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 3110

Analog input and output channels

9th edition 04/1999

Guarantee and responsibility

Basically are effective our "general terms of delivery and payment". The manager receives them at the latest with the invoice. Claims for guarantee and responsibility in case of injuries and material damages are excluded, if they are due to one or some of the following causes:
- if the board has not been used for the intended purpose
- improper installation, operation and maintenance of the board
- if the board has been operated with defective safety devices or with not appropriate or nonfunctioning safety equipment
- nonobservance of the instructions concerning: transport, storage, inserting the board, use, limit values, maintenance, device drivers
- altering the board at the user's own initiative
- altering the source files at the user's own initiative
- not checking properly the parts which are subject to wear
- disasters caused by the intrusion of foreign bodies and by influence beyond the user's control.

Licence for ADDI-DATA software products

Read carefully this licence before using the standard software. The right for using this software is given to the customer, if he/she agrees to the conditions of this licence.
- this software can only be used for configuring ADDI-DATA boards.
- copying the software is forbidden (except for archiving/ saving data and for replacing defective data carriers).
- deassembling, decompiling, decoding and reverse engineering of the software are forbidden.
- this licence and the software can be transfered 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

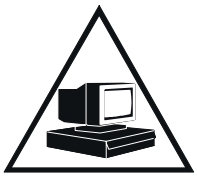
Borland C and Turbo Pascal are registered trademarks of Borland International, INC.
Burr-Brown is a registered trademark of Burr-Brown Corporation.
Intel is a registered trademark of Intel Corporation.
AT, IBM, ISA and XT are registered trademarks of International Business Machines Corporation.
Microsoft, 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 improper handling and if the board is not used for the purpose it is intended for:

people may be injured



the board, PC and peripheral devices may be damaged

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.

- **Do observe the instructions in the manual !**
  Make sure that you have not skipped any step. We are not liable for damage resulting from the improper use of the board.

- **Symbols used**

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

IMPORTANT!
designates hints and other useful information.

- **Do you have any questions?**
  Our technical support is always glad to help you.

CE

Declaration of Conformity

This declaration is valid for the following product:

ADDIALOG PA 3110
Analog inputs and outputs, 12/14 bits
optically isolated

It is made by

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

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

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

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

This declaration is valid for all units manufactured according to the manufacturing references listed in the form TD3110.020.

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.

09 October 1996

Legally valid signature of the manufacturer

Figures

Tables

1 INTENDED PURPOSE OF THE BOARD

The **PA 3110** board is the interface between an industrial process and a personal computer (PC).

It is to be used in a free 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 CE mark.

Data exchange between the **PA 3110** 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 3110** board.

The board has up to 16 / 8 input and output channels for processing analog signals.

The **PX 901** screw terminal board allows to connect the analog signals through a shielded cable.

The use of the **PA 3110** board in combination with external screw terminal boards is to occur in a closed switch cabinet; the installation is to be effected competently.

The connection with our standard cable ST010 complies with the specifications:
- metallized plastic hoods
- shielded cable
- cable shield folded back and firmly screwed to the connector housing.

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

The use of the board according to its intended purpose includes observing all advices given in the *Technical description* and in the *Safety* leaflet.

1.1 Limits of use

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

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

The use of the board in a PC could change the PC features regarding to noise emission and immunity.
Increased noise emission or decreased noise immunity could result in the system not being conform anymore.

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

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

2 USER

2.1 Qualification

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

- installation,
- use,
- maintenance.

2.2 Personal protection

Consider the country-specific regulations about

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

3 HANDLING THE BOARD

Fig. 3-1: Wrong handling

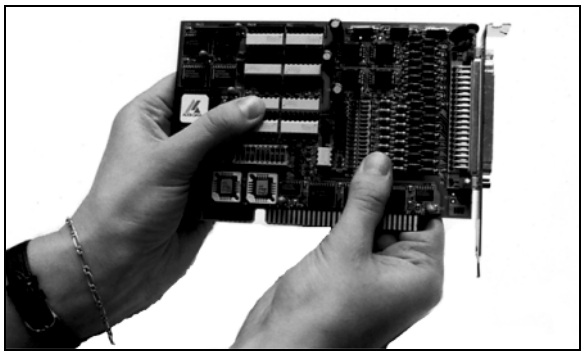
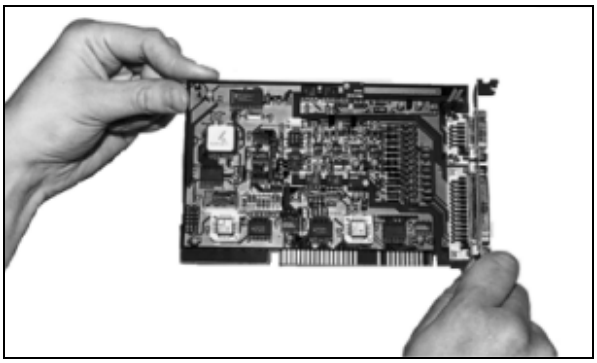


Fig. 3-2: Correct handling



4 TECHNICAL DATA

4.1 Electromagnetic compatibility (EMC)

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

	True 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
Noise emissions	B-class	

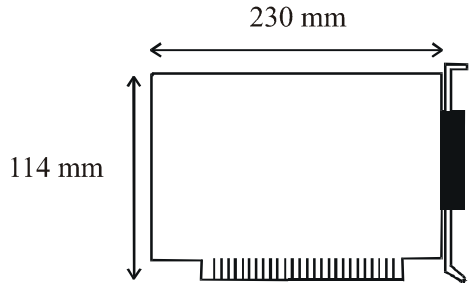
WARNING!
The EMC tests have been carried out in a specific appliance configuration. We guarantee these limit values **only** in this configuration. [1]

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:	AT slot
Connection to the peripheral	37-pin SUB-D male connector

Connection to the peripheral through:
- cable with twisted pairs directly to the analog signal transmitters
- our standard cable ST010 to screw terminal board PX 901-AG or PX 901-A

1 We transmit our appliance configuration on request.

4.3 Options

SF, DF: Precision filter for the analog input channels

PC: Precision current inputs 0-20 mA or 4-20 mA
Remark: with a current range of 4-20 mA, the accuracy is altered

4.4 Versions

The board **PA 3110** is available in the following versions:

Version	Analog input channels	Analog output channels
PA3110-8-4	8 SE / 4 Diff.[1]	4
PA3110-8-8	8 SE / 4 Diff.	8
PA3110-16-4	16 SE / 8 Diff.	4
PA3110-16-8	16 SE / 8 Diff.	8

4.5 Limit values

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

Minimum PC requirements:
- Operating system: .....MS DOS 3.3 or > Windows 3.1
- Bus speed: .....8 MHz

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

	PA3110-8-4	PA3110-8-8	PA3110-16-4	PA3110-16-8
+ 5 V from the PC	945	1082	1038	1182

1 SE: single ended
Diff.: differential

Analog input channels:

Number of analog input channels:	16/8 SE/diff. for **PA 3110-16x** 8/4 SE/diff. for **PA3110-8x**
Analog resolution:	14-bit, 1 among 16383
Max. sampling rate (for 1 channel):	100 kHz
Data transfer:	Data to the PC (16-bit only) via FIFO memory 1) Through I/O commands 2) Interrupt at EOC[1] & EOS[2] 3) DMA transfer at EOC
Start of conversion:	1) Trough software trigger 2) TIMER 0 3) TIMER 0 & 1
Monotony:	14-bit
Offset error:	After calibration: - Bipolar: ± 1/2 LSB - Unipolar: ± 1/2 LSB Drift (0°C to 60°C): - Bipolar: ± 2 ppm / °C - Unipolar: ± 2 ppm / °C
Gain error:	After calibration: - Bipolar: ± 1/2 LSB - Unipolar: ± 1/2 LSB Drift (0°C to 60°C): - Bipolar: ±7 ppm / °C - Unipolar: ±7 ppm / °C
Analog input ranges:	Voltage - Unipolar: 0-10 V - Bipolar: ±10 V - Selectable through software Current Unipolar: 0-20 mA Selection of the range 0-10 V and of gain x2 is necessary
Overvoltage protection:	20 V at POWER ON
Common mode rejection:	DC up to 10 Hz, 90 dB min. (Gain = 1)
Band width (-3dB):	Limited to 159 kHz (-3dB) with low-pass filter 1st order; yet the minimum SINAD is still 83 dB at 49 kHz (fin)
Bias currents for each input (multiplexer):	± 2 nA max.
Input impedance (PGA):	$10^{12}\,\Omega$ // 20 nF to GND
Integral non-linearity (INL):	± 1/2 LSB
Differential non-linearity (DNL):	± 1/2 LSB
Precision:	± 1 LSB

1 EOC: End of Conversion

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

Selectable gain:	through PGA 1, 2, 5, 10 (selectable by software)
System noise:	Bipolar: Gain x1:± 1/2 LSB Gain x2:± 1/2 LSB Gain x10:± 2 LSB Unipolar: Gain x1:± 1/2 LSB Gain x2:± 1/2 LSB Gain x10:± 2 LSB
Digital coding:	two's complement
Optical isolation to the PC:	500 V

Analog output channels:

Resolution:	12-bit
Overvoltage protection:	± 12 V
Number of output channels:	4 or 8
Data transfer:	The board is located in the I/O address space of the PC. The values are written on the board through 16-bit accesses. The outputs are updated with a dummy reading on an address (simultaneous updating of the output channels).
Settling time at 0,01% FS, (FS = Full scale) with 2 kΩ and 100 pF load:	- 2.5µs typ. for a 10 V voltage jump at 25°C - 10µs typ. for a 10 V voltage jump above the temperature range - 3.5µs typ. for a 20 V voltage jump at 25°C - 10 µs typ. for a 20 V voltage jump above the temperature range
Output voltage ranges:	Unipolar: 0-10 V Bipolar: ± 10 V
Digital coding:	Unipolar: Straight binary coding Bipolar: Offset binary coding
Output current:	± 5 mA max.
Capacitive load:	500 pF max.
Short-circuit current:	± 25 mA
Integral non-linearity (INL):	± 1 LSB max. above the temperature range
Differential non-linearity (DNL):	± 1/2 LSB max. above the temperature range

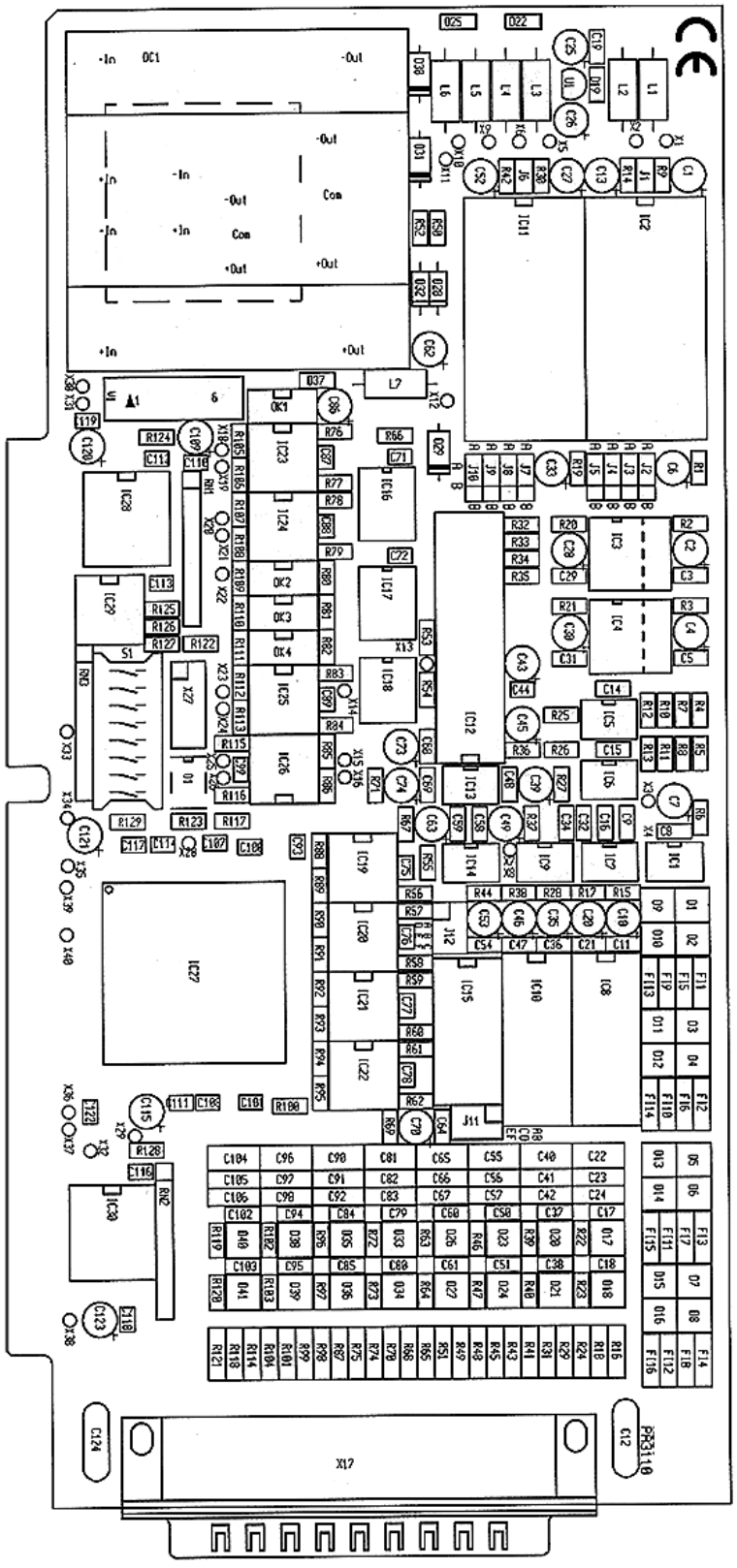
Analog output channels (continued):

Monotony:	12-bit
Offset error:	± 3 mV max. Unipolar ± 20 mV max. Bipolar
Gain error:	± 0.2 % of FSR max.
Galvanic separation to the PC:	500 VDC min.
Voltage after Reset:	0 V or -10V (depending on the mode, jumper settings, and on the voltage range)
Watchdog:	can be configured by software through timer2 times of 140 µs up to up to 1174 s are possible in steps of 70 µs

5 SETTINGS

5.1 Component scheme

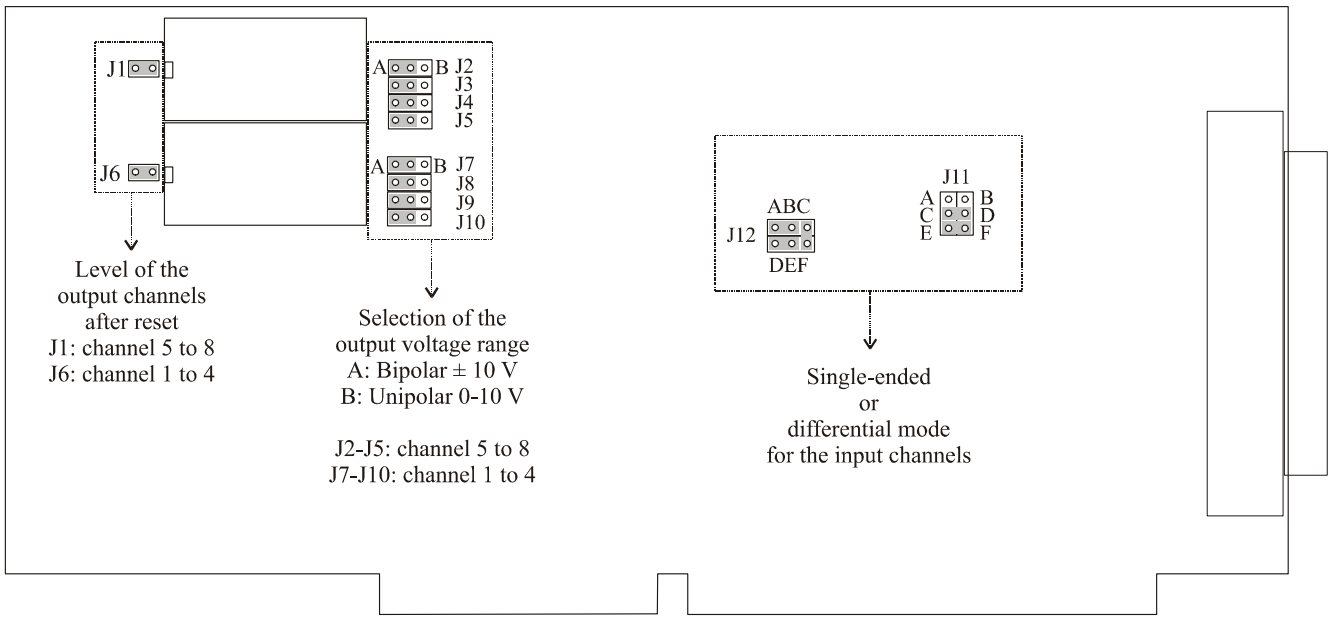
Fig. 5-1: Component scheme



5.2 Jumper settings

5.2.1 Jumper location and settings at delivery

Fig. 5-2: Jumper location and settings at delivery



5.2.2 Jumper settings

Table 5-1: Jumper settings

Jumper	Settings	Functions	Settings at delivery
J1	Set	Level after reset - Output channels 5 to 8 0 V (in unipolar mode) -10 V (in bipolar mode)	Set
	Not set	Level after reset - Output channels 5 to 8 5 V (in unipolar mode) 0 V (in bipolar mode)	
J6	Set	Level after reset - Output channels 1 to 4 0 V (in unipolar mode) -10 V (in bipolar mode)	Set
	Not set	Level after reset - Output channels 1 to 4 5 V (in unipolar mode) 0 V (in bipolar mode)	
J2, 3, 4, 5	A	Output voltage range - Channels 5 to 8 Bipolar = ± 10 V	A
	B	Output voltage range - Channels 5 to 8 Unipolar = 0-10 V	
J7, 8, 9, 10	A	Output voltage range - Channels 1 to 4 Bipolar = ± 10 V	A
	B	Output voltage range - Channels 1 to 4 Unipolar = 0-10 V	
J12*	A-B D-E C-F	Mode selection for the input channels Single-ended	A-B D-E C-F
	A-B E-F	Mode selection for the input channels Differential	
J11*	C-D E-F	Mode selection for the input channels Single-ended	C-D E-F
	A-C D-F	Mode selection for the input channels Differential	

* In the Single-Ended or differential mode, both jumper fields are to be modified.

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 (only once)
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 Base address

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

Before installing the board

The base address is set at delivery on the address 0300H.

- **Check**, that

  - the base address is free

  - the address range (32 I/O addresses) required by the board is not already used by the PC or by boards already installed in the PC.

The base address and/or the address range **are wrong**?
- **Select** another base address with the 8-pole block of DIP switches S1

The address 0300H is decoded in the following table.

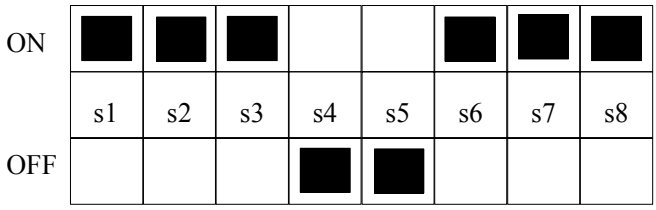
Table 6-1: Decoding table

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

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

Fig. 6-1: DIP switches

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



6.2 Inserting the board

IMPORTANT!
Please observe the *safety instructions*.

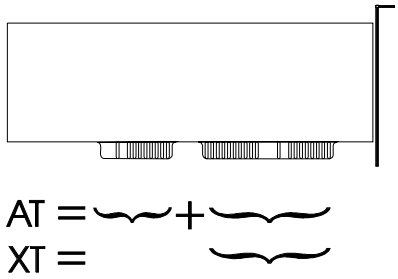
6.2.1 Opening the PC

- 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 AT slot

Fig. 6-2: Slot types



The board can be used in EISA slots under 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

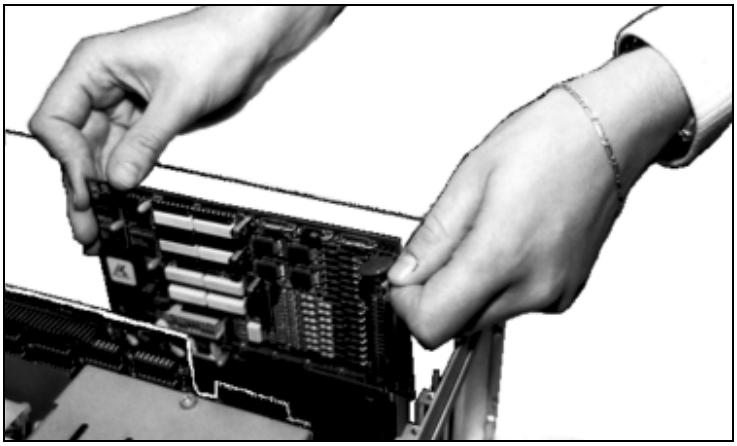
4. Take the board from its protective blister pack.

Fig. 6-3: Opening the blister pack

6.2.3 Inserting the board

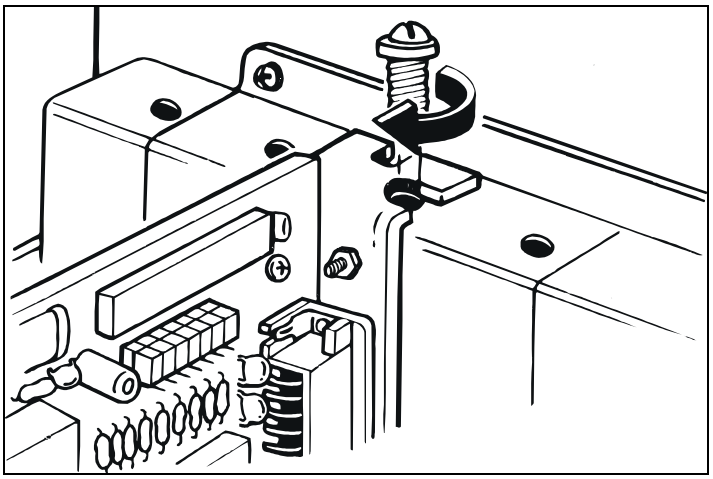
- **Discharge yourself from electrostatic charges**
- Insert the board **vertically into the chosen slot.**

Fig. 6-4: Inserting the board



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

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



- **Tighten all loosen screws**

6.2.4 Closing the PC

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

6.3 Installing the software

The board is delivered with a CD-ROM containing ADDIREG for Windows NT 4.0 and Windows 95.

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

http://www.addi-data.de
http://www.addi-data.com

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

6.3.1 Software installation under MS-DOS and Windows 3.11

- Copy the contents of PA3110\16bit on a disk.
  If several disks are to be used, the directory contents is stored in several sub-directories (Disk1, Disk2, Disk3...).
- Insert the (first) disk into a drive and change to this drive.
- Enter <INSTALL>.

The installation program gives you further instructions.

6.3.2 Software installation under Windows NT / 95

- Select the directory PA3110\32bit\Disk1.
- Start the setup program "setup.exe" (double click)
- Select one of the 3 parameters
  1- typical
  2- compact
  3- custom

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

The installation program gives you further instructions.

6.4 Board configuration with ADDIREG

The ADDIREG registration program is a 32-bit program for Windows NT 4.0/ 95. 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.

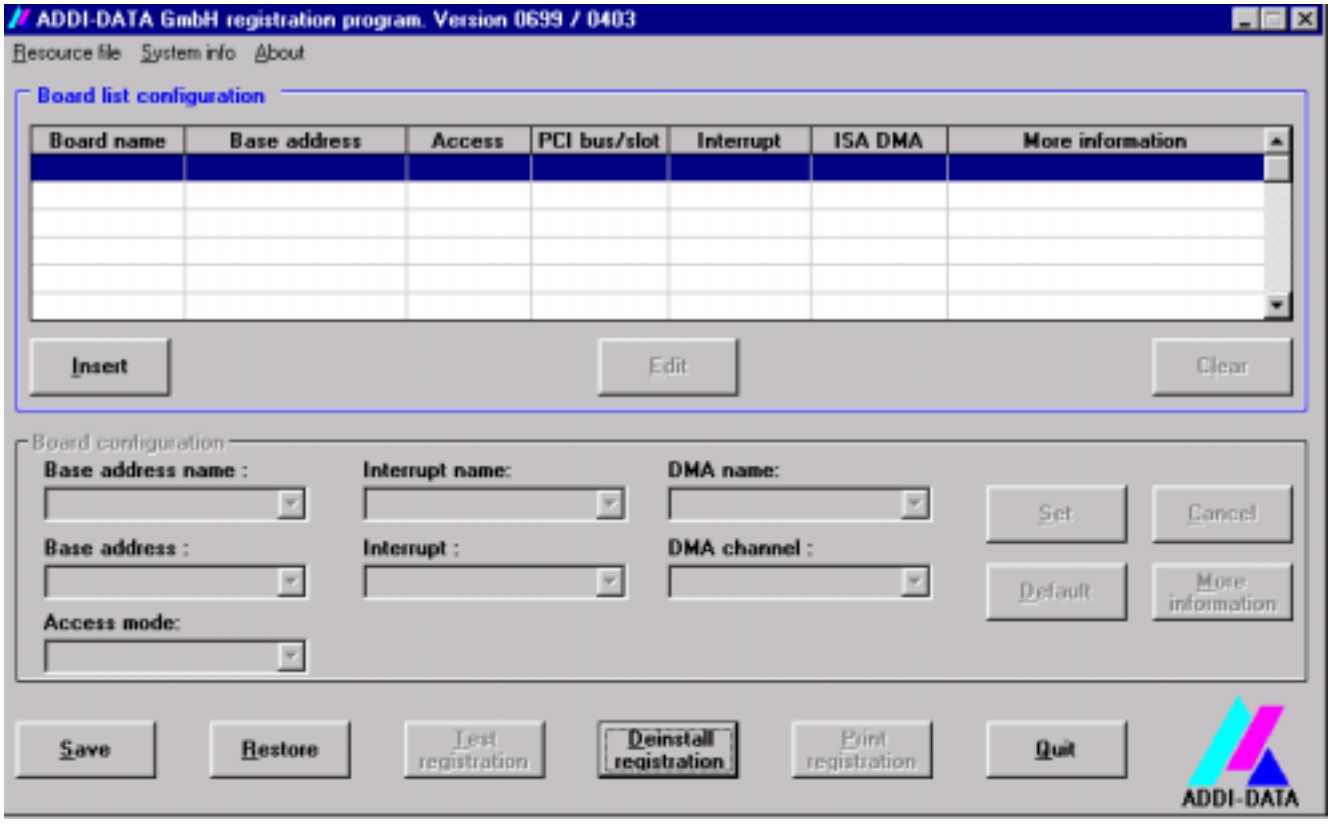
6.4.1 Program description

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

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

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

Fig. 6-6: The ADDIREG registration program



Board list configuration table:

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

Board name:
Names of the different registered boards

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.

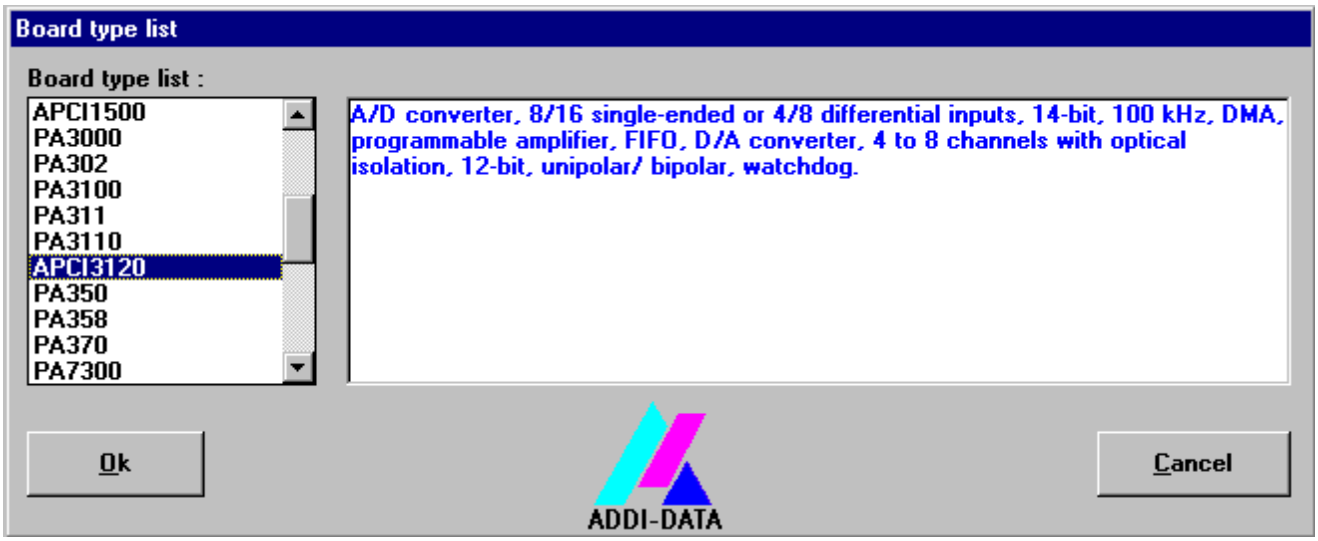
DMA channel:
Selection of the used DMA channel.

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:

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.

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

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 registration of all boards 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.

6.4.4 Removing the ADDIREG program

The ADDI_UNINSTAL program is delivered on the CD-ROM.

- Install the ADDI_UNINSTAL program on your computer.
- Start the ADDIREG program and click on "Deinstall registration"
- Quit ADDIREG
- Start the ADDI_UNINSTAL program
- Proceed as indicated until the complete removing of ADDIREG.

You can also download the programm from Internet.

6.4.5 Software downloads from the Internet

You can download the latest version of the device driver for the PA 3110 board.

http://www.addi-data.de. or
http://www.addi-data.com

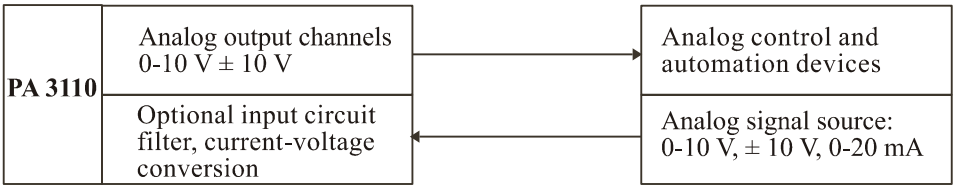
If you have any questions, do not hesitate to send us an e-mail to

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

7 CONNECTING THE PERIPHERAL

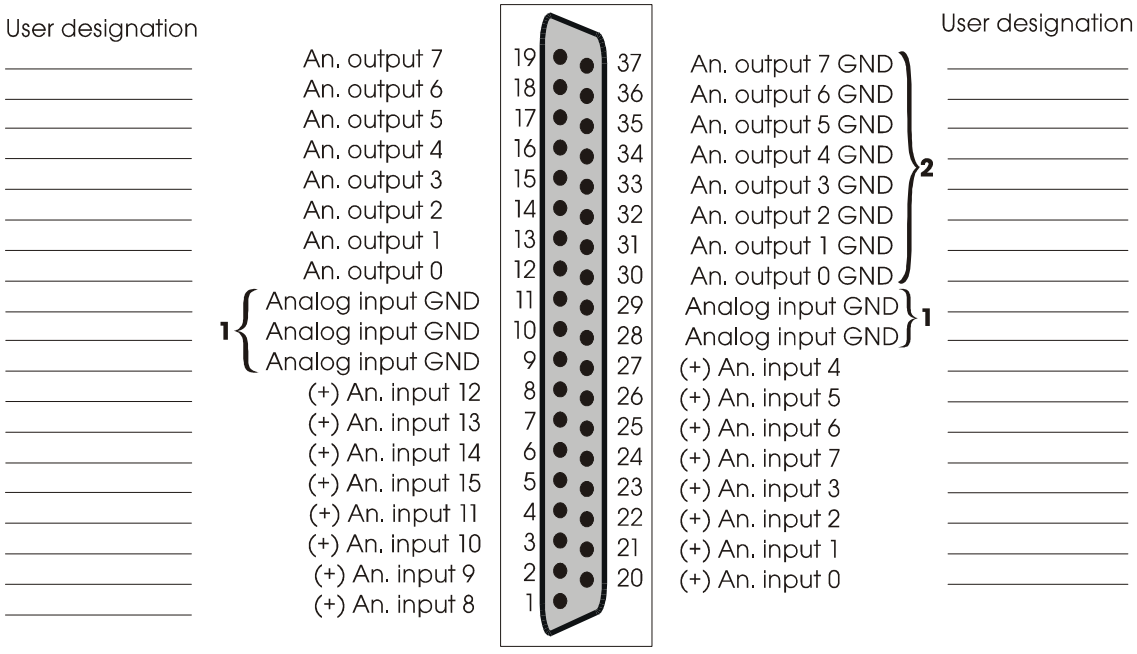
7.1 Connection principle

Fig. 7-1: Connection principle



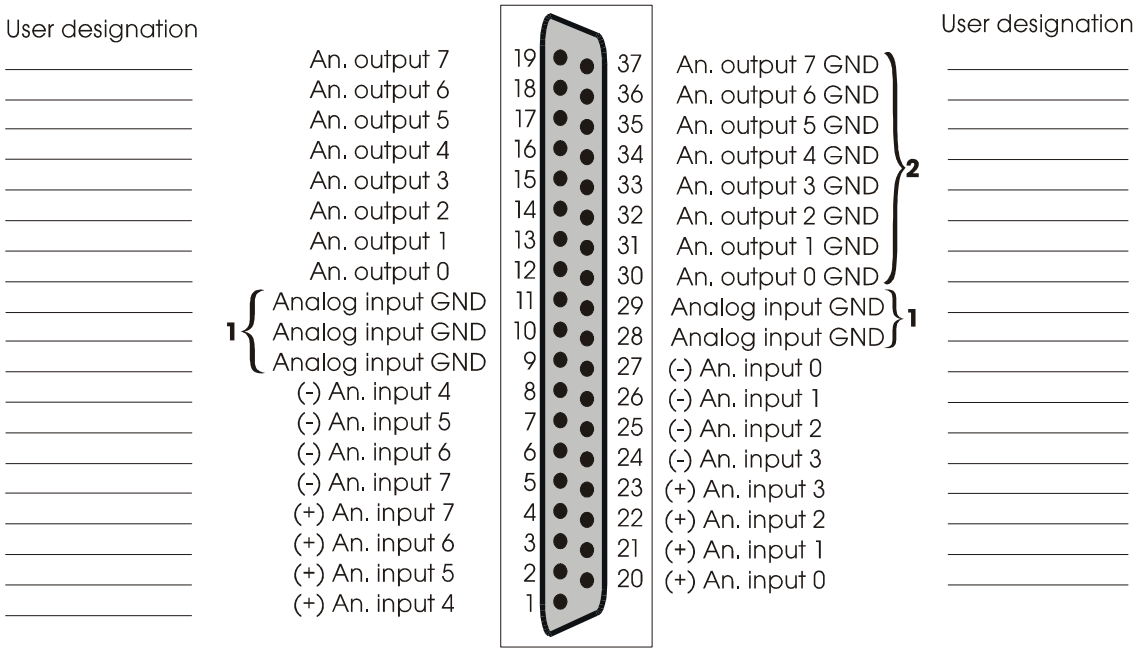
7.2 Connector pin assignment

Fig. 7-2: 37-pin SUB-D male connector X20 - Single-ended mode



1: The analog input channels have a common ground line
2: The analog output channels have separate ground lines

Fig. 7-3: 37-pin SUB-D male connector X20 - differential mode



1: The analog inputs have a common ground line
2: The analog outputs have separate ground lines

7.3 Connection examples

Fig. 7-4: Screw terminal board PX 901-A and cable ST010

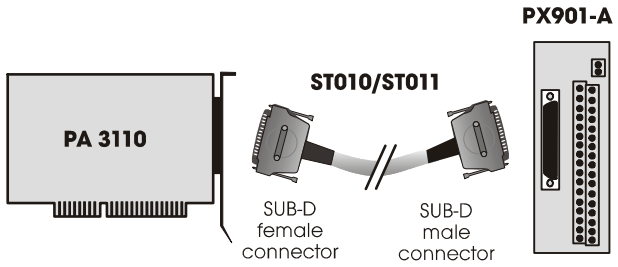


Fig. 7-5: Connection in single-ended mode

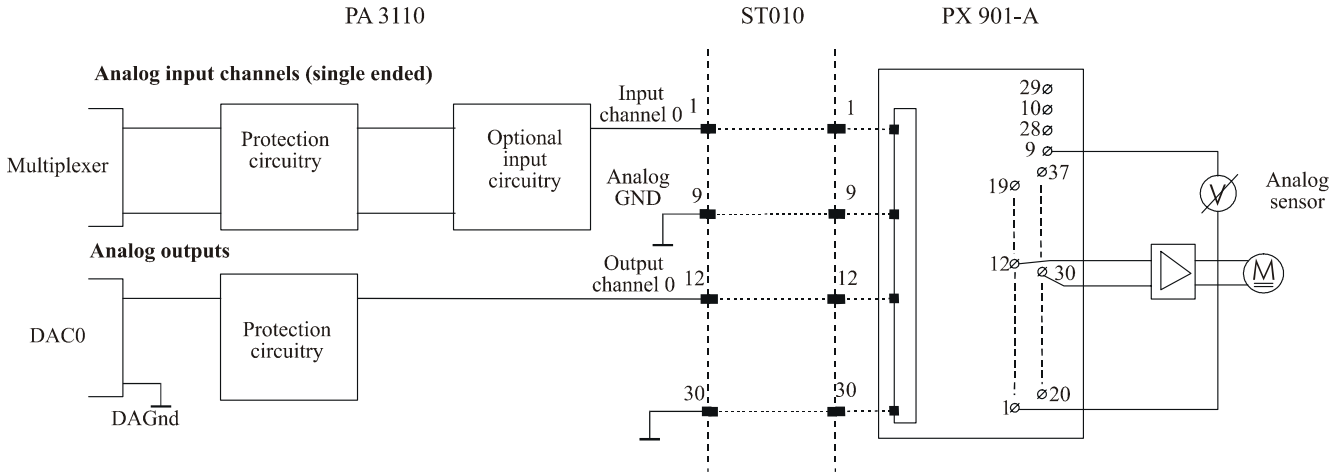
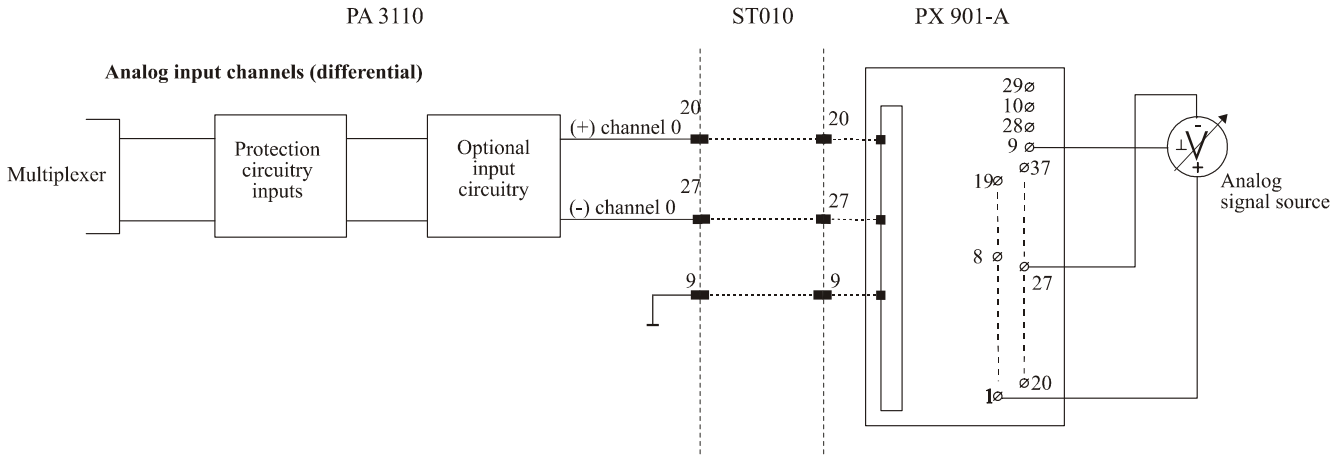


Fig. 7-6: Connection in differential mode



8 FUNCTIONS

8.1 Analog output channels

The board has 4 or 8 analog output channels.
They are set to a defined level after Power-ON Reset.

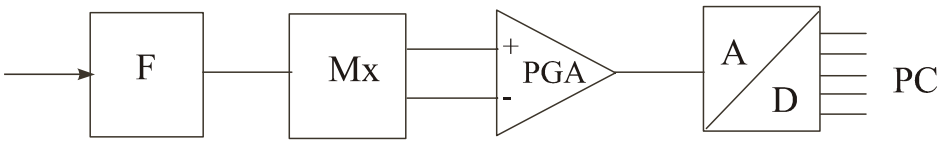
Update of the channels
1- 16-bit write commands on I/O addresses
2- dummy read

A bit of the *Status Register* indicates whether all channels are ready for updating.

Watchdog
A watchdog function is available through Timer2, which must be programmed as a retriggerable watchdog.
The watchdog is triggered through a write command on the output channels.
As long as data is written on the output channels, they are not reset.
The watchdog can also generate interrupts.
The state of the watchdog is indicated in the *Status Register*.

8.2 Analog input channels

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



After having reached the multiplexer (time-multiplexed system) through a filter (RC shield), the signals are led through a programmable gain amplifier to the 16-bit A/D converter.

Configuration through software
- analog input ranges 0-10 V (unipolar) and ± 10 V (bipolar)
- gain

It is possible to use a different voltage (or current) for each channel, and to use the highest resolution of the A/D converter.

Scan list
This list allows to acquire data in a more flexible way.
It can be stored in a 16-byte RAM.
The depth is determined through software.

The scan list allows to acquire data in different ways:
- 1 channel after the other through software trigger
  (each channel must be triggered),
- All channels at once through software trigger,
- through Timer0: the list is handled cyclically,
- through Timer0 & 1: the list is handled during a defined time interval.
- through Timer0, 1 & 2: a defined number of conversions or scans can be handled.

Data exchange with the PC occur through a 256-word FIFO.
- Polling is possible, analysis of the EOC and EOS bit
- Interrupt at EOC, EOS, END OF DMA,
- DMA mode at EOC
- Data is partly buffered (according to the type of conversion) in the FIFO.

8.3 Time-multiplex system

Data acquisition with the **PA 3110** is based on a time-multiplex system.

The board is equipped with a single A/D converter to which the channels are led through an analog multiplexer (hardware and software controlled).
The instrumental amplifier is highly resistive.
There is a capacitive distance from the output channel of the multiplexer to the input channel of the INA, so that each channel is a RC module (low-pass).
When switching from a channel to another, the output capacity of the multiplexer must be reloaded to the new value.

There is a certain delay between the reloading of the channel and the start of the A/D converter. This delay correspond to the settling time to an end value corresponding to the resolution of the acquisition, for ex.: 0.01% for 12-bit,
0.008% for 14-bit.

The delay time depends on the following factors:
- The settling time of the amplifier approx. 3.5 µs
- The max. voltage bounce from one channel to another
- The source impedance of the sensory mechanism
- Option SF = 10 K // 470 nF/-3dB approx. 30 Hz
- Option DF = 20 K // 470 nF/-3dB approx. 30 Hz

This delay time is supported through the 16-bit timer0.
You can set this time in steps of approx. 0.7 µs between 1.4 µs and 45 874 µs.
This time is defined as *ui_ConvertTiming* in the supplied API library
(see device driver).
Through the scan list it is possible to set the next channel during the conversion (conversion time = 10 µs). With low-impedance sensors, this allows to reach the max. data transfer rate of 100 kHz for several channels.

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

Table 8-1: Direct conversion (software controlled)

R of the signal source	ui_ConvertTiming
<100R	1..5
< 500R	10..60
< 1K	500..700
< 10K	1000..2000
< 50K	10000...65535

Table 8-2: Cyclic conversion (hardware controlled)

R of the signal source	ui_ConvertTiming
<100R	15 ..20
< 500R	30..80
< 1K	500.. 700
< 10K	1000..2000
< 50K	10000..65535

Observe the A/D conversion time.

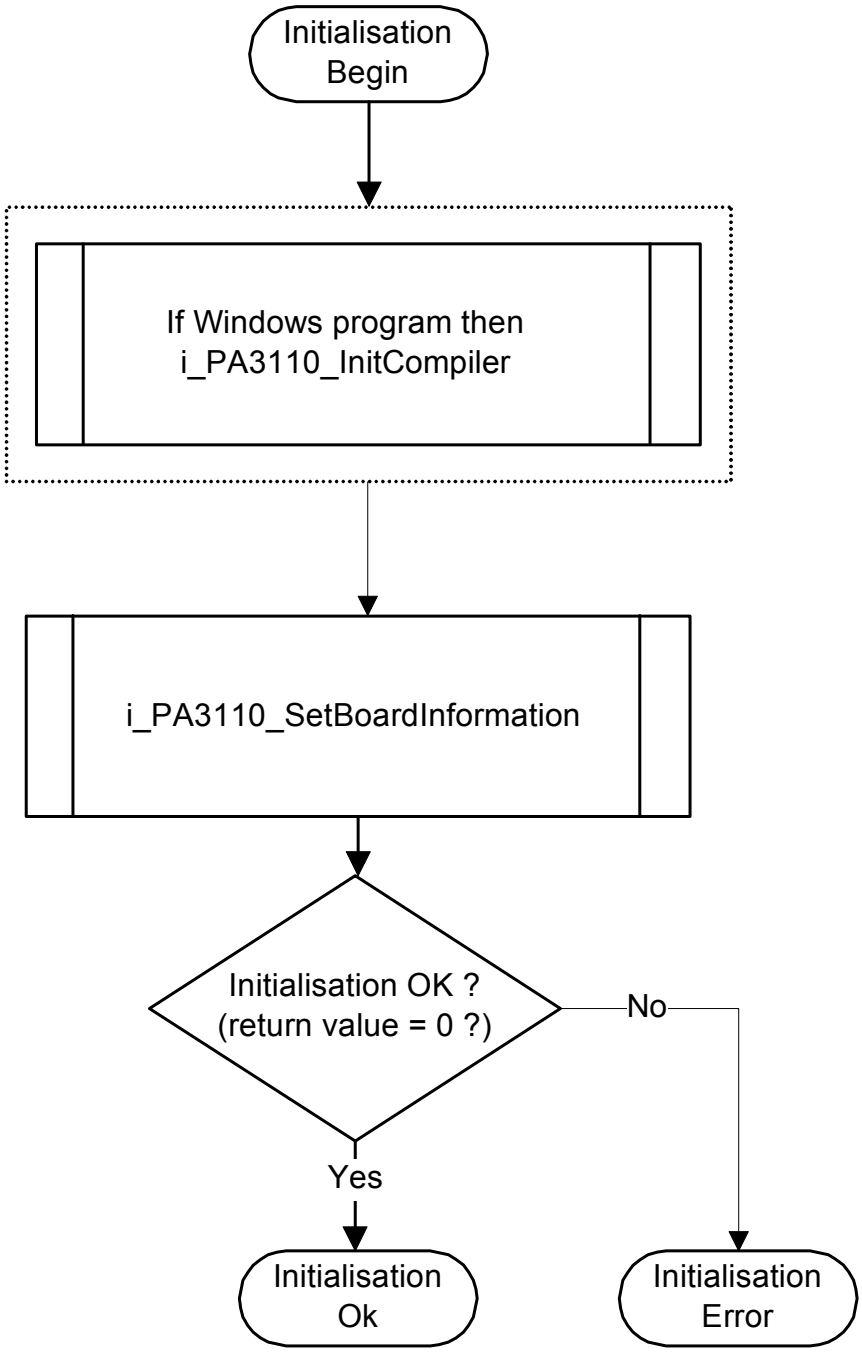
The time *l_DelayTiming* must be > than the number of channels to be converted x *ui_ConvertTiming*.

9 SOFTWARE EXAMPLES

9.1 Initialisation

9.1.1 Initialisation of the PA 3110 under DOS and Windows 3.11

a) Flow chart

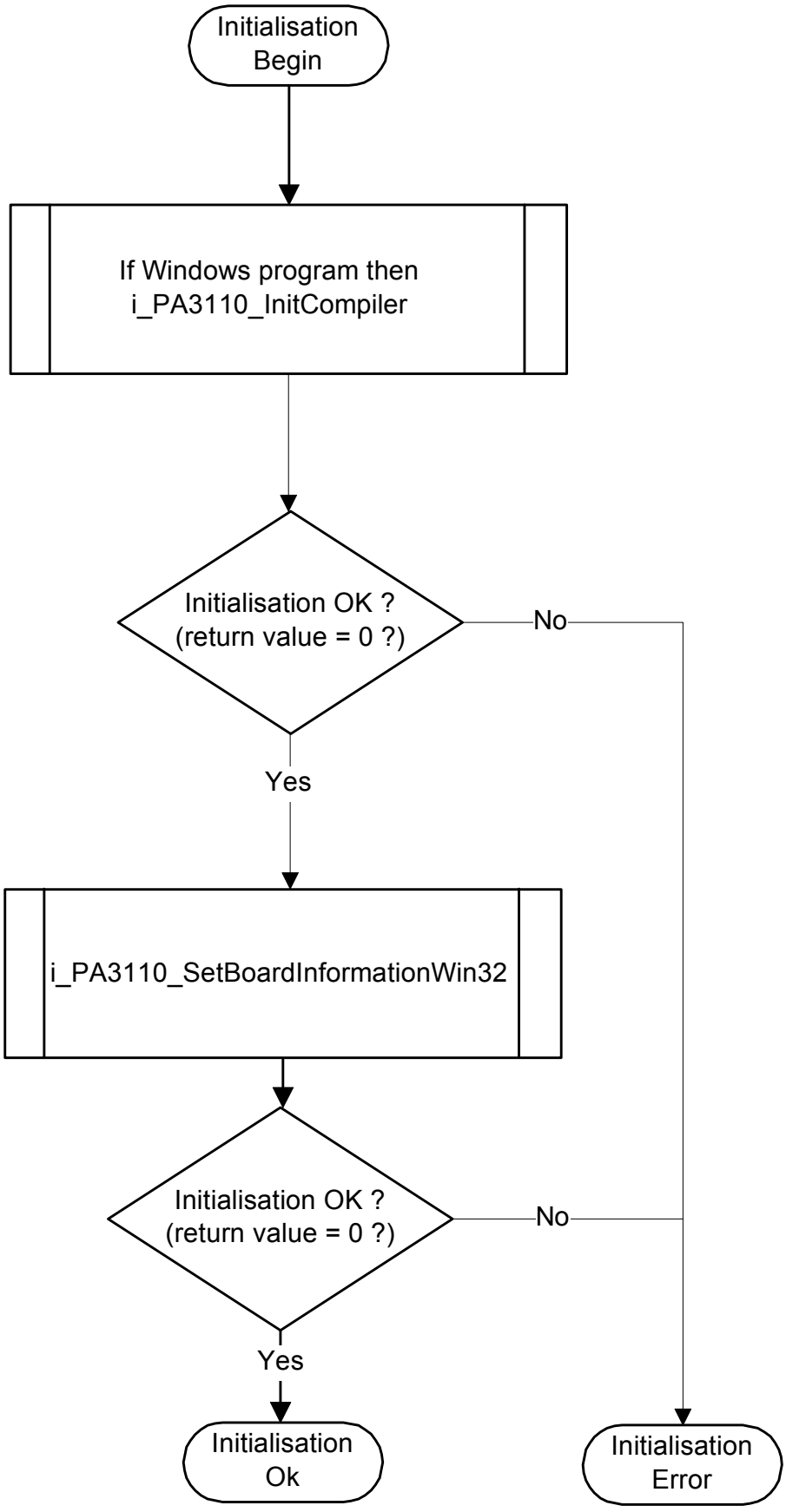


b) Example in C for DOS and Windows 3.11

```
int Initialisation(unsigned char *pb_BoardHandle)
    {
    #ifdef _Windows
       i_PA3110_InitCompiler (DLL_COMPILER_C);
    #endif

    if(i_PA3110_SetBoardInformation (0x390,
                                     3,  // IRQ3
                                     16,
                                     8,
                                     pb_BoardHandle) == 0)
       {
       return (0);         /* OK */
       }
    else
       {
       return (-1);        /* ERROR */
       }
    }
```

9.1.2 Initialisation of the PA 3110 under Windows NT / 95

a) Flow chart



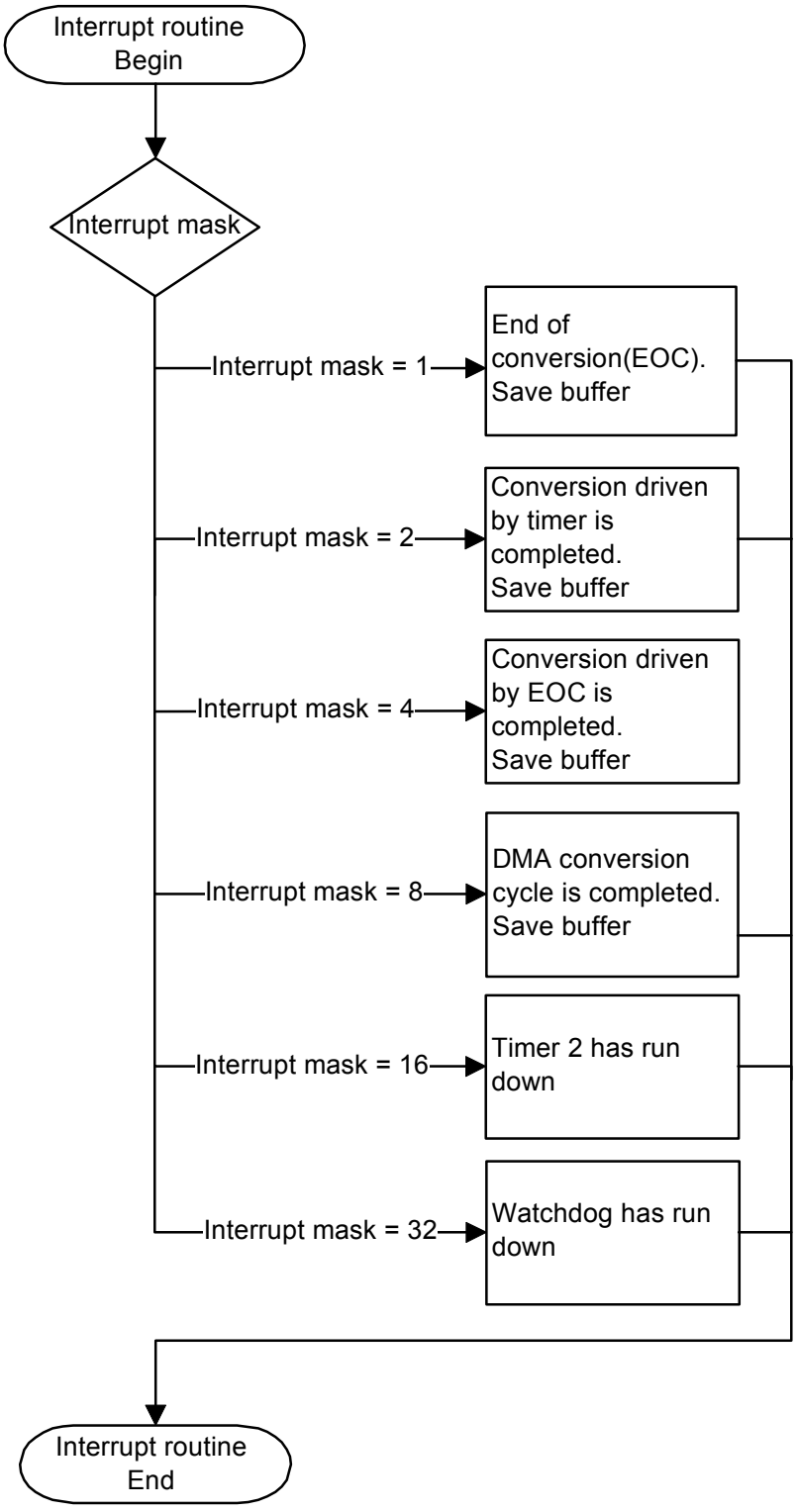
c) Example in C for Windows NT / 95

```
int Initialisation(unsigned char *pb_BoardHandle)
    {
    if (i_PA3110_InitCompiler (DLL_COMPILER_C) == 0)
       {
       if(i_PA3110_SetBoardInformationWin32 (“PA3110-00“,
                                             16,
                                             8,
                                             pb_BoardHandle) == 0)
          {
          return (0);      /* OK */
          }
       else
          {
          return (-1);     /* ERROR */
          }
       }
    else
       {
       return (-1);        /* ERROR */
       }
    }
```

9.2 Interrupt

9.2.1 Interrupt routine under DOS and Windows 3.11

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



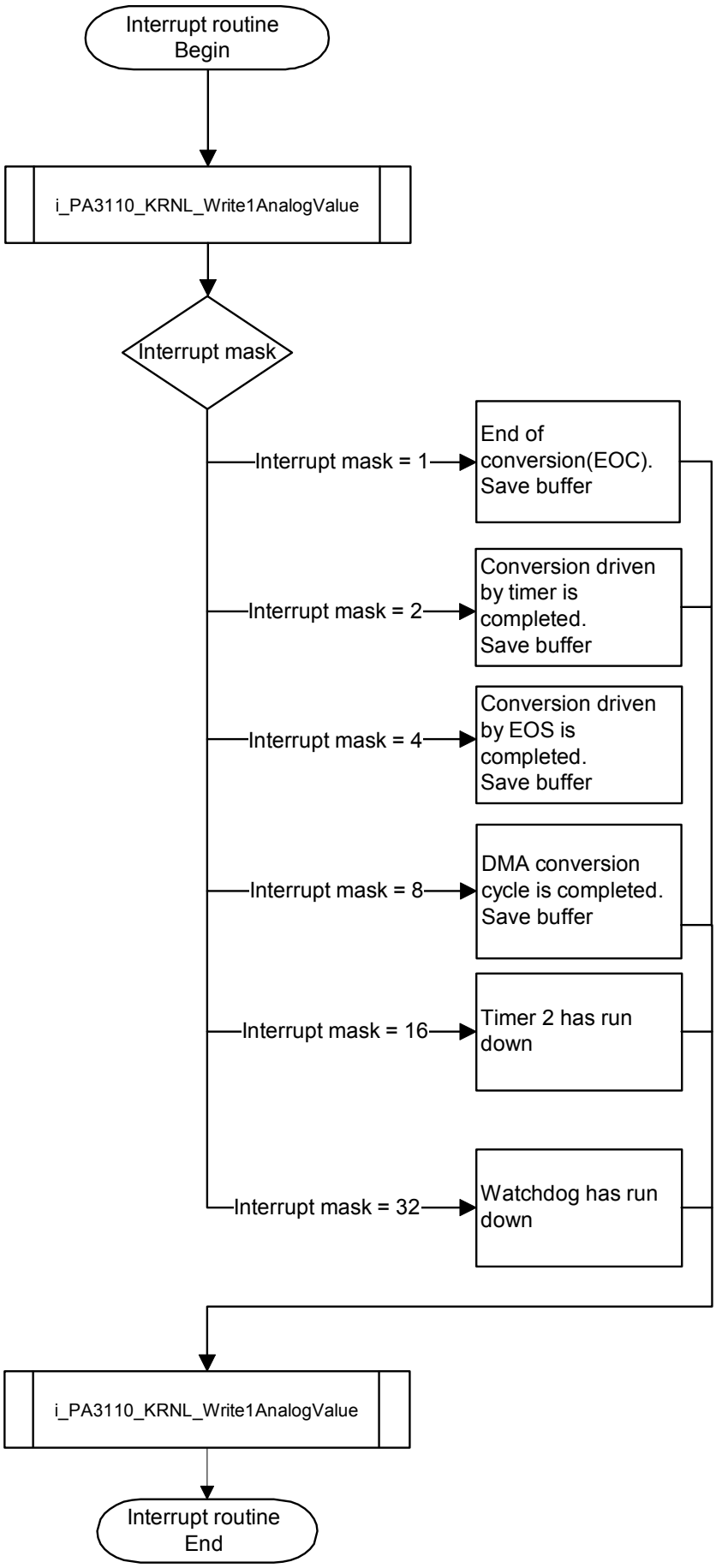
b) Example in C for DOS and Windows 3.11

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

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

     switch(b_InterruptMask)
        {
        case 1:
                /* EOC interrupt */
               ui_SaveArray[0] = pui_ValueArray[1];
               break;
        case 2:
                /* Timer conversion interrupt */
                for (ui_Cpt=0;ui_Cpt< pui_ValueArray [0] ;ui_Cpt++)
                  ui_SaveArray [ui_Cpt] = pui_ValueArray [ui_Cpt +1];
                             break;
        case 4:
                /* EOS interrupt */
               for (ui_Cpt=0;ui_Cpt< pui_ValueArray [0] ;ui_Cpt++)
                  ui_SaveArray [ui_Cpt] = pui_ValueArray [ui_Cpt+1];
                             break;
        case 8:
               /* DMA completed */
               for (ui_Cpt=0;ui_Cpt<16;ui_Cpt++)
                  ui_SaveArray [ui_Cpt] = pui_ValueArray [ui_Cpt];
               break;
        case 16:
               /* Timer 2 has run down */
               ui_TimerIntCpt = ui_TimerIntCpt + 1;
               break;
        case 32:
               /* Watchdog has run down */
               ui_TimerIntCpt = ui_TimerIntCpt + 1;
               break;
                     default :
      break;
               }
          b_ReceiveInterrupt = 1;
          }
```

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

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

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

   switch(b_InterruptMask)
        {
        case 1:
               /* EOC interrupt */
               ui_SaveArray[0] = pui_ValueArray[1];
               break;
        case 2:
               /* Timer conversion interrupt */
                for (ui_Cpt=0;ui_Cpt< pui_ValueArray [0] ;ui_Cpt++)
                   ui_SaveArray [ui_Cpt] = pui_ValueArray [ui_Cpt+1];
               break;
        case 4:
               /* EOS interrupt */
                for (ui_Cpt=0;ui_Cpt< pui_ValueArray [0] ;ui_Cpt++)
                   ui_SaveArray [ui_Cpt] = pui_ValueArray [ui_Cpt+1];
               break;
        case 8:
               /* DMA completed */
               for (ul_Cpt=0;ul_Cpt<ul_NbrAcquisitionDMA;ul_Cpt++)
                ui_SaveArray [ul_Cpt] = pusi_Index[ul_Cpt];
               break;
        case 16:
               /* Timer 2 has run down */
               ui_TimerIntCpt = ui_TimerIntCpt + 1;
               break;
        case 32:
               /* Watchdog has run down */
               ui_TimerIntCpt = ui_TimerIntCpt + 1;
               break;
        default :
               break;
        }
   b_ReceiveInterrupt = 1;

   }
```

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



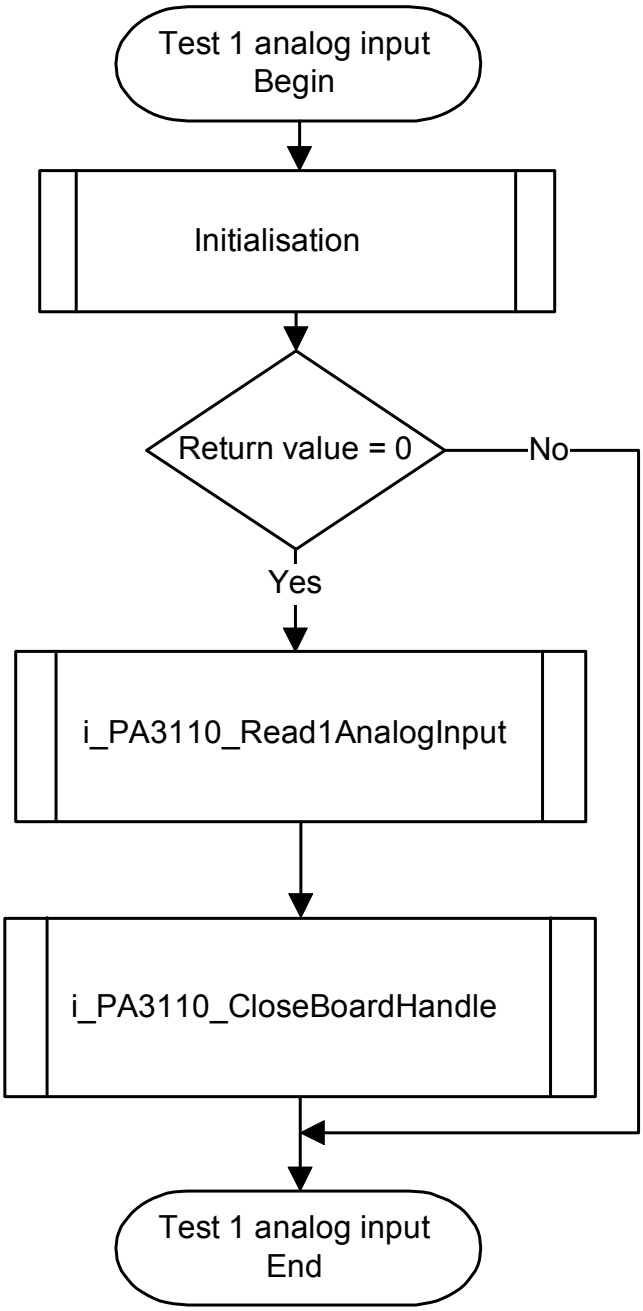
e) Example in C for Windows NT / 95 (synchronous mode)

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

str_UserStruct *ps_GlobalUserStruct;

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

   str_UserStruct *ps_UserStruct = (str_UserStruct *) pv_UserSharedMemory;
   pusi_Index = (unsigned short int *) pui_ValueArray;

   if ((b_InterruptMask&1) == 1) /* EOC interrupt */
      {
       ps_UserStruct->ui_SaveArray[0] = pui_ValueArray[1];
      }
   if ((b_InterruptMask&2) == 2) /* Timer conversion interrupt Acquisition */
      {
       for (ui_Cpt=0; ui_Cpt < pusi_Index[0]; ui_Cpt++)
           ps_UserStruct->ui_SaveArray [ui_Cpt] = pusi_Index[ui_Cpt];
       }
   if ((b_InterruptMask&4) == 4) /* EOS interrupt Acquisition */
      {
       for (ui_Cpt=0; ui_Cpt < pusi_Index[0]; ui_Cpt++)
           ps_UserStruct->ui_SaveArray [ui_Cpt] = pusi_Index[ui_Cpt];
       }
   if ((b_InterruptMask&8) == 8) /* DMA completed */
      {
       for (ui_Cpt=0;ui_Cpt<16;ui_Cpt++)
           ps_UserStruct->ui_SaveArray [ui_Cpt] = pusi_Index[ui_Cpt];
      }
   if ((b_InterruptMask&16) == 16) /* Timer 2 has run down */
      {
       ps_UserStruct->ui_TimerIntCpt = ps_UserStruct->ui_TimerIntCpt + 1;
      }
   if ((b_InterruptMask&32) == 32) /* Watchdog has run down */
      {
       ps_UserStruct->ui_TimerIntCpt = ps_UserStruct->ui_TimerIntCpt + 1;
      }

   i_PA3110_KRNL_Write1AnalogValue (0x300, 0, pui_ValueArray[1]);

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

   }
```

9.3 Direct conversion of analog inputs

9.3.1 Testing one analog input

a) Flow chart

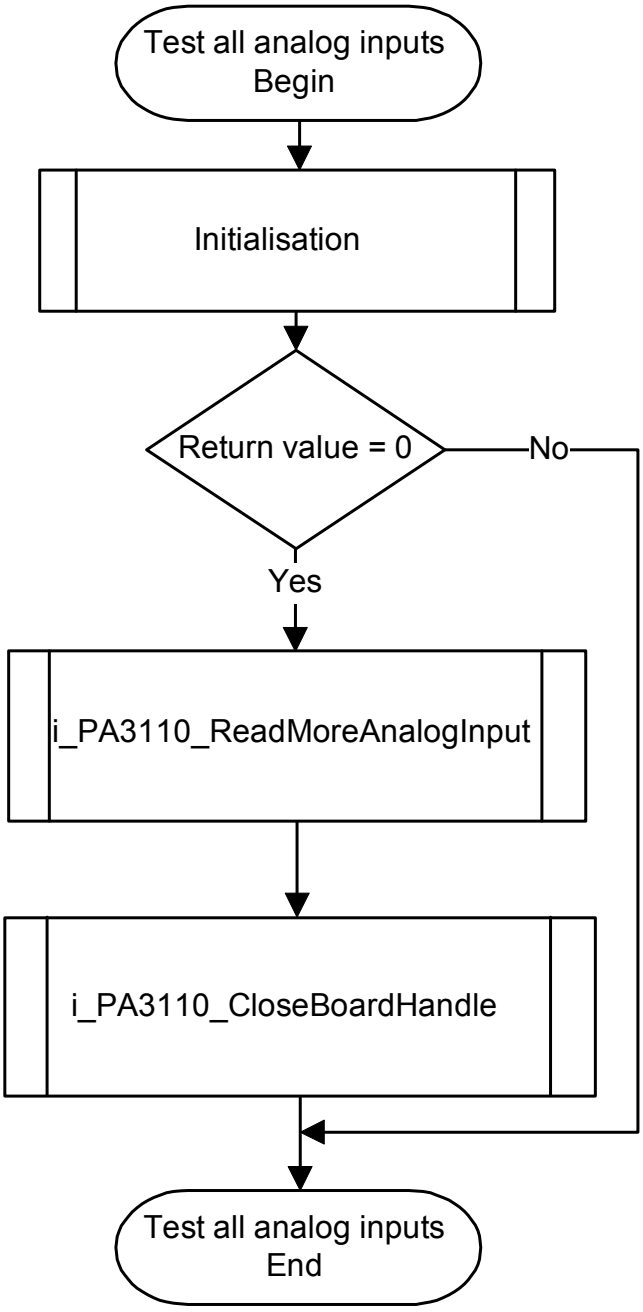


b) Example in C

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

     if (Initialisation (&b_BoardHandle) == 0)
        {
        if (i_PA3110_Read1AnalogInput   (b_BoardHandle,
                                         PA3110_CHANNEL_0,
                                         PA3110_1_GAIN,
                                         PA3110_UNIPOLAR,
                                         10,
                                         PA3110_DISABLE,
                                         &ui_ReadValue)  == 0)
           {
           printf ("ui_ReadValue = %u", ui_ReadValue);
           }
        else
           {
           printf ("Read value error");
           }
        i_PA3110_CloseBoardHandle (b_BoardHandle);
        }
     else
        {
        printf ("Initialisation error");
        }
     }
```

9.3.2 Testing all analog inputs

a) Flow chart



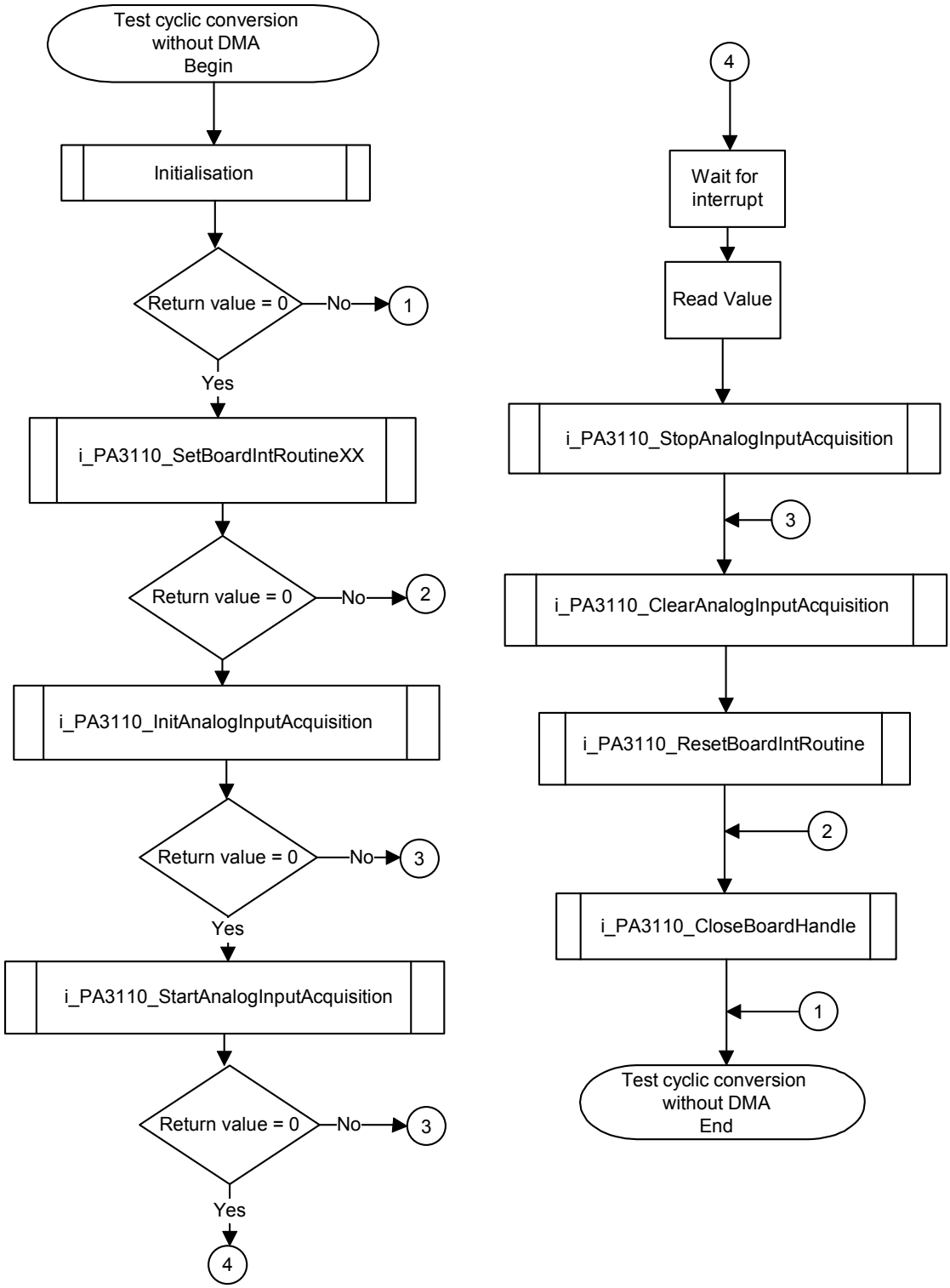
b) Example in C

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

     b_Channel [0] = PA3110_CHANNEL_0;
     b_Channel [1] = PA3110_CHANNEL_1;
     b_Channel [2] = PA3110_CHANNEL_2;
     b_Channel [3] = PA3110_CHANNEL_3;
     b_Channel [4] = PA3110_CHANNEL_4;
     b_Channel [5] = PA3110_CHANNEL_5;
     b_Channel [6] = PA3110_CHANNEL_6;
     b_Channel [7] = PA3110_CHANNEL_7;
     b_Gain [0] = PA3110_1_GAIN;
     b_Gain [1] = PA3110_1_GAIN;
     b_Gain [2] = PA3110_1_GAIN;
     b_Gain [3] = PA3110_1_GAIN;
     b_Gain [4] = PA3110_1_GAIN;
     b_Gain [5] = PA3110_1_GAIN;
     b_Gain [6] = PA3110_1_GAIN;
     b_Gain [7] = PA3110_1_GAIN;
      b_Polar [0] = PA3110_UNIPOLAR;
      b_Polar [1] = PA3110_UNIPOLAR;
      b_Polar [2] = PA3110_UNIPOLAR;
      b_Polar [3] = PA3110_UNIPOLAR;
      b_Polar [4] = PA3110_UNIPOLAR;
      b_Polar [5] = PA3110_UNIPOLAR;
      b_Polar [6] = PA3110_UNIPOLAR;
      b_Polar [7] = PA3110_UNIPOLAR;

     if (Initialisation (&b_BoardHandle) == 0)
        {
        if (i_PA3110_ReadMoreAnalogInput      (b_BoardHandle, 8, b_Channel, b_Gain,
                                               b_Polar, 10, ui_ReadValueArray)  == 0)
           {
           printf ("ui_ReadValue = %u %u %u %u %u %u %u %u",
                   ui_ReadValue [0], ui_ReadValue [1], ui_ReadValue [2], ui_ReadValue [3],
                   ui_ReadValue [4], ui_ReadValue [5], ui_ReadValue [6], ui_ReadValue [7]);
           }
        else
           {
           printf ("Read value error");
           }
        i_PA3110_CloseBoardHandle (b_BoardHandle);
        }
     else
        {
        printf ("Initialisation error");
        }
     }
```

9.4 Cyclic conversion of the analog inputs

9.4.1 Cyclic conversion without DMA and delay

a) Flow chart



b) Example in C for DOS

```
void main(void)
   {
   int i_Cpt;
   unsigned char b_Gain    [4];
   unsigned char b_Polar   [4];
   unsigned char b_Channel [4];
   unsigned char b_BoardHandle;

   if (Initialisation(&b_BoardHandle) == 0)
      {
      if (i_PA3110_SetBoardIntRoutineDos(b_BoardHandle, v_InterruptRoutine) == 0)
        {
        b_Channel[0] = PA3110_CHANNEL_0;
        b_Gain   [0] = PA3110_1_GAIN;
        b_Polar  [0] = PA3110_UNIPOLAR;
        b_Channel[1] = PA3110_CHANNEL_1;
        b_Gain   [1] = PA3110_1_GAIN;
        b_Polar  [1] = PA3110_UNIPOLAR;
        b_Channel[2] = PA3110_CHANNEL_2;
        b_Gain   [2] = PA3110_1_GAIN;
        b_Polar  [2] = PA3110_UNIPOLAR;
        b_Channel[3] = PA3110_CHANNEL_3;
        b_Gain   [3] = PA3110_1_GAIN;
        b_Polar  [3] = PA3110_UNIPOLAR;

        if (i_PA3110_InitAnalogInputAcquisition
                                        (b_BoardHandle,4,b_Channel,b_Gain,b_Polar,
                                        PA3110_SIMPLE_MODUS,1500,
                                        0,4,PA3110_DMA_NOT_USED,PA3110_SINGLE)==0)
           {
           b_ReceiveInterrupt = 0;
           if (i_PA3110_StartAnalogInputAcquisition (b_BoardHandle) == 0)
              {
              for (i_Cpt=0;i_Cpt<4;i_Cpt++)
                {
                while (b_ReceiveInterrupt == 0);
                b_ReceiveInterrupt = 0;
                printf("\n Acquisition 1 %u %u %u %u",ui_SaveArray[0], ui_SaveArray[1],
                                                     ui_SaveArray[2], ui_SaveArray[3]);
                }
              i_PA3110_StopAnalogInputAcquisition(b_BoardHandle);
              }
           else
              printf("\n Start acquisition error");
           i_PA3110_ClearAnalogInputAcquisition(b_BoardHandle);
           }
        else
           printf("\n Acquisition initialisation error");
        i_PA3110_ResetBoardIntRoutine(b_BoardHandle);
        }
      else
        printf("\n Interrupt routine initialisation error");
      i_PA3110_CloseBoardHandle(b_BoardHandle);
      }
   else
      printf("\n Initialisation error");

   }
```

c) Example in c for Windows 3.1x

```
void main(void)
   {
   int i_Cpt;
   unsigned char b_Gain [4];
   unsigned char b_Polar [4];
   unsigned char b_Channel [4];
   unsigned char b_BoardHandle;
   if (Initialisation(&b_BoardHandle) == 0)
      {
      if (i_PA3110_SetBoardIntRoutineWin16(b_BoardHandle, v_InterruptRoutine) == 0)
        {
        b_Channel[0] = PA3110_CHANNEL_0;
        b_Gain   [0] = PA3110_1_GAIN;
        b_Polar  [0] = PA3110_UNIPOLAR;
        b_Channel[1] = PA3110_CHANNEL_1;
        b_Gain   [1] = PA3110_1_GAIN;
        b_Polar  [1] = PA3110_UNIPOLAR;
        b_Channel[2] = PA3110_CHANNEL_2;
        b_Gain   [2] = PA3110_1_GAIN;
        b_Polar  [2] = PA3110_UNIPOLAR;
        b_Channel[3] = PA3110_CHANNEL_3;
        b_Gain   [3] = PA3110_1_GAIN;
        b_Polar  [3] = PA3110_UNIPOLAR;

       if (i_PA3110_InitAnalogInputAcquisition (b_BoardHandle,4,b_Channel,b_Gain,b_Polar,
         PA3110_SIMPLE_MODUS, 1500, 0,4,PA3110_DMA_NOT_USED,PA3110_SINGLE)==0)
           {
           b_ReceiveInterrupt = 0;
           if (i_PA3110_StartAnalogInputAcquisition (b_BoardHandle) == 0)
              {
              for (i_Cpt=0;i_Cpt<4;i_Cpt++)
                {
                while (b_ReceiveInterrupt == 0);
                b_ReceiveInterrupt = 0;
                printf("\n Acquisition 1 %u %u %u %u",ui_SaveArray[0], ui_SaveArray[1],
                                                     ui_SaveArray[2], ui_SaveArray[3]);
                }
              i_PA3110_StopAnalogInputAcquisition(b_BoardHandle);
              }
           else
              printf("\n Start acquisition error");
           i_PA3110_ClearAnalogInputAcquisition(b_BoardHandle);
           }
       else
          printf("\n Acquisition initialisation error");
       i_PA3110_ResetBoardIntRoutine(b_BoardHandle);
       }
      else
        printf("\n Interrupt routine initialisation error");
      i_PA3110_CloseBoardHandle(b_BoardHandle);
      }
   else
      printf("\n Initialisation error");

   }
```

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

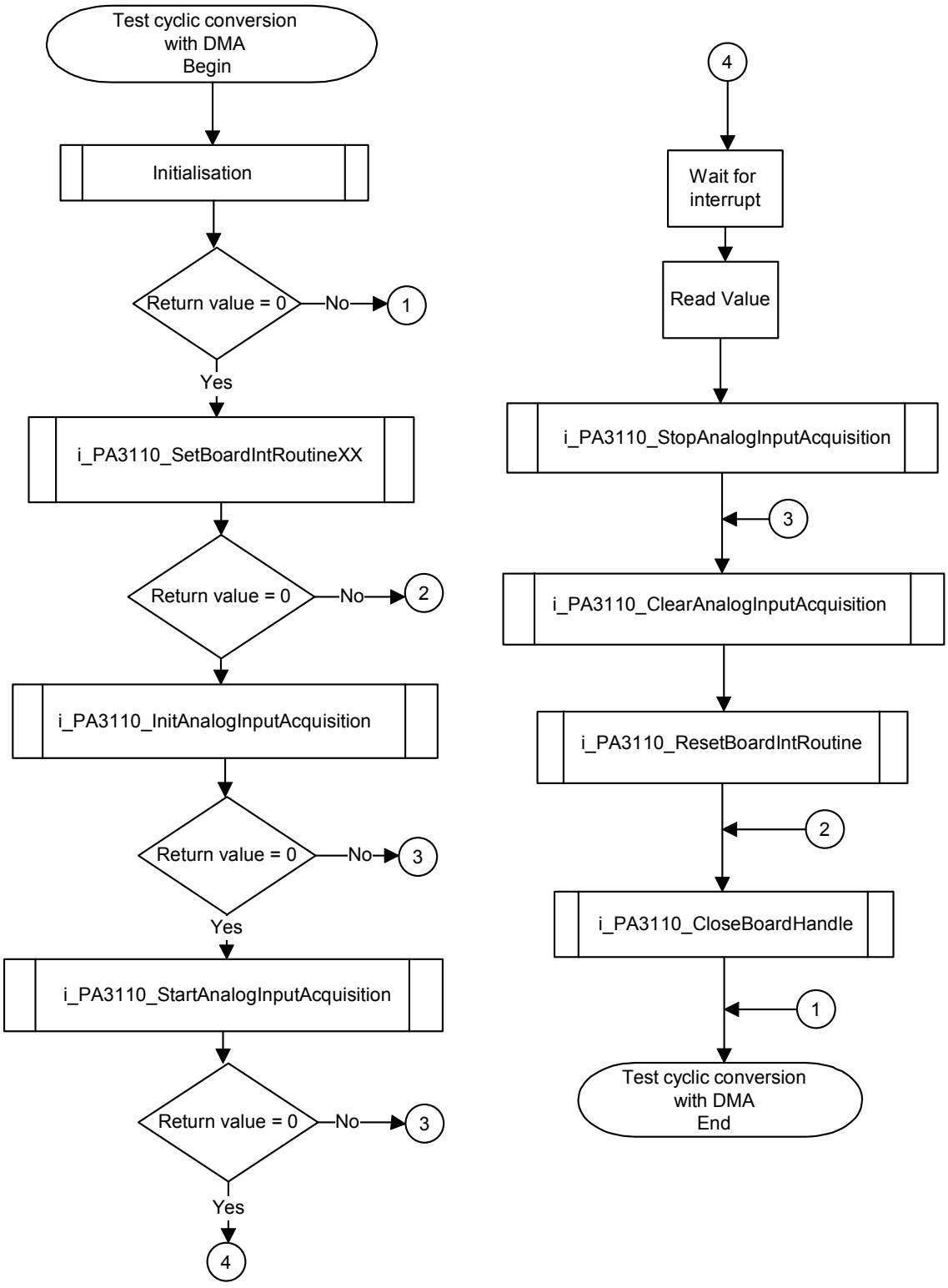
```
void main(void)
   {
   int i_Cpt;
   unsigned char b_Gain [4];
   unsigned char b_Polar [4];
   unsigned char b_Channel [4];
   unsigned char b_BoardHandle;
   if (Initialisation(&b_BoardHandle) == 0)
      {
      if (i_PA3110_SetBoardIntRoutineWin32(b_BoardHandle, ASYNCHRONOUS_MODE,
                                           0,NULL,v_InterruptRoutine) == 0)
        {
        b_Channel[0] = PA3110_CHANNEL_0;
        b_Gain   [0] = PA3110_1_GAIN;
        b_Polar  [0] = PA3110_UNIPOLAR;
        b_Channel[1] = PA3110_CHANNEL_1;
        b_Gain   [1] = PA3110_1_GAIN;
        b_Polar  [1] = PA3110_UNIPOLAR;
        b_Channel[2] = PA3110_CHANNEL_2;
        b_Gain   [2] = PA3110_1_GAIN;
        b_Polar  [2] = PA3110_UNIPOLAR;
        b_Channel[3] = PA3110_CHANNEL_3;
        b_Gain   [3] = PA3110_1_GAIN;
        b_Polar  [3] = PA3110_UNIPOLAR;

        if (i_PA3110_InitAnalogInputAcquisition (b_BoardHandle,4,b_Channel,b_Gain,b_Polar,
                                                 PA3110_SIMPLE_MODUS, 1500,
                                                 0,4,PA3110_DMA_NOT_USED,PA3110_SINGLE)==0)
           {
           b_ReceiveInterrupt = 0;
           if (i_PA3110_StartAnalogInputAcquisition (b_BoardHandle) == 0)
              {
              for (i_Cpt=0;i_Cpt<4;i_Cpt++)
                {
                while (b_ReceiveInterrupt == 0);
                b_ReceiveInterrupt = 0;
                printf("\n Acquisition 1 %u %u %u %u",ui_SaveArray[0], ui_SaveArray[1],
                                                     ui_SaveArray[2], ui_SaveArray[3]);
                }
              i_PA3110_StopAnalogInputAcquisition(b_BoardHandle);
              }
           else
              printf("\n Start acquisition error");
           i_PA3110_ClearAnalogInputAcquisition(b_BoardHandle);
           }
        else
           printf("\n Acquisition initialisation error");
        i_PA3110_ResetBoardIntRoutine(b_BoardHandle);
        }
      else
        printf("\n Interrupt routine initialisation error");
      i_PA3110_CloseBoardHandle(b_BoardHandle);
      }
   else
      printf("\n Initialisation error");

   }
```

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

```
void main(void)
   {
   int i_Cpt;
   unsigned char b_Gain [4];unsigned char b_Polar [4];unsigned char b_Channel [4];
   unsigned char b_BoardHandle;
   if (Initialisation(&b_BoardHandle) == 0)
      {
      if (i_PA3110_SetBoardIntRoutineWin32(b_BoardHandle,SYNCHRONOUS_MODE,
                                           sizeof(str_UserStruct),
                                           (void **) &ps_GlobalUserStruct,
                                           v_InterruptRoutine) == 0)
        {
        b_Channel[0] = PA3110_CHANNEL_0;
        b_Gain   [0] = PA3110_1_GAIN;
        b_Polar  [0] = PA3110_UNIPOLAR;
        b_Channel[1] = PA3110_CHANNEL_1;
        b_Gain   [1] = PA3110_1_GAIN;
        b_Polar  [1] = PA3110_UNIPOLAR;
        b_Channel[2] = PA3110_CHANNEL_2;
        b_Gain   [2] = PA3110_1_GAIN;
        b_Polar  [2] = PA3110_UNIPOLAR;
        b_Channel[3] = PA3110_CHANNEL_3;
        b_Gain   [3] = PA3110_1_GAIN;
        b_Polar  [3] = PA3110_UNIPOLAR;
        if (i_PA3110_InitAnalogInputAcquisition (b_BoardHandle,4,b_Channel,b_Gain,b_Polar,
                                                 PA3110_SIMPLE_MODUS, 1500,
                                                 0,4,PA3110_DMA_NOT_USED,PA3110_SINGLE)==0)
           {
           ps_GlobalUserStruct -> b_ReceiveInterrupt = 0;
           if (i_PA3110_StartAnalogInputAcquisition (b_BoardHandle) == 0)
              {
              for (i_Cpt=0;i_Cpt<4;i_Cpt++)
                {
                while (ps_GlobalUserStruct -> b_ReceiveInterrupt == 0);
                ps_GlobalUserStruct -> b_ReceiveInterrupt = 0;
                 printf("\n Acquisition 1 %u %u %u %u",
                        ps_GlobalUserStruct -> ui_SaveArray[0],
                        ps_GlobalUserStruct -> ui_SaveArray[1],
                        ps_GlobalUserStruct -> ui_SaveArray[2],
                       ps_GlobalUserStruct -> ui_SaveArray[3]);
                }
              i_PA3110_StopAnalogInputAcquisition(b_BoardHandle);
              }
           else
              printf("\n Start acquisition error");
           i_PA3110_ClearAnalogInputAcquisition(b_BoardHandle);
           }
        else
           printf("\n Acquisition initialisation error");
        i_PA3110_ResetBoardIntRoutine(b_BoardHandle);
        }
      else
        printf("\n Interrupt routine initialisation error");
      i_PA3110_CloseBoardHandle(b_BoardHandle);
      }
   else
      printf("\n Initialisation error");

   }
```

9.4.2 Cyclic conversion with DMA without delay

a) Flow chart



b) Example in C for DOS

```
void main(void)
   {
   int i_Cpt;
   unsigned char b_Gain [4];
   unsigned char b_Polar [4];
   unsigned char b_Channel [4];
   unsigned char b_BoardHandle;
   if (Initialisation(&b_BoardHandle) == 0)
      {
      if (i_PA3110_SetBoardIntRoutineDos(b_BoardHandle, v_InterruptRoutine) == 0)
        {
        b_Channel[0] = PA3110_CHANNEL_0;
        b_Gain   [0] = PA3110_1_GAIN;
        b_Polar  [0] = PA3110_UNIPOLAR;
        b_Channel[1] = PA3110_CHANNEL_1;
        b_Gain   [1] = PA3110_1_GAIN;
        b_Polar  [1] = PA3110_UNIPOLAR;
        b_Channel[2] = PA3110_CHANNEL_2;
        b_Gain   [2] = PA3110_1_GAIN;
        b_Polar  [2] = PA3110_UNIPOLAR;
        b_Channel[3] = PA3110_CHANNEL_3;
        b_Gain   [3] = PA3110_1_GAIN;
        b_Polar  [3] = PA3110_UNIPOLAR;

        if (i_PA3110_InitAnalogInputAcquisition (b_BoardHandle,4,b_Channel,b_Gain,b_Polar,
                                                 PA3110_SIMPLE_MODUS, 1500,0,16,
                                                 PA3110_DMA_USED,PA3110_SINGLE) == 0)
           {
           b_ReceiveInterrupt = 0;
           if (i_PA3110_StartAnalogInputAcquisition (b_BoardHandle) == 0)
              {
              while (b_ReceiveInterrupt == 0);
              b_ReceiveInterrupt = 0;
              printf("\n Acquisition 1 %u %u %u %u %u %u %u %u \n %u %u %u %u %u %u %u %u",
                      ui_SaveArray[0],ui_SaveArray[1],ui_SaveArray[2],ui_SaveArray[3],
                      ui_SaveArray[4],ui_SaveArray[5],ui_SaveArray[6],ui_SaveArray[7],
                      ui_SaveArray[8],ui_SaveArray[9],ui_SaveArray[10],ui_SaveArray[11],
                      ui_SaveArray[12],ui_SaveArray[13],ui_SaveArray[14],
                     ui_SaveArray[15]);

              i_PA3110_StopAnalogInputAcquisition(b_BoardHandle);
              i_PA3110_ClearAnalogInputAcquisition(b_BoardHandle);
              }
           else
              printf("\n Start acquisition error");
           }
        else
           printf("\n Acquisition initialisation error");
        i_PA3110_ResetBoardIntRoutine(b_BoardHandle);
        }
      else
        printf("\n Interrupt routine initialisation error");
      i_PA3110_CloseBoardHandle(b_BoardHandle);
      }
   else
      printf("\n Initialisation error");

   }
```

c) Example in C for Windows 3.1x

```
void main(void)
   {
   int i_Cpt;
   unsigned char b_Gain [4];
   unsigned char b_Polar [4];
   unsigned char b_Channel [4];
   unsigned char b_BoardHandle;
   if (Initialisation(&b_BoardHandle) == 0)
      {
      if (i_PA3110_SetBoardIntRoutineWin16(b_BoardHandle, v_InterruptRoutine) == 0)
        {
        b_Channel[0] = PA3110_CHANNEL_0;
        b_Gain   [0] = PA3110_1_GAIN;
        b_Polar  [0] = PA3110_UNIPOLAR;
        b_Channel[1] = PA3110_CHANNEL_1;
        b_Gain   [1] = PA3110_1_GAIN;
        b_Polar  [1] = PA3110_UNIPOLAR;
        b_Channel[2] = PA3110_CHANNEL_2;
        b_Gain   [2] = PA3110_1_GAIN;
        b_Polar  [2] = PA3110_UNIPOLAR;
        b_Channel[3] = PA3110_CHANNEL_3;
        b_Gain   [3] = PA3110_1_GAIN;
        b_Polar  [3] = PA3110_UNIPOLAR;

        if (i_PA3110_InitAnalogInputAcquisition (b_BoardHandle,4,b_Channel,b_Gain,b_Polar,
                                                 PA3110_SIMPLE_MODUS, 1500,0,16,
                                                 PA3110_DMA_USED,PA3110_SINGLE) == 0)
           {
           b_ReceiveInterrupt = 0;
           if (i_PA3110_StartAnalogInputAcquisition (b_BoardHandle) == 0)
              {
              while (b_ReceiveInterrupt == 0);
              b_ReceiveInterrupt = 0;
              printf("\n Acquisition 1 %u %u %u %u %u %u %u \n %u %u %u %u %u %u %u %u",
                      ui_SaveArray[0],ui_SaveArray[1],ui_SaveArray[2],ui_SaveArray[3],
                      ui_SaveArray[4],ui_SaveArray[5],ui_SaveArray[6],ui_SaveArray[7],
                      ui_SaveArray[8],ui_SaveArray[9],ui_SaveArray[10],ui_SaveArray[11],
                      ui_SaveArray[12],ui_SaveArray[13],ui_SaveArray[14],
                     ui_SaveArray[15]);
              i_PA3110_StopAnalogInputAcquisition(b_BoardHandle);
              i_PA3110_ClearAnalogInputAcquisition(b_BoardHandle);
              }
           else
              printf("\n Start acquisition error");
           }
        else
           printf("\n Acquisition initialisation error");
        i_PA3110_ResetBoardIntRoutine(b_BoardHandle);
        }
      else
        printf("\n Interrupt routine initialisation error");
      i_PA3110_CloseBoardHandle(b_BoardHandle);
      }
   else
      printf("\n Initialisation error");

   }
```

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

```
void main(void)
   {
   int i_Cpt;
   unsigned char b_Gain [4];
   unsigned char b_Polar [4];
   unsigned char b_Channel [4];
   unsigned char b_BoardHandle;
   if (Initialisation(&b_BoardHandle) == 0)
      {
      if (i_PA3110_SetBoardIntRoutineWin32(b_BoardHandleASYNCHRONOUS_MODE,0,NULL,
          v_InterruptRoutine) == 0)
       {
       b_Channel[0] = PA3110_CHANNEL_0;
       b_Gain   [0] = PA3110_1_GAIN;
       b_Polar  [0] = PA3110_UNIPOLAR;
       b_Channel[1] = PA3110_CHANNEL_1;
       b_Gain   [1] = PA3110_1_GAIN;
       b_Polar  [1] = PA3110_UNIPOLAR;
       b_Channel[2] = PA3110_CHANNEL_2;
       b_Gain   [2] = PA3110_1_GAIN;
       b_Polar  [2] = PA3110_UNIPOLAR;
       b_Channel[3] = PA3110_CHANNEL_3;
       b_Gain   [3] = PA3110_1_GAIN;
       b_Polar  [3] = PA3110_UNIPOLAR;

       if (i_PA3110_InitAnalogInputAcquisition (b_BoardHandle,4,b_Channel,b_Gain,b_Polar,
                                                PA3110_SIMPLE_MODUS, 1500,0,16,
                                                PA3110_DMA_USED,PA3110_SINGLE) == 0)
          {
          b_ReceiveInterrupt = 0;
          if (i_PA3110_StartAnalogInputAcquisition (b_BoardHandle) == 0)
             {
             while (b_ReceiveInterrupt == 0);
             b_ReceiveInterrupt = 0;
             printf("\n Acquisition 1 %u %u %u %u %u %u %u %u \n%u %u %u %u %u %u %u %u",
                     ui_SaveArray[0],ui_SaveArray[1],ui_SaveArray[2],ui_SaveArray[3],
                     ui_SaveArray[4],ui_SaveArray[5],ui_SaveArray[6],ui_SaveArray[7],
                     ui_SaveArray[8],ui_SaveArray[9],ui_SaveArray[10],ui_SaveArray[11],
                     ui_SaveArray[12],ui_SaveArray[13],ui_SaveArray[14],
                    ui_SaveArray[15]);
             i_PA3110_StopAnalogInputAcquisition(b_BoardHandle);
             i_PA3110_ClearAnalogInputAcquisition(b_BoardHandle);
             }
          else
             printf("\n Start acquisition error");
          }
       else
          printf("\n Acquisition initialisation error");
       i_PA3110_ResetBoardIntRoutine(b_BoardHandle);
       }
      else
       printf("\n Interrupt routine initialisation error");
      i_PA3110_CloseBoardHandle(b_BoardHandle);
      }
   else
      printf("\n Initialisation error");

   }
```

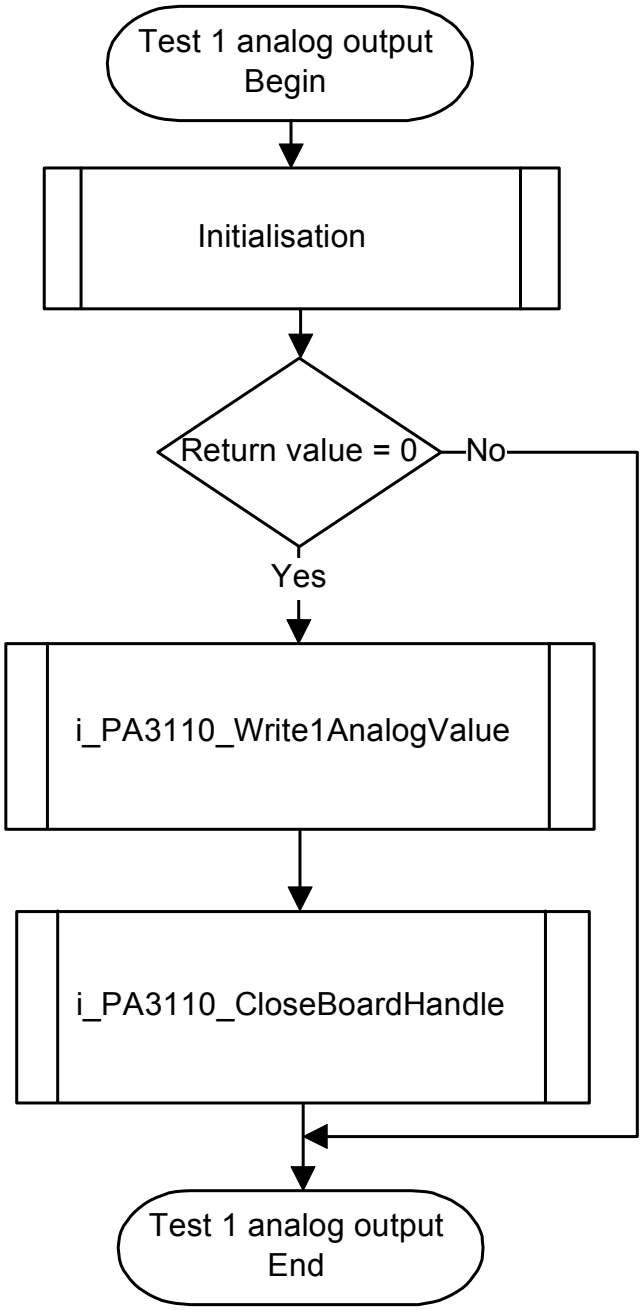
e) Example in C for Windows NT / 95 (synchronous mode)

```
void main(void)
   {
   int i_Cpt;
   unsigned char b_Gain [4];unsigned char b_Polar [4];unsigned char b_Channel [4];
   unsigned char b_BoardHandle;

   if (Initialisation(&b_BoardHandle) == 0)
      {
      if (i_PA3110_SetBoardIntRoutineWin32(b_BoardHandle,SYNCHRONOUS_MODE,
                                           sizeof(str_UserStruct),
                                           (void **) &GlobalUserStruct,
                                           v_InterruptRoutine) == 0)
       {
       b_Channel[0] = PA3110_CHANNEL_0;
       b_Gain   [0] = PA3110_1_GAIN;
       b_Polar  [0] = PA3110_UNIPOLAR;
       b_Channel[1] = PA3110_CHANNEL_1;
       b_Gain   [1] = PA3110_1_GAIN;
       b_Polar  [1] = PA3110_UNIPOLAR;
       b_Channel[2] = PA3110_CHANNEL_2;
       b_Gain   [2] = PA3110_1_GAIN;
       b_Polar  [2] = PA3110_UNIPOLAR;
       b_Channel[3] = PA3110_CHANNEL_3;
       b_Gain   [3] = PA3110_1_GAIN;
       b_Polar  [3] = PA3110_UNIPOLAR;

       if (i_PA3110_InitAnalogInputAcquisition (b_BoardHandle,4,b_Channel,b_Gain,b_Polar,
                                                PA3110_SIMPLE_MODUS, 1500,0,16,
                                                PA3110_DMA_USED,PA3110_SINGLE) == 0)
          {
          ps_GlobalUserStruct -> b_ReceiveInterrupt = 0;
          if (i_PA3110_StartAnalogInputAcquisition (b_BoardHandle) == 0)
             {
             while (ps_GlobalUserStruct -> b_ReceiveInterrupt == 0);
             ps_GlobalUserStruct -> b_ReceiveInterrupt = 0;
             printf("\n Acquisition 1 %u %u %u %u %u %u %u %u \n%u %u %u %u %u %u %u %u",
                    ps_GlobalUserStruct -> ui_SaveArray[0],
                    ps_GlobalUserStruct -> ui_SaveArray[1],
                    ps_GlobalUserStruct -> ui_SaveArray[2],
                    ps_GlobalUserStruct -> ui_SaveArray[3],
                    ps_GlobalUserStruct -> ui_SaveArray[4],
                    ps_GlobalUserStruct -> ui_SaveArray[5],
                    ps_GlobalUserStruct -> ui_SaveArray[6],
                    ps_GlobalUserStruct -> ui_SaveArray[7],
                    ps_GlobalUserStruct -> ui_SaveArray[8],
                    ps_GlobalUserStruct -> ui_SaveArray[9],
                    ps_GlobalUserStruct -> ui_SaveArray[10],
                    ps_GlobalUserStruct -> ui_SaveArray[11],
                    ps_GlobalUserStruct -> ui_SaveArray[12],
                    ps_GlobalUserStruct -> ui_SaveArray[13],
                    ps_GlobalUserStruct -> ui_SaveArray[14],
                    ps_GlobalUserStruct -> ui_SaveArray[15]);
             i_PA3110_StopAnalogInputAcquisition(b_BoardHandle);
             i_PA3110_ClearAnalogInputAcquisition(b_BoardHandle);
             }
          else
             printf("\n Start acquisition error");
          }
       else
          printf("\n Acquisition initialisation error");
       i_PA3110_ResetBoardIntRoutine(b_BoardHandle);
       }
      else
       printf("\n Interrupt routine initialisation error");
      i_PA3110_CloseBoardHandle(b_BoardHandle);
      }
   else
      printf("\n Initialisation error");
   }
```

9.5 Analog output channels

8.5.1 Testing one analog output channel

a) Flow chart

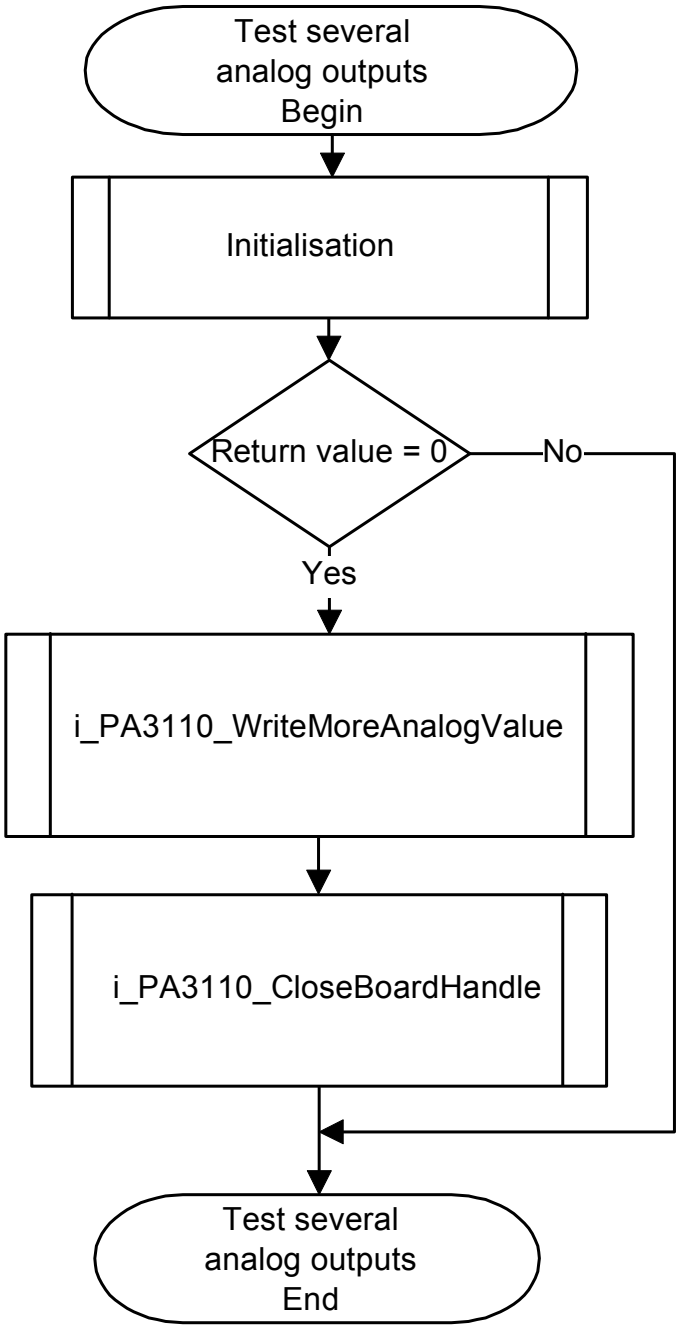


b) Example in C

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

     if (Initialisation (&b_BoardHandle) == 0)
        {
        if (i_PA3110_WriteAnalogValue   (b_BoardHandle,
                                         0,
                                         4095)  == 0)
           {
           printf ("Write test OK");
           }
        else
           {
           printf ("Write value error");
           }
        i_PA3110_CloseBoardHandle (b_BoardHandle);
        }
     else
        {
        printf ("Initialisation error");
        }
     }
```

9.5.2 Testing several analog output channels

a) Flow chart



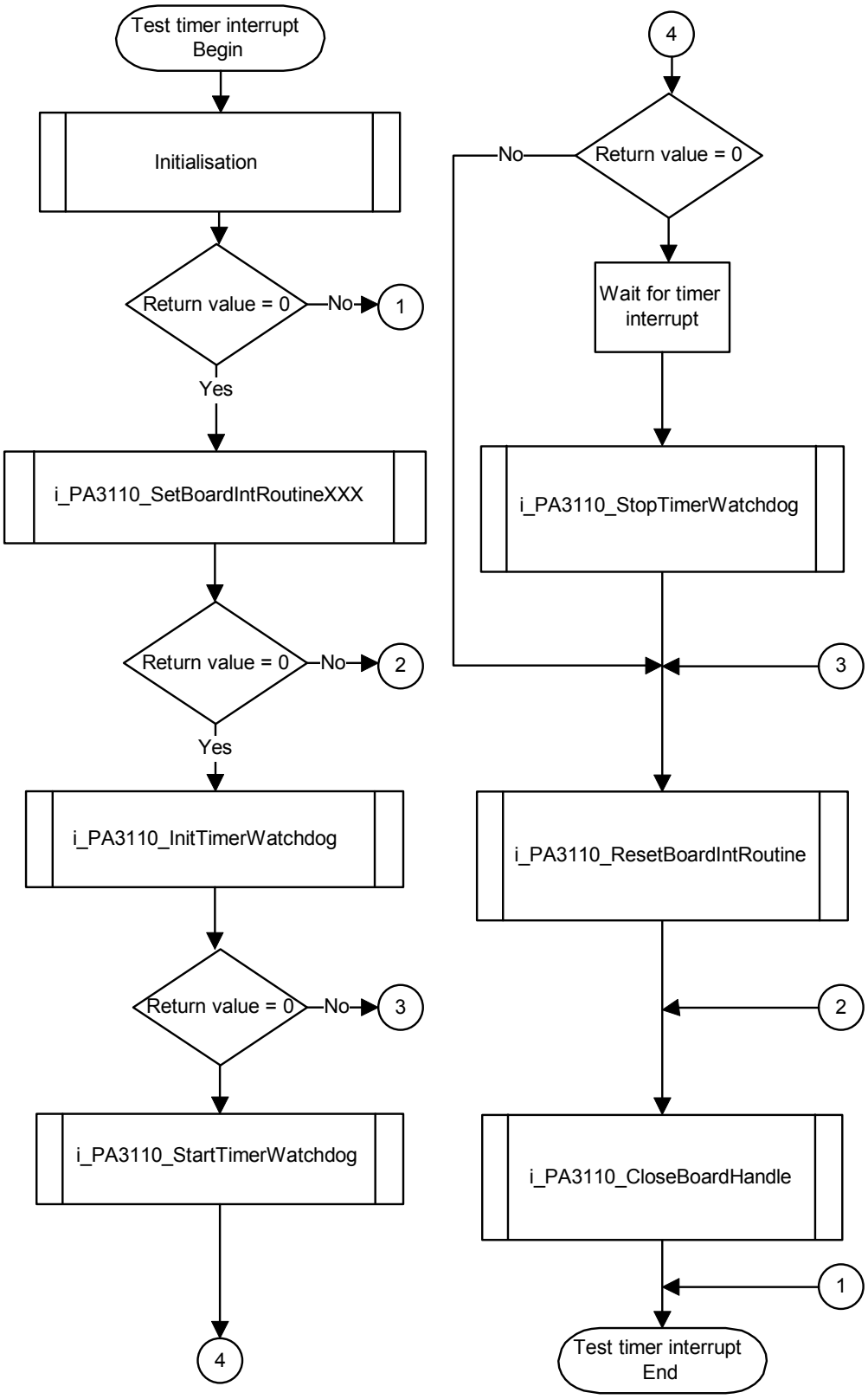
b) Example in C

```
void main (void)
     {
     unsigned char b_BoardHandle;
     unsigned char b_Channel            [8];
     unsigned int  ui_WriteValueArray   [8];

     b_Channel[0] = 0; ui_WriteValueArray [0] = 0;
     b_Channel[1] = 1; ui_WriteValueArray [1] = 2047;
     b_Channel[2] = 2; ui_WriteValueArray [2] = 4095;
     b_Channel[3] = 3; ui_WriteValueArray [3] = 0;
     b_Channel[4] = 4; ui_WriteValueArray [4] = 2047;
     b_Channel[5] = 5; ui_WriteValueArray [5] = 4095;
     b_Channel[6] = 6; ui_WriteValueArray [6] = 0;
     b_Channel[7] = 7; ui_WriteValueArray [7] = 2047;

     if (Initialisation (&b_BoardHandle) == 0)
        {
        if (i_PA3110_WriteMoreAnalogValue     (b_BoardHandle, 1, 8,ui_WriteValueArray)==0)
           {
           printf ("Write test OK");
           }
        else
           {
           printf ("Write value error");
           }

        i_PA3110_CloseBoardHandle (b_BoardHandle);
        }
     else
        {
        printf ("Initialisation error");
        }
     }
```

9.6 Timer

9.6.1 Testing the timer interrupt

a) Flow chart



b) Example in C for DOS

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

     if (Initialisation (&b_BoardHandle) == 0)
        {
        if (i_PA3110_SetBoardIntRoutineDOS (b_BoardHandle, v_InterruptRoutine) == 0)
           {
           ui_TimerIntCpt = 0;
           if (i_PA3110_InitTimerWatchdog     (b_BoardHandle, PA3110_TIMER,
                                               1000, PA3110_ENABLE) == 0)
              {
              if (i_PA3110_StartTimerWatchdog (b_BoardHandle) == 0)
                 {
                 while (ui_TimerIntCpt == 0);
                 printf ("Receive timer interrupt");
                 i_PA3110_StopTimerWatchdog (b_BoardHandle);
                 }
              else
                 printf ("Start timer error");
              }
           else
              printf ("Init timer error");
           i_PA3110_ResetBoardIntRoutine (b_BoardHandle);
           }
        else
           printf ("Interrupt routine initialisation error");
        i_PA3110_CloseBoardHandle (b_BoardHandle);
        }
     else
        printf ("Initialisation error");
     }
```

c) Example in C for Windows 3.1x

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

     if (Initialisation (&b_BoardHandle) == 0)
        {
        if (i_PA3110_SetBoardIntRoutineWin16 (b_BoardHandle, v_InterruptRoutine) == 0)
           {
           ui_TimerIntCpt = 0;
           if (i_PA3110_InitTimerWatchdog     (b_BoardHandle, PA3110_TIMER,
                                               1000, PA3110_ENABLE) == 0)
              {
              if (i_PA3110_StartTimerWatchdog (b_BoardHandle) == 0)
                 {
                 while (ui_TimerIntCpt == 0);
                 printf ("Receive timer interrupt");
                 i_PA3110_StopTimerWatchdog (b_BoardHandle);
                 }
              else
                 {
                 printf ("Start timer error");
                 }
              }
           else
              {
              printf ("Init timer error");
              }
           i_PA3110_ResetBoardIntRoutine (b_BoardHandle);
           }
        else
           {
           printf ("Interrupt routine initialisation error");
           }
        i_PA3110_CloseBoardHandle (b_BoardHandle);
        }
     else
        {
        printf ("Initialisation error");
        }
     }
```

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

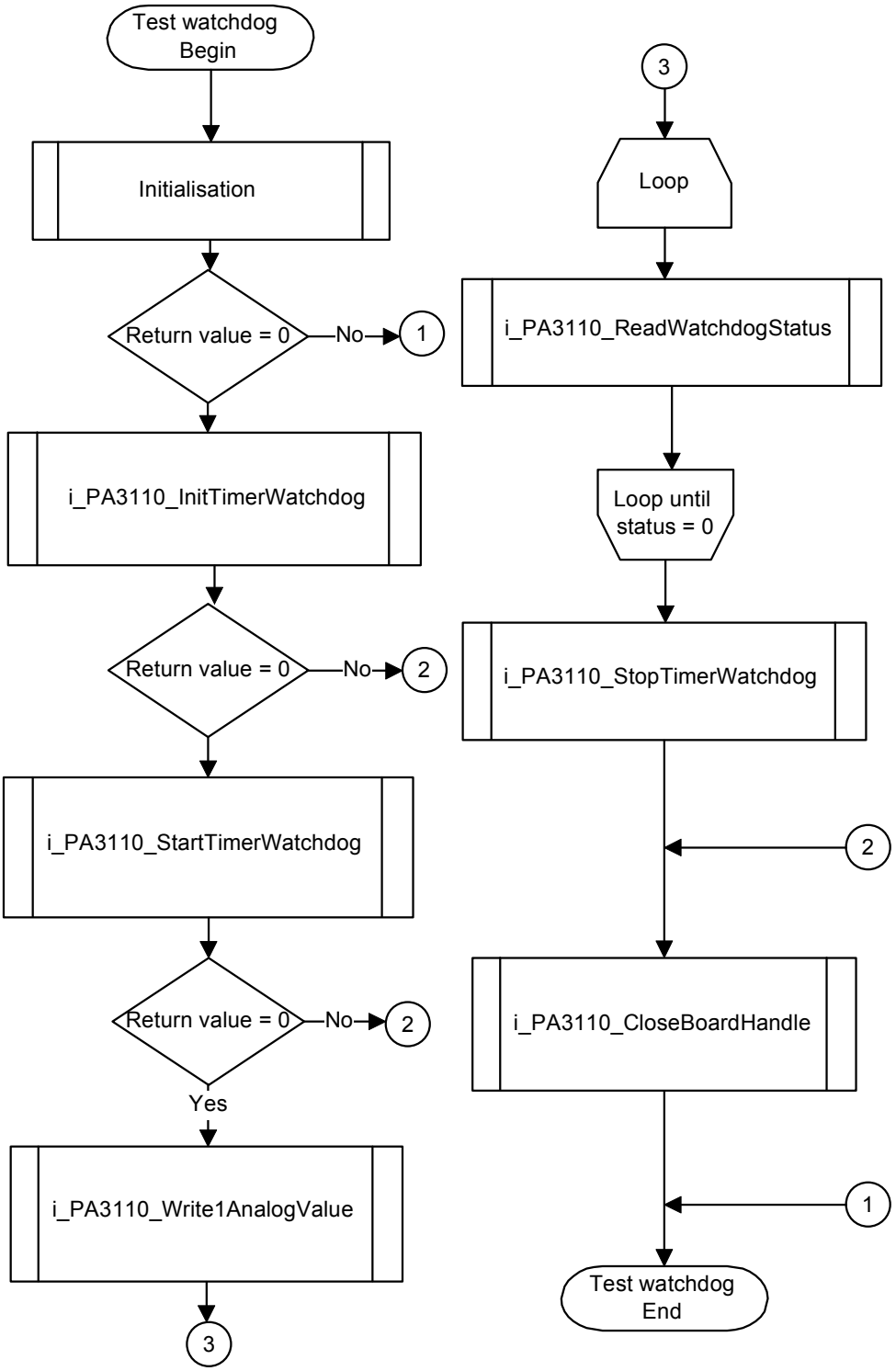
```
void main (void)
     {
     unsigned char b_BoardHandle;

     if (Initialisation (&b_BoardHandle) == 0)
        {
        if (i_PA3110_SetBoardIntRoutineWin32 (b_BoardHandle,PA3110_ASYNCHRONOUS_MODE,
                                              0,NULL,v_InterruptRoutine) == 0)
           {
           ui_TimerIntCpt = 0;
           if (i_PA3110_InitTimerWatchdog     (b_BoardHandle, PA3110_TIMER,
                                               1000, PA3110_ENABLE) == 0)
              {
              if (i_PA3110_StartTimerWatchdog (b_BoardHandle) == 0)
                 {
                 while (ui_TimerIntCpt == 0);
                 printf ("Receive timer interrupt");
                 i_PA3110_StopTimerWatchdog (b_BoardHandle);
                 }
              else
                 printf ("Start timer error");
              }
           else
              printf ("Init timer error");
           i_PA3110_ResetBoardIntRoutine (b_BoardHandle);
           }
        else
           printf ("Interrupt routine initialisation error");
        i_PA3110_CloseBoardHandle (b_BoardHandle);
        }
     else
         printf ("Initialisation error");
     }
```

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

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

     if (Initialisation (&b_BoardHandle) == 0)
        {
        if (i_PA3110_SetBoardIntRoutineWin32 (b_BoardHandle,PA3110_SYNCHRONOUS_MODE,
           sizeof(str_UserStruct),(void **) &ps_GlobalUserStruct,v_InterruptRoutine) == 0)
           {
           ps_GlobalUserStruct->ui_TimerIntCpt = 0;
           if (i_PA3110_InitTimerWatchdog     (b_BoardHandle, PA3110_TIMER,
                                               1000, PA3110_ENABLE) == 0)
              {
              if (i_PA3110_StartTimerWatchdog (b_BoardHandle) == 0)
                 {
                 while (ps_GlobalUserStruct->ui_TimerIntCpt == 0);
                 printf ("Receive timer interrupt");
                 i_PA3110_StopTimerWatchdog (b_BoardHandle);
                 }
              else
                 {
                 printf ("Start timer error");
                 }
              }
           else
              {
              printf ("Init timer error");
              }
           i_PA3110_ResetBoardIntRoutine (b_BoardHandle);
           }
        else
           {
           printf ("Interrupt routine initialisation error");
           }
        i_PA3110_CloseBoardHandle (b_BoardHandle);
        }
     else
        {
        printf ("Initialisation error");
        }
     }
```

9.6.2 Testing the watchdog

a) Flow chart



b) example in C

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

     if (Initialisation (&b_BoardHandle) == 0)
        {
        if (i_PA3110_InitTimerWatchdog  (b_BoardHandle, PA3110_WATCHDOG,
                                         1000, PA3110_DISABLE) == 0)
           {
           if (i_PA3110_StartTimerWatchdog (b_BoardHandle) == 0)
              {
              i_PA3110_Write1AnalogValue (b_BoardHandle, 1, PA3110_UNIPOLAR, 8192);
              do
                 {
                 i_PA3110_ReadWatchdogStatus (b_BoardHandle, &b_WatchdogSatus);
                 }
              while (b_WatchdogSatus == 0);
              printf ("Receive timer interrupt");
              i_PA3110_StopTimerWatchdog (b_BoardHandle);
              }
           else
              {
              printf ("Start watchdog error");
              }
           }
        else
           {
           printf ("Init watchdog error");
           }
        i_PA3110_CloseBoardHandle (b_BoardHandle);
        }
     else
        {
        printf ("Initialisation error");
        }
     }
```

INDEX