



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



**Technical description**

**ADDIALOG PA 3000**

**Analog inputs**

## 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. Offenses 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 nonfunctioning 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 carriers)
- deassembling, decompiling, decoding and reverse engineering of the software are forbidden.
- this licence and the software can be transferred to a third party, so far as this party has purchased a board, declares to agree to all the clauses of this licence contract and the preceding owner has not kept copies of the software.

## Trademarks

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

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

Intel is a registered trademark of Intel Corporation

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

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

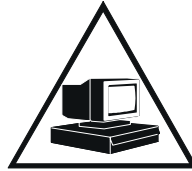
***The original version of this manual is in German. You can obtain it on request.***

# WARNING

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



**people may be injured**



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



**the environment may be polluted**

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

- **Do read the safety 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.

- **Symbols used**



**WARNING!**

It designates a possibly dangerous situation.

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



**IMPORTANT!**

designates hints and other useful information.

- **Any question?**

Our technical support is at your disposal



# Declaration of Conformity

This declaration is valid for the following product:

**ADDIALOG PA 3000**  
**Analog input channels, 12 bits**  
**with 4/8/16 Diff. or 4/8/16 SE input channels**  
**optically isolated**

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 manufacturing references listed in the form TD3000.020.

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.

<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
<b>5</b>	<b>SETTINGS</b> .....	<b>8</b>
5.1	Component scheme .....	8
5.2	Jumper settings .....	9
5.2.1	Jumper location and settings at delivery .....	9
5.2.2	Jumper settings for the different functions .....	9
<b>6</b>	<b>INSTALLATION</b> .....	<b>10</b>
6.1	Base address .....	11
6.2	Inserting the board .....	12
6.2.1	Opening the PC .....	12
6.2.2	Selecting a free slot .....	12
6.2.3	Inserting the board .....	13
6.2.4	Closing the PC .....	13
6.3	Installing the software .....	14
6.3.1	Software installation under MS-DOS and Windows 3.11 : .....	14
6.3.2	Software installation under Windows NT / 95: .....	14
6.4	Board configuration with ADDIREG for Windows NT 4.0/ 95 .....	15
6.4.1	Program description .....	15
6.4.2	Registrating a new board .....	18
6.4.3	Changing the registration of a board .....	19
6.4.4	Removing the ADDIREG program .....	19
6.5	Board configuration with ADDIMON for DOS .....	20
6.5.1	Help for configuration .....	20
6.5.2	Testing the functions .....	20
6.5.3	Hotline protocol, quick technical support .....	20
6.6	Error analysis per Internet .....	20

<b>7</b>	<b>CONNECTION TO THE PERIPHERAL .....</b>	<b>21</b>
<b>7.1</b>	<b>Connector pin assignments.....</b>	<b>21</b>
<b>7.2</b>	<b>Connection principle .....</b>	<b>22</b>
<b>7.3</b>	<b>Connection examples .....</b>	<b>23</b>
<b>8</b>	<b>SOFTWARE EXAMPLES .....</b>	<b>24</b>
<b>8.1</b>	<b>Initialisation.....</b>	<b>24</b>
8.1.1	Initialisation of the PA 3000 under DOS and Windows 3.11 .....	24
	a) Flow chart.....	24
	b) Example in C for DOS and Windows 3.11 .....	25
8.1.2	Initialisation of the PA 3000 under Windows NT / 95 .....	26
	a) Flow chart.....	26
	b) Example in C for Windows NT / 95.....	27
<b>8.2</b>	<b>Interrupt.....</b>	<b>28</b>
8.2.1	Interrupt routine under DOS and Windows 3.11 .....	28
	a) Flow chart for DOS, Windows 3.11 and Windows NT/95 (asynchronous mode) .....	28
	b) Example in C for DOS and Windows 3.11 .....	29
	c) Example in C for Windows NT / 95 (asynchronous mode).....	30
	d) Flow chart for Windows NT / 95 (synchronous mode).....	31
	e) Example in C for Windows NT / 95 (synchronous mode) .....	32
<b>8.3</b>	<b>Direct conversion of analog input channels .....</b>	<b>33</b>
8.3.1	Testing one analog input channel.....	33
	a) Flow chart.....	33
	b) Example in C .....	34
8.3.2	Testing all analog input channels .....	35
	a) Flow chart.....	35
	b) Example in C .....	36
<b>8.4</b>	<b>Cyclic conversion of the analog inputs.....</b>	<b>37</b>
8.4.1	Cyclic conversion without DMA, external trigger and delay .....	37
	a) Flow chart.....	37
	b) Example in C for DOS .....	38
	c) Example in C for Windows 3.1x.....	39
	d) Example in C for Windows NT / 95 (asynchronous mode).....	40
	e) Example in C for Windows NT / 95 (synchronous mode) .....	41
8.4.2	Cyclic conversion with DMA without external trigger and delay .....	42
	a) Flow chart.....	42
	b) Example in C for DOS .....	43
	c) Example in C for Windows 3.1x.....	44
	d) Example in C for Windows NT / 95 (aynchronous mode) .....	45
	e) Example in C for Windows NT / 95 (synchronous mode) .....	46

<b>8.5</b>	<b>Timer2</b> .....	<b>47</b>
8.5.1	Testing timer interrupt .....	47
	a) Flow chart .....	47
	b) Example in C for DOS .....	48
	c) example in C for Windows 3.1x.....	49
	d) Example in C for Windows NT / 95 (asynchronous mode) .....	50
	e) Example in C for Windows NT / 95 (synchronous mode) .....	51
<b>8.6</b>	<b>Digital input channels</b> .....	<b>52</b>
8.6.1	Reading one digital input channel .....	52
	a) flow chart.....	52
	b) Example in C .....	53
8.6.2	Reading 4 digital input channels .....	54
	a) Flow chart .....	54
	b) Example in C .....	55
<b>8.7</b>	<b>Digital output channels</b> .....	<b>56</b>
8.7.1	Testing the digital output memory .....	56
	a) flow chart.....	56
	b) Example in C .....	57
<b>INDEX</b>	.....	<b>59</b>

**Figures**

Fig. 3-1: Wrong handling..... 3  
 Fig. 3-2: Correct handling..... 3  
 Fig. 5-1: Component scheme..... 8  
 Fig. 5-2: Jumper location on the PA 3000 board..... 9  
 Fig. 6-1: DIP switches..... 11  
 Fig. 6-2: Slot types..... 12  
 Fig. 6-3: Opening the protective blister pack ..... 12  
 Fig. 6-4: Inserting the board..... 13  
 Fig. 6-5: Securing the board at the back cover..... 13  
 Fig. 6-6: ADDIREG registration program ..... 15  
 Fig. 6-7: Configuring a new board..... 17  
 Fig. 7-1: 37-pin SUB-D male connector X18..... 21  
 Fig. 7-2: Connection of the ribbon cable FB3000 ..... 21  
 Fig. 7-3: Connection principle ..... 22  
 Fig. 7-4: Analog input channels..... 23  
 Fig. 7-5: Digital output channels..... 23  
 Fig. 7-6: Digital input channels..... 23

**Tables**

Table 5-1: Jumper settings for the different functions ..... 9  
 Table 6-1: Decoding table ..... 11

# 1 INTENDED PURPOSE OF THE BOARD

The board **PA 3000** is the interface between an industrial process and a personal computer (PC). It is to be used in a free PC ISA 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  mark.

Data exchange between the board **PA 3000** and the peripheral is to occur through a shielded cable, which has to be connected to the 37-pin SUB-D male connector of the board **PA 3000**.

The board has up to 16 differential input channels intended for processing analog signals. The screw terminal board **PX 901** allows to connect the analog signals through a shielded cable. The use of the 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 specifications:

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

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

The use of the board according to its intended purpose includes observing all advices given in this manual 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.

**Our boards are not to be used for securing emergency stop functions.**

The emergency stop functions are to be secured separately.  
This securing must not be influenced by the board or the PC.

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:

- installation,
- use,
- maintenance.

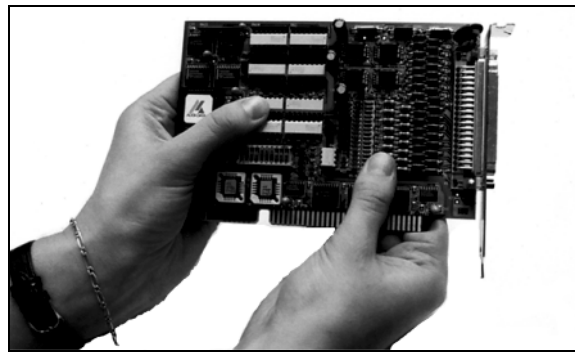
### **2.2 Personal protection**

Consider the country-specific regulations about

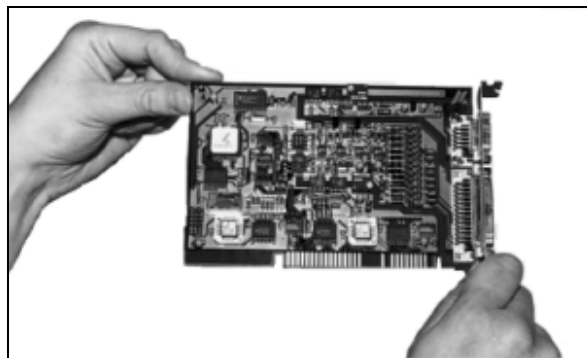
- the prevention of accidents
- electrical and mechanical installations
- radio interference suppression.

### 3 HANDLING THE BOARD

**Fig. 3-1: Wrong handling**



**Fig. 3-2: Correct handling**



## 4 TECHNICAL DATA

### 4.1 Electromagnetic compatibility (EMC)

The board has been subjected in an accepted laboratory to the EMC tests. The board complies as follows with the limit values set by the norms EN50082-2, EN55011, EN55022:

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



#### **WARNING!**

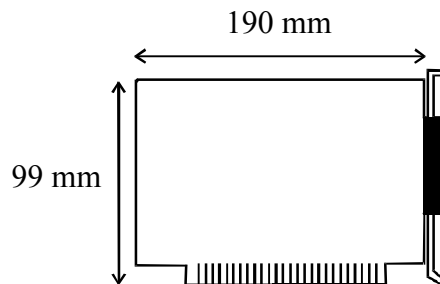
The EMC tests have been carried out in a specific appliance configuration. We guarantee these limit values **only** in this configuration<sup>1</sup>.

#### **Consider the following aspects:**

- Use well shielded lines.
- your test program must be able to recognize errors of operation.
- **Set your system up**, so that you can know what caused the errors.

### 4.2 Physical set-up of the board

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



Weight:	170 g
Installation in:	AT slot
Connection to the peripheral	37-pin SUB-D male connector
	Cables: ST010, ST011, FB3000 for the connection of digital I/O.
	Screw terminal boards: PX 901-A(G) PX 901-ZG for dig. I/O

See Fig. 7-3: Connection principle

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

### 4.3 Options

For version PA3000-16

**Option D2F:** Precision filter for 16 differential input channels

**Option D2C:** Current input channels 0-20 mA or 4-20 mA for 16 differential input channels.



**IMPORTANT!**

By 4-20mA the precision is altered

### 4.4 Versions

The PA 3000 is available in the following versions:

Version	Differential or Single-Ended inputs	Digital inputs and outputs
PA3000-4	4	4 /4
PA3000-8	8	4 /4
PA3000-16	16	4 /4

### 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:**

- operating system: ..... MS DOS 3.3 or > Windows 3.1

- bus speed: ..... 8 MHz

**Energy requirements :**

- operating voltage of the PC: ..... 5V ± 5%

- current consumption in mA(without load)

typically: ..... see table below ± 10%

	PA3000-4	PA3000-8	PA3000-16
+ 5 V from the PC	530 mA	580 mA	630 mA

**Analog input channels:**

Number of analog input channels: .....	16 diff. for <b>PA 3000-16</b>
Analog resolution: .....	12-bit, 1 among 4096
Data transfer rate (1 input): .....	150 kHz
Data transfer: .....	Data to the PC (16-bit only) 1) through I/O commands 2) Interrupt at EOC <sup>1</sup> 3) DMA transfer at EOC
Start of conversion: .....	1) per software trigger 2) TIMER 0 of 82C54 3) TIMER 0 & 1 of 82C54 4) through ext. trigger (dig. input 1)
Monotony: .....	12-bit
Offset error: .....	after calibration: - Bipolar: $\pm 1/2$ LSB - Unipolar: $\pm 1/2$ LSB Drift (0°C to 60°C): - Bipolar: $\pm 2$ ppm / °C - Unipolar: $\pm 2$ ppm / °C
Gain error: .....	after calibration: - Bipolar: $\pm 1/2$ LSB - Unipolar: $\pm 1/2$ LSB Drift (0°C to 60°C): - Bipolar: $\pm 7$ ppm / °C - Unipolar: $\pm 7$ ppm / °C
Analog input ranges: .....	Voltage - Unipolar: 0-10 V - Bipolar: $\pm 10$ V Software programmable Current (Option D2C) - Unipolar: 0-20 mA Unipolar mode and gain x2 must be selected
Overvoltage protection: .....	70 V <sub>pp</sub> <sup>2</sup> when POWER ON
Common mode rejection: .....	from 8 dB (min.) to 10 Hz, gain = 1
Bandwidth (-3dB): .....	Limited to 159 kHz (-3 dB) with low-pass filter 1st order; but the minimum SINAD is still 74 dB at 47 kHz (fin)
Bias currents for the input channels: .....	$\pm 10$ nA max.
Input impedance: .....	$10^{12} \Omega$ // 20 nF to GND
Integral non-linearity (INL): .....	$\pm 1/2$ LSB
Differential non-linearity (DNL): .....	$\pm 1/2$ LSB
Accuracy: .....	$\pm 1$ LSB
Selectable gain: .....	via PGA gain 1, 2, 5, 10, 20, 50, 100, 200, 500, 1000 (software programmable for each channel)

---

<sup>1</sup> EOC: End of Conversion

<sup>2</sup> V<sub>pp</sub>: peak-to-peak voltage

System noise: .....	Bipolar: Gain x1: $\pm 1/2$ LSB Gain x2: $\pm 1/2$ LSB Gain x10: $\pm 2$ LSB Gain x100: $\pm 5$ LSB
	Unipolar: Gain x1: $\pm 1/2$ LSB Gain x2: $\pm 1/2$ LSB Gain x10: $\pm 2$ LSB Gain x100: $\pm 10$ LSB
Digital coding: .....	Offset binary
Optical isolation from the PC .....	500 VDC minimum
DMA access: .....	channels 5,6 and 7 (1- or 2-channel DMA)
Interrupts: .....	IRQ 3, 5 for XT,
(software pogrammable)	IRQ 9, 10, 11, 12, 14, 15 for AT
Timers: .....	3 x 16-bit timer (2 for sequeunce programming)

**Digital input channels:**

Number of digital input channels: .....	4
Input current at 24V: .....	3 mA typ.
Input voltage range: .....	0-30 V
Optical isolation from the PC: .....	1000 VAC
Logic "0" level: .....	0-5 V
Logic "1" level: .....	10-30 V

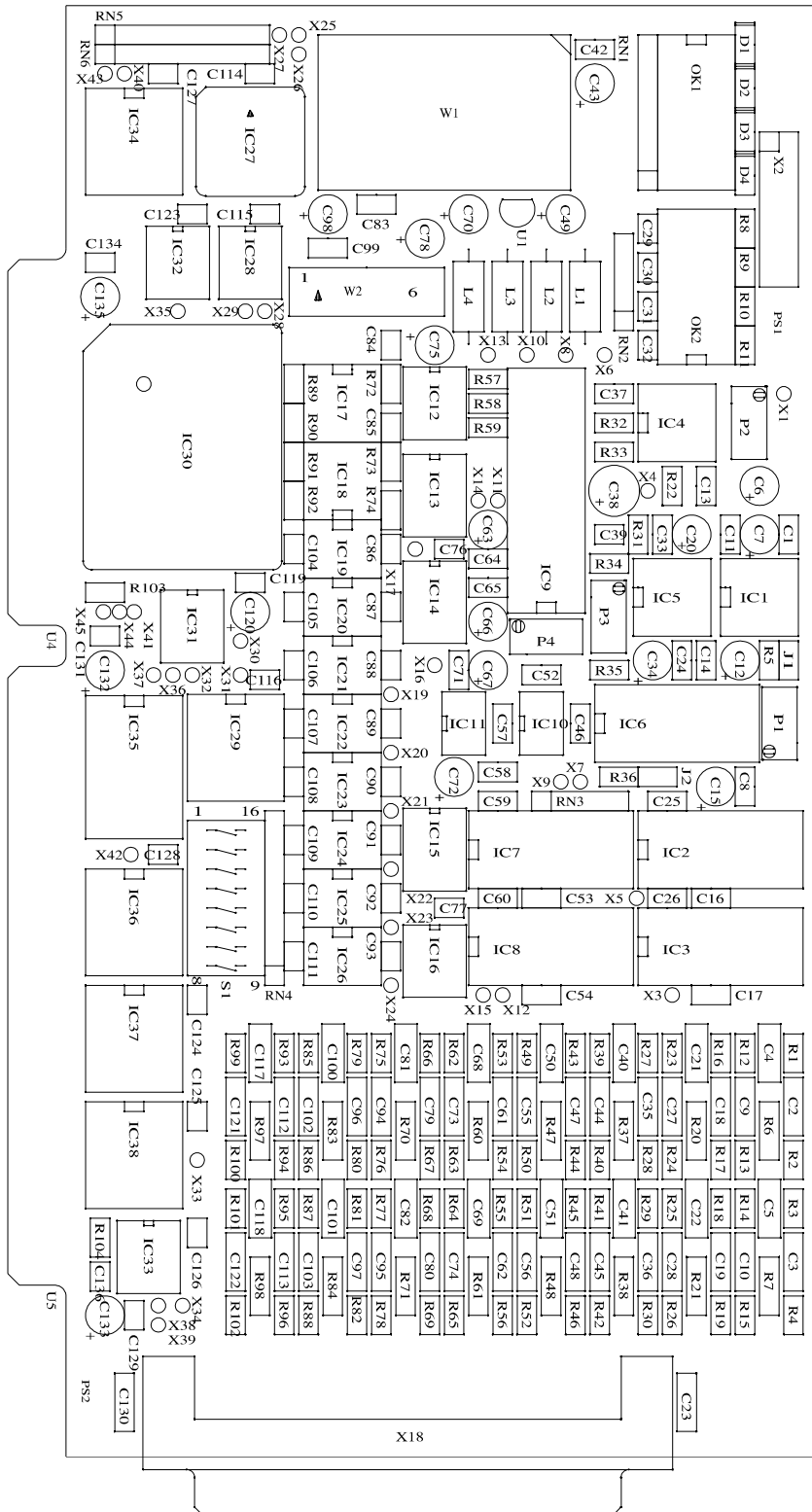
**Digital output channels:**

Number of digital output channels: .....	4, 24 V
Maximum switching current: .....	5 mA typ.
Voltage range: .....	5-30 V
Optical isolation from the PC: .....	1000 VAC

# 5 SETTINGS

## 5.1 Component scheme

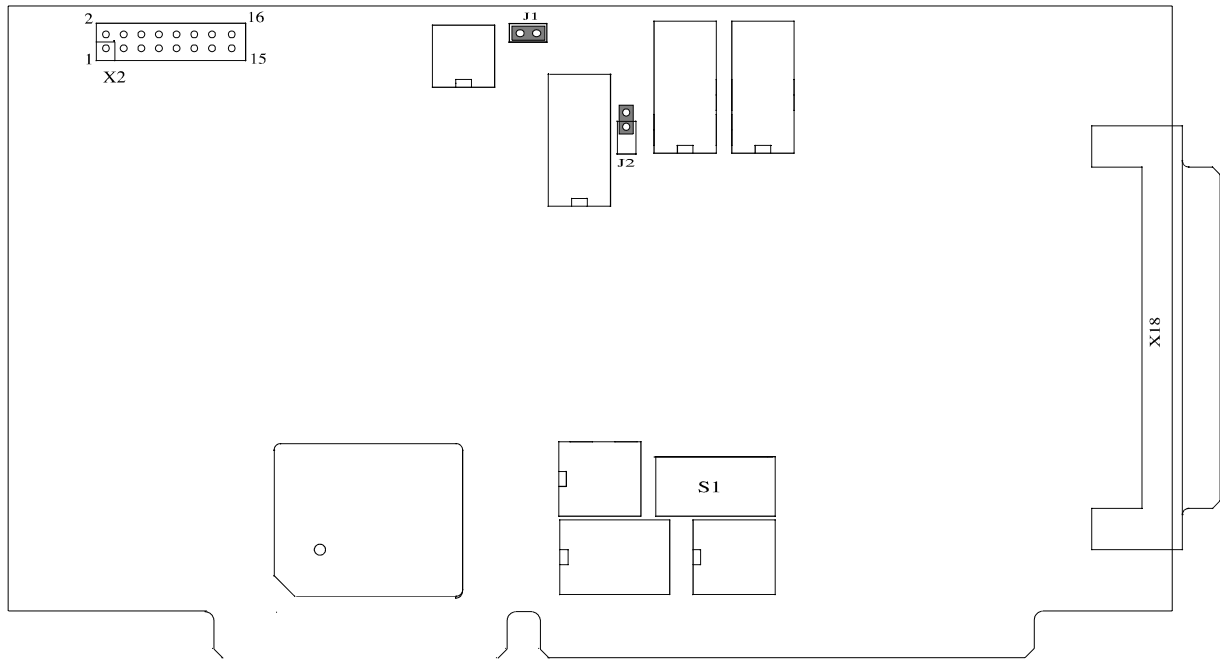
Fig. 5-1: Component scheme



## 5.2 Jumper settings

### 5.2.1 Jumper location and settings at delivery

Fig. 5-2: Jumper location on the PA 3000 board



### 5.2.2 Jumper settings for the different functions

Table 5-1: Jumper settings for the different functions

Designation	Position	Function	Remarks	Delivery
J1	set	gains 1, 2, 5, 10, 20, 50, 100 selectable by software		✓
J1	not set	gains 100, 200, 500, 1000 selectable by software	needed when all input channels receive small signals	
J2	set	a virtual ground is created at the INAs (-) input through a 1MΩ resistor connected to the signal ground	(should a problem occur in differential mode) the signal is not within the INA common-mode range	
J2	not set	real differential mode		✓

## 6 INSTALLATION

### i

#### IMPORTANT!

If you want to install simultaneously **several** ADDI-DATA boards, 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 the software (once is enough)
  5. Configure the board
  
  6. Switch off the PC
  7. Install the **second** board
  8. Start the PC
  9. Configure the board

etc

You will find additional information to these different steps in the sections 6.1 to 6.5.

### i

#### IMPORTANT!

You have installed already **one or more** ADDI-DATA boards in your PC, and you wish to install **an additional** board?

Proceed as if you wished to install one single board.

## 6.1 Base address



**WARNING!**

If the base address set is wrong, the board and/or the PC may be destroyed.

**Before installing the board**

The base address is set at delivery on the address 0390H.

- **Check**, that
  - the base address is free
  - the address range (8 I/O addresses ) required by the board is not already used by the PC or by boards already installed in the PC.

The base address and/or the address range **are wrong?**

- **Select** another base address with the 8-pole block of DIP switches S1

The address 0390H is decoded in the following figure.

**Table 6-1: Decoding table**

	MSB												LSB			
Decoded address bus	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
Wished base address Hex	0				3				9				0			
Wished base address binary	0	0	0	0	0	0	1	1	1	0	0	1	0	0	0	0
DIP switch S1 Logic "0"= ON Logic "1" = OFF	*	*	*	*	*	s8	s7	s6	s5	s4	s3	s2	s1	X	X	X
						ON	OFF	OFF	OFF	ON	ON	OFF	ON			

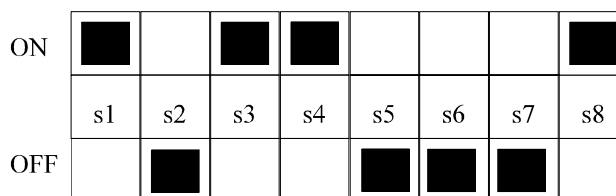
X: decoded address range of the board  
 \*: permanently decoded on logic "0"

**Fig. 6-1: DIP switches**

**IMPORTANT!**

You will find the switch **s1 on the left** of the DIP switches!

**S1**



## 6.2 Inserting the board

**i**

### IMPORTANT!

Please follow absolutely the *safety instructions*.

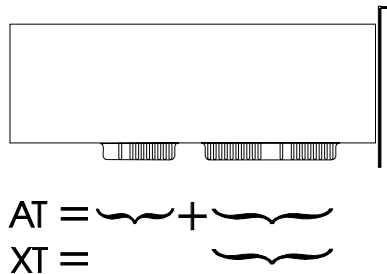
### 6.2.1 Opening the PC

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

### 6.2.2 Selecting a free slot

1. Select a free AT slot

**Fig. 6-2: Slot types**



The board can also be used in EISA slots.

2. **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.
3. **Discharge yourself from electrostatic charges**
4. **Take the board from its protective blister pack.**

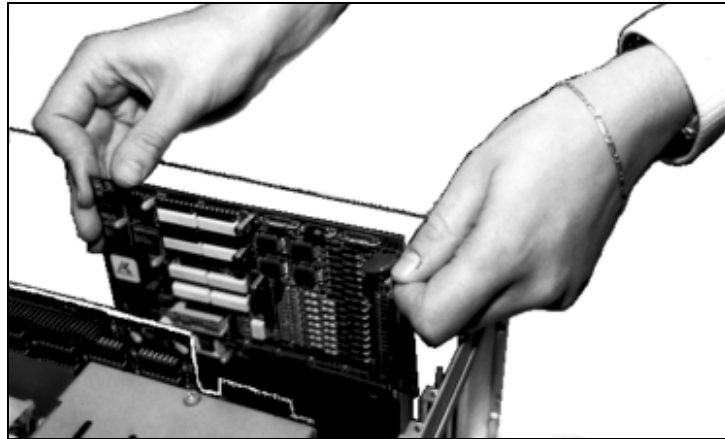
**Fig. 6-3: Opening the protective blister pack**



### 6.2.3 Inserting the board

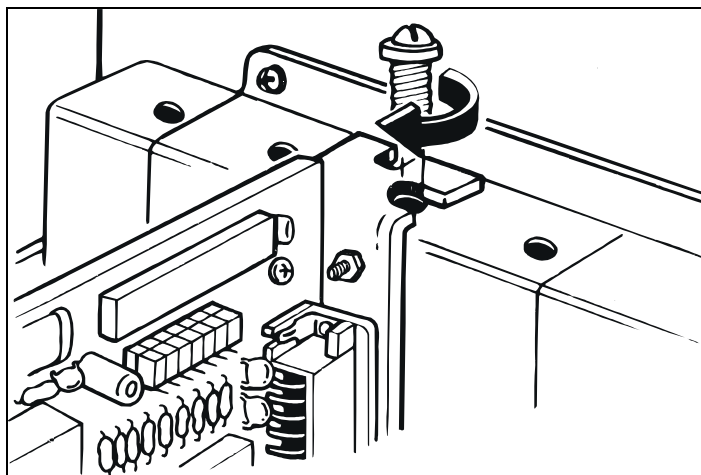
- Discharge yourself from electrostatic charges
- Insert the board vertically into the chosen slot.

**Fig. 6-4: Inserting the board**



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

**Fig. 6-5: Securing the board at the back cover**



- Tighten all the loosed screws.

### 6.2.4 Closing the PC

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

## 6.3 Installing the software

The CD contains:

- ADDIREG for Windows NT 4.0 and Windows 95,  
You can also download the ADDIREG program from Internet.
- Standard software for the ADDI-DATA boards:
  - 16-bit for MS-DOS and Windows 3.11
  - 32-bit for Windows NT/95.

### 6.3.1 Installation under MS-DOS and Windows 3.11

- Copy the contents of PA3000\16bit 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.

### 6.3.2 Installation under Windows NT / 95

- Select the directory PA3000\32bit\Disk1 corresponding to the board.
- Start the setup 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.

## 6.4 Board configuration with ADDIREG for Windows NT 4.0/ 95

The ADDIREG registration program is a 32-bit program. The user can registrate all hardware informations necessary to operate the ADDI-DATA PC boards.

### 6.4.1 Program description

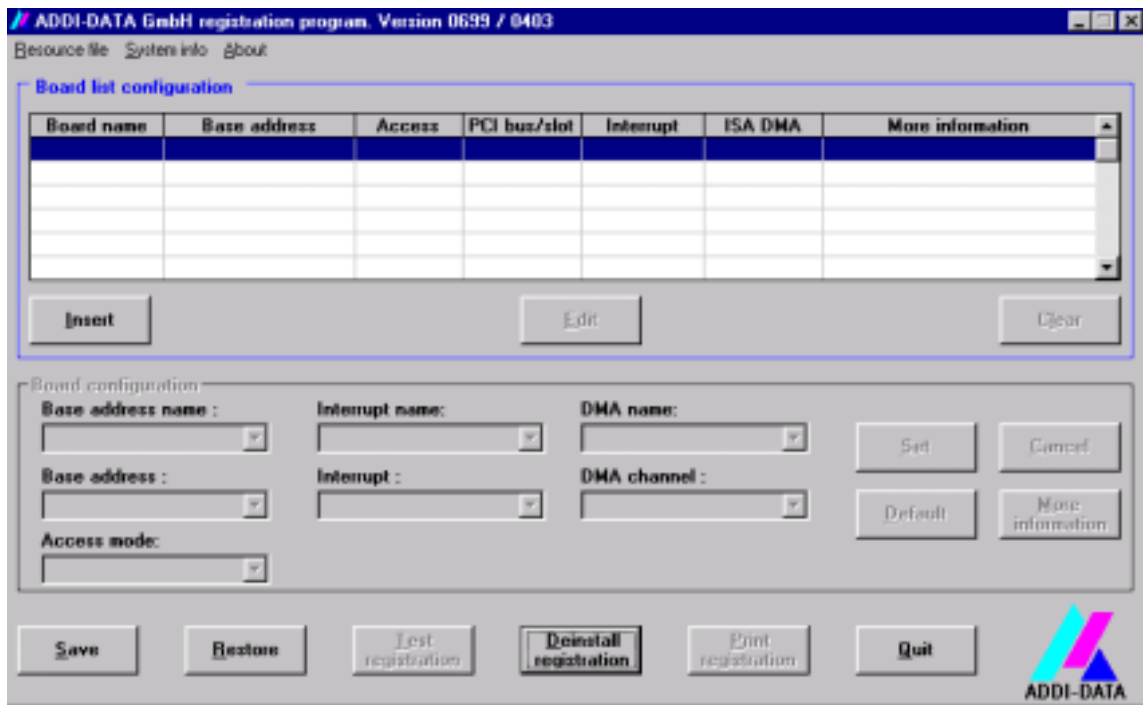
**i**

**IMPORTANT!**

Insert the ADDI-DATA boards to be registrated 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 box appears.

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



Screen explanation:

**Table:**

The table in the middle lists the registrated boards and their respective parameters.

**Board name:**

Names of the different registrated boards (eg.: APCI-7800).

When you start the program for the first time, no board is registrated in this table.

**Base address:**

Selected base address of the board.

**i**

**IMPORTANT!**

The base address selected with the ADDIREG program must correspond to the one set through DIP-switches.

**Access:**

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

**PCI bus / slot:**

Used PCI slot. If the board is no PCI board, the message „NO“ is displayed.

**Interrupt:**

Used interrupt of the board. If the board uses no interrupt, the message „Not available“ is displayed.

**i****IMPORTANT!**

The interrupt selected with the ADDIREG program must correspond to the one set through DIP-switches.

**ISA DMA:**

Indicates the selected DMA channel or „Not available“ if the board uses no DMA.

**More information:**

Additional information like the identifier string (eg.: PCI1500-50) or the installed COM interfaces.

**Text boxes:**

Under the table you will find 6 text boxes in which you can change the parameters of the board.

**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.

**Interrupt name:**

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

**Interrupt:**

Selection of the interrupt number which the board has to use.

**DMA name:**

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

**DMA channel:**

Selection of the used DMA channel.

**Buttons:**

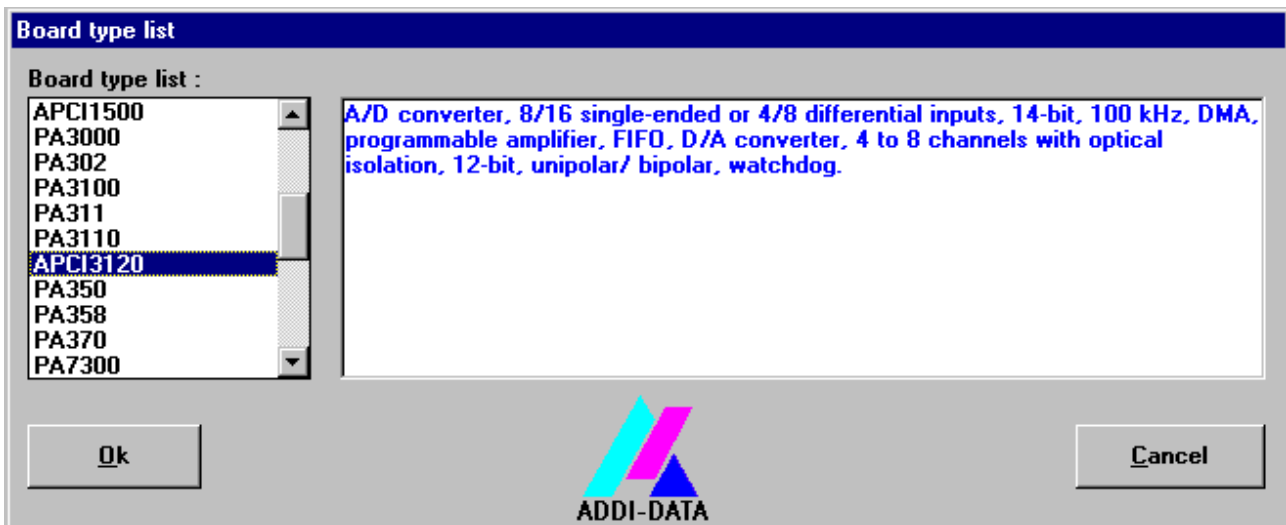
**Edit <sup>1</sup>:**

Selection of the highlighted board with the different parameters set in the text boxes. Click on „Edit“ to activate the data or click twice on the selected board.

**Insert:**

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

**Fig. 6-7: Configuring a new board**



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

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

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

**Clear:**

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

**Set:**

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

**Cancel:**

Reactivates the former parameters of the saved configuration.

**Default:**

Sets the standard parameters of the board.

**More information:**

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

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

<sup>1</sup> "x": Keyboard shortcuts; e.g. "Alt + e" for Edit

**Save:**

Saves the parameters and registers the board.

**Restore:**

Reactivates the last saved parameters and registration.

**Test registration:**

Controls if there is a conflict between the board and other devices.

A message indicates the parameter which has generated the conflict. If there is no conflict, „OK“ is displayed.

**Deinstall registration:**

Deinstalls the registrations of all board listed in the table.

**Print registration:**

Prints the registration parameter on your standard printer.

**Quit:**

Quits the ADDIREG program.

## 6.4.2 Registering a new board

### **i**

**IMPORTANT!**

To register a new board, you must have administrator rights.

Only an administrator is allowed to register a new board or change a registration.

- Call up the ADDIREG program. The figure X-X is displayed on the screen. Click on „Insert“. Select the wished board.
- Click on „OK“. The default address, interrupt, and the other parameters are automatically set in the lower fields. The parameters are listed in the lower fields.  
If the parameters are not automatically set by the BIOS, you can change them. Click on the wished scroll function(s) and choose a new value. Activate your selection with a click.
- Once the wished configuration is set, click on „Set“.
- Save the configuration with „Save“.
- You can test if the registration is „OK“.  
This test controls if the registration is right and if the board is present. If the test has been successfully completed you can quit the ADDIREG program. The board is initialised with the set parameters and can now be operated.

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

### 6.4.3 Changing the registration of a board

#### i

#### IMPORTANT!

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

- Call up the ADDIREG program. Select the board to be changed. The board parameters (Base address, DMA channel, ..) are listed in the lower fields.
- Click on the parameter(s) you want to set and open the scroll function(s).
- Select a new value. Activate it with a click. Repeat the operation for each parameter to be modified.
- Once the wished configuration is set, click on „Set“.
- Save the configuration with „Save“.
- You can test if the registration is „OK“.  
This test controls if the registration is right and if the board is present. If the test has been successfully completed you can quit the ADDIREG program. The board is initialised with the set parameters and can now be operated.

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

### 6.4.4 Removing the ADDIREG program

The ADDI\_UNINSTAL program is delivered on the CD-ROM.

- Install the program on your computer.
- Start the program
- Proceed as indicated until the complete removing of ADDIREG.

You can also download the programm from Internet.

## 6.5 Board configuration with ADDIMON for DOS

- In the directory MONITOR enter <MON3000>.
- The monitoring program of the board PA 3000 is loaded.

### 6.5.1 Help for configuration

- Load ADDIMON
- Select the wished configuration
- Visualise the jumper settings  
(or DIP switches settings for the base address)
- Configure the board as indicated on the screen

### 6.5.2 Testing the functions

- Insert the board in your PC.
- Load ADDIMON
- The basic functions of the board are available
- You can build up and test a simple AD conversion

### 6.5.3 Hotline protocol, quick technical support

Should a problem arise:

- fill in the hotline protocol.
  - Print it and send it to our technical support
- You will then receive a quick solution to your problem.

## 6.6 Error analysis per Internet

Do not hesitate to e-mail us your questions.

Our Internet page is accessed:

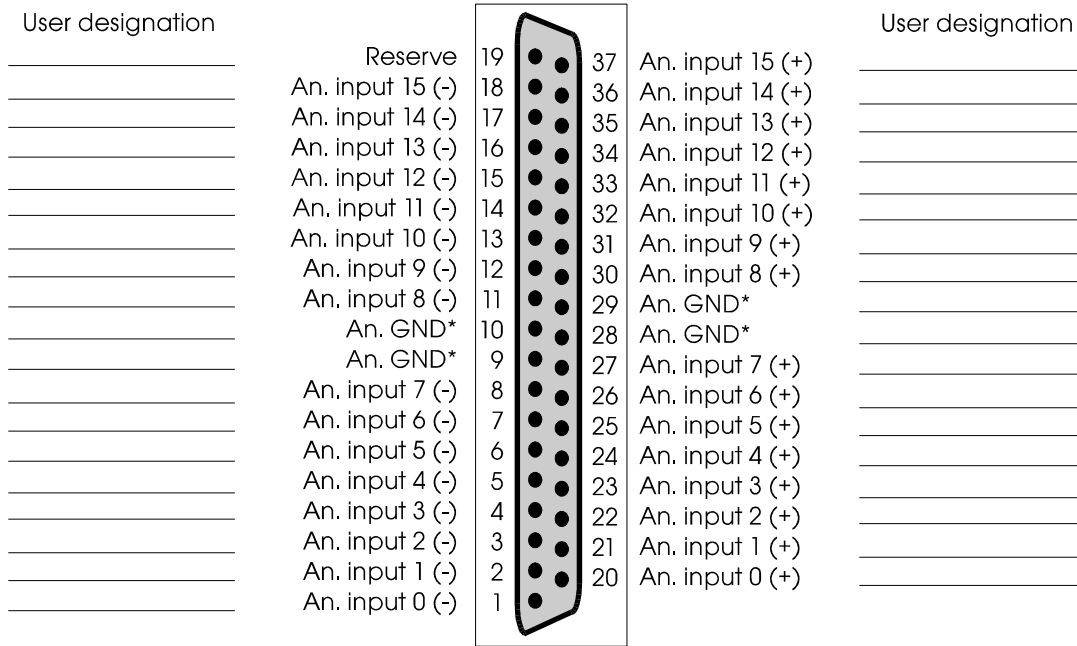
- per e-mail: [info@addi-data.de](mailto:info@addi-data.de)
- per Internet: <http://www.addi-data.de> or  
<http://www.addi-data.com>

**Free downloads of the standard softwares**

# 7 CONNECTION TO THE PERIPHERAL

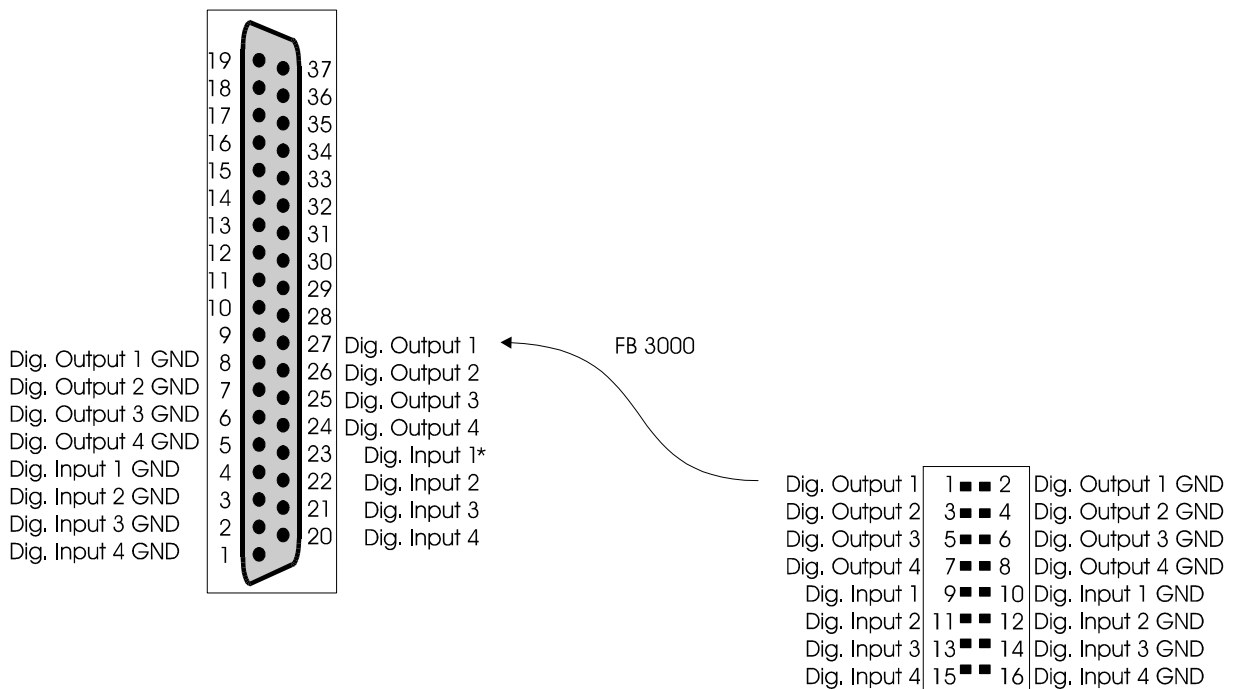
## 7.1 Connector pin assignments

**Fig. 7-1: 37-pin SUB-D male connector X18**



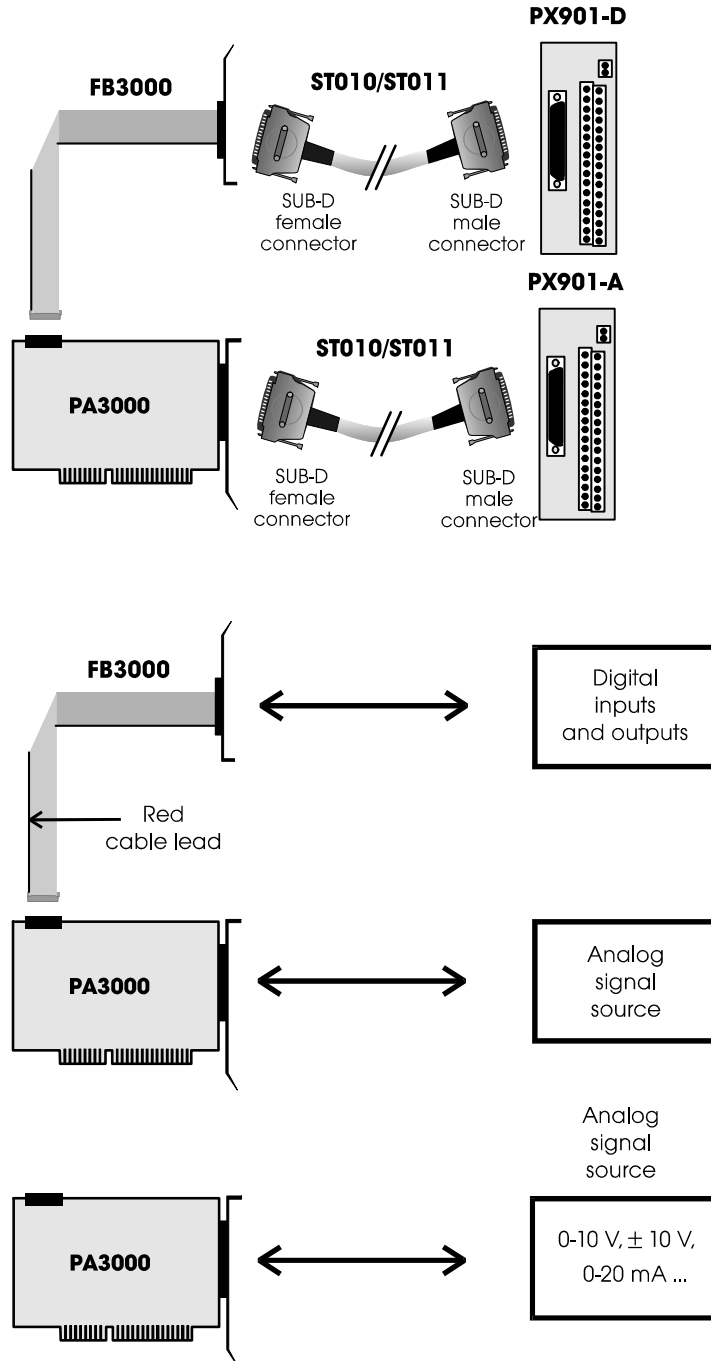
\* Common ground for all analog input channels

**Fig. 7-2: Connection of the ribbon cable FB3000**



## 7.2 Connection principle

Fig. 7-3: Connection principle



**i**

**IMPORTANT!**

Insert the FB3000 on the connector with the red cable lead on the left of the board.

### 7.3 Connection examples

Fig. 7-4: Analog input channels

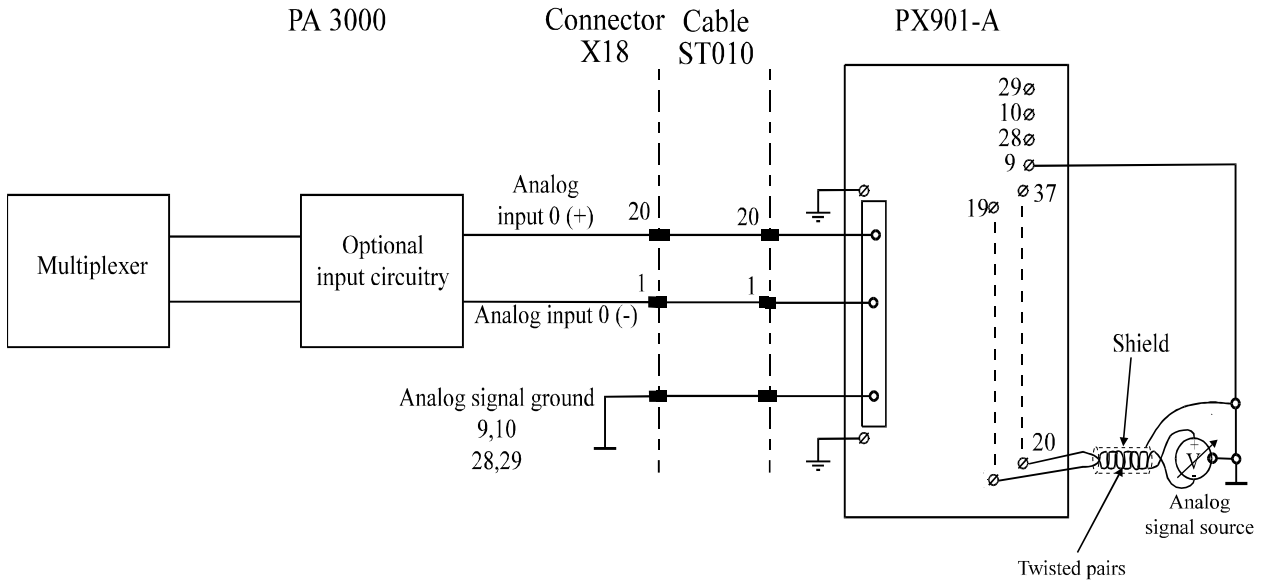


Fig. 7-5: Digital output channels

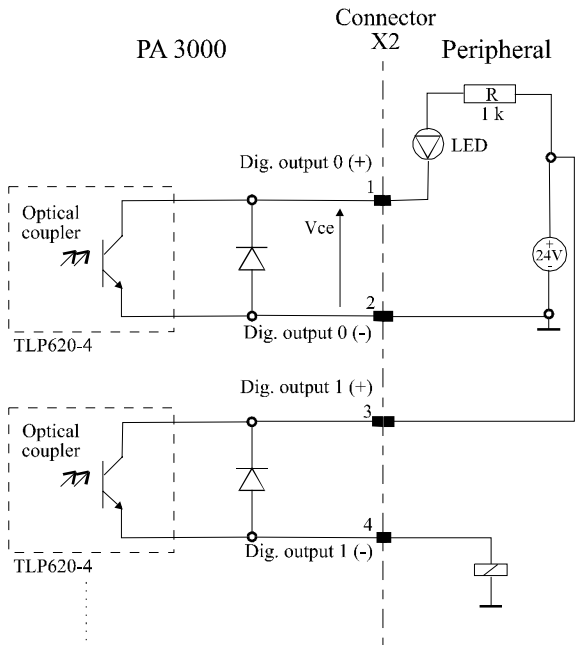
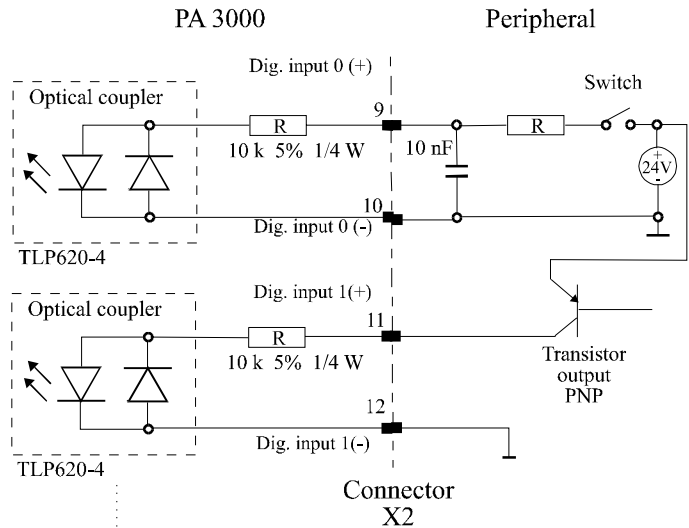


Fig. 7-6: Digital input channels

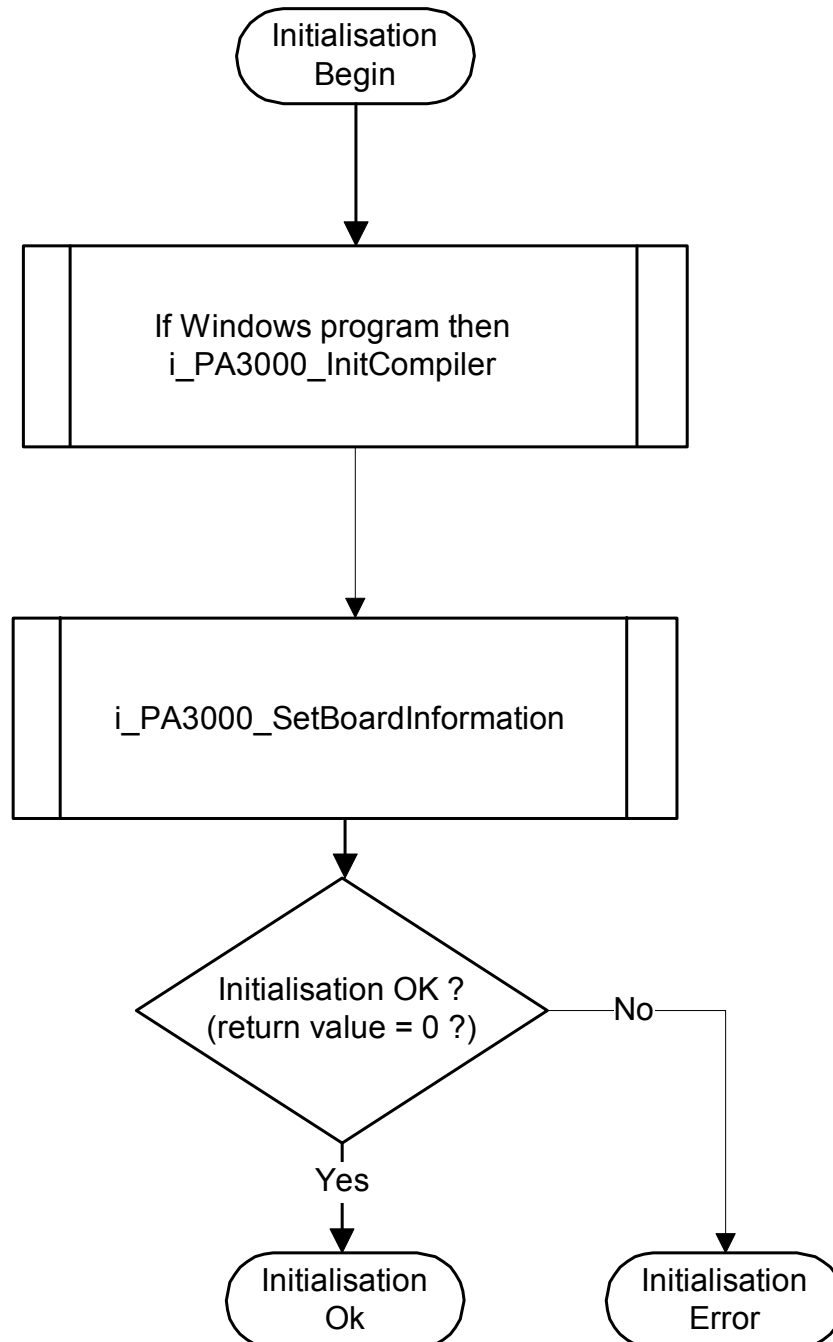


## 8 SOFTWARE EXAMPLES

### 8.1 Initialisation

#### 8.1.1 Initialisation of the PA 3000 under DOS and Windows 3.11

##### a) Flow chart



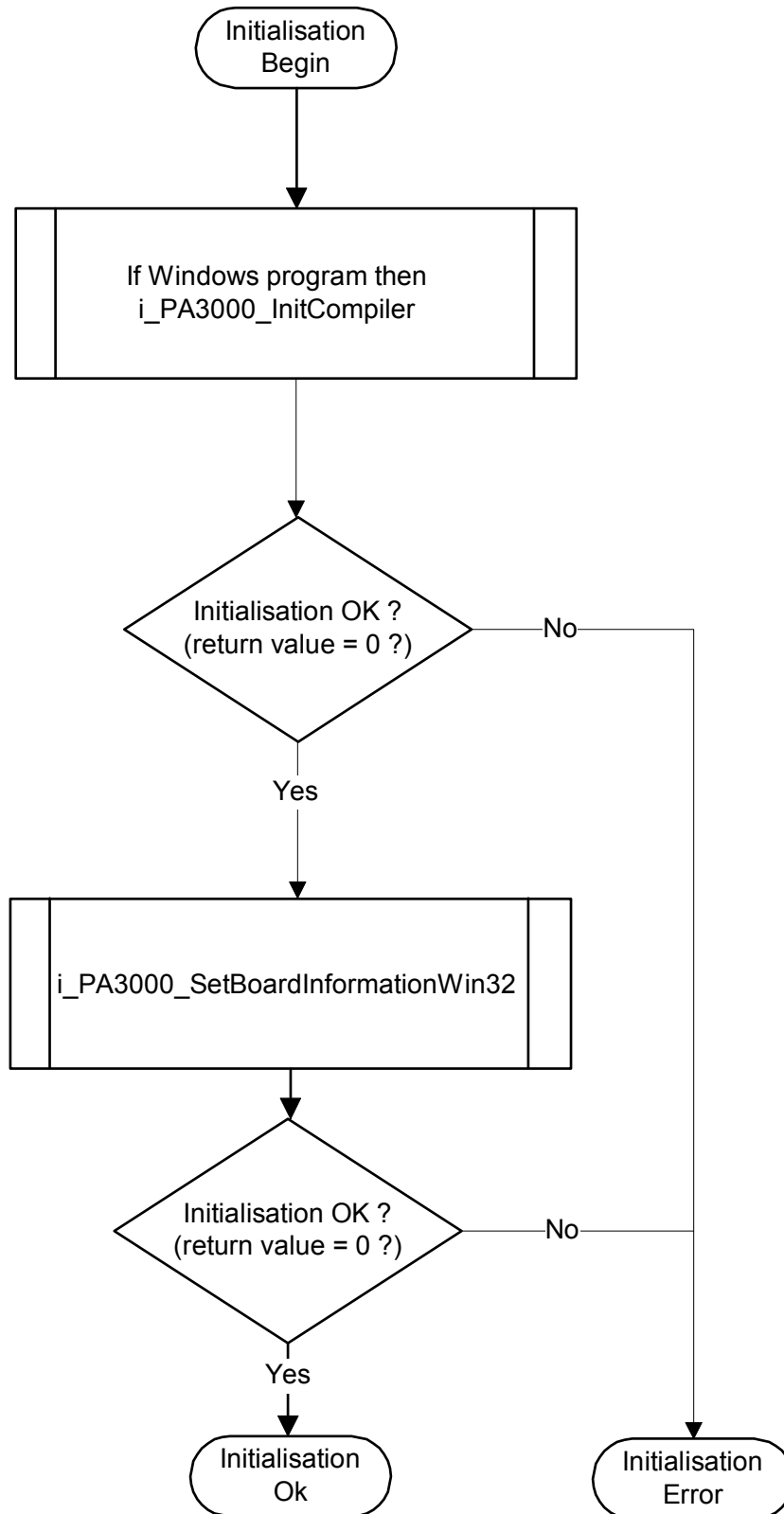
**b) Example in C for DOS and Windows 3.11**

```
int Initialisation(unsigned char *pb_BoardHandle)
{
#ifdef _Windows
    i_PA3000_InitCompiler (DLL_COMPILER_C);
#endif

    if(i_PA3000_SetBoardInformation (0x390,
                                    3, // IRQ3
                                    5, // DMA5
                                    6, // DMA6
                                    16,
                                    pb_BoardHandle) == 0)
    {
        return (0);        /* OK */
    }
    else
    {
        return (-1);      /* ERROR */
    }
}
```

### 8.1.2 Initialisation of the PA 3000 under Windows NT / 95

#### a) Flow chart



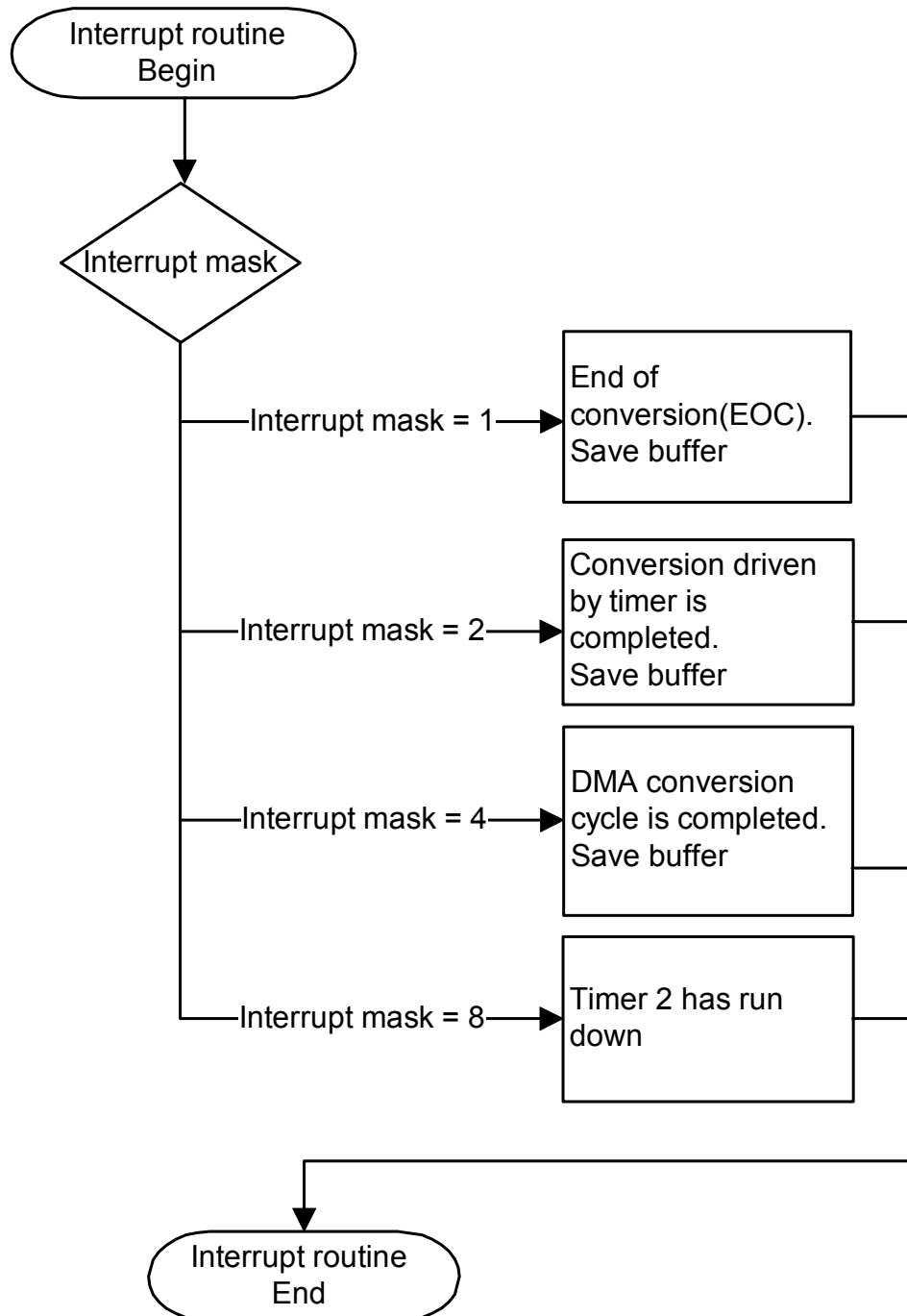
**b) Example in C for Windows NT / 95**

```
int Initialisation(unsigned char *pb_BoardHandle)
{
    if (i_PA3000_InitCompiler (DLL_COMPILER_C) == 0)
    {
        if(i_PA3000_SetBoardInformationWin32 ("PA3000-00",
                                             16,
                                             pb_BoardHandle) == 0)
        {
            return (0);      /* OK */
        }
        else
        {
            return (-1);    /* ERROR */
        }
    }
    else
    {
        return (-1);      /* ERROR */
    }
}
```

## 8.2 Interrupt

### 8.2.1 Interrupt routine under DOS and Windows 3.11

a) Flow chart for DOS, Windows 3.11 and Windows NT / 95 (asynchronous mode)



**b) Example in C for DOS and Windows 3.11**

```
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:
            /* EOC interrupt */
            ui_SaveArray[0] = pui_ValueArray[1];
            break;

        case 4:
            /* DMA completed */
            for (ui_Cpt=0;ui_Cpt<16;ui_Cpt++)
                ui_SaveArray [ui_Cpt] = pui_ValueArray [ui_Cpt];
            break;

        case 8:
            /* Timer 2 has run down */
            ui_TimerIntCpt = ui_TimerIntCpt + 1;
            break;

        default :
            break;
    }
    b_ReceiveInterrupt = 1;
}
```

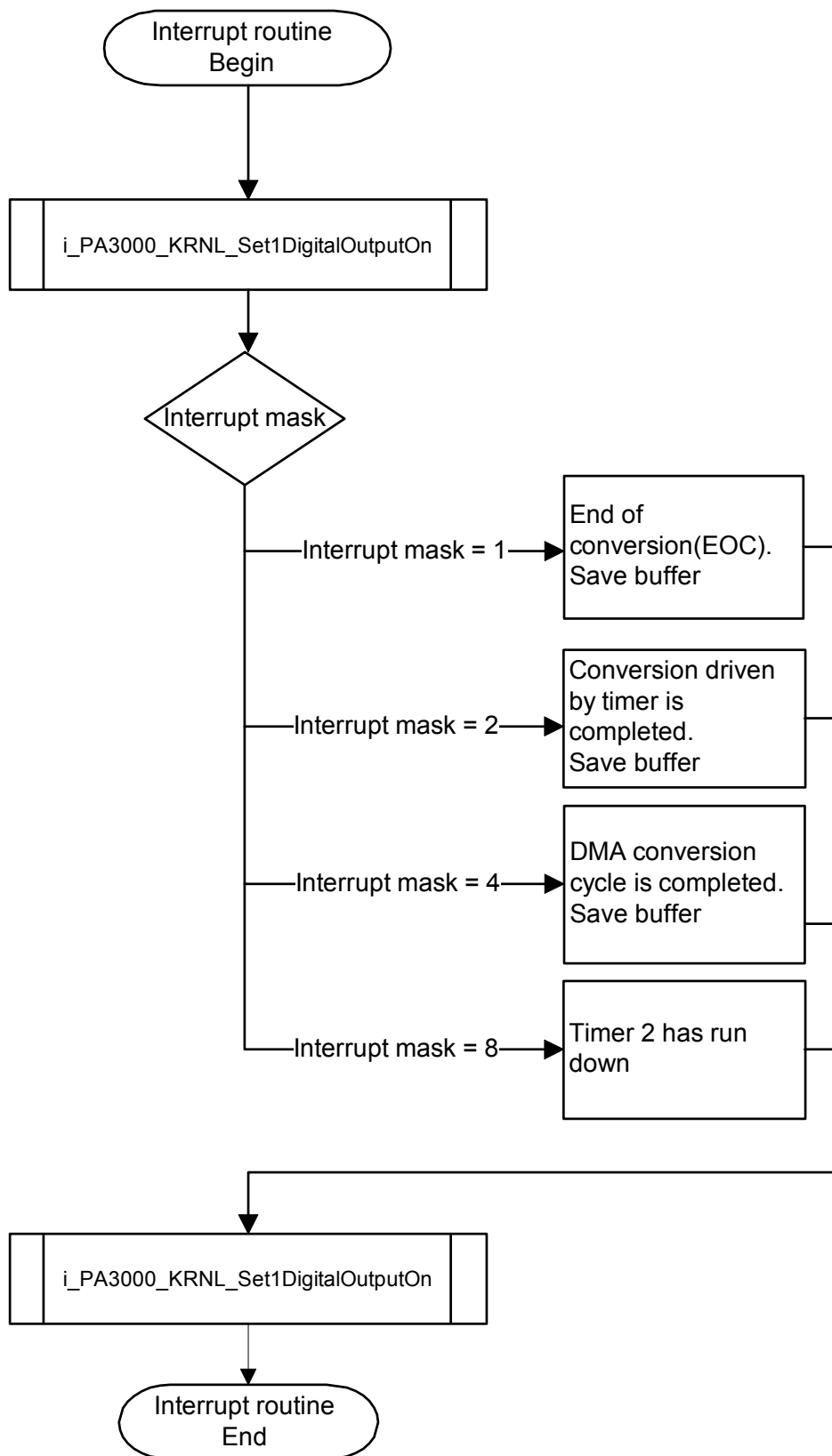
**c) Example in C for Windows NT / 95 (asynchronous mode)**

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

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



## e) Example in C for Windows NT / 95 (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 = (unsigned short int *) pui_ValueArray;

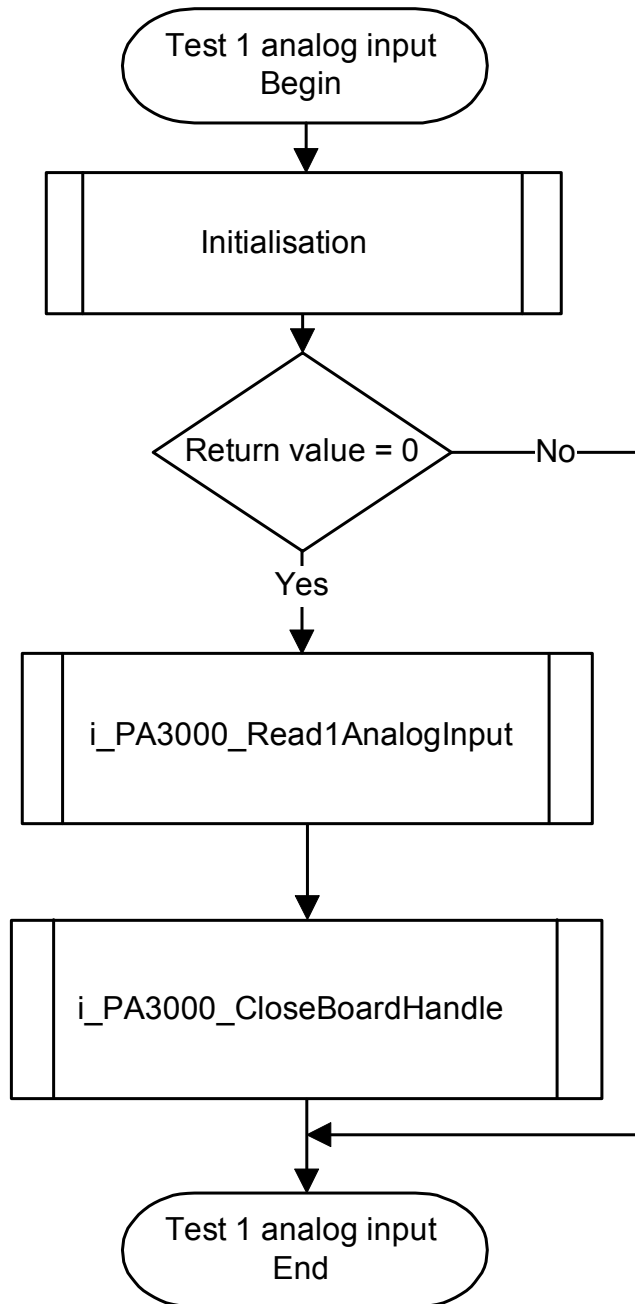
    i_PA3000_KRNL_Set1DigitalOutputOn(0x390,1);
    if ((b_InterruptMask&1) == 1) /* EOC interrupt */
    {
        ps_UserStruct->ui_SaveArray[0] = pui_ValueArray[1];
    }
    if ((b_InterruptMask&2) == 2) /* EOS interrupt Acquisition */
    {
        ps_UserStruct->ui_SaveArray[0] = pui_ValueArray[1];
    }
    if ((b_InterruptMask&4) == 4) /* DMA completed */
    {
        for (ui_Cpt=0;ui_Cpt<16;ui_Cpt++)
            ps_UserStruct->ui_SaveArray [ui_Cpt] = pusi_Index[ui_Cpt];
    }
    if ((b_InterruptMask&8) == 8)
    {
        /* Timer 2 has run down */
        ps_UserStruct->ui_TimerIntCpt = ps_UserStruct->ui_TimerIntCpt + 1;
    }
    i_PA3000_KRNL_Set1DigitalOutputOn(0x390,2);
    ps_UserStruct->b_ReceiveInterrupt =ps_UserStruct->b_ReceiveInterrupt + 1;
}

```

### 8.3 Direct conversion of analog input channels

#### 8.3.1 Testing one analog input channel

a) Flow chart



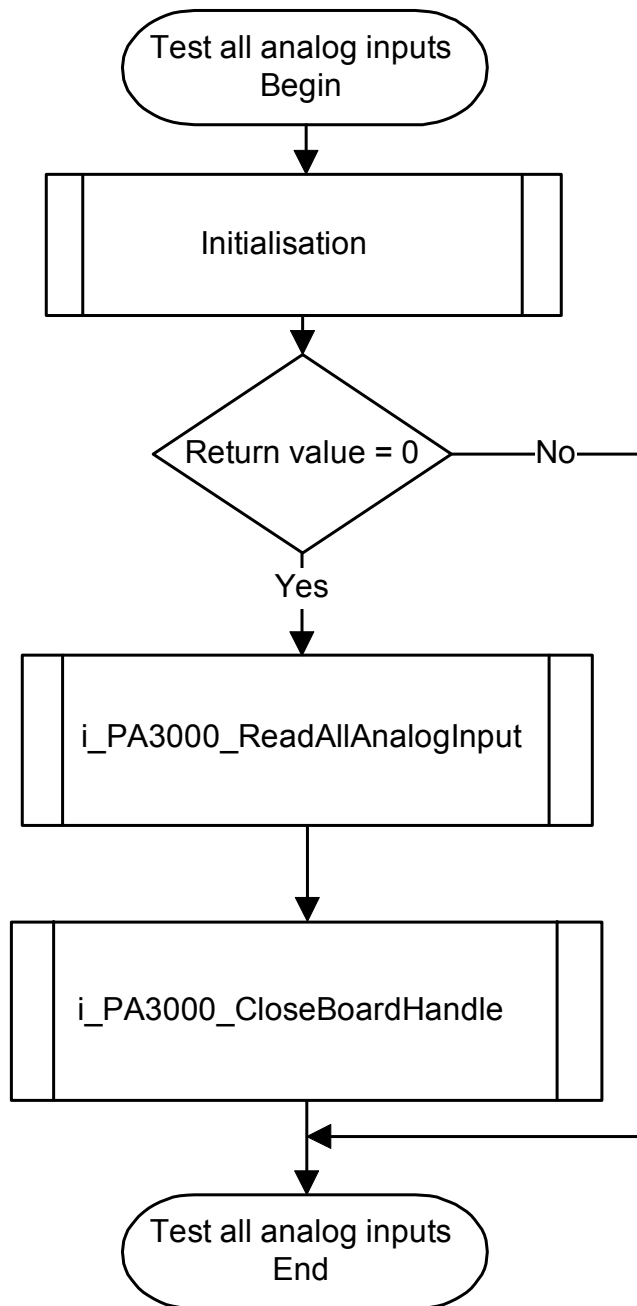
**b) Example in C**

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

    if (Initialisation (&b_BoardHandle) == 0)
    {
        if (i_PA3000_ReadAnalogInput (b_BoardHandle,
                                     PA3000_CHANNEL_0,
                                     PA3000_1_GAIN,
                                     PA3000_UNIPOLAR,
                                     10,
                                     PA3000_DISABLE,
                                     &ui_ReadValue) == 0)
        {
            printf ("ui_ReadValue = %u", ui_ReadValue);
        }
        else
        {
            printf ("Read value error");
        }
        i_PA3000_CloseBoardHandle (b_BoardHandle);
    }
    else
    {
        printf ("Initialisation error");
    }
}
```

### 8.3.2 Testing all analog input channels

a) Flow chart



**b) Example in C**

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

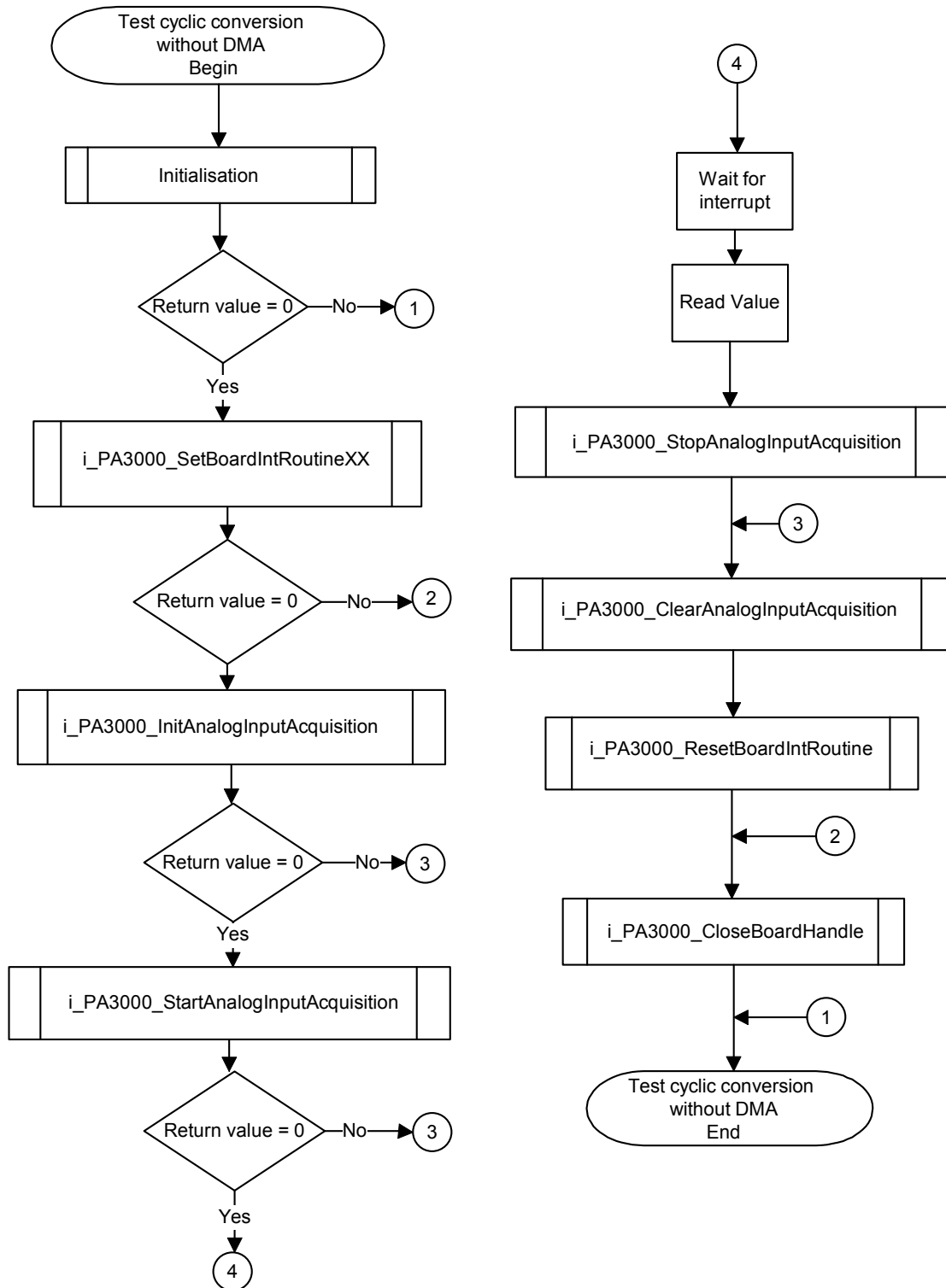
    b_Gain = PA3000_1_GAIN;
    b_Polar = PA3000_UNIPOLAR;

    if (Initialisation (&b_BoardHandle) == 0)
    {
        if (i_PA3000_ReadMoreAnalogInput (b_BoardHandle, b_Gain,
                                          b_Polar, 10, 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_PA3000_CloseBoardHandle (b_BoardHandle);
    }
    else
    {
        printf ("Initialisation error");
    }
}
```

## 8.4 Cyclic conversion of the analog inputs

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

#### a) Flow chart



## b) 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_PA3000_SetBoardIntRoutineDos(b_BoardHandle, v_InterruptRoutine) == 0)
        {
            b_Channel[0] = PA3000_CHANNEL_0;b_Gain[0] = PA3000_1_GAIN;b_Polar[0] = PA3000_UNIPOLAR;
            b_Channel[1] = PA3000_CHANNEL_1;b_Gain[1] = PA3000_1_GAIN;b_Polar[1] = PA3000_UNIPOLAR;
            b_Channel[2] = PA3000_CHANNEL_2;b_Gain[2] = PA3000_1_GAIN;b_Polar[2] = PA3000_UNIPOLAR;
            b_Channel[3] = PA3000_CHANNEL_3;b_Gain[3] = PA3000_1_GAIN;b_Polar[3] = PA3000_UNIPOLAR;
            if (i_PA3000_InitAnalogInputAcquisition
                (b_BoardHandle,4,b_Channel,b_Gain,b_Polar,
                 PA3000_SIMPLE_MODUS,PA3000_DISABLE,1500,
                 0,4,PA3000_DMA_NOT_USED,PA3000_SINGLE)==0)
            {
                b_ReceiveInterrupt = 0;
                if (i_PA3000_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_PA3000_StopAnalogInputAcquisition(b_BoardHandle);
                }
                else
                    printf("\n Start acquisition error");
                i_PA3000_ClearAnalogInputAcquisition(b_BoardHandle);
            }
            else
                printf("\n Acquisition initialisation error");
            i_PA3000_ResetBoardIntRoutine(b_BoardHandle);
        }
        else
            printf("\n Interrupt routine initialisation error");
        i_PA3000_CloseBoardHandle(b_BoardHandle);
    }
    else
        printf("\n Initialisation error");
}

```

## c) 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_PA3000_SetBoardIntRoutineWin16(b_BoardHandle, v_InterruptRoutine) == 0)
        {
            b_Channel[0] = PA3000_CHANNEL_0;b_Gain[0] = PA3000_1_GAIN;b_Polar[0] = PA3000_UNIPOLAR;
            b_Channel[1] = PA3000_CHANNEL_1;b_Gain[1] = PA3000_1_GAIN;b_Polar[1] = PA3000_UNIPOLAR;
            b_Channel[2] = PA3000_CHANNEL_2;b_Gain[2] = PA3000_1_GAIN;b_Polar[2] = PA3000_UNIPOLAR;
            b_Channel[3] = PA3000_CHANNEL_3;b_Gain[3] = PA3000_1_GAIN;b_Polar[3] = PA3000_UNIPOLAR;
            if (i_PA3000_InitAnalogInputAcquisition (b_BoardHandle,4,b_Channel,b_Gain,b_Polar,
                PA3000_SIMPLE_MODUS,PA3000_DISABLE,1500, 0,4,PA3000_DMA_NOT_USED,PA3000_SINGLE)==0)
            {
                b_ReceiveInterrupt = 0;
                if (i_PA3000_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_PA3000_StopAnalogInputAcquisition(b_BoardHandle);
                }
                else
                    printf("\n Start acquisition error");
                i_PA3000_ClearAnalogInputAcquisition(b_BoardHandle);
            }
            else
                printf("\n Acquisition initialisation error");
            i_PA3000_ResetBoardIntRoutine(b_BoardHandle);
        }
        else
            printf("\n Interrupt routine initialisation error");
        i_PA3000_CloseBoardHandle(b_BoardHandle);
    }
    else
        printf("\n Initialisation error");
}

```

**d) Example in C for Windows NT / 95 (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_PA3000_SetBoardIntRoutineWin32(b_BoardHandle,ASYNCHRONOUS_MODE,0,NULL,v_InterruptR
outine) == 0)
        {
            b_Channel[0] = PA3000_CHANNEL_0;b_Gain[0] = PA3000_1_GAIN;b_Polar[0] = PA3000_UNIPOLAR;
            b_Channel[1] = PA3000_CHANNEL_1;b_Gain[1] = PA3000_1_GAIN;b_Polar[1] = PA3000_UNIPOLAR;
            b_Channel[2] = PA3000_CHANNEL_2;b_Gain[2] = PA3000_1_GAIN;b_Polar[2] = PA3000_UNIPOLAR;
            b_Channel[3] = PA3000_CHANNEL_3;b_Gain[3] = PA3000_1_GAIN;b_Polar[3] = PA3000_UNIPOLAR;
            if (i_PA3000_InitAnalogInputAcquisition (b_BoardHandle,4,b_Channel,b_Gain,b_Polar,
                PA3000_SIMPLE_MODUS,PA3000_DISABLE,1500,
                0,4,PA3000_DMA_NOT_USED,PA3000_SINGLE)==0)
            {
                b_ReceiveInterrupt = 0;
                if (i_PA3000_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_PA3000_StopAnalogInputAcquisition(b_BoardHandle);
                }
                else
                    printf("\n Start acquisition error");
                i_PA3000_ClearAnalogInputAcquisition(b_BoardHandle);
            }
            else
                printf("\n Acquisition initialisation error");
            i_PA3000_ResetBoardIntRoutine(b_BoardHandle);
        }
        else
            printf("\n Interrupt routine initialisation error");
        i_PA3000_CloseBoardHandle(b_BoardHandle);
    }
    else
        printf("\n Initialisation error");
}

```

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

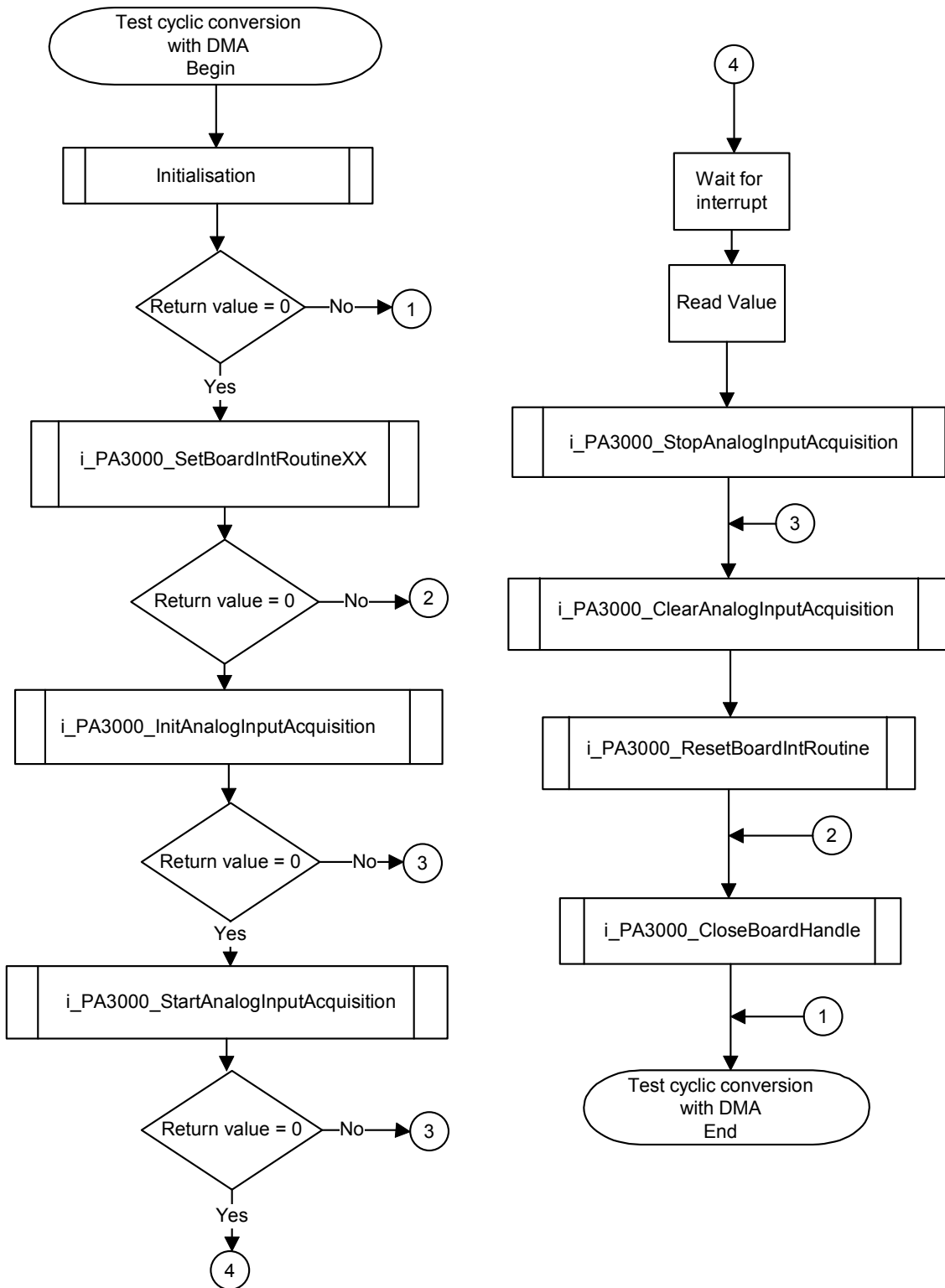
```

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_PA3000_SetBoardIntRoutineWin32(b_BoardHandle, SYNCHRONOUS_MODE, sizeof(str_UserStruct)
(void **) &ps_GlobalUserStruct, v_InterruptRoutine) == 0)
        {
            b_Channel[0] = PA3000_CHANNEL_0; b_Gain[0] = PA3000_1_GAIN; b_Polar[0] = PA3000_UNIPOLAR;
            b_Channel[1] = PA3000_CHANNEL_1; b_Gain[1] = PA3000_1_GAIN; b_Polar[1] = PA3000_UNIPOLAR;
            b_Channel[2] = PA3000_CHANNEL_2; b_Gain[2] = PA3000_1_GAIN; b_Polar[2] = PA3000_UNIPOLAR;
            b_Channel[3] = PA3000_CHANNEL_3; b_Gain[3] = PA3000_1_GAIN; b_Polar[3] = PA3000_UNIPOLAR;
            if (i_PA3000_InitAnalogInputAcquisition (b_BoardHandle,4,b_Channel,b_Gain,b_Polar,
                PA3000_SIMPLE_MODUS,PA3000_DISABLE,1500,
                0,4,PA3000_DMA_NOT_USED,PA3000_SINGLE)==0)
            {
                ps_GlobalUserStruct -> b_ReceiveInterrupt = 0;
                if (i_PA3000_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_PA3000_StopAnalogInputAcquisition(b_BoardHandle);
                }
                else
                    printf("\n Start acquisition error");
                i_PA3000_ClearAnalogInputAcquisition(b_BoardHandle);
            }
            else
                printf("\n Acquisition initialisation error");
            i_PA3000_ResetBoardIntRoutine(b_BoardHandle);
        }
        else
            printf("\n Interrupt routine initialisation error");
        i_PA3000_CloseBoardHandle(b_BoardHandle);
    }
    else
        printf("\n Initialisation error");
}

```

### 8.4.2 Cyclic conversion with DMA without external trigger and delay

a) Flow chart



**b) 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_PA3000_SetBoardIntRoutineDos(b_BoardHandle, v_InterruptRoutine) == 0)
        {
            b_Channel[0] = PA3000_CHANNEL_0;b_Gain[0] = PA3000_1_GAIN;b_Polar[0] = PA3000_UNIPOLAR;
            b_Channel[1] = PA3000_CHANNEL_1;b_Gain[1] = PA3000_1_GAIN;b_Polar[1] = PA3000_UNIPOLAR;
            b_Channel[2] = PA3000_CHANNEL_2;b_Gain[2] = PA3000_1_GAIN;b_Polar[2] = PA3000_UNIPOLAR;
            b_Channel[3] = PA3000_CHANNEL_3;b_Gain[3] = PA3000_1_GAIN;b_Polar[3] = PA3000_UNIPOLAR;
            if (i_PA3000_InitAnalogInputAcquisition (b_BoardHandle,4,b_Channel,b_Gain,b_Polar,
                PA3000_SIMPLE_MODUS,PA3000_DISABLE,1500,0,16,PA3000_DMA_USED,PA3000_SINGLE) == 0)
            {
                b_ReceiveInterrupt = 0;
                if (i_PA3000_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_PA3000_StopAnalogInputAcquisition(b_BoardHandle);
                    i_PA3000_ClearAnalogInputAcquisition(b_BoardHandle);
                }
                else
                    printf("\n Start acquisition error");
            }
            else
                printf("\n Acquisition initialisation error");
            i_PA3000_ResetBoardIntRoutine(b_BoardHandle);
        }
        else
            printf("\n Interrupt routine initialisation error");
        i_PA3000_CloseBoardHandle(b_BoardHandle);
    }
    else
        printf("\n Initialisation error");
}

```

## c) 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_PA3000_SetBoardIntRoutineWin16(b_BoardHandle, v_InterruptRoutine) == 0)
        {
            b_Channel[0] = PA3000_CHANNEL_0;b_Gain[0] = PA3000_1_GAIN;b_Polar[0] = PA3000_UNIPOLAR;
            b_Channel[1] = PA3000_CHANNEL_1;b_Gain[1] = PA3000_1_GAIN;b_Polar[1] = PA3000_UNIPOLAR;
            b_Channel[2] = PA3000_CHANNEL_2;b_Gain[2] = PA3000_1_GAIN;b_Polar[2] = PA3000_UNIPOLAR;
            b_Channel[3] = PA3000_CHANNEL_3;b_Gain[3] = PA3000_1_GAIN;b_Polar[3] = PA3000_UNIPOLAR;
            if (i_PA3000_InitAnalogInputAcquisition (b_BoardHandle,4,b_Channel,b_Gain,b_Polar,
                PA3000_SIMPLE_MODUS,PA3000_DISABLE,1500,0,16,PA3000_DMA_USED,PA3000_SINGLE) == 0)
            {
                b_ReceiveInterrupt = 0;
                if (i_PA3000_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_PA3000_StopAnalogInputAcquisition(b_BoardHandle);
                    i_PA3000_ClearAnalogInputAcquisition(b_BoardHandle);
                }
                else
                    printf("\n Start acquisition error");
            }
            else
                printf("\n Acquisition initialisation error");
            i_PA3000_ResetBoardIntRoutine(b_BoardHandle);
        }
        else
            printf("\n Interrupt routine initialisation error");
        i_PA3000_CloseBoardHandle(b_BoardHandle);
    }
    else
        printf("\n Initialisation error");
}

```

d) Example in C for Windows NT / 95 (aynchronous 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_PA3000_SetBoardIntRoutineWin32(b_BoardHandleASYNCHRONOUS_MODE,0,NULL,
            v_InterruptRoutine) == 0)
        {
            b_Channel[0] = PA3000_CHANNEL_0;b_Gain[0] = PA3000_1_GAIN;b_Polar[0] = PA3000_UNIPOLAR;
            b_Channel[1] = PA3000_CHANNEL_1;b_Gain[1] = PA3000_1_GAIN;b_Polar[1] = PA3000_UNIPOLAR;
            b_Channel[2] = PA3000_CHANNEL_2;b_Gain[2] = PA3000_1_GAIN;b_Polar[2] = PA3000_UNIPOLAR;
            b_Channel[3] = PA3000_CHANNEL_3;b_Gain[3] = PA3000_1_GAIN;b_Polar[3] = PA3000_UNIPOLAR;
            if (i_PA3000_InitAnalogInputAcquisition (b_BoardHandle,4,b_Channel,b_Gain,b_Polar,
                PA3000_SIMPLE_MODUS,PA3000_DISABLE,1500,0,16,PA3000_DMA_USED,PA3000_SINGLE) == 0)
            {
                b_ReceiveInterrupt = 0;
                if (i_PA3000_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_PA3000_StopAnalogInputAcquisition(b_BoardHandle);
                    i_PA3000_ClearAnalogInputAcquisition(b_BoardHandle);
                }
                else
                    printf("\n Start acquisition error");
            }
            else
                printf("\n Acquisition initialisation error");
            i_PA3000_ResetBoardIntRoutine(b_BoardHandle);
        }
        else
            printf("\n Interrupt routine initialisation error");
        i_PA3000_CloseBoardHandle(b_BoardHandle);
    }
    else
        printf("\n Initialisation error");
}

```

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

```

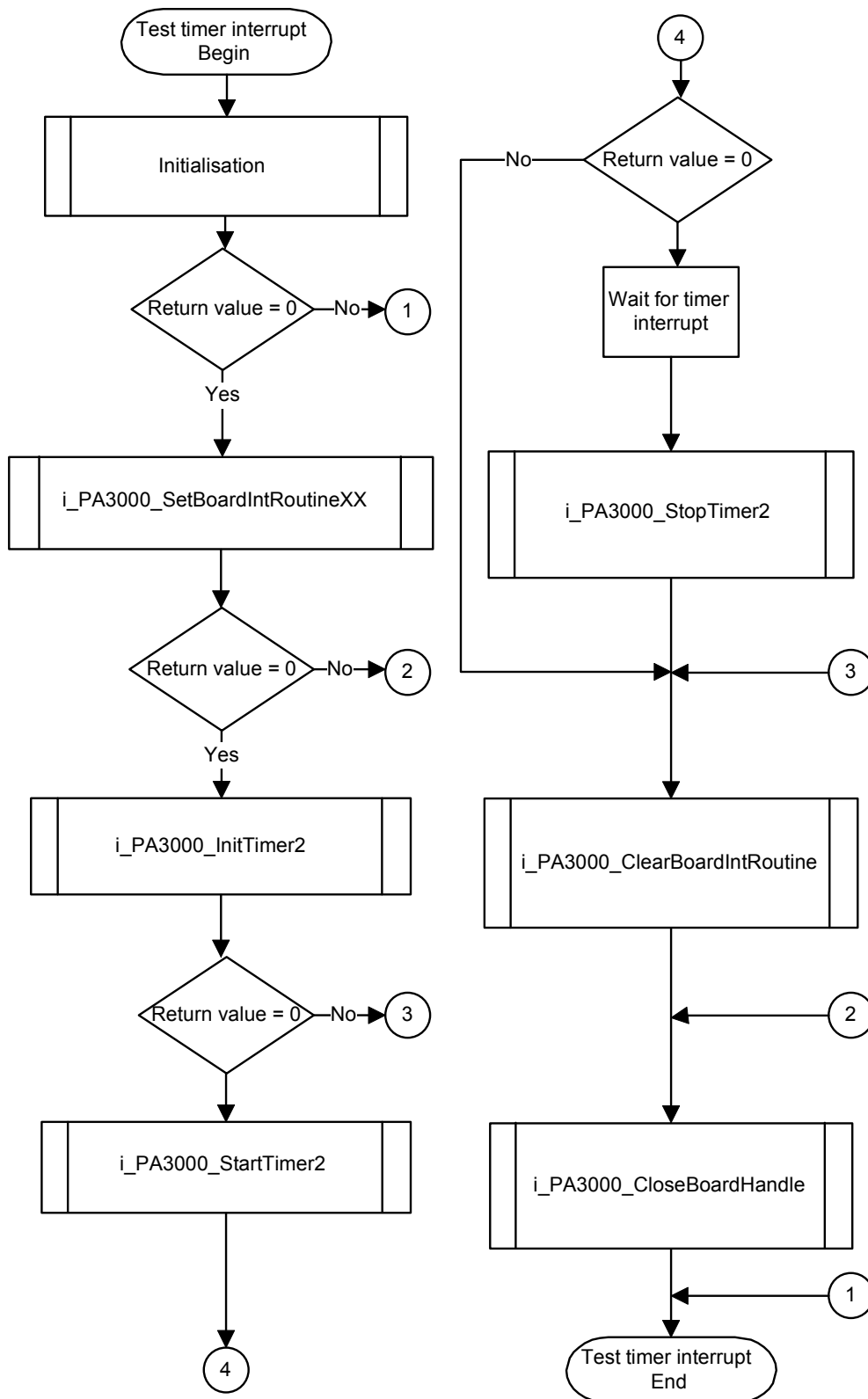
void main(void)
{
    int i_Cpt; unsigned char b_Gain [4]; unsigned char b_Polar [4]; unsigned char b_Channel
[4];
    unsigned char b_BoardHandle;
    if (Initialisation(&b_BoardHandle) == 0)
    {
        if
(i_PA3000_SetBoardIntRoutineWin32(b_BoardHandle, SYNCHRONOUS_MODE, sizeof(str_UserStruct),
        (void **) &GlobalUserStruct, v_InterruptRoutine) == 0)
        {
            b_Channel[0] = PA3000_CHANNEL_0; b_Gain[0] = PA3000_1_GAIN; b_Polar[0] = PA3000_UNIPOLAR;
            b_Channel[1] = PA3000_CHANNEL_1; b_Gain[1] = PA3000_1_GAIN; b_Polar[1] = PA3000_UNIPOLAR;
            b_Channel[2] = PA3000_CHANNEL_2; b_Gain[2] = PA3000_1_GAIN; b_Polar[2] = PA3000_UNIPOLAR;
            b_Channel[3] = PA3000_CHANNEL_3; b_Gain[3] = PA3000_1_GAIN; b_Polar[3] = PA3000_UNIPOLAR;
            if (i_PA3000_InitAnalogInputAcquisition (b_BoardHandle, 4, b_Channel, b_Gain, b_Polar,
            PA3000_SIMPLE_MODUS, PA3000_DISABLE, 1500, 0, 16, PA3000_DMA_USED, PA3000_SINGLE) == 0)
            {
                ps_GlobalUserStruct -> b_ReceiveInterrupt = 0;
                if (i_PA3000_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_PA3000_StopAnalogInputAcquisition(b_BoardHandle);
                    i_PA3000_ClearAnalogInputAcquisition(b_BoardHandle);
                }
                else
                    printf("\n Start acquisition error");
            }
            else
                printf("\n Acquisition initialisation error");
            i_PA3000_ResetBoardIntRoutine(b_BoardHandle);
        }
        else
            printf("\n Interrupt routine initialisation error");
        i_PA3000_CloseBoardHandle(b_BoardHandle);
    }
    else
        printf("\n Initialisation error");
}

```

## 8.5 Timer2

### 8.5.1 Testing 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_PA3000_SetBoardIntRoutineDOS (b_BoardHandle, v_InterruptRoutine) == 0)
        {
            ui_TimerIntCpt = 0;
            if (i_PA3000_InitTimer2 (b_BoardHandle,
                                    1000,
                                    PA3000_ENABLE) == 0)
            {
                if (i_PA3000_StartTimer2 (b_BoardHandle) == 0)
                {
                    while (ui_TimerIntCpt == 0);
                    printf ("Receive timer interrupt");
                    i_PA3000_StopTimer2 (b_BoardHandle);
                }
                else
                    printf ("Start timer error");
            }
            else
                printf ("Init timer error");
            i_PA3000_ResetBoardIntRoutine (b_BoardHandle);
        }
        else
            printf ("Interrupt routine initialisation error");
        i_PA3000_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_PA3000_SetBoardIntRoutineWin16 (b_BoardHandle, v_InterruptRoutine) == 0)
        {
            ui_TimerIntCpt = 0;
            if (i_PA3000_InitTimer2 (b_BoardHandle,
                                    1000,
                                    PA3000_ENABLE) == 0)
            {
                if (i_PA3000_StartTimer2 (b_BoardHandle) == 0)
                {
                    while (ui_TimerIntCpt == 0);
                    printf ("Receive timer interrupt");
                    i_PA3000_StopTimer2 (b_BoardHandle);
                }
                else
                {
                    printf ("Start timer error");
                }
            }
            else
            {
                printf ("Init timer error");
            }
            i_PA3000_ResetBoardIntRoutine (b_BoardHandle);
        }
        else
        {
            printf ("Interrupt routine initialisation error");
        }
        i_PA3000_CloseBoardHandle (b_BoardHandle);
    }
    else
    {
        printf ("Initialisation error");
    }
}
```

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

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

    if (Initialisation (&b_BoardHandle) == 0)
    {
        if (i_PA3000_SetBoardIntRoutineWin32 (b_BoardHandle,PA3000_ASYNCHRONOUS_MODE,
                                             0,NULL,v_InterruptRoutine) == 0)
        {
            ui_TimerIntCpt = 0;
            if (i_PA3000_InitTimer2      (b_BoardHandle,
                                         1000,
                                         PA3000_ENABLE) == 0)
            {
                if (i_PA3000_StartTimer2 (b_BoardHandle) == 0)
                {
                    while (ui_TimerIntCpt == 0);
                    printf ("Receive timer interrupt");
                    i_PA3000_StopTimer2 (b_BoardHandle);
                }
                else
                    printf ("Start timer error");
            }
            else
                printf ("Init timer error");
            i_PA3000_ResetBoardIntRoutine (b_BoardHandle);
        }
        else
            printf ("Interrupt routine initialisation error");
        i_PA3000_CloseBoardHandle (b_BoardHandle);
    }
    else
        printf ("Initialisation error");
}
```

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

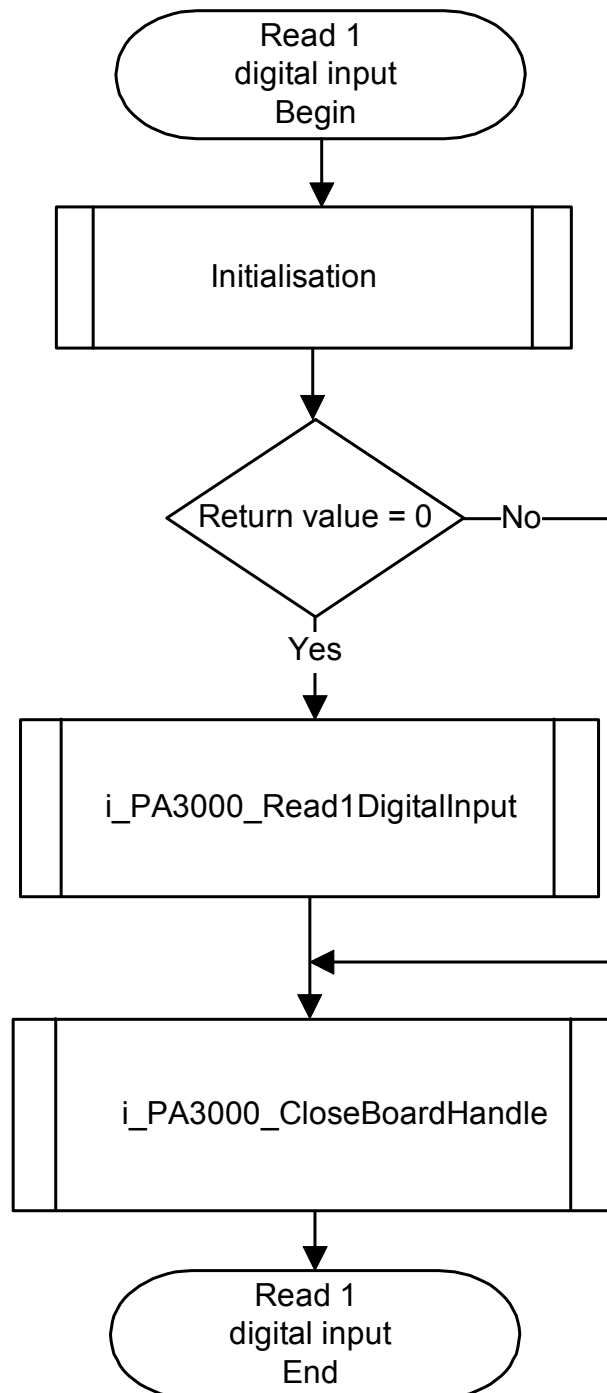
```
void main (void)
{
    unsigned char b_BoardHandle;

    if (Initialisation (&b_BoardHandle) == 0)
    {
        if (i_PA3000_SetBoardIntRoutineWin32 (b_BoardHandle,PA3000_SYNCHRONOUS_MODE,
            sizeof(str_UserStruct),(void **) &ps_GlobalUserStruct,v_InterruptRoutine) == 0)
        {
            ps_GlobalUserStruct->ui_TimerIntCpt = 0;
            if (i_PA3000_InitTimer2 (b_BoardHandle,
                1000,
                PA3000_ENABLE) == 0)
            {
                if (i_PA3000_StartTimer2 (b_BoardHandle) == 0)
                {
                    while (ps_GlobalUserStruct->ui_TimerIntCpt == 0);
                    printf ("Receive timer interrupt");
                    i_PA3000_StopTimer2 (b_BoardHandle);
                }
                else
                {
                    printf ("Start timer error");
                }
            }
            else
            {
                printf ("Init timer error");
            }
            i_PA3000_ResetBoardIntRoutine (b_BoardHandle);
        }
        else
        {
            printf ("Interrupt routine initialisation error");
        }
        i_PA3000_CloseBoardHandle (b_BoardHandle);
    }
    else
    {
        printf ("Initialisation error");
    }
}
```

## 8.6 Digital input channels

### 8.6.1 Reading one digital input channel

a) flow chart



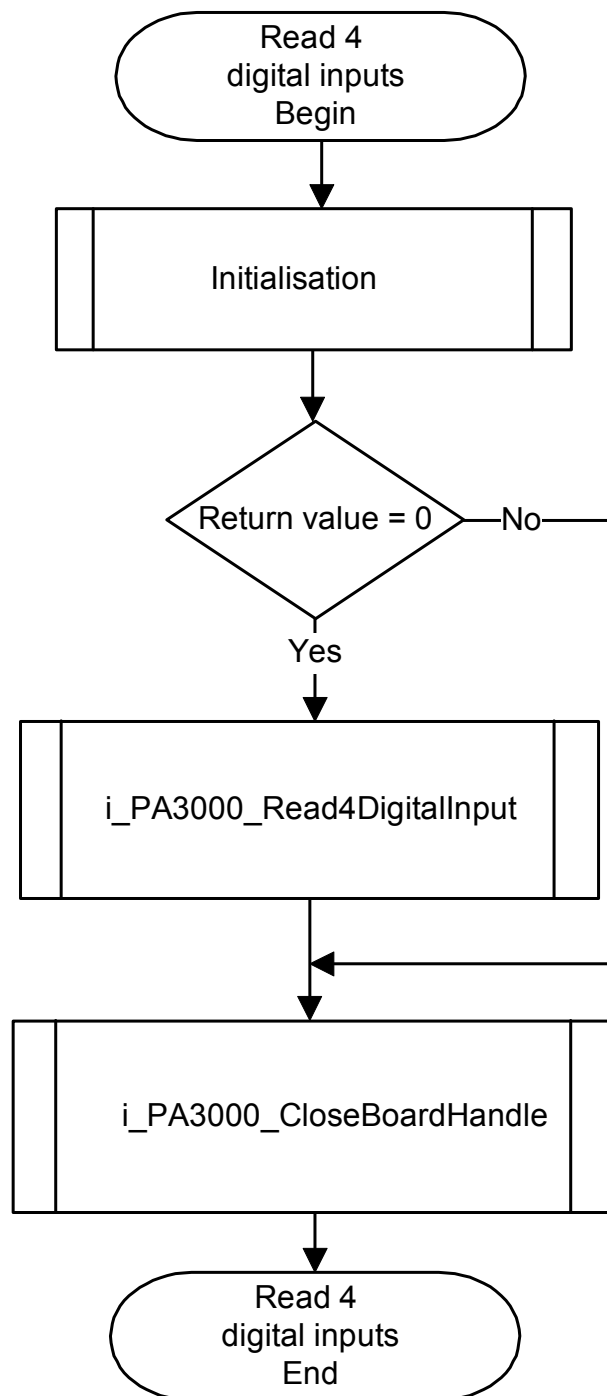
**b) Example in C**

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

    if (Initialisation (&b_BoardHandle) == 0)
    {
        if (i_PA3000_Read1DigitalInput (b_BoardHandle,
                                        1
                                        &ui_ReadValue) == 0)
        {
            printf ("ui_ReadValue = %u", ui_ReadValue);
        }
        else
        {
            printf ("Read value error");
        }
        i_PA3000_CloseBoardHandle (b_BoardHandle);
    }
    else
    {
        printf ("Initialisation error");
    }
}
```

## 8.6.2 Reading 4 digital input channels

### a) Flow chart



**b) Example in C**

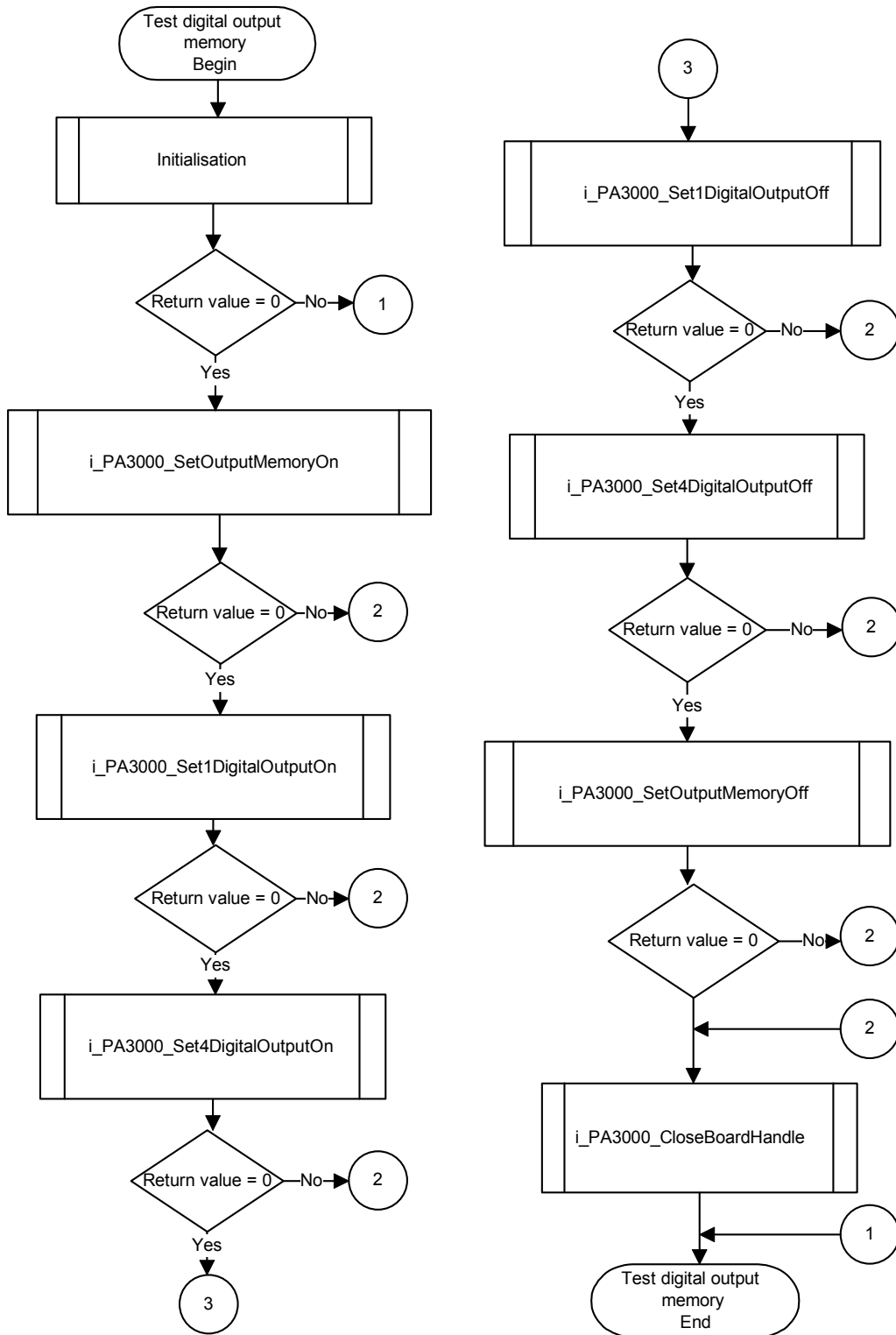
```
void main (void)
{
    unsigned char b_BoardHandle;
    unsigned int  ui_ReadValue;

    if (Initialisation (&b_BoardHandle) == 0)
    {
        if (i_PA3000_Read4DigitalInput (b_BoardHandle,
                                       &ui_ReadValue) == 0)
        {
            printf ("ui_ReadValue = %u", ui_ReadValue);
        }
        else
        {
            printf ("Read value error");
        }
        i_PA3000_CloseBoardHandle (b_BoardHandle);
    }
    else
    {
        printf ("Initialisation error");
    }
}
```

## 8.7 Digital output channels

### 8.7.1 Testing 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_PA3000_SetOutputMemory On (b_BoardHandle) == 0)
        {
            printf ("Digital output memory is activated");
            if (i_PA3000_Set1DigitalOutputOn (b_BoardHandle, 1) == 0)
            {
                printf(" Output 1 is set ");
                getch();
                if (i_PA3000_Set4DigitalOutputOn (b_Boardhandle, 14) ==0)
                {
                    printf ("All Output are set ");
                    getch();
                    if (i_PA3000_Set1DigitalOutputOff (b_Boardhandle, 1 ) == 0)
                    {
                        printf ("Output 1 is reset");
                        getch();
                        if (i_PA3000_Set4DigitalOutputOff (b_Boardhandle, 14) == 0)
                        {
                            printf ("Output 2,3,4 are reset");
                            if (i_PA3000_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_PA3000_CloseBoardHandle (b_BoardHandle);
    }
    else printf ("Initialisation error");
}
```

# INDEX

- ADDIMON
  - configuration 20
- ADDIREG 15–19
  - changing the configuration 19
  - deinstallieren 19
- addressing see base address
- base address 11
- board
  - configuration with ADDIMON 20
  - configuration with ADDIREG 15–19
  - handling 3
  - insert 13
  - options 5
  - physical set-up 4
  - secure 13
  - slot 4
  - versions 5
- component scheme 8
- connection 21–23
  - examples 23
  - peripheral 21
  - principle 22
- DIP switches 11
- installation 9–14
- intended purpose 1
- Internet
  - address 20
- jumpers
  - location 9
  - settings at delivery 9
- limit values 5–7, 5
  - energy requirements 5
- options 5
- PC
  - close 13
  - open 12
  - remove the back cover 12
- peripheral
  - connection 21
- ribbon cable
  - pin assignment 21
- slot
  - select 12
  - types, figure 12
- software
  - delivered 14
- SUB-D connector
  - pin assignment 21
- technical data 4–8
- use
  - limits 1
- versions 5