DIN EN ISO 9001:2000
certified

ADDI-DATA®

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

Technical description

PA 1000

Digital input board, optically isolated

Edition: 07.03 – 10/2006

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

CE

ADDI-DATA®

Declaration of Conformity

Document-Number/Month-Year: B-25805 / 10.1995

Manufacturer/Importer: ADDI-DATA GmbH
Dieselstraße 3
D-77833 OTTERSWEIER

Type: **PA 1000**

Product description: **Board to be inserted in an ISA slot of a PC**
32 digital inputs
Timer, counter

The above named product complies with the following European directives:

Directive 72/23/EEC of 19 February 1973 on the harmonization of the laws of Member States relating to electrical equipment designed for use within certain voltage limits.

Directive 89/336/EEC of 3 May 1989 on the approximation of the laws of the Member States relating to electromagnetic compatibility.

The following norms have been applied:

IEC 61010-1 2002-08

IEC 61326-2 2004

2004/11/10
Date

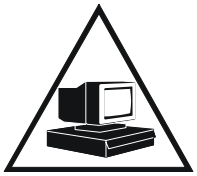
Legally valid signature of the manufacturer

ADDI-DATA GmbH ● Dieselstraße 3 ● D-77833 Ottersweier
Phone: +49 (0)7223/9493-0 ● Fax: + 49 (0)7223/9493-92

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, others and the environment★★★

- **Read the yellow safety leaflet carefully !**
  If this leaflet is not with the documentation , please contact us and ask for it.

- **Observe the instructions in the manual!**
  Make sure that you do not forget or skip any step. We are not liable for damage resulting from a wrong use of the board.

- **Symbols used**

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

IMPORTANT!
designates hints and other useful information.

- **Do you have any question?**

  Our technical support is at your disposal

Figures

Tables

1 INTENDED PURPOSE OF THE BOARD

The **PA 1000** board is the interface between an industrial process and a personal computer (PC).

The board **PA 1000** must be inserted in a PC with ISA slots, which is used as electrical equipment for measurement, control and laboratory use as definedn in the norm IEC 61010-1.

The PC is to comply with the norm IEC61326 for measurement, control and laboratory use and with the specifications for EMC protection.

Products complying with these specifications bear the CE mark.

Data exchange between the **PA 1000** board and the peripheral is to occur through a shielded cable. It has to be connected to the 37-pin SUB-D male connector of the **PA 1000** board.

The board has 32 input channels for processing digital 24 V signals.

The use of the **PA 1000** board in combination with external screw terminal panels is to occur in a closed switch cabinet; the installation is to be effected competently.

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

The connection with our standard cable ST010 complies with the specifications:
- metallized plastic hoods
- shielded cable
- cable shield folded back and firmly screwed to the cable 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 advice given in this manual and in the safety leaflet.

1.1 Limits of use

The PA 1000 board is not to be used as safety related part 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.

WARNING!
The EMC tests have been carried out in a specific appliance configuration. We guarantee these limit values **only** in this configuration

The tested appliance configuration is at your disposal on request.

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.

The installation of the board PA 1000 in sites lying under risk of explosion is excluded.

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

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

2 USER

2.1 Qualification

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

- installation,
- use,
- maintenance.

2.2 Personal protection

Consider the country-specific regulations about

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

3 HANDLING THE BOARD

Fig. 3-1: Wrong handling

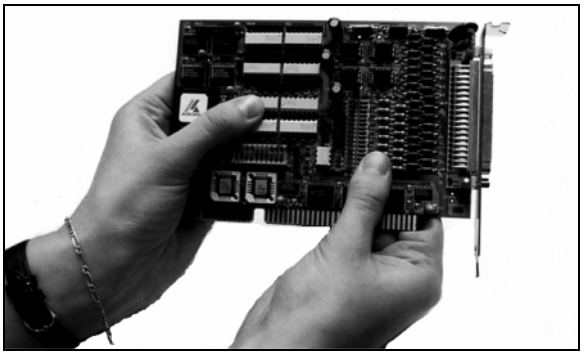
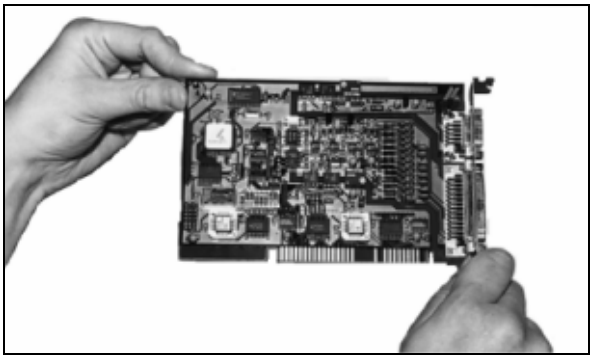


Fig. 3-2: Correct handling



4 TECHNICAL DATA

4.1 Electromagnetic compatibility (EMC)

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

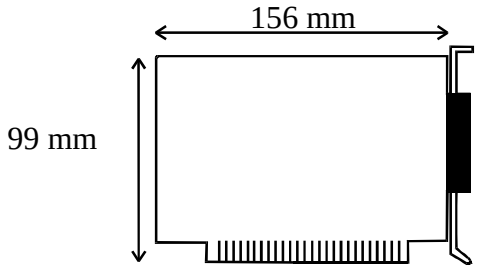
	True value	Set value
ESD	4 kV	4 kV
Fields	10 V/m	10 V/m
Burst	4 kV	2 kV
Conducted radio interferences	10 V	10 V

WARNING!
The EMC tests have been carried out in a specific appliance configuration. We guarantee these limit values **only** in this configuration[1] .

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

4.2 Physical set-up of the board

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



Weight:	160 g
Installation in:	XT / AT slot
Connection to the peripheral	37-pin SUB-D male connector Cables: standard cable **ST010**, **ST011** Screw terminal panels: **PX 901-D**, **PX 901-DG** or **PX 9000**

See Fig 7-4: Connection to screw terminal panels

[1] We transmit our appliance configuration on request.

4.3 Limit values

Operating temperature: ..... 0 to 60°C
Storage temperature: ..... -25 to 70°C
Relative humidity: ..... 30% to 95% non condensing

Minimum PC requirements:
ISA bus interface
Bus speed: ..... 8 MHz
Energy requirements:
Operating voltage: ..... 5 V ± 5%
Current consumption (+ 5 V of the PC): ..... 194 mA ± 10%

24 V digital input channels
Input type: ..... common ground in accordance with IEC1131-2
Number of input channels: 32
Interruptible input channels: 14
Interrupt lines: ..... IRQ 3, 5 for XT; IRQ 10, 11, 12, 14, 15 for AT
Nominal voltage: ..... 24 VDC
Input current at nominal voltage: ..... 6 mA
Logic input level: ..... U_H[1)] max.: 30 V, 9 mA typ.
U_H min.: 17 V, 2 mA typ.
U_L[2)] max.: 14 V, 0.7 mA typ.
U_L min.: 0 V

Logic input level for the 24 V control
(Channel 16 selectable through jumper) ..... U_H max.: 30 V, 10 mA typ
U_H min.: 17 V, 3 mA typ.
U_L max.: 14 V, 1 mA typ.
U_L min.: 0 V

Signal delay:
Input channels 1-16 ..... 70 µs (at nominal voltage)
Input channels 17-32 ..... 40 µs (at nominal voltage)
Maximum input frequency: ..... 5 kHz (at nominal voltage)

Safety
Optical isolation
(DIN VDE 0411-100): ..... 1000 V (from the PC to the external peripheral).
Counters or timers: 3

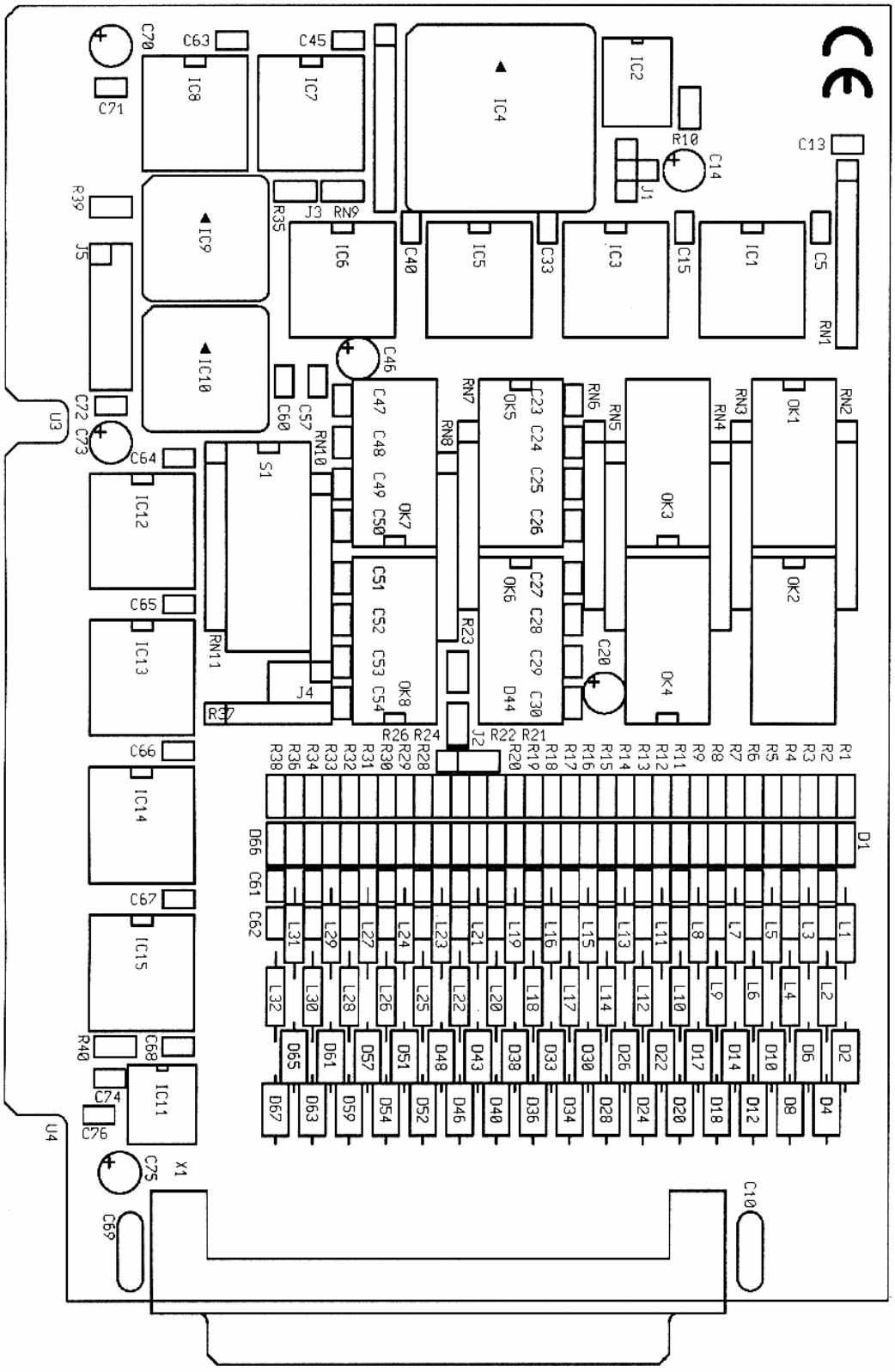
[1] U_H: input voltage which corresponds to logic "1"
[2] U_L: input voltage which corresponds to logic "0"

5 SETTINGS

5.1 Settings at delivery

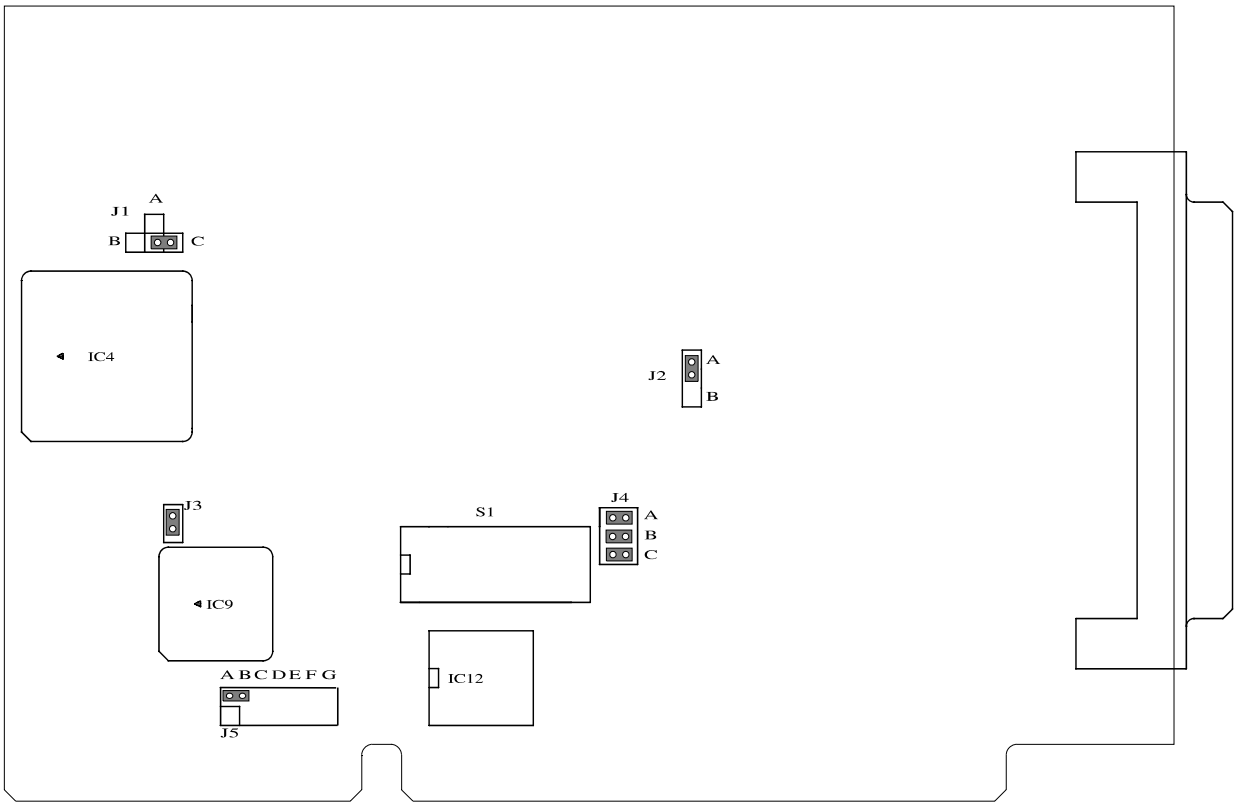
5.1.1 Component scheme

Fig. 5-1: Component scheme



5.1.2 Jumper location and settings at delivery

Fig. 5-2: Jumper location on the PA 1000 board



5.1.3 Jumper settings at delivery

IMPORTANT!
J1-A. It means that jumper J1 is set in position A .

Base address

Table 5-1: Base address

Jumper	Settings at delivery
S1	DIP switches set to 0390H.
J4 J4-A, J4-B, J4-C	Adress bits A13, A14, A15 of the address decoding logic decoded to "0"

Data bus width

Table 5-2: Data bus width

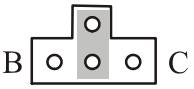
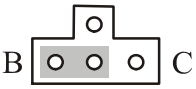
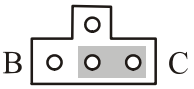
Jumper	Settings	At delivery
J3 	16-bit data bus access on the addreses **Base +0**, **Base +2**	✓
J3 	8-bit access	

Base = base address

Time base for the timers

Select the input frequnecy through jumper J1.

Table 5-3: Time base of the timers

Jumper	Settings	At delivery
J1 A B C 	3,45 kHz ± 1 %,	-
J1 A B C 	1.75 kHz ± 1 %,	-
J1 A B C 	111,5 kHz ± 1 %.	✓

Selection of the 24 V switching threshold

Select the switching threshold of input 16 for the control of the 24 V supply voltage.

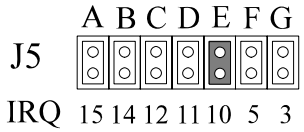
Table 5-4: Control of the 24 V supply voltage

Jumper	Settings	Delivery
J2 A B logical"1" logical"0"	 Ext. input voltage > 17 V Ext. input voltage < 15 V.	✓
J2 A B logical"1" logical"0"	 Ext. input voltage > 20 V Ext. input voltage < 18 V	-

Selection of an interrupt line to the PC bus

No interrupt line is selected at delivery. (no jumper is set)
Select a free interrupt line to the PC bus.

Fig. 5-3: Jumper field J5



e.g.:
Jumper J5-E is set
IRQ10 is selected

5.2 I/O mapping

The board **PA 1000** requires an address range of 8 I/O addresses within the I/O address range of the PC.
The functionalities of the board are accessed with a write or read command on this address range.

The decoding logic is related to the whole 64 KB I/O address range of the PC.
- **Set the base address** (beginning of the address range) with the DIP-switches S1 and the jumper field J4 in steps of 8 I/O addresses.
- **The data bus width** is determined with jumper J3. (See jumper settings)

There are therefore **two** I/O maps.

Table 5-5: I/O map for 8-bit access (J3 open)

	I/O Read	I/O Write
Base +0	CHANNEL 1-8	-
Base +1	CHANNEL 9-16	-
Base +2	CHANNEL 17-24	-
Base +3	CHANNEL 25-32	-
Base +4	Z8536_PORT_C	Z8536_PORT_C
Base +5	Z8536_PORT_B	Z8536_PORT_B
Base +6	Z8536_PORT_A	Z8536_PORT_A
Base +7	Z8536_CONTROL_REGISTER	Z8536_CONTROL_REGISTER

-: not decoded

Table 5-6: I/O map for 16-bit access (J3 is set)

	I/O Read	I/O Write
Base +0	CHANNEL 1-16	-
Base +2	CHANNEL 17-32	-
Base +4*	Z8536_PORT_C	Z8536_PORT_C
Base +5*	Z8536_PORT_B	Z8536_PORT_B
Base + 6*	Z8536_PORT_A	Z8536_PORT_A
Base +7*	Z8536_CONTROL_REGISTER	Z8536_CONTROL_REGISTER

- : not decoded
*: only 8-bit accesses possible

5.2.1 Description of the I/O map

8-bit access

The 32 digital input channels (24 V) are directly read with an I/O read command.

Ex.: Input 1 corresponds to bit D0 of the address **Base +0**
Input 16 corresponds to bit D7 of the address **Base +1**.
Input 17 corresponds to bit D0 of the address **Base +2**.
Input 32 corresponds to bit D7 of the address **Base +3**.

Table 5-7: Description of the address Base +0 (8-bit read access for the inputs 1-8)

D7							D0
Input 8	Input 7	Input 6	Input 5	Input 4	Input 3	Input 2	Input 1

Table 5-8: Description of the address Base +1 (8-bit read access for the inputs 9-16)

D7							D0
Input 16	Input 15	Input 14	Input 13	Input 12	Input 11	Input 10	Input 9

Table 5-9: Description of the address Base +2 (8-bit read access for the inputs 17-24)

D7							D0
Input 24	Input 23	Input 22	Input 21	Input 20	Input 19	Input 18	Input 17

Table 5-10: Description of the address Base +3 (8-bit read access for the inputs 25-32)

D7							D0
Output 32	Output 31	Output 30	Output 29	Output 28	Output 27	Output 26	Output 25

Addresses Base +4 to Base +7
They are intended to program the functionalities of the CIO Z8536.
Ex.: counter or timer, (time dependent interrupt request)
An interrupt is generated with events

The accesses occur on these addresses **only** in 8-bit data bus width (independently from the setting of jumper J3).

16-bit access

The 32 digital input channels (24V) are read directly with a read command.

Ex.: Input 1 corresponds to bit D0 of the address **Base +0**
Input 16 corresponds to bit D15 of the address **Base +0**.
Input 17 corresponds to bit D0 of the address **Base +2**.
Input 32 corresponds to bit D15 of the address **Base +2**.

Table 5-11: Description of the address Base + 0 (16-bit read access for the inputs 1-16)

D15	D14					D1	D0
Input 16	Input 15	------	------	------	------	Input 2	Input 1

Table 5-12: Description of the address Base + 2 (16-bit read access for the inputs 17-32)

D15	D14					D1	D0
Output 32	Output 31	------	------	------	------	Output 18	Output 17

6 INSTALLATION

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.

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 Setting the base address through DIP switches

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

At delivery the base address is set to the address 0390H.

Before installing the board
Check, that
- the base address is free
- the address range required by the board is not already used by the PC or by boards already installed in the PC.

If the base address or the address range **are wrong**
• **Select** another base address with the 10-pin DIP switches S1 and the J4.

Decoding the base address
The base address is decoded in steps of each time 8 I/O addresses.
The base address can be selected between 0100H and 0FFFFH within the PC I/O address space.

In table 6-1 the address 0390H is decoded. (Settings at delivery).

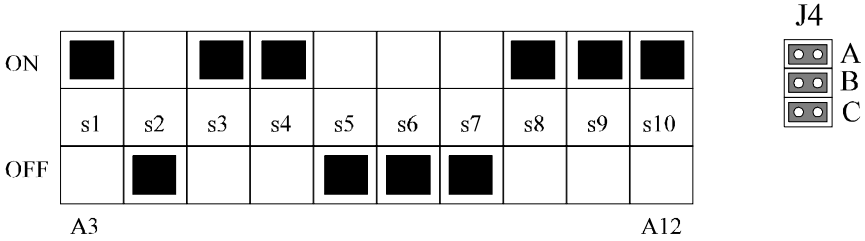
Table 6-1: Decoding table

	MSB															LSB
Decoded address bus	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
Hex base address to be set	0				3				9				0			
Binary base address to be set	0	0	0	0	0	0	1	1	1	0	0	1	0	0	0	0
DIP switches S1 Logic "0"= ON Logic "1" = OFF	–	–	–	s10 ON	s9 ON	s8 ON	s7 OFF	s6 OFF	s5 OFF	s4 ON	s3 ON	s2 OFF	s1 ON	X	X	X
Jumper field J4 ON = jumper set OFF = jumper open	J4-C ON	J4-B ON	J4-A ON	_	_	_	_	_	_	_	_	_	_	X	X	X

X : Decoded address range of the board (8 I/O addresses)
_ : Not selectable through this component

Fig. 6-1: DIP switches S1 (Address 0390H)

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



6.2 Inserting the board

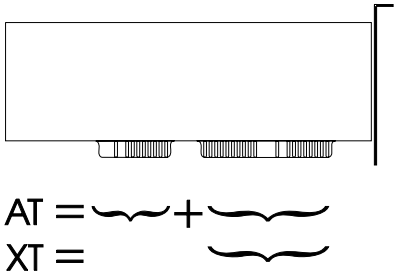
IMPORTANT!
Do observe the *safety instructions.*

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.

1. Select a free ISA slot

Fig. 6-2: Types of slots



The board can be inserted either in a slot XT or AT. It can also be inserted in EISA slots with respect of certain conditions.

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.

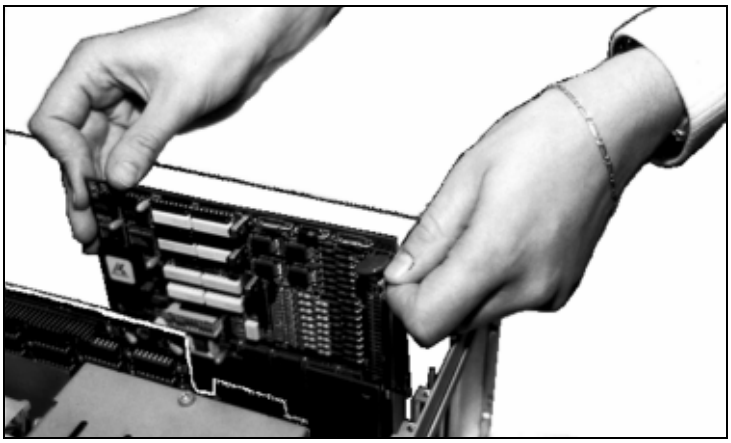
3. Discharge yourself from electrostatic charges

4. Take the board from its protective pack.

6.2.2 Plugging the board into the slot

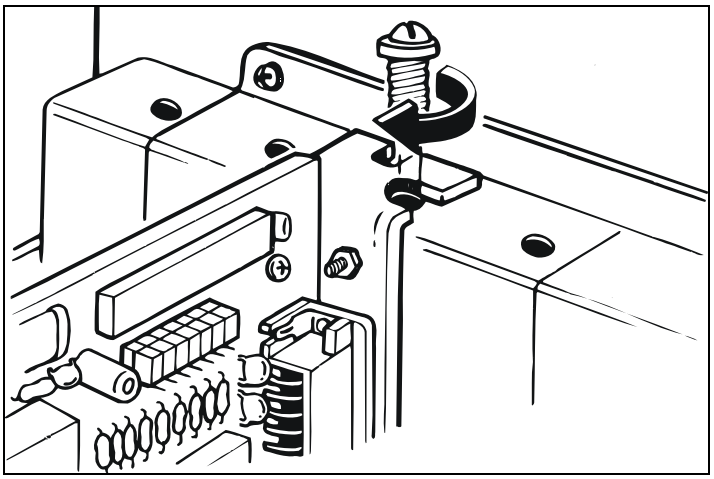
- **Discharge yourself from electrostatic charges**
- Insert the board **vertically into the chosen slot.**

Fig. 6-4: Inserting the board



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

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



- **Tigthen all loosen screws.**

6.2.3 Closing the PC

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

6.3 Installing the 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 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 (CD1) containing
- the driver and software samples for Windows NT 4.0 and Windows 2000/98/95,
- the ADDIREG registration program for Windows NT 4.0 and Windows 2000/98/95/NT.

6.4 Board configuration with ADDIREG

The ADDIREG registration program is a 32-bit program for Windows NT 4.0/ 95/98/2000.
The user can register all hardware information necessary to operate the ADDI-DATA PC boards.

IMPORTANT!
If you use one or several resources of the board, you cannot start the ADDIREG program.

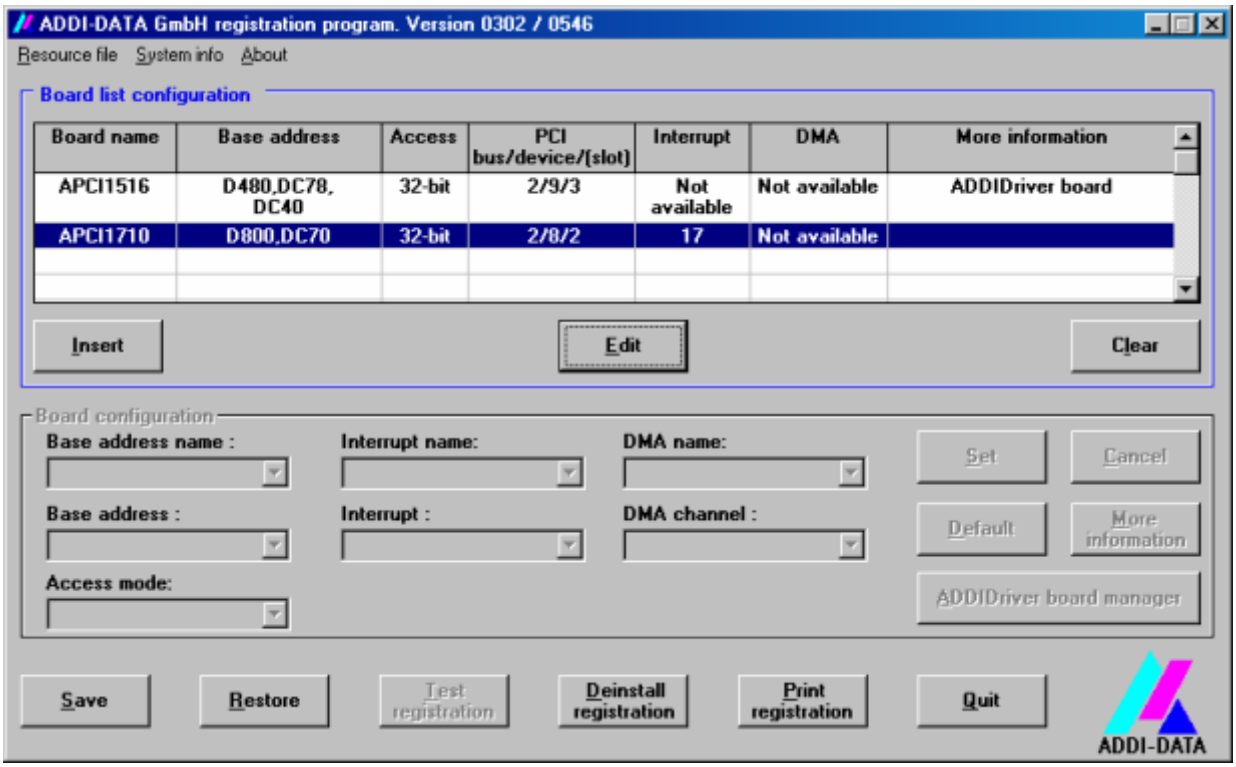
6.4.1 Program description

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

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

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

Fig. 6-6: ADDIREG registration program



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

Board name:
Names of the different registered boards (eg.: APCI-3120).
When you start the program for the first time, no board is registered in this table.

Base address:
Selected base address of the board.

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.

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 uses.

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.

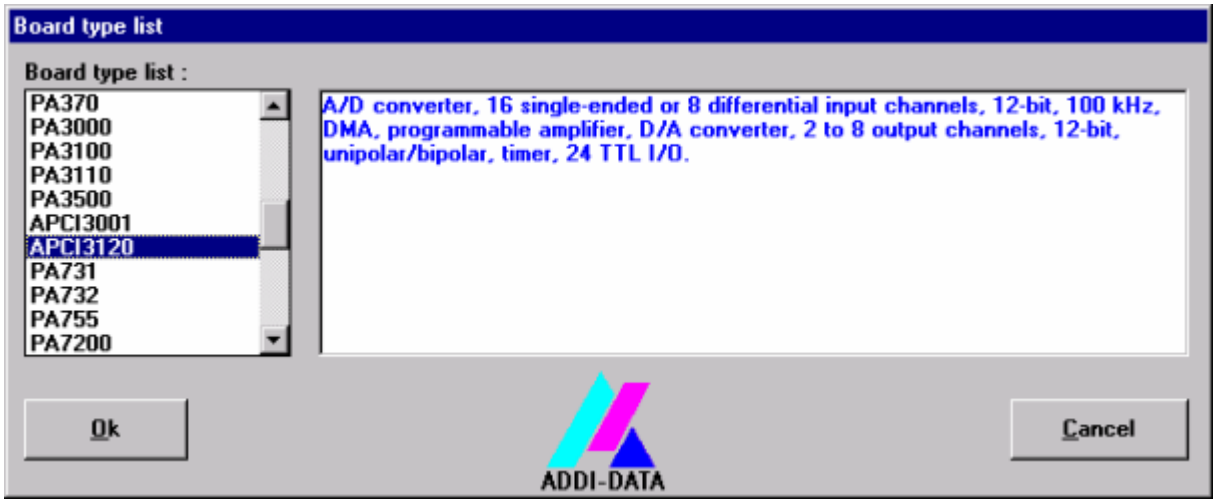
Buttons:

Edit [1]:
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:

[1] "x": Keyboard shortcuts; e.g. "Alt + e" for Edit

Fig. 6-7: Configuring a new board



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 parameterized 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 identifier string, the COM number, the operating mode of a communication board, etc...

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

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

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 6-6 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

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.

6.4.5 Software downloads from the Internet

You can download the latest version of the device driver for the PA1000 board**:**

http://www.addi-data.com

If you have any questions, do not hesitate to send us an e-mail

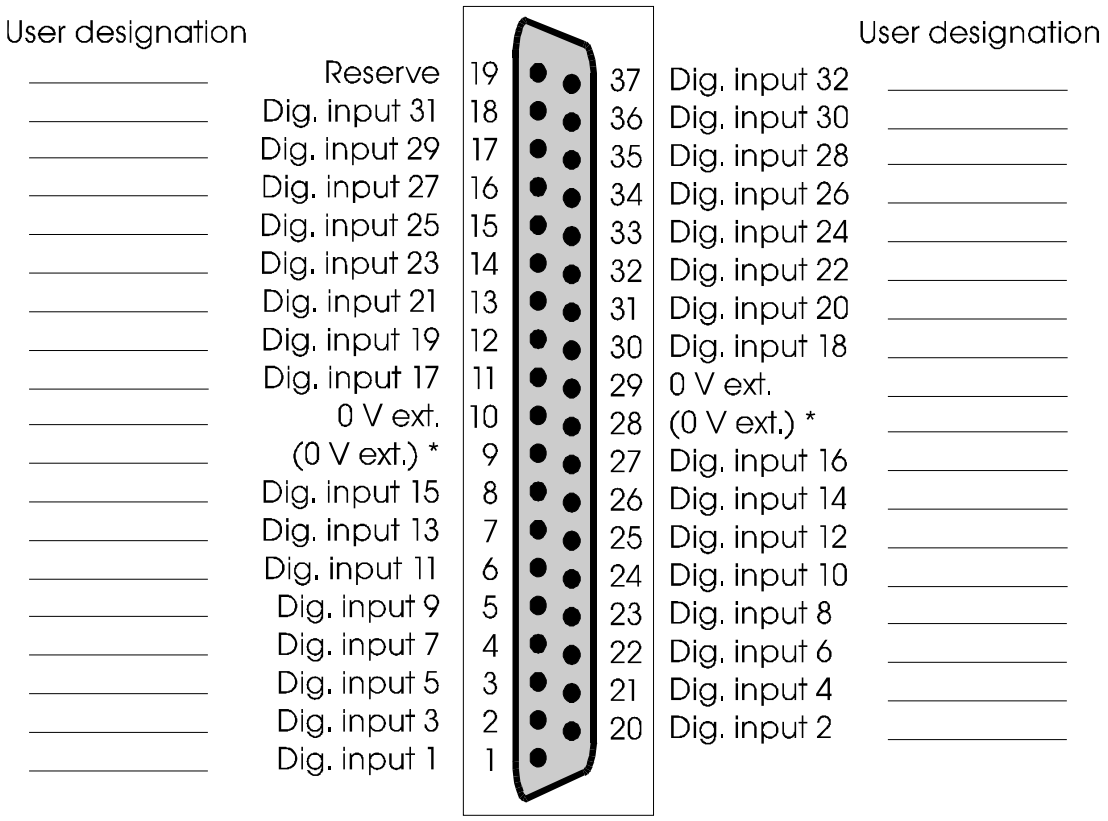
info@addi-data.de or
hotline@addi-data.com

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.

7 CONNECTION TO THE PERIPHERAL

7.1 Connector pin assignment

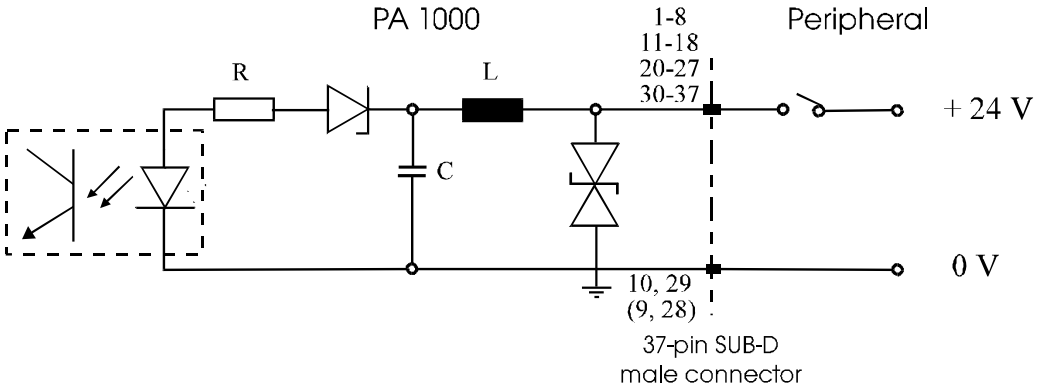
Fig. 7-1: 37-pin SUB-D male connector



* No signal connected at delivery
Intended for current return lines

7.2 Connection principle

Fig. 7-2: Connection principle



7.3 Connection examples

Fig. 7-3: Connection examples

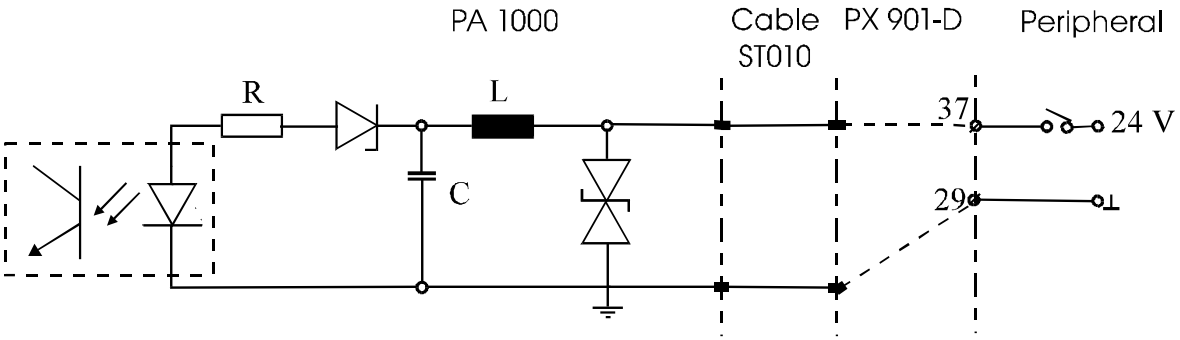
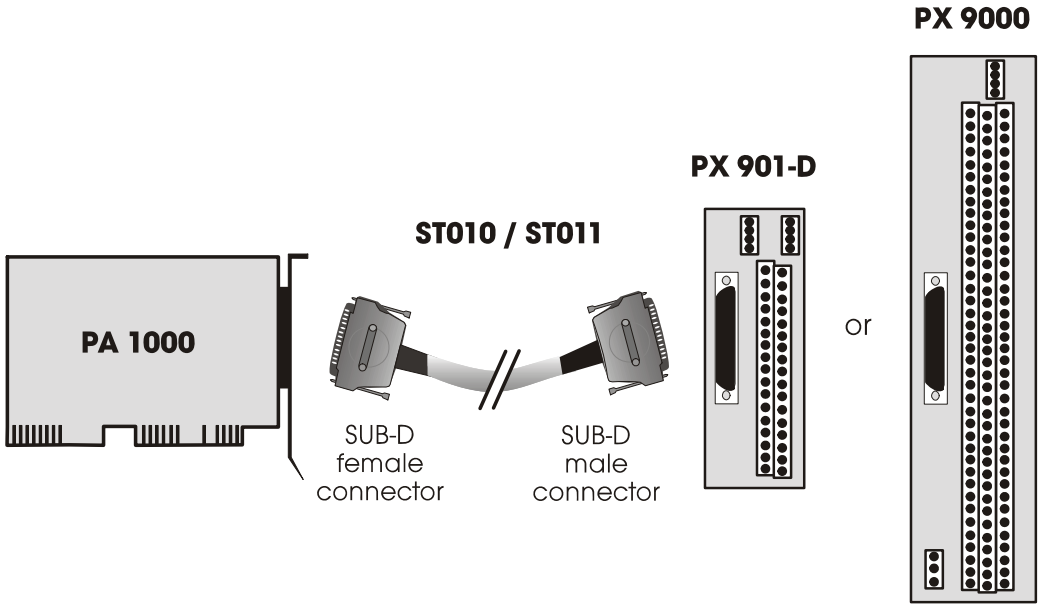


Fig. 7-4: Connection to the screw terminal panels PX 901 and PX 9000



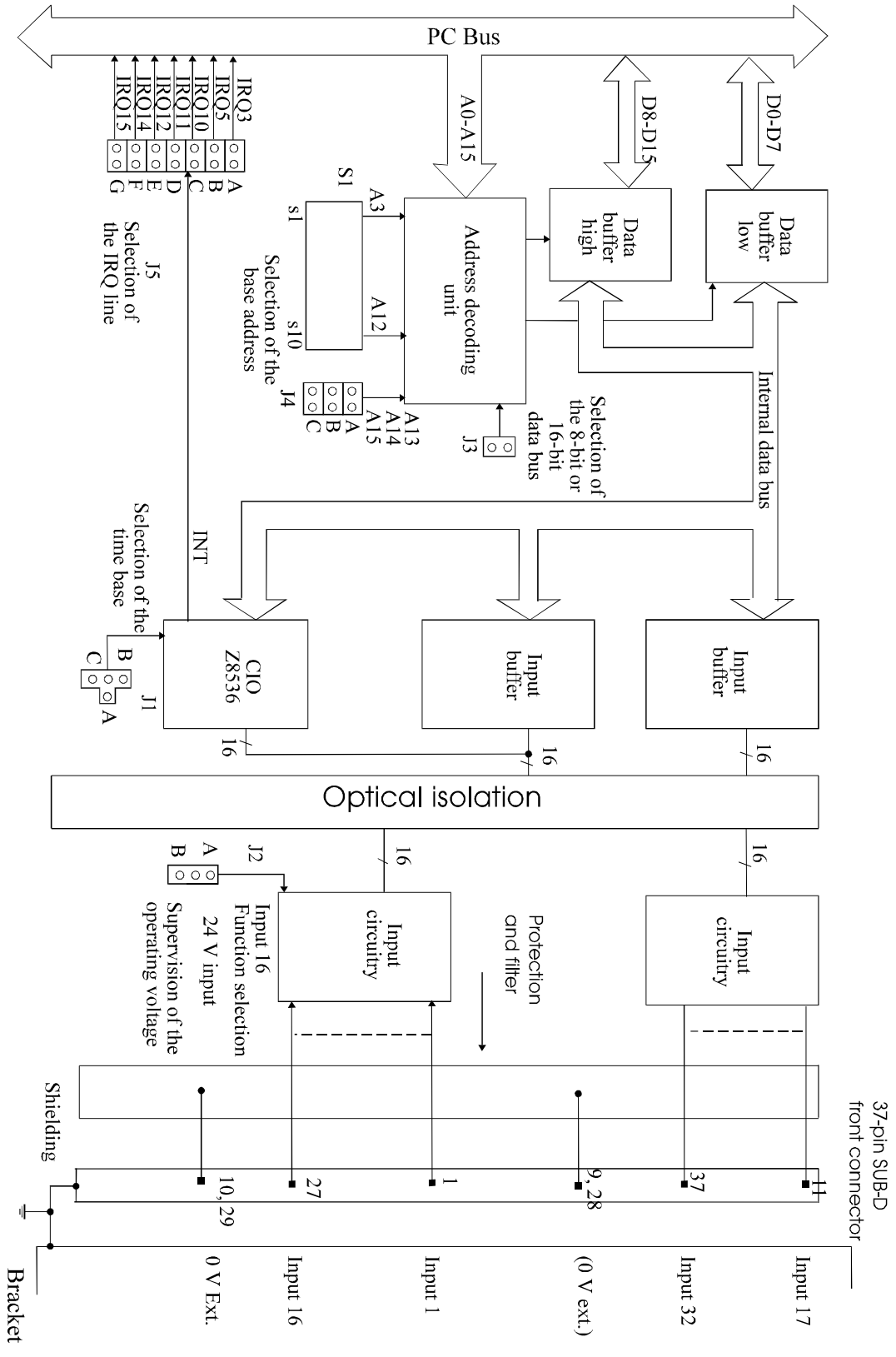
Our technical support is at your disposal for further information about our cables and screw terminal panels.

8 FUNCTIONS

8.1 Board description

8.1.1 Block diagram

Fig. 8-1: Block diagram of the PA 1000



8.1.2 Board description

32 optoisolated inputs
They comply with the 24 V industry standard (IEC1131-2)
- logic "1" corresponds to an input voltage > 17 V
- logic "0" corresponds to an input voltage < 15 V.

Input 16
special input for supervising the operating voltage:
- logic "1" corresponds to an input voltage > 20 V
- logic "0" corresponds to an input voltage < 18 V

All the inputs have two / four [1)] common current return lines: 0 V EXT (inputs): Pin 10, 29 (9, 28) of the 37-pin SUB-D male connector.

Each input needs 6 mA with a nominal voltage of 24 V.

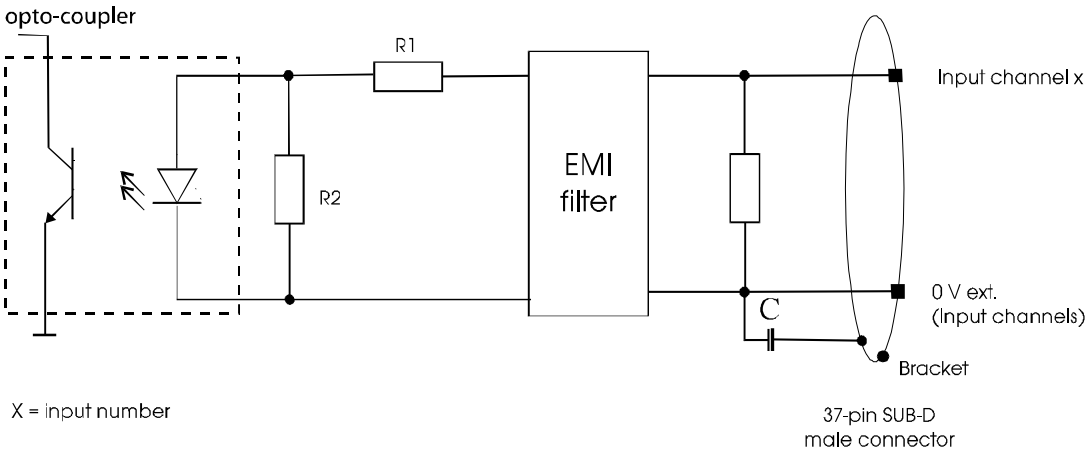
WARNING!
If you operate all the inputs with the same voltage supply, the voltage supply must be able to deliver at least 32 x 6 = 192 mA at 24 V.

The maximum input voltage is 30 V / 9.2 mA typically.

Transorb diodes, Z diodes, LC filters and optical couplers filter noise from the peripheral to the system bus.
The effects of inductive and capacitive noise are thus reduced.

The board requires no initialisation to read directly the 24 V digital information. After successful power ON reset data is immediately available on the board.

Fig. 8-2: Input circuitry



[1] Pin 9 and 28: no signal is connected at delivery. See connector pin assignment

Reading in 8-bit access

Four addresses are available in the I/O address range:
- **Base +0** for the inputs 1 to 8
- **Base +1** for the inputs 9 to 16.
- **Base +2** for the inputs 17 to 24.
- **Base +3** for the inputs 25 to 32.

Example with the DEBUG program under DOS

```
c:> DEBUG (CR)
- i 390 (CR)          (* Read the inputs 1-8                *)
00                    (* All the inputs are on logic "0"     *)
- i 391 (CR)          (* Read the inputs 9-16               *)
00                    (* All the inputs are on logic "0"     *)
- i 392 (CR)          (* Read the inputs 17-24              *)
00                    (* All the inputs are on logic "0"     *)
- i 393 (CR)          (* Read the inputs 25-32              *)
00                    (* All the inputs are on logic "0"     *)
- q                   (* Quit the DEBUG program             *)
```

Example in BASIC

```
A = INP(&H390);       (* Read the inputs 1-8                *)
B = INP(&H391);       (* Read the inputs 9-16               *)
C = INP(&H392);       (* Read the inputs 17-24              *)
D = INP(&H393);       (* Read the inputs 25-32              *)
                      (* Test input 4                       *)
IF (A and &H08) THEN PRINT "INPUT4 = 1" ELSE
PRINT "INPUT4  = 0 "
```

Reading in 16-bit access

• **Make sure that** jumper J3 is set. (See jumper settings)

Two addresses are available in the I/O address range
- **Base +0** for the inputs 1-16
- **Base +2** for the inputs 17-32.

Example in ASSEMBLER

```
.....
MOV DX, 390           ; Base address on 0390H
IN    AX, DX          ; Read the inputs 1 to 16
.....
MOV DX, 392           ; Base +2
IN    AX, DX          ; Read the inputs 17 to 32
.......
```

Example in PASCAL

Function Read_Inputs(w_Base : **WORD**) : **WORD**;

begin
(* Read the inputs 1-16 in 16-bit data bus width w_Base := $390 *)
Read_Inputs_16 := portw[w_Base];
(* Read the inputs 17-32 in 16-bit data bus width w_Base := $390 *)
Read_Inputs_32 := portw[w_Base + 2];

end;

Special functions of the digital input channels

1) Interrupt

The input channels 1 to 16 can trigger an interrupt.

Inputs 1 to 8:
It is possible to declare an **OR** or **AND** event which then generates an interrupt.

Inputs 9 to 16:
It is possible to declare an **OR** event which then generates an interrupt.

2) Counter

Counter 1: **Input 14** signal input
Input 16 can be used as a gate function
Input 15 can be used as a trigger function.

Counter 2: **Input 10** signal input
Input 12 can be used as a gate function
Input 11 can be used as a trigger function.

Counter 3: **Input 18** signal input
Input 20 can be used as a gate function
Input 19 can be used as a trigger function.

8.1.3 Interrupt

The board **PA 1000** has one interrupt source.
It must be connected through the jumper field J5 to an interrupt line of the PC bus.

The following interrupt lines are available :
IRQ3, IRQ5, IRQ10, IRQ11, IRQ12, IRQ14, IRQ15.

Possible interrupt sources :
- Event 1 has occurred (inputs 1-8),
- Event 2 has occurred (inputs 9-16),
- Counter/Timer 1 has run down,
- Counter/Timer 2 has run down,
- Counter/Timer 3 has run down,

The interrupt source information are available for the user program through an interrupt routine
See API functions: i_PA1000_SetBoardIntRoutine,
i_PA1000_ResetBoardIntRoutine.

An **event** indicates a change of state (level):
- on an input channel (ex. "0"→"1" or "1"→"0")
- or on several input channels if a logic has been defined between the input channels.

Counter/timer has run down : if the counter content changes from von 1 → 0.

8.1.4 Counter timer

On the board **PA 1000** three 16-bit counters/timers are available in component Z8536 (down counter). Each counter/timer can be programmed by software.

If the component Z8536 functions as a counter, the corresponding input channels are used as follows.

1) Counter

Counter 1: **Input 14** signal input
Input 16 can be used as a gate function
Input 15 can be used as a trigger function.

Counter 2: **Input 10** signal input
Input 12 can be used as a gate function
Input 11 can be used as a trigger function.

Counter 3: **Input 18** signal input
Input 20 can be used as a gate function
Input 19 can be used as a trigger function.

2) Timer

If the component Z8536 is used as a timer, the frequency is used as a reference. You set it with jumper J1. Gate and trigger are possible through the input channels.

Gate : can be driven by software or by an input channel. The polarity of the input channel can be programmed. This gate stopps counting when it is set.

Trigger: can be driven by software or by an input channel. The polarity of the input channel can be programmed. The trigger re-loads the counter/timer with the start value when it is set.

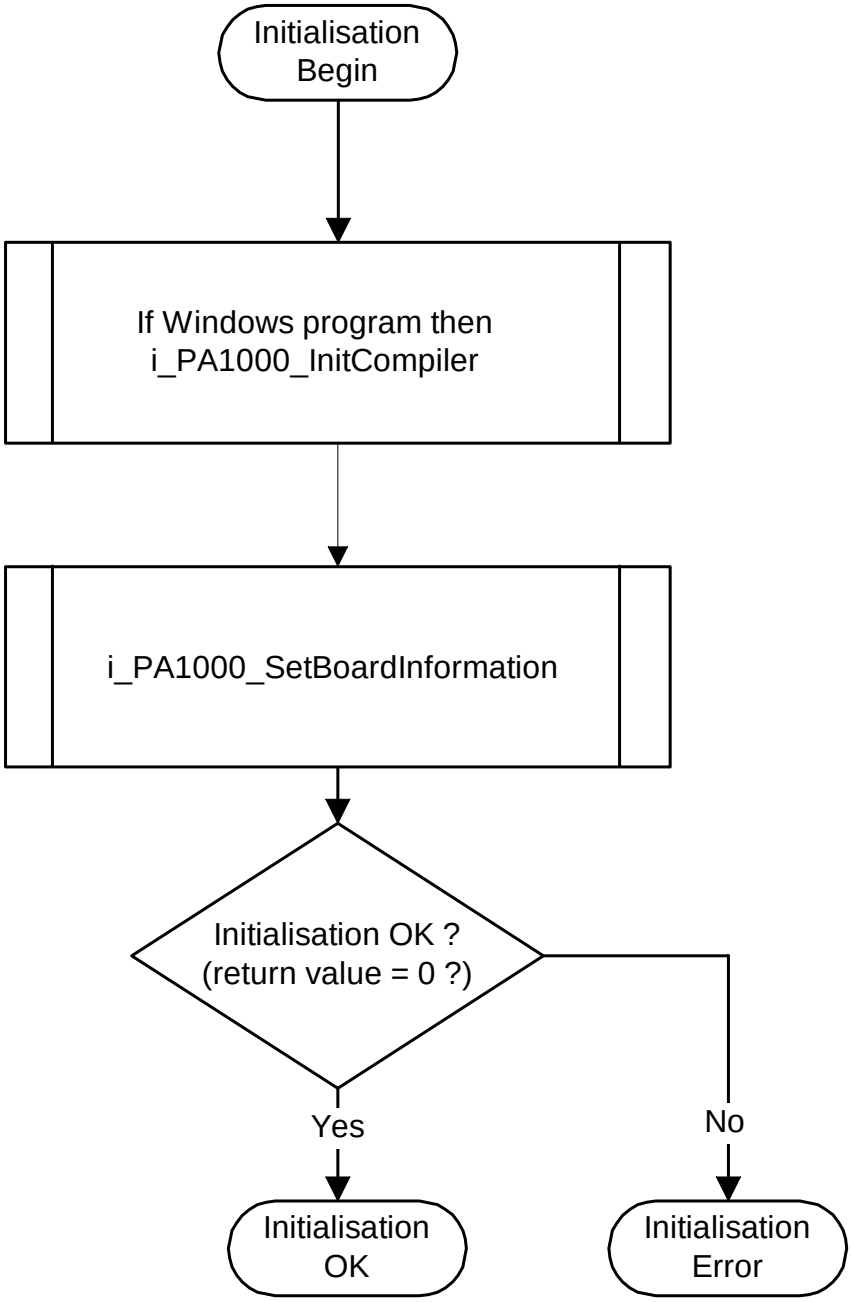
The following functionalities are available:

- Initialising the counters/timers,
- Starting the counters/timers,
- Stopping the counters/timers,
- Reading the counting value of the counters/timers

9 SOFTWARE EXAMPLES

9.1 Initialisation

a) Flow chart



b) Example in C for DOS and Windows 3.11

```
int Initialisation(unsigned char *pb_BoardHandle)
    {
    #if defined (_Windows) || defined (_WINDOWS) || defined (_WIN32)
       i_PA1000_InitCompiler (DLL_COMPILER_C);
    #endif

    if(i_PA1000_SetBoardInformation    (0x300,
                                        PA1000_16BIT,
                                        10,
                                        pb_BoardHandle) == 0)
       {
       return (0);         /* OK */
       }
    else
       {
       return (-1);        /* ERROR */
       }
    }
```

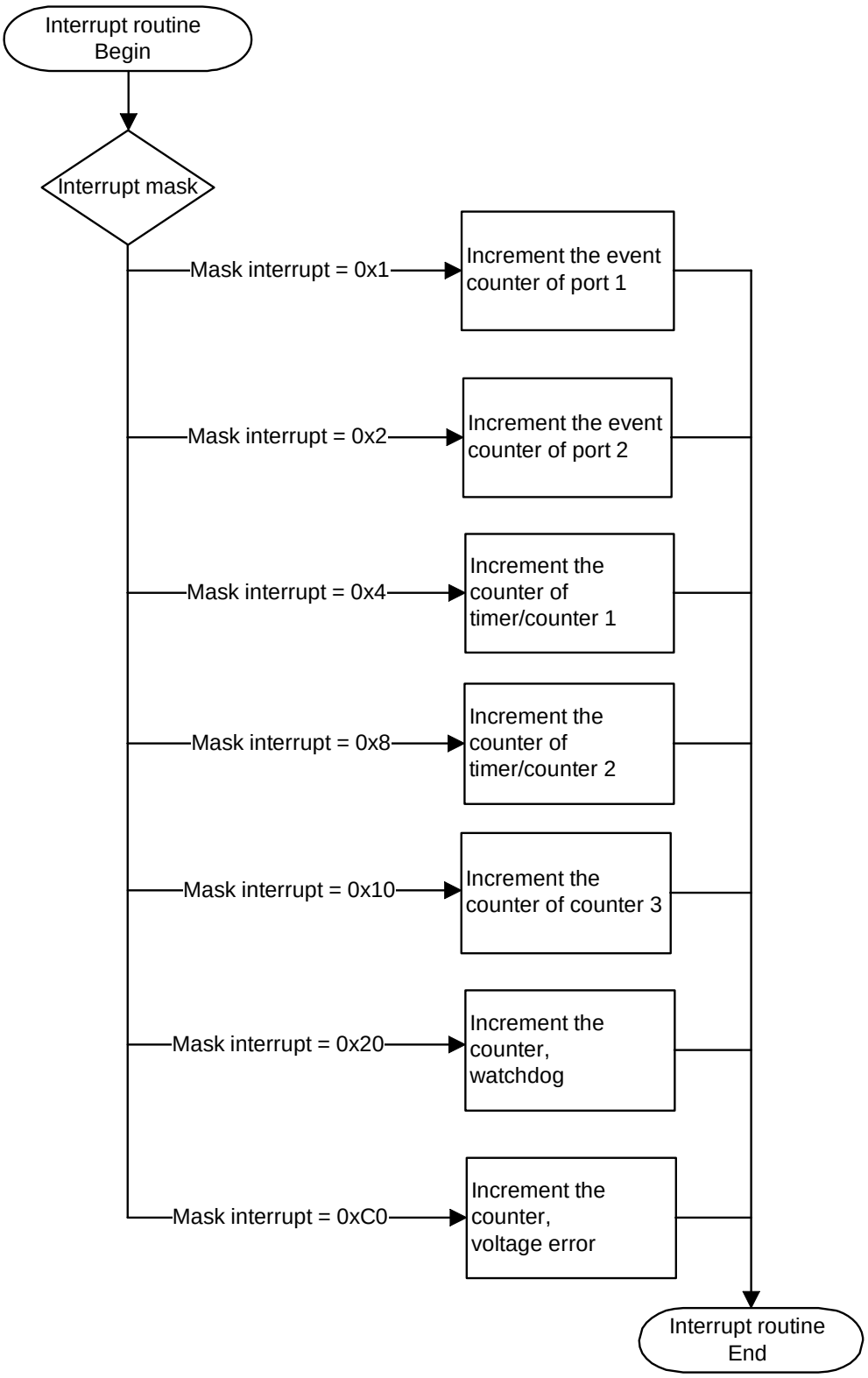
c) Example in C for Windows NT / 95

```
int Initialisation(unsigned char *pb_BoardHandle)
    {
    #if defined (_Windows) || defined (_WINDOWS) || defined (_WIN32)
       i_PA1000_InitCompiler (DLL_COMPILER_C);
    #endif

    if(i_PA1000_SetBoardInformation   (“PA1000-00”,
                                       PA1000_16BIT,
                                       Pb_BoardHandle) == 0)
       {
       return (0);         /* OK */
       }
    else
       {
       return (-1);        /* ERROR */
       }
    }
```

9.2 Interrupt

9.2.1 Interrupt routine of each example

a) Flow chart



b) Example in C for DOS / Windows 3.11

```
unsigned char b_EventCpt [2], b_TimerCounterCpt [3];       /* Global buffer    */

_VOID_ v_InterruptRoutine (unsigned char b_BoardHandle,
                           unsigned char b_InterruptMask,
                           unsigned char b_InputChannelNbr)
    {

    if (b_InterruptMask & 0x1)
       {                                                   /*************************/
       b_EventCpt [0] = b_EventCpt [0] + 1;                /* Event port 1 interrupt */
       }                                                   /*************************/

    if (b_InterruptMask & 0x2)
       {                                                   /*************************/
       b_EventCpt [1] = b_EventCpt [1] + 1;                /* Event port 2 interrupt */
       }                                                   /*************************/

    if (b_InterruptMask & 0x4)
       {                                                   /*****************************/
        b_TimerCounterCpt [0] = b_TimerCounterCpt [0] + 1; /* Timer/Counter 1 interrupt */
       }                                                   /*****************************/

    if (b_InterruptMask & 0x8)
       {                                                   /*****************************/
        b_TimerCounterCpt [1] = b_TimerCounterCpt [1] + 1; /* Timer/Counter 2 interrupt */
       }                                                   /*****************************/

    if (b_InterruptMask & 0x10)
       {                                                   /***********************/
        b_TimerCounterCpt [2] = b_TimerCounterCpt [2] + 1; /* Counter 3 interrupt */
       }                                                   /***********************/
```

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

```
Unsigned char b_EventCpt [2], b_TimerCounterCpt [3];       /* Global buffer    */

_VOID_ v_InterruptRoutine (unsigned char b_BoardHandle,
                           unsigned char b_InterruptMask,
                           unsigned char b_InputChannelNbr,
                           unsigned char b_UserCallingMode,
                           void *        pv_UserSharedMemory)
    {

    if (b_InterruptMask & 0x1)
       {                                                    /*************************/
       b_EventCpt [0] = b_EventCpt [0] + 1;                 /* Event port 1 interrupt */
       }                                                    /*************************/

    if (b_InterruptMask & 0x2)
       {                                                    /*************************/
       b_EventCpt [1] = b_EventCpt [1] + 1;                 /* Event port 2 interrupt */
       }                                                    /*************************/

    if (b_InterruptMask & 0x4)
       {                                                    /*****************************/
       b_TimerCounterCpt [0] = b_TimerCounterCpt [0] + 1;   /* Timer/Counter 1 interrupt */
       }                                                    /*****************************/

    if (b_InterruptMask & 0x8)
       {                                                    /*****************************/
       b_TimerCounterCpt [1] = b_TimerCounterCpt [1] + 1;   /* Timer/Counter 2 interrupt */
       }                                                    /*****************************/

    if (b_InterruptMask & 0x10)
       {                                                    /***********************/
       b_TimerCounterCpt [2] = b_TimerCounterCpt [2] + 1;   /* Counter 3 interrupt */
       }                                                    /***********************/
```

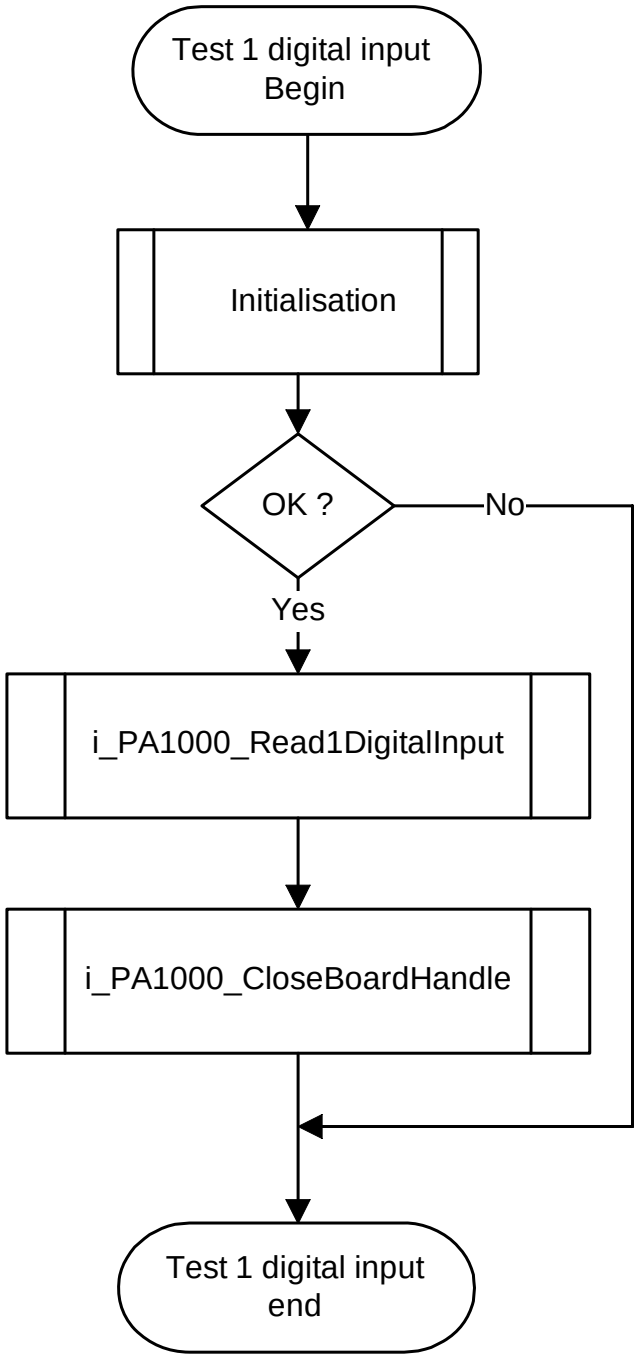
d) Example in C for Windows NT/95 (synchronous mode)

```
Typedef struct
    {
    Unsigned char b_EventCpt [2];
    Unsigned char b_TimerCounterCpt [3];
    }str_UserStruct;

str_UserStruct * ps_GlobalUserStruct;

_VOID_ v_InterruptRoutine (unsigned char b_BoardHandle,
                           unsigned char b_InterruptMask,
                           unsigned char b_InputChannelNbr,
                           unsigned char b_UserCallingMode,
                           void *        pv_UserSharedMemory)
    {
    str_UserStruct * ps_UserStruct = (str_UserStruct *) pv_UserSharedMemory;

    if (b_InterruptMask & 0x1)                              /*************************/
       {                                                    /* Event port 1 interrupt */
       ps_UserStruct ->b_EventCpt [0] =                     /*************************/
       ps_UserStruct ->b_EventCpt [0] + 1;
       }

    if (b_InterruptMask & 0x2)                              /*************************/
       {                                                    /* Event port 2 interrupt */
       ps_UserStruct ->b_EventCpt [1] =                     /*************************/
       ps_UserStruct ->b_EventCpt [1] + 1;
       }

    if (b_InterruptMask & 0x4)                              /*****************************/
       {                                                    /* Timer/Counter 1 interrupt */
       ps_UserStruct ->b_TimerCounterCpt [0] =              /*****************************/
       ps_UserStruct ->b_TimerCounterCpt [0] + 1;
       }

    if (b_InterruptMask & 0x8)                              /*****************************/
       {                                                    /* Timer/Counter 2 interrupt */
       ps_UserStruct ->b_TimerCounterCpt [1] =              /*****************************/
       ps_UserStruct ->b_TimerCounterCpt [1] + 1;
       }

    if (b_InterruptMask & 0x10)                             /*****************************/
       {                                                    /* Timer/Counter 3 interrupt */
       ps_UserStruct ->b_TimerCounterCpt [2] =              /*****************************/
       ps_UserStruct ->b_TimerCounterCpt [2] + 1;
       }
```

9.3 Digital input channels

9.3.1 Read 1 digital input

a) Flow chart

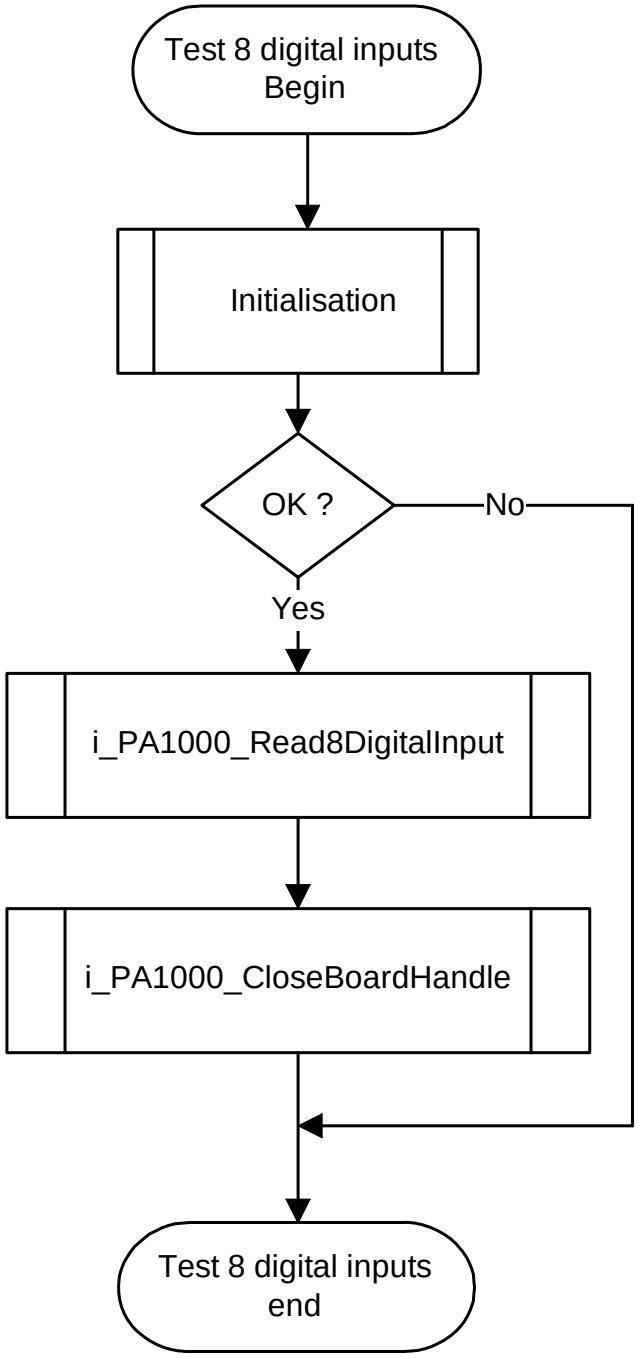


b) Example in C

```
Void    main   (void)
        {
        unsigned char b_BoardHandle;
        unsigned char b_ChannelValue;
        int           i_ReturnValue;

        if (Initialisation (&b_BoardHandle) == 0)
           {
           i_ReturnValue = i_PA1000_Read1DigitalInput (b_BoardHandle, 1, &b_ChannelValue);
           printf ("i_PA1000_Read1DigitalInput return %d", i_ReturnValue);
           printf ("Channel 1 value = %d", b_ChannelValue);

           i_ReturnValue = i_PA1000_CloseBoardHandle (b_BoardHandle);
           printf ("i_PA1000_CloseBoardHandle = %d", i_ReturnValue);
           }
        else
           {
           printf ("Initialisation error");
           }
        }
```

9.3.2 Read 8 digital inputs

a) Flow chart

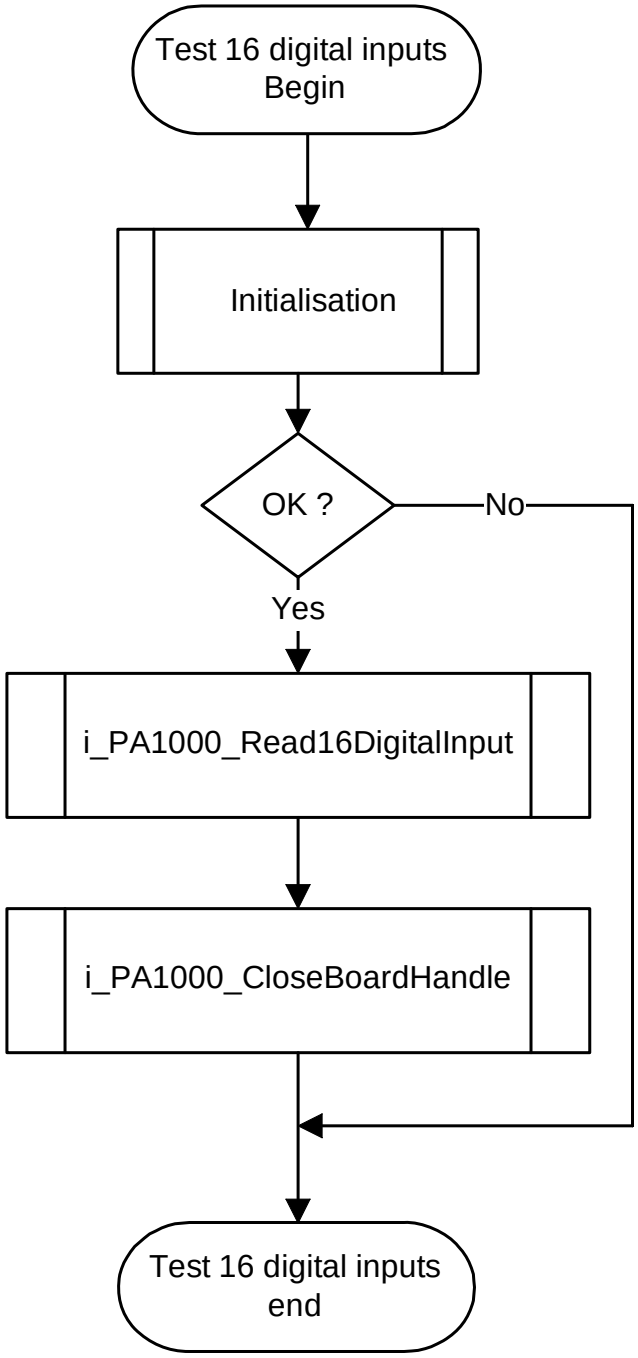


b) Example in C

```
Void    main   (void)
        {
        unsigned char b_BoardHandle;
        unsigned char b_PortValue;
        int           i_ReturnValue;

        if (Initialisation (&b_BoardHandle) == 0)
           {
           i_ReturnValue = i_PA1000_Read8DigitalInput (b_BoardHandle, 1, &b_PortValue);
           printf ("i_PA1000_Read8DigitalInput return %d", i_ReturnValue);
           printf ("Port 1 value = %d", b_PortValue);

           i_ReturnValue = i_PA1000_CloseBoardHandle (b_BoardHandle);
           printf ("i_PA1000_CloseBoardHandle = %d", i_ReturnValue);
           }
        else
           {
           printf ("Initialisation error");
           }
        }
```

9.3.3 Read 16 digital inputs

a) Flow chart

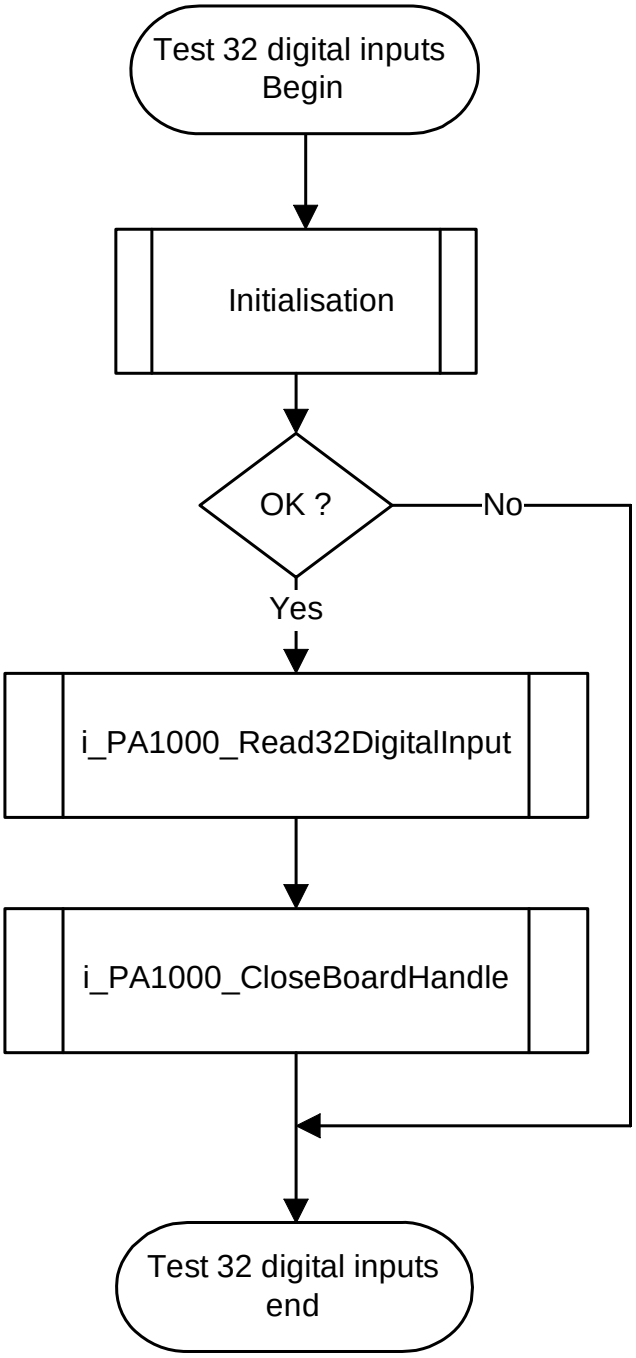


b) Example in C

```
Void    main   (void)
        {
        unsigned char b_BoardHandle;
        long          l_PortValue;
        int           i_ReturnValue;

        if (Initialisation (&b_BoardHandle) == 0)
           {
           i_ReturnValue = i_PA1000_Read16DigitalInput (b_BoardHandle, 1, &l_PortValue);
           printf ("i_PA1000_Read16DigitalInput return %d", i_ReturnValue);
           printf ("Port 1 value = %d", l_PortValue);

           i_ReturnValue = i_PA1000_CloseBoardHandle (b_BoardHandle);
           printf ("i_PA1000_CloseBoardHandle = %d", i_ReturnValue);
           }
        else
           {
           printf ("Initialisation error");
           }
        }
```

9.3.4 Read 32 digital inputs

a) Flow chart



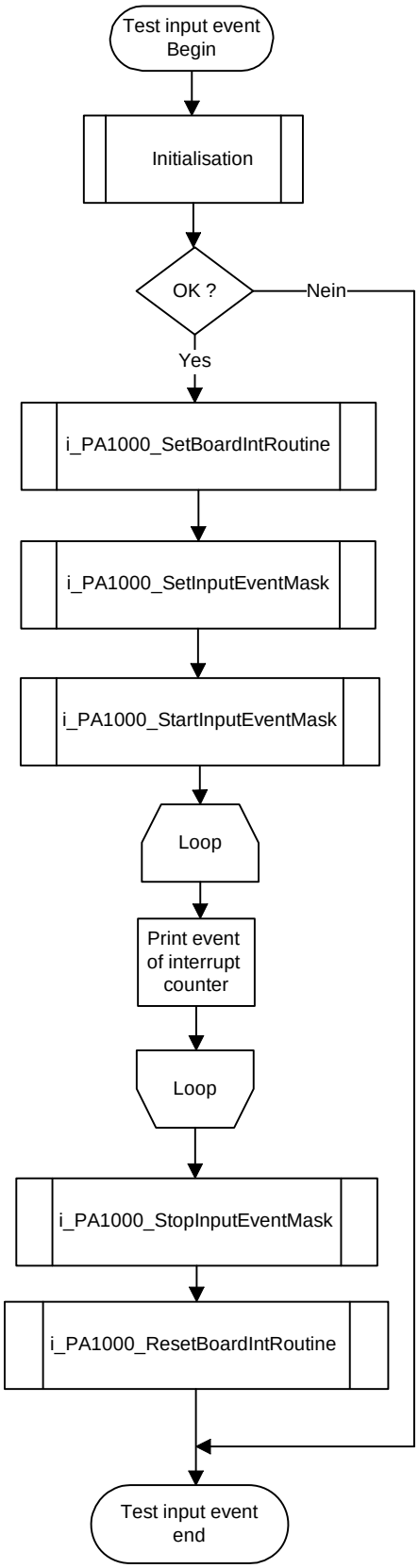
b) Example in C

```
Void    main   (void)
        {
        unsigned char b_BoardHandle;
        long          l_PortValue;
        int           i_ReturnValue;

        if (Initialisation (&b_BoardHandle) == 0)
           {
           i_ReturnValue = i_PA1000_Read32DigitalInput (b_BoardHandle, &l_PortValue);
           printf ("i_PA1000_Read32DigitalInput return %d", i_ReturnValue);
           printf ("Port 1 value = %d", l_PortValue);

           i_ReturnValue = i_PA1000_CloseBoardHandle (b_BoardHandle);
           printf ("i_PA1000_CloseBoardHandle = %d", i_ReturnValue);
           }
        else
           {
           printf ("Initialisation error");
           }
        }
```

9.4 Event

9.4.1 Set event of port 1

a) Flow chart



b) Example in C for DOS

```
Void    main   (void)
        {
        unsigned char b_BoardHandle;
        int           i_ReturnValue;

        if (Initialisation (&b_BoardHandle) == 0)
           {
           i_ReturnValue = i_PA1000_SetBoardIntRoutineDos (b_BoardHandle,
                                                           v_InterruptRoutine);
           printf ("i_PA1000_SetBoardIntRoutineDos return %d", i_ReturnValue);

           i_ReturnValue = i_PA1000_SetInputEventMask (b_BoardHandle, 1,
                                                       PA1000_OR, "11111111");
           printf ("i_PA1000_SetInputEventMask return %d", i_ReturnValue);

           i_ReturnValue = i_PA1000_StartInputEvent (b_BoardHandle, 1);
           printf ("i_PA1000_StartInputEvent return %d", i_ReturnValue);

           do
              {
              printf ("Input event counter = %d", b_InterruptMask [0]);
              }
           while (!kbhit ());

           i_ReturnValue = i_PA1000_StopInputEvent (b_BoardHandle, 1);
           printf ("i_PA1000_StopInputEvent return %d", i_ReturnValue);

           i_ReturnValue = i_PA1000_ResetBoardIntRoutine (b_BoardHandle, 1);
           printf ("i_PA1000_ResetBoardIntRoutine return %d", i_ReturnValue);

           i_ReturnValue = i_PA1000_CloseBoardHandle (b_BoardHandle);
           printf ("i_PA1000_CloseBoardHandle = %d", i_ReturnValue);
           }
        else
           {
           printf ("Initialisation error");
           }
        }
```

c) Example in C for Windows 3.11

```
Void    main   (void)
        {
        unsigned char b_BoardHandle;
        int           i_ReturnValue;

        if (Initialisation (&b_BoardHandle) == 0)
           {
           i_ReturnValue = i_PA1000_SetBoardIntRoutineWin16 (b_BoardHandle,
                                                             v_InterruptRoutine);
           printf ("i_PA1000_SetBoardIntRoutineWin16 return %d", i_ReturnValue);

           i_ReturnValue = i_PA1000_SetInputEventMask (b_BoardHandle, 1,
                                                       PA1000_OR, "11111111");
           printf ("i_PA1000_SetInputEventMask return %d", i_ReturnValue);

           i_ReturnValue = i_PA1000_StartInputEvent (b_BoardHandle, 1);
           printf ("i_PA1000_StartInputEvent return %d", i_ReturnValue);

           do
              {
              printf ("Input event counter = %d", b_InterruptMask [0]);
              }
           while (!kbhit ());

           i_ReturnValue = i_PA1000_StopInputEvent (b_BoardHandle, 1);
           printf ("i_PA1000_StopInputEvent return %d", i_ReturnValue);

           i_ReturnValue = i_PA1000_ResetBoardIntRoutine (b_BoardHandle, 1);
           printf ("i_PA1000_ResetBoardIntRoutine return %d", i_ReturnValue);

           i_ReturnValue = i_PA1000_CloseBoardHandle (b_BoardHandle);
           printf ("i_PA1000_CloseBoardHandle = %d", i_ReturnValue);
           }
        else
           {
           printf ("Initialisation error");
           }
        }
```

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

```
Void    main   (void)
        {
        unsigned char b_BoardHandle;
        int           i_ReturnValue;

        if (Initialisation (&b_BoardHandle) == 0)
           {
           i_ReturnValue = i_PA1000_SetBoardIntRoutineWin32 (b_BoardHandle,
                                                             ASYNCHRONOUS_MODE,
                                                             0,
                                                             NULL,
                                                             v_InterruptRoutine);
           printf ("i_PA1000_SetBoardIntRoutineWin32 return %d", i_ReturnValue);

           i_ReturnValue = i_PA1000_SetInputEventMask (b_BoardHandle, 1,
                                                       PA1000_OR, "11111111");
           printf ("i_PA1000_SetInputEventMask return %d", i_ReturnValue);

           i_ReturnValue = i_PA1000_StartInputEvent (b_BoardHandle, 1);
           printf ("i_PA1000_StartInputEvent return %d", i_ReturnValue);

           do
              {
              printf ("Input event counter = %d", b_InterruptMask [0]);
              }
           while (!kbhit ());

           i_ReturnValue = i_PA1000_StopInputEvent (b_BoardHandle, 1);
           printf ("i_PA1000_StopInputEvent return %d", i_ReturnValue);

           i_ReturnValue = i_PA1000_ResetBoardIntRoutine (b_BoardHandle, 1);
           printf ("i_PA1000_ResetBoardIntRoutine return %d", i_ReturnValue);

           i_ReturnValue = i_PA1000_CloseBoardHandle (b_BoardHandle);
           printf ("i_PA1000_CloseBoardHandle = %d", i_ReturnValue);
           }
        else
           {
           printf ("Initialisation error");
           }
        }
```

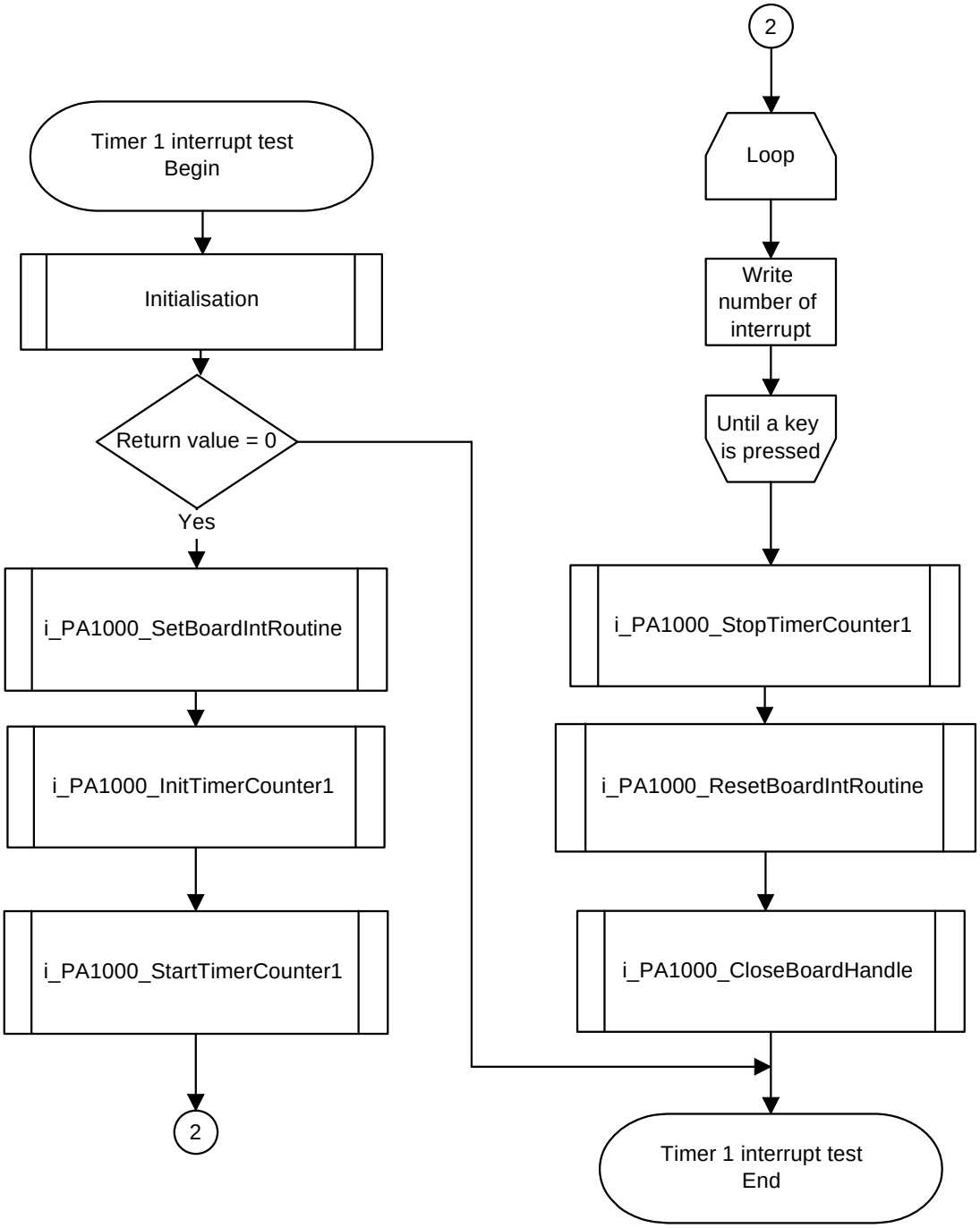
e) Example in C for Windows NT/95 (synchronous mode)

```
Void    main   (void)
        {
        unsigned char b_BoardHandle;
        int           i_ReturnValue;

        if (Initialisation (&b_BoardHandle) == 0)
           {
           i_ReturnValue = i_PA1000_SetBoardIntRoutineWin32 (b_BoardHandle,
                                                             SYNCHRONOUS_MODE,
                                                             sizeof (str_UserStruct),
                                                             (void**)&ps_GlobalUserStruct,
                                                             v_InterruptRoutine);
           printf ("i_PA1000_SetBoardIntRoutineWin32 return %d", i_ReturnValue);

           i_ReturnValue = i_PA1000_SetInputEventMask (b_BoardHandle, 1,
                                                       PA1000_OR, "11111111");
           printf ("i_PA1000_SetInputEventMask return %d", i_ReturnValue);

           i_ReturnValue = i_PA1000_StartInputEvent (b_BoardHandle, 1);
           printf ("i_PA1000_StartInputEvent return %d", i_ReturnValue);

           do
              {
              printf ("Input event counter = %d", ps_GlobalUserStruct->
                                                  b_InterruptMask [0]);
              }
           while (!kbhit ());

           i_ReturnValue = i_PA1000_StopInputEvent (b_BoardHandle, 1);
           printf ("i_PA1000_StopInputEvent return %d", i_ReturnValue);

           i_ReturnValue = i_PA1000_ResetBoardIntRoutine (b_BoardHandle, 1);
           printf ("i_PA1000_ResetBoardIntRoutine return %d", i_ReturnValue);

           i_ReturnValue = i_PA1000_CloseBoardHandle (b_BoardHandle);
           printf ("i_PA1000_CloseBoardHandle = %d", i_ReturnValue);
           }
        else
           {
           printf ("Initialisation error");
           }
        }
```

9.5 Timer/counter

9.5.1 Test the interrupt of timer 1

a) Flow chart



b) Example in C for DOS

```
void main (void)
     {
     int          i_ReturnValue;
     long         l_TimerCounterValue;
     unsigned charb_BoardHandle;

     if (Initialisation (&b_BoardHandle) == 0)
        {
        i_PA1000_SetBoardIntRoutineDos (b_BoardHandle, v_InterruptRoutine);
        printf ("\ni_PA1000_SetBoardIntRoutineDos return value = %d", i_ReturnValue);

       i_ReturnValue = i_PA1000_InitTimerCounter1    (b_BoardHandle,
                                                      PA1000_TIMER,
                                                      1000UL,
                                                      PA1000_CONTINUOUS,
                                                      PA1000_ENABLE);

        printf ("\nTimer initialisation return value = %d", i_ReturnValue);

        i_ReturnValue = i_PA1000_StartTimerCounter1  (b_BoardHandle);
        printf ("\nTimer start return value = %d", i_ReturnValue);

        do
           {
           printf ("\nTimer interrupt counter = %d", b_TimerCounterCpt [0]);
           i_ReturnValue = i_PA1000_ReadTimerCounter1 (b_BoardHandle,
                                                       &l_TimerCounterValue);
           printf ("\ni_PA1000_ReadTimerCounter1 return value = %d", i_ReturnValue);
           printf ("\nTimer value = %l", l_TimerCounterValue);
           }
        while (!kbhit());

        getch ();

        i_ReturnValue = i_PA1000_StopTimerCounter1   (b_BoardHandle);
        printf ("\nTimer stop return value = %d", i_ReturnValue);

        i_ReturnValue = i_PA1000_ResetBoardIntRoutine      (b_BoardHandle);
        printf ("\nReset board interrupt return value = %d", i_ReturnValue);

        i_ReturnValue = i_PA1000_CloseBoardHandle    (b_BoardHandle);
        printf ("\nClose board handle return value = %d", i_ReturnValue);
        }
     else
        {
        printf ("Initialisation error");
        }
```

c) Example in C for Windows 3.11

```
void main (void)
     {
     int          i_ReturnValue;
     long         l_TimerCounterValue;
     unsigned charb_BoardHandle;

     if (Initialisation (&b_BoardHandle) == 0)
        {
        i_PA1000_SetBoardIntRoutineWin16 (b_BoardHandle, v_InterruptRoutine);
        printf (“\ni_PA1000_SetBoardIntRoutineDos return value = %d“, i_ReturnValue);

       i_ReturnValue = i_PA1000_InitTimerCounter1    (b_BoardHandle,
                                                      PA1000_TIMER,
                                                      1000UL,
                                                      PA1000_CONTINUOUS,
                                                      PA1000_ENABLE);

        printf (“\nTimer initialisation return value = %d“, i_ReturnValue);

        i_ReturnValue = i_PA1000_StartTimerCounter1  (b_BoardHandle);
        printf (“\nTimer start return value = %d“, i_ReturnValue);

        do
           {
           printf (“\nTimer interrupt counter = %d“, b_TimerCounterCpt [0]);
           i_ReturnValue = i_PA1000_ReadTimerCounter1 (b_BoardHandle,
                                                       &l_TimerCounterValue);
           printf (“\ni_PA1000_ReadTimerCounter1 return value = %d“, i_ReturnValue);
           printf (“\nTimer value = %l”, l_TimerCounterValue);
           }
        while (!kbhit());

        getch ();

        i_ReturnValue = i_PA1000_StopTimerCounter1   (b_BoardHandle);
        printf (“\nTimer stop return value = %d“, i_ReturnValue);

        i_ReturnValue = i_PA1000_ResetBoardIntRoutine      (b_BoardHandle);
        printf (“\nReset board interrupt return value = %d“, i_ReturnValue);

        i_ReturnValue = i_PA1000_CloseBoardHandle    (b_BoardHandle);
        printf (“\nClose board handle return value = %d“, i_ReturnValue);
        }
     else
        {
        printf (“Initialisation error”);
        }
```

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

```
void main (void)
     {
     int          i_ReturnValue;
     long         l_TimerCounterValue;
     unsigned charb_BoardHandle;

     if (Initialisation (&b_BoardHandle) == 0)
        {
        i_PA1000_SetBoardIntRoutineWin32 (b_BoardHandle,
                                          ASYNCHRONOUS_MODE,
                                          0,
                                          NULL,
                                          v_InterruptRoutine);
        printf ("\ni_PA1000_SetBoardIntRoutineDos return value = %d", i_ReturnValue);

         i_ReturnValue = i_PA1000_InitTimerCounter1   (b_BoardHandle,
                                                       PA1000_TIMER,
                                                       1000UL,
                                                       PA1000_CONTINUOUS,
                                                       PA1000_ENABLE);

        printf ("\nTimer initialisation return value = %d", i_ReturnValue);

        i_ReturnValue = i_PA1000_StartTimerCounter1  (b_BoardHandle);
        printf ("\nTimer start return value = %d", i_ReturnValue);

        do
           {
           printf ("\nTimer interrupt counter = %d", b_TimerCounterCpt [0]);
           i_ReturnValue = i_PA1000_ReadTimerCounter1 (b_BoardHandle,
                                                       &l_TimerCounterValue);
           printf ("\ni_PA1000_ReadTimerCounter1 return value = %d", i_ReturnValue);
           printf ("\nTimer value = %l", l_TimerCounterValue);
           }
        while (!kbhit());

        getch ();

        i_ReturnValue = i_PA1000_StopTimerCounter1   (b_BoardHandle);
        printf ("\nTimer stop return value = %d", i_ReturnValue);

        i_ReturnValue = i_PA1000_ResetBoardIntRoutine      (b_BoardHandle);
        printf ("\nReset board interrupt return value = %d", i_ReturnValue);

        i_ReturnValue = i_PA1000_CloseBoardHandle    (b_BoardHandle);
        printf ("\nClose board handle return value = %d", i_ReturnValue);
        }
     else
        {
         printf ("Initialisation error");
        }
```

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

```
void main (void)
     {
     int          i_ReturnValue;
     long         l_TimerCounterValue;
     unsigned charb_BoardHandle;

     if (Initialisation (&b_BoardHandle) == 0)
        {
        i_PA1000_SetBoardIntRoutineWin32 (b_BoardHandle,
                                          ASYNCHRONOUS_MODE,
                                          sizeof (str_UserStruct),
                                          (void**)&ps_GlobalUserStruct,
                                          v_InterruptRoutine);
        printf ("\ni_PA1000_SetBoardIntRoutineDos return value = %d", i_ReturnValue);

         i_ReturnValue = i_PA1000_InitTimerCounter1   (b_BoardHandle,
                                                       PA1000_TIMER,
                                                       1000UL,
                                                       PA1000_CONTINUOUS,
                                                       PA1000_ENABLE);

        printf ("\nTimer initialisation return value = %d", i_ReturnValue);

        i_ReturnValue = i_PA1000_StartTimerCounter1  (b_BoardHandle);
        printf ("\nTimer start return value = %d", i_ReturnValue);

        do
           {
           printf ("\nTimer interrupt counter = %d", ps_GlobalUserStruct->
                                                     b_TimerCounterCpt [0]);
           i_ReturnValue = i_PA1000_ReadTimerCounter1 (b_BoardHandle,
                                                       &l_TimerCounterValue);
           printf ("\ni_PA1000_ReadTimerCounter1 return value = %d", i_ReturnValue);
           printf ("\nTimer value = %l", l_TimerCounterValue);
           }
        while (!kbhit());

        getch ();

        i_ReturnValue = i_PA1000_StopTimerCounter1   (b_BoardHandle);
        printf ("\nTimer stop return value = %d", i_ReturnValue);

        i_ReturnValue = i_PA1000_ResetBoardIntRoutine      (b_BoardHandle);
        printf ("\nReset board interrupt return value = %d", i_ReturnValue);

        i_ReturnValue = i_PA1000_CloseBoardHandle    (b_BoardHandle);
        printf ("\nClose board handle return value = %d", i_ReturnValue);
        }
     else
        {
         printf ("Initialisation error");
        }
```

INDEX