ISO 9001 certified

ADDI-DATA®

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

Attention!
Product discontinuation due to EC RoHS directive
More info: www.addi-data.com

CE

Technical description

ADDIALOG PA 3500

Analog output board

Hardware, installation, software examples

6th edition 08/2001

Guarantee and responsibility

Basically are effective our "general terms of delivery and payment". The manager receives them at the latest with the invoice. Claims for guarantee and responsibility in case of injuries and material damages are excluded, if they are due to one or some of the following causes:

- if the board has not been used for the intended purpose
- improper installation, operation and maintenance of the board
- if the board has been operated with defective safety devices or with not appropriate or non-functioning safety equipment
- non-observance of the instructions concerning: transport, storage, inserting the board, use, limit values, maintenance, device drivers
- altering the board at the user's own initiative
- altering the source files at the user's own initiative
- not checking properly the parts which are subject to wear
- disasters caused by the intrusion of foreign bodies and by influence beyond the user's control.

Licence for ADDI-DATA software products

Read carefully this licence before using the software ADDISET and ADDIMON. The right for using this software is given to the customer, if he/she agrees to the conditions of this licence.
- this software can only be used for configuring ADDI-DATA boards.
- copying the software is forbidden (except for archiving/ saving data and for replacing defective data media).
- deassembling, decompiling, decoding and reverse engineering of the software are forbidden.
- this licence and the software can be transferred to a third party, so far as this party has purchased a board, declares to agree to all the clauses of this licence contract and the preceding owner has not kept copies of the software.

Trademarks

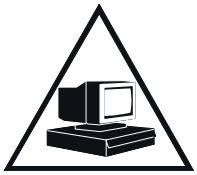
Borland C and Turbo Pascal are registered trademarks of Borland International, INC.
Burr-Brown is a registered trademark of Burr-Brown Corporation
Intel is a registered trademark of Intel Corporation
AT, IBM, ISA and XT are registered trademarks of International Business Machines Corporation
Microsoft, 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 use 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 manual, 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**

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

CE

Declaration of Conformity

This declaration is valid for the following product:

ADDIALOG PA 3500
4/8 analog outputs, 14-bit
optical isolation

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 TD3500.020.

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.

17. February 1999

Legally valid signature of the manufacturer

Figures

Tables

1 INTENDED PURPOSE OF THE BOARD

The board **PA 3500** 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 normed CE mark.

The board **PA 3500** operates in your PC as an automation interface for standard analog applications. It is equipped with up to 8 analog outputs (voltage or current outputs) and 2 digital inputs and outputs.

The functions of the **PA 3500** are to be used according to the intended purpose of the board as follows:

- **Analog outputs**
  The 8 analog outputs as well as both digital outputs switch off by timeout or voltage fall. Make sure that no function failure occurs in the application. In this case adapt your connection and the program control.

- **Watchdog**
  The watchdog function is recommended when the analog outputs are used for important control functions.
  Make sure by writing your controlling program that the outputs are updated minimum once within 4.2 s while the functions are running.

- **Digital inputs:**
  Both digital inputs convert 24 V and 0 V levels into logical states "1" and "0".

- **Trigger input:**
  The analog outputs are triggered through the trigger input and synchronised with other functions or events.

- **Digital outputs:**
  Operation states can be transmitted through both digital outputs. In trigger mode they are used for the synchronisation of the external trigger event.

Please only use the board :
- in conditions providing absolute security
- in a closed housing which is adequately protected against environmental influences
- **with the accessories** we recommend

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

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

1.1 Limits of use

The board PA 3500 is not to be used for securing emergency stop functions.

No emergency stop functions are to be controlled through the relays.

The board must not operate in a PC which does not provide for an adequate protection against environmental conditions (e.g.: liquids, dust, ...) and which is not equipped with an appropriate aeration system.

Make sure that your PC is equipped with a safety system and a shielded metal housing.
The installation of the board in sites lying under risk of explosion is excluded.
The board should not be submitted to vibrations without additional fixation.

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.

Please control that the PC-housing and the cable are effectively shielded before putting the device into operation.

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

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

2 USER

Only a person trained in electronics is entitled to:
- the installation, the operation, the maintenance of the board.

The basic knowledge of a high-level programming language is sufficient for programming the board.

The knowledge of conversion between the different numerative systems (binary, decimal, hexadecimal) is necessary.

For the CE conformity declaration, the knowledge of EMC is strongly recommended.

2.2 Personal protection

Consider the country-specific regulations about

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

Clothing: antistatic work-clothes (cotton-made clothes, shoes with conductive soles) and earthing strip on the wrist are strictly recommended before putting the board into operation.

Work conditions: Select a work place which protects you from electrostatic charges (conductive table, table layers in cotton, conductive floor layers, ...).

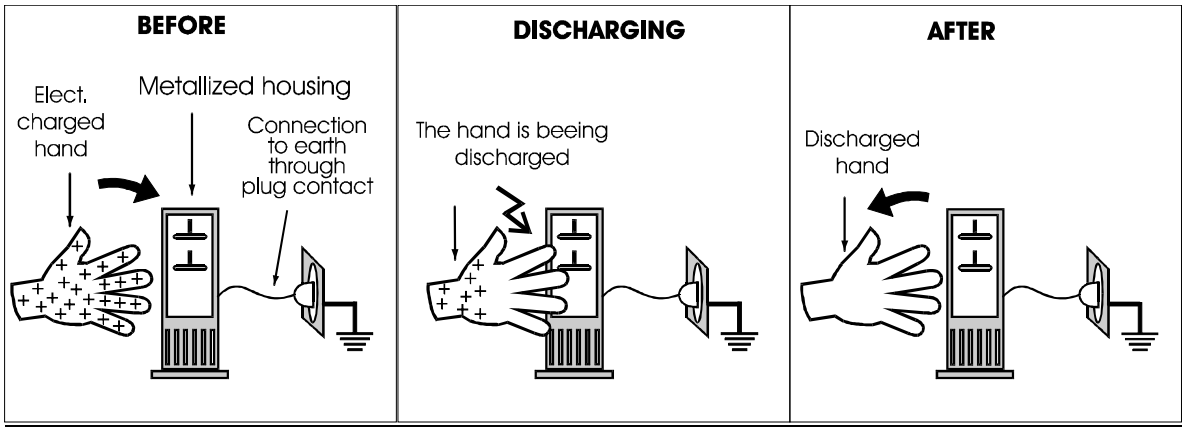
2.3 In case of emergency

System failure: when possible, check the diagnosis outputs before switching off the system. Plug the connector and the cable out. Wait 10 min before removing the board from the PC.
Send the board back to us for an inspection and an eventual repair.

In case of fire, dangerous vapours and gas can occur. Disconnect the PC from all external connections and plug the connector out. Only use non-residual carbon dioxide to extinguish the flames.
Consider the professional regulations of your sector.

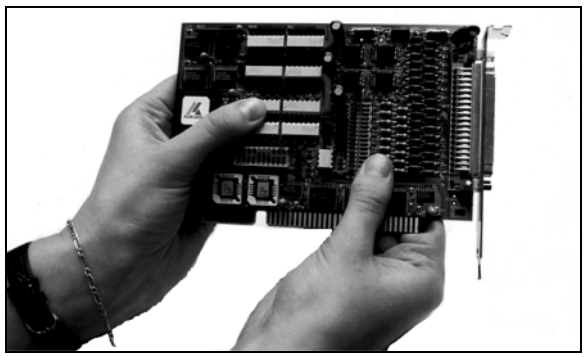
3 HANDLING THE BOARD

Fig. 3-1: Handling of the board

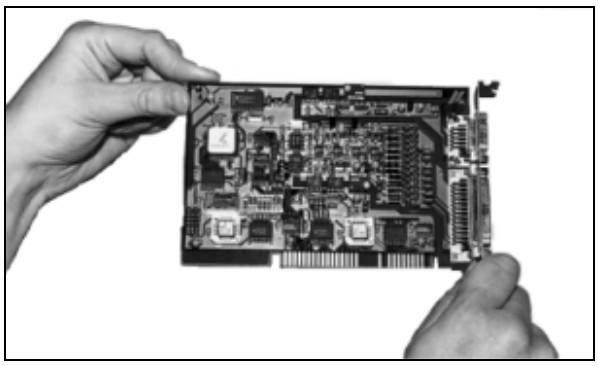


Discharge yourself; Wear an earthing strip.
Put your hand on an earthed, metallized surface (e.g.: radiator, tap, PC housing), before touching the board. If you are electrostatically charged, this could lead to an electric shock. Make sure your state of health allows it. Persons with cardiac problems or pacemaker act at their own risk.

Avoid the direct touching of the board



WRONG



CORRECT

4 TECHNICAL DATA

4.1 Electromagnetic compatibility (EMC)

The board has been subjected to EMC tests in an accredited laboratory.
In the tested appliance configuration, the emission limit values of the board complies with the norms EN55011 and EN55022.

If you have to consider the CE conformity of your PC notice the following indications:

Emission spectrum:
The **PA 3500** operates with bus signals which are controlled with **8 MHz** clock by the PC processor.
According to the internal installation of your PC the processor clock of the PC couples more or less stronger on the board. The optical isolation separates the low frequency signals from the peripheral. Coupling or isolation capacities (Ci) of the optical couplers are determined in the limit values.

High frequency emissions are suppressed by the EMI filters.
The PC emission are of various forms in the emission spectrum: the EMI filter must also have different ranges of influence. When PCI slots are available, the 33 MHz bus clock coupling of intermodulation products should be tested.
The board needs a 8 MHz oscillator as clock transmitter. A 4 MHz clock signal is transmitted to the D/A converter.
As the TLL part of the PC is coupled trough the galvanic isolation to the external part, the coupling capacities of the optical couplers and of the DC/DC converter determine the frequency spectrum which may be transmitted through the creep distance.

The board complies with the limit values of the norm EN50082-2 as follows:

	Real value	Set value
ESD	8 kV	4 kV
Fields	10 V/m	10 V/m
Burst	4 kV	2 kV
Conducted radio interferences	10 V	10 V

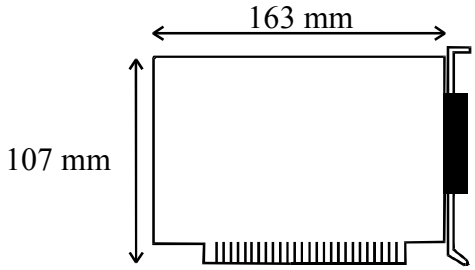
WARNING!
The EMC tests have been carried out in a specific appliance configuration.
We guarantee these limit values **only** in this configuration. [1]

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.

[1] We transmit our appliance configuration on request.

4.2 Physical set-up of the board

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



Breadth:	19.2 mm
Installation in:	AT slot
Connection to the peripheral:	37-pin SUB-D male connector 16-pin connector for ribbon cable
Accessories:	Screw terminal boards: PX 901-A PX-901-ZG for the connection of dig. I/Os External cable: ST010 Internal ribbon cable: FB3000 See connection examples

4.3 Versions

The PA 3500 board is available in **4 versions**:

PA 3500-8:	8 voltage outputs
PA 3500-4:	4 voltage outputs
PA 3500-8C:	8 current outputs
PA 3500-4C:	4 current outputs

4.4 Options

The board can be delivered with an amplifier for the voltage outputs.

PA 3500-8 with **option 2P**: 8 voltage outputs for high load
PA 3500-4 with **option P**: 4 voltage outputs for high load

4.5 Limit values

Max. Altitude storage:	up to 2000 m above 0
Operating temperature:	15 to 60 °C
Storage temperature:	-25 to 70 °C
Relative humidity:	30-99% non condensing

Minimum PC requirements:

Operating system:	MS DOS 3.3 or > Windows 3.1
Slot types:	ISA (AT)
Number of the necessary slots:	
for analog outputs:	1 slot + 1 open slot
for digital inputs and outputs:	only open slot used

Resources:

I/O memory requirement	
for version PA 3500-4/4C:	3 Bytes
for version PA 3500-8/8C:	4 Bytes
IRQ (selectable):	3, 4, 5, 6, 7, 9, 10, 11, 12, 14, 15

Energy requirements:

Operating voltage Vcc from PC:	5 V ± 5 %
Current consumption Vcc (No load) :	500 mA ± 10 %
External operating voltage Vext. [1]:	15-40 V
External current requirement Iext. [1]:	max. 20 mA per channel
Current consumption (Vext) [1]:	depending on the load

ISA Bus interface:

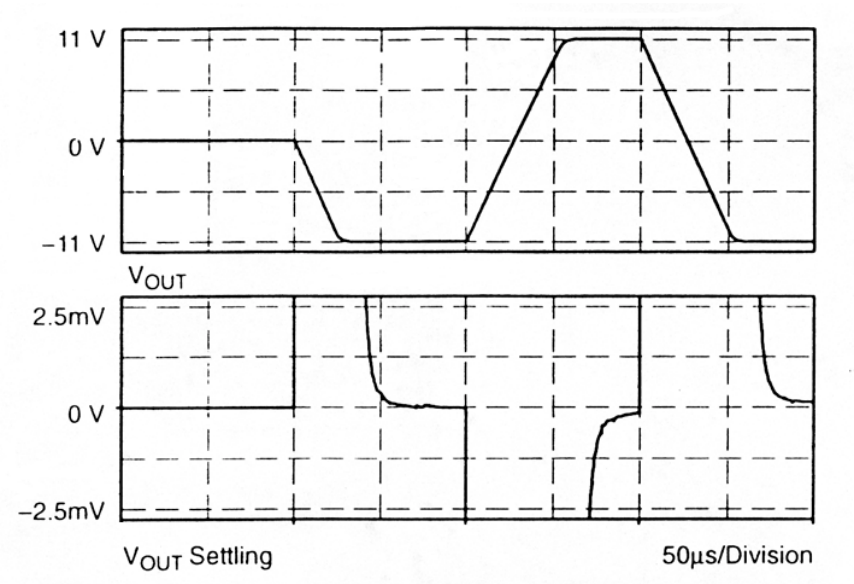
Bus speed:	8 MHz
Data bus access:	16 and 8 bits
Address decoding (A12-A15 fixed at0):	A3-A11 per DIP-switches
Addressing:	I/O-mapped

8 analog voltage outputs

Output type:	Single Ended
Resolution (Bipolar):	14-bit
Resolution (Unipolar):	13-bit
Precision:	12-bit
Output voltage range:	± 10 V, 0-10 V
Time to ready (tr):	typ. 4 µs
Settling time: (ts); 20 V steps:	typ. 30 µs

[1] This parameter only refers to the current versions (PA 3500-8C and PA 3500-4C), which need an external supply voltage to function

Fig. 4-1: Typical settling time



The first diagram indicates the settling time at 10 V and 20 V steps. The second diagram figures the variation of the output voltage at ideal value.

Non-linearity:

Integral: ..... ± 4 LSB
Differential: ..... ± 4 LSB
Bipolar Null Offset: ..... ± 16 LSB
Maximum load: ..... 5 mA at 10 V [1]

8 analog amplified voltage outputs (Option -2P with AD712JR)

Settling time (ts); 20 V step: ..... typ. 32 µs
max. 52 µs
Short-circuit current max.: ..... 25 mA
Max. available load current: ..... 140 mA

8 analog current outputs (version -C)

Output type: ..... Single Ended
Resolution: 14-bit
Precision: 12-bit
Output current: ..... 0-20 mA, 4-20 mA, 5-25 mA
Time to ready (tr): ..... typ 4 µs
Settling time (ts); 20 mA step: ..... typ. 45 µs
max. 70 µs

Non-linearity:

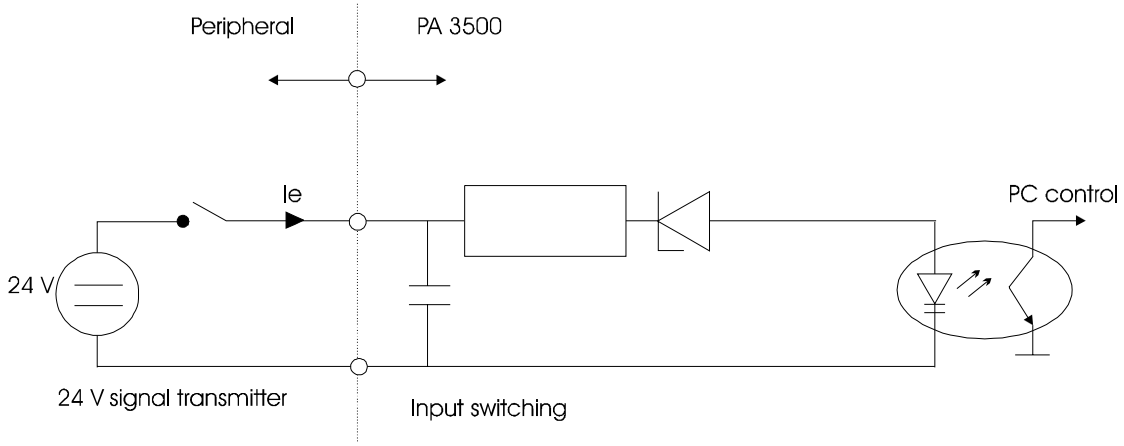
Integral: ..... ± 8 LSB
Differential: ..... ± 7 LSB
Bipolar Null Offset: ..... ± 16 LSB
Maximum load: ..... max. 500 Ω at 20 mA

[1] Higher load (25 mA on request)

Watchdog : ..... Time: 4.2 s
Programmable: ..... per software
Output after timeout: ..... all outputs switch off
Interrupt (selectable): ..... at alarm state or timeout

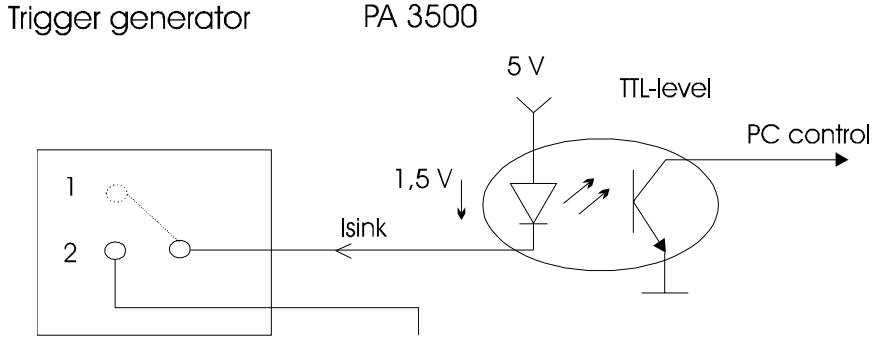
2 digital inputs:
Logic: ..... active High
Interrupt: ..... by rising edge
Nominal voltage: ..... 24 VDC
Switching voltage, typ.: ..... 16.5 V
Input current at nominal voltage Ie: ..... 6 mA
Signal delay. ..... 70 µs, at nominal voltage
Maximum input frequency: ..... 5 kHz, at nominal voltage
Isolation capacity (optical couplers) Ci: ..... typ. 0.8 pF (f = 1MHz; V = 0)

Fig. 4-2: Switching principle of the digital inputs



Trigger input:
Logic: ..... active Low
Trigger edge: ..... falling
Interrupt: ..... at falling edge
Nominal voltage: ..... 5 V
Isink (at 0 V): ..... 10.6 mA
Max. transmitted trigger frequency: ..... 5 MHz
Common mode transient immunity, min.: ..... typ. 10 kV/µs
Isolation capacity (optical couplers) Ci: ..... typ. 0.6 pF (f = 1 MHz, V = 0)

Fig. 4-3: Switching principle of the trigger input



Switching state: 1 = open or 5 V
2 = closed or 0 V

Timing:

Test conditions: ..... 25 °C, DOS operating system, Polling in synchronous mode with trigger simulation per software, Pentium 60 MHz, 5 V voltage supply of the digital outputs, load resistance R1 = 680 Ω, Measurement time depends on the switching threshold

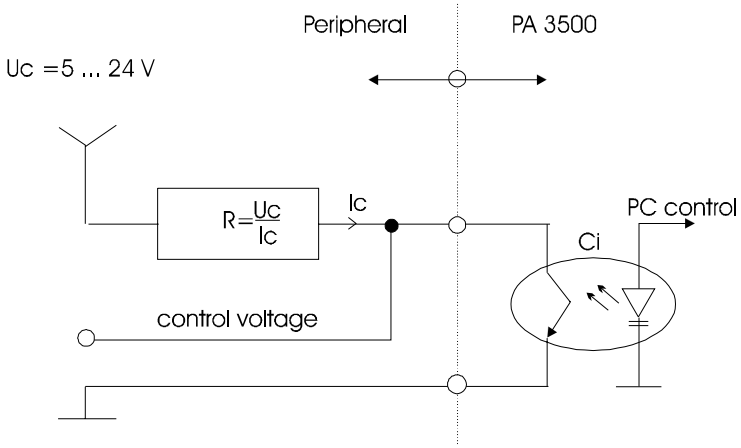
Switch-ON time of output 1
after Write t_{set} (A1)..... 4.4-5.3 µs
Switch-ON time of output 2
after Write t_{set} (A2)..... 11-11.7 µs
Switch-OFF time of the outputs 1/2
after Write t_{reset} ..... 33-38.5 µs

2 digital outputs with optical isolation

Typ: ..... Open collector
Open emitter
Recommended collector current IC: ..... typ. 1-10 mA
max. 20 mA
Switch-ON time: ..... typ. 3 µs
Switch-OFF time
(strongly depending on the load): ..... 3 µs [1]
Collector-emitter-saturation voltage: ..... max 0.4 V
Collector current (OFF state): ..... max. 10 µA
Isolation capacity (optical couplers); Ci ..... typ. 0.8 pF (f = 1 MHz, V = 0)

[1] test conditions: VCC = 10 V; R1 = 100 Ω

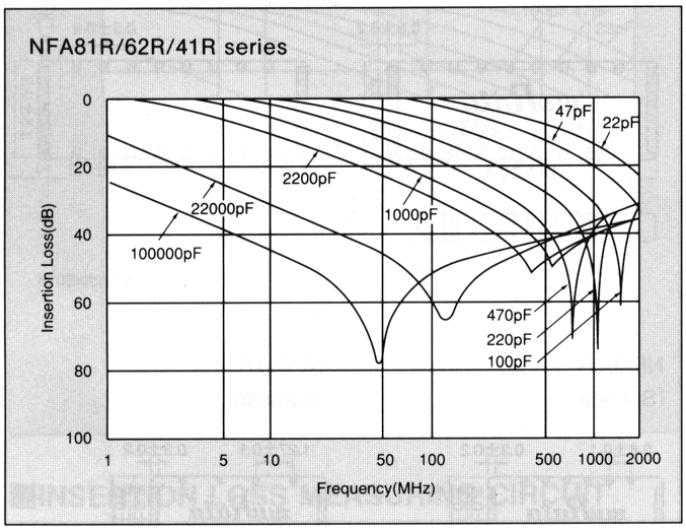
Fig. 4-4: Switching principle of the digital outputs



Safety

Galvanic isolation:	500 V
Dimensioning voltage of creep distance:	DIN VDE 0411-100
Optical couplers (digital inputs and outputs): ..	DIN VDE 0884/08.87 UL1577; file N° E67349
Test voltage (TTL peripheral):	800 Vrms.
EMI filters:	100 pF +50% / -20 %
EMV:	EN55022, EN55011, EN50082-2

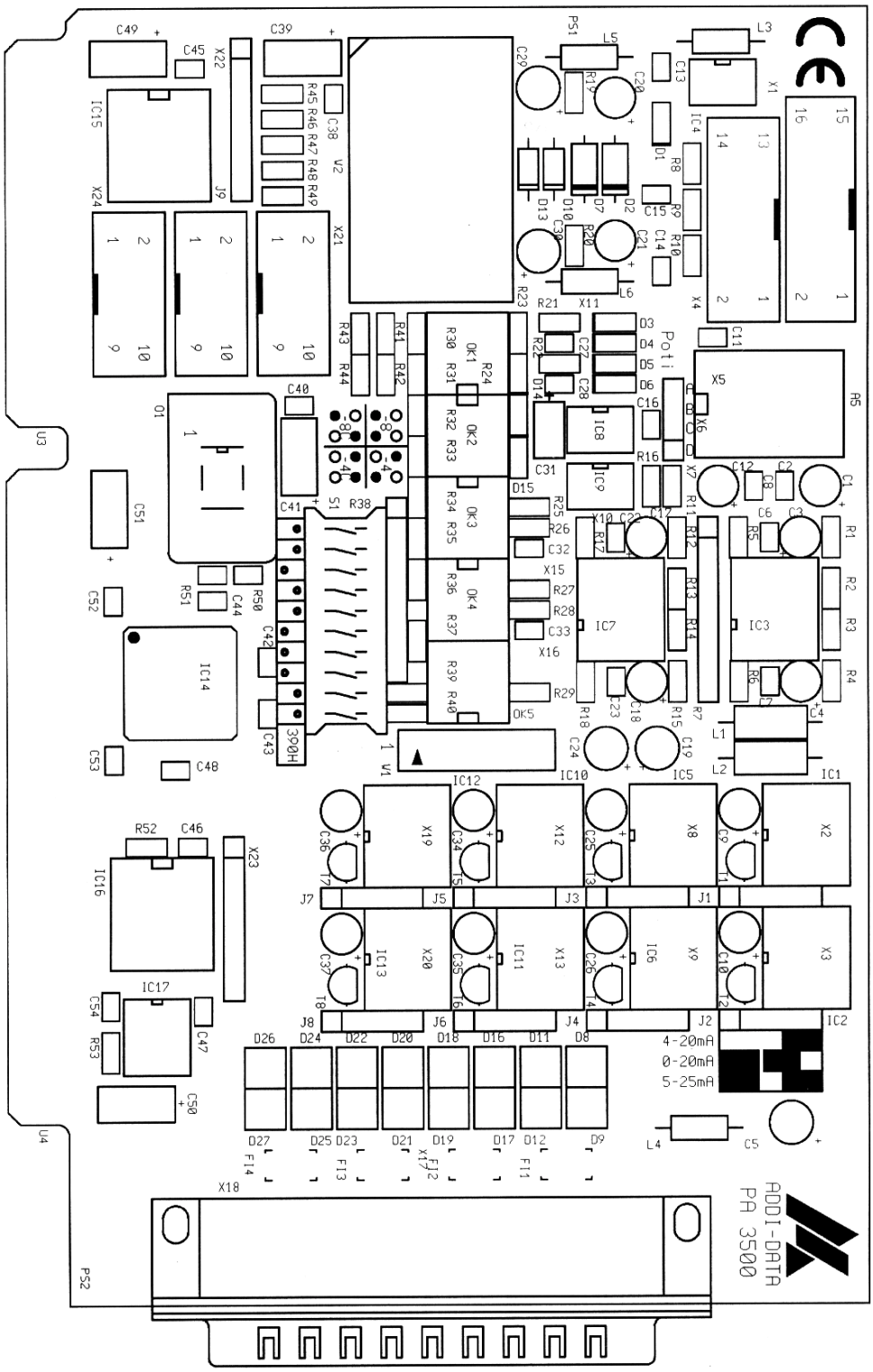
Fig. 4-5: Insertion loss characteristics of the EMI filters



5 SETTINGS

5.1 Component scheme

Fig. 5-1: Component scheme of the board PA 3500



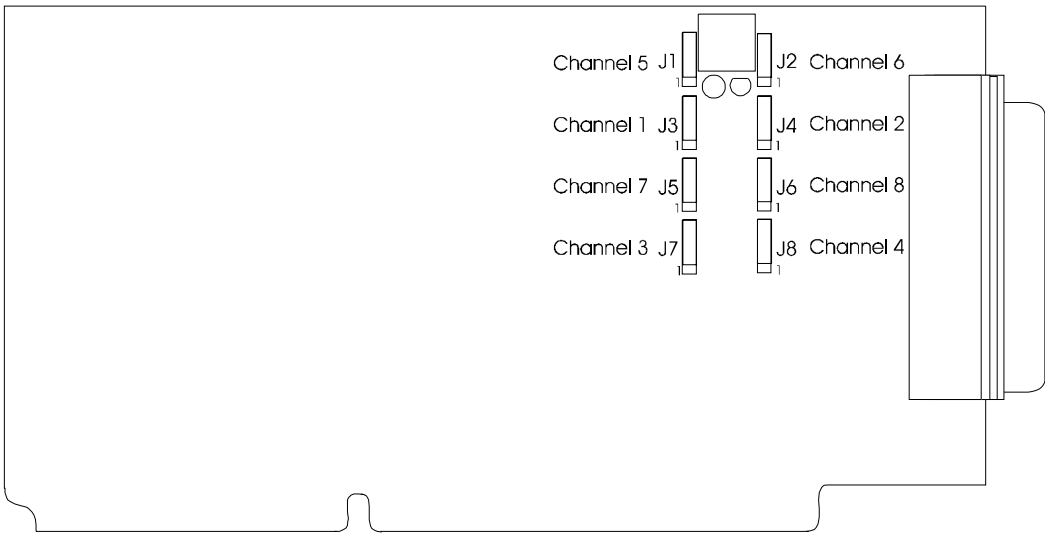
5.2 Jumper settings

IMPORTANT!
In the current version the board has no jumper.
Please do read this chapter if you use a PA 3500-4C or PA 3500-8C.

5.2.1 Position of the jumpers on the board and settings at delivery

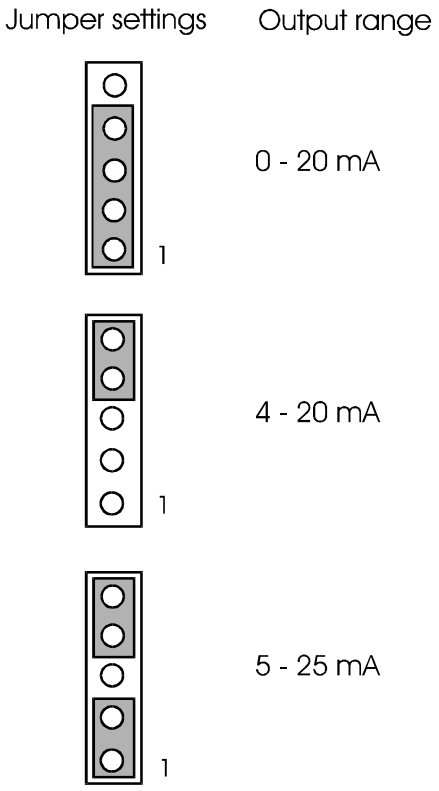
Settings at delivery: 0-20 mA on all channels

Fig. 5-2: Position of the jumpers on the PA 3500 board



5.2.2 Jumper settings

Fig. 5-3: Jumper settings according to the selected output range



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 ADDIREG (once is enough)
5. Configure the board
6. Install the driver and the samples if necessary
7. Switch off the PC
8. Install the **second** board
9. Start the PC
10. Configure the board
11. Install the driver and the samples if necessary.

etc.

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

6.1 Base address

WARNING!
If the base address is set wrong, the board and/or other PC components may be strongly damaged.

- Check attentively the resources. (6.1.1)
- To change the base address, refer to the decoding table. (6.1.2)

6.1.1 Testing the resources

Under Windows 95/98 and Windows NT you will find the resources which are already occupied as follows:

Select under "Start":

- SETTINGS
- CONTROL PANEL
- ICON: SYSTEM
- DEVICE MANAGER
- REFRESH
- PROPERTIES
- RESOURCES: I/O PORT

Under DOS or other operating systems please refer to the PC documentation and check the system address range of your PC.

The preset address range begins at the address 390Hex and ends at the address 391Hex. Select another address range by changing the base address if other PC components are occupying the resources.

6.1.2 Changing the base address

Set the base address through the 10-pin DIP switches S1 according to the figure below.

To change the base address, use the decoding table.

As an example, the address 0390H is decoded in the following figure.

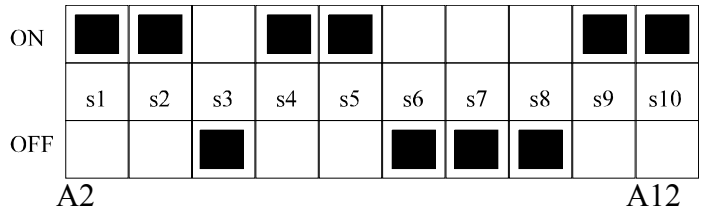
Table 6-1: Decoding of the base address

	MSB												LSB	
Decoded address bus	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2
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
DIP switches S1 Logic "0"= ON Logic "1" = OFF	*	*	*	*	S10 ON	S9 ON	S8 OFF	S7 OFF	S6 OFF	S5 ON	S4 ON	S3 OFF	S2 ON	S1 ON

* : Decoded at logic “0”

Fig. 6-1: Settings of the DIP switches

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



6.2 Inserting the board

IMPORTANT!
Do observe the *safety instructions*.

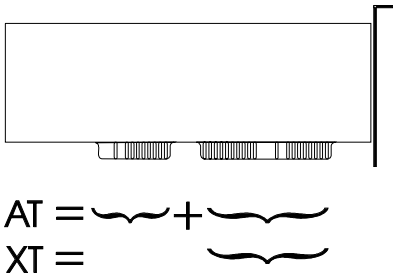
6.2.1 Opening the PC

- 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.

6.2.2 Selecting a free slot

1. **Select a free ISA slot.**

Fig. 6-2: Slot types



The board can be inserted either in an XT or AT slot. In some cases the board can also be inserted in an EISA slot provided it respects the instructions of the PC manufacturer.

2. **Remove the back cover of the selected slot**
   according to the instructions of the PC manufacturer.
   Keep the back cover in order to be able to close the PC properly if you wish to remove the board.
3. **Discharge yourself from electrostatic charges**

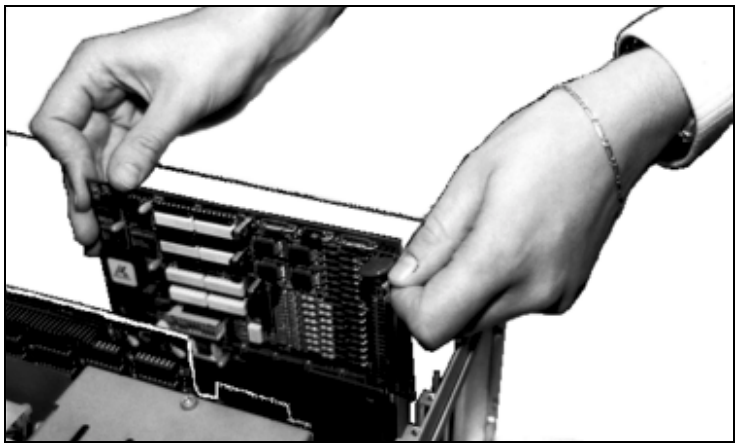
4. Take the board out of its protective blister pack.

Fig. 6-3: Opening the protective blister pack

6.2.3 Inserting the board

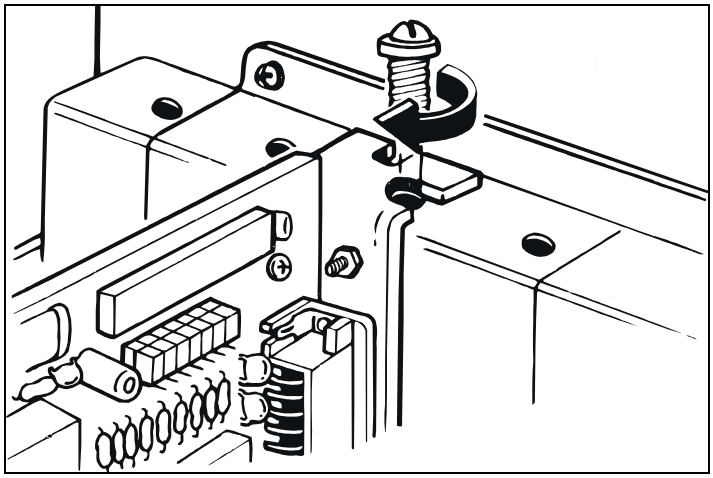
- **Discharge yourself from electrostatic charges.**
- Insert the board **vertically into the chosen slot.**

Fig. 6-4: Inserting the board



- **Secure the board** to the rear of the PC housing with the screw which held the back cover.

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



- **Tighten all loosen screws.**

6.2.4 Closing the PC

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

7 SOFTWARE

The board is supplied with a CD-ROM. The CD contains ADDIREG for Windows NT 4.0 and Windows 95/98.

IMPORTANT!
To install the new version of ADDIREG, please uninstall first the current version from your PC with the **ADDI_UNINSTALL** program (See paragraph 7.4).

You can also download the ADDIREG program from the Internet:
http://www.addi-data.de
http://www.addi-data.com.

The CD includes also:
- Standard software for the ADDI-DATA boards:
  - 16-bit for MS-DOS and Windows 3.11
  - 32-bit for Windows NT/95/98.

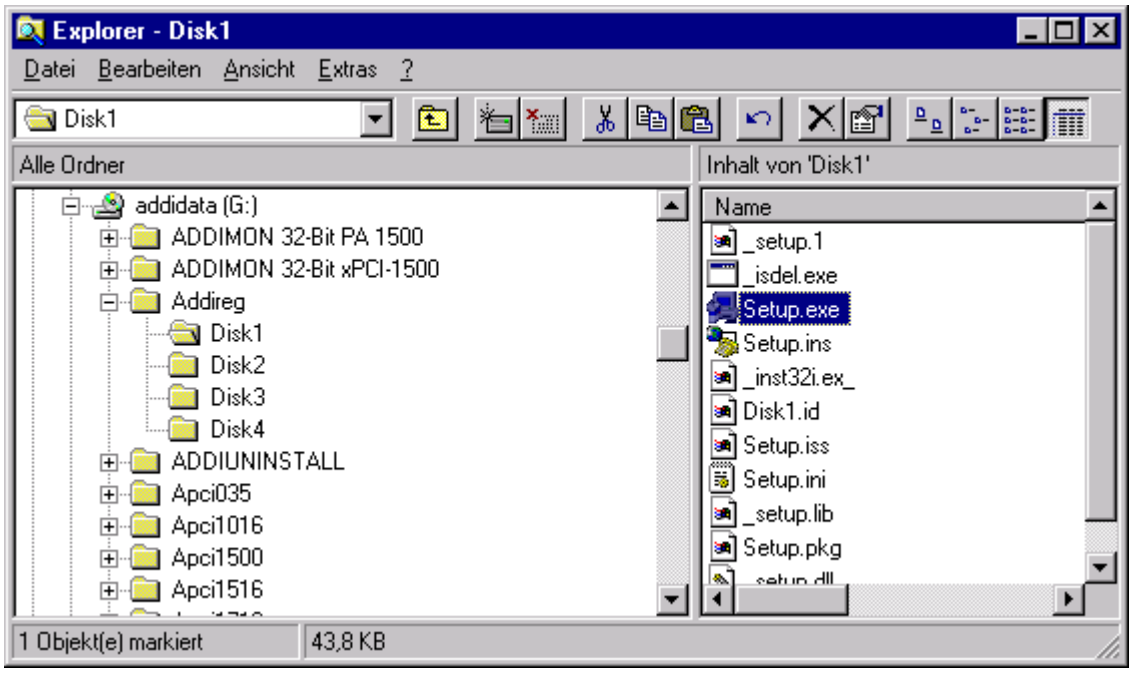
7.1 Board configuration with ADDIREG

7.1.1 Installing the ADDIREG program

IMPORTANT!
Install first the ADDIREG program before installing or starting any other application of the board.

- Change to the CD drive.

Fig. 7-1: Installation of the ADDIREG program



- Start the set-up file.

- Select one of the 3 parameters
  1- typical
  2- compact
  3- custom

Proceed as indicated on the screen and read the "Software License" and "Readme". In "custom", you can select your operating system.
The installation program gives you further instructions.

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

7.1.2 Program description

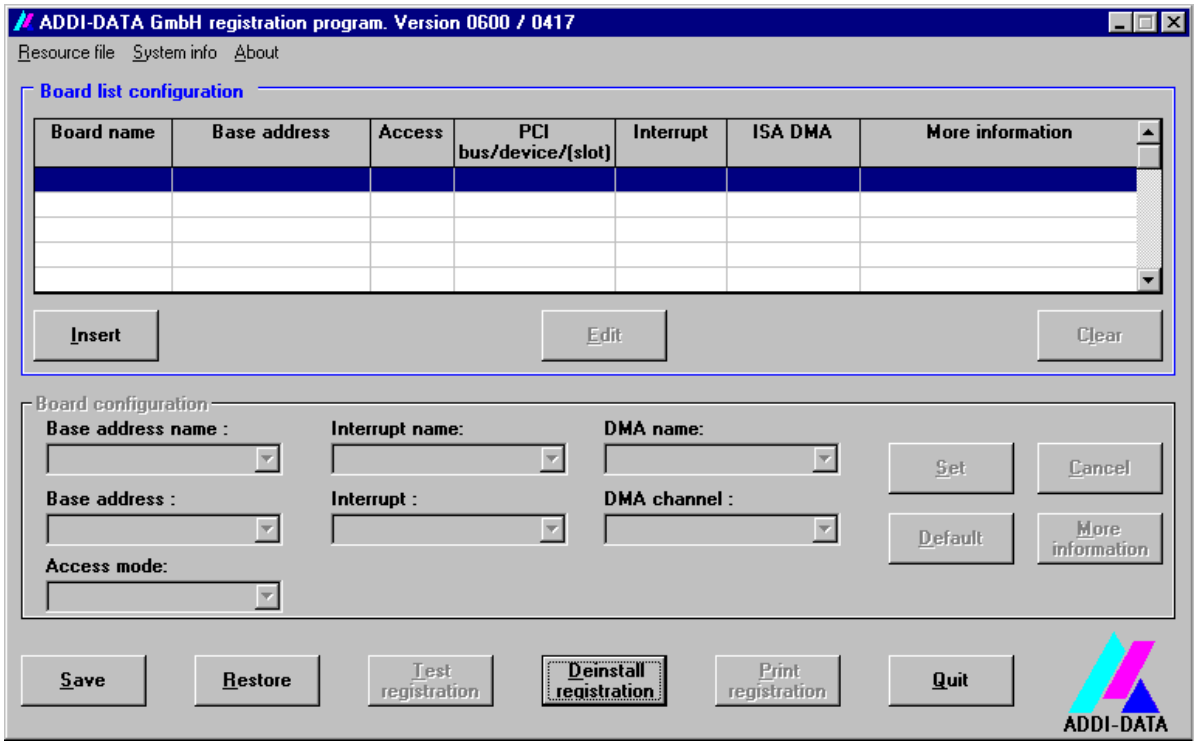
IMPORTANT!
If you use one or several resources of the board, you cannot start the ADDIREG program.

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

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

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

Fig. 7-2: ADDIREG registration program



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

Board name:
Names of the different registered boards (e.g.: 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.

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

More information:
Additional information like the identifier string (e.g.: 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.

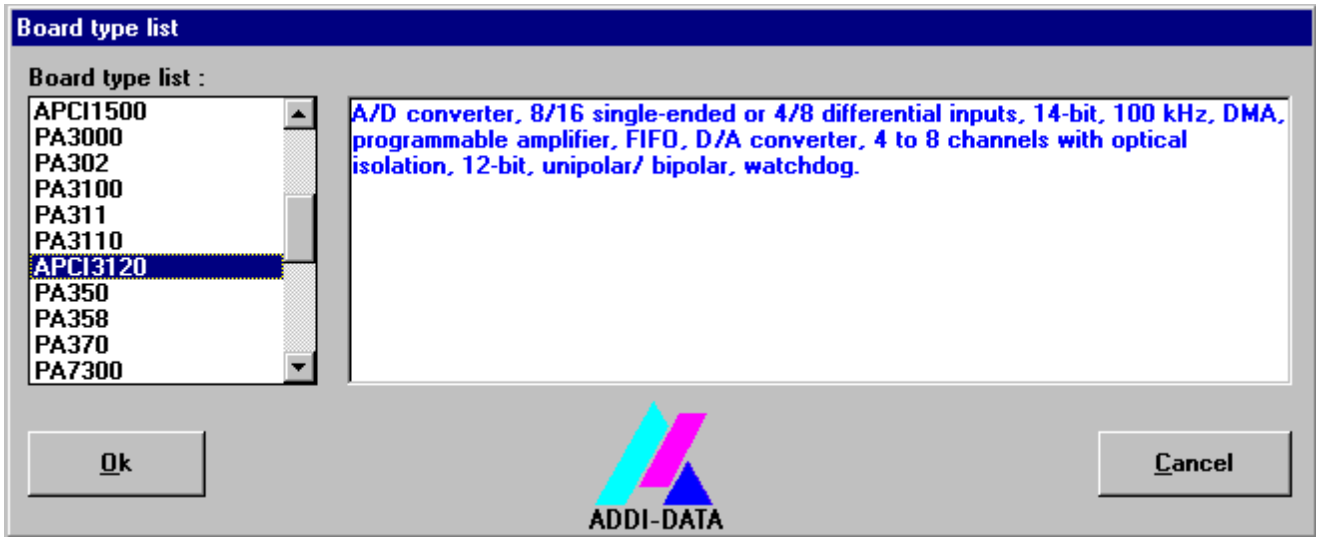
DMA channel:
Selection of the used DMA channel.

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:

Fig. 7-3: 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 parameterised 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.

---

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

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

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

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.

7.1.3 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 7-2 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.

7.1.4 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.

7.2 Installing the drivers

IMPORTANT!
Install first the ADDIREG program before installing or starting any other application of the board.

7.2.1 Installation under MS-DOS

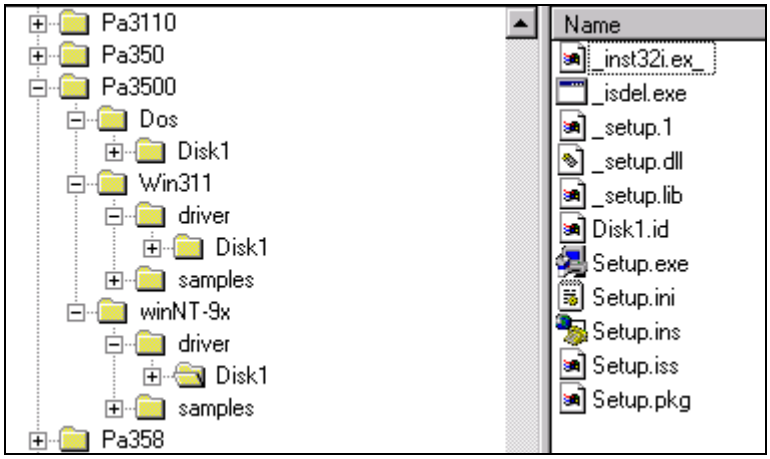
- Copy the contents of PA3500/Dos/Disk1 on a diskette.
- Insert the diskette into a driver and change to this drive.
- Enter <INSTALL>.

The installation program gives you further instructions.

7.2.2 Installation under Windows 3.11/NT/95/98

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

Fig. 7-4: Installing the driver



- Under PA3500/Win311/Driver/Disk1 or PA3500/winNT-9x/Driver/Disk1 start the set-up program "setup.exe" (double click).

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

7.3 Installing the samples

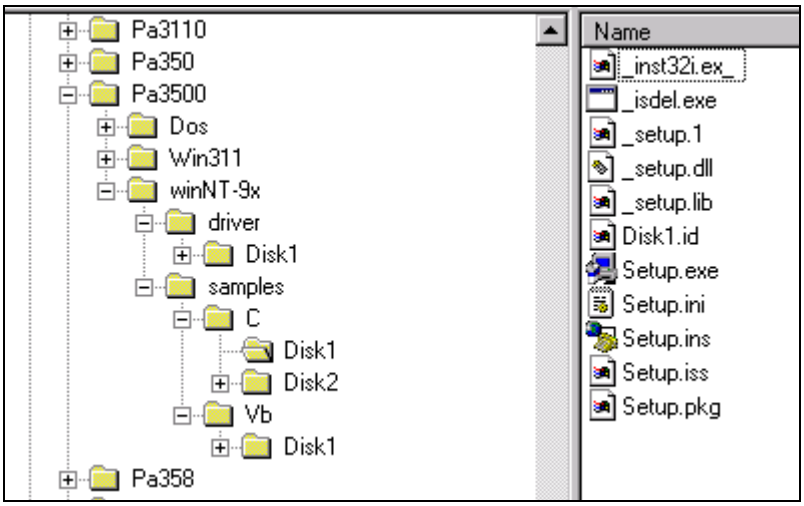
7.3.1 Installation under MS-DOS

Under DOS the software samples are automatically installed with the driver.

7.3.2 Installation under Windows 3.11/NT/95/98

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

Fig. 7-5: Installation of the samples



- Select the required programming language.
- Start the Set-up file setup.exe.
- 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.

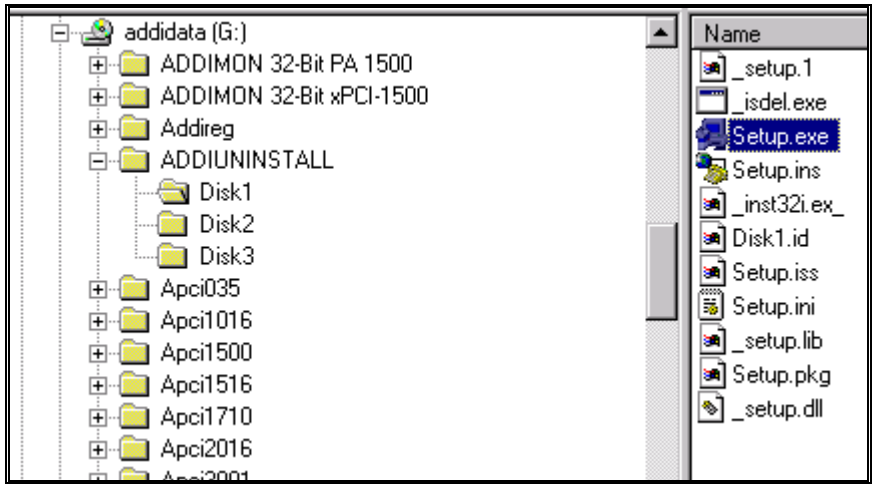
7.4 The ADDI-UNINSTALL program

7.4.1 Installation of ADDI-UNINSTALL

The ADDI_UNINSTALL program is delivered on the CD-ROM.

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

Fig. 7-6: Installation of the ADDI-UNINSTALL program

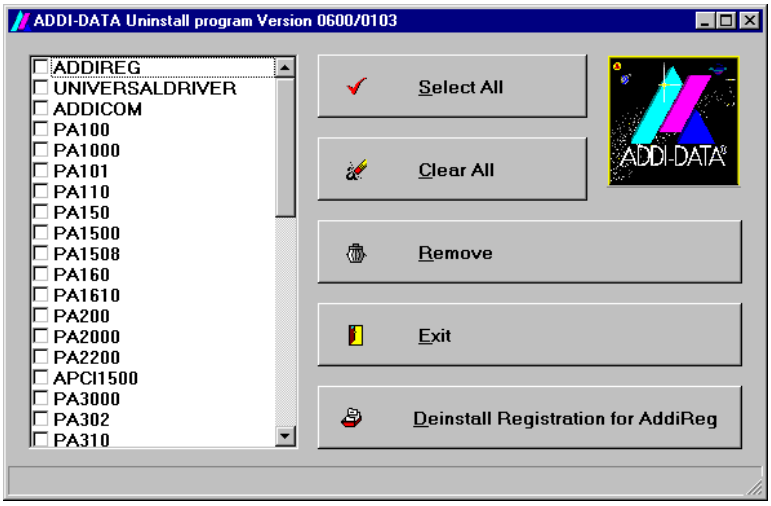


- Proceed as indicated on the screen.

7.4.2 Software uninstalling with ADDI-UNINSTALL

- Start the ADDI_UNINSTALL program.

Fig. 7-7: The ADDI_UNINSTALL program



- Select the software or the board driver(s) to be deinstalled and click in the corresponding check box.

- Click on "Remove". Proceed as indicated until the complete removal of the program.

Uninstall ADDIREG

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

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

7.5 Software downloads from the Internet

Do not hesitate to visit us or e-mail your questions.
Our Internet page is accessed:
- per e-mail: info@addi-data.de
- per Internet : http://www.addi-data.de. or
  http://www.addi-data.com

Free downloads of the standard software
You can download the latest version of the software for the board **PA 3500**.

8 CONNECTION TO THE PERIPHERAL

WARNING!
Emissions may be diffused and coupled through a wrong connection cable and could damage the operation and function safeties of your system.

We recommend to use our standard connection cable.

Make sure by the installation of the connection cable that:
- it is installed within a sufficient distance from sensitive analog signals
- the distance from potential interference sources (e.g.: frequency converters, supply circuits) is as long as possible.

If the outputs are operating at maximum load, you may install the connection cable with the appropriate cross section in a well-aerated room.

A voltage fall of 1.22 mV is possible on the connection cable and this may modify the measurement results by 1 LSB.

To avoid this error, you can calculate the maximum length of your cable (lmax with diameter A) according to the maximum voltage fall (Umax) possible for your application as follows.

R_{loop} = = 2 x l x ρ / A
ρ (copper)=0.01724 Ω mm^2/m

Condition
R_{loop} x I< U_{max}
U_{max} = 0.3448 Ω x mm^2/m x lmax / A x I

lmax= A x U_{max} / (0.3448 Ω x mm^2/m x I)

lmax = maximum length in mm
I = current through the conductor in Ampère
A = cross section of the conductor in mm^2

WARNING ! (Version PA 3500-8 and -4)
If more than 5 mA current flows through the load, the D/A converter may be damaged. (See technical data)

Notice the version of your board. The maximum output current is limited at 5 mA in the versions -8 and -4.
Rmax = 10 V / 5 mA = 2000 Ω

8.1 Connector pin assignment

IMPORTANT!
To avoid any confusion, both pins of a signal are displayed in front of one another on the connector (except from the trigger input on the SUB-D connector).

To avoid galvanic coupling the trigger input is to be connected to the ribbon connector through the SUB-D connector

Notice !

2 connections (Pin 6,25) are available for the trigger on the ribbon connector. If you do not use the digital inputs and outputs, you can also connect the trigger input on 2 pins (10,19) of the SUB-D connector.

Fig. 8-1: 37-pin SUB-D male connector

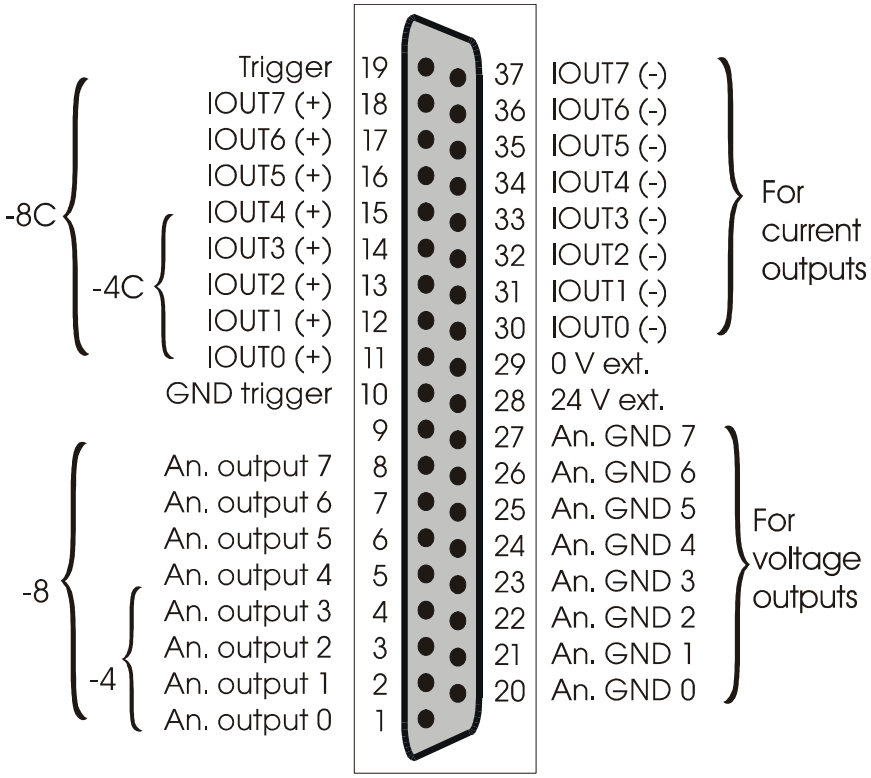
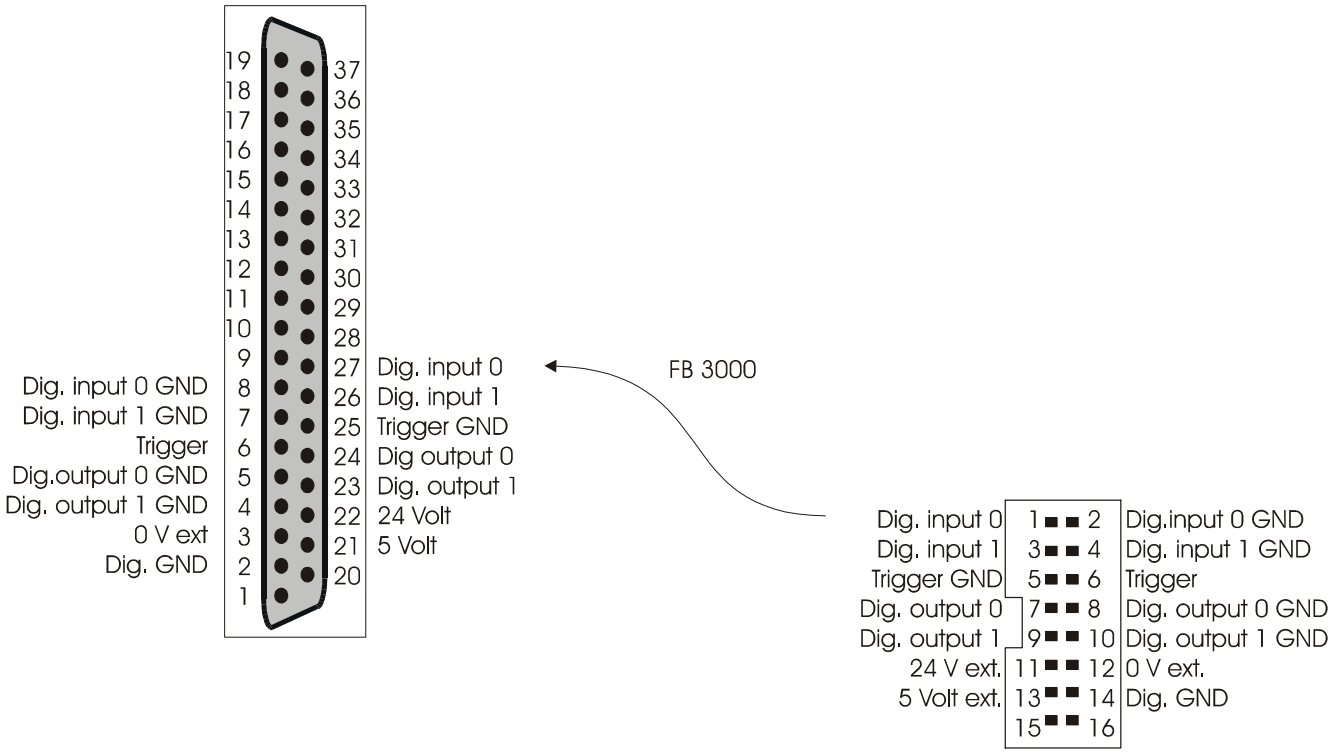
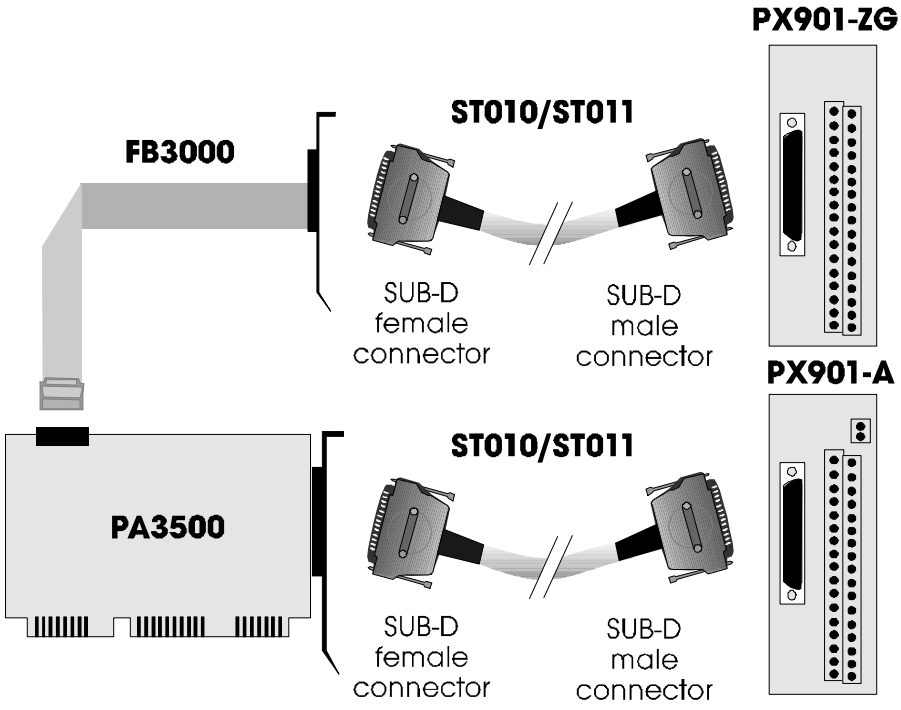


Fig. 8-2: Connection to the ribbon cable FB3000 (16-pin to 37-pin connector)



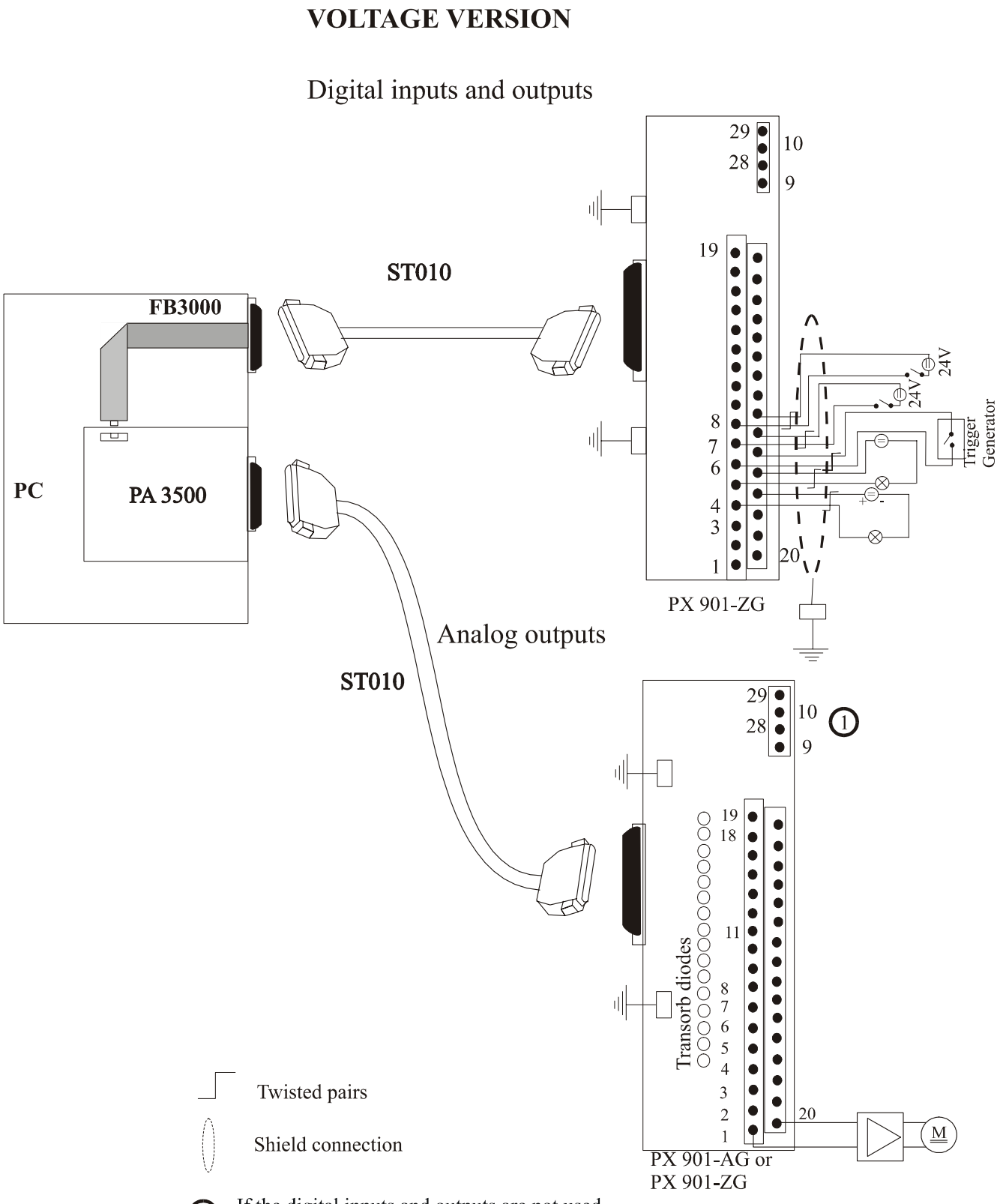
The 24 V lines on the front connector (pin 28) and on the connector for ribbon cable (pin 11) are connected to each other.

Fig. 8-3: Connection of the ribbon connector through the SUB-D connector

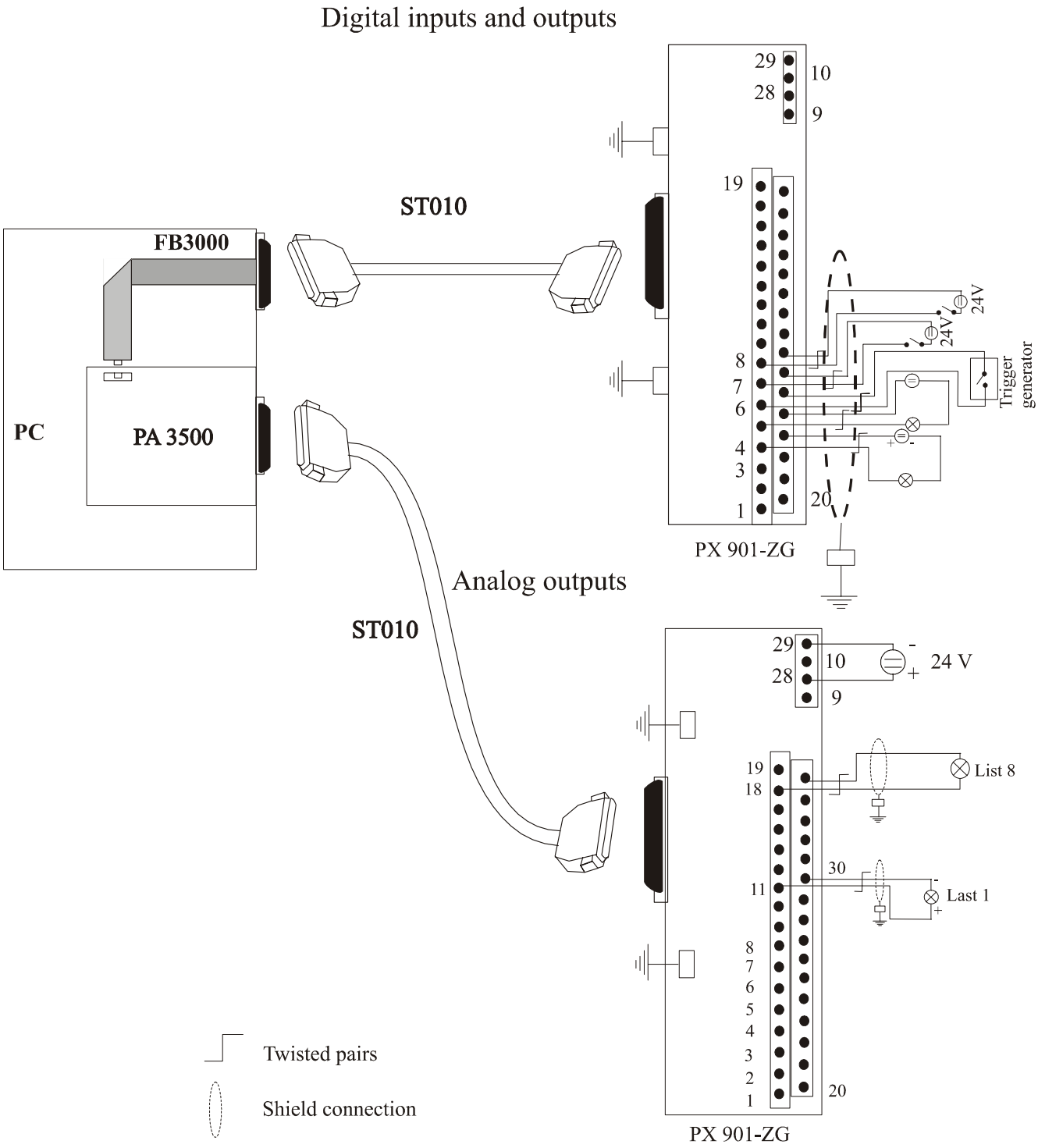


8.2 Connection examples

Fig. 8-4: Connection to the screw terminal board PX 901



CURRENT VERSION



9 BOARD FUNCTIONS

9.1 Analog outputs

The inputs are controlled by electronics through the PC bus. They modify the output voltage [1] or the output current [2] according to the programmed data bus information.
The used data bus information is transmitted to the corresponding command functions of the driver in the form of a number value with 14-bit decoding.
The A/D converter loads the number value and only returns the new value to the output after a strobe. (update).
The strobe generation depends on the programmed operation mode:

Strobe	Operating mode
Automatic	Simple mode
per command	Trigger and synchronous mode
per trigger	Trigger and synchronous mode

Table 9-1: Translation table

Output range	Current version			Voltage version	
Resolution	13-bit			14-bit	
Digital value D13..D0	0-20 mA	4-20 mA	5-25 mA	0-10 V	± 10 V
3FFF Hex	20 mA	20 mA	25 mA	10 V	+10 V
2FFF Hex	10 mA	12 mA	15 mA	5 V	+ 5 V
2000 Hex	0 mA	4 mA	5 mA	0 V	0 V
0000 Hex	-	-	-	-	-10 V
LSB	2,44 µA	1,96 µA	2,44 µA	1,22 mV	2,44 m V

Reset: After reset the outputs are set on 0 V.

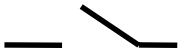
[1] Refers to the voltage version.
[2] Refers to the current version

9.2 Digital outputs

The outputs are used as electronic switches which modify their switching status according to the programmed bus information.

The used data bus information is transmitted to the corresponding command functions of the driver in the form of a number value with logic "1" or "0".

Warning!
When switching the digital outputs, make sure that you do not exceed the limit values given in the chapter 4 "technical data".

Logical state	Switching state	Meaning
1		Output transistor is conductive
0		Output transistor is high ohmic

Synchronous mode:
In this mode, the switching status of the outputs cannot be controlled through the data bus; The outputs indicate the status of the analog outputs operations.

Table 9-2: Status display in synchronous mode

Output 1 = Ready	Output 2 = Load	Strobe-effect	State
0	**1**	-	Transmission of new values
1	**1**	Update	Waiting for strobe
1	**0**	-	Update has occurred

Reset: After reset the outputs are high ohmic.

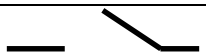
9.3 Digital inputs

The inputs acquire external signal status: the PC loads an input information per driver command as a number value (1 or 0). The user can read if inputs signals higher than 17 V are transmitted to the inputs.

Logical state	Meaning
1	Entering signal voltage > 17 V
0	Entering signal voltage < 17 V

9.4 Trigger

The trigger is a digital input which determines the status of the connected mechanic [1] or electronic switches. The open [2] or closed status of the switches is converted in logic signals and is translated by the corresponding reading function of the driver as number value "1" or "0".

Switching state	Logic	Meaning
	0	**Trigger is active**
	**1**	**No trigger** [3]

Update: The trigger function consists in releasing at a fixed period of time the loaded analog value. (update by closing)

IMPORTANT !
The trigger is used for trigger function as a digital input with update function only in trigger and synchronous mode.

Table 9-3: Trigger function according to the operating mode

Operating mode	Interrupt	Function
SIMPLE MODE	release by Enable	Digital input, interrupt source
TRIGGER MODE	release by Enable	Digital input with update function and interrupt source
SYNCHRONOUS MODE	release by Enable	

[1] For mechanic switches a debounce device is necessary

[2] For electronic switches "open" means high-ohmic

[3] "Regular" status, when no switch is connected for the triggering.

9.5 Watchdog

The watchdog function controls the function errors in the application.

The updating of the outputs triggers the watchdog.
If the watchdog has been activated per software and if no retrigger occurs after 4.2 s, the outputs switch off and cannot be activated.

3 different states of the watchdog can be read at any time per software.

"OFF" The watchdog is switched off. It has no influence on the outputs.

"ON" The watchdog is active. It controls the program course and switches the outputs off by timeout (> 4.2 s).

"Alarm" After running down of the watchdog (timeout > 4,2) the alarm bit is set. The outputs are not reset; an interrupt can be generated.

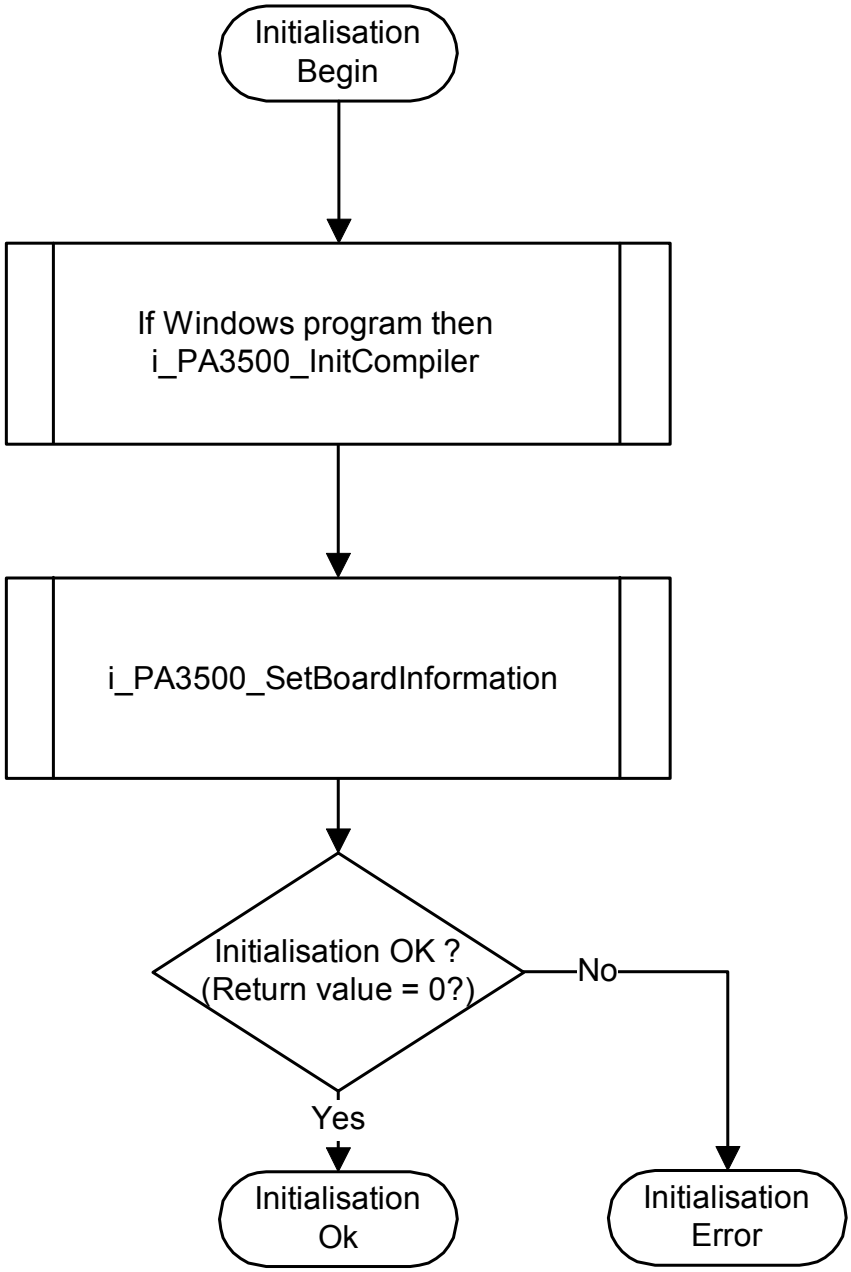
Reset settings
After PC start the watchdog is off. The watchdog time is 4,2 s.

10 EXAMPLES

10.1 Initialisation

10.1.1 Initialisation of one PA 3500 under DOS and Windows 3.11

a) Flow chart

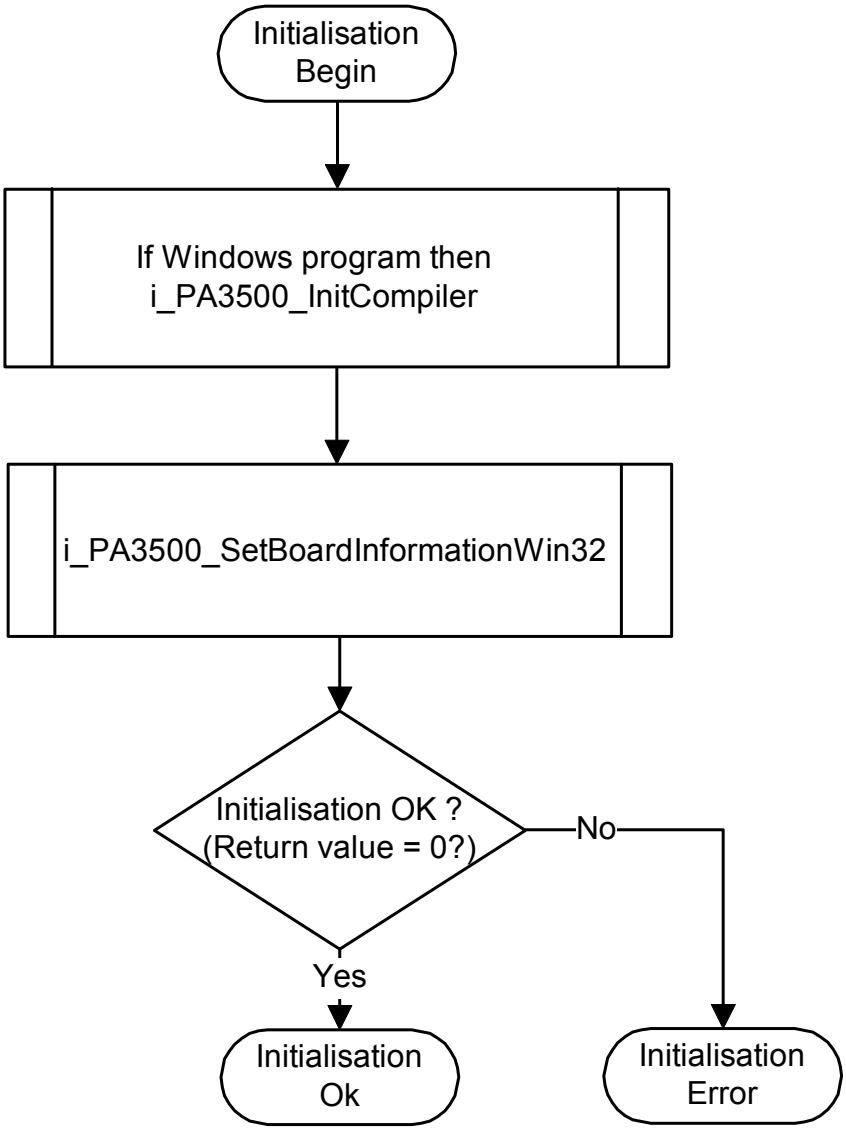


b) Example in C

```
int Initialisation(unsigned char *pb_BoardHandle)
    {
    #ifdef _Windows
       i_PA3500_InitCompiler (DLL_COMPILER_C);
    #endif

    if(i_PA3500_SetBoardInformation (0x390,
                                     3,  // IRQ3
                                     8,  // Analog output Number
                                     PA3500_SIMPLE_MODE,
                                     pb_BoardHandle) == 0)
       {
       return (0);         /* OK */
       }
    else
       {
       return (-1);        /* ERROR */
       }
    }
```

10.1.2 Initialisation of one PA 3500 under Windows NT/95

a) Flow chart

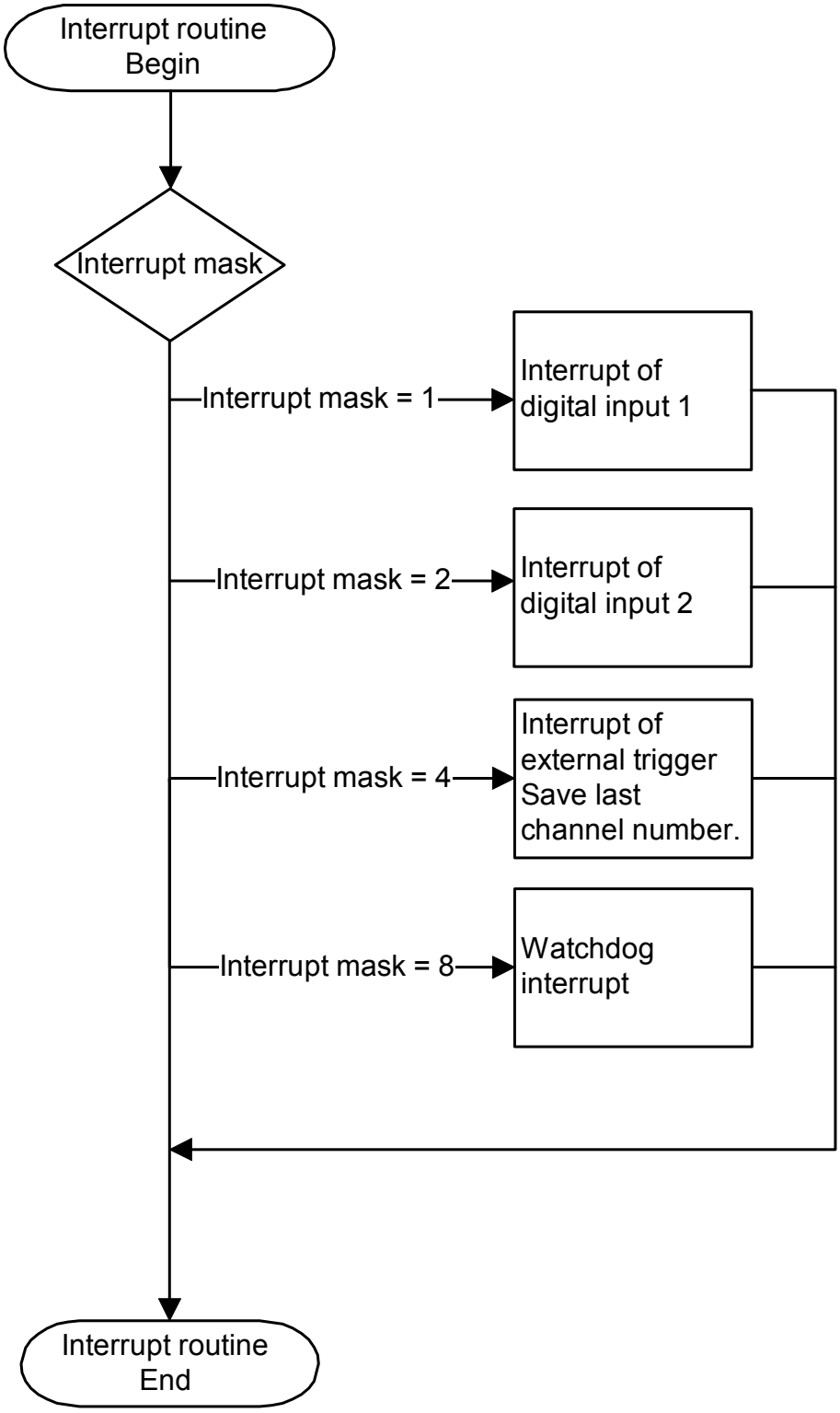


b) Example in C

```
int Initialisation(unsigned char *pb_BoardHandle)
    {
    if (i_PA3500_InitCompiler (DLL_COMPILER_C) == 0)
       {
       if(i_PA3500_SetBoardInformationWin32 (“PA3500-00“,
                                             8,
                                             PA3500_SIMPLE_MODE
                                             pb_BoardHandle) == 0)
          {
          return (0);      /* OK */
          }
       else
          {
          return (-1);     /* ERROR */
          }
       }
    else
       {
       return (-1);        /* ERROR */
       }
    }
```

10.2 Interrupt

a) Flow chart



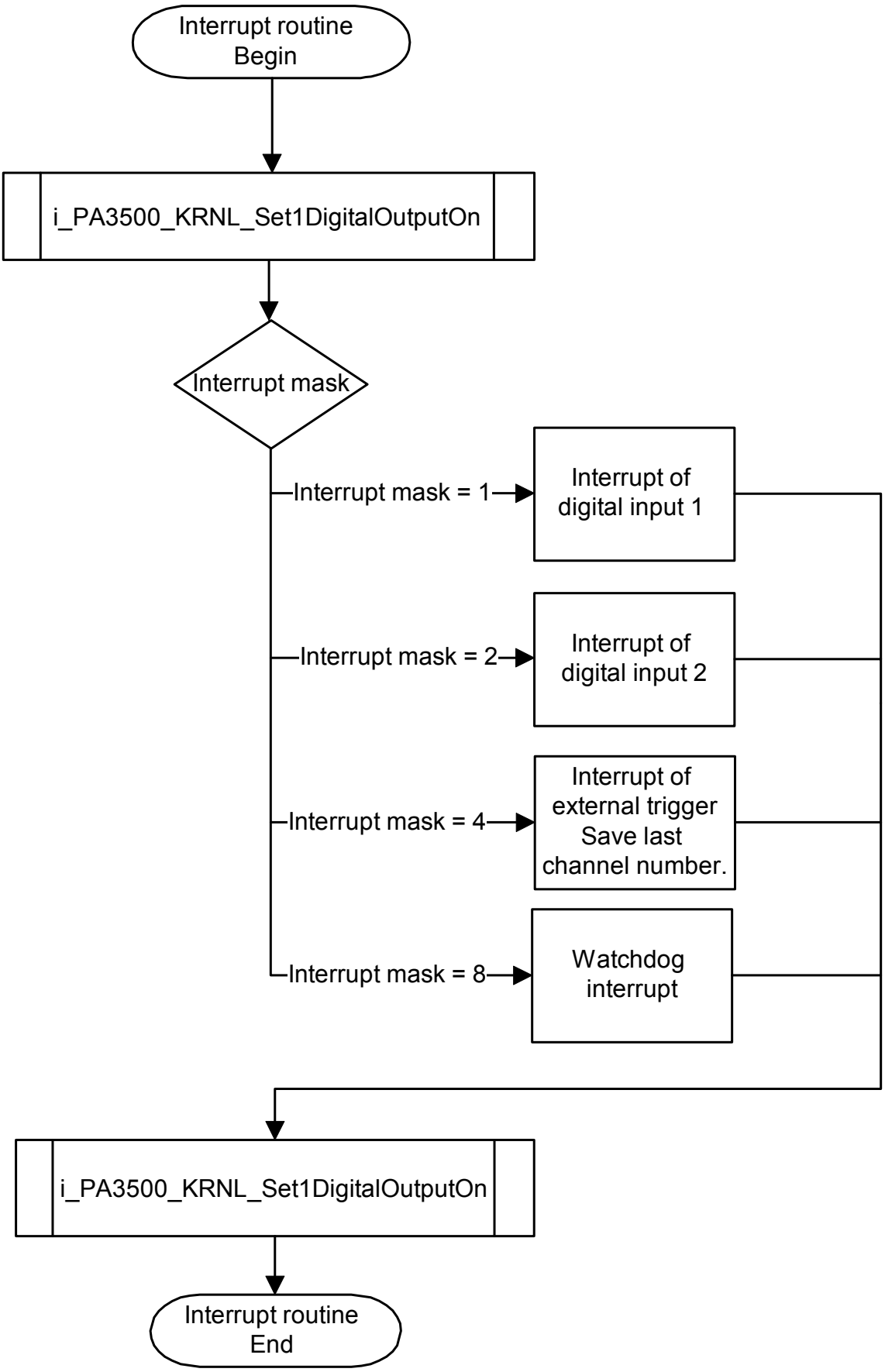
b) Example in C for DOS and Windows 3.1x

```
unsigned char b_ReceiveInterruptSource  = 0;  /* Interrupt flag                */
unsigned char b_LastChannelNbr = 0;    /* Number of the last channel triggered */

_VOID_ v_InterruptRoutine (BYTE_ b_BoardHandle, BYTE_ b_InterruptMask, BYTE_ b_LastChannelNumber)
     {

     switch(b_InterruptMask)
        {
        case 1:
                /* DIGITAL INPUT 1 interrupt */
               b_ReceiveInterruptSource = 1;
               break;
        case 2:
               /* DIGITAL INPUT 2 interrupt */
                b_ReceiveInterruptSource = 2;
               break;
         case 4:
               /* EXTERN TRIGGER interrupt */
               b_LastChannelNbr = b_LastChannelNumber;
               b_ReceiveInterruptSource = 4;
               break;
        case 8:
               /* WATCHDOG RUN DOWN */
               b_ReceiveInterruptSource = 8;
               break;
         default :
              b_ReceiveInterruptSource = 0;
               break;
        }

   }
```

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

```
unsigned char b_ReceiveInterruptSource = 0;   /* Interrupt flag */
unsigned char b_LastChannelNbr = 0;           /* Number of the last channel triggered
*/

_VOID_ v_InterruptRoutine ( BYTE_ b_BoardHandle, BYTE_ b_InterruptMask,
                            BYTE_ b_LastChannelNbr, BYTE_ b_UserCallingMode,
                            VOID *pv_UserSharedMemory)
   {

   switch(b_InterruptMask)
        {
        case 1:
                /* DIGITAL INPUT 1 interrupt */
               b_ReceiveInterruptSource = 1;
               break;
        case 2:
               /* DIGITAL INPUT 2 interrupt */
                b_ReceiveInterruptSource = 2;
               break;
         case 4:
               /* EXTERN TRIGGER interrupt */
               b_LastChannelNbr = b_LastChannelNumber;
               b_ReceiveInterruptSource = 4;
               break;
        case 8:
               /* WATCHDOG RUN DOWN */
               b_ReceiveInterruptSource = 8;
               break;
         default :
              b_ReceiveInterruptSource = 0;
               break;
        }

   }
```

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



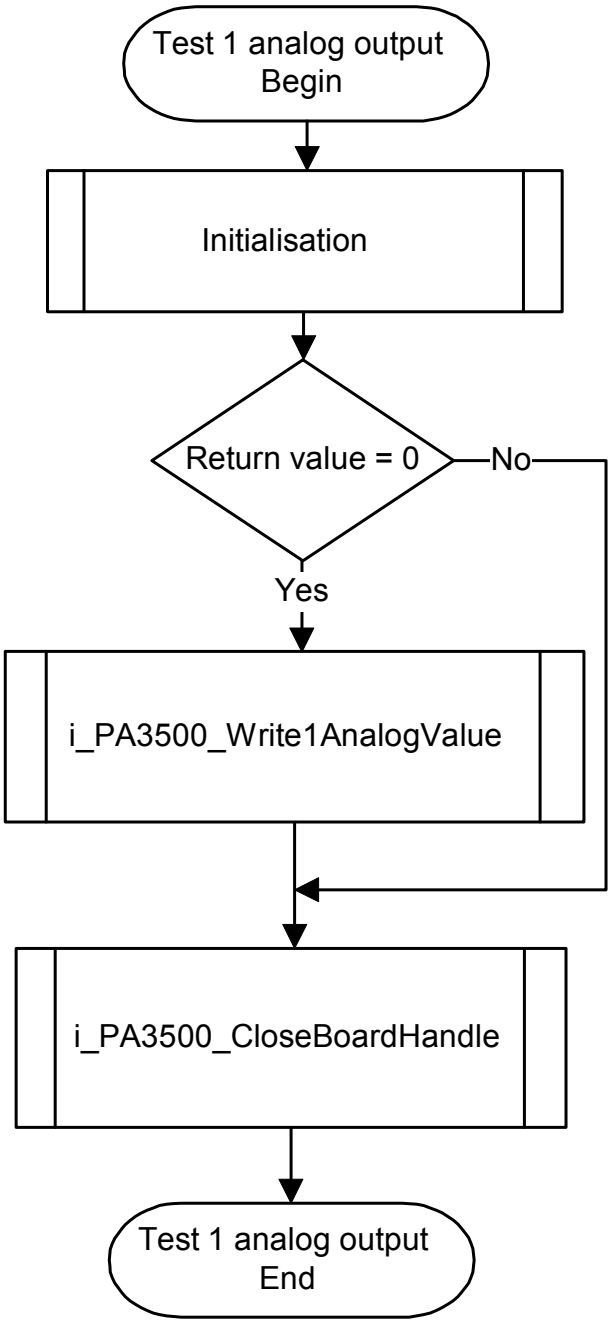
e) Example in C for Windows NT/95 (synchronous mode)

```
typedef struct
{
   unsigned char b_ReceiveInterruptSource;   /* Interrupt flag */
   unsigned char b_LastChannelNbr;           /* Number of the last channel triggered */
}str_UserStruct;
str_UserStruct *ps_GlobalUserStruct;

_VOID_ v_InterruptRoutine ( BYTE_ b_BoardHandle,BYTE_ b_InterruptMask,
                            BYTE_ b_LastChannelNumber,
                           BYTE_ b_UserCallingMode,VOID *pv_UserSharedMemory)
{

   str_UserStruct *ps_UserStruct = (str_UserStruct *) pv_UserSharedMemory;

   i_PA3500_KRNL_Set1DigitalOutputOn(0x390,1);
   if ((b_InterruptMask&1) == 1) /* DIGITAL INPUT 1 interrupt */
      {
       ps_UserStruct->b_ReceiveInterruptSource = 1;
      }
   if ((b_InterruptMask&2) == 2) /* DIGITAL INPUT 2 interrupt */
      {
       ps_UserStruct->b_ReceiveInterruptSource = 2;
       }
   if ((b_InterruptMask&4) == 4) /* EXTERN TRIGGER interrupt */
      {
      ps_UserStruct->b_LastChannelNbr = b_LastChannelNumber;
      ps_UserStruct->b_ReceiveInterruptSource = 4;
      }
   if ((b_InterruptMask&8) == 8) /* WATCHDOG interrupt */
      {
       ps_UserStruct->b_ReceiveInterruptSource = 8;
      }
   i_PA3500_KRNL_Set1DigitalOutputOn(0x390,2);

}
```

10.3 Analog output channels

10.3.1 Testing 1 analog output (simple mode)

a) Flow chart

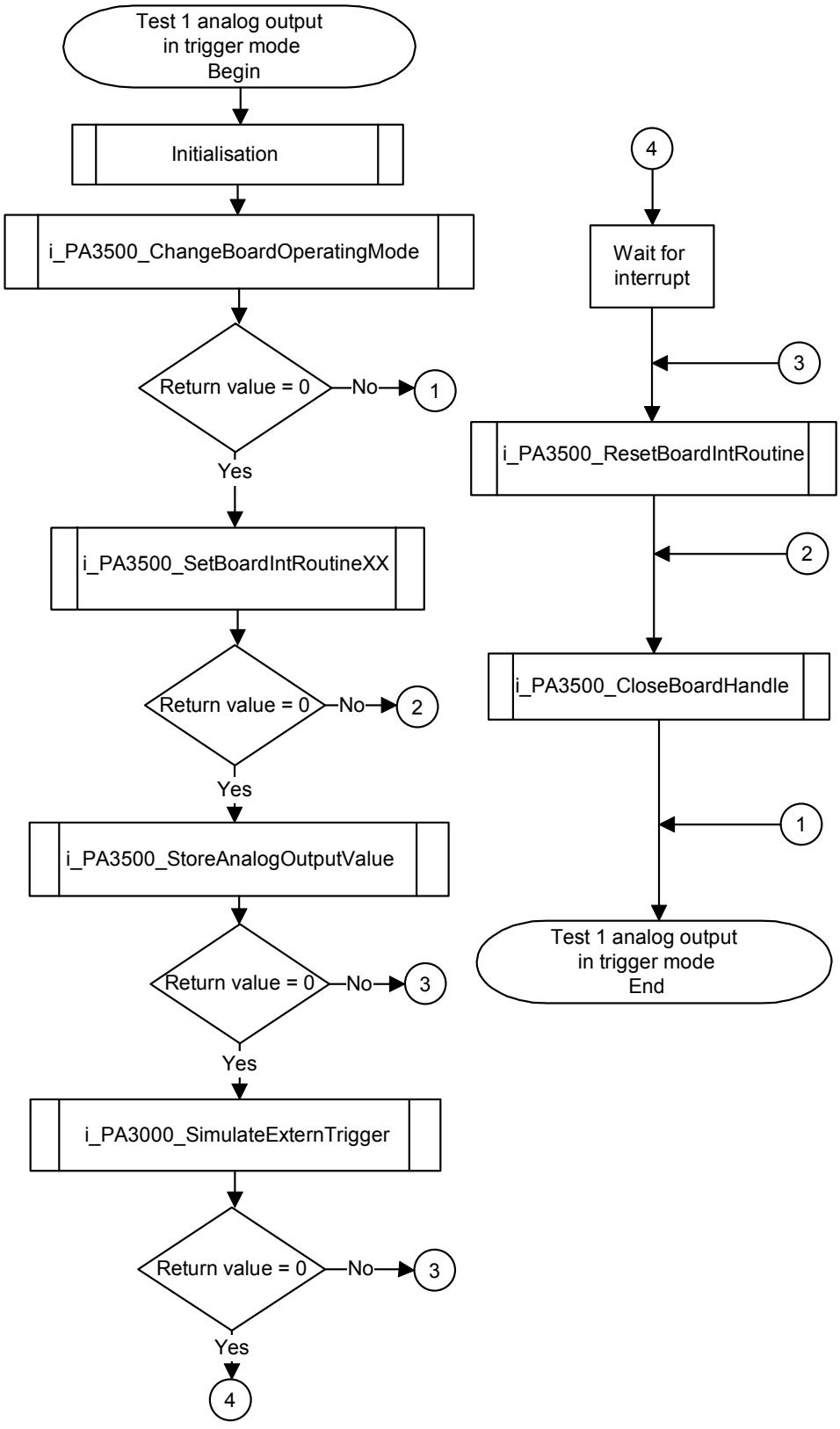


b) Example in C

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

    if (Initialisation (&b_BoardHandle) == 0)
       {
       if (i_PA3500_Write1AnalogValue  (b_BoardHandle,
                                        PA3500_CHANNEL_0,
                                        PA3500_UNIPOLAR,
                                        4095)  == 0)
          {
          printf ("\n output 0 = 10 V");
          }
       else
          {
          printf ("i_PA3500_Write1AnalogValue error");
          }
       i_PA3500_CloseBoardHandle (b_BoardHandle);
       }
    else
       {
       printf ("Initialisation error");
       }
    }
```

10.3.2 Testing one analog output (trigger mode)

a) Flow chart



b) Example in C for DOS

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

   if (Initialisation(&b_BoardHandle) == 0)
      {
      if(i_PA3500_ChangeBoardOperatingMode(b_BoardHandle,PA3500_TRIGGER_MODE) != 0)
         return(0);
      if (i_PA3500_SetBoardIntRoutineDos(b_BoardHandle, v_InterruptRoutine) == 0)
        {
        b_ReceiveInterruptSource = 0;
        if (i_PA3500_StoreAnalogOutputValue (b_BoardHandle,PA3500_ENABLE,PA3500_CHANNEL0,
                                             PA3500_UNIPOLAR,1024)==0)
          {
          printf("\n Press a key to simulate the trigger");
          getch();
          if (i_PA3500_SimulateExternalTrigger(b_BoardHandle) == 0)
             {
              while (b_ReceiveInterruptSource == 0);
              printf("\n interrupt source = %u, output 0 = 2.5V",b_ReceiveInterruptSource);

             }
          else
             printf("\n i_PA3500_SimulateExternalTrigger error ");

          }
        else
           printf("\n i_PA3500_StoreAnalogOutputValue error");
        i_PA3500_ResetBoardIntRoutine(b_BoardHandle);
        }
      else
        printf("\n Interrupt routine initialisation error");
      i_PA3500_CloseBoardHandle(b_BoardHandle);
      }
   else
      printf("\n Initialisation error");
   }
```

c) Example in C for Windows 3.1x

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

   if (Initialisation(&b_BoardHandle) == 0)
      {
      if(i_PA3500_ChangeBoardOperatingMode(b_BoardHandle,PA3500_TRIGGER_MODE) != 0)
         return(0);
      if (i_PA3500_SetBoardIntRoutineWin16(b_BoardHandle, v_InterruptRoutine) == 0)
        {
        b_ReceiveInterruptSource = 0;
        if (i_PA3500_StoreAnalogOutputValue (b_BoardHandle,PA3500_ENABLE,PA3500_CHANNEL0,
                                             PA3500_UNIPOLAR,1024)==0)
           {
           printf("\n Press a key to simulate the trigger");
           getch();
           if (i_PA3500_SimulateExternalTrigger(b_BoardHandle) == 0)
              {
               while (b_ReceiveInterruptSource == 0);
               printf("\n interrupt source = %u, output 0 = 2.5V",b_ReceiveInterruptSource);

              }
           else
              printf("\n i_PA3500_SimulateExternalTrigger error ");

           }
        else
           printf("\n i_PA3500_StoreAnalogOutputValue error");
        i_PA3500_ResetBoardIntRoutine(b_BoardHandle);
        }
      else
        printf("\n Interrupt routine initialisation error");
      i_PA3500_CloseBoardHandle(b_BoardHandle);
      }
   else
      printf("\n Initialisation error");
   }
```

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

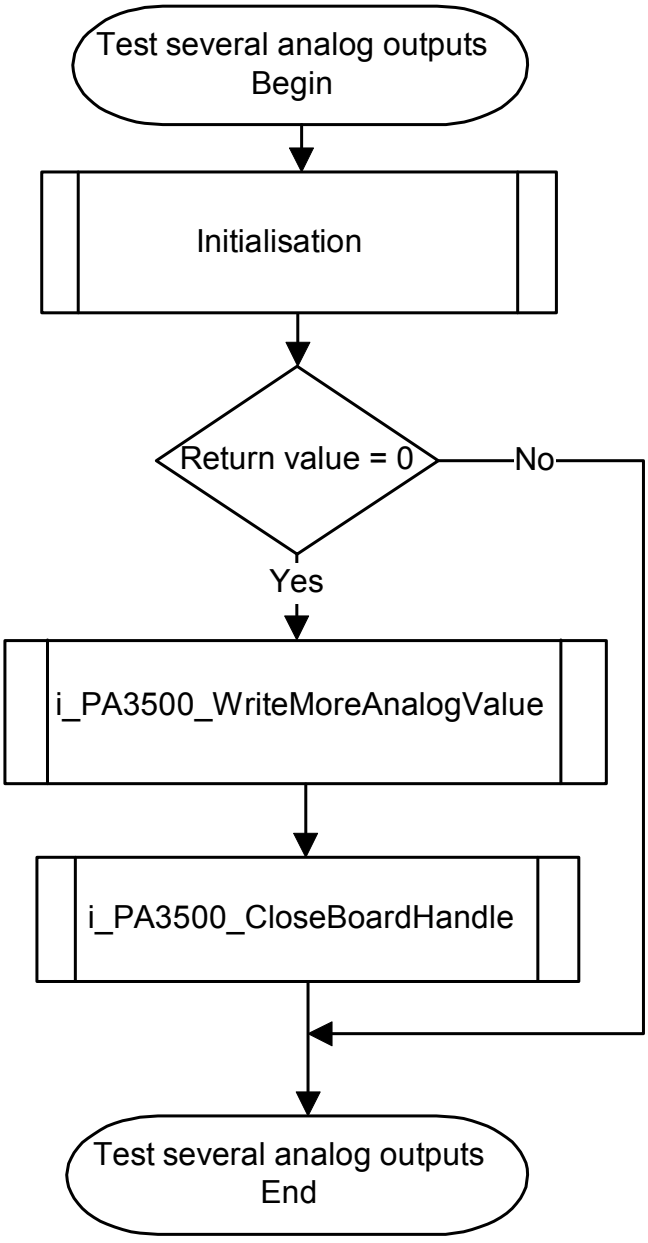
```
void main(void)
   {
   unsigned char b_BoardHandle;

   if (Initialisation(&b_BoardHandle) == 0)
      {
      if(i_PA3500_ChangeBoardOperatingMode(b_BoardHandle,PA3500_TRIGGER_MODE) != 0)
         return(0);
      if (i_PA3500_SetBoardIntRoutineWin32(b_BoardHandle,PA3500_ASYNCHRONOUS_MODE,0,
                                           NULL, v_InterruptRoutine) == 0)
        {
        b_ReceiveInterruptSource = 0;
        if (i_PA3500_StoreAnalogOutputValue (b_BoardHandle,PA3500_ENABLE,PA3500_CHANNEL0,
                                             PA3500_UNIPOLAR,1024)==0)
           {
           printf("\n Press a key to simulate the trigger");
           getch();
           if (i_PA3500_SimulateExternalTrigger(b_BoardHandle) == 0)
              {
               while (b_ReceiveInterruptSource == 0);
               printf("\n interrupt source = %u, output 0 = 2.5V",b_ReceiveInterruptSource);

              }
           else
              printf("\n i_PA3500_SimulateExternalTrigger error ");

           }
        else
           printf("\n i_PA3500_StoreAnalogOutputValue error");
        i_PA3500_ResetBoardIntRoutine(b_BoardHandle);
        }
      else
        printf("\n Interrupt routine initialisation error");
      i_PA3500_CloseBoardHandle(b_BoardHandle);
      }
   else
      printf("\n 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_PA3500_ChangeBoardOperatingMode(b_BoardHandle,PA3500_TRIGGER_MODE) != 0)
         return(0);
      if (i_PA3500_SetBoardIntRoutineWin32(b_BoardHandle,PA3500_SYNCHRONOUS_MODE,
                                     sizeof(str_UserStruct),(void **)&ps_GlobalUserStruct,
                                          v_InterruptRoutine) == 0)
        {
        ps_GlobalUserStruct -> b_ReceiveInterruptSource = 0;
        if (i_PA3500_StoreAnalogOutputValue (b_BoardHandle,PA3500_ENABLE,PA3500_CHANNEL0,
                                           PA3500_UNIPOLAR,1024)==0)
           {
           printf("\n Press a key to simulate the trigger");
           getch();
           if (i_PA3500_SimulateExternalTrigger(b_BoardHandle) == 0)
              {
                while (ps_GlobalUserStruct -> b_ReceiveInterruptSource == 0);
         printf("\n interrupt source = %u, output 0 = 2.5V", ps_GlobalUserStruct ->
                                                    b_ReceiveInterruptSource);

              }
           else
              printf("\n i_PA3500_SimulateExternalTrigger error ");

           }
        else
           printf("\n i_PA3500_StoreAnalogOutputValue error");
        i_PA3500_ResetBoardIntRoutine(b_BoardHandle);
        }
      else
        printf("\n Interrupt routine initialisation error");
      i_PA3500_CloseBoardHandle(b_BoardHandle);
      }
   else
      printf("\n Initialisation error");
   }
```

10.3.3 Testing several analog outputs (simple mode)

a) Flow chart



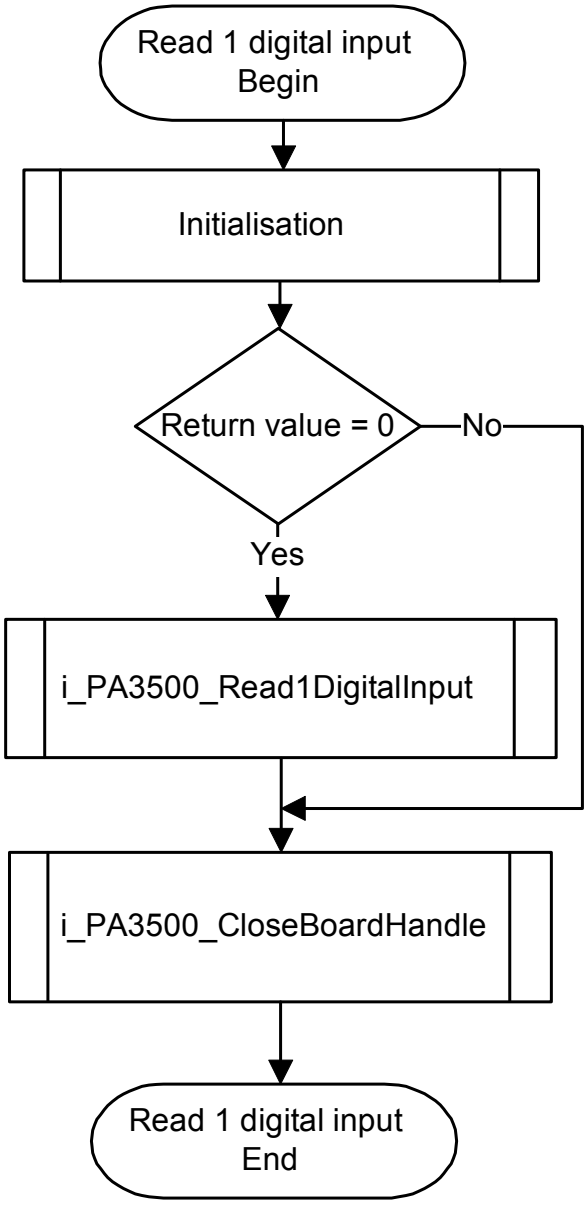
b) Example in C

```
void main (void)
     {
      unsigned char b_BoardHandle;
      unsigned char b_Polar [4];
      unsigned int ui_WriteValueArray [4];

     b_Polar[0] = PA3500_UNIPOLAR;ui_WriteValueArray[0] = 0;
     b_Polar[1] = PA3500_UNIPOLAR;ui_WriteValueArray[1] = 2047;
     b_Polar[2] = PA3500_UNIPOLAR;ui_WriteValueArray[2] = 4095;
     b_Polar[3] = PA3500_UNIPOLAR;ui_WriteValueArray[3] = 0;


     if (Initialisation (&b_BoardHandle) == 0)
        {
        if (i_PA3500_WriteMoreAnalogValue     (b_BoardHandle,PA3500_CHANNEL0,4,
                                               b_Polar,ui_WriteValueArray)  == 0)
           {
      printf ("Output 0 value=0V \nOuptut 1 value=5V \nOutput 2 value=10V\nOutput 3 value=0V");
           }
        else
           {
           printf ("i_PA3500_WriteMoreAnalogValue error");
           }
        i_PA3500_CloseBoardHandle (b_BoardHandle);
        }
     else
        {
        printf ("Initialisation error");
        }
     }
```

10.4 Digital inputs

10.4.1 Reading one digital input

a) Flow chart

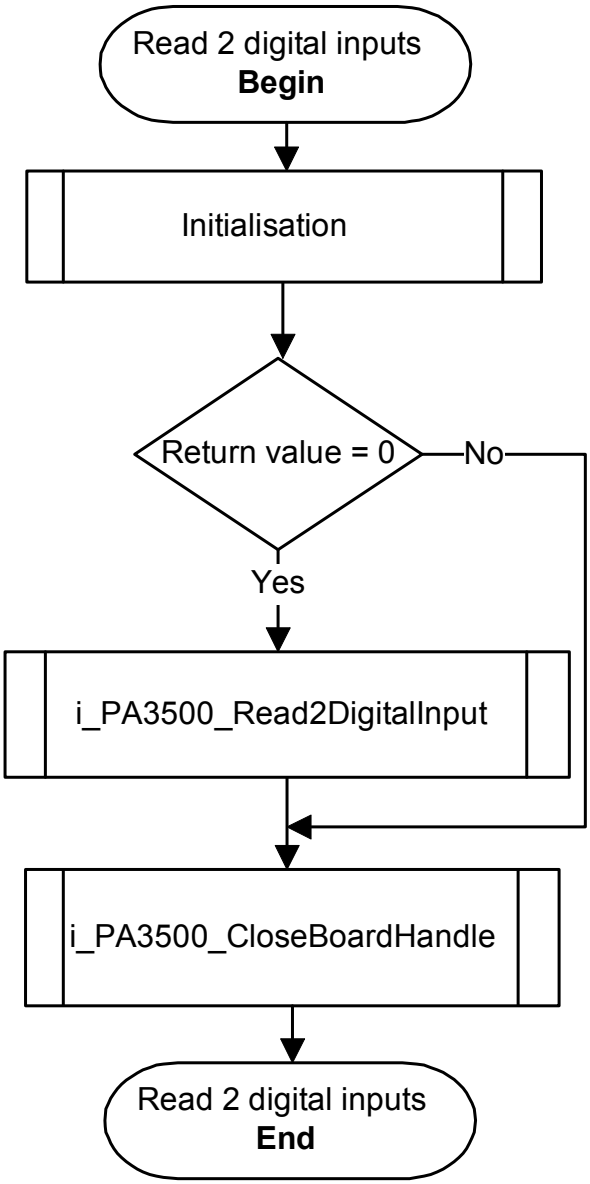


b) Example in C

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

     if (Initialisation (&b_BoardHandle) == 0)
        {
        if (i_PA3500_Read1DigitalInput  (b_BoardHandle,
                                         1
                                         &b_ReadValue)  == 0)
           {
           printf (“b_ReadValue = %u“, b_ReadValue);
           }
        else
           {
           printf (“i_PA3500_Read1DigitalInput error“);
           }
        i_PA3500_CloseBoardHandle (b_BoardHandle);
        }
     else
        {
        printf (“Initialisation error“);
        }
     }
```

10.4.2 Reading 2 digital inputs

a) Flow chart

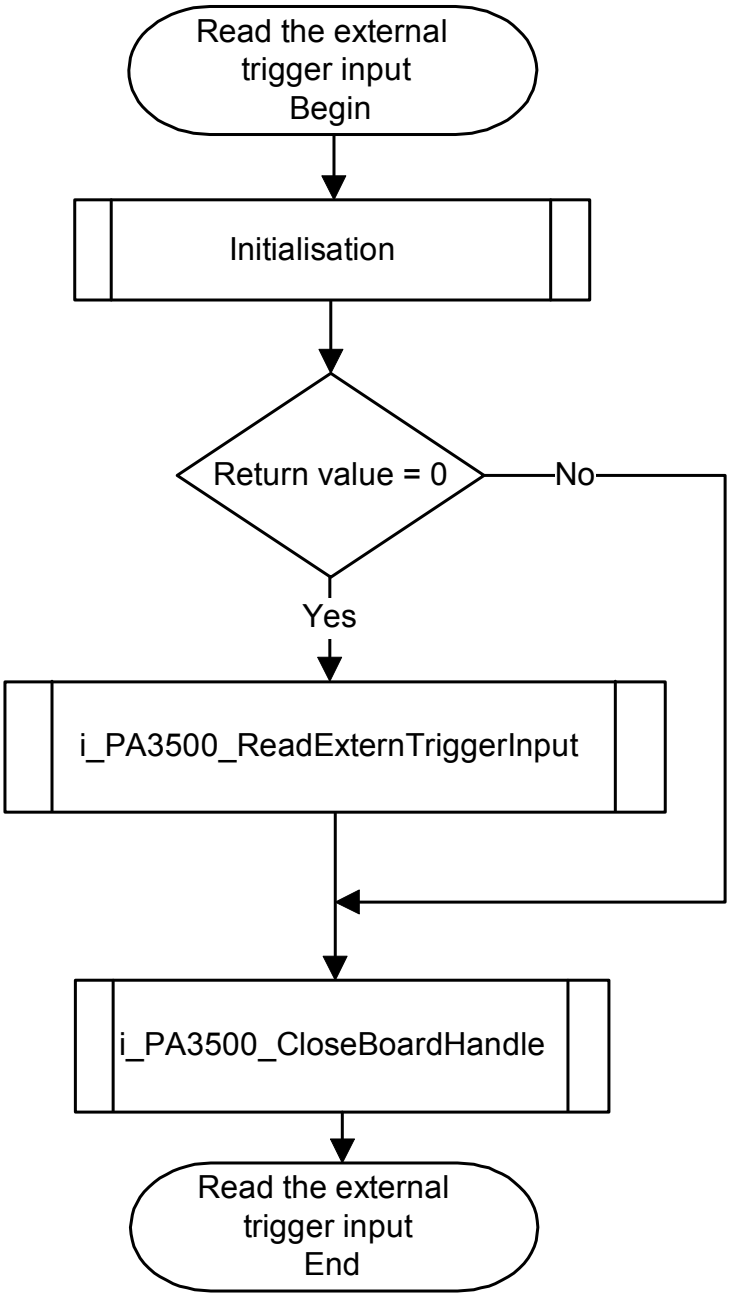


b) Example in C

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

     if (Initialisation (&b_BoardHandle) == 0)
        {
        if (i_PA3500_Read2DigitalInput  (b_BoardHandle,
                                         &b_ReadValue)  == 0)
           {
           printf ("b_ReadValue = %u", b_ReadValue);
           }
        else
           {
           printf ("i_PA3500_Read2DigitalInput error");
           }
        i_PA3500_CloseBoardHandle (b_BoardHandle);
        }
     else
        {
        printf ("Initialisation error");
        }
     }
```

10.4.3 Reading the external trigger input

a) Flow chart

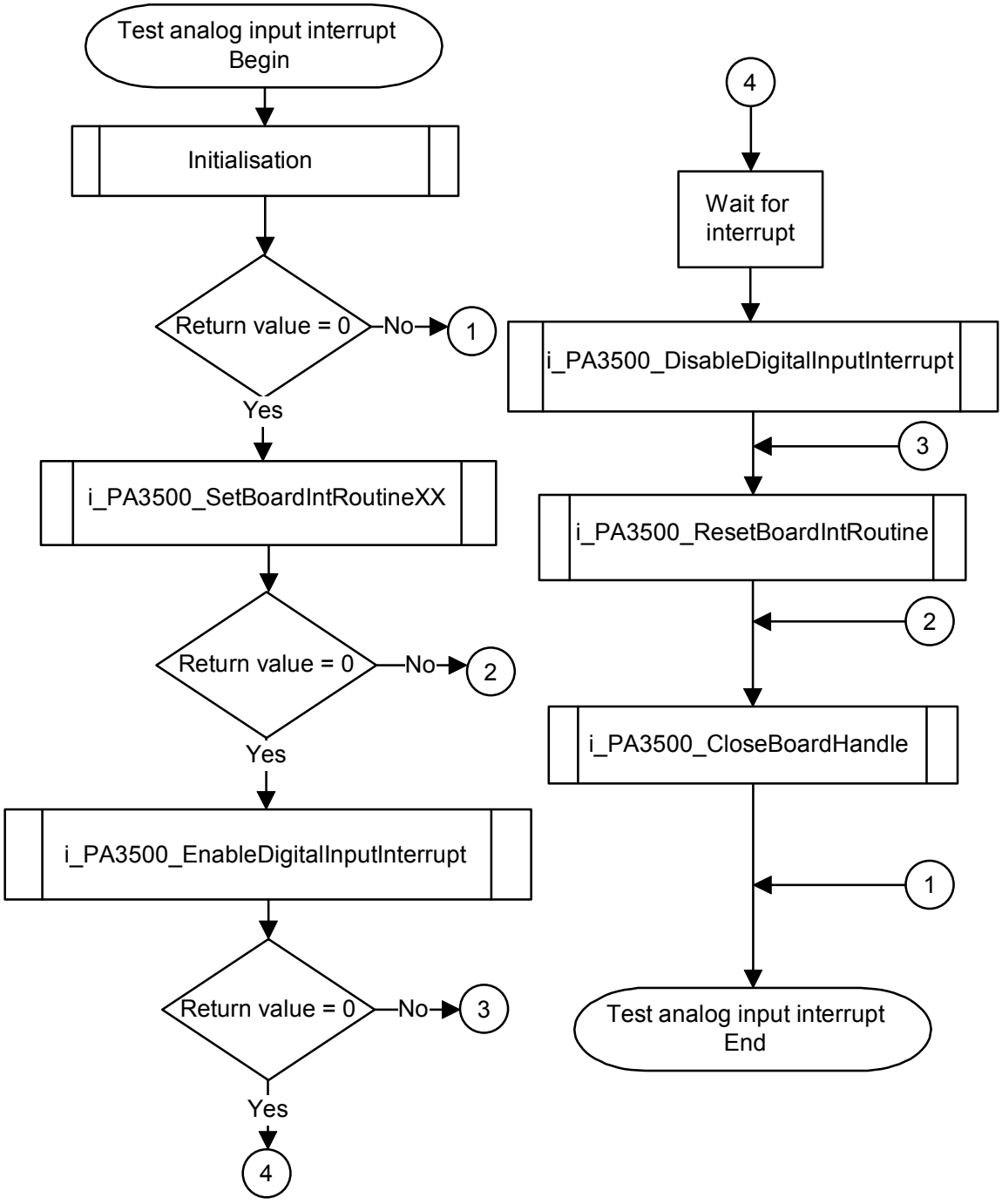


b) Example in C

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

     if (Initialisation (&b_BoardHandle) == 0)
        {
        if (i_PA3500_ReadExternTriggerInput
                                        (b_BoardHandle,
                                         &b_ReadValue)  == 0)
           {
           printf (“b_ReadValue = %u“, b_ReadValue);
           }
        else
           {
           printf (“i_PA3500_ReadExternTriggerInput error“);
           }
        i_PA3500_CloseBoardHandle (b_BoardHandle);
        }
     else
        {
        printf (“Initialisation error“);
        }
     }
```

10.4.4 Interrupt on the digital inputs

a) Flow chart



b) Example in C for DOS

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

   if (Initialisation(&b_BoardHandle) == 0)
      {
      if (i_PA3500_SetBoardIntRoutineDos(b_BoardHandle, v_InterruptRoutine) == 0)
        {
        b_ReceiveInterruptSource = 0;
        if (i_PA3500_EnableDigitalInputInterrupt(b_BoardHandle,3)==0)
           {
           while (b_ReceiveInterruptSource == 0);
           printf("\n interrupt source = input %u",b_ReceiveInterruptSource);
           i_PA3500_DisableDigitalInputInterrupt(b_BoardHandle);
           }
        else
           printf("\n i_PA3500_EnableDigitalInputInterrupt error");
        i_PA3500_ResetBoardIntRoutine(b_BoardHandle);
        }
      else
        printf("\n Interrupt routine initialisation error");
      i_PA3500_CloseBoardHandle(b_BoardHandle);
      }
   else
      printf("\n Initialisation error");
   }
```

c) Example in C for Windows 3.1x

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

   if (Initialisation(&b_BoardHandle) == 0)
      {
      if (i_PA3500_SetBoardIntRoutineWin16(b_BoardHandle, v_InterruptRoutine) == 0)
        {
        b_ReceiveInterruptSource = 0;
        if (i_PA3500_EnableDigitalInputInterrupt(b_BoardHandle,3)==0)
           {
           while (b_ReceiveInterruptSource == 0);
           printf("\n interrupt source = input %u",b_ReceiveInterruptSource);
           i_PA3500_DisableDigitalInputInterrupt(b_BoardHandle);
           }
        else
           printf("\n i_PA3500_EnableDigitalInputInterrupt error");
        i_PA3500_ResetBoardIntRoutine(b_BoardHandle);
        }
      else
        printf("\n Interrupt routine initialisation error");
      i_PA3500_CloseBoardHandle(b_BoardHandle);
      }
   else
      printf("\n Initialisation error");
   }
```

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

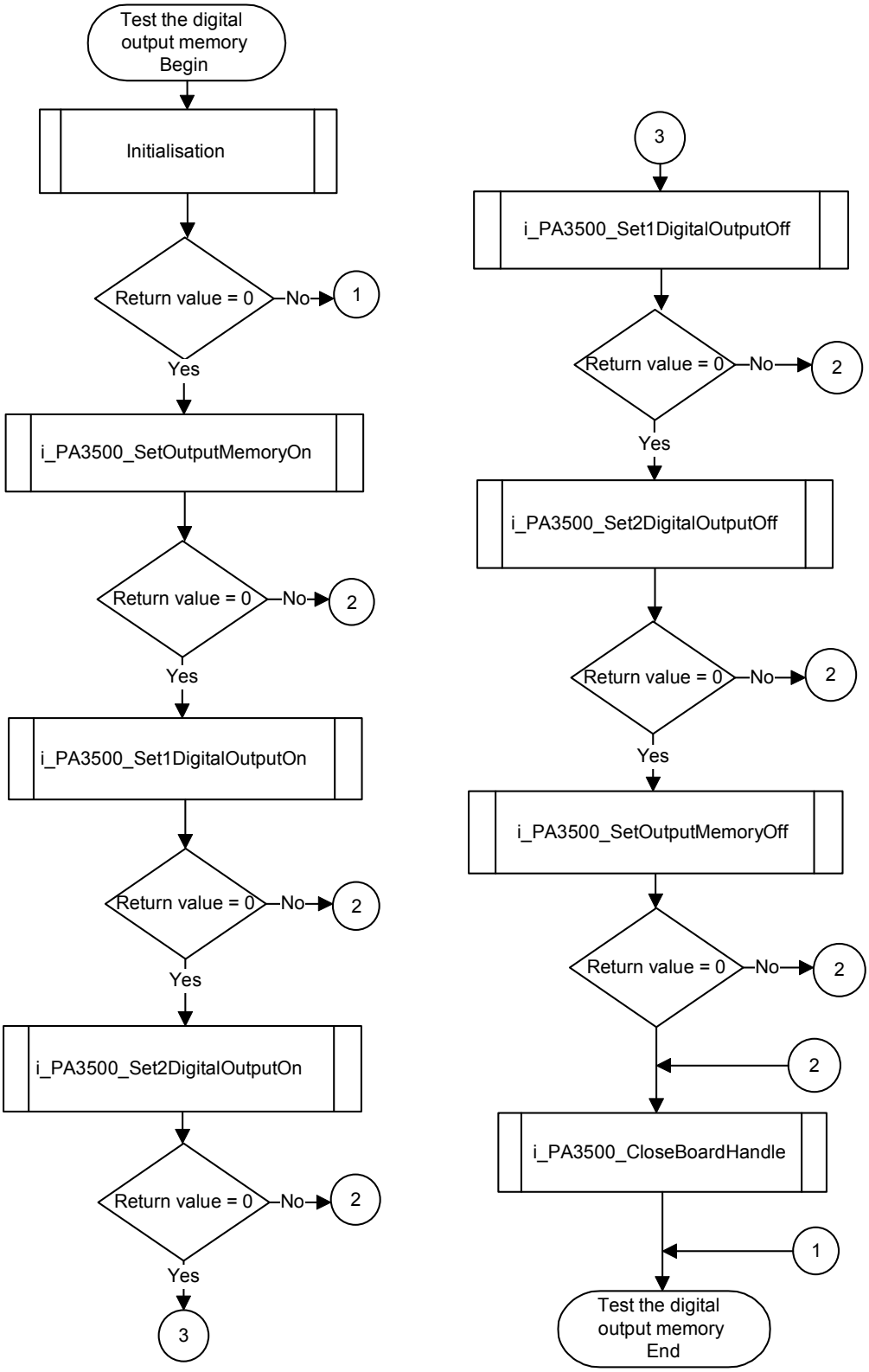
```
void main(void)
   {
   unsigned char b_BoardHandle;

   if (Initialisation(&b_BoardHandle) == 0)
      {
      if (i_PA3500_SetBoardIntRoutineWin32(b_BoardHandle,PA3500_ASYNCHRONOUS_MODE,0,NULL,
                                           v_InterruptRoutine) == 0)
        {
        b_ReceiveInterruptSource = 0;
        if (i_PA3500_EnableDigitalInputInterrupt(b_BoardHandle,3)==0)
           {
           while (b_ReceiveInterruptSource == 0);
           printf("\n interrupt source = input %u",b_ReceiveInterruptSource);
           i_PA3500_DisableDigitalInputInterrupt(b_BoardHandle);
           }
        else
           printf("\n i_PA3500_EnableDigitalInputInterrupt error");
        i_PA3500_ResetBoardIntRoutine(b_BoardHandle);
        }
      else
        printf("\n Interrupt routine initialisation error");
      i_PA3500_CloseBoardHandle(b_BoardHandle);
      }
   else
      printf("\n 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_PA3500_SetBoardIntRoutineWin32(b_BoardHandle,PA3500_ASYNCHRONOUS_MODE,
                                           sizeof(str_UserStruct),
                                           (void **)&ps_GlobalUserStruct,
                                           v_InterruptRoutine) == 0)
        {
        ps_GlobalUserStruct -> b_ReceiveInterruptSource = 0;
        if (i_PA3500_EnableDigitalInputInterrupt(b_BoardHandle,3)==0)
           {
           while (ps_GlobalUserStruct -> b_ReceiveInterruptSource == 0);
           printf("\n interrupt source = input %u", ps_GlobalUserStruct ->
b_ReceiveInterruptSource);
           i_PA3500_DisableDigitalInputInterrupt(b_BoardHandle);
           }
         else
            printf("\n i_PA3500_EnableDigitalInputInterrupt error");
        i_PA3500_ResetBoardIntRoutine(b_BoardHandle);
        }
      else
        printf("\n Interrupt routine initialisation error");
      i_PA3500_CloseBoardHandle(b_BoardHandle);
      }
   else
      printf("\n Initialisation error");
   }
```

10.5 Digital outputs

a) Flow chart

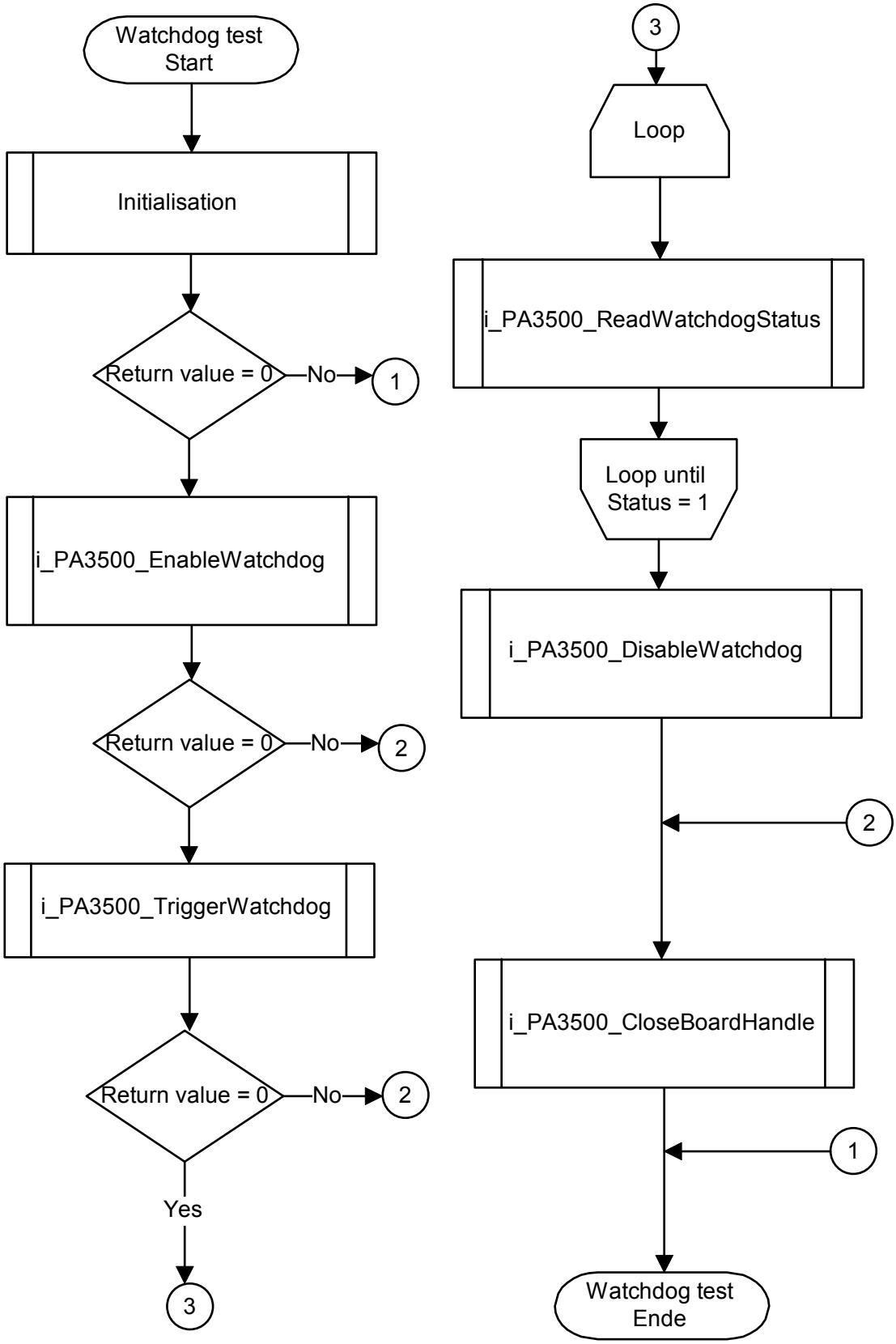


b) Example in C

```
void main (void)
     {
     unsigned char b_BoardHandle;
     if (Initialisation (&b_BoardHandle) == 0)
        {
        if (i_PA3500_SetOutputMemoryOn (b_BoardHandle)  == 0)
           {
           printf (“Digital output memory is activated“);
           if (i_PA3500_Set1DigitalOutputOn (b_BoardHandle, 1) == 0)
              {
              printf(“ Output 1 is set “);
              getch();
              if (i_PA3500_Set2DigitalOutputOn (b_BoardHandle, 2) ==0)
                 {
                 printf (“All Output are set “);
                 getch();
                 if (i_PA3500_Set1DigitalOutputOff (b_BoardHandle, 1 ) == 0)
                    {
                     printf (“Output 1 is reset“);
                     getch();
                     if (i_PA3500_Set2DigitalOutputOff (b_BoardHandle, 2) == 0)
                        {
                        printf (“Output 2 is reset“);
                        if (i_PA3500_SetOutputMemoryOff (b_BoardHandle) == 0)
                           printf (“Digital Output Memory deactivated“);
                        else   printf (“i_PA3500_SetOutputMemoryOff error“);
                        }
                     else   printf (“i_PA3500_Set2digitalOutputOff error“);
                    }
                 else   printf (“i_PA3500_Set1DigitalOutputOff error “);
                 }
              else   printf (“i_PA3500_Set2digitalOutputOn error“);
              }
           else   printf (“i_PA3500_Set1DigitalOutputOn error“);
           }
        else   printf (“i_PA3500_SetOutputMemoryOn error“);
        i_PA3500_CloseBoardHandle (b_BoardHandle);
        }
     else   printf (“Initialisation error“);
     }
```

10.6 Testing the watchdog

a) Flow chart



b) Example in C

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

     if (Initialisation (&b_BoardHandle) == 0)
        {
        if (i_PA3500_EnableWatchdog(b_BoardHandle,PA3500_DISABLE)  == 0)
           {
           if (i_PA3500_TriggerWatchdog(b_BoardHandle) == 0)
              {
              printf(" Watchdog is triggered ");
              do
                 {
                 i_ReturnValue =
i_PA3500_GetWatchdogStatus(b_Boardhandle,&b_WatchdogStatus);
                 if(i_ReturnValue == 0)
                    {
                    printf ("\nWatchdog Status = %u ",b_WatchdogStatus);
                    }
                 else   printf ("i_PA3500_GetWatchdogStatus error");
                 }
              while((i_ReturnValue == 0)&&(b_WatchdogStatus == 0));
              }
           else   printf ("i_PA3500_TriggerWatchdog error");
           i_PA3500_DisableWatchdog(b_BoardHandle);
           }
        else   printf ("i_PA3500_EnableWatchdog error");
        i_PA3500_CloseBoardHandle (b_BoardHandle);
        }
     else   printf ("Initialisation error");
     }
```

INDEX