DIN EN ISO 9001:2000
certified

ADDI-DATA GmbH
Dieselstraße 3
D-77833 OTTERSWEIER

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

Technical description

APCI-/CPCI-3001

Analog input board, optically isolated

Edition: 04.07 – 01/2008

Product information

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

Warranty and liability

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

Copyright

ADDI-DATA software product licence

Please read this licence carefully before using the standard software. The customer is only granted the right to use this software if he/she agrees with the conditions of this licence.
The software must only be used to set up the ADDI-DATA boards.
Reproduction of the software is forbidden (except for back-up and for exchange of faulty data carriers). Disassembly, decompilation, decryption and reverse engineering of the software are forbidden. This licence and the software may be transferred to a third party if this party has acquired a board by purchase, has agreed to all the conditions in this licence contract and the original owner does not keep any copies of the software.

Trademarks

WARNING

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

- **people may be injured,**
- **the board, PC and peripheral may be destroyed,**
- **the environment may be polluted**

- **Protect yourself, the others and the environment!**
- **Read carefully the safety precautions (yellow leaflet).**

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

- **Observe the instructions of the manual.**

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

- **Used symbols:**

IMPORTANT!
designates hints and other useful information.

WARNING!
It designates a possibly dangerous situation.
If the instructions are ignored the board, PC and/or peripheral may be destroyed.

Figures

1 DEFINITION OF APPLICATION

1.1 Intended use

The **APCI-3001** board must be inserted in a PC with PCI 5V/32-bit slots which is used as electrical equipment for measurement, control and laboratory pursuant to the norm EN 61010-1 (IEC 61010-1). The used personal computer (PC) must fulfil the requirements of IEC 60950-1 or EN 60950-1 and 55022 or IEC/CISPR 22 and EN 55024 or IEC/CISPR 24.

The use of the board **APCI-3001** in combination with external screw terminal panels requires correct installation according to IEC 60439-1 or EN 60439-1 (switch cabinet / switch box).

The **CPCI-3001** board must be inserted in a CompactPCI/PXI computer with CompactPCI 5V/32-bit slots which is used as electrical equipment for measurement, control and laboratory pursuant to the norm EN 61010-1 (IEC 61010-1). The used personal computer (PC) must fulfil the requirements of IEC 60950-1 or EN 60950-1 and 55022 or IEC/CISPR 22 and EN 55024 or IEC/CISPR 24.

The use of the board **CPCI-3001** in combination with external screw terminal panels requires correct installation according to IEC 60439-1 or EN 60439-1 (switch cabinet / switch box).

1.2 Usage restrictions

The **APCI-3001/CPCI-3001** board must not to be used as safety related part (SRP).

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

The **APCI-3001/CPCI-3001** board must not be used in potentially explosive atmospheres.

The **APCI-3001/CPCI-3001** board must not be used as electrical equipment according to the Low Voltage Directive 2006/95/EC.

1.3 General description of the board

Data exchange between the **XPCI-3001**[1] 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 **XPCI-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.

[1] Common designation for the board APCI-3001 and CPCI-3001.

The **PX901-A** screw terminal panel allows the connection of the analog signals with a shielded cable.

The use of the board **XPCI-3001** in combination with external screw terminal panel or relay 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 following specifications:

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

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

Uses beyond these specifications are not allowed. The manufacturer is not liable for any damages which would result from the non-observance of this clause.

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

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

2 USER

2.1 Qualification

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

- installation
- use,
- maintanance.

2.2 Country-specific regulations

Consider the country-specific regulations about:

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

3 HANDLING OF THE BOARD

Fig. 3-1: Correct handling of the APCI-3001

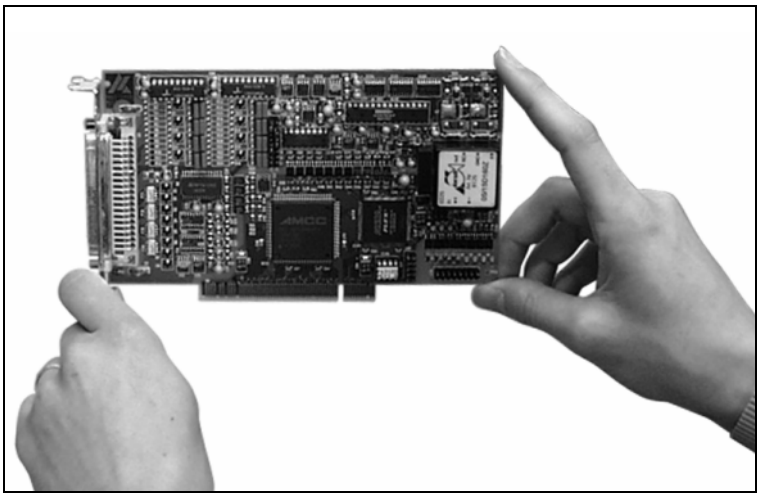
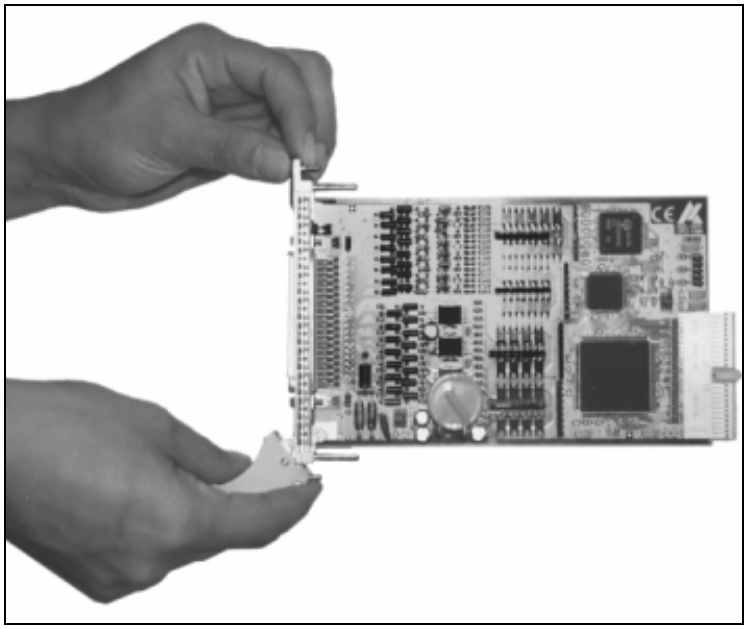


Fig. 3-2: Correct handling of the APCI-3001



4 TECHNICAL DATA

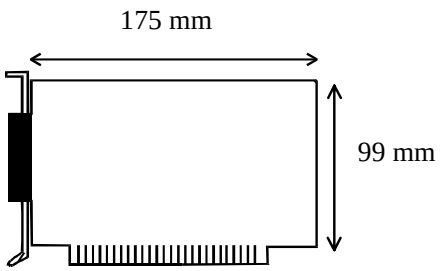
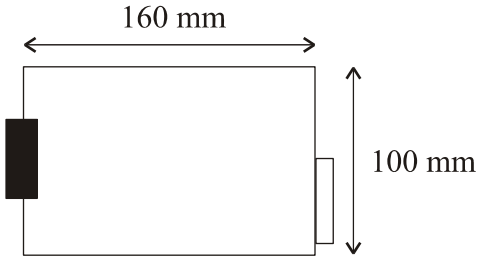
4.1 Electromagnetic compatibility (EMC)

The board **APCI-3001/CPCI-3001** complies with the European EMC directive. The tests were carried out by a certified EMC laboratory in accordance with the norm from the EN 61326 series (IEC 61326). The limit values as set out by the European EMC directive for an industrial environment are complied with.

The respective EMC test report is available on request.

4.2 Physical set-up of the board

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

	APCI-3001	CPCI-3001
Dimensions		
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

Accessories[1]:
Standard cable: **ST010**
Ribbon cable (Digital I/Os): ..... **FB3000** or **FB3001**
Screw terminal panels
Analog inputs: ..... **PX901-A** or
PX901-AG with housing
Digital I/Os: **PX901-ZG**

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

1 Not included in the standard delivery.

4.3 Options

SF, DF: Precision filter for the analog input channels
SC Current inputs 0-20 mA or 4-20 mA for the SE analog inputs
DC Current inputs 0-20 mA or 4-20 mA for the diff. analog inputs (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 **XPCI-3001** is available in the following versions.

Version	Analog input channels
XPCI-3001-4	4 SE[1]
XPCI-3001-8	8 SE / 4 Diff.[2]
XPCI-3001-16	16 SE / 8 Diff.

4.5 Limit values

Max. altitude: ..... 2000 m
Operating temperature: ..... 0 to 60°C
Storage temperature: ..... -25 to 70°C
Relative humidity at indoor installation
50% at +40 °C
80% at +31 °C

Minimum PC requirements (APCI-3001):
PCI BIOS from Version 1.0
Bus speed: ..... < 33 MHz
Operating system: ..... Windows NT, 98, 2000, XP

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, 9x, 2000, XP

[1] SE for Single-Ended
[2] diff. for differential

Energy requirements:

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

	XPCI-3001
+ 5 V from 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 [1] & EOS [2]
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: ..... ±40 V

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): ..... $10^{12}\,\Omega$ // 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: ..... ±40 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.

Timer:

Counter depth of Timer 0 and 1: 16-bit

Counter depth of Timer 2: 24-bit

Timer 0 (A/D conversion): ..... 10 µs to 32.736 ms in 0.5 µs step

Timer 1 (delay time): ..... 100 µs to 3.27685 s in 50 µs step

Timer 2 (free programmable): ..... 100 µs to 838.8608 s in 50 µs step

Digital input channels:

Number: 4
Input current at 24 V: ..... 3 mA typ.
Input voltage range: ..... 0-30 V
Max. transfer rate: ..... 5 kHz
Optical isolation: ..... 1000 VAC
Logic „0“ level: ..... 0-5 V
Logic „1“ level ..... 12-30 V

Digital output channels:

Number: 4
Max. switch current: ..... 10 mA typ.
Voltage range: ..... 5-30 V
Max. transfer rate: ..... 5 kHz
Optical isolation: ..... 1000 VAC
Type: ..... Open Collector

4.6 Component scheme

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

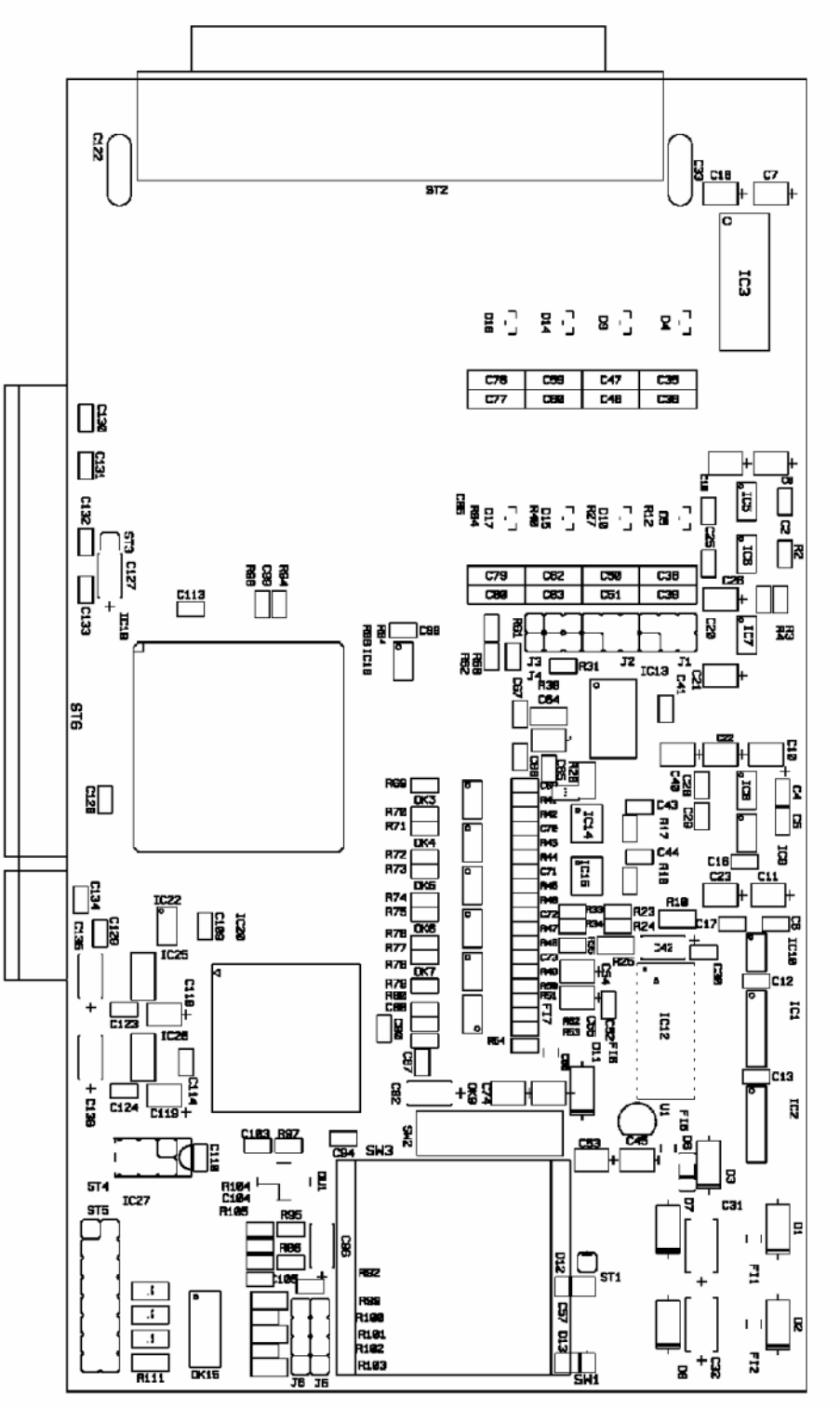
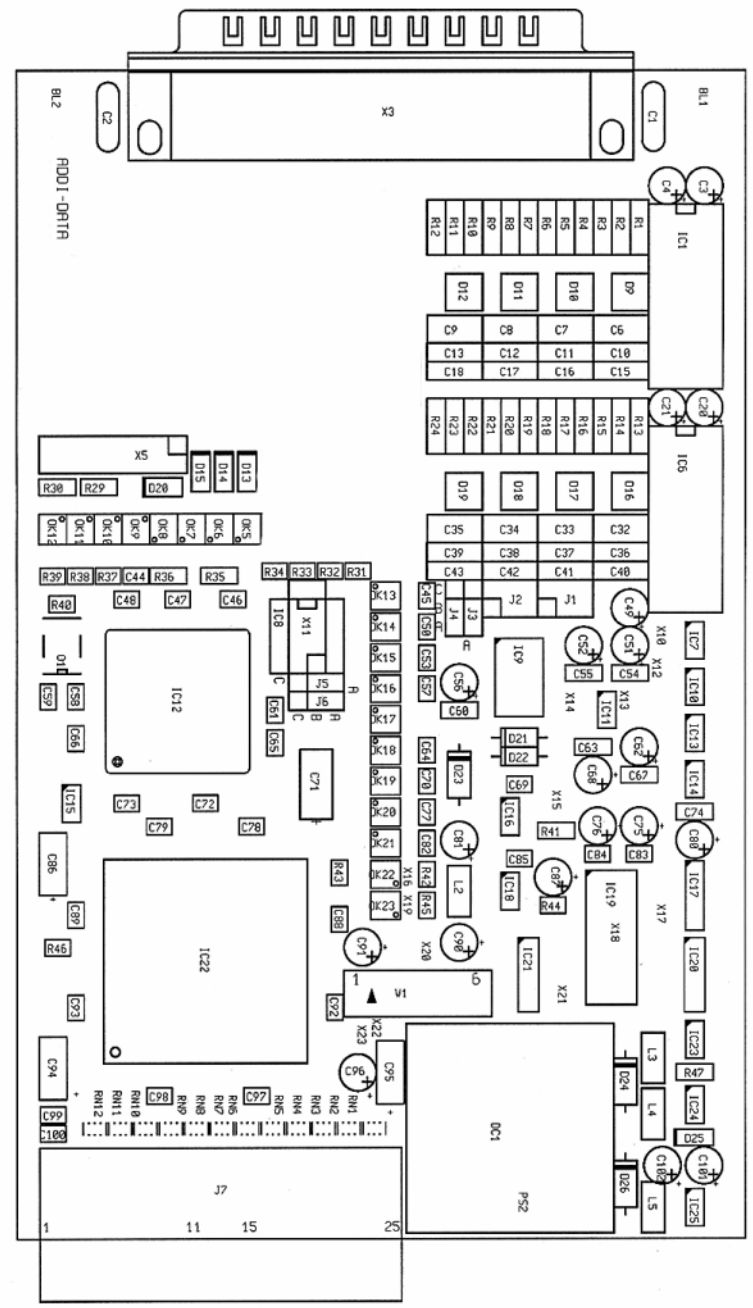


Fig. 4-2: Component scheme of the CPCI-3001



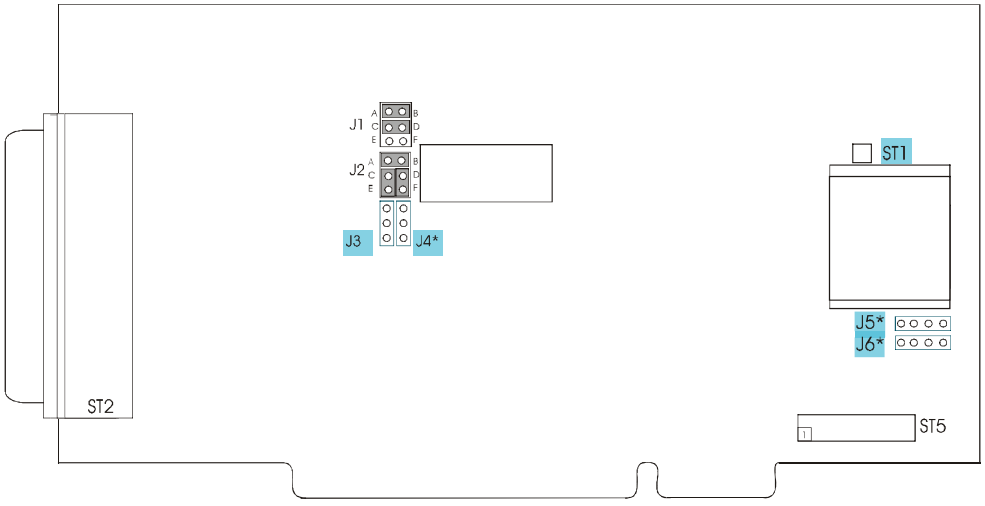
5 SETTING THE BOARD

IMPORTANT!
Read carefully the safety precautions!

5.1 Settings at delivery

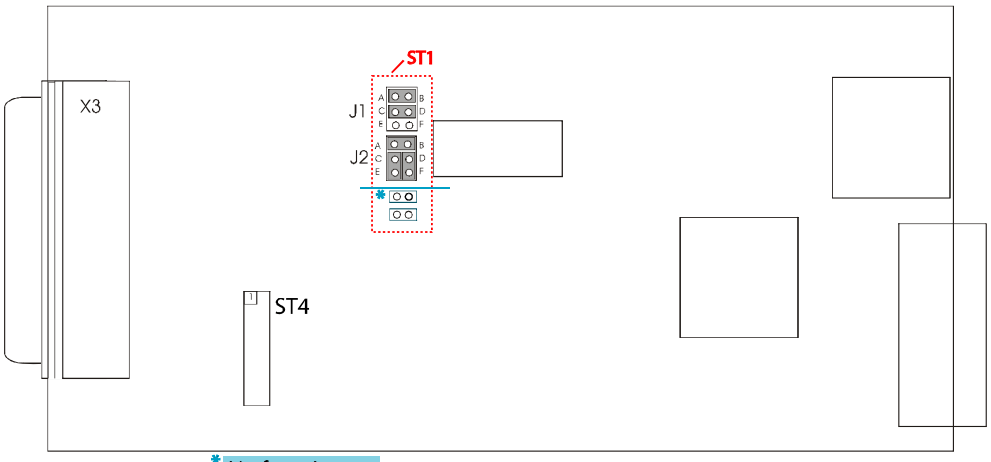
5.1.1 Jumper location (standard delivery)

Fig. 5-1: Jumper location on the APCI-3001 (settings at standard delivery)



*J3, J4, J5, J6, ST1: No function
Don't attach jumpers because not occupied!

Fig. 5-2: Jumper location on the CPCI-3001 (settings at standard delivery)



* No function:
Don't attach jumpers because not occupied !

5.2 Setting the input channels

Fig. 5-3: Setting the channels in single-ended mode (Settings at delivery)

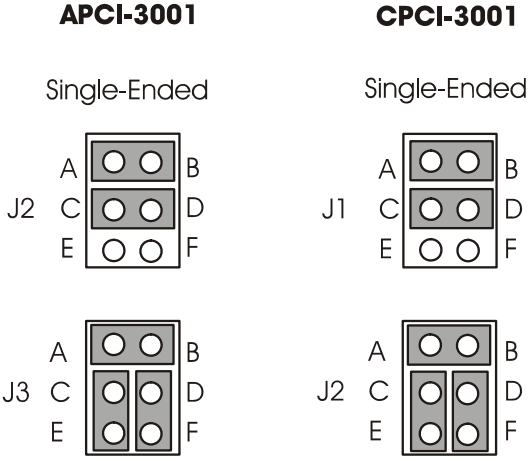
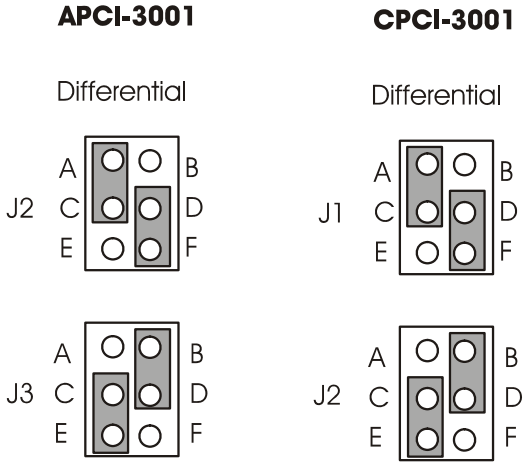


Fig. 5-4: Setting the channels in differential mode



6 INSTALLATION OF THE BOARD

IMPORTANT!
Do observe the safety precautions (yellow leaflet)!

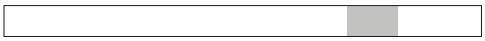
- **Switch off your PC/CompactPCI/PXI computer and all the units connected to it.**
- **Pull the PC/CompactPCI/PXI computer mains plug from the socket.**
- **Open your PC/CompactPCI/PXI computer as described in the manual of the PC manufacturer.**

6.1 Installation of an APCI-3001 plug-in board

6.1.1 Selecting a free slot

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

Fig. 6-1: PCI-5V slot (32-bit)



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

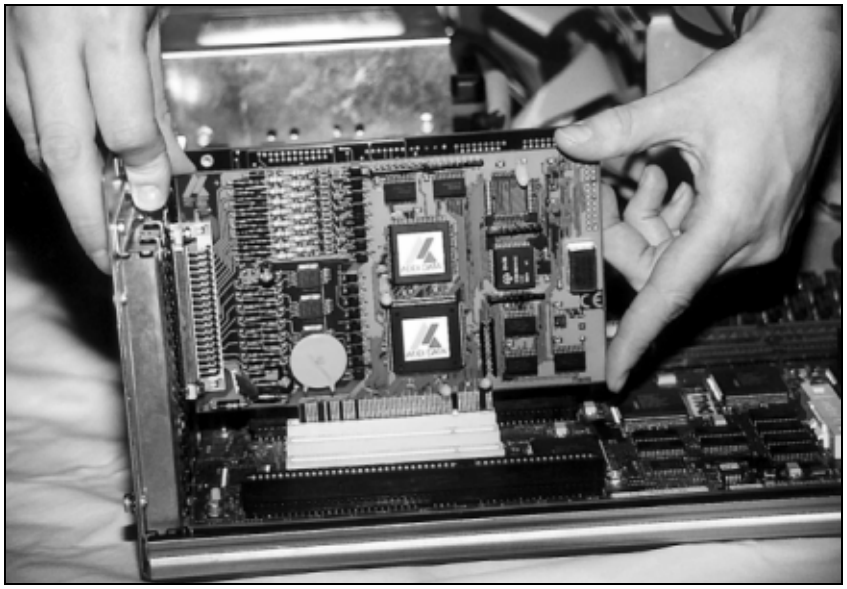
Discharge yourself from electrostatic charges.

Take the board out of its protective pack.

6.1.2 Plugging the board into the slot

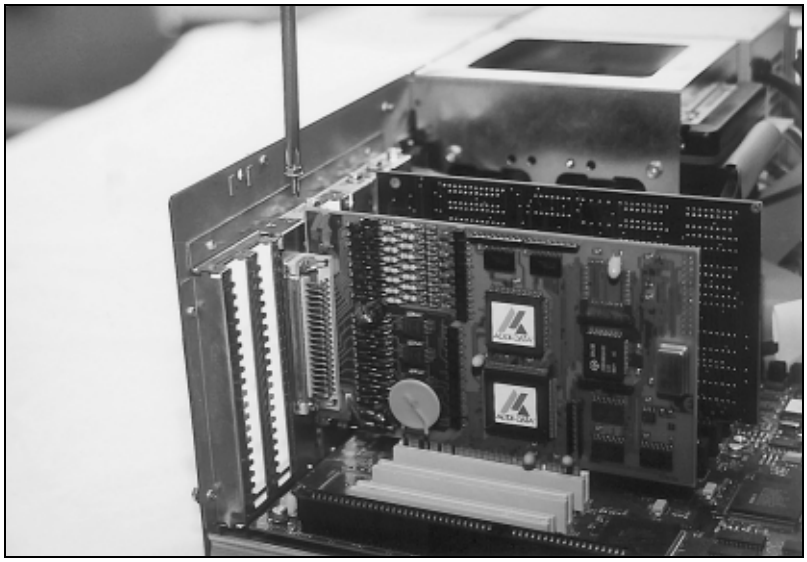
♦ Insert the board **vertically into the chosen slot.**

Fig. 6-2: Inserting the board



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

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



♦ **Tighten all the loosen screws.**

6.1.3 Closing the PC

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

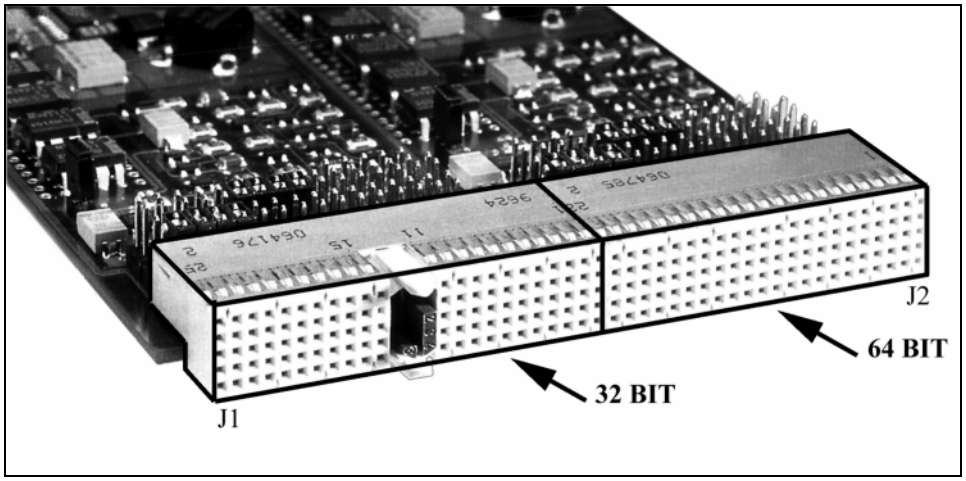
6.2 Installation of a CPCI-3001

6.2.1 Selecting a free slot

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

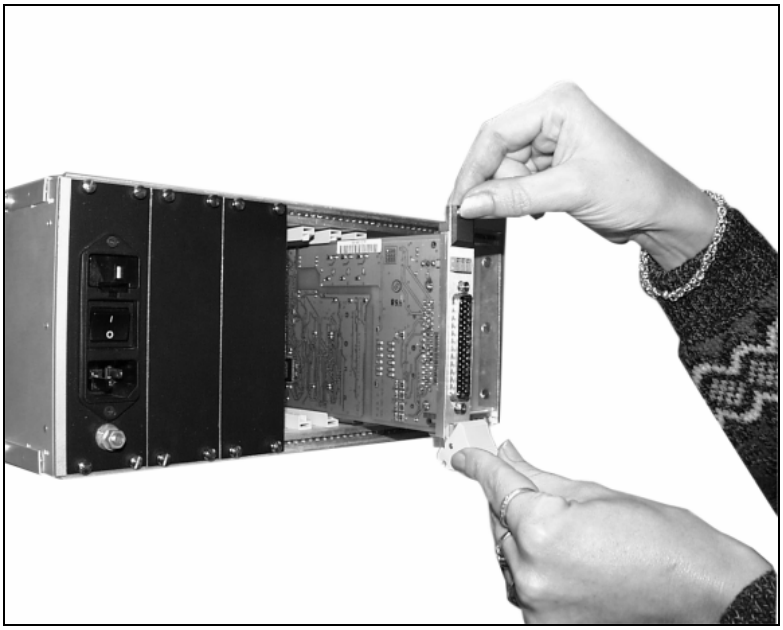
See in the computer manual which types of slots are free.

Fig. 6-4: Types of slots for CompactPCI boards



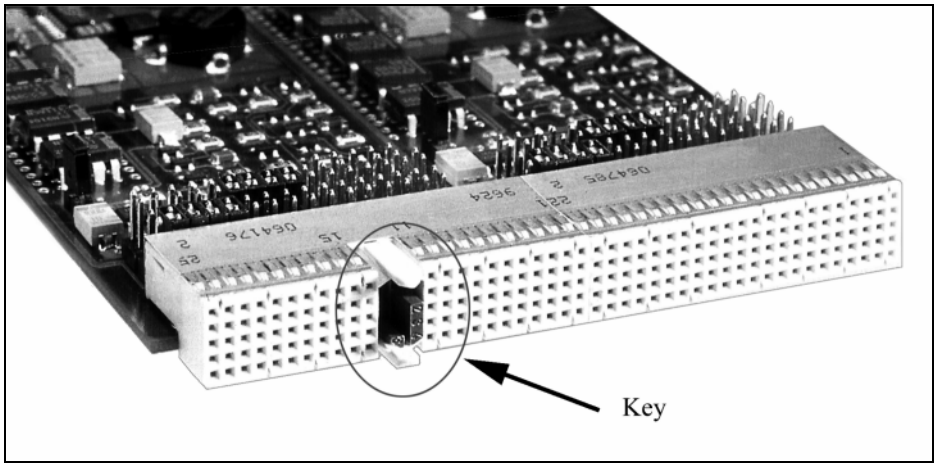
- **Discharge yourself from electrostatic charges.**
- **Hold the board at its grip (See handling of the board in chapter 3).**
- **Insert the board into the guiding rails and push it to the back cover of the rack. In order to fully insert the board, a small resistance has to be overcome.**

Fig. 6-5: Pushing a CPCI board into a rack



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

Fig. 6-6: Connector keying



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

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

7 SOFTWARE

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

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

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

The board is supplied with a CD-ROM containing the ADDIREG program for Windows NT 4.0 and Windows XP/2000/98.

The ADDIREG registration program is a 32-bit program for Windows NT 4.0 and Windows XP/2000/98. The user can register all hardware information necessary to operate the ADDI-DATA PC boards.

7.1 Program description

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

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

Start ADDIREG under Start/Programme/ADDIPACK/ADDIREG.

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

Fig. 7-1: ADDIREG registration program (example)

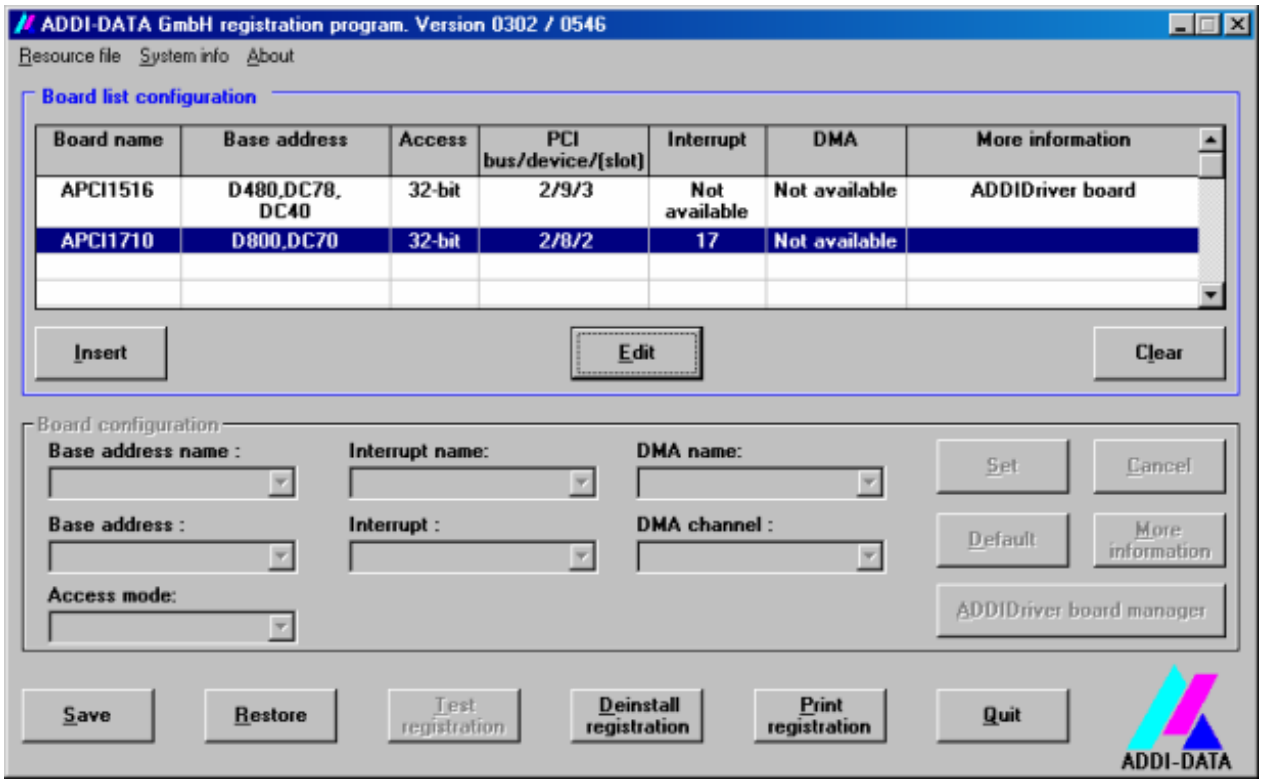


Table:

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

Board name:
Name of the different registered boards will be displayed (e.g. APCI-1710).
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.

Textboxes

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

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

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

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.

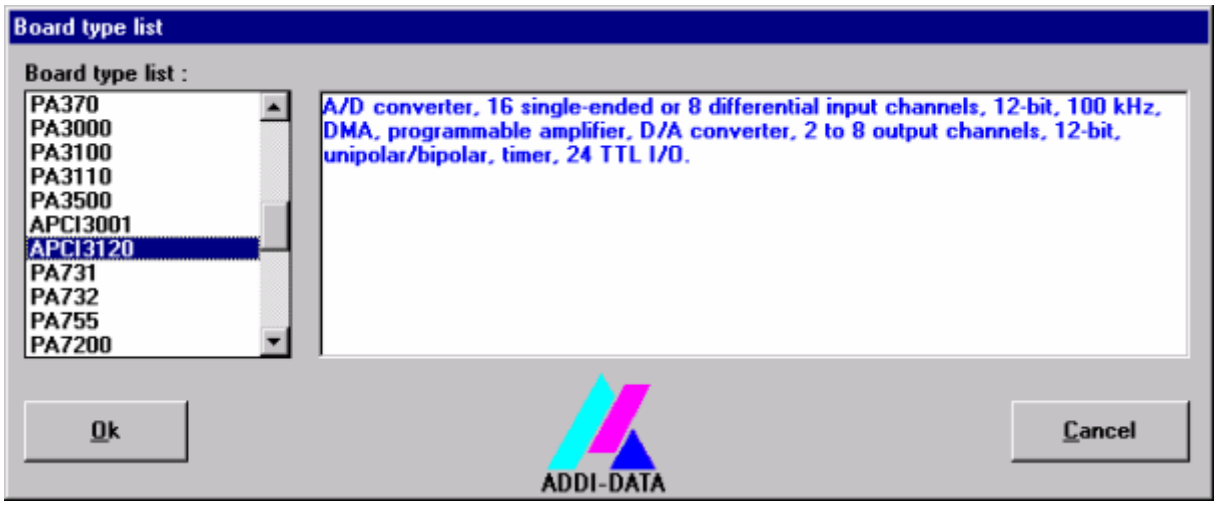
DMA channel (ISA boards only):
Selection of the used the DMA channel.

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. 7-2: 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 confilict 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.

ADDIDriver Board Manager (only for the boards with ADDIPACK):
Under Edit/ADDIDriver Board Manager you can check or change the current settings of the board set through the ADDEVICE Manager. ADDevice Manager starts and displays a list of all resources available for the virtual board.

More information (not available for the boards with ADDIPACK)
You can change the board specific parameters like the identifier string, the COM number, the operating mode of a communication board, etc...

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

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

7.1.1 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.

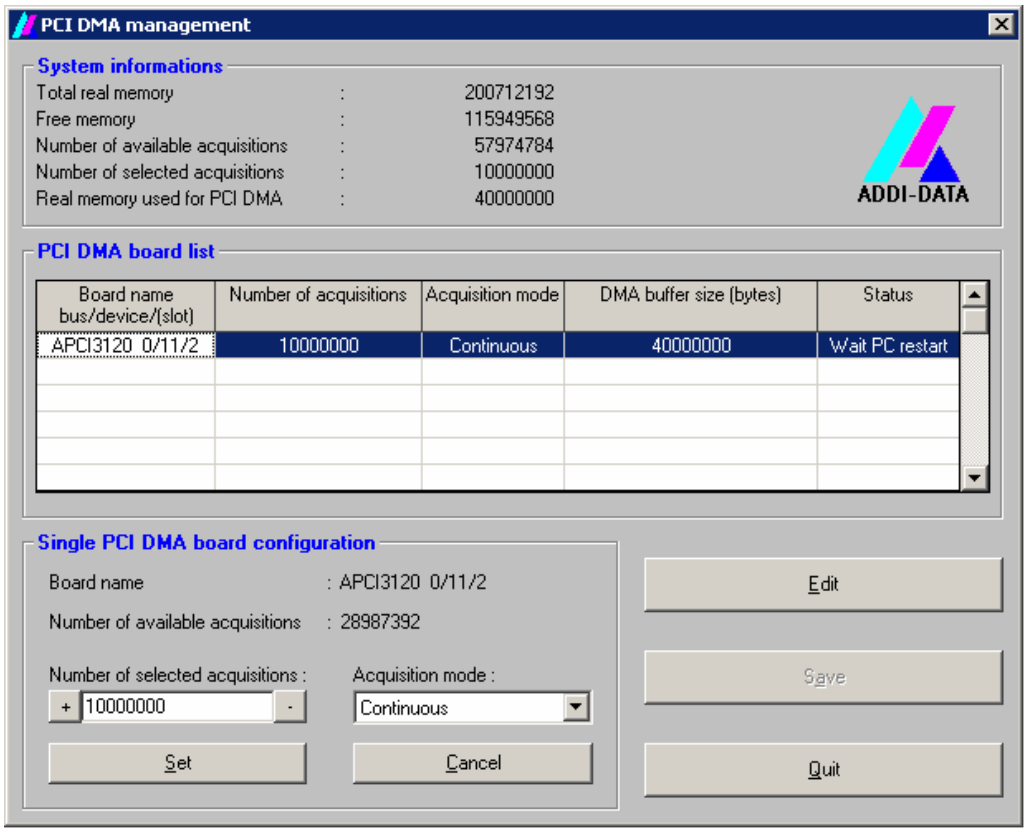
7.1.2 PCI analog input boards with DMA

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. 7-3: PCI DMA management (Example)



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:
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).

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

7.2 Registering a new board

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 Fig. 7-1 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 configurated 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.

7.3 Changing the registration of an existing board

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

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

7.4 Questions and software downloads on the web

Do not hesitate to e-mail us your questions.
per e-mail: info@addi-data.de or
hotline@addi-data.de

Free downloads of standard software

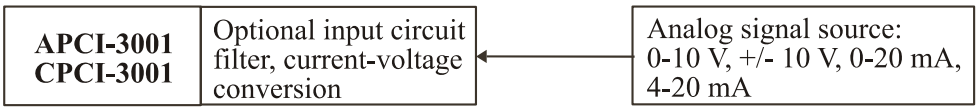
You can download the latest version of the software for the **XPCI-3001.**

http://www.addi-data.com

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

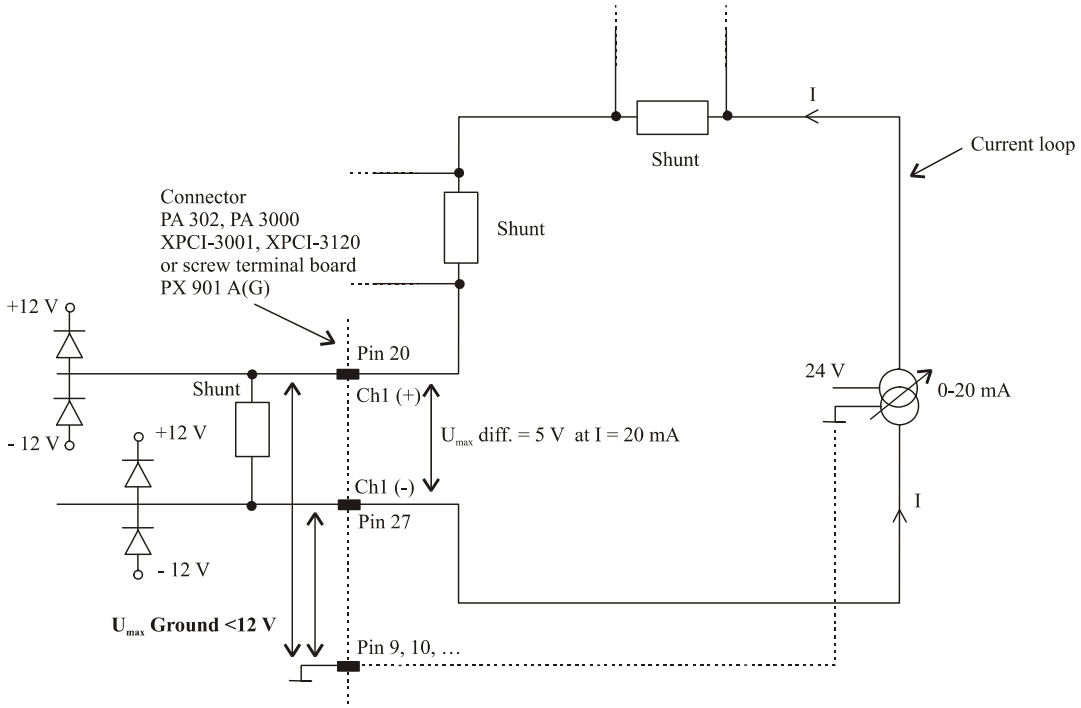
8 CONNECTING THE PERIPHERAL

8.1 Connection principle



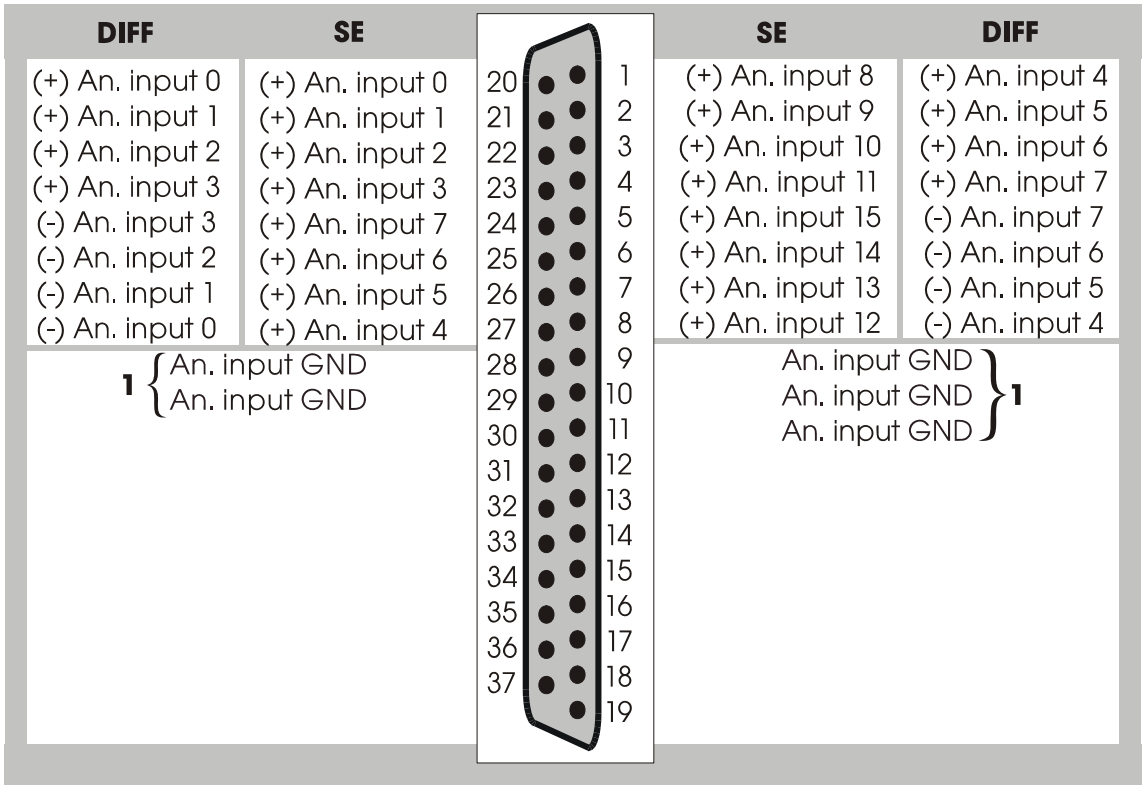
At option DC the board is to be inserted at the end of the current loop so that the voltage (**U_{max} Ground**) amounts maximum 12 V between the differential input pin and the ground.

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



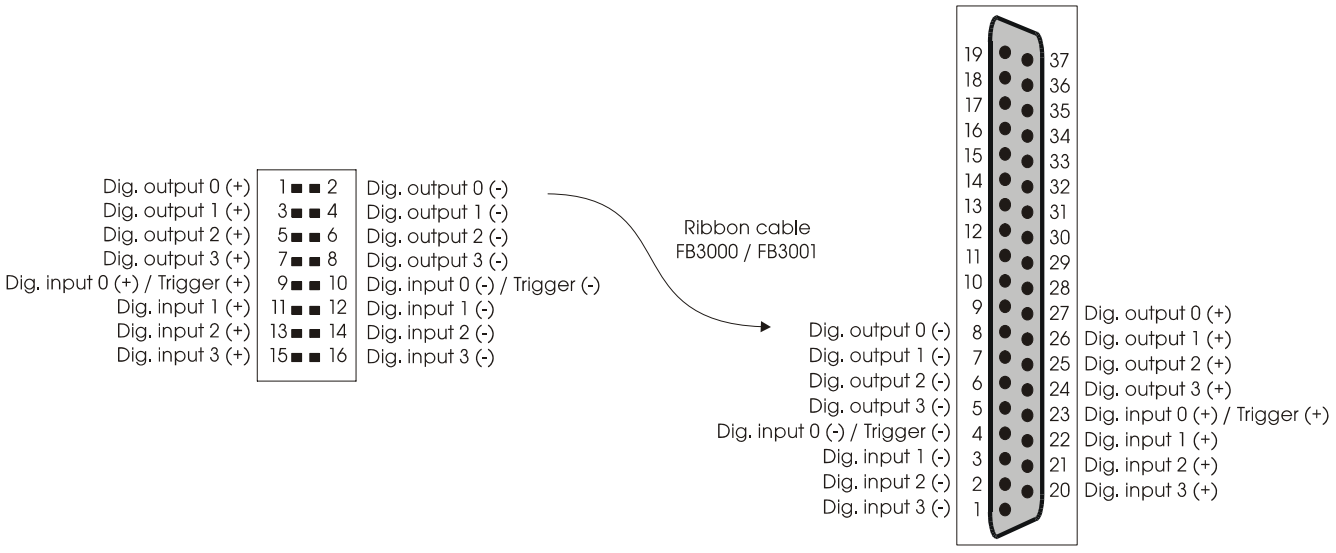
8.2 Connector pin assignment

Fig. 8-2: 37-pin SUB-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



IMPORTANT!
Insert the FB3000/FB3001 on the connector with the red cable lead on the side of the pin 1. See 5.1.1 "Jumper location on the board".

8.3 Connection examples

8.3.1 Analog input channels

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

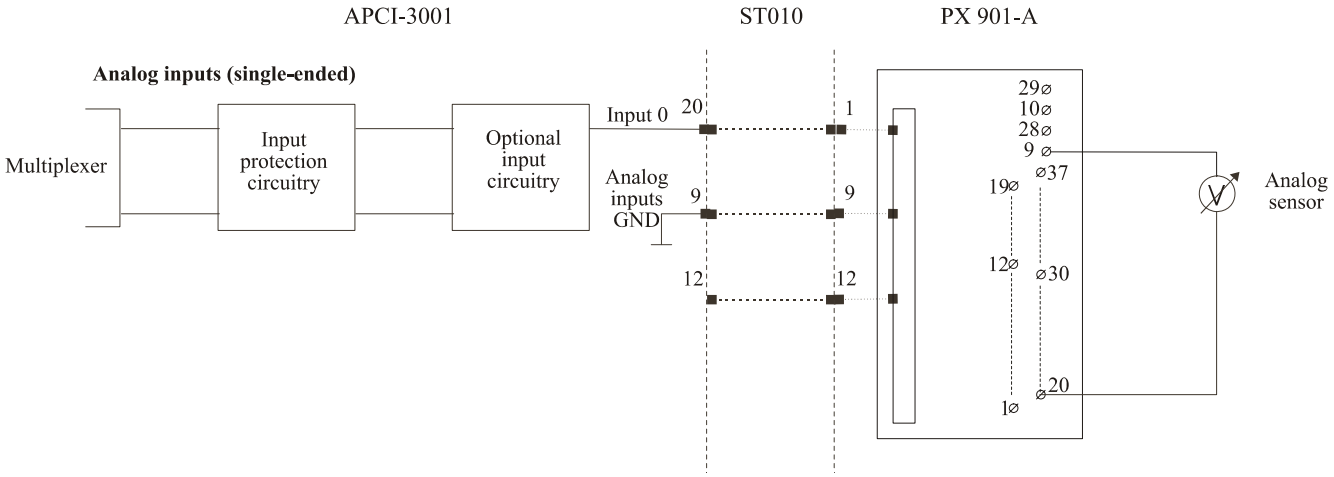
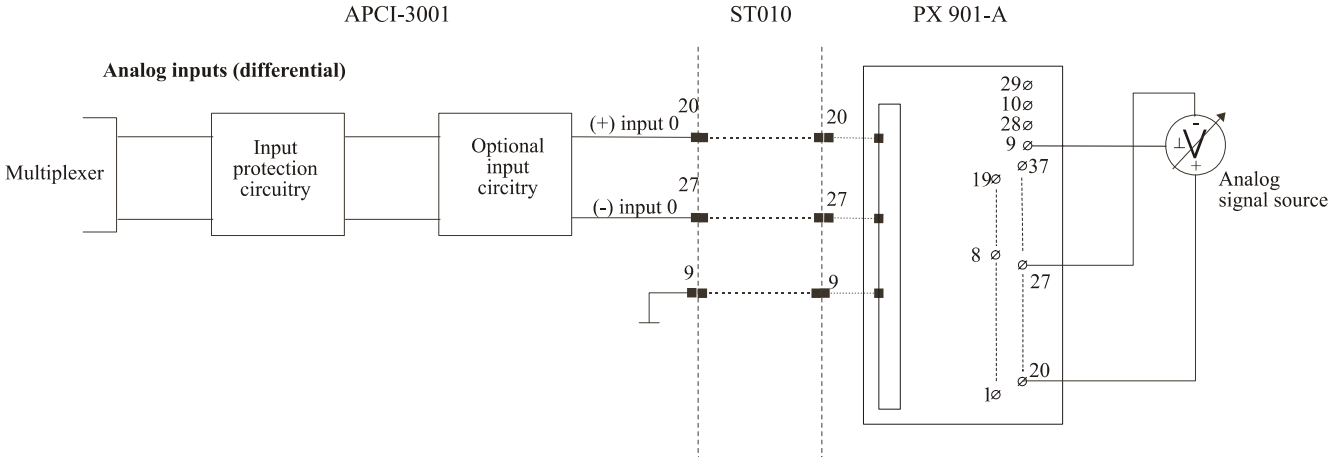


Fig. 8-5: Analog input channels (diff.)



8.3.2 Digital inputs and outputs

Fig. 8-6: Digital outputs

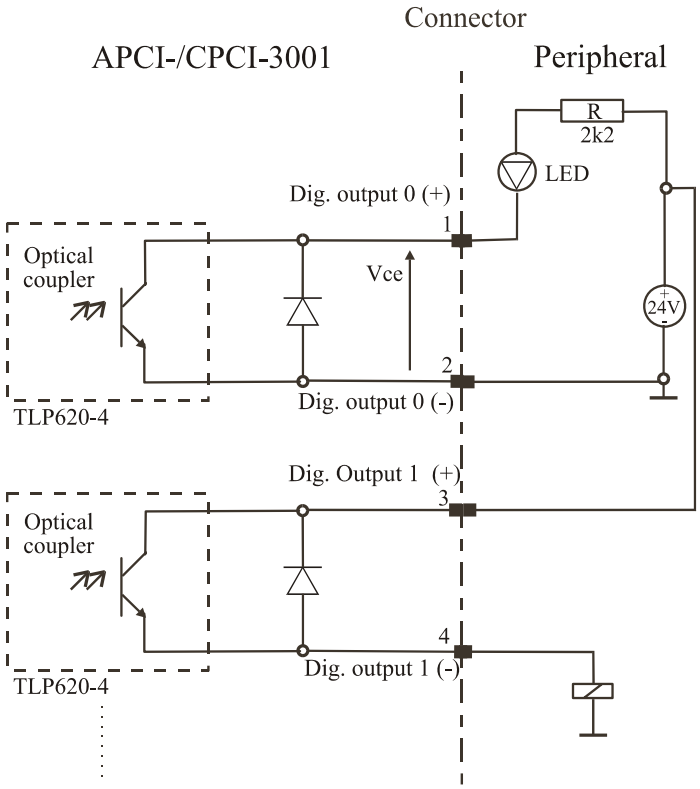
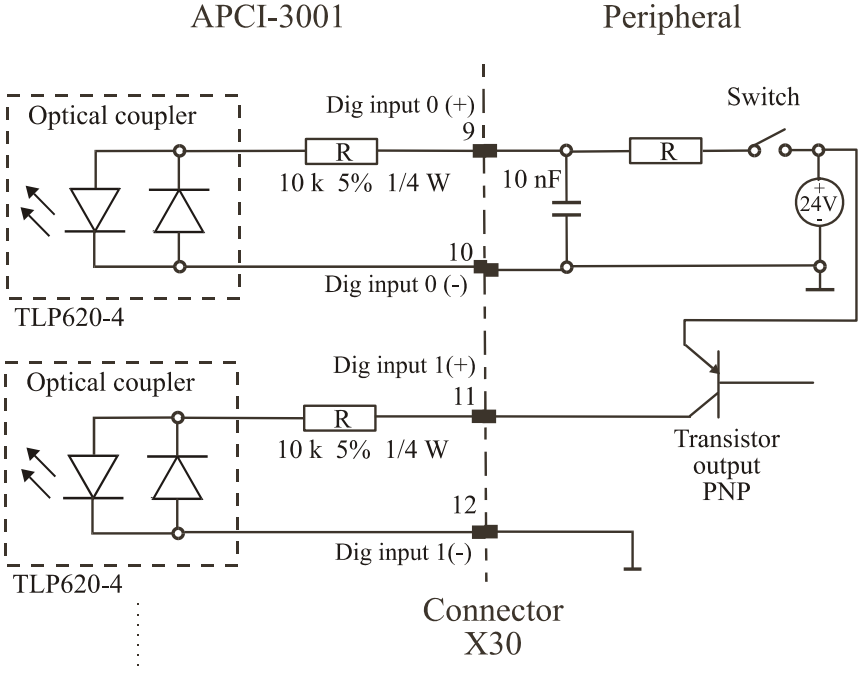
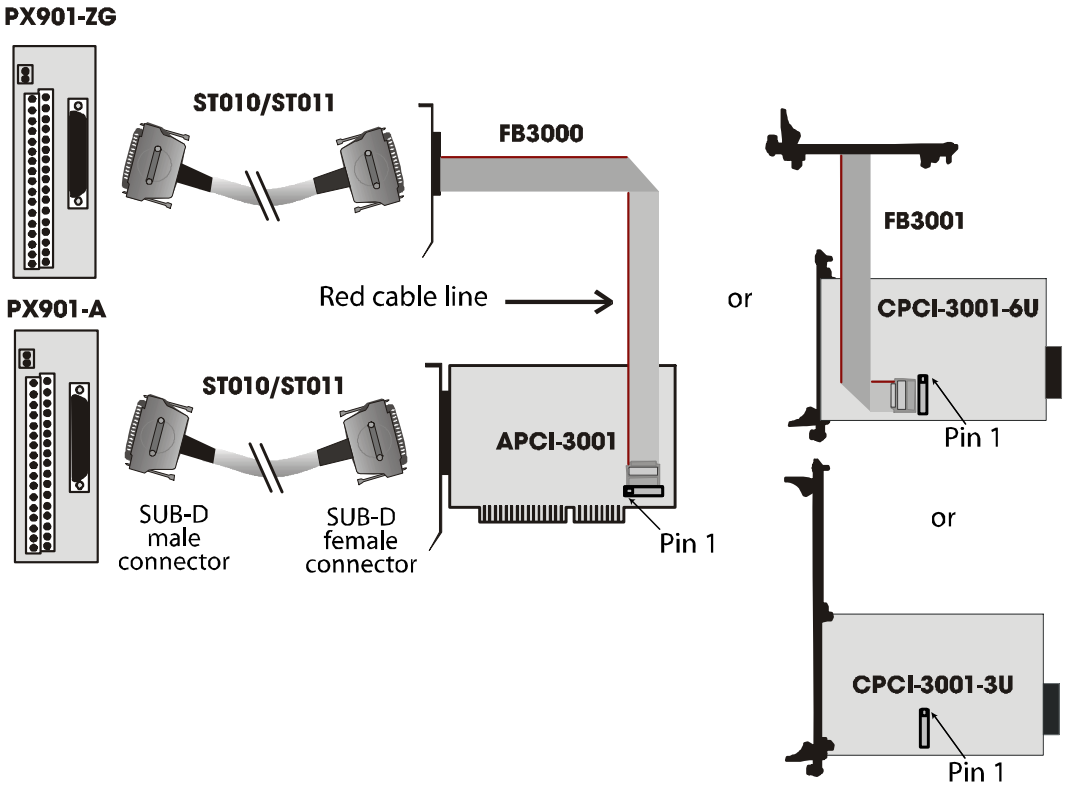


Fig. 8-7: Digital inputs



8.3.3 Connection to screw terminal panels

Fig. 8-8: Connection to the PX901 screw terminal panel

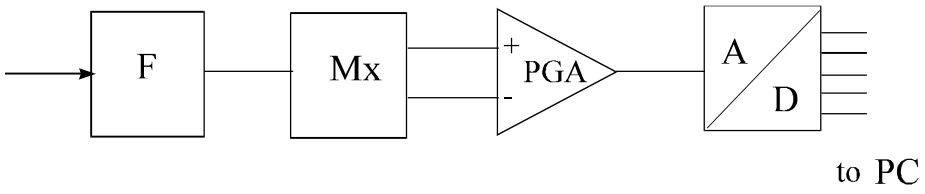


IMPORTANT!
Insert the **FB3000/FB3001** on the connector with the red cable lead on the side of the pin 1. See also Fig. 5-1 and Fig. 5-2 (“Jumper location”).

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).



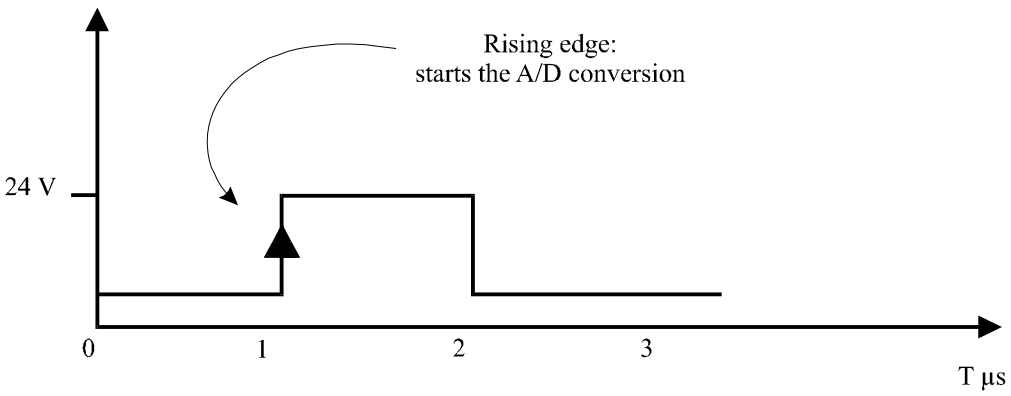
After reaching the multiplexer via a filter (RC module), the signals are led through a programmable gain amplifier to the 16-bit A/D converter.

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,
- the list can be cyclically handled through Timer0,
- the list can be handled during a defined time interval through Timer0 and 1.
- a defined number of conversions or scans is processed through Timer0, 1 and 2.
- 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 µs.
- 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 µs from 10 µs to 32767 µs. 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 µs). 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:

Rsource	ui_ConvertTiming ui_AcquisitionTiming
<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.

Rsource	ui_ConvertTiming ui_AcquisitionTiming
<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.*

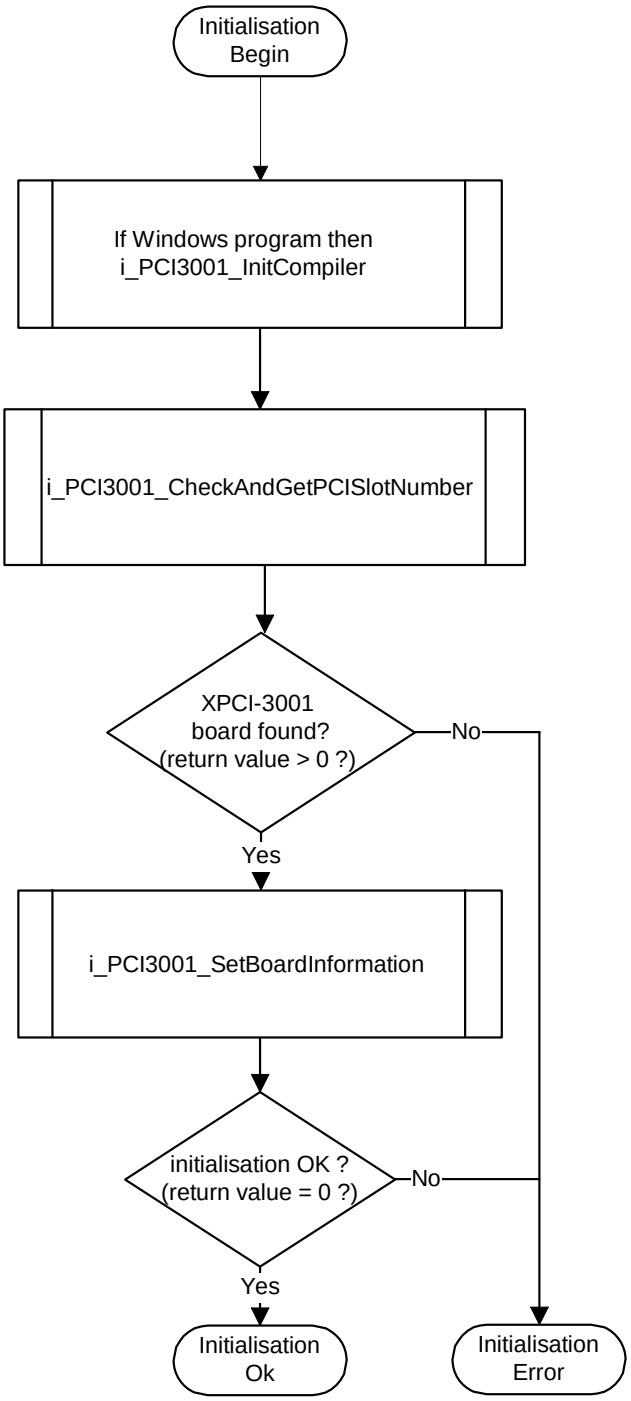
10 STANDARD SOFTWARE

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

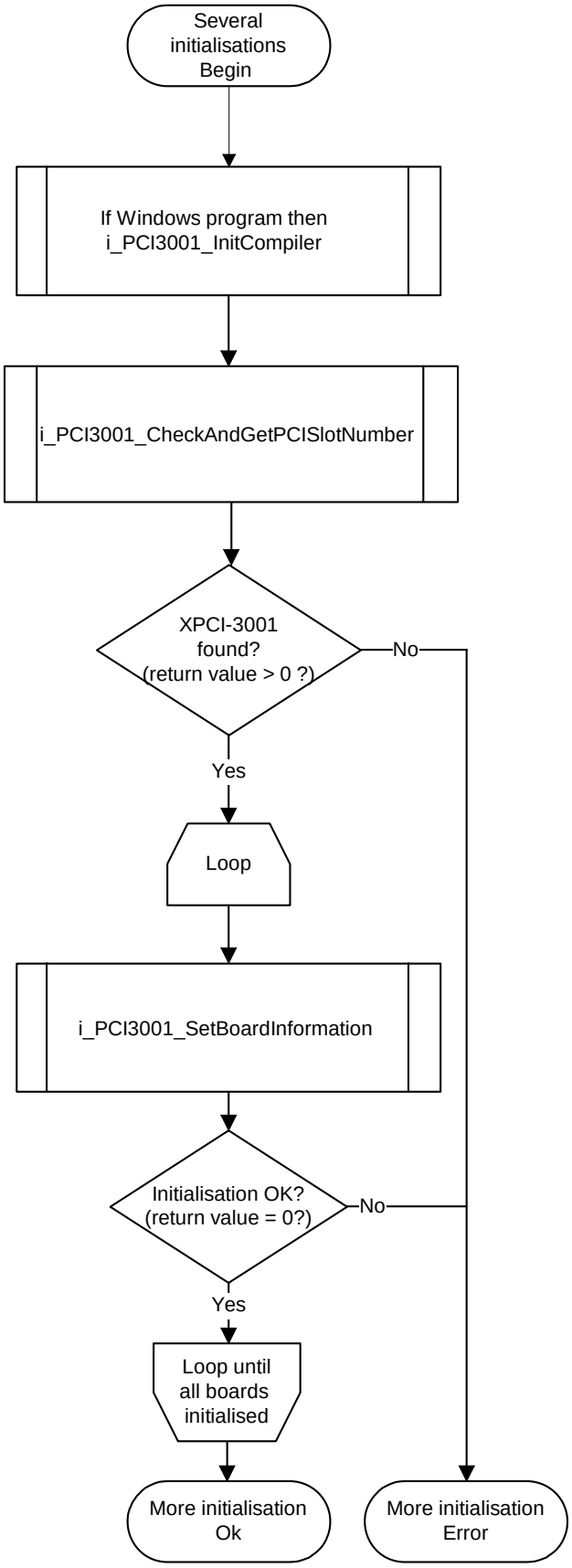


b) Example in C

```
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

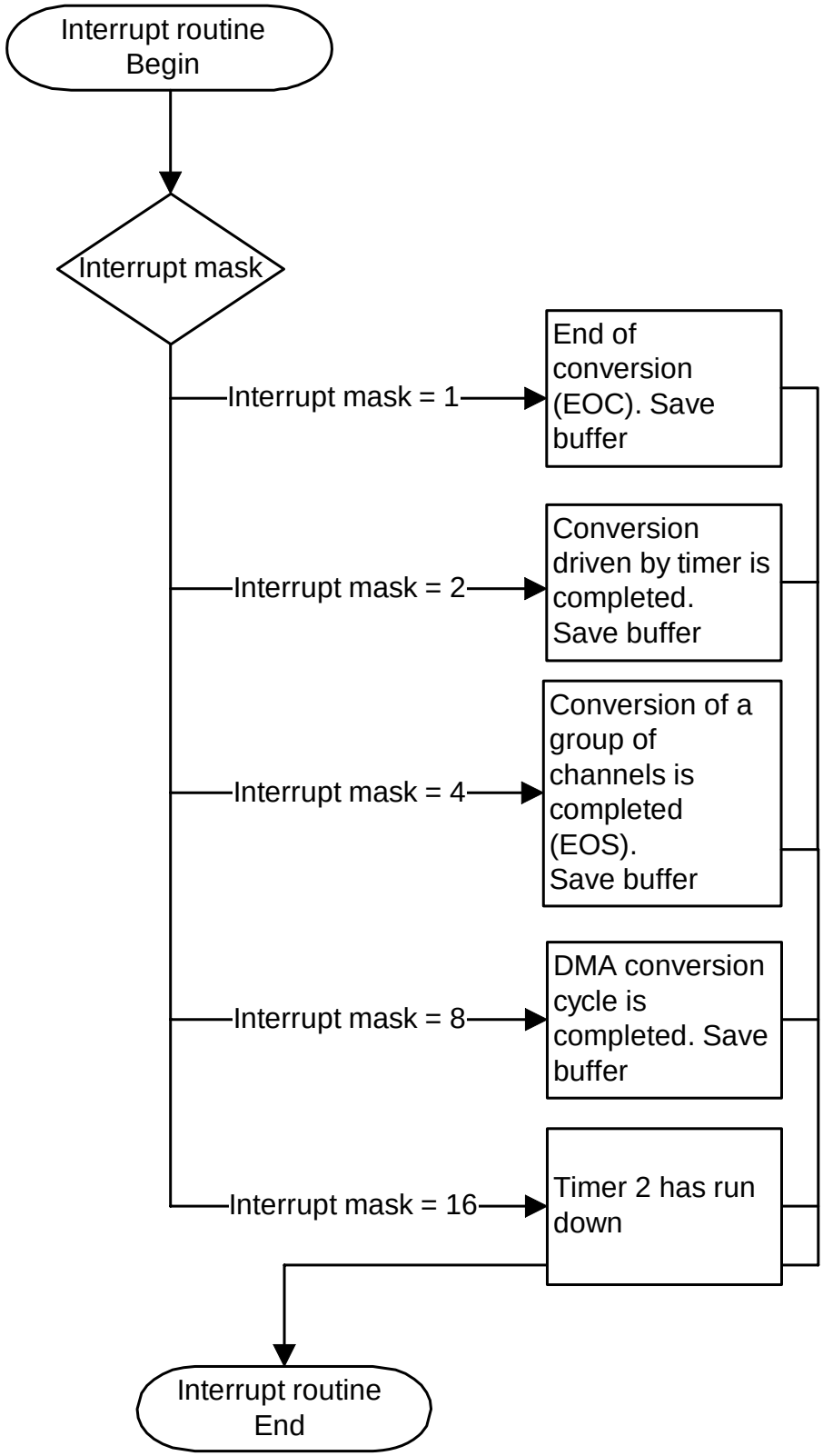


b) Example in C

```
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



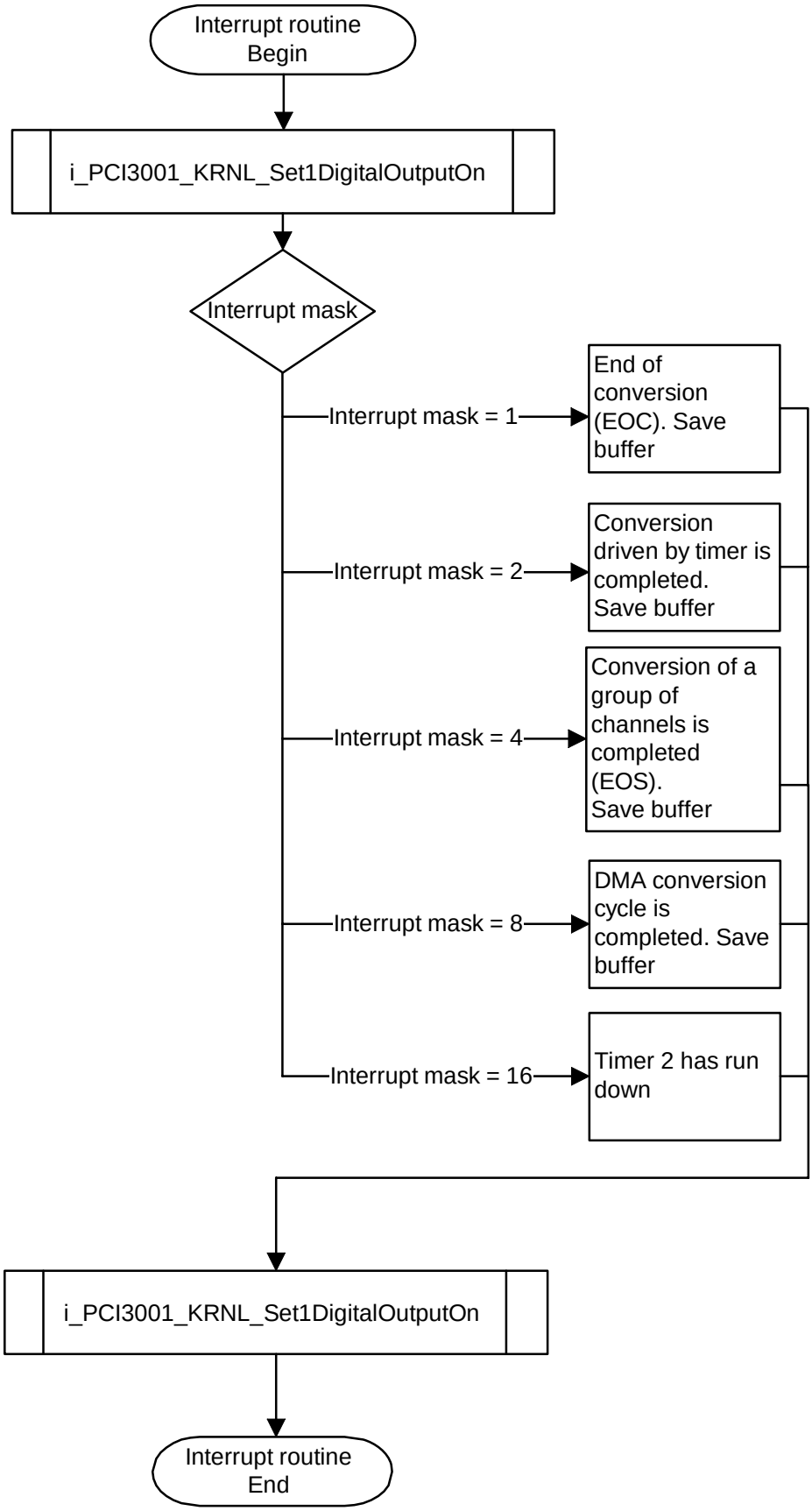
b) Example in C for DOS und Windows 3.1x

```
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)



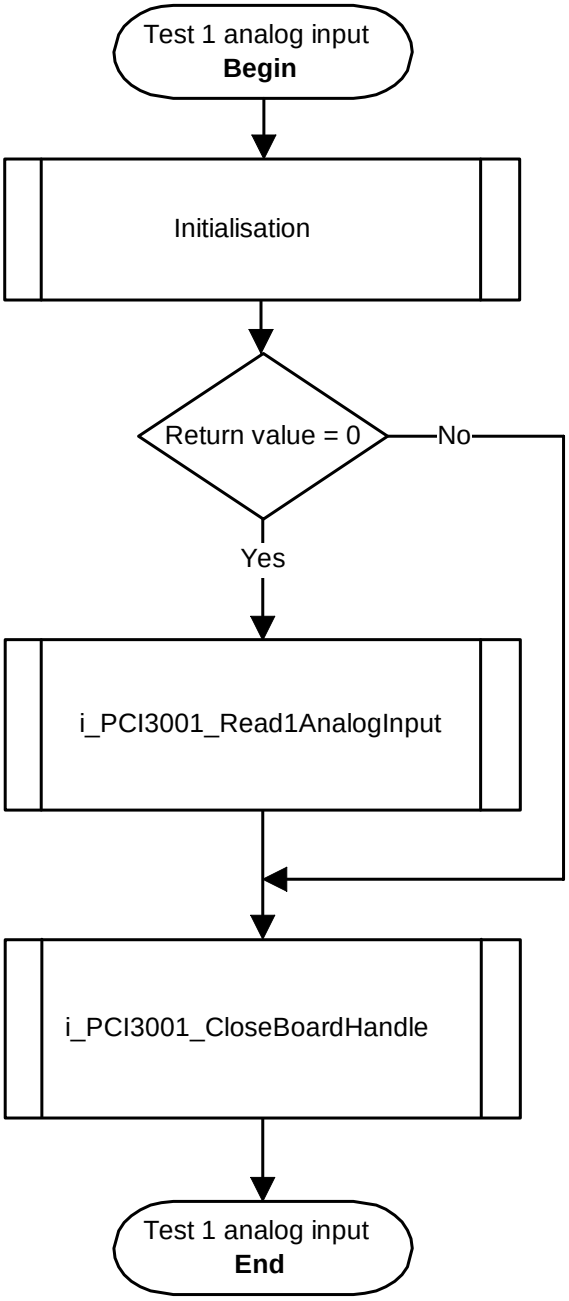
e) Example in C for Windows NT/95/98 (synchronous mode)

```
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

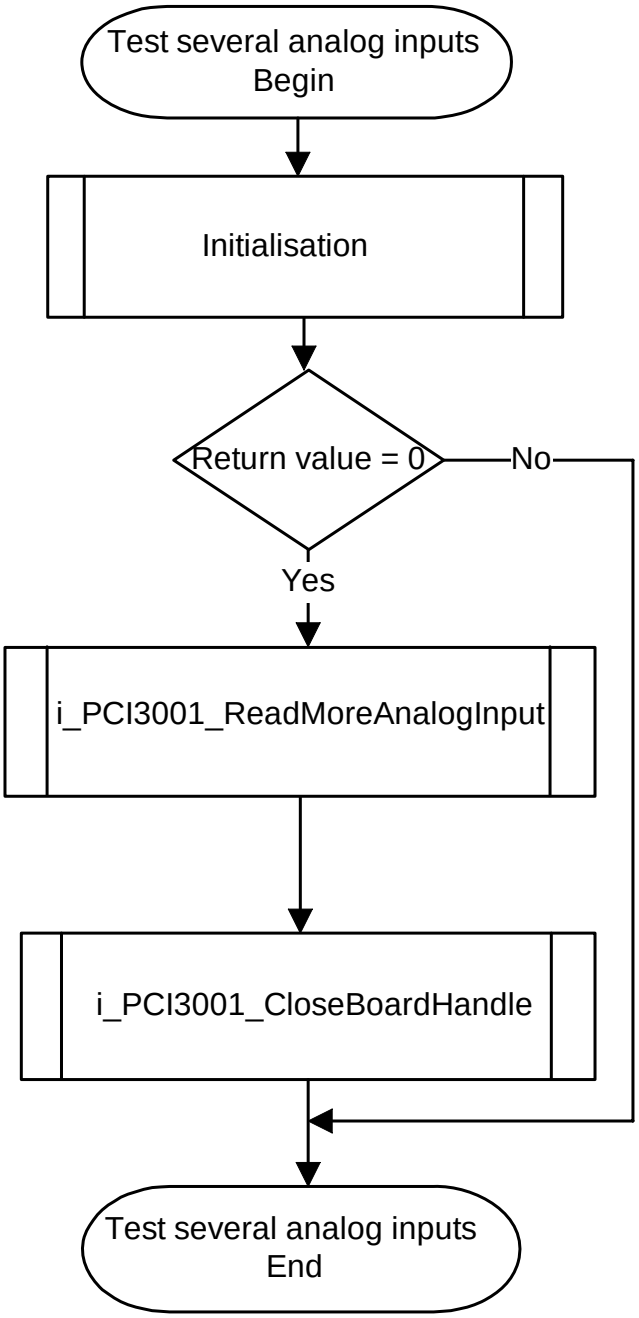


b) Example in C

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

     if (Initialisation (&b_BoardHandle) == 0)
        {
        if (i_PCI3001_Read1AnalogInput  (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



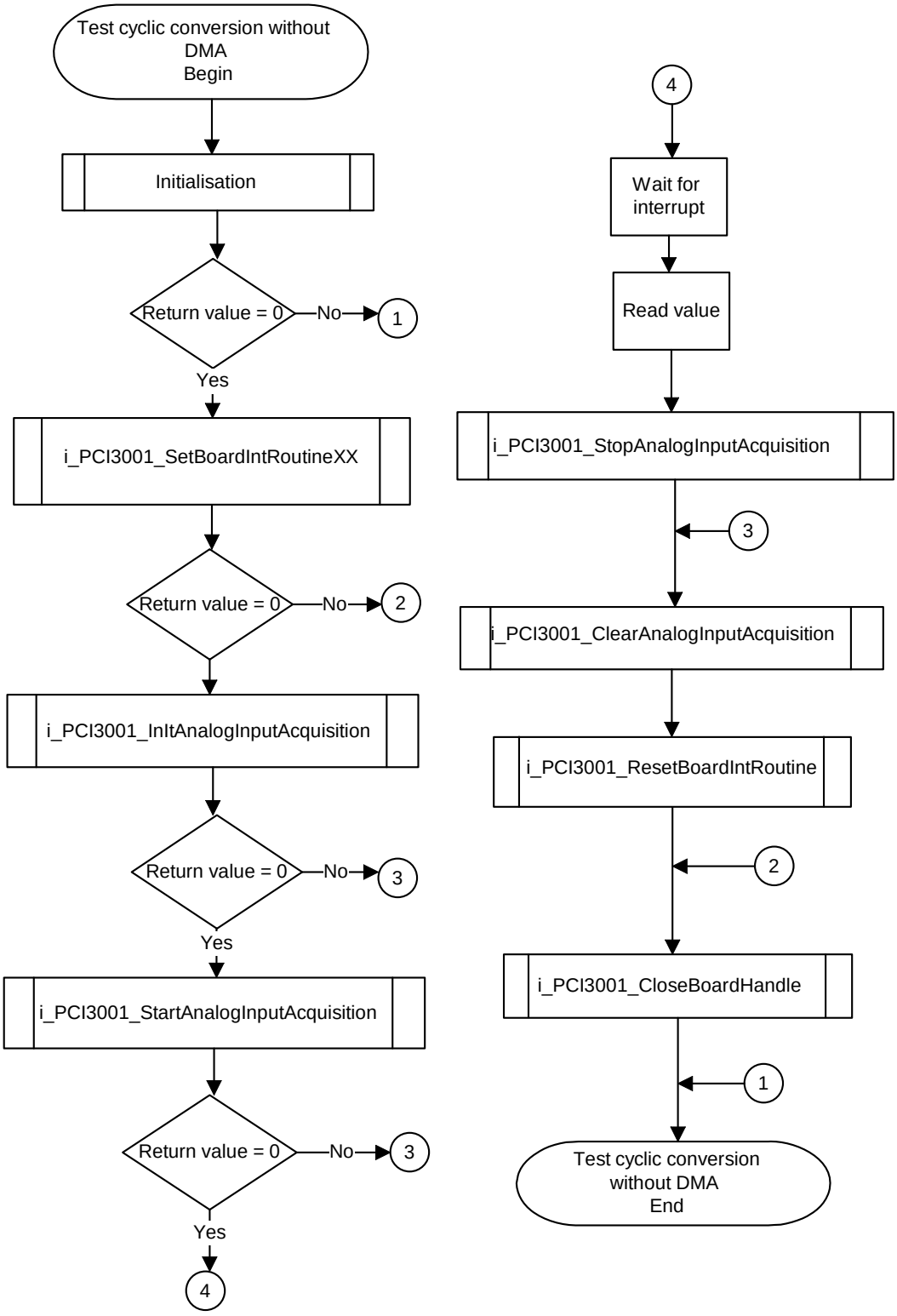
b) Example in C

```
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)

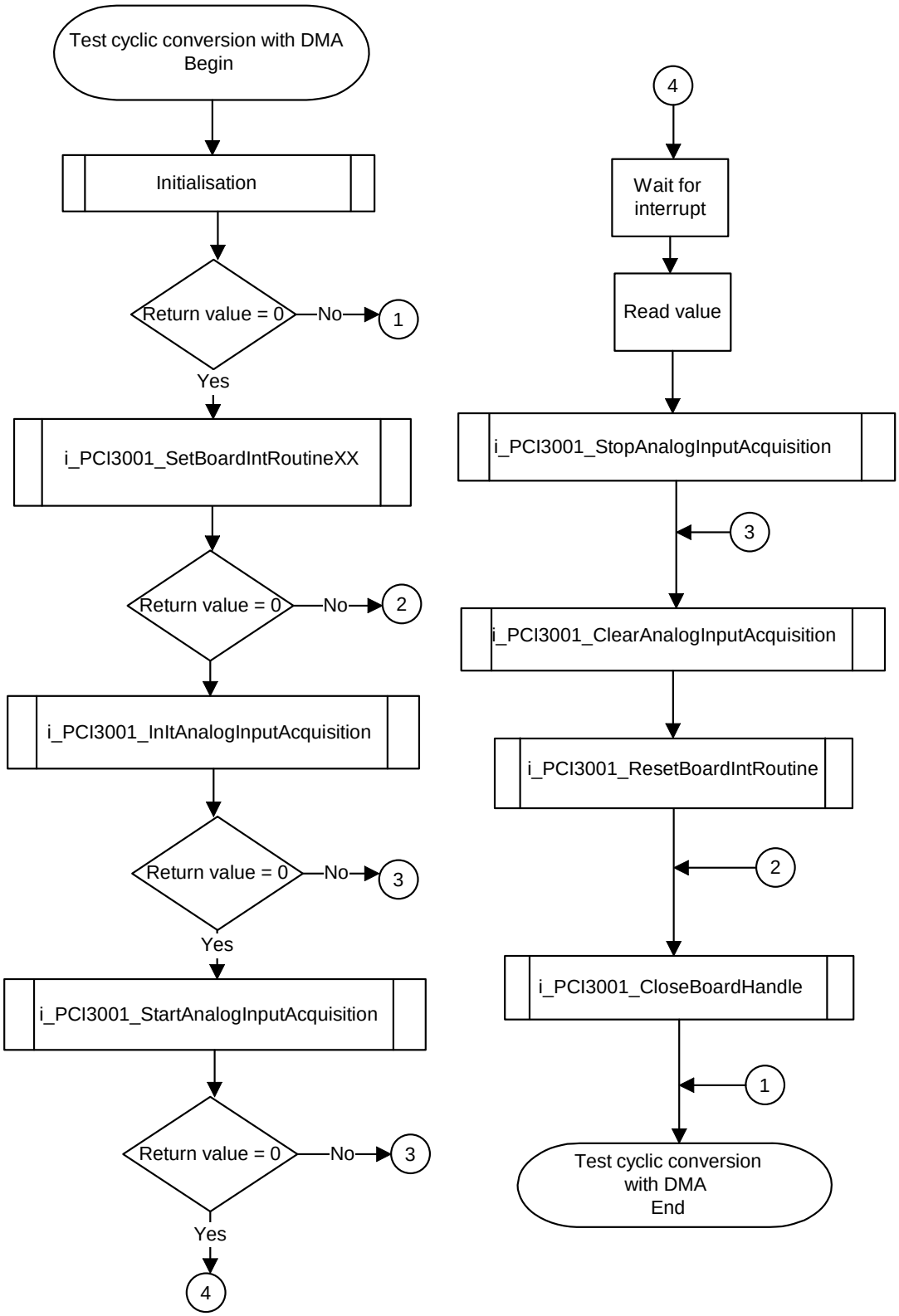
```
void main(void)
   {
   int i_Cpt;
   unsigned char b_Gain [4];
   unsigned char b_Polar [4];
   unsigned char b_Channel [4];
   unsigned char b_BoardHandle;
   if (Initialisation(&b_BoardHandle) == 0)
      {
     if (i_PCI3001_SetBoardIntRoutineWin32(b_BoardHandle,0,NULL,v_InterruptRoutine) == 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 %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");
   }
```

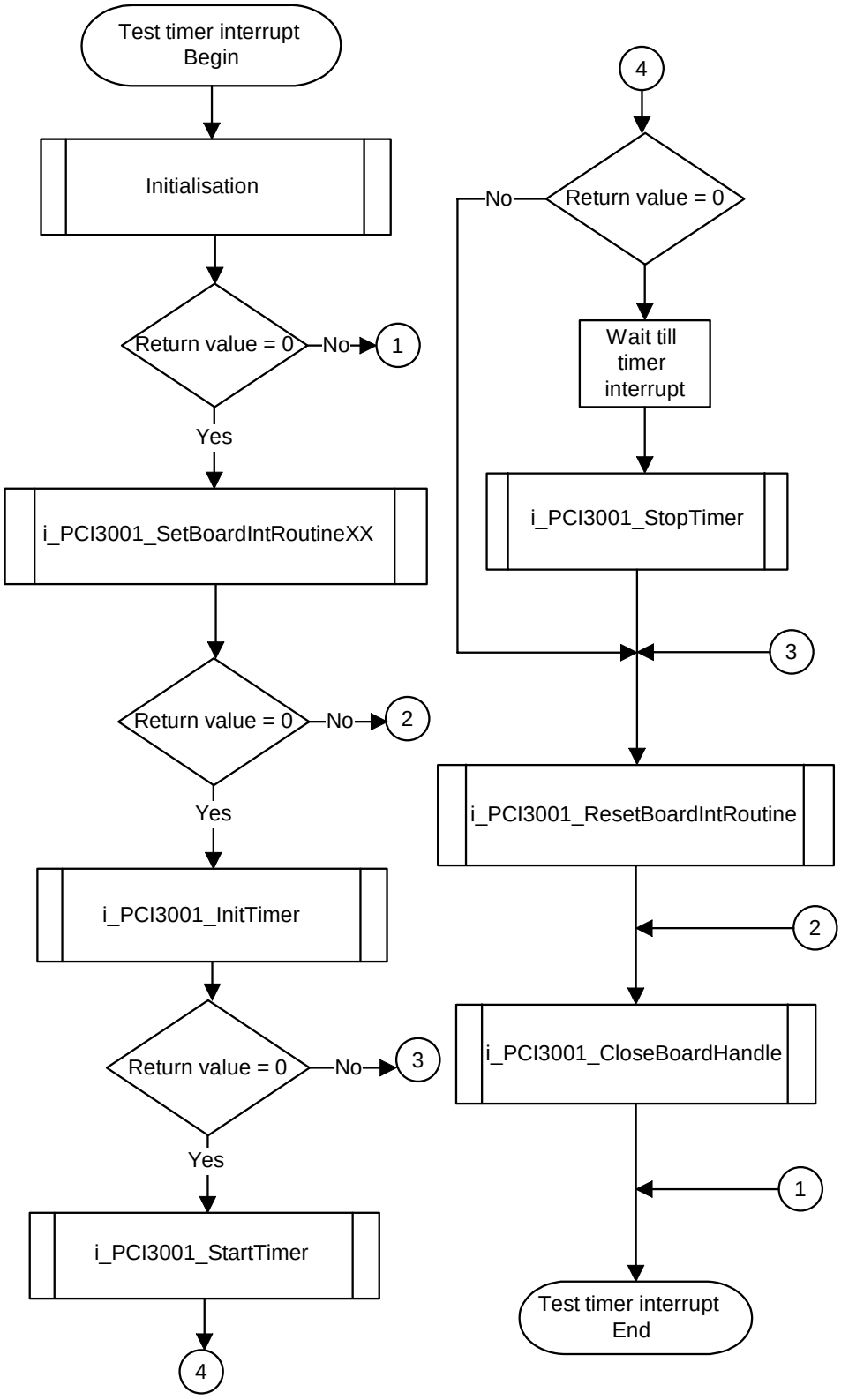
f) Example in C for Windows NT/95/98 (synchronous mode)

```
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 %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");
     }
```

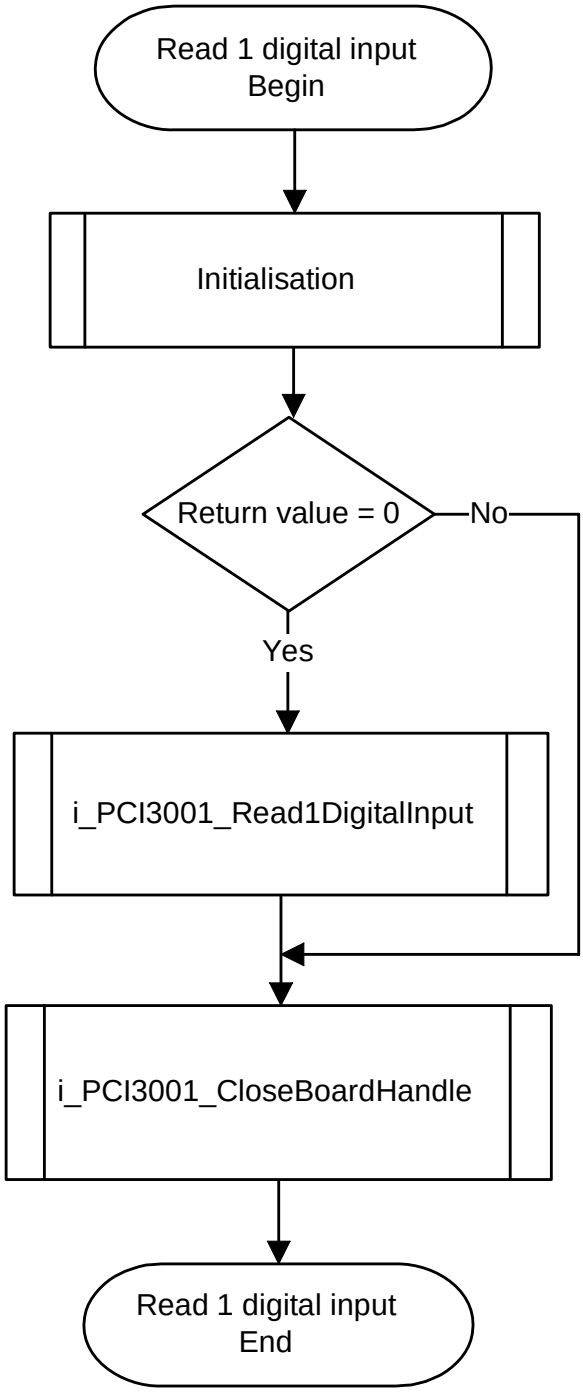
e) Example in C for Windows NT/95/98 (synchronous mode)

```
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.

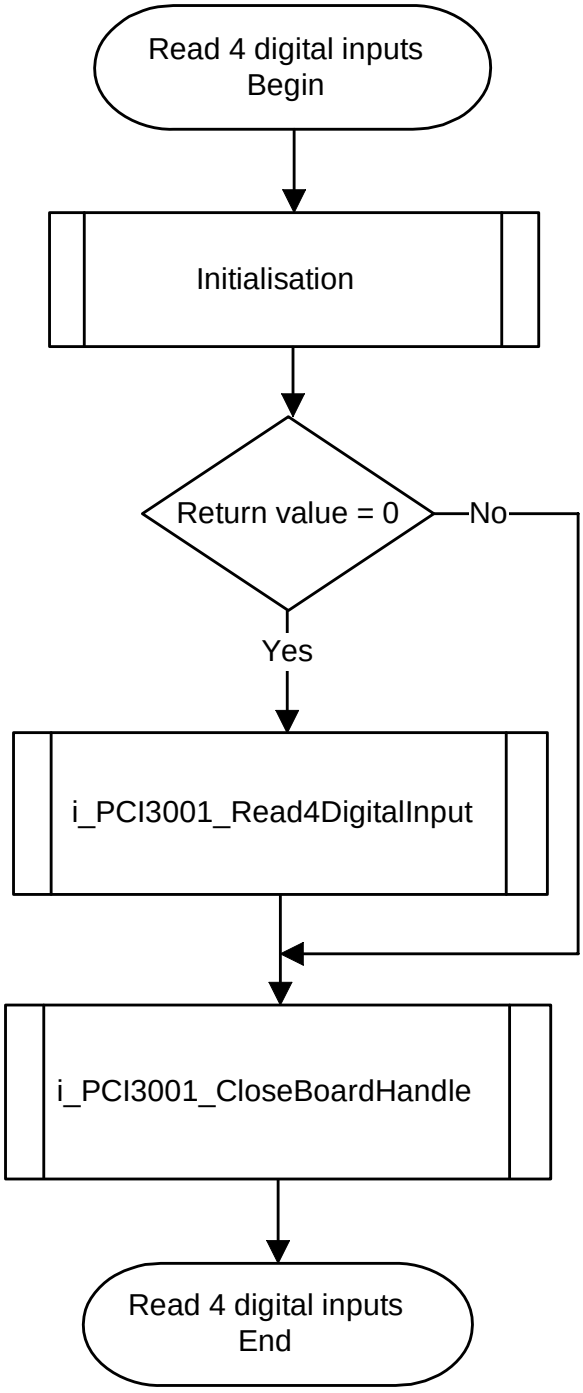
The test result is 1.

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



b) Pin assignment

Set all digital input channels to 24 V.
Pin assignment of the 37-pin SUD.D male connector: - at pins 4, 3, 2, 1
+ at pins 23, 22, 21, 20

The test result is 15.

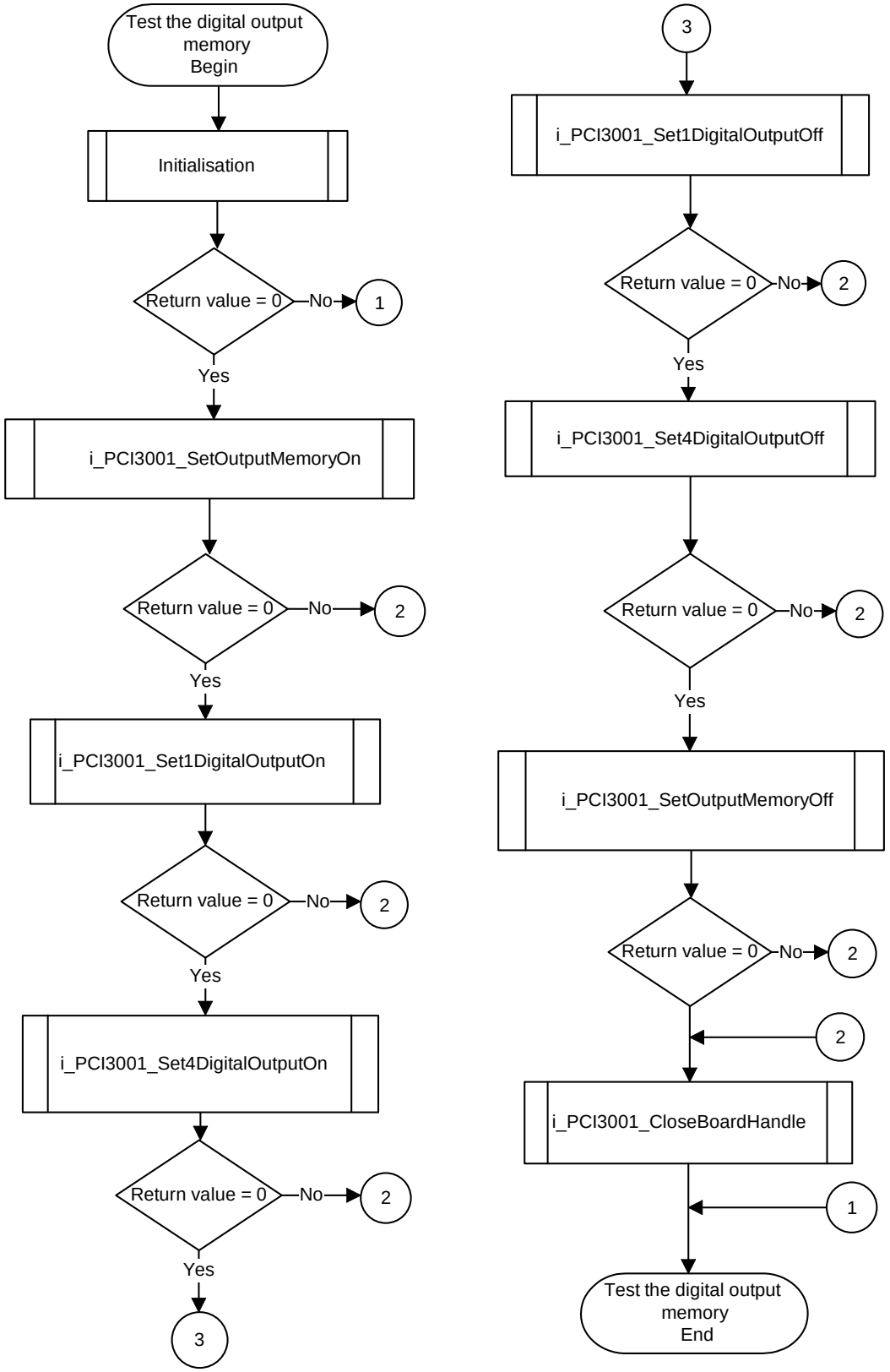
c) Example in C

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

     if (Initialisation (&b_BoardHandle) == 0)
        {
        if (i_PCI3001_Read4DigitalInput       (b_BoardHandle,
                            &ui_ReadValue)  == 0)
           {
           printf ("ui_ReadValue = %u", ui_ReadValue);
           }
        else
           {
           printf ("Read value error");
           }
        i_PCI3001_CloseBoardHandle (b_BoardHandle);
        }
     else
        {
        printf ("Initialisation error");
        }
     }
```

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 desactivated");
                        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");
     }
```

11 GLOSSARY

Table 11-1: Glossary

Term	Description
A/D converter	= *ADC* An electronic device that produces a digital output directly proportional to an analog signal output.
Acquisition	The process by which data is gathered by the computer for analysis or storage.
Clock	A circuit that generates time and clock pulses for the synchronisation of the conversion
D/A converter	= *DAC* A device that converts digital information into a corresponding analog voltage or current.
Data acquisition	Gathering information from sources such as sensors and transducers in an accurate, timely and organized manner. Modern systems convert this information to digital data which can be stored and processed by a computer.
DC voltage	= *Direct current voltage* DC voltage means that the voltage is constant respecting the time. It will always fluctuate slightly. Especially at switching on and switching off the transition behaviour is of high significance.
Disturb signal	Interferences that occur during the transfer caused by reduced bandwidth, attenuation, gain, noise, delay time etc.
Driver	A part of the software that is used to control a specific hardware device such as a data acquisition board or a printer.
FIFO	= *First In First Out* The first data into the buffer is the first data out of the buffer.
Gain	The factor by which an incoming signal is multiplied.
Ground	A common reference point for an electrical system.
Impedance	The reciprocal of admittance. Admittance is the complex ratio of the voltage across divided by the current flowing through a device, circuit element, or network.
INA	= *Instrumentation amplifier* A precision amplifier cirucuit with both high-impedance differential inputs and high common-mode rejection
Inductive loads	The voltage over the inductor is U=L.(dI/dt), whereas L is the inductivity and I is the current. If the current is switched on fast, the voltage over the load can become very highly for a short time.
Input impedance	The measured resistance and capacitance between the high and low inputs of a circuit.
Interrupt	A signal to the CPU indicating that the board detected the occurrence of a specified condition or event.

Limit value	Exceeding the limit values, even for just a short time, can lead to the destruction or to a loss of functionality.
MUX	= *Multiplexer* An array of semiconductor or electromechanical switches with a common output used for selecting one of a number of input signals.
Noise immunity	Noise immunity is the ability of a device to work during an electromagnetic interference without reduced functions.
Noise suppression	The suppression of undesirable electrical interferences to a signal. Sources of noise include the ac power line, motors, generators, transformers, fluorescent lights, CRT displays, computers, electrical storms, welders, radio transmitters, and others.
Operating voltage	The operating voltage is the voltage that occurs during the continuous operation of the device. It may not exceed the continuous limit voltage. Furthermore, any negative operation situations, such as net overvoltages over one minute at switching on the device must be taken in consideration.
Optical isolation	The technique of using an optoelectric transmitter and receiver to transfer data without electrical continuity, to eliminate high-potential differences and transients.
Parameter	The parameters of a control comprise all fort he control process required numeric values, e.g. for limit values and technological number.
PCI bus	PCI bus is a fast local bus with a clock rate up to 33 MHz. This bus is used for processing a great number of data. The PCI bus is not limited like the ISA and EISA systems.
Protective circuitry	A protective circuitry of the active part is done in order to protect the control electronic. The simplest protective circuitry is the parallel switching of a resistance.
Resolution	The smallest significant number to which a measurement can be determined. For example a converter with 12-bit resolution can resolve 1 part in 4096.
Sensor	A device that responds to physical stimuli (heat, light, sound, pressure, motion, etc.) and produces a corresponding electrical output.
Settling time	The time required, after application of a step input signal, for the output voltage to settle and remain within a specified error band around the final value. The settling time of a system includes that of all of the components of the system.
Short circuit	A short circuit of two clamps of an electric switch is when the concerning clamp voltage is zero.
Short circuit current	Short circuit current is the current between tow short-circuited clamps.
Signal delay	The change of a signal affects the following circuitries with finite velocity; the signal will be delayed. Besides the signal delay times that are not wanted, the signal delay can be extended by time switches and delay lines.

Synchronous	In hardware, it is an event that occurs in a fixed time relationship to another event. In software, it refers to a function that begins an operation and returns to the calling program only when the operation is complete.
Timer	The timer allows the adaptation of program processes between processor and peripheral devices. It usually contains from each other independent counters and can be programmed for several operation types over a control word register.
Trigger	Internal trigger: A software generated event that starts an operation. External trigger: An analog or digital hardware event from an external source that starts an operation. Digital trigger: An event that occurs at a user-selected point on a digital input signal. The polarity and sensitivity of the digital trigger can often be programmed.

12 INDEX

T

U

V

W