001_SYNCHRONOUS_MODE,
            sizeof(str_UserStruct),(void **) &ps_GlobalUserStruct,v_InterruptRoutine) == 0)
        {
            ps_GlobalUserStruct->ui_TimerIntCpt = 0;
            if (i_PCI3001_InitTimer (b_BoardHandle, 1000, PCI3001_ENABLE) == 0)
            {
                if (i_PCI3001_StartTimer (b_BoardHandle) == 0)
                {
                    while (ps_GlobalUserStruct->ui_TimerIntCpt == 0);
                    printf ("Receive timer interrupt");
                    i_PCI3001_StopTimer (b_BoardHandle);
                }
                else
                {
                    printf ("Start timer error");
                }
            }
            else
            {
                printf ("Init timer error");
            }
            i_PCI3001_ResetBoardIntRoutine (b_BoardHandle);
        }
        else
        {
            printf ("Interrupt routine initialisation error");
        }
        i_PCI3001_CloseBoardHandle (b_BoardHandle);
    }
    else
    {
        printf ("Initialisation error");
    }
}
```

## 10.6 Digital Input channels

### 10.6.1 Reading 1 digital input channel

a) Flow chart



**b) Pin assignment**

Set the digital input 1 to 24 V:

Pin assignment of the 37-pin SUB-D male connector: - at pin 4  
+ at pin 23.

The test result is 1.

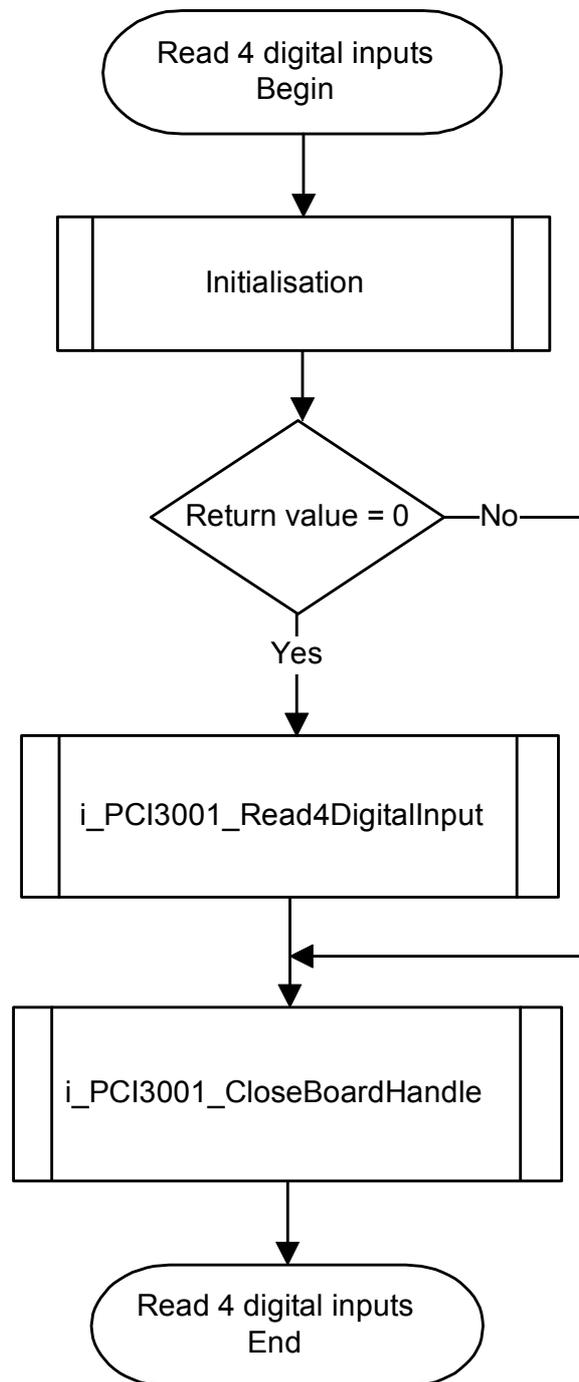
**c) Example in C**

```
void main (void)
{
    unsigned char b_BoardHandle;
    unsigned int  ui_ReadValue;

    if (Initialisation (&b_BoardHandle) == 0)
    {
        if (i_PCI3001_Read1DigitalInput      (b_BoardHandle,
                                             1
                                             &ui_ReadValue) == 0)
        {
            printf ("ui_ReadValue = %u", ui_ReadValue);
        }
        else
        {
            printf ("Read value error");
        }
        i_PCI3001_CloseBoardHandle (b_BoardHandle);
    }
    else
    {
        printf ("Initialisation error");
    }
}
```

### 10.6.2 Reading 4 digital input channels

a) Flow chart

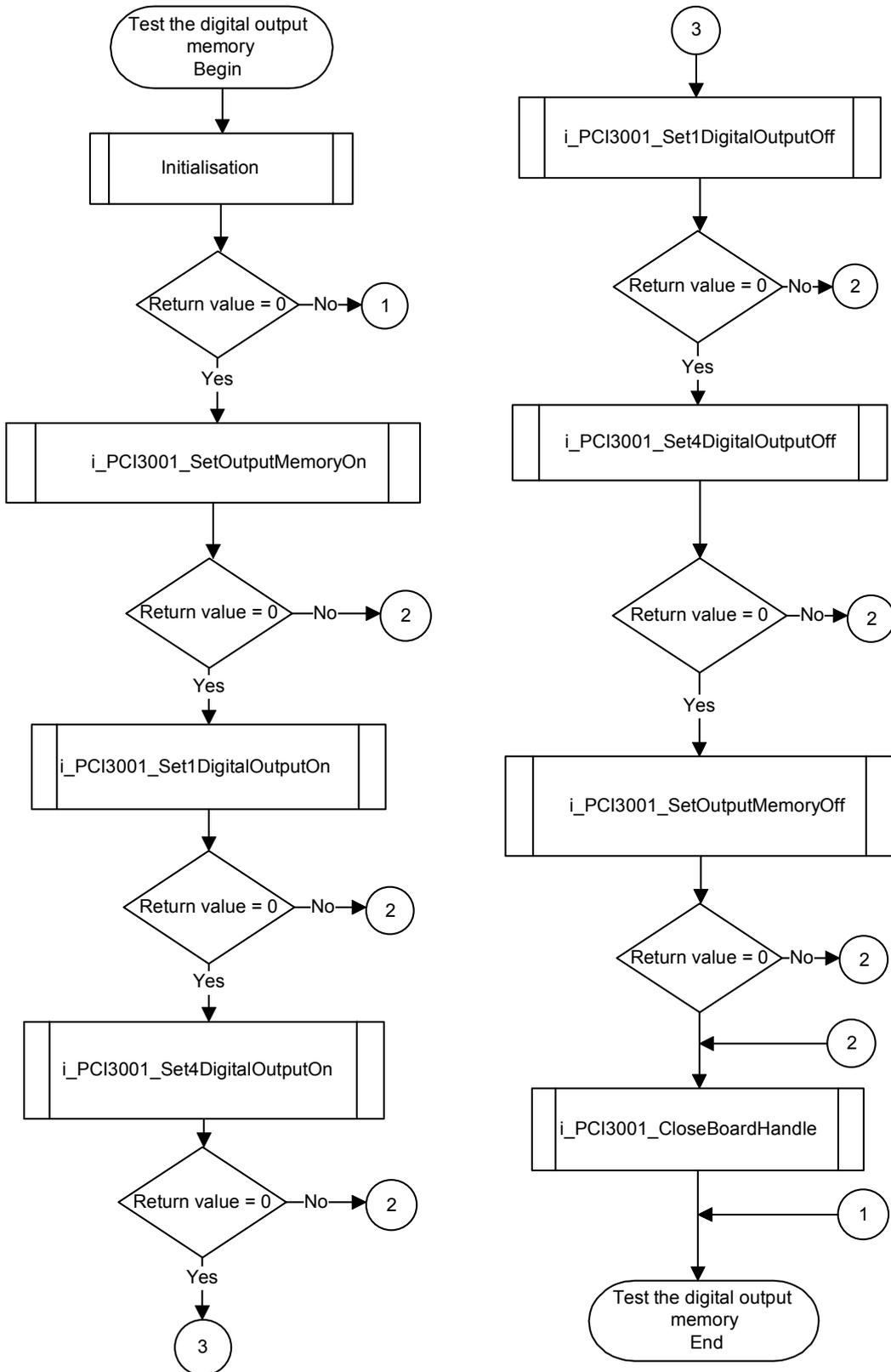




## 10.7 Digital output channels

### 10.7.1 Test of the digital output memory

#### a) Flow chart



**b) Example in C**

```

void main (void)
{
    unsigned char b_BoardHandle;
    if (Initialisation (&b_BoardHandle) == 0)
    {
        if (i_PCI3001_SetOutputMemoryOn (b_BoardHandle) == 0)
        {
            printf ("Digital output memory is activated");
            if (i_PCI3001_Set1DigitalOutputOn (b_BoardHandle, 1) == 0)
            {
                printf (" Output 1 is set ");
                getch();
                if (i_PCI3001_Set4DigitalOutputOn (b_Boardhandle, 14) ==0)
                {
                    printf ("All Output are set ");
                    getch();
                    if (i_PCI3001_Set1DigitalOutputOff (b_Boardhandle, 1 ) == 0)
                    {
                        printf ("Output 1 is reset");
                        getch();
                        if (i_PCI3001_Set4DigitalOutputOff (b_Boardhandle, 14) == 0)
                        {
                            printf ("Output 2,3,4 are reset");
                            if (i_PCI3001_SetOutputMemoryOff (b_Boardhandle) == 0)
                                printf ("Digital Output Memory deactivated");
                            else printf ("Digital Output Memory off error");
                        }
                        else printf ("Reset 4 digital Output error");
                    }
                    else printf ("Reset 1 digital output error ");
                }
                else printf ("Set 4 digital output error");
            }
            else printf ("Set 1 digital output error");
        }
        else printf ("Set Digital Output Memory On error");
        i_PCI3001_CloseBoardHandle (b_BoardHandle);
    }
    else printf ("Initialisation error");
}

```

**INDEX**

- 16-pin male connector 31
- 37-pin SUD-D connector 31
- ADDIREG
  - removing 28
- analog inputs 32, 34
- board
  - component scheme 8
  - function 34–36
  - handling 3
  - inserting 13
  - intended purpose 1
  - jumper settings 11
  - options 5
  - physical set-up 4
  - plugging 13
  - versions 5
- component scheme 8
- connection
  - examples 32
  - pin assignment 31
  - principle 30
  - screw terminal board 33
- connection to the peripheral 25–33
- digital inputs 33
- digital outputs 33
- EMC 4
- functions of the board 34–36
  - analog inputs 34
  - time multiplex system 35
- installation 10–14
- intended purpose 1
- Internet
  - error analysis 29
- jumper settings 11
- limit values 5–7
- limits of use 1
- options 5
- PC
  - closing 14
  - selecting a slot 13
- PX 901 33
- slot
  - selecting a 13
  - types 13
- software
  - examples 37
- technical data 4–7
- time multiplex system 35
- use
  - limits 1
- versions 5